Dr. Joseph Teguh Santoso, S.Kom, M.Kom

Kecerdasan Buatan
&
Jaringan Syaraf Buatan

YAYASAN PRIMA AGUS TEKNIK

Kecerdasan Buatan & Jaringan Syaraf Buatan

Dr. Joseph Teguh Santoso, S.Kom, M.Kom

BIODATA PENULIS

Dr. Joseph Teguh Santoso, S.Kom, M.Kom adalah Rektor dari Universitas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *e-commerce*sejak Tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek Sepeda Raleigh, dengan omzet tahunan pada Tahun 2019 mencapai lebih dari Rp. 35 Milyar rupiah dan terus meningkat. Dr. Joseph T.S memiliki lisensi tunggal sepeda merek "Raleigh" untuk penjualan *Online* di seluruh China. Di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek "Fengjiu", yaitu Pabrik Sepeda Listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online store* di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-21-7

Kecerdasan Buatan
&
Jaringan Syaraf Buatan

Dr. Joseph Teguh Santoso, S.Kom, M.Kom

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Kecerdasan Buatan & Jaringan Syaraf Buatan

Penulis :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom

ISBN : 9 786235 734217

Editor :
Muhammad Sholikan, M.Kom

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa bahwa buku yang berjudul *"Kecerdasan Buatandan Jaringan Syaraf Buatan*" ini dapat diselesaikan dengan baik. Berbicara tentang Artificial Intelligence atau kecerdasan buatan, merupakan kemampuan komputer yang dikembangkan untuk memecahkan suatu masalah. Sedangkan Jaringan syaraf buatan adalah Algoritmapembentuk kecerdasan buatan, dalam buku ini akan mempelajari secara mendalam tentang Kecerdasan Buatan dan Jaringan Syaraf tiruan meliputi pengetahuan apa itu kecerdasan buatan dan jaringan syaraf buatan, cara memrancang, hingga pengaplikasiannya dalam kehidupan sehari-hari.

Buku ini terbagi menjadi 2 bagian. Bagian pertama akan membahas tentang Artificial Intelligence, sedangkan bagian kedua akan membahas tentang Jaringan Syaraf tiruan. Buku inijuga dilengkapi dengan latihan soal agar mempermudah pembaca dalam penguasaan materi.Dalam buku ini juga akan diterangkan *machine Learning* dan cara untuk membangunnya. Data Mining atau yang sering kita sebut penambangan data adalah metode dalam ilmu komputer yang biasa digunakan dalam proses pencarian *knowledge.* Metode-metode dan tahapan *datamining* akan diuraikan secara detail dalam buku ini. Perkembangan dan pengembanganJaringan syaraf tiruan akan dijelaskan dan diuji dalam buku ini.

Pengaplikasian *Artificial Intelligence* dalam juga akan dibahas dan diujikan dalam buku ini. dalam buku ini memperkenalkan Anda pada bentuk kecerdasan buatan yang disebut jaringan saraf. Ini menjelaskan semua konsep penting dari jaringan saraf seperti lapisan, neuron, koneksi, bobot, dan fungsi aktivasi. Bab ini juga menjelaskan konvensi yang digunakanuntuk menggambar diagram jaringan saraf. Bab berikut menunjukkan semua detail pemrosesan jaringan neuron dengan menjelaskan cara menghitung semua hasil jaringan secara manual. Untuk mempermudah, dua istilah—jaringan saraf dan jaringan—akan digunakan secara bergantian untuk sisa buku ini.

Buku ini mengeksplorasi mesin bagian dalam pemrosesan jaringan saraf dengan menjelaskan bagaimana semua hasil pemrosesan dihitung. Serta memperkenalkan pembaca pada turunan dan gradien dan menjelaskan bagaimana konsep-konsep ini digunakan dalam menemukan salah satu fungsi kesalahan minimum. Lalu menunjukkan contoh sederhana di mana setiap hasil dihitung secara manual. Menjelaskan aturan perhitungan saja tidak cukup untuk memahami subjek karena menerapkan aturan pada arsitektur jaringan tertentu sangatrumit.

Buku ini menerangkan bahwa model berfungsi dengan benar untuk proyek pembaca sebelum memulai pengembangan apa pun. Kegagalan untuk memilih model yang benar akan meninggalkan Anda dengan aplikasi yang salah bekerja. Jaringan juga akan menghasilkan hasilyang salah ketika digunakan untuk memprediksi hasil dari semua jenis permainan (judi, olahraga, dan sebagainya). Akhir kata semoga buku ini bermanfaat bagi para pembaca.

Semarang, Desember 2021
Penulis

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

DAFTAR ISI

BAGIAN 1
BAB 1
ARTIFICIAL INTELLIGENCE

1.1 APAKAH *ARTIFICAL INTELLIGENCE* (KECERDASAN BUATAN) ITU?

Istilah *Artificial Intelligence* /kecerdasan buatan membangkitkan emosi. Untuk satu hal ada daya tarik kita dengan kecerdasan, yang tampaknya memberikan kepada kita manusia tempat khusus di antara bentuk-bentuk kehidupan. Muncul pertanyaan seperti "Apakah kecerdasan itu?", "Bagaimana seseorang dapat mengukur kecerdasan?" atau "Bagaimana cara kerja otak?". Semua pertanyaan ini bermakna ketika mencoba memahami kecerdasan buatan. Namun, pertanyaan sentral bagi para insinyur, terutama bagi ilmuwan komputer, adalah pertanyaan tentang mesin cerdas yang berperilaku seperti manusia, menunjukkan perilaku cerdas.

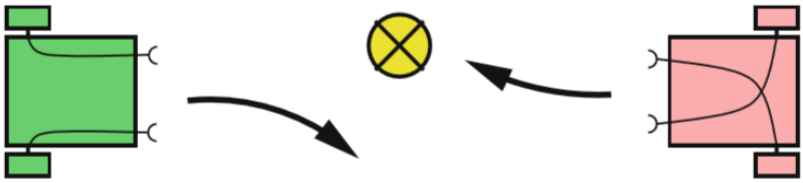
Atribut buatan mungkin membangkitkan asosiasi yang jauh berbeda. Ini memunculkan ketakutan akan cyborg cerdas. Ini mengingat gambar dari novel fiksi ilmiah. Ini menimbulkan pertanyaan apakah kebaikan tertinggi kita, jiwa, adalah sesuatu yang harus kita coba pahami, contohkan, atau bahkan rekonstruksi.

Dengan interpretasi begitu saja yang berbeda, menjadi sulit untuk mendefinisikan istilah kecerdasan buatan atau AI secara sederhana dan kokoh. Namun demikian saya ingin mencoba, dengan menggunakan contoh dan definisi historis, untuk mengkarakterisasi bidang AI. Pada tahun 1955, John McCarthy, salah satu pelopor AI, adalah orang pertama yang mendefinisikan istilah kecerdasan buatan, kira-kira sebagai berikut:

Tujuan AI adalah untuk mengembangkan mesin yang berperilaku seolah-olah mereka cerdas.

Untuk menguji definisi ini, pembaca dapat membayangkan skenario berikut. Lima belas atau lebih kendaraan robotik kecil bergerak di atas permukaan persegi empat kali empat meter tertutup. Seseorang dapat mengamati berbagai pola perilaku. Beberapa kendaraan membentuk kelompok kecil dengan gerakan yang relatif sedikit. Yang lain bergerak dengan damai melalui ruang dan dengan anggun menghindari tabrakan apa pun. Yang lain lagi tampaknya mengikuti seorang pemimpin. Perilaku agresif juga dapat diamati. Apakah yang kita lihat adalah perilaku cerdas?

Menurut definisi McCarthy, robot-robot yang disebutkan di atas dapat digambarkan sebagai robot yang cerdas. Psikolog Valentin Braitenberg telah menunjukkan bahwa perilaku yang tampaknya kompleks ini dapat dihasilkan oleh rangkaian listrik yang sangat sederhana. Kendaraan yang disebut Braitenberg memiliki dua roda, yang masing-masing digerakkan oleh motor listrik independen. Kecepatan masing-masing motor dipengaruhi oleh sensor cahaya di bagian depan kendaraan seperti yang ditunjukkan pada Gambar 1.1. Semakin banyak cahaya yang mengenai sensor, semakin cepat motor berjalan. Kendaraan 1 di bagian kiri gambar, menurut konfigurasinya, bergerak menjauh dari sumber cahaya titik. Kendaraan 2 di sisi lain bergerak menuju sumber cahaya. Modifikasi kecil lebih lanjut dapat menciptakan pola perilaku lain, sehingga dengan kendaraan yang sangat sederhana ini kita dapat mewujudkan perilaku mengesankan yang dijelaskan di atas.



Gambar 1.1 Dua kendaraan Braitenberg yang sangat sederhana dan reaksinya terhadap sumber cahaya

Jelas definisi di atas tidak cukup karena AI memiliki tujuan untuk memecahkan masalah praktis yang sulit yang tentunya terlalu menuntut untuk kendaraan Braitenberg. Dalam Encyclopedia Britannica [Bri91] ditemukan definisi yang berbunyi seperti:

> *AI adalah kemampuan komputer digital atau robot yang dikendalikan komputer untuk memecahkan masalah yang biasanya dikaitkan dengan kemampuan pemrosesan intelektual yang lebih tinggi dari manusia ...*

Kaya, singkat dan padat, mencirikan apa yang telah dilakukan peneliti AI selama 50 tahun terakhir. Bahkan di tahun 2050 definisi ini akan up to date.

Tugas-tugas seperti pelaksanaan banyak komputasi dalam waktu singkat adalah poin kuat dari komputer digital. Dalam hal ini mereka mengungguli manusia dengan banyak kelipatan. Namun, di banyak bidang lain, manusia jauh lebih unggul daripada mesin. Misalnya, seseorang yang memasuki ruangan yang tidak dikenalnya akan mengenali lingkungan sekitar dalam sepersekian detik dan, jika perlu, dengan cepat membuat keputusan dan merencanakan tindakan. Sampai saat ini, tugas ini terlalu menuntut untuk robot otonom[1]. Menurut definisi Rich, ini adalah tugas AI. Faktanya, penelitian tentang robot otonom adalah tema penting saat ini dalam AI. Konstruksi komputer catur, di sisi lain, telah kehilangan relevansi karena mereka sudah bermain di atau di atas tingkat grandmaster.

Akan berbahaya, bagaimanapun, untuk menyimpulkan dari definisi Rich bahwa AI hanya berkaitan dengan implementasi pragmatis dari proses cerdas. Sistem cerdas, dalam pengertian definisi Rich, tidak dapat dibangun tanpa pemahaman mendalam tentang penalaran manusia dan tindakan cerdas secara umum, karena itu ilmu saraf sangat penting bagi AI. Ini juga menunjukkan bahwa definisi lain yang dikutip mencerminkan aspek penting AI.

Kekuatan khusus kecerdasan manusia adalah kemampuan beradaptasi. Kita mampu menyesuaikan diri dengan berbagai kondisi lingkungan dan mengubah perilaku kita sesuai dengan itu melalui pembelajaran. Justru karena kemampuan belajar kita jauh lebih unggul daripada komputer, pembelajaran mesin, menurut definisi Rich, adalah subbidang utama AI.

1.2 ILMU OTAK DAN PEMECAHAN MASALAH

Melalui penelitian sistem cerdas kita dapat mencoba memahami cara kerja otak manusia dan kemudian memodelkan atau mensimulasikannya di komputer. Banyak ide dan prinsip di bidang jaringan saraf (lihat Bab 9) berasal dari ilmu otak dengan bidang ilmu saraf yang terkait.

Pendekatan yang sangat berbeda dihasilkan dari mengambil garis tindakan yang berorientasi pada tujuan, mulai dari masalah dan mencoba menemukan solusi yang paling optimal. Bagaimana manusia memecahkan masalah diperlakukan sebagai tidak penting di

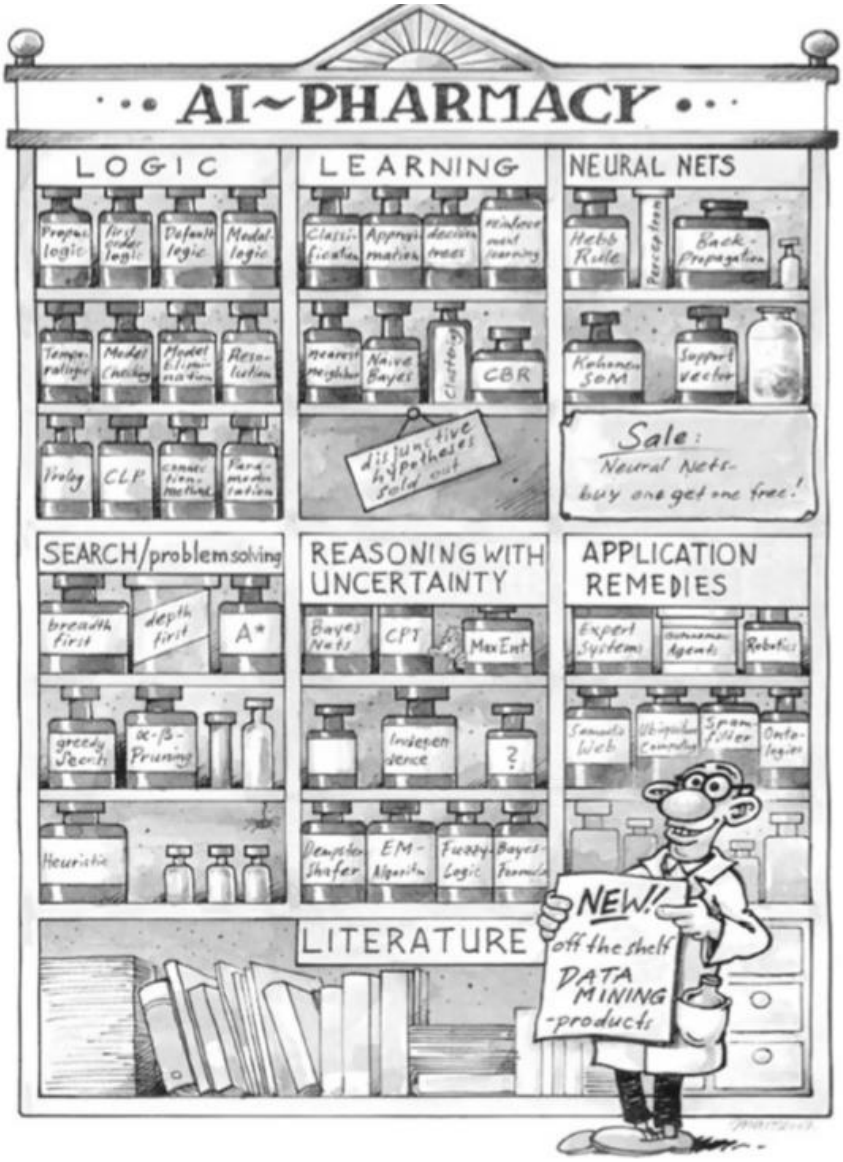
[1] Robot otonom bekerja secara mandiri, tanpa dukungan manual, khususnya tanpa kendali jarak jauh.

sini. Metode, dalam pendekatan ini, bersifat sekunder. Pertama dan terpenting adalah solusi cerdas yang optimal untuk masalah tersebut. Alih-alih menggunakan metode tetap (seperti, misalnya, logika predikat), AI memiliki tujuan tetapnya untuk menciptakan agen cerdas untuk sebanyak mungkin tugas yang berbeda.

Karena tugas AI sangat berbeda, tidak mengherankan bahwa metode yang saat ini digunakan dalam AI seringkali juga sangat berbeda. Mirip dengan obat-obatan, yang mencakup banyak prosedur diagnostik dan terapi yang berbeda, seringkali menyelamatkan jiwa, AI juga menawarkan beragam solusi efektif untuk berbagai aplikasi. Untuk inspirasi mental, perhatikan Gambar 1.2. Sama seperti dalam kedokteran, tidak ada metode universal untuk semua bidang aplikasi AI, melainkan sejumlah besar kemungkinan solusi untuk sejumlah besar berbagai masalah sehari-hari, besar dan kecil.

Ilmu kognitif dikhususkan untuk penelitian pemikiran manusia pada tingkat yang agak lebih tinggi. Sama halnya dengan ilmu otak, bidang ini melengkapi AI praktis dengan banyak ide penting. Di sisi lain, algoritma dan implementasi mengarah pada kesimpulan penting lebih lanjut tentang bagaimana fungsi penalaran manusia. Jadi ketiga bidang ini mendapat manfaat dari pertukaran interdisipliner yang bermanfaat. Subjek buku ini, bagaimanapun, adalah AI yang berorientasi pada masalah sebagai subdisiplin ilmu komputer.

Ada banyak pertanyaan filosofis yang menarik seputar kecerdasan dan kecerdasan buatan. Kita manusia memiliki kesadaran; yaitu, kita dapat berpikir tentang diri kita sendiri dan bahkan merenungkan bahwa kita dapat memikirkan diri kita sendiri. Bagaimana kesadaran muncul? Banyak filsuf dan ahli saraf sekarang percaya bahwa pikiran dan kesadaran terkait dengan materi, yaitu dengan otak. Pertanyaan apakah mesin suatu hari nanti bisa memiliki pikiran atau kesadaran bisa menjadi relevan di beberapa titik di masa depan. Masalah pikiran-tubuh khususnya menyangkut apakah pikiran terikat pada tubuh atau tidak. Kami tidak akan membahas pertanyaan-pertanyaan ini di sini. Pembaca yang tertarik dapat berkonsultasi dan diundang, selama studi teknologi AI, untuk membentuk pendapat pribadi tentang pertanyaan-pertanyaan ini.



Gambar 1.2 Contoh kecil dari solusi yang ditawarkan oleh AI

1.3 TES TURING DAN CHATTERBOTS

Alan Turing membuat dirinya terkenal sebagai pelopor awal AI dengan definisinya tentang mesin cerdas, di mana mesin tersebut harus lulus tes berikut. Orang yang diuji, Alice, duduk di ruang terkunci dengan dua terminal komputer. Satu terminal terhubung ke mesin, yang lain dengan orang yang tidak jahat, Bob. Alice dapat mengetik pertanyaan ke kedua terminal. Dia diberi tugas untuk memutuskan, setelah lima menit, terminal mana yang menjadi milik mesin. Mesin lulus tes jika bisa menipu Alice setidaknya 30% dari waktu.

Sementara tes ini sangat menarik secara filosofis, untuk AI praktis, yang berhubungan dengan pemecahan masalah, itu bukan tes yang sangat relevan. Alasan untuk ini serupa dengan yang disebutkan di atas terkait dengan kendaraan Braitenberg (lihat Latihan 1.3).

Pelopor AI dan kritikus sosial Joseph Weizenbaum mengembangkan sebuah program bernama Eliza, yang dimaksudkan untuk menjawab pertanyaan subjek tes seperti psikolog manusia. Dia terbukti mampu menunjukkan keberhasilan dalam banyak kasus. Seharusnya sekretarisnya sering berdiskusi panjang lebar dengan program tersebut. Saat ini di internet ada banyak yang disebut chatterbots, beberapa di antaranya tanggapan

awalnya cukup mengesankan. Namun, setelah jangka waktu tertentu, sifat buatan mereka menjadi jelas. Beberapa dari program ini sebenarnya mampu belajar, sementara yang lain memiliki pengetahuan luar biasa tentang berbagai mata pelajaran, misalnya geografi atau pengembangan perangkat lunak. Sudah ada aplikasi komersial untuk chatterbots dalam dukungan pelanggan online dan mungkin ada yang lain di bidang e-*learning*. Bisa dibayangkan bahwa pelajar dan sistem e-*learning* dapat berkomunikasi melalui chatterbot. Pembaca mungkin ingin membandingkan beberapa chatterbots dan mengevaluasi kecerdasan mereka dalam Latihan 1.1

1.4 SEJARAH AI

AI memanfaatkan banyak pencapaian ilmiah masa lalu yang tidak disebutkan di sini, karena AI sebagai ilmu yang berdiri sendiri baru ada sejak pertengahan abad ke-20. Tabel 1.1 , dengan pencapaian AI terpenting, dan representasi grafis dari pergerakan utama AI pada Gambar 1.3 melengkapi teks berikut.

Tabel 1.1 Milestones in the development of AI from Gödel to today	
1931	Kurt Gödel Austria menunjukkan bahwa dalam logika predikat orde pertama semua pernyataan benar dapat diturunkan. Dalam logika tingkat tinggi, di sisi lain, ada pernyataan benar yang tidak dapat dibuktikan. (Dalam Gödel menunjukkan bahwa logika predikat yang diperluas dengan aksioma aritmatika tidak lengkap.)
1937	Alan Turing menunjukkan batas-batas mesin cerdas dengan masalah penghentian
1943	McCulloch dan Pitts memodelkan jaringan saraf dan membuat koneksi ke logika proposisional.
1950	Alan Turing mendefinisikan kecerdasan mesin dengan tes Turing dan menulis tentang mesin pembelajaran dan algoritma genetika.
1951	Marvin Minsky mengembangkan mesin jaringan saraf. Dengan 3000 tabung vakum ia mensimulasikan 40 neuron.
1955	Arthur Samuel (IBM) membuat program pemeriksa pembelajaran yang bermain lebih baik daripada pengembangnya.
1956	McCarthy menyelenggarakan konferensi di Dartmouth College. Disinilah nama *Artificial Intelligence* pertama kali diperkenalkan. Newell dan Simon dari Carnegie Mellon University (CMU) mempresentasikan Logic Theorist, program komputer pemrosesan simbol pertama.
1958	McCarthy menciptakan di MIT (Massachusetts Institute of Technology) tingkat tinggi
1959	bahasa LISP. Dia menulis program yang mampu memodifikasi dirinya sendiri.
1961	Gelernter (IBM) membangun Pembukti Teorema Geometri.
1963	Pemecah Masalah Umum (GPS) oleh Newell dan Simon meniru pemikiran manusia.
1965	McCarthy mendirikan Lab AI di Universitas Stanford.
1966	Robinson menemukan kalkulus resolusi untuk logika predikat.
1969	Program Weizenbaum, Eliza, melakukan dialog dengan orang-orang dalam bahasa alami.
1972	Minsky dan Papert menunjukkan dalam buku mereka Perceptrons bahwa perceptron, jaringan saraf yang sangat sederhana, hanya dapat mewakili fungsi linier.
1976	Ilmuwan Prancis Alain Colmerauer menemukan bahasa pemrograman logika PROLOG (Bab 5). Dokter Inggris de Dombal mengembangkan sistem pakar untuk diagnosis nyeri perut akut. Ini tidak diperhatikan dalam komunitas AI arus utama saat itu .
	Shortliffe dan Buchanan mengembangkan MYCIN, sistem pakar untuk diagnosis penyakit menular, yang mampu menangani ketidakpastian (Bab 7).

1981	Jepang memulai, dengan biaya besar, "Proyek Generasi Kelima" dengan tujuan membangun mesin PROLOG yang kuat.
1982	R1, sistem pakar untuk mengkonfigurasi komputer, menghemat Digital Equipment Corporation 40 juta dolar per tahun.
1986	Renaissance jaringan saraf melalui, antara lain, Rumelhart, Hinton dan Sejnowski. Sistem Nettalk belajar membaca teks dengan keras (Bab 9).
1990	Pearl, Cheeseman, Whittaker, Spiegelhalter membawa teori probabilitas ke AI dengan jaringan Bayesian Sistem multi-agen menjadi populer.
1992	Program Tesauros TD-gammon menunjukkan keuntungan dari pembelajaran penguatan.
1993	Inisiatif RoboCup di seluruh dunia untuk membangun robot otonom yang bisa bermain sepak bola [Roba].
1995	Dari teori pembelajaran statistik, Vapnik mengembangkan mesin vektor pendukung, yang sangat penting saat ini.
1997	Komputer catur IBM Deep Blue mengalahkan juara dunia catur Gary Kasparov. Kompetisi RoboCup internasional pertama di Jepang.
2003	Robot-robot di RoboCup menunjukkan secara mengesankan apa yang mampu dicapai oleh AI dan robotika.
2006	Robotika layanan menjadi area penelitian AI utama.
2009	Mobil self-driving Google pertama yang dikendarai di jalan bebas hambatan California.
2010	Robot otonom mulai meningkatkan perilaku mereka melalui pembelajaran.
2011	IBM "Watson" mengalahkan dua juara manusia di acara permainan televisi "Jeopardy!". Watson memahami bahasa alami dan dapat menjawab pertanyaan sulit dengan sangat cepat
2015	Daimler meluncurkan truk otonom pertama di Autobahn.
	Mobil self-driving Google telah melaju lebih dari satu juta mil dan beroperasi di dalam kota.
	Pembelajaran mendalam memungkinkan klasifikasi gambar yang sangat baik.
	Lukisan dengan gaya Old Masters dapat dihasilkan secara otomatis dengan pembelajaran mendalam. AI menjadi kreatif!
2016	Program Go AlphaGo oleh Google DeepMind [SHM+16] mengalahkan juara Eropa 5:0 di bulan Januari dan Lee Sedol dari Korea, salah satu pemain Go terbaik dunia, 4:1 di bulan Maret. Teknik pembelajaran mendalam yang diterapkan pada pengenalan pola, serta pembelajaran penguatan dan pencarian pohon Monte Carlo mengarah pada kesuksesan ini.

Awal Pertama

Pada tahun 1930-an Kurt Gödel, AlonsoChurch, dan Alan Turing meletakkan dasar-dasar penting untuk logika dan ilmu komputer teoretis. Yang menarik bagi AI adalah teorema Gödel. Teorema kelengkapan menyatakan bahwa logika predikat orde pertama selesai. Ini berarti bahwa setiap pernyataan benar yang dapat dirumuskan dalam logika predikat dapat dibuktikan dengan menggunakan aturan kalkulus formal. Atas dasar ini, penyempurnaor otomatis nantinya dapat dibangun sebagai implementasi dari kalkulus formal. Dengan teorema ketidaklengkapan, Gödel menunjukkan bahwa dalam logika tingkat tinggi terdapat pernyataan benar yang tidak dapat dibuktikan[2] Dengan ini ia menemukan batasan menyakitkan dari sistem formal.

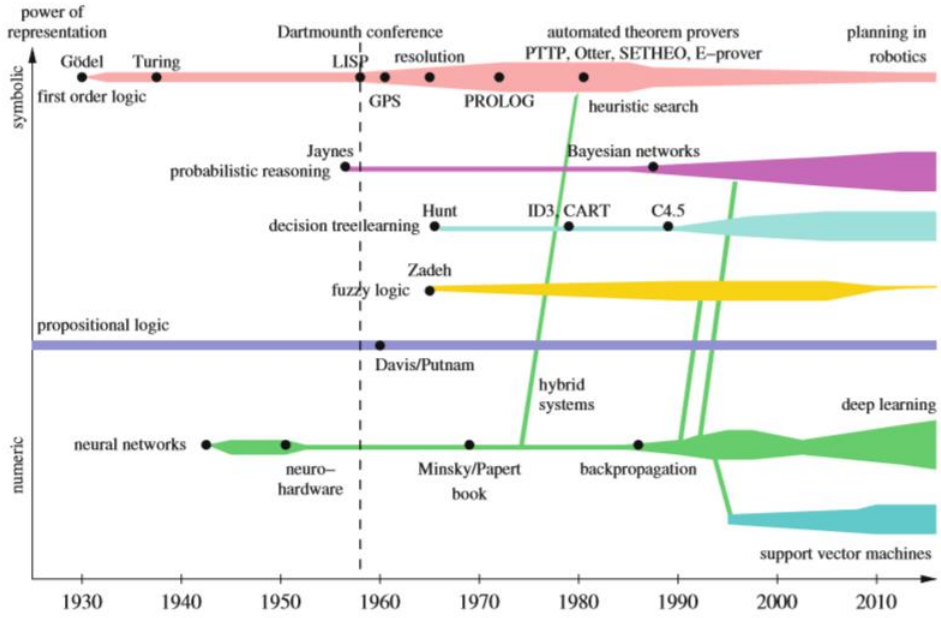
[2] Logika tingkat tinggi adalah perpanjangan dari logika predikat, di mana tidak hanya variabel, tetapi juga simbol fungsi atau predikat dapat muncul sebagai istilah dalam kuantifikasi. Memang, Gödel hanya menunjukkan bahwa sistem apa pun yang didasarkan pada logika predikat dan dapat merumuskan aritmatika Peano tidak lengkap.

Bukti Alan Turing tentang ketidakpastian masalah penghentian juga termasuk dalam periode waktu ini. Dia menunjukkan bahwa tidak ada program yang dapat memutuskan apakah program arbitrer yang diberikan (dan inputnya masing-masing) akan berjalan dalam loop tak berhingga. Dengan Turing ini juga mengidentifikasi batas untuk program cerdas. Ini mengikuti, misalnya, bahwa tidak akan pernah ada sistem verifikasi program universal.[3] Pada tahun 1940-an, berdasarkan hasil dari ilmu saraf, McCulloch, Pitts dan Hebb merancang model matematis pertama dari jaringan saraf. Namun, komputer pada waktu itu tidak memiliki kekuatan yang cukup untuk mensimulasikan otak sederhana.

1.5 LOGIKA MEMCAHKAN (HAMPIR) SEMUA MASALAH

AI sebagai ilmu praktis mekanisasi pemikiran tentu saja hanya bisa dimulai setelah ada komputer yang dapat diprogram. Ini adalah kasus di tahun 1950-an. Newell dan Simon memperkenalkan Logic Theorist, ahli teorema otomatis pertama, dan dengan demikian juga menunjukkan bahwa dengan komputer, yang sebenarnya hanya bekerja dengan angka, seseorang juga dapat memproses simbol. Pada saat yang sama McCarthy memperkenalkan, dengan bahasa LISP, bahasa pemrograman yang dibuat khusus untuk pemrosesan struktur simbolik. Kedua sistem ini diperkenalkan pada tahun 1956 di Konferensi Dartmouth yang bersejarah, yang dianggap sebagai hari lahir AI.

Di AS, LISP berkembang menjadi alat terpenting untuk implementasi sistem AI pemrosesan simbol. Setelah itu aturan inferensi logis yang dikenal sebagai resolusi berkembang menjadi kalkulus lengkap untuk logika predikat.



Gambar 1.3 Sejarah berbagai bidang AI. Lebar batang menunjukkan prevalensi penggunaan metode

Pada tahun 1970-an bahasa pemrograman logika PROLOG diperkenalkan sebagai mitra Eropa untuk LISP. PROLOG menawarkan keuntungan memungkinkan pemrograman langsung menggunakan klausa Horn, subset dari logika predikat. Seperti LISP, PROLOG memiliki tipe data untuk pemrosesan daftar yang nyaman.

[3] Pernyataan ini berlaku untuk "kebenaran total", yang menyiratkan bukti pelaksanaan yang benar serta bukti penghentian untuk setiap input yang valid.

Hingga memasuki tahun 1980-an, semangat terobosan mendominasi AI, terutama di antara banyak ahli logika. Alasan untuk ini adalah serangkaian pencapaian yang mengesankan dalam pemrosesan simbol. Dengan proyek Sistem Komputer Generasi Kelima di Jepang dan program ESPRIT di Eropa, investasi besar dilakukan untuk pembangunan komputer cerdas.

Untuk masalah kecil, pemeriksa otomatis dan sistem pemrosesan simbol lainnya terkadang bekerja dengan sangat baik. Ledakan kombinatorial dari ruang pencarian, bagaimanapun, mendefinisikan jendela yang sangat sempit untuk keberhasilan ini. Fase AI ini dijelaskan di [RN10] sebagai "Lihat, Ma, tidak ada tangan!" zaman. Karena keberhasilan ekonomi sistem AI jauh dari harapan, pendanaan untuk penelitian AI berbasis logika di Amerika Serikat turun drastis selama tahun 1980-an.

1.6 KONEKSIONISME BARU

Selama fase kekecewaan ini, ilmuwan komputer, fisikawan, dan ilmuwan kognitif mampu menunjukkan, dengan menggunakan komputer yang sekarang cukup kuat, bahwa jaringan saraf yang dimodelkan secara matematis mampu belajar menggunakan contoh-contoh pelatihan, untuk melakukan tugas-tugas yang sebelumnya membutuhkan pemrograman yang mahal. Karena toleransi kesalahan sistem tersebut dan kemampuan mereka untuk mengenali pola, keberhasilan yang cukup besar menjadi mungkin, terutama dalam pengenalan pola. Pengenalan wajah dalam foto dan pengenalan tulisan tangan adalah dua contoh aplikasi. Sistem Nettalk dapat mempelajari ucapan dari teks contoh [SR86]. Di bawah nama connectionism, subdisiplin baru AI lahir.

Koneksionisme berkembang pesat dan subsidi mengalir. Tapi segera bahkan di sini batas kelayakan menjadi jelas. Jaringan saraf dapat memperoleh kemampuan yang mengesankan, tetapi biasanya tidak mungkin untuk menangkap konsep yang dipelajari dalam rumus sederhana atau aturan logis. Upaya untuk menggabungkan jaring saraf dengan aturan logis atau pengetahuan ahli manusia menemui kesulitan besar. Selain itu, tidak ada solusi yang memuaskan untuk penataan dan modularisasi jaringan yang ditemukan.

1.7 PENALARAN DI BAWAH KETIDAKPASTIAN

AI sebagai ilmu praktis yang digerakkan oleh tujuan mencari jalan keluar dari krisis ini. Seseorang ingin menyatukan kemampuan logika untuk secara eksplisit mewakili pengetahuan dengan kekuatan jaringan saraf dalam menangani ketidakpastian. Beberapa alternatif diusulkan.

Penalaran probabilistik yang paling menjanjikan, bekerja dengan probabilitas bersyarat untuk rumus kalkulus proposisional. Sejak itu banyak sistem diagnostik dan pakar telah dibangun untuk masalah penalaran sehari-hari menggunakan jaringan Bayesian. Keberhasilan jaringan Bayesian berasal dari pemahaman intuitif mereka, semantik bersih dari probabilitas bersyarat, dan dari teori probabilitas yang didasarkan pada matematis berusia berabad-abad.

Kelemahan logika, yang hanya dapat bekerja dengan dua nilai kebenaran, dapat diselesaikan dengan logika fuzzy, yang secara pragmatis memperkenalkan banyak nilai antara nol dan satu. Meskipun saat ini landasan teoretisnya tidak sepenuhnya kokoh, ia berhasil digunakan, terutama dalam teknik kendali.

Jalan yang jauh berbeda mengarah pada sintesis logika dan jaringan saraf yang sukses dengan nama sistem hibrida. Misalnya, jaringan saraf digunakan untuk mempelajari heuristik untuk pengurangan ruang pencarian kombinatorial besar dalam penemuan bukti.

Metode pembelajaran pohon keputusan dari data juga bekerja dengan probabilitas. Sistem seperti CART, ID3 dan C4.5 dapat dengan cepat dan otomatis membangun pohon keputusan yang sangat akurat yang dapat mewakili konsep logika proposisional dan kemudian digunakan sebagai sistem pakar. Hari ini mereka adalah favorit di antara teknik pembelajaran mesin.

Sejak sekitar tahun 1990, *data mining* telah berkembang sebagai subdisiplin AI di bidang analisis data statistik untuk ekstraksi pengetahuan dari database besar. *Data mining* tidak membawa teknik baru ke AI, melainkan memperkenalkan persyaratan penggunaan database besar untuk mendapatkan pengetahuan eksplisit. Salah satu aplikasi dengan potensi pasar yang besar adalah mengarahkan kampanye iklan bisnis besar berdasarkan analisis jutaan pembelian oleh pelanggan mereka. Biasanya, teknik pembelajaran mesin seperti pembelajaran pohon keputusan ikut bermain di sini.

1.8 AGEN TERDISTRIBUSI, OTONOM DAN PEMBELAJARAN

Kecerdasan buatan terdistribusi, DAI, telah menjadi bidang penelitian yang aktif sejak sekitar tahun 1985. Salah satu tujuannya adalah penggunaan komputer paralel untuk meningkatkan efisiensi pemecah masalah. Namun, ternyata karena kompleksitas komputasi yang tinggi dari sebagian besar masalah, penggunaan sistem "cerdas" lebih menguntungkan daripada paralelisasi itu sendiri.

Pendekatan konseptual yang sangat berbeda dihasilkan dari pengembangan agen perangkat lunak otonom dan robot yang dimaksudkan untuk bekerja sama seperti tim manusia. Seperti kendaraan Braitenberg yang disebutkan di atas, ada banyak kasus di mana agen individu tidak mampu memecahkan masalah, bahkan dengan sumber daya yang tidak terbatas. Hanya kerja sama dari banyak agen yang mengarah pada perilaku cerdas atau solusi masalah. Koloni semut atau koloni rayap mampu mendirikan bangunan dengan kompleksitas arsitektur yang sangat tinggi, meskipun fakta bahwa tidak ada satu pemahaman pun yang menunjukkan bahwa semuanya cocok satu sama lain. Ini mirip dengan situasi penyediaan roti untuk kota besar seperti New York. Tidak ada badan perencanaan pusat untuk roti, melainkan ada ratusan pembuat roti yang mengetahui daerah masing-masing kota dan memanggang roti dalam jumlah yang sesuai di lokasi tersebut.

Akuisisi keterampilan aktif oleh robot adalah bidang penelitian yang menarik saat ini. Ada robot saat ini, misalnya, yang belajar berjalan atau melakukan berbagai keterampilan motorik yang berhubungan dengan sepak bola secara mandiri (Bab 10). Pembelajaran kooperatif beberapa robot untuk memecahkan masalah bersama masih dalam tahap awal.

1.9 AI TUMBUH DEWASA

Sistem di atas yang ditawarkan oleh AI saat ini bukanlah resep universal, tetapi lokakarya dengan sejumlah alat yang dapat dikelola untuk tugas yang sangat berbeda. Sebagian besar alat ini dikembangkan dengan baik dan tersedia sebagai perpustakaan perangkat lunak yang sudah jadi, seringkali dengan antarmuka pengguna yang nyaman. Pemilihan alat yang tepat dan penggunaannya yang masuk akal dalam setiap kasus diserahkan kepada pengembang AI atau insinyur pengetahuan. Seperti keahlian lainnya, ini membutuhkan pendidikan yang solid, yang dimaksudkan untuk dipromosikan oleh buku ini. Lebih dari hampir semua ilmu pengetahuan lainnya, AI bersifat interdisipliner, karena mengacu pada penemuan menarik dari berbagai bidang seperti logika, riset operasi, statistik, teknik kontrol, pemrosesan gambar, linguistik, filsafat, psikologi, dan neurobiologi.

Selain itu, ada area subjek aplikasi tertentu. Oleh karena itu, untuk berhasil mengembangkan proyek AI tidak selalu sesederhana itu, tetapi hampir selalu sangat menarik.

1.10 REVOLUSI AI

Sekitar tahun 2010 setelah sekitar 25 tahun penelitian tentang jaringan saraf, para ilmuwan dapat mulai memanen buah dari penelitian mereka. Jaringan pembelajaran mendalam yang sangat kuat, misalnya, dapat belajar mengklasifikasikan gambar dengan tingkat akurasi yang sangat tinggi. Karena klasifikasi gambar sangat penting untuk semua jenis robot pintar, ini memulai revolusi AI yang pada gilirannya mengarah ke mobil swakemudi pintar dan robot servis.

1.11 AI DAN MASYARAKAT

Ada banyak buku ilmiah dan novel fiksi ilmiah yang ditulis tentang semua aspek subjek ini. Karena kemajuan besar dalam penelitian AI, kita telah berada di ambang era robot otonom dan *Internet of Things* sejak kira-kira tahun 2005. Dengan demikian, kita semakin dihadapkan pada AI dalam kehidupan sehari-hari. Pembaca, yang mungkin akan segera bekerja sebagai pengembang AI, juga harus menghadapi dampak sosial dari pekerjaan ini. Sebagai penulis buku tentang teknik AI, saya memiliki tugas penting untuk memeriksa topik ini. Saya ingin membahas beberapa aspek AI yang sangat penting yang memiliki relevansi praktis yang besar bagi kehidupan kita.

1.12 APAKAH AI MENGHANCURKAN PEKERJAAN?

Pada Januari 2016, World Econonic Forum menerbitkan sebuah studi, yang sering dikutip oleh pers Jerman, yang memprediksi bahwa “industri 4.0 ” akan menghancurkan lebih dari lima juta pekerjaan dalam lima tahun ke depan. Perkiraan ini tidak mengejutkan karena otomatisasi di pabrik, kantor, administrasi, transportasi, di rumah dan di banyak area lain telah mendorong lebih banyak pekerjaan yang dilakukan oleh komputer, mesin, dan robot secara terus-menerus. AI telah menjadi salah satu faktor terpenting dalam tren ini sejak sekitar tahun 2010.

Agaknya, sebagian besar orang akan dengan senang hati menyerahkan pekerjaan dan tugas yang berat, kotor, dan tidak sehat secara fisik kepada mesin. Dengan demikian otomatisasi adalah berkah lengkap bagi umat manusia, dengan asumsi tidak menimbulkan efek samping negatif, seperti kerusakan lingkungan. Banyak dari pekerjaan tidak menyenangkan yang disebutkan di atas dapat dilakukan lebih cepat, lebih tepat, dan terutama lebih murah dengan mesin. Ini tampaknya hampir seperti tren menuju surga di Bumi, di mana manusia melakukan pekerjaan yang semakin tidak menyenangkan dan memiliki lebih banyak waktu untuk hal-hal baik dalam hidup. Ini sepertinya hampir seperti tren menuju surga di bumi. Kita harus melakukan pekerjaan yang semakin tidak menyenangkan dan pada gilirannya memiliki lebih banyak waktu untuk hal-hal baik dalam hidup.[4] Sementara itu, kita akan menikmati kemakmuran yang sama (atau bahkan berpotensi meningkat), karena perekonomian tidak akan menggunakan mesin-mesin ini jika mereka tidak secara nyata meningkatkan produktivitas.

Sayangnya kita tidak berada di jalan menuju surga. Selama beberapa dekade, kami telah bekerja lebih dari 40 jam per minggu, stres, mengeluh kelelahan dan penyakit lainnya,

[4] Kami, seperti ilmuwan, ilmuwan komputer dan insinyur, yang menikmatinya tentu saja dapat melanjutkan pekerjaan kami.

dan mengalami penurunan upah riil. Bagaimana ini bisa terjadi, jika produktivitas terus meningkat? Banyak ekonom mengatakan bahwa alasannya adalah tekanan persaingan. Dalam upaya untuk bersaing dan mengirimkan barang dengan harga terendah ke pasar, perusahaan perlu menurunkan biaya produksi dan dengan demikian memberhentikan pekerja. Hal ini mengakibatkan pengangguran di atas. Untuk menghindari penurunan volume penjualan karena penurunan harga, lebih banyak produk perlu diproduksi dan dijual. Ekonomi harus tumbuh!

Jika ekonomi terus tumbuh di negara di mana populasi tidak lagi tumbuh (seperti yang terjadi di sebagian besar negara industri modern), setiap warga negara harus mengkonsumsi lebih banyak. Agar itu terjadi, pasar baru harus diciptakan,[5] dan pemasaran memiliki tugas meyakinkan kita bahwa kita menginginkan produk baru. Ini—diduga—satu-satunya cara untuk "berkelanjutan" memastikan kemakmuran. Tampaknya tidak ada jalan keluar dari spiral pertumbuhan/konsumsi ini. Ini memiliki dua konsekuensi fatal. Untuk satu hal, peningkatan konsumsi ini seharusnya membuat orang lebih bahagia, tetapi memiliki efek sebaliknya: penyakit mental meningkat.

Yang lebih jelas dan, di atas segalanya, fatal, adalah efek pertumbuhan ekonomi terhadap kondisi kehidupan kita. Bukan rahasia lagi bahwa batas pertumbuhan bumi telah lama terlampaui, dan bahwa kita mengeksploitasi sumber daya alam yang tidak terbarukan secara berlebihan. Oleh karena itu kita hidup dengan mengorbankan anak-anak dan cucu-cucu kita, yang akibatnya akan memiliki kondisi kehidupan yang lebih buruk daripada yang kita miliki saat ini. Diketahui juga bahwa setiap dolar tambahan dari pertumbuhan ekonomi merupakan beban tambahan bagi lingkungan—misalnya melalui tambahan konsentrasi CO2 di atmosfer dan mengakibatkan perubahan iklim. Telah merusak dasar keberadaan kita sendiri. Jadi jelas bahwa kita harus meninggalkan jalan pertumbuhan ini demi masa depan yang layak huni. Tapi bagaimana caranya?

Mari kita pikirkan kembali jalan menuju surga yang seharusnya dipersiapkan AI untuk kita. Ternyata, seperti yang kita praktekkan, itu tidak mengarah ke surga. Memahami masalah ini dan menemukan jalan yang benar adalah salah satu tugas utama saat ini. Karena kompleksitas yang melekat, masalah ini tidak dapat sepenuhnya ditangani dalam buku teks pengantar AI. Namun, saya ingin memberi pembaca sedikit bahan untuk dipikirkan.

Meskipun produktivitas tumbuh dengan mantap di hampir semua bidang ekonomi, pekerja dituntut untuk bekerja sekeras biasanya. Mereka tidak mendapat manfaat dari peningkatan produktivitas. Jadi, kita harus bertanya, kemana keuntungannya? Jelas bukan kepada orang-orang yang kepadanya mereka berutang, yaitu para pekerja. Sebagai gantinya, sebagian dari keuntungan dihabiskan untuk investasi dan dengan demikian untuk pertumbuhan lebih lanjut dan sisanya diambil oleh pemilik modal, sementara karyawan bekerja pada jam yang sama untuk menurunkan upah riil. Hal ini menyebabkan konsentrasi modal yang terus meningkat di antara segelintir orang kaya dan bank swasta, sementara di sisi lain meningkatnya kemiskinan di seluruh dunia menciptakan ketegangan politik yang mengakibatkan perang, pengusiran, dan pelarian.

Apa yang hilang adalah distribusi keuntungan yang adil dan merata. Bagaimana ini bisa dicapai? Politisi dan ekonom terus berusaha mengoptimalkan sistem ekonomi kita, tetapi politik belum menawarkan solusi yang berkelanjutan, dan terlalu sedikit ekonom yang menyelidiki pertanyaan ekonomi yang sangat menarik ini. Jelas upaya untuk

[5] Banyak program pendanaan Kementerian Pendidikan dan Penelitian Uni Eropa dan Jerman, misalnya, mengharuskan para ilmuwan yang mengajukan proposal menunjukkan bukti bahwa penelitian mereka akan membuka pasar baru.

mengoptimalkan parameter sistem ekonomi kapitalis kita saat ini tidak mengarah pada distribusi kekayaan yang lebih adil, tetapi sebaliknya.

Inilah sebabnya mengapa para ekonom dan ilmuwan keuangan harus mulai mempertanyakan sistem dan mencari alternatif. Kita harus bertanya pada diri sendiri bagaimana mengubah aturan dan hukum ekonomi sehingga semua orang mendapat untung dari peningkatan produktivitas. Komunitas ekonom dan ilmuwan keberlanjutan yang berkembang telah menawarkan solusi menarik, beberapa di antaranya akan saya jelaskan secara singkat di sini.

Masalah Nomor Satu adalah penciptaan uang kertas oleh bank. Uang baru—yang dibutuhkan antara lain untuk menjaga pertumbuhan ekonomi kita—sekarang sedang diciptakan oleh bank-bank swasta. Hal ini dimungkinkan oleh fakta bahwa bank hanya memiliki sebagian kecil, yaitu rasio cadangan kas minimum, dari uang yang mereka berikan sebagai pinjaman. Di UE pada tahun 2016, rasio cadangan kas minimum adalah satu persen. Negara kemudian meminjam uang ini dari bank swasta dalam bentuk obligasi pemerintah dan dengan demikian jatuh ke dalam hutang. Ini adalah bagaimana krisis utang pemerintah kita saat ini telah berkembang. Masalah ini dapat diselesaikan dengan mudah dengan melarang penciptaan uang oleh bank dengan meningkatkan cadangan kas minimum 100%. Bank sentral negara kemudian akan mendapatkan kembali monopoli penciptaan uang, dan uang yang baru dibuat dapat digunakan langsung oleh negara untuk tujuan kesejahteraan sosial. Harus jelas bahwa tindakan sederhana ini akan secara signifikan meringankan masalah utang publik.

Komponen menarik lebih lanjut dari reformasi ekonomi semacam itu dapat berupa konversi sistem suku bunga saat ini menjadi apa yang disebut tatanan ekonomi alami, dan pengenalan "ekonomi untuk kebaikan bersama" dan ekonomi biofisik. Implementasi praktis ekonomi untuk kebaikan bersama akan melibatkan reformasi pajak, yang elemen terpentingnya adalah penghapusan pajak penghasilan dan secara substansial meningkatkan pajak pertambahan nilai untuk konsumsi energi dan sumber daya. Dengan demikian, kita akan sampai pada dunia manusia yang sangat makmur dan berkelanjutan dengan lebih sedikit kerusakan lingkungan dan lebih banyak perdagangan lokal. Pembaca dapat mempelajari literatur dan menilai apakah ide-ide yang dikutip di sini menarik dan, jika perlu, membantu membuat perubahan yang diperlukan.

Untuk mengakhiri bagian ini, saya ingin mengutip fisikawan terkenal Stephen Hawking. Wawancara berbasis komunitas di www.reddit.com memberikan jawaban berikut apakah dia memiliki pemikiran tentang pengangguran yang disebabkan oleh otomatisasi:

Jika mesin menghasilkan semua yang kita butuhkan, hasilnya akan tergantung pada bagaimana hal itu didistribusikan. Setiap orang dapat menikmati kehidupan rekreasi yang mewah jika kekayaan yang dihasilkan mesin dibagikan, atau kebanyakan orang dapat menjadi sangat miskin jika pemilik mesin berhasil melobi menentang redistribusi kekayaan. Sejauh ini, tren tampaknya mengarah pada opsi kedua, dengan teknologi mendorong ketimpangan yang semakin meningkat.

Kutipan Hawking lainnya juga pas. Selama wawancara yang sama[6], untuk pertanyaan seorang profesor AI tentang ide moral mana yang harus dia berikan kepada murid-muridnya, Hawking menjawab:

… Harap dorong mahasiswa Anda untuk berpikir tidak hanya tentang cara membuat AI, tetapi juga tentang bagaimana memastikan penggunaan yang bermanfaat.

[6] https://www.reddit.com/user/Prof-Stephen-Hawking.

Sebagai konsekuensinya, kita harus mempertanyakan kewajaran aplikasi AI seperti ekspor rudal jelajah cerdas ke negara-negara Arab “sekutu”, penyebaran robot tempur humanoid, dll.

1.13 AI DAN TRANSPORTASI

Dalam 130 tahun terakhir, insinyur industri otomotif telah membuat langkah besar. Di Jerman, satu dari setiap dua orang memiliki mobil sendiri. Mobil-mobil ini sangat andal. Hal ini membuat kita sangat mobile dan kita menggunakan mobilitas yang sangat nyaman ini dalam pekerjaan, kehidupan sehari-hari, dan rekreasi. Selain itu, kita dapat memakainya dengan bergantung. Saat ini, kita tidak dapat berkendara tanpa kendaraan bermotor, terutama di daerah pedesaan dengan infrastruktur transportasi umum yang lemah, seperti misalnya di Swabia Atas, tempat penulis dan muridnya tinggal.

Tahap selanjutnya dari peningkatan kenyamanan dalam transportasi jalan raya sekarang sudah dekat. Dalam beberapa tahun, kita akan dapat membeli mobil listrik self-driving, yaitu mobil robot, yang secara otomatis akan membawa kita ke hampir semua tujuan. Semua penumpang di dalam mobil robotik itu akan bisa membaca, bekerja atau tidur selama perjalanan. Ini sudah mungkin dilakukan di angkutan umum, tetapi penumpang di mobil robot akan dapat melakukannya kapan saja dan di rute apa pun.

Kendaraan otonom yang dapat beroperasi secara mandiri juga dapat melakukan perjalanan tanpa penumpang. Ini akan mengarah pada peningkatan kenyamanan lainnya: taksi robotik. Melalui aplikasi smartphone, kami akan dapat memesan taksi yang optimal, dalam hal ukuran dan peralatan, untuk tujuan transportasi apa pun yang memungkinkan. Kita akan dapat memilih apakah kita ingin bepergian sendiri di dalam taksi atau apakah kita mau berbagi tumpangan dengan penumpang lain. Kita tidak akan membutuhkan mobil kita sendiri lagi. Semua tanggung jawab dan pengeluaran terkait, seperti pengisian bahan bakar, layanan teknis, pembersihan, pencarian parkir, pembelian dan penjualan, sewa garasi, dll. tidak berlaku, yang menghemat uang dan tenaga.

Selain keuntungan langsung dalam kenyamanan dan kemudahan, mobil robot akan menawarkan keuntungan signifikan lainnya. Misalnya, menurut studi McKinsey, kita akan membutuhkan mobil yang jauh lebih sedikit dan, di atas segalanya, tempat parkir yang jauh lebih sedikit di era mobil yang dapat mengemudi sendiri, yang akan mengarah pada pengurangan besar dalam konsumsi sumber daya. Menurut penelitian Lawrence Berkeley National Laboratory, mobil listrik self-driving akan menyebabkan pengurangan 90% emisi rumah kaca per mil penumpang karena efisiensi energi kendaraan dan kesesuaian yang dioptimalkan antara kendaraan dan tujuannya. Karena pemanfaatan sumber daya yang optimal, taksi robotik akan jauh lebih ramah lingkungan daripada, misalnya, bus berat, yang sering berjalan dengan kapasitas rendah, terutama di daerah pedesaan. Secara keseluruhan, taksi robot akan berkontribusi secara dramatis terhadap penghematan energi dan dengan demikian, antara lain, untuk perbaikan yang signifikan dalam masalah CO_2 dan iklim.

Keselamatan penumpang akan jauh lebih tinggi daripada saat ini. Para ahli saat ini memperkirakan tingkat kecelakaan di masa depan antara nol dan sepuluh persen dibandingkan dengan hari ini. Mengemudi secara emosional ("kemarahan di jalan"), mengemudi yang terganggu dan mengemudi di bawah pengaruh obat-obatan dan alkohol tidak akan ada lagi.

Pengemudi taksi kehilangan pekerjaan sering disebut sebagai kerugian dari mobil robot. Hampir dapat dipastikan bahwa tidak akan ada lagi pengemudi taksi mulai sekitar tahun 2030 dan seterusnya, tetapi itu tidak selalu menjadi masalah. Seperti yang telah

dijelaskan pada bagian sebelumnya, masyarakat kita hanya perlu menyikapi produktivitas yang baru diperoleh dengan baik.

Selain banyak keuntungan yang disebutkan di atas, mobil robot memiliki dua masalah kritis. Pertama, apa yang disebut efek pantulan akan meniadakan setidaknya sebagian keuntungan dalam penghematan sumber daya, energi, dan waktu. Waktu mengemudi yang lebih singkat juga lebih nyaman dan lebih murah mengemudi akan menggoda kita untuk mengemudi lebih banyak. Di sini kenakan semua undangan untuk refleksi kritis.

Masalah lain yang harus kita tangani dengan serius adalah bahwa mobil robot perlu terhubung ke jaringan. Pada prinsipnya, ini memberi peretas dan teroris kemampuan untuk mengakses dan memanipulasi kontrol kendaraan melalui lubang keamanan di protokol jaringan mereka. Jika seorang peretas berhasil melakukan ini sekali, ia dapat mengulangi serangan dalam skala besar, yang berpotensi menghentikan seluruh armada kendaraan, menyebabkan kecelakaan, memata-matai penumpang kendaraan, atau memulai tindakan kriminal lainnya. Di sini, seperti di area lain seperti otomatisasi rumah dan Internet of Things, pakar keamanan TI akan diperlukan untuk memastikan jaminan keamanan setinggi mungkin dengan menggunakan alat perdagangan seperti metode kriptografi. Omong-omong, algoritme pembelajaran mesin yang ditingkatkan akan berguna dalam mendeteksi serangan peretasan.

1.14 LAYANAN ROBOTIKA

Dalam beberapa tahun, tak lama setelah mobil self-driving, umpan konsumsi berikutnya di rak-rak toko elektronik adalah robot servis. Baru-baru ini, anak perusahaan Google, Boston Dynamics, memberikan contoh yang mengesankan dalam robot humanoidnya Atlas[7] Seperti mobil baru, robot servis menawarkan keuntungan besar dalam kenyamanan dan kemudahan yang mungkin ingin kita nikmati. Kita hanya perlu membayangkan robot seperti itu dengan patuh membersihkan dan menggosok setelah pesta dari malam hingga pagi tanpa menggerutu. Atau pikirkan tentang bantuan yang dapat diberikan oleh robot bantuan seperti Marvin, ditunjukkan pada Gambar 1.4 kepada orang tua[8] atau orang cacat.

Namun, berbeda dengan mobil robot, manfaat ini datang dengan pengorbanan yang lebih mahal. Pasar yang benar-benar baru akan diciptakan, lebih banyak sumber daya alam dan lebih banyak energi akan dikonsumsi, dan bahkan tidak pasti bahwa kehidupan orang akan disederhanakan dengan penggunaan robot layanan di semua bidang. Salah satu aplikasi pertama untuk robot seperti Atlas, yang dikembangkan oleh Boston Dynamics dalam kontrak dengan Google, mungkin adalah pertempuran militer.

[7] https://youtu.be/rVlhMGQgDkY.

[8] Dalam pergeseran demografis yang akan datang, robot bantuan bisa menjadi penting bagi orang tua dan dengan demikian bagi seluruh masyarakat kita

Oleh karena itu, semakin penting bahwa, sebelum robot-robot ini datang ke pasar, kita terlibat dalam wacana sosial tentang topik ini. Film fiksi ilmiah, seperti “Ex Machina” (2015) dengan android wanitanya, “I, Robot” (2004) yang mengerikan, atau “Robot and Frank” (2012) yang lucu, yang menggambarkan sisi menyenangkan dari robot layanan sebagai pembantu orang tua, juga dapat berkontribusi pada diskusi semacam itu.

Gambar 1.4 Robot bantuan Marvin, yang digunakan dalam proyek penelitian AsRoBe

1.15 AGEN

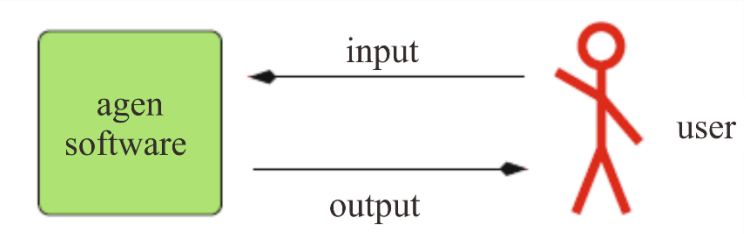
Meskipun istilah agen cerdas bukanlah hal baru bagi AI, hanya dalam beberapa tahun terakhir istilah ini menjadi terkenal antara lain melalui. Agen menunjukkan secara umum suatu sistem yang memproses informasi dan menghasilkan output dari input. Agen-agen ini dapat diklasifikasikan dalam berbagai cara.

Dalam ilmu komputer klasik, agen perangkat lunak terutama digunakan (Gambar 1.5). Dalam hal ini agen terdiri dari program yang menghitung hasil dari input pengguna. Dalam robotika, di sisi lain, agen perangkat keras (juga disebut robot otonom) digunakan, yang juga memiliki sensor dan aktuator yang tersedia (Gambar 1.6). Agen dapat melihat lingkungannya dengan sensor. Dengan aktuator ia melakukan tindakan dan mengubah lingkungannya.

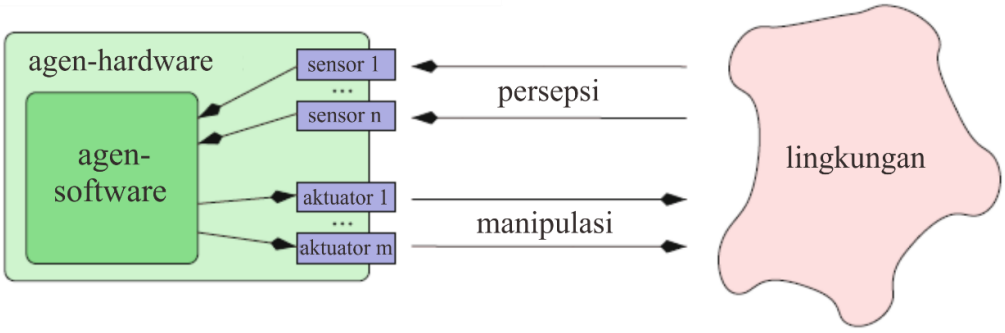
Sehubungan dengan kecerdasan agen, ada perbedaan antara agen refleks, yang hanya bereaksi terhadap input, dan agen dengan memori, yang juga dapat memasukkan masa lalu dalam keputusan mereka. Misalnya, robot penggerak yang melalui sensornya mengetahui posisi persisnya (dan waktu) tidak memiliki cara, sebagai agen refleks, untuk menentukan kecepatannya. Namun, jika ia menyimpan posisi, pada langkah waktu yang pendek dan diskrit, ia dapat dengan mudah menghitung kecepatan rata-ratanya dalam interval waktu sebelumnya.

Jika agen refleks dikendalikan oleh program deterministik, itu mewakili fungsi dari himpunan semua input ke himpunan semua output. Agen dengan memori, di sisi lain, pada umumnya bukan fungsi. Mengapa? (Lihat Latihan 1.5.) Agen refleks cukup dalam kasus di mana masalah yang harus dipecahkan melibatkan proses keputusan Markov. Ini adalah proses di mana hanya keadaan saat ini yang diperlukan untuk menentukan tindakan selanjutnya yang optimal (lihat Bab 10).

Sebuah mobile robot yang seharusnya berpindah dari ruangan 112 ke ruangan 179 dalam sebuah gedung melakukan aksi yang berbeda dengan robot yang harus berpindah ke ruangan 105. Dengan kata lain, aksinya bergantung pada tujuannya. Agen semacam itu disebut berbasis tujuan.



Gambar 1.5 Agen perangkat lunak dengan interaksi pengguna



Gambar 1.6 Agen perangkat keras

Contoh 1.1

Filter spam adalah agen yang memasukkan email masuk ke dalam kategori yang diinginkan atau tidak diinginkan (spam), dan menghapus email yang tidak diinginkan. Tujuannya sebagai agen berbasis sasaran adalah menempatkan semua email dalam kategori yang tepat. Dalam menjalankan tugas yang tidak begitu sederhana ini, agen terkadang dapat membuat kesalahan. Karena tujuannya adalah untuk mengklasifikasikan semua email dengan benar, ia akan berusaha membuat kesalahan sesedikit mungkin. Namun, itu tidak selalu apa yang ada dalam pikiran pengguna. Mari kita bandingkan dua agen berikut ini. Dari 1.000 email, Agen 1 hanya membuat 12 kesalahan. Agen 2 di sisi lain membuat 38 kesalahan dengan 1.000 email yang sama. Apakah karena itu lebih buruk dari Agen 1? Kesalahan kedua agen ditampilkan secara lebih rinci dalam tabel berikut, yang disebut "matriks kebingungan":

Tabel 1.2 tabel kesalahan kedua agen

Agent 1:

		correct class	
		wanted	spam
spam filter decides	wanted	189	1
	spam	11	799

Agent 2:

		correct class	
		wanted	spam
spam filter decides	wanted	200	38
	spam	0	762

Agen 1 sebenarnya membuat lebih sedikit kesalahan daripada Agen 2, tetapi beberapa kesalahan itu parah karena pengguna kehilangan 11 email yang berpotensi penting. Karena dalam kasus ini terdapat dua jenis kesalahan dengan tingkat keparahan yang berbeda, setiap kesalahan harus diberi bobot dengan faktor biaya yang sesuai.

Jumlah semua kesalahan tertimbang memberikan total biaya yang disebabkan oleh keputusan yang salah. Tujuan dari agen berbasis biaya adalah untuk meminimalkan biaya

keputusan yang salah dalam jangka panjang, yaitu rata-rata. Kita akan terbiasa dengan sistem diagnosis medis LEXMED sebagai contoh agen berbasis biaya.

Secara analog, tujuan dari agen berbasis utilitas adalah untuk memaksimalkan utilitas yang berasal dari keputusan yang benar dalam jangka panjang, yaitu rata-rata. Jumlah dari semua keputusan yang diboboti oleh faktor utilitas masing-masing memberikan utilitas total.

Yang menarik dalam AI adalah agen Pembelajaran, yang mampu mengubah diri mereka sendiri dengan memberikan contoh pelatihan atau melalui umpan balik positif atau negatif, sehingga utilitas rata-rata dari tindakan mereka tumbuh dari waktu ke waktu (lihat Bab 8).

Seperti yang disebutkan, agen terdistribusi semakin banyak digunakan, yang kecerdasannya tidak terlokalisasi dalam satu agen, melainkan hanya dapat dilihat melalui kerja sama banyak agen.

Desain agen berorientasi, bersama dengan tujuannya, kuat terhadap lingkungannya, atau secara bergantian gambarannya tentang lingkungan, yang sangat bergantung pada sensornya. Lingkungan dapat diamati jika agen selalu mengetahui keadaan dunia secara lengkap. Jika tidak, lingkungan hanya dapat diamati sebagian. Jika suatu tindakan selalu mengarah pada hasil yang sama, maka lingkungannya bersifat deterministik. Kalau tidak, itu nondeterministik. Dalam lingkungan diskrit hanya sedikit keadaan dan tindakan yang terjadi, sedangkan lingkungan yang berkesinambungan menghasilkan banyak keadaan atau tindakan tanpa batas.

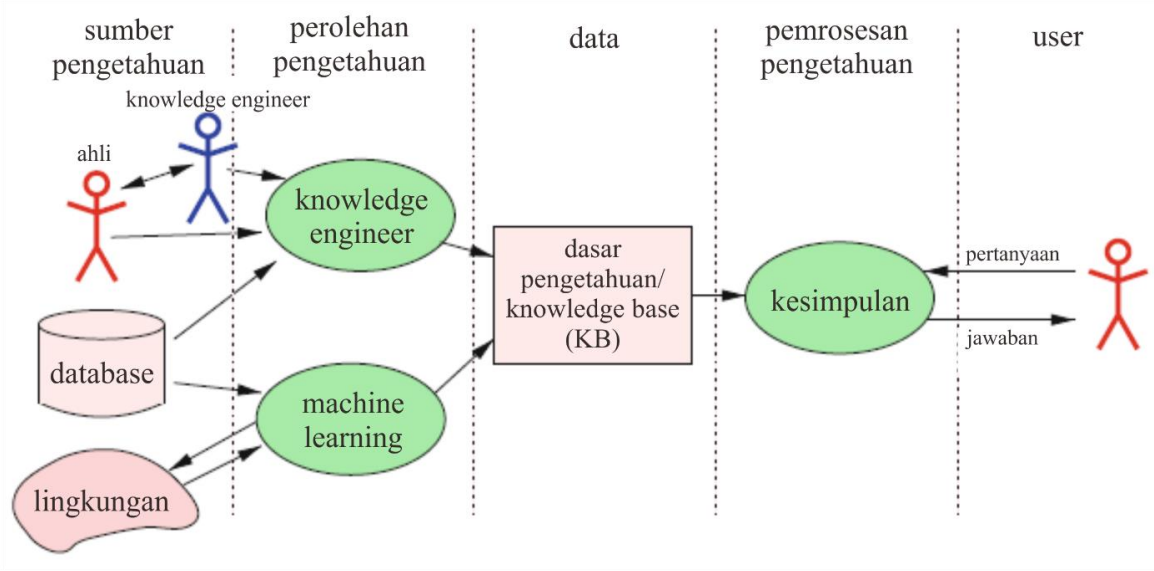
1.16 SISTEM BERBASIS PENGETAHUAN

Agen adalah program yang mengimplementasikan pemetaan dari persepsi ke tindakan. Untuk agen sederhana, cara melihat masalah ini sudah cukup. Untuk aplikasi yang kompleks di mana agen harus dapat mengandalkan sejumlah besar informasi dan dimaksudkan untuk melakukan tugas yang sulit, pemrograman agen bisa sangat mahal dan tidak jelas bagaimana melanjutkannya. Di sini AI menyediakan jalur yang jelas untuk diikuti yang akan sangat menyederhanakan pekerjaan.

Pertama, kita memisahkan pengetahuan dari sistem atau program, yang menggunakan pengetahuan untuk, misalnya, mencapai kesimpulan, menjawab pertanyaan, atau membuat rencana. Sistem ini disebut mekanisme inferensi. Pengetahuan tersebut disimpan dalam basis pengetahuan (KB). Akuisisi pengetahuan dalam basis pengetahuan dilambangkan dengan Rekayasa Pengetahuan dan didasarkan pada berbagai sumber pengetahuan seperti pakar manusia, insinyur pengetahuan, dan basis data. Sistem pembelajaran aktif juga dapat memperoleh pengetahuan melalui eksplorasi aktif dunia (lihat Bab 10). Pada Gambar 1.7 arsitektur umum sistem berbasis pengetahuan disajikan

Bergerak menuju pemisahan pengetahuan dan kesimpulan memiliki beberapa keuntungan penting. Pemisahan pengetahuan dan inferensi dapat memungkinkan sistem inferensi diimplementasikan dengan cara yang sebagian besar tidak bergantung pada aplikasi. Misalnya, penerapan sistem pakar medis untuk penyakit lain jauh lebih mudah dengan mengganti basis pengetahuan daripada memprogram sistem yang sama sekali baru.

Melalui decoupling basis pengetahuan dari inferensi, pengetahuan dapat disimpan secara deklaratif. Dalam basis pengetahuan hanya ada deskripsi pengetahuan, yang independen dari sistem inferensi yang digunakan. Tanpa pemisahan yang jelas ini, pengetahuan dan pemrosesan langkah-langkah inferensi akan terjalin, dan setiap perubahan pada pengetahuan akan sangat mahal.



Gambar 1.7 Struktur sistem pemrosesan pengetahuan klasik

Bahasa formal sebagai antarmuka yang nyaman antara manusia dan mesin cocok untuk representasi pengetahuan dalam basis pengetahuan. Dalam bab-bab berikut kita akan mengenal serangkaian bahasa tersebut. Pertama, dalam Bab. 2 dan 3 ada kalkulus proposisional dan logika predikat orde pertama (PL1). Tetapi formalisme lain seperti logika probabilistik dan pohon keputusan juga disajikan. Kita mulai dengan kalkulus proposisional dan sistem inferensi terkait. Berdasarkan itu, kami akan menyajikan logika predikat, bahasa yang kuat yang dapat diakses oleh mesin dan sangat penting dalam AI.

Sebagai contoh untuk sistem berbasis pengetahuan skala besar, kami ingin merujuk ke agen perangkat lunak "Watson". Dikembangkan di IBM bersama dengan sejumlah universitas, Watson adalah program penjawab pertanyaan, yang dapat diisi dengan petunjuk yang diberikan dalam bahasa alami. Ia bekerja pada basis pengetahuan yang terdiri dari empat terabyte penyimpanan hard disk, termasuk teks lengkap Wikipedia. Watson dikembangkan dalam proyek DeepQA IBM yang dicirikan dalam sebagai berikut:

Proyek DeepQA di IBM membentuk tantangan besar dalam Ilmu Komputer yang bertujuan untuk menggambarkan bagaimana aksesibilitas yang luas dan berkembang dari konten bahasa alami dan integrasi dan kemajuan Pemrosesan Bahasa Alami, Pengambilan Informasi, Pembelajaran Mesin, Representasi dan Penalaran Pengetahuan, dan paralel secara masif komputasi dapat mendorong teknologi *Question Answering* otomatis domain terbuka ke titik di mana teknologi ini secara jelas dan konsisten menyaingi kinerja manusia terbaik.

Dalam acara kuis televisi AS "Jeopardy!", Pada Februari 2011, Watson mengalahkan dua juara manusia Brad Rutter dan Ken Jennings dalam dua pertandingan, pertandingan poin gabungan dan memenangkan harga satu juta dolar. Salah satu kekuatan khusus Watson adalah reaksinya yang sangat cepat terhadap pertanyaan sehingga Watson sering menekan bel (menggunakan solenoida) lebih cepat daripada pesaing manusianya dan kemudian mampu memberikan jawaban pertama untuk pertanyaan itu.

Kinerja tinggi dan waktu reaksi yang singkat dari Watson disebabkan oleh implementasi pada 90 server IBM Power 750, yang masing-masing berisi 32 prosesor, menghasilkan 2880 prosesor paralel.

1.17 LATIHAN

Latihan 1.1

Uji beberapa chatterbot yang tersedia di internet. Mulai misalnya dengan www.hs-weingarten.de/*ertel/aibook dalam kumpulan tautan di bawah Turingtest/Chatterbots, atau di www.simonlaven.com atau www.alicebot.org. Tulis pertanyaan awal dan ukur waktu yang dibutuhkan, untuk setiap program, sampai Anda tahu pasti bahwa itu bukan manusia.

Latihan 1.2

Di www.pandorabots.com Anda akan menemukan server tempat Anda dapat membangun chatterbot dengan bahasa markup AIML dengan cukup mudah. Bergantung pada tingkat minat Anda, kembangkan chatterbot sederhana atau kompleks, atau ubah yang sudah ada.

Latihan 1.3

Berikan alasan ketidaksesuaian tes Turing sebagai definisi "kecerdasan buatan" dalam AI praktis.

Latihan 1.4

Banyak proses inferensi terkenal, proses pembelajaran, dll adalah NP-lengkap atau bahkan tidak dapat diputuskan. Apa artinya ini bagi AI?

Latihan 1.5

(a) Mengapa agen deterministik dengan memori bukan merupakan fungsi dari himpunan semua input ke himpunan semua output, dalam pengertian matematis?

(b) Bagaimana seseorang dapat mengubah agen dengan memori, atau memodelkannya, sehingga menjadi ekuivalen dengan suatu fungsi tetapi tidak kehilangan memorinya?

Latihan 1.6

Misalkan ada agen dengan memori yang dapat bergerak di dalam pesawat. Dari sensornya, ia menerima detak jam dari interval reguler Δt posisi yang tepat (x, y) dalam koordinat Cartesian.

(a) Berikan rumus yang dapat digunakan agen untuk menghitung kecepatannya dari waktu sekarang t dan pengukuran sebelumnya dari $t - \Delta t$.

(b) Bagaimana agen harus diubah agar dapat juga menghitung percepatannya? Berikan juga rumus di sini.

Latihan 1.7

(a) Tentukan untuk kedua agen pada Contoh 1.1 biaya yang ditimbulkan oleh kesalahan dan bandingkan hasilnya. Asumsikan di sini bahwa menghapus email spam secara manual membutuhkan biaya satu sen dan mengambil email yang dihapus, atau kehilangan email, membutuhkan biaya satu dolar.

(b) Tentukan untuk kedua agen keuntungan yang diciptakan oleh klasifikasi yang benar dan bandingkan hasilnya. Asumsikan bahwa untuk setiap email yang diinginkan dikenali, keuntungan satu dolar diperoleh dan untuk setiap email spam yang dihapus dengan benar, keuntungan satu sen.

BAB 2
LOGIKA PROPORSI

Dalam logika proposisional, seperti namanya, proposisi dihubungkan oleh operator logika. Pernyataan "jalan basah" adalah proposisi, seperti halnya "hujan". Kedua proposisi ini dapat dihubungkan untuk membentuk proposisi baru

jika hujan jalan basah.

Ditulis lebih formal

Hujan ⟹ jalanan basah.

Notasi ini memiliki keuntungan bahwa proposisi unsur muncul kembali dalam bentuk yang tidak berubah. Agar kita dapat bekerja dengan logika proposisional secara tepat, kita akan mulai dengan definisi himpunan semua rumus logika proposisional.

2.1 SINTAKS

Definisi 2.1
Diketahui Op = {¬, ∧, ∨, ⟹, ⇔, (,)} menjadi himpunan operator logika dan ∑ himpunan simbol. Himpunan Op, ∑ dan {t, f} berpasangan terpisah. ∑ disebut tanda tangan dan elemen-elemennya adalah variabel proposisi. Himpunan rumus logika proposisional sekarang didefinisikan secara rekursif:

- t dan f adalah rumus (atom).
- Semua variabel proposisi, yaitu semua elemen dari ∑, adalah rumus (atom).
- Jika A dan B adalah rumus, maka ¬A,(A), A ∧ B, A ∨ B, A ⟹ B, A ⇔ B juga rumus.

Definisi rekursif yang elegan dari himpunan semua rumus ini memungkinkan kita untuk menghasilkan banyak rumus yang tak berhingga. Misalnya, diberikan ∑ ={ A, B, C},

A ∧ B, A ∧ B ∧ C, A ∧ A ∧ A, C ∧ B ∨ A, (¬ A ∧ B) ⟹ (¬ C ∨ A)

adalah rumus. (((A)) ∨ B) juga merupakan rumus yang benar secara sintaksis.

Definisi 2.2
Kami membaca simbol dan operator dengan cara berikut:

t:	"true"	
f:	"false"	
¬A:	"not A"	*(negasiasin)*
A ∧ B:	" A and B"	*(konjungsi)*
A ∨ B:	"A or B"	*(fisjungsi)*
A⇒B:	"if A then B"	*(implikasi (juga disebut implikasi material))*
A ⇔ B:	"A if and only if B"	*(equivalence)*

Rumus yang didefinisikan dengan cara ini sejauh ini murni konstruksi sintaksis tanpa makna. Kami masih kehilangan semantik.

2.2 SEMANTIK

Dalam logika proposisional ada dua nilai kebenaran: t untuk "benar" dan f untuk "salah". Kita mulai dengan sebuah contoh dan bertanya pada diri sendiri apakah rumus A ^ B benar. Jawabannya adalah: tergantung apakah variabel A dan B benar. Misalnya, jika A singkatan dari "Hari ini hujan" dan B untuk "Hari ini dingin" dan keduanya benar, maka A^B benar. Namun, jika B menyatakan "Hari ini panas" (dan ini salah), maka A ^ B salah. Kita

jelas harus menetapkan nilai kebenaran yang mencerminkan keadaan dunia ke variabel proposisi. Oleh karena itu kami mendefinisikan :

Definisi 2.3
Sebuah pemetaan $I : \sum \rightarrow \{t, f\}$, yang memberikan nilai kebenaran pada setiap variabel proposisi, disebut interpretasi
Karena setiap variabel proposisi dapat mengambil dua nilai kebenaran, setiap rumus logika proposisional dengan n variabel berbeda memiliki 2n interpretasi yang berbeda. Kami mendefinisikan nilai kebenaran untuk operasi dasar dengan menunjukkan semua kemungkinan interpretasi dalam tabel kebenaran (lihat Tabel 2.1).

Tabel 2.1 Definisi operator logika berdasarkan tabel kebenaran

A	B	(A)	$\neg A$	$A \wedge B$	$A \vee B$	$A \Longrightarrow B$	$A \Longleftrightarrow B$
t	t	t	f	t	t	t	t
t	f	t	f	f	t	f	f
f	t	f	t	f	t	t	f
f	f	f	t	f	f	t	t

Rumus kosong berlaku untuk semua interpretasi. Untuk menentukan nilai kebenaran untuk rumus kompleks, kita juga harus menentukan urutan operasi untuk operator logika. Jika ekspresi dikurung, istilah dalam kurung dievaluasi terlebih dahulu. Untuk rumus yang tidak dikurung, prioritasnya diurutkan sebagai berikut, dimulai dengan ikatan terkuat: $\neg, \wedge, \vee, \Longrightarrow, \Longleftrightarrow$
Untuk membedakan dengan jelas antara persamaan rumus dan persamaan sintaksis, kita definisikan

Definisi 2.4
Dua rumus F dan G disebut ekuivalen secara semantik jika memiliki nilai kebenaran yang sama untuk semua interpretasi. Kami menulis $F \equiv G$.
Kesepadanan semantik berfungsi di atas segalanya untuk dapat menggunakan bahasa meta, yaitu bahasa alami, untuk berbicara tentang bahasa objek, yaitu logika. Pernyataan *"$A \equiv B$"* menyampaikan bahwa kedua rumus A dan B adalah ekuivalen secara semantik. Pernyataan *"$A \Longleftrightarrow B$"* di sisi lain adalah objek sintaksis dari bahasa formal logika proposisional.

Menurut jumlah interpretasi di mana suatu rumus benar, kita dapat membagi rumus ke dalam kelas-kelas berikut:

Definisi 2.5
Suatu rumus disebut

- Memuaskan jika benar untuk setidaknya satu interpretasi.
- Valid secara logis atau cukup valid jika benar untuk semua interpretasi. Rumus sejati juga disebut tautologi.
- Tidak memuaskan jika tidak benar untuk interpretasi apapun. Setiap interpretasi yang memenuhi rumus disebut model rumus.

Jelaslah bahwa negasi dari setiap rumus yang berlaku umum tidak memuaskan.
Negasi dari rumus F yang memuaskan, tetapi tidak valid secara umum adalah memuaskan.
Kami sekarang dapat membuat tabel kebenaran untuk rumus kompleks untuk memastikan nilai kebenarannya. Kami segera menerapkannya dengan menggunakan persamaan rumus yang penting dalam praktik.

Teorema 2.1

Operasi ∧, ∨ bersifat komutatif dan asosiatif, dan persamaan berikut umumnya valid:

$\neg A \vee B$	$\Leftrightarrow$	$A \Rightarrow B$	*(Implikasi)*
$A \Rightarrow B$	$\Leftrightarrow$	$\neg B \Rightarrow \neg A$	*(kontraposisi)*
$(A \Rightarrow B) \wedge (B \Rightarrow A)$	$\Leftrightarrow$	$(A \Leftrightarrow B)$	*(Ekuivalen)*
$\neg (A \wedge B)$	$\Leftrightarrow$	$\neg A \vee \neg B$	*(Hukum De Morgan)*
$\neg (A \vee B)$	$\Leftrightarrow$	$\neg A \wedge \neg B$	
$A \vee (B \wedge C)$	$\Leftrightarrow$	$(A \vee B) \wedge (A \vee C)$	*(hukum distribusi)*
$A \wedge (B \vee C)$	$\Leftrightarrow$	$(A \wedge B) \vee (A \wedge C)$	
$A \vee \neg A$	$\Leftrightarrow$	w	*(Tautologi)*
$A \wedge \neg A$	$\Leftrightarrow$	f	*(kontradiksi)*
$A \vee f$	$\Leftrightarrow$	A	
$A \vee w$	$\Leftrightarrow$	w	
$A \wedge f$	$\Leftrightarrow$	f	
$A \wedge w$	$\Leftrightarrow$	A	

Pembuktian Untuk menunjukkan ekivalensi pertama, kita menghitung tabel kebenaran untuk $\neg A \vee B$ *dan* $A \Longrightarrow B$ dan melihat bahwa nilai kebenaran untuk kedua rumus adalah sama untuk semua interpretasi. Oleh karena itu, rumusnya setara, dan dengan demikian semua nilai kolom terakhir adalah *"t"*.

A	B	$\neg A$	$\neg A \vee B$	$A \Longrightarrow B$	$(\neg A \vee B) \Leftrightarrow (A \Longrightarrow B)$
t	*t*	*f*	*t*	*t*	*t*
t	*f*	*f*	*f*	*f*	*t*
f	*t*	*t*	*t*	*t*	*t*
f	*f*	*t*	*t*	*t*	*t*

Bukti untuk ekivalensi lainnya serupa dan direkomendasikan sebagai latihan untuk pembaca (Latihan 2.2).

2.3 SISTEM BUKTI

Di AI, kami tertarik untuk mengambil pengetahuan yang ada dan dari itu memperoleh pengetahuan baru atau menjawab pertanyaan. Dalam logika proposisional ini berarti menunjukkan bahwa basis pengetahuan KB—yaitu, rumus logika proposisional (mungkin ekstensif)—sebuah rumus Q[9] berikut. Jadi, pertama-tama kita definisikan istilah "entailmen".

Setiap rumus yang tidak valid memilih sehingga untuk berbicara bagian dari himpunan semua interpretasi sebagai modelnya. Tautologi seperti A ∨ ¬A, misalnya, tidak membatasi jumlah interpretasi yang memuaskan karena proposisinya kosong. Oleh karena itu, rumus kosong itu benar dalam semua interpretasi. Untuk setiap tautologi T maka $\emptyset \models T$. Secara intuitif ini berarti bahwa tautologi selalu benar, tanpa batasan interpretasi oleh formula. Untuk singkatnya, tulis $\models T$. Sekarang kami menunjukkan hubungan penting antara konsep semantik entailment dan implikasi sintaksis.

Definisi 2.6

Rumus KB memerlukan rumus *Q* (atau *Q* mengikuti dari *KB*) jika setiap model KB juga merupakan model *Q*. Kita tulis $KB \models Q$.

[9] Di sini Q adalah singkatan dari query.

Dengan kata lain, dalam setiap interpretasi di mana *KB* benar, *Q* juga benar. Lebih ringkasnya, setiap kali *KB* benar, *Q* juga benar. Karena, untuk konsep entailment, interpretasi variabel dibawa, kita berhadapan dengan konsep semantik.

Teorema 2.2 (Teorema Deduktif)

A |= B jika dan hanya jika |= A ⟹ B.

Bukti Perhatikan tabel kebenaran implikasinya:

A	*B*	*A ⟹ B*
t	*t*	*t*
t	*f*	*f*
f	*t*	*t*
f	*f*	*t*

Implikasi arbitrer A ⟹ B jelas selalu benar kecuali dengan interpretasi A ↦t, B ↦ f. Asumsikan bahwa A |= B berlaku. Ini berarti bahwa untuk setiap interpretasi yang membuat A benar, B juga benar. Baris kedua kritis dari tabel kebenaran bahkan tidak berlaku dalam kasus itu. Oleh karena itu A ⟹ B benar, yang berarti bahwa A ⟹ B adalah tautologi. Jadi satu arah pernyataan telah ditunjukkan. Sekarang asumsikan bahwa A ⟹ B berlaku. Dengan demikian baris kedua kritis dari tabel kebenaran juga terkunci. Setiap model A juga merupakan model B. Kemudian A |= B berlaku.

Jika kita ingin menunjukkan bahwa KB memerlukan Q, kita juga dapat menunjukkan melalui metode tabel kebenaran bahwa KB ⟹ Q adalah tautologi. Jadi kami memiliki sistem bukti pertama kami untuk logika proposisional, yang mudah diotomatisasi. Kerugian dari metode ini adalah waktu komputasi yang sangat lama dalam kasus terburuk. Khususnya, dalam kasus terburuk dengan n variabel proposisi, untuk semua 2^n interpretasi variabel, rumus KB ⟹ Q harus dievaluasi. Oleh karena itu, waktu komputasi tumbuh secara eksponensial dengan jumlah variabel. Oleh karena itu proses ini tidak dapat digunakan untuk jumlah variabel yang besar, setidaknya dalam kasus terburuk.

Jika rumus KB memerlukan rumus Q, maka dengan teorema pengurangan KB)Q adalah tautologi. Oleh karena itu, negasi ¬(KB ⟹ Q) tidak terpenuhi. Kita punya :

¬(KB ⟹ Q) ≡ ¬(¬KB ∨ Q) ≡ KB ∧ ¬Q.

Oleh karena itu, *KB ∧ ¬Q* juga tidak memuaskan. Kami merumuskan konsekuensi sederhana namun penting dari teorema deduksi sebagai teorema.

Teorema 2.3 *(Bukti dengan kontradiksi)*

KB |= Q jika dan hanya jika KB ∧ ¬Q tidak memuaskan.

Untuk menunjukkan bahwa kueri Q mengikuti dari basis pengetahuan KB, kita juga dapat menambahkan kueri yang dinegasikan ¬Q ke basis pengetahuan dan menurunkan kontradiksi. Karena ekivalensi A ∧ ¬A , f dari Teorema 2.1 kita tahu bahwa kontradiksi tidak terpenuhi. Oleh karena itu, Q telah terbukti. Prosedur ini, yang sering digunakan dalam matematika, juga digunakan dalam berbagai kalkulus pembuktian otomatis seperti kalkulus resolusi dan dalam pemrosesan program PROLOG.

Salah satu cara untuk menghindari keharusan menguji semua interpretasi dengan metode tabel kebenaran adalah manipulasi sintaksis dari rumus KB dan Q dengan penerapan aturan inferensi dengan tujuan menyederhanakannya, sehingga pada akhirnya kita dapat langsung melihat bahwa KB |= Q Kami menyebutnya derivasi proses sintaksis dan menulis KB Q.

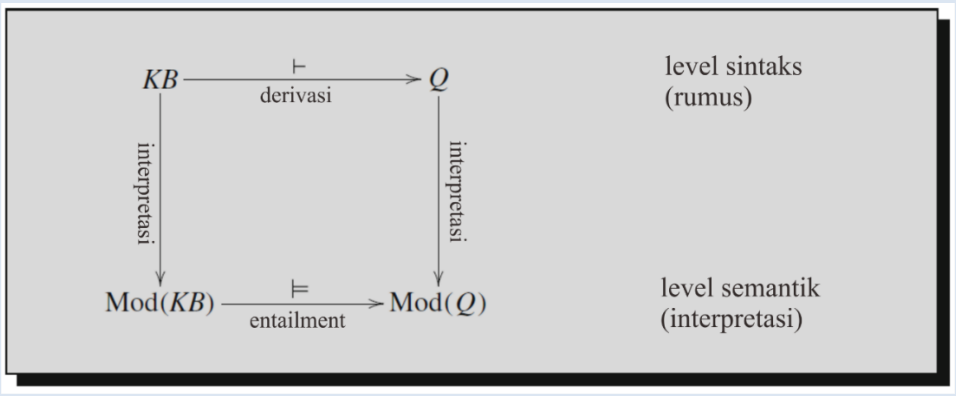
Sistem pembuktian sintaksis seperti itu disebut kalkuli. Untuk memastikan bahwa kalkulus tidak menghasilkan kesalahan, kami mendefinisikan dua sifat dasar kalkulus.

Definisi 2.7
Kalkulus disebut bunyi jika setiap proposisi turunan mengikuti secara semantik. Artinya, jika berlaku untuk rumus KB dan Q itu

jika KB |- Q maka KB |= Q.

Sebuah kalkulus disebut lengkap jika semua konsekuensi semantik dapat diturunkan. Artinya, untuk rumus KB dan Q berlaku:

jika KB |= Q maka KB |- Q.



Gambar 2.1 Derivasi sintaksis dan entailment semantik. Mod(X) mewakili himpunan model dari rumus X

Kesehatan kalkulus memastikan bahwa semua rumus turunan sebenarnya adalah konsekuensi semantik dari basis pengetahuan. Kalkulus tidak menghasilkan "konsekuensi palsu". Kelengkapan kalkulus, di sisi lain, memastikan bahwa kalkulus tidak mengabaikan apa pun. Kalkulus yang lengkap selalu menemukan bukti jika rumus yang akan dibuktikan mengikuti dari basis pengetahuan. Jika kalkulus sehat dan lengkap, maka derivasi sintaksis dan entailmen semantik adalah dua hubungan yang setara (lihat Gambar 2.1). Untuk menjaga sistem pembuktian otomatis sesederhana mungkin, ini biasanya dibuat untuk beroperasi pada rumus dalam bentuk normal konjungtif.

Definisi 2.8
Suatu rumus dikatakan dalam bentuk normal konjungtif (CNF) jika dan hanya jika terdiri dari konjungsi
dari klausa. Klausa K_i terdiri dari disjungsi
dari literal. Akhirnya, literal adalah variab$K_1 \wedge K_2 \wedge ... \wedge K_m$
$(L_{i1} \vee L_{i2} \vee ... \vee L_{ini})$
el (literal positif) atau variabel negatif (literal negatif).

Teorema 2.4
Setiap rumus logika proposisional dapat ditransformasikan ke dalam bentuk normal konjungtif ekuivalen.

Rumus (A ∨ B ∨ ¬C) ∧ (A ∨ B) ∧ (¬B ∨ C) dalam bentuk normal konjungsi. Bentuk normal konjungsi tidak membatasi himpunan rumus karena:

Contoh 2.1

Kita masukkan A ∨ B ⇒ C ∧ D ke dalam bentuk normal konjungtif dengan menggunakan ekuivalensi dari Teorema 2.1:

A ∨ B ⇒ C ∧ D
≡ ¬ (A ∨ B) ∨ (C ∧ D)
≡ (¬A ∨ B) ∨ (C D) *(implikasi)*
≡ (¬A ∧ ¬B) ∨ (C D) (*de Morgan)*
≡ (¬A ∧ (C ∧ D)) (¬B (C D)) *(hukum distribusi)*
≡ ((¬A ∨ C) ∧ (¬A ∨ D)) ∧ ((¬B ∨ C) ∧ (¬B ∨ D)) *(hukum distribusi)*
≡ ((¬A ∨C) ∧ (¬A ∨ D)) ∧ ((¬B ∨ C) ∧ (¬B ∨ D)) *(hukum aosiasi)*

Kita sekarang hanya kehilangan kalkulus untuk pembuktian sintaksis dari rumus logika proposisional. Kita mulai dengan modusponens, sederhana, rumus intuitif, yang, dari validitas A dan A ⇒ B, memungkinkan turunan dari B. Kami menulis ini secara formal sebagai :

$$\frac{A,\ \mathrm{A} \Rightarrow \mathrm{B}}{A}$$

Notasi ini berarti bahwa kita dapat menurunkan rumus di bawah garis dari rumus yang dipisahkan koma di atas garis. Modus ponens sebagai aturan dengan sendirinya, sementara suara, tidak lengkap. Jika kita menambahkan aturan tambahan, kita dapat membuat kalkulus lengkap, yang, bagaimanapun, tidak ingin kita pertimbangkan di sini. Sebagai gantinya kami akan menyelidiki aturan resolusi

$$\frac{A \vee \mathrm{B},\ \mathrm{B} \Rightarrow \mathrm{C}}{A \vee \mathrm{C}}$$

sebagai alternatif. Klausa turunan disebut resolvent. Melalui transformasi sederhana kita memperoleh bentuk yang setara

$$\frac{A \vee \mathrm{B},\ \mathrm{B} \Rightarrow \mathrm{C}}{A \vee \mathrm{C}}$$

Jika kita menetapkan A ke f, kita melihat bahwa aturan resolusi adalah generalisasi dari modus ponens. Aturan resolusi sama-sama dapat digunakan jika C tidak ada atau jika A dan C tidak ada. Dalam kasus terakhir, klausa kosong dapat diturunkan dari kontradiksi B ^ B (Latihan 2.7).

Resolusi

Kami sekarang menggeneralisasi aturan resolusi lagi dengan mengizinkan klausa dengan jumlah literal yang berubah-ubah. Dengan literal A_1, …, A_m, B, C_1, …, C_n aturan resolusi umum berbunyi

$$\frac{(A_1 \vee \ldots \vee A_m \vee \mathrm{B},\ (\neg \mathrm{B} \vee C_1 \vee \ldots \vee C_n)}{(A_1 \vee \ldots . \vee A_m \vee C_1 \vee \ldots \vee C_n)}$$

Kami menyebut literal B dan B saling melengkapi. Aturan resolusi menghapus sepasang literal pelengkap dari dua klausa dan menggabungkan sisa literal menjadi klausa baru.

Untuk membuktikan bahwa dari basis pengetahuan KB, kueri Q berikut, kami melakukan pembuktian dengan kontradiksi. Mengikuti Teorema 2.3 kita harus menunjukkan bahwa kontradiksi dapat diturunkan dari KB ∧¬Q. Pada rumus-rumus dalam bentuk normal konjungsi, muncul kontradiksi berupa dua klausa (A) dan (¬A), yang mengarah pada klausa kosong sebagai penyelesainya. Teorema berikut memastikan kita bahwa proses ini benar-benar bekerja seperti yang diinginkan.

Agar kalkulus menjadi lengkap, kita membutuhkan tambahan kecil, seperti yang ditunjukkan oleh contoh berikut. Biarkan rumus (A ∨ A) diberikan sebagai basis pengetahuan kita. Untuk menunjukkan dengan aturan resolusi bahwa dari sana kita dapat

menurunkan (A ∧ A), kita harus menunjukkan bahwa klausa kosong dapat diturunkan dari (A ∨ A) ∧ (¬A ∨ A). Dengan aturan resolusi saja, ini tidak mungkin. Dengan faktorisasi, yang memungkinkan penghapusan salinan literal dari klausa, masalah ini dihilangkan. Dalam contoh, penerapan faktorisasi ganda mengarah ke (A) ∧ (¬A), dan langkah resolusi ke klausa kosong.

Teorema 2.5

Kalkulus resolusi untuk pembuktian ketidakcukupan rumus dalam bentuk normal konjungtif adalah tepat dan lengkap.

Karena tugas kalkulus resolusi untuk menurunkan kontradiksi dari KB ∧¬Q, sangat penting bahwa basis pengetahuan KB konsisten:

Definisi 2.9

Suatu rumus *KB* disebut konsisten jika tidak mungkin diturunkan darinya suatu kontradiksi, yaitu suatu rumus berbentuk Ø ∧ ¬Ø.

Jika tidak, apa pun dapat diturunkan dari KB (lihat Latihan 2.8). Ini berlaku tidak hanya untuk resolusi, tetapi juga untuk banyak batu lainnya. Dari kalkuli untuk deduksi otomatis, resolusi memainkan peran yang luar biasa. Jadi kami ingin bekerja sedikit lebih dekat dengannya. Berbeda dengan kalkuli lain, resolusi hanya memiliki dua aturan inferensi, dan bekerja dengan rumus dalam bentuk normal konjungtif. Ini membuat implementasinya lebih sederhana. Keuntungan lebih lanjut dibandingkan dengan banyak batu terletak pada pengurangan jumlah kemungkinan penerapan aturan inferensi dalam setiap langkah pembuktian, dimana ruang pencarian berkurang dan waktu komputasi berkurang. Sebagai contoh, kita mulai dengan teka-teki logika sederhana yang memungkinkan langkah-langkah penting dari bukti resolusi ditampilkan.

Contoh 2.2

Teka-teki logika nomor 7 yang berjudul Keluarga Indonesia yang menawan, dari buku Jerman berbunyi (diterjemahkan Indonesia):

Meskipun belajar bahasa Inggris selama tujuh tahun, dan memiliki kesuksesan yang cemerlang, saya harus mengakui bahwa ketika saya mendengar orang Inggris berbicara bahasa Inggris, saya benar-benar bingung. Baru-baru ini, saya menjemput tiga pejalan kaki, seorang ayah, ibu, dan anak perempuan, mereka berbicara menggunakan bahasa Inggris. Saya bertanya kepada mereka dan mereka menjawab dengan kalimat yang hampir sama tapi terdengar memiliki maksud yang berbeda,saya menjadi ragu-ragu karena ada dua kemungkinan interpretasi. Mereka memberi tahu saya hal berikut (arti kedua yang mungkin ada dalam tanda kurung): Sang ayah: "Kami akan pergi ke Spanyol (kami dari Newcastle)." Sang ibu: "Kami tidak akan ke Spanyol dan dari Newcastle (kami berhenti di Paris dan tidak akan ke Spanyol)." Putri: “Kami bukan dari Newcastle (kami berhenti di Paris).” Bagaimana dengan keluarga Inggris yang menawan ini?

Untuk memecahkan masalah semacam ini, kami melanjutkan dalam tiga langkah: formalisasi, transformasi ke dalam bentuk normal, dan pembuktian. Dalam banyak kasus, formalisasi sejauh ini merupakan langkah yang paling sulit karena mudah membuat kesalahan atau melupakan detail-detail kecil. (Jadi latihan praktis sangat penting. Lihat Latihan 2.9–2.11.) Di sini kita menggunakan variabel S untuk “Kami akan ke Spanyol”, N untuk “Kami dari Newcastle”, dan P untuk “Kami berhenti di Paris ” dan diperoleh sebagai formalisasi dari tiga proposisi ayah, ibu, dan anak perempuan

$$(S \vee N) \wedge (\neg S \wedge N) \vee (P \wedge \neg S) \wedge (\neg N \vee P)$$

Memfaktorkan ¬S di sub-rumus tengah membawa rumus ke dalam CNF dalam satu langkah. Penomoran klausa dengan indeks subscript menghasilkan

$$KB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4$$

Sekarang kita mulai pembuktian resolusi, pertama masih tanpa pertanyaan Q. Ekspresi bentuk "Res(m, n): klausa$\rangle_k$" berarti bahwa klausa$\rangle$ diperoleh dengan resolusi klausa m dengan klausa n dan adalah bernomor k.

Res(1, 2) : $(n)_5$
Res(3,, 4): $(P)_6$
Res(1, 4): $(S \vee P)_7$

Kita dapat menurunkan klausa P juga dari Res(4, 5) atau Res(2, 7). Setiap langkah resolusi lebih lanjut akan mengarah pada derivasi klausa yang sudah tersedia. Karena tidak memungkinkan derivasi klausa kosong, oleh karena itu telah ditunjukkan bahwa basis pengetahuan tidak kontradiktif. Sejauh ini kami telah menurunkan N dan P. Untuk menunjukkan bahwa S berlaku, kami menambahkan klausa $(S)_8$ ke himpunan klausa sebagai kueri yang dinegasikan. Dengan langkah resolusi

Res(2, 8): $()_9$

dia buktinya lengkap. Jadi ¬S ∧ N ∧ P berlaku. "Keluarga Inggris yang menawan" ternyata berasal dari Newcastle, singgah di Paris, tetapi tidak akan ke Spanyol.

Contoh 2.3

Teka-teki logika nomor 28 dari [Ber89] berjudul *The High Jump,* berbunyi:

Tiga gadis berlatih lompat tinggi untuk ujian akhir pendidikan jasmani mereka. Bar diatur ke 1,20 meter. "Saya bertaruh", kata gadis pertama kepada yang kedua, "bahwa saya akan berhasil jika, dan hanya jika, Anda tidak melakukannya". Jika gadis kedua mengatakan hal yang sama kepada gadis ketiga, yang pada gilirannya mengatakan hal yang sama kepada yang pertama, apakah mungkin ketiganya memenangkan taruhan mereka?

Kami menunjukkan melalui bukti dengan resolusi bahwa tidak ketiganya dapat memenangkan taruhan mereka.

Formalisasi:

Lompatan gadis pertama berhasil: A, Gadis perama bertaruh: $(A \Leftrightarrow \neg B)$
Lompatan gadis kedua berhasil: B, Gadis kedua bertaruh : $(B \Leftrightarrow \neg C)$
Lompatan gadis ketiga berhasil: , Gadis ketiga bertaruh : $(C \Leftrightarrow \neg A)$

Klaim: ketiganya tidak dapat memenangkan semua taruhan mereka:

$$Q \equiv \neg((A \Leftrightarrow \neg B) \wedge (B \Leftrightarrow \neg C) \wedge (C \Leftrightarrow \neg A))$$

Sekarang harus ditunjukkan dengan resolusi bahwa Q tidak memuaskan. Transformasi menjadi CNF: Taruhan gadis pertama:

$$(A \Leftrightarrow \neg B) \equiv (A \Longrightarrow \neg B) \wedge (\neg B \Longrightarrow A) \equiv (\neg A \vee \neg B)) \wedge (A \vee B)$$

Taruhan dua gadis lainnya mengalami transformasi analog, dan kami mendapatkan klaim yang dinegasikan

$$\neg Q \equiv (\neg A \vee \neg B)_1 \wedge (A \vee B)_2 \wedge (\neg B \vee \neg C)_3 \equiv (B \vee C)_4 \wedge (\neg C \vee \neg B)_5 \wedge (C \vee A)_6$$

Dari sana kami menurunkan klausa kosong menggunakan resolusi:

Res(1, 6): $(C \vee \neg B)_7$
Res(4, 7): $(C)_8$
Res(2, 5): $(B \vee \neg C)_9$
Res*3, 9): $(\neg C)_{10}$
Res(8,1)): ()

Dengan demikian klaim telah terbukti.

2.4 KLAUSA TANDUK

Sebuah klausa dalam bentuk normal konjungsi mengandung literal positif dan negatif dan dapat direpresentasikan dalam bentuk

$$(\neg A \vee \ldots \vee \neg A_m \vee .. \vee B_n)$$

dengan variabel A_1, …, A_m dan B_1, …, B_n. Klausa ini dapat diubah dalam dua langkah sederhana menjadi bentuk yang setara

$$(A_1 \wedge \ldots \wedge A_m \Longrightarrow B_1 \vee B_n)$$

Implikasi ini mengandung premis, konjungsi variabel dan kesimpulan, disjungsi variabel. Misalnya, "Jika cuaca bagus, saya akan atau saya akan bekerja." adalah proposisi dari bentuk ini. Penerima pesan ini mengetahui dengan pasti bahwa pengirim tidak akan berenang. Pernyataan yang jauh lebih jelas adalah "Jika cuaca bagus, saya akan bermain bekerja.". Penerima sekarang memiliki informasi yang pasti. Jadi kami menyebut klausa dengan paling banyak satu klausa definit literal positif. Klausa-klausa ini memiliki keuntungan bahwa mereka hanya mengizinkan satu kesimpulan dan dengan demikian lebih mudah untuk ditafsirkan. Banyak relasi dapat dijelaskan dengan klausa jenis ini. Oleh karena itu kami mendefinisikan

Definisi 2.10

Klausa dengan paling banyak satu literal positif dari bentuk

$$(\neg A \vee \ldots \vee \neg A_m \vee B) \text{ atau } ((\neg A \vee \ldots \vee \neg A_m \text{ atau } B$$

atau (setara)

$$A_1 \wedge \ldots \wedge A_m \Longrightarrow B \quad \text{atau} \quad A_1 \wedge \ldots \wedge A_m \Longrightarrow f \text{ atau } B$$

diberi nama klausa Horn (setelah penemunya). Klausa dengan literal positif tunggal adalah fakta. Dalam klausa dengan literal negatif dan satu literal positif, literal positif disebut head. Untuk lebih memahami representasi klausa Horn, pembaca dapat menurunkannya dari definisi ekuivalensi yang kita gunakan saat ini (Latihan 2.12). Klausa tanduk lebih mudah ditangani tidak hanya dalam kehidupan sehari-hari, tetapi juga dalam penalaran formal, seperti yang dapat kita lihat pada contoh berikut. Biarkan basis pengetahuan terdiri dari klausa berikut ("∧" yang mengikat klausa ditinggalkan di sini dan dalam teks berikut):

$$(\text{cuaca\_bagus})_1$$
$$(\text{hujan turun})2$$
$$(\text{hujan turun} \Longrightarrow \text{hujan})3$$
$$(\text{cuaca\_bagus} \wedge \text{huajn turun} \Longrightarrow \text{ngopi})_4$$

Jika kita sekarang ingin tahu apakah ngopi berlaku, ini dapat dengan mudah diturunkan. Modus ponens yang sedikit digeneralisasikan cukup di sini sebagai aturan inferensi:

$$\frac{A_1 \wedge \ldots \wedge A_m \wedge \ldots \wedge A_m \Longrightarrow B}{B}$$

Bukti “ngopi” memiliki bentuk sebagai berikut (MP(i_1, …, i_k) merupakan penerapan modus ponens pada klausa i_1 sampai i_k:

$$MP(2, 3): \quad (\text{hujan})_5$$
$$MP(1, 5, 4): \quad (\text{ngopi})_6$$

Dengan modus ponens kita memperoleh kalkulus lengkap untuk rumus-rumus yang terdiri dari klausa-klausa Horn logika proposisional. Namun, dalam kasus basis pengetahuan yang besar, modus ponens dapat memperoleh banyak rumus yang tidak perlu jika dimulai dengan klausa yang salah. Oleh karena itu, dalam banyak kasus lebih baik menggunakan kalkulus yang dimulai dengan kueri dan bekerja mundur sampai fakta tercapai. Sistem seperti itu disebut dengan rantai mundur, berbeda dengan sistem rantai

maju, yang dimulai dengan fakta dan akhirnya menurunkan kueri, seperti dalam contoh di atas dengan modus ponens.

Untuk rantai mundur klausa Horn, resolusi SLD digunakan. SLD singkatan dari "Seleksi aturan didorong resolusi linier untuk klausa yang pasti". Dalam contoh di atas, ditambah dengan kueri yang dinegasikan (ngopi ⟹ f)

(cuaca_bagus)₁
(hujan turun)₂
(hujan turun ⟹ hujan)₃
(cuaca_bagus ∧ hujan ⟹ ngopi)₄
(ngopi ⟹ f)₅

kami melakukan resolusi SLD dimulai dengan langkah-langkah resolusi yang mengikuti dari klausa ini

Res(5, 4): (cuaca_bagus ∧ *hujan ⟹f)₆*
Res(6, 1) ; (hujan ⟹f)₇
Res(7, 3) : (hujan turun ⟹ f)₈
Res(8, 2): ()

dan mendapatkan kontradiksi dengan klausa kosong. Di sini kita dapat dengan mudah melihat "resolusi linier", yang berarti bahwa pemrosesan lebih lanjut selalu dilakukan pada klausa turunan saat ini. Hal ini menyebabkan pengurangan besar dari ruang pencarian. Selanjutnya, literal dari klausa saat ini selalu diproses dalam urutan yang tetap (misalnya, dari kanan ke kiri) ("Seleksi aturan didorong"). Literal dari klausa saat ini disebut subgoal. Literal dari kueri yang dinegasikan adalah tujuannya. Aturan inferensi untuk satu langkah berbunyi

$$\frac{A_1 \wedge \ldots \wedge A_m \Longrightarrow B_i,\ B_2\ \wedge \ldots \wedge B_n \Longrightarrow f}{A_1 \wedge \ldots \wedge A_m B_2\ \wedge \ldots \wedge B_n \Longrightarrow f}$$

Sebelum penerapan aturan inferensi, B_1, B_2, …, B_n—sub-tujuan saat ini—harus dibuktikan. Setelah aplikasi, B1 diganti dengan subgoal baru $A_1 \wedge ... \wedge A_m$. Untuk menunjukkan bahwa B_1 benar, sekarang kita harus menunjukkan bahwa $A_1 \wedge ... \wedge A_m$ benar. Proses ini berlanjut hingga daftar sub-tujuan dari klausa saat ini (yang disebut tumpukan tujuan) kosong. Dengan itu, kontradiksi telah ditemukan. Jika, untuk sub-tujuan $\neg B_i$, tidak ada klausa dengan literal pelengkap Bi sebagai kepala klausanya, buktinya berakhir dan tidak ada kontradiksi yang dapat ditemukan. Kueri dengan demikian tidak dapat dibuktikan.

Resolusi SLD memainkan peran penting dalam praktik karena program dalam bahasa pemrograman logika PROLOG terdiri dari klausa Horn logika predikat, dan pemrosesannya dicapai melalui resolusi SLD (lihat Latihan 2.13).

2.5 KOMPUTASI DAN KOMPLEKSITAS

Metode tabel kebenaran, sebagai sistem pembuktian semantik paling sederhana untuk logika proposisional, mewakili suatu algoritma yang dapat menentukan setiap model dari rumus apa pun dalam waktu yang berhingga. Dengan demikian, himpunan rumus yang tidak memuaskan, dapat memuaskan, dan valid dapat ditentukan. Waktu komputasi metode tabel kebenaran untuk kepuasan tumbuh dalam kasus terburuk secara eksponensial dengan jumlah n variabel karena tabel kebenaran memiliki 2n baris. Sebuah optimasi, metode pohon semantik, menghindari melihat variabel yang tidak muncul dalam klausa, dan dengan demikian menghemat waktu komputasi dalam banyak kasus, tetapi dalam kasus terburuk juga eksponensial.

Dalam resolusi, dalam kasus terburuk jumlah klausa turunan tumbuh secara eksponensial dengan jumlah klausa awal. Untuk memutuskan antara dua proses, oleh

karena itu kita dapat menggunakan aturan praktis bahwa dalam kasus banyak klausa dengan sedikit variabel, metode tabel kebenaran lebih disukai, dan dalam kasus beberapa klausa dengan banyak variabel, resolusi mungkin akan selesai lebih cepat.

Pertanyaannya tetap: dapatkah pembuktian dalam logika proposisional berjalan lebih cepat? Apakah ada algoritma yang lebih baik? Jawabannya: mungkin tidak. Bagaimanapun, S. Cook, pendiri teori kompleksitas, telah menunjukkan bahwa masalah 3-SAT adalah NP-complete. 3-SAT adalah himpunan semua rumus CNF yang klausanya memiliki tepat tiga literal. Jadi jelas bahwa mungkin (modulo masalah P/NP) tidak ada algoritma polinomial untuk 3-SAT, dan dengan demikian mungkin juga bukan yang umum. Namun, untuk klausa Horn, ada algoritme di mana waktu komputasi untuk menguji kepuasan tumbuh hanya secara linier seiring dengan meningkatnya jumlah literal dalam rumus.

2.6 APLIKASI DAN BATASAN

Pembukti teorema untuk logika proposisional adalah bagian dari perangkat sehari-hari pengembang dalam teknologi digital. Misalnya, verifikasi sirkuit digital dan pembuatan pola pengujian untuk pengujian mikroprosesor dalam fabrikasi adalah beberapa tugas ini. Sistem bukti khusus yang bekerja dengan diagram keputusan biner (BDD) juga digunakan sebagai struktur data untuk memproses rumus logika proposisional.

Dalam AI, logika proposisional digunakan dalam aplikasi sederhana. Misalnya, sistem pakar sederhana pasti dapat bekerja dengan logika proposisional. Namun, semua variabel harus diskrit, dengan hanya sedikit nilai, dan tidak boleh ada hubungan silang antar variabel. Koneksi logis yang kompleks dapat diekspresikan jauh lebih elegan menggunakan logika predikat. Logika probabilistik adalah kombinasi yang sangat menarik dan terkini dari logika proposisional dan komputasi probabilistik yang memungkinkan pemodelan pengetahuan yang tidak pasti. Ini ditangani secara menyeluruh di Bab. 7.

2.7 LATIHAN

Latihan 2.1

Berikan bentuk tata bahasa Backus–Naur untuk sintaks logika proposisional.

Latihan 2.2

Tunjukkan bahwa rumus berikut adalah tautologi:

a. $\neg(A \wedge B), \Leftrightarrow \neg A \vee \neg B$
b. $A \Longrightarrow B, \neg B \Longrightarrow \neg A$
c. $((A \Longrightarrow B) \wedge (B \Longrightarrow A)) \Leftrightarrow (A \Leftrightarrow B)$
d. $(A \vee B) \wedge (\neg B \vee C) \Longrightarrow (A \vee C)$

Latihan 2.3

Ubahlah rumus berikut ke dalam bentuk normal konjungtif:

a. $A \Leftrightarrow B$
b. $(A \wedge B) \Leftrightarrow A \vee B$
c. $A \wedge (A \Longrightarrow B) \Longrightarrow B$

Latihan 2.4

Periksa pernyataan berikut untuk kepuasan atau validitas.

a. (main_lottre ∧ enam_menang) ⟹menang
b. (main_lotre ∧ enam_menang ∧ (enam_menang)) ⟹menang
c. ¬(¬gas_dalam_tangki ∧ (gas_dalam_tangki ∨ ¬mobil_nyala ⟹ ¬mobil_nyala

Latihan 2.5

Dengan menggunakan bahasa pemrograman pilihan Anda, programlah pembuktian teorema untuk logika proposisional menggunakan metode tabel kebenaran untuk rumus-rumus dalam bentuk normal konjungtif. Untuk menghindari pemeriksaan sintaks yang mahal pada rumus, Anda dapat merepresentasikan klausa sebagai daftar atau kumpulan literal, dan rumus sebagai daftar atau kumpulan klausa. Program harus menunjukkan apakah rumus tersebut tidak memuaskan, memuaskan, atau benar, dan menampilkan sejumlah interpretasi dan model yang berbeda.

Latihan 2.6

(a) Tunjukkan bahwa modus ponens adalah aturan inferensi yang valid dengan menunjukkan bahwa $A \wedge (A \Longrightarrow B) \models B$

(b) Tunjukkan bahwa aturan resolusi (2.1) adalah aturan inferensi yang valid.

Latihan 2.7

Tunjukkan dengan penerapan aturan resolusi bahwa, dalam bentuk normal konjungsi, klausa kosong ekuivalen dengan pernyataan salah.

Latihan 2.8

Tunjukkan bahwa, dengan resolusi, seseorang dapat "mendapatkan" klausa arbitrer apa pun dari basis pengetahuan yang mengandung kontradiksi.

Latihan 2.9

Bentuklah fungsi logika berikut dengan operator logika dan tunjukkan bahwa rumus Anda valid. Sajikan hasilnya dalam CNF.

(a) Operasi XOR (eksklusif atau) antara dua variabel.

(b) Pernyataan paling sedikit dua dari tiga variabel A, B, C benar.

Latihan 2.10

Selesaikan kasus berikut dengan bantuan bukti resolusi: “Jika penjahat memiliki kaki tangan, maka dia datang dengan mobil. Penjahat tidak memiliki kaki tangan dan tidak memiliki kunci, atau dia memiliki kunci dan kaki tangan. Penjahat itu memiliki kuncinya. Apakah penjahat itu datang dengan mobil atau tidak?”

Latihan 2.11

Tunjukkan dengan resolusi bahwa rumus dari

(a) Latihan 2.2 (d) adalah tautologi.

(b) Latihan 2.4 (c) tidak memuaskan.

Latihan 2.12

Buktikan kesetaraan berikut, yang penting untuk bekerja dengan klausa Horn:

a. $(\neg A_1 \vee ... \vee \neg A_m \vee B) \equiv A_1 \wedge ... \wedge A_m \Longrightarrow B$

b. $(\neg A_1 \vee ... \vee \neg A_m) \equiv A_1 \wedge ... \wedge A_m \Longrightarrow f$

c. $A \equiv w \Longrightarrow B$

Latihan 2.13

Tunjukkan dengan resolusi SLD bahwa rangkaian klausa Horn berikut tidak memuaskan.

$(A)_1$

$(B)_2$

$(C)_3$

$(D)_4$

$(E)_5$

$(A \wedge B \wedge C \Longrightarrow F)_6$

$(A \wedge D \Longrightarrow G)_7$

$(C \wedge F \wedge E \Longrightarrow H)_8$

$(H \Longrightarrow f)_9$

Latihan 2.14

Dikatakan: “Jadi jelas bahwa mungkin (modulo masalah P/NP) tidak ada algoritma polinomial untuk 3-SAT, dan dengan demikian mungkin juga bukan yang umum.” Benarkan "mungkin" dalam kalimat ini.

BAB 3
LOGIKA PREDIKAT ORDE PERTAMA

Banyak masalah praktis dan relevan yang tidak dapat atau hanya dapat dengan sangat tidak nyaman dirumuskan dalam bahasa logika proposisional, seperti yang dapat kita kenali dengan mudah dalam contoh berikut. Pernyataan

"Robot 7 terletak pada posisi xy (35, 79)"

sebenarnya dapat langsung digunakan sebagai variabel logika proposisional

"Robot_7_is_terletak_at_xy_position_(35, 79)"

untuk penalaran dengan logika proposisional, tetapi penalaran dengan proposisi semacam ini sangat merepotkan. Asumsikan 100 robot ini dapat berhenti di mana saja pada grid 100 x 100 poin. Untuk menggambarkan setiap posisi dari setiap robot, kita membutuhkan 100 x 100 x 100 = 1.000.000 = 106 variabel yang berbeda. Definisi hubungan antar objek (di sini robot) menjadi sangat sulit. hubungan

"Robot A ada di sebelah kanan robot B."

secara semantik tidak lebih dari satu set pasangan. Dari 10.000 pasangan koordinat x yang mungkin ada (99 x 98)/2 = 4851 pasangan terurut. Bersama dengan semua 10.000 kombinasi nilai-y yang mungkin untuk kedua robot, ada (100 x 99) = 9900 rumus dari jenis

Robot _7_berada_di_sebelah_kanan_robot_12
Robot_7 terletak_pada_posisi_xy_(35; 79)
Robot_12_terletak_di_posisi_ xy_(10; 93) ∨.....

mendefinisikan hubungan ini, masing-masing dengan $(104)^2$ x 0,485 = 0,485 x 10^8 alternatif di sisi kanan. Dalam logika predikat orde pertama, kita dapat mendefinisikan posisi predikat (angka, xPosition, yPosition). Relasi di atas tidak boleh lagi disebutkan sebagai sejumlah besar pasangan, melainkan digambarkan secara abstrak dengan aturan bentuk

$$\forall_u \forall_v \text{ adalah\_ lebih\_kanan } (u, v) \Leftrightarrow$$
$$\exists x_u \exists y_u \exists y_v \text{ posisi } (u, x_u, y_u) \wedge \text{posisi } (v, x_v, y_v) \wedge x_u > x_v,$$

Dimana $\forall_u$ dibaca sebagai *"untuk setiap u"* dan $\exists v$ sebagai "ada *v*". Dalam bab ini kita akan mendefinisikan sintaks dan semantik logika predikat orde pertama (PL1), menunjukkan bahwa banyak aplikasi dapat dimodelkan dengan bahasa ini dan bahwa ada kalkulus lengkap dan suara untuk bahasa ini.

3.1 SINTAKS

Pertama kita memperkuat struktur sintaksis istilah.

Definisi 3.1
Misalkan *V* himpunan variabel, *K* himpunan konstanta, dan *F* himpunan simbol fungsi. Himpunan *V*, *K* dan *F* adalah himpunan lepas berpasangan. Kami mendefinisikan himpunan suku secara rekursif:

- Semua variabel dan konstanta adalah suku (atom).
- Jika $t_1, \ldots, tn$ adalah suku dan f merupakan simbol fungsi tempat ke-n, maka f ($t_1, \ldots, t_n$) juga merupakan suku.

Beberapa contoh suku adalah f (sin(ln(3)), exp(x)) dan g(g(g(x))). Untuk dapat membangun hubungan logis antar istilah, kita membangun rumus dari istilah.

Definisi 3.2
Misalkan P adalah himpunan simbol predikat. Rumus logika predikat dibangun sebagai berikut:

- Jika t_1, … , t_n adalah suku dan p adalah simbol predikat tempat-n, maka $p(t_1, …, t_n)$ adalah rumus (atom).
- Jika A dan B adalah rumus, maka ¬A, (A), A ∧ B, A ∨ B, A ⟹ B, A ⟺ B juga rumus.
- Jika x variabel dan A rumus, maka ∀*x* A dan ∃*x* A juga rumus. ∀ adalah quantifier universal dan ∃ quantifier eksistensial.
- $p(t_1,…, t_n)$ dan $p(t_1,…, t_n)$ disebut literal.
- Rumus di mana setiap variabel berada dalam ruang lingkup quantifier disebut kalimat orde pertama atau rumus tertutup. Variabel yang tidak termasuk dalam ruang lingkup suatu quantifier disebut variabel bebas.
- Definisi 2.8 (CNF) dan 2.10 (klausa Horn) berlaku untuk rumus literal logika predikat secara analog.

Tabel 3.1 Contoh rumus dalam logika predikat orde pertama. Harap dicatat bahwa ibu di sini adalah simbol fungsi

Rumus	***Deskripsi***
∀x katak(x) ⇒ hijaux)	Semua katak adalah hijau
∀x katak(x) ∧ coklat(x) ⇒ besar	Semua katak coklat adalah besar
∀x suka(x, kue)	Semua orang suka kue
¬ ∀ suka(x, kue)	Tidak semua orang suka kue
¬ ∃x suka(x, kue)	Ada sesuatu yang disukai semua orang
∃y ∀x suka(x, y)	Ada seseorang yang menykai semuanya
∃y ∀x suka(y, x)	Semuanya dicintai oleh seseorang
∀x ∃y suka(x, y)	Semua orang menyukai sesuatu
∀x ∃y suka(y, x)	Ada pelanggan yang disukai boby
∃x tukang roti(x) ∧ ∀y pelanggan(y) ⇒ suka(x, y)	Ada seorang tukang roti yang menyukai semua pelanggannya
∀x lebih tua dari ibunya(x), x)	Setiap ibu lebih tua dari anaknya
∀x lebih tua dari (ibunya(ibunya(x)), x)	Setiap nenek lebih tua dari anak putrinya
∀x ∀y ∀z rel(x, y) ∧rel(y, z) ⇒rel(x, z)	rel adalah relasi transitif

Pada Tabel 3.1 diberikan beberapa contoh rumus PL1 beserta interpretasi intuitifnya.

Definisi 3.3
Interpretasi I didefinisikan sebagai

- Pemetaan dari himpunan konstanta dan variabel K[V ke himpunan W nama objek di dunia.
- Pemetaan dari himpunan simbol fungsi ke himpunan fungsi di dunia. Setiap simbol fungsi n-tempat diberikan fungsi n-tempat.
- Pemetaan dari himpunan simbol predikat ke himpunan relasi di dunia. Setiap simbol predikat n-tempat diberikan relasi n-tempat.

3.2 SEMANTIK

Dalam logika proposisional, setiap variabel secara langsung diberi nilai kebenaran dengan interpretasi. Dalam logika predikat, makna rumus didefinisikan secara rekursif selama konstruksi rumus, di mana pertama-tama kita menetapkan konstanta, variabel, dan simbol fungsi ke objek di dunia nyata.

Contoh 3.1

Misalkan c_1, c_2, c_3 adalah konstanta, "plus" simbol fungsi dua tempat, dan *"gr"* simbol predikat dua tempat. Kebenaran rumus

$$F \equiv gr(plus(c_1, c_3(, c_2)$$

tergantung pada interpretasi ‖. Pertama-tama pilih interpretasi yang jelas berikut dari konstanta, fungsi, dan predikat dalam bilangan asli:

$$\|_1: c_1 \mapsto 1,\ c_2 \mapsto 2, c_3 \mapsto 3, \text{plus} \mapsto +, \text{gr} \mapsto >.$$

Dengan demikian rumus dipetakan ke

$$1 + 3 > 2, \text{ atau setelah evaluasi } 4 > 2$$

Relasi lebih besar dari pada himpunan {1, 2, 3, 4} adalah himpunan semua pasangan (x, y) bilangan dengan x > y, artinya himpunan G = {(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)}. Karena (4, 2) ∈ G, rumus F benar menurut interpretasi I1. Namun, jika kita memilih interpretasi

$$\|_2: c_1 \mapsto 1,\ c_2 \mapsto 2, c_3 \mapsto 3, \text{plus} \mapsto +, \text{gr} \mapsto >.$$

kita peroleh

$$2 - 1 > 3, \text{ atau } 1 > 3$$

Pasangan (1, 3) bukan anggota G. Rumus F salah menurut interpretasi $\|_1 2$. Jelas, kebenaran rumus di PL1 tergantung pada interpretasi. Sekarang, setelah pratinjau ini, kita mendefinisikan kebenaran.

Definisi 3.4

Rumus atom $p(t_1,\ldots, t_n)$ benar (atau valid) menurut interpretasi ‖ jika, setelah interpretasi dan evaluasi semua suku $t_1,\ldots, t_n$ dan interpretasi predikat p melalui relasi n-tempat r, itu menyatakan bahwa

$$(\|(t_1), \ldots, \|(t_n)) \in r$$

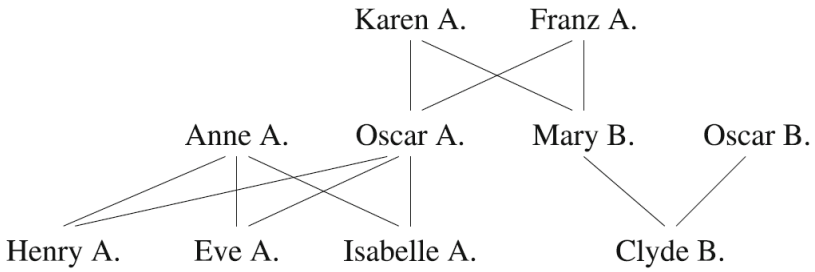
- Kebenaran rumus tanpa bilangan mengikuti dari kebenaran rumus atom—seperti dalam kalkulus proposisional—melalui semantik operator logika yang didefinisikan pada Tabel 2.1.
- Rumus $\forall x$ F benar di bawah interpretasi ‖ tepat ketika itu benar diberikan sewenang-wenang perubahan interpretasi untuk variabel x (dan hanya untuk x)
- Sebuah rumus $\exists_x$ F benar di bawah interpretasi ‖ persis ketika ada interpretasi untuk x yang membuat rumus benar. Definisi ekuivalensi semantik dari formula, untuk konsep satisfiable, true, unsatisfiable, dan model, bersama dengan semantic entailment (Definisi 2.4, 2.5, 2.6) tidak berubah dari kalkulus proposisi ke logika predikat.

Teorema 3.1

Teorema 2.2 (teorema deduksi) dan 2.3 (pembuktian dengan kontradiksi) berlaku analog untuk PL1

Contoh 3.2

Pohon keluarga yang diberikan pada Gambar 3.1 secara grafis mewakili (dalam tingkat semantik) hubungan tersebut



Gambar 3.1 Pohon keluarga. Tepi dari Clyde B. ke atas ke Mary B. dan Oscar B. mewakili elemen (Clyde B., Mary B., Oscar B.) sebagai hubungan anak

Anak = {(Oscar A., Karen A., Frank A.), (Mary B., Karen A., Frank A.),
(Henry A., Anne A., Oscar A.), (Eve A., Anne A., Oscar A.),
(Isabelle A., Anne A., Oscar A.), (Clyde B., Mary B., Oscar B.)}

Misalnya, rangkap tiga (Oscar A., Karen A., Frank A.) mewakili proposisi "Oscar A. adalah anak dari Karen A. dan Frank A.". Dari nama yang kita baca dari hubungan satu tempat

Wanita = {Karen A., Anne A., Mary B., Isabelle A.}

dari para wanita. Kami sekarang ingin menetapkan rumus untuk hubungan keluarga. Pertama kita mendefinisikan anak predikat tiga tempat (x, y, z) dengan semantik

‖ (anak (x, y, z)) = w ≡ (‖(x), ‖(y), ‖(z)) ∈ jenis

Di bawah interpretasi ‖ (oscar) = Oscar A., ‖(eve) = Eve A., ‖(anne) = Anne A., juga benar bahwa anak (eve, anne, oscar). Agar anak (eve, oscar, anne) benar, kami membutuhkan, dengan

$\forall x \forall y \forall z$ anak (x, y, z) ⇔ anak (x, y, z)

simetri anak predikat dalam dua argumen terakhir. Untuk definisi lebih lanjut, kami merujuk ke Latihan 3.1 an mendefinisikan turunan predikat secara rekursif sebagai

$\forall x \forall y$ keturunan (x, y) ⇔ ∃z anak (x, y, z) ∨ (∃u ∃v anak (x, u, v) ∧ keturunan (u, y))

Sekarang kita membangun basis pengetahuan kecil dengan aturan dan fakta. Membiarkan

KB = wanita (karen) ∧ wanita (anne) ∧ (wanita (mary)
∧ wanita (eve) ∧ wanita (isabelle)
∧ anak (eve, anne, oscar) ∧ anak (mary, karen, franz)
∧ anak (isabelle, anne, oscar) ∧ anak (henry, anne, oscar
∧ anak (isabelle, anne, oscar) ∧ anak (clyde, mary, oscar)
∧ (∀x ∀y ∀z anak (x, y, z) ⟹ (anak (x, y, z))
∧ (∀x ∀y keturunan (x, y) ⇔ ∃z anak (x, y, z)
∨ (∃u ∃v anak (x, u, v) ∧ keturunan (u, y))

Sekarang kita dapat menanyakan, misalnya, apakah proposisi child(eve, oscar, anne) atau descendant(eve, franz) dapat diturunkan. Untuk itu kita membutuhkan sebuah kalkulus.

3.3 EKUITAS

Untuk dapat membandingkan istilah, kesetaraan adalah hubungan yang sangat penting dalam logika predikat. Persamaan istilah dalam matematika merupakan hubungan ekivalensi, artinya bersifat refleksif, simetris dan transitif. Jika kita ingin menggunakan persamaan dalam rumus, kita harus memasukkan ketiga atribut ini sebagai aksioma dalam basis pengetahuan kita, atau kita harus mengintegrasikan persamaan ke dalam kalkulus. Kami mengambil cara mudah dan mendefinisikan predikat "=" yang menyimpang dari Definisi 3.2, ditulis menggunakan notasi infiks seperti biasa dalam matematika. (Persamaan x = y tentu saja dapat juga ditulis dalam bentuk eq(x, y).) Dengan demikian, aksioma persamaan memiliki bentuk:

Persamaan 3.1

$\forall x$	$x = x$	*(reflektivitas)*
$\forall x \forall y$	$x = y \Longrightarrow y = x$	*(simetris)*
$\forall x \forall y \forall z$	$x = y \wedge y = z \Longrightarrow x = z$	*(transitivitas)*

Untuk menjamin keunikan fungsi, kami juga membutuhkan

Persamaan 3.2

$$\forall x \forall y \; x = y \Longrightarrow f(x) = f(y) \qquad \text{(aksioma subtitusi)}$$

untuk setiap simbol fungsi. Secara analog kita membutuhkan untuk semua simbol predikat

Persamaan 3.3

$$\forall x \forall y \; x = y \Longrightarrow p(x) \Leftrightarrow p(y) \qquad \text{(aksioma subtitusi)}$$

Kami merumuskan hubungan matematika lainnya, seperti hubungan "<", dengan cara yang sama (Latihan 3.4). Seringkali variabel harus diganti dengan istilah. Untuk melakukan ini dengan benar dan menggambarkannya secara sederhana, kami memberikan definisi berikut.

Definisi 3.5

Kami menulis u[x/t] untuk rumus yang dihasilkan ketika kita mengganti setiap kemunculan bebas dari variabel x di u dengan istilah t. Dengan demikian kami tidak mengizinkan variabel apa pun dalam istilah t yang dikuantifikasi dalam u. Dalam kasus tersebut, variabel harus diganti namanya untuk memastikan hal ini.

Contoh 3.3

Jika, dalam rumus $\forall x \; x = y$, variabel bebas y diganti dengan suku x + 1, hasilnya adalah $\forall x \; x = y + 1$. Dengan substitusi yang benar kita memperoleh rumus $\forall x \; x = y + 1$, yang memiliki semantik yang sangat berbeda.

3.4 KUANTIFIER DAN BENTUK NORMAL

Berdasarkan Definisi 3.4 , rumus $\forall x \; p(x)$ benar jika dan hanya jika benar untuk semua interpretasi variabel x. Alih-alih quantifier, kita bisa menulis $p(a_1) \wedge ... \wedge p(a_n)$ untuk semua konstanta $a_1 \cdots a_n$ di K. Untuk $\exists x \; p(x)$ kita bisa menulis $p(a_1) \vee ... \vee p(a_n)$. Dari sini mengikuti hukum de Morgan bahwa

$$\forall x \; \varphi \equiv \neg \exists x \neg \varphi$$

Melalui ekuivalensi ini, kuantor universal dan eksistensial dapat saling menggantikan.

Contoh 3.4

Proposisi "Semua orang ingin dicintai" sama dengan proposisi "Tidak ada yang tidak ingin dicintai".

Kuantifikasi adalah komponen penting dari kekuatan ekspresif logika predikat. Namun, mereka mengganggu untuk inferensi otomatis di AI karena mereka membuat struktur rumus lebih kompleks dan meningkatkan jumlah aturan inferensi yang berlaku di setiap langkah pembuktian. Oleh karena itu, tujuan kita selanjutnya adalah menemukan, untuk setiap rumus logika predikat, sebuah rumus ekuivalen dalam bentuk normal terstandarisasi dengan sesedikit mungkin quantifier. Sebagai langkah pertama, kami membawa kuantor universal ke awal rumus dan dengan demikian mendefinisikan

Definisi 3.6

Sebuah rumus logika predikat u berada dalam bentuk normal prenex jika menyatakan bahwa

- $\Phi = Q_1 x_1 ... \; Q_n x_n \; \Psi$

- Ψ adalah rumus tanpa bilangan
- Qi ∈ {∀, ∃} untuk i = 1, ..., n.

Perhatian disarankan jika variabel terkuantifikasi muncul di luar lingkup quantifiernya, seperti misalnya x dalam

$$\forall x\ p(x) \Longrightarrow \exists x\ q(x)$$

Di sini salah satu dari dua variabel harus diganti namanya, dan di

$$\forall x\ p(x) \Longrightarrow \exists y\ q(y)$$

pengukur dapat dengan mudah dibawa ke depan, dan kami memperoleh sebagai output rumus yang setara

$$\forall x\ \exists y\ p(x) \Longrightarrow q(y)$$

Namun, jika kita ingin membawa quantifier dengan benar ke depan

$$(\forall x\ p(x) \Longrightarrow \exists y\ q(y)$$

pertama kita tulis rumusnya dalam bentuk yang setara

$$\neg(\forall x\ p(x)) \vee \exists y\ (q(y)$$

Kuantifier universal pertama sekarang berubah menjadi

$$\exists x\ \neg p(x)) \vee \exists y\ q(y)$$

dan sekarang kedua quantifier akhirnya dapat ditarik ke depan ke

$$\exists x\ \exists y\ \neg p(x) \vee q(y)$$

yang setara dengan

$$\exists x\ \exists y p(x) \Longrightarrow q(y)$$

Kemudian kita melihat bahwa di (3.4) halaman 46 kita tidak bisa begitu saja menarik kedua quantifier ke depan. Sebaliknya, pertama-tama kita harus menghilangkan implikasinya sehingga tidak ada negasi pada quantifiers. Ini menyatakan secara umum bahwa kita hanya dapat menarik quantifier jika negasi hanya ada secara langsung pada sub-rumus atom.

Contoh 3.5

Seperti diketahui dalam analisis, kekonvergenan deret (an)n2ℕ hingga batas a didefinisikan oleh

$$\forall \varepsilon > 0\ \exists n_0 \in \mathbb{N}\ \forall n > n_0 |a_n - a| < \varepsilon$$

Dengan fungsi abs(x) untuk |x|, a(n) untuk an, minus(x, y) untuk x–y dan predikat el(x, y) untuk x 2 y, gr(x, y) untuk x > y, rumusnya berbunyi

$$\forall \varepsilon(gr(\varepsilon, 0) \Longrightarrow \exists n_0(el(n_0, \mathbb{N}) \Longrightarrow \forall n(gr(n, n_0) \Longrightarrow gr(\varepsilon, abs(minus(a(n), a)))))$$

Ini jelas bukan dalam bentuk normal prenex. Karena variabel-variabel dari quantifier dalam $\exists n_0$ dan $\forall_n$ tidak muncul di sebelah kiri quantifiers masing-masing, maka tidak ada variabel yang harus diganti namanya. Selanjutnya kita menghilangkan implikasi dan memperoleh

$$\forall \varepsilon(\neg gr(\varepsilon, 0) \vee \exists n_0(\neg el(n_0, \mathbb{N}) \vee \forall n(\neg gr(n, n_0) \vee gr(\varepsilon, abs(minus(a(n), a)))))$$

Karena setiap negasi berada di depan rumus atom, kita majukan bilangannya, hilangkan tanda kurung yang berlebihan, dan dengan

$$\forall \varepsilon \vee \exists n_0 \forall n\ (\neg el(\varepsilon, 0) \vee (\neg el(n_0, \mathbb{N}) \vee \forall n(\neg gr(n, n_0) \vee gr(\varepsilon, abs(minus(a(n), a))))) (\neg gr(\varepsilon, 0)$$

itu menjadi klausa terkuantifikasi dalam bentuk normal penghubung.

Rumus yang diubah setara dengan rumus output. Fakta bahwa transformasi ini selalu mungkin dijamin oleh

Teorema 3.2

Setiap rumus logika predikat dapat ditransformasikan ke dalam persamaan setara dalam bentuk normal prenex.

Selain itu, kita dapat menghilangkan semua kuantor eksistensial. Namun, rumus yang dihasilkan dari apa yang disebut Skolemisasi tidak lagi setara dengan rumus output.

Kepuasannya, bagaimanapun, tetap tidak berubah. Dalam banyak kasus, terutama ketika seseorang ingin menunjukkan ketidakpuasaan KB ∧ ¬Q, ini sudah cukup. Rumus berikut dalam bentuk normal prenex sekarang akan diskolemisasi:

$$\forall x_1 \forall x_2 \exists y_1 \forall x_3 \exists y_2 \, p(f(x_1), x_2, y_1) \vee q(y_1, x_3, y_2)$$

Karena variabel y1 tampaknya bergantung pada x_1 dan x_2, setiap kemunculan y_1 digantikan oleh fungsi Skolem $g(x_1, x_2)$. Penting bahwa g adalah simbol fungsi baru yang belum muncul dalam rumus. Kami memperoleh

$$\forall x_1 \forall x_2 \forall x_3 \exists y_2 \, p(f(x_1), g(x_1, x_2) \vee q(g(x_1, x_2), x_3, y_2)$$

dan ganti y2 secara analog dengan $h(x_1, x_2, x_3)$, yang mengarah ke

$$\forall x_1 \forall x_2 \forall x_3 \, p(f(x_1), x_2, g(x_1, x_2) \vee q(g(x_1, x_2), x_3, h(x_1, x_2, x_3)).$$

Karena sekarang semua variabel terkuantifikasi secara universal, maka quantifier universal dapat diabaikan, menghasilkan

$$p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, h(x_1, x_2, x_3)).$$

Sekarang kita dapat menghilangkan kuantor eksistensial (dan dengan demikian juga kuantifikasi universal) di (3.5) dengan memperkenalkan fungsi Skolem $n_0(e)$. Prenex skolemized dan bentuk normal penghubung dari (3.5) berbunyi

$$\neg gr(\varepsilon, 0) \vee \neg el(n_0, (\varepsilon), \mathbb{N}) \vee \neg gr(n, n_0) \vee gr(\varepsilon, abs(minus(a(n), a)))).$$

Dengan menjatuhkan variabel n_0, fungsi Skolem dapat menerima nama n_0.

Saat mengkolemisasi rumus dalam bentuk normal prenex, semua kuantor eksistensial dihilangkan dari luar ke dalam, di mana rumus berbentuk $\forall x_1 \ldots \forall x_n \exists y \, \phi$ diganti dengan $\forall x_1 \ldots \forall x_n \, \phi[y/f(x_1,\ldots, x_n)]$, di mana f mungkin tidak muncul di ϕ. Jika suatu kuantor eksistensial berada jauh di luar, seperti pada ∃yp (y), maka y harus diganti dengan suatu konstanta (yaitu, dengan simbol fungsi tempat-nol).

TRANSFORMASI NORMALFORM(Rumus):

1. Transformasi ke bentuk normal prenex:

   Transformasi ke bentuk normal konjungtif (Teorema 2.1):

   Eliminasi ekivalensi.

   Eliminasi implikasi.

   Aplikasi berulang dari hukum de Morgan dan hukum distributif.

   Mengganti nama variabel jika perlu. Memfaktorkan kuantor universal.

2. Skolemisasi:

   Penggantian variabel yang dikuantifikasi secara eksistensial dengan fungsi Skolem baru.

   Penghapusan quantifier universal yang dihasilkan.

Gambar 3.2 Transformasi rumus logika predikat menjadi bentuk normal.

Prosedur untuk mengubah rumus dalam bentuk normal konjungtif diringkas dalam pseudocode yang ditunjukkan pada Gambar 3.2. Skolemization memiliki runtime polinomial dalam jumlah literal. Saat mengubah ke bentuk normal, jumlah literal dalam bentuk normal dapat bertambah secara eksponensial, yang dapat menyebabkan waktu komputasi eksponensial dan penggunaan memori eksponensial. Alasan untuk ini adalah penerapan berulang dari hukum distributif. Masalah sebenarnya, yang dihasilkan dari sejumlah besar klausa, adalah ledakan kombinatorial dari ruang pencarian untuk bukti

resolusi berikutnya. Namun, ada algoritma transformasi yang dioptimalkan yang hanya memunculkan banyak literal secara polinomial.

3.5 KALKULUS BUKTI

Untuk penalaran dalam logika predikat, telah dikembangkan berbagai kalkulus penalaran alami seperti kalkulus Gentzen atau kalkulus sekuen. Seperti namanya, batu ini dimaksudkan untuk diterapkan oleh manusia, karena aturan inferensi kurang lebih intuitif dan batu bekerja pada rumus PL1 yang berubah-ubah. Pada bagian berikutnya kita terutama akan berkonsentrasi pada kalkulus resolusi, yang dalam praktiknya merupakan kalkulus yang paling efisien dan dapat diotomatisasi untuk rumus dalam bentuk normal konjungtif. Di sini, dengan menggunakan Contoh 3.2 kami akan memberikan bukti "alami" yang sangat kecil. Kami menggunakan aturan inferensi

$$\frac{A, \mathrm{A} \implies \mathrm{B}}{B}\ (modus\ ponen, MP) \quad dan \quad \frac{\forall x\ \mathrm{A}}{A[x/t]}\ \ e\text{liminasi}\ \forall, \forall E$$

Modus ponens sudah familiar dari logika proposisional. Ketika menghilangkan kuantor universal, kita harus ingat bahwa variabel terkuantifikasi x harus diganti dengan suku dasar t, yang berarti suku yang tidak mengandung variabel. Bukti anak (eve, oscar, anne) dari basis pengetahuan yang dikurangi secara tepat disajikan pada Tabel 3.2.

Tabel 3.2 Pembuktian sederhana dengan modus ponens dan eliminasi kuantitatif

WB:	1	*child(eve, anne, oscar)*
WB:	2	$\forall x\ \forall y\ \forall z$ *child*$(x, y, z) \Rightarrow$ *child*(x, z, y)
$\forall E(2)$: *x/eve, y/anne, z/oscar*	3	*child(eve, anne, oscar)* $\Rightarrow$ *child(eve, oscar, anne)*
MP(1, 3)	4	*child(eve, oscar, anne)*

Dua rumus basis pengetahuan tereduksi tercantum pada baris 1 dan 2. Pada baris 3 kuantor universal dari baris 2 dihilangkan, dan pada baris 4 klaim diturunkan dengan modus ponens. Kalkulus yang terdiri dari dua aturan inferensi yang diberikan tidak lengkap. Namun, dapat diperluas menjadi prosedur lengkap dengan penambahan aturan inferensi lebih lanjut. Fakta nontrivial ini sangat penting untuk matematika dan AI. Ahli logika Austria Kurt Gödel membuktikan pada tahun 1931 bahwa.

Teorema 3.3
(Teorema Kelengkapan Gödel) Logika predikat orde pertama selesai. Artinya, ada kalkulus yang dengannya setiap proposisi yang merupakan konsekuensi dari basis pengetahuan KB |= ϕ dapat dibuktikan. Jika KB u, maka dinyatakan bahwa KB ⊢ ϕ.
Oleh karena itu, setiap proposisi benar dalam logika predikat orde pertama dapat dibuktikan. Tapi apakah kebalikannya juga benar? Apakah semua yang dapat kita peroleh secara sintaksis benar-benar benar? Jawabannya iya":

Teorema 3.4 (Kebenaran)
Ada kalkuli yang hanya dapat dibuktikan dengan proposisi benar. Artinya, jika KB |- ϕ berlaku, maka KB |= ϕ.

Faktanya, hampir semua batu yang diketahui benar. Lagi pula, tidak masuk akal untuk bekerja dengan metode pembuktian yang salah. Oleh karena itu, provabilitas dan konsekuensi semantik adalah konsep yang setara, selama kalkulus yang benar dan lengkap

digunakan. Dengan demikian logika predikat orde pertama menjadi alat yang ampuh untuk matematika dan AI. Kalkuli deduksi alami yang disebutkan di atas agak tidak cocok untuk otomatisasi. Hanya kalkulus resolusi, yang diperkenalkan pada tahun 1965 dan pada dasarnya bekerja dengan hanya satu aturan inferensi sederhana, yang memungkinkan pembangunan pembukti teorema otomatis yang kuat, yang kemudian digunakan sebagai mesin inferensi untuk sistem pakar.

3.6 RESOLUSI

Memang, kalkulus resolusi yang benar dan lengkap memicu euforia logika selama tahun 1970-an. Banyak ilmuwan percaya bahwa seseorang dapat merumuskan hampir setiap tugas representasi pengetahuan dan penalaran dalam PL1 dan kemudian menyelesaikannya dengan pengungkap otomatis. Logika predikat, bahasa ekspresif yang kuat, bersama dengan kalkulus bukti lengkap tampaknya menjadi mesin cerdas universal untuk mewakili pengetahuan dan memecahkan banyak masalah sulit (Gambar 3.3).

Jika seseorang memasukkan serangkaian aksioma (yaitu, basis pengetahuan) dan kueri ke dalam mesin logika seperti input, mesin mencari bukti dan mengembalikannya—karena ada dan akan ditemukan—sebagai output. Dengan teorema kelengkapan Gödel dan karya Herbrand sebagai landasan, banyak yang diinvestasikan ke dalam mekanisasi logika. Visi mesin yang bisa, dengan basis pengetahuan PL1 non-kontradiksi yang sewenang-wenang, membuktikan setiap permintaan yang benar sangat menarik. Oleh karena itu, hingga saat ini banyak batu pembuktian untuk PL1 yang sedang dikembangkan dan direalisasikan dalam bentuk pembuktian teorema. Sebagai contoh, di sini kami menjelaskan kalkulus resolusi yang penting secara historis dan banyak digunakan dan menunjukkan kemampuannya. Alasan pemilihan resolusi sebagai contoh kalkulus pembuktian dalam buku ini adalah, sebagaimana dinyatakan, kepentingan historis dan didaktiknya. Saat ini, resolusi hanya mewakili salah satu dari banyak batu yang digunakan dalam alat pengukur kinerja tinggi.

Gambar 3.3 Mesin logika universal

Kita mulai dengan mencoba mengkompilasi bukti pada Tabel 3.2 dengan basis pengetahuan Contoh 3.2 menjadi bukti resolusi. Pertama, rumus diubah menjadi bentuk normal konjungtif dan kueri yang dinegasikan

$$\neg Q \equiv \neg anak(eve, oscar, anne)$$

ditambahkan ke basis pengetahuan, yang memberikan

$$KB \wedge \neg Q \equiv (anak(eve, anne, oscar)_1 \wedge$$

(¬anak(x, y, z) ∨ anak(x, y, z))$_2$ ∧
(¬anak(eve, oscar, anne))$_3$

Buktinya kemudian bisa terlihat seperti

(2)x/eve, y/anne, z/oscar ≡ (¬anak(eve, oscar, anne) ∨
Anak(eve, oscar, anne))$_4$
Res(3, 4) : ((¬anak(eve, oscar, anne))$_5$
Res(1, 5) : ()$_6$

dimana, pada langkah pertama, variabel x, y, z digantikan oleh konstanta. Kemudian dua langkah resolusi mengikuti di bawah penerapan aturan resolusi umum dari (2.2), yang diambil tidak berubah dari logika proposisional.

Keadaan dalam contoh berikut agak lebih kompleks. Kami berasumsi bahwa setiap orang mengenal ibunya sendiri dan bertanya apakah Henry mengenal seseorang. Dengan simbol fungsi "ibu" dan predikat "tahu", kita harus menurunkan kontradiksi dari

$$(tahu(x, ibu(x))_1 \wedge (\neg tahu\ (henry, y))_2,$$

Dengan penggantian x/henry, y/mother(henry) kita memperoleh pasangan klausa kontradiktif

$$(tahu(henry, ibu\ (henry)))_1 \wedge (\neg tahu\ (henry, ibu(henry)))_2,$$

Langkah penggantian ini disebut unifikasi. Kedua literal itu saling melengkapi, yang berarti mereka sama selain tanda-tandanya. Klausa kosong sekarang dapat diturunkan dengan langkah resolusi, yang menunjukkan bahwa Henry memang mengenal seseorang (ibunya). Kami mendefinisikan

Definisi 3.7

Dua literal disebut unfifiable jika ada substitusi r untuk semua variabel yang membuat literalnya sama. R seperti itu disebut pemersatu. Sebuah σ disebut pemersatu paling umum (MGU) jika semua pemersatu lain dapat diperoleh darinya dengan substitusi variabel.

Contoh 3.6

Kita ingin menyatukan literal *p(f(g(x)), y, z)* dan *p(u, u, f(u)).* Beberapa pemersatu adalah

σ_1	:	*y/f(g(x)),*	*u/f(g(x)),*
σ_2	: *x/h(v),*	*y/f(g(h(v))),*	*u/f(g(h(v)))*
σ_3	: *x/h(h(v)),*	*y/f(g(h(h(v)))),*	*u/f(g(h(h(v))))*
σ_4	: *x/h(a)*	*y/f(g(h(a))),*	*u/f(g(h(a)))*
σ_5	: *x/a,*	*y/f(g(a)),*	*u/f(g(a)*

di mana σ_1 adalah pemersatu paling umum. Pemersatu lainnya dihasilkan dari σ_1 melalui substitusi *x/h(v), x/h(h(v), x/h(a), x/a*

Kita dapat melihat dalam contoh ini bahwa selama penyatuan literal, simbol predikat dapat diperlakukan seperti simbol fungsi. Artinya, literal diperlakukan seperti istilah. Implementasi algoritma unifikasi memproses argumen fungsi secara berurutan. Istilah disatukan secara rekursif selama struktur istilah. Algoritma penyatuan paling sederhana sangat cepat dalam banyak kasus. Dalam kasus terburuk, bagaimanapun, waktu komputasi dapat tumbuh secara eksponensial dengan ukuran istilah. Karena untuk pembuktian otomatis, banyak sekali upaya penyatuan yang gagal atau sangat sederhana, dalam banyak kasus kompleksitas kasus terburuk tidak memiliki efek dramatis. Algoritme penyatuan tercepat memiliki kompleksitas yang hampir linier bahkan dalam kasus terburuk.

Kami sekarang dapat memberikan aturan resolusi umum untuk logika predikat:

Definisi 3.8
Aturan resolusi untuk dua klausa dalam bentuk normal konjungtif berbunyi

$$\frac{(\mathrm{A}_1 \vee A_m \vee B, \quad (\neg B' \vee C_1 \vee \ldots \vee C_n) \quad \sigma(\mathrm{B}) = \sigma\mathrm{B}'}{(\sigma(A_1) \vee \sigma(A_m) \vee \sigma(C_1) \vee \ldots \vee \sigma(C_n))}$$

di mana σ adalah MGU dari B dan B'.

Teorema 3.5
Aturan resolusi benar. Artinya, pelarut adalah konsekuensi semantik dari dua klausa induk. Untuk Kelengkapan, bagaimanapun, kita masih membutuhkan tambahan kecil, seperti yang ditunjukkan pada contoh berikut.

Contoh 3.7
Paradoks Russell yang terkenal berbunyi "Ada tukang cukur yang mencukur semua orang yang tidak mencukur dirinya sendiri." Pernyataan ini kontradiktif, artinya tidak memuaskan. Kami ingin menunjukkan ini dengan resolusi. Diformalkan dalam PL1, paradoksnya berbunyi

$$\forall x \; mencukur(tukang\ cukur, x) \Leftrightarrow \neg mencukur(x, x)$$

dan transformasi ke dalam bentuk klausa menghasilkan (lihat Latihan 3.6)

$$(\neg mencukur(tukang\ cukur) \vee \neg mencukur(x, x))_1 \wedge (mencukur(tukang\ cukur, x) \vee mencukur(x, x))_2$$

Dari dua klausa ini kita dapat memperoleh beberapa tautologi, tetapi tidak ada kontradiksi. Dengan demikian resolusi tidak lengkap. Kita masih membutuhkan aturan inferensi lebih lanjut.

Definisi 3.9
Faktorisasi klausa dilakukan dengan

$$\frac{A_1 \vee A_2 \vee \ldots\ldots \vee A_n) \qquad \sigma(A_1) = (A_2)}{(\sigma(A_2) \vee \ldots\ldots \vee \sigma(A_n))}$$

di mana σ adalah MGU dari A_1 dan A_2.
Sekarang kontradiksi dapat diturunkan dari (3.7)

Fak(1, $\sigma : x/tukang\ cukur$*)* : $(\neg mencukur(tukang\ cukur, tukang\ cukur))_3$
Fak(2, $\sigma : x/tukang\ cukur$*)* : $(mencukur(tukang\ cukur, tukang\ cukur))_4$
Res(3, 4) : $()_5$

dan kami menegaskan:

Teorema 3.6
Aturan resolusi (3.6) bersama dengan aturan faktorisasi (3.9) adalah sanggahan selesai. Artinya, dengan penerapan langkah-langkah faktorisasi dan resolusi, klausa kosong dapat diturunkan dari rumus apa pun yang tidak memuaskan dalam bentuk normal konjungtif.

3.7 STRATEGI RESOLUSI

Sementara kelengkapan resolusi penting bagi pengguna, pencarian bukti bisa sangat membuat frustrasi dalam praktiknya. Alasan untuk ini adalah ruang pencarian kombinatorial yang sangat besar. Bahkan jika hanya ada sedikit pasangan klausa di KB∧¬Q di awal, Prover menghasilkan klausa baru dengan setiap langkah resolusi, yang meningkatkan jumlah kemungkinan langkah resolusi di iterasi berikutnya. Oleh karena itu, telah lama dicoba untuk mengurangi ruang pencarian menggunakan strategi khusus, lebih disukai tanpa kehilangan kelengkapan. Strategi yang paling penting adalah sebagai berikut.

Resolusi unit mengutamakan langkah-langkah resolusi di mana salah satu dari dua klausa hanya terdiri dari satu literal, yang disebut klausa unit. Strategi ini mempertahankan kelengkapan dan dalam banyak kasus, tetapi tidak selalu, pada pengurangan ruang pencarian. Oleh karena itu proses heuristik.

Seseorang memperoleh jaminan pengurangan ruang pencarian dengan penerapan *set strategi dukungan*. Di sini subset dari KB ∧ ¬Q didefinisikan sebagai set support (SOS). Setiap langkah resolusi harus melibatkan klausa dari SOS, dan pelarut ditambahkan ke SOS. Strategi ini tidak lengkap. Ini menjadi lengkap ketika dipastikan bahwa himpunan klausa terpenuhi tanpa SOS (lihat Latihan 3.7. Kueri yang dinegasikan ¬Q sering digunakan sebagai SOS awal.

Dalam *resolusi input*, klausa dari set input KB ∧ ¬Q harus dilibatkan dalam setiap langkah resolusi. Strategi ini juga mengurangi ruang pencarian, tetapi dengan mengorbankan kelengkapan.

Dengan aturan literal murni semua klausa yang mengandung literal yang tidak ada literal pelengkap dalam klausa lain dapat dihapus. Aturan ini mengurangi ruang pencarian dan lengkap, dan oleh karena itu digunakan oleh hampir semua pemecah resolusi. Jika literal klausa K1 mewakili subset dari literal klausa K_2, maka K_2 dapat dihapus. Misalnya, klausa

$$(hujan(hari\_ini) \Longrightarrow jalan\_basah(hari\_ini))$$

berlebihan jika street_wet(hari ini) sudah valid. Langkah reduksi yang penting ini disebut subsumption. Subsumsi juga selesai.

3.8 EKUITAS

Kesetaraan adalah penyebab yang sangat tidak nyaman dari pertumbuhan eksplosif dari ruang pencarian. Jika kita menambahkan (3.1) dan aksioma persamaan yang dirumuskan dalam (3.2) pada basis pengetahuan, maka klausa simetri ¬x = y ∨ y = x dapat disatukan dengan setiap persamaan positif atau negatif, misalnya . Ini mengarah pada penurunan klausa dan persamaan baru di mana aksioma kesetaraan dapat diterapkan lagi, dan seterusnya. Aksioma transitivitas dan substitusi memiliki konsekuensi yang serupa. Karena itu, aturan inferensi khusus untuk kesetaraan telah dikembangkan yang diperoleh tanpa aksioma kesetaraan eksplisit dan, khususnya, mengurangi ruang pencarian. Demodulasi, misalnya, memungkinkan substitusi istilah t_2 untuk t_1, jika persamaan $t_1 = t_2$ ada. Persamaan $t_1 = t_2$ diterapkan melalui penyatuan ke suku t sebagai berikut:

$$\frac{t_1 = t_2, \quad (\ldots.t.\ldots.), \quad \sigma(t_1) = \sigma(1)}{(\ldots.\sigma(t_2).\ldots.)}$$

Agak lebih umum adalah paramodulasi, yang bekerja dengan persamaan kondisional.

Persamaan $t_1 = t_2$ memungkinkan substitusi suku t_1 oleh t_2 serta substitusi t_2 oleh t_1. Biasanya tidak ada gunanya membalikkan substitusi yang telah dilakukan. Sebaliknya, persamaan sering digunakan untuk menyederhanakan istilah. Dengan demikian mereka sering digunakan dalam satu arah saja. Persamaan yang hanya digunakan dalam satu arah disebut persamaan berarah. Pemrosesan persamaan terarah yang efisien dicapai dengan apa yang disebut sistem penulisan ulang istilah. Untuk rumus dengan banyak persamaan, terdapat pembuktian persamaan khusus.

3.9 PEMBUKTI TEOREMA OTOMATIS

Implementasi dari proof calculi pada komputer disebut sebagai theoremprovers. Seiring dengan prover khusus untuk himpunan bagian dari PL1 atau aplikasi khusus, saat ini terdapat seluruh lini prover otomatis untuk logika predikat penuh dan logika tingkat tinggi,

yang hanya akan dibahas sedikit di sini. Ikhtisar sistem yang paling penting dapat ditemukan di [McC].

Salah satu aplikasi resolusi tertua dikembangkan di Argonne National Laboratory di Chicago. Berdasarkan perkembangan awal mulai tahun 1963, Otter, dibuat pada tahun 1984. Di atas semua, Otter berhasil menerapkan bidang khusus matematika, seperti yang dapat dipelajari dari halaman rumahnya:

> *"Saat ini, aplikasi utama Otter adalah penelitian aljabar abstrak dan logika formal. Berang-berang dan pendahulunya telah digunakan untuk menjawab banyak pertanyaan terbuka di bidang semigrup berhingga, aljabar Boolean terner, kalkuli logika, logika kombinasi, teori grup, teori kisi, dan geometri aljabar.*

Beberapa tahun kemudian University of Technology, Munich, menciptakan SETHEO yang berkinerja tinggi berdasarkan pada teknologi PROLOG cepat. Dengan tujuan mencapai kinerja yang lebih tinggi, sebuah implementasi untuk komputer paralel dikembangkan dengan nama PARTHEO. Ternyata tidak ada gunanya menggunakan perangkat keras khusus dalam pembuktian teorema, seperti halnya di bidang AI lainnya, karena komputer ini sangat cepat disusul oleh prosesor yang lebih cepat dan algoritme yang lebih cerdas. Munich juga merupakan tempat kelahiran E, seorang ahli persamaan modern pemenang penghargaan, yang akan kita kenal dalam contoh berikut. Di beranda E, seseorang dapat membaca karakterisasi ringkas dan ironis berikut, yang bagian keduanya secara kebetulan berlaku untuk semua pemeriksa otomatis yang ada saat ini.

> *"E adalah pembuktian teorema persamaan murni untuk logika klausa. Artinya, ini adalah program tempat Anda dapat memasukkan spesifikasi matematis (dalam logika klausa dengan kesetaraan) dan hipotesis, dan yang kemudian akan berjalan selamanya, menggunakan semua sumber daya mesin Anda. Sangat kadang-kadang ia akan menemukan bukti untuk hipotesis dan memberi tahu Anda ;-)."*

Menemukan bukti untuk proposisi yang benar tampaknya sangat sulit sehingga pencarian hanya berhasil sangat jarang, atau hanya setelah waktu yang sangat lama—jika sama sekali. Kami akan membahas ini secara lebih rinci di Bab. 4. Namun, harus disebutkan di sini bahwa tidak hanya komputer, tetapi juga kebanyakan orang kesulitan menemukan bukti formal yang ketat.

Meskipun ternyata komputer sendiri dalam banyak kasus tidak mampu menemukan bukti, hal terbaik berikutnya adalah membangun sistem yang bekerja secara semi-otomatis dan memungkinkan kerja sama yang erat dengan pengguna. Dengan demikian manusia dapat lebih menerapkan pengetahuannya tentang domain aplikasi khusus dan mungkin membatasi pencarian buktinya. Salah satu ahli interaktif paling sukses untuk logika predikat tingkat tinggi adalah Isabelle, produk umum Universitas Cambridge dan Universitas Teknologi, Munich.

Siapa pun yang mencari periset berkinerja tinggi harus melihat hasil terkini CASC (CADE ATP System Competition).[10] Di sini kami menemukan bahwa pemenang dari tahun 2001 hingga 2006 dalam kategori bentuk normal PL1 dan klausa adalah Vampir dari Manchester. Yang bekerja dengan varian resolusi dan pendekatan khusus untuk kesetaraan. Sistem Waldmeister dari Max Planck Institute di Saarbrücken telah memimpin selama bertahun-tahun dalam pembuktian kesetaraan. Banyak posisi teratas sistem Jerman di CASC menunjukkan bahwa kelompok penelitian Jerman di bidang pembuktian teorema otomatis memainkan peran utama, hari ini maupun di masa lalu.

[10] CADE adalah "Konferensi Pengurangan Otomatis" tahunan [CAD] dan ATP adalah singkatan dari "Pembukti Teorema Otomatis".

3.10 CONTOH MATEMATIKA

Kami sekarang ingin mendemonstrasikan penerapan proofer otomatis dengan Prover E yang disebutkan di atas.Eisprover kualitas khusus yang sangat mengecilkan ruang pencarian melalui perlakuan kesetaraan yang dioptimalkan. Kami ingin membuktikan bahwa elemen netral kiri dan kanan dalam semigrup adalah sama. Pertama kami meresmikan klaim langkah demi langkah.

Definisi 3.10

Suatu struktur (M,·) yang terdiri dari himpunan M dengan operasi dalam dua tempat "·" disebut semigrup jika hukum asosiatifitas

$$\forall x \ \forall y \ \forall z \ (x \,.\, y) \,.\, z = x \,.\, (y \,.\, z)$$

memegang. Suatu elemen e∈M disebut netral-kiri (netral-kanan) jika $\forall x \ e \,.\, x = x$ ($\forall x \ x \,.\, e = x$).

Masih harus ditunjukkan bahwa

Teorema 3.7

Jika suatu semigrup memiliki elemen netral kiri e_l dan elemen netral kanan e_r, maka $e_l = e_r$. Pertama kita membuktikan teorema semi-formal dengan penalaran matematis intuitif. Jelas berlaku untuk semua x 2 M bahwa

Persamaan 3.8

$$e_t \,.\, x = x$$

Jika kita menetapkan $x = e_r$ pada (3.8) dan $x = e_l$ pada (3.9), kita memperoleh dua persamaan $e_l \,.\, e_r = e_r$ dan $e_l \,.\, e_r = e_l$. Menggabungkan dua persamaan ini menghasilkan

Persamaan 3.9

$$x \,.\, e_r = x$$

yang ingin kita buktikan. Pada langkah terakhir, kebetulan, kami menggunakan fakta bahwa kesetaraan adalah simetris dan transitif. Sebelum kami menerapkan pembuktian otomatis, kami melakukan pembuktian resolusi secara manual. Pertama kita formalkan query yang dinegasi dan basis pengetahuan KB, yang terdiri dari aksioma sebagai klausa dalam bentuk normal konjungsi:

$(\neg e_t = e_r)_1$
$(m(m(x, y),z) = m(x, m(y, z)))_2$
$(m(et, x) = x)_3$
$(m(x, er) = x)_4$

Aksioma ekualitas:

$(x = x)_5$
$(\neg x = y \vee y = x)_6$
$(\neg x = y \vee \neg y = z \vee x = z)_7$
$(\neg x = y \vee m(x, z) = m(y, z))_8$
$(\neg x = y \vee m(z, x) = m(z, y))_9$

di mana perkalian diwakili oleh simbol fungsi dua tempat m. Aksioma persamaan dirumuskan secara analog dengan (Persamaan 3.1) dan (Persaman 3.2) . Bukti resolusi sederhana berbentuk

$Res(3, 6, x_6/m(e_t, x_3), y_6/x_3)$: $(x = m(e_1, x))_{10}$
$Res(7, 10, x_7/x_{10}, y_7/m(e_1, x_{10}))$: $(\neg m(e_t, x) = z \vee x = z)_{11}$
$Res(4, 11, x_4/et, x_{11}/e_r, z_{11}/e_t)$: $(e_r = e_r)_{12}$
$Res(1, 12, \emptyset$: $()$.

Di sini, misalnya, $Res(3,6,x_6/m(e_l, x_3), y_6/x_3)$ berarti bahwa dalam resolusi ayat 3 dengan ayat 6, variabel x dari ayat 6 diganti dengan $m(e_l, x_3)$ dengan variabel x dari ayat 3. Secara

analog, y dari ayat 6 diganti dengan x dari ayat 3. Sekarang kita ingin menerapkan E pada masalah tersebut. Klausa diubah menjadi klausa bentuk normal bahasa LOP melalui pemetaan

$$(\neg A \vee ..... \vee \neg Am \vee ..... \vee Bn) \longmapsto B_1; B_n <- A_1, ..., A_m.$$

Sintaks LOP merupakan perpanjangan dari sintaks PROLOG (lihat Bab 5) untuk klausa non Horn. Jadi kita peroleh sebagai file input untuk E

```
<- eq(el,er).                          # query
eq( m(m(X,Y),Z), m(X,m(Y,Z)) ).        # associativity of m
eq( m(el,X), X ).                      # left-neutral element of m
eq( m(X,er), X ).                      # right-neutral element of m
eq(X,X).                               # equality: reflexivity
eq(Y,X) <- eq(X,Y).                    # equality: symmetry
eq(X,Z) <- eq(X,Y), eq(Y,Z).           # equality: transitivity
eq( m(X,Z), m(Y,Z) ) <- eq(X,Y).       # equality: substitution in m
eq( m(Z,X), m(Z,Y) ) <- eq(X,Y).       # equality: substitution in m
```

dimana kesetaraan dimodelkan dengan simbol predikat **eq**. Memanggil peribahasa memberikan

```
unixprompt> eproof halbgr1.lop
# Problem status determined, constructing proof object
# Evidence for problem status starts
     0 : [--eq(el,er)] : initial
     1 : [++eq(X1,X2),--eq(X2,X1)] : initial
     2 : [++eq(m(el,X1),X1)] : initial
     3 : [++eq(m(X1,er),X1)] : initial
     4 : [++eq(X1,X2),--eq(X1,X3),--eq(X3,X2)] : initial
     5 : [++eq(X1,m(X1,er))] : pm(3,1)
     6 : [++eq(X2,X1),--eq(X2,m(el,X1))] : pm(2,4)
     7 : [++eq(el,er)] : pm(5,6)
     8 : [] : sr(7,0)
     9 : [] : 8 : {proof}
# Evidence for problem status ends
```

Literal positif diidentifikasi dengan ++ dan literal negatif ditandai dengan —. Pada baris 0 sampai 4, yang ditandai dengan **initial**, klausa dari data input didaftar lagi. pm(a, b) singkatan dari langkah resolusi antara klausa a dan klausa b. Kami melihat bahwa bukti yang ditemukan oleh E sangat mirip dengan bukti yang dibuat secara manual. Karena kita secara eksplisit memodelkan persamaan dengan predikat persamaan, kekuatan tertentu dari E tidak ikut bermain. Sekarang kita hilangkan aksioma persamaan dan dapatkan

```
<- el = er.                          % query
m(m(X,Y),Z) = m(X,m(Y,Z)) .          % associativity of m
m(el,X) = X .                        % left-neutral element of m
m(X,er) = X .                        % right-neutral element of m
```

sebagai input bagi para peribahasa. Buktinya juga menjadi lebih kompak. Kita melihat dalam output dari pembuktian berikut bahwa pembuktian pada dasarnya terdiri dari satu langkah inferensi pada dua klausa 1 dan 2.

```
unixprompt> eproof halbgr1a.lop
# Problem status determined, constructing proof object
# Evidence for problem status starts
     0 : [--equal(el, er)] : initial
     1 : [++equal(m(el,X1), X1)] : initial
     2 : [++equal(m(X1,er), X1)] : initial
     3 : [++equal(el, er)] : pm(2,1)
     4 : [--equal(el, el)] : rw(0,3)
     5 : [] : cn(4)
     6 : [] : 5 : {proof}
# Evidence for problem status ends
```

Pembaca sekarang dapat melihat lebih dekat pada kemampuan E (Latihan 3.9).

3.11 APLIKASI

Dalam matematika, pembuktian teorema otomatis digunakan untuk tugas-tugas khusus tertentu. Sebagai contoh, teorema empat warna yang penting dalam teori graf pertama kali dibuktikan pada tahun 1976 dengan bantuan sebuah pembuktian khusus. Namun, penyelidik otomatis masih memainkan peran kecil dalam matematika.

Di sisi lain, pada awal AI, logika predikat sangat penting untuk pengembangan sistem pakar dalam aplikasi praktis. Karena masalah ketidakpastian pemodelan, sistem pakar saat ini paling sering dikembangkan menggunakan formalisme lain.

Saat ini logika memainkan peran yang semakin penting dalam tugas-tugas verifikasi. Verifikasi program otomatis saat ini merupakan area penelitian penting antara AI dan rekayasa perangkat lunak. Sistem perangkat lunak yang semakin kompleks sekarang mengambil alih tugas dengan tanggung jawab dan relevansi keamanan yang semakin besar. Di sini diperlukan bukti karakteristik keamanan tertentu dari suatu program. Bukti seperti itu tidak dapat dilakukan melalui pengujian program yang sudah selesai, karena secara umum tidak mungkin untuk menerapkan program ke semua input yang mungkin. Oleh karena itu, ini adalah domain yang ideal untuk sistem inferensi umum atau bahkan khusus. Antara lain, protokol kriptografi yang digunakan saat ini yang karakteristik keamanannya telah diverifikasi secara otomatis. Tantangan lebih lanjut untuk penggunaan Prover otomatis adalah sintesis perangkat lunak dan perangkat keras. Untuk tujuan ini, misalnya, para ahli harus mendukung insinyur perangkat lunak dalam menghasilkan program dari spesifikasi.

Penggunaan kembali perangkat lunak juga sangat penting bagi banyak programmer saat ini. Pemrogram mencari program yang mengambil data input dengan properti tertentu dan menghitung hasilnya dengan properti yang diinginkan. Algoritme pengurutan menerima data input dengan entri dari tipe data tertentu dan dari ini membuat permutasi entri ini dengan properti bahwa setiap elemen kurang dari atau sama dengan elemen berikutnya. Pemrogram pertama-tama merumuskan spesifikasi kueri dalam PL1 yang terdiri dari dua bagian. Bagian pertama PRE_Q terdiri dari prasyarat, yang harus ada sebelum program yang diinginkan diterapkan. Bagian kedua $POST_Q$ berisi postconditions, yang harus ada setelah program yang diinginkan diterapkan.

$$PRE_Q \Longrightarrow PRE_M$$

Semua kondisi yang diperlukan sebagai prasyarat untuk penerapan modul M harus muncul sebagai prasyarat dalam kueri. Jika, misalnya, sebuah modul dalam database hanya menerima daftar bilangan bulat, maka daftar bilangan bulat sebagai input juga harus muncul sebagai prasyarat dalam kueri. Persyaratan tambahan dalam kueri yang, misalnya, hanya angka genap yang muncul, tidak menyebabkan masalah. Selanjutnya, itu harus berlaku untuk postconditions bahwa

$$POST_M \Longrightarrow POST_Q$$

Artinya, setelah penerapan modul, semua atribut yang dibutuhkan kueri harus dipenuhi. Kami sekarang menunjukkan penerapan pembuktian teorema untuk tugas ini.

Contoh 3.8 VDM-SL, Bahasa Spesifikasi Metode Pengembangan Wina, sering digunakan sebagai bahasa untuk spesifikasi pra dan pascakondisi. Asumsikan bahwa dalam database perangkat lunak deskripsi modul ROTATE tersedia, yang memindahkan elemen daftar pertama ke akhir daftar. Kami sedang mencari modul SHUFFLE, yang membuat permutasi arbitrer dari daftar. Dua spesifikasi membaca

ROTATE($l : List$) $l' : List$
pre *true*
post
$(l = [] \Rightarrow l' = []) \wedge$
$(l \neq [] \Rightarrow l' = (\text{tail}\, l)\hat{}[\text{head}\, l])$

SHUFFLE($x : List$) $x' : List$
pre *true*
post $\forall i : Item \cdot$
$(\exists x_1, x_2 : List \cdot x = x_1\hat{}[i]\hat{}x_2 \Leftrightarrow$
$\exists y_1, y_2 : List \cdot x' = y_1\hat{}[i]\hat{}y_2)$

Di sini "^" adalah singkatan dari rangkaian daftar, dan "·" memisahkan quantifier dengan variabelnya dari sisa rumus. Fungsi *"head l"* dan *"tail l"* masing-masing memilih elemen pertama dan sisanya dari daftar. Spesifikasi **SHUFFLE** menunjukkan bahwa setiap elemen daftar i yang ada pada daftar (x) sebelum penerapan **SHUFFLE** harus ada pada hasil (x') setelah penerapan, dan sebaliknya. Sekarang harus ditunjukkan bahwa rumus (PRE_Q) $\Longrightarrow$ PRE_M) $\wedge$ ($POST_M \Longrightarrow POST_Q$) adalah konsekuensi dari basis pengetahuan yang berisi deskripsi dari Daftar tipe data. Kedua spesifikasi VDM-SL menghasilkan tugas pembuktian

$$\forall l, l', x, x' : List \;.\; (l \Longrightarrow x\; l' = x' \wedge (w \Longrightarrow w)) \wedge$$
$$(l = x \wedge l' = x' \wedge ((l = [] \Longrightarrow = []) \wedge (l \neq [] \Longrightarrow l' = (tll) \hat{}\, [hd\ l]) \Longrightarrow$$
$$\forall i : Item \;.\; (\exists x_1, x_2 : List \;.\; x = x_1\, \hat{}[i]\hat{}x_2 \Leftrightarrow \exists y_1, y_2 : List \;.\; x' = y_i\hat{}[i]\hat{}y_2)))$$

yang kemudian dapat dibuktikan dengan SETHEO.

Di tahun-tahun mendatang web semantik kemungkinan akan mewakili aplikasi penting PL1. Isi dari World Wide Web seharusnya dapat diinterpretasikan tidak hanya untuk manusia, tetapi juga untuk mesin. Untuk tujuan ini situs web sedang dilengkapi dengan deskripsi semantik mereka dalam bahasa deskripsi formal. Pencarian informasi di

web dengan demikian akan menjadi jauh lebih efektif daripada saat ini, di mana pada dasarnya hanya blok penyusun teks yang dapat dicari.

Subset logika predikat yang dapat ditentukan digunakan sebagai bahasa deskripsi. Pengembangan kalkuli yang efisien untuk penalaran sangat penting dan terkait erat dengan bahasa deskripsi. Permintaan untuk mesin pencari yang beroperasi secara semantik di masa depan dapat (secara informal) berbunyi: Di mana di Semarang Minggu depan pada ketinggian di bawah 2000 meter akan ada cuaca yang baik ? Untuk menjawab pertanyaan seperti itu, diperlukan kalkulus yang mampu bekerja dengan sangat cepat pada kumpulan fakta dan aturan yang besar. Di sini, istilah fungsi bersarang yang kompleks kurang penting. Sebagai kerangka deskripsi dasar, World Wide Web Consortium mengembangkan bahasa RDF *(Resource Description Framework).* Membangun RDF, bahasa OWL (Web Ontology Language) yang secara signifikan lebih kuat memungkinkan deskripsi hubungan antara objek dan kelas objek, mirip dengan PL1. Ontologi adalah deskripsi hubungan antara objek yang mungkin.

Kesulitan ketika membangun deskripsi situs web yang tak terhitung banyaknya adalah pengeluaran pekerjaan dan juga memeriksa kebenaran deskripsi semantik. Di sini sistem pembelajaran mesin untuk pembuatan deskripsi otomatis bisa sangat membantu. Penggunaan yang menarik dari generasi semantik "otomatis" di web diperkenalkan oleh Luis von Ahn dari Carnegie Mellon University. Dia mengembangkan permainan komputer di mana para pemain, yang didistribusikan melalui jaringan, seharusnya secara kolaboratif menggambarkan gambar dengan kata-kata kunci . Jadi gambar-gambar tersebut diberi semantik dengan cara yang menyenangkan tanpa biaya.

3.12 RINGKASAN

Kami telah menyediakan dasar, istilah, dan prosedur yang paling penting dari logika predikat dan kami telah menunjukkan bahwa bahkan salah satu tugas intelektual yang paling sulit, yaitu pembuktian teorema matematika, dapat diotomatisasi. Prover otomatis dapat digunakan tidak hanya dalam matematika, melainkan, khususnya, dalam tugas-tugas verifikasi dalam ilmu komputer. Untuk penalaran sehari-hari, bagaimanapun, logika predikat dalam banyak kasus tidak cocok. Dalam bab-bab selanjutnya dan bab-bab berikutnya kami menunjukkan titik-titik lemahnya dan beberapa alternatif modern yang menarik. Selanjutnya, kami akan menunjukkan di Chap. 5 yang dapat diprogram secara elegan dengan logika dan ekstensi proseduralnya.

3.13 LATIHAN

Latihan 3.1

Berikan predikat tiga tempat "anak" dan predikat satu tempat "perempuan" dari Contoh 3.2 Tentukan:

a. Predikat satu tempat "laki-laki".
b. Predikat dua tempat "ayah" dan "ibu".
c. Predikat dua tempat "saudara".
d. Sebuah predikat “orang tua (x, y, z)”, yang benar jika dan hanya jika x adalah ayah dan y adalah ibu dari z.
e. Predikat “paman (x, y)”, yang benar jika dan hanya jika x adalah paman dari y (gunakan predikat yang telah ditentukan).
f. Predikat dua tempat “leluhur” yang artinya: nenek moyang adalah orang tua, kakek-nenek, dll dari banyak generasi secara sewenang-wenang.

Latihan 3.2

Bentuklah pernyataan-pernyataan berikut dalam logika predikat:

(a) Setiap orang memiliki ayah dan ibu.
(b) Beberapa orang memiliki anak.
(c) Semua burung terbang.
(d) Ada binatang yang memakan (sebagian) binatang pemakan biji-bijian.
(e) Setiap hewan memakan tumbuhan atau hewan pemakan tumbuhan yang jauh lebih kecil dari dirinya.

Latihan 3.3

Sesuaikan Latihan 3.1 dengan menggunakan simbol fungsi satu tempat dan persamaan alih-alih “ayah” dan “ibu”.

Latihan 3.4

Berikan aksioma logika predikat untuk relasi dua tempat “<” sebagai orde total. Untuk total order kita harus memiliki

(1) Setiap dua elemen sebanding.
(2) Itu simetris.
(3) Ini transitif.

Latihan 3.5

Satukan (jika mungkin) suku-suku berikut dan berikan MGU dan suku-suku yang dihasilkan.

a. p(x, f(y)), p(f(z), u)
b. p(x, f(x)), p(y, y)
c. x =4− 7 . x, cos y = z
d. x < 2 . x, 3 < 6
e. q(f(x, y, z), f(g(w; w), g(x, x), g(y, y))), q(u, u)

Latihan 3.6

(a) Ubah Paradoks Russell dari Contoh 3.7 menjadi CNF.
(b) Tunjukkan bahwa klausa kosong tidak dapat diturunkan menggunakan resolusi tanpa faktorisasi dari (3.7). Cobalah untuk memahaminya secara intuitif.

Latihan 3.7

(a) Mengapa resolusi dengan set strategi dukungan tidak lengkap?
(b) Membenarkan (tanpa membuktikan) mengapa set strategi dukungan menjadi lengkap jika (KB ∨ ¬Q)\SOS memenuhi.
(c) Mengapa resolusi dengan aturan literal murni lengkap?

Latihan 3.8

Rumuskan dan buktikan dengan resolusi bahwa dalam semigrup dengan setidaknya dua elemen berbeda a, b, elemen kiri-netral e, dan elemen nol kiri n, kedua elemen ini harus berbeda, yaitu n ≠ e. Gunakan demodulasi, yang memungkinkan penggantian "suka dengan suka".

Latihan 3.9

Dapatkan pembuktian teorema E [Sch02] atau pembuktian lainnya dan buktikan pernyataan-pernyataan berikut. Bandingkan bukti-bukti ini dengan yang ada di teks.

(a) Klaim dari Contoh 2.3
(b) Paradoks Russell dari Contoh 3.7
(c) Klaim dari Latihan 3.8.

BAB 4
KETERBATASAN LOGIKA

4.1 MASALAH RUANG PENCARIAN

Seperti yang telah disebutkan di beberapa tempat, dalam mencari bukti hampir selalu ada banyak kemungkinan (bergantung pada kalkulus, berpotensi tak terhingga) untuk penerapan aturan inferensi pada setiap langkah. Hasilnya adalah pertumbuhan eksplosif yang disebutkan di atas dari ruang pencarian (Gambar 4.1). Dalam kasus terburuk, semua kemungkinan ini harus dicoba untuk menemukan buktinya, yang biasanya tidak mungkin dilakukan dalam waktu yang wajar.

Jika kita membandingkan pembuktian otomatis atau sistem inferensi dengan ahli matematika atau ahli manusia yang memiliki pengalaman dalam domain khusus, kita membuat pengamatan yang menarik. Untuk satu hal, matematikawan berpengalaman dapat membuktikan teorema yang jauh dari jangkauan untuk pembukti otomatis. Di sisi lain, penyelidik otomatis melakukan puluhan ribu inferensi per detik. Sebaliknya, manusia mungkin melakukan satu inferensi per detik. Meskipun ahli manusia jauh lebih lambat pada tingkat objek (yaitu, dalam melakukan inferensi), mereka tampaknya memecahkan masalah yang sulit jauh lebih cepat.

Ada beberapa alasan untuk ini. Kita manusia menggunakan kalkuli intuitif yang bekerja pada tingkat yang lebih tinggi dan sering melakukan banyak kesimpulan sederhana dari pembukti otomatis dalam satu langkah. Lebih lanjut, kita menggunakan lemma, yaitu rumus turunan yang benar yang sudah kita ketahui dan oleh karena itu tidak perlu membuktikannya kembali setiap saat. Sementara itu ada juga mesin proofer yang bekerja dengan metode seperti itu. Tetapi bahkan mereka belum dapat bersaing dengan ahli manusia.

Lebih jauh, keuntungan yang jauh lebih penting dari kita manusia adalah intuisi, yang tanpanya kita tidak dapat memecahkan masalah yang sulit [PS09]. Upaya untuk memformalkan intuisi menyebabkan masalah. Pengalaman dalam proyek AI terapan menunjukkan bahwa dalam domain kompleks seperti kedokteran atau matematika, sebagian besar ahli tidak dapat merumuskan meta-pengetahuan intuitif ini secara verbal, apalagi memformalkannya. Oleh karena itu kita tidak dapat memprogram pengetahuan ini atau mengintegrasikannya ke dalam kalkuli dalam bentuk heuristik. Heuristik adalah metode yang dalam banyak kasus dapat sangat menyederhanakan atau mempersingkat jalan ke tujuan, tetapi dalam beberapa kasus (biasanya jarang) dapat sangat memperpanjang jalan ke tujuan. Pencarian heuristik penting tidak hanya untuk logika, tetapi umumnya untuk pemecahan masalah di AI dan karena itu akan ditangani secara menyeluruh di Bab 6.

Gambar 4.1 Kemungkinan konsekuensi dari ledakan ruang pencarian

Pendekatan yang menarik, yang telah dilakukan sejak sekitar tahun 1990, adalah penerapan teknik pembelajaran mesin untuk pembelajaran heuristik untuk mengarahkan pencarian sistem inferensi, yang akan kita sketsa singkat sekarang. Pembukti resolusi memiliki, selama mencari bukti, ratusan atau lebih kemungkinan untuk langkah resolusi pada setiap langkah, tetapi hanya sedikit yang mengarah ke tujuan. Akan sangat ideal jika si ahli dapat bertanya kepada oracle dua klausa mana yang harus digunakan pada langkah berikutnya untuk menemukan buktinya dengan cepat. Ada upaya untuk membangun modul pengarah bukti tersebut, yang mengevaluasi berbagai alternatif untuk langkah selanjutnya dan kemudian memilih alternatif dengan peringkat terbaik. Dalam kasus resolusi, peringkat klausa yang tersedia dapat dihitung dengan fungsi yang menghitung nilai berdasarkan jumlah literal positif, kompleksitas istilah, dll., untuk setiap pasangan klausa yang dapat diselesaikan.

Bagaimana fungsi ini dapat dilaksanakan? Karena pengetahuan ini "intuitif", programmer tidak terbiasa dengannya. Sebaliknya, seseorang mencoba untuk menyalin alam dan menggunakan algoritme pembelajaran mesin untuk belajar dari bukti yang berhasil.

Atribut dari semua pasangan klausa yang berpartisipasi dalam langkah resolusi yang berhasil disimpan sebagai positif, dan atribut dari semua resolusi yang gagal disimpan sebagai negatif. Kemudian, dengan menggunakan data pelatihan ini dan sistem pembelajaran mesin, sebuah program dihasilkan yang dapat menilai pasangan klausa secara heuristik.

Pendekatan yang berbeda dan lebih berhasil untuk meningkatkan penalaran matematis diikuti dengan sistem interaktif yang beroperasi di bawah kendali pengguna. Di sini orang dapat menamai program aljabar komputer seperti Mathematica, Maple, atau Maxima, yang secara otomatis dapat melakukan manipulasi matematika simbolis yang sulit. Pencarian bukti, bagaimanapun, diserahkan sepenuhnya kepada manusia. Pembukti interaktif yang disebutkan di atas, Isabelle memberikan lebih banyak dukungan selama pencarian bukti. Saat ini ada beberapa proyek, seperti Omega dan MKM, [11] untuk pengembangan sistem untuk mendukung matematikawan selama pembuktian.

Ringkasnya, dapat dikatakan bahwa, karena masalah ruang pencarian, penyelidik otomatis saat ini hanya dapat membuktikan teorema yang relatif sederhana dalam domain khusus dengan beberapa aksioma.

[11] www.mathweb.org/mathweb/demo.html.

4.2 *DECIDABILITY* DAN KETIDAKLENGKAPAN

Logika predikat orde pertama menyediakan alat yang ampuh untuk representasi pengetahuan dan penalaran. Kita tahu bahwa ada kalkuli dan pembuktian teorema yang benar dan lengkap. Ketika membuktikan teorema, yaitu pernyataan yang benar, pembuktian seperti itu sangat membantu karena, karena kelengkapan, seseorang mengetahui setelah waktu yang terbatas bahwa pernyataan itu benar-benar benar. Bagaimana jika pernyataan tersebut tidak benar? Teorema kelengkapan (Teorema 3.3) tidak menjawab pertanyaan ini.[12] Secara khusus, tidak ada proses yang dapat membuktikan atau menyangkal rumus apapun dari PL1 dalam waktu yang terbatas, karena menyatakan bahwa

Teorema 4.1

Himpunan rumus yang valid dalam logika predikat orde pertama adalah semidecidable.

Teorema ini menyiratkan bahwa ada program (pembukti teorema) yang, diberikan rumus yang benar (valid) sebagai input, menentukan kebenarannya dalam waktu yang terbatas. Namun, jika rumusnya tidak valid, mungkin saja peribahasa tidak pernah berhenti. (Pembaca dapat bergulat dengan pertanyaan ini dalam Latihan 4.1.) Logika proposisi dapat ditentukan karena metode tabel kebenaran menyediakan semua model rumus dalam waktu yang terbatas. Jelas logika predikat dengan quantifiers dan simbol fungsi bersarang adalah bahasa yang agak terlalu kuat untuk diputuskan.

Di sisi lain, logika predikat tidak cukup kuat untuk banyak tujuan. Seseorang sering ingin membuat pernyataan tentang himpunan predikat atau fungsi. Ini tidak bekerja di PL1 karena hanya mengetahui quantifier untuk variabel, tetapi tidak untuk predikat atau fungsi. Kurt Gödel menunjukkan, tak lama setelah teorema kelengkapannya untuk PL1, kelengkapan itu hilang jika kita memperluas PL1 bahkan secara minimal untuk membangun logika tingkat tinggi. Logika orde pertama hanya dapat mengukur variabel. Logika orde kedua juga dapat mengkuantifikasi rumus orde pertama, dan logika orde ketiga dapat mengkuantifikasi rumus orde kedua. Bahkan menambahkan hanya aksioma induksi untuk bilangan asli membuat logika tidak lengkap. Pernyataan "Jika predikat p(n) berlaku untuk n, maka p(n + 1) juga berlaku", atau

$$\forall p \; p(n) \Longrightarrow p(n+1)$$

adalah proposisi orde kedua karena ia menghitung di atas predikat. Godel membuktikan teorema berikut:

Teorema 4.2

(Teorema ketidaklengkapan Gödel) Setiap sistem aksioma untuk bilangan asli dengan penjumlahan dan perkalian (aritmatika) tidak lengkap. Artinya, ada pernyataan benar dalam aritmatika yang tidak dapat dibuktikan.

Pembuktian Gödel bekerja dengan apa yang disebut Gödelisasi, di mana setiap rumus aritmatika dikodekan sebagai angka. Ia memperoleh nomor Gödel yang unik. Godelisasi sekarang digunakan untuk merumuskan proposisi

F = "Saya tidak dapat dibuktikan"

dalam bahasa aritmatika. Rumus ini benar karena alasan berikut. Asumsikan F salah. Kemudian kita dapat membuktikan F dan karena itu menunjukkan bahwa F tidak dapat dibuktikan. Ini adalah kontradiksi. Jadi F benar dan karena itu tidak dapat dibuktikan.

[12] Hanya saja kasus ini sangat penting dalam praktik, karena jika saya sudah tahu bahwa suatu pernyataan itu benar, saya tidak lagi membutuhkan pembuktian.

Latar belakang yang lebih dalam dari teorema ini adalah bahwa teori matematika (sistem aksioma) dan, lebih umum, bahasa menjadi tidak lengkap jika bahasa menjadi terlalu kuat. Contoh serupa adalah teori himpunan. Bahasa ini sangat kuat sehingga orang dapat merumuskan paradoks dengannya. Ini adalah pernyataan yang bertentangan dengan dirinya sendiri, seperti pernyataan yang sudah kita ketahui dari Contoh 3.7 tentang tukang cukur yang semuanya mencukur mereka yang tidak mencukur sendiri (lihat Latihan 4.2). Dilemanya terletak di dalamnya bahwa dengan bahasa yang cukup kuat untuk menggambarkan matematika dan aplikasi yang menarik, kita menyelundupkan kontradiksi dan ketidaklengkapan melalui pintu belakang. Namun, ini tidak berarti bahwa logika tingkat tinggi sepenuhnya tidak cocok untuk metode formal. Tentu saja ada sistem formal serta pembuktian untuk logika tingkat tinggi.

4.3 PENGUIN TERBANG

Dengan contoh sederhana kami akan menunjukkan masalah mendasar logika dan pendekatan solusi yang mungkin. Mengingat pernyataan

1. Tweety adalah penguin
2. Penguin adalah burung
3. Burung dapat terbang

Diformalkan di PL1, basis pengetahuan hasil KB:

$$Pinguin(tweety)$$
$$Pinguin(x) \Longrightarrow burung(x)$$
$$Burung(x) \Longrightarrow terbang(x)$$

Dari sana (misalnya dengan resolusi) *terbang(tweety)* dapat diturunkan (Gambar 4.2).[13] Terbukti formalisasi atribut terbang penguin tidak cukup. Kami mencoba pernyataan tambahan Penguin tidak bisa y, yaitu

$$Pinguin(x) \Longrightarrow \neg terbang(x)$$

Dari sana ¬*terbang(tweety)* dapat diturunkan. Tapi *terbang(tweety)* masih benar. Oleh karena itu, basis pengetahuan tidak konsisten. Di sini kita melihat karakteristik penting dari logika, yaitu monoton. Meskipun kami secara eksplisit menyatakan bahwa penguin tidak dapat terbang, kebalikannya masih dapat diturunkan.

Definisi 4.1
Suatu logika disebut monotonik jika, untuk basis pengetahuan KB yang berubah-ubah dan suatu rumus sembarang ∅, himpunan rumus yang diturunkan dari KB adalah subset dari rumus yang diturunkan dari KB ∪ ∅.

Jika sekumpulan rumus diperpanjang, maka, setelah ekstensi, semua pernyataan turunan sebelumnya masih dapat dibuktikan, dan pernyataan tambahan berpotensi juga dapat dibuktikan. Himpunan pernyataan yang dapat dibuktikan dengan demikian tumbuh secara monoton ketika himpunan rumus diperpanjang. Untuk contoh kita, ini berarti bahwa perluasan basis pengetahuan tidak akan pernah mengarah ke tujuan kita. Oleh karena itu, kami memodifikasi KB dengan mengganti pernyataan yang jelas salah *"(semua) burung dapat terbang"* dengan pernyataan yang lebih tepat *"(semua) burung kecuali penguin dapat terbang" dan* memperoleh sebagai KB_2 klausa berikut:

[13] Eksekusi formal ini dan bukti sederhana berikut dapat diserahkan kepada pembaca (Latihan 4.3)

Gambar 4.2 Penguin Tweety terbang

Sekarang dunia tampaknya kembali teratur. Kita dapat menurunkan ¬*terbang(tweety)*, tetapi tidak *terbang(tweety*), karena untuk itu kita membutuhkan ¬penguin(x), yang, bagaimanapun, tidak dapat diturunkan. Selama hanya ada penguin di dunia ini, kedamaian akan berkuasa. Namun, setiap burung normal segera menimbulkan masalah. Kami ingin menambahkan gagak Abraxas (dari buku Jerman "The Little Witch") dan mendapatkan

Gagak(abraxas)
Gagak(x) ⟹ burung(x)
Penguin(tweety)
Penguin(x)⟹ bird(x)
Bird(x)) ∧¬ penguin(x) ⟹ terbang(x)
Penguin(x) ⟹ ¬fly(x)

Kita tidak bisa mengatakan apa-apa tentang atribut terbang Abraxas karena kita lupa merumuskan bahwa gagak bukanlah penguin. Jadi kami memperluas KB3 ke KB4:

Gagak(abraxas)
Gagak(x) ⟹ burung(x)
Gagak(x) ⟹ ¬pinguin(x)
Penguin(tweety)
Penguin(x)⟹ bird(x)
Bird(x)) ∧¬ penguin(x) ⟹ terbang(x)
Penguin(x) ⟹ ¬fly(x)

Fakta bahwa burung gagak bukanlah penguin, yang jelas bagi manusia, harus ditambahkan secara eksplisit di sini. Untuk membangun basis pengetahuan dengan semua 9.800 atau lebih jenis burung di seluruh dunia, oleh karena itu harus ditentukan untuk setiap jenis burung (kecuali penguin) yang bukan anggota penguin. Kita harus melanjutkan secara analog untuk semua pengecualian lain seperti burung unta. Untuk setiap objek dalam basis pengetahuan, selain atributnya, semua atribut yang tidak dimilikinya harus dicantumkan.

Untuk mengatasi masalah ini, berbagai bentuk logika non-monotonik telah dikembangkan, yang memungkinkan pengetahuan (rumus) dikeluarkan dari basis pengetahuan. Di bawah nama logika default, logika telah dikembangkan yang memungkinkan objek diberi atribut yang valid selama tidak ada aturan lain yang tersedia. Dalam contoh Tweety, aturan birds can y akan menjadi aturan default. Meskipun upaya besar, logika ini pada saat ini, karena masalah semantik dan praktis, tidak berhasil.

Monoton bisa sangat merepotkan dalam masalah perencanaan yang kompleks di mana dunia dapat berubah. Jika misalnya rumah biru dicat merah, maka setelah itu berwarna merah. Basis pengetahuan seperti

Warna(rumah, biru)
Cat(rumah, merah)
Cat(x, y) ⟹ warna (x, y)

mengarah pada kesimpulan bahwa, setelah dicat, rumah itu berwarna merah dan biru. Masalah yang muncul di sini dalam perencanaan dikenal sebagai masalah bingkai. Solusi untuk ini adalah kalkulus situasi.

Pendekatan yang menarik untuk masalah pemodelan seperti contoh Tweety adalah teori probabilitas. Pernyataan "semua burung bisa terbang" adalah salah. Pernyataan seperti "hampir semua burung bisa terbang" adalah benar. Pernyataan ini menjadi lebih tepat jika kita memberikan probabilitas untuk "burung dapat terbang". Ini mengarah pada logika probabilistik, yang saat ini mewakili sub-area penting AI dan alat penting untuk pemodelan ketidakpastian (lihat Bab 7).

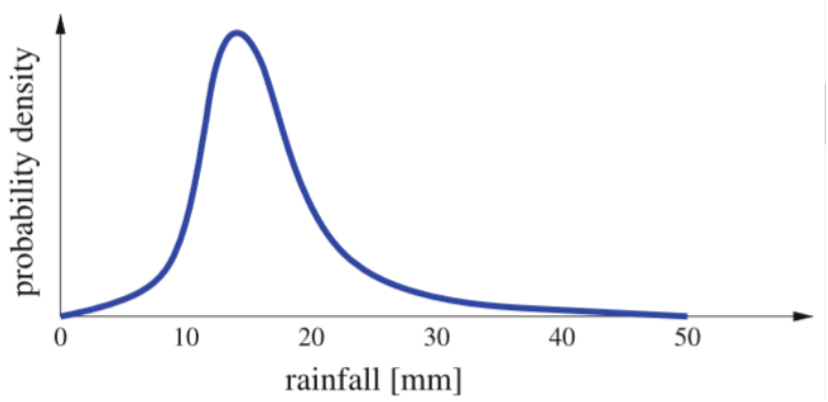
4.4 KETIDAKPASTIAN PEMODELAN

Logika dua nilai hanya dapat dan seharusnya memodelkan keadaan di mana ada nilai benar, salah, dan tidak ada nilai kebenaran lainnya. Untuk banyak tugas dalam penalaran sehari-hari, logika dua nilai karena itu tidak cukup ekspresif. Peraturan

Burung(x) ⟹ terbang(x)

benar untuk hampir semua burung, tetapi untuk beberapa itu salah. Seperti yang telah disebutkan, bekerja dengan probabilitas memungkinkan perumusan ketidakpastian yang tepat. Pernyataan "99% dari semua burung dapat terbang" dapat diformalkan dengan ekspresi

P(burung(x) ⟹ terbang(x)) = 0.99



Gambar 4.3 Probabilitas densitas dari curah hujan variabel kontinu

Dalam Bab. 7 kita akan melihat bahwa di sini lebih baik bekerja dengan probabilitas bersyarat seperti

P(fly|burung) = 0.99

Dengan bantuan jaringan Bayesian, aplikasi kompleks dengan banyak variabel juga dapat dimodelkan.

Model yang berbeda diperlukan untuk pernyataan "Cuacanya bagus". Di sini sering tidak masuk akal untuk berbicara dalam istilah benar dan salah. Variabel cuaca_cukup_bagus tidak boleh dimodelkan sebagai biner, melainkan terus-menerus dengan nilai, misalnya, dalam interval [0, 1]. *Cuaca_cukup_bagus* = 0,7 maka berarti "Cuaca cukup bagus". Logika fuzzy dikembangkan untuk jenis variabel kontinu (kabur).

Teori probabilitas juga menawarkan kemungkinan membuat pernyataan tentang probabilitas variabel kontinu. Sebuah pernyataan dalam laporan cuaca seperti "Ada

kemungkinan besar bahwa akan ada hujan" misalnya dapat diformulasikan secara tepat sebagai kerapatan probabilitas dari formulir

$$P(hujan\_turun = X) = Y$$

dan direpresentasikan secara grafis seperti pada Gambar 4.3. Representasi yang sangat umum dan bahkan dapat divisualisasikan dari kedua jenis ketidakpastian yang telah kita diskusikan, bersama dengan statistik induktif dan teori jaringan Bayesian, memungkinkan, pada prinsipnya, untuk menjawab pertanyaan probabilistik arbitrer. Teori probabilitas serta logika fuzzy tidak secara langsung dapat dibandingkan dengan logika predikat karena mereka tidak mengizinkan variabel atau kuantifier. Dengan demikian mereka dapat dilihat sebagai perpanjangan dari logika proposisional seperti yang ditunjukkan pada tabel berikut.

Tabel 4.1 tabel probabilitas di ekspresikan

Formalisasi	Jumlah nilai benar	Probabilitas di ekspresikan
Logika Proposisional	2	-
Logika Fuzzy	∞	-
Logika Probabilitas Diskrit	N	Ya
Logika Probabilitas Kontinyu	∞	Ya

4.5 LATIHAN

Latihan 4.1

a. Dengan argumen (salah) berikut, orang dapat mengklaim bahwa PL1 dapat ditentukan: Kami mengambil kalkulus bukti lengkap untuk PL1. Dengan itu kita dapat menemukan bukti untuk setiap rumus yang benar dalam waktu yang terbatas. Untuk setiap rumus lain Ø saya melanjutkan sebagai berikut: Saya menerapkan kalkulus pada Ø dan menunjukkan bahwa Ø benar. Jadi Ø salah. Dengan demikian saya dapat membuktikan atau menyangkal setiap rumus di PL1. Temukan kesalahan dalam argumen dan ubah sehingga menjadi benar.
b. Membangun proses keputusan untuk himpunan rumus benar dan tidak memuaskan dalam PL1.

Latihan 4.2

(a) Mengingat pernyataan "Ada seorang tukang cukur yang mencukur setiap orang yang tidak mencukur dirinya sendiri." Pertimbangkan apakah tukang cukur ini mencukur dirinya sendiri. (b) Misalkan M ={ x|x ∉ x}. Jelaskan himpunan ini dan pertimbangkan apakah M berisi dirinya sendiri.

Latihan 4.3

Gunakan pembuktian teorema otomatis (misalnya E [Sch02]) dan terapkan ke semua lima aksiomatisasi yang berbeda dari contoh Tweety. Validasi pernyataan contoh.

BAB 5
PEMROGRAMAN LOGIKA DENGAN PROLOG

Dibandingkan dengan bahasa pemrograman klasik seperti C atau Pascal, Logika memungkinkan untuk mengekspresikan hubungan secara elegan, kompak, dan deklaratif. Pembukti teorema otomatis bahkan mampu memutuskan apakah basis pengetahuan secara logis memerlukan kueri. Kalkulus bukti dan pengetahuan yang disimpan dalam basis pengetahuan dipisahkan secara ketat. Rumus yang dijelaskan dalam bentuk normal klausa dapat digunakan sebagai data input untuk pembuktian teorema apa pun, terlepas dari kalkulus pembuktian yang digunakan. Ini sangat berharga untuk penalaran dan representasi pengetahuan.

Jika seseorang ingin mengimplementasikan algoritma, yang pasti memiliki komponen prosedural, deskripsi deklaratif murni seringkali tidak cukup. Robert Kowalski, salah satu pelopor pemrograman logika, menyatakan hal ini dengan rumus

Algoritma = Logis + Kontrol

Ide ini diwujudkan dalam bahasa PROLOG. PROLOG digunakan dalam banyak proyek, terutama dalam AI dan linguistik komputasi. Kami sekarang akan memberikan pengantar singkat untuk bahasa ini, menyajikan konsep yang paling penting, menunjukkan kekuatannya, dan membandingkannya dengan bahasa pemrograman lain dan pembenar teorema.kan mere

Sintaks bahasa PROLOG hanya mengizinkan klausa Horn. Notasi logika dan sintaks PROLOG disandingkan dalam tabel berikut:

Tabel 5.1 Penyandingan sintaks PROLOG

PL1/klausa bentuk normal	PROLOG	Deskripsi
$(\neg A_1 \vee ... \vee \neg A_m \vee B)$	$B :- A_1, ..., A_m.$	*Aturan*
$(A_1 \wedge ... \wedge A_m) \Rightarrow B$	$B :- A_1, ..., A_m.$	*Atura*
A	$A.$	*Fakta*
$(\neg A_1 \vee ... \vee \neg A_m)$	$? - A_1, ..., A_m.$	*Kueri*
$\neg(A_1 \wedge ... \wedge A_m)$	$? - A_1, ..., A_m.$	*Kueri*

Di sini $A_1, \ldots, A_m$, A, B adalah literal. Literal, seperti dalam PL1, dibangun dari simbol predikat dengan istilah sebagai argumen. Seperti yang bisa kita lihat pada tabel di atas, dalam PROLOG tidak ada negasi dalam arti logis yang ketat karena tanda literal ditentukan oleh posisinya dalam klausa.

5.1 SISTEM DAN IMPLEMENTASI PROLOG

Untuk pembaca kami merekomendasikan sistem yang sangat kuat dan tersedia secara bebas (di bawah lisensi publik GNU) GNU-PROLOG dan SWI-PROLOG. Untuk contoh berikut, SWI-PROLOG digunakan.

Sebagian besar sistem PROLOG modern bekerja dengan juru bahasa berdasarkan mesin abstrak Warren (WAM). Kode sumber PROLOG dikompilasi menjadi apa yang disebut kode WAM, yang kemudian diinterpretasikan oleh WAM. Implementasi tercepat dari WAM mengelola hingga 10 juta inferensi logis per detik (LIPS) pada PC 1 GHz.

Contoh Sederhana

Kita mulai dengan hubungan keluarga dari Contoh 3.2. Basis pengetahuan kecil KB dikodekan—tanpa fakta untuk predikat perempuan—sebagai program PROLOG bernama rel.pl pada Gambar 5.1. Program dapat dimuat dan dikompilasi dalam penerjemah PROLOG dengan perintah

? – [rel]

```
 1 child(oscar,karen,frank).
 2 child(mary,karen,frank).
 3 child(eve,anne,oscar).
 4 child(henry,anne,oscar).
 5 child(isolde,anne,oscar).
 6 child(clyde,mary,oscarb).
 7
 8 child(X,Z,Y) :- child(X,Y,Z).
 9
10 descendant(X,Y) :- child(X,Y,Z).
11 descendant(X,Y) :- child(X,U,V), descendant(U,Y).
```

Gambar 5.1 Program PROLOG dengan hubungan keluarga

Permintaan awal mengembalikan dialog

?- child(eve,oscar,anne).

Yes

dengan jawaban yang benar Ya. Bagaimana jawaban ini muncul? Untuk kueri "?- child(eve,oscar,anne)." ada enam fakta dan satu aturan dengan predikat yang sama di kepala klausanya. Sekarang penyatuan dicoba antara kueri dan masing-masing literal komplementer dalam data input sesuai urutan kemunculannya. Jika salah satu alternatif gagal, hal ini mengakibatkan backtracking ke titik percabangan terakhir, dan alternatif berikutnya diuji. Karena penyatuan gagal dengan setiap fakta, kueri disatukan dengan aturan rekursif pada baris 8. Sekarang sistem mencoba menyelesaikan anak subtujuan (eve,anne,oscar), yang berhasil dengan alternatif ketiga.

?- descendant(X,Y).

X = oscar

Y = karen

Yes

dijawab dengan solusi pertama yang ditemukan, sebagaimana adanya

?- descendant(clyde,Y).

Y = mary

Yes

Pertanyaan

?- descendant(clyde,karen).

tidak dijawab, namun. Alasan untuk ini adalah klausa pada baris 8, yang menentukan simetri dari predikat anak. Klausa ini menyebut dirinya secara rekursif tanpa kemungkinan penghentian. Masalah ini dapat diselesaikan dengan program baru berikut (fakta telah dihilangkan di sini).

```
1 descendant(X,Y) :- child(X,Y,Z).
2 descendant(X,Y) :- child(X,Z,Y).
3 descendant(X,Y) :- child(X,U,V), descendant(U,Y).
```

Tapi sekarang pertanyaannya

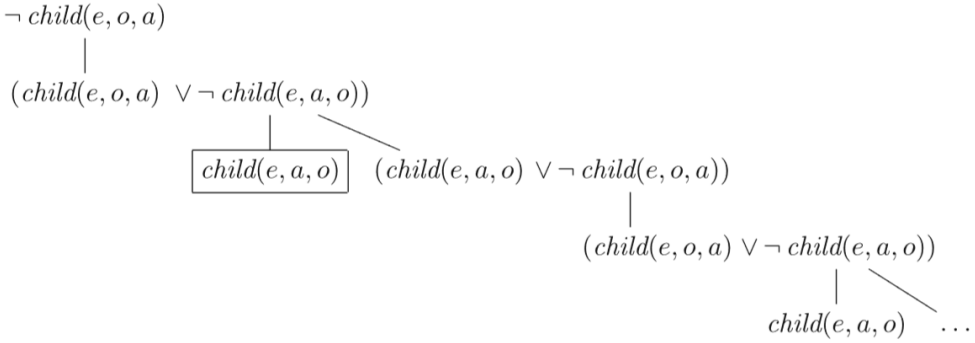
?- child(eve,oscar,anne).

tidak lagi dijawab dengan benar karena simetri anak pada dua variabel terakhir tidak lagi diberikan. Solusi untuk kedua masalah ditemukan dalam program

```
 1 child_fact(oscar,karen,franz).
 2 child_fact(mary,karen,franz).
 3 child_fact(eva,anne,oscar).
 4 child_fact(henry,anne,oscar).
 5 child_fact(isolde,anne,oscar).
 6 child_fact(clyde,mary,oscarb).
 7
 8 child(X,Z,Y) :- child_fact(X,Y,Z).
 9 child(X,Z,Y) :- child_fact(X,Z,Y).
10
11 descendant(X,Y) :- child(X,Y,Z).
12 descendant(X,Y) :- child(X,U,V), descendant(U,Y).
```

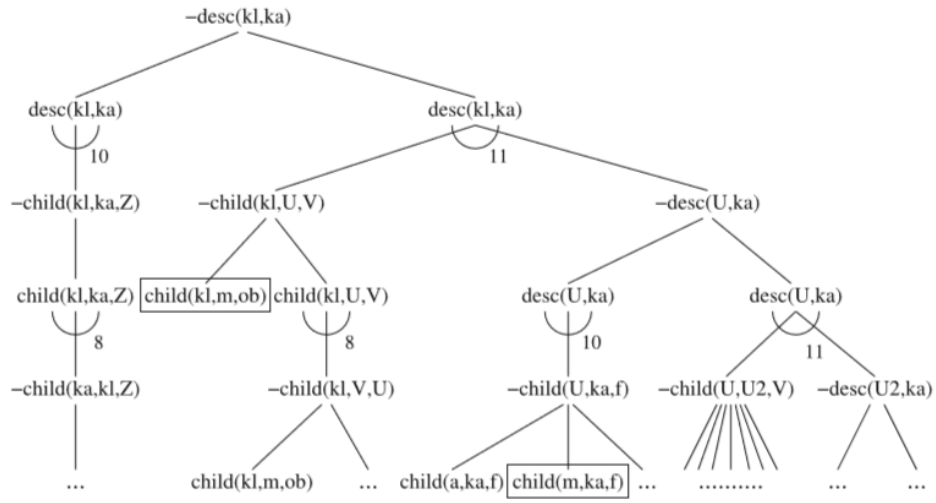
Dengan memperkenalkan predikat baru child_fact untuk fakta, anak predikat tidak lagi bersifat rekursif. Namun, program ini tidak lagi seanggun dan sesederhana varian pertama—yang benar secara logika—pada Gambar 5.1, yang mengarah ke loop tak berhingga. Pemrogram PROLOG harus, seperti dalam bahasa lain, memperhatikan pemrosesan dan menghindari pengulangan yang tidak terbatas. PROLOG hanyalah bahasa pemrograman dan bukan pembuktian teorema.

Kita harus membedakan di sini antara semantik deklaratif dan prosedural dari program PROLOG. Semantik deklaratif diberikan oleh interpretasi logis dari klausa tanduk. Semantik prosedural, sebaliknya, ditentukan oleh eksekusi program PROLOG, yang ingin kita amati secara lebih rinci sekarang. Eksekusi program dari Gambar 5.1 dengan query child(eve, oscar,anne) direpresentasikan pada Gambar 5.2 sebagai pohon pencarian.1 Eksekusi dimulai di kiri atas dengan query. Setiap tepi mewakili langkah resolusi SLD yang mungkin dengan literal terpadu yang saling melengkapi. Sementara pohon pencarian menjadi jauh tak terhingga oleh aturan rekursif, eksekusi PROLOG berakhir karena fakta terjadi sebelum aturan dalam data input.



Gambar 5.2 Pohon pencarian PROLOG untuk anak.

Dengan kueri turunan(clyde,karen), sebaliknya, eksekusi PROLOG tidak berhenti. Hal ini dapat kita lihat dengan jelas pada pohon dan-atau yang disajikan pada Gambar 5.3. Dalam representasi ini cabang, diwakili oleh gambar garis sudut diatas, memimpin dari kepala klausa ke subtujuan. Karena semua subgoals dari klausa harus diselesaikan, ini adalah dan cabang. Semua cabang lain adalah atau cabang, di mana setidaknya satu harus dapat disatukan dengan simpul induknya. Dua fakta yang diuraikan mewakili solusi untuk kueri. Akan tetapi, interpreter PROLOG tidak berhenti di sini, karena ia bekerja dengan menggunakan pencarian mendalam-pertama dengan pelacakan balik dan dengan demikian pertama-tama memilih jalur jauh tak terhingga ke paling kiri.



Gambar 5.3 Dan-atau pohon untuk desc.

5.2 KONTROL EKSEKUSI DAN ELEMEN PROSEDURAL

Seperti yang telah kita lihat dalam contoh hubungan keluarga, penting untuk mengontrol pelaksanaan PROLOG. Menghindari backtracking yang tidak perlu terutama dapat menyebabkan peningkatan efisiensi yang besar. Salah satu sarana untuk tujuan ini adalah pemotongan. Dengan menyisipkan tanda seru ke dalam klausa, kita dapat mencegah mundurnya poin ini. Dalam program berikut, predikat max(X,Y,Max) menghitung maksimum dari dua angka X dan Y.

```
1 max(X,Y,X) :- X >= Y.
2 max(X,Y,Y) :- X < Y.
```

Jika kasus pertama (klausa pertama) berlaku, maka yang kedua tidak akan tercapai. Sebaliknya, jika kasus pertama tidak berlaku, maka kondisi kasus kedua benar, artinya tidak perlu diperiksa. Misalnya, dalam kueri

?- max(3,2,Z), Z > 10.

backtracking digunakan karena Z=3, dan klausa kedua diuji untuk maks, yang pasti gagal. Jadi mundur di tempat ini tidak perlu. Kami dapat mengoptimalkan ini dengan pemotongan:

```
1 max(X,Y,X) :- X >= Y, !.
2 max(X,Y,Y).
```

Dengan demikian klausa kedua hanya dipanggil jika benar-benar diperlukan, yaitu jika klausa pertama gagal. Namun, pengoptimalan ini membuat program lebih sulit untuk dipahami.

Kemungkinan lain untuk kontrol eksekusi adalah predikat gagal bawaan, yang tidak pernah benar. Dalam contoh hubungan keluarga, kita cukup mencetak semua anak dan orang tua mereka dengan kueri

?- child_fact(X,Y,Z), write(X), write(' is a child of '), write(Y), write(' and '), write(Z), write('.'), nl, fail.

Output yang sesuai adalah
oscar adalah anak dari karen dan frank.
mary adalah anak dari karen dan frank.
eve adalah anak dari anne dan oscar.
...
Tidak.

di mana predikat nl menyebabkan jeda baris pada output. Apa yang akan menjadi output pada akhirnya tanpa menggunakan predikat fail? Dengan basis pengetahuan yang sama, kueri "?- child_fact(ulla,X,Y)." akan menghasilkan jawaban Tidak karena tidak ada fakta tentang ulla. Jawaban ini tidak benar secara logika. Secara khusus, tidak mungkin untuk membuktikan bahwa tidak ada objek dengan nama ulla. Di sini, si ahli E akan menjawab dengan benar "Tidak ada bukti yang ditemukan." Jadi jika PROLOG menjawab Tidak, ini hanya berarti bahwa pertanyaan Q tidak dapat dibuktikan. Untuk ini, bagaimanapun, ¬Q tidak harus dibuktikan. Perilaku ini disebut negasi sebagai kegagalan.

Membatasi diri kita pada klausa Horn tidak menyebabkan masalah besar dalam banyak kasus. Namun, penting untuk eksekusi prosedural menggunakan resolusi SLD. Melalui literal positif yang ditentukan secara tunggal per klausa, resolusi SLD, dan oleh karena itu eksekusi program PROLOG, memiliki titik masuk yang unik ke dalam klausa. Ini adalah satu-satunya cara yang memungkinkan untuk memiliki eksekusi program logika yang dapat direproduksi dan, oleh karena itu, semantik prosedural yang terdefinisi dengan baik.

Memang, pasti ada pernyataan masalah yang tidak dapat dijelaskan oleh klausa Horn. Contohnya adalah paradoks Russell dari Contoh 3.7, yang berisi klausa non-Horn *(shaves(barber, X) _ shaves(X, X)).*

5.3 LIST

Sebagai bahasa tingkat tinggi, PROLOG memiliki, seperti bahasa LISP, tipe data daftar umum yang nyaman. Sebuah daftar dengan elemen A, 2, 2, B, 3, 4, 5 memiliki bentuk [A,2,2,B,3,4,5]

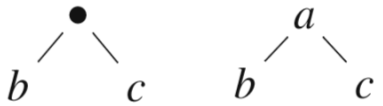
Konstruk [*Head*|*Tail*] memisahkan elemen pertama (Head) dari elemen lainnya (Tail) dari list. Dengan basis pengetahuan

list([A,2,2,B,3,4,5]).

PROLOG menampilkan dialog

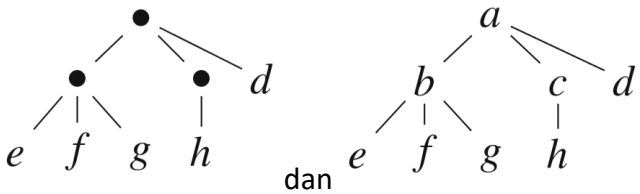
?- list([H|T]).
H=A
T = [2, 2, B, 3, 4, 5]
Ya

Dengan menggunakan daftar bersarang, kita dapat membuat struktur pohon arbitrer. Misalnya, dua pohon



dan

dapat diwakili oleh daftar[b,c]dan [a,b,c], masing-masing, dan dua pohon



dan

dengan daftar [[e,f,g],[h],d] dan [a,[b,e,f,g],[c,h],d], masing-masing. Di pohon di mana simpul dalam berisi simbol, simbol adalah *head* daftar dan simpul anak adalah *tail*.

Contoh pemrosesan daftar yang bagus dan elegan adalah definisi append(X,Y,Z) untuk menambahkan daftar Y ke daftar X. Hasilnya disimpan dalam Z. Program PROLOG yang sesuai berbunyi

```
1 append([],L,L).
2 append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

Ini adalah deskripsi logis deklaratif (rekursif) dari fakta bahwa L3 dihasilkan dari penambahan L2 ke L1. Namun, pada saat yang sama, program ini juga berfungsi saat dipanggil. Panggilan

?- append([a,b,c],[d,1,2],Z).

mengembalikan substitusi Z = [a, b, c, d, 1, 2], seperti panggilan

?- append(X,[1,2,3],[4,5,6,1,2,3]).

menghasilkan substitusi X = [4, 5, 6]. Di sini kita mengamati bahwa append bukanlah fungsi dua tempat, tetapi hubungan tiga tempat. Sebenarnya, kita juga bisa memasukkan "parameter output" Z dan menanyakan apakah bisa dibuat. Pembalikan urutan elemen daftar juga dapat dijelaskan secara elegan dan secara bersamaan diprogram oleh predikat rekursif

```
1 nrev([],[]).
2 nrev([H|T],R) :- nrev(T,RT), append(RT,[H],R).
```

yang mengurangi pembalikan daftar ke pembalikan daftar yang satu elemen lebih kecil. Memang, predikat ini sangat tidak efisien karena memanggil append. Program ini dikenal sebagai naive reverse dan sering digunakan sebagai benchmark PROLOG (lihat Latihan 5.). Segalanya menjadi lebih baik ketika seseorang melanjutkan menggunakan penyimpanan sementara, yang dikenal sebagai akumulator, sebagai berikut:

List	***Akumulator***
[a, b, c, d]	[]
[b, c, d]	[a]
[c, d]	[b, a]
[d]	[c, b, a]
[]	[d, c, b, a]

Program PROLOG yang sesuai berbunyi

```
1 accrev([],A,A).
2 accrev([H|T],A,R) :- accrev(T,[H|A],R).
```

5.4 PROGRAM MODIFIKASI SENDIRI

Program PROLOG tidak sepenuhnya dikompilasi, melainkan ditafsirkan oleh WAM. Oleh karena itu dimungkinkan untuk memodifikasi program saat runtime. Sebuah program bahkan dapat memodifikasi dirinya sendiri. Dengan perintah seperti menegaskan dan menarik kembali, fakta dan aturan dapat ditambahkan ke basis pengetahuan atau dikeluarkan darinya.

Aplikasi sederhana dari varian asserta adalah penambahan fakta turunan ke awal basis pengetahuan dengan tujuan menghindari turunan berulang yang berpotensi menghabiskan waktu (lihat Latihan 5.8). Jika dalam contoh hubungan keluarga kita mengganti dua aturan untuk predikat keturunan dengan

```
1 :- dynamic descendant/2.
2 descendant(X,Y) :- child(X,Y,Z), asserta(descendant(X,Y)).
3 descendant(X,Y) :- child(X,U,V), descendant(U,Y),
4                    asserta(descendant(X,Y)).
```

maka semua fakta turunan untuk predikat ini disimpan dalam basis pengetahuan dan dengan demikian di masa depan tidak diturunkan kembali. Pertanyaan

?- descendant(clyde, karen).

mengarah pada penambahan dua fakta

descendant(clyde, karen).
descendant(mary, karen).

Dengan memanipulasi aturan dengan menegaskan dan mencabut, bahkan program yang mengubah dirinya sendiri sepenuhnya dapat ditulis. Ide ini kemudian dikenal dengan istilah pemrograman genetik. Hal ini memungkinkan pembangunan program pembelajaran yang fleksibel secara sewenang-wenang. Namun, dalam praktiknya, ternyata, karena banyaknya kemungkinan perubahan yang tidak masuk akal, mengubah kode dengan coba-coba jarang mengarah pada peningkatan kinerja. Perubahan aturan yang sistematis, di sisi lain, membuat pemrograman jauh lebih kompleks sehingga, sejauh ini, program yang secara ekstensif memodifikasi kode mereka sendiri belum berhasil. Dalam Bab. 8 kami akan menunjukkan bagaimana pembelajaran mesin telah cukup berhasil. Namun, hanya modifikasi yang sangat terbatas dari kode program yang dilakukan di sini.

Contoh Perencanaan

Contoh 5.1 Teka-teki berikut berfungsi sebagai pernyataan masalah untuk program PROLOG yang khas.

> *Seorang petani ingin membawa kubis, kambing, dan serigala menyeberangi sungai, tetapi perahunya sangat kecil sehingga dia hanya bisa menyeberangkannya satu per satu. Petani itu memikirkannya dan kemudian berkata pada dirinya sendiri: "Jika saya pertama kali membawa serigala ke sisi lain, maka kambing itu akan memakan kubis. Jika saya mengangkut kubis terlebih dahulu, maka kambing akan dimakan oleh serigala. Apa yang harus saya lakukan?"*

Ini adalah tugas perencanaan yang dapat kita selesaikan dengan cepat dengan sedikit pemikiran. Program PROLOG yang diberikan pada Gambar 5.4 tidak dibuat secepat itu. Program ini bekerja berdasarkan kondisi bentuk Petani *(Farmer),* Serigala *(Wolf),*Kambing

(Goat), Kubis *(Cabbage),* yang menggambarkan keadaan dunia saat ini. Empat variabel dengan kemungkinan nilai *left* (kiri), *right* (kanan) memberikan lokasi objek. Rencana predikat rekursif pusat pertama-tama membuat status penerus Selanjutnya menggunakan go, menguji keamanannya dengan aman, dan mengulanginya secara rekursif hingga status awal dan tujuan sama (dalam baris program 15). Negara-negara bagian yang telah dikunjungi disimpan dalam argumen ketiga dari rencana. Dengan anggota predikat bawaan diuji apakah status Next telah dikunjungi. Jika ya, itu tidak dicoba.

Definisi dari predikat write_path untuk tugas mengeluarkan rencana yang ditemukan tidak ada di sini. Disarankan sebagai latihan untuk pembaca (Latihan 5.2). Untuk pengujian program awal, write_path(Path) literal dapat diganti dengan write(Path). Untuk kueri "?- start." kita mendapatkan jawabannya

Solusi:

Petani dan kambing dari kiri ke kanan
Petani dari kanan ke kiri
Petani dan serigala dari kiri ke kanan
Petani dan kambing dari kanan ke kiri
Petani dan kubis dari kiri ke kanan
Petani dari kanan ke kiri
Petani dan kambing dari kiri ke kanan
Ya

Untuk pemahaman yang lebih baik, kami menggambarkan definisi rencana dalam logika:

$$\forall z\ plan(z, z) \wedge \forall s\ \forall z\ \forall n\ [n[go(s, n) \wedge safe(n) \wedge plan(n, z) \Longrightarrow PLAN(S, Z)]$$

Definisi ini secara signifikan lebih ringkas daripada di PROLOG. Ada dua alasan untuk ini. Untuk satu hal, output dari rencana yang ditemukan tidak penting untuk logika. Selanjutnya, tidak perlu memeriksa apakah negara bagian berikutnya sudah dikunjungi jika perjalanan yang tidak perlu tidak mengganggu petani. Namun, jika \+ member(...) dikeluarkan dari program PROLOG, maka ada loop tak berhingga dan PROLOG mungkin tidak menemukan jadwal meskipun ada. Penyebabnya adalah strategi pencarian back chaining PROLOG, yang, menurut prinsip depth-first search selalu bekerja pada subgoals satu per satu tanpa membatasi kedalaman rekursi, dan karena itu tidak lengkap. Ini tidak akan terjadi pada pembuktian teorema dengan kalkulus lengkap.

Seperti dalam semua tugas perencanaan, keadaan dunia berubah saat tindakan dilakukan dari satu langkah ke langkah berikutnya. Hal ini menunjukkan pengiriman state sebagai variabel ke semua predikat yang bergantung pada state di dunia, seperti pada predikat safe. Transisi state terjadi pada predikat go. Pendekatan ini disebut kalkulus situasi. Kita akan menjadi akrab dengan ekstensi menarik untuk mempelajari urutan tindakan di dunia non-deterministik yang dapat diamati sebagian di Bab. 10.

Logika predikat dan sub-bahasa yang lebih sederhana untuk deskripsi dan solusi tugas perencanaan memainkan peran yang semakin penting untuk robot cerdas. Untuk menyelesaikan tugas-tugas kompleks, robot melunak bekerja pada deskripsi hubungan simbolik dunia. Aplikasi klasik untuk robot layanan adalah mengenali perintah yang dapat didengar secara linguistik dan kemudian mengubahnya menjadi rumus logis. Kemudian seorang perencana mempunyai tugas untuk menemukan suatu rencana yang pada akhirnya mencapai suatu keadaan di mana rumusan itu menjadi benar. Yang lebih menarik adalah ketika robot belajar untuk memenuhi tujuan pelatihnya dengan mengamatinya dan kemudian menggunakan perencana simbolik untuk mencapai tujuan. Satu hal yang sangat menarik tentang hal ini adalah robot mampu mengambil nilai sensor numerik seperti gambar piksel dan mengubahnya menjadi deskripsi simbolis dari situasi. Tugas ini

umumnya sangat sulit yang dikenal dalam AI sebagai simbol grounding. Pengembang robotik telah menemukan solusi menarik untuk masalah ini untuk tugas tertentu.

```
start :- action(state(left,left,left,left),
                state(right,right,right,right)).

action(Start,Goal):-
        plan(Start,Goal,[Start],Path),
        nl,write('Solution:'),nl,
        write_path(Path).
%         write_path(Path), fail.    % all solutions output

plan(Start,Goal,Visited,Path):-
      go(Start,Next),
      safe(Next),
      \+ member(Next,Visited),       % not(member(...))
      plan(Next,Goal,[Next|Visited],Path).
plan(Goal,Goal,Path,Path).

go(state(X,X,Z,K),state(Y,Y,Z,K)):-across(X,Y). % farmer, wolf
go(state(X,W,X,K),state(Y,W,Y,K)):-across(X,Y). % farmer, goat
go(state(X,W,Z,X),state(Y,W,Z,Y)):-across(X,Y). % farmer, cabbage
go(state(X,W,Z,K),state(Y,W,Z,K)):-across(X,Y). % farmer

across(left,right).
across(right,left).

safe(state(B,W,Z,K)):- across(W,Z), across(Z,K).
safe(state(B,B,B,K)).
safe(state(B,W,B,B)).
```

Gambar 5.4 Program PROLOG untuk masalah petani-serigala-kambing-kubis

5.5 *CONSTRAINT LOGIC PROGRAMMING* (CLP)

Pemrograman sistem penjadwalan, di mana banyak (terkadang kompleks) kondisi logis dan numerik harus dipenuhi, bisa sangat mahal dan sulit dengan bahasa pemrograman konvensional. Di sinilah tepatnya logika bisa berguna. Seseorang cukup menulis semua kondisi logis di PL1 dan kemudian memasukkan kueri. Biasanya pendekatan ini gagal total. Alasannya adalah masalah penguin. Fakta penguin(tweety) memastikan bahwa penguin(tweety) itu benar. Namun, tidak menutup kemungkinan bahwa gagak(tweety) juga benar. Untuk mengesampingkan hal ini dengan aksioma tambahan sangat merepotkan (Bagian 4.3).

Constraint logic programming (CLP), yang memungkinkan perumusan eksplisit kendala untuk variabel, menawarkan mekanisme yang elegan dan sangat efisien untuk memecahkan masalah ini. Penerjemah terus memantau pelaksanaan program untuk kepatuhan terhadap semua kendalanya. Pemrogram sepenuhnya dibebaskan dari tugas mengendalikan kendala, yang dalam banyak kasus dapat sangat menyederhanakan pemrograman. Hal ini diungkapkan dalam kutipan berikut oleh Eugene C. Freuder dari:

> *Pemrograman kendala merupakan salah satu pendekatan terdekat ilmu komputer yang pernah dibuat dengan Cawan Suci pemrograman: pengguna menyatakan masalahnya, komputer menyelesaikannya.*

Tanpa masuk ke teori constraint satisfaction problem (CSP), kita akan menerapkan mekanisme CLP GNU-PROLOG pada contoh berikut.

Contoh 5.2

Sekretaris Sekolah Menengah harus membuat rencana untuk mengalokasikan ruangan untuk ujian akhir. Dia memiliki informasi berikut: empat guru Mayer, Hoover, Miller dan Smith memberikan tes untuk mata pelajaran Jerman, Inggris, Matematika, dan Fisika di kamar bernomor 1, 2, 3 dan 4. Setiap guru memberikan tes untuk tepat satu subjek tepat di satu ruangan. Selain itu, dia mengetahui hal-hal berikut tentang guru dan mata pelajarannya.

1. Pak Mayer tidak pernah tes di room 4.
2. Pak Miller selalu tes bahasa Jerman.
3. Psk Smith dan Mr Miller tidak memberikan tes di kamar tetangga.
4. Ibu Hoover tes Matematika.
5. Fisika selalu diuji di ruang nomor 4.
6. Bahasa Jerman dan Inggris tidak diuji di ruang 1.

Siapa yang memberikan ujian di ruang mana?

Program GNU-PROLOG untuk memecahkan masalah ini diberikan pada Gambar 5.5. Program ini bekerja dengan variabel Mayer, Hoover, Miller, Smith serta Jerman, Inggris, Matematika, Fisika, yang masing-masing dapat mengambil nilai integer dari 1 hingga 4 sebagai nomor kamar (baris program 2 dan 5). Mayer yang mengikat = 1 dan German = 1 berarti bahwa Mr. Mayer memberikan tes bahasa Jerman di ruang 1. Baris 3 dan 6 memastikan bahwa keempat variabel tertentu memiliki nilai yang berbeda. Baris 8 memastikan bahwa semua variabel diberi nilai konkret dalam kasus solusi. Baris ini tidak mutlak diperlukan di sini. Namun, jika ada beberapa solusi, hanya interval yang akan dihasilkan. Pada baris 10 sampai 16 batasan diberikan, dan baris yang tersisa menampilkan nomor ruangan untuk semua guru dan semua mata pelajaran dalam format sederhana.

```
 1 start :-
 2   fd_domain([Mayer, Hoover, Miller, Smith],1,4),
 3   fd_all_different([Mayer, Miller, Hoover, Smith]),
 4
 5   fd_domain([German, English, Math, Physics],1,4),
 6   fd_all_different([German, English, Math, Physics]),
 7
 8   fd_labeling([Mayer, Hoover, Miller, Smith]),
 9
10   Mayer #\=4,                  % Mayer not in room 4
11   Miller #= German,            % Miller tests German
12   dist(Miller,Smith) #>= 2,    % Distance Miller/Smith >= 2
13   Hoover #= Math,              % Hoover tests mathematics
14   Physics #= 4,                % Physics in room 4
15   German #\=1,                 % German not in room 1
16   English #\=1,                % English not in room 1
17   nl,
18   write([Mayer, Hoover, Miller, Smith]), nl,
19   write([German, English, Math, Physics]), nl.
```

Gambar 5.5 Program CLP untuk masalah penjadwalan ruangan

Program dimuat ke dalam GNU-PROLOG dengan "['raumplan.pl'].", dan dengan "start." kami mendapatkan output

[3,1,2,4]
[2,3,4]

Diwakili dengan lebih mudah, kami memiliki jadwal kamar berikut:

Tabel 5.2 Jadwal kamar

No. Ruangan (room)	1	2	3	4
Guru (teacher)	Hoover	Miller	Mayer	Smith
Mata pejaran (subject)	Matematika (math)	Jerman (German)	English (inggris)	Physics (Fisika)

GNU-PROLOG memiliki, seperti kebanyakan bahasa CLP lainnya, yang disebut pemecah kendala domain terbatas, dengan variabel yang dapat diberikan rentang bilangan bulat terbatas. Ini tidak harus berupa interval seperti pada contoh. Kami juga dapat memasukkan daftar nilai. Sebagai latihan, pengguna diundang, dalam Latihan 5.9, untuk membuat program CLP, misalnya dengan GNU-PROLOG, untuk teka-teki logika yang tidak terlalu sederhana. Teka-teki ini, yang konon dibuat oleh Einstein, dapat dengan mudah dipecahkan dengan sistem CLP. Jika kami mencoba menggunakan PROLOG tanpa kendala, di sisi lain, kami dapat dengan mudah menggertakkan gigi. Siapa pun yang menemukan solusi elegan dengan PROLOG atau seorang ahli, tolong beri tahukan kepada penulisnya.

5.6 RINGKASAN

Penyatuan, daftar, pemrograman deklaratif, dan tampilan prosedur relasional, di mana argumen predikat dapat bertindak sebagai input dan output, memungkinkan pengembangan program pendek dan elegan untuk banyak masalah. Banyak program akan jauh lebih lama dan dengan demikian lebih sulit untuk dipahami jika ditulis dalam bahasa prosedural. Selain itu, fitur bahasa ini menghemat waktu programmer. Oleh karena itu PROLOG juga merupakan alat yang menarik untuk pembuatan prototipe cepat, terutama untuk aplikasi AI. Perpanjangan CLP PROLOG berguna tidak hanya untuk teka-teki logika, tetapi juga untuk banyak tugas pengoptimalan dan penjadwalan.

Sejak penemuannya pada tahun 1972, di Eropa PROLOG telah berkembang menjadi salah satu bahasa pemrograman terkemuka di Eropa dalam AI, bersama dengan bahasa prosedural. Di AS, di sisi lain, bahasa yang diciptakan secara asli LISP mendominasi pasar AI. PROLOG bukanlah pembuktian teorema. Ini disengaja, karena seorang programmer harus dapat dengan mudah dan fleksibel mengontrol pemrosesan, dan tidak akan terlalu jauh dengan pembuktian teorema. Di sisi lain, PROLOG sendiri tidak terlalu membantu untuk membuktikan teorema matematika. Namun, tentu ada pembuktian teorema menarik yang diprogram dalam PROLOG.

5.7 LATIHAN

Latihan 5.1

Coba buktikan teorema tentang persamaan elemen netral kiri dan kanan dari semi-grup dengan PROLOG. Masalah mana yang muncul? Apa penyebabnya?

Latihan 5.2

(a) Tulis predikat write_move(+State1, +State2), yang menghasilkan kalimat seperti "Petani dan serigala menyeberang dari kiri ke kanan" untuk setiap penyeberangan perahu. State1 dan State2 adalah istilah bentuk State (Petani, Serigala, Kambing, Kubis).

(b) Tulis predikat rekursif write_path(+Path), yang memanggil predikat write_move(+State1, +State2) dan menampilkan semua tindakan petani.

Latihan 5.3

(a) Sepintas variabel path dalam rencana predikat program PROLOG dari Contoh 5.1 tidak diperlukan karena tampaknya tidak berubah di mana pun. Untuk apa itu dibutuhkan?

(b) Jika kita menambahkan fail ke akhir action dalam contoh, maka semua solusi akan diberikan sebagai output. Mengapa setiap solusi sekarang dicetak dua kali? Bagaimana Anda bisa mencegah hal ini?

Latihan 5.4

(a) Tunjukkan dengan menguji bahwa pembuktian teorema E (berlawanan dengan PROLOG), berdasarkan pengetahuan dari Gambar 5.1, jawab pertanyaan "?- descendant(clyde, karen)." benar. Mengapa demikian?

(b) Bandingkan jawaban PROLOG dan E untuk pertanyaan "?- descendant (X, Y)".

Latihan 5.5

Tulis sebagai program PROLOG singkat yang mungkin menghasilkan 1024 satu

Latihan 5.6

Selidiki perilaku runtime dari predikat naif terbalik.

(a) Jalankan PROLOG dengan opsi trace dan amati panggilan rekursif dari nrev, append, dan accrev.

(b) Hitung kompleksitas waktu asimtotik append(L1,L2,L3), yaitu ketergantungan waktu berjalan pada panjang daftar untuk daftar besar. Asumsikan bahwa akses ke *head* daftar arbitrer membutuhkan waktu yang konstan.

(c) Hitung kompleksitas waktu dari nrev(L,R).

(d) Hitung kompleksitas waktu accrev(L,R).

(e) Secara eksperimental menentukan kompleksitas waktu dari predikat nrev, append, dan accrev, misalnya dengan melakukan pengukuran waktu (time(+Goal) memberikan inferensi dan waktu CPU.).

Latihan 5.7

Gunakan simbol fungsi alih-alih daftar untuk mewakili pohon yang diberikan dalam Bagian. 5.4.

Latihan 5.8

Barisan Fibonacci didefinisikan secara rekursif oleh fib(0) = 1, fib(1) = 1 dan *fib(n)=fib(n - 1) + fib(n - 2)*.

(a) Tentukan predikat PROLOG kursif fib(N,R) yang menghitung fib(N) dan mengembalikannya dalam R.

(b) Tentukan kompleksitas runtime dari predikat fib secara teoritis dan dengan pengukuran.

(c) Ubah program Anda dengan menggunakan asserta sehingga inferensi yang tidak perlu tidak lagi dilakukan.

(d) Tentukan kompleksitas runtime dari predikat yang dimodifikasi secara teoritis dan dengan pengukuran (perhatikan bahwa ini tergantung pada apakah fib sebelumnya dipanggil).

(e) Mengapa fib dengan asserta juga lebih cepat ketika dimulai pertama kali setelah PROLOG dimulai?

Latihan 5.9

Teka-teki logika tipikal berikut ini seharusnya ditulis oleh Albert Einstein. Lebih jauh, ia diduga mengklaim bahwa hanya 2% dari populasi dunia yang mampu menyelesaikannya. Pernyataan berikut diberikan.

- Ada lima rumah, masing-masing dicat dengan warna berbeda.

- Setiap rumah ditempati oleh seseorang yang berkebangsaan berbeda.
- Setiap penduduk lebih suka minuman tertentu, merokok merek tertentu, dan memiliki hewan peliharaan tertentu.
- Tak satu pun dari lima orang yang meminum hal yang sama, merokok hal yang sama, atau memiliki hewan peliharaan yang sama.
- Petunjuk:
    - Orang Semarang itu tinggal di rumah merah.
    - Orang Jepara memiliki seekor anjing.
    - Orang Demak suka minum teh.
    - Rumah hijau berada di sebelah kiri rumah putih.
    - Pemilik rumah kaca minum kopi.
    - Orang yang merokok Djarum punya burung.
    - Pria yang tinggal di rumah tengah minum susu.
    - Pemilik rumah kuning merokok Dunhill.
    - Orang Jakarta itu tinggal di rumah pertama.
    - Perokok Marlboro tinggal di sebelah orang yang memiliki kucing.
    - Pria dengan kuda itu tinggal di sebelah orang yang merokok Dunhill.
    - Perokok Sampoerna mild suka minum bir.
    - Orang Jakarta itu tinggal di sebelah rumah biru.
    - Orang Semarang merokok Esse.
    - Perokok Marlboro memiliki tetangga yang minum air.

Pertanyaan: Milik siapa ikan itu?

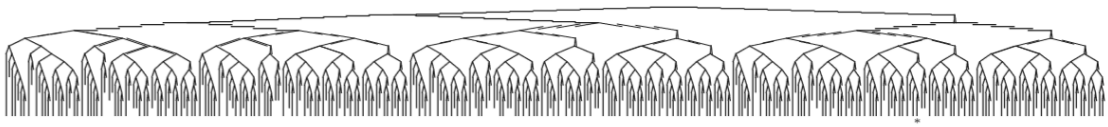
(a) Pecahkan teka-teki terlebih dahulu secara manual.

(b) Tulis program CLP (misalnya dengan GNU-PROLOG) untuk memecahkan teka-teki. Orientasikan diri Anda dengan masalah penjadwalan ruangan pada Gambar 5.5.

BAB 6
SEARCH, *GAME*, DAN *PROBLEM SOLVING*

6.1 PENDAHULUAN

Pencarian solusi dalam pohon pencarian yang sangat besar menghadirkan masalah untuk hampir semua sistem inferensi. Dari keadaan awal ada banyak kemungkinan untuk langkah inferensi pertama. Untuk setiap kemungkinan ini ada lagi banyak kemungkinan di langkah berikutnya, dan seterusnya. Bahkan dalam pembuktian rumus yang sangat sederhana dengan tiga klausa Horn, masing-masing dengan paling banyak tiga literal, pohon pencarian untuk resolusi SLD memiliki bentuk sebagai berikut:



Gambar. 6.1 bentuk dari pohon pencarian resolusi SLD

Pohon itu dipotong pada kedalaman 14 dan memiliki solusi di simpul daun yang ditandai dengan . Hal ini hanya mungkin untuk mewakili semuanya karena faktor percabangan kecil paling banyak dua dan *cut_off* pada kedalaman 14. Untuk masalah realistis, faktor percabangan dan kedalaman solusi pertama dapat menjadi jauh lebih besar.

Asumsikan faktor percabangan adalah konstanta yang sama dengan 30 dan solusi pertama adalah pada kedalaman 50. Pohon pencarian memiliki $30^{50} \approx 7{,}2 \times 10^{73}$ simpul daun. Tetapi jumlah langkah inferensi bahkan lebih besar karena tidak hanya setiap simpul daun, tetapi juga setiap simpul dalam pohon berkorespondensi dengan langkah inferensi. Oleh karena itu kita harus menjumlahkan node di semua level dan mendapatkan jumlah total node dari pohon pencarian

$$\sum_{d=0}^{50} 30^d = \frac{1 - 30^{51}}{1 - 30} = 7.4 \ x \ 10^3$$

yang tidak banyak mengubah jumlah simpul. Terbukti, hampir semua node dari pohon pencarian ini berada di level terakhir. Seperti yang akan kita lihat, ini umumnya terjadi. Tapi sekarang kembali ke pohon pencarian dengan $7{,}4 \times 1.0^{73}$ node. Asumsikan kita memiliki 10.000 komputer yang masing-masing dapat melakukan satu miliar inferensi per detik, dan bahwa kita dapat mendistribusikan pekerjaan ke semua komputer tanpa biaya. Total waktu komputasi untuk semua 7.4×10^{73} inferensi kira-kira sama dengan

$$\frac{7.4 \ x \ 10^{73} \ interfensi}{10000 \ x \ 10^9 \ Interfensi/detik} = 7.4 \ x \ 10^{60} \ detik \approx 2.3 \ x \ 10^{53} \ tahun$$

yaitu sekitar 10^{43} kali lebih lama dari usia alam semesta kita. Dengan latihan berpikir sederhana ini, kita dapat dengan cepat mengenali bahwa tidak ada peluang realistis untuk mencari ruang pencarian semacam ini sepenuhnya dengan sarana yang tersedia bagi kita di dunia ini. Selain itu, asumsi yang terkait dengan ukuran ruang pencarian benar-benar realistis. Dalam catur misalnya, ada lebih dari 30 kemungkinan gerakan untuk situasi tertentu, dan permainan yang berlangsung 50 setengah putaran relatif singkat.

Bagaimana mungkin, ada pemain catur yang bagus—dan sekarang ini juga ada komputer catur yang bagus? Bagaimana mungkin matematikawan menemukan bukti

teorema di mana ruang pencarian bahkan jauh lebih besar? Jelas bahwa kita manusia menggunakan strategi cerdas yang secara dramatis mengurangi ruang pencarian. Pemain catur berpengalaman, seperti halnya ahli matematika berpengalaman, akan, hanya dengan mengamati situasi, segera mengesampingkan banyak tindakan sebagai tidak masuk akal. Melalui pengalamannya, ia memiliki kemampuan untuk mengevaluasi berbagai tindakan untuk kegigihannya dalam mencapai tujuan. Seringkali seseorang akan merasakannya. Jika seseorang bertanya kepada ahli matematika bagaimana ia menemukan bukti, ia dapat menjawab bahwa intuisi itu datang ke dalam mimpinya. Dalam kasus-kasus sulit, banyak dokter menemukan diagnosis yang murni berdasarkan perasaan, berdasarkan semua gejala yang diketahui. Dalam masalah sehari-hari, seperti pencarian kucing yang melarikan diri pada Gambar 6.2 , intuisi memainkan peran besar. Kami akan berurusan dengan metode pencarian heuristik dan juga menjelaskan proses yang dengannya komputer dapat, seperti halnya manusia, meningkatkan strategi pencarian heuristik mereka dengan belajar.

Namun, pertama-tama, kita harus memahami bagaimana *pencarian tanpa informasi*, yaitu, secara membabi buta mencoba semua kemungkinan, bekerja. Kita mulai dengan beberapa contoh.

Gambar 6.2 Pohon pencarian yang dipangkas tebal—atau: “Di mana kucing saya?”

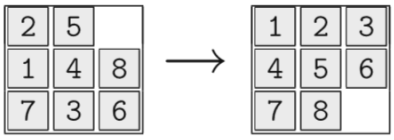
Contoh 6.1

Dengan 8-puzzle, contoh klasik untuk algoritma pencarian, berbagai algoritme dapat diilustrasikan dengan sangat jelas. Kotak dengan angka 1 sampai 8 didistribusikan dalam matriks 3 x 3 seperti yang ada pada Gambar 6.3 . Tujuannya adalah untuk mencapai urutan kotak tertentu, misalnya dalam urutan menaik dengan baris seperti yang ditunjukkan pada Gambar. 6.3. Dalam setiap langkah, sebuah persegi dapat dipindahkan ke kiri, kanan, atas, atau bawah ke dalam ruang kosong. Oleh karena itu, ruang kosong

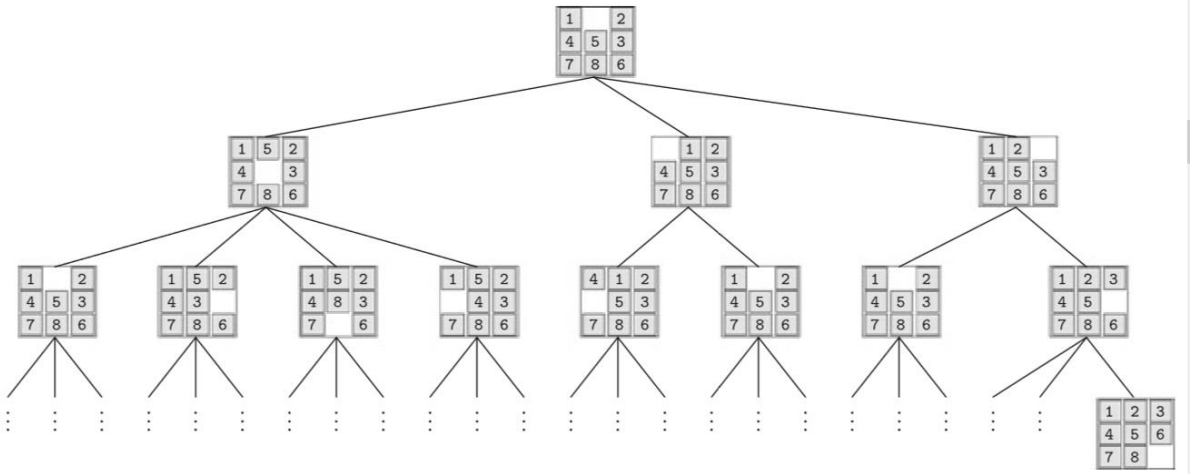
bergerak ke arah berlawanan yang sesuai. Untuk analisis ruang pencarian, akan lebih mudah untuk selalu melihat kemungkinan pergerakan bidang kosong.

Pohon pencarian untuk keadaan awal ditunjukkan pada Gambar 6.4 . Kita dapat melihat bahwa faktor percabangan bergantian antara dua, tiga, dan empat. Dirata-ratakan lebih dari dua tingkat pada satu waktu, kami memperoleh faktor percabangan rata-rata[14] dariV8 ≈ 2.83. Kami melihat bahwa setiap keadaan diulang beberapa kali dua tingkat lebih dalam karena dalam pencarian sederhana tanpa informasi, setiap tindakan dapat dibalik pada langkah berikutnya. Jika kita tidak mengizinkan siklus dengan panjang 2, maka untuk keadaan awal yang sama kita memperoleh pohon pencarian yang ditunjukkan pada Gambar 6.5. Faktor percabangan rata-rata berkurang sekitar 1 dan menjadi 1.8[15].

Sebelum kita mulai menjelaskan algoritma pencarian, beberapa istilah baru diperlukan. Kami berurusan dengan masalah pencarian diskrit di sini. Berada di state s, aksi a1 mengarah ke state baru s. Jadi s' = a_1(s). Tindakan yang berbeda dapat menyebabkan keadaan *s"*, dengan kata lain: s = a_2(s). Aplikasi rekursif dari semua tindakan yang mungkin ke semua status, dimulai dengan status awal, menghasilkan pohon pencarian.



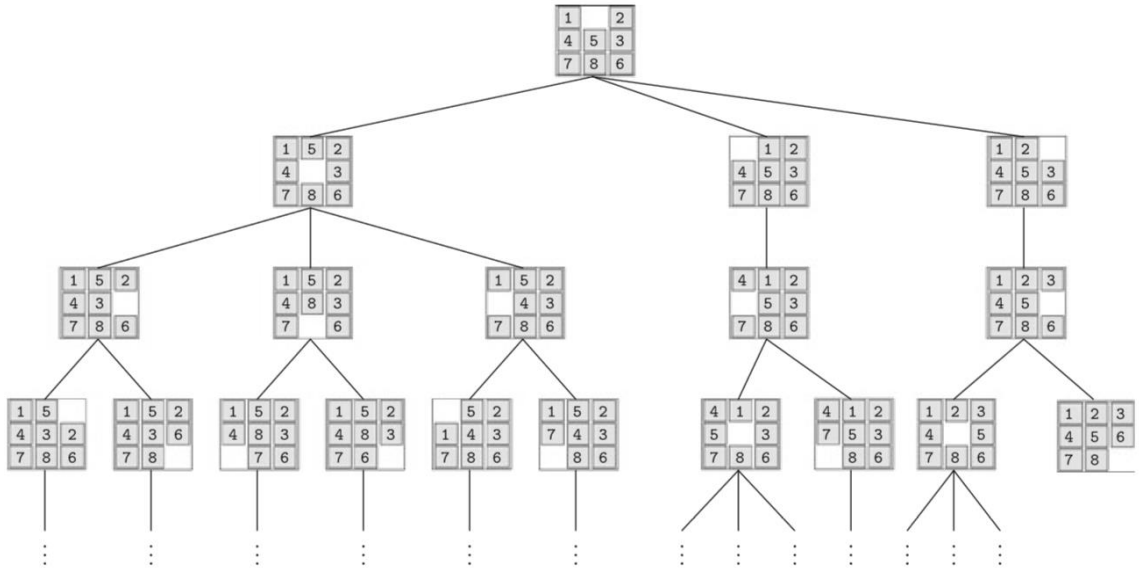
Gambar. 6.3 Kemungkinan status awal dan tujuan dari 8-puzzle



Gambar 6.4 Cari pohon untuk 8-puzzle. Kanan bawah State tujuan di kedalaman 3 diwakili. Untuk menghemat ruang, node lain pada level ini telah dihilangkan

[14] Faktor percabangan rata-rata pohon adalah faktor percabangan yang dimiliki pohon dengan faktor percabangan konstan, kedalaman yang sama, dan jumlah simpul daun yang sama.

[15] Untuk teka-teki 8, faktor percabangan rata-rata tergantung pada keadaan awal (lihat Latihan 6.2).



Gambar 6.5 Cari pohon untuk 8 teka-teki tanpa siklus dengan panjang 2

Definisi 6.1
Masalah pencarian didefinisikan oleh nilai-nilai berikut
State: Deskripsi keadaan dunia di mana agen pencarian menemukan dirinya sendiri.
***Starting State*:** Status awal di mana agen pencarian dimulai. Status tujuan: Jika agen mencapai status tujuan, maka agen akan berhenti dan mengeluarkan solusi (jika diinginkan).
Actions: Semua agen mengizinkan tindakan.
Solution: Jalur di pohon pencarian dari status awal ke status tujuan.
Cost Function: Menetapkan nilai biaya untuk setiap tindakan. Diperlukan untuk menemukan solusi biaya-optimal.
State space: Himpunan semua state.
Diterapkan pada teka-teki 8, kita dapatkan

State : 3 x 3 matriks S dengan nilai 1, 2, 3, 4, 5, 6, 7, 8 (masing-masing sekali) dan satu kotak kosong.

Starting State : Status arbitrer. Status tujuan: Status arbitrer, mis. keadaan yang diberikan ke kanan pada Gambar 6.3.

Action : Pergerakan kotak kosong S_{ij} ke kiri (jika $j \neq 1$), kanan (jika $j \neq 3$), atas (jika $i \neq 1$), bawah (jika $i \neq 3$).

Costing Function : Fungsi konstan 1, karena semua tindakan memiliki biaya yang sama. *State Space*: Ruang keadaan merosot dalam domain yang tidak dapat dijangkau bersama

(Latihan 6.4). Jadi ada masalah 8 - teka-teki yang tidak terpecahkan. Untuk analisis algoritma pencarian, istilah berikut diperlukan:

Definisi 6.2

- Banyaknya keadaan penerus dari suatu keadaan s disebut faktor percabangan b(s), atau b jika faktor percabangan konstan.
- Faktor percabangan efektif dari pohon dengan kedalaman d dengan n total simpul didefinisikan sebagai faktor percabangan yang dimiliki pohon dengan faktor percabangan konstan, kedalaman sama, dan n sama (lihat Latihan 6.3).

- Sebuah algoritma pencarian disebut lengkap jika menemukan solusi untuk setiap masalah yang dapat dipecahkan. Jika algoritma pencarian lengkap berakhir tanpa menemukan solusi, maka masalahnya tidak dapat dipecahkan.

Untuk kedalaman tertentu d dan jumlah simpul n, faktor percabangan efektif dapat dihitung dengan menyelesaikan persamaan

Persamaan 6.1

$$n = \frac{b^{d+1} - 1}{b - 1}$$

untuk b karena pohon dengan faktor percabangan konstan dan kedalaman d memiliki jumlah simpul.

Persamaan 6.2

$$n = \sum_{i=0}^{d} b^i = \frac{b^{d+1} - 1}{b - 1}$$

Untuk aplikasi praktis dari algoritma pencarian untuk pohon pencarian hingga, tingkat terakhir sangat penting karena:

Teorema 6.1

Untuk pohon pencarian hingga yang sangat bercabang dengan faktor percabangan konstan yang besar, hampir semua node berada pada level terakhir.

Pembuktian sederhana dari teorema ini direkomendasikan kepada pembaca sebagai latihan (Latihan 6.1).

Contoh 6.2

Kita diberikan peta, seperti yang ditunjukkan pada Gambar 6.5, sebagai grafik dengan kota sebagai simpul dan hubungan jalan raya antara kota sebagai tepi tertimbang dengan jarak. Kami mencari rute optimal dari kota A ke kota B. Deskripsi skema yang sesuai berbunyi

State: A city as the current location of traveler.

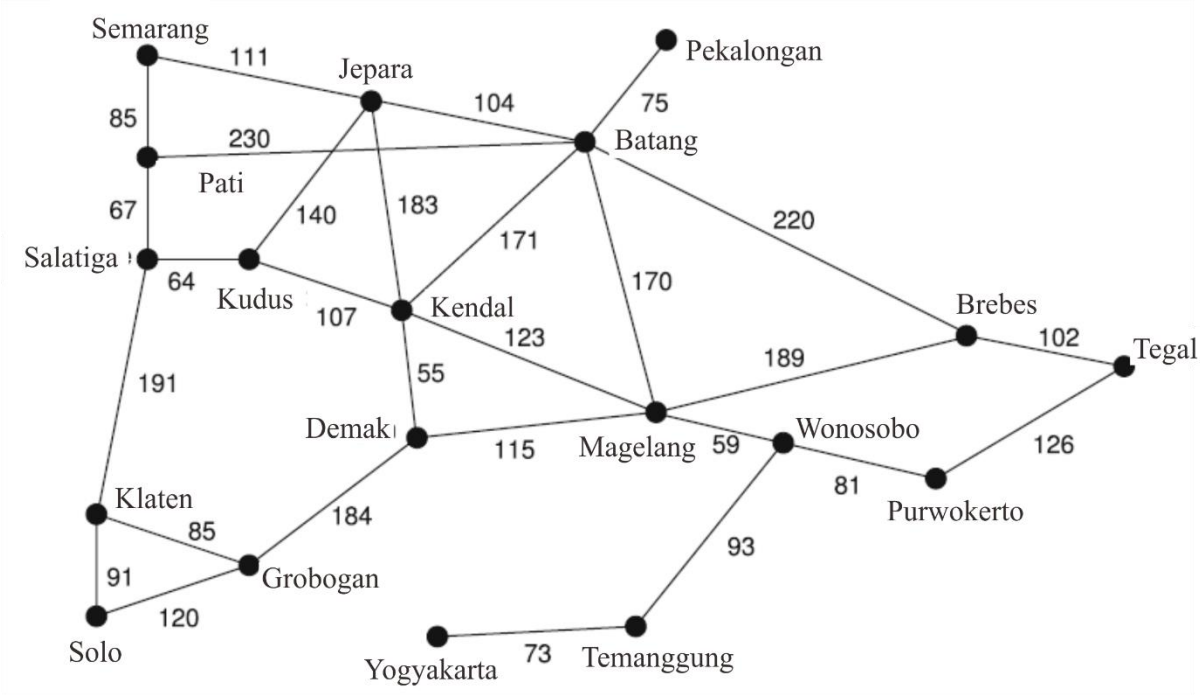
Tujuan Awal : Kota A.

Status tujuan: Kota B.

Action: Perjalanan dari kota saat ini ke kota tetangga.

Cost Function: Jarak antar kota. Setiap tindakan sesuai dengan tepi dalam grafik dengan jarak sebagai bobot.

State space: Semua kota, yaitu node dari grafik. Untuk menemukan rute dengan panjang minimal, biaya harus diperhitungkan karena tidak konstan seperti pada teka-teki 8.



Gambar 6.6 Grafik Indonesia (Jawa tenganh) sebagai contoh tugas pencarian dengan fungsi biaya

Definisi 6.3
Sebuah algoritma pencarian disebut optimal jika, jika ada solusi, selalu menemukan solusi dengan biaya terendah.
Masalah 8 teka-teki adalah *deterministik*, yang berarti bahwa setiap tindakan mengarah dari suatu keadaan ke keadaan penerus yang unik. Lebih jauh lagi dapat diamati, yaitu, agen selalu mengetahui statusnya. Dalam perencanaan rute dalam aplikasi nyata kedua karakteristik tidak selalu diberikan. Tindakan "Berkendara dari Munich ke Kendal" dapat—misalnya karena kecelakaan—mengakibatkan State penerus "Munich". Bisa juga terjadi pengelana tidak tahu lagi dimana karena tersesat. Kami ingin mengabaikan komplikasi semacam ini pada awalnya. Oleh karena itu dalam bab ini kita hanya akan melihat masalah yang bersifat deterministik dan dapat diamati.
Masalah seperti teka-teki 8, yang deterministik dan dapat diamati, membuat perencanaan tindakan relatif sederhana karena, karena memiliki model abstrak, dimungkinkan untuk menemukan urutan tindakan untuk solusi masalah tanpa benar-benar melakukan tindakan di dunia nyata. . Dalam kasus teka-teki 8, tidak perlu benar-benar memindahkan kotak di dunia nyata untuk menemukan solusinya. Kita dapat menemukan solusi optimal dengan apa yang disebut algoritma ofine. Seseorang menghadapi banyak tantangan berbeda ketika, misalnya, membuat robot yang seharusnya bermain sepak bola. Di sini tidak akan pernah ada model abstrak yang tepat dari tindakan. Misalnya, robot yang menendang bola ke arah tertentu tidak dapat memprediksi dengan pasti kemana bola akan bergerak karena antara lain tidak mengetahui apakah lawan akan menangkap atau membelokkan bola. Di sini algoritme online kemudian diperlukan, yang membuat keputusan berdasarkan sinyal sensor di setiap situasi. Pembelajaran penguatan, dijelaskan dalam Bab. 10, bekerja menuju optimalisasi keputusan ini berdasarkan pengalaman.

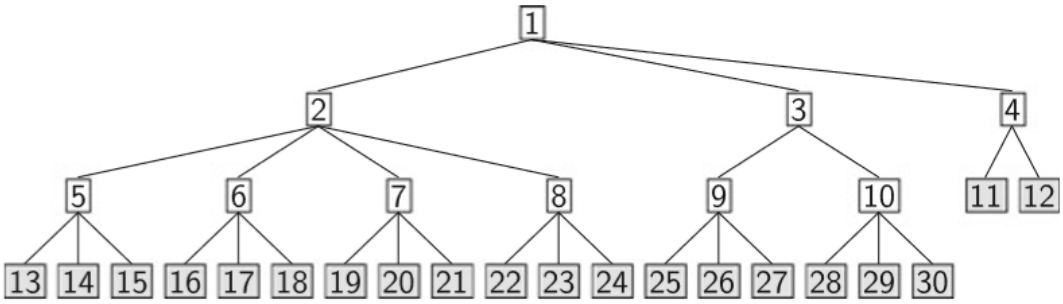
6.2 PENCARIAN LUAS-PERTAMA

Dalam pencarian luas pertama, pohon pencarian dieksplorasi dari atas ke bawah sesuai dengan algoritma yang diberikan pada Gambar 6.6 sampai solusi ditemukan. Pertama, setiap node dalam daftar node diuji apakah itu node tujuan, dan jika berhasil, program dihentikan. Jika tidak, semua penerus node dihasilkan. Pencarian kemudian dilanjutkan secara rekursif pada daftar semua node yang baru dibuat. Semuanya berulang sampai tidak ada lagi penerus yang dihasilkan.

Algoritma ini bersifat umum. Artinya, ini berfungsi untuk aplikasi arbitrer jika dua fungsi khusus aplikasi "*GoalReached*" dan "*Successors*" disediakan. "*GoalReached*" menghitung apakah argumen adalah simpul tujuan, dan "Penerus" menghitung daftar semua simpul penerus dari argumennya. Gambar 6.7 menunjukkan snapshot dari pencarian luas-pertama.

```
BREADTHFIRSTSEARCH(NodeList, Goal)
NewNodes = ∅
For all Node ∈ NodeList
    If GoalReached(Node, Goal)
        Return("Solution found", Node)
    NewNodes = Append(NewNodes, Successors(Node))
If NewNodes ≠ ∅
    Return(BREADTH-FIRST-SEARCH(NewNodes, Goal))
Else
    Return("No solution")
```

Gambar 6.7 Algoritma untuk pencarian luas pertama



Gambar 6.8 Pencarian *Breadth-first* selama perluasan node tingkat ketiga.

Node diberi nomor sesuai dengan urutan pembuatannya. Penerus node 11 dan 12 belum dihasilkan Analisis karena pencarian luas-pertama benar-benar mencari melalui setiap kedalaman dan mencapai setiap kedalaman dalam waktu yang terbatas, itu lengkap jika faktor percabangan b berhingga. Solusi optimal (yaitu, terpendek) ditemukan jika biaya semua tindakan adalah sama (lihat Latihan 6.7). Waktu komputasi dan ruang memori tumbuh secara eksponensial dengan kedalaman pohon. Untuk pohon dengan faktor percabangan konstan b dan kedalaman d, total waktu komputasi diberikan oleh

$$c \cdot \sum_{i=0}^{d} b^i = \frac{b^{d+1} - 1}{b - 1} = O(b^d)$$

Meskipun hanya level terakhir yang disimpan dalam memori, kebutuhan ruang memori juga $O(b^d)$.

Dengan kecepatan komputer saat ini, yang dapat menghasilkan miliaran node dalam hitungan menit, memori utama dengan cepat terisi dan pencarian berakhir. Masalah solusi terpendek tidak selalu ditemukan dapat diselesaikan dengan apa yang disebut Uniform Cost Search, di mana node dengan biaya terendah dari daftar node (yang diurutkan berdasarkan biaya) selalu diperluas, dan yang baru node diurutkan. Jadi kami menemukan solusi optimal. Masalah memori belum terpecahkan. Sebuah solusi untuk masalah ini disediakan oleh pencarian mendalam-pertama.

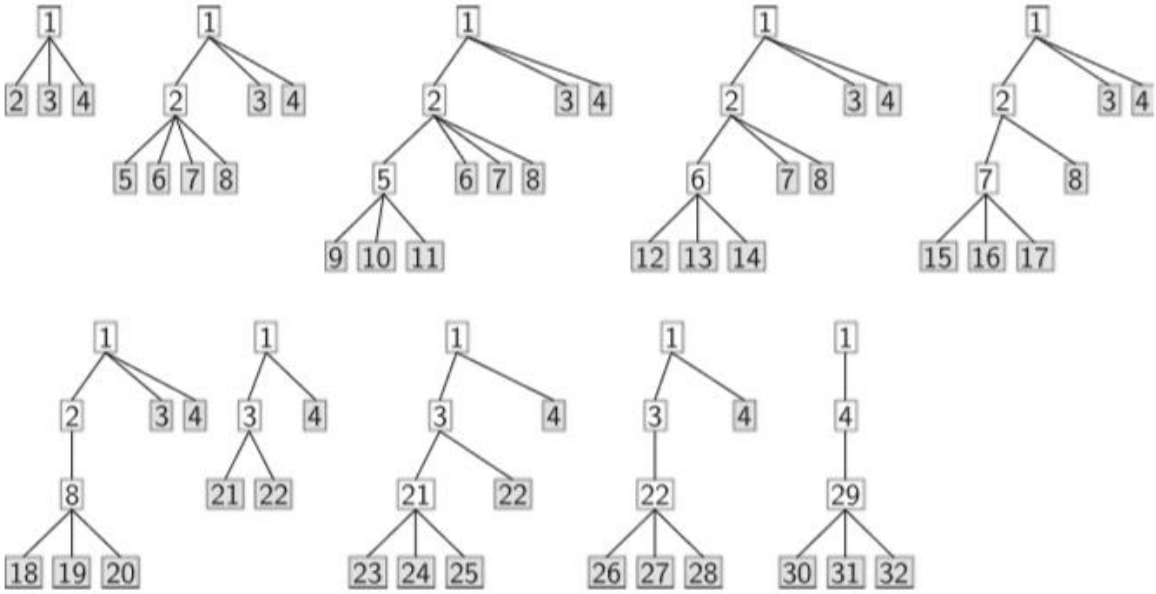
6.3 ANALISIS PENCARIAN

Depth-first membutuhkan memori yang jauh lebih sedikit daripada pencarian breadth-first karena paling banyak node menyimpan data setiap kedalaman. Oleh karena itu, dibutuhkan paling banyak sel memori.

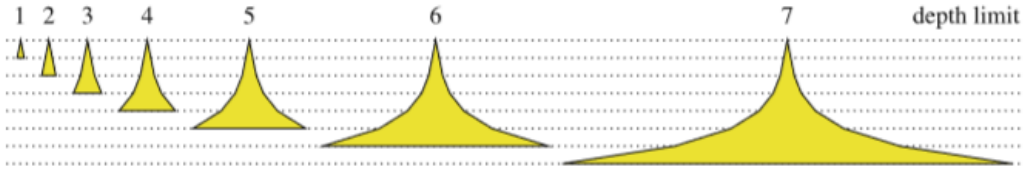
Namun, pencarian kedalaman-pertama tidak lengkap untuk pohon tak hingga karena pencarian mendalam-pertama berjalan ke loop tak terbatas ketika tidak ada solusi di cabang paling kiri. Oleh karena itu pertanyaan untuk menemukan solusi optimal sudah usang. Karena loop tak terbatas, tidak ada batasan waktu komputasi yang dapat diberikan . Dalam kasus pohon pencarian yang sangat dalam dengan kedalaman d, total sekitar bd node dihasilkan. Jadi waktu komputasi tumbuh, hanya dalam pencarian pertama yang luas, secara eksponensial dengan kedalaman. Kita dapat membuat pohon pencarian terhingga dengan menetapkan batas kedalaman. Sekarang jika tidak ada solusi yang ditemukan di pohon pencarian yang dipangkas, tetap saja ada solusi di luar batas.

```
DEPTHFIRSTSEARCH(Node, Goal)
If GoalReached(Node, Goal) Return("Solution found")
NewNodes = Successors(Node)
While NewNodes ≠ Ø
    Result = DEPTH-FIRST-SEARCH(First(NewNodes), Goal)
    If Result = "Solution found" Return("Solution found")
    NewNodes = Rest(NewNodes)
Return("No solution")
```

Gambar 6.9 Algoritma untuk pencarian depth-first. Fungsi "Pertama" mengembalikan elemen pertama dari daftar, dan "Istirahat" sisa daftar



Gambar 6.10 Eksekusi pencarian depth-first. Semua node pada kedalaman tiga tidak berhasil dan menyebabkan backtracking. Node diberi nomor sesuai urutan yang dihasilkan



Gambar 6.10 Representasi skematis dari pengembangan pohon pencarian dalam pendalaman iteratif dengan batas dari 1 hingga 7.

Lebar pohon sesuai dengan faktor percabangan 2. Dengan demikian pencarian menjadi tidak lengkap. Namun, ada ide yang jelas untuk membuat pencarian menjadi lengkap.

6.4 PENDALAMAN BERULANG

Kami memulai pencarian kedalaman-pertama dengan batas kedalaman 1. Jika tidak ada solusi yang ditemukan, kami menaikkan batas tersebut dengan 1 dan mulai mencari dari awal, dan seterusnya, seperti yang ditunjukkan pada Gambar 6.10. Peningkatan berulang dari batas kedalaman ini disebut pendalaman berulang. Kita harus menambah program pencarian kedalaman-pertama yang diberikan pada Gambar. 6.8 dengan dua parameter tambahan "Kedalaman" dan "Batas". "Kedalaman" dinaikkan satu pada panggilan rekursif, dan garis utama dari loop while digantikan *oleh "While NewNodes ≠ Ø And Depth < Limit".* Algoritma yang dimodifikasi ditunjukkan pada Gambar 6.11 .

6.5 ANALISIS

Persyaratan memori sama dengan pencarian mendalam-pertama. Orang bisa berargumen bahwa berulang kali memulai kembali pencarian *depth-first* pada kedalaman nol menyebabkan banyak pekerjaan yang berlebihan. Untuk faktor percabangan besar hal ini tidak terjadi. Kami sekarang menunjukkan bahwa jumlah jumlah node dari semua

kedalaman sampai dengan yang sebelum d_{max} - 1 terakhir di semua pohon yang dicari jauh lebih kecil daripada jumlah node di pohon terakhir yang dicari.

```
ITERATIVEDEEPENING(Node, Goal)
DepthLimit = 0
Repeat
    Result = DEPTHFIRSTSEARCH-B(Node, Goal, 0, DepthLimit)
    DepthLimit = DepthLimit + 1
Until Result = "Solution found"
```

```
DEPTHFIRSTSEARCH-B(Node, Goal, Depth, Limit)
If GoalReached(Node, Goal) Return("Solution found")
NewNodes = Successors(Node)
While NewNodes ≠ ∅ And Depth < Limit
    Result =
        DEPTHFIRSTSEARCH-B(First(NewNodes), Goal, Depth + 1, Limit)
    If Result = "Solution found" Return("Solution found")
    NewNodes = Rest(NewNodes)
Return("No solution")
```

Gambar 6.11 Algoritme untuk pendalaman berulang, yang menyebut pencarian depth-first yang sedikit dimodifikasi dengan batas kedalaman (TIEFENSUCHE-B)

Biarkan Nb(d) menjadi jumlah node dari pohon pencarian dengan faktor percabangan b dan kedalaman d dan dmax menjadi kedalaman terakhir yang dicari. Pohon terakhir yang dicari berisi

$$N_b\,(d_{msx}) \;=\; \sum_{i\,=\,0}^{d_{max}} b^i \;=\; \frac{b^{d_{max}} \;-\; 1}{b \;-\; 1}$$

node. Semua pohon yang dicari sebelumnya bersama-sama memiliki

$$\sum_{d=\,0}^{d_{max}\,-\,1} \;=\; \sum_{i=\,0}^{d_{max}\,-\,1} b^i \;=\; \frac{b^{d_{max}\,+1} \;-\; 1}{b \;-\; 1} \;=\; \frac{1}{b \;-\; 1}\left(\left(\sum_{d=1}^{d_{max}} b^{d+1}\right) - d_{max} \;+\; 1\right)$$

$$= \frac{1}{b \;-\; 1}\left(\left(\sum_{d=2}^{d_{max}} b^{d}\right) - d_{max} \;+\; 1\right)$$

$$= \frac{1}{b \;-\; 1}\left(\frac{b^{d_{max}} \;-\; 1}{b \;-\; 1} - 1 \;-\; b \;-\; d_{max} \;+\; 1\right)$$

$$\approx \frac{1}{b \;-\; 1} = \left(\frac{b^{d_{max}} \;-\; 1}{b \;-\; 1}\right) = \frac{1}{b \;-\; 1} \;-\; 1 \;-\; N_b(d_{max})$$

node. Untuk b > 2 ini lebih kecil dari jumlah $N_b(d_{max})$ node di pohon terakhir. Untuk b=20 pohon pertama d_{max} – 1 pertama yang bersama-sama hanya berisi sekitar 1/b-1 = 1/9 dari jumlah simpul pada pohon terakhir. Waktu komputasi untuk semua iterasi selain yang

terakhir dapat diabaikan. Sama seperti pencarian luas pertama, metode ini selesai, dan dengan biaya konstan untuk semua tindakan, metode ini menemukan solusi terpendek.

6.6 PERBANDINGAN

Algoritma pencarian yang dijelaskan telah diletakkan berdampingan dalam Tabel 6.1. Kita dapat melihat dengan jelas bahwa pendalaman berulang adalah pemenang tes ini karena mendapat nilai terbaik di semua kategori. Faktanya, dari keempat algoritma yang disajikan, itu adalah satu-satunya yang dapat digunakan secara praktis.

Kami memang memiliki pemenang untuk tes ini, meskipun untuk aplikasi realistis biasanya tidak berhasil. Bahkan untuk 15 teka-teki, kakak laki-laki dari 8 teka-teki (lihat Latihan 6.4), ada sekitar 2×10^{13} keadaan yang berbeda. Untuk sistem inferensi non-sepele ruang keadaan jauh lebih besar. Seperti yang ditunjukkan sebelumnya, semua kekuatan komputasi di dunia tidak akan banyak membantu. Sebaliknya, yang dibutuhkan adalah pencarian cerdas yang hanya mengeksplorasi sebagian kecil dari ruang pencarian dan menemukan solusi di sana.

6.7 PEMERIKSAAN SIKLUS

Node dapat dikunjungi berulang kali selama pencarian. Dalam teka-teki 8, misalnya, setiap gerakan dapat segera dibatalkan, yang mengarah ke siklus panjang dua yang tidak perlu. Siklus tersebut dapat dicegah dengan merekam dalam setiap node semua pendahulunya dan, ketika memperluas sebuah node, membandingkan node penerus yang baru dibuat dengan node pendahulunya. Semua duplikat yang ditemukan dapat dihapus dari daftar node penerus. Pemeriksaan sederhana ini hanya memerlukan faktor konstan kecil dari ruang memori tambahan dan meningkatkan waktu komputasi konstan c dengan konstanta tambahan δ untuk pemeriksaan itu sendiri untuk total c + δ. Overhead untuk pemeriksaan siklus ini (semoga) diimbangi dengan pengurangan biaya pencarian. Pengurangan tersebut tentu saja tergantung pada aplikasi tertentu dan oleh karena itu tidak dapat diberikan secara umum.

Tabel 6.1 Perbandingan algoritma uninformed search.

	Breadth-first search	Uniform cost search	Depth-first search	Iterative deepening
Completeness	Yes	Yes	No	Yes
Optimal solution	Yes (*)	Yes	No	Yes (*)
Computation time	b^d	b^d	∞ or b^d	b^d
Memory use	$\boldsymbol{b^d}$	$\boldsymbol{b^d}$	$\boldsymbol{bd}$	$\boldsymbol{bd}$

(*) berarti pernyataan tersebut hanya benar jika diberikan biaya tindakan yang konstan. ds adalah kedalaman maksimal untuk pohon pencarian hingga untuk 8-puzzle kita mendapatkan hasil sebagai berikut. Jika, misalnya, selama pencarian pertama dengan faktor percabangan efektif b pada pohon hingga kedalaman d, waktu komputasi tanpa pemeriksaan siklus adalah c x b^d, waktu yang diperlukan dengan pemeriksaan siklus menjadi

$$(c + \delta) \cdot (b - 1)^d$$

Pemeriksaan demikian secara praktis selalu menghasilkan keuntungan yang besar karena pengurangan faktor percabangan satu per satu memiliki efek yang tumbuh secara eksponensial seiring bertambahnya kedalaman, sedangkan waktu komputasi tambahan d hanya sedikit meningkatkan faktor konstanta.

Sekarang muncul pertanyaan tentang bagaimana pemeriksaan pada siklus dengan panjang sembarang akan mempengaruhi kinerja pencarian. Daftar pendahulu yang jatuh harus sekarang disimpan untuk setiap node, yang dapat dilakukan dengan sangat efisien (lihat Latihan 6.8). Selama pencarian, setiap node yang baru dibuat sekarang harus dibandingkan dengan semua pendahulunya. Waktu komputasi pencarian depth-first atau pencarian pertama luas diberikan oleh

$$c_1 \cdot \sum_{i=0}^{d} b^i + c_2 \cdot \sum_{i=0}^{d} i \cdot b^i$$

Di sini, suku pertama adalah biaya pembangkitan node yang sudah diketahui, dan suku kedua adalah biaya pemeriksaan siklus. Kita dapat menunjukkan bahwa untuk nilai b dan d yang besar,

$$\sum_{i=0}^{d} i \cdot b^i \approx b^d$$

Kompleksitas pencarian dengan pemeriksaan siklus penuh hanya meningkat satu faktor lebih cepat daripada pencarian tanpa pemeriksaan siklus. Pohon pencarian yang tidak terlalu dalam, kompleksitas tambahan ini tidak penting. Untuk tugas pencarian dengan pohon yang sangat dalam dan bercabang lemah, mungkin lebih baik menggunakan tabel [CLR90] untuk menyimpan daftar pendahulunya. Pencarian dalam tabel dapat dilakukan dalam waktu yang konstan sehingga waktu komputasi dari algoritma pencarian hanya bertambah dengan faktor konstan yang kecil. Singkatnya, kita dapat menyimpulkan bahwa pemeriksaan siklus menyiratkan hampir tidak ada overhead tambahan dan oleh karena itu bermanfaat untuk aplikasi dengan node yang muncul berulang kali.

6.8 PENCARIAN HEURISTIK

Heuristik adalah strategi pemecahan masalah yang dalam banyak kasus menemukan solusi lebih cepat daripada pencarian tanpa informasi. Namun, ini tidak dijamin. Pencarian heuristik dapat membutuhkan lebih banyak waktu dan bahkan dapat mengakibatkan solusi tidak ditemukan.

Kita manusia berhasil menggunakan proses heuristik untuk segala macam hal. Saat membeli sayuran di supermarket, misalnya, kami menilai berbagai pilihan untuk satu pon stroberi hanya menggunakan beberapa kriteria sederhana seperti harga, penampilan, sumber produksi, dan kepercayaan pada penjual, dan kemudian kami memutuskan opsi terbaik dengan perasaan perut. Secara teoritis mungkin lebih baik untuk melakukan analisis kimia dasar stroberi sebelum memutuskan apakah akan membelinya. Misalnya, stroberi mungkin diracuni. Jika itu masalahnya, analisisnya akan sepadan dengan masalahnya. Namun, kami tidak melakukan analisis semacam ini karena ada kemungkinan yang sangat tinggi bahwa seleksi heuristik kami akan berhasil dan akan dengan cepat membawa kami ke tujuan kami untuk makan stroberi yang lezat.

Keputusan heuristik terkait erat dengan kebutuhan untuk membuat keputusan waktu nyata dengan sumber daya yang terbatas. Dalam prakteknya solusi yang baik ditemukan dengan cepat lebih disukai daripada solusi yang optimal, tetapi sangat mahal untuk diturunkan.

Fungsi evaluasi heuristik f(s) untuk keadaan digunakan untuk memodelkan heuristik secara matematis. Tujuannya adalah untuk menemukan, dengan sedikit usaha, solusi untuk masalah pencarian yang dinyatakan dengan biaya total yang minimal. Harap dicatat bahwa ada perbedaan tipis antara upaya untuk menemukan solusi dan total biaya solusi ini.

Misalnya, Google Maps mungkin memerlukan upaya setengah detik untuk menemukan rute dari Balai Kota di San Francisco ke Tuolumne Meadows di Taman Nasional Yosemite, tetapi perjalanan dari San Francisco ke Tuolumne Meadows dengan mobil mungkin memakan waktu empat jam dan sedikit uang. untuk bensin dll (biaya total).

Selanjutnya kita akan memodifikasi algoritma pencarian breadth-first dengan menambahkan fungsi evaluasi ke dalamnya. Node yang saat ini terbuka tidak lagi diperluas dari kiri ke kanan per baris, melainkan menurut peringkat heuristiknya. Dari himpunan node terbuka, node dengan rating minimal selalu diekspansi terlebih dahulu. Ini dicapai dengan segera mengevaluasi node saat mereka diperluas dan menyortirnya ke dalam daftar node terbuka. Daftar tersebut kemudian dapat berisi node dari kedalaman yang berbeda di pohon.

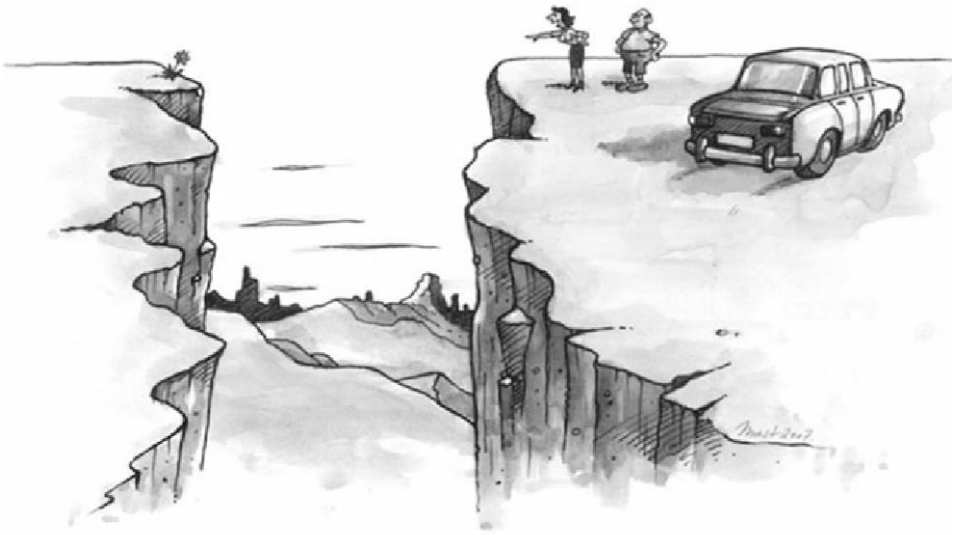
Karena evaluasi heuristik status sangat penting untuk pencarian, kami akan membedakan mulai sekarang antara status dan node terkaitnya. Node berisi status dan informasi lebih lanjut yang relevan dengan pencarian, seperti kedalamannya di pohon pencarian dan peringkat heuristik status. Akibatnya, fungsi "Penerus", yang menghasilkan penerus (anak) dari sebuah node, juga harus segera menghitung untuk node penerus ini peringkat heuristik mereka sebagai komponen dari setiap node. Kami mendefinisikan algoritma pencarian umum HEURISTICSEARCH pada Gambar. 6.12.

Daftar node diinisialisasi dengan node awal. Kemudian, dalam loop, simpul pertama dari daftar dihapus dan diuji apakah itu simpul solusi. Jika tidak, itu akan diperluas dengan fungsi "Penerus" dan penerusnya ditambahkan ke daftar dengan fungsi "SortIn". “SortIn(X,Y)” menyisipkan elemen dari daftar X yang tidak disortir ke dalam daftar Y yang diurutkan secara menaik. Peringkat heuristik digunakan sebagai kunci pengurutan. Jadi dijamin bahwa simpul terbaik (yaitu, yang memiliki nilai heuristik terendah) selalu berada di awal daftar.[16]

```
HEURISTICSEARCH(Start, Goal)
NodeList = [Start]
While True
    If NodeList = ∅ Return("No solution")
    Node = First(NodeList)
    NodeList = Rest(NodeList)
    If GoalReached(Node, Goal) Return("Solution found", Node)
    NodeList = SortIn(Successors(Node),NodeList)
```

Gambar 6.12 Algoritma untuk pencarian heuristik

[16] Saat menyortir node baru dari daftar node, mungkin bermanfaat untuk memeriksa apakah node sudah tersedia dan, jika demikian, menghapus duplikatnya.



Gambar 6.13 Laki-laki: “Sayang, pikirkan biaya bahan bakar! Aku akan memetik satu untukmu di tempat lain.” Perempuan : "Tidak, saya ingin yang itu di sana!"

Pencarian *Depth-First* dan *Breadth-First* juga merupakan kasus khusus dari fungsi HEURISTICSEARCH. Kita dapat dengan mudah menghasilkannya dengan memasukkan fungsi evaluasi yang sesuai (Latihan 6.11).

Heuristik terbaik adalah fungsi yang menghitung biaya aktual dari setiap node ke tujuan. Untuk melakukan itu, bagaimanapun, akan membutuhkan traversal dari seluruh ruang pencarian, yang seharusnya dicegah oleh heuristik. Oleh karena itu diperlukan suatu heuristik yang cepat dan mudah untuk dihitung. Bagaimana kita menemukan heuristik seperti itu?

Ide yang menarik untuk menemukan heuristik adalah penyederhanaan masalah. Tugas aslinya cukup disederhanakan sehingga dapat diselesaikan dengan sedikit biaya komputasi. Biaya dari keadaan ke tujuan dalam masalah yang disederhanakan kemudian berfungsi sebagai perkiraan untuk masalah yang sebenarnya (lihat Gambar 6.13). Fungsi estimasi biaya ini kita nyatakan h.

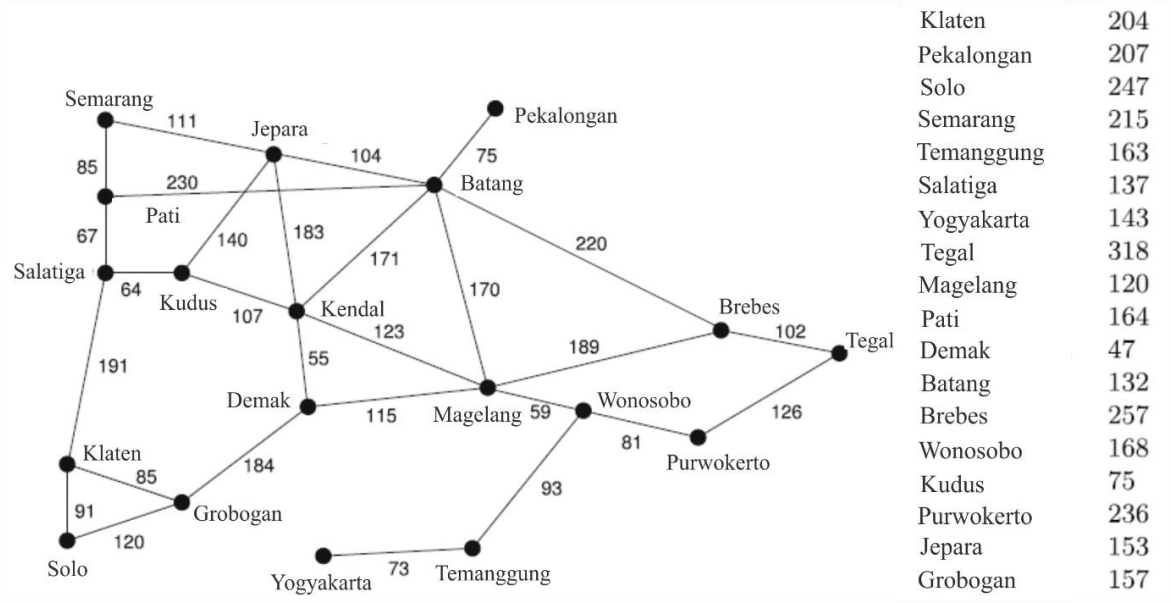
6.9 *GREEDY SEARCH*

Tampaknya masuk akal untuk memilih keadaan dengan perkiraan nilai h terendah (yaitu, yang dengan perkiraan biaya terendah) dari daftar keadaan yang tersedia saat ini. Estimasi biaya kemudian dapat digunakan secara langsung sebagai fungsi evaluasi. Untuk evaluasi dalam fungsi HEURISTICSEARCH kami menetapkan f (s)=h(s). Hal ini dapat dilihat dengan jelas pada contoh perencanaan perjalanan (Contoh 6.2). Kami menetapkan tugas untuk menemukan jalur garis lurus dari kota ke kota (yaitu, jarak terbang) sebagai penyederhanaan masalah. Alih-alih mencari rute yang optimal, pertama-tama kita menentukan dari setiap node sebuah rute dengan jarak terbang minimal ke tujuan. Kami memilih Kendal sebagai tujuan. Dengan demikian fungsi estimasi biaya menjadi

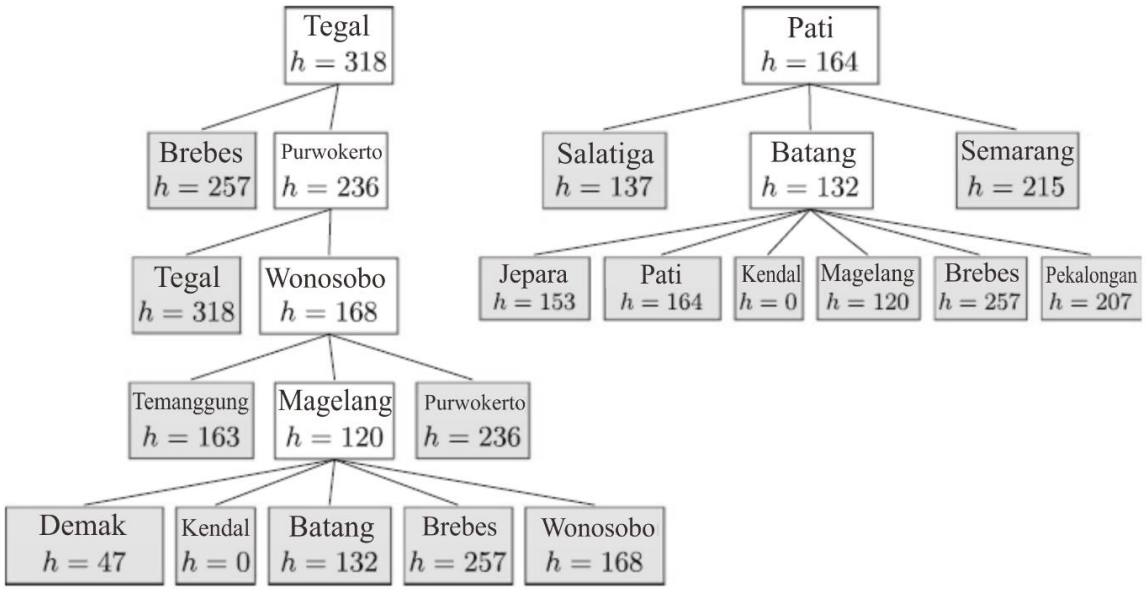
h(s) = jarak terbang dari kota s ke Kendal.

Jarak terbang dari semua kota ke Kendal diberikan pada Gambar 6.14 di sebelah grafik.
Pohon pencarian untuk memulai di Tegal ditunjukkan pada Gambar 6.15 kiri. Kita dapat melihat bahwa pohon itu sangat ramping. Dengan demikian, pencarian selesai dengan cepat. Sayangnya, pencarian ini tidak selalu menemukan solusi yang optimal. Misalnya, algoritma ini gagal menemukan solusi optimal ketika memulai di Mannheim (Gambar 6.15). Jalur Mannheim–Batang–Kendal memiliki panjang 401 km. Rute Mannheim–Salatiga–

Kudus–Kendal akan jauh lebih pendek pada 238 km. Saat kita mengamati grafik, penyebab masalah ini menjadi jelas. Batang sebenarnya agak lebih dekat daripada Salatiga ke Kendal, tetapi jarak dari Pati ke Batang secara signifikan lebih besar daripada jarak dari Mannheim ke Salatiga. Heuristik hanya melihat ke depan "dengan rakus" ke tujuan alih-alih juga memperhitungkan peregangan yang telah ditetapkan ke node saat ini. Inilah sebabnya kami memberinya nama pencarian serakah.



Gambar 6.14 Grafik kota dengan jarak terbang dari semua kota ke Kendal



Node	Magelang	Temanggung	Purwokerto	Brebes	Tegal
rating heuristik	120	163	236	257	318

Gambar 6.15 Pencarian serakah: dari Tegal ke Kendal (kiri) dan dari Pati ke Kendal (kanan).

Struktur data daftar node untuk pohon pencarian kiri, diurutkan berdasarkan peringkat node sebelum perluasan node Magelang diberikan

A★-Pencarian

Kami sekarang ingin memperhitungkan biaya yang timbul selama pencarian hingga node s saat ini. Pertama kita definisikan fungsi biaya

g(s) = Jumlah biaya yang masih harus dibayar dari awal hingga simpul saat ini,

kemudian tambahkan perkiraan biaya ke tujuan dan dapatkan sebagai fungsi evaluasi heuristik

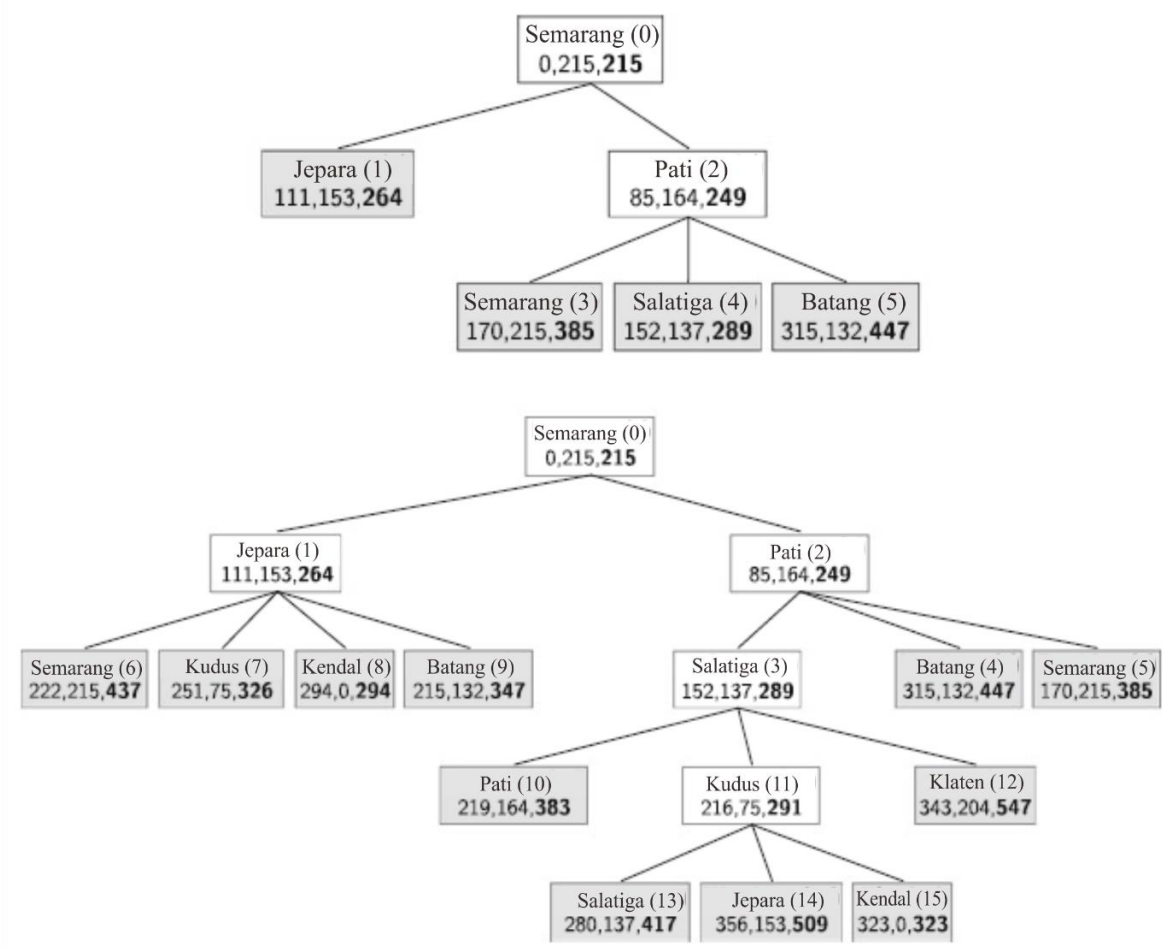
$$f(s) = g(s) + h(s)$$

Sekarang kami menambahkan satu lagi persyaratan kecil, tetapi penting.

Definisi 6.4

Fungsi estimasi biaya heuristik h(s) yang tidak pernah melebih-lebihkan biaya aktual dari keadaan s ke tujuan disebut dapat diterima.

Fungsi HEURISTICSEARCH bersama dengan fungsi evaluasi *f(s)=g(s)+h(s)* dan fungsi heuristik yang dapat diterima h disebut algoritma A★. Algoritma terkenal ini lengkap dan optimal. Dengan demikian A★ selalu menemukan solusi terpendek untuk setiap masalah pencarian yang dapat dipecahkan. Kami akan menjelaskan dan membuktikannya pada pembahasan berikut.



Gambar 6.16 Dua snapshot dari pohon pencarian A★ untuk rute optimal dari Semarang ke Kendal.

Dalam kotak di bawah nama kota s kita tunjukkan g(s), h(s), f(s). Angka dalam tanda kurung setelah nama kota menunjukkan urutan node yang dihasilkan oleh fungsi “Penerus”

Pertama kita terapkan algoritma A★ pada contoh. Kami mencari jalur terpendek dari Semarang ke Kendal. Di bagian atas Gambar 6.16 kita melihat bahwa penerus Mannheim dihasilkan sebelum penerus Jepara. Solusi optimal Semarang–Jepara–Kendal dihasilkan segera setelah itu pada langkah kedelapan, tetapi belum diakui seperti itu. Dengan demikian algoritme belum berhenti karena simpul Salatiga (3) memiliki nilai f yang lebih baik (lebih rendah) dan dengan demikian berada di depan simpul Kendal (8) sejalan. Hanya ketika semua nilai f lebih besar dari atau sama dengan simpul solusi Kendal (8) kami memastikan bahwa kami memiliki solusi optimal. Jika tidak, kemungkinan ada solusi lain dengan biaya lebih rendah. Kami sekarang akan menunjukkan bahwa ini benar secara umum.

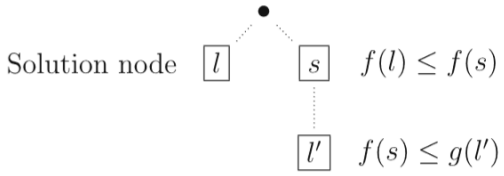
Teorema 6.2

Algoritma A★ optimal. Artinya, selalu menemukan solusi dengan total biaya terendah jika heuristik h dapat diterima.

Bukti Dalam algoritma HEURISTICSEARCH, setiap node s yang baru dibuat diurutkan berdasarkan fungsi "SortIn" menurut peringkat heuristiknya f(s). Node dengan nilai rating terkecil dengan demikian berada di awal daftar. Jika simpul l di awal daftar adalah simpul solusi, maka tidak ada simpul lain yang memiliki peringkat heuristik yang lebih baik. Untuk semua simpul lain s benar maka f(l) ≤ f(s). Karena heuristik dapat diterima, tidak ada solusi yang lebih baik l’ yang dapat ditemukan, bahkan setelah perluasan semua node lainnya (lihat Gambar 6.17). Ditulis secara resmi:

$$G(l) = g(l) + h(l) = f(l) \leq f(s) = g(s) + h(s) \leq g(l')$$

Persamaan pertama berlaku karena l adalah simpul solusi dengan h(l) = 0. Persamaan kedua adalah definisi dari f. Persamaan ketiga (dalam) berlaku karena daftar node terbuka diurutkan dalam urutan menaik. Persamaan keempat lagi-lagi merupakan definisi dari f. Akhirnya, yang terakhir (dalam)kesetaraan adalah diterimanya heuristik, yang tidak pernah melebih-lebihkan biaya dari node s ke solusi sewenang-wenang. Dengan demikian telah ditunjukkan bahwa g(l) ≤ g(l’), yaitu, solusi yang ditemukan *l* optimal



Gambar 6.17 Solusi pertama simpul l yang ditemukan oleh A★ tidak pernah memiliki biaya lebih tinggi daripada simpul arbitrer lain l’

6.10 PERENCANAAN RUTE DENGAN ALGORITMA PENCARIAN A★

Banyak sistem navigasi mobil saat ini menggunakan algoritma A★. Heuristik yang paling sederhana, tetapi sangat baik untuk menghitung A★ adalah jarak garis lurus dari node saat ini ke tujuan. Penggunaan 5 hingga 60 yang disebut landmark agak lebih baik. Untuk titik-titik yang dipilih secara acak ini, jalur terpendek ke dan dari semua node di peta dihitung dalam langkah pra-komputasi. Biarkan l menjadi tengara seperti itu, s simpul saat ini, dan z simpul tujuan. Juga biarkan c★(x, y) menjadi biaya jalur terpendek dari x ke y.

Kemudian kita peroleh untuk jalur terpendek dari s ke l pertidaksamaan segitiga (lihat Latihan 6.11)

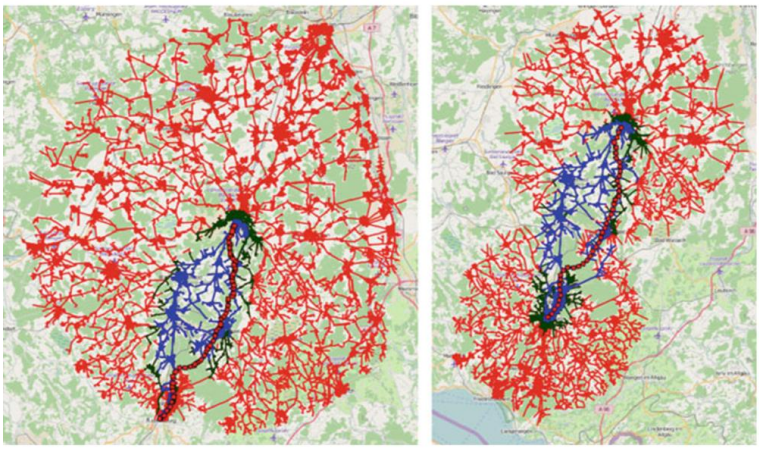
$$c^{\star}(s, l) \leq c^{\star}(s, z) + c^{\star}(z, l)$$

Memecahkan $c^{\star}(s, z)$ menghasilkan heuristik yang dapat diterima

$$H(s) = c^{\star}(s, l) - c^{\star}(z, l) \leq c^{\star}(s, z)$$

Dalam [Bat16], ditunjukkan bahwa sheuristik ini lebih baik daripada garis lurus untuk perencanaan rute. Di satu sisi, itu dapat dihitung lebih cepat daripada jarak garis lurus. Karena pra-komputasi, jarak ke landmark dapat dengan cepat diambil dari array, sedangkan jarak Euclidean harus dihitung secara individual. Ternyata heuristiklandmark semakin mempersempit ruang pencarian. Hal ini dapat dilihat pada gambar kiri Gambar 6.18, yang menggambarkan pohon pencarian pencarian $A^{\star}$ untuk merencanakan rute tepi tanpa heuristik (yaitu dengan h(s) = 0) diplot dalam warna merah, garis hijau tua menunjukkan pohon pencarian menggunakan heuristik jarak garis lurus, dan tepi heuristik tengara dengan dua puluh landmark diplot dengan warna biru.

Gambar kanan menunjukkan rute yang sama menggunakan pencarian dua arah, di mana rute dari Ravensburg ke Biberach dan satu arah berlawanan direncanakan secara efektif secara paralel. Jika rute bertemu, dengan kondisi heuristik tertentu, rute optimal telah ditemukan. Analisis kuantitatif ukuran pohon pencarian dan waktu komputasi pada PC dapat dilihat pada Tabel 6.2.



Gambar 6.18 $A^{\star}$ pohon pencarian tanpa heuristik (merah), dengan jarak garis lurus (hijau tua) dan dengan landmark (biru).

Gambar kiri menunjukkan pencarian satu arah dan gambar kanan menunjukkan pencarian dua arah. Perhatikan bahwa tepi hijau ditutupi oleh biru dan tepian tertutup oleh hijau dan biru.

Tabel 6.2 Perbandingan ukuran pohon pencarian dan waktu komputasi untuk perencanaan rute dengan dan tanpa masing-masing dari kedua heuristik

	Unidirectional		Bidirectional	
	Tree Size [nodes]	Comp. time [msec.]	Tree Size [nodes]	Comp. time [msec.]
No heuristic	62000	192	41850	122
Straight-line distance	9380	86	12193	84
Landmark heuristic	5260	16	7290	16

Heuristik tengara adalah pemenang yang jelas mengamati pencarian searah, kita melihat bahwa kedua heuristik jelas mengurangi ruang pencarian. Waktu komputasi benar-benar menarik. Dalam kasus heuristik tengara, kita melihat waktu komputasi dan ukuran ruang pencarian dikurangi dengan faktor sekitar 12. Dengan demikian, biaya komputasi heuristik tidak signifikan. Jarak garis lurus, bagaimanapun, menghasilkan pengurangan ruang pencarian faktor 6,6, tetapi hanya peningkatan faktor 2,2 dalam run time karena overhead komputasi jarak euclidean.

Dalam kasus pencarian dua arah, berbeda dengan pencarian satu arah, kami melihat pengurangan yang signifikan dari ruang pencarian bahkan tanpa heuristik. Di sisi lain, ruang pencarian lebih besar daripada kasus searah untuk kedua heuristik. Namun, karena node dipartisi menjadi dua daftar terurut dalam pencarian dua arah (lihat fungsi HEURISTICSEARCH pada Gambar 6.12), daftar ditangani lebih cepat dan waktu komputasi yang dihasilkan kira-kira sama.

Saat merencanakan rute, biasanya pengemudi lebih mementingkan waktu mengemudi daripada jarak yang ditempuh. Dengan demikian kita harus menyesuaikan heuristik yang sesuai dan mengganti jarak garis lurus d(s, z) dengan waktu t(s, z) = d(s, z)/v_{max}. Di sini kita harus membagi dengan kecepatan rata-rata maksimum, yang menurunkan heuristik karena menyebabkan waktu yang diperkirakan secara heuristik menjadi terlalu kecil. Heuristik tengara, sebaliknya, dibangun di atas rute optimal yang telah dihitung sebelumnya dan karenanya tidak menurun. Jadi, seperti yang ditunjukkan, pencarian rute yang dioptimalkan waktu dengan menggunakan heuristik landmark secara signifikan lebih cepat dibandingkan dengan jarak garis lurus yang dimodifikasi.

Algoritme hierarki kontraksi berkinerja lebih baik daripada A* dengan heuristik tengara. Ini didasarkan pada gagasan untuk menggabungkan, dalam langkah pra-komputasi, beberapa tepi menjadi apa yang disebut pintasan, yang kemudian digunakan untuk mengurangi ruang pencarian.

IDA*-Pencarian

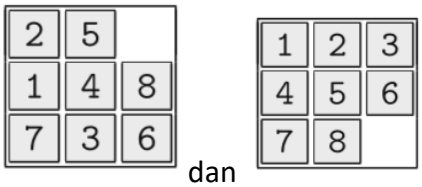
Pencarian A* mewarisi sebuahquirk dari pencarian pertama yang luas. Ini menyimpan banyak node dalam memori, yang dapat menyebabkan penggunaan memori yang sangat tinggi. Selanjutnya, daftar node yang terbuka harus disortir. Dengan demikian, penyisipan node ke dalam daftar dan penghapusan node dari daftar tidak dapat lagi berjalan dalam waktu yang konstan, yang meningkatkan kompleksitas algoritma sedikit. Berdasarkan algoritma heapsortal, kita dapat menyusun kemudian membuat daftar dengan kompleksitas waktu logaritmik untuk penyisipan dan penghapusan node.

Kedua masalah tersebut dapat diselesaikan—mirip dengan pencarian pertama yang luas—dengan pendalaman berulang. Kami bekerja dengan pencarian mendalam-pertama dan berturut-turut menaikkan batas. Namun, daripada bekerja dengan batas kedalaman, di sini kami menggunakan batas untuk evaluasi heuristik f(s). Proses ini disebut algoritma IDA*.

6.11 PERBANDINGAN EMPIRIS DARI ALGORITMA PENCARIAN

Di A*, atau (sebagai alternatif) IDA*, kami memiliki algoritma pencarian dengan banyak properti bagus. Lengkap dan optimal. Dengan demikian dapat digunakan tanpa risiko. Hal yang paling penting, bagaimanapun, adalah bahwa ia bekerja dengan heuristik, dan karena itu dapat secara signifikan mengurangi waktu komputasi yang dibutuhkan untuk menemukan solusi. Kami ingin mengeksplorasi ini secara empiris dalam contoh 8 teka-teki.

Untuk teka-teki 8 ada dua heuristik sederhana yang dapat diterima. Heuristik h1 hanya menghitung jumlah kotak yang tidak berada di tempat yang tepat. Jelas heuristik ini dapat diterima. Heuristik h2 mengukur jarak Manhattan. Untuk setiap bujur sangkar, jarak horizontal dan vertikal ke lokasi bujur sangkar itu di negara tujuan dijumlahkan. Nilai ini kemudian dijumlahkan ke semua kotak. Misalnya, jarak Manhattan dari kedua negara bagian



dan

dihitung sebagai

$$h_2(s) = 1 + 1 + 1 + 1 + 2 + 0 + 3 + 1 = 10$$

Dapat diterimanya jarak Manhattan juga jelas (lihat Latihan 6.13).

Algoritma yang dijelaskan diimplementasikan di Mathematica. Untuk perbandingan dengan pencarian yang tidak diinformasikan, algoritma A★ dengan dua heuristik h1 dan h2 dan pendalaman berulang diterapkan pada 132 masalah 8 teka-teki yang dihasilkan secara acak. Nilai rata-rata untuk jumlah langkah dan waktu komputasi diberikan pada Tabel 6.3. Kami melihat bahwa heuristik secara signifikan mengurangi biaya pencarian dibandingkan dengan pencarian tanpa informasi.

Tabel 6.3 Perbandingan biaya komputasi uninformed search dan heuristic search untuk memecahkan masalah 8 teka-teki dengan berbagai kedalaman.

Depth	Iterative deepening		A★ algorithm				Num. runs
			Heuristic h_1		Heuristic h_2		
	Steps	Time [sec]	Steps	Time [sec]	Steps	Time [sec]	
2	20	0.003	3.0	0.0010	3.0	0.0010	10
4	81	0.013	5.2	0.0015	5.0	0.0022	24
6	806	0.13	10.2	0.0034	8.3	0.0039	19
8	6455	1.0	17.3	0.0060	12.2	0.0063	14
10	50512	7.9	48.1	0.018	22.1	0.011	15
12	486751	75.7	162.2	0.074	56.0	0.031	12
			IDA★				
14	–	–	10079.2	2.6	855.6	0.25	16
16	–	–	69386.6	19.0	3806.5	1.3	13
18	–	–	708780.0	161.6	53941.5	14.1	4

Pengukuran dalam langkah dan detik. Semua nilai rata-rata selama beberapa kali berjalan (lihat kolom terakhir). Jika kita membandingkan pendalaman berulang ke A★ dengan h_1 pada kedalaman 1_2, misalnya, menjadi jelas bahwa h_1 mengurangi jumlah langkah dengan faktor sekitar 3.000, tetapi waktu komputasi hanya dengan faktor 1.023. Hal ini disebabkan biaya yang lebih tinggi per langkah untuk perhitungan heuristik.

Pemeriksaan lebih dekat mengungkapkan lompatan dalam jumlah langkah antara kedalaman 12 dan kedalaman 14 di kolom untuk h1. Lompatan ini tidak dapat dijelaskan hanya dengan pekerjaan berulang yang dilakukan oleh IDA★. Itu terjadi karena implementasi algoritma A★ menghapus duplikat dari node identik dan dengan demikian mengecilkan ruang pencarian. Ini tidak mungkin dengan IDA★ karena hampir tidak

menyimpan node. Meskipun demikian, A★ tidak dapat lagi bersaing dengan IDA★ di luar kedalaman 14 karena biaya penyortiran pada node baru mendorong waktu per langkah begitu banyak.

Perhitungan faktor percabangan efektif menurut (6.1) menghasilkan nilai sekitar 2,8 untuk pencarian tanpa informasi. Jumlah ini sesuai dengan nilai. Heuristik h1 mengurangi faktor percabangan menjadi nilai sekitar 1,5 dan h_2 menjadi sekitar 1,3. Kita dapat melihat pada tabel bahwa pengurangan kecil dari faktor percabangan dari 1,5 menjadi 1,3 memberi kita keuntungan besar dalam waktu komputasi. Pencarian heuristik dengan demikian memiliki signifikansi praktis yang penting karena dapat memecahkan masalah yang jauh dari jangkauan untuk pencarian tanpa informasi.

6.12 RINGKASAN

Dari berbagai algoritma pencarian untuk uninformed search, pendalaman iteratif adalah satu-satunya yang praktis karena lengkap dan dapat bertahan dengan memori yang sangat sedikit. Namun, untuk masalah pencarian kombinatorial yang sulit, bahkan pendalaman berulang biasanya gagal karena ukuran ruang pencarian. Pencarian heuristik membantu di sini melalui pengurangan faktor percabangan efektif. Algoritma IDA★, seperti pendalaman berulang, selesai dan membutuhkan sedikit memori.

Heuristik secara alami hanya memberikan keuntungan yang signifikan jika heuristik itu “baik”. Ketika memecahkan masalah pencarian yang sulit, tugas sebenarnya pengembang terdiri dari merancang heuristik yang sangat mengurangi faktor percabangan efektif. Kita akan menangani masalah ini dan juga menunjukkan bagaimana teknik pembelajaran mesin dapat digunakan untuk menghasilkan heuristik secara otomatis.

Sebagai penutup, perlu dicatat bahwa heuristik tidak memiliki keunggulan kinerja untuk masalah yang tidak dapat dipecahkan karena masalah yang tidak dapat dipecahkan hanya dapat ditentukan ketika pohon pencarian lengkap telah dicari. Untuk masalah yang dapat ditentukan seperti teka-teki 8 ini berarti bahwa seluruh pohon pencarian harus dilalui hingga kedalaman maksimal apakah heuristik sedang digunakan atau tidak. Heuristik selalu merupakan kerugian dalam hal ini, disebabkan oleh biaya komputasi untuk mengevaluasi heuristik. Kerugian ini biasanya dapat diperkirakan dengan faktor konstan yang tidak bergantung pada ukuran masalah. Untuk masalah yang tidak dapat diputuskan seperti pembuktian rumus PL1, pohon pencarian bisa sangat dalam. Ini berarti bahwa, dalam kasus yang tidak terpecahkan, pencarian berpotensi tidak pernah berakhir. Singkatnya kita dapat mengatakan sebagai berikut: untuk masalah yang dapat dipecahkan, heuristik sering mengurangi waktu komputasi secara dramatis, tetapi untuk masalah yang tidak dapat dipecahkan, biayanya bahkan bisa lebih tinggi dengan heuristik.

6.13 *GAME* DENGAN LAWAN

Permainan untuk dua pemain, seperti catur, catur, Othello, dan Go bersifat deterministik karena setiap tindakan (gerakan) menghasilkan status anak yang sama dengan status induk yang sama. Sebaliknya, backgammon bersifat non-deterministik karena status turunannya bergantung pada hasil pelemparan dadu. Semua *game* ini dapat diamati karena setiap pemain selalu mengetahui status *game* secara lengkap. Banyak permainan kartu, seperti poker, misalnya, hanya sebagian yang dapat diamati karena pemain tidak mengetahui kartu pemain lain, atau hanya memiliki sebagian pengetahuan tentang kartu tersebut.

Masalah yang dibahas sejauh ini dalam bab ini bersifat deterministik dan dapat diamati. Berikut ini kita akan melihat permainan yang juga deterministik dan dapat diamati.

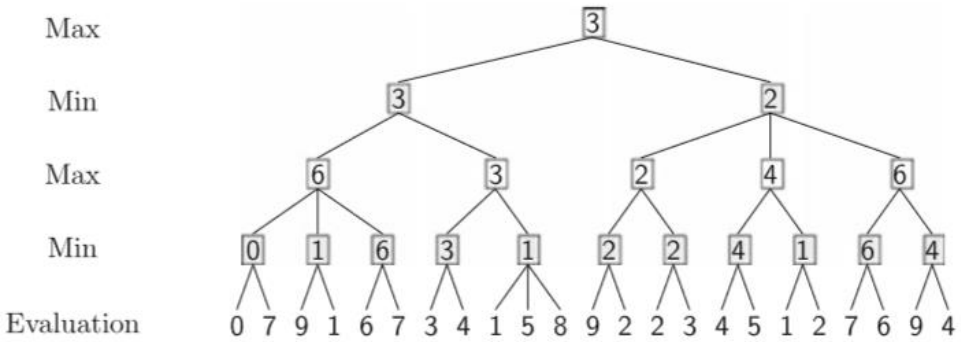
Selanjutnya, kami akan membatasi diri pada permainan zero-sum. Ini adalah permainan di mana setiap keuntungan yang diperoleh satu pemain berarti kerugian dengan nilai yang sama untuk lawan. Jumlah keuntungan dan kerugian selalu sama dengan nol. Ini berlaku untuk permainan catur, catur, Othello, dan Go, yang disebutkan di atas.

6.14 PENCARIAN MINIMAX

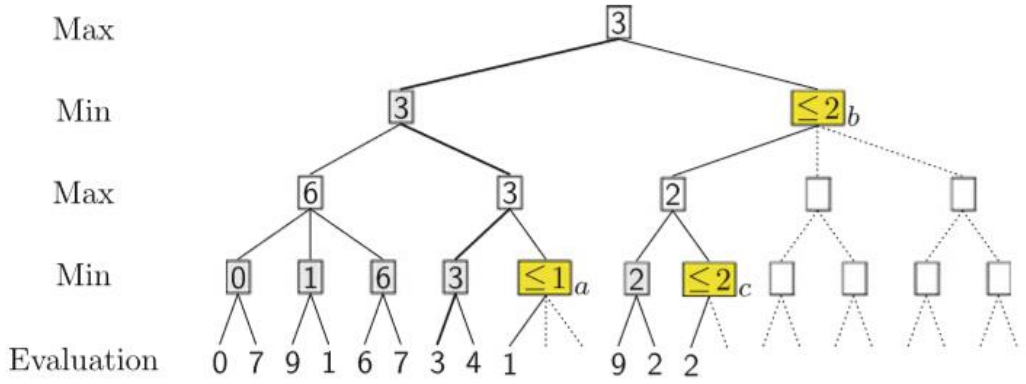
Tujuan setiap pemain adalah melakukan gerakan optimal yang menghasilkan kemenangan. Pada prinsipnya adalah mungkin untuk membangun pohon pencarian dan benar-benar mencari melaluinya (seperti dengan teka-teki 8) untuk serangkaian gerakan yang akan menghasilkan kemenangan. Namun, ada beberapa kekhasan yang harus diperhatikan:

1. Faktor percabangan efektif dalam catur adalah sekitar 30 hingga 35. Dalam permainan tipikal dengan 50 gerakan per pemain, pohon pencarian memiliki lebih dari $30^{100} \approx 10^{148}$ simpul daun. Jadi tidak ada kesempatan untuk sepenuhnya menjelajahi pohon pencarian. Selain itu, catur sering dimainkan dengan batas waktu. Karena persyaratan waktu nyata ini, pencarian harus dibatasi pada kedalaman yang sesuai di pohon, misalnya delapan setengah gerakan. Karena di antara simpul daun dari pohon yang dibatasi kedalaman ini biasanya tidak ada simpul solusi (yaitu, simpul yang mengakhiri permainan), fungsi evaluasi heuristik B untuk posisi papan digunakan. Tingkat permainan program sangat tergantung pada kualitas fungsi evaluasi ini. Oleh karena itu.
2. Berikut ini kita akan memanggil pemain yang permainannya ingin kita optimalkan Max, dan lawannya Min. Gerakan lawan (Min) tidak diketahui sebelumnya, dan dengan demikian juga bukan pohon pencarian yang sebenarnya. Masalah ini dapat diselesaikan secara elegan dengan mengasumsikan bahwa lawan selalu melakukan gerakan terbaik yang dia bisa. Semakin tinggi evaluasi B(s) untuk posisi s, semakin baik posisi s untuk pemain Max dan semakin buruk untuk lawannya Min. Max mencoba memaksimalkan evaluasi gerakannya, sedangkan Min membuat gerakan yang menghasilkan evaluasi serendah mungkin.

Sebuah pohon pencarian dengan empat setengah gerakan dan evaluasi semua daun diberikan pada Gambar 6.19. Evaluasi simpul dalam diturunkan secara rekursif sebagai maksimum atau minimum dari simpul anak, tergantung pada tingkat simpul.



Gambar 6.19 Pohon permainan minimax dengan pandangan ke depan dari empat gerakan setengah



Gambar 6.20 Pohon permainan alfa-beta dengan pandangan ke depan dari empat setengah gerakan.

Bagian pohon yang putus-putus tidak dilintasi karena tidak berpengaruh pada hasil akhirnya

6.15 PEMANGKASAN ALFA-BETA

Dengan beralih antara maksimalisasi dan minimalisasi, kita dapat menghemat banyak pekerjaan dalam beberapa keadaan. Pemangkasan alfa-beta bekerja dengan pencarian kedalaman-pertama hingga batas kedalaman yang telah ditentukan sebelumnya. Dengan cara ini pohon pencarian dicari dari kiri ke kanan. Seperti dalam pencarian minimax, di node minimum, minimum dihasilkan dari nilai minimum node penerus dan di node maksimum juga maksimum. Pada Gambar 6.20 proses ini digambarkan untuk pohon dari Gambar 6.19. Pada simpul bertanda a, semua penerus lainnya dapat diabaikan setelah anak pertama dievaluasi sebagai nilai 1 karena minimumnya pasti ≤1. Bahkan bisa menjadi lebih kecil lagi, tapi itu tidak relevan karena maksimal sudah ≥3 satu tingkat di atas. Terlepas dari bagaimana evaluasi penerus yang tersisa, nilai maksimum akan tetap 3. Secara analog pohon akan dipangkas pada simpul b. Karena anak pertama dari b memiliki nilai 2, minimum yang akan dibangkitkan untuk b hanya bisa kurang dari atau sama dengan 2. Tetapi nilai maksimum pada simpul akar sudah pasti adalah ≥3. Ini tidak dapat diubah dengan nilai ≤2. Dengan demikian, sisa subpohon b dapat dipangkas. Alasan yang sama berlaku untuk simpul c. Namun, simpul maksimum yang relevan bukanlah orang tua langsung, tetapi simpul akar. Ini bisa digeneralisasi.

- setiap simpul daun evaluasi dihitung.
- Untuk setiap node maksimum, nilai anak terbesar saat ini disimpan dalam a.
- Untuk setiap simpul minimum, nilai anak terkecil saat ini disimpan di b.
- Jika pada simpul minimum k nilai sekarang $\beta \leq \alpha$, maka pencarian di bawah k dapat berakhir. Di sini a adalah nilai terbesar dari simpul maksimum di jalur dari akar ke k.
- Jika pada node maksimum l nilai saat ini $\alpha \geq \beta$, maka pencarian di bawah l dapat berakhir. Di sini b adalah nilai terkecil dari simpul minimum dalam lintasan dari akar ke l.

```
ALPHABETAMAX(Node, α, β)
If DepthLimitReached(Node) Return(Rating(Node))
NewNodes = Successors(Node)
While NewNodes ≠ Ø
    α = Maximum(α, ALPHABETAMIN(First(NewNodes), α, β))
    If α ≥ β Return(β)
    NewNodes = Rest(NewNodes)
Return(α)
```

```
ALPHABETAMIN(Node, α, β)
If DepthLimitReached(Node) Return(Rating(Node))
NewNodes = Successors(Node)
While NewNodes ≠ Ø
    β = Minimum(β, ALPHABETAMAX(First(NewNodes), α, β))
    If β ≤ α Return(α)
    NewNodes = Rest(NewNodes)
Return(β)
```

Gambar 6.21 Algoritma pencarian alpha-beta dengan dua fungsi ALPHABETAMIN dan ALPHABETAMAX

Algoritme yang diberikan pada Gambar 6.21 merupakan perluasan dari pencarian depth-first dengan dua fungsi yang dipanggil secara bergantian. Ia menggunakan nilai-nilai yang didefinisikan di atas untuk α dan β. Panggilan pemangkasan alfa-beta awal dilakukan dengan perintah ALPHABETAMAX(RootNode, –∞, ∞).

Kompleksitas

Waktu komputasi yang dihemat oleh pemangkasan alfa-beta sangat bergantung pada urutan node anak yang dilalui. Dalam kasus terburuk, pemangkasan alfa-beta tidak menawarkan keuntungan apa pun. Untuk faktor percabangan konstan b, jumlah n_d dari simpul daun yang dievaluasi pada kedalaman d sama dengan

$$n_d = b^d$$

Dalam kasus terbaik, ketika penerus node maksimum diurutkan secara menurun dan penerus dari node minimum diurutkan secara menaik, faktor percabangan efektif dikurangi menjadi √b. Dalam catur ini berarti pengurangan substansial dari faktor percabangan efektif dari 35 menjadi sekitar 6. Kemudian hanya

$$n_d = \sqrt{b^d} = b^{d/2}$$

node daun akan dibuat. Ini berarti bahwa batas kedalaman dan dengan demikian juga cakrawala pencarian digandakan dengan pemangkasan alfa-beta. Namun, ini hanya berlaku dalam kasus penerus yang diurutkan secara optimal karena peringkat node anak tidak diketahui pada saat dibuat. Jika simpul anak diurutkan secara acak, maka faktor percabangan dikurangi menjadi $b^{3/4}$ dan jumlah simpul daun menjadi

$$dn = b^{3/d}$$

Dengan daya komputasi yang sama, komputer catur yang menggunakan pemangkasan alfa-beta dapat, misalnya, menghitung delapan setengah langkah ke depan, bukan enam, dengan faktor percabangan efektif sekitar 14.

Untuk menggandakan kedalaman pencarian seperti yang disebutkan di atas, kita membutuhkan node anak untuk diurutkan secara optimal, yang tidak terjadi dalam praktiknya. Jika tidak, pencarian tidak akan diperlukan. Dengan trik sederhana kita bisa mendapatkan urutan node yang relatif baik. Kami menghubungkan pemangkasan alfa-beta dengan pendalaman berulang di atas batas kedalaman. Jadi pada setiap batas kedalaman baru kita dapat mengakses peringkat semua node dari level sebelumnya dan memesan penerus di setiap cabang. Dengan demikian kita mencapai faktor percabangan efektif kira-kira 7 sampai 8, yang tidak jauh dari optimum teoritis √35.

***Game* Non-deterministik**

Pencarian Minimax dapat digeneralisasi untuk semua *game* dengan tindakan non-deterministik, seperti backgammon. Setiap pemain berguling sebelum bergerak, yang dipengaruhi oleh hasil lemparan dadu. Di pohon permainan sekarang ada tiga jenis level dalam urutan

Max, dice, Min, dice, ...,

di mana setiap simpul lemparan dadu bercabang enam cara.

6.16 FUNGSI EVALUASI HEURISTIK

Bagaimana kita menemukan fungsi evaluasi heuristik yang baik untuk tugas pencarian? Di sini pada dasarnya ada dua pendekatan. Cara klasik menggunakan pengetahuan ahli manusia. Insinyur pengetahuan diberi tugas yang biasanya sulit untuk memformalkan pengetahuan implisit pakar dalam bentuk program komputer. Kami sekarang ingin menunjukkan bagaimana proses ini dapat disederhanakan dalam contoh program catur.

Pada langkah pertama, para ahli ditanyai tentang faktor terpenting dalam pemilihan langkah. Kemudian dicoba untuk mengukur faktor-faktor ini. Kami memperoleh daftar fitur atau atribut yang relevan. Ini kemudian (dalam kasus paling sederhana) digabungkan menjadi fungsi evaluasi linier B(s) untuk posisi, yang dapat terlihat seperti:

Persamaan 6.3

$$B(s) = a_1 . material + a_2 . pawn\_structure + a_3 . king\_safety + a_4 . knight\_in\_center + a_5 . bishop\_diagonal\_coverage + ...,$$

di mana "material" sejauh ini merupakan fitur yang paling penting dan dihitung dengan

$$material = material(own\_team) – material\ (opponent)$$

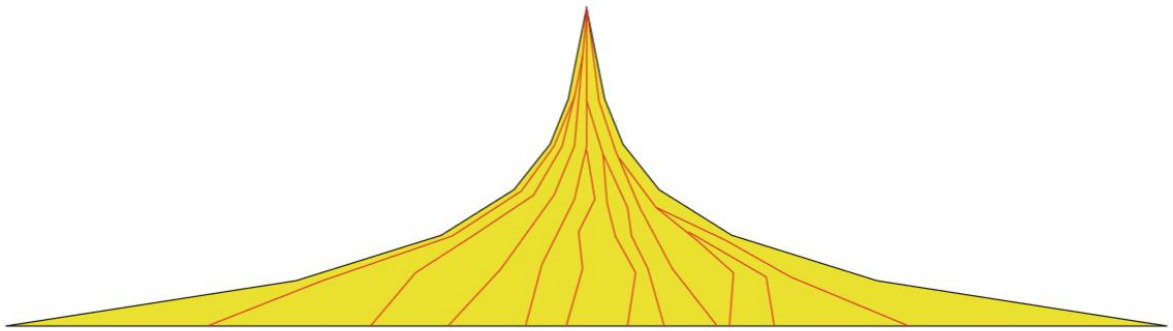
dengan

$$material(team) = num\_pawn(team) . 100 + num\_knights(team) . 300 + num\_bishops(team) . 300 + num\_rooks(team) . 500 + num\_queens(team) . 900$$

Hampir semua program catur membuat evaluasi materi yang serupa. Namun, ada perbedaan besar untuk semua fitur lainnya. Pada langkah berikutnya bobot ai dari semua fitur harus ditentukan. Ini diatur secara intuitif setelah berdiskusi dengan para ahli, kemudian diubah setelah setiap pertandingan berdasarkan pengalaman positif dan negatif. Fakta bahwa proses pengoptimalan ini sangat mahal dan lebih jauh lagi bahwa kombinasi linear fitur sangat terbatas menunjukkan penggunaan pembelajaran mesin.

6.17 BELAJAR EVALUASI HEURISTIK

Kami sekarang ingin secara otomatis mengoptimalkan bobot ai dari fungsi evaluasi B(s) dari (6.3). Dalam pendekatan ini pakar hanya ditanya tentang fitur yang relevan f1(s), … , fn(s) untuk status permainan s. Kemudian proses pembelajaran mesin digunakan dengan tujuan menemukan fungsi evaluasi yang seoptimal mungkin. Kami mulai dengan fungsi evaluasi yang telah ditentukan sebelumnya (ditentukan oleh proses pembelajaran), dan kemudian membiarkan program catur bermain. Di akhir permainan, peringkat diturunkan dari hasil (menang, kalah, atau seri). Berdasarkan peringkat ini, fungsi evaluasi diubah dengan tujuan membuat lebih sedikit kesalahan di waktu berikutnya. Pada prinsipnya, hal yang sama yang dilakukan oleh pengembang sekarang diurus secara otomatis oleh proses pembelajaran.



Gambar 6.22 Dalam sketsa pohon pencarian ini, beberapa jalur MCTS ke simpul daun ditunjukkan dengan warna merah.

Perhatikan bahwa hanya sebagian kecil dari pohon yang dicari, semudah ini terdengar, sangat sulit dalam prakteknya. Masalah utama dengan meningkatkan peringkat posisi berdasarkan pertandingan menang atau kalah dikenal saat ini sebagai masalah penugasan kredit. Kami sebenarnya memiliki peringkat di akhir permainan, tetapi tidak ada peringkat untuk gerakan individu. Jadi agen melakukan banyak tindakan tetapi tidak menerima umpan balik positif atau negatif sampai akhir. Lalu bagaimana seharusnya memberikan umpan balik ini ke banyak tindakan yang diambil di masa lalu? Dan bagaimana seharusnya meningkatkan tindakannya dalam kasus itu? Bidang menarik dari *reinforcement learning*/pembelajaran penguatan berhubungan dengan pertanyaan-pertanyaan ini (lihat Bab 10).

Pencarian pohon Monte Carlo (MCTS) bekerja dengan cara yang hampir sama. Untuk meningkatkan peringkat heuristik dari keadaan permainan s, sejumlah acak cabang pohon pencarian mulai dari keadaan ini dieksplorasi sampai akhir dan dievaluasi, atau dihentikan pada kedalaman tertentu dan kemudian simpul daun dievaluasi secara heuristik. Evaluasi B(s) dari keadaan s diberikan sebagai rata-rata dari semua skor simpul daun. Penggunaan jalur MCTS hanya membutuhkan sebagian kecil dari keseluruhan pohon yang meledak secara eksponensial untuk dicari. Hal ini diilustrasikan pada Gambar 6.22. Untuk banyak game simulasi komputer, seperti catur, algoritme ini dapat digunakan untuk mencapai permainan yang lebih baik untuk upaya komputasi yang sama atau untuk mengurangi upaya komputasi untuk tingkat kesulitan yang sama. Metode ini digunakan bersama dengan algoritme pembelajaran mesin pada tahun 2016 oleh program AlphaGo, yang merupakan program Go pertama yang mengalahkan pemain manusia kelas dunia.

6.18 NEGARA SENI

Untuk evaluasi kualitas proses pencarian heuristik, saya ingin mengulangi definisi Elaine Rich:

> *Kecerdasan Buatan adalah studi tentang bagaimana membuat komputer melakukan hal-hal di mana, pada saat ini, orang lebih baik.*

Hampir tidak ada tes yang lebih cocok untuk memutuskan apakah program komputer cerdas sebagai perbandingan langsung antara komputer dan manusia dalam permainan seperti catur, catur, backgammon atau Go.

Pada tahun 1950, Claude Shannon, Konrad Zuse, dan John von Neumann memperkenalkan program catur pertama, yang, bagaimanapun, tidak dapat diimplementasikan atau membutuhkan banyak waktu untuk diterapkan. Hanya beberapa tahun kemudian, pada tahun 1955, Arthur Samuel menulis sebuah program yang memainkan permainan catur dan dapat meningkatkan parameternya sendiri melalui proses pembelajaran yang sederhana. Untuk melakukan ini, ia menggunakan komputer logika pertama yang dapat diprogram, IBM 701. Namun, dibandingkan dengan komputer catur saat ini, komputer ini memiliki akses ke sejumlah besar permainan yang diarsipkan, yang setiap gerakannya telah dinilai oleh para ahli. Dengan demikian program meningkatkan fungsi evaluasinya. Untuk mencapai perbaikan lebih lanjut, Samuel membuat programnya bermain melawan dirinya sendiri. Dia memecahkan masalah penugasan kredit dengan cara yang sederhana. Untuk setiap posisi individu selama permainan, evaluasi yang dilakukan oleh fungsi B(s) dibandingkan dengan yang dihitung dengan pemangkasan alfa-beta dan perubahan B(s) yang sesuai. Pada tahun 1961 program caturnya mengalahkan pemain catur terbaik keempat di AS. Dengan karya terobosan ini, Samuel hampir 30 tahun lebih maju dari zamannya. Baru pada awal tahun sembilan puluhan, ketika pembelajaran penguatan muncul, Gerald Tersauro membangun program backgammon pembelajaran bernama TD-Gammon, yang dimainkan di tingkat juara dunia (lihat Bab 10).

Catur

Saat ini banyak program catur yang bermain di atas level grandmaster. Terobosan datang pada tahun 1997, saat IBM's Deep Blue mengalahkan juara dunia catur Gary Kasparov dengan skor 3,5 game berbanding 2,5. Deep Blue rata-rata dapat menghitung 12 setengah langkah ke depan dengan pemangkasan alfa-beta dan evaluasi posisi heuristik.

Sekitar tahun 2005 salah satu komputer catur yang paling kuat adalah Hydra, komputer paralel yang dimiliki oleh sebuah perusahaan di Uni Emirat Arab. Perangkat lunak ini dikembangkan oleh ilmuwan Christian Donninger (Austria) dan Ulf Lorenz (Jerman), serta juara catur utama Jerman Christopher Lutz. Hydra menggunakan 64 prosesor Xeon paralel dengan daya komputasi sekitar 3 GHz dan memori masing-masing 1 GByte. Untuk fungsi evaluasi posisi, setiap prosesor memiliki co-prosesor FPGA (field programmable gate array). Dengan demikian menjadi mungkin untuk mengevaluasi 200 juta posisi per detik bahkan dengan fungsi evaluasi yang mahal.

Dengan teknologi ini Hydra rata-rata dapat menghitung sekitar 18 langkah ke depan. Dalam situasi khusus dan kritis, cakrawala pencarian bahkan dapat diperpanjang hingga 40 setengah gerakan. Jelas cakrawala seperti ini melampaui apa yang bahkan dapat dilakukan oleh grand champion, karena Hydra sering membuat gerakan yang tidak dapat dipahami oleh grand champion, tetapi pada akhirnya mengarah pada kemenangan. Pada tahun 2005 Hydra mengalahkan grandmaster peringkat ketujuh Michael Adams dengan 5,5-0,5 game. Hydra menggunakan sedikit pengetahuan buku teks khusus tentang catur, bukan pencarian alfa-beta dengan heuristik yang relatif umum dan terkenal dan evaluasi posisi kode tangan

yang baik. Secara khusus, Hydra tidak mampu belajar. Perbaikan dilakukan antar *game* oleh pengembang. Akibatnya, Hydra segera diungguli oleh mesin yang menggunakan algoritma pembelajaran cerdas daripada perangkat keras yang mahal.

Pada tahun 2009 sistem Pocket Fritz 4, berjalan pada PDA, memenangkan turnamen catur Copa Mercosur di Buenos Aires dengan sembilan kemenangan dan satu hasil imbang melawan 10 pemain catur manusia yang luar biasa, tiga di antaranya adalah grandmaster. Meskipun tidak banyak informasi tentang struktur internal perangkat lunak yang tersedia, mesin catur ini mewakili tren dari kekuatan komputasi mentah menuju lebih banyak kecerdasan. Mesin ini bermain di level grandmaster, dan sebanding dengan, jika tidak lebih baik dari Hydra. Menurut pengembang Pocket Fritz Stanislav Tsukrov, Pocket Fritz dengan mesin pencari caturnya HIARCS 13 mencari kurang dari 20.000 posisi per detik, yang lebih lambat dari Hydra dengan faktor sekitar 10.000. Ini mengarah pada kesimpulan bahwa HIARCS 13 jelas menggunakan heuristik yang lebih baik untuk mengurangi faktor percabangan efektif daripada Hydra dan dengan demikian dapat disebut lebih cerdas daripada Hydra. Omong-omong, HIARCS adalah kependekan dari Sistem Catur Respon Otomatis Kecerdasan Tinggi.

Go

Meskipun saat ini tidak ada manusia yang memiliki peluang melawan komputer catur terbaik, masih ada banyak tantangan untuk AI. Misalnya Pergi. Dalam permainan Jepang kuno ini, dimainkan di papan persegi dengan 361 ruang dengan 181 batu putih dan 180 batu hitam, faktor percabangan efektifnya adalah sekitar 250. Setelah 8 setengah gerakan, sudah ada $1 \cdot 5 \cdot 10^{19}$ kemungkinan posisi. Mengingat kerumitan ini, tidak ada algoritme pencarian pohon permainan klasik yang terkenal yang memiliki peluang melawan pemain Go manusia yang baik. Padahal dalam edisi terbaru buku ini sebelumnya, disebutkan bahwa:

> *Para ahli setuju bahwa algoritma "benar-benar cerdas" diperlukan di sini. Pencacahan kombinatorik dari semua kemungkinan adalah pendekatan yang salah. Sebaliknya, diperlukan prosedur yang mengenali pola di papan tulis, melacak perkembangan bertahap, dan membuat keputusan "intuitif" yang cepat. Mirip dengan pengenalan objek dalam gambar yang kompleks, kita manusia masih jauh lebih unggul dari program komputer saat ini. Kami memproses gambar secara keseluruhan dengan cara yang sangat paralel, sedangkan komputer memproses jutaan piksel secara berurutan dan memiliki kesulitan besar untuk mengenali hal-hal penting dalam kelimpahan piksel. Program "The Many Faces of Go" mengenali 1100 pola yang berbeda dan mengetahui 200 strategi bermain yang berbeda. Namun, semua program Go masih memiliki kesulitan besar untuk mengenali apakah sekelompok batu itu hidup atau mati, atau di mana di antara keduanya untuk mengklasifikasikannya.*

Pernyataan ini sekarang sudah usang. Pada Januari 2016, Google dan Facebook menerbitkan terobosan tersebut secara bersamaan. Pada bulan yang sama, program AlphaGo, yang dikembangkan dan disajikan oleh Google DeepMind, mengalahkan juara Go Eropa Fan Hui 5:0. Dua bulan kemudian, pemain Korea Lee Sedol, salah satu yang terbaik di dunia, dikalahkan 4:1. Pembelajaran Mendalam untuk pengenalan pola, pembelajaran penguatan (lihat Bab 10) dan pencarian pohon Monte Carlo (MCTS) menghasilkan hasil yang sukses ini.

Program memainkan ratusan ribu permainan melawan dirinya sendiri dan menggunakan hasilnya (menang, kalah, seri) untuk mempelajari skor heuristik terbaik untuk posisi tertentu. Pencarian pohon Monte Carlo digunakan sebagai pengganti

pencarian Minimax, yang tidak cocok untuk Go. Setelah kami terbiasa dengan algoritma pembelajaran yang diperlukan, kami akan memperkenalkan AlphaGo.

6.19 LATIHAN

Latihan 6.1

(a) Buktikan Teorema 6.1, dengan kata lain, buktikan bahwa untuk pohon dengan faktor percabangan konstan besar b, hampir semua node berada pada level terakhir pada kedalaman d.

(b) Tunjukkan bahwa ini tidak selalu benar jika faktor percabangan efektifnya besar dan tidak konstan.

Latihan 6.2

(a) Hitung faktor percabangan rata-rata untuk teka-teki 8 tanpa pemeriksaan siklus. Faktor percabangan rata-rata adalah faktor percabangan yang dimiliki pohon dengan jumlah simpul yang sama pada tingkat terakhir, faktor percabangan konstan, dan kedalaman yang sama.

(b) Hitung faktor percabangan rata-rata untuk teka-teki 8 untuk pencarian tanpa informasi sambil menghindari siklus dengan panjang 2.

Latihan 6.3

(a) Apa perbedaan antara faktor percabangan rata-rata dan efektif (Definisi 6.2)?

(b) Mengapa faktor percabangan efektif lebih cocok untuk analisis dan perbandingan waktu komputasi algoritma pencarian daripada faktor percabangan rata-rata?

(c) Tunjukkan bahwa untuk pohon bercabang banyak dengan n simpul dan kedalaman d faktor percabangan efektif b kira-kira sama dengan faktor percabangan rata-rata dan dengan demikian sama dengan $\sqrt[d]{n}$.

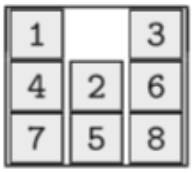
Latihan 6.4

(a) Hitung ukuran ruang keadaan untuk teka-teki 8, untuk teka-teki 3-analog (2 x 2-matriks), serta untuk teka-teki 15 (4 x 4-matriks).

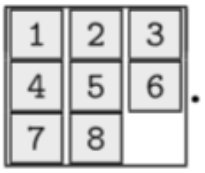
(b) Buktikan bahwa grafik keadaan yang terdiri dari keadaan (simpul) dan tindakan (sisi) untuk 3-puzzle jatuh ke dalam dua sub-grafik yang terhubung, di antaranya tidak ada hubungan.

Latihan 6.5

Dengan pencarian luas pertama untuk teka-teki 8, temukan jalan (secara manual) dari



simpul awal ke simpul tujuan



.

Latihan 6.6

(a) Programkan pencarian luas-pertama, pencarian mendalam-pertama, dan pendalaman berulang dalam bahasa pilihan Anda dan ujilah pada contoh 8 teka-teki.

(b) Mengapa tidak masuk akal untuk menggunakan pencarian mendalam-pertama pada teka-teki 8?

Latihan 6.7

(a) Tunjukkan bahwa pencarian luas-pertama yang diberikan biaya konstan untuk semua tindakan dijamin untuk menemukan solusi terpendek.

(b) Tunjukkan bahwa hal ini tidak berlaku untuk biaya yang bervariasi.

Latihan 6.8

Pendahulu dari semua node harus disimpan untuk memeriksa siklus selama pencarian depth-first.

(a) Untuk depth first search, kembangkan struktur data (bukan tabel hash) yang seefisien mungkin untuk menyimpan semua node dalam jalur pencarian dari pohon pencarian.

(b) Untuk faktor percabangan konstan b dan kedalaman d, berikan rumus untuk ruang penyimpanan yang dibutuhkan oleh pencarian kedalaman-pertama dengan dan tanpa menyimpan pendahulunya.

(c) Tunjukkan bahwa untuk b dan d besar, kita memiliki $\sum_{k=0}^{d} k \ . \ b^k \approx d \ . \ b^d$

Latihan 6.9

Gunakan pencarian A★ untuk teka-teki 8, cari (secara manual) jalur dari simpul awal

ke simpul tujuan

(a) gunakan heuristik h_1 (Bagian 6.3.4).

(b) gunakan heuristik h_2 (Bagian 6.3.4).

Latihan 6.10

Buatlah pohon pencarian A★ untuk grafik kota dari Gambar 6.14 dan gunakan jarak terbang ke Kendal sebagai heuristiknya. Mulai di Solo dengan Kendal sebagai tujuan. Berhati-hatilah agar setiap kota hanya muncul sekali per jalur.

Latihan 6.11

(a) Tunjukkan bahwa pertidaksamaan segitiga berlaku untuk jarak terpendek pada peta.

(b) Dengan menggunakan sebuah contoh, tunjukkan bahwa pertidaksamaan segitiga tidak selalu berlaku untuk simpul-simpul tetangga langsung x dan y, di mana jaraknya adalah d(x, y). Artinya, tidak demikian halnya dengan d(x; y) ≤ d(x; z) + d(z; y).

Latihan 6.12

Program A★ cari dalam bahasa pemrograman pilihan Anda menggunakan heuristik h_1 dan h_2 dan uji ini pada contoh 8 teka-teki.

Latihan 6.13

Berikan fungsi evaluasi heuristik untuk keadaan di mana HEURISTICSEARCH dapat diimplementasikan sebagai pencarian depth-first, dan satu untuk implementasi pencarian broadth-first.

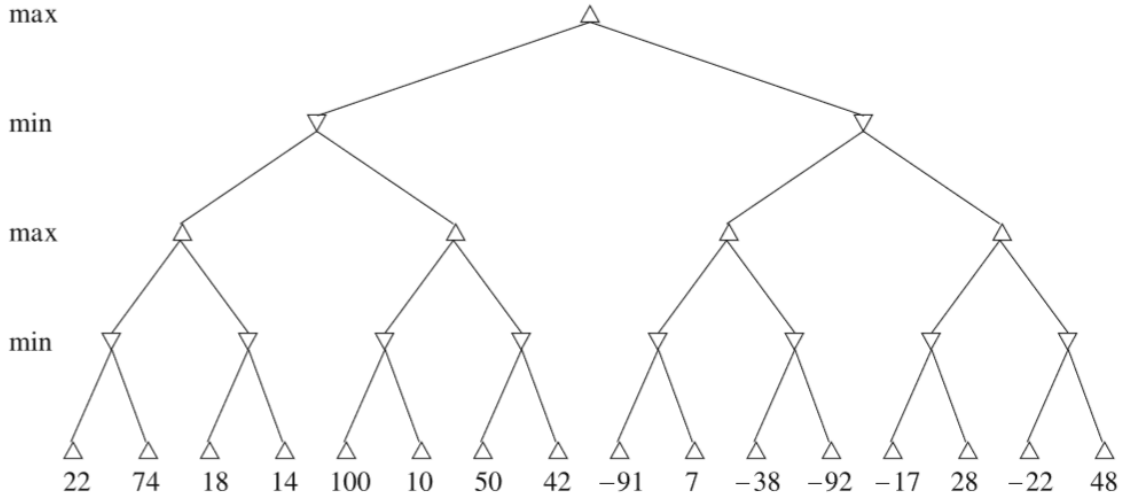
Latihan 6.14

Apa hubungan antara gambar pasangan di ngarai dari Gambar 6.13 dan heuristik yang dapat diterima? Latihan 6.15 Tunjukkan bahwa heuristik h1 dan h2 untuk teka-teki.

Latihan 6.16

(a) Pohon pencarian untuk permainan dua pemain diberikan pada Gambar 6.23 dengan peringkat semua simpul daun. Gunakan pencarian minimax dengan pemangkasan α – β dari kiri ke kanan. Coret semua node yang tidak dikunjungi dan berikan nilai yang dihasilkan optimal untuk setiap node dalam. Tandai jalur yang dipilih.

(b) Uji diri Anda menggunakan applet P. Winston [Win].



Gambar 6.23 Pohon pencarian Minimax

BAB 7
PEMIKIRAN KETIDAKPASTIAN

Kami telah menunjukkan di Bab 4 dengan masalah Tweety bahwa logika dua nilai mengarah pada masalah dalam penalaran sehari-hari. Dalam contoh ini, pernyataan Tweety adalah penguin, Penguin adalah burung, dan Semua burung dapat terbang mengarah pada kesimpulan (yang salah secara semantik) Tweety dapat terbang. Teori probabilitas menyediakan bahasa di mana kita dapat memformalkan pernyataan Hampir semua burung dapat terbang dan melakukan inferensi di atasnya. Teori probabilitas adalah metode yang terbukti dapat kita gunakan di sini karena ketidakpastian tentang apakah burung dapat terbang dapat dimodelkan dengan baik oleh nilai probabilitas. Kami akan menunjukkan, bahwa pernyataan seperti 99% dari semua burung dapat terbang, bersama dengan logika probabilistik, menghasilkan kesimpulan yang benar.

Penalaran di bawah ketidakpastian dengan sumber daya terbatas memainkan peran besar dalam situasi sehari-hari dan juga dalam banyak aplikasi teknis AI. Di area ini proses heuristik sangat penting, seperti yang telah kita bahas dalam Bab. 6.Misalnya, kita menggunakan teknik heuristik ketika mencari tempat parkir di lalu lintas kota. Heuristik saja seringkali tidak cukup, terutama ketika keputusan cepat diperlukan mengingat pengetahuan yang tidak lengkap, seperti yang ditunjukkan pada contoh berikut. Pejalan kaki melintasi jalan dan sebuah mobil dengan cepat mendekat. Untuk mencegah kecelakaan serius, pejalan kaki harus bereaksi cepat. Dia tidak mampu mengkhawatirkan informasi lengkap tentang keadaan dunia, yang dia perlukan untuk algoritma pencarian yang dibahas dalam Bab. 6. Karena itu dia harus mengambil keputusan yang optimal di bawah batasan yang diberikan (sedikit waktu dan sedikit, berpotensi pengetahuan yang tidak pasti). Jika dia berpikir terlalu lama, itu akan berbahaya. Dalam situasi ini dan banyak situasi serupa (lihat Gambar 7.1), diperlukan metode penalaran dengan pengetahuan yang tidak pasti dan tidak lengkap.

Kami ingin menyelidiki berbagai kemungkinan penalaran di bawah ketidakpastian dalam contoh diagnosis medis sederhana. Jika pasien mengalami nyeri di perut kanan bawah dan peningkatan jumlah sel darah putih (leukosit), hal ini menimbulkan kecurigaan bahwa itu mungkin radang usus buntu. Kami memodelkan hubungan ini menggunakan logika proposisional dengan rumus

$$\textit{Sakit perut sebelah kanan bawah} \wedge \textit{Leukosit} > 10000 \Longrightarrow \textit{Apendisitis}$$

Gambar 7.1 "Mari kita duduk dan berpikir tentang apa yang harus dilakukan!"

Jika kita kemudian tahu itu

Sakit perut kanan bawah ∧ Leukosit > 10000

benar, maka kita dapat menggunakan modus ponens untuk menurunkan Apendisitis. Model ini jelas terlalu kasar. Pada tahun 1976, Shortliffe dan Buchanan mengenali ini ketika membangun sistem pakar medis MYCIN [Sho76]. Mereka mengembangkan kalkulus menggunakan apa yang disebut faktor kepastian, yang memungkinkan kepastian fakta dan aturan untuk diwakili. Aturan A → B diberi faktor kepastian β. Semantik aturan A → β B didefinisikan melalui probabilitas bersyarat P (B | A)=β. Dalam contoh di atas, aturan kemudian bisa membaca

Sakit Perut Kanan Bawah ∧ Leukosit > 10.000 → $_{0,6}$ *Apendisitis,*

Untuk penalaran dengan rumus semacam ini, mereka menggunakan kalkulus untuk menghubungkan faktor-faktor aturan. Ternyata, bagaimanapun, bahwa dengan kalkulus ini, hasil yang tidak konsisten dapat diperoleh.

Seperti yang dibahas dalam Bab. 4, ada juga upaya untuk memecahkan masalah ini dengan menggunakan logika non-monotonik dan logika default, yang pada akhirnya tidak berhasil. Teori Dempster–Schäfer memberikan fungsi keyakinan Bel(A) ke proposisi logis A, yang nilainya memberikan tingkat bukti untuk kebenaran A. Tetapi bahkan formalisme ini memiliki kelemahan, yang ditunjukkan pada pengujian sebelumnya menggunakan varian dari contoh Tweety. Bahkan logika fuzzy, yang di atas segalanya berhasil dalam teori kontrol, menunjukkan kelemahan yang cukup besar ketika penalaran di bawah ketidakpastian dalam aplikasi yang lebih kompleks.

Sejak sekitar pertengahan 1980-an, teori probabilitas telah semakin banyak mempengaruhi AI. Di bidang penalaran dengan Bayesiannetworks, atau probabilitas subjektif, ia telah mengamankan dirinya sendiri di tempat yang kuat di antara teknik AI yang sukses. Daripada implikasi seperti yang dikenal dalam logika (implikasi material), probabilitas bersyarat digunakan di sini, yang memodelkan penalaran kausal sehari-hari secara signifikan lebih baik. Penalaran dengan probabilitas sangat diuntungkan dari fakta bahwa teori probabilitas berumur ratusan tahun, cabang matematika yang mapan.

Dalam bab ini kita akan memilih yang elegan, tetapi untuk buku instruksi yang agak tidak biasa, titik masuk ke bidang ini. Setelah pengenalan singkat tentang dasar-dasar terpenting yang diperlukan di sini untuk penalaran dengan probabilitas, kita akan mulai

dengan contoh sederhana, tetapi penting untuk penalaran dengan pengetahuan yang tidak pasti dan tidak lengkap. Dengan cara yang cukup alami dan hampir menarik, kita akan diarahkan ke metode entropi maksimum (MaxEnt). Kemudian kami akan menunjukkan kegunaan metode ini dalam praktik menggunakan sistem pakar medis LEXMED. Akhirnya kami akan memperkenalkan penalaran yang sekarang tersebar luas dengan jaringan Bayesian, dan menunjukkan hubungan antara kedua metode tersebut.

7.1 KOMPUTASI DAN PROBABILITAS

Pembaca yang akrab dengan teori probabilitas dapat melewatkan bagian ini. Untuk semua orang, kami akan memberikan peningkatan cepat. Probabilitas sangat cocok untuk pemodelan penalaran di bawah ketidakpastian. Salah satu alasannya adalah bahwa probabilitas secara intuitif mudah untuk ditafsirkan, yang dapat dilihat pada contoh dasar berikut.

Contoh 7.1

Untuk satu lemparan dadu *game* (percobaan), peluang munculnya "angka ganjil" sama dengan 1/6, sedangkan peluang munculnya "angka ganjil" sama dengan 1/2.

Definisi 7.1

Misalkan adalah himpunan kejadian berhingga untuk suatu eksperimen. Setiap kejadian 2Ω mewakili kemungkinan hasil eksperimen. Jika kejadian-kejadian tersebut saling mengecualikan satu sama lain, tetapi mencakup semua hasil yang mungkin dari upaya tersebut, maka kejadian-kejadian itu disebut kejadian-kejadian elementer.

Contoh 7.2 Untuk satu lemparan satu dadu *game*

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

karena tidak ada dua peristiwa ini yang dapat terjadi secara bersamaan. Oleh karena itu, pelemparan suatu bilangan genap ({2, 4, 6}) bukan merupakan kejadian dasar, demikian pula pengguliran suatu bilangan yang lebih kecil dari lima ({1, 2, 3, 4}) karena {2, 4, 6}∩ {1 , 2, 3, 4} = {2, 4} ≠∅. Diketahui dua kejadian A dan B, A ∪ B juga merupakan kejadian. sendiri dilambangkan dengan kejadian tertentu ∅, dan himpunan kosong ; peristiwa yang mustahil. Berikut ini kita akan menggunakan notasi logika proposisional untuk operasi himpunan. Yaitu, untuk himpunan A ∩ B kita tulis A ∧ B. Ini bukan hanya transformasi sintaksis, tetapi juga benar secara semantik karena perpotongan dua himpunan didefinisikan sebagai

$$x \in A \cap B \Leftrightarrow x \in A \wedge x \cap B$$

Karena ini adalah semantik dari A B, kita dapat dan akan menggunakan notasi ini. Hal ini juga berlaku untuk operasi himpunan lain dan komplemen, dan kita akan, seperti yang ditunjukkan pada tabel berikut, menggunakan notasi logika proposisional untuk mereka juga.

Set notation	Propositional logic	Description
$A \cap B$	$A \wedge B$	intersection / and
$A \cup B$	$A \vee B$	union / or
$\bar{A}$	$\neg A$	complement / negation
Ω	t	certain event / true
$\emptyset$	f	impossible event / false

Gambar 7.2 notasi logika proposional

Variabel yang digunakan di sini (misalnya A, B, dll.) disebut variabel acak dalam teori probabilitas. Kami hanya akan menggunakan variabel peluang diskrit dengan domain terbatas di sini. Variabel bilangan_muka pelemparan dadu adalah diskrit dengan nilai 1, 2, 3, 4, 5,6.Kemungkinan pelemparan limaorasix sama dengan 1/3.Hal ini dapat dijelaskan dengan

P(nomor_wajah ∈{5, 6}) = P(nomor_wajah) = 5 ∨ nomor_wajah = 6) = 1/3:

Konsep probabilitas seharusnya memberi kita deskripsi seobjektif mungkin tentang "keyakinan" atau "keyakinan" kita tentang hasil eksperimen. Semua bilangan dalam selang [0,1] seharusnya mungkin, di mana 0 adalah peluang kejadian yang tidak mungkin dan 1 peluang kejadian tertentu. Kami sampai pada ini dari definisi berikut.

Definisi 7.2

Misalkan 1Ω ={ 1ω1, 1ω2, … , 1ωn} hingga. Tidak ada peristiwa dasar yang disukai, yang berarti bahwa kita mengasumsikan simetri yang terkait dengan frekuensi seberapa sering setiap peristiwa dasar muncul. Peluang P(A) dari kejadian A adalah

$$P(A) = \frac{|A|}{|\Omega|} = \frac{Jumlah \text{ kasus yang menguntungkan untuk A}}{Jumlah \text{ kemungkinan kasus}}$$

Ini segera mengikuti bahwa setiap peristiwa dasar memiliki probabilitas $1/|\Omega|$. Persyaratan bahwa peristiwa elementer memiliki probabilitas yang sama disebut asumsi Laplace dan probabilitas yang dihitung dengan demikian disebut probabilitas Laplace. Definisi ini mencapai batasnya ketika jumlah kejadian elementer menjadi tak terhingga. Karena kita hanya melihat ruang kejadian yang terbatas di sini, hal ini tidak menimbulkan masalah. Untuk menggambarkan peristiwa kami menggunakan variabel dengan jumlah nilai yang sesuai. Misalnya, variabel eye_color dapat mengambil nilai *hijau, biru, coklat. eye-color=blue* kemudian menjelaskan suatu peristiwa karena kita berhadapan dengan proposisi dengan nilai kebenaran t atau f. Untuk variabel biner (boolean), variabel itu sendiri sudah merupakan proposisi. Di sini cukup, misalnya, untuk menulis *P(JohnCalls)* alih-alih *P(JohnCalls =t).*

Contoh 7.3

Dengan definisi ini, peluang munculnya bilangan genap adalah:

$$P(nomor\_wajah) \in \{2,4,6\}) = \frac{|\{2,4,6\}|}{|\{1,2,3,4,5,6\}|} = \frac{3}{6} = \frac{1}{2}$$

Aturan penting berikut mengikuti langsung dari definisi.

Teorema 7.1

1. P(Ω) = 1.
2. P(Ø) = 0, yang berarti bahwa kejadian yang tidak mungkin memiliki probabilitas 0.
3. Untuk kejadian ekslusif berpasanganA dan Bitis benar bahwaP(A∨B)=P(A)+P(B).
4. Untuk dua kejadian komplementer A dan A benar bahwa P(A)+P(¬A) = 1.
5. Untuk kejadian sembarang A dan B benar bahwa P(A ∨ B)=P(A) +P(B) = P(A ∧ B).
6. Untuk A ⊆ B benar bahwa P(A) ≤ P(B).
7. Jika $A_1, ..., A_n$ adalah kejadian elementer, maka $\sum_{i=1}^{n} P(A_i)$ = 1 (kondisi normalisasi).

Ekspresi P(A ∧ B) atau ekuivalen P(A, B) mewakili peluang kejadian A ∧ B. Kita sering tertarik pada peluang semua kejadian elementer, yaitu semua kombinasi semua nilai dari variabel A dan B. Untuk variabel biner A dan B adalah P(A, B), P(A, ¬B), P(¬A,B), P(¬A, ¬B). kita sebut vektor

(P(A, B), P(A, ¬B), P(¬A, B), P(¬A, ¬B))

terdiri dari empat nilai ini suatu distribusi atau distribusi probabilitas gabungan dari variabel A dan B. Singkatan untuk ini adalah P(A, B). Distribusi dalam kasus dua variabel dapat divisualisasikan dengan baik dalam bentuk tabel (matriks), direpresentasikan sebagai berikut:

$\boldsymbol{P}(A, B)$	$B = w$	$B = f$
$A = w$	$P(A, B)$	$P(A, \neg B)$
$A = f$	$P(\neg A, B)$	$P(\neg A, \neg B)$

Untuk variabel d X_1, ..., X_d dengan nilai masing-masing n, distribusinya memiliki nilai P(X_1 =x_1, ..., X_d = x_d) dan x_1,... ,x_d, yang masing-masing mengambil n nilai yang berbeda. Distribusi karena itu dapat direpresentasikan sebagai matriks d-dimensi dengan total elemen n^d. Karena kondisi normalisasi dari Teorema 7.1, bagaimanapun, salah satu dari nilai nd ini adalah redundan dan distribusinya dicirikan oleh nilai unik n^d −1.

7.2 PROBABILITAS BERSYARAT

Contoh 7.4

Di jalan Siliwangi Semarang, kecepatan 100 kendaraan diukur. Untuk setiap pengukuran juga dicatat apakah pengemudinya adalah seorang pelajar. Hasilnya adalah

Even	***Frekuensi***	***Frekuensi relatif***
Observasi kendaraan	100	1
Supir adalah siswa (S)	30	0.3
Kecepatan sangat tinggi (G)	10	0.1
Supir adalah siswa dan ngebut (S ∧ G)	5	0.05

Kami mengajukan pertanyaan: Apakah siswa mempercepat lebih sering daripada rata-rata orang, atau daripada non-siswa?[17]

Jawabannya diberikan oleh probabilitas
Pengemudi adalah seorang pelajar dan ngebut,

untuk ngebut dengan syarat pengemudi adalah pelajar. Ini jelas berbeda dari probabilitas apriori P(G) = 0,1 untuk ngebut. Untuk probabilitas apriori, ruang kejadian tidak dibatasi oleh kondisi tambahan

Definisi 7.3

Untuk dua kejadian A dan B, peluang P(A|B) untuk A pada kondisi B (probabilitas bersyarat) ditentukan oleh

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

Dalam Contoh 7.4 kita melihat bahwa dalam kasus ruang kejadian berhingga, probabilitas bersyarat P(A|B) dapat dipahami sebagai probabilitas A ∧ B jika kita hanya melihat kejadian B, yaitu sebagai

$$P(A|B) = \frac{|A \wedge B|}{|B|}$$

Rumus ini dapat dengan mudah diturunkan menggunakan Definisi 7.2

[17] Probabilitas yang dihitung hanya dapat digunakan untuk proposisi lanjutan jika sampel yang diukur (100 kendaraan) representatif. Jika tidak, hanya proposisi tentang 100 kendaraan yang diamati yang dapat dibuat.

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{\frac{|A \wedge B|}{|\Omega|}}{\frac{|B|}{|\Omega|}} = \frac{|A \wedge B|}{|B|}$$

Definisi 7.4
Jika, untuk dua kejadian A dan B,

$$P(A|B) = P(A)$$

maka peristiwa ini disebut independen.
Jadi A dan B saling bebas jika peluang kejadian A tidak dipengaruhi oleh kejadian B.

Teorema 7.2
Untuk kejadian bebas A dan B, berikut dari definisi bahwa

$$P(A \wedge B) = P(A) \ . \ P(B)$$

Contoh 7.5
Untuk pelemparan dua buah dadu, peluang munculnya dua dadu berenam adalah 1/36 jika kedua dadu saling bebas karena

$$P(D_1 = 6 \wedge D_2 = 6) = P(D_1 = 6) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$$

di mana persamaan pertama hanya benar jika kedua dadu saling bebas. Jika misalnya oleh beberapa kekuatan sihir mati 2 selalu sama dengan mati 1, maka

$$P(D_1 = 6 \wedge D_2 = 6) = \frac{1}{6}$$

7.3 ATURAN RANTAI

Memecahkan definisi probabilitas bersyarat untuk P(A ∧ B) menghasilkan apa yang disebut aturan produk

$$P(A \wedge B) = P(A|B)P(B)$$

yang segera kita generalisasi untuk kasus n variabel. Dengan penerapan berulang dari aturan di atas kita memperoleh aturan rantai
Persamaan 7.1

$$P(X_1, \ldots\ldots, X_n) = P(x_n|x_1, \ldots\ldots, x_{n-1}) * P(X_1, \ldots\ldots, X_{n-1})$$
$$= P(x_n|x_1, \ldots\ldots, x_{n-1}) * P(x_{n-1}|x_1, \ldots\ldots, x_{n-2}) * P(X_1, \ldots\ldots, X_{n-2})$$
$$= P(x_n|x_1, \ldots\ldots, x_{n-1}) * (x_{n-1}|x_1, \ldots\ldots, x_{n-2}), \ldots\ldots, P(x_n|x_1) * P(x_1)$$
$$= \prod_{i=1}^{n} P(x_n|x_1, \ldots\ldots, x_{i-1})$$

dengan mana kita dapat mewakili distribusi sebagai produk dari probabilitas bersyarat. Karena aturan rantai berlaku untuk semua nilai variabel $X_1, \ldots$, X_n, maka telah dirumuskan distribusinya menggunakan simbol ***P***.

7.4 MARGINALISASI

Karena A ⇔ (A ∧ B) ∨ (A ∧ ¬B) benar untuk variabel biner A dan B

$$P(A) = P((A \wedge B) \vee (A \wedge \neg B)) = P(A \wedge B) + P(A \wedge \neg B)$$

Dengan menjumlahkan dua nilai B, variabel B dihilangkan. Secara analog, untuk variabel arbitrer $X_1, \ldots$, X_d, sebuah variabel, misalnya X_d, dapat dihilangkan dengan menjumlahkan semua variabelnya dan kita dapatkan penerapan rumus ini disebut marginalisasi.

Penjumlahan ini dapat dilanjutkan dengan variabel X_1, … ,X_{d-1} sampai hanya satu variabel yang tersisa. Marginalisasi juga dapat diterapkan pada distribusi P(X_1, … ,X_d). Distribusi yang dihasilkan P(X_1, … , X_{d-1}) disebut distribusi marginal. Ini sebanding dengan proyeksi balok persegi panjang pada permukaan datar. Di sini objek tiga dimensi digambar di tepi atau "pinggiran" kubus, yaitu pada himpunan dua dimensi. Dalam kedua kasus, dimensi berkurang satu.

Contoh 7.6

Kami mengamati himpunan semua pasien yang datang ke dokter dengan nyeri perut akut. Untuk setiap pasien, nilai leukosit diukur, yang merupakan metrik untuk kelimpahan relatif sel darah putih dalam darah. Kami mendefinisikan variabel Leuko, yang benar jika dan hanya jika nilai leukosit lebih besar dari 10.000. Ini menandakan adanya infeksi di dalam tubuh. Jika tidak, kami mendefinisikan variabel App, yang memberi tahu kami apakah pasien menderita radang usus buntu, yaitu usus buntu yang terinfeksi. Distribusi *P(App, Leuko)* dari kedua variabel tersebut diberikan dalam tabel berikut:

P(*App*, *Leuko*)	*App*	¬*App*	Total
Leuko	0.23	0.31	0.54
¬*Leuko*	0.05	0.41	0.46
Total	0.28	0.72	1

Di baris terakhir jumlah baris diberikan, dan di kolom terakhir jumlah kolom diberikan. Jumlah ini diperoleh dengan marginalisasi. Misalnya, kita membaca

P(Leuko) = P(App, Leuko) + P(¬App, Leuko) = 0,54

Distribusi P(App, Leuko) yang diberikan dapat berasal dari survei dokter Jerman, misalnya. Dari situ kita kemudian dapat menghitung probabilitas bersyarat

$$P(Leuko|App) = \frac{P(Leuko, App)}{P(App)} = 0.82$$

yang memberitahu kita bahwa sekitar 82% dari semua kasus radang usus buntu menyebabkan nilai leukosit yang tinggi. Nilai-nilai seperti ini diterbitkan dalam literatur medis. Namun, probabilitas bersyarat *P(App|Leuko),* yang sebenarnya akan jauh lebih membantu untuk mendiagnosis apendisitis, tidak dipublikasikan. Untuk memahami ini, pertama-tama kita akan mendapatkan rumus yang sederhana, tetapi sangat penting.

Teorema Bayes

Menukar A dan B dalam Definisi 7.3 menghasilkan

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} \qquad dan \qquad P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

Dengan menyelesaikan kedua persamaan untuk P(A ∧ B) dan menyamakannya, kita memperoleh **teorema Bayes**

Persamaan 7.2

$$P(A|B) = \frac{P(B|A) \; . \; P(A)}{P(B)}$$

yang relevansinya dengan banyak aplikasi akan kami ilustrasikan menggunakan tiga contoh. Pertama kita terapkan pada contoh usus buntu dan dapatkan

Contoh 7.7

Persamaan 7.3

$$P(app|Leuko) = \frac{P(Leuko|App) \; . \; P(App)}{P(Leuko)} = \frac{0.82 \; . \; 0.28}{0.54} = 0.43$$

Lalu mengapa *P(Leuko|App)* diterbitkan, tetapi bukan *P(App|Leuko)?*

Dengan asumsi bahwa radang usus buntu mempengaruhi biologi semua manusia sama, terlepas dari etnis, PðLeukojApp adalah nilai universal yang berlaku di seluruh dunia. Dalam Persamaan 7.3 kita melihat bahwa *P(App|Leuko)* tidak universal, karena nilai ini dipengaruhi oleh probabilitas apriori *P(App)* dan *P(Leuko).* Masing-masing dapat bervariasi sesuai dengan keadaan kehidupan seseorang. Misalnya, P(Leuko) bergantung pada apakah suatu populasi memiliki tingkat keterpaparan yang tinggi atau rendah terhadap penyakit menular. Di daerah tropis, nilai ini dapat berbeda secara signifikan dari daerah dingin. Teorema Bayes, bagaimanapun, memudahkan kita untuk mengambil nilai yang valid secara universal *P(Leuko|App)*, dan menghitung *P(App|Leuko*) yang berguna untuk diagnosis.

Sebelum kita menyelam lebih dalam ke contoh ini dan membangun sistem pakar medis untuk radang usus buntu di bagian sebelumnya, pertama-tama mari kita terapkan teorema Bayes pada contoh medis lain yang menarik.

Contoh 7.8

Dalam diagnosis kanker, apa yang disebut penanda tumor sering diukur. Salah satu contohnya adalah penggunaan penanda tumor PSA (prostate spesific antigen) untuk diagnosis kanker prostat (PCa = kanker prostat) pada pria. Dengan asumsi bahwa tidak ada tes lebih lanjut untuk PCa telah dilakukan, tes dianggap positif, yaitu ada dugaan PCa, jika konsentrasi PSA mencapai tingkat pada atau di atas 4 ng/ml. Jika ini terjadi, probabilitas P(C|pos) PCa menarik bagi pasien.

Variabel biner C benar jika pasien memiliki PCa, dan pos mewakili nilai PSA 4 ≥ ng/ml. Sekarang mari kita hitung jpos PC . Untuk alasan yang mirip dengan yang disebutkan untuk diagnosis apendisitis, nilai ini tidak dilaporkan. Sebaliknya, peneliti menerbitkan sensitivitas P posjCðÞdan spesifisitas P(neg|¬C) dari tes.[18] Untuk sensitivitas 0,95, spesifisitas paling banyak 0,25, itulah sebabnya kami melanjutkan dari P(pos|C) = 0,95 dan P(neg|C) = 0,25 di bawah. Kami menerapkan teorema Bayes dan memperoleh RMG 148.

Di sini kita menggunakan P(pos|¬C) =1 – P(neg|C) = 1 – 0,25 = 0,75 dan P(C) = 0,0021 = 0,21% sebagai probabilitas apriori PCa selama satu tahun.[19] Masuk akal untuk mengasumsikan bahwa tes PSA dilakukan setahun sekali . Hasil ini agak mengejutkan dari sudut pandang pasien karena probabilitas PCa setelah tes positif, pada 0,27%, hanya sedikit lebih tinggi daripada probabilitas 0,21% untuk PCa untuk pria berusia 55 tahun. Jadi, nilai PSA lebih dari 4 ng/ml jelas bukan alasan bagi pasien untuk panik. Paling-paling digunakan sebagai dasar untuk pemeriksaan lebih lanjut, seperti biopsi atau MRI, jika perlu mengarah ke radiasi dan pembedahan. Situasinya serupa untuk banyak penanda tumor lainnya seperti untuk kanker kolorektal atau diagnosis kanker payudara dengan mamografi.

Penyebab masalah ini adalah spesifisitas yang sangat rendah P(neg|C) = 0,25, yang menyebabkan 75% pasien sehat (tanpa PCa) mendapatkan hasil tes positif palsu dan akibatnya menjalani pemeriksaan yang tidak perlu. Karena itu, pengujian PSA telah menjadi topik diskusi kontroversial selama bertahun-tahun.[20]

Asumsikan kita memiliki tes yang lebih baik dengan spesifisitas 99%, yang hanya akan memberikan hasil positif palsu untuk satu persen pria sehat. Kemudian, dalam perhitungan di atas, kita akan menetapkan P(pos|C) nilai 0,01 dan memperoleh hasil P(C|pos) = 0,17. Jelas, tes ini akan jauh lebih spesifik.

Contoh 7.9

[18] Untuk definisi sensitivitas dan spesifisitas lihat Persamaan. 7.16 dan 7.17.

[19] Lihat http://www.prostata.de/pca_haeufigkeit.html untuk usia 55- tahun-orang tua.

[20] Penulis bukanlah seorang dokter medis. Oleh karena itu perhitungan ini tidak boleh digunakan sebagai dasar untuk keputusan medis pribadi oleh individu yang berpotensi terkena dampak. Jika perlu, silakan berkonsultasi dengan dokter spesialis atau literatur spesialis yang relevan.

Seorang perwakilan penjualan yang ingin menjual sistem alarm dapat membuat argumen berikut:

> *Jika Anda membeli sistem alarm yang sangat andal ini, itu akan mengingatkan Anda tentang pembobolan dengan kepastian 99%. Sistem pesaing kami hanya menawarkan kepastian sebesar 85%.*

Mendengar ini, jika pembeli menyimpulkan bahwa dari peringatan A dia dapat menyimpulkan pembobolan B dengan kepastian tinggi, dia salah. Teorema Bayes menunjukkan alasannya. Apa yang dikatakan perwakilan tersebut adalah bahwa P(A|B) = 0,99. Apa yang tidak dia katakan, bagaimanapun, apa artinya ketika kita mendengar alarm berbunyi. Untuk mengetahuinya, kami menggunakan teorema Bayes untuk menghitung P(B|A) dan mengasumsikan bahwa pembeli tinggal di daerah yang relatif aman di mana pembobolan jarang terjadi, dengan P(B) = 0,001. Selain itu, kami berasumsi bahwa sistem alarm dipicu tidak hanya oleh pencuri, tetapi juga oleh hewan, seperti burung atau kucing di halaman, yang menghasilkan P(A) = 0,1. Dengan demikian kita peroleh

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)} = \frac{0.00 \;.\; 0.01}{0.1} = 0.01$$

yang berarti bahwa siapa pun yang membeli sistem ini tidak akan senang karena mereka akan dikejutkan oleh terlalu banyak alarm palsu. Ketika kita memeriksa penyebutnya

$$P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B) = 0.00099 + P(A|\neg B) \;.\; 0.999 = 0.1$$

dari teorema Bayes lebih dekat, kita melihat bahwa P(A|¬B) ≈ 0,1, yang berarti bahwa alarm akan dipicu kira-kira setiap hari kesepuluh jika tidak ada pembobolan. Dari contoh ini kita belajar, antara lain, bahwa penting untuk mempertimbangkan probabilitas mana yang benar-benar kita minati sebagai pembeli, terutama dalam hal keamanan. Ketika argumen probabilitas bersyarat dipertukarkan, nilainya dapat berubah secara dramatis ketika probabilitas sebelumnya berbeda secara signifikan.

7.5 PRINSIP ENTROPI MAKSIMUM

Kami sekarang akan menunjukkan, dengan menggunakan contoh inferensi, bahwa kalkulus untuk penalaran di bawah ketidakpastian dapat direalisasikan dengan menggunakan teori probabilitas. Namun, kita akan segera melihat bahwa jalur probabilistik yang sudah usang akan segera berakhir. Khususnya, ketika terlalu sedikit pengetahuan yang tersedia untuk memecahkan persamaan yang diperlukan, ide-ide baru diperlukan. Fisikawan Amerika E.T. Jaynes melakukan pekerjaan perintis di bidang ini pada 1950-an. Dia mengklaim bahwa dengan pengetahuan yang hilang, seseorang dapat memaksimalkan entropi dari distribusi probabilitas yang diinginkan, dan menerapkan prinsip ini ke banyak contoh.

Aturan Inferensi untuk Probabilitas

Kami ingin menurunkan aturan inferensi untuk pengetahuan yang tidak pasti yang analog dengan modus ponens. Dari pengetahuan tentang proposisi A dan aturan A ⇒ B, kesimpulan B harus dicapai. Diformulasikan secara ringkas, ini berbunyi

$$\frac{A, A \rightarrow B}{B}$$

Generalisasi untuk aturan probabilitas menghasilkan

$$\frac{P(A) = \alpha,\; P(B|A) = \beta}{P(B) = ?}$$

Biarkan dua aturan probabilitas α, β diberikan dan nilai P(B) yang diinginkan. Dengan marginalisasi kita memperoleh distribusi marginal yang diinginkan

$$P(B) = P(A, B) + P(\neg A, B) = P(B|A) \;.\; P(A) + P(B|\neg A)) \;.\; P(\neg A)$$

Tiga nilai P(A), P(¬A), P(B|A) di ruas kanan diketahui, tetapi nilai P (B|¬A) tidak diketahui. Kita tidak dapat membuat pernyataan eksak tentang P(B) dengan teori probabilitas klasik, tetapi paling banyak kita dapat memperkirakan P(B) ≥ P(B|A). P(A). Sekarang kita pertimbangkan distribusi

(P(A, B) = (P(A, B), P(A, ¬B), P(¬A, B), P(¬A, ¬B)

dan memperkenalkan singkatan empat yang tidak diketahui

P_1 = P(A, B)

P_2 = P(A, ¬B)

P_3 = P(¬A, B)

P_4 = P(¬A, ¬B)

Keempat parameter ini menentukan distribusi. Jika semuanya diketahui, maka setiap peluang untuk kedua variabel A dan B dapat dihitung. Untuk menghitung empat yang tidak diketahui, diperlukan empat persamaan. Satu persamaan sudah diketahui dalam bentuk kondisi normalisasi

$$P_1 + P_2 + P_3 + P_4 = 1$$

Oleh karena itu, diperlukan tiga persamaan lagi. Namun, dalam contoh kita, hanya dua persamaan yang diketahui. Dari nilai yang diberikan P(A) = α dan P(A|B) = β kami menghitung

P(A, B) = P(B|A) . P(A) = αβ

dan

P(A) = P(A, B) + P(A, ¬B)

Dari sini kita dapat mengatur sistem persamaan berikut dan menyelesaikannya sejauh mungkin:

Persamaan 7.4

P1 = αβ

Persamaan 7.5

P1 + P2 = α

Persamaan 7.6

$P_1 + P_2 + P_3 + P_4 = 1$

Persamaan 7.7

Persamaan 7.4 pada Persamaan 7.5 : P2 = α – αβ = α(1 – β)

Persamaan 7.8

Persamaan 7.5 pada 7.6 : P3 + P4 = 1 - α

Probabilitas p_1, p_2 untuk interpretasi (A, B) dan (A, ¬B) diketahui, tetapi untuk nilai p_3, p_4 hanya satu persamaan yang tersisa. Untuk sampai pada solusi pasti terlepas dari pengetahuan yang hilang ini, kami mengubah sudut pandang kami. Kami menggunakan persamaan yang diberikan sebagai kendala untuk solusi dari masalah optimasi.

Kami mencari distribusi p (untuk variabel p_3, p_4) yang memaksimalkan entropi

Persamaan 7.9

$$H(P) = -\sum_{i=1}^{n} p_i \, pada \; p_i = -p_i \, pada \, p_3 - p_4 \, pada \, p_4$$

di bawah batasan p3 + p4 =1−α (7.8). Mengapa tepatnya fungsi entropi harus dimaksimalkan? Karena kita kehilangan informasi tentang distribusi, entah bagaimana harus ditambahkan. Kita bisa memperbaiki nilai ad hoc, misalnya p_3 = 0,1. Namun lebih baik untuk menentukan nilai p_3 dan p_4 sedemikian rupa sehingga informasi yang ditambahkan minimal. Kita dapat menunjukkan bahwa entropi mengukur ketidakpastian distribusi hingga faktor konstan. Entropi negatif kemudian menjadi ukuran jumlah informasi yang

dikandung oleh suatu distribusi. Maksimalisasi entropi meminimalkan isi informasi dari distribusi. Untuk memvisualisasikannya, fungsi entropi untuk kasus dua dimensi direpresentasikan secara grafis pada Gambar 7.2.

Untuk menentukan entropi maksimum di bawah kendala $p_3 + p_4 - 1 + \alpha = 0$ kita menggunakan metode pengali Lagrange [Ste07].Fungsi Lagrange membaca

$$L = -p_3 \text{ pada } p_3 - p_4 \text{ pada } p_4 + \lambda(p_3 + p_4 - 1 + \alpha)$$

Mengambil turunan parsial terhadap p3 dan p4 kita peroleh

$$\frac{\partial L}{\partial p_3} = -\,pada\, p_3 \; - \; 1 \; + \; \lambda \; = \; 0,$$

$$\frac{\partial L}{\partial p_4} = -\,pada\, p_4 \; - \; 1 \; + \; \lambda \; = \; 0,$$

dan hitung

$$p_3 = p_{4=} \frac{1 \; - \; \alpha}{2}$$

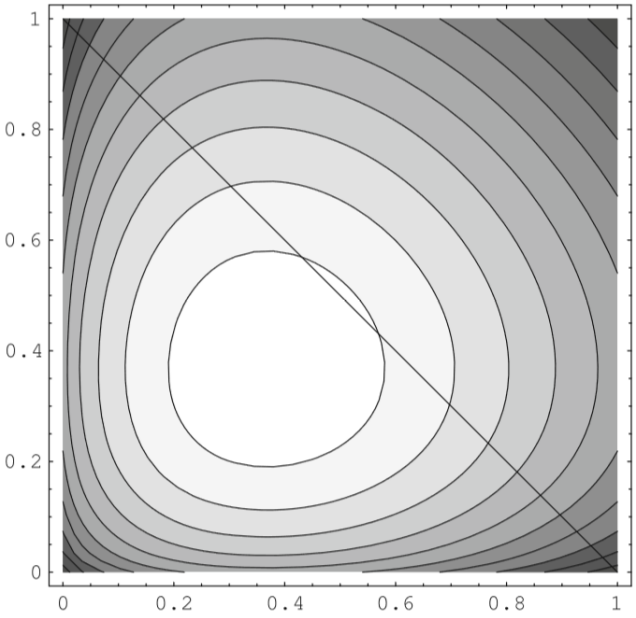
Sekarang kita dapat menghitung nilai yang diinginkan

$$P(B) \; = \; P(A,B) \; + \; P(\neg A,B) \; = \; p_1 + p_3 = \alpha\beta \; + \; \frac{1 \; - \; \alpha}{2} \; = \; \alpha(\beta \; - \; \frac{1}{2}) \; + \; \frac{1}{2}$$

Substitusi ke α dan β menghasilkan

$$P(B) \; = \; P(A) \; \left(P(B|A) \; - \; \frac{1}{2}\right) \; + \; \frac{1}{2}$$

P(B) ditunjukkan pada Gambar 7.3 untuk berbagai nilai P(B|A) . Kita melihat bahwa dalam kasus tepi dua nilai, yaitu, ketika P(B) dan P(B|A) mengambil nilai 0 atau 1, inferensi probabilistik mengembalikan nilai yang sama untuk P(B) sebagai modus ponens. Jika A dan B|A keduanya benar, B juga benar. Kasus yang menarik adalah P(A) = 0, di mana ¬A benar. Modus ponens tidak dapat diterapkan di sini, tetapi rumus kami menghasilkan nilai 1/2 untuk P(B) terlepas dari P(B|A).



Gambar 7.3 Diagram garis kontur dari fungsi entropi dua dimensi.

Kita melihat bahwa itu benar-benar cembung di seluruh unit persegi dan memiliki maksimum global yang terisolasi. Juga ditandai adalah kendala $p_3 + p_4 = 1$ sebagai kasus khusus dari kondisi $p_3 + p_4 - 1 + \alpha = 0$ untuk $\alpha = 0$ yang relevan di sana.

Ketika A salah, kita tidak tahu apa-apa tentang B, yang mencerminkan intuisi kita secara tepat. Kasus di mana P(A) = 1 dan P(B|A) = 0 juga dicakup oleh logika proposisional. Di sini A benar dan A $\Longrightarrow$ B salah, dan dengan demikian A $\wedge \neg$ B benar. Maka B salah. Garis horizontal pada gambar berarti bahwa kita tidak dapat membuat prediksi tentang B dalam kasus P(B|A)= 1/2. Di antara titik-titik ini, P (B) berubah secara linier untuk perubahan P(A) atau P(B|A).

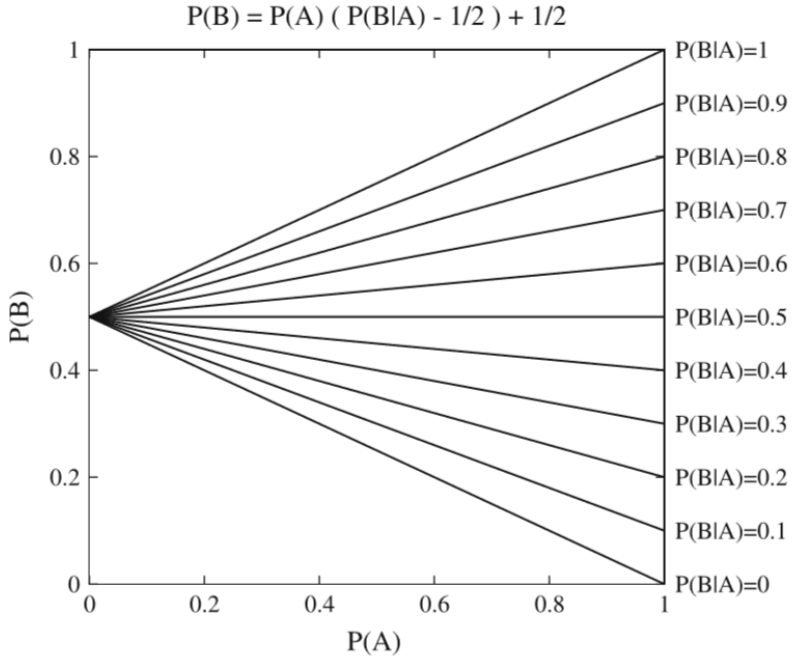
Teorema 7.3
Biarkan ada satu set persamaan probabilistik linier yang konsisten[21]. Kemudian ada maksimum unik untuk fungsi entropi dengan persamaan yang diberikan sebagai kendala. Distribusi MaxEnt dengan demikian didefinisikan memiliki konten informasi minimum di bawah batasan.

Ini mengikuti dari teorema ini bahwa tidak ada distribusi yang memenuhi kendala dan memiliki entropi lebih tinggi daripada distribusi MaxEnt. Kalkulus yang mengarah ke entropi yang lebih rendah memasukkan informasi ad hoc tambahan, yang tidak dibenarkan. Melihat lebih dekat pada perhitungan P(B) di atas, kita melihat bahwa kedua nilai p_3 dan p_4 selalu muncul secara simetris. Ini berarti bahwa menukar kedua variabel tidak mengubah hasilnya. Jadi hasil akhirnya adalah $p_3 = p_4$. Apa yang disebut ketidakpedulian dari dua variabel ini menyebabkan mereka disetel sama oleh MaxEnt. Hubungan ini berlaku secara umum:

Definisi 7.5
Jika pertukaran sembarang dua variabel atau lebih dalam persamaan Lagrange menghasilkan persamaan ekuivalen, variabel ini disebut indiferen.

Teorema 7.4
Jika himpunan variabel {p_1, ..., p_{ik}}indiferen, maka maksimum entropi di bawah kendala yang diberikan adalah pada titik di mana pi1 = p_{i2} = p_{ik}.
Dengan pengetahuan ini kita dapat dengan segera menetapkan kedua variabel p3 dan p4 sama (tanpa menyelesaikan persamaan Lagrange).

[21] Himpunan persamaan probabilistik disebut konsisten jika terdapat setidaknya satu solusi, yaitu satu distribusi yang memenuhi semua persamaan.



Gambar 7.4 Array kurva untuk P(B) sebagai fungsi P(A) untuk nilai P(B|A) yang berbeda

7.6 ENTROPI MAKSIMUM TANPA BATASAN EKSPLISIT

Kita sekarang melihat kasus di mana tidak ada pengetahuan yang diberikan. Artinya, selain kondisi normalisasi

$$P_1 + p_2 + ... + p_n = 1$$

tidak ada kendala. Semua variabel karena itu acuh tak acuh. Oleh karena itu, kita dapat mengaturnya sama dan mengikuti bahwa $p_1 = p_2 = = p_n = 1/ n$.[22] Untuk penalaran di bawah ketidakpastian, ini berarti bahwa dengan kurangnya pengetahuan, semua dunia memiliki kemungkinan yang sama. Artinya, distribusinya seragam. Misalnya, dalam kasus dua variabel A dan B akan menjadi kasus bahwa

$$P(A, B) = P(A, \neg B) = P(\neg A, B) = P(\neg A, \neg B\_ = 1/4$$

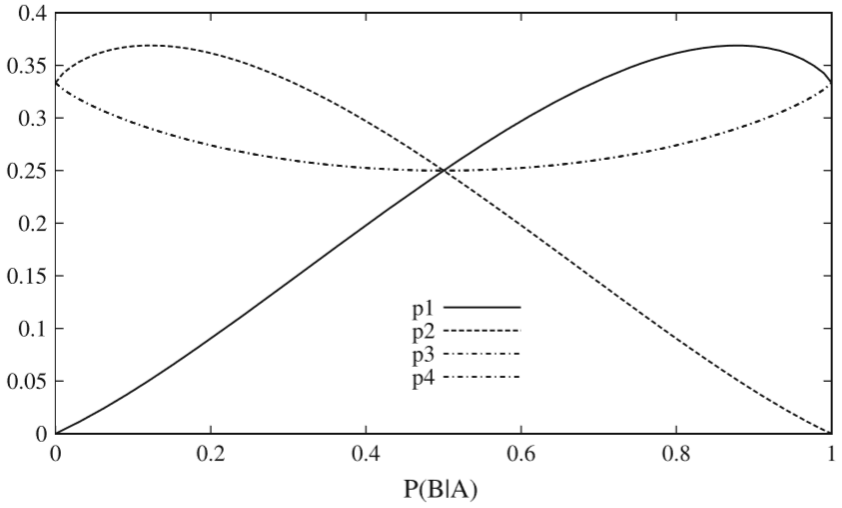
rom yang diikuti oleh P(A)=P(B) = 1/2 dan P(B|A) = 1/2. Hasil untuk kasus dua dimensi dapat dilihat pada Gambar 7.2 karena kondisi yang ditandai sama persis dengan kondisi normalisasi. Kita melihat bahwa entropi maksimum terletak pada garis tepat (1/2, 1/2). Segera setelah nilai suatu kondisi menyimpang dari yang diturunkan dari distribusi seragam, probabilitas dunia bergeser. Kami menunjukkan ini dalam contoh lebih lanjut. Dengan deskripsi yang sama seperti yang digunakan di atas, kami berasumsi bahwa hanya

$$P(B|A) = \beta$$

diketahui. Jadi P(A, B) =P(B|A)P(A) = βP(A) dari $p_1 = (p_1 + p_2)$ berikut dan kami menurunkan dua kendala

$$\beta p_2 + (\beta - 1) = 0$$
$$p_1 + p_2 + p_3 + p_4 - 1 = 0$$

[22] Pembaca dapat menghitung hasil ini dengan memaksimalkan entropi pada kondisi normalisasi (Latihan 7.5).



Gambar 7.4 p_1, p_2, p_3, p_4 bergantung pada β

Di sini persamaan Lagrange tidak bisa lagi diselesaikan secara simbolis dengan begitu mudah. Solusi numerik dari persamaan Lagrange menghasilkan gambar yang ditunjukkan pada Gambar 7.4, yang menunjukkan bahwa $p_3 = p_4$. Kita sudah dapat melihat ini dalam kendala, di mana p_3 dan p_4 indiferen. Untuk P(B|A) =1/2 kita mendapatkan distribusi seragam, yang tidak mengejutkan. Ini berarti bahwa kendala untuk nilai ini tidak menyiratkan pembatasan pada distribusi. Selanjutnya, kita dapat melihat bahwa untuk P(B|A) kecil, P(A, B) juga kecil.

7.7 PROBABILITAS BERSYARAT VERSUS IMPLIKASI MATERIAL

Kami sekarang akan menunjukkan bahwa, untuk penalaran pemodelan, probabilitas bersyarat lebih baik daripada apa yang dikenal dalam logika sebagai implikasi material. Pertama kita mengamati tabel kebenaran yang ditunjukkan pada Tabel 7.1, di mana probabilitas bersyarat dan implikasi material untuk kasus ekstrim probabilitas nol dan satu dibandingkan. Dalam kedua kasus dengan premis palsu (yang, secara intuitif, adalah kasus kritis), P(B|A) tidak terdefinisi, yang masuk akal.

Tabel 7.1 Tabel kebenaran implikasi material dan probabilitas bersyarat untuk limit logika proposisional

A	*B*	$A \Longrightarrow B$	*P(A)*	*P(B)*	*P(B|A)*
t	*t*	*t*	*1*	*1*	*1*
t	*f*	*f*	*1*	*0*	*0*
f	*t*	*t*	*0*	*1*	*Tak terdefinisi*
f	*f*	*t*	*0*	*0*	*Tak terdefinisi*

Sekarang kita bertanya pada diri kita sendiri nilai mana yang diambil oleh PB jA ketika nilai arbitrer P (A) = α dan P(B) = γ diberikan dan tidak ada informasi lain yang diketahui. Sekali lagi kami memaksimalkan entropi di bawah batasan yang diberikan. Seperti di atas kami mengatur

$$P_1 = P(A, B), \quad p_2 = P(A, \neg B), \quad p_3 = P(\neg A, B), \quad p_4 = P(\neg A, \neg B)$$

dan dapatkan sebagai kendala

Persamaan 7.10

$$P_1 + p_2 = \alpha$$

Persamaan 7.11

$$P_1 + p_3 = \gamma$$

Persamaan 7.12

$$P_1 + p_2 + p_3 + p_4 = 1$$

Dengan ini kami menghitung menggunakan maksimalisasi entropi (lihat Latihan 7.8)

$P_1 = \alpha\gamma$, $p_2 = \alpha(1 - \gamma)$, $p_3 = \gamma (1 - \alpha)$, $p_4 = (1 - \alpha) (1 - \gamma)$

Dari $p_1 = \alpha\gamma$ diperoleh P(A, B)=P(A) **.** P(B), yang berarti A dan B saling bebas. Karena tidak ada batasan yang menghubungkan A dan B, prinsip MaxEnt menghasilkan independensi variabel-variabel ini. Bagian kanan Tabel 7.1 membuatnya lebih mudah dipahami. Dari definisi

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

berikut untuk kasus P(A) ≠ 0, yaitu, ketika premis tidak salah, karena A dan B independen, P(B|A) = P(B). Untuk kasus P(A) = 0, P(B|A) tetap didenda.

7.8 MAXENT-SYSTEM

Seperti disebutkan sebelumnya, karena nonlinier dari fungsi entropi, optimasi MaxEnt biasanya tidak dapat dilakukan secara simbolis untuk masalah non-sepele. Jadi dua sistem dikembangkan untuk maksimalisasi entropi numerik. Sistem pertama, SPIRIT (Symmetrical Probabilistic Intensional Reasoning in Inference Networks in Transition, www.xspirit.de), dibangun di Fernuniversität Hagen. Yang kedua, PIT (Alat Induksi Probabilitas) dikembangkan di Universitas Teknik Munich. Kami sekarang akan memperkenalkan PIT secara singkat.

Sistem PIT menggunakan metode pemrograman kuadratik sekuensial (SQP) untuk menemukan ekstrem dari fungsi entropi di bawah batasan yang diberikan. Sebagai input, PIT mengharapkan data yang mengandung kendala. Misalnya, kendala P(A)=α dan P(B|A)=β memiliki bentuk

```
var A{t,f}, B{t,f};

P([A=t]) = 0.6;
P([B=t] | [A=t]) = 0.3;

QP([B=t]);
QP([B=t] | [A=t]);
```

Karena PIT melakukan perhitungan numerik, kita harus memasukkan nilai probabilitas eksplisit. Baris kedua hingga terakhir berisi kueri QP([B = t]). Ini berarti bahwa P(B) adalah nilai yang diinginkan. Di www.pit-systems.deunder“Contoh”kita sekarang memasukkan input ini ke halaman input kosong (“Halaman Kosong”) dan memulai PIT. Akibatnya kita mendapatkan

No	nilai kebenaran	Probabilitas	Kueri
1	TIDAK TERPESIFIKASI	3,80E-02	QP([b = t]);
2.	TIDAK TERPESIFIKASI	3,00E-02	QP([A = t]-|> [B = t]);

.dan dari sana baca P(B) = 0,38 dan P(B|A) = 0,3.

Contoh Tweety

Kami sekarang menunjukkan, menggunakan contoh *Tweety*, bahwa penalaran probabilistik dan khususnya MaxEnt tidak monoton dan memodelkan penalaran sehari-hari dengan sangat baik. Kami memodelkan aturan yang relevan dengan probabilitas sebagai berikut:

P(burung|penguin)=1 *''penguin adalah burung''*
P(lalat|burung) ∈ 2 [0.95,1] *''(hampir semua) burung bisa terbang''*
P(lalat|penguin)=0 *''penguin tidak bisa terbang''*

Aturan pertama dan ketiga mewakili prediksi yang pasti, yang juga dapat dengan mudah dirumuskan dalam logika. Namun, di bagian kedua, kami mengungkapkan pengetahuan kami bahwa hampir semua burung dapat terbang melalui interval probabilitas. Dengan data input PIT

```
var penguin{yes,no}, bird{yes,no}, flies{yes,no};

P([bird=yes] | [penguin=yes]) = 1;
P([flies=yes] | [bird=yes]) IN [0.95,1];
P([flies=yes] | [penguin=yes]) = 0;

QP([flies=yes]| [penguin=yes]);
```

kami mendapatkan kembali jawaban yang benar

No	nilai kebenaran	Probabilitas	Kueri
1	TIDAK TERPESIFIKASI	0.000e+00	QP([pInguin = yes]-|> [flies = yes]);

dengan proposisi bahwa penguin tidak bisa terbang.[23] Penjelasannya sangat sederhana. Dengan P(lalat|burung) ∈ [0.95, 1] ada kemungkinan ada burung yang tidak terbang. Jika aturan ini diganti dengan P(flies|bird) = 1, maka PIT tidak akan dapat melakukan apa pun dan akan menampilkan pesan kesalahan tentang batasan yang tidak konsisten.

Dalam contoh ini kita dapat dengan mudah melihat bahwa interval probabilitas seringkali sangat membantu untuk memodelkan ketidaktahuan kita tentang nilai probabilitas yang tepat. Kita bisa saja membuat formulasi yang lebih kabur dari aturan kedua dalam semangat "biasanya burung y" dengan P(flies|bird) ∈ (0,5, 1] Penggunaan interval setengah terbuka tidak termasuk nilai 0,5.

Pada penelitian sebelumnya, bahwa contoh ini dapat diselesaikan menggunakan logika probabilistik, bahkan tanpa MaxEnt. Ditunjukkan untuk sejumlah tolok ukur yang menuntut untuk alasan non-monotonik bahwa ini dapat diselesaikan secara elegan dengan MaxEnt. Pada bagian berikut, kami memperkenalkan aplikasi praktis MaxEnt yang berhasil dalam bentuk sistem pakar medis.

LEXMED, Sistem Pakar untuk Mendiagnosis Apendisitis

Sistem pakar medis LEXMED, yang menggunakan metode MaxEnt, dikembangkan di Ravensburg-Weingarten University of Applied Sciences oleh Manfred Schramm, Walter Rampf, dan penulisnya, bekerja sama dengan Weingarten 14-Nothelfer Hospital [SE00, Le999].[24] Akronim LEXMED adalah singkatan dari *learning* expert system for medical diagnosis.

[23] `QP([penguin=yes]-|> [flies=yes])` adalah bentuk alternatif dari sintaks PIT untuk `QP([flies=yes] | [penguin=yes])`.

[24] Proyek ini dibiayai oleh negara bagian Jerman Baden-Württemberg, perusahaan asuransi kesehatan AOK Baden-Württemberg, Universitas Ilmu Terapan Ravensburg-Weingarten, dan Rumah Sakit Nothelfer ke-14 di Weingarten.

7.9 DIAGNOSIS APENDISITIS DENGAN METODE FORMAL

Penyebab serius paling umum dari nyeri perut akut adalah radang usus buntu—peradangan pada usus buntu, saluran buntu yang terhubung ke sekum. Bahkan saat ini, diagnosis dapat menjadi sulit dalam banyak kasus. Misalnya, hingga sekitar 20% dari apendiks yang diangkat tidak memiliki temuan patologis, yang berarti bahwa operasi tidak diperlukan. Demikian juga, ada kasus reguler di mana usus buntu yang meradang tidak dikenali.

Sejak awal tahun 1970-an, telah ada upaya untuk mengotomatisasi diagnosis apendisitis, dengan tujuan mengurangi tingkat diagnosis palsu. Khususnya yang perlu diperhatikan adalah sistem pakar untuk diagnosis nyeri perut akut, yang dikembangkan oleh Dombalin Inggris Raya. Itu dipublikasikan pada tahun 1972, sehingga jelas lebih awal dari sistem MYCIN yang terkenal.

Hampir semua proses diagnostik formal yang digunakan dalam kedokteran sampai saat ini didasarkan pada skor. Sistem skor sangat mudah diterapkan: Untuk setiap nilai gejala (misalnya demam atau sakit perut kanan bawah) dokter mencatat sejumlah poin tertentu. Jika jumlah poin melebihi nilai tertentu (ambang), keputusan tertentu direkomendasikan (misalnya operasi). Untuk n gejala S_1, …, S_n skor untuk apendisitis dapat digambarkan secara formal sebagai:

$$Diagnosa = \begin{cases} Radang\ usus\ buntu & jika\ w_1\ s_1 + \ldots + w_n S_n > \Theta \\ Negatif & Lainnya \end{cases}$$

Dengan skor, kombinasi linear dari nilai gejala dibandingkan dengan ambang batas Θ. Bobot gejala diekstraksi dari database menggunakan metode statistik. Keuntungan dari skor adalah kesederhanaan penerapannya. Jumlah bobot poin dapat dihitung dengan tangan dengan mudah dan komputer tidak diperlukan untuk diagnosis.

Karena linearitas metode ini, skor terlalu lemah untuk memodelkan hubungan yang kompleks. Karena kontribusi w_iS_i dari gejala S_i terhadap skor dihitung secara independen dari gejala lainnya, jelas bahwa sistem skor tidak dapat memperhitungkan "konteks" apa pun. Pada prinsipnya, mereka tidak dapat membedakan kombinasi gejala, misalnya mereka tidak dapat membedakan jumlah sel darah putih pasien tua dan pasien muda.

Untuk serangkaian gejala yang diberikan tetap, probabilitas bersyarat jauh lebih kuat daripada skor untuk membuat prediksi karena yang terakhir tidak dapat menggambarkan ketergantungan antara gejala yang berbeda. Kita dapat menunjukkan bahwa skor secara implisit mengasumsikan bahwa semua gejala adalah independen.

Saat menggunakan skor, masalah lain muncul. Untuk mencapai kualitas diagnosis yang baik, kita harus menerapkan persyaratan ketat pada database yang digunakan untuk menentukan bobot w_i secara statistik. Secara khusus mereka harus mewakili kumpulan pasien di area di mana sistem diagnosis digunakan. Hal ini seringkali sulit, jika bukan tidak mungkin, untuk dijamin. Dalam kasus seperti itu, skor dan metode statistik lainnya tidak dapat digunakan, atau akan memiliki tingkat kesalahan yang tinggi.

7.10 BASIS PENGETAHUAN PROBABILISTIK HIBRIDA

Hubungan probabilistik yang kompleks sering muncul dalam kedokteran. Dengan LEXMED, hubungan ini dapat dimodelkan dengan baik dan dihitung dengan cepat. Di sini penggunaan proposisi probabilistik, yang dengannya informasi yang tidak pasti dan tidak lengkap dapat diekspresikan dan diproses dengan cara yang intuitif dan berlandaskan matematis, sangat penting. Pertanyaan berikut dapat berfungsi sebagai pertanyaan khas terhadap sistem pakar: “Seberapa tinggi kemungkinan radang usus buntu jika pasien adalah

pria berusia 23 tahun dengan nyeri di perut kanan bawah dan jumlah sel darah putih 13.000? ?" Diformulasikan sebagai probabilitas bersyarat, menggunakan nama dan rentang nilai untuk gejala yang digunakan pada Tabel 7.2, berbunyi:

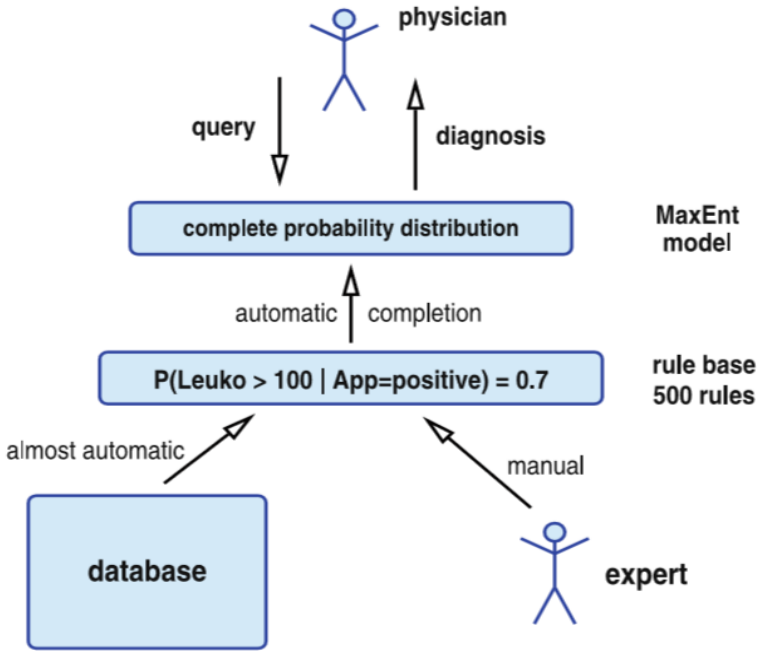
$$P(Diag4 = inflamed \vee Diag4 = berlubang \mid Jenis\ Kelamin2 = laki\text{-}laki \wedge Usia10 \in 21 - 25 \wedge Leuko7 \in 2k - 15k).$$

Dengan menggunakan probabilistik probabilistik, LEXMED memiliki kemampuan untuk menggunakan informasi dari database non-representatif karena informasi ini dapat dilengkapi dengan tepat dari sumber lain. Yang mendasari LEXMED adalah database yang hanya berisi data tentang pasien yang usus buntunya diangkat melalui pembedahan. Dengan metode statistik, (sekitar 400) aturan dihasilkan yang mengkompilasi pengetahuan yang terkandung dalam database ke dalam bentuk abstrak. Karena tidak ada pasien dalam database ini yang diduga menderita radang usus buntu tetapi memiliki diagnosis negatif (yaitu, tidak memerlukan pengobatan)[25], tidak ada pengetahuan tentang pasien negatif dalam database. Oleh karena itu, pengetahuan dari sumber lain harus ditambahkan. Dalam LEXMED, aturan yang dikumpulkan dari database dilengkapi dengan (sekitar 100) aturan dari pakar medis dan literatur medis. Ini menghasilkan database probabilistik hibrida, yang berisi pengetahuan yang diekstraksi dari data serta pengetahuan yang dirumuskan secara eksplisit oleh para ahli. Karena kedua jenis aturan diformulasikan sebagai probabilitas bersyarat (lihat misalnya (7.14)), mereka dapat dengan mudah digabungkan, seperti yang ditunjukkan pada Gambar 7.8 dan dengan rincian lebih lanjut pada Gambar 7.7.

Tabel 7. 2 Gejala yang digunakan untuk kueri di LEXMED dan nilainya. Jumlah nilai untuk setiap gejala diberikan pada kolom bertanda #

Symptom	Values	#	Short
Gender	*Male, female*	2	*Sex2*
Age	*0–5, 6–10, 11–15, 16–20, 21–25, 26–35, 36–45, 46–55, 56–65, 65–*	10	*Age10*
Pain 1st Quad.	*Yes, no*	2	*P1Q2*
Pain 2nd Quad.	*Yes, no*	2	*P2Q2*
Pain 3rd Quad.	*Yes, no*	2	*P3Q2*
Pain 4th Quad.	*Yes, no*	2	*P4Q2*
Guarding	*Local, global, none*	3	*Gua3*
Rebound tenderness	*Yes, no*	2	*Reb2*
Pain on tapping	*Yes, no*	2	*Tapp2*
Rectal pain	*Yes, no*	2	*RecP2*
Bowel sounds	*Weak, normal, increased, none*	4	*BowS4*
Abnormal ultrasound	*Yes, no*	2	*Sono2*
Abnormal urine sedim.	*Yes, no*	2	*Urin2*
Temperature (rectal)	*–37.3, 37.4–37.6, 37.7–38.0, 38.1–38.4, 38.5–38.9, 39.0–*	6	*TRec6*
Leukocytes	*0–6k, 6k–8k, 8k–10k, 10k–12k, 12k–15k, 15k–20k, 20k–*	7	*Leuko7*
Diagnosis	*Inflamed, perforated, negative, other*	4	*Diag4*

[25] Diagnosis negatif ini dilambangkan sebagai "nyeri perut non-spesifik" (NSAP).



Gambar 7.5 Aturan probabilistik dihasilkan dari data dan pengetahuan pakar, yang diintegrasikan dalam basis aturan (*knowledge base*) dan akhirnya dibuat lengkap menggunakan metode MaxEnt

LEXMED menghitung probabilitas berbagai diagnosis menggunakan distribusi probabilitas dari semua variabel yang relevan (lihat Tabel 7.2). Karena semua 14 gejala yang digunakan dalam LEXMED dan diagnosis dimodelkan sebagai variabel diskrit (bahkan variabel kontinu seperti nilai leukosit dibagi menjadi rentang), ukuran distribusi (yaitu, ukuran ruang kejadian) dapat ditentukan menggunakan Tabel 7.2 sebagai produk dari jumlah nilai semua gejala, atau

$$2^{10} \cdot 10 \cdot 3 \cdot 4 \cdot 6 \cdot 7 \cdot 4 = 20643840$$

elemen. Berdasarkan kondisi normalisasi dari Teorema 7.1, sehingga memiliki 20643839 nilai independen. Setiap aturan yang ditetapkan dengan nilai probabilitas kurang dari 20643839 berpotensi tidak sepenuhnya menggambarkan ruang peristiwa ini. Untuk dapat menjawab sembarang query, sistem pakar membutuhkan distribusi yang lengkap. Konstruksi distribusi yang ekstensif dan konsisten dengan menggunakan metode statistik sangat sulit.[26] Untuk meminta dari seorang ahli manusia semua nilai 20643839 untuk distribusi (bukan 100 aturan yang disebutkan di atas) pada dasarnya tidak mungkin.

Di sini metode MaxEnt ikut bermain. Generalisasi sekitar 500 aturan ke model probabilitas lengkap dilakukan di LEXMED dengan memaksimalkan entropi dengan 500 aturan sebagai kendala. Pengkodean yang efisien dari distribusi MaxEnt yang dihasilkan menghasilkan waktu respons untuk diagnosis sekitar satu detik.

7.11 PENERAPAN LEXMED

Penggunaan LEXMED sederhana dan cukup jelas. Dokter mengunjungi home page LEXMED di www.lexmed.de.[27] Untuk diagnosis otomatis, dokter memasukkan hasil pemeriksaannya ke formulir input pada Gambar 7.6. Setelah satu atau dua detik ia

[26] Tugas menghasilkan fungsi dari sekumpulan data dikenal sebagai pembelajaran mesin. Kami akan membahas ini secara menyeluruh di Bab. 8.

[27] Versi dengan fungsionalitas terbatas dapat diakses tanpa kata sandi

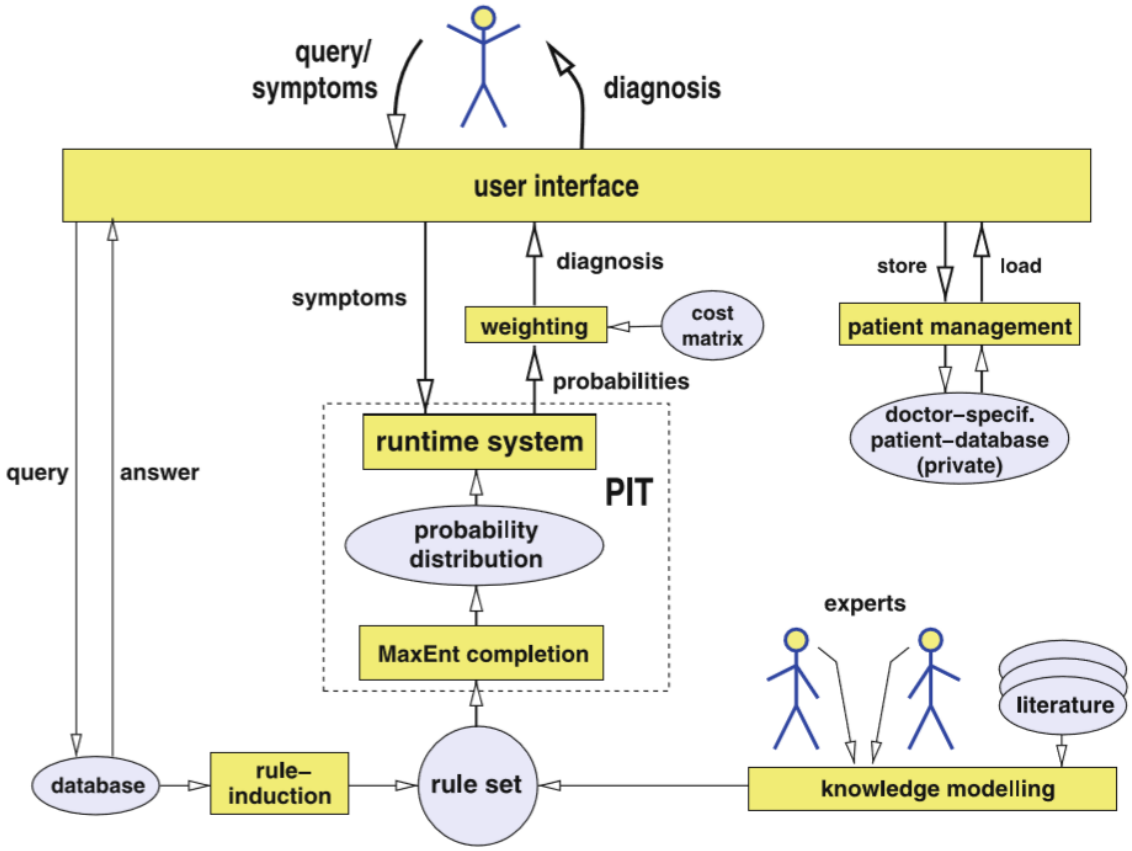
menerima probabilitas untuk empat diagnosis yang berbeda serta saran untuk pengobatan (Bagian 7.3.5). Jika hasil pemeriksaan tertentu hilang sebagai input (misalnya hasil sonogram), maka dokter memilih entri yang tidak diperiksa. Secara alami kepastian diagnosis lebih tinggi ketika lebih banyak nilai gejala yang dimasukkan.

Personal Details	unknown	values	
Gender	◉	○ male ○ female	?
Age-group	○	○ 0-5 ○ 6-10 ○ 11-15 ○ 16-20 ◉ 21-25 ○ 26-35 ○ 36-45 ○ 46-55 ○ 56-65 ○ 65-	?
Results of examination	**not done**	**values**	
1st quadrant	◉	○ yes ○ no	?
2nd quadrant	◉	○ yes ○ no	?
3rd quadrant	○	◉ yes ○ no	?
4th quadrant	◉	○ yes ○ no	?
guarding	○	◉ local ○ global ○ none	?
rebound tenderness	○	◉ yes ○ no	?
pain on tapping	○	◉ yes ○ no	?
rectal pain	◉	○ yes ○ no	?
bowel sounds	○	○ weak ◉ normal ○ increased ○ none	?
abnormal ultrasound	◉	○ yes ○ no	?
abnormal urine sediment	◉	○ yes ○ no	?
temperature range (rectal)	○	○ -37.3 ○ 37.4-37.6 ○ 37.7-38.0 ○ 38.1-38.4 ◉ 38.5-38.9 ○ 39.0-	?
leucocyte count	○	○ 0-6k ○ 6k-8k ○ 8k-10k ○ 10k-12k ◉ 12k-15k ○ 15k-20k ○ 20k-	?

	Result of the PIT diagnosis			
Diagnosis	App. inflamed	App. perforated	Negative	Other
Probability	0.70	0.17	0.06	0.07

Gambar. 7.6 Masker input LEXMED untuk input gejala yang diperiksa dan di bawahnya output probabilitas diagnosis yang dihasilkan

Setiap pengguna terdaftar memiliki akses ke database pasien pribadi, di mana data input dapat diarsipkan. Dengan demikian data dan diagnosis dari pasien sebelumnya dapat dengan mudah dibandingkan dengan pasien baru. Ikhtisar proses di LEXMED diberikan pada Gambar 7.7.



Gambar 7.7 Aturan dihasilkan dari database dan juga dari pengetahuan ahli.

Dari ini, MaxEnt membuat distribusi probabilitas lengkap. Untuk kueri pengguna, probabilitas setiap diagnosis yang mungkin dihitung. Menggunakan matriks biaya keputusan kemudian disarankan

Fungsi LEXMED

Pengetahuan diformalkan menggunakan proposisi probabilistik. Misalnya proposisi

$$P(Leuko7 > 20000 | Diag4 = meradang) = 0{,}09$$

memberikan frekuensi 9% untuk nilai leukosit lebih dari 20.000 dalam kasus apendiks yang meradang.[28]

Pembelajaran Aturan dengan Induksi Statistik

Data mentah dalam database LEXMED berisi 54 nilai berbeda (dianonimkan) untuk 14.646 pasien. Seperti disebutkan sebelumnya, hanya pasien yang usus buntunya diangkat melalui pembedahan yang dimasukkan dalam database ini. Dari 54 atribut yang digunakan dalam database, setelah analisis statistik, 14 gejala yang ditunjukkan pada Tabel 7.2 digunakan. Sekarang aturan dibuat dari database ini dalam dua langkah. Langkah pertama menentukan struktur ketergantungan gejala. Langkah kedua mengisi struktur ini dengan aturan probabilitas masing-masing.[29]

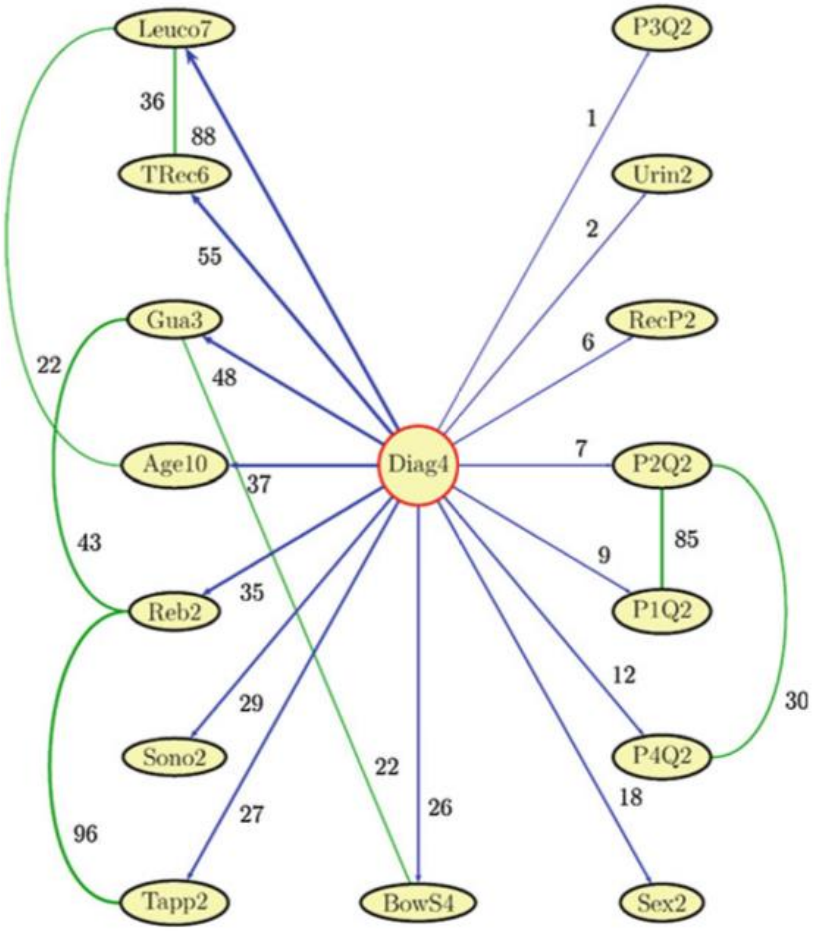
7.12 MENENTUKAN GRAFIK KETERGANTUNGAN

Grafik pada Gambar 7.8 berisi untuk setiap variabel (gejala dan diagnosis) sebuah simpul dan tepi terarah yang menghubungkan berbagai simpul. Ketebalan tepi antara variabel mewakili ukuran ketergantungan statistik atau korelasi variabel. Korelasi dua

[28] Alih-alih nilai numerik individual, interval juga dapat digunakan di sini (misalnya [0.06, 0.12]).

[29] Untuk pengenalan sistematis untuk pembelajaran mesin, kami merujuk pembaca ke Bab. 8.

variabel bebas sama dengan nol. Korelasi pasangan untuk masing-masing dari 14 gejala dengan Diag4 dihitung dan dicantumkan dalam grafik. Selanjutnya, semua korelasi rangkap tiga antara diagnosis dan dua gejala dihitung. Dari jumlah tersebut, hanya nilai terkuat yang telah ditarik sebagai tepi tambahan antara dua gejala yang berpartisipasi.



Gambar 7.8 Grafik ketergantungan dihitung dari database

```
 1 P([Leuco7=0-6k]    | [Diag4=negativ] * [Age10=16-20]) = [0.132,0.156];
 2 P([Leuco7=6-8k]    | [Diag4=negativ] * [Age10=16-20]) = [0.257,0.281];
 3 P([Leuco7=8-10k]   | [Diag4=negativ] * [Age10=16-20]) = [0.250,0.274];
 4 P([Leuco7=10-12k]  | [Diag4=negativ] * [Age10=16-20]) = [0.159,0.183];
 5 P([Leuco7=12-15k]  | [Diag4=negativ] * [Age10=16-20]) = [0.087,0.112];
 6 P([Leuco7=15-20k]  | [Diag4=negativ] * [Age10=16-20]) = [0.032,0.056];
 7 P([Leuco7=20k-]    | [Diag4=negativ] * [Age10=16-20]) = [0.000,0.023];
 8 P([Leuco7=0-6k]    | [Diag4=negativ] * [Age10=21-25]) = [0.132,0.172];
 9 P([Leuco7=6-8k]    | [Diag4=negativ] * [Age10=21-25]) = [0.227,0.266];
10 P([Leuco7=8-10k]   | [Diag4=negativ] * [Age10=21-25]) = [0.211,0.250];
11 P([Leuco7=10-12k]  | [Diag4=negativ] * [Age10=21-25]) = [0.166,0.205];
12 P([Leuco7=12-15k]  | [Diag4=negativ] * [Age10=21-25]) = [0.081,0.120];
13 P([Leuco7=15-20k]  | [Diag4=negativ] * [Age10=21-25]) = [0.041,0.081];
14 P([Leuco7=20k-]    | [Diag4=negativ] * [Age10=21-25]) = [0.004,0.043];
```

Gambar 7.9 Beberapa aturan LEXMED dengan interval probabilitas."*" adalah singkatan dari "∧" di sini

Memperkirakan Probabilitas Aturan

Struktur grafik dependensi menggambarkan struktur aturan yang dipelajari.[30] Aturan di sini memiliki kompleksitas yang berbeda: ada aturan yang hanya menggambarkan distribusi kemungkinan diagnosis (aturan prioritas, misalnya (7.13)), aturan yang menggambarkan ketergantungan antara diagnosis dan asimto (aturan dengan kondisi sederhana, misalnya (7.14)), dan akhirnya menjelaskan dua aturan ketergantungan. gejala, seperti yang diberikan pada Gambar 7.9 dalam sintaks PIT.

Persamaan 7.13

$$P(Diag4 = meradang) = 0{,}40,$$

Persamaan 7.14

$$P(Sono2 = ya \mid Diag4 = meradang) = 0{,}43,$$

Persamaan 7.15

$$P(P4Q2) = ya \mid Diag4 = meradang \wedge P2Q2 = ya) = 0{,}61$$

Untuk menjaga ketergantungan konteks dari pengetahuan yang disimpan sekecil mungkin, semua aturan berisi diagnosis dalam kondisinya dan bukan sebagai kesimpulan. Ini sangat mirip dengan konstruksi banyak buku kedokteran dengan formulasi semacam "Dengan radang usus buntu yang biasa kita lihat ...". Seperti yang ditunjukkan sebelumnya pada Contoh 7.6, bagaimanapun, ini tidak menimbulkan masalah karena, menggunakan rumus Bayesian, LEXMED secara otomatis menempatkan aturan-aturan ini ke dalam bentuk yang benar.

Nilai numerik untuk aturan ini diperkirakan dengan menghitung frekuensinya dalam database. Misalnya, nilai dalam (7.14) diberikan dengan menghitung dan menghitung

$$\frac{|Diag4 = meradang \wedge Sono2 = yes|}{|Diag4 = meradang|}$$

Aturan Ahli

Karena database apendisitis hanya berisi pasien yang telah menjalani operasi, aturan untuk nyeri perut non-spesifik (NSAP) menerima nilainya dari proposisi para ahli medis. Pengalaman di LEXMED menegaskan bahwa aturan probabilistik mudah dibaca dan dapat langsung diterjemahkan ke dalam bahasa alami. Pernyataan para ahli medis tentang hubungan frekuensi gejala tertentu dan diagnosis, baik dari literatur atau sebagai hasil wawancara, oleh karena itu dapat dimasukkan ke dalam dasar aturan dengan sedikit biaya. Untuk memodelkan ketidakpastian pengetahuan pakar, penggunaan interval probabilitas telah terbukti efektif. Pengetahuan ahli terutama diperoleh dari ahli bedah yang berpartisipasi, Dr. Rampf dan Dr. Hontschik, dan publikasi mereka [Hon94]. Setelah aturan ahli telah dibuat, basis aturan selesai. Kemudian model probabilitas lengkap dihitung dengan metode entropi maksimum oleh sistem PIT.

Kueri Diagnosis Dengan menggunakan model probabilitas yang disimpan secara efisien, LEXMED menghitung probabilitas untuk empat kemungkinan diagnosis dalam beberapa detik. Sebagai contoh, kita asumsikan output berikut:

	Hasil diagnosa PIT			
Diagnosis	Lampiran meradang	Lampiran Berlubang	Negatif	Lainnya
Probabilitas	0.24	0.16	0.57	0.03

[30] Perbedaan antara ini dan jaringan Bayesian adalah, misalnya, bahwa aturan dilengkapi dengan interval probabilitas dan hanya setelah menerapkan prinsip entropi maksimum model probabilitas unik yang dihasilkan.

Keputusan harus dibuat berdasarkan empat nilai probabilitas ini untuk mengejar salah satu dari empat perawatan: operasi, operasi darurat, observasi stasioner, atau observasi ambulan.[31] Sementara probabilitas untuk diagnosis negatif dalam kasus ini lebih besar daripada yang lain, mengirim pasien pulang sebagai sehat bukanlah keputusan yang baik. Kita dapat melihat dengan jelas bahwa, bahkan ketika probabilitas diagnosis telah dihitung, diagnosis belum selesai. Sebaliknya, tugas sekarang adalah untuk memperoleh keputusan yang optimal dari probabilitas ini. Untuk tujuan ini, pengguna dapat meminta LEXMED menghitung keputusan yang disarankan.

7.13 MANAJEMEN RISIKO MENGGUNAKAN MATRIKS BIAYA

Bagaimana probabilitas yang dihitung sekarang diterjemahkan secara optimal ke dalam keputusan? Sebuah algoritma naif akan menetapkan keputusan untuk setiap diagnosis dan akhirnya memilih keputusan yang sesuai dengan probabilitas tertinggi. Asumsikan bahwa probabilitas yang dihitung adalah 0,40 untuk diagnosis apendisitis (meradang atau berlubang), 0,55 untuk diagnosis negatif, dan 0,05 untuk diagnosis lainnya. Sebuah algoritma naif sekarang akan memilih keputusan (terlalu berisiko) "tidak ada operasi" karena sesuai dengan diagnosis dengan probabilitas yang lebih tinggi. Metode yang lebih baik terdiri dari membandingkan biaya kesalahan yang mungkin terjadi untuk setiap keputusan. Kesalahan dikuantifikasi dalam bentuk "(hipotetis) biaya tambahan dari keputusan saat ini dibandingkan dengan yang optimal". Nilai yang diberikan mengandung biaya untuk rumah sakit, perusahaan asuransi, pasien (misalnya risiko komplikasi pasca operasi), dan pihak lain (misalnya tidak masuk kerja), dengan mempertimbangkan konsekuensi jangka panjang. Biaya ini diberikan pada Tabel 7.3.

Tabel 7.3 Matriks biaya LEXMED bersama dengan probabilitas diagnosis yang dihitung pasien

Terapi	***Probabilitas variasi diagnosa***				
	Meradang	***Berlubang***	***Negatif***	***Lainnya***	
	0.25	***0.15***	***0.55***	***0.05***	
Operasi	0	500	5800	6000	3565
Operasi Darurat	500	0	6300	6500	3915
Observasi Ambulan	12000	**150000**	0	16500	26325
Lainnya	3000	5000	1300	0	2215
Observasi Perlengkapan	3500	7000	400	600	2175

Entri akhirnya dirata-ratakan untuk setiap keputusan, yaitu, dijumlahkan dengan mempertimbangkan frekuensinya. Ini tercantum dalam kolom terakhir pada Tabel 7.3. Akhirnya, keputusan dengan biaya kesalahan rata-rata terkecil disarankan. Pada Tabel 7.3 matriks diberikan bersama dengan vektor probabilitas yang dihitung untuk pasien (dalam hal ini: (0,25, 0,15, 0,55, 0,05)). Kolom terakhir dari tabel berisi hasil perhitungan biaya rata-rata yang diharapkan dari kesalahan. Nilai Operasi di baris pertama dihitung sebagai 0,25 **.** 0 + 0,15 **.** 500 + 0,55 **.** 5800 + 0,05 **.** 6000 = 3565, rata-rata tertimbang dari semua biaya. Keputusan optimal dimasukkan dengan biaya (tambahan) 0. Sistem memutuskan perawatan dengan biaya rata-rata minimal. Dengan demikian adalah contoh dari agen berorientasi biaya.

[31] Observasi ambulan berarti pasien diperbolehkan tinggal di rumah.

Matriks Biaya dalam Kasus Biner Untuk lebih memahami matriks biaya dan manajemen risiko, kami sekarang akan membatasi sistem LEXMED pada keputusan dua nilai antara diagnosis apendisitis dengan probabilitas

$$p_1 = P(appendicitis) = P(Diag4 = meradang) + P(Diag4 = perforated)$$

dan NSAP dengan probabilitas

$$p_2 = P(NSAP) = P(Diag4)\ negatif) + P(Diag4 = other)$$

Satu-satunya perawatan yang tersedia adalah operasi dan observasi keliling. Matriks biaya dengan demikian adalah matriks 2 × 2 berbentuk

$$\begin{pmatrix} 0 & k_2 \\ k_1 & 0 \end{pmatrix}$$

Dua nol pada diagonal berdiri untuk operasi keputusan yang benar dalam kasus apendisitis dan observasi ambulan untuk NSAP. Parameter k_2 menunjukkan biaya yang diharapkan yang terjadi ketika pasien tanpa radang usus buntu dioperasi. Kesalahan ini disebut positif palsu. Di sisi lain, keputusan observasi ambulan pada kasus apendisitis adalah negatif palsu. Vektor probabilitas $(p_1, p_2)^T$ sekarang dikalikan dengan matriks ini dan kita memperoleh vektor

$$(k_2\ p_2,\ k_1\ p_1)^T$$

dengan biaya tambahan rata-rata untuk dua kemungkinan perawatan. Karena keputusan hanya memperhitungkan hubungan dua komponen, vektor dapat dikalikan dengan faktor skalar apa pun. Kami memilih $1/k_1$ dan memperoleh $((k_2/k_1)\ p_2, p_1)$. Jadi hanya hubungan $k = k_2/k_1$ yang relevan di sini. Hasil yang sama diperoleh dengan matriks biaya yang lebih sederhana

$$\begin{pmatrix} 0 & k \\ 1 & 0 \end{pmatrix}$$

yang hanya berisi variabel k. Parameter ini sangat penting karena menentukan manajemen risiko. Dengan mengubah k kita dapat menyesuaikan "titik kerja" dari sistem diagnosis. Untuk k → sistem ditempatkan dalam pengaturan yang sangat berisiko karena tidak ada pasien yang akan dioperasi, dengan konsekuensi tidak memberikan klasifikasi positif palsu, tetapi banyak negatif palsu. Dalam kasus k = 0 kondisinya terbalik dan semua pasien dioperasi.

Pertunjukan

LEXMED dimaksudkan untuk digunakan dalam praktik medis atau ambulans. Prasyarat untuk penggunaan LEXMED adalah nyeri perut akut selama beberapa jam (tetapi kurang dari lima hari). Lebih lanjut, LEXMED (saat ini) khusus untuk radang usus buntu, yang berarti bahwa untuk penyakit lain, sistem ini berisi sangat sedikit informasi.

Dalam lingkup studi prospektif, database representatif dengan 185 kasus dibuat di 14 Rumah Sakit Nothelfer. Ini berisi pasien rumah sakit yang datang ke klinik setelah beberapa jam mengalami sakit perut akut dan dugaan radang usus buntu. Dari pasien ini, gejala dan diagnosis (diverifikasi dari sampel jaringan dalam kasus operasi) dicatat.

Jika pasien diperbolehkan pulang (tanpa operasi) setelah tinggal beberapa jam atau 1-2 hari dengan sedikit atau tanpa keluhan, kemudian ditanyakan melalui telepon apakah pasien tetap bebas dari gejala atau apakah ditemukan diagnosis positif pada pasien tersebut. pengobatan selanjutnya.

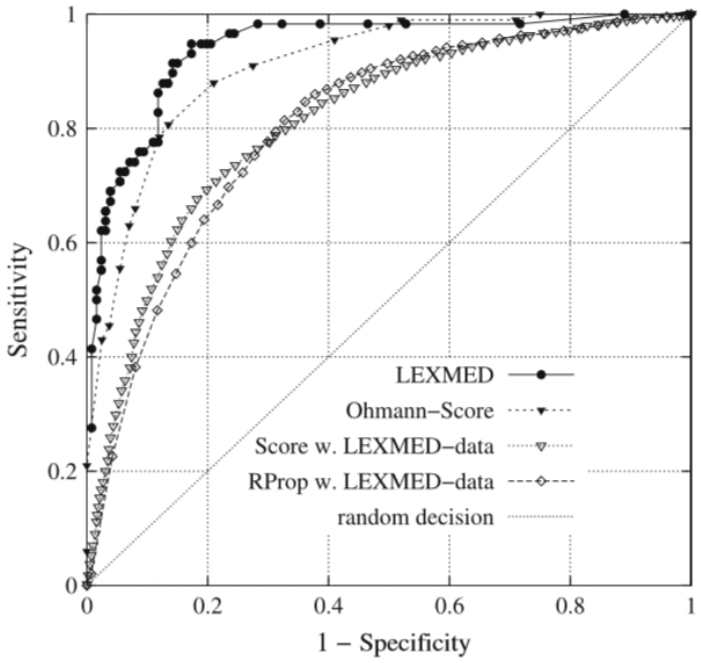
Untuk menyederhanakan representasi dan membuat perbandingan yang lebih baik untuk penelitian serupa, LEXMED dibatasi pada perbedaan dua nilai antara radang usus buntu dan NSAP. Sekarang k divariasikan antara nol dan tak terhingga dan untuk setiap nilai k sensitivitas dan spesifisitas diukur terhadap data uji. Langkah-langkah sensitivitas

$$P(clasifikasi\ positif|positive) = \frac{|positif\ dan\ clasifikasi\ positif|}{|positif|}$$

yaitu, bagian relatif dari kasus positif yang diidentifikasi dengan benar. Ini menunjukkan seberapa sensitif sistem diagnostik. Spesifikitas, di sisi lain, mengukur yaitu, bagian relatif dari kasus negatif yang diidentifikasi dengan benar.

$$P(clasifikasi\ negatif|negatif) = \frac{|negatif\ dan\ clasifikasi\ negatif|}{|negatif|}$$

Kami memberikan hasil sensitivitas dan spesifisitas pada Gambar. 7.10 untuk $0 \leq k < \infty$. Kurva ini dilambangkan dengan kurva ROC, atau karakteristik pengoperasian receiver. Sebelum kita sampai pada analisis kualitas LEXMED, beberapa kata tentang arti kurva ROC. Garis yang membagi diagram secara diagonal digambar untuk orientasi. Semua titik pada garis ini sesuai dengan keputusan acak. Misalnya, titik (0.2, 0.2) sesuai dengan spesifisitas 0,8 dengan sensitivitas 0,2. Kita dapat sampai pada hal ini dengan cukup mudah dengan mengklasifikasikan kasus baru, tanpa melihatnya, dengan probabilitas 0,2 untuk positif dan 0,8 untuk negatif. Oleh karena itu, setiap sistem diagnosis berbasis pengetahuan harus menghasilkan ROC yang jelas terletak di atas diagonal.



Gambar 7.10 Kurva ROC dari LEXMED dibandingkan dengan skor Ohmann dan dua model tambahan

Nilai ekstrim dalam kurva ROC juga menarik. Pada titik (0,0) ketiga kurva berpotongan. Sistem diagnosis yang sesuai akan mengklasifikasikan semua kasus sebagai negatif. Nilai ekstrim lainnya (1, 1) sesuai dengan sistem yang akan memutuskan untuk melakukan operasi untuk setiap pasien dan dengan demikian memiliki sensitivitas 1. Kita dapat menyebut kurva ROC sebagai kurva karakteristik untuk sistem diagnostik dua nilai. Sistem diagnostik yang ideal akan memiliki kurva karakteristik yang hanya terdiri dari titik (0, 1), dan dengan demikian memiliki spesifisitas 100% dan sensitivitas 100%.

Sekarang mari kita menganalisis kurva ROC. Pada sensitivitas 88%, LEXMED mencapai spesifisitas 87% (k = 0,6). Sebagai perbandingan, skor Ohmann, skor yang mapan dan terkenal untuk apendisitis berikan. Karena LEXMED berada di atas atau di sebelah kiri skor Ohmann hampir di semua tempat, kualitas diagnosis rata-ratanya jelas lebih baik. Ini

tidak mengejutkan karena skor terlalu lemah untuk memodelkan proposisi yang menarik. Pada Latihan 8.17 kami akan menunjukkan bahwa skor ekuivalen dengan kasus khusus naive Bayes, yaitu dengan asumsi bahwa semua gejala tidak berpasangan saat diagnosis diketahui. Ketika membandingkan LEXMED dengan skor, bagaimanapun, harus disebutkan bahwa database yang representatif secara statistik digunakan untuk skor Ohmann, tetapi database non-representatif yang ditingkatkan dengan pengetahuan ahli digunakan untuk LEXMED. Untuk mendapatkan gagasan tentang kualitas data LEXMED dibandingkan dengan data Ohmann, skor linier dihitung menggunakan metode kuadrat terkecil, yang juga digambar untuk perbandingan. Selanjutnya, jaringan saraf dilatih pada data LEXMED dengan algoritma RPop. Kekuatan menggabungkan data dan pengetahuan ahli ditampilkan dengan jelas dalam perbedaan antara kurva LEXMED dan kurva sistem skor dan algoritma RProp.

Area Aplikasi dan Pengalaman

LEXMED tidak boleh menggantikan penilaian ahli bedah yang berpengalaman. Namun, karena spesialis tidak selalu tersedia dalam pengaturan klinis, permintaan LEXMED menawarkan pendapat kedua yang substantif. Yang sangat menarik dan bermanfaat adalah penerapan sistem dalam ambulans klinis dan untuk dokter umum.

Kemampuan belajar LEXMED, yang memungkinkan untuk memperhitungkan gejala lebih lanjut, data pasien lebih lanjut, dan aturan lebih lanjut, juga menghadirkan kemungkinan baru di klinik. Untuk kelompok yang sangat jarang yang sulit didiagnosis, misalnya anak di bawah usia enam tahun, LEXMED dapat menggunakan data dari dokter anak atau database khusus lainnya, untuk mendukung bahkan ahli bedah yang berpengalaman.

Selain penggunaan langsung dalam diagnosis, LEXMED juga mendukung langkah-langkah jaminan kualitas. Misalnya, perusahaan asuransi dapat membandingkan kualitas diagnosis rumah sakit dengan sistem pakar. Dengan mengembangkan lebih lanjut matriks biaya yang dibuat di LEXMED (dengan persetujuan dokter, asuransi, dan pasien), kualitas diagnosis dokter, diagnosis komputer, dan institusi medis lainnya akan menjadi lebih mudah untuk dibandingkan.

LEXMED telah menunjukkan cara baru untuk membangun sistem diagnostik otomatis. Menggunakan bahasa teori probabilitas dan algoritma MaxEnt, secara induktif, pengetahuan yang diturunkan secara statistik digabungkan dengan pengetahuan dari para ahli dan dari literatur. Pendekatan berdasarkan model probabilistik secara teoretis elegan, dapat diterapkan secara umum, dan telah memberikan hasil yang sangat baik dalam sebuah penelitian kecil.

LEXMED telah digunakan secara praktis di 14 Rumah Sakit Bukan Pembantu di Weingartensince 1999 dan telah tampil di sana dengan sangat baik. Ini juga tersedia di www.lexmed.de, tentu saja tanpa jaminan. Kualitas diagnosisnya sebanding dengan dokter bedah berpengalaman dan lebih baik daripada dokter umum biasa, atau dokter yang tidak berpengalaman di klinik.

Terlepas dari keberhasilan ini, terbukti bahwa sangat sulit untuk memasarkan sistem seperti itu secara komersial di sistem medis Jerman. Salah satu alasannya adalah karena tidak ada pasar bebas untuk mempromosikan kualitas yang lebih baik (di sini diagnosis yang lebih baik) melalui mekanisme pemilihannya. Lebih jauh lagi, dalam kedokteran, waktu untuk penggunaan teknik-teknik cerdas secara luas belum tiba—bahkan pada tahun 2010. Salah satu penyebabnya bisa jadi adalah ajaran konservatif dalam hal ini di fakultas-fakultas sekolah kedokteran Jerman.

Masalah selanjutnya adalah keinginan banyak pasien untuk mendapatkan nasihat dan perawatan pribadi dari dokter, bersama dengan ketakutan bahwa, dengan pengenalan sistem pakar, pasien hanya akan berkomunikasi dengan mesin. Namun, ketakutan ini sama sekali tidak berdasar. Bahkan dalam jangka panjang, sistem pakar medis tidak dapat menggantikan dokter. Namun, mereka dapat, seperti operasi laser dan pencitraan resonansi magnetik, digunakan secara menguntungkan untuk semua peserta. Sejak sistem diagnostik komputer medis pertama di de Dombal pada tahun 1972, hampir 40 tahun telah berlalu. Masih harus dilihat apakah kedokteran akan menunggu 40 tahun lagi sampai diagnosa komputer menjadi alat medis yang mapan.

7.14 PENALARAN DENGAN *BAYESIAN NETWORKS*

Satu masalah dengan penalaran menggunakan probabilitas dalam prakteknya. Jika variabel $X_{1,}\ldots$, X_d digunakan masing-masing dengan n nilai, maka distribusi probabilitas terkait memiliki nilai total n^d. Ini berarti bahwa dalam kasus terburuk penggunaan memori dan waktu komputasi untuk menentukan probabilitas tertentu tumbuh secara eksponensial dengan jumlah variabel.

Dalam pelaksanaannya aplikasi biasanya sangat terstruktur dan pendistribusiannya banyak mengandung redudansi. Ini berarti bahwa hal itu dapat sangat dikurangi dengan metode yang tepat. Penggunaan jaringan Bayesian telah membuktikan kekuatan di sini dan merupakan salah satu teknik AI yang telah berhasil digunakan dalam praktik. Jaringan Bayesian memanfaatkan pengetahuan tentang independensi variabel untuk menyederhanakan model.

7.15 VARIABLE INDEPENDEN

Dalam kasus yang paling sederhana, semua variabel bebas berpasangan dan dalam kasus ini, semua entri dalam distribusi dapat dihitung dari nilai $P(X_1),\ldots,P(X_d)$. Aplikasi yang menarik, bagaimanapun, biasanya tidak dapat dimodelkan karena probabilitas bersyarat menjadi sepele.[32] Karena semua probabilitas bersyarat direduksi menjadi probabilitas apriori. Situasi menjadi lebih menarik ketika hanya sebagian dari variabel independen atau independen dalam kondisi tertentu. Untuk penalaran dalam AI, ketergantungan antar variabel menjadi penting dan harus dimanfaatkan.

$$P(X_1, ..., X_d) = P(X_1) \cdot P(X_2) ... P(X_d)$$

Kami ingin menguraikan penalaran dengan jaringan Bayesian melalui contoh sederhana dan sangat ilustratif oleh J. Pearl [Pea88], yang menjadi terkenal melalui [RN10] dan sekarang menjadi pengetahuan dasar AI.

Contoh 7.10

(Contoh Alarm) Bob, yang masih lajang, telah memasang sistem alarm di rumahnya untuk melindungi dari pencuri. Bob tidak bisa mendengar alarm ketika dia sedang bekerja di kantor. Karena itu dia meminta kedua tetangganya, John di rumah sebelah kiri, dan Mary di rumah sebelah kanan, untuk meneleponnya di kantornya jika mereka mendengar alarmnya. Setelah beberapa tahun, Bob mengetahui seberapa andal John dan Mary dan memodelkan perilaku panggilan mereka menggunakan probabilitas bersyarat sebagai berikut.[33]

$$P(J|Al) = 0.90$$

[32] Dalam metode naive Bayes, independensi semua atribut diasumsikan, dan metode ini telah berhasil diterapkan pada klasifikasi teks.

[33] Variabel biner J dan M mewakili dua peristiwa "panggilan John", dan "panggilan Mary", masing-masing, Al untuk "bunyi sirene alarm", Bur untuk "perampokan" dan Ear untuk "gempa bumi".

$P(M|Al) = 0.70$
$P(J|\neg Al) = 0.05$
$P(M)|\ \neg Al) = 0.01$

Karena Mary sulit mendengar, dia gagal mendengar alarm lebih sering daripada John. Namun, John terkadang mencampuradukkan alarm di rumah Bob dengan alarm di rumah lain. Alarm dipicu oleh pencurian, tetapi juga dapat dipicu oleh gempa (lemah), yang dapat menyebabkan alarm palsu karena Bob hanya ingin tahu tentang pencurian saat berada di kantornya. Hubungan ini dimodelkan oleh

$P(Al|Bur, Ear) = 0.95$
$P(Al|Bur, \neg Ear) = 0.94$
$P(Al|\neg Bur, Ear) = 0.29$
$P(Al|\neg Bur, \neg Ear) = 0.001$

Serta probabilitas apriori P(Bur) = 0,001 dan P(Ear) = 0,002. Kedua variabel ini independen karena gempa bumi tidak membuat rencana berdasarkan kebiasaan pencuri, dan sebaliknya tidak ada cara untuk memprediksi gempa, sehingga pencuri tidak memiliki kesempatan untuk mengatur jadwal mereka sesuai.
Kueri sekarang dibuat berdasarkan basis pengetahuan ini. Misalnya, Bob mungkin tertarik pada P(Bur|J M), P(J|Bur) atau P(M|Bur). Artinya, dia ingin tahu seberapa sensitif variabel J dan M bereaksi terhadap laporan pembobolan.

7.16 REPRENSENTASI GRAFIS PENGETAHUAN SEBAGAI JARINGAN BAYESIAN

Kita dapat sangat menyederhanakan kerja praktek dengan secara grafis mewakili pengetahuan yang dirumuskan sebagai probabilitas bersyarat. Gambar 7.11 menunjukkan jaringan Bayesian untuk contoh alarm. Setiap node dalam jaringan mewakili variabel dan setiap tepi terarah pernyataan probabilitas bersyarat. Tepi dari Al ke J misalnya mewakili dua nilai P(J|Al) dan P(J|¬Al), yang diberikan dalam bentuk tabel, yang disebut CPT (tabel probabilitas bersyarat). CPT dari sebuah node mencantumkan semua probabilitas bersyarat dari variabel node yang dikondisikan pada semua node yang dihubungkan oleh edge yang masuk.

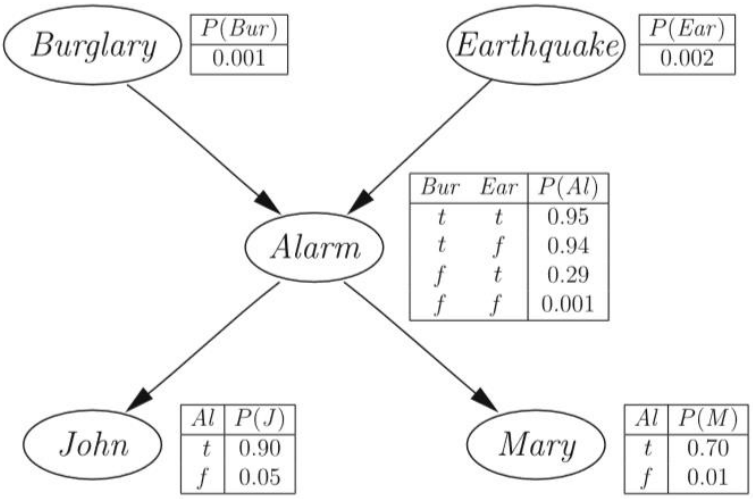
Saat mempelajari jaringan, kita mungkin bertanya pada diri sendiri mengapa tidak ada sisi lain yang disertakan selain empat yang digambar. Kedua node Bur dan Ear tidak terhubung karena variabelnya independen. Semua node lain memiliki node induk, yang membuat alasannya sedikit lebih kompleks. Pertama-tama kita membutuhkan konsep independensi bersyarat.

7.17 INDEPENDENSI KONDISIONAL

Analog dengan independensi variabel acak, kami memberikan

Definisi 7.6
Dua variabel A dan B disebut bebas bersyarat, diberikan C jika

$$P(A, B|C) = P(A|C) \ . \ \boldsymbol{P}(B|C)$$



Gambar 7.11 Jaringan Bayesian untuk contoh alarm dengan CPT terkait

Persamaan ini berlaku untuk semua kombinasi nilai untuk ketiga variabel (yaitu, untuk distribusi), yang kita lihat dalam notasi. Kita sekarang melihat node J dan M dalam contoh alarm, yang memiliki node induk yang sama Al. Jika John dan Mary secara independen bereaksi terhadap alarm, maka kedua variabel J dan M adalah independen diberikan Al, yaitu:

$$\boldsymbol{P}(J, M|Al) = \boldsymbol{P}(J|Al) \;.\; \boldsymbol{P}(M|Al)$$

Jika nilai Al diketahui, misalnya karena alarm dipicu, maka variabel J dan M adalah independen (dalam kondisi Al = w). Karena independensi bersyarat dari dua variabel J dan M, tidak ada tepi antara dua node yang ditambahkan. Namun, J dan M tidak independen (lihat Latihan 7.11).

Cukup mirip adalah hubungan antara dua variabel J dan Bur, karena John tidak bereaksi terhadap pencurian, melainkan alarm. Ini bisa jadi, misalnya, karena tembok tinggi yang menghalangi pandangannya ke properti Bob, tapi dia masih bisa mendengar alarmnya. Jadi J dan Bur independen diberikan Al dan

$$\boldsymbol{P}(J, Bur|Al) = \boldsymbol{P}(J|Al) \;.\; \boldsymbol{P}(Bur|Ai)$$

Diberi alarm, variabel J dan Ear, M dan Bur, serta M dan Ear juga independen. Untuk komputasi dengan independensi bersyarat, karakterisasi berikut, yang setara dengan definisi di atas, sangat membantu:

Teorema 7.5

Persamaan berikut adalah ekuivalen berpasangan, yang berarti bahwa setiap persamaan individu menggambarkan independensi bersyarat untuk variabel A dan B yang diberikan C.

Persamaan 7.18

$$\boldsymbol{P}(A, B|C) = \boldsymbol{P}(A|C) \;.\; \boldsymbol{P}(B|C),$$

Persamaan 7.19

$$\boldsymbol{P}(A|B, C) = \boldsymbol{P}(A|C)$$

Persamaan 7.20

$$\boldsymbol{P}(B|A, C) = \boldsymbol{P}(B|C)$$

Bukti Di satu sisi, dengan menggunakan independensi bersyarat (7.18) kita dapat menyimpulkan bahwa

$$\boldsymbol{P}(A, B, C) = \boldsymbol{P}(A, B|C)\boldsymbol{P}(C) = \boldsymbol{P}(A|C)\boldsymbol{P}(B|C)\boldsymbol{P}(C)$$

Di sisi lain, aturan produk memberi kita

$$\boldsymbol{P}(A, B, C) = \boldsymbol{P}(A|B, C)\boldsymbol{P}(B|C)\boldsymbol{P}(C)$$

Jadi **P**(A|B, C) = **P**(A|C) ekuivalen dengan (7.18). Kami memperoleh (7.20) secara analog dengan menukar A dan B dalam derivasi ini.

Aplikasi praktis

Sekarang kita kembali ke contoh alarm dan menunjukkan bagaimana jaringan Bayesian pada Gambar 7.11 dapat digunakan untuk penalaran. Bob tertarik, misalnya, pada kepekaan dua reporter alarmnya John dan Mary, yaitu, dalam P(J|Bur) dan P(M|Bur). Namun, nilai P(Bur|J) dan P(Bur|M), serta P(Bur|M) bahkan lebih penting baginya. Kita mulai dengan P(J|Bur) dan hitung

Persamaan 7.21

$$P(J|Bur) = \frac{P(J, Bur)}{P(Bur)} = \frac{P(J, Bur, Al) \ + \ P(J, Bur, \neg Al)}{P(Bur)}$$

Persamaan 7.22

P(J,Bur, Al) = P(J|Bur, Al)P(Al|Bur) = P(J|Al)P(Al|Bur)P(Bur)

dimana untuk dua persamaan terakhir kita telah menggunakan aturan perkalian dan independensi bersyarat J dan Bur diberikan Al. Dimasukkan ke dalam (7.21) kita peroleh

$$P(J|Bur) = \frac{P(J|Al)P(Al|Bur) + P(J|\neg Al)P(\neg Al|Bur)P(Bur)}{P(Bur)}$$
$$= P(J|Al)P(Al|Bur) \ + \ P(J|\neg Al)P(\neg Al|Bur)$$

Di sini P(Al|Bur) dan P(¬Al|Bur) tidak ada. Oleh karena itu kami menghitung

$$P(J|Bur) = \frac{P(Al|Bur)}{P(Bur)}$$
$$= \frac{P(Al, Bur, Ear) + P(Al, Bur, \neg Ear)}{P(Bur)}$$
$$= \frac{P(Al, Bur, Ear)P(Bur)P(Ear) + P(Al, Bur, \neg Ear)P(Bur)P(\neg Ear)}{P(Bur)}$$
$$= P(Al|Bur, Ear)P(Ear) + P(Al|Bur, \neg Ear)P(\neg Ear)$$
$$= 0.95 \ . \ 0.002 \ + \ 0.94 \ . \ 0.998 \ = \ 0.94$$

serta P(¬Al|Bur) = 0,06 dan masukkan ini ke dalam (7.23) yang memberikan hasil

P(J|Bur) = 0.9 . 0.94 + 0.05 . 0.06 = 0.849

Secara analog kita hitung P(M|Bur) = 0,659. Kita sekarang tahu bahwa John meminta sekitar 85% dari semua pembobolan dan Mary untuk sekitar 66% dari semua pembobolan. Probabilitas bahwa keduanya memanggil dihitung, karena independensi bersyarat, sebagai

P(J, M|Bur) = P(J, M|Al)P(Al|Bur) + P(J, M|¬Al)P(¬Al|Bur)
= P(J|Al)P(M|Al)P(Al|Bur) + P(J|¬Al)P(M|¬Al)P(¬A|Bur)
= 0.9 . 0.7 0.94 + 0.05 . 0.01 . 0.06 = 0.5922

Lebih menarik, bagaimanapun, adalah kemungkinan panggilan dari John atau Mary

P(J ∨ M|Bar) = P(¬(¬J, ¬M|Bur)
= 1 – P(¬J, ¬M|Bar)
= 1 – [P(¬J|Al)P(¬M|Al)P(Al|Bur) + P(¬J|¬Al)P(¬Al|Bur)]
= 1 – [0.1 . 0.3 0.94 + 0.95 . 0.99 . 0.06]
= 1 – 0.085 = 0.915

Bob dengan demikian menerima pemberitahuan untuk sekitar 92% dari semua perampokan. Sekarang untuk menghitung P(Bur|J) , kita terapkan teorema Bayes, yang memberikan kita

$$P(Bur|J) = \frac{P(J|Bur)P(Bur)}{P(J)} = \frac{0.849 \ . \ 0.001}{0.052} = 0.016$$

Terbukti hanya sekitar 1,6% dari semua panggilan dari John sebenarnya karena pembobolan. Karena probabilitas alarm palsu lima kali lebih kecil untuk Mary, dengan P(Bur|M) 0.056, kita memiliki keyakinan yang jauh lebih tinggi saat menerima panggilan dari Mary. Bob seharusnya benar-benar memperhatikan rumahnya jika keduanya menelepon, karena P(Bur|J, M) = 0,284 (lihat Latihan 7.11). Dalam (7.23) kami menunjukkan dengan

P(J|Bur) = P(J|Al)P(Al|Bur) + P(J|¬Al)P(¬Al|Bur)

bagaimana kita bisa "slide in" variabel baru. Hubungan ini berlaku secara umum untuk dua variabel A dan B yang diberikan pengenalan variabel tambahan C dan disebut **pengkondisian**:

$$P(A|B) = \sum_c P(A|B, C = c)P(C = c|B)$$

Jika selanjutnya A dan B bebas bersyarat diberikan C, rumus ini disederhanakan menjadi:

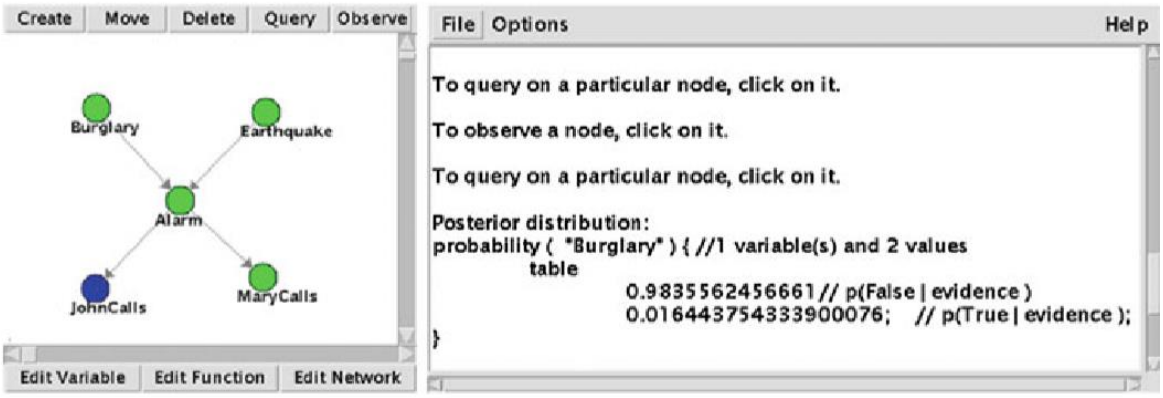
$$P(A|B) = \sum_c P(A|C = c)P(C = c|B)$$

7.18 PERANGKAT LUNAK UNTUK JARINGAN BAYESIAN

Kami akan memberikan pengantar singkat untuk dua alat menggunakan contoh alarm. Kita sudah tidak asing lagi dengan sistem PIT. Kami memasukkan nilai dari CPT dalam sintaks PIT ke jendela input online di www.pit-systems.de. Setelah input yang ditunjukkan pada Gambar 7.12 kami menerima jawabannya:
P([Einbruch=t] | [John=t] AND [Mary=t]) = 0.2841.

```
 1 var Alarm{t,f}, Burglary{t,f}, Earthquake{t,f}, John{t,f}, Mary{t,f};
 2
 3 P([Earthquake=t]) = 0.002;
 4 P([Burglary=t]) = 0.001;
 5 P([Alarm=t] | [Burglary=t] AND [Earthquake=t]) = 0.95;
 6 P([Alarm=t] | [Burglary=t] AND [Earthquake=f]) = 0.94;
 7 P([Alarm=t] | [Burglary=f] AND [Earthquake=t]) = 0.29;
 8 P([Alarm=t] | [Burglary=f] AND [Earthquake=f]) = 0.001;
 9 P([John=t] | [Alarm=t]) = 0.90;
10 P([John=t] | [Alarm=f]) = 0.05;
11 P([Mary=t] | [Alarm=t]) = 0.70;
12 P([Mary=t] | [Alarm=f]) = 0.01;
13
14 QP([Burglary=t] | [John=t] AND [Mary=t]);
```

Gambar 7.12 Input PIT untuk contoh alarm



Gambar 7.13 Antarmuka pengguna JavaBayes: kiri editor grafis dan kanan konsol tempat jawaban diberikan sebagai output

Meskipun PIT bukan alat jaringan Bayesian klasik, PIT dapat mengambil probabilitas dan kueri bersyarat arbitrer sebagai input dan menghitung hasil yang benar. Hal ini dapat ditunjukkan, bahwa pada input CPT beberapa aturan ekuivalen, prinsip Max Ent mengimplikasikan independensi bersyarat yang sama dan dengan demikian juga jawaban yang sama sebagai jaringan Bayesian. Jaringan Bayesian dengan demikian merupakan kasus khusus dari MaxEnt.

Selanjutnya kita akan melihat JavaBayes, sebuah sistem klasik yang juga tersedia secara bebas di Internet dengan antarmuka grafis yang ditunjukkan pada Gambar 7.13. Dengan editor jaringan grafis, node dan edge dapat dimanipulasi dan nilai dalam CPT dapat diedit. Selanjutnya, nilai variabel dapat ditetapkan dengan "Amati" dan nilai variabel lain disebut dengan "Kueri". Jawaban atas pertanyaan kemudian muncul di jendela konsol.
Sistem komersial dan profesional Hugin secara signifikan lebih kuat dan nyaman. Misalnya, Hugin dapat menggunakan variabel kontinu selain variabel diskrit. Itu juga dapat mempelajari jaringan Bayesian, yaitu, menghasilkan jaringan sepenuhnya secara otomatis dari data statistik.

7.19 PENGEMBANGAN JARINGAN BAYESIAN

Jaringan Bayesian yang kompak sangat jelas dan secara signifikan lebih informatif bagi pembaca daripada distribusi probabilitas penuh. Selain itu, ini membutuhkan lebih sedikit memori. Untuk variabel v_1, ... ,v_n dengan $|v_1|$, ...,$|v_n|$ nilai yang berbeda masing-masing, distribusi memiliki total entri independen. Dalam contoh alarm, semua variabel adalah biner. Jadi untuk semua variabel $|v_i| = 2$, dan distribusi memiliki $2^5 - 1 = 31$ entri independen. Untuk menghitung jumlah entri independen untuk jaringan Bayesian, jumlah total semua entri dari semua CPT harus ditentukan. Untuk simpul v_i dengan k_i simpul induk e_{i1}, ...,e_{iki}, CPT terkait memiliki entri. Kemudian semua CPT dalam jaringan bersama-sama memiliki entri[34].

$$\prod_{i=1}^{n} |v_i| - 1$$

$$(|v_i| - 1) \prod_{j=1}^{k_i} |e_{ij}|$$

[34] Untuk kasus simpul tanpa leluhur, produk dalam jumlah ini kosong. Untuk ini kami mengganti nilai 1 karena CPT untuk node tanpa ancestor mengandung, dengan probabilitas apriori, tepat satu nilai.

Persamaan 7.24

$$\sum_{i=1}^{n}(|v_i| - 1)\prod_{j=1}^{k_i}|e_{ij}|$$

Untuk contoh alarm, hasilnya adalah entri independen yang secara unik menggambarkan jaringan. Perbandingan kompleksitas memori antara distribusi penuh dan jaringan Bayesian menjadi lebih jelas ketika kita mengasumsikan bahwa semua n variabel memiliki jumlah nilai b yang sama dan setiap node memiliki k node induk. Kemudian (7.24) dapat disederhanakan dan semua CPT bersama-sama memiliki $n(b - 1)b^k$ entri.

2 + 2 + 4 + 1+ 1 = 10

Distribusi lengkap berisi $b^n - 1$ entri. Sebuah keuntungan yang signifikan hanya dibuat kemudian jika jumlah rata-rata node induk jauh lebih kecil dari jumlah variabel. Ini berarti bahwa node hanya terhubung secara lokal. Karena koneksi lokal, jaringan menjadi termodulasi, yang—seperti dalam rekayasa perangkat lunak—mengakibatkan pengurangan kompleksitas. Dalam contoh alarm, node alarm memisahkan node Bur dan Ear dari node J dan M. Kita juga dapat melihat ini dengan jelas dalam contoh LEXMED.

LEXMED sebagai Jaringan Bayesian Sistem LEXMED juga dapat dimodelkan sebagai jaringan Bayesian. Dengan membuat garis-garis luar yang digambar tipis menjadi terarah (dengan memberikan panah), grafik independensi pada Gambar 7.8 dapat diinterpretasikan sebagai jaringan Bayesian. Jaringan yang dihasilkan ditunjukkan pada Gambar 7.14.

Ukuran distribusi untuk LEXMED dihitung sebagai nilai 20643839. Jaringan Bayesian di sisi lain dapat sepenuhnya dijelaskan dengan hanya 521 nilai. Nilai ini dapat ditentukan dengan memasukkan variabel dari Gambar 7.14 ke (7.24). Dalam urutan (Leuko, TRek, Gua, Age, Reb, Sono, Tapp, BowS, Sex, P4Q, P1Q, P2Q, RecP , Urin, P3Q, Diag4) kami menghitung Contoh ini menunjukkan bahwa secara praktis tidak mungkin membangun distribusi penuh untuk aplikasi nyata. Jaringan Bayesian dengan 22 tepi dan 521 nilai probabilitas di sisi lain masih dapat dikelola.

$$\sum_{i=1}^{n}(|v_i|-1)\prod_{j=1}^{k_i}|e_{ij}|$$

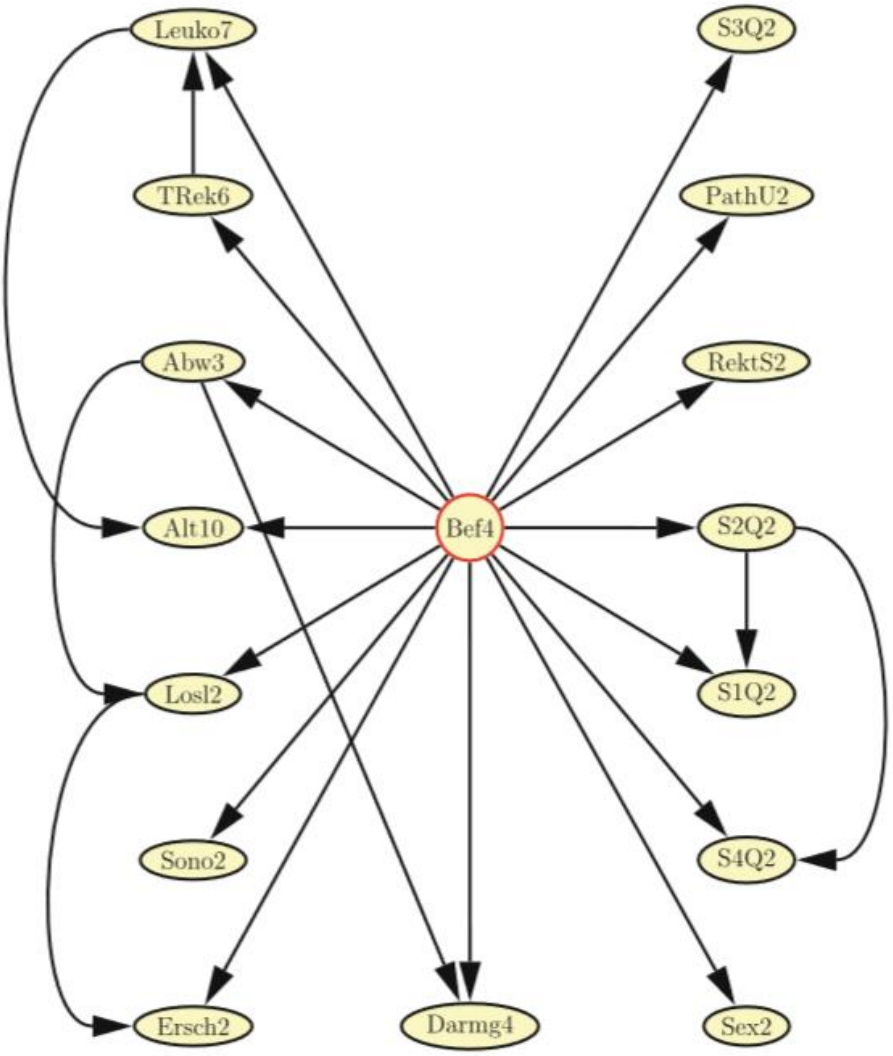
$$= 6 . 6 . 4 + 5 . 4 + 2 . 4 + 9 . 7 . 4 + 1 . 4 + 1 . 2 . 4 + 3 . 3 . 4 + 1 . 4 + 1 . 4 . 2 + 1 . 4 . 2 + 1 . 4 + 1 . 4 + 1 . 4 + 1 . 4 + 1 = 521.$$

Kausalitas dan Struktur Jaringan

Konstruksi jaringan Bayesian biasanya berlangsung dalam dua tahap.

1. Desain struktur jaringan: Langkah ini biasanya dilakukan secara manual dan akan dijelaskan berikut ini.
2. Memasukkan probabilitas dalam CPT: Memasukkan nilai secara manual untuk banyak variabel sangat membosankan. Jika (seperti misalnya dengan LEXMED) database tersedia, langkah ini dapat diotomatisasi dengan memperkirakan entri CPT dengan menghitung frekuensi.

Kami sekarang akan menjelaskan konstruksi jaringan dalam contoh alarm (lihat Gambar 7.15). Pada awalnya kita mengetahui dua penyebab Pencurian dan Gempa Bumi dan dua gejala John dan Mary. Namun, karena John dan Mary tidak langsung bereaksi terhadap pencuri atau gempa bumi, melainkan hanya alarm, maka tepat untuk menambahkan ini sebagai variabel tambahan yang tidak dapat diamati oleh Bob. Proses penambahan edge dimulai dengan cause, yaitu variabel yang tidak memiliki parent node.

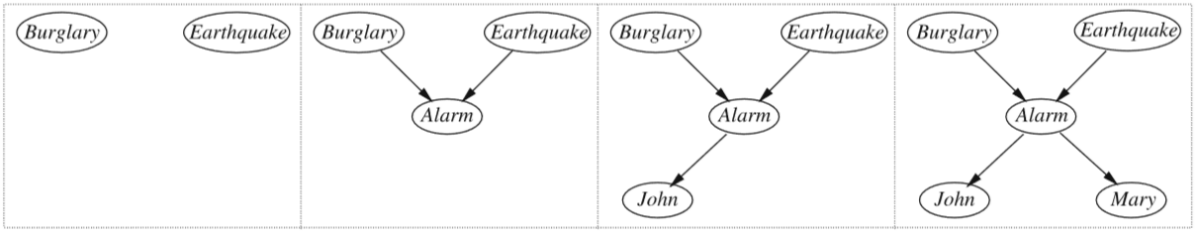


Gambar 7.14 Jaringan Bayesian untuk aplikasi LEXMED

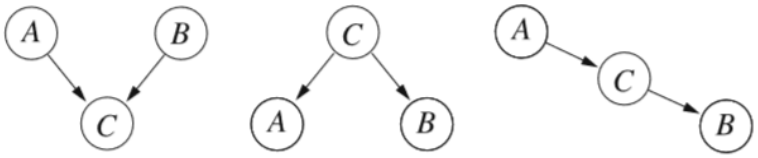
Pertama kita pilih Burglary dan selanjutnya Earthquake. Sekarang kita harus memeriksa apakah Gempa tidak tergantung pada Pencurian. Ini diberikan, dan dengan demikian tidak ada edge yang ditambahkan dari Burglary ke Earthquake. Karena Alarm secara langsung bergantung pada Burglary dan Earthquake, variabel ini dipilih berikutnya dan edge ditambahkan dari Burglary dan Earthquake ke Alarm. Kemudian kami memilih John. Karena Alarm dan John tidak independen, sebuah edge ditambahkan dari alarm ke John. Hal yang sama berlaku untuk Maria. Sekarang kita harus memeriksa apakah John independen bersyarat dari Alarm yang diberikan Pencurian. Jika tidak demikian, maka sisi lain harus dimasukkan dari Burglary ke John. Kita juga harus memeriksa apakah edge diperlukan dari Earthquake ke John dan dari Burglary atau Earthquake ke Mary. Karena independensi bersyarat, keempat sisi ini tidak diperlukan. Batas antara John dan Mary juga tidak diperlukan karena John dan Mary independen bersyarat karena diberi Alarm.

Struktur jaringan Bayesian sangat tergantung pada urutan variabel yang dipilih. Jika urutan variabel dipilih untuk mencerminkan hubungan kausal yang dimulai dengan penyebab dan berlanjut ke variabel diagnosis, maka hasilnya akan menjadi jaringan sederhana. Jika tidak, jaringan mungkin mengandung lebih banyak tepi secara signifikan. Jaringan non-kausal semacam itu seringkali sangat sulit untuk dipahami dan memiliki

kompleksitas penalaran yang lebih tinggi. Pembaca dapat merujuk ke Latihan 7.11 untuk pemahaman yang lebih baik.



Gambar 7.15 Konstruksi bertahap dari jaringan alarm dengan mempertimbangkan kausalitas



Gambar 7.16 Tidak ada edge antara A dan B jika keduanya bebas (kiri) atau bebas bersyarat (tengah, kanan)

7.20 SEMANTIK DARI *BAYESIAN NETWORKS*

Seperti yang telah kita lihat di bagian sebelumnya, tidak ada sisi yang ditambahkan ke jaringan Bayesian antara dua variabel A dan B ketika A dan B independen atau independen bersyarat jika diberikan variabel ketiga C. Situasi ini ditunjukkan pada Gambar 7.16. Kami sekarang membutuhkan jaringan Bayesian untuk tidak memiliki siklus dan kami berasumsi bahwa variabel diberi nomor sedemikian rupa sehingga tidak ada variabel yang memiliki angka lebih rendah daripada variabel apa pun yang mendahuluinya. Ini selalu mungkin ketika jaringan tidak memiliki siklus.[35] Kemudian, ketika menggunakan semua independensi bersyarat, kita memiliki

$$P(X_n|X_1, ..., X_{n-1}) = P(X_n|Keluarga(X_n))$$

Persamaan ini dengan demikian merupakan proposisi bahwa variabel arbitrer Xi dalam jaringan Bayesian adalah independen bersyarat dari nenek moyangnya, mengingat orang tuanya. Proposisi yang agak lebih umum yang digambarkan pada Gambar 7.17 dapat dinyatakan secara ringkas sebagai:

Teorema 7.6
Sebuah node dalam jaringan Bayesian secara kondisional independen dari semua node non-penerus, mengingat orang tuanya.

Sekarang kita dapat menyederhanakan aturan rantai ((Persamaan 7.1)):
Persamaan 7.25

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_1|X_1, \ldots, X_{i-1}) = \prod_{i=1}^{n} P(X_i|Keluarga(X_i))$$

Dengan menggunakan aturan ini, kita dapat, misalnya, menulis (7.22) secara langsung

$$\boldsymbol{P}(J, Bur, Al) = \boldsymbol{P}(J|Al)\boldsymbol{P}(Al|Bur)\boldsymbol{P}(Bur)$$

[35] Jika misalnya tiga simpul X1, X2, X3 membentuk suatu siklus, maka ada sisi-sisinya (X1, X2), (X2, X3) dan (X3, X1) dimana X1 memiliki X3 sebagai penerusnya.

Kita sekarang mengetahui konsep dan dasar paling penting dari jaringan Bayesian:

Definisi 7.7

Sebuah jaringan Bayesian didefinisikan oleh:

- Satu set variabel dan satu set tepi diarahkan antara variabel-variabel ini.
- Setiap variabel memiliki hingga banyak kemungkinan nilai.
- Variabel bersama-sama dengan tepi membentuk grafik asiklik berarah (DAG). DAG adalah graf tanpa siklus, yaitu tanpa jalur berbentuk (A, …, A).
- Untuk setiap variabel A CPT (yaitu, tabel probabilitas bersyarat P(A|Parents(A))) diberikan.

Dua variabel A dan B disebut bebas bersyarat diberikan C jika P(A, B|C) = P(A|C) . P(B|C) atau, ekuivalen, jika P(A|B, C) = P(A|C). Selain aturan dasar perhitungan probabilitas, aturan berikut juga benar:

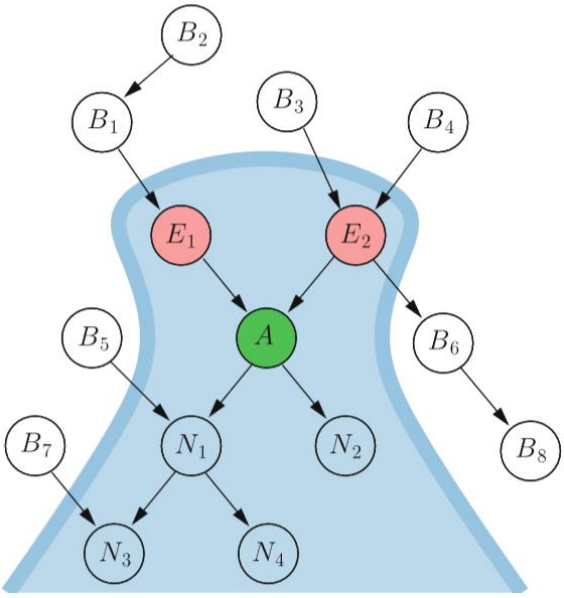
Teorema Bayes: $P(A|B) = \frac{P(B|A)\ .\ P(A)}{P(B)}$

Marjinalisasi: P(B) = P(A, B) + P(¬A, B)
= P(B|A) . P(A) + P(B|¬A) . P(¬A)

Pengkondisian: P(A|B) = $\sum_c$ P(A|B, C = c) P(C = c|B)

Sebuah variabel dalam jaringan Bayesian adalah independen bersyarat dari semua variabel non-penerus yang diberikan variabel induknya. Jika $X_1,\ldots,X_{n-1}$ bukan penerus X_n, kita memiliki $P(X_n|X_1,\ldots,X_{n-1})=P(X_n|\text{Induk}(X_n))$. Kondisi ini harus dihormati selama pembangunan jaringan. Selama konstruksi jaringan Bayesian, variabel harus diurutkan menurut kausalitas. Pertama penyebabnya, lalu variabel tersembunyi, dan terakhir variabel diagnosis.

Aturan rantai: $P(X_i,\ldots,X_n) = \prod_{i=1}^{n} P\ (X_i|Keluarga\ (X_i))$



Gambar 7.17 Contoh independensi bersyarat dalam jaringan Bayesian. Jika node induk E1 dan E2 diberikan, maka semua node non-penerus B1,…, B8 independen dari A

Istilah d-separation diperkenalkan untuk jaringan Bayesian, dari mana teorema yang mirip dengan Teorema 7.6 berikut. Kami akan menahan diri dari memperkenalkan istilah ini dan dengan demikian mencapai representasi yang agak lebih sederhana, meskipun secara teoritis tidak bersih.

7.21 RINGKASAN

Dengan cara yang mencerminkan tren yang berkepanjangan dan berkelanjutan menuju sistem probabilistik, kami telah memperkenalkan logika probabilistik untuk penalaran dengan pengetahuan yang tidak pasti. Setelah memperkenalkan bahasa—logika proposisional yang ditambah dengan probabilitas atau interval probabilitas—kami memilih pendekatan alami, jika tidak biasa melalui metode entropi maksimum sebagai titik masuk dan menunjukkan bagaimana kami dapat memodelkan penalaran non-monotonik dengan metode ini. Jaringan Bayesian kemudian diperkenalkan sebagai kasus khusus dari metode MaxEnt.

Mengapa jaringan Bayesian merupakan kasus khusus MaxEnt? Saat membangun jaringan Bayesian, asumsi tentang independensi dibuat yang tidak diperlukan untuk metode MaxEnt. Selanjutnya, ketika membangun jaringan Bayesian, semua CPT harus terisi penuh sehingga distribusi probabilitas lengkap dapat dibangun. Jika tidak, penalaran dibatasi atau tidak mungkin. Dengan MaxEnt, di sisi lain, pengembang dapat merumuskan semua pengetahuan yang dimilikinya dalam bentuk probabilitas. MaxEnt kemudian melengkapi model dan menghasilkan distribusi. Bahkan jika (misalnya ketika mewawancarai seorang ahli) hanya tersedia proposisi yang tidak jelas, ini dapat dimodelkan dengan tepat. Proposisi seperti "Saya cukup yakin bahwa A benar." misalnya dapat dimodelkan menggunakan $P(A) \in 0.6, 1]$ sebagai interval probabilitas. Saat membangun jaringan Bayesian, nilai konkret harus diberikan untuk probabilitas, jika perlu dengan menebak. Ini berarti, bagaimanapun, ahli atau pengembang memasukkan informasi ad hoc ke dalam sistem. Satu keuntungan lebih lanjut dari MaxEnt adalah kemungkinan merumuskan (hampir) proposisi sewenang-wenang. Untuk jaringan Bayesian, CPT harus diisi.

Kebebasan yang dimiliki pengembang saat membuat model dengan MaxEnt dapat menjadi kerugian (terutama bagi seorang pemula) karena, berbeda dengan pendekatan Bayesian, belum tentu jelas pengetahuan apa yang harus dimodelkan. Ketika pemodelan dengan jaringan Bayesian pendekatannya cukup jelas: menurut dependensi kausal, dari penyebab hingga efek, satu demi satu dimasukkan ke dalam jaringan dengan menguji independensi bersyarat.[36] Pada akhirnya semua CPT diisi dengan nilai.

Namun, kombinasi menarik dari dua metode berikut ini dimungkinkan: kita mulai dengan membangun jaringan menurut metodologi Bayesian, memasukkan semua tepi sesuai dengan itu dan kemudian mengisi CPT dengan nilai. Jika nilai tertentu untuk CPT tidak tersedia, maka nilai tersebut dapat diganti dengan interval atau dengan rumus logika probabilistik lainnya. Secara alami jaringan seperti itu—atau lebih baik: seperangkat aturan—tidak lagi memiliki semantik khusus jaringan Bayesian. Kemudian harus diproses dan diselesaikan oleh sistem MaxEnt.

Namun, kemampuan untuk menggunakan MaxEnt dengan kumpulan aturan arbitrer memiliki kelemahan. Sama halnya dengan situasi dalam logika, kumpulan aturan seperti itu bisa jadi tidak konsisten. Misalnya, dua aturan P(A) = 0,7 dan P(A) = 0,8 tidak konsisten. Sedangkan PIT sistem MaxEnt misalnya dapat mengenali inkonsistensi, jika tidak dapat memberikan petunjuk tentang cara menghilangkan masalah tersebut.

Kami memperkenalkan sistem pakar medis LEXMED, aplikasi klasik untuk penalaran dengan pengetahuan yang tidak pasti, dan menunjukkan bagaimana hal itu dapat dimodelkan dan diimplementasikan menggunakan jaringan MaxEnt dan Bayesian, dan

[36] Ini juga tidak selalu begitu sederhana.

bagaimana alat ini dapat menggantikan sistem penilaian linier yang sudah mapan, tetapi terlalu lemah yang digunakan dalam kedokteran.[37]

Dalam contoh LEXMED kami menunjukkan bahwa adalah mungkin untuk membangun sistem pakar untuk penalaran di bawah ketidakpastian yang mampu menemukan (belajar) pengetahuan dari data dalam database. Kami akan memberikan lebih banyak wawasan tentang metode pembelajaran jaringan Bayesian di Bab. 8, setelah fondasi yang diperlukan untuk pembelajaran mesin diletakkan.

Saat ini penalaran Bayesian adalah bidang yang luas dan independen, yang hanya dapat kami jelaskan secara singkat di sini. Kami benar-benar mengabaikan penanganan variabel kontinu. Untuk kasus variabel acak terdistribusi normal ada prosedur dan sistem. Untuk distribusi sewenang-wenang, bagaimanapun, kompleksitas komputasi adalah masalah besar. Selain jaringan terarah yang banyak didasarkan pada kausalitas, ada juga jaringan tidak terarah. Terkait dengan ini adalah diskusi tentang makna dan kegunaan kausalitas dalam jaringan Bayesian. Pembaca yang tertarik diarahkan ke buku teks yang sangat baik seperti [Pea88, Jen01, Whi96, DHS01], serta prosiding konferensi tahunan *Association for Uncertainty in Artificial Intelligence* (AUAI) (www.auai.org).

7.22 LATIHAN

Latihan 7.1

Buktikan proposisi dari Teorema 7.1.

Latihan 7.2

Penghobi berkebun Max ingin menganalisis secara statistik panen kacang polong tahunannya. Untuk setiap polong yang dia petik, dia mengukur panjangnya xi dalam sentimeter dan beratnya y_i dalam gram. Dia membagi kacang polong menjadi dua kelas, yang baik dan yang buruk (polong kosong). Data terukur (x_i, y_i) adalah

kacang polong yang baik:	x	1 2 2 3 3 4 4 5 6
	y	2 3 4 4 5 5 6 6 6
kacang polong buruk:	x	4 6 6 7
	y	2 2 3 3

(a) Dari data tersebut, hitunglah peluang P(y >3|Kelas = baik) dan P(y≤3| Kelas = baik). Kemudian gunakan rumus Bayes untuk menentukan P(Kelas = bagus|y > 3) dan P(Kelas = bagus|y≤ 3).

(b) Manakah dari probabilitas yang dihitung dalam submasalah (a) yang bertentangan dengan pernyataan “Semua kacang polong yang baik lebih berat dari 3 gram”?

Latihan 7.3

Anda seharusnya memprediksi cuaca sore menggunakan beberapa nilai cuaca sederhana dari pagi hari ini. Perhitungan probabilitas klasik untuk ini membutuhkan model yang lengkap, yang diberikan dalam tabel berikut.

Langit	Bar	Prec	P(Langit, Bar, Prec)
Cerah	Naik	Kering	0.40
Cerah	Naik	Hujaan	0.07
Cerah	Turun	Kering	0.08
Cerah	Turun	Hujan	0.10
Mendung	Naik	Kering	0.09
Mendung	Naik	Hujan	0.11

[37] Bagian dalam Latihan 8.17 kami akan menunjukkan bahwa skornya ekuivalen dengan kasus khusus naive Bayes, yaitu dengan asumsi bahwa semua gejala tidak tergantung pada diagnosis yang diberikan.

Mendung	Turun	Kering	0.03

Langit: Langit cerah atau berawan di pagi hari
Bar: Barometer naik atau turun di pagi hari
Sebelum: Hujan atau kering di sore hari

(a) Berapa banyak kejadian dalam distribusi ketiga variabel tersebut?
(b) Hitung P(Prec = kering|Langit = cerah, Bar = naik).
(c) Hitung P(Prec = hujan|Langit = mendung).
(d) Apa yang akan Anda lakukan jika baris terakhir hilang dari tabel?

Latihan 7.4

Dalam acara kuis televisi, kontestan harus memilih antara tiga pintu tertutup. Di balik satu pintu hadiah menanti: sebuah mobil. Di belakang kedua pintu lainnya adalah kambing. Kontestan memilih pintu, mis. nomor satu. Tuan rumah, yang tahu di mana mobil itu, membuka pintu lain, mis. nomor tiga, dan seekor kambing muncul. Kontestan sekarang diberi kesempatan untuk memilih antara dua pintu yang tersisa (satu dan dua). Apa pilihan yang lebih baik dari sudut pandangnya? Untuk tetap dengan pintu yang awalnya dia pilih atau beralih ke pintu tertutup lainnya?

Latihan 7.5

Dengan menggunakan metode pengali Lagrange, tunjukkan bahwa, tanpa kendala eksplisit, distribusi seragam $p_1 = p_2 = \ldots = p_n = 1/n$ mewakili entropi maksimum. Jangan lupa batasan yang selalu ada secara implisit $p_1 + p_2 + \ldots + p_n = 1$. Bagaimana kita bisa menunjukkan hasil yang sama ini menggunakan indiferen?

Latihan 7.6

Gunakan sistem PIT (http://www.pit-systems.de) atau SPIRIT (http://www.xspirit.de) untuk menghitung solusi MaxEnt untuk P(B) di bawah batasan $P(A) = \alpha$ dan $P(B|A) = \beta$. Kelemahan PIT mana, dibandingkan dengan perhitungan manual, yang Anda perhatikan di sini?

Latihan 7.7

Mengingat kendala $P(A) = \alpha$ dan $P(A \vee B) = \beta$, hitung P(B) secara manual menggunakan metode MaxEnt. Gunakan PIT untuk memeriksa solusi Anda.

Latihan 7.8

Diketahui kendala dari (7.10), (7.11), (7.12): $p_1 + p_2 = \alpha$, $p_1 + p_3 = \gamma$, $p_1 + p_2 + p_3 + p_4 = 1$. Tunjukkan bahwa $p_1 = \alpha\gamma$, $p_2 = \alpha(1 - \gamma)$, $p_3 = \gamma\ (1 - \alpha)$, $p_4 = (1-\alpha)(1 - \gamma)$ mewakili entropi maksimum di bawah batasan ini.

Latihan 7.9

Algoritme probabilistik menghitung kemungkinan p bahwa email masuk adalah spam. Untuk mengklasifikasikan email di kelas yang dihapus dan dibaca, matriks biaya kemudian diterapkan pada hasilnya.

(a) Berikan matriks biaya (matriks 2 × 2) untuk filter spam. Asumsikan di sini bahwa pengguna memerlukan biaya 10 sen untuk menghapus email, sedangkan kehilangan email memerlukan biaya 10 dolar (bandingkan dengan Contoh 1.1 dan Latihan 1.7).
(b) Tunjukkan bahwa, untuk kasus matriks 2 × 2, penerapan matriks biaya setara dengan penerapan ambang batas pada probabilitas spam dan tentukan ambang batasnya.

Latihan 7.10

Diberikan jaringan Bayesian dengan tiga variabel biner A, B, C dan P(A) = 0,2, P(B) = 0,9, serta CPT yang ditunjukkan di bawah ini:

(a) Hitung P(A |B).
(b) Hitung P(C|A).]

Latihan 7.11

Untuk contoh alarm (Contoh 7.10), hitung probabilitas bersyarat berikut:

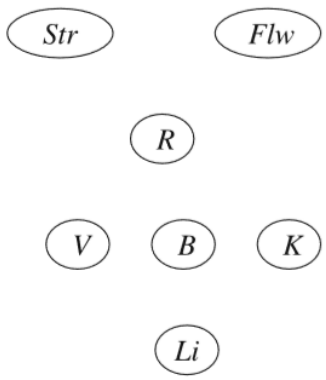
(a) Hitung probabilitas apriori P(Al), P(J), P(M).
(b) Hitung P(M|Bur) menggunakan aturan perkalian, marginalisasi, aturan rantai, dan independensi bersyarat.
(c) Gunakan rumus Bayes untuk menghitung P(Bur|M)
(d) Hitung P(Al|J, M) dan P(Bur|J, M).
(e) Tunjukkan bahwa variabel J dan M tidak bebas.
(f) Periksa semua hasil Anda dengan JavaBayes dan dengan PIT (lihat [Ert11] untuk program demo).
(g) Rancang jaringan Bayesian untuk contoh alarm, tetapi dengan urutan variabel yang diubah M, Al, Ear, Bur, J. Menurut semantik jaringan Bayesian, hanya tepi yang diperlukan yang harus digambar. (Petunjuk: variabel urutan yang diberikan di sini TIDAK mewakili kausalitas, sehingga akan sulit untuk secara intuitif menentukan independensi bersyarat.)
(h) Dalam jaringan Bayesian asli dari contoh alarm, node gempa dihilangkan. CPT mana yang berubah? (Mengapa ini khususnya?)
(i) Hitung CPT dari node alarm di jaringan baru.

Latihan 7.12

Sistem diagnostik harus dibuat untuk lampu sepeda bertenaga dinamo menggunakan jaringan Bayesian. Variabel dalam tabel berikut diberikan.

Abb.	***Maksud***	***Nilai***
Li	Lampu menyala	*t/f*
Str	Kondisi Jalan	*kering, basah, gerimis*
Flw	Dinamo roda aus	*t/f*
R	Dinamo roda geser	*t/f*
V	Dinamo menunjukkan Voltase	*t/f*
B	Lampu menyala ok	*t/f*
K	Kabel ok	*t/f*

Variabel-variabel berikut adalah bebas berpasangan: Str, Flw, B, K. Selanjutnya: (R, B), (R, K), (V, B), (V,K) adalah bebas dan persamaan berikut berlaku:



V	*B*	*K*	*P(Li)*
t	*t*	*t*	0.99
t	*t*	*f*	0.01
t	*f*	*t*	0.01
t	*f*	*f*	0.001
f	*t*	*t*	0.3
f	*t*	*f*	0.005
f	*f*	*t*	0.005
f	*f*	*f*	0

P(Li|V, R) = P(Li|V)
P(V|R, Str) = P(V|R)
P(V|R, Flw) = P(V|R)

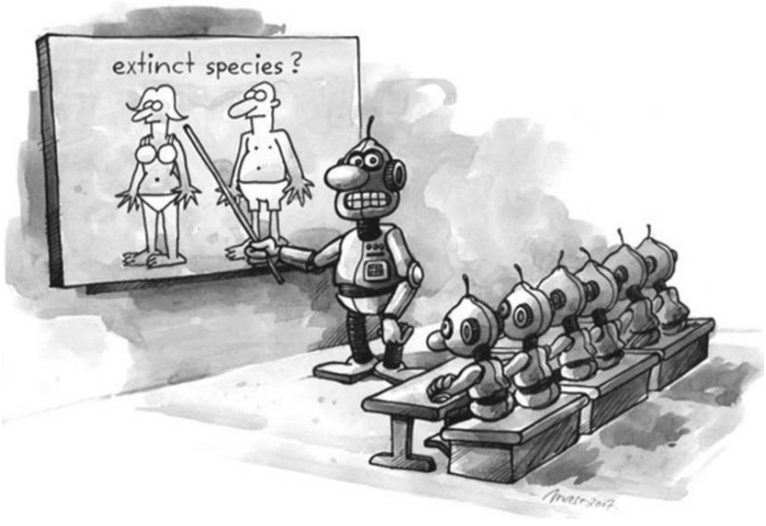
(a) Gambarkan semua sisi ke dalam grafik (dengan mempertimbangkan kausalitas).
(b) Masukkan semua CPT yang hilang ke dalam grafik (tabel probabilitas bersyarat). Masukkan secara bebas nilai yang masuk akal untuk probabilitas.
(c) Tunjukkan bahwa jaringan tidak mengandung sisi (Str, Li).
(d) Hitung P(V|Str = gerimis).

BAB 8
MACHINE *LEARNING* DAN *DATA MINING*

Jika kita mendefinisikan AI seperti yang dilakukan dalam buku Elaine Rich: *Kecerdasan Buatan adalah studi tentang bagaimana membuat komputer melakukan hal-hal di mana, pada saat ini, orang lebih baik.* dan jika kita menganggap bahwa kemampuan belajar komputer lebih rendah daripada manusia, maka penelitian tentang mekanisme pembelajaran, dan pengembangan algoritme pembelajaran mesin adalah salah satu cabang AI yang paling penting.

Ada juga permintaan untuk pembelajaran mesin dari sudut pandang pengembang perangkat lunak yang memprogram, misalnya, perilaku robot otonom. Struktur perilaku cerdas dapat menjadi sangat kompleks sehingga sangat sulit atau bahkan tidak mungkin untuk diprogram mendekati secara optimal, bahkan dengan bahasa tingkat tinggi modern seperti PROLOG dan Python.[38] Algoritma pembelajaran mesin bahkan digunakan saat ini untuk memprogram robot dalam cara yang mirip dengan bagaimana manusia, seringkali dalam campuran hibrida dari perilaku yang diprogram dan dipelajari.

Tugas bab ini adalah menjelaskan algoritma pembelajaran mesin yang paling penting dan aplikasinya. Topik akan diperkenalkan di bagian ini, diikuti oleh algoritma pembelajaran dasar yang penting di bagian berikutnya. Teori dan terminologi akan dibangun secara paralel untuk ini. Bab ini akan ditutup dengan ringkasan dan gambaran umum tentang berbagai algoritma dan aplikasinya. Kami akan membatasi diri dalam bab ini untuk algoritma pembelajaran terawasi dan tidak terawasi. Sebagai kelas penting dari algoritma pembelajaran, jaringan saraf akan dibahas dalam Bab. 9. Karena tempat khusus dan peran penting robot otonom, pembelajaran penguatan juga akan memiliki bab tersendiri (Bab 10).



Gambar 8.1 Pembelajaran terawasi

8.1 APA ITU *LEARNING*?

Mempelajari kosakata bahasa asing, atau istilah teknis, atau bahkan menghafal puisi bisa jadi sulit bagi banyak orang. Namun, untuk komputer, tugas-tugas ini cukup

[38] Python adalah bahasa scripting modern dengan sintaks yang sangat mudah dibaca, tipe data yang kuat, dan perpustakaan standar yang luas, yang dapat digunakan untuk tujuan ini.

sederhana karena tidak lebih dari sekadar menyimpan teks dalam sebuah file. Jadi menghafal tidak menarik untuk AI. Sebaliknya, perolehan keterampilan matematika biasanya tidak dilakukan dengan menghafal. Untuk penjumlahan bilangan asli, ini tidak mungkin, karena untuk setiap suku dalam penjumlahan + y ada banyak nilai tak hingga. Untuk setiap kombinasi dua nilai x dan y, tiga kali lipat (x, y, x + y) harus disimpan, yang tidak mungkin. Untuk angka desimal, ini benar-benar tidak mungkin. Ini menimbulkan pertanyaan: bagaimana kita belajar matematika? Jawabannya berbunyi: Guru menjelaskan proses dan siswa mempraktikkannya pada contoh-contoh sampai mereka tidak lagi membuat kesalahan pada contoh-contoh baru. Setelah 50 contoh siswa (semoga) mengerti penjumlahan. Artinya, setelah hanya 50 contoh, dia dapat menerapkan apa yang telah dipelajari ke banyak contoh baru, yang sampai saat itu tidak terlihat. Proses ini dikenal sebagai generalisasi. Kita mulai dengan contoh sederhana.

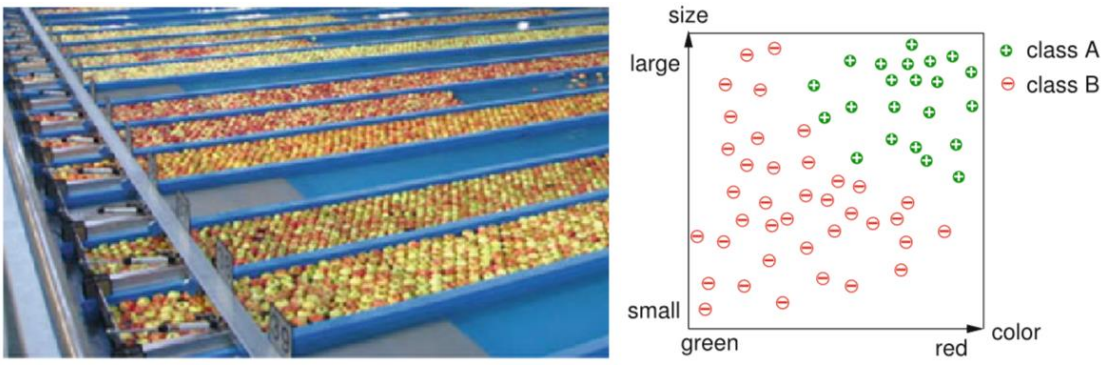
Contoh 8.1

Seorang petani buah ingin secara otomatis membagi apel yang dipanen ke dalam kelas barang dagangan A dan B. Perangkat penyortiran dilengkapi dengan sensor untuk mengukur dua fitur, ukuran dan warna, dan kemudian memutuskan yang mana dari dua kelas apel tersebut. Ini adalah tugas klasifikasi yang khas. Sistem yang mampu membagi vektor fitur menjadi sejumlah kelas yang terbatas disebut pengklasifikasi.

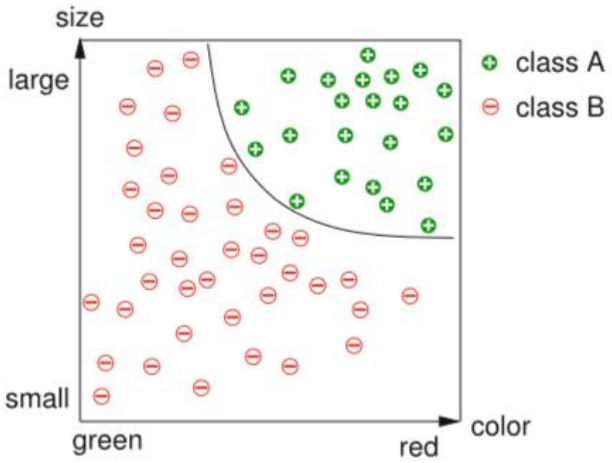
Untuk mengonfigurasi mesin, apel dipetik dengan tangan oleh seorang spesialis, yaitu, diklasifikasikan. Kemudian kedua ukuran tersebut dimasukkan bersama-sama dengan label kelasnya ke dalam sebuah tabel (Tabel 8.1. Ukuran diberikan dalam bentuk diameter dalam sentimeter dan warna dengan nilai numerik antara 0 (untuk hijau) dan 1 (untuk merah). visualisasi data tercantum sebagai titik-titik dalam diagram scatterplot di sebelah kanan Gambar 8.2.

Tabel 8.1 Data pelatihan untuk agen penyortiran apel

Size [cm]	8	8	6	3	…
Color	0.1	0.3	0.9	0.8	…
Merchandise class	B	A	A	B	…



Gambar 8.2 Peralatan penyortiran apel perusahaan BayWa di Kressbronn dan beberapa apel yang diklasifikasikan ke dalam kelas barang dagangan A dan B di ruang fitur (Foto: BayWa)



Gambar 8.3 Kurva yang digambar ke dalam diagram membagi kelas dan kemudian dapat diterapkan ke apel baru yang sewenang-wenang

Tugas dalam pembelajaran mesin terdiri dari menghasilkan fungsi dari data yang dikumpulkan dan diklasifikasikan yang menghitung nilai kelas (A atau B) untuk apel baru dari dua fitur ukuran dan warna. Pada Gambar 8.3 fungsi seperti itu ditunjukkan oleh garis pemisah yang ditarik melalui diagram. Semua apel dengan vektor ciri di kiri bawah garis dimasukkan ke dalam kelas B, dan yang lainnya ke dalam kelas A.

Dalam contoh ini masih sangat sederhana untuk menemukan garis pemisah seperti itu untuk dua kelas. Ini jelas merupakan tugas yang lebih sulit, dan di atas semua itu, jauh lebih tidak dapat divisualisasikan, ketika objek yang akan diklasifikasikan dijelaskan tidak hanya oleh dua, tetapi banyak fitur. Dalam praktiknya biasanya digunakan 30 atau lebih fitur. Untuk n fitur, tugas terdiri dari menemukan n – 1 dimensi hyperplane dalam ruang fitur n-dimensi yang membagi kelas sebaik mungkin. Pembagian "baik" berarti bahwa persentase objek yang diklasifikasikan salah adalah sekecil mungkin.

Gambar 8.4 Struktur fungsional agen pembelajaran untuk penyortiran apel (kiri) dan secara umum (kanan)

Pengklasifikasi memetakan vektor fitur ke nilai kelas. Di sini ia memiliki sejumlah alternatif yang tetap, biasanya kecil. Pemetaan yang diinginkan disebut juga fungsi target. Jika fungsi target tidak dipetakan ke domain berhingga, maka itu bukan klasifikasi, melainkan masalah aproksimasi. Menentukan nilai pasar saham dari fitur yang diberikan adalah masalah perkiraan. Pada bagian berikut kami akan memperkenalkan beberapa agen pembelajaran untuk kedua jenis pemetaan.

***Learning* Agent**

Kita dapat secara formal menggambarkan agen pembelajaran sebagai fungsi yang memetakan vektor fitur ke nilai kelas diskrit atau secara umum ke bilangan real. Fungsi ini

tidak diprogram, melainkan muncul atau berubah sendiri selama fase pembelajaran, dipengaruhi oleh data pelatihan. Pada Gambar 8.4 agen seperti itu disajikan dalam contoh penyortiran apel. Selama pembelajaran, agen diberi makan dengan data yang sudah diklasifikasikan dari Tabel 8.1. Setelah itu agen membuat pemetaan sebaik mungkin dari vektor fitur ke nilai fungsi (misalnya kelas barang dagangan).

Kita sekarang dapat mencoba mendekati definisi istilah "pembelajaran mesin". Tom Mitchell memberikan definisi ini:

Machine Learning adalah studi tentang algoritma komputer yang meningkat secara otomatis melalui pengalaman. Menggambar pada ini, kami memberikan

Definisi 8.1
Agen adalah agen pembelajaran jika meningkatkan kinerjanya (diukur dengan kriteria yang sesuai) pada data baru yang tidak diketahui dari waktu ke waktu (setelah melihat banyak contoh pelatihan).

Penting untuk menguji kemampuan generalisasi dari algoritma pembelajaran pada data yang tidak diketahui, data uji. Jika tidak, setiap sistem yang baru saja menyimpan data pelatihan akan tampil optimal hanya dengan memanggil data yang disimpan. Agen pembelajaran dicirikan oleh istilah-istilah berikut:

Task: tugas algoritma pembelajaran adalah mempelajari pemetaan. Ini bisa misalnya pemetaan dari ukuran dan warna apel ke kelas barang dagangannya, tetapi juga pemetaan dari 15 gejala pasien hingga keputusan apakah usus buntunya diangkat atau tidak.

Agen variabel: (lebih tepatnya kelas agen): di sini kita harus memutuskan algoritma pembelajaran mana yang akan digunakan. Jika ini telah dipilih, maka kelas dari semua fungsi yang dapat dipelajari ditentukan.

Training Data: (pengalaman): data pelatihan (sampel) berisi pengetahuan yang seharusnya diambil dan dipelajari oleh algoritma pembelajaran. Dengan pilihan data pelatihan, seseorang harus memastikan bahwa itu adalah sampel yang representatif untuk tugas yang akan dipelajari.

Test Data: penting untuk mengevaluasi apakah agen terlatih dapat menggeneralisasi dengan baik dari data pelatihan ke data baru.

Performance Measure untuk perangkat penyortiran apel, jumlah apel yang diklasifikasikan dengan benar. Kami membutuhkannya untuk menguji kualitas agen. Mengetahui ukuran kinerja biasanya jauh lebih mudah daripada mengetahui fungsi agen. Misalnya, mudah untuk mengukur kinerja (waktu) seorang pelari 10.000 meter. Namun, ini sama sekali tidak menyiratkan bahwa wasit yang mengukur waktu dapat berlari secepat itu. Wasit hanya tahu bagaimana mengukur kinerja, tetapi bukan "fungsi" agen yang kinerjanya dia ukur.

Gambar 8.5 *Data mining*

8.2 APA ITU *DATA MINING*?

Tugas Machine *Learning* untuk mengekstrak pengetahuan dari data pelatihan. Seringkali pengembang atau pengguna menginginkan Machine *Learning* untuk membuat pengetahuan yang diekstraksi dapat dibaca oleh manusia juga. Masih lebih baik jika pengembang bahkan dapat mengubah pengetahuan. Proses induksi pohon keputusan adalah contoh dari jenis metode ini.

Tantangan serupa datang dari bisnis elektronik dan manajemen pengetahuan. Masalah klasik muncul di sini: dari tindakan pengunjung ke portal webnya, pemilik bisnis Internet ingin membuat hubungan antara karakteristik pelanggan dan kelas produk yang menarik bagi pelanggan itu. Kemudian penjual akan dapat menempatkan iklan khusus pelanggan. Hal ini ditunjukkan dengan cara jitu di www.amazon.com, di mana pelanggan direkomendasikan produk yang serupa dengan yang terlihat pada kunjungan sebelumnya. Di banyak bidang periklanan dan pemasaran, serta dalam manajemen hubungan pelanggan (CRM), teknik *data mining* mulai digunakan. Kapanpun sejumlah besar data tersedia, seseorang dapat mencoba menggunakan data ini untuk analisis preferensi pelanggan untuk menampilkan iklan khusus pelanggan. Bidang pembelajaran preferensi yang muncul didedikasikan untuk tujuan ini.

> *Proses memperoleh pengetahuan dari data, serta representasi dan aplikasinya, disebut Data mining. Metode yang digunakan biasanya diambil dari statistik atau pembelajaran mesin dan harus dapat diterapkan pada jumlah data yang sangat besar dengan biaya yang wajar.*

Dalam konteks memperoleh informasi, misalnya di Internet atau di intranet, penambangan teks memainkan peran yang semakin penting. Tugas umum termasuk menemukan teks serupa di mesin pencari atau klasifikasi teks, yang misalnya diterapkan dalam filter spam untuk email. Kami akan memperkenalkan algoritma naive Bayes yang tersebar luas untuk klasifikasi teks. Tantangan yang relatif baru untuk *data mining* adalah ekstraksi informasi struktural, statis, dan dinamis dari struktur grafik seperti jaringan sosial, jaringan lalu lintas, atau lalu lintas Internet.

Karena dua tugas pembelajaran mesin dan *Data mining* yang dijelaskan secara formal sangat mirip, metode dasar yang digunakan di kedua area sebagian besar identik. Oleh karena itu, dalam deskripsi algoritma pembelajaran, tidak akan ada perbedaan antara pembelajaran mesin dan *Data mining*.

Karena dampak komersial yang besar dari teknik *data mining*, sekarang ada banyak pengoptimalan canggih dan seluruh lini sistem *data mining* yang kuat, yang menawarkan palet besar alat yang nyaman untuk ekstraksi pengetahuan dari data.

Analisis data

Statistik menyediakan sejumlah cara untuk menggambarkan data dengan parameter sederhana. Dari sini kami memilih beberapa yang sangat penting untuk analisis data pelatihan dan mengujinya pada subset data LEXMED. Dalam kumpulan data contoh ini, gejala x_1,…, x_{15} dari N = 473 pasien, secara ringkas dijelaskan pada Tabel 8.2, serta label kelas—yaitu diagnosis (apendisitis positif/negatif)—didaftarkan. Pasien nomor satu, misalnya, digambarkan oleh vektor

$$x^1 = (26, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 37.9, 38.8, 23100, 0, 1)$$

dan pasien nomor dua oleh

$$x^2 = (17, 2, 0, 0, 1, 0, 1, 0, 1, 1, 0, 36.9, 37.4, 8100, 0, 0)$$

Pasien nomor dua memiliki nilai leukosit $x_{14}^2 = 8100$. Untuk setiap variabel x_i, rata-rata x_i didefinisikan sebagai:

$$x_i := \frac{1}{N}\sum_{p=1}^{N} x_i^p$$

dan standar deviasi si sebagai ukuran deviasi rata-ratanya dari nilai rata-rata sebagai

$$s_i := \sqrt{\frac{1}{N-1}\sum_{p=1}^{N}(x_i^p - x_i)^2}$$

Pertanyaan apakah dua variabelxi dan xjar bergantung secara statistik (berkorelasi) penting untuk analisis data multidimensi. Misalnya, kovarians memberikan informasi tentang ini. Dalam jumlah ini, summand mengembalikan entri positif untuk vektor data ke-p tepat ketika deviasi komponen ke-i dan ke-j dari rata-rata keduanya memiliki tanda yang sama. Jika mereka memiliki tanda yang berbeda, maka entrinya negatif. Oleh karena itu kovarians $\sigma_{12,13}$ dari dua nilai demam yang berbeda harus cukup jelas positif.

Tabel 8.2 Deskripsi variabel x_1,…, x_{16}. Formalisasi yang sedikit berbeda digunakan pada Tabel 7.2

Var. num.	Description	Values
1	Age	Continuous
2	Sex (1 = male, 2 = female)	1, 2
3	Pain quadrant 1	0, 1
4	Pain quadrant 2	0, 1
5	Pain quadrant 3	0, 1
6	Pain quadrant 4	0, 1
7	Local muscular guarding	0, 1
8	Generalized muscular guarding	0, 1
9	Rebound tenderness	0, 1
10	Pain on tapping	0, 1
11	Pain during rectal examination	0, 1
12	Axial temperature	Continuous
13	Rectal temperature	Continuous
14	Leukocytes	Continuous
15	Diabetes mellitus	0, 1
16	Appendicitis	0, 1

Tabel 8.3 Matriks korelasi untuk 16 variabel apendisitis yang diukur pada 473 kasus

1.	−0.009	0.14	0.037	−0.096	0.12	0.018	0.051	−0.034	−0.041	0.034	0.037	0.05	−0.037	0.37	0.012
−0.009	1.	−0.0074	−0.019	−0.06	0.063	−0.17	0.0084	−0.17	−0.14	−0.13	−0.017	−0.034	−0.14	0.045	−0.2
0.14	−0.0074	1.	0.55	−0.091	0.24	0.13	0.24	0.045	0.18	0.028	0.02	0.045	0.03	0.11	0.045
0.037	−0.019	0.55	1.	−0.24	0.33	0.051	0.25	0.074	0.19	0.087	0.11	0.12	0.11	0.14	−0.0091
−0.096	−0.06	−0.091	−0.24	1.	0.059	0.14	0.034	0.14	0.049	0.057	0.064	0.058	0.11	0.017	0.14
0.12	0.063	0.24	0.33	0.059	1.	0.071	0.19	0.086	0.15	0.048	0.11	0.12	0.063	0.21	0.053
0.018	−0.17	0.13	0.051	0.14	0.071	1.	0.16	0.4	0.28	0.2	0.24	0.36	0.29	−0.0001	0.33
0.051	0.0084	0.24	0.25	0.034	0.19	0.16	1.	0.17	0.23	0.24	0.19	0.24	0.27	0.083	0.084
−0.034	−0.17	0.045	0.074	0.14	0.086	0.4	0.17	1.	0.53	0.25	0.19	0.27	0.27	0.026	0.38
−0.041	−0.14	0.18	0.19	0.049	0.15	0.28	0.23	0.53	1.	0.24	0.15	0.19	0.23	0.02	0.32
0.034	−0.13	0.028	0.087	0.057	0.048	0.2	0.24	0.25	0.24	1.	0.17	0.17	0.22	0.098	0.17
0.037	−0.017	0.02	0.11	0.064	0.11	0.24	0.19	0.19	0.15	0.17	1.	0.72	0.26	0.035	0.15
0.05	−0.034	0.045	0.12	0.058	0.12	0.36	0.24	0.27	0.19	0.17	0.72	1.	0.38	0.044	0.21
−0.037	−0.14	0.03	0.11	0.11	0.063	0.29	0.27	0.27	0.23	0.22	0.26	0.38	1.	0.051	0.44
0.37	0.045	0.11	0.14	0.017	0.21	−0.0001	0.083	0.026	0.02	0.098	0.035	0.044	0.051	1.	−0.0055
0.012	−0.2	0.045	−0.0091	0.14	0.053	0.33	0.084	0.38	0.32	0.17	0.15	0.21	0.44	−0.0055	1.

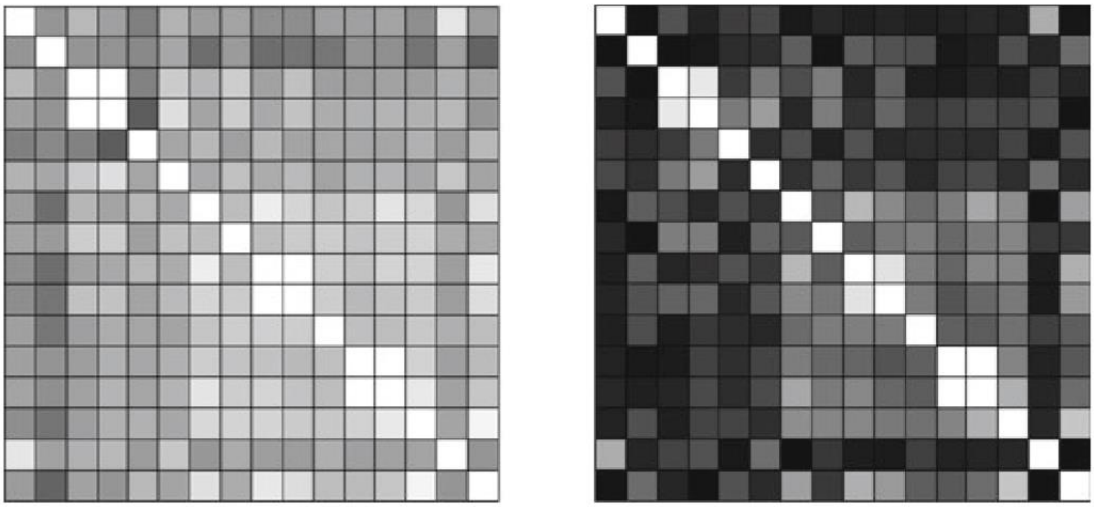
$$\sigma_{ij} = \frac{1}{N - 1} \sum_{p=1}^{N} x_i^p - x_i)(x_j^p - x_j)$$

Namun, kovarians juga bergantung pada nilai absolut variabel, yang membuat perbandingan nilai menjadi sulit. Untuk dapat membandingkan tingkat ketergantungan dalam kasus beberapa variabel, oleh karena itu kami mendefinisikan koefisien korelasi

$$K_{ij} = \frac{\sigma_{ij}}{s_i \ . \ s_j}$$

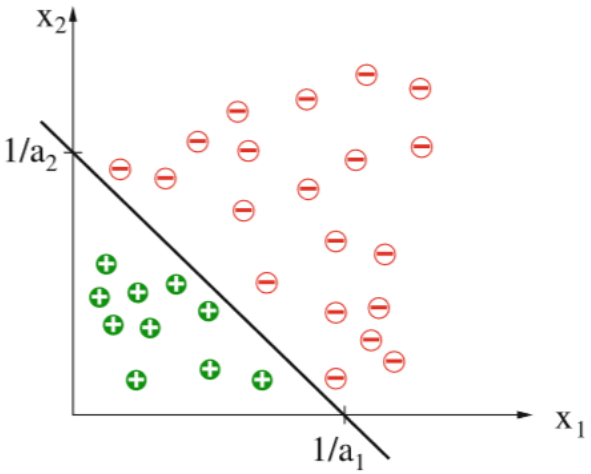
untuk dua nilai xi dan xj, yang tidak lain adalah kovarians yang dinormalisasi. Matriks K dari semua koefisien korelasi mengandung nilai antara – 1 dan 1, simetris, dan semua elemen diagonalnya bernilai 1. Matriks korelasi untuk ke-16 variabel diberikan pada Tabel 8.3.

Matriks ini menjadi agak lebih mudah dibaca ketika kami merepresentasikannya sebagai plot kepadatan. Alih-alih nilai numerik, elemen matriks pada Gambar 8.6 diisi dengan nilai abu-abu. Dalam diagram yang tepat, nilai-nilai absolut ditampilkan. Dengan demikian kita dapat dengan cepat melihat variabel mana yang menunjukkan ketergantungan yang lemah atau kuat. Kita dapat melihat, misalnya, bahwa variabel 7, 9, 10 dan 14 menunjukkan korelasi yang paling kuat dengan variabel kelas apendisitis dan oleh karena itu lebih penting untuk diagnosis daripada variabel lainnya. Kami juga melihat, bagaimanapun, bahwa variabel 9 dan 10 berkorelasi kuat. Ini bisa berarti bahwa salah satu dari dua nilai ini berpotensi cukup untuk diagnosis.



Gambar 8.6 Matriks korelasi sebagai grafik frekuensi.

Pada diagram kiri, gelap berarti negatif dan terang berarti positif. Di gambar kanan, nilai absolut dicantumkan. Di sini hitam berarti $K_{ij} \approx 0$ (tidak berkorelasi) dan putih $|K_{ij}| \approx 1$ (sangat berkorelasi)



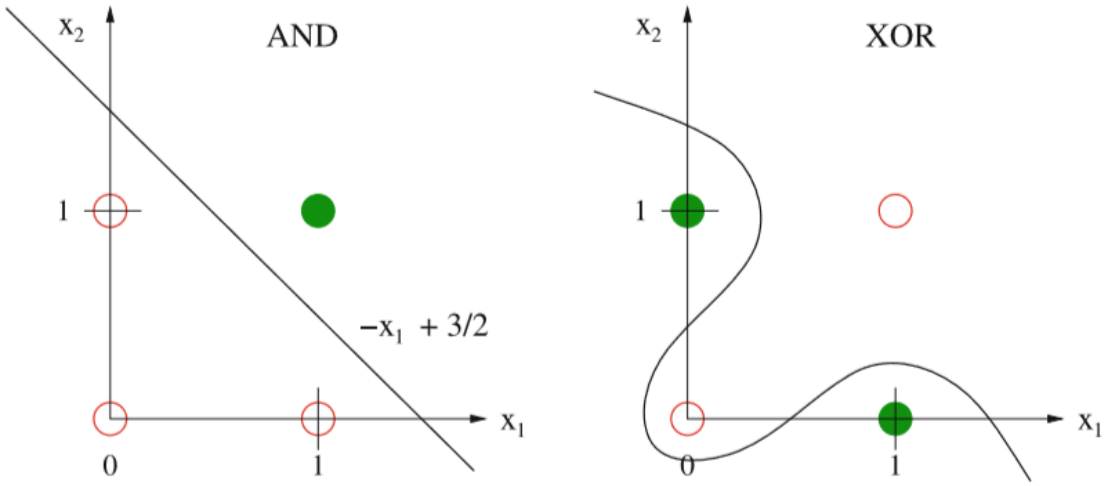
Gambar 8.7 Kumpulan data dua dimensi yang dapat dipisahkan secara linier. Persamaan garis lurus pembagi adalah $a_1x_1 + a_2x_2 = 1$

8.3 *PERCEPTRON*, PENGKLASIFIKASI *LINIER*

Dalam contoh klasifikasi penyortiran apel, garis pemisah melengkung digambar antara dua kelas pada Gambar 8.3. Kasus yang lebih sederhana ditunjukkan pada Gambar 8.7. Di sini contoh pelatihan dua dimensi dapat dipisahkan oleh garis lurus. Kami menyebut kumpulan data pelatihan seperti itu dapat dipisahkan secara linier. Dimensi Inn hyperplane diperlukan untuk pemisahan. Ini merepresentasikan subruang linier berdimensi n – 1. Karena setiap hyperplane (n - 1) dimensi dalam n dapat dijelaskan dengan persamaan

$$\sum_{i=1}^{n} a_i \, x_i \; = \; 0$$

masuk akal untuk mendefinisikan keterpisahan linier sebagai berikut.



Gambar 8.8 Fungsi boolean AND dapat dipisahkan secara linear, tetapi XOR tidak benar

(true) ○ salah(false) ●

Definisi 8.2
Dua himpunan $M_1 \subset \mathbb{R}^n$ dan $M_2 \subset \mathbb{R}^n$ disebut terpisah linier jika bilangan real $a_1, \ldots, a_n, \theta$ ada dengan

$$\sum_{i=1}^{n} a_i x_i > \theta \; untuk \; semua \; x \in M_1 \; dan \; \sum_{i=1}^{n} a_i x \, i \leq \theta \; untuk \; semua \; x \in M_2$$

Nilai θ dilambangkan dengan ambang batas.

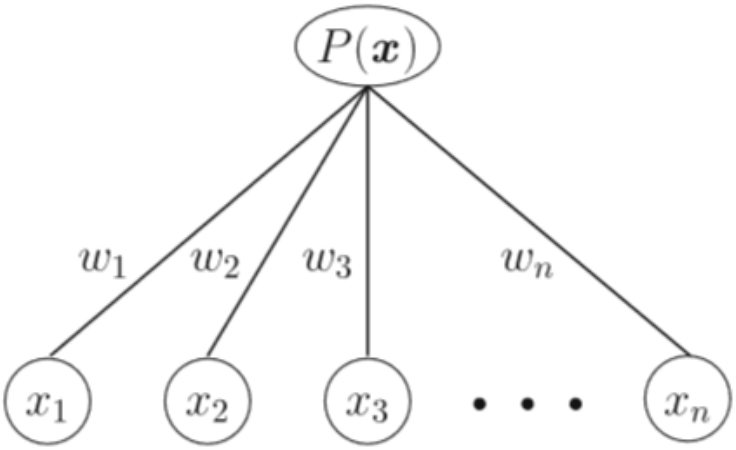
Pada Gambar 8.8 kita melihat bahwa fungsi AND dapat dipisahkan secara linier, tetapi fungsi XOR tidak. Untuk AND, misalnya, garis – x_1 + 3/2 memisahkan interpretasi benar dan salah dari rumus $x_1 \wedge x_2$. Sebaliknya, fungsi XOR tidak memiliki garis pemisah yang lurus. Jelas fungsi XOR memiliki struktur yang lebih kompleks daripada fungsi AND dalam hal ini. Dengan perceptron, kami menyajikan algoritma pembelajaran yang sangat sederhana yang dapat memisahkan set yang dapat dipisahkan secara linier.

Definisi 8.3
Misalkan $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$ adalah vektor bobot dan $x \in \mathbb{R}^n$ vektor input. Sebuah perceptron mewakili fungsi $P: \mathbb{R}^n \rightarrow \{0, 1\}$ yang sesuai dengan aturan berikut:

$$P(x) = \begin{cases} 1 & jika \; wx = \sum_{i=1}^{n} w_i x_i > 0 \\ 0 & Lainnya \end{cases}$$

Perceptron adalah algoritma klasifikasi yang sangat sederhana. Ini setara dengan jaringan saraf dua lapis dengan aktivasi oleh fungsi ambang batas, ditunjukkan pada Gambar 8.9. Seperti yang ditunjukkan pada Bab. 9, setiap node dalam jaringan mewakili neuron, dan semua tepi sinaps. Untuk saat ini, bagaimanapun, kita hanya akan melihat perceptron sebagai agen pembelajaran, yaitu, sebagai fungsi matematika yang memetakan vektor fitur ke nilai fungsi. Di sini variabel input xi adalah fitur yang dilambangkan. Seperti yang dapat kita lihat dalam rumus $\sum_{i=1}^{n} w_i \, x_i > 0$, semua titik x di atas hyperplane $\sum_{i=1}^{n} w_i \, x_i = 0$ diklasifikasikan sebagai positif (p(x) =1), dan semua lainnya sebagai negatif

Px = 0. Hyperplane pemisah melewati titik asal karena θ = 0. Kami akan menggunakan sedikit trik untuk menunjukkan bahwa tidak adanya ambang batas yang sewenang-wenang menunjukkan tidak ada pembatasan kekuasaan. Namun, pertama-tama, kami ingin memperkenalkan algoritma pembelajaran sederhana untuk perceptron.



Gambar 8.9 Representasi grafis dari perceptron sebagai jaringan saraf dua lapis

Aturan Belajar

Dengan notasi M+ dan M_ untuk himpunan pola latihan positif dan negatif, aturan pembelajaran perceptron berbunyi:

```
PERCEPTRONLEARNING[M+, M−]
w = arbitrary vector of real numbers
Repeat
    For all x ∈ M+
        If w x ≤ 0 Then w = w + x
    For all x ∈ M−
        If w x > 0 Then w = w − x
Until all x ∈ M+ ∪ M− are correctly classified
```

Perceptron harus menampilkan nilai 1 untuk semua x ∈ 2 M+. Dengan Definisi 8.3 adalah masalah ketika wx > 0. Jika ini bukan masalahnya, maka ditambahkan ke vektor bobot w, di mana vektor bobot diubah menjadi arah yang benar. Kita melihat ini ketika kita menerapkan perceptron ke vektor wþx yang diubah karena

$$(w+x) \cdot x = wx + x^2$$

Jika langkah ini cukup sering diulang, maka pada titik tertentu nilai wx akan menjadi positif, sebagaimana mestinya. Secara analog, kita melihat bahwa, untuk data pelatihan negatif, perceptron menghitung nilai yang semakin kecil

$$(w - x) \cdot x = wx - x^2$$

yang pada titik tertentu menjadi negatif.[39]

Contoh 8.2 Sebuah perceptron harus dilatih pada himpunan M+ {(0, 1.8), (2, 0.6)} dan M− {(−1.2, 1.4), (0.4, 1)}. w (1, 1) digunakan sebagai vektor bobot awal. Data pelatihan dan garis yang ditentukan oleh vektor bobot $wx = x_1 + x_2 = 0$ ditunjukkan pada Gambar 8.10

[39] Peringatan! Ini bukan bukti konvergensi untuk aturan pembelajaran perceptron. Ini hanya menunjukkan bahwa perceptron konvergen ketika dataset pelatihan terdiri dari satu contoh.

pada gambar pertama di baris atas. Selain itu, vektor bobot digambar sebagai garis putus-putus. Karena wx=0, ini ortogonal terhadap garis. Dalam iterasi pertama melalui loop dari algoritma pembelajaran, satu-satunya contoh pelatihan yang diklasifikasikan secara salah adalah (−1.2, 1.4) karena

$$(-1.2, 1.4).\begin{pmatrix}1\\1\end{pmatrix} = 0.2 > 0.$$

Hasil ini di (1,1)−(−1.2, 1.4) = (2.2, −0.4), seperti yang digambar pada gambar kedua di baris atas Gambar 8.10. Gambar lainnya menunjukkan bagaimana, setelah total lima perubahan, garis pemisah terletak di antara dua kelas. Persepsi dengan demikian mengklasifikasikan semua data dengan benar. Kita lihat dengan jelas pada contoh bahwa setiap titik data yang salah diklasifikasikan dari M+ "menarik" vektor bobot w ke arahnya dan setiap titik yang salah diklasifikasikan dari M− "mendorong" vektor bobot ke arah yang berlawanan. Telah ditunjukkan bahwa perceptron selalu konvergen untuk data yang dapat dipisahkan secara linier. Kita punya

Hasil ini di (1,1)−(−1.2,1.4) = (2.2,−0.4),seperti yang digambar pada gambar kedua di baris atas Gambar 8.10 . Gambar lainnya menunjukkan bagaimana, setelah total lima perubahan, garis pemisah terletak di antara dua kelas. Persepsi dengan demikian mengklasifikasikan semua data dengan benar. Kita lihat dengan jelas pada contoh bahwa setiap titik data yang salah diklasifikasikan dari M+ "menarik" vektor bobot w ke arahnya dan setiap titik yang salah diklasifikasikan dari M− "mendorong" vektor bobot ke arah yang berlawanan.

Telah ditunjukkan bahwa perceptron selalu konvergen untuk data yang dapat dipisahkan secara linier. Kita punya

Teorema 8.1

Misalkan kelas M+ dan M− dapat dipisahkan secara linear dengan hyperplane wx = 0. Kemudian PERCEPTRON*LEARNING* konvergen untuk setiap inisialisasi vektor w. Perceptron P dengan vektor bobot yang dihitung membagi kelas M+ dan M−, yaitu:

(x) = 1 ⇔ x ∈ M+

Dan

P(x) = 0 ⇔ x ∈ M_

Seperti yang dapat kita lihat dengan jelas pada Contoh 8.2, perceptron seperti yang didefinisikan di atas tidak dapat membagi himpunan yang dapat dipisahkan secara linier, melainkan hanya yang habis dibagi oleh garis melalui titik asal, atau dalam $\mathbb{R}$ oleh hyperplane melalui titik asal, karena suku konstanta θ hilang dari persamaan $\sum_{i=1}^{n} w_i\, x_i > 0$,.

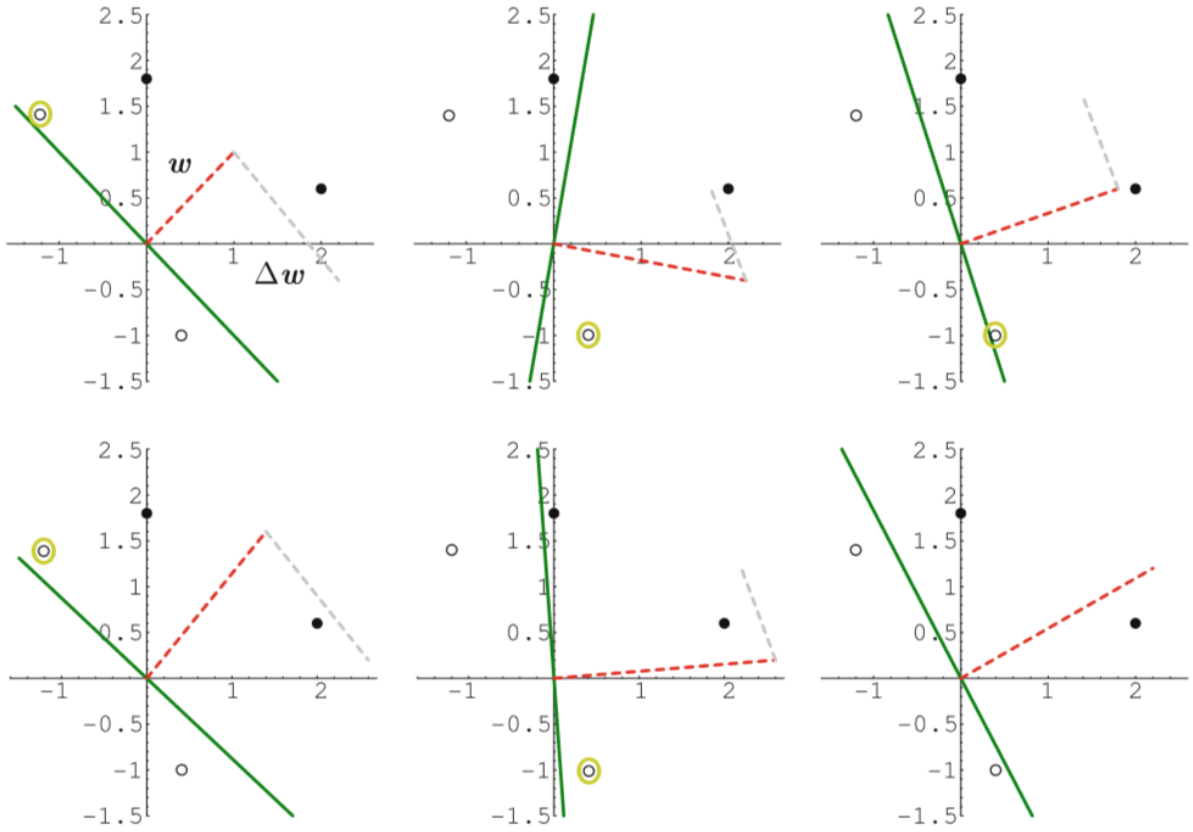
Dengan trik berikut kita dapat men ghasilkan suku konstan. Kami memegang komponen terakhir x_n dari vektor input x konstan dan mengaturnya ke nilai 1. Sekarang bobot w_n =: -θ bekerja seperti ambang batas karena

$$\sum_{i=1}^{n} w_i\, x_i = \sum_{i=1}^{n-1} w_i\, x_i - 0 > 0 \;\Leftrightarrow\; \sum_{i=1}^{n-1} w_i\, x_i > 0$$

Nilai konstan seperti x_n = 1 dalam input disebut unit bias. Karena bobot yang terkait menyebabkan pergeseran hyperplane yang konstan, istilah "bias" sangat cocok. Dalam penerapan algoritma pembelajaran perceptron, bit dengan nilai konstanta 1 ditambahkan ke vektor data pelatihan. Kami mengamati bahwa bobot wn, atau ambang batas h, dipelajari selama proses pembelajaran. Sekarang telah ditunjukkan bahwa perceptron Pθ: $\mathbb{R}^{n-1}$ → {0, 1}

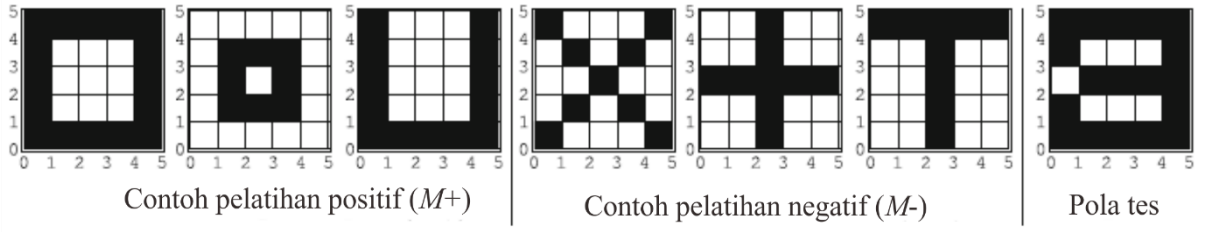
$$P_0(x_{1,}, \dots x_{n-1}) = \begin{cases} 1 & jika\ wx = \sum_{i=1}^{n} w_i x_i > 0 \\ 0 & Lainnya \end{cases}$$

dengan ambang arbitrer dapat disimulasikan oleh perceptron Pθ: $\mathbb{R}^{n-1} \rightarrow \{0, 1\}$ dengan threshold 0. Jika kita bandingkan (8.1) dengan definisi linear separable, maka kita melihat bahwa kedua pernyataan tersebut ekuivalen. Singkatnya, kami telah menunjukkan bahwa:



Gambar 8.10 Penerapan aturan pembelajaran perceptron pada dua titik data positif (•) dan dua negatif ( ).

Garis padat menunjukkan garis pemisah saat ini wx=0. Garis putus-putus ortogonal adalah vektor bobot w dan garis putus-putus kedua adalah vektor perubahan Δw = x atau Δw = -x yang akan ditambahkan, yang dihitung dari titik data aktif saat ini yang dikelilingi warna hijau muda

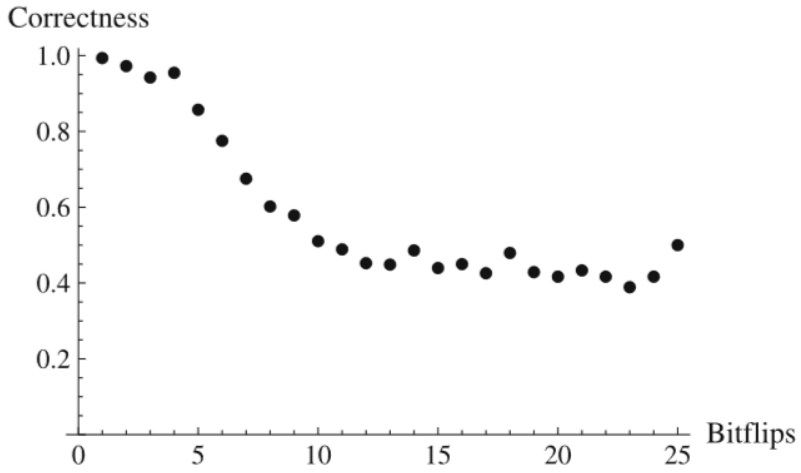


Gambar 8.11 Enam pola yang digunakan untuk pelatihan. Seluruh pola yang benar adalah salah satu dari 22 pola pengujian untuk pola pertama dengan urutan empat bit terbalik

Teorema 8.2
Suatu fungsi f: n!{0, 1} dapat diwakili oleh perceptron jika dan hanya jika dua himpunan vektor input positif dan negatif dapat dipisahkan secara linier.

Contoh 8.3
Kami sekarang melatih perceptron dengan ambang pada enam sederhana, pola biner grafis, diwakili pada Gambar. 8.11, dengan 5 x 5 piksel masing-masing.

Data pelatihan dapat dipelajari dengan *PERCEPTRON LEARNING* dalam empat iterasi pada semua pola. Pola dengan sejumlah variabel bit terbalik yang diperkenalkan sebagai noise digunakan untuk menguji kemampuan generalisasi sistem. Bit terbalik dalam pola pengujian dalam setiap kasus berurutan satu demi satu. Pada Gambar 8.12 persentase pola yang diklasifikasikan dengan benar diplot sebagai fungsi dari jumlah bit palsu. Setelah sekitar lima bit terbalik berturut-turut, ketepatan turun tajam, yang tidak mengejutkan mengingat kesederhanaan model. Di bagian selanjutnya kami akan menyajikan algoritma yang berkinerja jauh lebih baik dalam kasus ini.



Gambar 8.12 Kebenaran relatif perceptron sebagai fungsi dari jumlah bit terbalik dalam data uji

8.4 PENGOPTIMALAN DAN OUTLOOK

Sebagai salah satu algoritma pembelajaran berbasis jaringan saraf yang paling sederhana, perceptron dua lapis hanya dapat membagi kelas yang dapat dipisahkan secara linier. Selanjutnya kita akan melihat bahwa jaringan berlapis-lapis secara signifikan lebih kuat. Meskipun strukturnya sederhana, perceptron dalam bentuk yang disajikan menyatu dengan sangat lambat. Ini dapat dipercepat dengan normalisasi vektor pengubah berat. Rumus w w x diganti dengan w w x=jxj. Dengan demikian setiap titik data memiliki bobot yang sama selama pembelajaran, terlepas dari nilainya.

Kecepatan konvergensi sangat bergantung pada inisialisasi vektor w. Idealnya tidak perlu mengubah data di tanah, algoritma akan konvergen setelah satu iterasi. Kita bisa lebih dekat ke tujuan ini dengan menggunakan inisialisasi heuristik yang akan kita selidiki lebih dekat dalam Latihan 8.5. Jika kita membandingkan rumus perceptron dengan metode penilaian yang disajikan di Bagian. 7.3.1, kita segera melihat kesetaraannya. Selanjutnya, perceptron, sebagai model jaringan saraf paling sederhana, setara dengan naive Bayes, jenis jaringan Bayesian yang paling sederhana (lihat Latihan 8.17). Jadi ternyata beberapa algoritma klasifikasi yang sangat berbeda memiliki asal yang sama. Dalam Bab. 9 kita akan menjadi akrab dengan generalisasi perceptron dalam bentuk algoritma back-propagation,

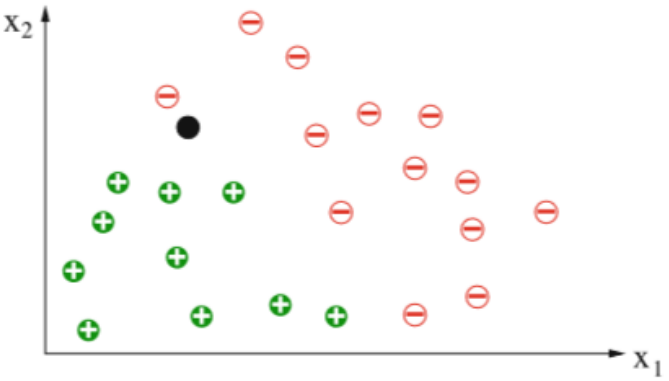
yang dapat membagi set yang tidak dapat dipisahkan secara linier melalui penggunaan beberapa lapisan, dan yang memiliki aturan pembelajaran yang lebih baik.

8.5 METODE TERTANGGA TERDEKAT

Untuk perceptron, pengetahuan yang tersedia dalam data pelatihan diekstraksi dan disimpan dalam bentuk terkompresi dalam bobot w_i. Dengan demikian informasi tentang data tersebut hilang. Ini adalah persis apa yang diinginkan, bagaimanapun, jika sistem seharusnya menggeneralisasi dari data pelatihan ke data baru. Generalisasi dalam hal ini adalah proses yang memakan waktu dengan tujuan menemukan representasi data yang kompak dalam bentuk fungsi yang mengklasifikasikan data baru sebaik mungkin.

Menghafal semua data hanya dengan menyimpannya sangat berbeda. Di sini pembelajarannya sangat sederhana. Namun, seperti yang disebutkan sebelumnya, pengetahuan yang disimpan tidak begitu mudah diterapkan pada contoh baru yang tidak diketahui. Pendekatan seperti itu sangat tidak cocok untuk mempelajari cara bermain ski, misalnya. Seorang pemula tidak akan pernah bisa menjadi pemain skateboard yang baik hanya dengan menonton video pemain skateboard yang baik. Terbukti, ketika gerakan belajar jenis ini dilakukan secara otomatis, hal serupa terjadi seperti dalam kasus perceptron. Setelah cukup lama berlatih, pengetahuan yang tersimpan dalam contoh-contoh pelatihan diubah menjadi representasi internal di otak.

Namun, ada contoh sukses menghafal di mana generalisasi juga mungkin. Selama mendiagnosis kasus yang sulit, seorang dokter dapat mencoba mengingat kasus serupa di masa lalu. Jika ingatannya baik, maka dia mungkin menemukan kasus ini, mencarinya di arsipnya dan akhirnya mendapatkan diagnosis serupa. Untuk pendekatan ini dokter harus memiliki perasaan yang baik untuk kesamaan, untuk mengingat kasus yang paling mirip. Jika dia telah menemukan ini, maka dia harus bertanya pada dirinya sendiri apakah itu cukup mirip untuk membenarkan diagnosis yang sama.



Gambar 8.13 Dalam contoh ini dengan contoh pelatihan negatif dan positif, metode tetangga terdekat mengelompokkan titik baru yang ditandai dengan warna hitam ke dalam kelas negatif

Apa arti kesamaan dalam konteks formal yang kita bangun? Kami mewakili sampel pelatihan seperti biasa dalam ruang fitur multidimensi dan mendefinisikan: Semakin kecil jarak mereka dalam ruang fitur, semakin banyak dua contoh yang serupa. Kami sekarang menerapkan definisi ini pada contoh dua dimensi sederhana dari Gambar 8.13. Di sini tetangga berikutnya ke titik hitam adalah contoh negatif. Jadi itu ditugaskan ke kelas negatif. Jarak d(x, y) antara dua titik $x \in \mathbb{R}^n$ dan $y \in \mathbb{R}^n$ misalnya dapat diukur dengan jarak Euclidean

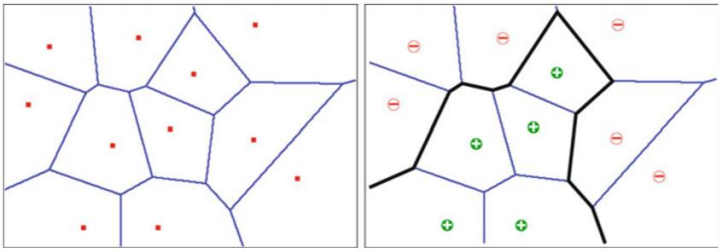
$$d(x,y) = |x - y| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Karena ada banyak metrik jarak lain selain yang ini, masuk akal untuk memikirkan alternatif untuk aplikasi konkret. Dalam banyak aplikasi, fitur tertentu lebih penting daripada yang lain. Oleh karena itu seringkali masuk akal untuk menskalakan fitur secara berbeda berdasarkan bobot w_i. Rumusnya kemudian berbunyi

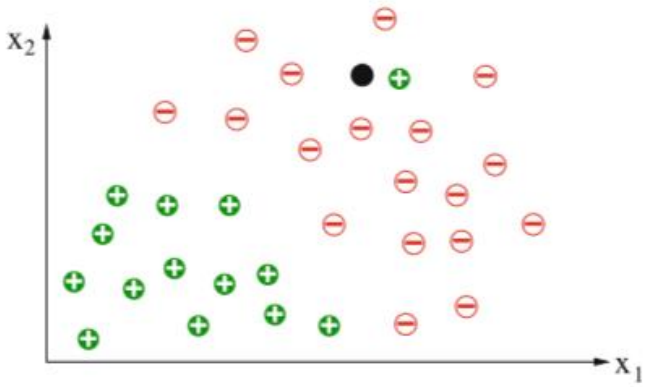
$$d_w(x,y) = |x - y| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Program klasifikasi tetangga terdekat sederhana berikut mencari data pelatihan untuk tetangga terdekat t ke contoh baru s dan kemudian mengklasifikasikan s persis seperti t.[40]

NEARESTNEIGHBOR[M_+, M_-, s]
$t = \text{argmin}_{x \in M_+ \cup M_-}\{d(s,x)\}$
If $t \in M_+$ **Then Return** („+")
Else Return(„–")



Gambar 8.14 Sekumpulan titik bersama dengan Diagram Voronoi (kiri) dan garis pemisah yang dihasilkan untuk dua kelas M+ dan M−



Gambar 8.15 Metode tetangga terdekat memberikan titik baru yang ditandai dengan warna hitam ke kelas yang salah (positif) karena tetangga terdekat kemungkinan besar diklasifikasikan salah

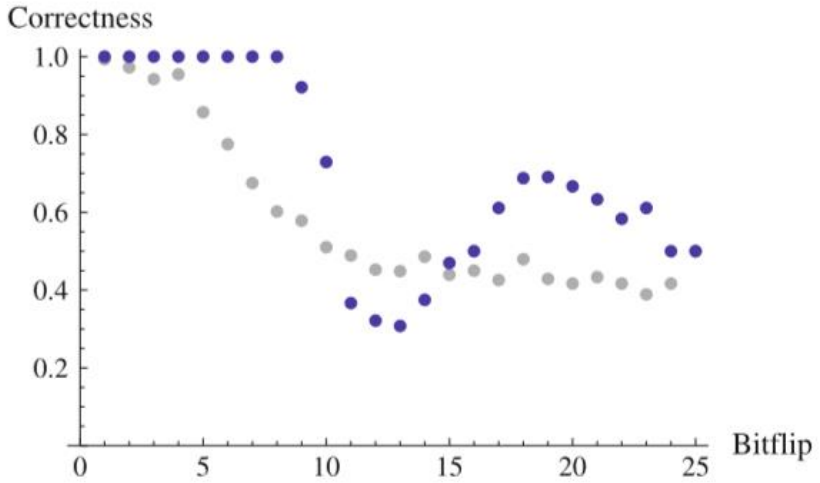
[40] Fungsi argmin dan argmax menentukan, sama seperti min dan max, minimum atau maksimum dari suatu himpunan atau fungsi. Namun, alih-alih mengembalikan nilai maksimum atau minimum, mereka memberikan posisi, yaitu argumen di mana ekstrem muncul.

Berbeda dengan perceptron, metode tetangga terdekat tidak menghasilkan garis yang membagi titik data latih. Namun, garis imajiner yang memisahkan kedua kelas pasti ada. Kita dapat menghasilkan ini dengan terlebih dahulu menghasilkan apa yang disebut diagram Voronoi. Dalam diagram Voronoi, setiap titik data dikelilingi oleh poligon cembung, yang dengan demikian mendefinisikan lingkungan di sekitarnya. Diagram Voronoi memiliki sifat bahwa untuk titik baru yang berubah-ubah, tetangga terdekat di antara titik data adalah titik data, yang terletak di lingkungan yang sama. Jika diagram Voronoi untuk satu set data pelatihan ditentukan, maka mudah untuk menemukan tetangga terdekat untuk titik baru yang akan diklasifikasikan. Keanggotaan kelas kemudian akan diambil dari tetangga terdekat.

Pada Gambar 8.14 kita melihat dengan jelas bahwa metode tetangga terdekat secara signifikan lebih kuat daripada perceptron. Ia mampu dengan benar mewujudkan garis pemisah kompleks yang sewenang-wenang (secara umum: hyperplanes). Namun, ada bahaya di sini. Satu titik yang salah dalam keadaan tertentu dapat menyebabkan hasil klasifikasi yang sangat buruk. Kasus seperti itu terjadi pada Gambar 8.15 selama klasifikasi titik hitam. Metode tetangga terdekat dapat mengklasifikasikan ini salah. Jika titik hitam tepat di sebelah titik positif yang merupakan outlier dari kelas positif, maka akan diklasifikasikan positif daripada negatif seperti yang dimaksudkan di sini. Pemasangan yang salah terhadap kesalahan acak (noise) disebut *overfitting*.

```
K-NEARESTNEIGHBOR(M+, M−, s)
    V = {k nearest neighbors in M+ ∪ M−}
    If      |M+ ∩ V| > |M− ∩ V| Then    Return(„+")
    ElseIf  |M+ ∩ V| < |M− ∩ V| Then    Return(„−")
    Else                                Return(Random(„+", „−"))
```

Gambar 8.16 ALGORITMA K-NEARESTNEIGHBOR



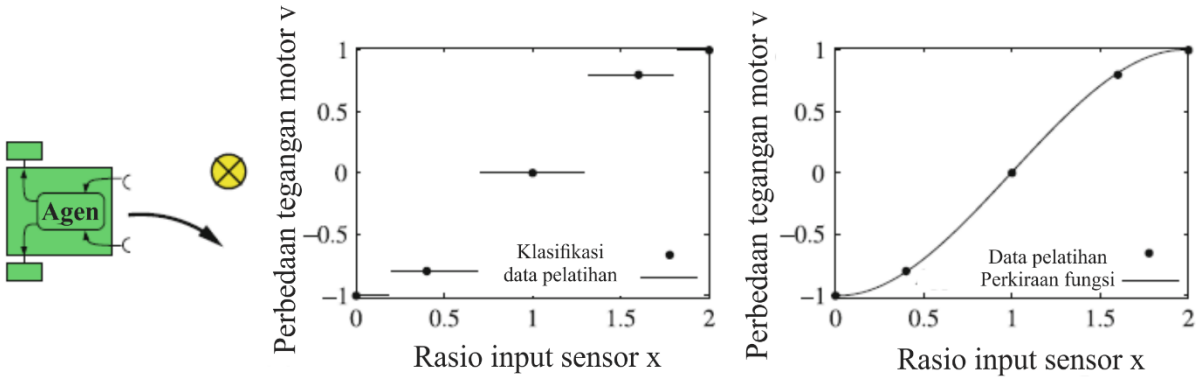
Gambar 8.17 Ketepatan relatif klasifikasi tetangga terdekat sebagai fungsi dari jumlah bit terbalik.

Struktur kurva dengan minimum pada 13 dan maksimum pada 19 terkait dengan struktur khusus dari data latih. Untuk perbandingan nilai perceptron dari Contoh 8.3 ditampilkan dalam warna abu-abu

Untuk mencegah klasifikasi yang salah karena outlier tunggal, direkomendasikan untuk menghaluskan permukaan pembagian. Ini dapat dicapai dengan, misalnya, dengan algoritma K-NEARESTNEIGHBOR pada Gambar 8.16, yang membuat keputusan mayoritas di antara k tetangga terdekat.

Contoh 8.4

Kita sekarang menerapkan NEARESTNEIGHBOR pada Contoh 8.3. Karena kita berurusan dengan data biner, kita menggunakan jarak Hamming sebagai metrik jarak.[41] Sebagai contoh pengujian, kita kembali menggunakan contoh pelatihan yang dimodifikasi dengan masing-masing n bit terbalik berturut-turut. Pada Gambar 8.17 persentase contoh uji yang diklasifikasikan dengan benar ditunjukkan sebagai fungsi dari jumlah bit terbalik b. Untuk hingga delapan bit terbalik, semua pola diidentifikasi dengan benar. Melewati titik itu, jumlah kesalahan meningkat dengan cepat. Hal ini tidak mengherankan karena pola latihan nomor 2 dari Gambar 8.11 dari kelas Mþ memiliki jarak hamming 9 ke dua contoh latihan, nomor 4 dan 5 dari kelas lainnya. Artinya pola tes sangat mungkin mendekati pola kelas lainnya. Cukup jelas kita melihat bahwa klasifikasi tetangga terdekat lebih unggul dari perceptron dalam aplikasi ini hingga delapan bit palsu.



Gambar 8.18 *Learning* Agent, yang seharusnya menghindari cahaya (kiri), direpresentasikan sebagai pengklasifikasi (tengah), dan sebagai aproksimasi (kanan)

Dua Kelas, Banyak Kelas, Perkiraan

Klasifikasi tetangga terdekat juga dapat diterapkan pada lebih dari dua kelas. Sama seperti kasus dua kelas, kelas vektor fitur yang akan diklasifikasikan secara sederhana ditetapkan sebagai kelas tetangga terdekat. Untuk metode knearestneighbor, kelas ditentukan sebagai kelas dengan anggota paling banyak di antara k tetangga terdekat. Jika jumlah kelas besar, maka biasanya tidak masuk akal lagi untuk menggunakan algoritma klasifikasi karena ukuran data pelatihan yang diperlukan bertambah dengan cepat seiring dengan jumlah kelas. Selanjutnya, dalam keadaan tertentu informasi penting hilang selama klasifikasi banyak kelas. Ini akan menjadi jelas dalam contoh berikut.

Contoh 8.5

Sebuah robot otonom dengan sensor sederhana mirip dengan kendaraan Braitenberg yang disajikan pada Gambar 1.1 seharusnya belajar menjauh dari cahaya. Ini berarti ia harus belajar seoptimal mungkin untuk memetakan nilai sensornya ke sinyal

[41] Jarak Hamming antara dua vektor bit adalah jumlah bit yang berbeda dari dua vektor.

kemudi yang mengontrol arah mengemudi. Robot ini dilengkapi dengan dua sensor cahaya sederhana di sisi depannya. Dari dua sinyal sensor (dengan s_l untuk sensor kiri dan s_r untuk sensor kanan), hubungan $x = s_r / s_l$ dihitung. Untuk mengendalikan motor listrik kedua roda dari nilai x ini, selisih $v = U_r - U_l$ dari dua tegangan U_r dan U_l masing-masing motor kiri dan kanan. Tugas agen pembelajaran sekarang adalah menghindari sinyal cahaya. Karena itu ia harus mempelajari pemetaan f yang menghitung nilai "benar" v= f(x).[42]

Untuk ini kami melakukan percobaan di mana, untuk beberapa nilai x yang terukur, kami menemukan nilai v seoptimal yang kami bisa. Nilai-nilai ini diplot sebagai titik data pada Gambar 8.18 dan akan berfungsi sebagai data pelatihan untuk agen pembelajaran. Selama klasifikasi tetangga terdekat, setiap titik dalam ruang fitur (yaitu, pada sumbu x) diklasifikasikan persis seperti tetangga terdekatnya di antara data pelatihan. Fungsi untuk kemudi motor kemudian menjadi fungsi langkah dengan lompatan besar (Gbr. 8.18 tengah). Jika kita menginginkan langkah yang lebih halus, maka kita harus menyediakan lebih banyak data pelatihan. Di sisi lain, kita dapat memperoleh fungsi kontinu jika kita mendekati fungsi mulus agar sesuai dengan lima titik (Gbr. 8.18). Mengharuskan fungsi f untuk terus menerus menghasilkan hasil yang sangat baik, bahkan tanpa titik data tambahan.

Untuk aproksimasi fungsi pada titik data terdapat banyak metode matematika, seperti interpolasi polinomial, interpolasi spline, atau metode kuadrat terkecil. Penerapan metode ini menjadi masalah di dimensi yang lebih tinggi. Kesulitan khusus dalam AI adalah bahwa metode aproksimasi bebas model diperlukan. Artinya, perkiraan yang baik dari data harus dibuat tanpa pengetahuan tentang sifat khusus dari data atau aplikasi. Hasil yang sangat baik telah dicapai di sini dengan jaringan saraf dan aproksimator fungsi nonlinier lainnya, yang disajikan dalam Bab. 9.

Metode k-nearestneighbor dapat diterapkan secara sederhana untuk masalah aproksimasi. Pada algoritma K-NEARESTNEIGHBOR, setelah ditentukan himpunan V ={ x_1, x_2, … , x_k}, nilai fungsi rata-rata k tetangga terdekat
Persamaan 8.2

$$f(x) = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$$

dihitung dan diambil sebagai aproksimasi f untuk vektor kueri x. Semakin besar k, semakin halus fungsi f.

8.6 JARAK ADALAH RELEVAN

Dalam aplikasi praktis varian diskrit maupun varian kontinu dari metode k tetangga terdekat, masalah sering terjadi. Ketika k menjadi besar, biasanya ada lebih banyak tetangga dengan jarak yang jauh daripada mereka yang memiliki jarak kecil. Dengan demikian perhitungan *f* didominasi oleh tetangga yang berjauhan. Untuk mencegah hal ini, k tetangga diberi bobot sedemikian rupa sehingga tetangga yang lebih jauh memiliki pengaruh yang lebih kecil pada hasil. Selama keputusan mayoritas dalam algoritma K-NEARESTNEIGHBOR, “suara” dibobot dengan bobot
Persamaan 8.3

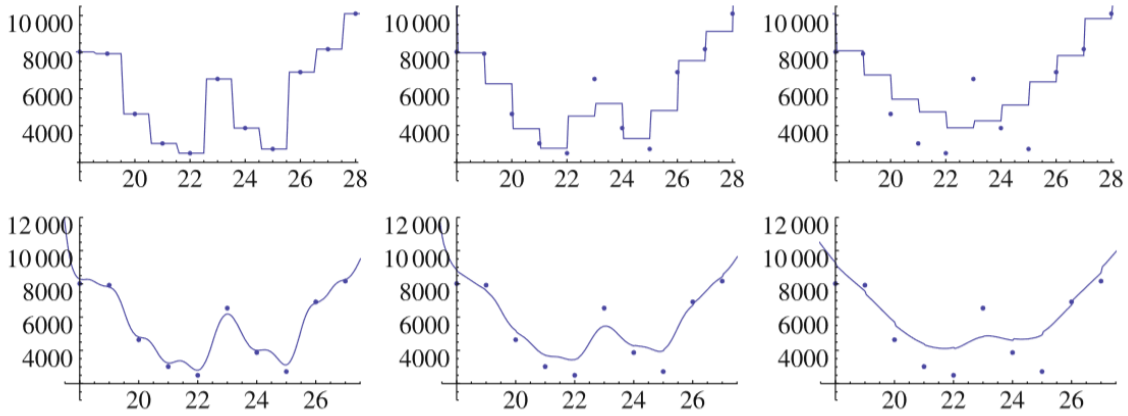
$$w_i = \frac{1}{1 + \alpha d(x, xi)^2}$$

yang berkurang dengan kuadrat jarak. Konstanta a menentukan kecepatan penurunan bobot. Persamaan (8.2) sekarang digantikan oleh

[42] Untuk menjaga agar contoh tetap sederhana dan mudah dibaca, vektor fitur x sengaja disimpan satu dimensi.

$$f(x) = \frac{\sum_{i=1}^{k} wi\, f(x)}{\sum_{i=1}^{k} wi}$$

Untuk konsentrasi titik yang terdistribusi secara merata dalam ruang fitur, hal ini memastikan bahwa pengaruh titik secara asimtotik mendekati nol seiring bertambahnya jarak. Dengan demikian menjadi mungkin untuk menggunakan banyak atau bahkan semua data pelatihan untuk mengklasifikasikan atau memperkirakan vektor input yang diberikan.



Gambar 8.19 Perbandingan metode k-nearest neighbor (baris atas) dengan k 1(kiri), k = 2 (tengah) dan k = 6(kanan), terhadap varian bobot jaraknya (baris bawah) dengan 20 (kiri), α = 4 (tengah) dan α = 1 (kanan) pada dataset satu dimensi

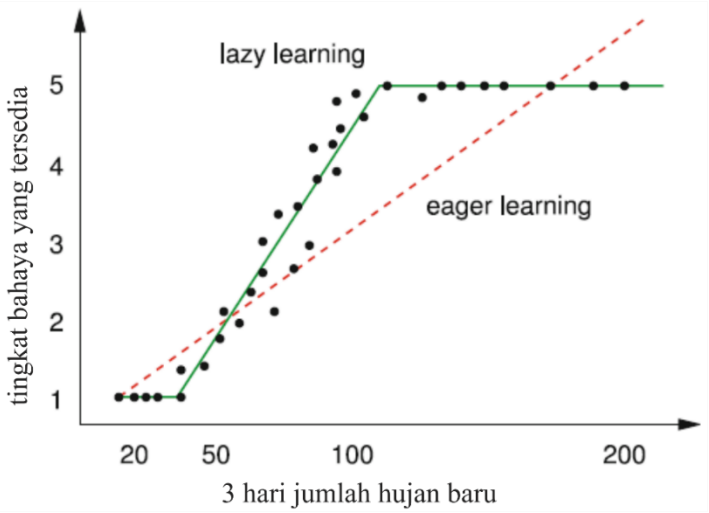
Untuk memahami metode ini, pada Gambar 8.19 metode k-nearest neighbor (di baris atas) dibandingkan dengan optimasi pembobotan jarak. Karena rata-rata, kedua metode dapat menggeneralisasi, atau dengan kata lain, membatalkan kebisingan, jika jumlah tetangga untuk k-tetangga terdekat atau parameter a diatur dengan tepat. Diagram menunjukkan dengan baik bahwa metode pembobotan jarak memberikan perkiraan yang jauh lebih halus daripada k-nearest tetangga. Sehubungan dengan kualitas aproksimasi, metode yang sangat sederhana ini dapat bersaing dengan baik dengan algoritma aproksimasi yang canggih seperti jaringan saraf nonlinier, mesin vektor pendukung, dan proses Gaussian.

Ada banyak alternatif untuk fungsi bobot (juga disebut kernel) yang diberikan pada (8.3). Misalnya, fungsi Gaussian atau fungsi berbentuk lonceng yang serupa dapat digunakan. Untuk sebagian besar aplikasi, hasilnya tidak terlalu masuk akal dalam pemilihan kernel. Namun, parameter lebar a yang untuk semua fungsi ini harus diatur secara manual memiliki pengaruh besar pada hasil, seperti yang ditunjukkan pada Gambar 8.19. Untuk menghindari adaptasi manual yang tidak nyaman, metode pengoptimalan telah dikembangkan untuk menyetel parameter ini secara otomatis.

Waktu Komputasi

Seperti disebutkan sebelumnya, pelatihan dilakukan di semua varian metode tetangga terdekat hanya dengan menyimpan semua vektor pelatihan bersama dengan labelnya (nilai kelas), atau nilai fungsi f(x). Dengan demikian tidak ada algoritma pembelajaran lain yang belajar secepat itu. Namun, menjawab kueri untuk klasifikasi atau aproksimasi vektor x bisa menjadi sangat mahal. Hanya menemukan k tetangga terdekat untuk n data pelatihan membutuhkan biaya yang tumbuh secara linier dengan n. Untuk klasifikasi atau aproksimasi, terdapat tambahan biaya yang linier pada k. Dengan demikian,

total waktu komputasi bertambah sebagai H(n + k). Untuk sejumlah besar data pelatihan, ini dapat menyebabkan masalah.



Gambar 8.20 Untuk menentukan bahaya longsor, sebuah fungsi didekati dari data pelatihan. Di sini untuk perbandingan adalah model lokal (garis padat), dan model global (garis putus-putus)

Ringkasan dan Pandangan

Karena tidak ada yang terjadi dalam fase pembelajaran dari metode tetangga terdekat yang disajikan, algoritme seperti itu juga dilambangkan sebagai pembelajaran malas, berbeda dengan pembelajaran bersemangat, di mana fase pembelajaran bisa mahal, tetapi penerapan pada contoh-contoh baru sangat efisien. Perceptron dan semua jaringan saraf lainnya, pembelajaran pohon keputusan, dan pembelajaran jaringan Bayesian adalah metode pembelajaran yang bersemangat. Karena metode pembelajaran malas memerlukan akses ke memori dengan semua data pelatihan untuk mendekati input baru, metode ini juga disebut pembelajaran berbasis memori.

Untuk membandingkan dua kelas proses pembelajaran ini, kita akan menggunakan sebagai contoh tugas menentukan bahaya longsoran saat ini dari rendah/tingginya jumlah curah hujan yang baru turun di wilayah tertentu di Swiss.[43] Pada Gambar 8.20 nilai yang ditentukan oleh para ahli dimasukkan, yang ingin kita gunakan sebagai data pelatihan. Selama penerapan algoritma pembelajaran bersemangat yang melakukan pendekatan linier data, garis putus-putus yang ditunjukkan pada gambar dihitung. Karena pembatasan pada garis lurus, kesalahannya relatif besar dengan maksimum sekitar 1,5 tingkat bahaya. Selama pembelajaran malas, tidak ada yang dihitung sebelum kueri untuk tingkat bahaya saat ini tiba. Kemudian jawabannya dihitung dari beberapa tetangga terdekat, yaitu lokal. Ini bisa menghasilkan kurva yang ditunjukkan pada gambar, yang disatukan dari segmen garis dan menunjukkan kesalahan yang jauh lebih kecil. Keuntungan dari metode lazy adalah lokalitasnya. Perkiraan diambil secara lokal dari tingkat hujan baru saat ini dan tidak secara global. Jadi untuk kelas fungsi yang pada dasarnya sama (misalnya fungsi linier), algoritma malas lebih baik.

Metode tetangga terdekat sangat cocok untuk semua situasi masalah di mana pendekatan lokal yang baik diperlukan, tetapi tidak menempatkan persyaratan tinggi pada

[43] Total tiga hari hujan sebenarnya merupakan fitur penting untuk menentukan tingkat bahaya. Namun, dalam praktiknya, atribut tambahan digunakan [Bra01]. Contoh yang digunakan di sini disederhanakan.

kecepatan sistem. Prediktor genangan air yangmenutupi lubang pada jalan yang disebutkan di sini, yang berjalan sekali sehari, adalah aplikasi semacam itu. Metode tetangga terdekat tidak cocok ketika deskripsi pengetahuan yang diekstraksi dari data harus dapat dimengerti oleh manusia, yang saat ini berlaku untuk banyak aplikasi *Data mining*. Dalam beberapa tahun terakhir metode pembelajaran berbasis memori ini menjadi populer, dan berbagai varian yang ditingkatkan (misalnya regresi linier berbobot lokal) telah diterapkan. Untuk dapat menggunakan metode yang dijelaskan, data pelatihan harus tersedia dalam bentuk vektor bilangan bulat atau bilangan real. Dengan demikian mereka tidak cocok untuk aplikasi di mana data direpresentasikan secara simbolis, misalnya sebagai rumus logika orde pertama. Sekarang kita akan membahas ini secara singkat.

Feature	Query	Case from case base
Defective part:	Rear light	Front light
Bicycle model:	Marin Pine Mountain	VSF T400
Year:	1993	2001
Power source:	Battery	Dynamo
Bulb condition:	ok	ok
Light cable condition:	?	ok
	Solution	
Diagnosis:	?	Front electrical contact missing
Repair:	?	Establish front electrical contact

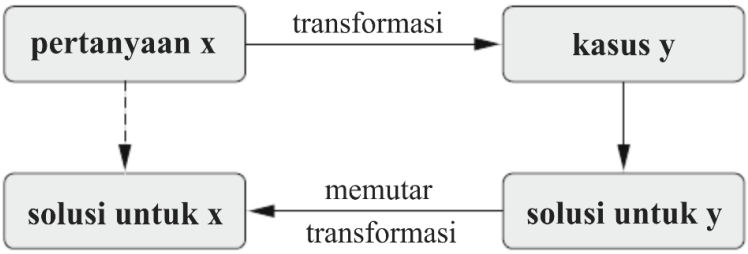
Gambar 8.21 Contoh diagnosis sederhana untuk kueri dan kasus yang sesuai dari basis kasus

8.7 PENALARAN BERBASIS KASUS

Dalam penalaran berbasis kasus (CBR), metode tetangga terdekat diperluas ke deskripsi masalah simbolis dan solusinya. CBR digunakan dalam diagnosis masalah teknis dalam layanan pelanggan atau untuk hotline telepon. Contoh yang ditunjukkan pada Gambar 8.21 tentang diagnosis lampu sepeda padam menggambarkan situasi seperti ini.

Sebuah solusi dicari untuk permintaan pelanggan dengan lampu sepeda belakang yang rusak. Di kolom kanan, diberikan kasus yang mirip dengan kueri di kolom tengah. Ini berasal dari basis kasus, yang sesuai dengan data pelatihan dalam metode tetangga terdekat. Jika kita hanya mengambil kasus yang paling mirip, seperti yang kita lakukan dalam metode tetangga terdekat, maka kita akhirnya akan mencoba memperbaiki lampu depan ketika lampu belakang rusak. Oleh karena itu, kami membutuhkan transformasi terbalik dari solusi untuk masalah serupa yang ditemukan kembali ke kueri. Langkah terpenting dalam penyelesaian kasus CBR dilakukan pada Gambar 8.22. Transformasi dalam contoh ini sederhana: lampu belakang dipetakan ke lampu depan contoh ini sederhana: lampu belakang dipetakan ke lampu depan. Seindah dan sesederhana metode ini dalam teori, dalam praktiknya konstruksi sistem diagnostik CBR adalah tugas yang sangat sulit. Tiga kesulitan utama adalah:

Pemodelan : Domain aplikasi aplikasi harus dimodelkan dalam konteks formal. Di sini logika monoton, yang kita ketahui dari Bab. 4, menyajikan kesulitan. Dapatkah pengembang memprediksi dan memetakan semua kemungkinan kasus khusus dan varian masalah?



Gambar 8.22 Jika untuk kasus x ditemukan kasus serupa y, maka untuk mendapatkan solusi x, transformasi harus ditentukan dan inversnya diterapkan pada kasus y yang ditemukan

Kesamaan : Menemukan metrik kesamaan yang sesuai untuk fitur simbolis, non-numerik. *Transformasi :* Sekalipun kasus serupa ditemukan, belum jelas bagaimana pemetaan transformasi dan inversnya.

Memang ada sistem CBR praktis untuk aplikasi diagnostik yang digunakan saat ini. Namun, karena alasan yang disebutkan, ini masih jauh di belakang ahli manusia dalam kinerja dan fleksibilitas. Alternatif yang menarik untuk CBR adalah jaringan Bayesian yang disajikan di Bab. 7. Seringkali representasi masalah simbolik juga dapat dipetakan dengan cukup baik ke fitur numerik diskrit atau kontinu. Kemudian metode pembelajaran induktif tersebut seperti pohon keputusan atau jaringan saraf dapat digunakan dengan sukses.

8.8 PEMBELAJARAN POHON KEPUTUSAN

Pembelajaran pohon keputusan adalah algoritma yang sangat penting untuk AI karena sangat kuat, tetapi juga sederhana dan efisien untuk mengekstraksi pengetahuan dari data. Dibandingkan dengan dua algoritma pembelajaran yang sudah diperkenalkan, ia memiliki keunggulan penting. Pengetahuan yang diekstraksi tidak hanya tersedia dan dapat digunakan sebagai fungsi kotak hitam, melainkan dapat dengan mudah dipahami, ditafsirkan, dan dikendalikan oleh manusia dalam bentuk pohon keputusan yang dapat dibaca. Ini juga membuat pembelajaran pohon keputusan menjadi alat penting untuk *Data mining*.

Kami akan membahas fungsi dan aplikasi pembelajaran pohon keputusan menggunakan algoritma C4.5. C4.5 diperkenalkan pada tahun 1993 oleh Ross Quinlan dari Australia dan merupakan peningkatan dari ID3 pendahulunya (Iterative Dichotomiser 3, 1986). Ini tersedia secara bebas untuk penggunaan nonkomersial. Pengembangan lebih lanjut, yang bekerja bahkan lebih efisien dan dapat memperhitungkan biaya keputusan, adalah C5.0. Sistem CART (Classification and Regression Trees, 1984) yang dikembangkan oleh Leo Breiman bekerja sama dengan C4.5. Ini memiliki antarmuka pengguna grafis yang nyaman, tetapi sangat mahal.

Dua puluh tahun sebelumnya, pada tahun 1964, sistem CHAID (Chi-square Automatic Interaction Detectors), yang dapat secara otomatis menghasilkan pohon keputusan, diperkenalkan oleh J. Sonquist dan J. Morgan. Ia memiliki karakteristik yang patut diperhatikan bahwa ia menghentikan pertumbuhan pohon sebelum menjadi terlalu besar, tetapi hari ini ia tidak memiliki relevansi lagi.

Tabel 8.4 Variabel untuk masalah klasifikasi ski

Variabel	***Nilai***	***Deskripsi***
Ngopi (variabel goal)	Ya, tidak	Haruskah aku pergi ke kafe ketika hujan sudah cukup reda untuk ngopi?

Matahari (fitur)	Ya, tidak	Apakah ada sinar matahari (cerah) hari ini?
Hujan_turun (fitur)	≤100, >100	Jarak antara rumah dengan kafe (dibawah 10km)
Akhir pekan (fitur)	Ya, tidak	Apakah ini akhir pekan?

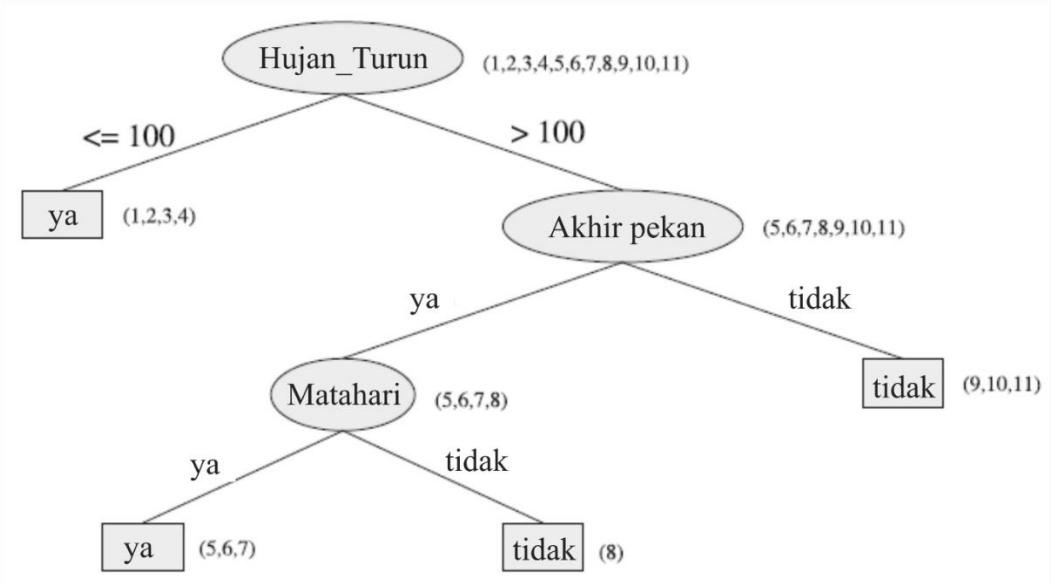
Yang juga menarik adalah alat *Data mining* KNIME (Konstanz Information Miner), yang memiliki antarmuka pengguna yang ramah dan, dengan menggunakan pustaka WEKA Java, juga memungkinkan induksi pohon keputusan. Kami akan memperkenalkan KNIME. Sekarang pertama-tama kita tunjukkan dalam contoh sederhana bagaimana pohon keputusan dapat dibangun dari data pelatihan, untuk kemudian menganalisis algoritme dan menerapkannya pada contoh LEXMED yang lebih kompleks untuk diagnosis medis.

Contoh Sederhana

Seorang pemain pendaki setia yang tinggal di daerah bukit tinggi, pegunungan yang indah di Indonesia, menginginkan pohon keputusan untuk membantunya memutuskan apakah perlu mengendarai mobilnya di pegunungan. Dengan demikian, kami memiliki masalah dua kelas pendaki ya/tidak berdasarkan variabel yang tercantum dalam Tabel 8.4. Gambar 8.23 menunjukkan pohon keputusan untuk masalah ini. Pohon keputusan adalah pohon yang simpul-simpul dalamnya merepresentasikan fitur (atribut). Setiap tepi mewakili nilai atribut. Pada setiap simpul daun nilai kelas diberikan. Data yang digunakan untuk konstruksi pohon keputusan ditunjukkan pada Tabel 8.5. Setiap baris dalam tabel berisi data untuk satu hari dan dengan demikian mewakili sampel. Setelah pemeriksaan lebih dekat kita melihat bahwa baris 6 dan baris 7 saling bertentangan. Jadi tidak ada algoritma klasifikasi deterministik yang dapat mengklasifikasikan semua data dengan benar. Oleh karena itu, jumlah data yang diklasifikasikan salah harus ≥1. Pohon pada Gambar 8.23 dengan demikian mengklasifikasikan data secara optimal.

Bagaimana pohon seperti itu dibuat dari data? Untuk menjawab pertanyaan ini, pertama-tama kita akan membatasi diri kita pada atribut-atribut diskrit dengan banyak nilai berhingga. Karena jumlah atribut juga terbatas dan setiap atribut dapat muncul paling banyak satu kali per jalur, maka terdapat banyak pohon keputusan yang berbeda. Sebuah algoritma yang sederhana dan jelas untuk konstruksi sebuah pohon hanya akan menghasilkan semua pohon, kemudian untuk setiap pohon menghitung jumlah klasifikasi data yang salah, dan pada akhirnya memilih pohon dengan jumlah kesalahan minimum. Jadi kita bahkan akan memiliki algoritma yang optimal (dalam arti kesalahan untuk data pelatihan) untuk pembelajaran pohon keputusan.

Kerugian yang jelas dari algoritma ini adalah waktu komputasi yang sangat tinggi, segera setelah jumlah atribut menjadi agak lebih besar. Kami sekarang akan mengembangkan algoritma heuristik yang, mulai dari root, secara rekursif membangun pohon keputusan. Pertama, atribut dengan perolehan informasi tertinggi (Hujan_turun) dipilih untuk simpul akar dari himpunan semua atribut. Untuk setiap nilai atribut (≤100, >100) ada cabang di pohon. Sekarang untuk setiap cabang proses ini diulang secara rekursif. Selama pembuatan node, atribut dengan perolehan informasi tertinggi di antara atribut yang belum digunakan selalu dipilih, dengan semangat strategi serakah.



Gambar 8.23 Pohon keputusan untuk masalah klasifikasi ski.

Dalam daftar di sebelah kanan node, nomor data pelatihan yang sesuai diberikan. Perhatikan bahwa dari simpul daun cerah = ya hanya dua dari tiga contoh yang diklasifikasikan dengan benar

Tabel 8.5 Kumpulan data untuk masalah klasifikasi ski

Hari	*Hujan_Turun*	*Akhir Pekan*	*Matahari*	*Ngopi*
1	≤100	ya	ya	ya
2	≤100	ya	ya	ya
3	≤100	ya	tidak	ya
4	≤100	tidak	ya	ya
5	>100	ya	ya	ya
6	>100	ya	ya	ya
7	>100	ya	ya	tidak
8	>100	ya	tidak	tidak
9	>100	tidak	ya	tidak
10	>100	tidak	ya	tidak
11	>100	tidak	tidak	tidak

8.9 ENTROPI SEBAGAI METRIK UNTUK KONTEN INFORMASI

Algoritma *top-down* yang dijelaskan untuk konstruksi pohon keputusan, pada setiap langkah memilih atribut dengan perolehan informasi tertinggi. Kami sekarang memperkenalkan entropi sebagai metrik untuk konten informasi dari satu set data pelatihan D. Jika kita hanya melihat variabel biner pendaki pada contoh di atas, maka D dapat digambarkan sebagai

D = (ya, ya, ya, ya, ya, ya, tidak, tidak, tidak, tidak, tidak)

dengan probabilitas yang diperkirakan

$P_1 = P(ya) = 6/11$ dan $P_2 = P(tidak) = 5/11$

Di sini kita ternyata memiliki distribusi probabilitas p (6/11, 5/11). Secara umum, untuk masalah kelas n ini berbunyi

$$P = (p_1, ..., p_n)$$

Dengan

$$\sum_{i=1}^{n} pi = 1$$

Untuk memperkenalkan isi informasi dari suatu distribusi, kami mengamati dua kasus ekstrim. pertama mari

Persamaan 8.4

$$P=(1, 0, 0, ..., 0)$$

Artinya, yang pertama dari n peristiwa pasti akan terjadi dan yang lainnya tidak. Ketidakpastian tentang hasil dari peristiwa demikian minimal. Sebaliknya, untuk distribusi seragam

Persamaa 8.5

$$p = \left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$$

ketidakpastiannya maksimal karena tidak ada peristiwa yang dapat dibedakan dari yang lain. Di sini Claude Shannon bertanya pada dirinya sendiri berapa banyak bit yang diperlukan untuk mengkodekan peristiwa semacam itu. Dalam kasus tertentu (8.4) nol bit diperlukan karena kita tahu bahwa kasus 1 selalu terjadi. Dalam kasus yang terdistribusi seragam dari (8.5) ada kemungkinan yang sama. Untuk pengkodean biner, bit $\log_2 n$ diperlukan di sini. Karena semua probabilitas individu adalah pi = 1/n, bit $\log_2 n$ 1/pi diperlukan untuk pengkodean ini. Dalam kasus umum p = (p_1, …, p_n), jika probabilitas kejadian elementer menyimpang dari distribusi seragam, maka nilai harapan H untuk jumlah bit dihitung. Untuk tujuan ini kita akan menimbang semua nilai $\log_2 1/pi = - \log_2 pi$ dengan probabilitasnya dan memperoleh

$$H = \sum_{i=1}^{n} p_i(-log_2 pi) = -\sum_{i=}^{n} pi\, log_2 pi$$

Semakin banyak bit yang kita butuhkan untuk mengkodekan suatu peristiwa, jelas semakin tinggi ketidakpastian tentang hasilnya. Oleh karena itu kami mendefinisikan:

Definisi 8.4

Entropi H sebagai metrik untuk ketidakpastian distribusi probabilitas didefinisikan oleh[44]

$$H(P) = H(H(p_1, \dots, p_n) := -\sum_{i=1}^{n} Pi\, log_2 p_i$$

Derivasi rinci dari rumus ini ditemukan di [SW76]. Jika kita mengganti kejadian tertentu p=(1, 0, 0, … , 0), maka 0log20, hasil ekspresi yang tidak ditentukan. Kami memecahkan masalah ini dengan definisi $0\log_2 0 := 0$ (lihat Latihan 8.10. Sekarang kita dapat menghitung H(1, 0, … , 0) = 0. Kita akan menunjukkan bahwa entropi dalam hiperkubus $[0, 1]^n$ di bawah batasan $\sum_{i=1}^{n} Pi = 1$ mengambil nilai maksimumnya dengan distribusi seragam 1/n ..., 1/n. Dalam kasus sebuah peristiwa dengan dua kemungkinan hasil, yang sesuai dengan dua kelas, hasilnya adalah

$$H(p) = H(p_1, p_2) = H(p1, 1\text{-}p_1) = -(p_1 \log_2 P_1 + (1 – p_1) \log_2(1\text{-}P_1))$$

[44] logaritma natural daripada log2 digunakan dalam definisi entropi. Karena di sini, dan juga dalam kasus metode MaxEnt, entropi hanya dibandingkan, perbedaan ini tidak berperan. (lihat Latihan 8.12)

Ekspresi ini ditunjukkan sebagai fungsi p1 pada Gambar 8.24 dengan maksimum pada $p_1 = 1/2$. Karena setiap dataset diklasifikasikan D diberikan distribusi probabilitas p dengan memperkirakan probabilitas kelas, kita dapat memperluas konsep entropi ke data dengan definisi

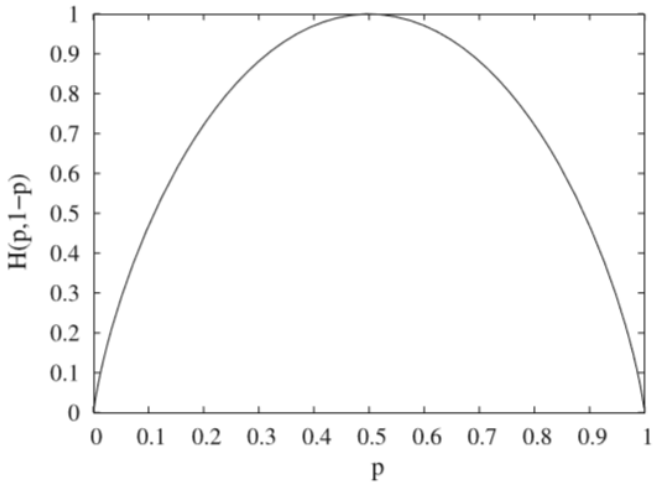
$$H(D) = H(p)$$

Sekarang, karena konten informasi I(D) dari dataset D dimaksudkan sebagai kebalikan dari ketidakpastian. Jadi kita mendefinisikan:

Definisi 8.5
Isi informasi dari dataset didefinisikan sebagai:
Persamaan 8.6

$$I(D) := 1 - H(D)$$

Perolehan Informasi



Gambar 8.24 Fungsi entropi untuk kasus dua kelas. Kita melihat maksimum pada p 1/2 dan simetri terhadap pertukaran p dan 1 p

Jika kita menerapkan rumus entropi pada contoh, hasilnya adalah

$$H(6/11,5/11) = 0.994$$

Selama konstruksi pohon keputusan, dataset dibagi lagi oleh setiap atribut baru. Semakin banyak atribut meningkatkan konten informasi dari distribusi dengan membagi data, semakin baik atribut tersebut. Kami mendefinisikan sesuai:

Definisi 8.6
Perolehan informasi G(D, A) melalui penggunaan atribut A ditentukan oleh selisih rata-rata isi informasi dari kumpulan data $D = D_1[D_2 \cup \cdots \cup D_n$ dibagi dengan nilai-n atribut A dan konten informasi I(D) dari dataset tak terbagi, yang menghasilkan

$$G(D, A) = \sum_{i=1}^{n} \frac{|Di|}{|D|} \, 1(Di) - I(D)$$

Dengan (8.6) kita peroleh dari ini

$$G(D < A) = \sum_{i=1}^{n} \frac{|Di|}{|D|} \, 1(Di) - I(D) = \sum_{i=1}^{n} \frac{|Di|}{|D|} \, 1H(Di)) - (IH()D))$$

$= 1 - \sum_{i=1}^{n} \frac{|Di|}{|D|} H(Di) - I + H(D)$ =H(D) - $\sum_{i=1}^{n} \frac{|Di|}{|D|} H(Di)$

Diterapkan pada contoh kami untuk atribut Snow_Dist, ini menghasilkan

G(D, weekend) = 0.150

Secara analog kita peroleh

G(D, Senin) = 0.049

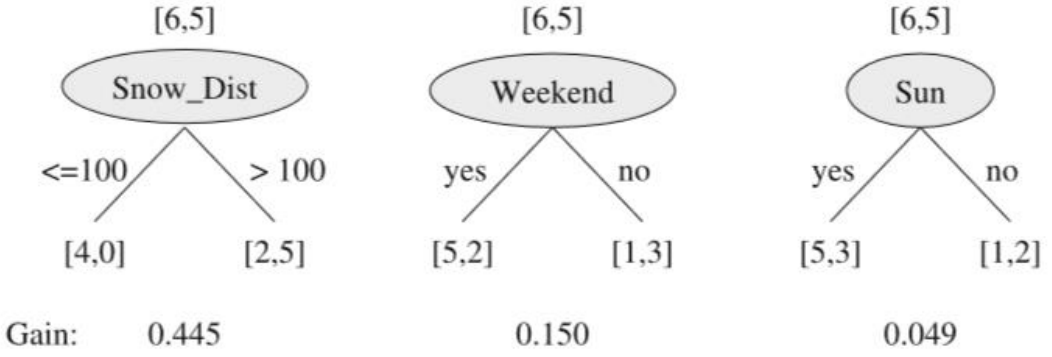
Atribut Snow_Dist sekarang menjadi simpul akar dari pohon keputusan. Situasi pemilihan atribut ini sekali lagi diklarifikasi pada Gambar 8.25. Dua nilai atribut 100 dan >100 menghasilkan dua sisi di pohon, yang sesuai dengan himpunan bagian D 100 dan $D_{>100}$. Untuk subset $D_{\leq 100}$ klasifikasinya jelas ya. dengan demikian pohon berakhir di sini. Di cabang lain $D_{>100}$ tidak ada hasil yang jelas. Dengan demikian algoritma berulang secara rekursif. Dari dua atribut yang masih tersedia, Sun dan Weekend, harus dipilih yang lebih baik. Kami menghitung

$G(D_{>100}, Weekend) = 0.292$

Dan

$G(D_{>100}, Sun) = 0.170$

Node dengan demikian mendapatkan atribut Weekend yang ditetapkan. Untuk Akhir Pekan = tidak pohon berakhir dengan keputusan pendaki = tidak. Perhitungan keuntungan di sini mengembalikan nilai 0. Untuk Akhir Pekan = ya, Sun menghasilkan keuntungan 0,171. Kemudian konstruksi pohon dihentikan karena tidak ada atribut lebih lanjut yang tersedia, meskipun contoh nomor 7 salah diklasifikasikan. Pohon yang sudah jadi sudah familiar dari Gambar 8.23.



Gambar 8.25 Perolehan yang dihitung untuk berbagai atribut mencerminkan apakah pembagian data oleh masing-masing atribut menghasilkan pembagian kelas yang lebih baik.

Semakin banyak distribusi yang dihasilkan oleh atribut menyimpang dari distribusi seragam, semakin tinggi perolehan informasi

8.10 PENERAPAN C4.5

Pohon keputusan yang baru saja kita buat juga dapat dihasilkan oleh C4.5. Data pelatihan disimpan dalam file data ski.data dalam format berikut:

```
<=100, yes,  yes,  yes
<=100, yes,  yes,  yes
<=100, yes,  no,   yes
<=100, no,   yes,  yes
>100,  yes,  yes,  yes
>100,  yes,  yes,  yes
>100,  yes,  yes,  no
>100,  yes,  no,   no
>100,  no,   yes,  no
>100,  no,   yes,  no
>100,  no,   no,   no
```

Informasi tentang atribut dan kelas disimpan dalam file ski.names (baris yang diawali dengan "|" adalah komentar):

```
|Classes: no: do not ski, yes: go skiing
|
no,yes.
|
|Attributes
|
Snow_Dist:        <=100,>100.
Weekend:          no,yes.
Sun:              no,yes.
```

C4.5 kemudian dipanggil dari baris perintah Unix dan menghasilkan pohon keputusan yang ditunjukkan di bawah ini, yang diformat menggunakan lekukan. Opsi -f adalah untuk nama file input, dan opsi -m menentukan jumlah minimum titik data pelatihan yang diperlukan untuk menghasilkan cabang baru di pohon. Karena jumlah titik data pelatihan dalam contoh ini sangat kecil, -m 1 masuk akal di sini. Untuk kumpulan data yang lebih besar, nilai setidaknya -m 10 harus digunakan.

```
unixprompt> c4.5 -f ski -m 1

C4.5 [release 8] decision tree generator

                                    Wed Aug 23 10:44:49 2010
----------------------------------------
    Options:
        File stem <ski>
        Sensible test requires 2 branches with >=1 cases

Read 11 cases (3 attributes) from ski.data

Decision Tree:

Snow_Dist = <=100: ja (4.0)
Snow_Dist = >100:
|   Weekend = no: no (3.0)
|   Weekend = yes:
|   | Sun = no: no (1.0)
|   | Sun = yes: yes (3.0/1.0)

Simplified Decision Tree:

Snow_Dist = <=100: yes (4.0/1.2)
Snow_Dist = >100: no (7.0/3.4)

Evaluation on training data (11 items):

       Before Pruning             After Pruning
       ----------------    ---------------------------
       Size    Errors      Size    Errors    Estimate

        7     1(9.1%)       3      2(18.2%)   (41.7%) <<
```

GENERATEDECISIONTREE(*Data,Node*)

A_{max} = Attribute with maximum information gain
If $G(A_{max}) = 0$
  Then *Node* becomes leaf node with most frequent class in *Data*
  Else assign the attribute A_{max} to *Node*
    For each value $a_1, \ldots, a_n$ of A_{max}, generate
      a successor node: $K_1, \ldots, K_n$
    Divide *Data* into $D_1, \ldots, D_n$ with $D_i = \{x \in Data | A_{max}(x) = a_i\}$
    For all $i \in \{1, \ldots, n\}$
      If all $x \in D_i$ belong to the same class C_i
        Then generate leaf node K_i of class C_i
        Else GENERATEDECISIONTREE(D_i, K_i)

Gambar 8.26 Algoritma untuk konstruksi pohon keputusan

Selain itu, pohon yang disederhanakan dengan hanya satu atribut diberikan. Pohon ini, yang dibuat dengan pemangkasan, akan menjadi penting untuk peningkatan jumlah data pelatihan. Dalam contoh kecil ini, hal itu masih kurang masuk akal. Tingkat kesalahan untuk kedua pohon pada data pelatihan juga diberikan. Angka-angka dalam tanda kurung setelah keputusan memberikan ukuran dataset yang mendasari dan jumlah kesalahan. Misalnya, garis Sun = yes: yes (3.0/1.0) di pohon teratas menunjukkan bahwa untuk simpul

daun ini Matahari = ya, ada tiga contoh pelatihan, salah satunya diklasifikasikan secara salah. Dengan demikian, pengguna dapat membaca dari sini apakah keputusan tersebut didasarkan secara statistik dan/atau pasti.

Pada Gambar 8.26 kita sekarang dapat memberikan skema algoritma pembelajaran untuk menghasilkan pohon keputusan. Kita sekarang akrab dengan dasar-dasar pembuatan pohon keputusan secara otomatis. Untuk aplikasi praktis, bagaimanapun, ekstensi penting diperlukan. Kami akan memperkenalkan ini menggunakan aplikasi LEXMED yang sudah dikenal.

8.11 PEMBELAJARAN DIAGNOSIS APENDISITIS

```
|Definition of the classes and attributes
|
|Classes 0=appendicitis negative
| 1=appendicitis positive
0,1.
|
|Attributes
|
Age: continuous.
Sex_(1=m___2=w): 1,2.
Pain_Quadrant1_(0=no__1=yes): 0,1.
Pain_Quadrant2_(0=no__1=yes): 0,1.
Pain_Quadrant3_(0=no__1=yes): 0,1.
Pain_Quadrant4_(0=no__1=yes): 0,1.
Local_guarding_(0=no__1=yes): 0,1.
Generalized_guarding_(0=no__1=yes): 0,1.
Rebound_tenderness_(0=no__1=yes): 0,1.
Pain_on_tapping_(0=no__1=yes): 0,1.
```

```
Pain_during_rectal_examination_(0=no__1=yes): 0,1.
Temp_axial: continuous.
Temp_rectal: continuous.
Leukocytes: continuous.
Diabetes_mellitus_(0=no__1=yes): 0,1
```

Dalam proyek penelitian LEXMED, sistem pakar untuk diagnosis apendisitis dikembangkan di atas database data pasien. Sistem, yang bekerja dengan metode entropi maksimum, dijelaskan dalam Bagian. 7.3. Kami sekarang menggunakan database LEXMED untuk menghasilkan pohon keputusan untuk mendiagnosis apendisitis dengan C4.5. Gejala yang digunakan sebagai atribut didefinisikan dalam file app.names:

Kami melihat bahwa selain banyak atribut biner seperti berbagai gejala nyeri, gejala terus menerus seperti usia dan suhu demam juga terjadi. Dalam file data pelatihan berikut, app.data, di setiap baris sebuah kasus dijelaskan. Baris pertama adalah pasien laki-laki 19 tahun dengan nyeri pada kuadran ketiga (kanan bawah, di mana usus buntu), dua nilai demam 36,2 dan 37,8 derajat Celcius nilai leukosit 13400 dan diagnosis positif, yaitu , apendiks yang meradang.

```
19,1,0,0,1,0,1,0,1,1,0,362,378,13400,0,1
13,1,0,0,1,0,1,0,1,1,1,383,385,18100,0,1
32,2,0,0,1,0,1,0,1,1,0,364,374,11800,0,1
18,2,0,0,1,1,0,0,0,0,0,362,370,09300,0,0
73,2,1,0,1,1,1,0,1,1,1,376,380,13600,1,1
30,1,1,1,1,1,0,1,1,1,1,377,387,21100,0,1
56,1,1,1,1,1,0,1,1,1,0,390,?,14100,0,1
36,1,0,0,1,0,1,0,1,1,0,372,382,11300,0,1
36,2,0,0,1,0,0,0,1,1,1,370,379,15300,0,1
33,1,0,0,1,0,1,0,1,1,0,367,376,17400,0,1
19,1,0,0,1,0,0,0,1,1,0,361,375,17600,0,1
12,1,0,0,1,0,1,0,1,1,0,364,370,12900,0,0
...
```

Tanpa merinci database, penting untuk disebutkan bahwa hanya pasien yang diduga menderita radang usus buntu saat tiba di rumah sakit dan kemudian dioperasi yang dimasukkan dalam database. Kami melihat di baris ketujuh bahwa C4.5 juga dapat menangani nilai yang hilang. Data tersebut berisi 9764 kasus.

```
unixprompt> c4.5 -f app -u -m 100

C4.5 [release 8] decision tree generator
                                        Wed Aug 23 13:13:15 2006
----------------------------------------

Read 9764 cases (15 attributes) from app.data

Decision Tree:
```

```
Leukocytes <= 11030 :
| Rebound_tenderness = 0:
| | Temp_rectal > 381 : 1 (135.9/54.2)
| | Temp_rectal <= 381 :
| | | Local_guarding = 0: 0 (1453.3/358.9)
| | | Local_guarding = 1:
| | | | Sex_(1=m___2=w) = 1: 1 (160.1/74.9)
| | | | Sex_(1=m___2=w) = 2: 0 (286.3/97.6)
| Rebound_tenderness = 1:
| | Leukocytes <= 8600 :
| | | Temp_rectal > 378 : 1 (176.0/59.4)
| | | Temp_rectal <= 378 :
| | | | Sex_(1=m___2=w) = 1:
| | | | | Local_guarding = 0: 0 (110.7/51.7)
| | | | | Local_guarding = 1: 1 (160.6/68.5)
| | | | Sex_(1=m___2=w) = 2:
| | | | | Age <= 14 : 1 (131.1/63.1)
| | | | | Age > 14 : 0 (398.3/137.6)
| | Leukocytes > 8600 :
| | | Sex_(1=m___2=w) = 1: 1 (429.9/91.0)
| | | Sex_(1=m___2=w) = 2:
| | | | Local_guarding = 1: 1 (311.2/103.0)
| | | | Local_guarding = 0:
| | | | | Temp_rectal <= 375 : 1 (125.4/55.8)
| | | | | Temp_rectal > 375 : 0 (118.3/56.1)
Leukocytes > 11030 :
| Rebound_tenderness = 1: 1 (4300.0/519.9)
| Rebound_tenderness = 0:
| | Leukocytes > 14040 : 1 (826.6/163.8)
| | Leukocytes <= 14040 :
| | | Pain_on_tapping = 1: 1 (260.6/83.7)
| | | Pain_on_tapping = 0:
| | | | Local_guarding = 1: 1 (117.5/44.4)
| | | | Local_guarding = 0:
| | | | | Temp_axial <= 368 : 0 (131.9/57.4)
| | | | | Temp_axial > 368 : 1 (130.5/57.8)

Simplified Decision Tree:

Leukocytes > 11030 : 1 (5767.0/964.1)
Leukocytes <= 11030 :
| Rebound_tenderness = 0:
| | Temp_rectal > 381 : 1 (135.9/58.7)
| | Temp_rectal <= 381 :
| | | Local_guarding = 0: 0 (1453.3/370.9)
| | | Local_guarding = 1:
| | | | Sex_(1=m___2=w) = 1: 1 (160.1/79.7)
| | | | Sex_(1=m___2=w) = 2: 0 (286.3/103.7)
| Rebound_tenderness = 1:
| | Leukocytes > 8600 : 1 (984.7/322.6)
| | Leukocytes <= 8600 :
```

```
| | | Temp_rectal > 378 : 1 (176.0/64.3)
| | | Temp_rectal <= 378 :
| | | | Sex_(1=m___2=w) = 1:
| | | | | Local_guarding = 0: 0 (110.7/55.8)
| | | | | Local_guarding = 1: 1 (160.6/73.4)
| | | | Sex_(1=m___2=w) = 2:
| | | | | Age <= 14 : 1 (131.1/67.6)
| | | | | Age > 14 : 0 (398.3/144.7)

Evaluation on training data (9764 items):

         Before Pruning           After Pruning
        ----------------   --------------------------
        Size      Errors   Size      Errors   Estimate
          37  2197(22.5%)     21  2223(22.8%)    (23.6%)   <<

Evaluation on test data (4882 items):

         Before Pruning           After Pruning
        ----------------   --------------------------
        Size      Errors   Size      Errors   Estimate
          37  1148(23.5%)     21  1153(23.6%)    (23.6%)   <<

         (a)  (b)      <-classified as
        ----  ----
         758  885      (a): class 0
         268 2971      (b): class 1
```

Atribut Berkelanjutan

Pada pohon yang dihasilkan untuk diagnosis apendisitis terdapat node Leukosit > 11030 yang jelas berasal dari atribut kontinu Leukosit dengan menetapkan ambang batas pada nilai 11030. C4.5 telah membuat atribut biner Leukosit > 11030 dari atribut kontinu Leukosit. Ambang batas ΘD;A untuk atribut A ditentukan oleh algoritma berikut: untuk semua nilai v yang ada dalam data pelatihan D, atribut biner A > v dihasilkan dan perolehan informasinya dihitung. Ambang batas ΘD;A kemudian diatur ke nilai v dengan perolehan informasi maksimum, yaitu:

$$\Theta_{D,A} = argmax_v\{G\{D, A > v\}\}$$

Untuk atribut seperti nilai leukosit atau usia pasien, keputusan berdasarkan diskritisasi biner mungkin terlalu tidak tepat. Namun demikian, tidak perlu melakukan diskritisasi yang lebih baik karena setiap atribut kontinu diuji pada setiap simpul yang baru dibangkitkan dan dengan demikian dapat terjadi berulang kali dalam satu pohon dengan ambang batas ΘD;A yang berbeda. Jadi pada akhirnya kita memperoleh diskritisasi yang sangat baik yang kehalusannya sesuai dengan masalah.

8.12 MEMANGKAS – MEMOTONG POHON

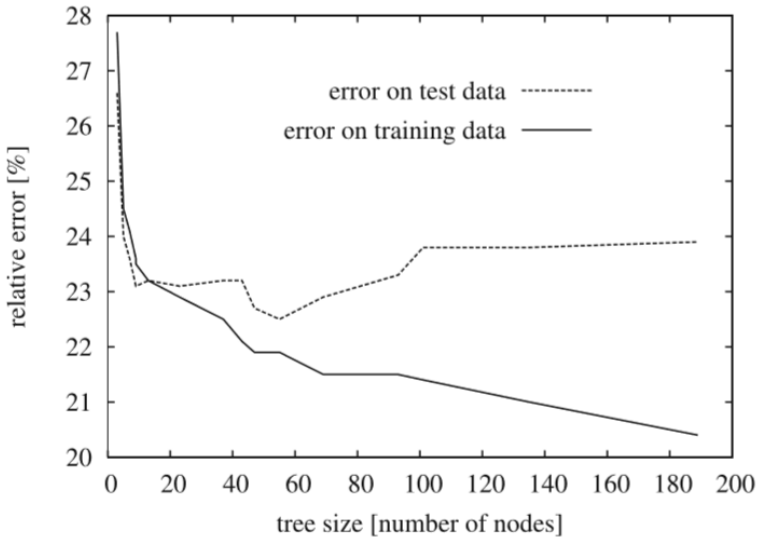
Sejak zaman Aristoteles telah ditetapkan bahwa dari dua teori ilmiah yang menjelaskan situasi yang sama sama baiknya, lebih disukai yang lebih sederhana. Hukum

ekonomi ini, juga sekarang dikenal sebagai pisau cukur Occam, sangat penting untuk pembelajaran mesin dan *Data mining*.

Sebuah pohon keputusan adalah teori untuk menggambarkan data pelatihan. Sebuah teori yang berbeda untuk menggambarkan data adalah data itu sendiri. Jika pohon mengklasifikasikan semua data tanpa kesalahan, tetapi jauh lebih kompak dan dengan demikian lebih mudah dipahami oleh manusia, maka lebih disukai menurut pisau cukur Occam. Hal yang sama berlaku untuk dua pohon keputusan dengan ukuran berbeda. Jadi tujuan dari setiap algoritma untuk menghasilkan pohon keputusan harus menghasilkan pohon keputusan sekecil mungkin untuk tingkat kesalahan yang diberikan. Di antara semua pohon dengan tingkat kesalahan tetap, pohon terkecil harus selalu dipilih.

Sampai sekarang kita belum mendefinisikan istilah tingkat kesalahan secara tepat. Seperti yang telah disebutkan beberapa kali, penting agar pohon yang dipelajari tidak hanya menghafal data pelatihan, tetapi juga menggeneralisasikannya dengan baik. Untuk menguji kemampuan pohon untuk menggeneralisasi, kami membagi data yang tersedia menjadi satu set data pelatihan dan satu set v. Data uji disembunyikan dari algoritma pembelajaran dan hanya digunakan untuk pengujian. Jika dataset besar tersedia, seperti data usus buntu, maka kita dapat misalnya menggunakan dua pertiga dari data untuk pembelajaran dan sepertiga sisanya untuk pengujian.

Selain pemahaman yang lebih baik, pisau cukur Occam memiliki pembenaran penting lainnya: kemampuan generalisasi. Semakin kompleks model (di sini pohon keputusan), semakin banyak detail yang diwakili, tetapi pada tingkat yang sama semakin sedikit model yang dapat ditransfer ke data baru. Hubungan ini diilustrasikan pada Gambar 8.27. Pohon keputusan dari berbagai ukuran dilatih terhadap data apendisitis. Dalam grafik, kesalahan klasifikasi pada data pelatihan dan data uji diberikan. Tingkat kesalahan pada data pelatihan menurun secara monoton dengan ukuran pohon. Hingga ukuran pohon 55 node, tingkat kesalahan pada data uji juga menurun. Namun, jika pohon tumbuh lebih jauh, maka tingkat kesalahan mulai meningkat lagi! Efek ini, yang telah kita lihat dalam metode tetangga terdekat, disebut *overfitting*.



Gambar 8.27 Kurva pembelajaran C4.5 pada data apendisitis. Kami dengan jelas melihat *overfitting* pohon dengan lebih dari 55 node

Kami akan memberikan konsep ini, yang penting untuk hampir semua proses pembelajaran, definisi umum yang diambil dari Penelitian sebelumnya:

Definisi 8.7
Biarkan algoritma pembelajaran tertentu, yaitu agen pembelajaran, diberikan. Kami menyebut agen A overfit ke data pelatihan jika ada agen lain A' yang kesalahannya pada data pelatihan lebih besar dari pada A, tetapi kesalahannya pada seluruh distribusi data lebih kecil daripada kesalahan A.

Bagaimana kita sekarang dapat menemukan titik kesalahan minimum ini pada data uji? Algoritma yang paling jelas disebut validasi silang. Selama konstruksi pohon, kesalahan pada data uji diukur secara paralel. Segera setelah kesalahan meningkat secara signifikan, pohon dengan kesalahan minimum disimpan. Algoritma ini digunakan oleh sistem CART yang disebutkan sebelumnya.

C4.5 bekerja agak berbeda. Pertama, menggunakan algoritma GENERATEDECISIONTREE dari Gambar 8.26, menghasilkan sebuah pohon yang biasanya overfit. Kemudian, dengan menggunakan pemangkasan, ia mencoba untuk memotong simpul pohon sampai kesalahan pada data uji, yang diperkirakan oleh kesalahan pada data pelatihan, mulai meningkat.[45] Seperti konstruksi pohon, ini juga merupakan algoritma serakah. Ini berarti bahwa setelah sebuah simpul dipangkas, ia tidak dapat dimasukkan kembali, bahkan jika ini ternyata lebih baik.

Nilai yang hilang

Seringkali nilai atribut individu hilang dari data pelatihan. Dalam kumpulan data LEXMED, entri berikut terjadi:

56, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 390, ?, 14100, 0, 1.

di mana salah satu nilai demam hilang. Namun demikian, data tersebut dapat digunakan selama konstruksi pohon keputusan. Kita dapat menetapkan atribut nilai yang paling sering dari seluruh dataset atau yang paling sering dari semua titik data dari kelas yang sama. Bahkan lebih baik untuk mengganti distribusi probabilitas dari semua nilai atribut untuk nilai atribut yang hilang dan untuk membagi contoh pelatihan menjadi cabang-cabang sesuai dengan distribusi ini. Ini secara kebetulan merupakan alasan untuk kemunculan nilai non-bilangan bulat dalam ekspresi dalam tanda kurung di sebelah simpul daun dari pohon C4.5. Nilai yang hilang dapat terjadi tidak hanya selama pembelajaran, tetapi juga selama klasifikasi. Ini ditangani dengan cara yang sama seperti selama pembelajaran.

Ringkasan

Belajar pohon keputusan adalah pendekatan favorit untuk tugas-tugas klasifikasi. Alasan untuk ini adalah aplikasi dan kecepatannya yang sederhana. Pada kumpulan data sekitar 10.000 titik data LEXMED dengan masing-masing 15 atribut, C4.5 membutuhkan sekitar 0,3 detik untuk pembelajaran. Ini sangat cepat dibandingkan dengan algoritma pembelajaran lainnya.

Namun, bagi pengguna juga penting bahwa pohon keputusan sebagai model yang dipelajari dapat dipahami dan berpotensi diubah. Juga tidak sulit untuk secara otomatis mengubah pohon keputusan menjadi serangkaian pernyataan if-then-else dan dengan demikian secara efisien membangunnya ke dalam program yang sudah ada.

Karena algoritma serakah digunakan untuk konstruksi pohon serta selama pemangkasan, pohon pada umumnya kurang optimal. Pohon keputusan yang ditemukan biasanya memiliki tingkat kesalahan yang relatif kecil. Namun, ada potensi pohon yang lebih baik, karena pencarian keserakahan heuristik C4.5 lebih memilih pohon kecil dan

[45] Akan lebih baik untuk menggunakan kesalahan pada data uji secara langsung. Setidaknya ketika jumlah data pelatihan cukup untuk membenarkan set pengujian yang terpisah.

atribut menunjukkan nilai kelemahan dengan perolehan informasi yang tinggi di bagian atas pohon.

8.13 *CROSS* – VALIDASI DAN *OVERFITTING*

Banyak algoritma pembelajaran memiliki masalah *overfitting*. Algoritma pembelajaran yang kuat, seperti misalnya pembelajaran pohon keputusan, dapat menyesuaikan kompleksitas model yang dipelajari dengan kompleksitas data pelatihan. Hal ini menyebabkan *overfitting* jika ada noise dalam data.

Dengan validasi silang, salah satu upaya untuk mengoptimalkan kompleksitas model sedemikian rupa sehingga meminimalkan kesalahan klasifikasi atau aproksimasi pada kumpulan data uji yang tidak diketahui. Hal ini membutuhkan kompleksitas model yang dapat dikontrol oleh suatu parameter c. Untuk pohon keputusan ini, misalnya, ukuran pohon γ. Untuk k metode tetangga terdekat, parameternya adalah k, jumlah tetangga terdekat, sedangkan untuk jaringan saraf adalah jumlah neuron tersembunyi (lihat Bab 9).

Kami memvariasikan parameter γ saat melatih algoritme pada kumpulan data pelatihan dan memilih nilai γ yang meminimalkan kesalahan pada kumpulan data pengujian independen. k-ary *cross*-validation bekerja sesuai dengan skema berikut:

CROSSVALIDATION($\boldsymbol{X}, k$)

Partition data into k equally sized blocks $\boldsymbol{X} = \boldsymbol{X}_1 \cup \ldots \cup \boldsymbol{X}_k$
For all $\gamma \in \{\gamma_{min}, \ldots, \gamma_{max}\}$
  For all $i \in \{1, \ldots, k\}$
    Train a model of complexity γ on $\boldsymbol{X} \backslash \boldsymbol{X}_i$
    Compute the error $E(\gamma, \boldsymbol{X}_i)$ on the test set $\boldsymbol{X}_i$
  Compute the mean error $E(\gamma) = \frac{1}{k} \sum_{i=1}^{k} E(\gamma, \boldsymbol{X}_i)$
Choose the value $\gamma_{opt} = \text{argmin}_\gamma E(\gamma)$ with smallest mean error
Train the final model with complexity γ_{opt} on the whole data set $\boldsymbol{X}$

Seluruh kumpulan data X dibagi menjadi k blok berukuran sama. Kemudian algoritma dilatih k kali pada k = 1 blok dan diuji pada blok yang tersisa. Kesalahan yang dihitung k dirata-ratakan dan nilai γ_{opt} dengan kesalahan rata-rata terkecil dipilih untuk melatih model akhir pada seluruh kumpulan data X.

Jika set pelatihan X besar, dapat dibagi, misalnya, menjadi k = 3 atau k = 10 blok. Peningkatan yang dihasilkan dalam kompleksitas komputasi biasanya dapat diterima. Untuk set pelatihan kecil, lebih baik melatih semua n vektor fitur jika memungkinkan. Kemudian seseorang dapat memilih k n, yang menghasilkan apa yang disebut validasi silang tanpa-satu.

Validasi silang adalah metode optimasi otomatis yang paling penting dan paling banyak digunakan untuk kompleksitas model. Ini memecahkan masalah *overfitting*. Kita dapat memperoleh wawasan tambahan tentang masalah ini dari tinjauan singkat pada apa yang disebut sebagai tradeoff varians bias. Jika model yang terlalu sederhana digunakan, ini memaksa perkiraan data yang tidak optimal dalam arah tertentu (bias). Di sisi lain, jika model yang terlalu kompleks digunakan, model tersebut akan cenderung memenuhi kumpulan data apa pun. Jadi, untuk sampel data baru dari distribusi yang sama, ia mungkin mempelajari model yang sangat berbeda. Oleh karena itu model sangat bervariasi untuk perubahan data (varians).

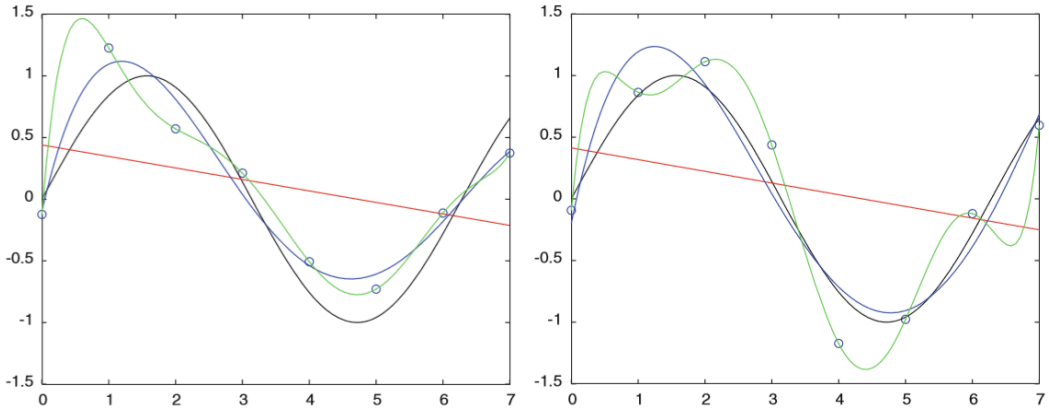
Hubungan ini dapat diilustrasikan dengan contoh aproksimasi fungsi menggunakan polinomial. Jika seseorang memilih derajat polimonial sebagai parameter kompleksitas γ, derajat 1 mendekati garis, yang bukan merupakan pendekatan yang baik untuk data non-

trivial. Sebaliknya, jika derajat polinomial sama dengan n - 1 untuk n titik, ia cenderung mengenai setiap titik dan mengurangi kesalahan pada data pelatihan menjadi nol. Hal ini dapat dilihat pada Gambar 8.28 kiri, di mana delapan titik data telah dihasilkan menggunakan

$$Y(x) = \sin(x) + g(0,02)$$

dimana g(0, 0.2) menghasilkan derau terdistribusi normal dengan rata-rata nol dan simpangan baku 0,2.

Pada gambar kiri, selain titik data, fungsi sinus yang mendasari (hitam) tertulis, serta garis lurus (merah) yang didekati menggunakan metode kuadrat terkecil. Selain itu ,polinomial derajat ketujuh (hijau) diinterpolasi melalui titik-titik, mengenai setiap titik, serta polinomial derajat keempat (biru), yang paling dekat dengan kurva sinus. Pada gambar kanan, perkiraan yang sama dilakukan lagi pada delapan titik data baru. Orang dapat melihat dengan sangat baik bahwa meskipun data berbeda, kedua garis lurus sangat menyimpang dari data (bias besar), tetapi memiliki grafik fungsi yang serupa (varians sangat kecil). Kedua polinomial derajat ketujuh, di sisi lain, cocok dengan data dengan sempurna (bias nol), tetapi mereka memiliki grafik fungsi yang sangat berbeda (varians sangat besar). Ini adalah efek *overfitting* yang jelas. Polinomial derajat keempat menyajikan kompromi yang baik. Validasi silang mungkin akan menghasilkan derajat empat sebagai solusi optimal.



Gambar 8.28 Dua kumpulan data berbeda yang didekati dengan polinomial derajat 1, 4 dan 7. Tradeoff varians bias terlihat jelas

8.14 MEMPELAJARI JARINGAN BAYESIAN

Pada bagian sebelumnya, ditunjukkan bagaimana membangun jaringan Bayesian secara manual. Sekarang kami akan memperkenalkan algoritma untuk induksi jaringan Bayesian. Mirip dengan proses pembelajaran yang dijelaskan sebelumnya, jaringan Bayesian secara otomatis dihasilkan dari file yang berisi data pelatihan. Proses ini biasanya didekomposisi menjadi dua bagian.

1. Mempelajari struktur jaringan: Untuk variabel yang diberikan, topologi jaringan dihasilkan dari data pelatihan. Langkah pertama ini jauh lebih sulit dan akan diberikan perhatian lebih lanjut nanti.
2. Mempelajari probabilitas bersyarat: Untuk topologi jaringan yang diketahui, CPT harus diisi dengan nilai. Jika data pelatihan cukup tersedia, semua probabilitas bersyarat yang diperlukan dapat diperkirakan dengan menghitung frekuensi dalam data. Langkah ini dapat diotomatisasi dengan relatif mudah. Kami sekarang akan

menjelaskan bagaimana jaringan Bayesian belajar menggunakan algoritma sederhana.

Mempelajari Struktur Jaringan

Selama pengembangan jaringan Bayesian, ketergantungan kausal dari variabel harus diperhitungkan untuk mendapatkan jaringan sederhana dengan kualitas yang baik. Pengembang manusia bergantung pada pengetahuan latar belakang, yang tidak tersedia untuk mesin. Oleh karena itu, prosedur ini tidak dapat dengan mudah diotomatisasi.

Menemukan struktur optimal untuk jaringan Bayesian dapat dirumuskan sebagai masalah pencarian klasik. Biarkan satu set variabel V_1, ..., V_n dan file dengan data pelatihan diberikan. Kami mencari satu set tepi berarah tanpa siklus antara node V_1, … , V_n, yaitu, grafik asiklik terarah (DAG) yang mereproduksi data yang mendasarinya sebaik mungkin.

Pertama kita amati ruang pencarian. Jumlah DAG yang berbeda tumbuh lebih dari eksponensial dengan jumlah node. Untuk lima node ada 29281 dan untuk sembilan node sekitar 1015 DAG yang berbeda. Pencarian kombinatorial tanpa informasi di ruang semua grafik dengan sekumpulan variabel tertentu tidak ada harapan jika jumlah variabel bertambah. Oleh karena itu algoritma heuristik harus digunakan. Ini menimbulkan pertanyaan tentang fungsi evaluasi untuk jaringan Bayesian. Dimungkinkan untuk mengukur kesalahan klasifikasi jaringan selama aplikasi ke satu set data uji, seperti yang dilakukan, misalnya, di C4.5. Untuk ini, bagaimanapun, probabilitas yang dihitung oleh jaringan Bayesian harus dipetakan ke sebuah keputusan.

Pengukuran langsung kualitas jaringan dapat dilakukan melalui distribusi probabilitas. Kami berasumsi bahwa, sebelum membangun jaringan dari data, kami dapat menentukan (memperkirakan) distribusinya. Kemudian kami memulai pencarian di ruang semua DAG, memperkirakan nilai CPT untuk setiap DAG (yaitu, untuk setiap jaringan Bayesian) menggunakan data, dan dari situ kami menghitung distribusi dan membandingkannya dengan distribusi yang diketahui dari data. Untuk perbandingan distribusi kita jelas membutuhkan metrik jarak. Mari kita perhatikan contoh prediksi cuaca dari Latihan 7.3 dengan tiga variabel Sky, Bar, Prec, dan distribusi

***P**(Sky, Bar, Prec) = (0.40, 0.07, 0.08, 0.10, 0.09, 0.11, 0.03, 0.12)*

Pada Gambar 8.29 disajikan dua jaringan Bayesian, yang sekarang akan kita bandingkan dalam hal kualitasnya. Masing-masing jaringan ini membuat asumsi independensi, yang divalidasi karena kita menentukan distribusi jaringan dan kemudian membandingkannya dengan distribusi aslinya (lihat Latihan 8.16).

Karena, untuk variabel konstan yang ditentukan sebelumnya, distribusinya jelas diwakili oleh vektor dengan panjang konstan, kita dapat menghitung norma Euclidian dari perbedaan dua vektor sebagai jarak antara distribusi. Kami mendefinisikan

$$d_q(x, y) = \sum_i (x_i - y_i)^2$$

sebagai jumlah kuadrat jarak komponen vektor dan hitung jarak d_q (P_a, P) 0,0029 P_a distribusi jaringan 1 dari distribusi aslinya. Untuk jaringan 2 kita hitung d_q (P_b, P) 0,014. Jelas jaringan 1 adalah perkiraan distribusi yang lebih baik. Seringkali, alih-alih jarak kuadrat, yang disebut jarak Kullback–Leibler

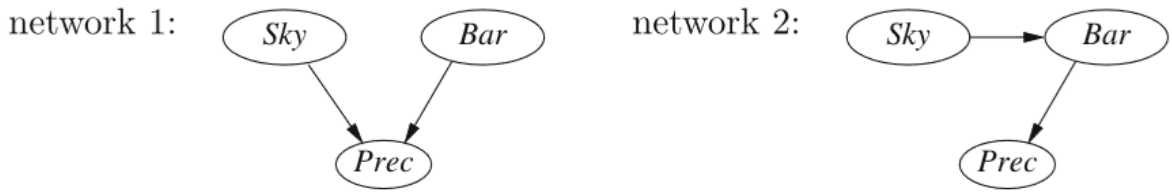
$$d_q(x, y) = \sum_i y_i(log_2\, y_i - log_2\, x_i)$$

metrik teori informasi, digunakan. Dengan itu kami menghitung d_k (P_a, P) 0,017 dan dk (P_b, P) 0,09 dan sampai pada kesimpulan yang sama seperti sebelumnya. Diharapkan bahwa jaringan dengan banyak tepi mendekati distribusi lebih baik daripada yang memiliki sedikit tepi. Jika semua tepi dalam jaringan dibangun, maka itu menjadi sangat membingungkan

dan menciptakan risiko *overfitting*, seperti yang terjadi pada banyak algoritma pembelajaran lainnya. Untuk menghindari *overfitting*, kami memberikan bobot yang lebih besar pada jaringan kecil menggunakan fungsi evaluasi heuristik

$$f(N) = Ukuran(N) + w \cdot d_k(\boldsymbol{P_N}, \boldsymbol{P})$$

Di sini Ukuran(N) adalah jumlah entri dalam CPT dan PN adalah distribusi jaringan N. w adalah faktor bobot, yang harus disesuaikan secara manual.

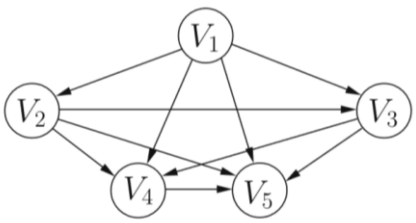


Gambar 8.29 Dua jaringan Bayesian untuk pemodelan contoh prediksi cuaca dari Latihan 7.3

Algoritma pembelajaran untuk jaringan Bayesian dengan demikian menghitung evaluasi heuristik f(N) untuk banyak jaringan yang berbeda dan kemudian memilih jaringan dengan nilai terkecil. Seperti disebutkan sebelumnya, kesulitan terdiri dari pengurangan ruang pencarian untuk topologi jaringan yang kita cari. Sebagai algoritme sederhana, dimungkinkan, mulai dari pengurutan (misalnya kausal) variabel V_1, … , V_n, untuk memasukkan dalam grafik hanya sisi-sisi yang i < j. Kita mulai dengan model maksimal yang memenuhi kondisi ini. Jaringan ini ditunjukkan pada Gambar 8.30 untuk lima variabel terurut.

Sekarang, misalnya dalam semangat pencarian serakah, satu demi satu tepi dihilangkan sampai nilai f tidak lagi berkurang. Algoritma ini tidak praktis untuk jaringan yang lebih besar dalam bentuk ini. Ruang pencarian yang besar, penyetelan manual bobot w, dan perbandingan yang diperlukan dengan distribusi tujuan P adalah alasan untuk ini, karena ini bisa menjadi terlalu besar, atau kumpulan data yang tersedia mungkin terlalu kecil.

Faktanya, penelitian tentang pembelajaran jaringan Bayesian masih berjalan lancar, dan ada sejumlah besar algoritma yang disarankan, misalnya algoritma EM, metode rantai Markov Monte Carlo, dan pengambilan sampel Gibbs. Selain pembelajaran batch, yang telah disajikan di sini, di mana jaringan dihasilkan satu kali dari seluruh dataset, ada juga algoritma inkremental, di mana setiap kasus baru digunakan untuk meningkatkan jaringan. Implementasi dari algoritma ini juga ada, seperti Hugin (www. hugin.com) dan Bayesware (www.bayesware.com).



Gambar 8.30 Jaringan maksimal dengan lima variabel dan edge (V_i, V_j) yang memenuhi kondisi i < j

Pengklasifikasi Naive Bayes

Pada Gambar 7.14 diagnosis apendisitis dimodelkan sebagai jaringan Bayesian. Karena ujung yang diarahkan memulai diagnosis simpul dan tidak ada ujungnya, rumus Bayes harus digunakan untuk menjawab pertanyaan diagnosis. Untuk gejala S_1, … , S_n dan nilai k diagnosis D dengan nilai b_1, … , b_k kita hitung probabilitas

$$P(D|S_1, \dots, S_n) = \frac{P(S_{1,} \dots ., S_n|D) \,.\, P(D)}{(D|S_1, \dots, S_n)}$$

untuk diagnosis yang diberikan gejala pasien. Dalam kasus terburuk, yaitu, jika tidak ada variabel bebas, semua kombinasi dari semua gejala dan D perlu ditentukan untuk semua 20 643 840 probabilitas distribusi P(S_1,…, S_n, D). Ini akan membutuhkan database yang sangat besar. Dalam kasus jaringan Bayesian LEXMED, jumlah nilai yang diperlukan (dalam CPT) dikurangi menjadi 521. Jaringan dapat lebih disederhanakan, bagaimanapun, di mana kita menganggap semua variabel gejala independen bersyarat diberikan D, yaitu:

$$P(S_1, \dots, S_n|D) = P(S_1|D) \dots P(S_n|D)$$

Jaringan Bayesian untuk apendisitis kemudian disederhanakan menjadi bintang yang ditunjukkan pada Gambar 8.31. Dengan demikian kita memperoleh rumus

Persamaan 8.8

$$P(D|S_1, \dots, S_n) = \frac{P(D) \prod_{i=1}^{n} P\,(S_i|D)}{(D|S_1, \dots, S_n)}$$

Probabilitas yang dihitung ditransformasikan menjadi keputusan oleh pengklasifikasi naif Bayes sederhana, yang memilih P(D = d_i|S_1,…, S_n) maksimum dari semua nilai di dalam D. Artinya, ia menentukan

$$d_{\text{Naive-Bayes}} = \text{argmax}_{i \in \{1, \dots, k\}} P(D = d_i|S_1, \dots, S_n)$$

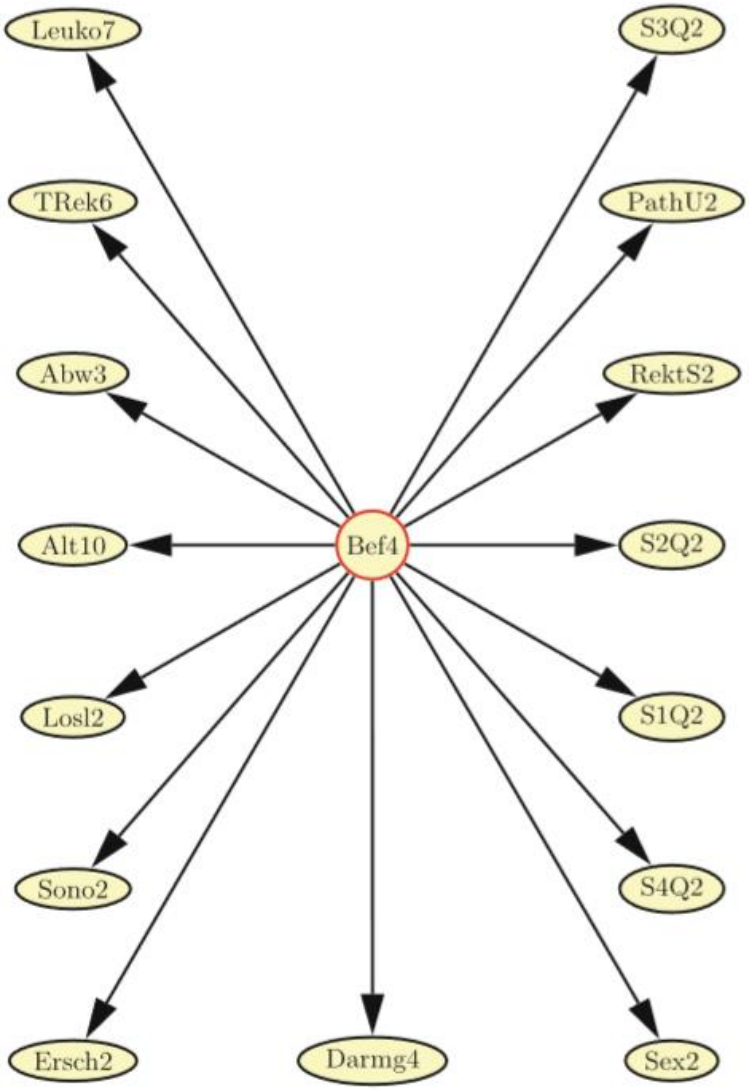
Karena penyebut dalam (8.8) adalah konstan, penyebut dapat dihilangkan selama maksimalisasi, yang menghasilkan rumus naive Bayes

$$d_{\text{Naive-Bayes}} = \text{argmax}_{\text{i}\in\{\text{l},\dots,\text{k}\}}\text{P(D = di)} = \prod_{i=1}^{n} P\,(S_i|D)$$

Karena beberapa node sekarang memiliki lebih sedikit ancestor, jumlah nilai yang diperlukan untuk menggambarkan distribusi LEXMED di CPT berkurang, menurut (7.24), menjadi

$$6 \,.\, 4 + 5 \,.\, 4 + 9 \,.\, 4 + 3 \,.\, 4 + 10 \,.\, (1 \,.\, 4) + 3 = 143$$

Untuk sistem diagnostik medis seperti LEXMED, penyederhanaan ini tidak dapat diterima. Tetapi untuk tugas dengan banyak variabel independen, naive Bayes sebagian atau bahkan sangat cocok, seperti yang akan kita lihat dalam contoh klasifikasi teks.



Gambar 8.31 Jaringan Bayesian untuk aplikasi LEXMED dengan asumsi bahwa semua gejala tidak tergantung pada diagnosis yang diberikan

Omong-omong, klasifikasi naif Bayes sama ekspresifnya dengan sistem skor linier. Artinya, semua skor berbagi asumsi yang mendasari bahwa semua gejala independen bersyarat diberikan diagnosis. Namun demikian, skor tidak digambarkan dalam pengobatan hari ini. Meskipun fakta bahwa itu dihasilkan dari database yang lebih baik dan representatif, skor Ohmann dibandingkan dengan LEXMED pada Gambar. 7.10 memiliki kualitas diagnosis yang lebih buruk. Ekspresinya yang terbatas tentu menjadi alasan untuk ini. Misalnya, seperti naive Bayes, tidak mungkin untuk memodelkan ketergantungan antara gejala menggunakan skor.

Estimasi Probabilitas

Jika kita mengamati rumus Bayes Naive (Persamaan 8.8), kita melihat bahwa seluruh ekspresi menjadi nol segera setelah salah satu faktor $P(S_i|D)$di ruas kanan menjadi nol. Secara teoritis tidak ada yang salah di sini. Namun, dalam praktiknya, ini dapat menyebabkan efek nyaman yang berlebihan jika $P(S_i|D)$ kecil. diperkirakan dengan menghitung frekuensi dan mensubstitusikannya ke dalam

$$P(S_i = x|D = y) - \frac{|S_i = x \ \wedge D = y|}{|D = y|}$$

Asumsikan bahwa untuk variabel S_i: P(S_i x|D = y) 0,01 dan ada 40 kasus pelatihan dengan D = y. Kemudian dengan probabilitas tinggi tidak ada kasus pelatihan dengan S_i = x dan D = y, dan kami memperkirakan P(S_i = x|D = y) = 0. Untuk nilai yang berbeda D = z, asumsikan bahwa hubungan terletak serupa, tetapi estimasi menghasilkan nilai yang lebih besar dari nol untuk semua P(Si = x|D = z). Jadi nilai D = z selalu lebih disukai, yang tidak mencerminkan distribusi probabilitas yang sebenarnya. Oleh karena itu, ketika memperkirakan probabilitas, rumusnya

$$P(A|B) \approx \frac{|A \wedge B|}{|B|} = \frac{n_{AB}}{n_B}$$

digantikan oleh

$$P(A|B) \approx = \frac{n_{AB} + n_p}{n_B + m}$$

di mana p =P(A) adalah probabilitas apriori untuk A, dan m adalah konstanta yang dapat dipilih secara bebas dan dikenal sebagai "ukuran data ekuivalen". Semakin besar m, semakin besar bobot probabilitas apriori dibandingkan dengan nilai yang ditentukan dari frekuensi yang diukur.

Klasifikasi Teks dengan Naive Bayes

Naive Bayes sangat sukses dan produktif saat ini dalam klasifikasi teks. Aplikasi utamanya, dan pada saat yang sama sangat penting, adalah pemfilteran email secara otomatis menjadi email yang diinginkan dan tidak diinginkan, atau spam. Dalam filter spam seperti SpamAssassin, di antara metode lainnya digunakan pengklasifikasi Bayes yang belajar memisahkan email yang diinginkan dari spam. SpamAssassin adalah sistem hybrid yang melakukan penyaringan awal menggunakan daftar hitam dan putih. Daftar hitam adalah daftar alamat email yang diblokir dari pengirim spam yang emailnya selalu dihapus, dan daftar putih adalah daftar dengan pengirim yang emailnya selalu terkirim. Setelah prafilter ini, email yang tersisa diklasifikasikan oleh pengklasifikasi naif Bayes menurut konten sebenarnya, dengan kata lain, menurut teks. Nilai kelas yang terdeteksi kemudian dievaluasi dengan skor, bersama dengan atribut lain dari header email seperti domain pengirim, tipe MIME, dll., dan akhirnya disaring.

Di sini kemampuan belajar dari filter naive Bayes cukup penting. Untuk ini pengguna harus terlebih dahulu secara manual mengklasifikasikan sejumlah besar email yang diinginkan atau spam. Kemudian filter dilatih. Agar tetap up to date, filter harus dilatih ulang secara teratur. Untuk ini, pengguna harus mengklasifikasikan dengan benar semua email yang salah diklasifikasikan oleh filter, yaitu, meletakkannya di folder yang sesuai. Filter kemudian terus dilatih ulang dengan email-email ini.

Selain penyaringan spam, ada banyak aplikasi lain untuk klasifikasi teks otomatis. Aplikasi penting termasuk pemfilteran entri yang tidak diinginkan di forum diskusi Internet, dan melacak situs web dengan konten kriminal seperti aktivitas militan atau teroris, pornografi anak, atau rasisme. Ini juga dapat digunakan untuk menyesuaikan mesin pencari agar sesuai dengan preferensi pengguna untuk mengklasifikasikan hasil pencarian dengan lebih baik. Dalam lingkungan industri dan ilmiah, pencarian di seluruh perusahaan dalam database atau dalam literatur berada di latar depan penelitian. Melalui kemampuan belajarnya, sebuah filter dapat beradaptasi dengan kebiasaan dan keinginan masing-masing pengguna.

Kami akan memperkenalkan penerapan naive Bayes pada analisis teks pada contoh teks singkat oleh Alan Turing:

> *"Kita mungkin berharap bahwa mesin pada akhirnya akan bersaing dengan manusia di semua bidang intelektual murni. Tapi mana yang terbaik untuk memulai?*

Bahkan ini adalah keputusan yang sulit. Banyak orang berpikir bahwa aktivitas yang sangat abstrak, seperti bermain catur, adalah yang terbaik. Juga dapat dipertahankan bahwa yang terbaik adalah memberikan mesin itu organ-organ indera terbaik yang dapat dibeli dengan uang, dan kemudian mengajarinya untuk memahami dan berbicara bahasa Inggris. Proses ini bisa mengikuti pengajaran normal seorang anak. Hal-hal akan ditunjukkan dan diberi nama, dll. Sekali lagi saya tidak tahu apa jawaban yang benar, tetapi saya pikir kedua pendekatan itu harus dicoba."

Misalkan teks seperti yang diberikan harus dibagi menjadi dua kelas: "I" untuk menarik dan "¬I untuk tidak menarik. Misalkan juga ada database teks yang sudah diklasifikasikan. Atribut mana yang harus digunakan? Dalam pendekatan klasik untuk konstruksi jaringan Bayesian, kami mendefinisikan satu set atribut seperti panjang teks, panjang kalimat rata-rata, frekuensi relatif dari tanda baca tertentu, frekuensi beberapa kata penting seperti "I" , "mesin", dll. Selama klasifikasi menggunakan naive Bayes, sebaliknya, algoritma primitif yang mengejutkan dipilih. Untuk masing-masing dari n posisi kata dalam teks, sebuah atribut si didefinisikan. Semua kata yang muncul dalam teks diperbolehkan sebagai nilai yang mungkin untuk semua posisi si. Sekarang untuk kelas I dan I nilainya

Persamaan 8.9

$$\mathrm{P}(1|S_1,\ldots,S_n) = \mathrm{c}\ .\ \mathrm{P}(1)\prod\prod_{i=1}^{n} P\,(S_i|I)$$

dan P(¬I|s_1,..., s_n) harus dihitung dan kemudian kelas dengan nilai maksimum yang dipilih. Dalam contoh di atas dengan total 113 kata, ini menghasilkan

P(1|S_1,..., S_n) = c . P(1) . P(S_1 = "Kita"|1) . P(S_2 = "mungkin"|1) ... P(S_{13} = "mencoba"|I)

Dan

P(¬1|s_1, ..., s_n) = c . (P¬) . P(s_1 = "Kita"| ¬I) . P(s_2 ="mungkin"|¬I) ... P(s_{13} = "mencoba"| ¬I)

Pembelajaran di sini cukup sederhana. Probabilitas bersyarat P(s_i |I), P(s_i |¬I) dan probabilitas apriori P(I), P(¬I) harus dihitung secara sederhana. Kami sekarang juga mengasumsikan bahwa P(s_i| I) tidak bergantung pada posisi dalam teks. Ini berarti, misalnya, bahwa

P(s_{61} = "dan"|I) = P(s_{69} = "dan"|I) = P(s_{86} = "dan"|I)

Dengan demikian kita dapat menggunakan ekspresi P(dan|I), dengan variabel biner baru dan, sebagai probabilitas kemunculan "dan" pada posisi arbitrer. Implementasinya dapat dipercepat jika kita menemukan frekuensi ni dari setiap kata wi yang muncul dalam teks dan menggunakan rumus

Persamaan 8.10

$$\mathrm{P}(1|S_1,\ldots,S_n) = \mathrm{c}\ .\ \mathrm{P}(1)\prod\prod_{i=1}^{n} P\,(w_i|I))^{n_i}$$

yang setara dengan (Persamaan 8.9). Harap dicatat bahwa indeks i dalam produk hanya menuju ke nomor l dari kata-kata berbeda yang muncul dalam teks.

Terlepas dari kesederhanaannya, naive Bayes memberikan hasil yang sangat baik untuk klasifikasi teks. Filter spam yang bekerja dengan naive Bayes mencapai tingkat kesalahan jauh di bawah satu persen. Sistem DSPAM dan CRM114 bahkan dapat dilatih dengan sangat baik sehingga mereka hanya salah mengklasifikasikan masing-masing satu dari 7000 atau 8000 email. Ini sesuai dengan kebenaran hampir 99,99%.

8.15 *ONE – CLASS LEARNING*

Tugas klasifikasi dalam pembelajaran terawasi mengharuskan semua data pelatihan diberi label kelas. Namun, ada aplikasi yang hanya tersedia satu kelas label. Contoh klasik adalah mendeteksi kesalahan dalam sistem teknis yang kompleks. Tugasnya adalah mengenali apakah suatu perangkat rusak atau tidak. Ini terdengar seperti masalah dua kelas biasa. Dalam data pelatihan, status perangkat (baik atau rusak) harus disediakan secara manual. Dalam praktiknya, pelatihan pengklasifikasi terjadi selama penyebaran perangkat atau nanti sesuai permintaan saat sedang digunakan dalam produksi. Pengambilan data dalam kondisi bebas kesalahan tidak menjadi masalah. Pengumpulan data dari sistem yang rusak bermasalah karena alasan berikut:

- Pengukuran pada peralatan produksi yang sengaja dibuat cacat dikaitkan dengan biaya tinggi karena pengukuran menyebabkan downtime yang mahal.
- Pengukuran menggunakan peralatan yang rusak dengan kesalahan yang sebenarnya terjadi dalam praktik seringkali tidak mungkin dilakukan karena, selama penyebaran peralatan, kesalahan potensial yang terjadi kemudian mungkin tidak diketahui.
- Tidak ada insinyur yang tahu sebelumnya secara pasti jenis kesalahan apa yang akan terjadi pada peralatan baru. Jika sekarang, selama pelatihan, pengukuran dengan beberapa jenis kesalahan diambil, ini dapat menyebabkan hasil klasifikasi yang buruk. Jika kesalahan yang digunakan untuk pelatihan tidak mewakili perangkat, yaitu jika beberapa jenis kesalahan hilang dalam data pelatihan, maka hypersurface pemisah kelas yang dipelajari di ruang fitur akan sering menyebabkan klasifikasi positif palsu.

Dalam kasus seperti itu, tidak mungkin melatih pengklasifikasi dua kelas dengan cara standar. Sebagai gantinya, seseorang dapat menggunakan pembelajaran satu kelas, yang diperoleh dengan data dari satu kelas selama pelatihan. Asumsikan bahwa tidak ada data negatif yang tersedia. Data positif, yaitu data dari operasi bebas kesalahan, tersedia dalam jumlah yang cukup. Oleh karena itu diperlukan algoritma pembelajaran yang dapat, berdasarkan data bebas kesalahan, menangkap semua status operasional perangkat yang bebas kesalahan dan mengklasifikasikan semua yang lain sebagai negatif.

Untuk tujuan ini, ada sejumlah algoritma berbeda yang dikenal sebagai algoritma pembelajaran satu kelas, sepertideskripsi data tetangga terdekat(NNDD), deskripsi data vektor dukungan (SVDD, lihat juga Bagian 9.6), dll. Dalam statistik, ini dan algoritma terkait termasuk dalam kategori deteksi outlier.

Salah satu algoritma yang paling dikenal saat ini adalah faktor outlier lokal yang menghitung skor outlier untuk suatu titik yang akan diklasifikasikan berdasarkan estimasi densitasnya. Untuk kumpulan data besar dengan jutaan titik, dan untuk titik data berdimensi tinggi dengan ribuan dimensi, algoritme yang dibahas sejauh ini tidak cocok. Algoritme EXPoSE memiliki tingkat kegagalan serendah LOF, tetapi cocok untuk kumpulan data berdimensi tinggi yang besar karena waktu komputasinya yang konstan. Dengan contoh sederhana NNDD, sekarang kami akan memperkenalkan prinsip pembelajaran satu kelas secara singkat.

8.16 DESKRIPSI DATA TETANGGA TERDEKAT

Mirip dengan metode tetangga terdekat, algoritma deskripsi data tetangga terdekat termasuk dalam kategori algoritma pembelajaran malas. Selama pembelajaran, satu-satunya hal yang terjadi adalah normalisasi dan penyimpanan vektor fitur. Dengan demikian pembelajaran yang berkualitas “malas”. Normalisasi setiap fitur individu

diperlukan untuk memberikan bobot yang sama pada setiap fitur. Tanpa normalisasi, fitur dalam kisaran [0, 10^{-4}] akan hilang dalam kebisingan fitur lain dalam kisaran [- 10.000, +10000] Jadi semua fitur secara linier diskalakan ke interval [0, 1].[46] Algoritma tetangga terdekat yang sebenarnya pertama kali berperan selama klasifikasi.

Misalkan X = (x_1,... ,x_n) merupakan himpunan pelatihan yang terdiri dari n vektor ciri. Sebuah pointq baru, yang belum diklasifikasikan, diterima jika

Persamaan 8.11

$$D(q, NN(q)) \leq \gamma D$$

yaitu, jika jarak ke tetangga terdekat tidak lebih besar dari c D. Di sini Dðx;yÞ adalah metrik jarak, misalnya, seperti yang digunakan di sini, jarak Euclidean. Fungsinya

$$NN(q) = argmin\{D(q, x)\}$$

mengembalikan tetangga terdekat dari q dan

$$D = \frac{1}{n}\sum_{i=1}^{n} D\,(x_i, NN(x_i))$$

adalah jarak rata-rata dari tetangga terdekat ke semua titik data di X. Untuk menghitung D, jarak setiap titik ke tetangga terdekatnya dihitung. D adalah mean aritmatika dari jarak yang dihitung ini. Seseorang dapat menggunakan γ=1 dalam Persamaan. 8.11, tetapi kemudian akan mendapatkan hasil klasifikasi yang suboptimal. Lebih baik menentukan parameter c menggunakan validasi silang (Bab 8.5) sehingga tingkat kesalahan pengklasifikasi NNDD sekecil mungkin.

Algoritma NNDD yang digunakan di sini merupakan modifikasi dari algoritma NNDDT yang digunakan, dimana titik q diterima jika jarak ke tetangga terdekatnya dalam data pelatihan tidak lebih besar dari jarak tetangga terdekat yang ditemukan ke tetangga terdekatnya sendiri di data pelatihannya. Secara formal ini berbunyi

Persamaan 8.12

$$D(q), NN(q)) \leq D(NN(q), NN(q))).$$

Rumus ini memiliki beberapa kelemahan dibandingkan dengan Ketimpangan 8.5. Pertama, penggunaan NNDDT (Ketidaksamaan 8.12) menghasilkan garis pemisah kelas yang jelek secara intuitif, seperti yang ditunjukkan pada Gambar 8.32 dalam contoh sederhana dua dimensi. Di bagian tengah Gambar 8.32 kita dapat melihat bahwa titik data kiri bawah menentukan area pengaruh yang besar, meskipun atau secara langsung karena jauh dari semua titik data lainnya. NNDD dengan Persamaan.8.5 dibidang lainnya di bagi kelas melingkar dengan radius konstan yang ditunjukkan pada Gambar 8.32 di sebelah kiri. Dugaan intuitif ini dikonfirmasi oleh eksperimen praktis.

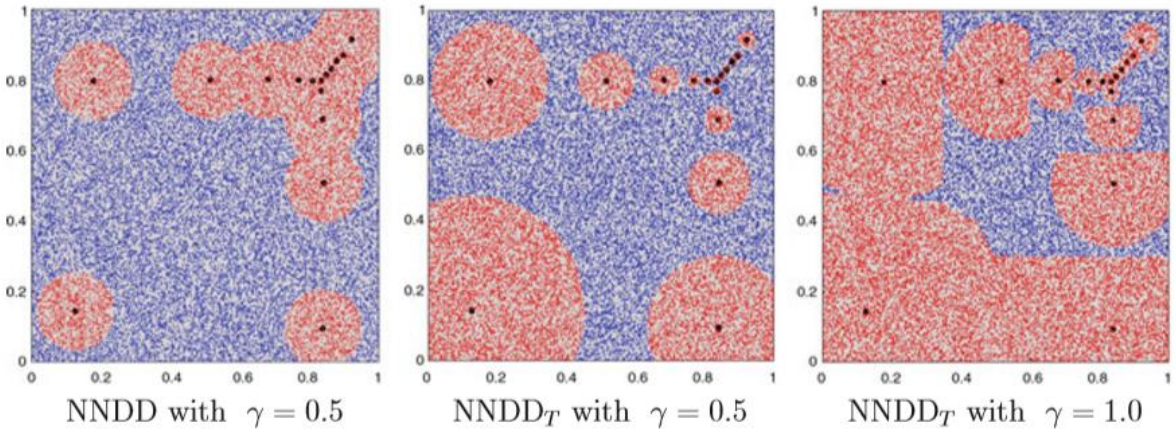
8.17 *CLUSTERING*

Jika kita mencari di mesin pencari untuk istilah “mars”, kita akan mendapatkan hasil seperti “the planet mars” dan “Chocolate, confectionery and beverage conglomerate” yang secara semantik sangat berbeda. Dalam kumpulan dokumen yang ditemukan ada dua kelompok yang sangat berbeda. Google, misalnya, masih mencantumkan hasil dengan cara yang tidak terstruktur. Akan lebih baik jika mesin pencari memisahkan cluster dan menyajikannya kepada pengguna karena biasanya pengguna hanya tertarik pada salah satu cluster.

Perbedaan pengelompokan berbeda dengan pembelajaran yang diawasi adalah bahwa data pelatihan tidak berlabel. Dengan demikian pra-penataan data oleh supervisor tidak ada. Sebaliknya, menemukan struktur adalah inti dari pengelompokan. Dalam ruang

[46] Penskalaan fitur diperlukan atau bermanfaat untuk banyak algoritme pembelajaran mesin.

data pelatihan, akumulasi data seperti pada Gambar 8.33 dapat ditemukan. Dalam sebuah cluster, jarak titik-titik tetangga biasanya lebih kecil daripada jarak antara titik-titik dari cluster yang berbeda. Oleh karena itu, pemilihan metrik jarak yang sesuai untuk titik, yaitu untuk objek yang akan dikelompokkan dan untuk cluster, adalah sangat penting. Seperti sebelumnya, kita asumsikan berikut ini bahwa setiap objek data dideskripsikan oleh vektor atribut numerik.



Gambar 8.32 NNDD dan NNDDT diterapkan pada satu set 16 titik data dua dimensi yang dipilih (titik hitam).

Dalam setiap kasus, keanggotaan kelas telah ditentukan untuk 10.000 poin yang dipilih secara acak. Poin yang ditandai dengan warna merah diklasifikasikan sebagai positif, yaitu ditetapkan kelas set pelatihan, sedangkan poin biru diklasifikasikan sebagai negatif.

Gambar 8.33 Contoh sederhana dua dimensi dengan empat cluster yang terpisah dengan jelas

8.18 METRIK JARAK

Sesuai untuk setiap aplikasi, berbagai metrik jarak didefinisikan untuk jarak d antara dua vektor x dan y dalam n. Yang paling umum adalah jarak Euclidean

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Agak lebih sederhana adalah jumlah jarak kuadrat

$$d_q(x,y) = \sum_{i=1}^{n}(x_i - y_i)^2$$

yang, untuk algoritma yang hanya membandingkan jarak, setara dengan jarak Euclidean (Latihan 8.2). Juga digunakan adalah jarak Manhattan yang disebutkan di atas

$$d_m(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

serta jarak komponen maksimum

$$d_\infty(x,y) = \max_{i=1,\ldots,n} |x_i - y_i|$$

yang didasarkan pada norma maksimum. Selama klasifikasi teks, proyeksi ternormalisasi dari dua vektor satu sama lain, yaitu, produk skalar ternormalisasi

$$\frac{xy}{|x||y|}$$

sering dihitung, di mana |x| adalah norma Euclidian dari x. Karena rumus ini adalah metrik untuk kesamaan dua vektor, sebagai metrik jarak kebalikannya

$$d_s(x,y) = \frac{|x||y|}{xy}$$

dapat digunakan, atau ">" dan "<" dapat ditukar untuk semua perbandingan. Dalam pencarian teks, atribut $x_1,\ldots,x_n$ dihitung mirip dengan naive bayes sebagai komponen vektor x sebagai berikut. Untuk kamus dengan 50.000 kata, nilai xi sama dengan frekuensi kata kamus ke-i dalam teks. Karena biasanya hampir semua komponen adalah nol dalam vektor seperti itu, selama perhitungan produk skalar, hampir semua suku penjumlahannya adalah nol. Dengan memanfaatkan informasi semacam ini, implementasi dapat dipercepat secara signifikan (Latihan 8.21).

8.19 K – MEANS DAN ALGORITMA EM

Kapanpun jumlah cluster sudah diketahui sebelumnya, algoritma k-means dapat digunakan. Seperti namanya, k cluster ditentukan oleh nilai rata-ratanya. Pertama k cluster titik tengah $\boldsymbol{\mu_1, \ldots, \mu_k}$ diinisialisasi ke koordinat k titik data yang dipilih secara acak atau manual. (Catatan: Pemilihan titik arbitrer (yang bukan titik data) sebagai pusat klaster dapat menyebabkan klaster kosong.) Kemudian dua langkah berikut dilakukan berulang kali:

- Klasifikasi semua data ke titik tengah klaster terdekat
- Perhitungan ulang titik tengah klaster

Hasil skema berikut sebagai algoritma:

K-MEANS($\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, k$)
initialize cluster centers $\boldsymbol{\mu}_1 = \boldsymbol{x}_{i_1}, \ldots, \boldsymbol{\mu}_k = \boldsymbol{x}_{i_k}$ (e.g. randomly)
Repeat
  classify $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ to each's nearest $\boldsymbol{\mu}_i$
  recalculate $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
Until no change in $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
Return($\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$)

Perhitungan titik tengah cluster l untuk titik $x_1, \ldots, x_l$ dilakukan dengan

$$\mu = \frac{1}{l}\sum_{i=1}^{1} x_i$$

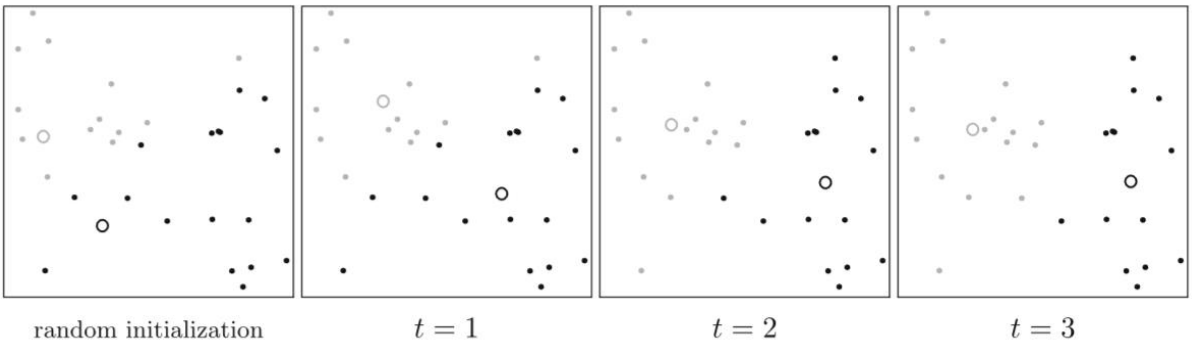
Eksekusi pada contoh ditunjukkan pada Gambar 8.34 untuk kasus dua kelas. Kita melihat bagaimana setelah tiga iterasi, pusat kelas, yang pertama kali dipilih secara acak, menjadi

stabil. Sementara algoritma ini tidak menjamin konvergensi, biasanya konvergen sangat cepat. Ini berarti bahwa jumlah langkah iterasi biasanya jauh lebih kecil daripada jumlah titik data. Kompleksitasnya adalah O(*ndkt*), di mana n adalah jumlah total titik, d dimensi ruang fitur, dan t jumlah langkah iterasi.

Dalam banyak kasus, keharusan memberikan jumlah kelas di muka menimbulkan batasan yang tidak nyaman. Oleh karena itu, selanjutnya kami akan memperkenalkan algoritma yang lebih fleksibel. Namun, sebelum itu, kami akan menyebutkan algoritma EM, yang merupakan varian kontinu dari k-means, karena tidak membuat penetapan pasti dari data ke kelas, melainkan, untuk setiap titik mengembalikan probabilitas itu milik kelas. berbagai kelas. Di sini kita harus mengasumsikan bahwa jenis distribusi probabilitas diketahui. Sering digunakan distribusi normal. Tugas dari algoritma EM adalah untuk menentukan parameter (mean μi dan matriks kovarians ∑i dari distribusi normal multi-dimensi k) untuk setiap cluster. Sama halnya dengan k-means, dua langkah berikut dijalankan berulang kali:

Harapan: Untuk setiap titik data, probabilitas $P(C_j|x_i)$ yang dimiliki setiap klaster dihitung.

Maksimalisasi: Menggunakan probabilitas yang baru dihitung, parameter distribusi dihitung ulang.



Gambar 8.34 k-means dengan dua kelas (k = 2) diterapkan pada 30 titik data. Paling kiri adalah dataset dengan pusat awal, dan ke kanan adalah cluster setelah setiap iterasi. Setelah tiga iterasi, konvergensi tercapai

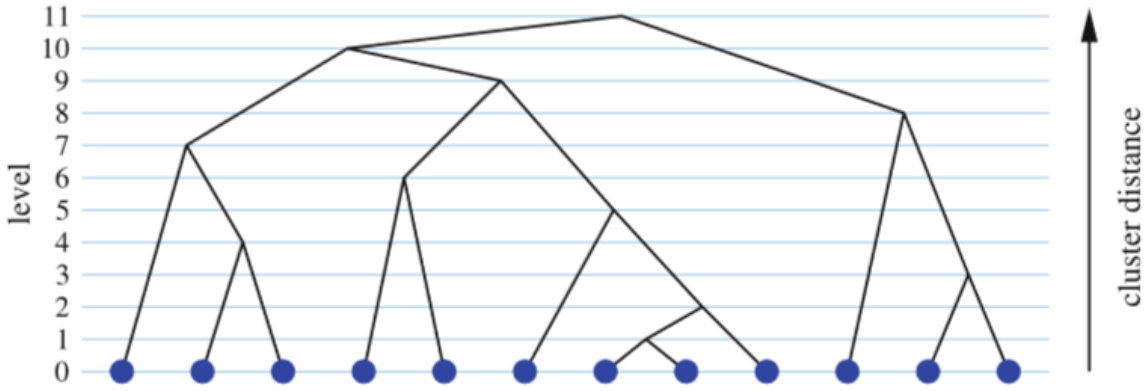
Dengan demikian pengelompokan yang lebih lembut tercapai, yang dalam banyak kasus mengarah pada hasil yang lebih baik. Pergantian antara harapan dan maksimalisasi ini memberi nama algoritma. Selain pengelompokan, misalnya, algoritma EM digunakan untuk mempelajari jaringan Bayesian [DHS01].

8.20 PENGELOMPOKAN HIRARKIS

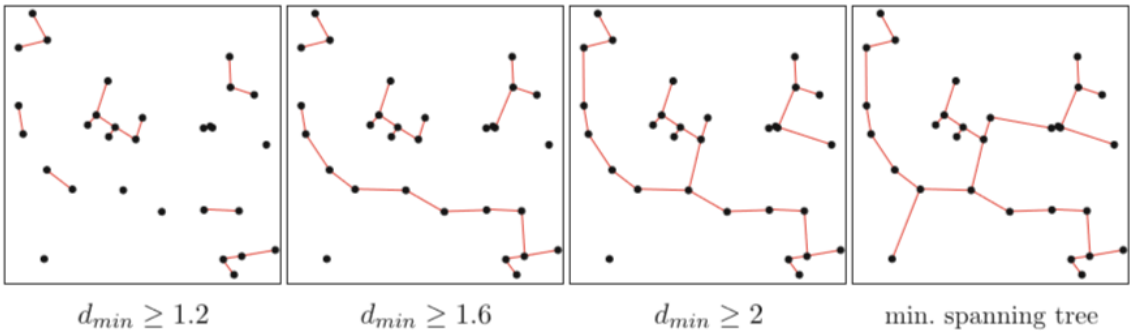
Dalam *clustering* hirarkis kita mulai dengan n cluster yang masing-masing terdiri dari satu titik. Kemudian cluster tetangga terdekat digabungkan sampai semua titik telah digabungkan menjadi satu cluster, atau sampai kriteria terminasi tercapai. Kami mendapatkan skema

HIERARCHICALCLUSTERING$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, k)$
initialize $C_1 = \{\boldsymbol{x}_1\}, \ldots, C_n = \{\boldsymbol{x}_n\}$
Repeat
  Find two clusters C_i and C_j with the smallest distance
  Combine C_i and C_j
Until Termination condition reached
Return(tree with clusters)

Kondisi terminasi dapat dipilih sebagai, misalnya, jumlah cluster yang diinginkan atau jarak maksimum antar cluster. Pada Gambar 8.35 algoritma ini direpresentasikan secara skematis sebagai pohon biner, di mana dari bawah ke atas di setiap langkah, yaitu, pada setiap tingkat, dua subpohon terhubung. Di tingkat atas semua titik disatukan menjadi satu kelompok besar.



Gambar 8.35 Pada hirarki *clustering*, dua cluster dengan jarak terkecil digabungkan pada setiap langkah



Gambar 8.36 Algoritma tetangga terdekat diterapkan pada data dari Gambar 8.34 pada tingkat yang berbeda dengan 12, 6, 3, 1 cluster

Sejauh ini tidak jelas bagaimana jarak antara cluster dihitung. Memang, di bagian sebelumnya kami mendefinisikan berbagai metrik jarak untuk titik, tetapi ini tidak dapat

digunakan pada cluster. Metrik yang mudah digunakan dan sering digunakan adalah jarak antara dua titik terdekat dalam dua cluster C_i dan C_j:

$$d_{min}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$

Jadi kita memperoleh algoritma tetangga terdekat, yang penerapannya ditunjukkan pada Gambar. 8.36. [47] Kita melihat bahwa algoritma ini menghasilkan pohon merentang minimum. [48] Contoh lebih lanjut menunjukkan bahwa dua algoritma yang dijelaskan menghasilkan cluster yang cukup berbeda. Ini menunjukkan bahwa untuk graf dengan cluster yang tidak dipisahkan secara jelas, hasilnya sangat tergantung pada algoritma atau metrik jarak yang dipilih.

Untuk implementasi yang efisien dari algoritma ini, pertama-tama kita membuat matriks ketetanggaan di mana jarak antara semua titik disimpan, yang membutuhkan waktu dan memori $O(n^2)$. Jika jumlah cluster tidak memiliki batas atas, loop akan melakukan iterasi n - 1 kali dan waktu komputasi asimtotik menjadi $O(n^3)$. Untuk menghitung jarak antara dua cluster, kita juga dapat menggunakan jarak antara dua titik terjauh

$$d_{max}(C_i, C_j) = \underset{x \in C_i, y \in C_j}{mac} d(x, y)$$

dan mendapatkan algoritma tetangga terjauh. Sebagai alternatif, jarak titik tengah cluster $d_\mu (C_i, C_j) = d(\mu_i, \mu_j)$ digunakan.

Bagaimana Jumlah Cluster Ditentukan?

Dalam semua algoritma pengelompokan yang dibahas sejauh ini, pengguna harus menentukan jumlah cluster. Dalam banyak kasus, pengguna tidak memiliki gagasan tentang jumlah cluster yang masuk akal, tetapi mereka hanya ingin partisi yang "baik", yaitu pemisahan titik data mereka ke dalam cluster.

Sebelum kita membahas kriteria ini, pertama-tama mari kita tunjukkan bagaimana hal itu dapat diterapkan. Asumsikan bahwa kita memiliki program yang secara kombinatorial dapat menghitung semua partisi yang mungkin untuk satu set n titik data. Kemudian kita cukup menerapkan kriteria lebar siluet untuk setiap partisi dan menggunakan maksimum untuk menentukan partisi terbaik. Tetapi karena jumlah partisi bertambah dengan cepat dengan jumlah titik data, metode ini tidak praktis. Kita bisa, bagaimanapun, hanya menerapkan salah satu algoritma pengelompokan yang disajikan (misalnya k-means), biarkan berjalan untuk k dari 1 hingga n, dan kemudian gunakan kriteria lebar siluet untuk menentukan k terbaik dan partisinya masing-masing.

Apa yang masih kita lewatkan adalah kriteria yang mengukur kualitas partisi. Idenya, secara kasar, adalah sebagai berikut: jarak rata-rata antara dua titik arbitrer dalam cluster yang sama harus lebih kecil dari jarak antara dua titik arbitrer yang berada di cluster tetangga yang berbeda. Rasio jarak rata-rata antara titik-titik dalam cluster tetangga dan jarak rata-rata antara titik-titik dalam cluster harus dimaksimalkan.

Biarkan titik data $(x_1, \ldots, x_n)$ diberikan, serta fungsi pengelompokan

$$C : x_1 \rightarrow c(x_i)$$

yang menetapkan setiap titik ke sebuah cluster. Misal d(i, l) menjadi jarak rata-rata dari x_i ke semua titik ($\neq x_i$) dalam gugus l. Maka a(i) = d(i, c, (x_i)) adalah jarak rata-rata dari xi ke semua titik lain dalam clusternya sendiri. Jika xi adalah satu-satunya titik dalam cluster, kami menetapkan a(i) = 0.

$$b(i) = \min_{j \neq c(xi)} \{d(i, j\}$$

[47] Algoritma tetangga terdekat tidak menjadi bingung dengan metode tetangga terdekat untuk klasifikasi.

[48] Pohon merentang minimum adalah graf tak berarah asiklik dengan jumlah panjang rusuk minimum.

adalah jarak rata-rata terkecil dari titik x_i ke sebuah cluster yang xi tidak termasuk. Kami sekarang mendefinisikan fungsi

$$s(i) = \begin{cases} 0 & jika\ a(i) = 0 \\ \dfrac{b(i) = a(i)}{\max\{a(i), b(i)\}} & Lainnya \end{cases}$$

yang mengukur seberapa baik titik xi cocok ke dalam sebuah cluster. Maka kita memiliki 1 ≤s(i) ≤ 1 dan

$$s(i) = \begin{cases} 1 & \text{jika xi berada di tengah} - \text{tengah cluster yang digambarkan dengan jelas.} \\ 0 & \text{jika xi terletak di perbatasan dua cluster.} \\ -1 & \text{jika xi berada pada cluster “salah”} \end{cases}$$

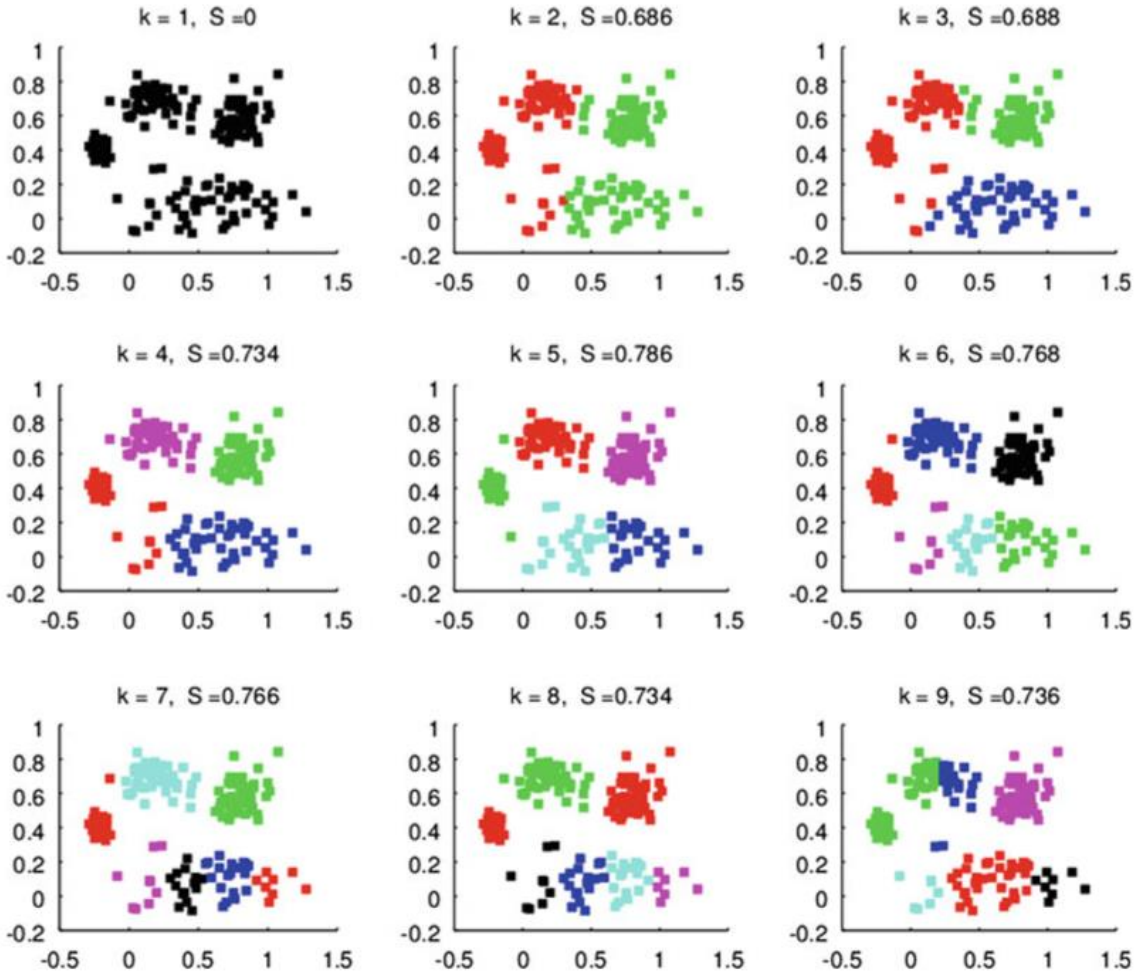
Mencerminkan ide yang dirumuskan di atas, sekarang kita mencari partisi yang memaksimalkan mean

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

s(i) pada semua titik, yang disebut kriteria lebar siluet. Untuk pengelompokan k-means, ini dapat dilakukan dengan algoritma OMRk yang disajikan:

OMR$_K$ ($\varkappa_1$, ..., $\varkappa_n$, p, k_{max}.)
S* = - ∞
    For i = 1 **To** p
    Hasilkan partisi acak dengan k cluster (inisialisasi)
    Dapatkan partisi P dengan k-means
    Tentukan S untuk P
    If S > S* **Then**
        S* = S; k* = k; P* = P
Return (k*, P*)

Algoritma ini berulang kali menerapkan k-means untuk nilai k yang berbeda. Karena hasil dari k-means sangat bergantung pada inisialisasinya, untuk setiap k, p inisialisasi acak yang berbeda dicoba di loop dalam, dan kemudian fungsi mengembalikan k* optimal dan partisi terbaik yang sesuai P*. Algoritma OMRk juga dapat digunakan dengan algoritma *clustering* lainnya seperti algoritma EM dan hirarki *clustering*.



Gambar 8.37 Hasil algoritma OMRk untuk k = 2 sampai 9. Nilai S = 0,786 terbaik ditemukan dengan k = 5

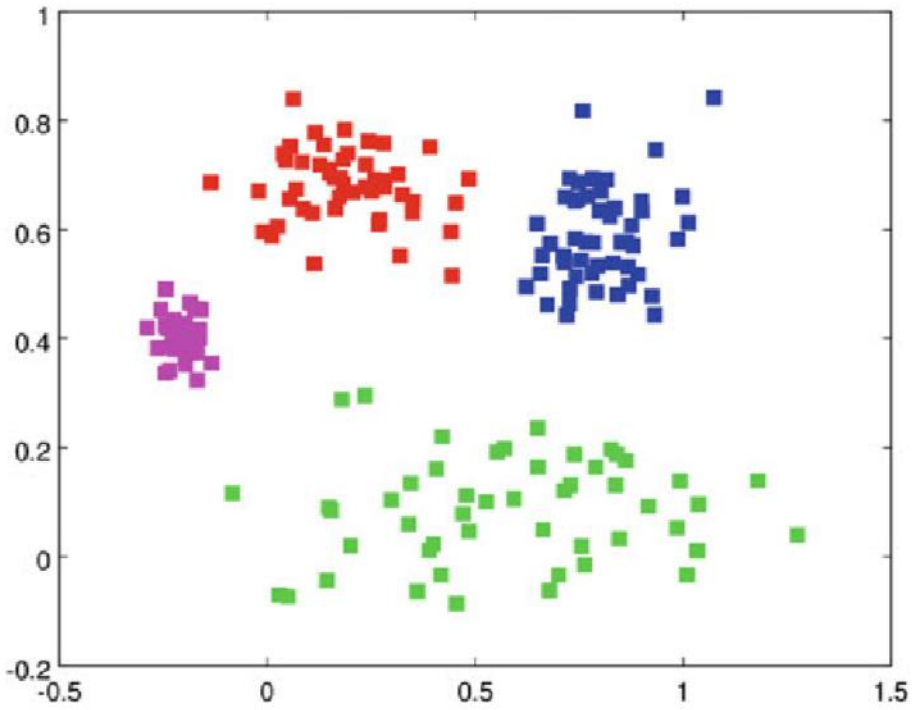
Contoh 4

Diagram kiri atas pada Gambar 8.37 menunjukkan satu set titik data dua dimensi dengan empat cluster yang jelas. Algoritma OMRk dijalankan pada data ini dengan p = 30 dan k_{max} = 9. Dalam delapan diagram berikut, gambar menunjukkan partisi terbaik bersama dengan kualitasnya S untuk setiap k. Algoritme menemukan nilai maksimum S = 0:786 pada k = 5. Ini tidak mencerminkan pengelompokan titik-titik yang alami (untuk mata manusia) menjadi empat kelompok. Dalam partisi yang ditemukan untuk k = 4, beberapa titik yang seharusnya menjadi bagian dari cluster biru ditugaskan ke cluster merah. Hal ini karena k-means meminimalkan jarak ke titik pusat cluster, dan titik-titik lebih dekat ke pusat cluster merah. Kepadatan titik yang lebih tinggi dalam klaster merah tidak diperhitungkan.

Algoritma EM, yang dapat memperkirakan perbedaan densitas titik dengan menggunakan distribusi normal, berkinerja jauh lebih baik. Seperti ditunjukkan pada Gambar 8.38, algoritma EM menemukan distribusi alami yang hampir sama persis untuk k = 4.

Akhirnya kami harus menegaskan kembali bahwa semua metode yang telah kami jelaskan hanyalah algoritma pencarian serakah heuristik yang tidak menjelajahi seluruh ruang dari semua partisi. Kriteria lebar siluet yang dijelaskan hanyalah estimasi heuristik dari "kualitas" partisi. Tidak ada ukuran mutlak kualitas untuk partisi karena, seperti yang telah kami tunjukkan bahkan dalam contoh dua dimensi yang sederhana, orang yang berbeda dalam kasus tertentu dapat memilih pengelompokan yang sangat berbeda. Kami

juga harus menyebutkan bahwa ada banyak algoritma pengelompokan menarik lainnya seperti algoritma DBSCAN berbasis kepadatan.



Gambar 8.38 Sebuah partisi yang dihasilkan oleh algoritma EM dengan k = 4

8.21 *DATA MINING* DALAM PRAKTEKNYA

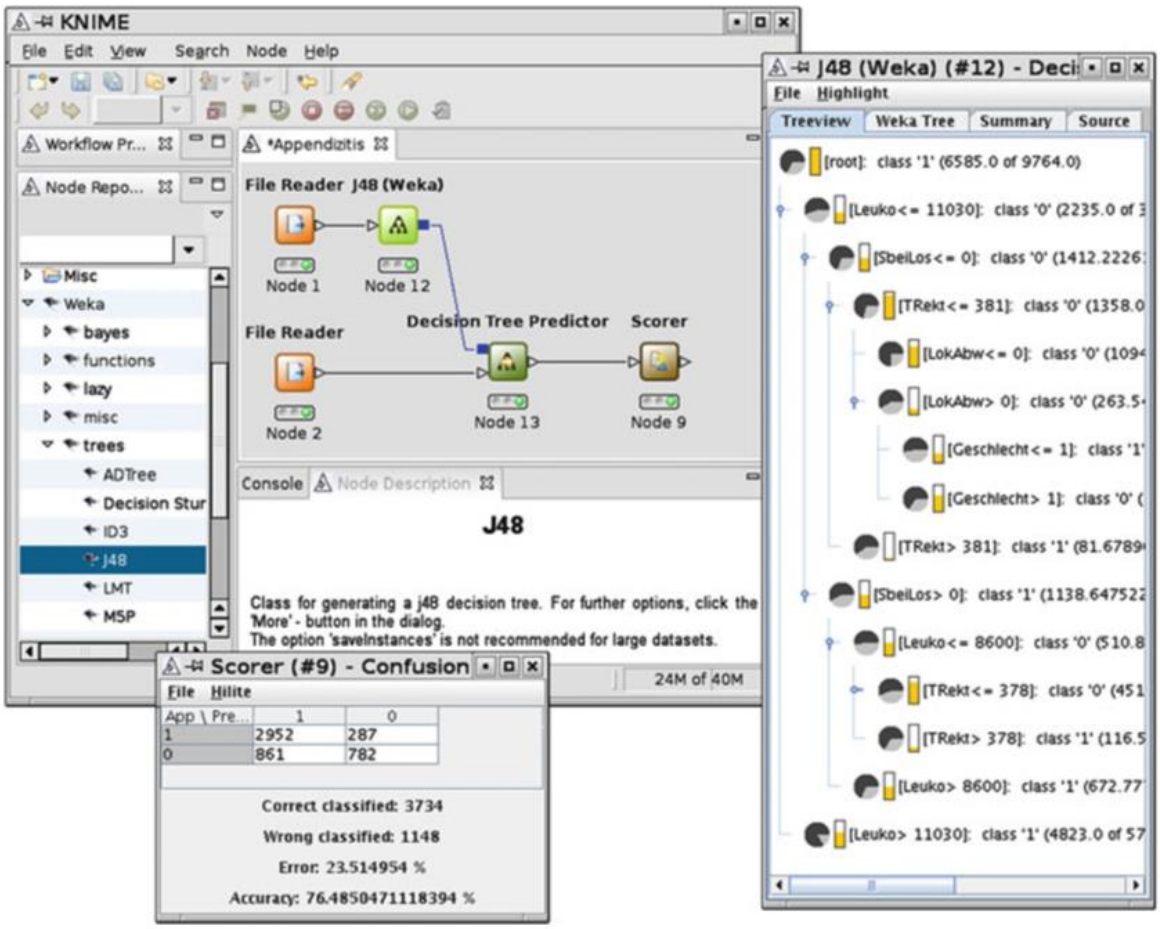
Semua algoritma pembelajaran yang disajikan sejauh ini dapat digunakan sebagai alat untuk *Data mining*. Namun, bagi pengguna terkadang cukup merepotkan untuk membiasakan diri dengan perangkat lunak baru untuk setiap aplikasi dan lebih jauh lagi untuk memasukkan data yang akan dianalisis ke dalam format yang sesuai untuk setiap kasus tertentu.

Sejumlah sistem *data mining* mengatasi masalah ini. Sebagian besar sistem ini menawarkan antarmuka pengguna grafis yang nyaman dengan beragam alat untuk visualisasi data, untuk pemrosesan awal seperti manipulasi nilai yang hilang, dan untuk analisis. Untuk analisis, algoritma pembelajaran yang disajikan di sini digunakan, antara lain. Open-source JavalibraryWEKA yang komprehensif layak mendapatkan perhatian khusus. Ia menawarkan sejumlah besar algoritma dan menyederhanakan pengembangan algoritma baru.

Sistem KNIME yang tersedia secara gratis, yang akan kami perkenalkan secara singkat di bagian berikut, menawarkan antarmuka pengguna yang nyaman dan semua jenis alat yang disebutkan di atas. KNIME juga menggunakan modul WEKA. Selain itu, ia menawarkan cara sederhana untuk mengontrol aliran data dari alat visualisasi, prapemrosesan, dan analisis yang dipilih dengan editor grafis. Sementara itu, sejumlah besar sistem lain menawarkan fungsionalitas yang sangat mirip, seperti proyek sumber terbuka Rapid Miner (www.rapidminer.com), system Clementine (www.spss.com/clementine) sold by SPSS, dan kerangka kerja analitik KXEN (www.kxen.com).

Alat *Data mining* KNIME

Dengan menggunakan data LEXMED, sekarang kami akan menunjukkan cara mengekstrak pengetahuan dari data menggunakan KNIME (Penambang Informasi Konstanz, www.knime.org). Pertama kita membuat pohon keputusan seperti yang ditunjukkan pada Gambar 8.39. Setelah membuat proyek baru, alur kerja dibangun secara grafis. Untuk melakukan ini, alat yang sesuai dikeluarkan dari repositori node dengan mouse dan diseret ke jendela alur kerja utama.



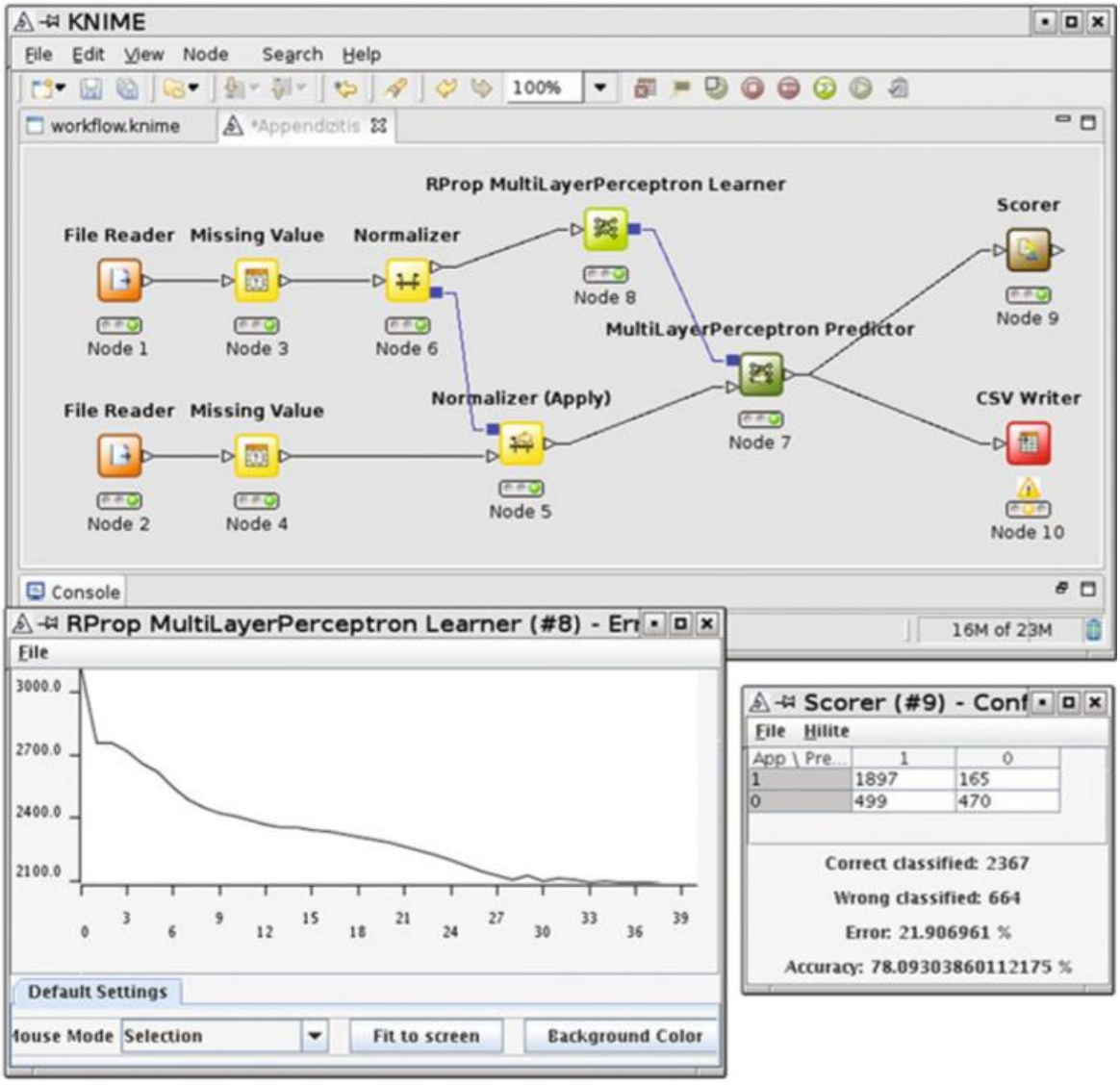
Gambar 8.39 Antarmuka pengguna KNIME dengan dua tampilan tambahan, yang menunjukkan pohon keputusan dan matriks kebingungan

Data pelatihan dan pengujian dari file C4.5 dapat dibaca dengan dua node pembaca file tanpa masalah. Node-node ini juga dapat dengan mudah dikonfigurasi untuk format file lain. Lampu lalu lintas menyamping di bawah node menunjukkan statusnya (belum siap, dikonfigurasi, dieksekusi). Kemudian node J48 dipilih dari perpustakaan WEKA, yang berisi implementasi Java C4.5. Konfigurasi untuk ini cukup sederhana. Sekarang node prediktor dipilih, yang menerapkan pohon yang dihasilkan ke data uji. Ini memasukkan kolom baru ke dalam tabel data uji "Prediksi" dengan klasifikasi yang dihasilkan oleh pohon. Dari sana, simpul pencetak menghitung matriks kebingungan yang ditunjukkan pada gambar, yang memberikan jumlah kasus yang diklasifikasikan dengan benar untuk kedua kelas dalam diagonal, dan sebagai tambahan jumlah titik data positif dan negatif palsu.

Setelah aliran sepenuhnya dibangun dan semua node dikonfigurasi, maka node arbitrer dapat dieksekusi. Ini secara otomatis memastikan bahwa node pendahulu dieksekusi, jika perlu. Node J48 menghasilkan tampilan pohon keputusan, yang ditunjukkan

di sebelah kanan gambar. Pohon ini identik dengan yang dihasilkan oleh C4.5, meskipun di sini simpul TRekt<=378 ditampilkan runtuh.

Sebagai perbandingan, sebuah proyek untuk mempelajari perceptron multilayer ditunjukkan pada Gambar 8.40. Ini bekerja mirip dengan perceptron linier yang diperkenalkan sebelumnya, tetapi juga dapat membagi kelas yang tidak dapat dipisahkan secara linier. Di sini alurnya agak lebih kompleks. Node tambahan diperlukan untuk menangani kesalahan nilai yang hilang saat melakukan prapemrosesan masing-masing dari dua file. Kami mengaturnya sehingga garis-garis itu akan dihapus. Karena jaringan saraf tidak dapat menangani nilai arbitrer, nilai semua variabel diskalakan secara linier ke dalam interval [0, 1] menggunakan node “normalizer”.



Gambar 8.40 Antarmuka pengguna KNIME dengan jendela alur kerja, kurva pembelajaran, dan matriks kebingungan

Setelah menerapkan pembelajar RProp, peningkatan pada propagasi balik, kita dapat menganalisis perkembangan waktu kesalahan aproksimasi dalam kurva pembelajaran yang ditunjukkan. Dalam matriks konfusi, pencetak skor mengeluarkan analisis data uji. Node “CSV Writer” di kanan bawah berfungsi untuk mengekspor file hasil,

yang kemudian dapat digunakan secara eksternal untuk menghasilkan kurva ROC yang ditunjukkan pada Gambar 7.10, yang sayangnya belum ada alat KNIME (belum).

Singkatnya, kita dapat mengatakan yang berikut tentang KNIME (dan alat serupa): untuk proyek dengan persyaratan analisis data yang tidak terlalu eksotis, ada baiknya bekerja dengan meja kerja yang kuat untuk analisis data. Pengguna sudah menghemat banyak waktu dengan tahap preprocessing. Ada banyak pilihan "alat penambangan" yang mudah digunakan, seperti pengklasifikasi tetangga terdekat, algoritma pembelajaran jaringan Bayesian sederhana, serta algoritma pengelompokan k-means (Bagian 8.9.2). Evaluasi hasil, misalnya validasi silang, dapat dilakukan dengan mudah. Perlu disebutkan, bahwa selain yang ditampilkan, ada banyak alat lain untuk visualisasi data. Selain itu, pengembang KNIME telah menyediakan ekstensi untuk KNIME, yang dengannya pengguna dapat memprogram alatnya sendiri dalam Java atau Python.

Namun, juga harus disebutkan bahwa pengguna sistem *Data mining* jenis ini harus membawa pengetahuan awal yang kuat tentang pembelajaran mesin dan penggunaan teknik *Data mining*. Perangkat lunak itu sendiri tidak dapat menganalisis data, tetapi di tangan seorang spesialis, perangkat lunak ini menjadi alat yang ampuh untuk mengekstraksi pengetahuan dari data. Untuk pemula di bidang pembelajaran mesin yang menarik, sistem seperti itu menawarkan kesempatan yang ideal dan sederhana untuk menguji pengetahuan seseorang secara praktis dan untuk membandingkan berbagai algoritme. Pembaca dapat memverifikasi ini dalam Latihan 8.22.

8.22 RINGKASAN

Kami telah membahas beberapa algoritme secara menyeluruh dari bidang pembelajaran terawasi yang mapan, termasuk pembelajaran pohon keputusan, jaringan Bayesian, dan metode tetangga terdekat. Algoritme ini stabil dan dapat digunakan secara efisien dalam berbagai aplikasi dan dengan demikian termasuk dalam repertoar standar dalam AI dan *Data mining*. Hal yang sama berlaku untuk algoritme pengelompokan, yang bekerja tanpa "pengawas" dan dapat ditemukan, misalnya, dalam aplikasi mesin telusur. Pembelajaran penguatan sebagai bidang lain dari pembelajaran mesin juga tidak menggunakan pengawas. Berbeda dengan pembelajaran terawasi, di mana pembelajar menerima tindakan atau jawaban yang benar sebagai label dalam data pelatihan, dalam pembelajaran penguatan hanya sesekali dan kemudian umpan balik positif atau negatif diterima dari lingkungan. Dalam Bab. 10 kami akan menunjukkan cara kerjanya. Tidak sesulit tugas dalam pembelajaran semi-diawasi, sub-bidang pembelajaran mesin yang masih muda, di mana hanya sedikit dari sejumlah besar data pelatihan yang diberi label.

Pembelajaran yang diawasi sekarang menjadi area mapan dengan banyak aplikasi yang berhasil. Untuk pembelajaran data yang diawasi dengan label berkelanjutan, algoritma aproksimasi fungsi apa pun dapat digunakan. Dengan demikian ada banyak algoritma dari berbagai bidang matematika dan ilmu komputer. Dalam Bab. 9 kami akan memperkenalkan berbagai jenis jaringan saraf, algoritma kuadrat terkecil dan mesin vektor pendukung, yang semuanya merupakan aproksimator fungsi. Saat ini, proses Gaussian sangat populer karena sangat universal dan mudah diterapkan serta memberikan perkiraan ketidakpastian nilai output kepada pengguna.

Taksonomi berikut memberikan gambaran tentang algoritma pembelajaran yang paling penting dan klasifikasinya.

Pembelajaran terawasi

- ***Pembelajaran malas***
    - k metode tetangga terdekat (klasifikasi + aproksimasi)

    - Regresi berbobot lokal (perkiraan)
    - Pembelajaran berbasis kasus (klasifikasi + aproksimasi)
- ***Eager learning – Induksi pohon keputusan (klasifikasi)***
    - Pembelajaran jaringan Bayesian (klasifikasi + aproksimasi)
    - Jaringan syaraf tiruan (klasifikasi + aproksimasi)
    - Proses Gaussian (klasifikasi + aproksimasi)
    - Mendukung mesin vektor
    - Wavelet, splines, fungsi basis radial, …

Pembelajaran tanpa pengawasan (clustering)

- Algoritma tetangga terdekat
- Algoritma tetangga terjauh
- k-means
- Jaringan saraf

Pembelajaran penguatan

- Iterasi nilai
- Pembelajaran Q
- Pembelajaran TD
- Metode gradien kebijakan
- Jaringan saraf

Apa yang telah dikatakan tentang pembelajaran terawasi hanya benar, bagaimanapun, ketika bekerja dengan seperangkat atribut yang diketahui. Bidang yang menarik dan masih terbuka di bawah penelitian intensif adalah pemilihan fitur otomatis. Untuk pembelajaran dengan pohon keputusan, kami menyajikan algoritma untuk penghitungan perolehan informasi atribut yang mengurutkan atribut menurut relevansinya dan hanya menggunakan atribut yang meningkatkan kualitas klasifikasi. Dengan jenis metode ini dimungkinkan untuk secara otomatis memilih atribut yang relevan dari kumpulan basis yang berpotensi besar. Set dasar ini, bagaimanapun, harus dipilih secara manual.

Masih terbuka dan juga tidak didefinisikan dengan jelas adalah pertanyaan tentang bagaimana mesin dapat menemukan atribut baru. Mari kita bayangkan sebuah robot yang seharusnya memetik apel. Untuk ini ia harus belajar membedakan antara apel matang dan mentah dan benda-benda lainnya. Secara tradisional kami akan menentukan atribut tertentu seperti warna dan bentuk wilayah piksel dan kemudian melatih algoritma pembelajaran menggunakan gambar yang diklasifikasikan secara manual. Mungkin juga bahwa misalnya jaringan saraf dapat dilatih secara langsung dengan semua piksel gambar sebagai input, yang untuk resolusi tinggi terkait dengan masalah waktu komputasi yang parah. Pendekatan yang secara otomatis membuat saran untuk fitur yang relevan akan diinginkan di sini.

Clustering menyediakan satu pendekatan untuk pemilihan fitur. Sebelum melatih mesin pengenalan apel, kami membiarkan pengelompokan berjalan pada data. Untuk pembelajaran (diawasi) kelas apple dan non apple, inputnya tidak lagi semua piksel, melainkan hanya kelas yang ditemukan selama pengelompokan, berpotensi bersama dengan atribut lainnya. Pengelompokan pada tingkat apa pun dapat digunakan untuk "penemuan" fitur yang otomatis dan kreatif. Namun, tidak pasti apakah fitur yang ditemukan relevan. Algoritme pembelajaran mendalam yang relatif muda namun sukses, yang menggabungkan jaringan saraf kompleks yang besar dengan pra-pemrosesan tanpa pengawasan, merupakan terobosan dalam pembuatan fitur, dan oleh karena itu merupakan tonggak sejarah dalam AI. Kami akan membahas topik ini.

Masalah berikut ini lebih sulit lagi: asumsikan bahwa kamera video yang digunakan untuk pengenalan apel hanya mentransmisikan gambar hitam putih. Tugas tidak lagi dapat diselesaikan dengan baik. Alangkah baiknya jika mesin berkreasi sendiri, misalnya dengan menyarankan agar kamera diganti dengan kamera berwarna. Ini akan meminta terlalu banyak hari ini.

Selain karya khusus tentang semua subbidang pembelajaran mesin. Di atas segalanya, saya merekomendasikan buku luar biasa karya Peter Flach , yang membuka jalan menuju pemahaman mendalam tentang konsep dan algoritme pembelajaran mesin dengan penjelasan dan contoh yang sangat baik. Untuk hasil penelitian saat ini, lihat Jurnal Penelitian Pembelajaran Mesin yang tersedia secara gratis (http://jmlr.csail.mit.edu), Jurnal Pembelajaran Mesin, serta prosiding Konferensi Internasional tentang Pembelajaran Mesin (ICML) direkomendasikan. Untuk setiap pengembang algoritme pembelajaran, Repositori Pembelajaran Mesin dari University of California di Irvine (UCI) menarik, dengan banyak koleksi data pelatihan dan pengujian untuk algoritme pembelajaran dan alat *Data mining*. MLOSS, yang merupakan singkatan dari perangkat lunak sumber terbuka pembelajaran mesin, adalah direktori tautan yang sangat baik ke perangkat lunak yang tersedia secara bebas (www.mloss.org).

8.23 LATIHAN

Pengantar

Latihan 8.1

(a) Tentukan tugas agen yang harus memprediksi cuaca untuk hari berikutnya dengan nilai terukur untuk suhu, tekanan udara, dan kelembaban. Cuaca harus dikategorikan ke dalam salah satu dari tiga kelas: cerah, berawan, dan hujan.

(b) Jelaskan struktur file dengan data pelatihan untuk agen ini.

Latihan 8.2

Tunjukkan bahwa matriks korelasi simetris dan semua elemen diagonalnya sama dengan 1.

Perceptron

Latihan 8.3

Menerapkan aturan pembelajaran perceptron ke himpunan

M+ = {(0, 1.8), (2, 0.6)} dan M_ = {(-11.2, 1.4), (0.4, -1)}

dari Contoh 8.2 berikan hasil dari nilai vektor bobot.

Latihan 8.4

Diberikan tabel di sebelah kanan dengan data pelatihan:

Num.	x_1	x_2	Class
1	6	1	0
2	7	3	0
3	8	2	0
4	9	0	0
5	8	4	1
6	8	6	1
7	9	2	1
8	9	5	1

(a) Dengan menggunakan grafik, tunjukkan bahwa data tersebut dapat dipisahkan secara linier.

(b) Secara manual menentukan bobot w1 dan w2, serta ambang H dari perceptron (dengan ambang batas) yang mengklasifikasikan data dengan benar.

(c) Programkan aturan pembelajaran perceptron dan terapkan program Anda ke tabel. Bandingkan bobot yang ditemukan dengan bobot yang dihitung secara manual.

Latihan 8.5

(a) Berikan interpretasi visual dari inisialisasi heuristik

$$w_0 = \sum_{x_i \in M_+} x_i - \sum_{x_i \in M_-} x_i$$

(b) Berikan contoh kumpulan data yang dapat dipisahkan secara linier di mana heuristik ini tidak menghasilkan garis pemisah.

Metode Tetangga Terdekat

Latihan 8.6

(a) Tunjukkan diagram Voronoi untuk himpunan titik-titik yang bertetangga.

(b) Kemudian gambarlah garis pembagian kelas.

Latihan 8.7

Diberikan tabel dengan data pelatihan dari Latihan 8.4. Berikut ini, gunakan jarak Manhattan d(a, b), yang didefinisikan sebagai d(a, b) $|a_1 - b_1| + |a_2 - b_2|$, untuk menentukan jarak d antara dua titik data a = (a_1, a_2) dan b = (b_1, b_2).

(a) Klasifikasikan vektor v = (8, 3.5) dengan metode tetangga terdekat.

(b) Klasifikasikan vektor v = (8, 3.5) dengan metode k tetangga terdekat untuk k =2,3,5.

Latihan 8.8

(a) Tunjukkan bahwa dalam ruang fitur dua dimensi masuk akal, seperti yang diklaim pada (8.3), untuk menimbang k tetangga terdekat dengan kebalikan dari jarak kuadrat.

(b) Mengapa pembobotan menggunakan $w'_i = 1/1+ ad(x, x_i)$ kurang masuk akal?

Latihan 8.9

(a) Tulislah program yang mengimplementasikan Metode-Nearest-Neighbor-Metode untuk klasifikasi.

(b) Terapkan program ini untuk k = 1 (yaitu satu tetangga terdekat) ke Lexmed-data dari http://www.hs-weingarten.de/*ertel/kibuch/uebungen/. Gunakan untuk melatih file app1.data dan untuk menguji file app1.test. Jangan lupa untuk menormalkan semua fitur sebelum pelatihan.

(c) Terapkan validasi silang leave-one-out pada seluruh kumpulan data app1.data [app1.test untuk menentukan jumlah optimal k tetangga terdekat dan menentukan kesalahan klasifikasi.

(d) Ulangi validasi silang dengan data yang tidak dinormalisasi.

(e) Bandingkan hasil Anda dengan yang diberikan pada Gambar 9.14 untuk metode kuadrat terkecil dan RProp.

Pohon Keputusan

Latihan 8.10

Tunjukkan bahwa definisi $0\log_2 0 := 0$ masuk akal, dengan kata lain, bahwa fungsi $f(x) = x \log_2 x$ dengan demikian menjadi kontinu di titik asal.

Latihan 8.11

Tentukan entropi untuk distribusi berikut.

a. (1, 0, 0, 0, 0,)
b. $\left(\frac{1}{2}, \frac{1}{2}, 0, 0, 0\right)$
c. $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, 0, 0\right)$
d. $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}\right)$
e. $\left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right)$
f. $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \ldots\right)$

Latihan 8.12

(a) Tunjukkan bahwa dua definisi entropi yang berbeda dari (Persamaan 7.9) dan Definisi 8.4 hanya berbeda oleh faktor konstan, yaitu bahwa

$$\sum_{i=1}^{n} p_i \log_2 p_i = c \sum_{i=1}^{n} p_i \, dalam \, p_i$$

dan berikan konstanta c.

(b) Tunjukkan bahwa untuk metode MaxEnt dan untuk pohon keputusan tidak ada bedanya yang mana dari dua rumus yang kita gunakan.

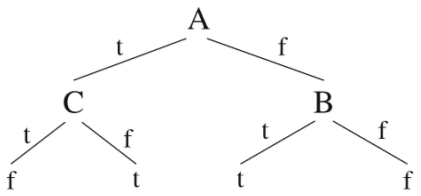
Latihan 8.13

Kembangkan pohon keputusan untuk dataset D dari Latihan 8.4

(a) Perlakukan kedua atribut sebagai diskrit.
(b) Sekarang perlakukan atribut x2 sebagai kontinu dan x1 sebagai diskrit.
(c) Minta C4.5 menghasilkan pohon dengan kedua varian. Gunakan -m 1 -t 10 sebagai parameter untuk mendapatkan saran yang berbeda.

Latihan 8.14

Diberikan pohon keputusan dan tabel berikut untuk data pelatihan dan pengujian:



Training data

A	B	C	Class
t	t	f	t
t	f	f	t
t	t	t	f
f	f	t	f
f	f	f	f
f	t	t	t

Test data

A	B	C	Class
t	t	f	t
t	f	f	t
f	t	f	f
f	f	t	f

(a) Berikan kebenaran pohon untuk data pelatihan dan pengujian.
(b) Berikan rumus logika proposisi yang ekuivalen dengan pohon.
(c) Lakukan pemangkasan pada pohon, gambar pohon yang dihasilkan, dan berikan kebenarannya untuk data latih dan uji.

Latihan 8.15

(a) Saat menentukan atribut saat ini, algoritme untuk menghasilkan pohon keputusan (Gambar 8.26) tidak menghilangkan atribut yang telah digunakan lebih lanjut di

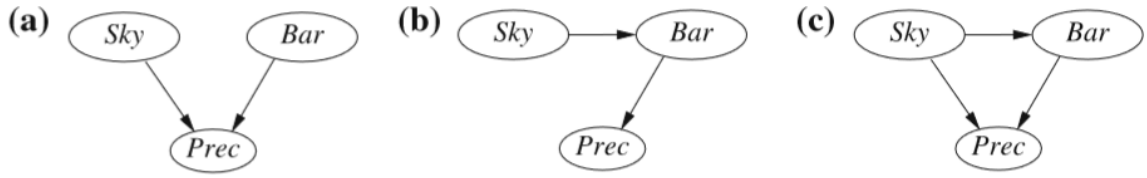
pohon. Meskipun demikian, atribut diskrit terjadi di jalur paling banyak satu kali. Mengapa?

(b) Mengapa atribut kontinu dapat muncul beberapa kali?

Pembelajaran Jaringan Bayesian

Latihan 8.16

Gunakan distribusi yang diberikan dalam Latihan 7.3 dan tentukan CPT untuk tiga jaringan Bayesian:



(d) Tentukan distribusi untuk dua jaringan dari (a) dan (b) dan bandingkan dengan distribusi aslinya. Jaringan mana yang "lebih baik"?

(e) Sekarang tentukan distribusi untuk jaringan (c). Apa yang terjadi pada Anda? Jelaskan!

Latihan 8.17

Tunjukkan bahwa untuk variabel biner $S_1,\ldots,S_n$ dan variabel kelas biner K, skor linier berbentuk

$$keputusan\begin{cases}positif & jika\ w_1S_1 + \cdots + w_nS_n > \Theta \\ negatif & lainnya\end{cases}$$

sama-sama ekspresif dalam kaitannya dengan perceptron dan pengklasifikasi Bayes naif, yang keduanya memutuskan menurut rumus

$$keputusan\begin{cases}positif & jika\ P(K|S_1 \ldots S_n > \frac{1}{2} \\ negatif & lainnya\end{cases}$$

Latihan 8.18

Dalam penerapan klasifikasi teks dengan naive Bayes, aliran bawah eksponen dapat terjadi dengan cepat karena faktor $P(w_i|K)$ (yang muncul pada (8.10) biasanya semuanya sangat kecil, yang dapat menghasilkan hasil yang sangat kecil . Bagaimana kita bisa mengurangi masalah ini?

Latihan 8.19

Tulislah program untuk analisis teks naive Bayes. Kemudian latih dan uji pada benchmark teks menggunakan alat pilihan Anda. Menghitung frekuensi kata dalam teks dapat dilakukan dengan mudah di Linux dengan perintah

cat <datei> | tr -d“[:punct:]” | tr -s“[:space:]”“ \n” | sort | uniq -ci

Dapatkan data Twenty Newsgroups oleh Tom Mitchell dalam koleksi benchmark pembelajaran mesin UCI (Machine *Learning* Repository) [DNM98]. Di sana Anda juga akan menemukan referensi ke program naif Bayes untuk klasifikasi teks oleh Mitchell.

Kekelompokan

Latihan 8.20

Tunjukkan bahwa untuk algoritme yang hanya membandingkan jarak, menerapkan fungsi f yang meningkat secara monoton secara ketat ke jarak tidak ada bedanya. Dengan kata lain, Anda harus menunjukkan bahwa jarak $d_1(x, y)$ dan jarak $d_2(x, y) := f(d_1(x,y))$ menghasilkan hasil yang sama terhadap relasi pemesanan.

Latihan 8.21

Tentukan jarak d_s (perkalian skalar) teks-teks berikut satu sama lain.

x_1: Kami akan memperkenalkan penerapan naive Bayes pada analisis teks pada contoh teks singkat oleh Alan Turing.

x_2: Kita mungkin berharap bahwa mesin pada akhirnya akan bersaing dengan manusia di semua bidang intelektual murni. Tapi mana yang terbaik untuk memulai?

x_3: Sekali lagi saya tidak tahu apa jawaban yang benar, tapi saya pikir kedua pendekatan itu harus dicoba.

Data mining

Latihan 8.22

Menggunakan KNIME (www.knime.de)

(a) Memuat file contoh dengan data Iris dari direktori KNIME dan bereksperimen dengan berbagai representasi data, terutama dengan diagram scatterplot.

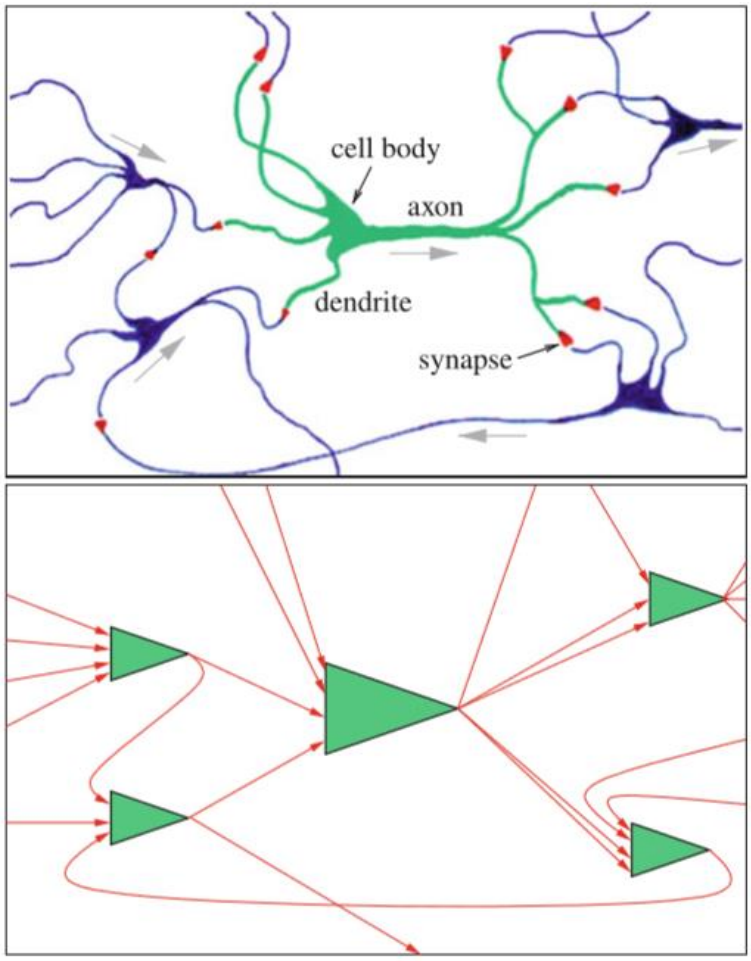
(b) Pertama latih pohon keputusan untuk tiga kelas, dan kemudian latih jaringan RPop.

(c) Muat data apendisitis di situs web buku ini. Bandingkan kualitas klasifikasi metode k tetangga terdekat dengan jaringan RPop. Optimalkan k serta jumlah neuron tersembunyi dari jaringan RPop.

(d) Dapatkan kumpulan data pilihan Anda dari pengumpulan data UCI untuk *Data mining* di http://kdd.ics.uci.edu atau untuk pembelajaran mesin di http://mlearn.ics.uci.edu/MLRepository.html dan bereksperimen dengannya.

BAGIAN 2
BAB 9
NEURAL NETWORK (JARINGAN SYARAF)

Jaringan saraf adalah jaringan sel saraf di otak manusia dan hewan. Otak manusia memiliki sekitar 100 miliar sel saraf. Kita manusia berutang kecerdasan dan kemampuan kita untuk mempelajari berbagai kemampuan motorik dan intelektual pada relai dan adaptasi otak yang kompleks. Selama berabad-abad para ahli biologi, psikolog, dan dokter telah mencoba memahami bagaimana fungsi otak. Sekitar tahun 1900 muncul kesadaran revolusioner bahwa blok bangunan fisik kecil dari otak, sel-sel saraf dan koneksinya, bertanggung jawab atas kesadaran, asosiasi, pikiran, kesadaran, dan kemampuan untuk belajar.



Gambar 9.1 Dua tahap pemodelan jaringan saraf. Di atas model biologis dan di bawah model formal dengan neuron dan koneksi terarah di antara mereka

Langkah besar pertama menuju jaringan saraf di AI dibuat pada tahun 1943 oleh McCullochand Pitt dalam sebuah artikel berjudul “Sebuah kalkulus logis dari gagasan aktivitas saraf inti”. Mereka pertama kali menampilkan model matematis neuron sebagai elemen switching dasar otak. Artikel ini meletakkan dasar untuk konstruksi jaringan saraf tiruan dan dengan demikian untuk cabang AI yang sangat penting ini.

Kita bisa mempertimbangkan bidang pemodelan dan simulasi jaringan saraf menjadi cabang bionik dalam AI.1 Hampir semua bidang AI mencoba untuk menciptakan kembali proses kognitif, seperti dalam logika atau dalam penalaran probabilistik. Namun, alat yang digunakan untuk pemodelan—yaitu matematika, bahasa pemrograman, dan komputer digital—memiliki sedikit kesamaan dengan otak manusia. Dengan jaringan saraf tiruan, pendekatannya berbeda. Berawal dari pengetahuan tentang fungsi jaringan syaraf tiruan, kami mencoba untuk memodelkan, mensimulasikan, dan bahkan merekonstruksinya dalam perangkat keras. Setiap peneliti di bidang ini menghadapi tantangan yang menarik dan mengasyikkan untuk membandingkan hasil dengan kinerja manusia.

Dalam bab ini kami akan mencoba untuk menguraikan perkembangan sejarah dengan mendefinisikan model neuron dan interkonektivitasnya, mulai dari wawasan biologis yang paling penting. Kemudian kami akan menyajikan beberapa model penting dan mendasar: model Hopfield, dua model memori asosiatif sederhana, dan—yang sangat penting dalam praktik—algoritma backpropagation.

9.1 DARI BIOLOGI KE SIMULASI

Masing-masing dari sekitar 100 miliar neuron di otak manusia memiliki, seperti yang ditunjukkan dalam representasi yang disederhanakan pada Gambar 9.1, struktur dan fungsi berikut. Selain badan sel, neuron memiliki akson, yang dapat membuat koneksi lokal ke neuron lain melalui dendrit. Akan tetapi, akson dapat tumbuh hingga satu meter dalam bentuk serabut saraf di seluruh tubuh.

Badan sel neuron dapat menyimpan muatan listrik kecil, mirip dengan kapasitor atau baterai. Penyimpanan ini dimuat oleh impuls listrik yang masuk dari neuron lain. Semakin banyak impuls listrik yang masuk, semakin tinggi tegangannya. Jika tegangan melebihi ambang batas tertentu, neuron akan menyala. Ini berarti bahwa ia membongkar simpanannya, dalam arti ia mengirimkan lonjakan di atas akson dan sinapsis. Arus listrik membelah dan mencapai banyak neuron lain melalui sinapsis, di mana proses yang sama berlangsung.

Sekarang pertanyaan tentang struktur jaringan saraf muncul. Masing-masing dari sekitar 10^{11} neuron di otak terhubung ke sekitar 1000 hingga 10.000 neuron lain, yang menghasilkan total lebih dari 10^{14} koneksi. Jika kita mempertimbangkan lebih lanjut bahwa sejumlah besar sambungan yang sangat tipis ini terdiri dari jaringan lunak tiga dimensi dan bahwa eksperimen pada otak manusia tidak mudah dilakukan, maka menjadi jelas mengapa kita tidak memiliki diagram rangkaian rinci dari otak. Agaknya kita tidak akan pernah mampu sepenuhnya memahami diagram sirkuit otak kita, hanya berdasarkan ukurannya yang sangat besar.

Dari perspektif hari ini, tidak ada gunanya lagi mencoba membuat diagram sirkuit otak yang lengkap, karena struktur otak bersifat adaptif. Ia berubah dengan sendirinya dan beradaptasi sesuai dengan aktivitas individu dan pengaruh lingkungan. Peran sentral di sini dimainkan oleh sinapsis, yang menciptakan hubungan antar neuron. Pada titik koneksi antara dua neuron, seolah-olah dua kabel bertemu. Namun, kedua sadapan tersebut tidak ikat konduktif sempurna, melainkan ada celah kecil, yang tidak dapat dilompati elektron secara langsung. Kesenjangan ini diisi dengan zat kimia, yang disebut neurotransmiter. Ini dapat terionisasi oleh tegangan yang diberikan dan kemudian mengangkut muatan melalui celah. Konduktivitas celah ini tergantung pada banyak parameter, misalnya konsentrasi dan komposisi kimia neurotransmitter. Sungguh mencerahkan bahwa fungsi otak bereaksi

sangat sensitif terhadap perubahan koneksi sinaptik ini, misalnya melalui pengaruh alkohol atau obat-obatan lain.

Bagaimana cara kerja pembelajaran dalam jaringan saraf seperti itu? Hal yang mengejutkan di sini adalah bahwa bukan unit aktif yang sebenarnya, yaitu neuron, yang adaptif, melainkan hubungan di antara mereka, yaitu sinapsis. Secara khusus, ini dapat mengubah konduktivitasnya. Kita tahu bahwa sinapsis dibuat lebih kuat oleh lebih banyak arus listrik yang harus dibawanya. Lebih kuat di sini berarti sinapsis memiliki konduktivitas yang lebih tinggi. Sinapsis yang digunakan seringkali memperoleh bobot yang semakin tinggi. Untuk sinapsis yang jarang digunakan atau tidak aktif sama sekali, konduktivitasnya terus menurun. Ini bahkan dapat menyebabkan mereka sekarat.

Semua neuron di otak bekerja secara asinkron dan paralel, tetapi, dibandingkan dengan komputer, pada kecepatan yang sangat rendah. Waktu untuk impuls saraf membutuhkan waktu sekitar satu milidetik, persis sama dengan waktu untuk ion diangkut melalui celah sinaptik. Frekuensi clock neuron kemudian berada di bawah satu kilohertz dan dengan demikian lebih rendah daripada komputer modern dengan faktor 106. Kerugian ini, bagaimanapun, lebih dari kompensasi untuk banyak tugas kognitif kompleks, seperti pengenalan gambar, oleh tingkat pemrosesan paralel yang sangat tinggi dalam jaringan saraf. sel.

Hubungan dengan dunia luar terjadi melalui neuron sensor, misalnya pada retina di mata, atau melalui sel saraf dengan akson yang sangat panjang yang menjangkau dari otak ke otot dan dengan demikian dapat melakukan tindakan seperti gerakan kaki.
Namun, masih belum jelas bagaimana prinsip-prinsip yang dibahas memungkinkan perilaku cerdas. Sama seperti banyak peneliti dalam ilmu saraf, kami akan mencoba menjelaskan menggunakan simulasi model matematika sederhana bagaimana tugas kognitif, misalnya pengenalan pola, menjadi mungkin.

9.2 MODEL MATEMATIKA

Pertama kita ganti sumbu waktu kontinu dengan skala waktu diskrit. Neuron i melakukan perhitungan berikut dalam langkah waktu. "Pemuatan *(loading)*" dari potensi aktivasi dicapai hanya dengan menjumlahkan nilai output tertimbang x_1,…, x_n dari semua koneksi masuk melalui rumus

$$\sum_{j=1}^{n} w_{ij} x_j$$

Jumlah tertimbang ini dihitung oleh sebagian besar model saraf. Kemudian fungsi aktivasi f diterapkan padanya dan hasilnya

$$x_i = f\left(\sum_{j=1}^{n} w_{ij} x_j\right)$$

diteruskan ke neuron tetangga sebagai output melalui bobot sinaptik. Pada Gambar 9.2 model neuron seperti ini diperlihatkan. Untuk fungsi aktivasi ada beberapa kemungkinan. Yang paling sederhana adalah identitasnya: f(x) = x. Dengan demikian, neuron hanya menghitung jumlah bobot dari nilai input dan meneruskannya. Namun, ini sering menyebabkan masalah konvergensi dengan dinamika saraf karena fungsi f(x) = x tidak terbatas dan nilai fungsi dapat tumbuh melampaui semua batas dari waktu ke waktu. Sangat dibatasi dengan baik, sebaliknya, adalah fungsi ambang (fungsi langkah Heaviside)

$$H_\Theta(x) = \begin{cases} 0 & jika\ x < \Theta \\ 1 & lainnya \end{cases}$$

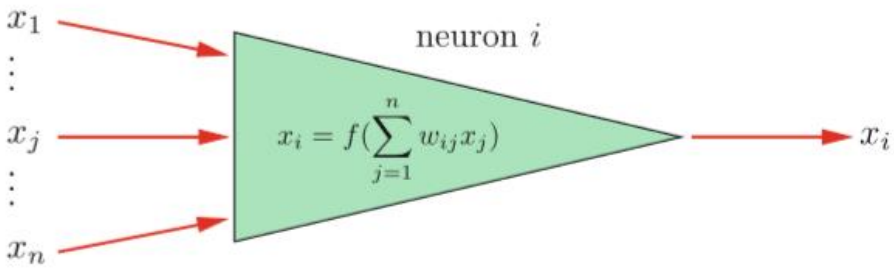
Seluruh neuron kemudian menghitung outputnya sebagai

$$x_i = \begin{cases} 0 & jika \sum_{j=1}^{n} w_{ij}x_j < \Theta \\ 1 & lainnya \end{cases}$$

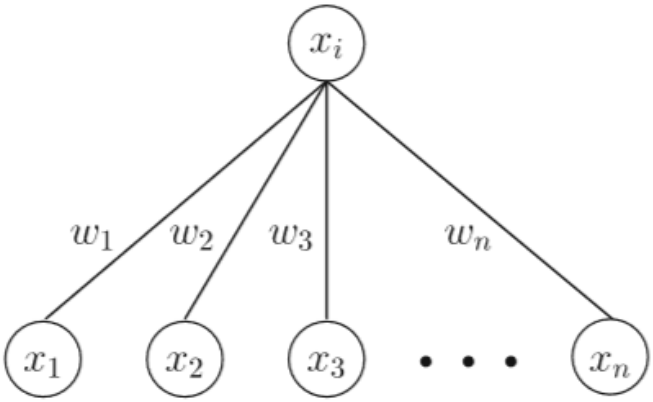
Rumus ini identik dengan (Persamaan 8.1), dengan kata lain, persepsi terhadap threshold Θ (Gambar 9.3). Neuron input 1,…, n di sini hanya memiliki fungsi variabel yang meneruskan nilai yang ditetapkan secara eksternal x_1,…, x_n tidak berubah. Fungsi langkah cukup masuk akal untuk neuron biner karena aktivasi neuron hanya dapat mengambil nilai nol atau satu saja. Sebaliknya, untuk neuron kontinu dengan aktivasi antara 0 dan 1, fungsi langkah menciptakan diskontinuitas. Namun, ini dapat dihaluskan dengan fungsi sigmoid, seperti:

$$f(x)\frac{1}{1+e^{-\frac{x-\Theta}{T}}}$$

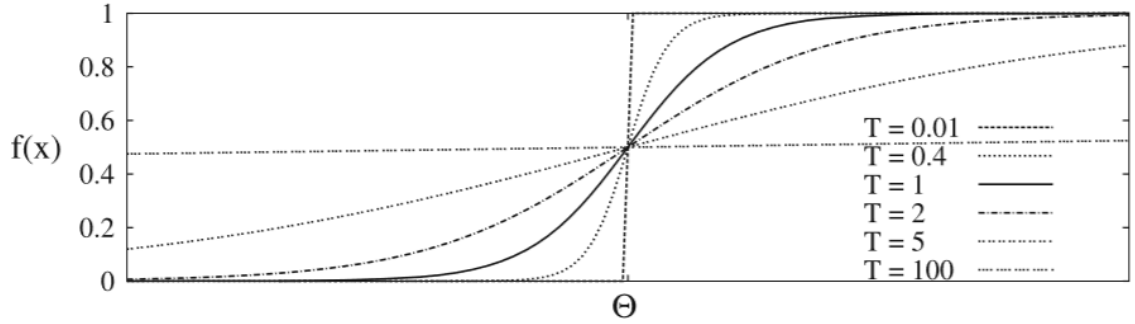
dengan grafik pada Gambar 9.4. Dekat daerah kritis di sekitar ambang H, fungsi ini berperilaku mendekati linier dan memiliki batas asimtotik. Pemulusan dapat divariasikan dengan parameter T.



Gambar 9.2 Struktur neuron formal, yang menerapkan fungsi aktivasi f ke jumlah bobot semua input



Gambar 9.3 Neuron dengan fungsi langkah bekerja seperti perceptron dengan ambang batas



Gambar 9.4 Fungsi sigmoid untuk berbagai nilai parameter T. Kita dapat melihat bahwa pada limit T → 0 hasil fungsi langkah

Pembelajaran pemodelan adalah inti dari teori jaringan saraf. Seperti disebutkan sebelumnya, satu kemungkinan pembelajaran terdiri dari penguatan sinapsis sesuai dengan berapa banyak impuls listrik yang harus ditransmisikan. Prinsip ini didalilkan oleh D. Hebb pada tahun 1949 dan dikenal sebagai aturan Hebb:
Jika ada koneksi w_{ij} antara neuron j dan neuron i dan sinyal berulang dikirim dari neuron j ke neuron i, yang mengakibatkan kedua neuron aktif secara bersamaan, maka bobot w_{ij} diperkuat. Rumus yang mungkin untuk perubahan berat Δw_{ij} adalah

$$\Delta w_{ij} = \eta x_i x_j$$

dengan konstanta η (laju pembelajaran), yang menentukan ukuran langkah pembelajaran individu.
Ada banyak modifikasi dari aturan ini, yang kemudian menghasilkan berbagai jenis jaringan atau algoritma pembelajaran. Pada bagian berikut, kita akan mengenal beberapa di antaranya.

9.3 JARINGAN *HOPFIELD*

Melihat aturan Hebb, kita melihat bahwa untuk neuron dengan nilai antara nol dan satu, bobotnya hanya dapat tumbuh seiring waktu. Tidak mungkin untuk neuron melemah atau bahkan mati menurut aturan ini. Ini dapat dimodelkan, misalnya, dengan konstanta peluruhan yang melemahkan bobot yang tidak digunakan dengan faktor konstan per langkah waktu, seperti 0,99.

Masalah ini diselesaikan dengan cara yang sangat berbeda dengan model yang disajikan oleh Hopfield pada tahun 1982. Itu adalah neuron biner, tetapi dengan dua nilai -1 untuk tidak aktif dan 1 untuk aktif. Menggunakan aturan Hebb, kami memperoleh kontribusi positif terhadap bobot setiap kali dua neuron aktif secara bersamaan. Namun, jika hanya satu dari dua neuron yang aktif, Δw_{ij} negatif.

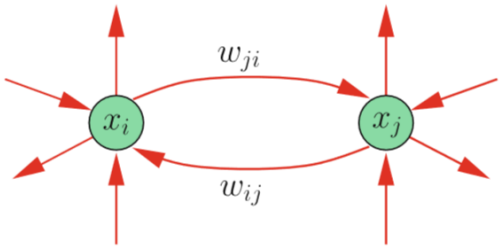
Jaringan hopfield, yang merupakan contoh memori auto-asosiatif yang indah dan dapat divisualisasikan, didasarkan pada gagasan ini. Pola dapat disimpan dalam memori *auto-associative.* Untuk memanggil pola yang disimpan, cukup menyediakan pola yang sama. Toko kemudian menemukan pola tersimpan yang paling mirip. Aplikasi klasik dari ini adalah pengenalan tulisan tangan.

Dalam fase pembelajaran dari jaringan Hopfield, N pola kode biner, disimpan dalam vektor $q^1,\ldots,q^N$, seharusnya dipelajari. Setiap komponen $q^j_i \in \{-1, 1\}$ dari vektor tersebut q^j mewakili piksel dari pola. Untuk vektor yang terdiri dari n piksel, jaringan aneural dengan n neuron digunakan, satu untuk setiap posisi piksel. Neuron terhubung sepenuhnya dengan batasan bahwa matriks bobot simetris dan semua elemen diagonal adalah nol. Artinya, tidak ada hubungan antara neuron dan dirinya sendiri.

Jaringan yang sepenuhnya terhubung mencakup loop umpan balik yang kompleks, yang disebut pengulangan, dalam jaringan (Gambar 9.5). N pola dapat dipelajari hanya dengan menghitung semua bobot dengan rumus
Persamaan 9.1

$$w_{ij} = \frac{1}{N}\sum_{k=1}^{N} q_i^k q_j^k$$

Rumus ini menunjukkan hubungan yang menarik dengan aturan Hebb. Setiap pola di mana piksel dan j memiliki nilai yang sama memberikan kontribusi positif terhadap bobot wij. Setiap pola memberikan kontribusi negatif. Karena setiap piksel berhubungan dengan neuron, di sini bobot antar neuron yang secara bersamaan memiliki nilai yang sama diperkuat. Harap perhatikan perbedaan kecil ini dengan aturan Hebb.



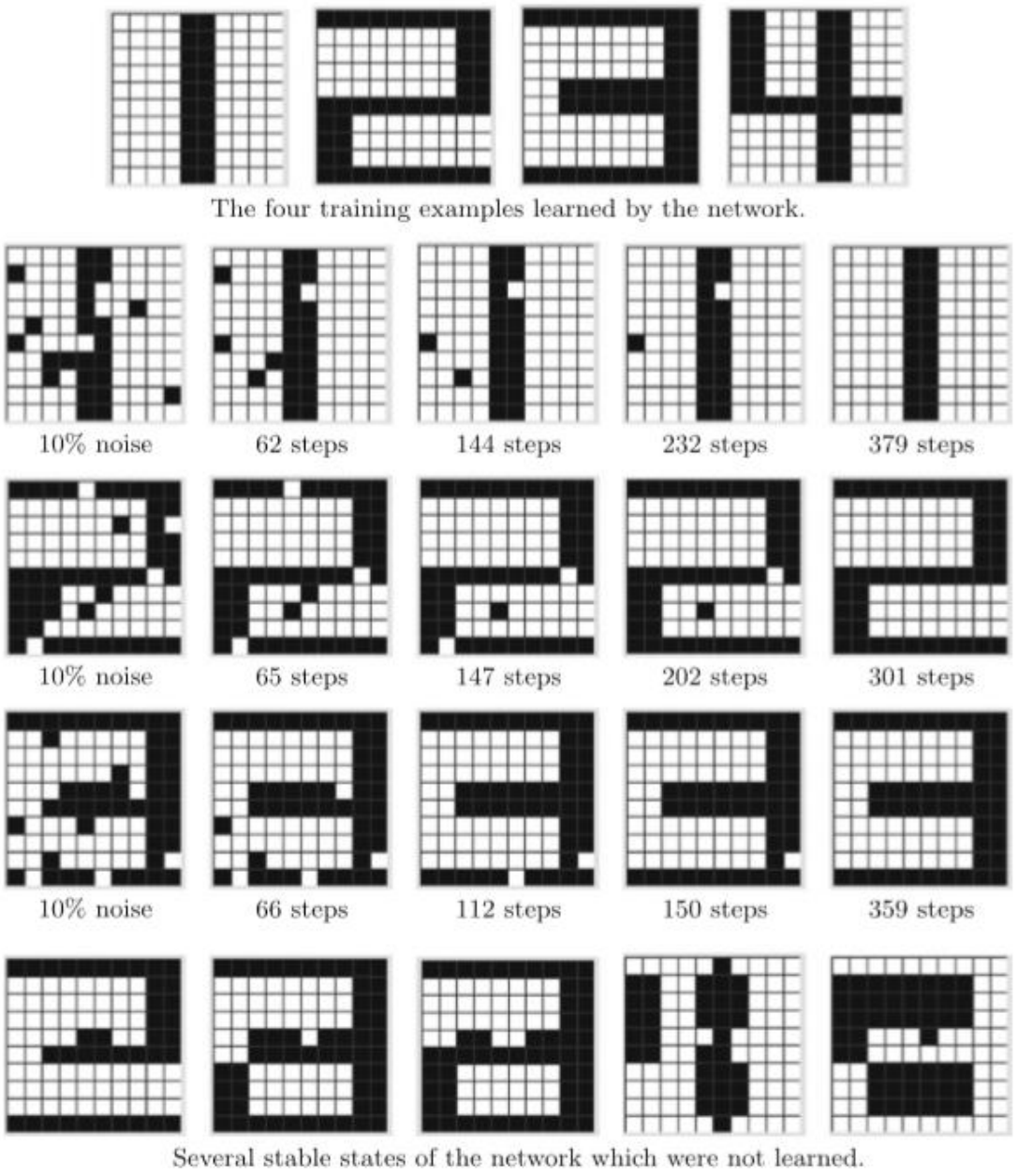
Gambar 9.5 Hubungan berulang antara dua neuron dalam jaringan Hopfield

Setelah semua pola telah disimpan, jaringan dapat digunakan untuk pengenalan pola. Kami memberikan jaringan pola x baru dan memperbarui aktivasi semua neuron dalam proses asinkron sesuai dengan aturan

$$x_i = \begin{cases} -1 & jika \sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij}x_j < 0 \\ 1 & lainnya \end{cases}$$

sampai jaringan menjadi stabil, yaitu sampai tidak ada lagi perubahan aktivasi. Sebagai skema program ini berbunyi sebagai berikut:

HOPFIELDASSOCIATOR($\boldsymbol{q}$)
Initialize all neurons: $\boldsymbol{x} = \boldsymbol{q}$
Repeat
  i = Random(1, n)
  Update neuron i according to (9.2)
Until $\boldsymbol{x}$ converges
Return ($\boldsymbol{x}$)



Gambar 9.6 Dinamika jaringan Hopfield.

Di baris 2, 3 dan 4 kita dapat dengan mudah melihat bagaimana jaringan menyatu dan pola yang dipelajari dikenali setelah sekitar 300 hingga 400 iterasi. Pada baris terakhir beberapa keadaan stabil ditunjukkan yang dicapai oleh jaringan ketika pola input menyimpang terlalu banyak dari semua pola yang dipelajari.

9.4 APLIKASI UNTUK CONTOH PENGENALAN POLA

Kami menerapkan algoritme yang dijelaskan ke contoh pengenalan pola sederhana. Ini harus mengenali digit dalam bidang 10 x 10 piksel. Jaringan Hopfield dengan demikian memiliki 100 neuron dengan total

$$\frac{1100 \ . \ 99}{2} = 4950$$

bobot. Pertama pola angka 1, 2, 3, 4 pada Gambar 9.6 di atas dilatih. Artinya, bobot dihitung dengan (9.1) . Kemudian kita masukkan pola dengan noise tambahan dan biarkan dinamika Hopfield berjalan hingga konvergen. Pada baris 2 sampai 4 pada gambar, lima snapshot dari

pengembangan jaringan ditampilkan selama pengenalan. Pada kebisingan 10%, keempat pola yang dipelajari dikenali dengan sangat andal. Di atas sekitar 20% noise, algoritme sering menyatu dengan pola yang dipelajari lainnya atau bahkan ke pola yang tidak dipelajari. Beberapa pola tersebut ditunjukkan pada Gambar 9.6 di bawah ini.

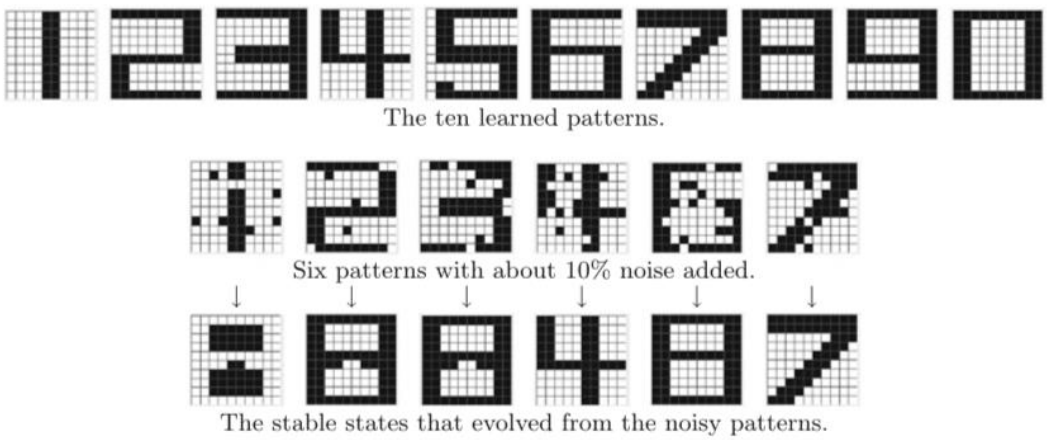
Sekarang kita menyimpan angka 0 sampai 9 (Gambar 9.7) di jaringan yang sama dan menguji jaringan lagi dengan pola yang memiliki jumlah acak sekitar 10% piksel terbalik. Dalam gambar, kita dengan jelas melihat bahwa iterasi Hopfield sering kali tidak konvergen ke keadaan terpelajar yang paling mirip bahkan untuk noise hanya 10%. Jelas jaringan dapat dengan aman menyimpan dan mengenali empat pola, tetapi untuk sepuluh pola, kapasitas memorinya terlampaui. Untuk memahami ini lebih baik, kita akan melihat sekilas teori jaringan ini.

Analisis

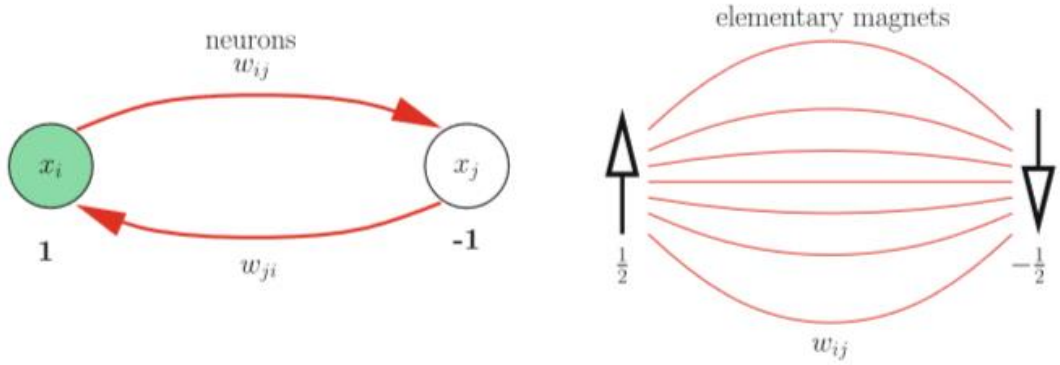
Pada tahun 1982, John Hopfield menunjukkan bahwa model ini secara formal setara dengan model fisik magnetisme. Magnet elementer kecil, yang disebut spin, saling mempengaruhi satu sama lain di atas medan magnetnya (lihat Gambar 9.8). Jika kita mengamati dua putaran seperti i dan j, keduanya berinteraksi pada wij konstan dan energi total sistem adalah

$$E = -\frac{1}{2}\sum_{ij} w_{ij} x_i x_j$$

Omong-omong, $w_{ii} = 0$ dalam fisika juga, karena partikel tidak memiliki interaksi diri. Karena interaksi fisik bersifat simetris, $w_{ij} = w_{ji}$.



Gambar 9.7 Untuk sepuluh status terpelajar, jaringan menunjukkan perilaku kacau. Bahkan dengan sedikit noise, jaringan menyatu dengan pola atau artefak yang salah



Gambar 9.8 Perbandingan antara interpretasi saraf dan fisik dari model Hopfield

Sebuah sistem fisik dalam kesetimbangan mengambil keadaan (stabil) energi minimal dan dengan demikian meminimalkan E(x, y). Jika sistem seperti itu dibawa ke keadaan sewenang-wenang, maka ia bergerak menuju keadaan energi minimal. Dinamika Hopfield yang didefinisikan dalam (Persamaan 9.2) sesuai dengan prinsip ini karena ia memperbarui keadaan di setiap iterasi sedemikian rupa sehingga, dari dua keadaan – 1 dan 1, yang diambil dengan energi total lebih kecil. Kontribusi neuron i terhadap energi total adalah

$$-\frac{1}{2}x_i\sum_{j\neq 1}^{n}w_{ij}x_j$$

Jika sekarang

$$\sum_{j\neq 1}^{n}w_{ij}x_j<0$$

maka x_i = - 1 m enghasilkan kontribusi negatif terhadap total energi, dan xi 1 menghasilkan kontribusi positif. Untuk x_i = -1, jaringan mengambil keadaan energi yang lebih rendah daripada untuk x_i = 1. Secara analog, kita dapat menyatakan bahwa dalam kasus

$$\sum_{j\neq 1}^{n}w_{ij}x_j\geq 0$$

harus benar bahwa x_i = 1.

Jika setiap iterasi individu dari di≠namika saraf menghasilkan pengurangan fungsi energi, maka energi total sistem berkurang secara monoton dengan waktu. Karena hanya ada banyak keadaan, jaringan bergerak dalam waktu ke keadaan energi minimal. Sekarang kita memiliki pertanyaan menarik: apa arti minima dari fungsi energi ini?

Saat kita memenangkan eksperimen pengenalan pola, dalam kasus pola yang dipelajari, sistem konvergen ke salah satu pola yang dipelajari. Pola-pola yang dipelajari merepresentasikan minima dari fungsi energi dalam ruang keadaan. Namun jika terlalu banyak pola yang dipelajari, maka sistem konvergen ke minima yang tidak sesuai dengan pola yang dipelajari. Di sini kita memiliki transisi dari dinamika yang teratur menjadi dinamis yang kacau.

Hopfield dan fisikawan lain telah menyelidiki dengan tepat proses ini dan telah menunjukkan bahwa sebenarnya ada transisi fase pada sejumlah pola yang dipelajari. Jika jumlah pola yang dipelajari melebihi nilai ini, maka sistem berubah dari fase teratur menjadi kacau.

Dalam fisika magnetik ada transisi seperti itu dari mode feromagnetik, di mana semua magnet elementer mencoba mengorientasikan diri secara paralel, ke apa yang disebut kaca spin, di mana putaran berinteraksi secara kacau. Contoh yang lebih dapat divisualisasikan dari transisi fase fisik seperti itu adalah pencairan kristal es. Kristal berada dalam keteraturan yang tinggi karena molekul H_2O sangat teratur. Sebaliknya, dalam air cair, struktur molekulnya larut dan posisinya lebih acak.

Dalam jaringan saraf kemudian terjadi transisi fase dari pembelajaran yang teratur dan pengenalan pola ke pembelajaran yang kacau dalam hal terlalu banyak pola, yang tidak lagi dapat dikenali secara pasti. Di sini kita pasti melihat kesejajaran dengan efek yang kadang-kadang kita alami sendiri.

Kita dapat memahami transisi fase ini jika kita membawa semua neuron ke status pola, misalnya q^1, dan memasukkan bobot yang dipelajari dari (Persamaan 9.1) ke dalam istilah $\sum^{n}{}_{j=1,\, j\neq i} w_{ij}1_j$, yang relevan untuk memperbarui neuron i . Ini menghasilkan

$$\sum_{\substack{j=1\\ j\neq 1}}^{n} w_{ij}q_j^i = \frac{1}{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\sum_{k=2}^{N} q_i^k q_j^k q_j^1 = \frac{1}{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\left(q_j^1(q_j^1)^2 + \sum_{k=2}^{N} q_i^k q_j^k q_j^1\right)$$

$$= q_i^1 + \frac{1}{n}\sum_{\substack{j=1\\ j\neq 1}}^{n}\sum_{k=2}^{N} q_i^k q_j^k q_j^1$$

Di sini kita melihat komponen ke-i dari pola input ditambah jumlah dengan suku (n - 1) (N - 1). Jika penjumlahan ini semuanya independen secara statistik, maka kita dapat menggambarkan jumlah tersebut dengan variabel acak terdistribusi normal dengan standar deviasi

$$\frac{1}{n}\sqrt{(n-1)(N-1)} \approx \sqrt{\frac{N-1}{n}}$$

Independensi statistik dapat dicapai, misalnya, dengan pola acak yang tidak berkorelasi. Jumlah tersebut kemudian menghasilkan noise yang tidak mengganggu selama $N \ll n$, yang berarti jumlah pola yang dipelajari tetap jauh lebih kecil daripada jumlah neuron. Namun, jika $N \approx n$, maka pengaruh kebisingan menjadi sebesar pola dan jaringan bereaksi kacau. Perhitungan yang lebih tepat dari transisi fase memberikan $N = 0.146\ n$ sebagai titik kritis. Diterapkan pada contoh kita, ini berarti bahwa untuk 100 neuron, hingga 14 pola dapat disimpan. Karena pola dalam contoh sangat berkorelasi, nilai kritis jauh lebih rendah, jelas antara 0,04 dan 0,1. Bahkan nilai 0,146 jauh lebih rendah daripada kapasitas penyimpanan daftar memori tradisional (Latihan 9.3).

Jaringan hopfield dalam bentuk yang disajikan hanya bekerja dengan baik ketika pola dengan sekitar 50% 1-bit dipelajari. Jika bit terdistribusi sangat asimetris, maka neuron harus dilengkapi dengan ambang. Fisika ini analog dengan penerapan medan magnet luar, yang juga menyebabkan asimetri spin 1/2 dan spin -1/2 negara bagian.

Ringkasan dan Pandangan

Melalui kemungkinan biologisnya, model matematika yang dipahami dengan baik, dan di atas semua itu melalui simulasi yang mengesankan dalam pengenalan pola, model Hopfield berkontribusi pada gelombang kegembiraan tentang jaringan saraf dan munculnya neuroinformatika sebagai cabang penting AI.[49] Selanjutnya banyak model jaringan lebih lanjut dikembangkan. Di satu sisi, jaringan tanpa back-coupling diselidiki karena dinamikanya secara signifikan lebih mudah dipahami daripada jaringan Hopfield berulang. Di sisi lain, upaya dilakukan untuk meningkatkan kapasitas penyimpanan jaringan, yang akan kita bahas di bagian selanjutnya.

Masalah khusus dari banyak model saraf sudah terbukti dalam model Hopfield. Bahkan jika ada jaminan konvergensi, tidak pasti apakah jaringan akan konvergen ke keadaan yang dipelajari atau macet di minimum lokal. Mesin Boltzmann, dengan nilai aktivasi berkelanjutan dan aturan pembaruan probabilistik untuk dinamika jaringannya, dikembangkan sebagai upaya untuk memecahkan masalah ini. Dengan menggunakan parameter "suhu", kita dapat memvariasikan jumlah perubahan keadaan acak dan dengan demikian mencoba untuk keluar dari minimum lokal, dengan tujuan menemukan minimum global yang stabil. Algoritma ini disebut "simulated annealing". Annealing adalah proses perlakuan panas logam dengan tujuan membuat logam lebih kuat dan lebih “stabil”.

Model Hopfield melakukan pencarian fungsi energi minimum dalam ruang nilai aktivasi. Dengan demikian menemukan pola yang disimpan dalam bobot, dan yang dengan

[49] Bahkan penulisnya terbawa oleh gelombang ini, yang membawanya dari fisika ke AI pada tahun 1987

demikian direpresentasikan dalam fungsi energi. Dinamika Hopfield juga dapat diterapkan pada fungsi energi lainnya, selama matriks bobotnya simetris dan elemen diagonalnya nol. Ini berhasil didemonstrasikan oleh Hopfield dan Tank pada masalah travelling salesman [HT85, Zel94]. Tugas di sini adalah, dengan n kota dan matriks jaraknya, untuk menemukan perjalanan pulang pergi terpendek yang mengunjungi setiap kota tepat satu kali.

9.5 MEMORI ASOSIATIF SARAF

Memori daftar tradisional dalam kasus yang paling sederhana dapat berupa file teks di mana deretan angka disimpan baris demi baris. Jika file diurutkan berdasarkan baris, maka pencarian elemen dapat dilakukan dengan sangat cepat dalam waktu logaritmik, bahkan untuk file yang sangat besar.

Memori daftar juga dapat digunakan untuk membuat pemetaan. Misalnya, buku telepon adalah pemetaan dari himpunan semua nama yang dimasukkan ke himpunan semua nomor telepon. Pemetaan ini diimplementasikan sebagai tabel sederhana, biasanya disimpan dalam database.

Kontrol akses ke gedung menggunakan pengenalan wajah adalah tugas yang serupa. Di sini kita juga bisa menggunakan database di mana foto setiap orang disimpan bersama dengan nama orang tersebut dan mungkin data lainnya. Kamera di pintu masuk kemudian mengambil gambar orang tersebut dan mencari database untuk foto yang sama. Jika foto ditemukan, maka orang tersebut diidentifikasi dan mendapat akses ke gedung. Namun, bangunan dengan sistem kontrol seperti itu tidak akan mendapatkan banyak pengunjung karena kemungkinan foto yang sekarang cocok dengan foto yang disimpan sangat kecil.

Dalam hal ini tidak cukup hanya menyimpan foto dalam sebuah tabel. Sebaliknya, yang kita inginkan adalah memori asosiatif, yang tidak hanya mampu menetapkan nama yang tepat untuk foto itu, tetapi juga untuk semua foto "serupa" yang berpotensi tak terbatas. Sebuah fungsi untuk menemukan kesamaan harus dihasilkan dari kumpulan data pelatihan yang terbatas, yaitu foto-foto yang disimpan yang diberi label dengan nama. Pendekatan sederhana untuk ini adalah metode tetangga terdekat. Selama pembelajaran, semua foto disimpan begitu saja.

Untuk menerapkan fungsi ini, foto yang paling mirip dengan yang sekarang harus ditemukan di database. Untuk database dengan banyak foto beresolusi tinggi, proses ini, tergantung pada metrik jarak yang digunakan, dapat memerlukan waktu komputasi yang sangat lama dan dengan demikian tidak dapat diimplementasikan dalam bentuk sederhana ini. Oleh karena itu, daripada algoritme malas seperti itu, kami akan memilih yang mentransfer data ke dalam fungsi yang kemudian menciptakan asosiasi yang sangat cepat saat diterapkan.

Menemukan metrik jarak yang sesuai menghadirkan masalah lebih lanjut. Kami ingin seseorang dikenali bahkan jika wajah orang tersebut muncul di tempat lain pada foto (terjemahan), atau jika lebih kecil, lebih besar, atau bahkan diputar. Sudut pandang dan pencahayaan mungkin juga berbeda.

Di sinilah jaringan saraf menunjukkan kekuatan mereka. Tanpa mengharuskan pengembang untuk memikirkan metrik kesamaan yang sesuai, mereka tetap memberikan hasil yang baik. Kami akan memperkenalkan dua model memori asosiatif paling sederhana dan mulai dengan model oleh Teuvo Kohonen, salah satu pelopor di bidang ini.

Model Hopfield yang disajikan dalam bab sebelumnya akan terlalu sulit untuk digunakan karena dua alasan. Pertama, ini hanya memori *auto-associative,* yaitu pemetaan yang kurang lebih identik yang memetakan objek yang mirip dengan aslinya yang dipelajari.

Kedua, dinamika berulang yang kompleks seringkali sulit dikelola dalam praktik. Oleh karena itu sekarang kita akan melihat jaringan feedforward dua lapis sederhana.

9.6 MEMORI MATRIKS KORELASI

Dalam Penelitian Kohonen memperkenalkan model memori asosiatif berdasarkan aljabar linier dasar. Ini memetakan vektor kueri x ∈ $\mathbb{R}^n$ ke vektor hasil y ∈ $\mathbb{R}^m$. Kami mencari matriks W yang memetakan dengan benar dari satu set data pelatihan

$$T = \{(q^1, t^1, ..., (q^N, t^N))$$

dengan N kueri-respons memasangkan semua vektor kueri ke responsnya.[50] Artinya, untuk p = 1, …, N harus terjadi bahwa

Persamaan 9.3

$$T^p = \boldsymbol{W} \boldsymbol{.} \boldsymbol{q}^p$$

Atau

Persamaan 9.4

$$t_i^p = \sum_{j=1}^{n} w_{ij} q_j^p$$

Untuk menghitung elemen matriks wij, aturannya

Persamaan 9.5

$$w_{ij} = \sum_{p=1}^{N} q_j^p t_i^p$$

digunakan. Kedua persamaan linier ini dapat dipahami secara sederhana sebagai jaringan saraf jika kita mendefinisikan, seperti pada Gambar 9.9, jaringan dua lapis dengan q sebagai lapisan input dan t sebagai lapisan *output*. Neuron dari lapisan *output* memiliki fungsi aktivasi linier, dan (9.5) digunakan sebagai aturan pembelajaran, yang sesuai dengan aturan Hebb. Sebelum kami menunjukkan bahwa jaringan mengenali data pelatihan, kami membutuhkan definisi berikut:

Definisi 9.1

Dua buah vektor x dan y disebut ortonormal jika

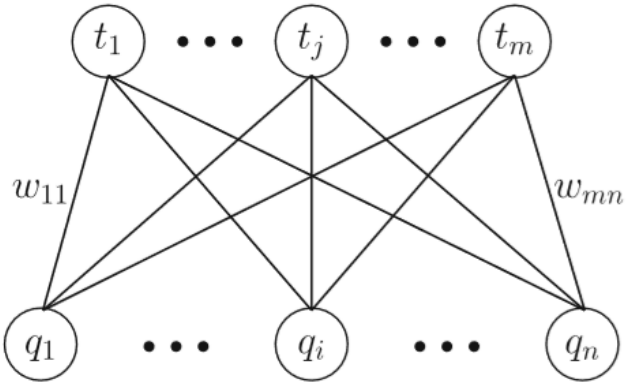
$$x^T \ . \ y = \begin{cases} 1 & jika\ x = y \\ 0 & lainnya \end{cases}$$

Dengan demikian

Teorema 9.1

Jika semua N vektor query q^p dalam data pelatihan ortonormal, maka setiap vektor q^p dipetakan ke vektor target t^p dengan perkalian dengan matriks W dari (Persamaan 9.5).

[50] Untuk perbedaan yang jelas antara data pelatihan dan nilai-nilai lain dari neuron, dalam diskusi berikut kita akan merujuk ke vektor kueri sebagai q dan respons yang diinginkan sebagai t (target).



Gambar 9.9 Representasi memori asosiatif Kohonen sebagai jaringan saraf dua lapis

Bukti: Kami mengganti (9.5) menjadi (9.4) dan memperoleh

$$(W \ . \ q^p)_i = \sum_{j=1}^{n} w_{ij} q_j^p = \sum_{j=1}^{n} \sum_{r=1}^{N} q_j^p t \, q_j^p t \, q_j^p = \sum_{j=1}^{n} w_{ij} \left(q_j^p \, q_j^p t_i^p \sum_{\substack{r=1 \\ r \neq p}}^{N} q_j^r \, q_j^p t_{ji}^r \right)$$

Jadi, jika vektor kueri ortonormal, semua pola akan dipetakan dengan benar ke target masing-masing. Namun, ortonormalitas adalah batasan yang terlalu kuat. Bagian selanjutnya kami akan menyajikan pendekatan yang mengatasi keterbatasan ini.

Karena pemetaan linier kontinu dan injektif, kita tahu bahwa pemetaan dari vektor kueri ke vektor target mempertahankan kesamaan. Kueri serupa dengan demikian dipetakan ke target serupa karena kontinuitas. Namun, pada saat yang sama kami tahu bahwa kueri yang berbeda dipetakan ke target yang berbeda. Jika jaringan dilatih untuk memetakan wajah ke nama dan jika nama Henry ditugaskan ke wajah, maka kami yakin bahwa untuk input wajah yang serupa, output yang mirip dengan "Henry" akan dihasilkan, tetapi "Henry" itu sendiri dijamin tidak akan dihitung. Jika output dapat diartikan sebagai string, maka, misalnya, itu bisa menjadi "Genry" atau "Gfnry". Untuk sampai pada kasus terpelajar yang paling mirip, Kohonen menggunakan pengkodean biner untuk neuron *output*. Hasil penghitungan kueri dibulatkan jika nilainya bukan nol atau satu. Meski begitu kami tidak memiliki jaminan bahwa kami akan mencapai vektor target. Atau, kita dapat menambahkan pemetaan berikutnya dari jawaban yang dihitung ke vektor target yang dipelajari dengan jarak terkecil.

9.7 ATURAN BINER HERB

Dalam konteks memori asosiatif, apa yang disebut aturan Hebb biner diusulkan. Ini membutuhkan pola yang dikodekan biner. Artinya untuk semua pola $q^p \in \{0, 1\}^n$ dan $t^p \in \{0, 1\}^m$. Selanjutnya, penjumlahan dari (9.5)diganti dengan logika OR sederhana dan kita memperoleh aturan biner Hebb

Persamaan 9.10

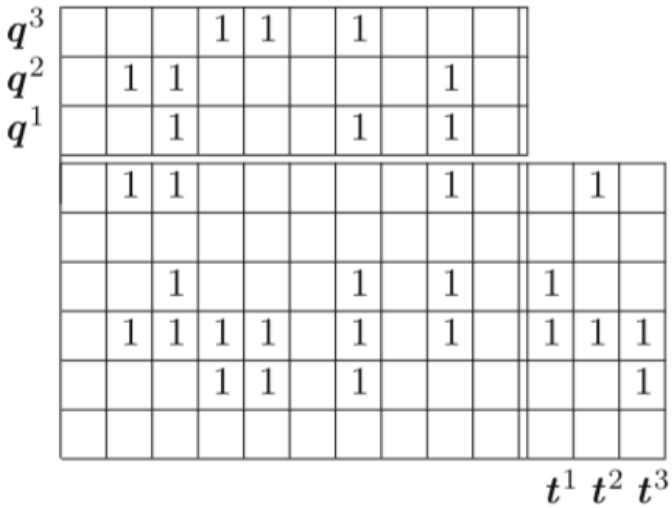
$$w_{ij} = \bigvee_{p=1}^{N} q_j^p t_i^p$$

Matriks bobot dengan demikian juga biner, dan elemen matriks wij sama dengan satu jika dan hanya jika setidaknya salah satu entri $q_j^p t_i^1, \ldots, q_j^N t_i^N$ tidak nol. Semua elemen matriks lainnya adalah nol. Kami tergoda untuk percaya bahwa banyak informasi yang hilang di sini selama pembelajaran karena, ketika elemen matriks mengambil nilai 1 sekali, itu tidak

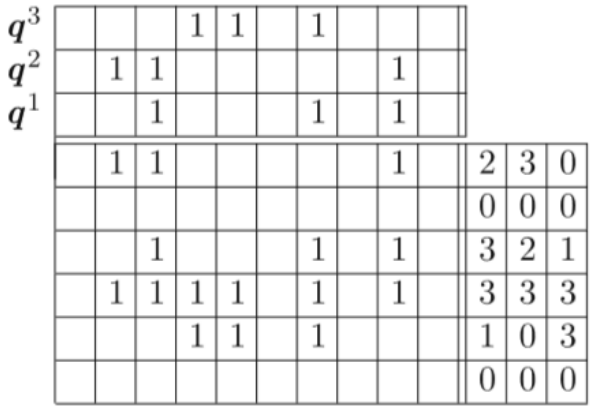
dapat diubah dengan pola tambahan. Gambar 9.10 menunjukkan bagaimana matriks diisi dengan satu untuk contoh dengan n = 10; m = 6 setelah belajar tiga pasang.

Untuk mengambil pola yang disimpan, kita cukup mengalikan vektor kueri ***q*** dengan matriks dan lihat hasilnya ***Wq***. Ujilah ini pada contoh dan dapatkan Gambar. 9.11. Kita melihat bahwa di vektor target di sebelah kanan ada nilai 3 di tempat vektor target yang dipelajari memiliki satu. Hasil yang benar akan diperoleh dengan menetapkan nilai ambang 3. Dalam kasus umum kami memilih jumlah yang dalam vektor kueri sebagai ambang batas. Setiap neuron *output* dengan demikian bekerja seperti perceptron, meskipun dengan ambang variabel.

Selama matriks bobot jarang, algoritma ini bekerja dengan baik. Namun, jika banyak pola yang berbeda disimpan, matriks menjadi semakin padat. Dalam kasus ekstrim itu hanya berisi satu. Kemudian, setelah menetapkan ambang batas, semua jawaban hanya terdiri dari satu dan tidak lagi berisi informasi apa pun.



Gambar 9.10 Matriks W setelah menyimpan tiga pasangan (q^1, t^1), (q^2, t^2), (q^3, t^3). Bidang kosong sesuai dengan nilai 0



Gambar 9.11 Perhitungan produk $\mathbf{W_q^1}$, $\mathbf{W_q^2}$, $\mathbf{W_q^3}$

Kasus ini jarang terjadi selama jumlah bit yang disimpan dalam matriks tidak menjadi terlalu besar. Matriks memiliki ukuran mnelement. Pasangan yang akan disimpan memiliki m + n bit. Seseorang dapat menunjukkan bahwa jumlah pola yang dapat diingat N_{max} ditentukan oleh kondisi berikut:

$$\alpha = \frac{jumlah\ bit\ yang\ dapat\ disimpan,}{jumlah\ elemen\ matriks\ biner} = \frac{(m+n)N_{max}}{mn} \leq 2 \approx 0.69$$

Untuk memori daftar kita memiliki α = 1. Memori asosiatif dengan aturan Hebb biner memiliki efisiensi memori maksimum 0:69 dibandingkan dengan α = 0,72 untuk memori asosiatif Kohonen dan α = 0,292 untuk jaringan Hopfield. Kapasitas memori aturan Hebb biner dengan demikian sangat tinggi dibandingkan dengan model Kohonen dengan neuron kontinu.

Jelas bahwa memori seperti itu menjadi "penuh" lebih cepat ketika kueri dan vektor target jarang diisi. Ini bukan satu-satunya alasan mengapa pengkodean input dan output untuk memori asosiatif—seperti juga untuk jaringan saraf lainnya—sangat penting untuk kinerja yang baik. Kami sekarang akan mendemonstrasikan ini pada aplikasi memori ini dengan pengkodean input dan output yang dipilih dengan tepat.

Stored words:

agathe, agnes, alexander, andreas, andree, anna, annemarie, astrid, august, bernhard, bjorn, cathrin, christian, christoph, corinna, corrado, dieter, elisabeth, elvira, erdmut, ernst, evelyn, fabrizio, frank, franz, geoffrey, georg, gerhard, hannelore, harry, herbert, ingilt, irmgard, jan, johannes, johnny, juergen, karin, klaus, ludwig, luise, manfred, maria, mark, markus, marleen, martin, matthias, norbert, otto, patricia, peter, phillip, quit, reinhold, renate, robert, robin, sabine, sebastian, stefan, stephan, sylvie, ulrich, ulrike, ute, uwe, werner, wolfgang, xavier

Associations of the program:

```
input pattern: harry

threshold: 4, answer: harry
threshold: 3, answer: harry
threshold: 2, answer: horryrrde

------------------------------
input pattern: ute

threshold: 2, answer: ute

------------------------------
input pattern: gerhar

threshold: 5, answer: gerhard
threshold: 4, answer: gerrarn
threshold: 3, answer: jerrhrrd
threshold: 2, answer: jurtyrrde

------------------------------
input pattern: egrhard

threshold: 6, answer:
threshold: 5, answer:
threshold: 4, answer: gerhard
threshold: 3, answer: gernhrrd
threshold: 2, answer: irrryrrde

------------------------------
input pattern: andr

threshold: 3, answer: andrees
threshold: 2, answer: anexenser
```

```
input pattern: andrees

threshold: 6, answer: a
threshold: 5, answer: andree
threshold: 4, answer: andrees
threshold: 3, answer: mnnrens
threshold: 2, answer: morxsnssr

------------------------------
input pattern: johanne

threshold: 6, answer: johnnnes
threshold: 5, answer: johnnnes
threshold: 4, answer: jornnnrse
threshold: 3, answer: sorrnyrse
threshold: 2, answer: wtrrsyrse

------------------------------
input pattern: johnnnes

threshold: 6, answer: joh
threshold: 5, answer: johnnnes
threshold: 4, answer: johnnyes
threshold: 3, answer: jonnnyes
threshold: 2, answer: jornsyrse

------------------------------
input pattern: johnnyes

threshold: 7, answer:
threshold: 6, answer: joh
threshold: 5, answer: johnny
threshold: 4, answer: johnnyes
threshold: 3, answer: johnnyes
threshold: 2, answer: jonnnyes
```

Gambar 9.12 Penerapan program koreksi ejaan untuk berbagai input yang dipelajari atau salah.

Input yang benar ditemukan dengan ambang batas maksimum (yaitu percobaan pertama). Untuk input yang salah, ambang batas harus diturunkan untuk asosiasi yang benar

9.8 PROGRAM KOREKSI EJAAN

Sebagai aplikasi dari memori asosiatif yang dijelaskan dengan aturan biner Hebb, kami memilih program yang mengoreksi input yang salah dan memetakannya ke kata-kata yang disimpan dari kamus. Jelas memori auto-associative akan dibutuhkan di sini. Namun, karena kami menyandikan kueri dan vektor target secara berbeda, hal ini tidak terjadi. Untuk vektor kueri q kami memilih penyandian pasangan. Untuk alfabet dengan 26 karakter terdapat 26 . 26 = 676 pasangan huruf yang berurutan. Dengan 676 bit, vektor kueri memiliki satu bit untuk setiap pasangan yang mungkin

Aa, ab, ..., az, ba, ..., bz, ..., za, ..., zz.

Jika sepasang huruf muncul dalam kata, maka satu akan dimasukkan di tempat yang sesuai. Untuk kata “hans”, misalnya, slot untuk “ha”, “an”, dan “ns” diisi satu. Untuk vektor target t, 26 bit dicadangkan untuk setiap posisi dalam kata hingga panjang maksimum (misalnya sepuluh karakter). Untuk huruf ke-i pada abjad pada posisi j pada kata maka bilangan bit (j - 1) 26 + i diatur. Untuk kata “hans”, bit 8, 27, 66, dan 97 diset. Untuk maksimum 10 huruf per kata, vektor target memiliki panjang 260 bit. Matriks bobot ***W*** dengan demikian memiliki ukuran 676 . 260 bit = 199420 bit, yang dengan(9.11) dapat menyimpan paling banyak

$$N_{max} \leq 0.69 \frac{mn}{m+n} = 0.69 \frac{676 \ . \ 260}{676 + 260} \approx 130$$

kata-kata. Dengan 72 nama depan, kami menghemat sekitar setengahnya dan menguji sistem. Nama yang disimpan dan *output* program untuk beberapa contoh input diberikan pada Gambar 9.12 Ambang selalu diinisialisasi ke jumlah bit dalam kueri yang dikodekan. Ini adalah jumlah pasangan huruf, jadi panjang kata dikurangi satu. Kemudian secara bertahap dikurangi menjadi dua. Kami selanjutnya dapat mengotomatiskan pilihan ambang batas dengan membandingkan dengan kamus untuk setiap ambang batas yang dicoba dan mengeluarkan kata yang ditemukan ketika perbandingan berhasil.

Reaksi terhadap input ambigu "andr" dan "johanne" menarik. Dalam kedua kasus, jaringan membuat campuran dua kata tersimpan yang cocok. Kami melihat di sini kekuatan penting dari jaringan saraf. Mereka mampu membuat asosiasi ke objek serupa tanpa metrik kesamaan eksplisit. Namun, mirip dengan pencarian heuristik dan pengambilan keputusan manusia, tidak ada jaminan untuk solusi yang "benar".

Karena data pelatihan harus tersedia dalam bentuk pasangan input-output untuk semua model saraf yang telah diperkenalkan sejauh ini, kita berurusan dengan pembelajaran terawasi, yang juga merupakan kasus untuk jaringan yang diperkenalkan di bagian berikut.

9.9 JARINGAN LINIER DENGAN KESALAHAN MINIMAL

Aturan Hebb yang digunakan dalam model saraf yang disajikan sejauh ini bekerja dengan asosiasi antara neuron tetangga. Memori inasosiatif, ini dieksploitasi untuk mempelajari pemetaan dari vektor kueri ke target. Ini bekerja dengan sangat baik dalam banyak kasus, terutama ketika vektor kueri linearindependen. Jika kondisi ini tidak terpenuhi, misalnya ketika terlalu banyak data pelatihan tersedia, muncul pertanyaan: bagaimana kita menemukan matriks bobot optimal? Optimal berarti meminimalkan kesalahan rata-rata.

Kita manusia mampu belajar dari kesalahan. Aturan Hebb tidak menawarkan kemungkinan ini. Algoritma backpropagation, yang dijelaskan berikut ini, menggunakan solusi elegan yang diketahui dari aproksimasi fungsi untuk mengubah bobot sedemikian

rupa sehingga kesalahan pada data pelatihan diminimalkan. Biarkan N pasang vektor pelatihan

$$T = \{(q^1, t^1), ..., (q^N, t^N)$$

diberikan dengan $q^p \in [0, 1]^n t^p \in [0, 1]^m$. Kami mencari fungsi $f : [0, 1]^n \rightarrow [0, 1]^m$ yang meminimalkan kesalahan kuadrat

$$\sum_{p=1}^{N} (f(q^p) - t^p)^2$$

pada datanya. Mari kita asumsikan dulu bahwa data tidak mengandung kontradiksi. Artinya, tidak ada vektor kueri dalam data pelatihan yang harus dipetakan ke dua target yang berbeda. Dalam hal ini tidak sulit untuk menemukan fungsi yang meminimalkan kesalahan kuadrat. Faktanya, ada banyak sekali fungsi yang membuat kesalahan menjadi nol. Kami mendefinisikan fungsi

$$f(q) = 0, \quad \text{jika } q \notin \{q^1, ..., q^N\}$$

dan

$$f(q^p) = t^p \; \forall \; p \in \{1, ..., N\}.$$

Ini adalah fungsi yang bahkan membuat kesalahan pada data pelatihan menjadi nol. Apa lagi yang kita inginkan? Mengapa kita tidak senang dengan fungsi ini?

Jawabannya adalah: karena kami ingin membangun sistem yang cerdas! Cerdas berarti, antara lain, bahwa fungsi yang dipelajari dapat menggeneralisasi dengan baik dari data pelatihan ke data baru yang tidak diketahui dari kumpulan data representatif yang sama. Dengan kata lain itu berarti: kita tidak ingin data yang berlebihan dengan menghafal. Lalu apa yang sebenarnya kita inginkan?

Kami menginginkan fungsi yang mulus dan "meratakan" ruang di antara titik-titik. Kontinuitas dan kemampuan untuk mengambil beberapa turunan akan menjadi persyaratan yang masuk akal. Karena bahkan dengan kondisi ini masih ada banyak fungsi yang membuat kesalahan nol, kita harus membatasi kelas fungsi ini lebih jauh.

9.10 METODE KUADRAT TERKECIL

Pilihan paling sederhana adalah pemetaan linier. Kita mulai dengan jaringan dua lapis (Gambar 9.13) di mana neuron tunggal y dari lapisan kedua menghitung aktivasinya menggunakan

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

dengan f(x) = x. Fakta bahwa kita hanya melihat neuron *output* di sini tidak menimbulkan batasan nyata karena jaringan dua lapis dengan dua atau lebih neuron *output* selalu dapat dipisahkan menjadi jaringan independen dengan neuron input yang identik untuk setiap neuron *output* asli. Bobot dari subnetwork semuanya independen. Menggunakan fungsi sigmoid alih-alih aktivasi linier tidak memberikan keuntungan apa pun di sini karena fungsi sigmoid benar-benar meningkat secara monoton dan tidak mengubah hubungan urutan antara berbagai nilai *output*. Yang diinginkan adalah vektor w yang meminimalkan kesalahan kuadrat

$$E(w) = \sum_{p=1}^{N} (wq^p - t^p)^2 = \sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i q_i^p - t^p\right)^2$$

Sebagai kondisi yang diperlukan untuk minimum fungsi kesalahan ini, semua turunan parsial harus nol. Jadi kita mensyaratkan bahwa untuk j = 1,...,n:

$$\frac{\partial E}{\partial wj} = 2\sum_{p=1}^{N}\left(\sum_{i=1}^{n} w_i q_i^p - t^p\right) q_i^p = 0$$

Mengalikan ini menghasilkan

$$\sum_{p=1}^{N}\left(\sum_{i=1}^{n} w_i q_i^p q_j^p - t^p q_i^p\right) = 0$$

dan menukar hasil penjumlahan dalam sistem persamaan linear

$$\sum_{i=1}^{n} w_i \sum_{p=1}^{N} q_i^p q_j^p = \sum_{p=1}^{N} = t^p \; q_i^p$$

yang dengan

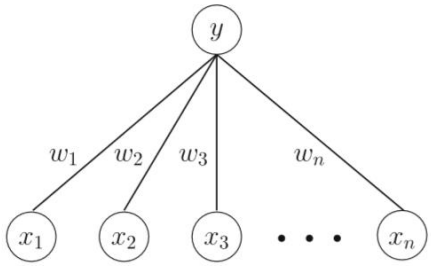
$$Aij = \sum_{p=1}^{N} q_i^p q_j^p \qquad dan \quad bj = \sum_{p=1}^{N} = t^p \; q_j^p$$

dapat dituliskan sebagai persamaan matriks

Aw = b

Yang disebut persamaan normal ini selalu memiliki setidaknya satu solusi dan, jika A dapat dibalik, tepat satu. Selanjutnya, matriks A adalah definit positif, yang berimplikasi bahwa solusi yang ditemukan dalam kasus unik adalah minimum global. Algoritma ini dikenal sebagai metode kuadrat terkecil.

Waktu hitung untuk menyetel matriks bertambah dengan $\Theta(N \cdot n^2)$ dan waktu untuk menyelesaikan sistem persamaan sebagai $O(n^3)$. Metode ini dapat diperluas dengan sangat sederhana untuk menggabungkan beberapa neuron *output* karena, seperti yang telah disebutkan, untuk jaringan umpan maju dua lapis, neuron *output* tidak bergantung satu sama lain.



Gambar 9.13 Jaringan dua lapis dengan neuron *output*

9.11 APLIKASI KE DATA APENDISITIS

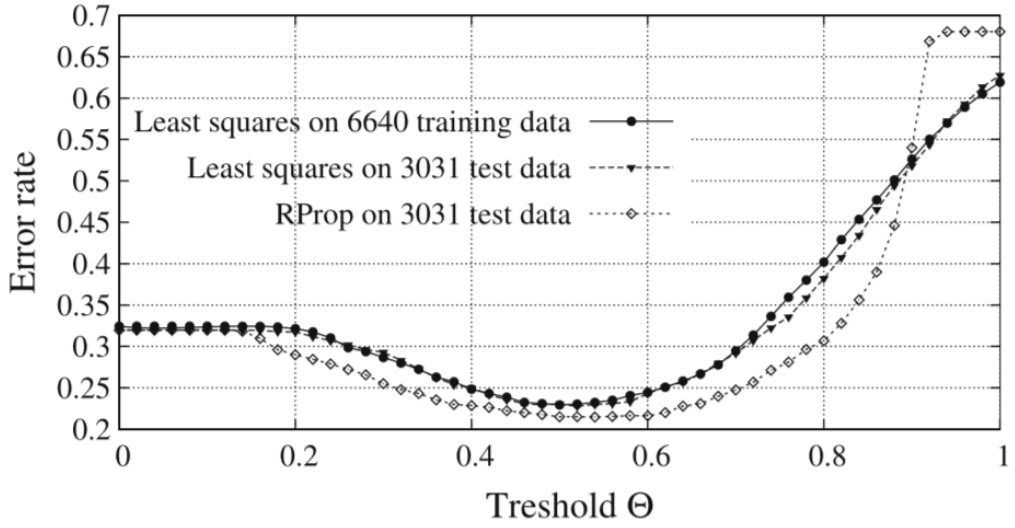
Sebagai aplikasi, kami sekarang menentukan skor linier untuk diagnosis apendisitis. Dari data proyek LEXMED, kami menggunakan metode kuadrat terkecil untuk menentukan lineararmapping dari gejala ke variabel kelas kontinu AppScore dengan nilai dalam interval[0,1]dan mendapatkan kombinasi linear

AppScore = 0.00085 Usia – 0.125 Jenis kelamin + 0.025 P1Q – 0.035 P2Q – 0.021 P3Q
- 0.025 P4Q + 0.12 TensLoc + 0.031 TensGlo + 0.13 Losl + 0.081 Conv
+ 0:0034 RectS
= 0.0027 Taxi + 0.0031 TRec + 0.000021 Leuko – 0.11 Diab – 1.83.

Fungsi ini mengembalikan variabel kontinu untuk AppScore, meskipun variabel kelas biner sebenarnya App hanya mengambil nilai 0 dan 1. Jadi kita harus memutuskan nilai ambang, seperti halnya perceptron. Kesalahan klasifikasi skor sebagai fungsi dari

ambang batas yang tercantum dalam Gambar. 9.14 untuk data pelatihan dan data pengujian. Kita dengan jelas melihat bahwa kedua kurva hampir sama dan memiliki minimum pada H=0,5. Dalam perbedaan kecil dari dua kurva kita melihat bahwa *overfitting* tidak menjadi masalah untuk metode ini karena model menggeneralisasi dari data uji dengan sangat baik.

Juga dalam gambar adalah hasil untuk jaringan RPop tiga lapis nonlinier (Bagian 9.5) dengan kesalahan yang agak lebih rendah untuk nilai ambang antara 0,2 dan 0,9. Untuk penerapan praktis skor turunan dan penentuan ambang batas H yang benar, penting untuk tidak hanya melihat kesalahan, tetapi juga membedakan berdasarkan jenis kesalahan (yaitu positif palsu dan negatif palsu), seperti yang dilakukan dalam aplikasi LEXMED pada Gambar 7.10. Pada kurva



Gambar 9.14 Kesalahan kuadrat terkecil untuk data pelatihan dan pengujian

ROC yang ditunjukkan di sana, skor yang dihitung di sini juga ditampilkan. Kami melihat bahwa model linier sederhana jelas lebih rendah daripada sistem LEXMED. Terbukti, pendekatan linier tidak cukup kuat untuk banyak aplikasi yang kompleks.

9.12 ATURAN DELTA

Kuadrat terkecil adalah, seperti pembelajaran perceptron dan pohon keputusan, yang disebut algoritma pembelajaran batch, sebagai lawan dari pembelajaran inkremental. Dalam pembelajaran batch, semua data pelatihan harus dipelajari dalam satu kali proses. Jika data pelatihan baru ditambahkan, itu tidak bisa begitu saja dipelajari selain apa yang sudah ada. Seluruh proses pembelajaran harus diulang dengan himpunan yang diperbesar. Masalah ini diselesaikan dengan algoritma pembelajaran tambahan, yang dapat mengadaptasi model yang dipelajari untuk setiap contoh baru tambahan. Dalam algoritme yang akan kita lihat dalam diskusi berikut, kami akan secara aditif memperbarui bobot untuk setiap contoh pelatihan baru dengan aturan

$$wj = wj + \Delta wj$$

Untuk menurunkan varian inkremental dari metode kuadrat terkecil, kami mempertimbangkan kembali n turunan parsial yang dihitung di atas dari fungsi kesalahan

$$\frac{\partial E}{\partial wj} = 2\sum_{p=1}^{N}\left(\sum_{i=1}^{n} w_i q_i^p - t^p\right) q_j^p$$

Gradien

$$\nabla \mathrm{E} = \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}$$

sebagai vektor dari semua turunan parsial dari fungsi kesalahan menunjuk ke arah kenaikan terkuat dari fungsi kesalahan dalam ruang n-dimensi dari bobot. Saat mencari minimum, karena itu kami akan mengikuti arah gradien negatif. Sebagai rumus untuk mengubah bobot yang kita peroleh

$$\Delta wj = -\frac{\eta}{2}\frac{\partial E}{\partial w_j} - \eta \sum_{p=1}^{N} \left(\sum_{i=1}^{n} w_i q_i^p - t^p \right) q_j^p$$

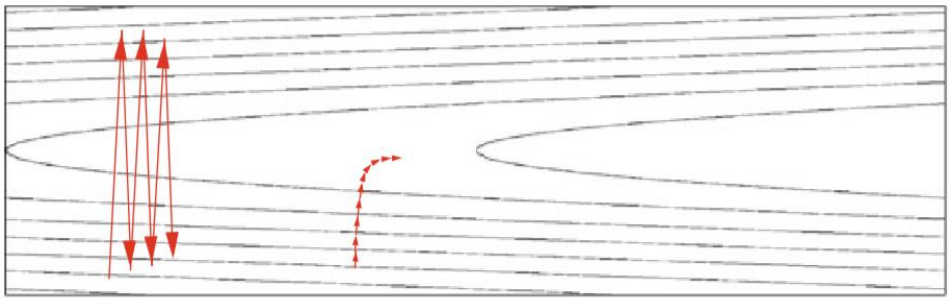
di mana laju pembelajaran adalah konstanta positif yang dapat dipilih secara bebas. yang lebih besar mempercepat konvergensi tetapi pada saat yang sama meningkatkan risiko osilasi di sekitar lembah minimal atau datar. Oleh karena itu, pilihan optimal bukanlah tugas yang sederhana (lihat Gambar 9.15). Sebuah besar, misalnya η - 1, sering digunakan untuk memulai, dan kemudian menyusut perlahan. Dengan mengganti aktivasi

$$y^P = \sum_{i=1}^{n} w_i q_i^p$$

dari neuron *output* untuk contoh pelatihan yang diterapkan qp, rumusnya disederhanakan dan kita memperoleh aturan delta

$$\Delta wj = \eta \sum_{p=1}^{N} (t^p - y^p)\ q_j^p$$

Jadi untuk setiap contoh pelatihan, perbedaan antara target t^p dan output aktual y^p dari jaringan dihitung untuk input q^p yang diberikan. Setelah menjumlahkan semua pola, bobot kemudian diubah secara proporsional dengan jumlah. Algoritma ini ditunjukkan pada Gambar 9.16.



Gambar 9.15 Penurunan gradien untuk besar (kiri) dan yang sangat kecil (kanan) ke dalam lembah yang turun datar ke kanan. Untuk besar ada osilasi di sekitar lembah. Untuk yang terlalu kecil, sebaliknya, konvergensi di di lembah terjadi sangat lambat

DELTALEARNING(*TrainingExamples*, η)
Initialize all weights w_j randomly
Repeat
  $\Delta \boldsymbol{w} = \mathbf{0}$
  For all $(\boldsymbol{q}^p, t^p) \in$ *TrainingExamples*
    Calculate network output $y^p = \boldsymbol{w}^p \boldsymbol{q}^p$
    $\Delta \boldsymbol{w} = \Delta \boldsymbol{w} + \eta(t^p - y^p)\boldsymbol{q}^p$
  $\boldsymbol{w} = \boldsymbol{w} + \Delta \boldsymbol{w}$
Until $\boldsymbol{w}$ converges

Gambar 9.16 Mempelajari jaringan linier dua lapis dengan aturan delta. Perhatikan bahwa perubahan bobot selalu terjadi setelah semua data pelatihan diterapkan

Kami melihat bahwa algoritma tersebut masih belum benar-benar inkremental karena perubahan bobot hanya terjadi setelah semua contoh pelatihan diterapkan satu kali. Kita dapat memperbaiki kekurangan ini dengan secara langsung mengubah bobot (penurunan gradien inkremental) setelah setiap contoh pelatihan (Gambar 9.17), yang sebenarnya bukan lagi implementasi aturan delta yang benar.

9.13 PERBANDINGAN DENGAN PERCEPTRON

Aturan pembelajaran untuk perceptron , metode kuadrat terkecil, dan aturan delta dapat digunakan untuk menghasilkan fungsi linier dari data. Namun untuk perceptron, berbeda dengan metode lain, pengklasifikasi untuk kelas yang dapat dipisahkan secara linier dipelajari melalui keputusan ambang batas. Dua metode lainnya, bagaimanapun, menghasilkan pendekatan linier ke data. Pengklasifikasi dapat dihasilkan dari pemetaan linier, jika diinginkan, dengan penerapan fungsi ambang batas.

DELTALEARNINGINCREMENTAL(*TrainingExamples*, η)
Initialize all weights w_j randomly
Repeat
  For all $(\boldsymbol{q}^p, t^p) \in$ *TrainingExamples*
    Calculate network output $y^p = \boldsymbol{w}^p \boldsymbol{q}^p$
    $\boldsymbol{w} = \boldsymbol{w} + \eta(t^p - y^p)\boldsymbol{q}^p$
Until $\boldsymbol{w}$ converges

Gambar 9.17 Varian tambahan dari aturan delta

Perceptron dan aturan delta adalah algoritma iteratif dimana waktu sampai konvergensi sangat bergantung pada data. Dalam kasus data yang dapat dipisahkan secara linier, batas atas jumlah langkah iterasi dapat ditemukan untuk perceptron. Untuk aturan delta, sebaliknya, hanya ada jaminan konvergensi asimtotik tanpa batas.

Untuk kuadrat terkecil, pembelajaran terdiri dari menyiapkan dan memecahkan sistem persamaan linear untuk vektor bobot. Dengan demikian ada batasan keras pada waktu komputasi. Karena itu, metode kuadrat terkecil selalu lebih disukai ketika pembelajaran tambahan tidak diperlukan.

9.14 ALGORITMA *BACKPROPAGATION*

Dengan algoritma *backpropagation*, kami sekarang memperkenalkan model saraf yang paling banyak digunakan. Alasan penggunaannya yang luas keserbagunaan universalnya untuk tugas-tugas perkiraan yang sewenang-wenang. Algoritma berasal langsung dari aturan delta tambahan. Berbeda dengan aturan delta, aturan ini menerapkan fungsi sigmoid nonlinier pada jumlah bobot input sebagai fungsi aktivasinya. Selanjutnya, jaringan *backpropagation* dapat memiliki lebih dari dua lapisan neuron.

Pada Gambar 9.18 jaringan propagasi balik yang khas dengan lapisan input, lapisan tersembunyi, dan lapisan output ditampilkan. Karena nilai *output* saat ini x^p_j dari neuron lapisan *output* dibandingkan dengan nilai *output* target t^p_j , ini digambar sejajar satu sama lain. Selain neuron input, semua neuron menghitung nilai saat ini xj dengan aturan

$$x_j = f\left(\sum_{i=1}^{n} w_{ji} xi\right)$$

dimana n adalah jumlah neuron pada lapisan sebelumnya. Kami menggunakan fungsi sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

Analog dengan aturan delta tambahan, bobot diubah sebanding dengan gradien negatif dari fungsi kesalahan kuadrat yang dijumlahkan di atas neuron *output*

$$E_p = \frac{1}{2} \sum_{k \in output} (t_k^p - x_k^p)^2$$

untuk pola pelatihan p:

$$\Delta p\, wji = -\eta \frac{\partial Ep}{\partial w_j}$$

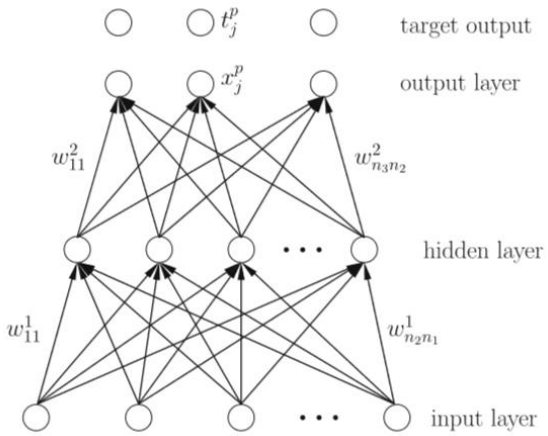
Untuk menurunkan aturan pembelajaran, ekspresi di atas diganti dengan Ep. Dalam ekspresi, x_k digantikan oleh (9.14). Dalam persamaan, output x_i dari neuron berikutnya, lapisan yang lebih dalam terjadi secara rekursif, dll. Dengan beberapa aplikasi aturan rantai (lihat [RHR86] atau [Zel94]) kami mendapatkan aturan pembelajaran propagasi balik

$$\delta_j^p = \begin{cases} x_j^p(1-x_j^p)(t_j^p - x_j^p) & jika \text{ j } adalah\ output\ neuron \\ x_j^p(1-x_j^p)\sum_k \delta_k^p w_{kj} \end{cases}$$

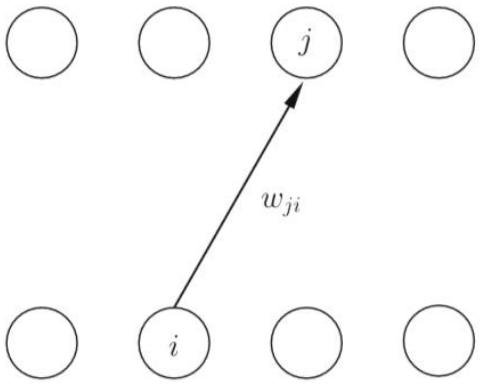
yang juga dilambangkan dengan aturan delta umum. Untuk semua neuron, rumus untuk mengubah bobot w_{ji} dari neuron i ke neuron j (lihat Gambar 9.19) mengandung, seperti aturan Hebb, sebuah istilah $\eta\, x_i^p - x_j^p$. Faktor baru $1 - x_j^p$ menciptakan simetri, yang hilang dari aturan Hebb, antara aktivasi 0 dan 1 neuron j. Untuk neuron *output*, faktor δ_j^p jÞ menangani perubahan bobot yang sebanding dengan kesalahan. Untuk neuron tersembunyi, nilai dp j neuron j dihitung secara rekursif dari semua perubahan δ_k^p neuron tingkat berikutnya yang lebih tinggi.

Keseluruhan pelaksanaan proses pembelajaran ditunjukkan pada Gambar 9.20. Setelah menghitung output jaringan (propagasi maju) untuk contoh pelatihan, kesalahan perkiraan dihitung. Ini kemudian digunakan selama propagasi mundur untuk mengubah bobot mundur dari lapisan ke lapisan. Seluruh proses kemudian diterapkan ke semua contoh pelatihan dan diulang sampai bobot tidak lagi berubah atau batas waktu tercapai.
Jika kita membangun jaringan dengan setidaknya satu lapisan tersembunyi, pemetaan nonlinier dapat dipelajari. Tanpa lapisan tersembunyi, neuron output tidak lebih kuat dari neuron linier, meskipun fungsi sigmoid. Alasan untuk ini adalah bahwa fungsi sigmoid

sangat monoton. Hal yang sama berlaku untuk jaringan multi-layer yang hanya menggunakan fungsi linier sebagai fungsi aktivasi, misalnya fungsi identitas. Ini karena eksekusi berantai dari pemetaan linier adalah linier secara agregat. Sama seperti perceptron, kelas fungsi yang dapat direpresentasikan juga diperbesar jika kita menggunakan variabel fungsi sigmoid.



Gambar 9.18 Jaringan backpropagation tiga lapis dengan n_1 neuron di lapisan pertama, n_2 neuron di lapisan kedua, dan n_3 neuron di lapisan ketiga



Gambar 9.19 Penunjukan neuron dan bobot untuk penerapan aturan backpropagation

$$f(x) = \frac{1}{1 + e^{-(x - \Theta)}}$$

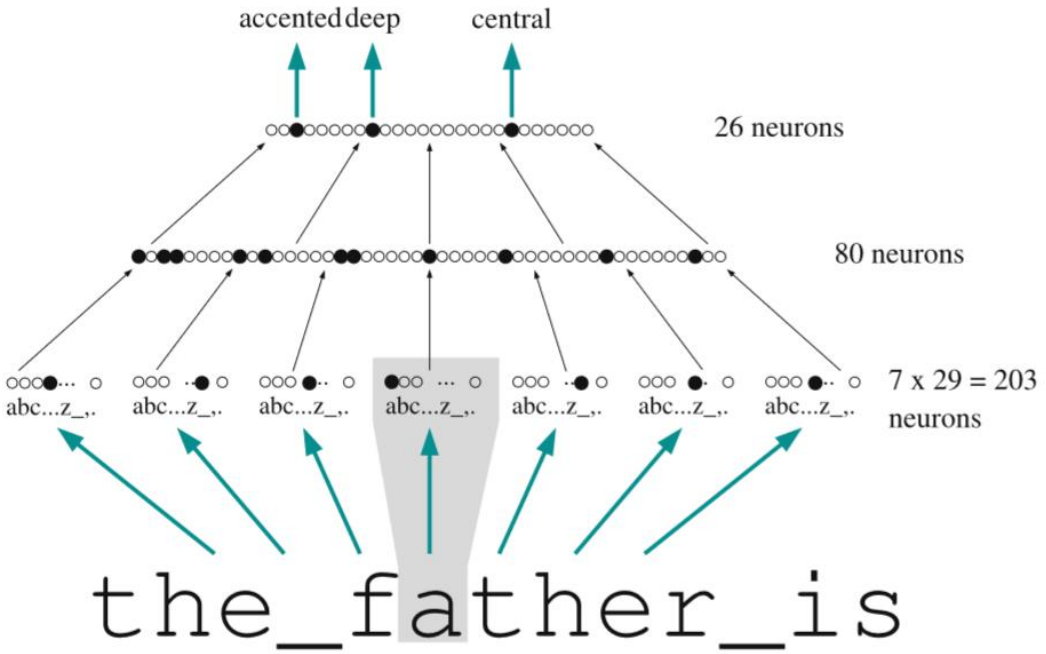
dengan ambang Θ. Ini diimplementasikan secara analog, dimana neuron yang aktivasinya selalu bernilai satu dan terhubung dengan neuron pada level tertinggi berikutnya dimasukkan ke dalam input layer dan ke dalam setiap hidden layer. Bobot koneksi ini dipelajari secara normal dan mewakili ambang Θ dari neuron penerus.

BACKPROPAGATION(*TrainingExamples*, η)
Initialize all weights w_j to random values
Repeat
  For all $(q^p, t^p) \in$ *TrainingExamples*
    1. **Apply the query vector** q^p to the input layer
    2. **Forward propagation**:
      For all layers from the first hidden layer upward
        For each neuron of the layer
          Calculate activation $x_j = f(\sum_{i=1}^{n} w_{ji} x_i)$
    3. **Calculation of the square error** $E_p(w)$
    4. **Backward propagation**:
      For all levels of weights from the last downward
        For each weight w_{ji}
          $w_{ji} = w_{ji} + \eta \delta_j^p x_i^p$
Until w converges or time limit is reached

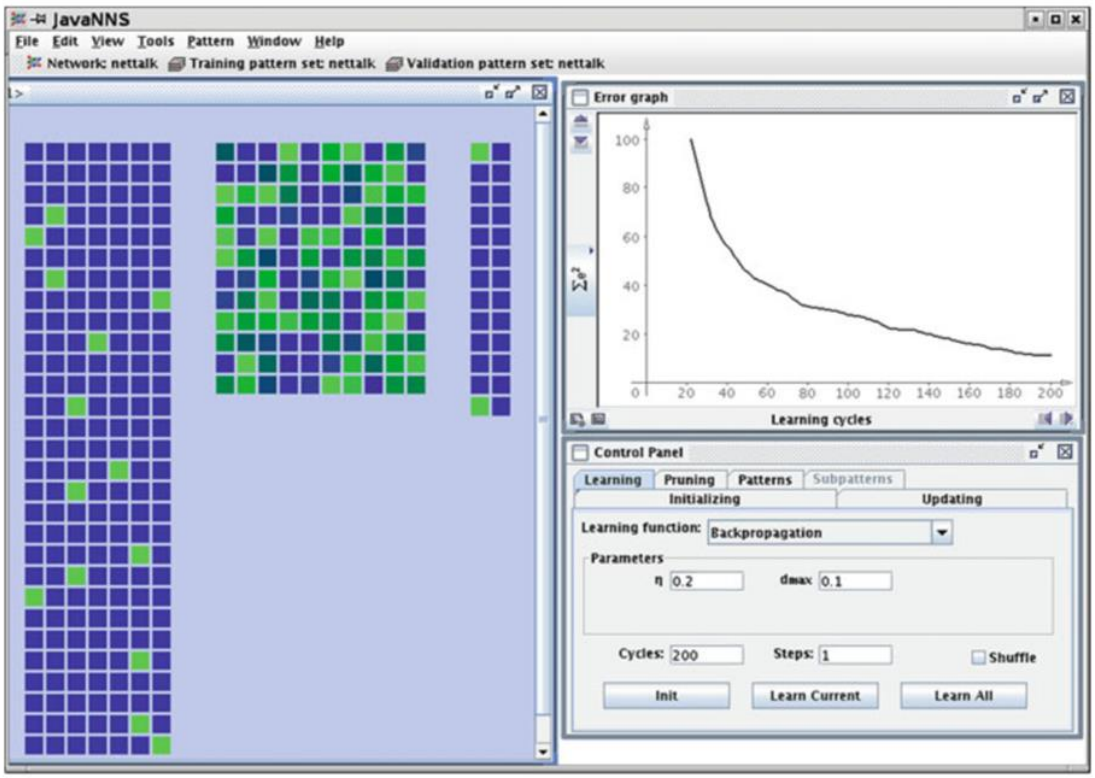
Gambar 9.20 Algoritma backpropagation

9.15 NETTALK: JARINGAN BELAJAR BERBICARA

Sejnowski dan Rosenberg mendemonstrasikan dengan sangat mengesankan pada tahun 1986 apa yang mampu dilakukan oleh backpropagation. Mereka membangun sebuah sistem yang dapat dimengerti untuk membaca teks bahasa Inggris dengan keras dari file teks. Arsitektur jaringan yang ditunjukkan pada Gambar 9.21 terdiri dari lapisan input dengan 7 x 29 = 203 neuron di mana huruf saat ini dan tiga huruf sebelumnya, serta tiga huruf berikutnya dikodekan. Untuk masing-masing dari tujuh huruf ini, 29 neuron dicadangkan untuk karakter "a...z ,.". Input dipetakan ke 26 neuron output lebih dari 80 neuron tersembunyi, yang masing-masing mewakili fonem tertentu. Misalnya, "a" dalam "ayah" diucapkan dalam, beraksen, dan sentral. Jaringan dilatih dengan 1.000 kata, yang diterapkan secara acak satu demi satu huruf demi huruf. Untuk setiap huruf, output target diberikan secara manual untuk intonasinya. Untuk menerjemahkan atribut intonasi menjadi suara yang sebenarnya, bagian dari sistem sintesis ucapan DECtalk digunakan. Melalui interkonektivitas lengkap, jaringan berisi total 203 x 80 + 80 x 26 = 18320 bobot.



Gambar 9.21 Jaringan NETtalk memetakan teks ke atribut pengucapannya



Gambar 9.22 Jaringan NETtalk di JNNS.

Di jendela kiri dari kiri ke kanan kita melihat 7 . 29 neuron input, 80 tersembunyi, dan 26 neuron output. Karena jumlahnya yang besar, bobot dihilangkan dari jaringan. Kanan atas adalah kurva pembelajaran yang menunjukkan perkembangan kesalahan kuadrat dari waktu ke waktu.

Sistem dilatih menggunakan simulator pada VAX 780 dengan sekitar 50 siklus untuk semua kata. Jadi, pada sekitar 5 karakter per kata rata-rata, diperlukan sekitar 5 . 50 . 1000 = 250.000 iterasi dari algoritma backpropagation. Dengan kecepatan sekitar satu karakter

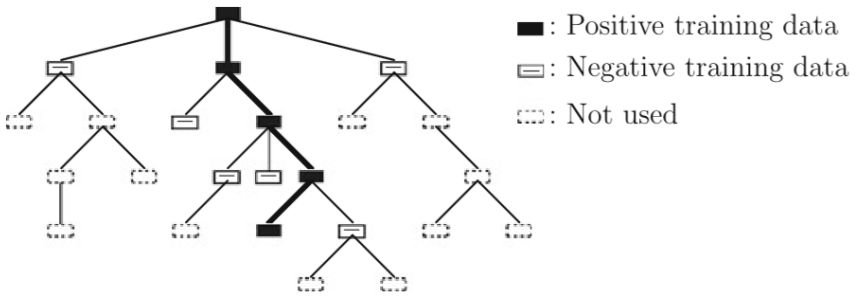
per detik, ini berarti kira-kira 69 jam waktu komputasi. Para pengembang mengamati banyak properti dari sistem yang sangat mirip dengan pembelajaran manusia. Pada awalnya sistem hanya dapat berbicara dengan tidak jelas atau menggunakan kata-kata sederhana. Seiring waktu itu terus membaik dan akhirnya mencapai 95% kebenaran huruf yang diucapkan.

Untuk eksperimen visual dengan jaringan saraf, disarankan menggunakan Java Neural Network Simulator (JNNS). Jaringan NETtalk, dimuat dan dilatih dengan JNNS, ditunjukkan pada Gambar 9.22.

9.16 PEMBELAJARAN HEURISTIK UNTUK PEMBUKTI TEOREMA

Dalam Bab. 6, algoritma untuk pencarian heuristik, seperti algoritma A⋆ dan algoritma IDA⋆ dibahas. Untuk mencapai pengurangan ruang pencarian yang signifikan, diperlukan heuristik yang baik untuk implementasi algoritma ini. Masalah ruang pencarian yang meledak selama pencarian pembuktian teorema dibuktikan. Masalah ini disebabkan oleh banyaknya kemungkinan langkah inferensi di setiap langkah.

Kami sekarang mencoba membangun modul pengontrol bukti heuristik yang mengevaluasi berbagai alternatif untuk langkah berikutnya dan kemudian memilih alternatif dengan peringkat terbaik. Dalam kasus resolusi, peringkat klausa yang tersedia dapat dilakukan dengan fungsi yang, berdasarkan atribut klausa tertentu seperti jumlah literal, kompleksitas istilah, dll., Menghitung nilai untuk setiap pasangan yang dapat dipecahkan. Seperti heuristik dipelajari untuk teorema SETHEO menggunakan backpropagation. Saat mempelajari heuristik, atribut klausa saat ini disimpan sebagai data pelatihan untuk kelas positif untuk setiap bukti yang ditemukan di setiap cabang selama pencarian. Di semua cabang yang tidak mengarah ke bukti, atribut klausa saat ini disimpan sebagai data pelatihan untuk kelas negatif. Pohon seperti itu, dengan jalur yang berhasil dan peringkat simpul yang sesuai, digambarkan pada Gambar 9.23.



Gambar 9.23 Cari pohon dengan jalur sukses yang simpulnya dievaluasi sebagai positif, dan cabang yang tidak berhasil dinilai negatif

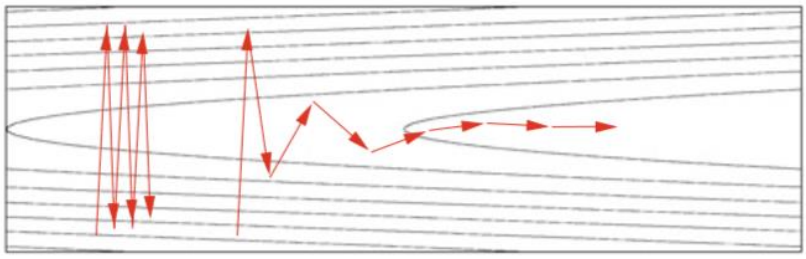
Sebuah jaringan backpropagation dilatih pada data ini dan kemudian digunakan untuk mengevaluasi klausa dalam peribahasa. Untuk setiap klausa, 16 atribut numerik dihitung, dinormalisasi, dan dikodekan dalam neuron input. Jaringan dilatih dengan 25 neuron tersembunyi dan satu neuron output untuk kelas (positif/negatif).

Ditunjukkan bahwa jumlah inferensi yang dicoba dapat dikurangi dengan banyak urutan besarnya untuk masalah yang sulit, yang pada akhirnya mengurangi waktu komputasi dari jam ke detik. Dengan demikian menjadi mungkin untuk membuktikan teorema yang, tanpa heuristik, berada di luar jangkauan.

Masalah dan Perbaikan

Backpropagation sekarang berusia 25 tahun dan telah membuktikan dirinya dalam berbagai aplikasi, misalnya dalam pengenalan pola dan robotika. Namun, penerapannya terkadang bermasalah. Terutama ketika jaringan memiliki ribuan bobot dan ada banyak data pelatihan untuk dipelajari, dua masalah muncul:

Jaringan sering konvergen ke minimum lokal dari fungsi kesalahan. Selanjutnya, backpropagation sering menyatu dengan sangat lambat. Ini berarti bahwa banyak iterasi pada semua pola pelatihan diperlukan. Banyak perbaikan telah disarankan untuk mengatasi masalah ini. Seperti yang disebutkan sebelumnya, osilasi dapat dihindari dengan mengurangi kecepatan belajar η secara perlahan seperti yang ditunjukkan pada Gambar 9.15.



Gambar 9.24 Perubahan arah yang tiba-tiba diperhalus dengan menggunakan istilah momentum. Iterasi tanpa suku momentum (kiri) dibandingkan dengan iterasi dengan suku momentum (kanan)

Metode lain untuk mengurangi osilasi adalah penggunaan istilah momentum saat memperbarui bobot, yang memastikan bahwa arah penurunan gradien tidak berubah terlalu dramatis dari satu langkah ke langkah berikutnya. Di sini untuk perubahan bobot saat ini $\Delta_p w_{ji}$ (t) pada waktu t bagian lain dari perubahan $\Delta_p w_{ji}$ (t - 1) dari langkah sebelumnya ditambahkan. Aturan belajar kemudian berubah menjadi

$$\Delta_p w_{ji}(t) = \eta \delta_j^p x^p_j + \gamma \Delta_p w_{ji}(t-1)$$

dengan parameter γ antara nol dan satu. Hal ini digambarkan dalam contoh dua dimensi pada Gambar 9.24. Ide lain adalah meminimalkan fungsi kesalahan linier daripada fungsi kesalahan kuadrat, yang mengurangi masalah konvergensi lambat menjadi lembah datar. Penurunan gradien dalam backpropagation pada akhirnya didasarkan pada pendekatan linier dari fungsi kesalahan. Quickprop, sebuah algoritma yang menggunakan pendekatan kuadrat untuk fungsi kesalahan dan dengan demikian mencapai konvergensi lebih cepat, diciptakan oleh Scott Fahlmann.

Melalui penyatuan cerdas dari peningkatan yang disebutkan dan trik heuristik lainnya, Martin Riedmiller mencapai optimasi lebih lanjut dengan algoritma RProp [RB93].Prophasmenggantikan propagasi mundur dan keadaan baru dari algoritma aproksimasi jaringan saraf umpan maju. Kami menerapkan RProp pada klasifikasi data apendisitis dan mencapai kesalahan yang kira-kira berukuran sama dengan pohon keputusan yang dipelajari.

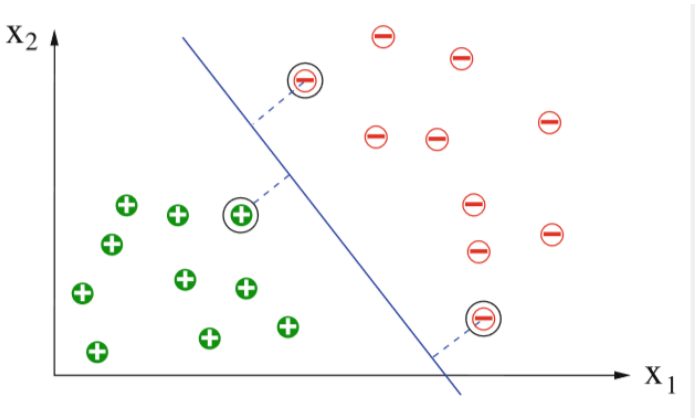
9.17 MENDUKUNG MESIN VEKTOR

Jaringan saraf feedforward dengan hanya satu lapisan bobot bersifat linier. Linearitas mengarah ke jaringan sederhana dan pembelajaran cepat dengan konvergensi yang terjamin. Selanjutnya, bahaya *overfitting* kecil untuk model linier. Namun, untuk banyak aplikasi, model linier tidak cukup kuat, misalnya karena kelas yang relevan tidak dapat dipisahkan secara linier. Di sini jaringan multi-layer seperti backpropagation mulai

digunakan, dengan konsekuensi bahwa minima lokal, masalah konvergensi, dan *overfitting* dapat terjadi.

Pendekatan yang menjanjikan, yang menyatukan keunggulan model linier dan nonlinier, mengikuti teori mesin vektor pendukung (SVM), yang secara kasar akan kami uraikan menggunakan masalah dua kelas.[51]

Dalam kasus dua kelas yang dapat dipisahkan secara linier, mudah untuk menemukan bidang hiper yang membagi, misalnya dengan aturan pembelajaran perceptron. Namun, biasanya ada tak terhingga banyak bidang seperti itu, seperti pada contoh dua dimensi pada Gambar 9.25. Kami mencari pesawat yang memiliki jarak minimum terbesar ke kedua kelas. Bidang ini biasanya secara unik didefinisikan oleh beberapa titik di daerah perbatasan. Titik-titik ini, yang disebut vektor pendukung, semuanya memiliki jarak yang sama ke garis pemisah. Untuk menemukan vektor pendukung, ada algoritma pengoptimalan yang efisien. Menariknya, hyperplane pembagi yang optimal ditentukan oleh beberapa parameter, yaitu oleh support vector. Jadi bahaya *overfitting* kecil.



Gambar 9.25 Dua kelas dengan garis pemisah maksimal. Titik yang dilingkari adalah vektor pendukung

Mesin vektor pendukung menerapkan algoritme ini untuk masalah yang tidak dapat dipisahkan secara linier dalam proses dua langkah: Pada langkah pertama, transformasi nonlinier diterapkan pada data, dengan properti bahwa data yang ditransformasikan dapat dipisahkan secara linier. Pada langkah kedua, vektor pendukung kemudian ditentukan dalam ruang yang diubah.

Langkah pertama sangat menarik, tetapi tidak cukup sederhana. Nyatanya, selalu mungkin untuk membuat kelas-kelas dapat dipisahkan secara linier dengan mentransformasikan ruang vektor, selama data tersebut tidak mengandung kontradiksi.[52] Pemisahan seperti itu dapat dicapai misalnya dengan memasukkan dimensi baru (n + 1) dan definisi

$$x_{n-1} = \begin{cases} 1 & if\ x\ \in kelas\ 1 \\ 0 & if\ x\ \in kela\ 0 \end{cases}$$

Namun, rumus ini tidak banyak membantu karena tidak dapat diterapkan pada titik-titik baru dari kelas yang tidak diketahui yang akan diklasifikasikan. Oleh karena itu, kita membutuhkan transformasi umum yang se-independen mungkin dari data saat ini. Dapat

[51] Dukungan mesin vektor bukan jaringan saraf. Namun, karena perkembangan historis dan hubungan matematisnya dengan jaringan linier, mereka dibahas di sini.

[52] Sebuah titik data kontradiktif jika milik kedua kelas.

ditunjukkan bahwa ada transformasi generik seperti itu bahkan untuk batas pembagian kelas yang berbentuk sewenang-wenang dalam ruang vektor asli. Dalam ruang yang diubah, data kemudian dipisahkan secara linier. Namun, jumlah dimensi ruang vektor baru tumbuh secara eksponensial dengan jumlah dimensi ruang vektor asli. Namun, banyaknya dimensi baru tidak begitu bermasalah karena, ketika menggunakan vektor pendukung, bidang pembagi, seperti disebutkan di atas, ditentukan hanya oleh beberapa parameter.

Transformasi nonlinier pusat dari ruang vektor disebut kernel, karena itu mesin vektor pendukung juga dikenal sebagai metode kernel. Teori SVM asli yang dikembangkan untuk tugas-tugas klasifikasi telah diperluas dan sekarang dapat digunakan pada masalah regresi juga. Matematika yang digunakan di sini sangat menarik, tetapi terlalu luas untuk pengenalan awal.

9.18 DEEP *LEARNING*

Dalam Bab. 8 dan 9 kita melihat bahwa ada banyak algoritma pembelajaran yang baik saat ini yang mampu mempelajari klasifikasi atau aproksimasi non-sepele, kadang-kadang kompleks untuk semua jenis aplikasi, seperti diagnosis dan prognosis berdasarkan input sensor. Kami juga melihat bahwa hingga saat ini, pembuatan fitur belum berhasil. Alih-alih, tugas seorang ilmuwan data adalah menemukan sekumpulan fitur kecil yang masuk akal, yang kemudian dapat digunakan sebagai input untuk algoritme pembelajaran. Mengapa kita tidak menggunakan semua data sensor yang tersedia, yaitu gambar langsung dunia, sebagai input? Misalnya, untuk pengenalan objek dalam foto, kita dapat menggunakan sepuluh juta piksel sebagai vektor input, yang memiliki panjang 30 juta (dalam kasus gambar RGB atau HSV, yang masing-masing memiliki tiga nilai warna). Apa masalahnya dengan pendekatan ini? Jawabannya dikenal sebagai "kutukan dimensi". Ini berarti, antara lain, waktu pelatihan tumbuh sangat cepat, seringkali secara eksponensial, dengan dimensi data input. Untuk menjaga waktu komputasi dalam batas, karena itu pertama-tama kita harus mengurangi data input menjadi vektor fitur pendek. Seperti dijelaskan di atas, pemetaan ini biasanya dibuat secara manual. Namun, untuk banyak aplikasi, seperti klasifikasi objek dalam gambar, sulit atau bahkan tidak mungkin untuk menemukan rumus fitur secara manual. Salah satu metode lama untuk reduksi otomatis dimensi adalah analisis komponen utama (PCA), yang menentukan arah varians tertinggi (yaitu komponen utama) dalam ruang vektor data pelatihan dan memproyeksikan data ke dalam subruang komponen utama menggunakan linear transformasi. Karena ketidakjelasan yang hilang dari pemetaan kompresi, PCA tidak sekuat metode baru yang dijelaskan di bawah ini.

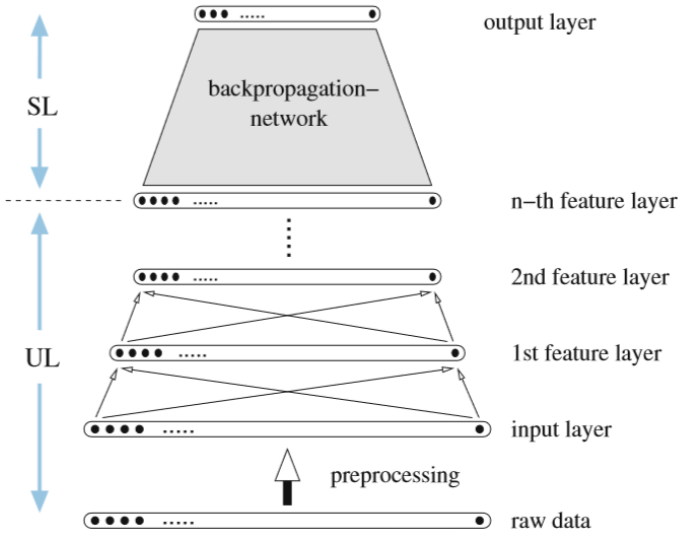
Sejak sekitar tahun 1995 pekerjaan telah dilakukan pada pembelajaran yang mendalam, kelas algoritma yang sangat menjanjikan untuk memecahkan masalah ini, dan sekarang ada port penyimpanan sukses yang mengesankan. Pembelajaran mendalam mencakup metode seperti jaringan saraf convolutional (CNN), atau jaringan kepercayaan mendalam dan variasinya. Arsitektur jaringan saraf berlapis-lapis dengan hingga dua puluh atau lebih lapisan merupakan bagian yang sangat kompleks dan tidak dapat dijelaskan secara rinci di sini.

Pengenalan pola sederhana dalam ruang dimensi rendah atau dalam kasus klasifikasi ketika kelas dipisahkan secara linier. Namun, untuk kelas yang tidak dapat dipisahkan secara linier dalam ruang berdimensi tinggi, masalah muncul karena di sini pembelajaran menimbulkan masalah optimasi nonlinier. Pada prinsipnya ada solusi menggunakan algoritma gradient descent seperti backpropagation. Namun, masalah konvergensi dan waktu komputasi yang sangat tinggi muncul untuk algoritma klasik,

terutama ketika jaringan dengan banyak lapisan tersembunyi digunakan. Jadi metode lain telah dicari.

Alam sebagai Contoh

Semua pendekatan yang berhasil dalam pembelajaran mendalam hingga saat ini bekerja dengan banyak lapisan neuron. Jaringan dibagi menjadi dua bagian, mirip dengan jaringan feedforward yang ditunjukkan secara skematis pada Gambar 9.26. Setelah lapisan preprocessing ada beberapa lapisan yang dilatih sebelumnya oleh unsupervised learning (UL). Setiap lapisan dalam jaringan UL ini mewakili fitur pola input. Semakin rendah lapisannya, semakin sederhana fiturnya. Dalam pengenalan objek dalam foto, fitur lapisan bawah biasanya mewakili tepi atau garis dalam orientasi yang berbeda.[53] Fitur kompleks seperti keberadaan wajah dapat terbentuk pada lapisan yang lebih tinggi. Arsitektur ini menunjukkan kesamaan tertentu dengan struktur otak manusia dan hewan. Mulai dari alat indera, misalnya mata, otak dibangun dalam banyak lapisan, dan semakin tinggi lapisannya, semakin abstrak informasi yang dapat ditemukan di sana. Namun, masih terlalu sedikit yang diketahui tentang bagaimana jaringan saraf bekerja di alam yang mengarah pada manfaat yang signifikan dalam pembelajaran mendalam.



Gambar 9.26 Arsitektur sederhana dari jaringan pembelajaran mendalam.

Ini terdiri dari preprocessor, beberapa lapisan (dalam hal ini dua) untuk deteksi fitur tanpa pengawasan dan kemudian satu jaringan saraf klasik, yang, dalam contoh ini, dilatih oleh backpropagation

Terlampir ke jaringan UL adalah jaringan pembelajaran terawasi klasik (SL), yang dapat dilatih dengan backpropagation atau RProp. Dengan demikian proses pembelajaran berjalan sebagai berikut:

1. Pelatihan tanpa pengawasan untuk semua bobot lapisan fitur.
2. Pelatihan jaringan SL yang diawasi dengan penurunan gradien.

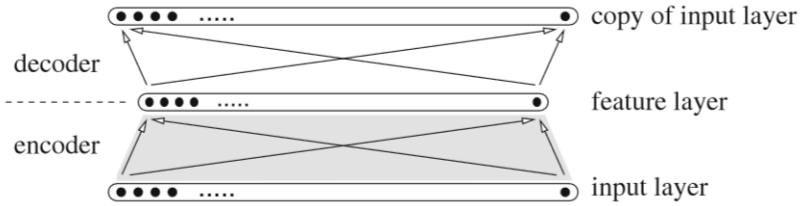
Pembelajaran tanpa pengawasan dari bobot ke lapisan fitur masih harus dijelaskan. Dengan mempelajari bobot ini, fitur akan diekstraksi. Ekstraksi fitur harus memiliki properti yang memasukkan data yang dipetakan ke dalam ruang dimensi yang lebih rendah, jika memungkinkan tanpa (terlalu banyak) kehilangan informasi. Oleh karena itu, seseorang

[53] Representasi visual dari fitur tepi tersebut, termasuk penjelasan, dapat ditemukan di deeplearning.stanford.edu

dapat melihat ekstraksi fitur sebagai bentuk kompresi. Kami sekarang menguraikan salah satu dari banyak kemungkinan algoritme untuk pembelajaran fitur.

9.19 *AUTOENCODER DENOISING* BERTUMPUK

Bobot lapisan fitur ditentukan oleh algoritma yang sederhana dan jelas secara fundamental yang disebut stacked denoising autoencoder, seperti yang ditunjukkan pada Gambar 9.27. Untuk melatih lapisan tersembunyi pertama*, autoencoder* dilatih dengan metode pembelajaran terawasi, misalnya RProp. Tugasnya adalah mempelajari pemetaan identitas semua vektor input x ke dirinya sendiri. Lapisan fitur bertindak sebagai lapisan tersembunyi. Untuk memastikan bahwa fitur terbentuk di lapisan ini, penting untuk menghindari *overfitting*. Pendekatan klasik akan menggunakan validasi silang atas jumlah neuron tersembunyi di lapisan fitur. Itu akan menyebabkan sejumlah kecil neuron tersembunyi dan dengan demikian beberapa fitur. Telah ditunjukkan bahwa pengkodean fitur yang dikompresi seperti itu tidak optimal. Apa yang disebut pengkodean jarang mengarah ke hasil yang lebih baik.



Gambar 9.27 *Autoencoder* untuk mempelajari bobot lapisan fitur. Jaringan *backpropagation* klasik dapat digunakan di sini untuk mempelajari pemetaan identitas dari lapisan input saat ini ke dirinya sendiri. Fitur terbentuk di lapisan tersembunyi selama pembelajaran

Untuk menemukan banyak fitur yang berbeda melalui pengkodean yang jarang, denoising, daripada validasi silang, akan digunakan. Selama setiap siklus pembelajaran, nilai beberapa neuron input diubah secara acak, misalnya dengan noise Gaussian atau dengan mengatur aktivasi ke 0 atau 1. Jarak

$$||y - x||$$

dari vektor output yang dihitung y dari vektor input asli x digunakan sebagai fungsi kesalahan untuk perubahan bobot. Jadi encoderis dilatih untuk menjadi kuat terhadap kebisingan. Selama pembelajaran, sebuah parameter variabel memberi bobot pada neuron-neuron yang terganggu lebih kuat daripada yang lain, sehingga dapat mengoptimalkan efek kebisingan yang diinginkan pada pembelajaran.

Setelah *autoencoder* dilatih, lapisan decoder yang sebenarnya tidak dibutuhkan, yaitu lapisan kedua dari bobot, dihilangkan. Lapisan pertama dibekukan dan digunakan untuk menghitung fitur dalam jaringan UL pada Gambar 9.26. Kemudian, dengan lapisan bobot pertama yang tetap ini, lapisan fitur kedua dilatih dengan algoritma *autoencoder,* dibekukan, dan seterusnya hingga lapisan fitur terakhir. Dengan demikian, bagian pembelajaran yang tidak diawasi telah selesai.

Sekarang bagian SL dari jaringan dilatih dengan algoritma pembelajaran terawasi klasik. Untuk setiap contoh pelatihan, *output* dari jaringan UL, yaitu, lapisan fitur terakhir digunakan sebagai *input*, dan label data tertentu adalah output target. Selama *backpropagation,* bobot lapisan fitur dapat dilatih lagi untuk tujuan *fine-tuning*, atau mereka dapat dibiarkan tidak berubah.

Kami belum menyebutkan langkah pra-pemrosesan, yang berbeda tergantung pada aplikasinya. Seringkali semua variabel input dinormalisasi. Untuk foto, gambar piksel sering diubah menjadi ruang berdimensi lebih rendah, mirip dengan PCA, menggunakan proses yang disebut pemutihan, dengan tujuan membuat fitur yang dihasilkan tidak terlalu berlebihan. Di sini orang juga bisa langsung menggunakan PCA.

Metode lain

Selain autoencoder denoising bertumpuk, jaringan saraf convolutional yang disebutkan sebelumnya memainkan peran penting. Untuk mengurangi kerumitan dan menghemat waktu, lapisan fitur tidak sepenuhnya terhubung, melainkan setiap neuron fitur menahan input hanya dari beberapa neuron di lapisan di bawahnya. Lapisan juga bergantian antara lapisan konvolusi dan penyatuan. Dalam setiap lapisan konvolusi, filter linier yang dapat dilatih digunakan pada neuron input, dan pada lapisan penyatuan, fungsi rata-rata, maksimum atau lebih kompleks dihitung dari neuron input. Juga cukup populer adalah jaringan kepercayaan mendalam, yang menggunakan mesin Boltzmann terbatas untuk belajar.

Penemuan fitur dengan jaringan UL dapat sepenuhnya diganti dengan pengelompokan, di mana, untuk setiap cluster yang ditemukan, fitur biner menentukan apakah suatu titik termasuk dalam cluster tertentu. Seseorang juga dapat menggunakan kernel PCA, generalisasi nonlinier dari PCA, untuk mempelajari fitur. Seperti disebutkan sebelumnya, jaringan SL yang terhubung sepenuhnya juga dapat digantikan oleh algoritma pembelajaran lainnya, misalnya dengan mesin vektor dukungan.

9.20 SISTEM DAN IMPLEMENTASI

Bahkan implementasi sistem pembelajaran mendalam terbaik saat ini sangat intensif secara komputasi. Penyebab dari waktu komputasi yang lama adalah ukuran input layer dan banyaknya jumlah layer dalam jaringan. Efek ini diperkuat lebih lanjut jika, dalam kasus lapisan input yang besar, data pelatihan adalah elemen dari ruang vektor berdimensi tinggi. Untuk mewakili kelas yang terlatih dengan baik, diperlukan banyak vektor data, yang membuat waktu komputasi menjadi lebih lama.

Ini berarti bahwa latihan lari dapat berlangsung dari menit hingga hari. Selain itu, parameter sistem dari jaringan yang kompleks harus dikonfigurasi, yang pada gilirannya akan berdampak besar pada kualitas hasil. Seperti yang kita ketahui dari Bab. 8, metaparameter optimal untuk algoritma pembelajaran dapat ditemukan dengan validasi silang, yaitu dengan mencoba semua kombinasi nilai. Karena kompleksitas algoritma pembelajaran yang mendalam, ada banyak parameter, dan kumpulan kombinasi parameter tumbuh secara eksponensial dengan jumlah parameter. Oleh karena itu, penerapan validasi silang yang naif tidak praktis. Sebagai gantinya, algoritme digunakan yang mencari titik di ruang metaparameter sistem yang meminimalkan kesalahan pada data validasi. Contoh parametermeta tersebut adalah jumlah lapisan di jaringan UL serta jaringan SL, jumlah neuron di lapisan individu, tingkat pembelajaran, dan tingkat keterkaitan untuk jaringan CNN. Algoritme untuk optimasi metaparameter menggunakan, misalnya, pencarian acak atau penurunan gradien dalam ruang metaparameter [MDA15].

Jadi, untuk kumpulan data berdimensi tinggi, PC sederhana kewalahan. Saat ini seseorang bekerja dengan mesin multiprosesor dan algoritme pembelajaran dijalankan sangat paralel pada kartu grafis modern, yang pada akhirnya mengarah pada waktu pelatihan berjam-jam hingga berminggu-minggu untuk pelatihan dengan optimasi parameter hiper.

Tiga puluh delapan sistem perangkat lunak berbeda yang tersedia secara bebas terdaftar di http://deeplearning.net/software_links[54] Theano, dengan dukungan khusus untuk kartu grafis, dan sistem Pylearn2 und Keras, yang didasarkan pada Theano, sangat menarik. Masing-masing diprogram dalam bahasa pemrograman Python. Tensorflow dari Google Deep Brain juga menggunakan Python. Satu-satunya persyaratan tambahan untuk proyek yang sukses adalah mesin cepat yang sesuai, yang dapat disewa dari penyedia layanan cloud sesuai kebutuhan.

9.21 APLIKASI PEMBELAJARAN MENDALAM

Gambar 9.28 Contoh data MNIST (kiri), data SVHN (tengah) dan contoh foto untuk sistem deskripsi gambar

Khususnya dalam klasifikasi objek dalam foto, pembelajaran mendalam telah menunjukkan kemajuan yang mengesankan. Ada kumpulan hasil untuk berbagai kumpulan data. Misalnya, dalam pengenalan angka tulisan tangan, akurasi klasifikasi 99,8% telah dicapai pada kumpulan data MNIST yang disajikan dalam Gambar 9.28. Pada kumpulan data SVHN (Gambar 9.28) yang dihasilkan sebagai bagian dari Proyek Google Street View dengan foto nomor alamat rumah, akurasinya sekitar 98,3%. Bahkan dimungkinkan untuk menggambarkan foto dalam kalimat lengkap. Untuk foto yang ditampilkan di sebelah kanan Gambar. 9.28, sistem ini menghasilkan kalimat "Seseorang mengendarai sepeda motor di jalan tanah." Ini menggunakan jaringan saraf convolutional untuk pengenalan objek dan jaringan saraf berulang untuk pembuatan teks. Selain banyak aplikasi lain, ada varian pembelajaran mendalam yang mengesankan untuk menghasilkan karya seni secara kreatif.

Kreativitas

Dengan pembelajaran mendalam, program telah muncul yang dapat, misalnya, menyusun melodi jazz. Jaringan saraf saat ini digunakan untuk menghasilkan teks, kode XML yang benar secara sintaksis, bahkan sumber LATEX dengan notasi matematika dan program komputer pendek. Oleh karena itu jaringan saraf berulang dapat mempelajari tata bahasa. Sebuah program yang dilatih pada tubuh kerja Shakespeare dapat, dari awal, menghasilkan teks baru seperti:

RAJA LEAR: O, jika Anda adalah penglihatan yang lemah,
kesopanan hukum Anda, penglihatan Anda dan beberapa napas,
akan memakai para dewa Dengan kepala, dan tangan saya heran pada perbuatan,
Jadi jatuhkan di atas kepala Yang Mulia,
dan pendapat Anda Akan bertentangan dengan kehormatan Anda.

[54] Akses April 2016

Wikipedia menggambarkan kreativitas sebagai fenomena di mana sesuatu yang baru dan entah bagaimana berharga terbentuk. Sangat mudah untuk menghargai definisi ini berdasarkan teks di atas. Mari kita bayangkan sebuah program yang secara acak menghasilkan teks huruf demi huruf. Hasilnya mungkin dimulai seperti

NitDjgMQsQfi 6zz1B:6xZkgp1xe.EqyD7z(C

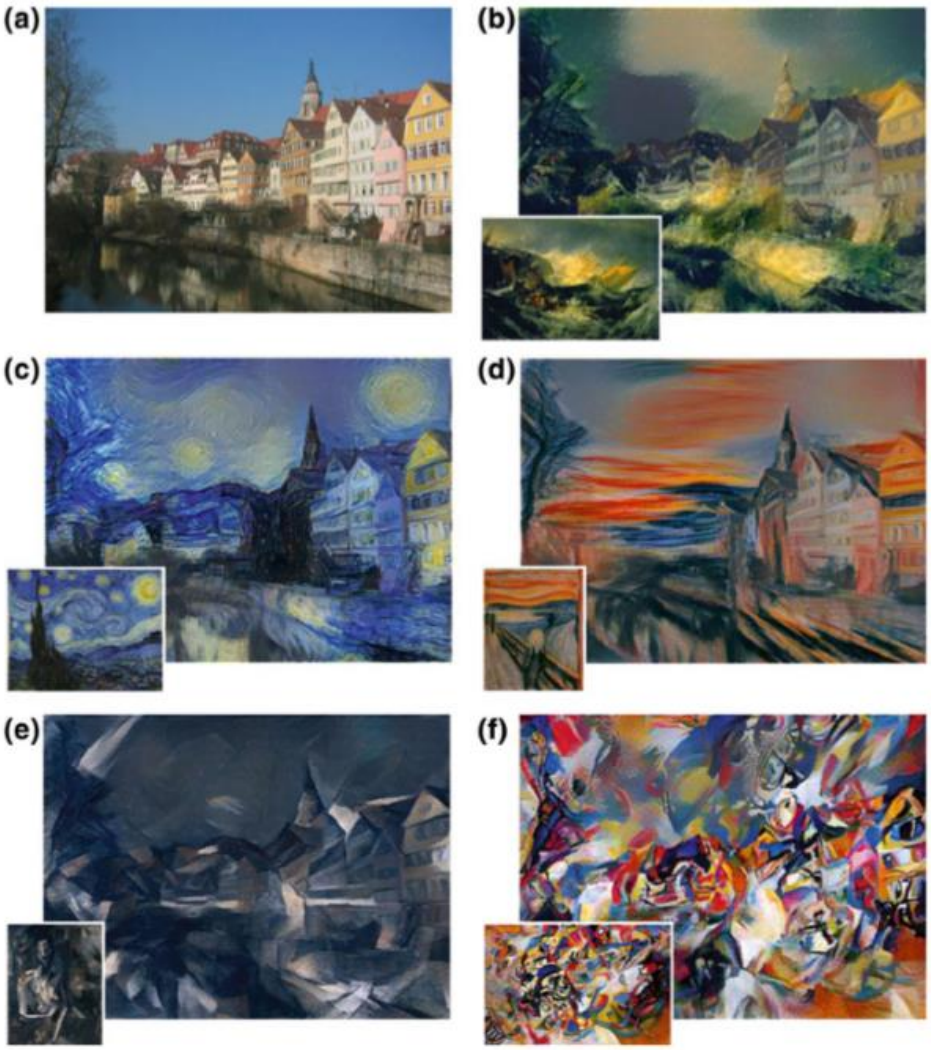
Meskipun teksnya baru, itu tidak terlalu berharga dan oleh karena itu bukan artefak kreatif. Di sisi lain, program yang mengkueri dan mengeluarkan teks dari database juga tidak dianggap kreatif karena tidak menghasilkan sesuatu yang baru. Program kreatif, misalnya, harus menulis novel yang menarik sekaligus baru. Seseorang dapat dengan mudah memeriksa kebaruan teks, misalnya dengan menggunakan perangkat lunak untuk menguji orisinalitas makalah tesis mahasiswa. Ketertarikan lebih sulit. Akan lebih baik jika sebuah program dapat menghasilkan teks-teks baru yang akan dinikmati seseorang berdasarkan peringkat yang diberikan orang tersebut untuk membaca literatur sebelumnya.

Secara lebih formal, kita dapat mendefinisikan program kreatif adaptif sebagai program yang mengambil beberapa objek dari distribusi data pelatihan tertentu, kemudian mendekati distribusi tersebut dan secara acak membuat objek baru dari distribusi ini.

Salah satu contoh yang sangat bagus dari upaya kreatif semacam itu adalah CNN, yang mengambil dua gambar input dan menghasilkan gambar baru yang mengasumsikan gaya gambar pertama dan konten gambar kedua. Hasil dari program ini secara mengesankan ditunjukkan pada Gambar 9.29. Foto di kiri atas diubah menjadi gaya masing-masing lukisan oleh para empu tua.

Program ini menggunakan jaringan VGG, yang merupakan jaringan konvolusi yang telah dilatih sebelumnya pada tantangan pengenalan visual skala besar imagenet dengan ratusan kategori objek dan jutaan gambar. Dari jaringan klasifikasi objek yang telah dilatih sebelumnya ini, hanya lapisan fitur, tetapi bukan lapisan klasifikasi yang terhubung sepenuhnya, yang digunakan di sini. Sebaliknya, penulis hanya menggunakan enam belas lapisan convolutional dan lima lapisan penyatuan.

Untuk mendapatkan pemahaman yang lebih baik tentang prosedur, pertama-tama kita memecahkan dua masalah yang lebih sederhana. Foto diterapkan dan disebarkan ke depan melalui jaringan, menghasilkan fitur gambar foto. Sekarang gambar piksel dengan derau putih diterapkan dan disebarkan ke depan melalui jaringan, menghasilkan fitur gambar derau putih. Menggunakan pencarian penurunan gradien, piksel dari gambar white noise sekarang divariasikan hingga jarak antara fitur fitur gambar dengan fitur foto diminimalkan. Pada akhirnya ini mereproduksi konten foto.



Gambar 9.29 Gambar artistik pemandangan Tübingen di kiri atas dihasilkan dengan jaringan. Untuk setiap pasangan gambar,lukisan terkenal yang gayanya digunakan untuk mengubah foto dapat dilihat di kiri bawah.

Kemudian foto tersebut digantikan oleh karya seni lama yang diterapkan pada jaringan lain dengan struktur yang sama dan, mirip dengan foto, fitur-fiturnya dihitung. Sekali lagi gambar piksel dengan white noise diterapkan ke jaringan. Menggunakan pencarian penurunan gradien, itu diubah hingga jarak antara korelasi fitur yang dihasilkan dan lukisan diminimalkan. Pada akhirnya ini mereproduksi gaya lukisan.

Sekarang, untuk mentransfer gaya lukisan ke foto, kedua metode digabungkan. Kami menerapkan pencarian penurunan gradien pada gambar piksel dengan white noise tetapi sekarang kami meminimalkan kombinasi linier dari kedua fungsi jarak di atas. Maka terciptalah sebuah gambar ala lukisan dengan isi foto tersebut.

9.22 APLIKASI *NEURAL NETWORK*/JARINGAN SYARAF

Selain aplikasi yang diberikan sebagai contoh sejauh ini, ada banyak aplikasi untuk jaringan saraf di semua bidang industri, terutama untuk pembelajaran mendalam. Pengenalan pola dalam segala bentuknya adalah area yang sangat penting, baik analisis foto untuk mengenali orang atau wajah, pengenalan kawanan ikan dalam pembacaan sonar, pengenalan dan klasifikasi kendaraan militer dalam pemindaian radar, atau sejumlah

aplikasi lainnya. Jaringan saraf juga dapat dilatih untuk mengenali bahasa lisan dan teks tulisan tangan.

Jaringan saraf tidak hanya digunakan untuk mengenali objek dan pemandangan. Mereka dapat dilatih untuk mengontrol mobil atau robot yang dapat mengemudi sendiri berdasarkan data sensor, serta untuk mengontrol pencarian secara heuristik di komputer backgammon dan catur. Penggunaan jaringan yang menarik untuk pembelajaran penguatan dijelaskan dalam Bagian 10.8

Untuk beberapa waktu, jaringan saraf, selain metode statistik, telah berhasil digunakan untuk meramalkan harga saham dan menilai kelayakan kredit nasabah bank. Perdagangan cepat dari transaksi keuangan internasional tidak akan mungkin terjadi tanpa bantuan jaringan saraf yang cerdas dan cepat yang secara mandiri memutuskan untuk membeli atau menjual.

Algoritme pembelajaran mesin lainnya juga dapat digunakan untuk banyak aplikasi ini. Karena kesuksesan komersial yang besar dari *Data mining*, pembelajaran pohon keputusan dan mesin vektor dukungan, ada algoritma saraf untuk banyak aplikasi serta yang lain yang tidak termotivasi secara biologis sama sekali. Bidang jaringan saraf adalah subarea pembelajaran mesin.

Ringkasan dan Pandangan

Dengan perceptron, aturan delta, backpropagation, dan, dibangun di atasnya, pembelajaran mendalam, kami telah memperkenalkan kelas paling penting dari jaringan feedforward dan menunjukkan hubungannya dengan skor dan naive Bayes, dan juga dengan metode kuadrat terkecil.

Yang paling dekat dengan panutan biologis mereka adalah jaringan Hopfield yang menarik. Namun, karena dinamikanya yang kompleks, mereka sulit untuk ditangani dalam praktik. Dalam semua model saraf, informasi yang disimpan didistribusikan melalui banyak bobot. Oleh karena itu, kematian beberapa neuron memiliki sedikit efek nyata pada fungsi otak. Jaringan ini kuat terhadap gangguan kecil karena penyimpanan data yang terdistribusi. Terkait dengan ini adalah kemampuan untuk mengenali pola bising, yang telah ditunjukkan dalam beberapa contoh.

Representasi pengetahuan yang terdistribusi memiliki kelemahan, namun, sulit bagi insinyur pengetahuan untuk melokalisasi pengetahuan. Secara praktis tidak mungkin untuk menganalisis dan memahami banyak bobot dalam jaringan saraf yang terhubung penuh yang terlatih. Sebaliknya, relatif mudah untuk memahami pengetahuan yang dipelajari dalam pohon keputusan, dan bahkan untuk merepresentasikannya sebagai rumus logis. Logika predikat, yang memungkinkan formalisasi hubungan, sangat ekspresif dan elegan. Misalnya, predikat nenek (katrin, klaus) mudah dipahami. Jenis hubungan ini juga dapat dipelajari dengan jaringan saraf. Namun, tidak mungkin untuk menemukan "nenek saraf" di jaringan. Jadi, bahkan hari ini ada masalah saat menghubungkan jaringan saraf dengan sistem pemrosesan simbol. Langkah yang sangat penting ke arah ini adalah pembuatan fitur otomatis jaringan pembelajaran mendalam.

Dari sekian banyak model saraf yang menarik, banyak yang belum dibahas. Misalnya, peta yang mengatur sendiri, yang diperkenalkan oleh Kohonen, sangat menarik. Peta yang mengatur diri sendiri melakukan pemetaan yang terinspirasi secara biologis dari lapisan saraf sensorik ke lapisan kedua neuron dengan properti bahwa pemetaan ini adaptif dan mempertahankan kesamaan.

Satu masalah dengan jaringan yang disajikan di sini adalah pembelajaran tambahan. Jika, misalnya, jaringan backpropagation yang terlatih penuh dilatih lebih lanjut dengan pola-pola baru, banyak dan akhirnya semua pola lama akan mulai dilupakan dengan cepat.

Untuk mengatasi masalah ini, Carpenter dan Grossberg mengembangkan teori resonansi adaptif (ART), yang mengarah pada serangkaian model saraf.

9.23 LATIHAN

Dari Biologi ke Simulasi

Latihan 9.1

Tunjukkan simetri titik dari fungsi sigmoid ke h; 1 2Þ.

Jaringan Hopfield

Latihan 9.2

Gunakan applet jaringan Hopfield di http://www.cbu.edu/*pong/ai/hopfield/hopfieldapplet.htm dan uji kapasitas memori untuk pola yang berkorelasi dan tidak berkorelasi (dengan bit yang diatur secara acak) dengan kisi 10 10 ukuran. Gunakan pola dengan noise 10% untuk pengujian.

Latihan 9.3

Bandingkan batas teoretis N = 0,146 n untuk jumlah maksimum pola yang dapat disimpan, diberikan dalam Bagian. 9.2.2, dengan kapasitas penyimpanan biner klasik dengan ukuran yang sama.

Jaringan Linier dengan Kesalahan Minimal

Latihan 9.4

(a) Buatlah program untuk mempelajari pemetaan linier dengan metode kuadrat rata-rata terkecil. Ini cukup sederhana: kita hanya perlu menyiapkan persamaan normal dengan (9.12) dan kemudian menyelesaikan sistem persamaannya.

(b) Terapkan program ini ke data apendisitis di situs web buku ini dan tentukan skor liniernya. Berikan kesalahan sebagai fungsi dari ambang batas, seperti pada Gambar 9.14.

(c) Sekarang tentukan kurva ROC untuk skor ini.

Propagasi balik

Latihan 9.5

Menggunakan program backpropagation (misalnya dalam JNNS atau KNIME), buatlah sebuah jaringan dengan delapan neuron input, delapan neuron output, dan tiga neuron tersembunyi. Kemudian latih jaringan dengan delapan pasangan data pelatihan q^p, q^p dengan properti bahwa, dalam vektor pth q^p, bit pth adalah satu dan semua bit lainnya adalah nol. Jaringan dengan demikian memiliki tugas sederhana untuk mempelajari pemetaan yang identik. Jaringan seperti itu disebut encoder 8-3-8. Setelah proses pembelajaran, amati pengkodean tiga neuron tersembunyi untuk input kedelapan vektor input yang dipelajari. Apa yang Anda perhatikan? Sekarang ulangi percobaan, kurangi jumlah neuron tersembunyi menjadi dua dan kemudian satu.

Latihan 9.6

Untuk menunjukkan bahwa jaringan backpropagation dapat membagi himpunan yang tidak dapat dipisahkan secara linier, latih fungsi XOR. Data pelatihan untuk ini adalah: ((0, 0), 0), ((0, 1), 1), ((1, 0), 1), ((1, 1), 0).

(a) Buat jaringan dengan dua neuron masing-masing di lapisan input dan lapisan tersembunyi, dan satu neuron di lapisan output, dan latih jaringan ini.

(b) Sekarang hapus neuron tersembunyi dan hubungkan lapisan input langsung ke neuron output. Apa yang Anda amati?

Latihan 9.7

(a) Tunjukkan bahwa setiap jaringan propagasi balik multi-lapisan dengan fungsi aktivasi linier sama kuatnya dengan jaringan dua-lapisan. Untuk ini, cukup

ditunjukkan bahwa eksekusi berturut-turut dari pemetaan linier adalah pemetaan linier.

(b) Tunjukkan bahwa jaringan backpropagation dua lapis dengan fungsi aktivasi monotonik yang ketat tidak lebih kuat untuk tugas klasifikasi daripada jaringan tanpa fungsi aktivasi atau dengan fungsi aktivasi linier.

Mendukung Mesin Vektor

Latihan 9.8

Diberikan dua set data pelatihan dua dimensi yang tidak dapat dipisahkan secara linier, M+ dan M−. Semua titik di M+ berada di dalam lingkaran satuan $x^2_1 + x_2^2 = 1$ dan semua titik di M− berada di luar. Berikan transformasi koordinat f : $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ yang membuat data dapat dipisahkan secara linier. Berikan persamaan garis pemisah dan buat sketsa dua spasi dan titik datanya.

BAB 10
REINFORCEMENT *LEARNING*

10.1 PENDAHULUAN

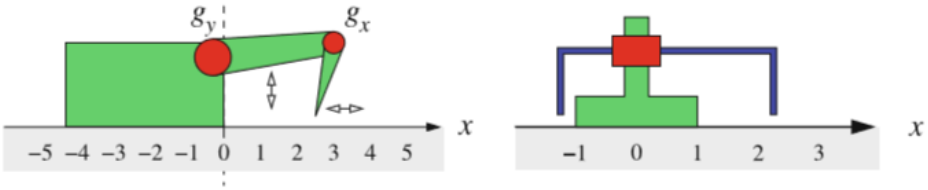
Semua algoritma pembelajaran yang dijelaskan sejauh ini—kecuali algoritme pengelompokan— termasuk dalam kelas pembelajaran terawasi. Dalam pembelajaran terawasi, agen seharusnya mempelajari pemetaan dari variabel input ke variabel output. Di sini penting bahwa untuk setiap contoh pelatihan individu, semua nilai untuk variabel input dan variabel output disediakan. Dengan kata lain, kita membutuhkan seorang guru atau database di mana pemetaan yang akan dipelajari didefinisikan secara kira-kira untuk sejumlah nilai input yang memadai. Satu-satunya tugas algoritme pembelajaran mesin adalah menyaring noise dari data dan menemukan fungsi yang mendekati pemetaan dengan baik, bahkan di antara titik data yang diberikan.

Dalam pembelajaran penguatan situasinya berbeda dan lebih sulit karena tidak ada data pelatihan yang tersedia. Kita mulai dengan contoh ilustrasi sederhana dari robotika, yang kemudian digunakan sebagai aplikasi untuk berbagai algoritma. Pembelajaran penguatan sangat berharga di bidang robotika, di mana tugas-tugas yang harus dilakukan seringkali cukup kompleks untuk menentang pengkodean sebagai program dan tidak ada data pelatihan yang tersedia. Tugas robot terdiri dari mencari tahu, melalui trial and error (atau sukses), tindakan mana yang baik dalam situasi tertentu dan mana yang tidak. Dalam banyak kasus, kita manusia belajar dengan cara yang sangat mirip. Misalnya, ketika seorang anak belajar berjalan, ini biasanya terjadi tanpa instruksi, melainkan hanya melalui penguatan. Upaya yang berhasil dalam berjalan dihargai dengan kemajuan ke depan, dan upaya yang gagal dihukum dengan jatuh yang seringkali menyakitkan. Penguatan positif dan negatif juga merupakan faktor penting dalam keberhasilan pembelajaran di sekolah dan di banyak olahraga (lihat Gambar 10.1). Sebuah tugas gerakan yang sangat disederhanakan dipelajari dalam contoh berikut.

Contoh 10.1 Robot yang mekanismenya hanya terdiri dari balok persegi panjang dan lengan dengan dua sambungan gy dan gx ditunjukkan pada Gambar 10.2. Satu-satunya tindakan yang mungkin dilakukan robot adalah pergerakan gy ke atas atau ke bawah dan gx ke kanan atau ke kiri. Selanjutnya, kami hanya mengizinkan pergerakan unit diskrit tetap (misalnya, kenaikan 10 derajat). Tugas agen terdiri dari mempelajari kebijakan yang memungkinkannya bergerak secepat mungkin ke kanan. Robot berjalan pada Gambar 10.2 bekerja secara analog dalam ruang keadaan dua dimensi yang sama

Gambar 10.1 "Mungkin lain kali saya harus mulai berputar sedikit lebih cepat atau lebih lambat?"—Belajar dari penguatan negatif

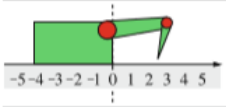
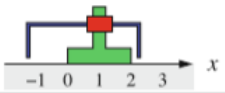
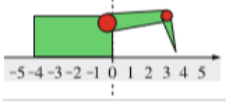
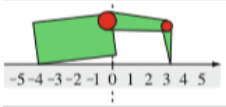
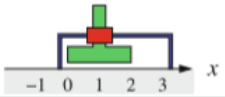
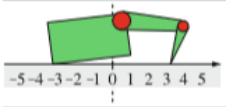
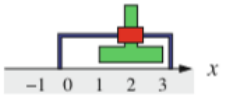
Urutan tindakan yang berhasil ditunjukkan pada Tabel 10.1. Tindakan pada waktu t = 2 menghasilkan lengan yang dibebani menggerakkan tubuh satu satuan panjang ke kanan. Sebelum kita masuk ke algoritma pembelajaran, kita harus memodelkan tugas secara matematis. Kami menggambarkan keadaan robot dengan dua variabel g_x dan g_y untuk posisi sambungan, masing-masing dengan banyak nilai diskrit. Keadaan robot dengan demikian dikodekan sebagai vektor (g_x, g_y). Banyaknya kemungkinan posisi joint adalah nx, atau masing-masing ny. Kami menggunakan posisi horizontal tubuh robot, yang dapat mengambil nilai nyata, untuk mengevaluasi tindakan robot. Gerakan ke kanan dihargai dengan perubahan positif ke x, dan gerakan ke kiri dihukum dengan perubahan negatif. Pada Gambar 10.3 ruang keadaan untuk dua varian ini ditunjukkan dalam bentuk yang disederhanakan.[55] Pada contoh kiri, kedua sambungan memiliki dua posisi, dan pada contoh tengah masing-masing memiliki empat posisi. Sebuah kebijakan yang optimal diberikan pada Gambar. 10.3 kanan.



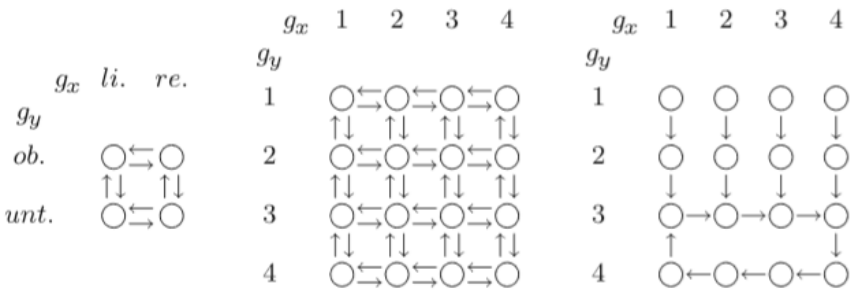
Gambar 10.2 Dengan menggerakkan kedua sendinya, robot perayap di sebelah kiri gambar dapat bergerak maju dan mundur.

Robot berjalan di sisi kanan juga harus menggerakkan bingkai ke atas dan ke bawah atau ke kiri dan ke kanan. Umpan balik untuk gerakan robot adalah positif untuk gerakan ke kanan dan negatif untuk gerakan ke kiri.

Tabel 10.1 Siklus deret periodik gerakan dengan gerakan maju sistematis

Crawling robot	Running robot	Time	State		Reward	Action
		t	g_y	g_x	x	a_t
		0	Up	Left	0	Right
		1	Up	Right	0	Down
		2	Down	Right	0	Left
		3	Down	Left	1	Up

[55] Ruang gerakan lengan yang terdiri dari busur dirender sebagai kisi siku-siku.

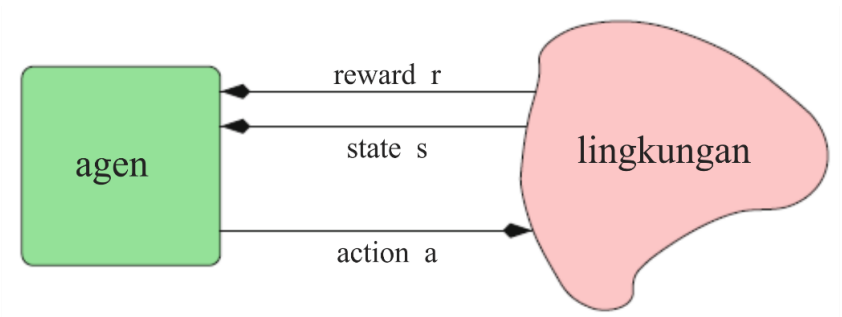


Gambar 10.3 Ruang keadaan robot

Contoh dalam kasus dua posisi yang mungkin untuk masing-masing sambungan (kiri) dan dalam kasus empat posisi horizontal dan vertikal untuk masing-masing (tengah). Pada gambar yang tepat diberikan kebijakan yang optimal.

10.2 TUGAS

Seperti yang ditunjukkan pada Gambar. 10.4, kami membedakan antara agen dan lingkungannya. Pada saat t dunia, yang meliputi agen dan lingkungannya, dijelaskan oleh keadaan $s_t \in S$. Himpunan S adalah abstraksi dari kemungkinan keadaan dunia yang sebenarnya karena, di satu sisi, dunia tidak dapat dideskripsikan secara tepat, dan di sisi lain agen seringkali hanya memiliki informasi yang tidak lengkap tentang keadaan sebenarnya karena kesalahan pengukuran. Agen kemudian melakukan aksi pada $a_t \in A$ pada waktu t. Tindakan ini mengubah dunia dan dengan demikian menghasilkan keadaan $s_t + 1$ pada waktu t + 1. Fungsi transisi keadaan δ yang didefinisikan oleh lingkungan menentukan keadaan baru $s_t +1 = \delta (s_t, a_t)$. Itu tidak dapat dipengaruhi oleh agen.



Gambar 10.4 Agen dan interaksinya dengan lingkungan

Setelah melakukan aksi di, agen langsung mendapatkan reward $r_t = r(s_t, a_t)$ (lihat Gambar 10.4). Hadiah langsung r_t = r(st, at) selalu bergantung pada status saat ini dan tindakan yang dieksekusi. r(st, at) = 0 berarti bahwa agen tidak menerima umpan balik langsung untuk tindakan di. Selama pembelajaran, $r_t > 0$ harus menghasilkan penguatan positif dan $r_t < 0$ negatif dari evaluasi tindakan di keadaan st. Dalam pembelajaran penguatan khususnya, aplikasi sedang dipelajari di mana tidak ada hadiah langsung yang terjadi untuk waktu yang lama. Seorang pemain catur misalnya belajar untuk meningkatkan permainannya dari pertandingan yang dimenangkan atau kalah, bahkan jika dia tidak mendapat hadiah langsung untuk semua gerakan individu. Di sini kita dapat melihat kesulitan dalam menetapkan hadiah di akhir urutan tindakan untuk semua tindakan dalam urutan yang mengarah ke titik ini (masalah penugasan kredit).

Persamaan 10.1

$$v^{\pi}(St) = r_t + \gamma r_t + 1 + \gamma^2 r_t + 2 + \cdots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

dari kebijakan p ketika kita mulai di negara awal st. Di sini 0 c < 1 adalah konstanta, yang memastikan bahwa umpan balik di masa mendatang lebih didiskontokan semakin jauh di masa depan hal itu terjadi. Hadiah langsung rt diberi bobot yang paling kuat. Fungsi penghargaan ini adalah yang paling banyak digunakan. Alternatif yang terkadang menarik adalah hadiah rata-rata

Persamaan 10.2

$$v^{\pi}(St) \lim_{h \to \infty h} \frac{1}{h} \sum_{i=0}^{h} r_{t+1}$$

Suatu kebijakan p⋆disebut optimal, jika untuk semua keadaan s

Persamaan 10.3

$$v^{\pi}(S) \geq v^{\pi}(S)$$

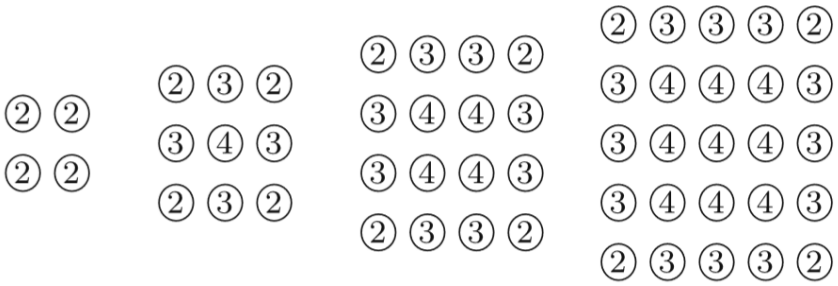
Artinya, setidaknya sama bagusnya dengan semua kebijakan lain menurut fungsi nilai yang ditentukan. Untuk keterbacaan yang lebih baik, fungsi nilai optimal VpH akan dilambangkan $v^{\pi*}$. Agen yang dibahas di sini, atau kebijakan mereka, hanya menggunakan informasi tentang keadaan saat ini st untuk menentukan keadaan selanjutnya, dan bukan sejarah sebelumnya. Ini dibenarkan jika imbalan dari suatu tindakan hanya bergantung pada keadaan saat ini dan tindakan saat ini. Proses seperti itu disebut proses keputusan Markov (MDP). Dalam banyak aplikasi, terutama dalam robotika, keadaan sebenarnya dari agen tidak diketahui secara pasti, yang membuat tindakan perencanaan menjadi lebih sulit. Alasan untuk ini mungkin karena sinyal sensor yang bising. Kami menyebut proses seperti itu sebagai proses keputusan Markov yang dapat diamati sebagian (POMDP).

10.3 PENCARIAN KOMBINATORIAL TANPA INFORMASI

Kemungkinan paling sederhana untuk menemukan kebijakan yang berhasil adalah enumerasi kombinatorial dari semua kebijakan, seperti yang dijelaskan dalam Bab. 6. Namun, bahkan dalam Contoh 10.1 yang sederhana terdapat banyak sekali kebijakan, yang menyebabkan pencarian kombinatorial dikaitkan dengan biaya komputasi yang sangat tinggi. Pada Gambar 10.5 jumlah tindakan yang mungkin diberikan untuk setiap keadaan. Dari sana, jumlah kebijakan yang mungkin dihitung sebagai produk dari nilai-nilai yang diberikan, seperti yang ditunjukkan pada Tabel 10.2 .

Untuk nilai sembarang n_x dan n_y selalu ada empat simpul sudut dengan dua kemungkinan aksi, $2(n_x - 2) + 2(n_y - 2)$ simpul tepi dengan tiga aksi, dan $(n_x - 2)(n_y - 2)$ simpul dalam dengan empat tindakan. Jadi ada kebijakan yang berbeda untuk nx dan ny tetap. Jumlah kebijakan dengan demikian tumbuh secara eksponensial dengan jumlah negara bagian. Hal ini berlaku secara umum jika ada lebih dari satu kemungkinan tindakan per negara bagian. Untuk aplikasi praktis algoritma ini karena itu tidak berguna. Bahkan pencarian heuristik, dijelaskan dalam Bab. 6, tidak dapat digunakan di sini. Karena imbalan langsung untuk hampir semua tindakan adalah nol, ia tidak dapat digunakan sebagai fungsi evaluasi heuristik.

$$2^4 3^2 (n_x - 2) + 2(n_y - 2)\ 4(n_x - 2)\ (n_y - 2)$$



Gambar 10.5 Ruang keadaan untuk contoh dengan nilai 2, 3, 4, 5 untuk nx dan ny. Jumlah tindakan yang mungkin diberikan untuk setiap keadaan di lingkaran masing-masing

Tabel 10.2 Jumlah kebijakan untuk ruang negara berukuran berbeda dalam contoh

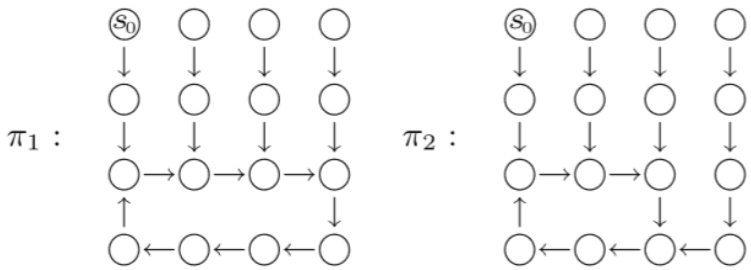
n_x, n_y	*Jumlah State*	*Jumlah kebijakan*
24	4	$2^4 = 26$
39	9	$2^4 3^4 4 = 5184$
416	16	$2^4 3^8 4^4 \approx 2.7 \times 10^7$
525	25	$2^4 3^{12} 4^9 \approx 10^{12}$

Biaya komputasi naik lebih tinggi lagi ketika kita mempertimbangkan bahwa (selain enumerasi semua kebijakan), nilai V^{π} (s) harus dihitung untuk setiap kebijakan yang dihasilkan p dan setiap status awal s. Jumlah tak terhingga dalam V^{π} (s) harus dipotong untuk digunakan dalam perhitungan praktis; namun, karena reduksi eksponensial dari faktor γ^i dalam (10.1) , hal ini tidak menimbulkan masalah.

Pada Contoh 10.1 perbedaan x_{t+1} - x_t dapat digunakan sebagai hadiah langsung untuk suatu tindakan di, yang berarti bahwa setiap gerakan tubuh robot ke kanan dihargai dengan 1 dan setiap gerakan tubuh robot ke kiri dihukum dengan -1. Pada Gambar 10.6, dua kebijakan ditampilkan. Di sini hadiah langsung adalah nol di mana-mana selain di baris bawah ruang keadaan. Kebijakan kiri p1 lebih baik dalam jangka panjang karena, untuk urutan tindakan yang panjang, kemajuan rata-rata per tindakan adalah 3/8 = 0,375 untuk π_1 dan 2/6 ≈ 0,333 untuk π_2. Jika kita menggunakan (10.1) untuk V^{π} (s), hasilnya adalah tabel berikut dengan status awal s0 di kiri atas dan berbagai nilai γ:

γ	0.9	0.8375	0.8
$V^{\pi_1}(s_0)$	2.52	1.156	0.77
$V^{\pi_2}(s_0)$	2.39	1.156	0.80

Di sini kita melihat bahwa kebijakan π_1 lebih unggul daripada kebijakan π_2 ketika gamma = 0,9, dan kebalikannya benar ketika gamma = 0,8. Untuk γ ≈ 0.8375 kedua polis sama-sama bagus. Kita dapat melihat dengan jelas bahwa c yang lebih besar menghasilkan cakrawala waktu yang lebih besar untuk evaluasi kebijakan.



Gambar 10.6 Dua kebijakan berbeda untuk contoh

10.4 ITERASI NILAI DAN PEMROGRAMAN DINAMIS

Dalam pendekatan naif untuk menghitung semua kebijakan, banyak pekerjaan yang berlebihan dilakukan, karena banyak kebijakan yang sebagian besar identik. Mereka mungkin hanya sedikit berbeda. Namun demikian, setiap kebijakan benar-benar baru dibuat dan dievaluasi. Ini menyarankan untuk menyimpan hasil antara untuk sebagian kebijakan dan menggunakannya kembali. Pendekatan untuk memecahkan masalah optimasi ini diperkenalkan sebagai pemrograman dinamis oleh Richard Bellman pada tahun 1957. Bellman menyadari bahwa untuk kebijakan yang optimal adalah:

> *Terlepas dari keadaan awal st dan tindakan pertama di, semua keputusan selanjutnya yang melanjutkan dari setiap kemungkinan keadaan penerus st+1 harus optimal.*

Berdasarkan apa yang disebut prinsip Bellman, menjadi mungkin untuk menemukan kebijakan yang optimal secara global melalui optimasi lokal dari tindakan individu. Kami akan menurunkan prinsip ini untuk MDP bersama dengan algoritma iterasi yang sesuai. Diinginkan adalah kebijakan optimal π* yang memenuhi (Persamaan 10.3) dan (Persamaan 10.1) . Kami menulis ulang kedua persamaan dan memperoleh

Persamaan 10.4

$$V*(s_t) = \max_{at,at+1,at+2,\ldots} (r(s_t, a_t) + \gamma r(s_t + 1, at + 1) + \gamma^2 r(s_t + 2, a_t + 2) + \cdots))$$

Karena imbalan langsung $r(s_t, a_t)$ hanya bergantung pada s_t dan a_t, tetapi tidak pada status dan tindakan penerus, maksimalisasi dapat didistribusikan, yang pada akhirnya menghasilkan karakterisasi rekursif dari V*:

Persamaan 10.5

$$\begin{aligned} V*(s_t) &= \max_{at}[r(s_t, a_t) + \gamma \max_{at+1,at+2,\ldots} (r((s_t + 1, at + 1) \\ &+ \gamma r(s_t + 1, at + 1) + \gamma^2 r(s_t + 2, a_t + 2) + \cdots)) \\ &= \max_{at}[r(s_t, a_t) + \gamma V^*(s_t + 1)] \end{aligned}$$

Persamaan (10.5) dihasilkan dari substitusi t→t + 1 pada (10.4). Ditulis agak lebih sederhana:

Persamaan 10.6

$$V*(s) = \max_{a}[r(s, a) + \gamma V*(\delta(s, a))]$$

Persamaan ini menyiratkan, seperti halnya (10.1) bahwa, untuk menghitung V*(s), hadiah langsung ditambahkan ke hadiah dari semua negara penerus, didiskontokan oleh faktor γ. Jika V*(δ(s, a)) diketahui, maka V*(s) jelas dihasilkan oleh optimasi lokal sederhana untuk semua kemungkinan tindakan a dalam keadaan s. Ini sesuai dengan prinsip Bellman, karena itu (10.6) juga disebut persamaan Bellman.

Kebijakan optimal π*(s) melakukan aksi pada keadaan s yang menghasilkan nilai maksimum V*. Dengan demikian,

Persamaan 10.7

$$\pi^*(s) = \underset{a}{argmax}[r(s, a) + \gamma V^*(\delta(s, a))]$$

Dari persamaan rekursi (10.6) aturan iterasi untuk mendekati V*: berikut secara langsung:

Persamaan 10.8

$$V^*(s) = \max_{a}[r(s, a) + \gamma V(\delta(s, a))]$$

Untuk memulai nilai perkiraan V(s) untuk semua keadaan diinisialisasi, misalnya dengan nilai nol. Sekarang V(s) diperbarui berulang kali untuk setiap keadaan dengan mundur secara rekursif pada nilai V(δ(s,a)) dari keadaan penerus terbaik. Proses penghitungan V ini disebut iterasi nilai dan ditunjukkan secara skematis pada Gambar 10.7. Hal ini dapat ditunjukkan bahwa iterasi nilai konvergen. Analisis yang sangat baik dari algoritma pemrograman dinamis dapat ditemukan berdasarkan sifat kontraksi dari algoritma

tertentu (misalnya iterasi nilai), konvergensi dapat dibuktikan menggunakan fixed- Banach. teorema titik.

Pada Gambar 10.8 algoritma ini diterapkan pada Contoh 10.1 dengan πγ = 0.9. Dalam setiap iterasi, status diproses berdasarkan baris dari kiri bawah ke kanan atas. Ditampilkan beberapa iterasi awal dan pada gambar kedua di baris bawah nilai batas stabil untuk V⋆.

Kami dengan jelas melihat perkembangan pembelajaran dalam urutan ini. Agen berulang kali menjelajahi semua status, melakukan iterasi nilai untuk setiap status dan menyimpan kebijakan dalam bentuk fungsi tabulasi V*, yang kemudian dapat dikompilasi lebih lanjut menjadi tabel π* yang dapat digunakan secara efisien.

Kebetulan, untuk menemukan kebijakan yang optimal dari V* akan salah jika memilih aksi pada state st yang menghasilkan state dengan nilai V* maksimum. Sesuai dengan (10.7), hadiah langsung r(s_t, a_t) juga harus ditambahkan karena kita mencari V*(s_t) dan bukan V*(s_t+1). Diterapkan pada keadaan s = (2, 3) pada Gambar 10.8, ini berarti

$$\pi^*(2,3) = \underset{a \in \{kiri,kanan,atas\}}{argmax} [r(s,a) + \gamma V^*(\delta(s,a))]$$

$$= \underset{a \in \{kiri,kanan,atas\}}{argmax} \{1 + 0.9 + \ .\ 2.66, -1 + 0.9\ .\ 4.05, 0 + 0.9\ .\ 3,28\}$$

$$= \underset{a \in \{kiri,kanan,atas\}}{argmax} \{3.39, 2.65, 2.95\} = kiri$$

Pada (Persamaan 10.7) kita melihat bahwa agen dalam keadaan s_t harus mengetahui imbalan langsung r_t dan keadaan penerus $s_{t+1} = \delta(s_t, a_t)$ untuk memilih tindakan optimal a_t. Itu juga harus memiliki model fungsi r dan δ. Karena ini tidak berlaku untuk banyak aplikasi praktis, diperlukan algoritma yang juga dapat bekerja tanpa pengetahuan tentang r dan δ. Bagian 10.6 didedikasikan untuk algoritma semacam itu.

VALUEITERATION()

For all $s \in \mathcal{S}$

$\hat{V}(s) = 0$

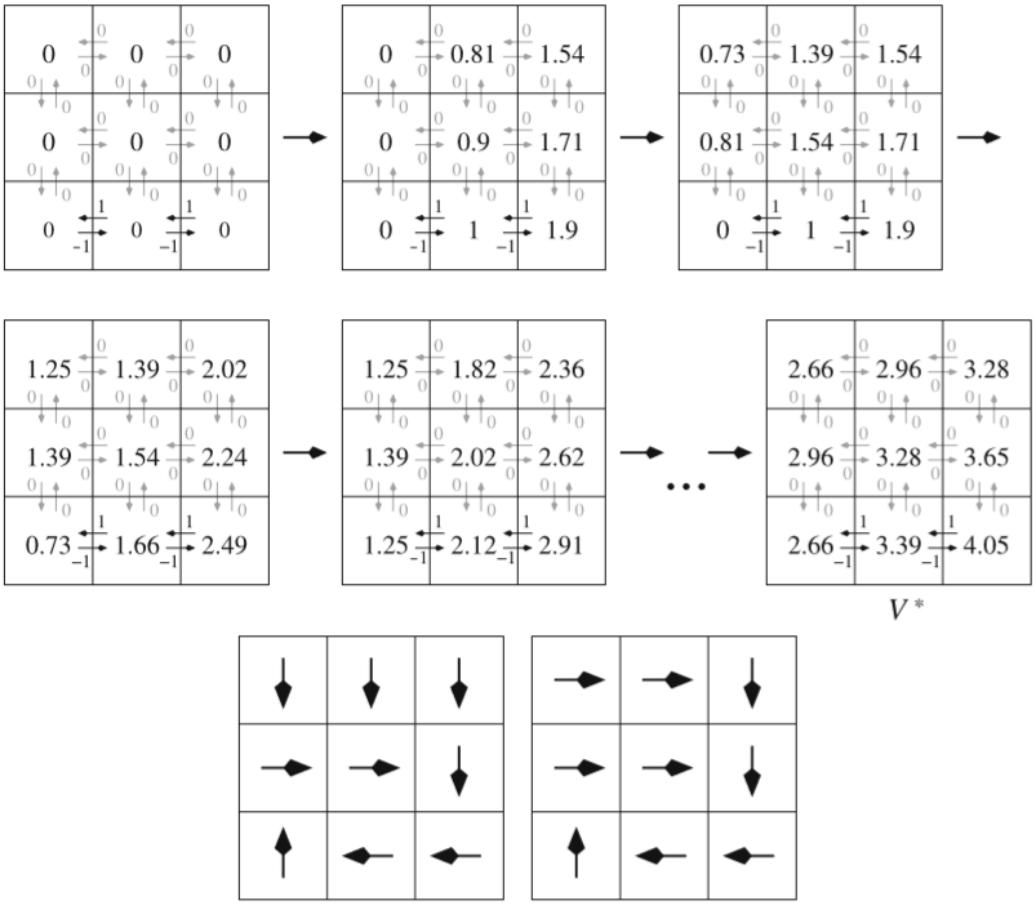
Repeat

For all $s \in \mathcal{S}$

$\hat{V}(s) = \max_a [r(s,a) + \gamma \hat{V}(\delta(s,a))]$

Until $\hat{V}(s)$ does not change

Gambar 10.7 Algoritma untuk iterasi nilai

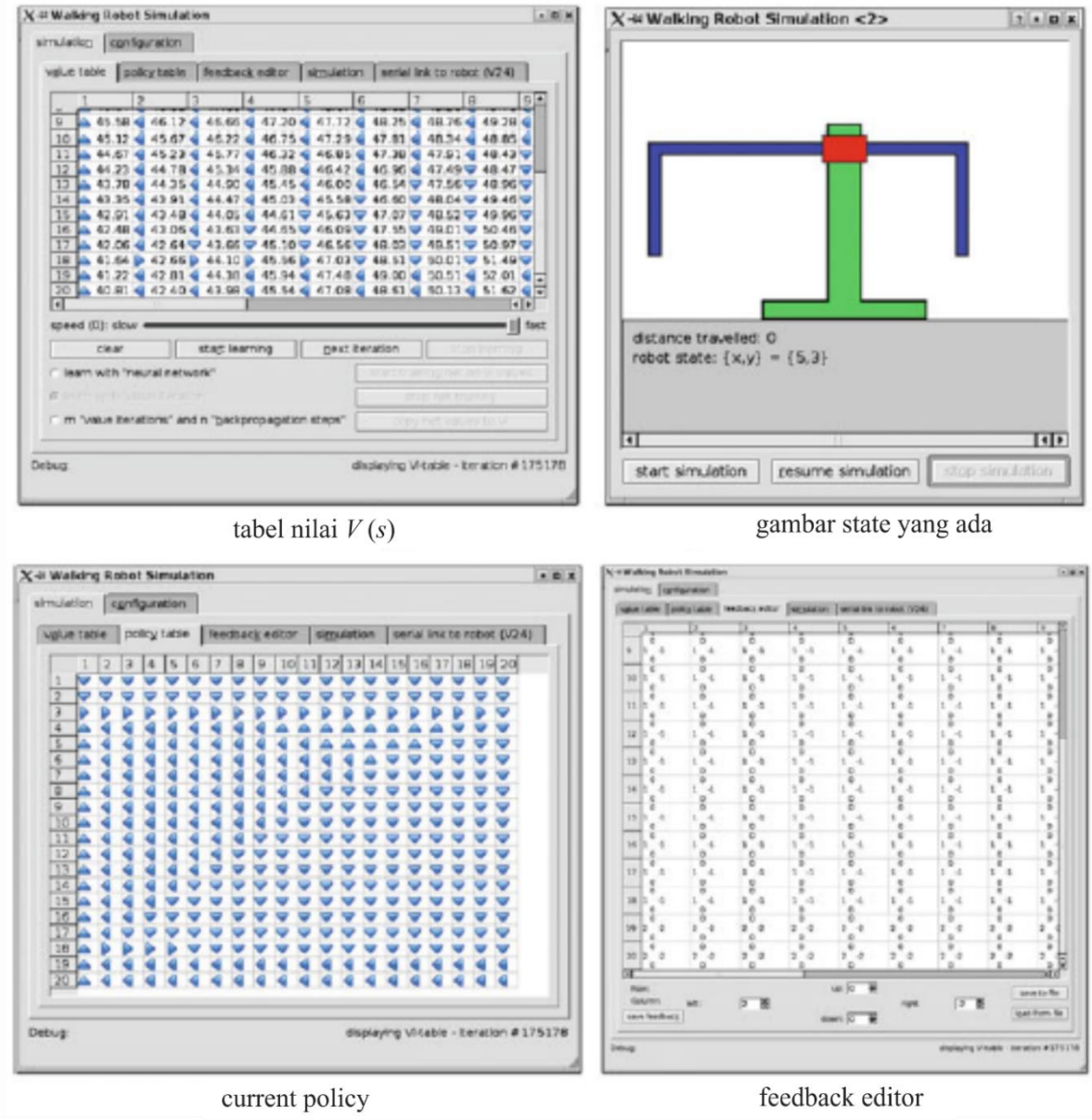


Gambar 10.8 Nilai iterasi pada contoh dengan keadaan 3 x 3.

Dua gambar terakhir menunjukkan dua kebijakan optimal. Angka-angka di sebelah panah memberikan hadiah langsung r(s, a) dari setiap tindakan

10.5 ROBOT BELAJAR BERJALAN DAN SIMULASINYA

Antarmuka pengguna grafis untuk eksperimen sederhana dengan pembelajaran penguatan ditunjukkan pada Gambar 10.9. Pengguna dapat mengamati pembelajaran penguatan untuk ruang keadaan dua dimensi berukuran berbeda. Untuk generalisasi yang lebih baik, jaringan backpropagation digunakan untuk menyimpan statu. Editor umpan balik yang ditampilkan di kanan bawah, yang dengannya pengguna dapat memberikan umpan balik secara manual tentang lingkungan, sangat menarik untuk eksperimen. Tidak ditampilkan menu untuk mengatur parameter untuk iterasi nilai dan pembelajaran backpropagation.



Gambar 10.9 Empat jendela berbeda dari simulator robot berjalan

Selain simulasi, dua kecil, robot merangkak nyata dengan ruang keadaan diskrit dua dimensi yang sama dikembangkan secara khusus untuk mengajar.[56] Kedua robot ditunjukkan pada Gambar. 10.10. Masing-masing bergerak dengan aktuator servo. Servo dikendalikan oleh mikrokontroler atau melalui antarmuka nirkabel langsung dari PC. Menggunakan perangkat lunak simulasi, matriks umpan balik robot dapat divisualisasikan pada PC. Dengan umpan balik yang disimpan ini, kebijakan dapat dilatih di PC (yang menghitung lebih cepat), kemudian dimuat lagi ke robot dan dieksekusi. Namun, robot juga bisa belajar secara mandiri. Untuk ruang keadaan ukuran 5 x 5 ini membutuhkan waktu sekitar 30 detik.

Sangat menarik untuk mengamati perbedaan antara simulasi dan robot "nyata". Berbeda dengan simulasi, crawler mempelajari kebijakan di mana ia tidak pernah mengangkat lengannya dari tanah, tetapi tetap bergerak maju dengan sangat efisien. Alasan untuk ini adalah bahwa, tergantung pada permukaan bawah tanah, ujung "ketiak"

[56] Informasi lebih lanjut dan sumber terkait tentang robot perayapan tersedia melalui www.hs-weingarten.de/*ertel/kibuch.

dapat mencengkeram tanah selama gerakan mundur, tetapi meluncur selama gerakan maju. Efek ini sangat masuk akal dirasakan melalui sensor pengukur jarak dan dievaluasi sesuai selama pembelajaran.

Adaptasi robot menghasilkan efek yang mengejutkan. Sebagai contoh, kita dapat mengamati bagaimana crawler, meskipun servo rusak yang tergelincir pada sudut tertentu, tetap belajar berjalan (lebih seperti terpincang-pincang). Ia bahkan mampu beradaptasi dengan situasi yang berubah dengan mengubah kebijakan. Efek yang benar-benar diinginkan adalah kemampuan, mengingat permukaan halus yang berbeda (misalnya, karpet kasar yang berbeda) untuk mempelajari kebijakan optimal untuk masing-masing. Ternyata robot yang sebenarnya juga sangat mudah beradaptasi mengingat ruang keadaan yang kecil berukuran 5 x 5.

Pembaca dapat (tidak memiliki robot sungguhan) memodelkan berbagai permukaan atau cacat servo dengan memvariasikan nilai umpan balik dan kemudian mengamati kebijakan yang dihasilkan (Latihan 10.3).

Gambar 10.10 Dua versi robot perayapan

10.6 Q-*LEARNING*

Kebijakan yang didasarkan pada evaluasi kemungkinan negara penerus jelas tidak dapat digunakan jika agen tidak memiliki model dunia, yaitu, ketika agen tidak tahu ke negara mana tindakan yang mungkin mengarah. Dalam sebagian besar aplikasi realistis, agen tidak dapat menggunakan model dunia seperti itu. Misalnya, robot yang seharusnya memegang objek kompleks tidak dapat memprediksi apakah objek akan dipegang dengan aman di genggamannya setelah tindakan mencengkeram, atau apakah itu akan tetap di tempatnya.

Jika tidak ada model dunia, maka diperlukan evaluasi terhadap suatu tindakan yang dilakukan di negara bagian meskipun masih belum diketahui kemana arah tindakan tersebut. Jadi kita sekarang bekerja dengan fungsi evaluasi $Q(s_t, a_t)$ untuk keadaan dengan tindakan yang terkait. Dengan fungsi ini, pilihan tindakan optimal dibuat oleh aturan
Persamaan 10.9

$$\pi^*(s) = \underset{a}{argmax}\ Q(s,a)$$

Untuk mendefinisikan fungsi evaluasi kita kembali menggunakan pendiskontoan bertahap dari evaluasi untuk pasangan keadaan-tindakan yang terjadi lebih jauh ke masa depan, seperti pada (10.1) . Dengan demikian kita ingin memaksimalkan $r_t + \gamma r_t+1+ \gamma^2 r_{t+2}+ ....$ Oleh karena itu, untuk mengevaluasi tindakan a_t pada keadaan s_t kita definisikan dalam analogi (10.4) :
Persamaan 10.10

$$Q\left(s_t, a_t\right) = \underset{at+1,at+2,\ldots}{max}\left(r(s_t, a_t) + \gamma r\,(s_t + 1, a_t + 1) + \gamma^2 r(s_t + 2, a_t + 2\right) + \cdots)$$

Analog dengan pendekatan untuk iterasi nilai, kami membawa persamaan ini ke dalam bentuk rekursif sederhana dengan:

Persamaan 10.11

$$\begin{aligned} Q\left(s_t, a_t\right) &= \max_{at+1,at+2,\ldots}\left(r(s_t, a_t) + \gamma r(s_t+1, a_t+1) + \gamma^2 r(s_t+2, a_t+2)\right) + \cdots) \\ &= r(s_t, a_t) + \gamma \max_{at+1,at+2,\ldots}(r(s_t+1, a_t+1) + \gamma^2 r(s_t+2, a_t+2) + \cdots) \\ &= r(s_t, a_t) + \gamma \max_{at+1}(r(s_t+1, a_t+1) + \gamma \max_{at+2}(r(s_t+2, a_t+2) + \cdots) \\ &= r(s_t, a_t) + \gamma \max_{at+1} Q(s_t+1, a_t+1) \\ &= r(s_t, a_t) + \gamma \max_{at+1} Q(\delta(s_t+1,)\, a_t+1) \\ &= r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a') \end{aligned}$$

Lalu apa keuntungannya dibandingkan dengan iterasi nilai? Persamaan lama hanya sedikit ditulis ulang, tetapi ini ternyata merupakan pendekatan yang tepat untuk algoritma baru. Alih-alih menyimpan V*, sekarang fungsi Q disimpan, dan agen dapat memilih tindakannya dari fungsi δ dan r tanpa model dunia. Kami masih belum memiliki proses, bagaimanapun, yang dapat mempelajari Q secara langsung, yaitu tanpa pengetahuan tentang V*.

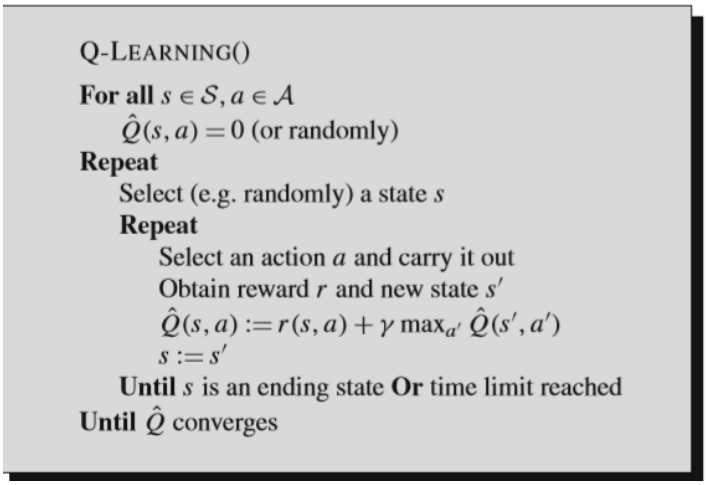
Dari formulasi rekursif Q(s, a), algoritma iterasi untuk menentukan Q(s, a) dapat diturunkan secara langsung. Kami menginisialisasi tabel Q(s, a) untuk semua menyatakan secara sewenang-wenang, misalnya dengan nol, dan melakukan iteratif

Persamaan 10.12

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$$

Perlu dicatat bahwa kita tidak mengetahui fungsi r dan d. Kami memecahkan masalah ini cukup pragmatis dengan membiarkan agen di lingkungannya dalam keadaan melakukan tindakan a. Keadaan penerus kemudian jelas δ(s, a) dan agen menerima hadiahnya dari lingkungan. Algoritma yang ditunjukkan pada Gambar 10.11 mengimplementasikan algoritma ini untuk Q-*learning*.

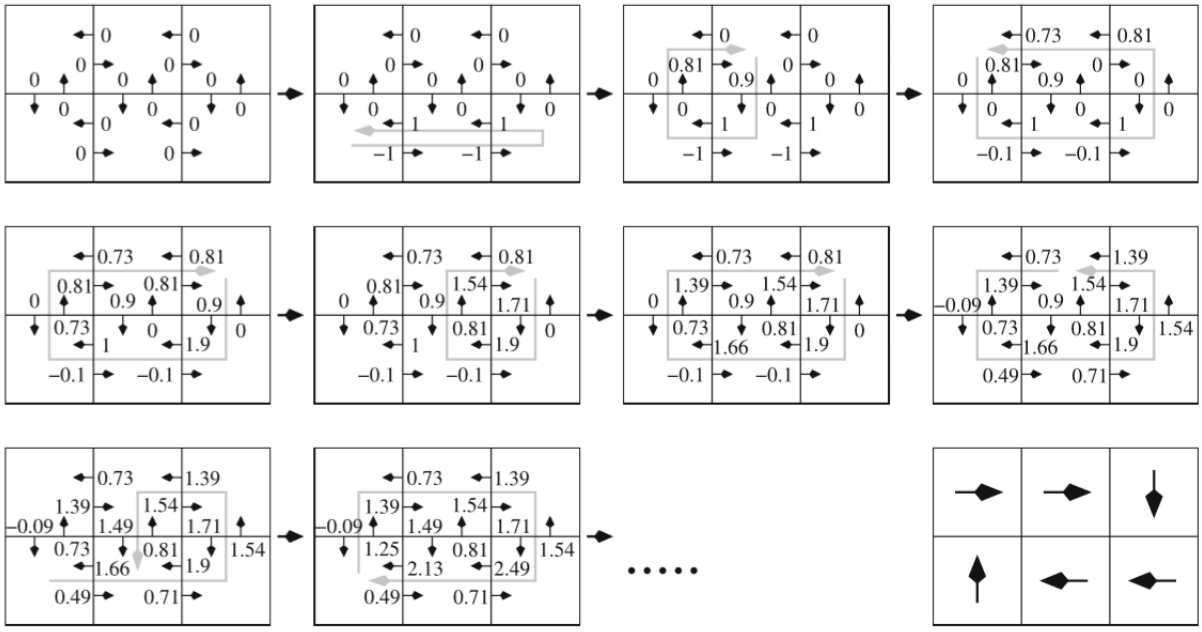
Penerapan algoritma untuk Contoh 10.1 dengan γ = 0.9 dan n_x = 3, n_y = 2 (yaitu, dalam kotak 2 x 3) ditunjukkan pada Gambar 10.12 sebagai contoh. Pada gambar pertama, semua nilai Q diinisialisasi ke nol. Pada gambar kedua, setelah urutan tindakan pertama, empat nilai r yang tidak sama dengan nol menjadi terlihat sebagai nilai Q. Pada gambar terakhir, kebijakan optimal yang dipelajari diberikan.



Gambar 10.11 Algoritma untuk Q-*learning*

Teorema 10.1
Biarkan MDP deterministik dengan hadiah langsung terbatas r(s, a) diberikan. Persamaan (10.12) dengan $0 \le \gamma < 1$ digunakan untuk pembelajaran. Biarkan $Q_n(s, a)$ menjadi nilai untuk Q(s, a) setelah n pembaruan. Jika setiap pasangan keadaan-aksi sering dikunjungi tak berhingga, maka $Q_n(s, a)$ konvergen ke Q(s, a) untuk semua nilai dan untuk $n \to \infty$.

Bukti : Karena setiap transisi keadaan-aksi sering terjadi tak berhingga, kita melihat interval waktu yang berurutan dengan sifat bahwa, dalam setiap interval, semua transisi keadaan-aksi terjadi setidaknya sekali. Kami sekarang menunjukkan bahwa kesalahan maksimum untuk semua entri dalam tabel Q dikurangi oleh setidaknya faktor γ di setiap interval ini.



Gambar 10.12 Q-*learning* diterapkan pada contoh dengan nx = 3, ny = 2. Panah abu-abu menandai tindakan yang dilakukan pada setiap gambar

Nilai Q yang diperbarui diberikan. Pada gambar terakhir, kebijakan saat ini, yang juga optimal, ditampilkan
Diketahui

$$\Delta n = \max_{s,a} | Q_n(s,a) - Q(s,a)|$$

menjadi kesalahan maksimum dalam tabel Q_n dan s′ = δ(s, a). Untuk setiap entri tabel ^ Qnðs;aÞkami menghitung kontribusinya terhadap kesalahan setelah interval sebagai

$$\begin{aligned} |Q_{n+1}(s,a) - Q(s,a) &= |(r + \gamma \max_{a'} Q_n(s',a') - (r + \gamma \max_{a'} Q_n(s',a')| \\ &= \gamma | \max_{a'} Q_n(s',a') - \max_{a'} Q_n(s',a')| \\ &\le \gamma \max_{a'} | Q_n(s',a') - Q_n(s',a')| \\ &\le \gamma \max_{a'} | Q_n(s'',a') - Q_n(s'',a')| = \gamma \Delta n \end{aligned}$$

Pertidaksamaan pertama benar karena, untuk fungsi sembarang f dan g,

$$l \left| \max_x f(x) - j \max_x g(x) \right| \le \max_x \left| f(x) - g(x) \right|$$

dan pertidaksamaan kedua benar karena, dengan tambahan variasi keadaan s′′, maksimum yang dihasilkan tidak dapat menjadi lebih kecil. Dengan demikian telah ditunjukkan bahwa $\Delta_{n+1} \le \gamma \Delta_n$. Karena kesalahan dalam setiap interval dikurangi dengan faktor paling sedikit γ,

setelah interval k paling banyak adalah $\gamma^k \Delta D_\Theta$, dan, sebagai hasilnya, Δ_Θ terbatas. Karena setiap state dikunjungi berkali-kali tak berhingga, ada tak hingga banyak interval dan Δ_n konvergen ke nol.

Q-*Learning* dalam Lingkungan Nondeterministik

Dalam banyak aplikasi robotika, lingkungan agen bersifat nondeterministik. Ini berarti bahwa reaksi lingkungan terhadap tindakan a dalam keadaan s pada dua titik waktu yang berbeda dapat menghasilkan keadaan dan imbalan pengganti yang berbeda. Proses Markov nondeterministik seperti itu dimodelkan dengan fungsi transisi probabilitas d(s, a) dan imbalan langsung probabilistik r(s, a). Untuk mendefinisikan fungsi Q, setiap kali nilai yang diharapkan harus dihitung pada semua kemungkinan status penerus. Persamaan (10.11) dengan demikian digeneralisasikan menjadi:

Persamaan 10.13

$$Q(s_1, a_1)) = E\big(r(s,a)\big) + \gamma \sum_{x'} P(s'|s,a) \max_d Q(s', a')$$

di mana P(s'|a) adalah probabilitas berpindah dari keadaan s ke keadaan penerus s' dengan aksi a. Sayangnya tidak ada jaminan konvergensi untuk Q-*learning* dalam kasus nondeterministik jika kita melanjutkan seperti sebelumnya menurut (10.12) . Hal ini karena, berturut-turut berjalan melalui loop luar dari algoritma pada Gambar. 10.11, penghargaan dan status penerus dapat benar-benar berbeda untuk status yang sama s dan tindakan yang sama a. Hal ini dapat mengakibatkan urutan bolak-balik yang melompat bolak-balik antara beberapa nilai. Untuk menghindari lompatan nilai Q yang kuat seperti ini, kami menambahkan nilai Q tertimbang lama ke sisi kanan (10.12) . Ini menstabilkan iterasi. Aturan belajar kemudian berbunyi

Persamaan 10.14

$$Q_n(s,a) = (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n \left[r(s,a) + \gamma \max_{a'} \hat{Q}_{n-1}\left(\delta(s,a), a'\right)\right]$$

dengan faktor pembobotan yang bervariasi terhadap waktu

$$\alpha_n = \frac{1}{1 + b_n(s,a)}$$

Nilai b_n (s, a) menunjukkan seberapa sering aksi a dieksekusi dalam keadaan s pada iterasi ke-n. Untuk nilai b_n yang kecil (yaitu, pada awal pembelajaran) istilah stabilisasi $\hat{Q}_{n-1}(s,a)$ tidak ikut berperan, karena kami ingin proses pembelajaran membuat kemajuan yang cepat. Namun, kemudian, bn menjadi lebih besar dan dengan demikian mencegah lompatan yang terlalu besar dalam urutan nilai $\hat{Q}$. Saat mengintegrasikan (10.14) ke dalam Q-*learning*, nilai b_n(s, a) harus disimpan untuk semua pasangan aksi-status. Ini dapat dicapai dengan memperluas tabel nilai ^ Q.

Untuk pemahaman yang lebih baik tentang (10.14), kami menyederhanakan ini dengan mengasumsikan $\alpha_n = \alpha$ adalah konstanta dan mengubahnya sebagai berikut:

$$\begin{aligned}\hat{Q}_n(s,a) &= (1-\alpha)\hat{Q}_{n-1}(s,a) + \alpha[r(s,a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s,a), a')] \\ &= \hat{Q}_{n-1}(s,a) + \underbrace{\alpha[r(s,a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s,a), a') - \hat{Q}_{n-1}(s,a)]}_{\text{TD-error}}.\end{aligned}$$

Nilai Q baru $\hat{Q}_{n-1}(s,a)$ dapat dengan jelas direpresentasikan sebagai $\hat{Q}_{n-1}(s,a)$ lama ditambah atimes suku koreksi yang sama dengan perubahan nilai Q pada langkah ini. Istilah koreksi disebut kesalahan TD, atau kesalahan perbedaan temporal, dan persamaan di atas untuk mengubah nilai Q adalah kasus khusus dari TD-*Learning*, kelas penting dari algoritma pembelajaran. Untuk 1 kita memperoleh Q-*learning* dijelaskan di atas. Untuk a 0 nilai $\hat{Q}$ sama sekali tidak berubah. Dengan demikian tidak ada pembelajaran yang terjadi.

10.7 EKSPLORASI DAN EKSPLOITASI

Untuk Q-*learning* sejauh ini, hanya skema algoritma kasar yang telah diberikan. Yang kurang adalah deskripsi pilihan keadaan awal setiap kali dan tindakan yang akan dilakukan di loop dalam Gambar 10.11 . Untuk pemilihan tindakan berikutnya ada dua kemungkinan. Di antara tindakan yang mungkin, satu dapat dipilih secara acak. Dalam jangka panjang, hal ini menghasilkan eksplorasi yang seragam dari semua tindakan atau kebijakan yang mungkin, tetapi dengan konvergensi yang sangat lambat. Alternatif untuk ini adalah eksploitasi nilai $\hat{Q}$ yang dipelajari sebelumnya. Di sini agen selalu memilih aksi dengan nilai ^ Q tertinggi. Hal ini menghasilkan konvergensi yang relatif cepat dari lintasan tertentu. Namun, jalan lain tetap tidak dikunjungi sampai akhir. Dalam kasus ekstrim maka kita dapat memperoleh kebijakan yang tidak optimal. Oleh karena itu, dalam Teorema 10.1 diperlukan bahwa setiap pasangan keadaan-aksi dikunjungi berkali-kali tak berhingga. Disarankan untuk menggunakan kombinasi eksplorasi dan eksploitasi dengan porsi eksplorasi yang tinggi di awal dan semakin berkurang seiring waktu.

Pilihan keadaan awal juga mempengaruhi kecepatan belajar. Dalam tiga gambar pertama pada Gambar 10.12 kita dapat dengan jelas melihat bahwa, untuk iterasi pertama, hanya nilai Q di sekitar pasangan keadaan-aksi yang diubah oleh imbalan langsung. Memulai lebih jauh dari titik semacam ini menghasilkan banyak pekerjaan yang tidak perlu. Ini menyarankan mentransfer pengetahuan sebelumnya tentang pasangan tindakan negara dengan hadiah langsung ke negara awal di dekat titik-titik ini. Dalam proses pembelajaran maka keadaan awal yang lebih jauh dapat dipilih.

10.8 PENDEKATAN, GENERALISASI DAN KONVERGENSI

Seperti yang telah dijelaskan Q-*learning* sejauh ini, sebuah tabel dengan semua nilai Q secara eksplisit disimpan dalam sebuah tabel. Ini hanya mungkin ketika bekerja dengan ruang keadaan berhingga dengan banyak aksi berhingga. Namun, jika ruang keadaan tak berhingga, misalnya dalam kasus variabel kontinu, maka tidak mungkin menyimpan semua nilai Q atau mengunjungi semua pasangan tindakan-keadaan selama pembelajaran.

Meskipun demikian ada cara sederhana menggunakan Q-*learning* dan iterasi nilai pada variabel kontinu. Tabel Q(s, a) diganti dengan jaringan saraf, misalnya jaringan backpropagation dengan variabel input s, a dan nilai Q sebagai output target. Untuk setiap pembaruan nilai Q, jaringan saraf disajikan contoh pelatihan dengan (s, a) sebagai input dan Q(s, a) sebagai target output. Pada akhirnya kita memiliki representasi terbatas dari fungsi Q(s, a). Karena kita hanya memiliki banyak contoh pelatihan berhingga, tetapi fungsi Q(s, a) didefinisikan untuk banyak input tak terhingga, dengan demikian kita secara otomatis memperoleh generalisasi jika ukuran jaringan dipilih dengan tepat (lihat Bab 9). Alih-alih jaringan saraf, kita juga dapat menggunakan algoritme pembelajaran terawasi lainnya atau aproksimator fungsi seperti mesin vektor pendukung atau proses Gaussian.

Namun, langkah dari banyak contoh pelatihan hingga fungsi kontinu bisa menjadi sangat mahal dalam situasi tertentu. Q-*learning* dengan aproksimasi fungsi mungkin tidak konvergen karena Teorema 10.1 hanya benar jika setiap pasangan state-action sering dikunjungi tak berhingga.

Namun, masalah konvergensi juga dapat muncul dalam kasus banyak pasangan aksi-negara saat Q-*learning* digunakan pada POMDP. Q-*learning* dapat diterapkan—dalam kedua varian yang dijelaskan—untuk proses Markov deterministik dan nondeterministik (MDP). Untuk POMDP dapat terjadi bahwa agen, karena sensor bising misalnya, merasakan banyak keadaan yang berbeda sebagai satu. Seringkali banyak negara bagian di dunia nyata sengaja dipetakan ke satu yang disebut observasi. Ruang observasi yang dihasilkan

kemudian jauh lebih kecil daripada ruang keadaan, dimana pembelajaran menjadi lebih cepat dan *overfitting* dapat dihindari

Namun, dengan menggabungkan beberapa keadaan, agen tidak dapat lagi membedakan antara keadaan sebenarnya, dan suatu tindakan dapat mengarahkannya ke banyak keadaan penerus yang berbeda, tergantung pada keadaan yang sebenarnya. Hal ini dapat menyebabkan masalah konvergensi untuk iterasi nilai atau untuk Q-*learning*. Dalam literatur banyak pendekatan berbeda untuk solusi yang disarankan.

Yang juga sangat menjanjikan adalah apa yang disebut metode perbaikan kebijakan dan metode gradien kebijakan turunannya, di mana nilai Q tidak diubah, melainkan kebijakan diubah secara langsung. Dalam skema ini, sebuah kebijakan dicari di ruang semua kebijakan, yang memaksimalkan hadiah diskon kumulatif ((Persamaan 10.1)). Salah satu kemungkinan untuk mencapai ini adalah dengan mengikuti gradien hadiah kumulatif secara maksimal. Kebijakan yang ditemukan dengan cara ini kemudian dengan jelas mengoptimalkan imbalan kumulatif.

10.9 APLIKASI

Utilitas praktis dari pembelajaran penguatan telah ditunjukkan berkali-kali. Dari sejumlah besar contoh ini, kami akan secara singkat menyajikan pilihan kecil. TD-*learning*, bersama dengan jaringan backpropagation dengan 40 hingga 80 neuron tersembunyi digunakan dengan sangat sukses di TD-gammon, sebuah program permainan backgammon. Hanya hadiah langsung untuk program ini adalah hasil akhir permainan. Versi program yang dioptimalkan dengan tampilan dua langkah dilatih melawan dirinya sendiri dalam 1,5 juta *game*. Itu kemudian mengalahkan pemain dan permainan kelas dunia serta tiga pemain manusia terbaik.

Ada banyak aplikasi dalam robotika. Misalnya, di Liga Simulasi Sepak Bola RoboCup, tim sepak bola robot terbaik sekarang berhasil menggunakan pembelajaran penguatan. Balancingapole, yang relatif mudah bagi manusia, telah berhasil diselesaikan berkali-kali dengan pembelajaran penguatan.

Demonstrasi yang mengesankan tentang kemampuan belajar robot diberikan oleh Russ Tedrake di IROS 2008 dalam presentasinya tentang model pesawat yang belajar mendarat di titik yang tepat, seperti burung yang mendarat di cabang. Karena arus udara menjadi sangat turbulen selama pendekatan pendaratan yang sangat dinamis, persamaan diferensial terkait, persamaan Navier-Stokes, tidak dapat dipecahkan. Oleh karena itu, pendaratan tidak dapat dikontrol dengan cara matematika klasik. Komentar Tedrake tentang ini:

"Burung tidak memecahkan Navier–Stokes!"

Burung dapat dengan jelas belajar terbang dan mendarat bahkan tanpa persamaan Navier–Stokes. Tedrake menunjukkan bahwa ini sekarang juga mungkin untuk pesawat terbang. Hari ini juga dimungkinkan untuk belajar mengendalikan mobil nyata hanya dalam 20 menit menggunakan Q-*learning* dan pendekatan fungsi. Contoh ini menunjukkan bahwa aplikasi industri nyata di mana beberapa pengukuran harus dipetakan ke tindakan dapat dipelajari dengan sangat baik dalam waktu singkat.

Robot nyata masih memiliki kesulitan belajar di ruang tindakan keadaan dimensi tinggi karena, dibandingkan dengan simulasi, robot nyata mendapatkan umpan balik dari lingkungan relatif lambat. Karena keterbatasan waktu, jutaan siklus pelatihan yang diperlukan tidak dapat direalisasikan. Di sini, selain algoritme pembelajaran cepat, diperlukan metode yang memungkinkan setidaknya sebagian pembelajaran terjadi secara offline, yaitu tanpa umpan balik dari lingkungan.

10.10 ALPHAGO, TEROBOSAN DI GO

Meskipun pemangkasan alfa-beta telah berhasil digunakan oleh komputer catur, itu tidak dapat mencapai keberhasilan yang sama dalam program Go karena faktor percabangan besar permainan sekitar 250, seperti yang dijelaskan sebelumnya. Telah diketahui selama beberapa waktu bahwa di Go, langkah selanjutnya harus dipilih menggunakan algoritma pengenalan pola pada posisi papan saat ini. Namun semua upaya sebelumnya tidak terlalu berhasil. Ini berubah ketika Google DeepMind menghadirkan AlphaGo. Program ini menggunakan pencarian pohon Monte Carlo (MCTS) untuk menghasilkan data pelatihan, pembelajaran mendalam untuk mengevaluasi posisi papan dan pembelajaran penguatan untuk meningkatkan strateginya dengan bermain melawan dirinya sendiri. Secara kasar sketsa, algoritma bekerja sebagai berikut:

Tujuan proses pembelajaran yang paling penting adalah mempelajari kebijakan p yang, untuk posisi s, menghitung probabilitas p(a|s) menang untuk setiap kemungkinan langkah a. Jika fungsi probabilitas ini ditemukan, maka itu akan digunakan dalam permainan sedemikian rupa sehingga ketika berada di posisi s, gerakan terbaik dari semua kemungkinan

$$a^* = arg\max_a \; p(a|s)$$

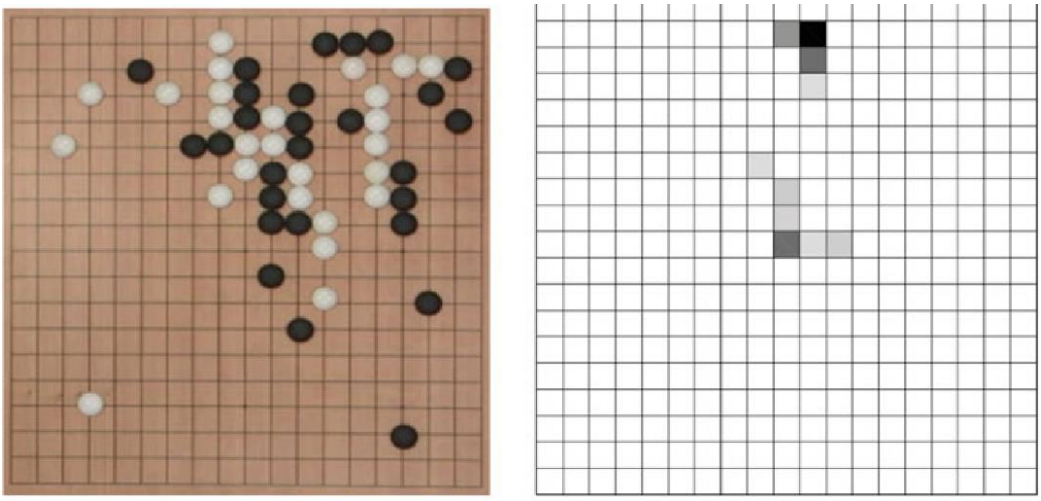
akan dipilih. Karena program pada akhirnya harus berperan lebih baik daripada panutannya sebagai manusia, ini adalah proses multi-tahap untuk mempelajari p kebijakan yang sebaik mungkin. Pertama, menggunakan *game* tingkat juara yang disimpan, dua fungsi pemilihan gerakan yang berbeda dipelajari (tahap 1). Yang lebih kuat dari keduanya kemudian akan lebih ditingkatkan menggunakan pembelajaran penguatan (tahap 2) dan diubah menjadi fungsi evaluasi posisi (tahap 3). Terakhir, program bermain dengan menggabungkan fungsi evaluasi posisi yang dihasilkan dengan dua kebijakan pemilihan langkah dari tahap 1 dalam proses pencarian MCTS yang kompleks.

Tahap 1: Pembelajaran Mendalam Berdasarkan Game yang Disimpan

KGS Go Server berisi banyak pertandingan Go lengkap oleh pemain juara dengan total tiga puluh juta posisi papan dan gerakan masing-masing pemain. Semua gerakan ini digunakan untuk melatih jaringan saraf convolutional (CNN).

CNN mengambil posisi dewan s sebagai input dan, setelah pelatihan, harus mengembalikan probabilitas $P_\sigma(a|s)$ untuk semua langkah hukum a. Realisasi de facto dari $P_\sigma(a|s)$ adalah matriks 19 x 19, yang mewakili papan. Nilai untuk masing-masing dari 361 poin di papan menunjukkan kemungkinan menang untuk langkah yang sesuai. Contoh penerapan fungsi kebijakan tersebut dapat dilihat pada Gambar 10.13. Untuk keadaan papan di sebelah kiri, $P_\sigma(a|s)$ direpresentasikan dalam grafik kepadatan di sebelah kanan. Pada giliran ini, pemain dapat meletakkan batu putih di salah satu titik. Untuk hampir semua poin yang tersedia, probabilitas menang adalah nol, tetapi untuk beberapa poin menarik, probabilitas direpresentasikan sebagai nilai skala abu-abu. Rupanya tempat yang paling diinginkan untuk meletakkan batu adalah di tengah atas di baris 2, kolom 11.

Selama pelatihan, status papan 19 19 untuk masing-masing dari tiga puluh juta gerakan juara dalam database diberikan sebagai input ke jaringan, dan output target adalah matriks 19 x 19 dengan semua nilai disetel ke nol dan kotak dipilih oleh ahli diatur ke satu. Pada kumpulan data uji yang belum terlihat, jaringan CNN tiga belas lapis kompleks yang digunakan oleh AlphaGo memilih langkah yang benar hingga 57% dari waktu, yang menunjukkan peningkatan yang mengesankan atas kinerja terbaik hingga saat ini sekitar 44% oleh program Go lainnya.



Gambar 10.13 Grafik kepadatan (kanan) dari probabilitas $P_\sigma(a|s)$ bergerak untuk pemain dengan batu putih dalam keadaan papan yang ditunjukkan di sebelah kiri. Untuk setiap kotak putih, $P_\sigma(a|s) = 0$. Semakin gelap alun-alun, semakin tinggi nilai untuk pergerakan putih ke titik kisi yang sesuai

Selain kebijakan $P_\sigma(a|s)$, kebijakan peluncuran yang lebih sederhana dilatih pada data pelatihan yang sama dengan CNN yang lebih sederhana. Ia memilih gerakan dengan benar hanya 24% dari waktu, tetapi melakukan perhitungan sekitar 1000 kali lebih cepat.

Tahap 2: Memperbaiki Kebijakan yang Dipelajari dengan Reinforcement Learning

Selanjutnya, fungsi pemilihan gerak ditingkatkan menggunakan pembelajaran penguatan. Langkah kedua ini diperlukan agar program dapat bermain lebih baik daripada panutannya sebagai manusia. Kebijakan $P_\sigma(a|s)$ yang dipelajari dari database digunakan oleh AlphaGo untuk bermain melawan dirinya sendiri. Setelah setiap pertandingan, kebijakan saat ini $P_\sigma(a|s)$ ditingkatkan menggunakan penurunan gradien stokastik. Untuk menghindari *overfitting*, AlphaGo tidak selalu bermain melawan versi saat ini, melainkan melawan versi sebelumnya yang dipilih secara acak.

Tahap 3: Mempelajari Nilai Posisi

Selanjutnya, $P_\sigma(a|s)$ kebijakan saat ini digunakan untuk melatih fungsi evaluasi status dewan V(s) menggunakan iterasi nilai.

Kebijakan Game Final AlphaGo dan Performanya

Algoritme permainan AlphaGo yang sebenarnya menggunakan algoritme MCTS yang kompleks di mana pohon diperluas sedikit dari posisi saat ini. Dari simpul daun yang dihasilkan, permainan dimainkan sampai akhir menggunakan kebijakan peluncuran yang cepat namun sederhana $P_\pi(a|s)$. Posisi tersebut kemudian dievaluasi menggunakan hasil simulasi permainan bersama dengan fungsi nilai V(s). Meskipun memiliki kebijakan yang sangat baik, biaya komputasinya sangat besar karena kompleksitas algoritma MCTS ini. Kemampuan bermain tertinggi AlphaGo, yang digunakan untuk mengalahkan Go grandmaster Lee Sedol dari Korea 4:1, dicapai pada komputer paralel dengan 1202 CPU dan 176 GPU

Singkatnya, AlphaGo merupakan tonggak besar dalam sejarah AI. Pencapaian ini dimungkinkan dengan menggunakan pembelajaran mendalam dan dengan upaya rekayasa yang menakjubkan dari tim ahli yang besar di berbagai bidang pembelajaran mesin.

10.11 KUTUKAN DIMENSI

Meskipun sukses dalam beberapa tahun terakhir, pembelajaran penguatan tetap menjadi bidang penelitian aktif dalam AI, paling tidak karena bahkan algoritma pembelajaran terbaik yang dikenal saat ini masih tidak praktis untuk keadaan dan ruang tindakan dimensi tinggi karena waktu komputasi yang sangat besar. Masalah ini dikenal sebagai "kutukan dimensi".

Dalam mencari solusi untuk masalah ini, para ilmuwan mengamati hewan dan manusia selama belajar. Di sini kita melihat bahwa belajar di alam terjadi pada banyak tingkat abstraksi. Seorang bayi pertama kali belajar keterampilan motorik dan bahasa sederhana pada tingkat terendah. Ketika ini dipelajari dengan baik, mereka disimpan dan nantinya dapat dipanggil kapan saja dan digunakan. Diterjemahkan ke dalam bahasa ilmu komputer, ini berarti bahwa setiap kemampuan yang dipelajari dikemas dalam sebuah modul dan kemudian, pada tingkat yang lebih tinggi, mewakili suatu tindakan. Dengan menggunakan tindakan kompleks seperti itu pada tingkat yang lebih tinggi, ruang tindakan menjadi sangat berkurang dan dengan demikian pembelajaran dipercepat. Dengan cara yang sama, keadaan dapat diabstraksikan dan dengan demikian ruang keadaan dapat diperkecil. Pembelajaran pada berbagai tingkatan ini disebut pembelajaran hierarkis [BM03].

Pendekatan lain untuk modularisasi pembelajaran adalah pembelajaran terdistribusi, atau pembelajaran multi-agen. Ketika mempelajari keterampilan motorik robot humanoid, hingga 50 motor yang berbeda harus dikontrol secara bersamaan, yang menghasilkan ruang keadaan 50 dimensi dan juga ruang aksi 50 dimensi. Untuk mengurangi kompleksitas raksasa ini, kontrol pusat diganti dengan kontrol terdistribusi. Misalnya, setiap motor individu bisa mendapatkan kontrol individu yang mengarahkannya secara langsung, jika mungkin secara independen dari motor lainnya. Di alam, kami menemukan kontrol semacam ini pada serangga. Misalnya, banyak kaki kaki seribu tidak dikendalikan oleh otak pusat, melainkan setiap pasang kaki memiliki "otak" kecilnya sendiri.

Mirip dengan pencarian kombinatorial tanpa informasi, pembelajaran penguatan memiliki tugas menemukan yang terbaik dari sejumlah besar kebijakan. Tugas belajar menjadi lebih mudah secara signifikan jika agen memiliki kebijakan yang kurang lebih baik sebelum pembelajaran dimulai. Maka tugas belajar yang berdimensi tinggi dapat diselesaikan lebih cepat. Tapi bagaimana kita menemukan kebijakan awal seperti itu? Di sini ada dua kemungkinan utama.

Kemungkinan pertama adalah pemrograman klasik. Pemrogram menyediakan agen dengan kebijakan yang terdiri dari program yang dia anggap baik. Kemudian terjadi peralihan, misalnya ke Q-*learning*. Agen memilih, setidaknya pada awal pembelajaran, tindakannya sesuai dengan kebijakan yang diprogramkan dan dengan demikian dibawa ke area "menarik" dari ruang tindakan negara. Hal ini dapat menyebabkan pengurangan dramatis dalam ruang pencarian pembelajaran penguatan.

Jika pemrograman tradisional menjadi terlalu rumit, kita dapat mulai melatih robot atau agen dengan meminta manusia untuk melarang tindakan yang benar. Dalam kasus yang paling sederhana, ini dilakukan dengan kendali jarak jauh robot secara manual. Robot kemudian menyimpan tindakan terlarang untuk setiap keadaan dan menggeneralisasi menggunakan algoritma pembelajaran yang diawasi seperti backpropagation atau pembelajaran pohon keputusan. Ini disebut pembelajaran demonstrasi dengan demikian juga memberikan kebijakan awal untuk pembelajaran penguatan berikutnya.

Ringkasan dan Pandangan

Hari ini kami memiliki akses ke algoritma pembelajaran yang berfungsi dengan baik dan mapan untuk melatih mesin kami. Tugas pelatih atau pengembang manusia, bagaimanapun, masih menuntut untuk aplikasi yang kompleks. Ada banyak kemungkinan bagaimana menyusun pelatihan robot dan itu tidak akan berhasil tanpa eksperimen. Eksperimen ini bisa sangat membosankan dalam praktiknya karena setiap proyek pembelajaran baru harus dirancang dan diprogram. Alat diperlukan di sini yang, selain berbagai algoritma pembelajaran, juga menawarkan pelatih kemampuan untuk menggabungkannya dengan pemrograman tradisional dan pembelajaran demonstrasi. Salah satu yang pertama dari jenis alat ini adalah Teaching-Box, yang selain perpustakaan program yang luas juga menawarkan template untuk konfigurasi proyek pembelajaran dan untuk komunikasi antara robot dan lingkungan. Misalnya, guru manusia dapat memberikan robot umpan balik lebih lanjut dari keyboard atau melalui antarmuka ucapan selain umpan balik dari lingkungan.

Pembelajaran penguatan adalah bidang penelitian yang menarik dan aktif yang akan semakin banyak digunakan di masa depan. Semakin banyak sistem kontrol robot, tetapi juga program lain, akan belajar melalui umpan balik dari lingkungan. Saat ini terdapat banyak variasi dari algoritma yang disajikan dan juga algoritma yang sama sekali berbeda. Masalah penskalaan tetap belum terpecahkan. Untuk tindakan kecil dan ruang keadaan dengan beberapa derajat kebebasan, hasil yang mengesankan dapat dicapai. Jika jumlah derajat kebebasan dalam ruang keadaan bertambah menjadi 18, misalnya untuk robot humanoid sederhana, maka pembelajaran menjadi sangat mahal.

Untuk kuliah dasar lebih lanjut, kami merekomendasikan pengenalan ringkas ke dalam pembelajaran penguatan dalam buku Tom Mitchell. Pekerjaan standar oleh Sutton dan Barto menyeluruh dan komprehensif, seperti artikel survei oleh Kaelbling, Littman dan Moore.

10.12 LATIHAN

Latihan 10.1

(a) Hitung jumlah kebijakan yang berbeda untuk n keadaan dan n tindakan. Dengan demikian transisi dari setiap keadaan ke setiap keadaan dimungkinkan.
(b) Bagaimana jumlah kebijakan berubah dalam submasalah (a) jika tindakan kosong, yaitu tindakan dari satu negara ke negara itu sendiri, tidak diperbolehkan.
(c) Dengan menggunakan diagram panah seperti pada Gambar 10.3, berikan semua kebijakan untuk dua keadaan.
(d) Dengan menggunakan diagram panah, berikan semua kebijakan tanpa tindakan kosong untuk tiga keadaan.

Latihan 10.2

Gunakan iterasi nilai secara manual pada Contoh 10.1 dengan $n_x = n_y = 2$.

Latihan 10.3

Lakukan berbagai eksperimen menggunakan simulator iterasi nilai.

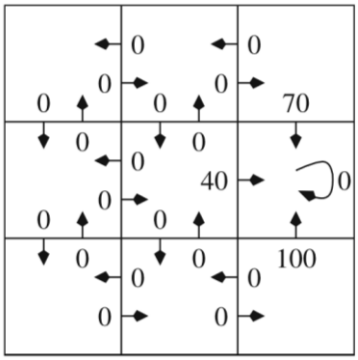
(a) Instal simulator iterasi nilai dari M. Tokic 2006.
(b) Reproduksi hasil dari Latihan 10.2 dengan terlebih dahulu memasukkan umpan balik dengan editor umpan balik dan kemudian melakukan iterasi nilai.
(c) Model permukaan dengan kehalusan yang berbeda dan amati bagaimana kebijakan berubah.

(d) Dengan matriks umpan balik yang serupa, perbesar ruang keadaan secara bertahap hingga sekitar 100 x 100 dan sesuaikan dengan faktor diskon c sedemikian rupa sehingga menghasilkan kebijakan yang masuk akal.

Latihan 10.4

Tunjukkan bahwa untuk contoh perhitungan pada Gambar 10.8 nilai eksaknya adalah $V^*(3, 3) = 1.9/(1 - 0.9^6) \approx 4.05499$.

Latihan 10.5



Kerjakan Q-*learning* pada kisi 3 x 3 di sebelah kanan. Keadaan di kanan tengah adalah keadaan tujuan yang menyerap.

Latihan 10.6

Sebuah lengan robot dengan *n* sendi (dimensi) dan *l* status diskrit per sendi diberikan. Tindakan dari setiap status ke setiap status dimungkinkan (jika robot tidak melakukan apa-apa, ini dievaluasi sebagai tindakan (kosong).

(a) Berikan rumus untuk jumlah keadaan dan jumlah tindakan di setiap keadaan dan jumlah kebijakan untuk robot.

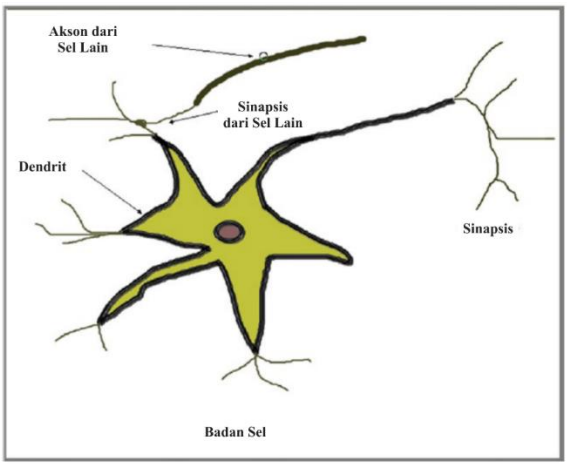
(b) Buatlah tabel dengan jumlah strategi untuk n = 1, 2, 3, 4, 8 dan *l* = 1, 2, 3, 4, 10.

(c) Untuk mengurangi jumlah kemungkinan strategi, asumsikan bahwa jumlah kemungkinan tindakan per sambungan selalu sama dengan 2 dan robot hanya dapat menggerakkan satu sambungan pada satu waktu. Berikan rumus baru untuk jumlah strategi dan buat tabel terkait.

(d) Dengan hasil perhitungan, justifikasi bahwa suatu agen yang beroperasi secara otonom dan adaptif dengan n = 8 dan l = 10 pasti dapat disebut cerdas.

BAB 11
ARSITEKTUR *NEURAL NETWORK*

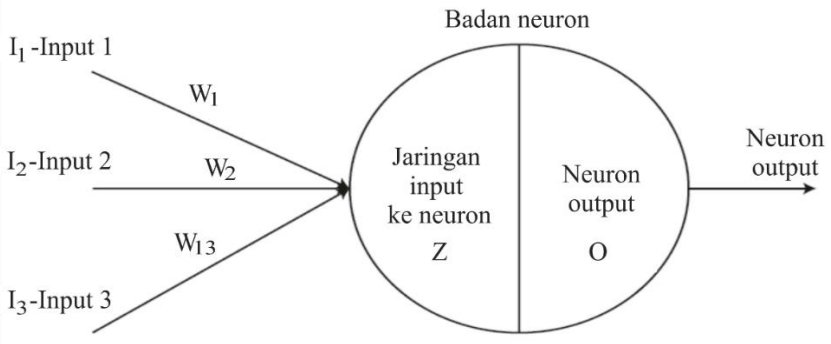
Arsitektur jaringan saraf kecerdasan buatan secara skematis meniru jaringan otak manusia. Ini terdiri dari lapisan neuron yang terhubung secara terarah satu sama lain. Gambar 11.1 menunjukkan gambar skema neuron manusia.



Gambar 11.1 Gambar skema dari neuron manusia

11.1 NEURON BIOLOGIS DAN BUATAN

Neuron biologis (pada tingkat yang disederhanakan) terdiri dari badan sel dengan nukleus, akson, dan sinapsis. Sinapsis menerima impuls, yang diproses oleh badan sel. Badan sel mengirimkan respons melalui akson ke sinapsisnya, yang terhubung ke neuron lain. Meniru neuron biologis, neuron buatan terdiri dari badan neuron dan koneksi ke neuron lain (lihat Gambar 11.2).



Gambar 11.2 Neuron buatan tunggal

Setiap input ke neuron diberi bobot, W. Bobot yang diberikan ke neuron menunjukkan dampak input ini dalam penghitungan output jaringan. Jika bobot yang diberikan pada neuron1 (W_1) lebih besar daripada bobot yang diberikan pada neuron2 (W_2), maka pengaruh input dari neuron1 pada output jaringan lebih signifikan daripada dari neuron2.

Tubuh neuron digambarkan sebagai lingkaran yang dibagi menjadi dua bagian oleh garis vertikal. Bagian kiri disebut input jaringan ke neuron, dan ini menunjukkan bagian dari perhitungan yang dilakukan oleh badan neuron. Bagian ini biasanya ditandai pada diagram

jaringan sebagai Z. Misalnya, nilai Z untuk neuron yang ditunjukkan pada Gambar 11.2 dihitung sebagai jumlah dari setiap input ke neuron dikalikan dengan bobot yang sesuai dan akhirnya menambahkan bias. Itu adalah bagian linear dari perhitungan (Persamaan 11-1).

Persamaan 11.1

$$Z = W_1*I_1 + W_2*I_2 + W_3*I_3 + B_1$$

11.2 FUNGSI AKTIVASI

Untuk menghitung output O dari neuron yang sama (Gambar 11.2), Anda dapat menerapkan fungsi nonlinier khusus (disebut fungsi aktivasi, σ) ke bagian linier perhitungan Z (Persamaan 11-2).

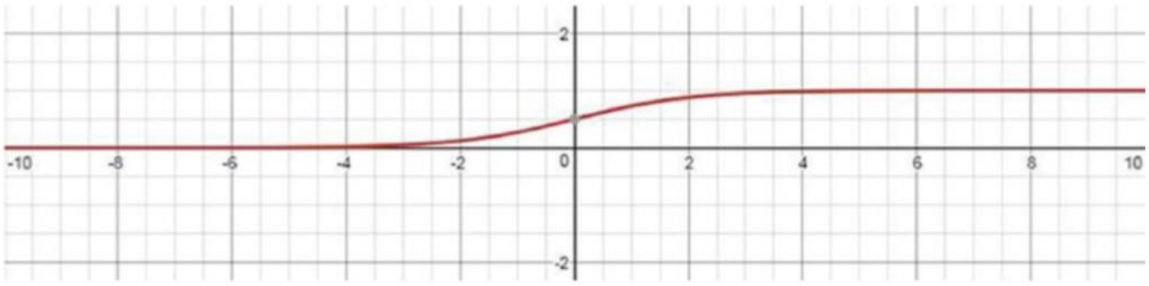
Persamaan 11.2

$$O = \sigma(Z)$$

Ada banyak fungsi aktivasi yang digunakan untuk jaringan. Penggunaannya tergantung pada berbagai faktor, seperti interval di mana mereka berperilaku baik (tidak jenuh), seberapa cepat fungsi berubah ketika argumennya berubah, dan hanya preferensi Anda. Mari kita lihat salah satu fungsi aktivasi yang paling sering digunakan; itu disebut sigmoid. Fungsi tersebut memiliki rumus yang ditunjukkan pada Persamaan 11-3.

Persamaan 11.3

$$\sigma(Z) = \frac{1}{1 + e^{-z}}$$



Gambar 11.3 menunjukkan grafik fungsi aktivasi sigmoid.

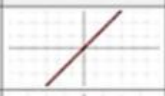
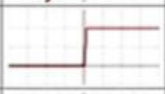
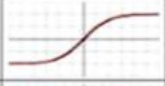
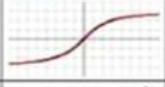
Seperti yang ditunjukkan pada grafik pada Gambar 11.3, fungsi sigmoid (terkadang juga disebut fungsi logistik) berperilaku paling baik pada interval [-1, 1]. Di luar interval ini, ia cepat jenuh, artinya nilainya praktis tidak berubah dengan perubahan argumennya. Itulah sebabnya (seperti yang akan Anda lihat di semua contoh buku ini) data input jaringan biasanya dinormalisasi pada interval [-1, 1].

Beberapa fungsi aktivasi berperilaku baik pada interval [0, 1], sehingga data input juga dinormalisasi pada interval [0, 1]. Gambar 11.4 mencantumkan fungsi aktivasi yang paling sering digunakan. Ini termasuk nama fungsi, plot, persamaan, dan turunan. Informasi ini akan berguna ketika Anda mulai menghitung berbagai bagian dalam jaringan.

Menggunakan fungsi aktivasi tertentu tergantung pada fungsi yang didekati dan pada banyak kondisi lainnya. Banyak publikasi terbaru menyarankan penggunaan tanh sebagai fungsi aktivasi untuk lapisan tersembunyi, menggunakan fungsi aktivasi linier untuk regresi, dan menggunakan fungsi softmax untuk klasifikasi. Sekali lagi, ini hanya rekomendasi umum. Saya sarankan Anda bereksperimen dengan berbagai fungsi aktivasi untuk proyek Anda dan memilih yang menghasilkan hasil terbaik.

Untuk contoh buku ini, saya menemukan secara eksperimental bahwa fungsi aktivasi tanh bekerja paling baik. Ini juga berperilaku baik (sebagai fungsi aktivasi sigmoid)

pada interval [-1,1], tetapi laju perubahannya pada interval ini lebih cepat daripada fungsi sigmoid. Ini juga jenuh lebih lambat. Saya menggunakan tanh di hampir semua contoh buku ini.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Gambar 11.4 Fungsi aktivasi

11.3 RINGKASAN

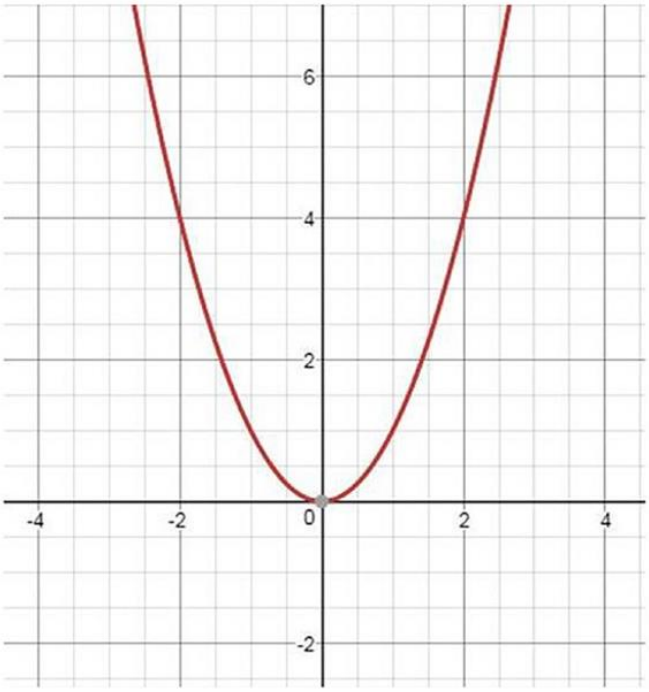
Bab ini memperkenalkan Anda pada bentuk kecerdasan buatan yang disebut jaringan saraf. Ini menjelaskan semua konsep penting dari jaringan saraf seperti lapisan, neuron, koneksi, bobot, dan fungsi aktivasi. Bab ini juga menjelaskan konvensi yang digunakan untuk menggambar diagram jaringan saraf. Bab berikut menunjukkan semua detail pemrosesan jaringan neuron dengan menjelaskan cara menghitung semua hasil jaringan secara manual. Untuk mempermudah, dua istilah—jaringan saraf dan jaringan—akan digunakan secara bergantian untuk sisa buku ini.

BAB 12
MEKANISME PEMROSESAN JARINGAN *NEURAL INTERNAL*

Bab ini membahas cara kerja dalam pemrosesan jaringan saraf. Ini menunjukkan bagaimana jaringan dibangun, dilatih, dan diuji.

12.1 FUNGSI UNTUK DIPERKIRAKAN

Mari kita pertimbangkan fungsi $y(x) = x^2$, seperti yang ditunjukkan pada Gambar 12.1; namun, kita akan berpura-pura bahwa rumus fungsi tidak diketahui dan bahwa fungsi tersebut diberikan kepada kita oleh nilainya di empat titik.



Gambar 12.1 Grafik fungsi

Tabel 12.1 mencantumkan nilai fungsi pada empat titik.

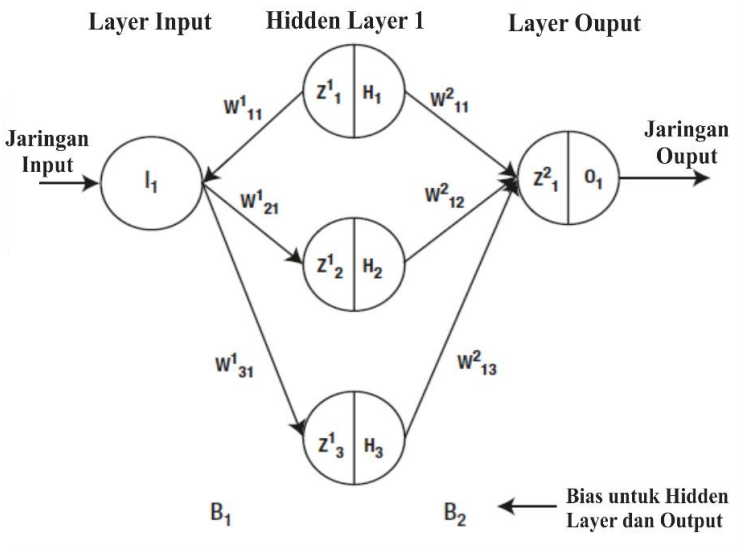
Tabel 12.1 Nilai Fungsi Diberikan di Empat Titik

x	*F(x)*
1	1
3	9
5	25
7	49

Dalam bab ini, Anda akan membangun dan melatih jaringan yang dapat digunakan untuk memprediksi nilai fungsi pada beberapa argumen (x) yang tidak digunakan untuk pelatihan. Untuk dapat memperoleh nilai fungsi pada titik yang tidak terlatih (tetapi dalam rentang latihan), Anda harus terlebih dahulu memperkirakan fungsi ini. Ketika aproksimasi selesai, Anda kemudian dapat menemukan nilai fungsi di sembarang tempat. Untuk itulah jaringan digunakan, karena jaringan adalah mekanisme pendekatan universal.

12.2 ARSITEKTUR JARINGAN

Bagaimana jaringan dibangun? Sebuah jaringan dibangun dengan memasukkan lapisan-lapisan neuron. Lapisan pertama di sebelah kiri adalah *lapisan input*, dan berisi neuron yang menerima input dari luar. Lapisan terakhir di sebelah kanan adalah *output layer*, dan berisi neuron yang membawa *jaringan output*. Satu atau lebih lapisan tersembunyi terletak di antara lapisan *input* dan *output*. Neuron lapisan tersembunyi digunakan untuk melakukan sebagian besar perhitungan selama aproksimasi fungsi. Gambar 12.2 menunjukkan diagram jaringan.



Gambar 12.2 Arsitektur jaringan saraf

Koneksi ditarik dari neuron di lapisan sebelumnya ke neuron di lapisan berikutnya. Setiap neuron pada lapisan sebelumnya terhubung ke semua neuron pada lapisan berikutnya. Setiap koneksi membawa bobot, dan setiap bobot diberi nomor dengan dua indeks. Indeks pertama adalah jumlah neuron penerima, dan indeks kedua adalah jumlah neuron pengirim. Misalnya, hubungan antara neuron kedua di lapisan tersembunyi (H_2) dan satu-satunya neuron di lapisan input (I_1) diberi bobot W^1_{21} . Superscript 1 menunjukkan nomor lapisan dari neuron pengirim. Setiap lapisan diberi bias, yang mirip dengan bobot yang diberikan ke neuron tetapi diterapkan ke tingkat lapisan. Bias membuat bagian linier dari setiap perhitungan output neuron lebih fleksibel dalam mencocokkan topologi fungsi yang didekati.

Ketika pemrosesan jaringan dimulai, nilai awal untuk bobot dan bias biasanya ditetapkan secara acak. Biasanya, untuk menentukan jumlah neuron di lapisan tersembunyi, Anda menggandakan jumlah neuron di lapisan input dan menambahkan jumlah neuron di lapisan output. Dalam kasus kami, itu adalah (1*2+1 = 3), atau tiga neuron. Jumlah lapisan tersembunyi yang akan digunakan dalam jaringan tergantung pada kompleksitas fungsi yang akan didekati. Biasanya, satu lapisan tersembunyi cukup untuk fungsi kontinu yang mulus, dan lebih banyak lapisan tersembunyi diperlukan untuk topologi fungsi yang lebih kompleks. Dalam praktiknya, jumlah lapisan dan neuron di lapisan tersembunyi yang mengarah ke hasil aproksimasi terbaik ditentukan secara eksperimental. Pemrosesan jaringan terdiri dari dua lintasan: lintasan maju dan lintasan mundur. Dalam passing depan, perhitungan bergerak dari kiri ke kanan. Untuk setiap neuron, jaringan mendapatkan input ke neuron dan menghitung output dari neuron.

12.3 PERHITUNGAN *FORWARD-PASS*

Perhitungan berikut memberikan output dari neuron H_1, H_2, dan H_3:

Neuron H_1

$$Z_1^1 = W_{11}^1 * I_1 + B_1 * 1$$
$$H_1 = \sigma(Z_1^1)$$

Neuron H_2

$$Z_2^1 = W_{21}^1 * I_1 + B_1 * 1$$
$$H_2 = \sigma(Z_2^1)$$

Neuron H_3

$$Z_3^1 = W_{31}^1 * I_1 + B_1 * 1$$
$$H_3 = \sigma(Z_3^1)$$

Nilai-nilai ini digunakan saat memproses neuron di lapisan berikutnya (dalam hal ini, lapisan output):

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_1 * 1$$
$$O_1 = \sigma(Z_1^2)$$

Perhitungan yang dilakukan pada lintasan pertama memberikan output jaringan (disebut nilai prediksi jaringan). Saat melatih jaringan, Anda menggunakan output yang diketahui untuk titik pelatihan yang disebut nilai aktual atau target. Dengan mengetahui nilai output yang harus dihasilkan jaringan untuk input yang diberikan, Anda dapat menghitung kesalahan jaringan, yaitu selisih antara nilai target dan kesalahan yang dihitung jaringan (nilai prediksi). Untuk fungsi yang ingin Anda aproksimasi pada contoh ini, nilai aktual (target) ditampilkan pada Tabel 12.2, kolom 2.

Tabel 12.2 Input Data Set untuk Contoh

x	*F(x)*
1	1
3	9
5	25
7	49

Perhitungan dilakukan untuk setiap record dalam kumpulan data input. Misalnya, pemrosesan record pertama dari kumpulan data input dilakukan dengan menggunakan rumus berikut.

Record input 1

Berikut adalah rumus untuk record input pertama:

Neuron H_1

$$Z_1^1 = W_{11}^1 * I_1 + B_1 * 1.00 = W_{11}^1 * 1.00 + B_1 * 1.00$$
$$H_1 = \sigma(Z_1^1)$$

Neuron H_2

$$Z_2^1 = W_{21}^1 * I_1 + B_1 * 1.00 = W_{21}^1 * 1.00 + B_1 * 1.00$$
$$H_2 = \sigma(Z_2^1)$$

Neuron H_3

$$Z_3^1 = W_{31}^1 * I_1 + B_1 * 1.00 = W_{31}^1 * 1.00 + B_1 * 1.00$$
$$H_3 = \sigma(Z_3^1)$$

Neuron O_1

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$
$$O_1 = \sigma(Z_1^2)$$

Berikut adalah kesalahan untuk catatan 1:

$$O_1 = \sigma(Z_1^2) \text{ Nilai target untuk Rekam } O_1 = \sigma(Z_1^2) - 100$$

Record input 2

Berikut adalah rumus untuk record input kedua:

Neuron H_1

$$Z_1^1 = W_{11}^1 * I_1 + B_1 * 1.00 = W_{11}^1 * 3.00 + B_1 * 1.00$$
$$H_1 = \sigma(Z_1^1)$$

Neuron H_2

$$Z_2^1 = W_{21}^1 * I_1 + B_1 * 1.00 = W_{21}^1 * 3.00 + B_1 * 1.00$$
$$H_2 = \sigma(Z_2^1)$$

Neuron H_3

$$Z_3^1 = W_{31}^1 * I_1 + B_1 * 1.00 = W_{31}^1 * 3.00 + B_1 * 1.00$$
$$H_3 = \sigma(Z_3^1)$$

Neuron O_1

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$
$$O_1 = \sigma(Z_1^2)$$

Berikut adalah kesalahan untuk catatan 2:

$$E_1 = \sigma(Z_1^2) - \text{ Nilai target untuk Rekam } 2 = \sigma(Z_1^2) - 9.00$$

Record input 3

Berikut adalah rumus untuk record input ketiga:

Neuron H_1

$$Z_1^1 = W_{11}^1 * I_1 + B_1 * 1.00 = W_{11}^1 * 5.00 + B_1 * 1.00$$
$$H_1 = \sigma(Z_1^1)$$

Neuron H_2

$$Z_2^1 = W_{21}^1 * I_1 + B_1 * 1.00 = W_{21}^1 * 5.00 + B_1 * 1.00$$
$$H_2 = \sigma(Z_2^1)$$

Neuron H_3

$$Z_3^1 = W_{31}^1 * I_1 + B_1 * 1.00 = W_{31}^1 * 5.00 + B_1 * 1.00$$
$$H_3 = \sigma(Z_3^1)$$

Neuron O_1

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$
$$O_1 = \sigma(Z_1^2)$$

Berikut adalah kesalahan untuk catatan 3:

$$E_1 = \sigma(Z_1^2) - \text{ Nilai target untuk Rekam } 3 = \sigma(Z_1^2) - 25.00$$

Record input 4

Berikut adalah rumus untuk record input keempat:

Neuron H_1

$$Z_1^1 = W_{11}^1 * I_1 + B_1 * 1.00 = W_{11}^1 * 7.00 + B_1 * 1.00$$
$$H_1 = \sigma(Z_1^1)$$

Neuron H_2

$$Z_2^1 = W_{21}^1 * I_1 + B_1 * 1.00 = W_{21}^1 * 7.00 + B_1 * 1.00$$
$$H_2 = \sigma(Z_2^1)$$

Neuron H_3

$$Z_3^1 = W_{31}^1 * I_1 + B_1 * 1.00 = W_{31}^1 * 7.00 + B_1 * 1.00$$
$$H_3 = \sigma(Z_3^1)$$

Neuron O_1

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$
$$O_1 = \sigma(Z_1^2)$$

Berikut adalah kesalahan untuk catatan 2:

$$E_1 = \sigma(Z_1^2) - \text{ Nilai target untuk Rekam } 4 = \sigma(Z_1^2) - 49.00$$

Ketika semua record telah diproses untuk batch ini (dan batch di sini adalah seluruh set pelatihan), titik dalam pemrosesan itu disebut epoch. Pada titik itu, Anda dapat mengambil rata-rata kesalahan jaringan untuk semua catatan—seperti dalam E =(E1 + E2 + E3 + E4)/4— dan itu adalah kesalahan pada zaman saat ini. Rata-rata kesalahan mencakup setiap tanda kesalahan. Jelas, kesalahan pada epoch pertama (dengan bobot/bias yang dipilih secara acak) akan terlalu besar untuk aproksimasi fungsi yang baik; oleh karena itu, Anda perlu mengurangi kesalahan ini ke nilai yang dapat diterima (diinginkan) yang disebut *batas kesalahan* (ERROR LIMIT), yang Anda tetapkan di awal pemrosesan. Pengurangan kesalahan jaringan dilakukan di jalur mundur (disebut juga *backpropagation*). Batas kesalahan ditentukan secara eksperimental. Batas kesalahan diatur ke kesalahan minimum yang dapat dicapai jaringan, tetapi tidak mudah. Jaringan harus bekerja keras untuk mencapai batas kesalahan seperti itu. Batas kesalahan dijelaskan secara lebih rinci dalam banyak contoh kode di seluruh buku ini.

12.4 PERHITUNGAN *BACKPROPAGATION-PASS*

Bagaimana kesalahan jaringan dapat dikurangi? Jelas, bobot awal dan nilai bias ditetapkan secara acak, dan mereka tidak baik. Mereka menyebabkan kesalahan yang signifikan untuk Epoch 1. Anda perlu menyesuaikannya sedemikian rupa sehingga nilai barunya akan mengarah ke kesalahan perhitungan jaringan yang lebih kecil. Backpropagation melakukan ini dengan mendistribusikan ulang kesalahan antara semua neuron jaringan di output dan lapisan tersembunyi dan dengan menyesuaikan nilai bobot awal mereka. Penyesuaian juga dilakukan untuk setiap lapisan bias. Untuk menyesuaikan bobot setiap neuron, Anda menghitung turunan parsial dari fungsi kesalahan sehubungan dengan output neuron. Misalnya, turunan parsial yang dihitung untuk neuron O_1 adalah $\frac{\partial E}{\partial O_1}$. Karena turunan parsial menunjuk ke arah nilai fungsi yang meningkat (tetapi Anda perlu menurunkan nilai fungsi kesalahan), maka penyesuaian bobot harus dilakukan dalam arah yang berlawanan.

$$\textbf{Nilai bobot yang disesuaikan = nilai bobot asli} - \eta * \frac{\partial E}{\partial O_1}$$

Di sini η adalah tingkat pembelajaran untuk jaringan, dan ini mengontrol seberapa cepat jaringan belajar. Nilainya biasanya diatur antara 0,1 dan 1,0.

Perhitungan serupa dilakukan untuk bias setiap lapisan. Untuk bias B1, jika turunan parsial yang dihitung adalah$\frac{\partial E}{\partial B_1}$, maka bias yang disesuaikan dihitung sebagai berikut:

$$\textbf{Nilai bias yang disesuaikan} - \textbf{Nilai bias asli } B_1 - \eta * \frac{\partial E}{\partial B_1} =$$

Dengan mengulangi perhitungan ini untuk setiap neuron jaringan dan setiap bias lapisan, Anda mendapatkan satu set nilai bobot/bias baru yang disesuaikan. Memiliki serangkaian nilai bobot/bias baru, Anda kembali ke lintasan maju dan menghitung output jaringan baru menggunakan bobot/bias yang disesuaikan. Anda juga menghitung ulang kesalahan output jaringan.

Karena Anda menyesuaikan bobot/bias ke arah yang berlawanan dengan gradien (turunan parsial), kesalahan penghitungan jaringan yang baru akan berkurang. Anda mengulangi umpan maju dan mundur dalam satu lingkaran hingga kesalahan menjadi kurang dari batas kesalahan. Pada saat itu, jaringan dilatih, dan Anda menyimpan jaringan terlatih pada disk. Jaringan yang dilatih mencakup semua parameter bobot dan bias yang mendekati nilai fungsi yang diprediksi dengan tingkat presisi yang dibutuhkan. Di bab berikutnya, Anda akan mempelajari cara memproses beberapa contoh secara manual dan

melihat semua perhitungan mendetail. Namun, sebelum melakukan itu, Anda perlu menyegarkan kembali pengetahuan Anda tentang turunan fungsi dan gradien.

12.5 FUNGSI DERIVATIF DAN DIVERGEN FUNGSI

Derivatif (turunan) dari suatu fungsi didefinisikan sebagai berikut:

$$\frac{\partial f}{\partial x} = lim_{n \to 0} \frac{f(x + dx) - f(x)}{dx}$$

di mana:

∂x adalah perubahan kecil dalam argumen fungsi.

f(x) adalah nilai fungsi sebelum mengubah argumen.

f(x + ∂x) adalah nilai fungsi setelah mengubah argumen.

Turunan fungsi menunjukkan laju perubahan fungsi variabel tunggal f(x) di titik x. Gradien adalah turunan (laju perubahan) dari fungsi multivariabel f(x, y, z) di titik (x, y, z). Gradien fungsi multivariabel f(x, y, z) adalah produk dari komponen yang dihitung untuk setiap arah ($\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$). Setiap komponen disebut turunan parsial dari fungsi f(x, y, z) terhadap variabel spesifik (arah) x, y, z.

Gradien pada setiap titik fungsi selalu menunjuk ke arah kenaikan terbesar suatu fungsi. Pada maksimum lokal atau minimum lokal, gradiennya adalah nol karena tidak ada arah kenaikan tunggal di lokasi tersebut. Saat Anda mencari fungsi minimum (misalnya, untuk fungsi kesalahan) yang ingin Anda perkecil, Anda bergerak ke arah yang berlawanan dengan gradien.

Ada beberapa aturan untuk menghitung turunan:

Ini adalah aturan pangkat:

$$\frac{\partial}{\partial x}(u^a) = a * u^{a-1} * \frac{\partial u}{\partial x}$$

Ini adalah aturan produk:

$$\frac{\partial (u * v)}{\partial x} = u * \frac{\partial v}{\partial x} + v * \frac{\partial u}{\partial x}$$

Ini adalah aturan hasil bagi:

$$\frac{\partial f}{\partial x}\left(\frac{u}{v}\right) = \frac{v * \frac{\partial u}{\partial x} - u * \frac{\partial v}{\partial x}}{v^2}$$

Aturan rantai memberi tahu Anda cara membedakan fungsi komposit.

Ini menyatakan bahwa

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} * \frac{\partial u}{\partial x}, dimana\ u = f(x)$$

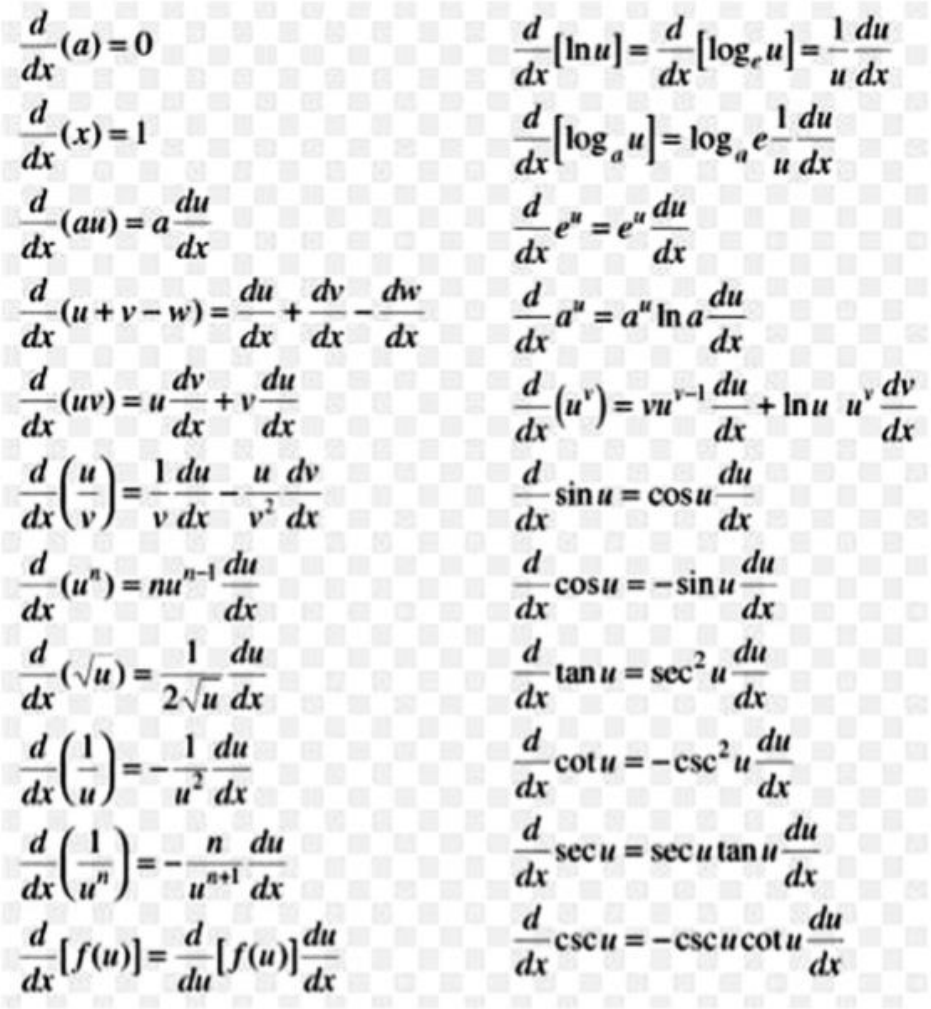
Berikut ini contohnya: $y = u^8$. $u = x^2 + 5$.

Menurut aturan rantai,

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} * \frac{\partial u}{\partial x} = 8u^7 * 2x * x(x^2 + 6)^7$$

Fungsi Derivatif/Turunan yang Paling Umum Digunakan

Gambar 12.3 mencantumkan turunan fungsi yang paling umum digunakan.



$$\frac{d}{dx}(a)=0 \qquad \frac{d}{dx}[\ln u]=\frac{d}{dx}[\log_e u]=\frac{1}{u}\frac{du}{dx}$$

$$\frac{d}{dx}(x)=1 \qquad \frac{d}{dx}[\log_a u]=\log_a e\frac{1}{u}\frac{du}{dx}$$

$$\frac{d}{dx}(au)=a\frac{du}{dx} \qquad \frac{d}{dx}e^u=e^u\frac{du}{dx}$$

$$\frac{d}{dx}(u+v-w)=\frac{du}{dx}+\frac{dv}{dx}-\frac{dw}{dx} \qquad \frac{d}{dx}a^u=a^u\ln a\frac{du}{dx}$$

$$\frac{d}{dx}(uv)=u\frac{dv}{dx}+v\frac{du}{dx} \qquad \frac{d}{dx}\left(u^v\right)=vu^{v-1}\frac{du}{dx}+\ln u\ u^v\frac{dv}{dx}$$

$$\frac{d}{dx}\left(\frac{u}{v}\right)=\frac{1}{v}\frac{du}{dx}-\frac{u}{v^2}\frac{dv}{dx} \qquad \frac{d}{dx}\sin u=\cos u\frac{du}{dx}$$

$$\frac{d}{dx}(u^n)=nu^{n-1}\frac{du}{dx} \qquad \frac{d}{dx}\cos u=-\sin u\frac{du}{dx}$$

$$\frac{d}{dx}(\sqrt{u})=\frac{1}{2\sqrt{u}}\frac{du}{dx} \qquad \frac{d}{dx}\tan u=\sec^2 u\frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{1}{u}\right)=-\frac{1}{u^2}\frac{du}{dx} \qquad \frac{d}{dx}\cot u=-\csc^2 u\frac{du}{dx}$$

$$\frac{d}{dx}\left(\frac{1}{u^n}\right)=-\frac{n}{u^{n+1}}\frac{du}{dx} \qquad \frac{d}{dx}\sec u=\sec u\tan u\frac{du}{dx}$$

$$\frac{d}{dx}[f(u)]=\frac{d}{du}[f(u)]\frac{du}{dx} \qquad \frac{d}{dx}\csc u=-\csc u\cot u\frac{du}{dx}$$

Gambar 12.3 Turunan fungsi paling umum dipakai

Hal ini juga membantu untuk mengetahui turunan dari fungsi aktivasi sigmoid karena sering digunakan dalam langkah backpropagation dari pemrosesan jaringan.

σ(Z) = 1/(1+exp(Z))

$$\frac{\partial 6(Z)}{\partial Z}=Z*(1-Z)$$

Turunan dari fungsi aktivasi sigmoid memberikan laju perubahan fungsi aktivasi pada setiap neuron.

12.6 RINGKASAN

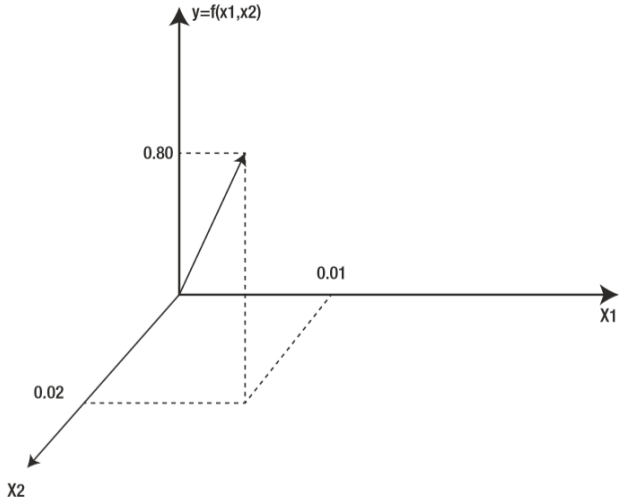
Bab ini mengeksplorasi mesin bagian dalam pemrosesan jaringan saraf dengan menjelaskan bagaimana semua hasil pemrosesan dihitung. Ini memperkenalkan Anda pada turunan dan gradien dan menjelaskan bagaimana konsep-konsep ini digunakan dalam menemukan salah satu fungsi kesalahan minimum. Bab berikutnya menunjukkan contoh sederhana di mana setiap hasil dihitung secara manual. Menjelaskan aturan perhitungan saja tidak cukup untuk memahami subjek karena menerapkan aturan pada arsitektur jaringan tertentu sangat rumit.

BAB 13
PEMROSESAN JARINGAN SARAF MANUAL

Dalam bab ini, Anda akan belajar tentang internal pemrosesan jaringan saraf dengan melihat contoh sederhana. Saya akan memberikan penjelasan rinci langkah demi langkah tentang perhitungan yang terlibat dalam pemrosesan lintasan propagasi maju dan mundur. **Catatan**: Semua perhitungan dalam bab ini didasarkan pada informasi di Bab 2. Jika Anda memiliki masalah dalam membaca Bab 3, lihat Bab 2 untuk penjelasannya.

Contoh 1: Perkiraan Manual untuk suatu Fungsi pada Satu Titik

Gambar 13.1 menunjukkan vektor dalam ruang 3-D



Gambar 13.1 Vektor dalam ruang 3-D

Vektor mewakili nilai fungsi y=f(x1,x2), di mana x1 = 0,01 dan x2 = 0,02.

Y(0.01, 0.02) = 0.80

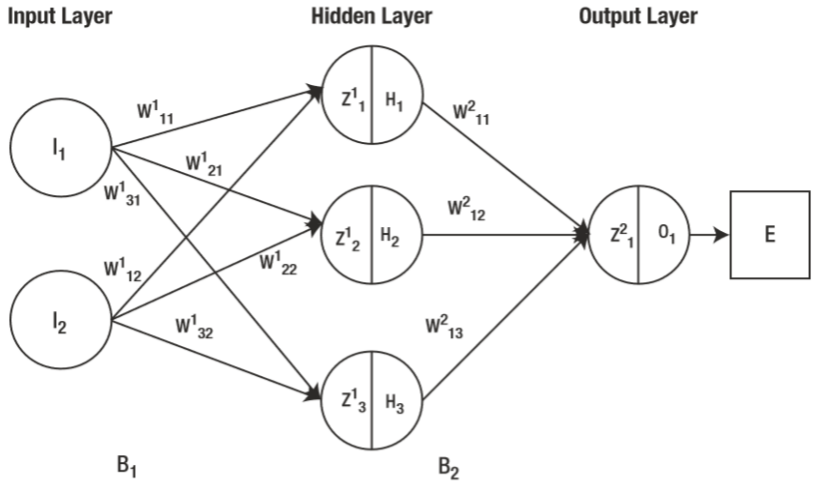
13.1 MEMBANGUN *NEURAL NETWORK*/JARINGAN SARAF

Untuk contoh ini, katakanlah Anda ingin membangun dan melatih jaringan yang untuk input tertentu (x1=0,01, x2=0,02) menghitung hasil output y = 0,80 (nilai target untuk jaringan). Gambar 13.2 menunjukkan diagram jaringan sebagai contoh.

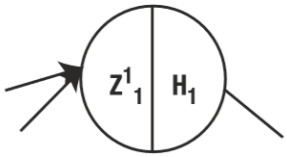
Jaringan memiliki tiga lapisan neuron (input, hidden, dan output). Terdapat dua neuron (I_1 dan I_2) pada lapisan input, tiga neuron (H_1, H_2, H_3) pada lapisan tersembunyi, dan satu neuron (O1) pada lapisan output. Bobot digambarkan di dekat panah yang menunjukkan tautan (koneksi) antara neuron (misalnya, neuron I1 dan I2 memberikan input untuk neuron H_1 dengan bobot yang sesuai $W^1{}_{11}$ dan $W^1{}_{12}$.

Badan neuron di lapisan tersembunyi dan lapisan output (H_1, H_2, H_3, dan O_1) ditampilkan sebagai lingkaran yang dibagi menjadi dua bagian (lihat Gambar 3-3). Bagian kiri badan neuron menunjukkan nilai input jaringan yang dihitung untuk neuron $Z_1^1 = W_{11}^1 * I_1 + W_{12}^1 * I_2 + B_1 * 1$. Nilai awal untuk bias biasanya diatur ke 1,00. Output neuron dihitung dengan menerapkan fungsi aktivasi sigmoid ke input jaringan ke dalam neuron.

$H_1 = \sigma(W_{12}^1) = 1/(1 + \exp(-Z_1^1)$

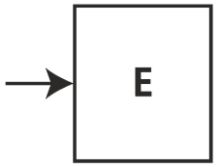


Gambar 13.2 Diagram jaringan



Gambar 13.3 Presentasi neuron di lapisan tersembunyi dan output

Perhitungan fungsi error ditampilkan sebagai kuadran karena bukan neuron (Gambar 13.4).



Gambar 13.4 Representasi *error function/*fungsi kesalahan

B1 dan B2 adalah bias untuk lapisan jaringan yang sesuai. Berikut ini ringkasan pengaturan jaringan awal:

- Input ke neuron I_1 = 0,01.
- Input ke neuron I_2 = 0,02.
- T_1 - (Target output dari neuron O_1) = 0.80

Anda juga perlu menetapkan nilai awal untuk parameter bobot dan bias. Nilai parameter awal biasanya ditetapkan secara acak, tetapi untuk contoh ini (di mana semua perhitungan dilakukan secara manual) Anda menetapkan nilai berikut:

$W_{11}^1 = 0.05 \quad W_{12}^1 = 0.06 \quad W_{21}^1 = 0.07 \quad W_{22}^1 = 0.08 \quad W_{31}^1 = 0.09 \quad W_{32}^1 = 0.10$

$W_{11}^2 = 0.11 \quad W_{12}^2 = 0.12 \quad W_{13}^2 = 0.13$

$B_1 = 0.20$

$B_2 = 0.25$

Batas kesalahan untuk contoh ini diatur ke 0,01.

13.2 PERHITUNGAN *FORWARD-PASS*

Perhitungan forward-pass dimulai dari lapisan tersembunyi.

Lapisan Tersembunyi

Untuk neuron H1, berikut langkah-langkahnya:

1. Menghitung total input bersih untuk neuron H_1 (Persamaan 3-1).

Persamaan 13.1

$$Z_1^1 = W_{11}^1 * I_1 + W_{11}^1 * I_1 + B_1 * 1.00$$
$$= 0.05 * 0.01 + 0.06 * 0.02 * 1.00 = 0.2017000000000000$$

2. Menggunakan fungsi logistik untuk mendapatkan output dari H_1 (Persamaan 13-2).

Persamaan 13.2

$$H_1 = \delta(Z_1^1) = \left(\frac{1}{1 + exp(Z_1^1))}\right) = \left(\frac{1}{1 + exp - 0.2017000000000000))}\right)$$
$$= 0.5502547397403884$$

Lihat Persamaan 13-3 untuk neuron H_2.

Persamaan 13.3

$$Z_2^1 = W_{21}^1 * I_1 + W_{21}^1 * I_1 + W_{22}^1 * I_2 + B_1$$
$$= 0.07 * 0.01 + 0.08 * 0.02 * 1.00$$
$$= 0.2023$$
$$H_2 = \frac{1}{(1 + \exp(-0.2023))} = 0.5504032199355139$$

Lihat Persamaan 13-4 untuk neuron H_3.

Persamaan 13.4

$$Z_1^1 = W_{11}^1 * I_1$$

Layer Output

Perhitungan neuron O_1 lapisan output mirip dengan perhitungan neuron lapisan tersembunyi tetapi dengan satu perbedaan: input untuk neuron output O_1 adalah output dari neuron lapisan tersembunyi yang sesuai. Juga, perhatikan bahwa ada tiga neuron lapisan tersembunyi yang berkontribusi pada neuron lapisan output O_1.

Berikut langkah-langkah untuk neuron O1:

1. Menghitung total input bersih untuk neuron O (Persamaan 13-5).

Persamaan 13.5

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$
$$= 0.11 * 0.5502547397403884 + 0.12 * 0.5504032199355139$$
$$+0.13 * 0.55055169115025556 + 0.25 * 1.00 = 0.44814812761323763$$

2. Gunakan fungsi logistik untuk mendapatkan output dari O_1 (Persamaan 13-6).

Persamaan 13.6

$$O_1 = \sigma(Z_1^2) = \left(\frac{1}{1 + exp(-Z_1^2))}\right) = \left(\frac{1}{1 + exp - 0.4481481276132376))}\right)$$
$$= 0.6101988445912522$$

Output yang dihitung dari neuron O_1 adalah 0.6101988445912522, sedangkan target output dari neuron O_1 harus = 0.80; oleh karena itu, galat kuadrat untuk output untuk neuron O_1 adalah seperti yang ditunjukkan pada Persamaan 13-7.

Persamaan 13.7

$$E = 0.5*(T_1 - O_1)^2 = 0.5 * (0.80 - 0.6101988445912522) = 0.01801223929724783$$

Dalam rumus ini, pengali 0,5 digunakan untuk meniadakan eksponen selama perhitungan turunan. Untuk alasan efisiensi, kerangka Encog (yang akan Anda pelajari dan gunakan nanti dalam buku ini) memindahkan kuadrat ke nanti dalam perhitungan. Apa yang diperlukan di sini adalah meminimalkan kesalahan perhitungan jaringan untuk mendapatkan hasil aproksimasi yang baik. Hal ini dilakukan dengan mendistribusikan kesalahan jaringan antara bobot dan bias neuron output dan lapisan tersembunyi, sambil mempertimbangkan bahwa dampak setiap neuron pada output jaringan bergantung pada bobotnya. Perhitungan ini dilakukan pada lintasan perambatan mundur.

Untuk mendistribusikan ulang kesalahan ke semua neuron output dan lapisan tersembunyi dan menyesuaikan bobotnya, Anda perlu memahami seberapa besar nilai kesalahan akhir berubah ketika bobot untuk setiap neuron berubah. Hal yang sama berlaku untuk bias untuk setiap lapisan. Dengan mendistribusikan ulang kesalahan jaringan ke semua neuron output dan lapisan tersembunyi, Anda sebenarnya menghitung penyesuaian untuk setiap bobot neuron dan setiap bias lapisan.

13.3 PERHITUNGAN *BACKWARD-PASS*

Perhitungan bobot dan penyesuaian bias untuk setiap neuron/lapisan jaringan dilakukan dengan bergerak mundur (dari kesalahan jaringan ke lapisan output dan kemudian dari lapisan output ke lapisan tersembunyi).

Menghitung Penyesuaian Bobot untuk Neuron Lapisan Keluar

Mari kita hitung penyesuaian bobot untuk neuron W_{11}^2 . Seperti yang sudah Anda ketahui, turunan parsial dari suatu fungsi menentukan dampak dari perubahan kecil dalam argumen fungsi kesalahan pada perubahan nilai fungsi yang sesuai. Menerapkannya ke neuron W_{11}^2 , di sini Anda ingin tahu bagaimana perubahan pada W_{11}^2 mempengaruhi kesalahan jaringan E. Untuk melakukan ini, Anda perlu menghitung turunan parsial dari fungsi kesalahan E terhadap W_{11}^2 yang mana $\frac{\partial E}{\partial W_{11}^2}$

Menghitung Penyesuaian untuk W_{11}^2

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial W_{11}^2}$ dapat dinyatakan dengan Persamaan 3-8.

Persamaan 13.8

$$\frac{\partial E}{\partial W_{11}^2} = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial W_{11}^2}$$

Mari kita hitung secara terpisah setiap bagian persamaan menggunakan kalkulus turunan (Persamaan 13-9).

Persamaan 13.9

$$E = 0.5 * (T_1 - O_1)^2$$

$$\frac{\partial E}{\partial O_1} = 2 * 0.5 * (T_{1\_} O_1) * \frac{\partial (0.5(T_{1\_} O_1)}{\partial O_1} = (T_{1\_} O_1) * (-1) = (T_1 - O_1)$$

$\frac{\partial O_1}{\partial Z_1^2}$ adalah turunan dari fungsi aktivasi sigmoid dan sama dengan Persamaan 13-10.

Persamaan 13.10

$O_1 * (1 - O)$ = 0.6101988445912522 * (1 – 0.6101988445912522)
= 0.23785621465075305

Lihat Persamaan 3-11 dan Persamaan 13-12 untuk menghitung $\frac{\partial Z_1^2}{\partial Z_{11}^2}$

Persamaan 13.11

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$

Persamaan 13.12

$$\frac{\partial Z_1^2}{\partial W_{11}^2} = H_1 = 0.5502547397403884$$

Catatan : $\frac{\partial Z_1^2 \, (W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00}{\partial W_{11}^2} = 0$ $karena\ bagian\ ini\ tidak\ tergantung\ pasa\ W_{11}^2$

Mari kita gabungkan semuanya (Persamaan 13-13).

Persamaan 13.13

$$\frac{\partial E}{\partial W_{11}^2} = -0.18980115540874787 * 0.23785621465075305 * 0.5502547397403884$$
$$= -0.024841461722517316$$

Untuk mengurangi kesalahan, Anda perlu menghitung nilai penyesuaian baru untuk W_{11}^2 dengan mengurangkan nilai $\frac{\partial E}{\partial W_{11}^2}$ (secara opsional dikalikan dengan beberapa tingkat pembelajaran η) dari nilai awal W_{11}^2 (Persamaan 3-14).

Persamaan 13.14

$$Disesuaikan\ W_{11}^2 = W_{11}^2 - \eta * \frac{\partial E}{\partial W_{11}^2}\ untuk\ contoh\ ini, \eta = 1$$

$$Disesuaikan\ W_{11}^2 = 0.11 + 0.024841461722517316 = 0.1348414672251732$$

Menghitung Penyesuaian untuk W_{11}^2

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial W_{12}^2}$dapat dinyatakan dengan Persamaan 3-15.

Persamaan 13.15

$$\frac{\partial E}{\partial W_{11}^2} = \frac{\partial E}{\partial O_1} * \frac{O_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial W_{11}^2}$$

Mari kita hitung secara terpisah setiap bagian persamaan menggunakan kalkulus turunan (Persamaan 13-16).

Persamaan 13.16

$$\frac{\partial E}{\partial O_1} = -0.18980115540874787\ (lihat\ persamaan\ 1.13)$$

Lihat Persamaan 13-17 dan Persamaan 13-18 untuk menghitung $\frac{\partial O_1}{\partial Z_1^2}$

Persamaan 13.17

$$\frac{\partial O_1}{\partial Z_1^2} = 0.23785621465075305\ (Lihat\ persamaan\ 1.14)$$

Lihat Persamaan 13-18 dan Persamaan 13-19 untuk menghitung $\frac{\partial Z_1^2}{\partial W_{12}^2}$

Persamaan 13.18

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$

Persamaan 13.19

$$\frac{\partial Z_1^2}{\partial W_{12}^2} = H_2 = 0.5504032199355139$$

Mari kita gabungkan semuanya (Persamaan 13-20).

Persamaan 13.20

$$\frac{\partial E}{\partial W_{11}^2} = -0.18980115540874787 * 0.23785621465075305 * 0.5502547397403884$$
$$= -0.0024841461722517316$$

Untuk mengurangi kesalahan, Anda perlu menghitung nilai baru yang disesuaikan untuk W_{11}^2 dengan mengurangkan nilai $\frac{\partial E}{\partial W_{11}^2}$ (opsional dikalikan dengan beberapa tingkat pembelajaran) dari nilai asli W_{11}^2 (Persamaan 13-21).

Persamaan 13.21

$$Disesuaikan\ W_{11}^2 = W_{11}^2 - \eta * \frac{\partial E}{\partial W_{11}^2}$$
$$Disesuaikan\ W_{11}^2 = 0.12 + 0.024841461722517316$$
$$= 0.1448414617225173$$

Menghitung Penyesuaian untuk W_{13}^2

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial W_{11}^3}$ dapat dinyatakan dengan Persamaan 13-22 dan Persamaan 13-23.

$$\frac{\partial E}{\partial W_{13}^2} = \frac{\partial E}{\partial O_1} * \frac{O_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial W_{13}^2}$$

Persamaan 13.22

$$\frac{\partial E}{\partial O_1} = -0.18980115540874787 \ (lihat\ persamaan\ 1.13)$$

Persamaan 13.23

$$\frac{O_1}{\partial Z_1^2} = 0.23785621465075305$$

Lihat Persamaan 13-24 dan Persamaan 13-25 untuk menghitung $\frac{\partial Z_1^2}{\partial W_{13}^2}$

Persamaan 13.24

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_2 * 1.00$$

Persamaan 3.25

$$\frac{\partial Z_1^2}{\partial W_{13}^2} = H_3 = 0.5505516911502556 \ (lihat\ persamaan\ 11.4)$$

Mari kita gabungkan semuanya (Persamaan 13-26 dan Persamaan 13-27).

Persamaan 13.26

$$\frac{\partial E}{\partial W_{13}^2} = -0.18980115540874787 * 0.23785621465075305 * 0.5505516911502556$$
$$= -0.024854867708052567$$

Persamaan 13.27

$$Disesuaikan\ W_{13}^2 = W_{13}^2 - \eta * \frac{\partial E}{\partial W_{13}^{\ 2}}. Untuk\ contoh\ ini, \eta = 1$$
$$Disesuaikan\ W_{11}^2 = 0.13 + 0.024841461722517316$$
$$= 0.1548144617225173$$

Oleh karena itu, pada iterasi kedua, Anda akan menggunakan nilai yang disesuaikan dengan bobot berikut:

$$Disesuaikan\ W_{13}^2 = 0.08515853827748268$$
$$Disesuaikan\ W_{13}^2 = 0.0951585382774828$$
$$Disesuaikan\ W_{13}^2 = 0.10515853827748269$$

Setelah menyesuaikan bobot untuk neuron output, Anda siap untuk menghitung penyesuaian bobot untuk neuron tersembunyi.

Menghitung Penyesuaian Berat untuk Neuron Lapisan Tersembunyi

Menghitung penyesuaian bobot untuk neuron di lapisan tersembunyi mirip dengan perhitungan yang sesuai di lapisan output tetapi dengan satu perbedaan penting. Untuk neuron di lapisan output, input sekarang terdiri dari hasil output dari neuron yang sesuai di lapisan tersembunyi.

Menghitung Penyesuaian untuk W_{11}^2

Menerapkan aturan rantai untuk turunan, X dapat dinyatakan dengan Persamaan 13-28, 13-29, 13-30, dan 13-31.

Persamaan 13.28

$$\frac{\partial E}{\partial W_{11}^1} = \frac{\partial E}{\partial H_1} * \frac{\partial H_1}{\partial Z_1^1} * \frac{\partial Z_1^1}{\partial W_{11}^1}$$

Persamaan 13.29

$$\frac{\partial E}{\partial H_1} = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^1} = -0.18980115540874787 * 0.23785621465075305$$
$$= -0.04514538436186407 \ (lihat\ persamaan\ 1.13\ dan\ 1.14)$$

Persamaan 13.30

$$\frac{\partial H_1}{\partial Z_1^1} = \sigma(H_1) = H_1 * (1 - H_1)$$
$$= 0.0.5502547397403884 * (1 - 0.5502547397403884)$$
$$= 0.24747446113362584$$

Persamaan 13.31

$$\frac{\partial Z_1}{\partial W_1^1} = \frac{\partial (W_{11}^1 * I_1 + W_{11}^1 * I_1 + B_1 * I)}{\partial W_{11}^1} = I_1 = 0.01$$

Mari kita gabungkan semuanya (Persamaan 13-32, 13-33, dan 13-34).

Persamaan 13.32

$$\frac{\partial E}{\partial W_{11}^1} = -0.04514538436186407 * 0.24747446113362584 * 0.01$$
$$= -0.0001117232966762273$$

Persamaan 13.33

$$Disesuaikan\ W_{11}^1 = W_{11}^1 - \eta * \frac{\partial E}{\partial W_{11}^1} = 0.05 - 0.0001117232966762273$$
$$= 0.049888276703323776$$

Persamaan 13.34

$$Disesuaikan\ W_{11}^1 = W_{11}^1 - \eta * \frac{\partial E}{\partial W_{11}^1} = 0.05 + 0.0001117232966762273$$
$$= 0.05011172329667623$$

Menghitung Penyesuaian untuk W_{12}^1

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial W_{12}^1}$ dapat dinyatakan dengan Persamaan 13-35, 13-36, dan 13-37).

Persamaan 13.35

$$\frac{\partial E}{\partial W_{12}^1} = \frac{\partial E}{\partial H_1} * \frac{\partial H_1}{\partial Z_1^1} * \frac{\partial Z_1^1}{\partial W_{12}^1}$$

$$\frac{\partial E}{\partial H_1} = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^1} = -0.18980115540874787 * 0.237856214650753\ .05$$
$$= -0.04514538436186407 \ (lihat\ prssamaan\ 1.13\ dan\ 1.14)$$

Persamaan 13.36

$$\frac{\partial H_1}{\partial Z_1^1} = 0.24747446113362584$$

Persamaan 13.37

$$\frac{\partial Z_1^1}{\partial W_{12}^1} = \frac{\partial (W_{11}^1 * I_1 + W_{12}^1 * I_1 + B_1 * I)}{\partial W_{12}^1} = I_2 = 0.02$$

Mari kita gabungkan semuanya (Persamaan 13-38 dan 13-39).

Persamaan 13.38

$$\frac{\partial E}{\partial Z_{12}^1} = = -0.04514538436186407 * 0.24747446113362584 * 0.02$$
$$= -00022344659335245464$$

Persamaan 13.39

$$Disesuaikan\ W_{12}^1 = W_{12}^1 - \eta * \frac{\partial E}{\partial W_{12}^1} = 0.06 + 0.00022344659335245464$$
$$= 0.06022344659335245$$

Menghitung Penyesuaian untuk W_{21}^1

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial W_{21}^1}$ dapat dinyatakan dengan Persamaan 13-40, 13-41, 13-42, dan 13-43.

Persamaan 3.40

$$\frac{\partial E}{\partial W_{21}^1} = \frac{\partial E}{\partial H_2} * \frac{\partial H_2}{\partial Z_2^1} * \frac{\partial Z_2^1}{\partial W_{21}^1}$$

Persamaan 13.41

$$\frac{\partial E}{\partial H_2} * \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial W_2^1} = -0.18980115540874787 * 0.23785621465075305$$
$$= 0.04514538436186407\ (Lihat\ Persamaan\ 3.9\ dan\ Persamaan\ 3.10)$$

Persamaan 13.42

$$\frac{\partial H_2}{\partial Z_2^1} = H_2 * (1 - H_2) = 0.5504032199355139 * 0.5504032199355139$$
$$= 0.059776553406647545$$

Persamaan 13.43

$$\frac{\partial Z_1^1}{\partial W_{21}^1} = \frac{\partial (W_{21}^1 * I_1 + W_{21}^1 * I_2 + B_1 * I)}{\partial W_{21}^1} = I_1 = 0.01$$

Mari kita gabungkan semuanya (Persamaan 3-44 dan 3-45).

Persamaan 13.44

$$\frac{\partial E}{\partial W_{21}^1} = -0.04514538436186407 * 0.059776553406647545 * 0.01$$
$$= -0.00026986354793705983$$

Persamaan 4.45

$$Disesuaikan\ W_{21}^1 = W_{21}^1 - \eta * \frac{\partial E}{\partial W_{21}^1} = 0.07 + 0.00026986354793705983$$
$$= 0.07002698635479371$$

Menghitung Penyesuaian untuk W_{22}^1

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial W_{22}^1}$ dapat dinyatakan dengan Persamaan 13-46, 13-47, 13-48, dan 13-49.

Persamaan 13.46

$$\frac{\partial E}{\partial W_{22}^1} = \frac{\partial E}{\partial H_2} * \frac{\partial H_2}{\partial Z_2^1} * \frac{\partial Z_2^1}{\partial W_{22}^1}$$

Persamaan 13.47

$$\frac{\partial E}{\partial H_2} * \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial W_2^1} = -0.04514538436186407\ (Lihat\ Persamaan\ 3.9\ dan\ 3.10)$$

Persamaan 13.48

$$\frac{\partial H_2}{\partial Z_2^1} = H_2 * (1 - H_2) = 0.5504032199355139 * (1 - 0.5504032199355139)$$
$$= 0.059776553406647545$$

Persamaan 13.49

$$\frac{\partial Z_1^1}{\partial W_{22}^1} = \frac{\partial (W_{22}^1 * I_1 + W_{22}^1 * I_2 + B_1 * I)}{\partial W_{22}^1} = I_2 = 0.02$$

Mari kita gabungkan semuanya (Persamaan 13-50 dan 13-51).

Persamaan 13.50

$$\frac{\partial E}{\partial W_{21}^1} = -0.04514538436186407 * 0.059776553406647545 * 0.02$$
$$= -0.00005397270958741 1966$$

Persamaan 13.51

$$Disesuaikan\ W_{22}^1 = W_{22}^1 - \eta * \frac{\partial E}{\partial W_2^1} = 0.08 + 0.00005397270958741 1966$$
$$= 0.08005397270958742$$

Menghitung Penyesuaian untuk W_{31}^1

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial H_{21}^1}$ dapat dinyatakan dengan Persamaan 13-52, 13-53, 13-54, dan 13-55.

Persamaan 13.52

$$\frac{\partial E}{\partial W_{31}^1} = \frac{\partial E}{\partial H_3} * \frac{\partial H_3}{\partial Z_3^1} * \frac{\partial Z_3^1}{\partial W_{31}^1}$$

Persamaan 13.53

$$\frac{\partial E}{\partial H_3} * \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial W_3^1} = -0.04514538436186407\ (Lihat\ persamaan\ 11.13\ dan\ 11.14)$$

Persamaan 13.54

$$\frac{\partial H_3}{\partial Z_3} = H_3 * (1 - H_3) = 0.5505516911502556 * (1 - 0.5505516911502556)$$
$$= 0.24744452652184917$$

Persamaan 13.55

$$\frac{\partial Z_3^1}{\partial W_{31}^1} = \frac{\partial (W_{31}^1 * I_1 + W_{32}^1 * I_2 + B_1 * I)}{\partial W_{31}^1} = I_1 = 0.01$$

Mari kita gabungkan semuanya (Persamaan 13-56 dan 13-57).

Persamaan 13.56

$$\frac{\partial E}{\partial W_{31}^1} = -0.04514538436186407 * 0.24744452652184917 * 0.01$$
$$= -0.0001117097825806835$$

Persamaan 13.57

$$Disesuaikan\ W_{31}^1 = W_{31}^1 - \eta * \frac{\partial E}{\partial W_{31}^1} = 0.09 + 0.0001117097825806835$$
$$= 0.09011170978258086$$

Menghitung Penyesuaian untuk W_{32}^1

Menerapkan aturan rantai untuk turunan, $\frac{\partial E}{\partial H_{31}^1}$ dapat dinyatakan dengan Persamaan 13-58, 13-59, 13-60, dan 13-61.

Persamaan 13.58

$$\frac{\partial E}{\partial W_{32}^1} = \frac{\partial E}{\partial H_3} * \frac{\partial H_3}{\partial Z_3^1} * \frac{\partial Z_3^1}{\partial W_{32}^1}$$

Persamaan 13.59

$$\frac{\partial E}{\partial H_3} * \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial W_3^1} = -0.0451453843618 6407(Lihat\ Persamaan\ 13.49)$$

Persamaan 13.60

$$\frac{\partial H_3}{\partial Z_3^1} = H_3 * (1 - H_3) = 0.5505516911502556 * (1 - 0.5505516911502556)$$
$$= 0.24744452652184917 (Lihat\ Persamaan\ 13.50)$$

Persamaan 13.61

$$\frac{\partial Z_3^1}{\partial W_{32}^1} = \frac{\partial (W_{31}^1 * I_1 + W_{32}^1 * I_2 + B_1 * I)}{\partial W_{32}^1} = I_2 = 0.02$$

Mari kita gabungkan semuanya (Persamaan 13-62 dan1 3-63).

Persamaan 13.62

$$\frac{\partial E}{\partial W_{32}^1} = -0.04514538436186407 * 0.24744452652184917 * 0.02$$
$$= -0.000223419565161367$$

Persamaan 13.63

$$Disesuaikan\ W_{32}^1 = W_{32}^1 - \eta * \frac{\partial E}{\partial W_{32}^1} = 0.10 + 0.000223419565161367$$
$$= 0.10022341956516137$$

13.4 MEMPERBARUI BIAS JARINGAN

Anda perlu menghitung penyesuaian kesalahan untuk bias B1 dan B2. Sekali lagi, dengan menggunakan aturan rantai, lihat Persamaan 13-64 dan 13-65.

Persamaan 13.64

$$\frac{\partial E}{\partial B_1} = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^1} * \frac{\partial Z_1^1}{\partial B_1}$$

Persamaan 13.65

$$\frac{\partial E}{\partial B_2} = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial B_2}$$

Hitung tiga bagian dari rumus sebelumnya untuk kedua ekspresi (Persamaan 13-66, 13-67, 13-68, dan 13-69).

Persamaan 13.66

$$\frac{\partial Z_1^1}{\partial B_1} = \frac{\partial (W_{11}^1 * I_1 + W_{12}^1 * I_2 + B_1 * I)}{\partial B_1} = 1$$

Persamaan 13.67

$$\frac{\partial Z_1^2}{\partial 2} = \frac{\partial (W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_1 * I)}{\partial B_2} = 1$$

Persamaan 13.68

$$\frac{\partial E}{\partial B_1} = \frac{\partial E}{\partial H_1} * \frac{\partial H_1}{\partial Z_1^1} * 1 = \delta_1^1$$

Persamaan 13.69

$$\frac{\partial E}{\partial B_2} = \frac{\partial E}{\partial H_2} * \frac{\partial H_2}{\partial Z_1^2} * 1 = \delta_1^2$$

Karena Anda menggunakan bias B1 dan B2 per lapisan dan bukan per neuron, Anda dapat menghitung rata-rata untuk lapisan (Persamaan 13-70 hingga 13-76)

Persamaan 13.70

$$\delta^1 = \delta_1^1 + \delta_2^1 + \delta_3^1$$

Persamaan 13.71

$$\frac{\partial E}{\partial B_1} = \delta^1$$

Persamaan 13.72

$$\frac{\partial E}{\partial B_2} = \delta^2$$

Persamaan 13.73

$$\delta^1 = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^2} = -0.18980115540874787 * 0.23785621465075305$$
$$= -0.04514538436186407$$

Persamaan 13.74

$$\delta_1^1 = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_2^1} = 0.04514538436186407$$

Persamaan 13.75

$$\delta_2^1 = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^1} = 0.04514538436186407$$

Persamaan 13.76

$$\delta_3^1 = \frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_3^1} = 0.04514538436186407$$

Karena untuk penyesuaian bias yang Anda hitung per lapisan, Anda dapat mengambil rata-rata dari penyesuaian bias yang dihitung untuk setiap neuron (Persamaan 13-77).

Persamaan 3.77

$$\delta^1 = \frac{\delta_1^1 + \delta_2^1 + \delta_3^1}{3} = -0.04514538436186407$$
$$\delta^1 = -0.04514538436186407$$

Dengan pengenalan variabel 1δ, Anda mendapatkan Persamaan 13-78 dan 13-79.

Persamaan 13.78

$$Disesuaikan\ B_1 = B_1 - \eta * \delta_1 = 0.20 + 0.04514538436186407$$
$$= 0.24514538618641$$

Persamaan 13.79

$$Disesuaikan\ B_2 = B_2 - \eta * \delta_2 = 0.25 + 0.04514538436186407$$
$$= 0.29514538436186405$$

Sekarang setelah Anda menghitung semua nilai bobot baru, Anda kembali ke fase maju dan menghitung kesalahan baru.

Kembali ke Pass

Hitung ulang output jaringan untuk lapisan tersembunyi dan output menggunakan bobot/bias baru yang disesuaikan.

Lapisan Tersembunyi (Layer Output)

Untuk neuron H1, berikut langkah-langkahnya:

1. Hitung total input bersih untuk neuron H_1 (Persamaan 13-80).

Persamaan 13.80

$$Z_1^1 = W_{11}^1 * I_1 + W_{12}^1 * I_2 + B_1 * 1.00 = 0.05011172329667623 * 0.01$$
$$+0.06022344659335245 * 0.02 + 0.2451453843618641 * 1.00$$
$$= 0.2468509705266979$$

2. Gunakan fungsi logistik untuk mendapatkan output dari H_1 (Persamaan 13-81).

Persamaan 13.81

$$H_1 = \delta(Z_1^1) = \frac{1}{(1 + \exp(Z_1^1))} = \frac{1}{(1 + \exp(-0.2468509705266979))}$$
$$= 0.561401266257945$$

Untuk neuron H_2, lihat Persamaan 13-82 dan Persamaan 13-83.

Persamaan 13.82

$$Z_1^1 = W_{21}^1 * I_1 + W_{22}^1 * I_2 + B_1 * 1.00 = 0.07002698635479371 * 0.01 + 0.08005397270958742 * 0.02 + 0.2451453843618641 * 1.00 = 0.24744673367960376$$

Persamaan 13.83

$$H_2 = \frac{1}{(1 + \exp(-0.24744673367960376))} = 0.5615479555799516$$

Untuk neuron H_3, lihat Persamaan 13-84.

Persamaan 13.84

$$Z_1^1 = W_{31}^1 * I_1 + W_{32}^1 * I_2 + B_1 * 1.00 = 0.09011170978258068 * 0.01 + 0.1002234196985099312 * 0.02 + 0.2451453843618641 * 1.00 = 0.24805096985099312$$

$$H_2 = \frac{1}{(1 + \exp(-0.24805096985099312))} = 0.5616967201480348$$

Layer Output

Untuk neuron O_1, berikut langkah-langkahnya:

1. Hitung total input bersih untuk neuron O_1 (Persamaan 13-85).

Persamaan 13.85

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_3 + B_2 * 1.00 = 0.13484146172251732 * 0.5502547397403884 + 0.1448414617225173 * 0.5504032199355139 + 0.15484414617225173 * 0.5505516911502556 + 0.29514538436186405 * 1.00 = 0.5343119733119508$$

2. Gunakan fungsi logistik untuk mendapatkan output dari O1 (Persamaan 13-86).

Persamaan 13.86

$$O_1 = \sigma(Z_1^2) = \frac{1}{(1 + \exp(Z_1^2))} = \frac{1}{(1 + \exp(-0.5343119733119508))} = 0.6304882485312977$$

Output yang dihitung dari neuron O_1 adalah 0.6304882485312977, sedangkan target output untuk O_1 adalah 0.80; oleh karena itu, lihat Persamaan 13-87 untuk galat kuadrat untuk output neuron O_1.

Persamaan `3.87

$$E = 0.5 * (T_1 - O_1)^2 (0.80 - 0.6304882485312977)^2 = 0.0143671169\ 42993556$$

Pada iterasi pertama, kesalahannya adalah 0.01801223929724783 (lihat Persamaan 13.7). Sekarang, pada iterasi kedua, kesalahan telah berkurang menjadi 0.014367116942993556. Anda dapat melanjutkan iterasi ini hingga jaringan menghitung kesalahan yang lebih kecil dari batas kesalahan yang telah ditetapkan. Mari kita lihat rumus untuk menghitung turunan parsial dari fungsi kesalahan E terhadap W_{11}^2 dan W_{12}^2 untuk simpul H1 (lihat Persamaan 13-88, 13-89, dan 13-90).

Persamaan 13.88

$$\frac{\partial E}{\partial W_{11}^2} = \frac{\partial E}{\partial O_1} * \frac{\partial H_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial W_{11}^2}$$

Persamaan 13.89

$$\frac{\partial E}{\partial W_{12}^2} = \frac{\partial E}{\partial O_1} * \frac{\partial H_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial W_{12}^2}$$

Persamaan 13.90

$$\frac{\partial E}{\partial W_{13}^2} = \frac{\partial E}{\partial O_1} * \frac{\partial H_1}{\partial Z_1^2} * \frac{\partial Z_1^2}{\partial W_{13}^2}$$

Anda dapat melihat bahwa ketiga rumus memiliki bagian yang sama: $\frac{\partial E}{\partial O_1} * \frac{\partial O_1}{\partial Z_1^2}$. Bagian ini disebut Node Delta . Dengan menggunakan , Anda dapat menulis ulang Persamaan 13-88, 13-89, dan 13-90 sebagai Persamaan 13-91, 13-92, dan 13-93.

Persamaan 13.91

$$\frac{\partial E}{\partial W_{11}^2} = \delta_1^2 * \frac{\partial Z_1^2}{\partial W_{11}^2}$$

Persamaan 13.92

$$\frac{\partial E}{\partial W_{12}^2} = \delta_1^2 * \frac{\partial Z_1^2}{\partial W_{12}^2}$$

Persamaan 13.93

$$\frac{\partial E}{\partial W_{13}^2} = \delta_1^2 * \frac{\partial Z_1^2}{\partial W_{13}^2}$$

Sejalan dengan itu, Anda dapat menulis ulang rumus untuk lapisan tersembunyi (lihat Persamaan 13-94 hingga 13-99).

Persamaan 13.94

$$\frac{\partial E}{\partial W_{11}^1} = \delta_1^1 * \frac{\partial Z_1^1}{\partial W_{11}^1}$$

Persamaan 13.95

$$\frac{\partial E}{\partial W_{12}^1} = \delta_1^1 * \frac{\partial Z_1^1}{\partial W_{12}^1}$$

Persamaan 13.96

$$\frac{\partial E}{\partial W_{21}^1} = \delta_2^1 * \frac{\partial Z_2^1}{\partial W_{21}^1}$$

Persamaan 13.97

$$\frac{\partial E}{\partial W_{22}^1} = \delta_2^1 * \frac{\partial Z_2^1}{\partial W_{22}^1}$$

Persamaan 13.98

$$\frac{\partial E}{\partial W_{31}^1} = \delta_3^1 * \frac{\partial Z_3^1}{\partial W_{31}^1}$$

Persamaan 13.99

$$\frac{\partial E}{\partial W_{32}^1} = \delta_3^1 * \frac{\partial Z_3^1}{\partial W_{32}^1}$$

Secara umum, menghitung turunan parsial dari fungsi kesalahan E terhadap bobotnya dapat dilakukan dengan mengalikan delta simpul dengan turunan parsial dari fungsi kesalahan terhadap bobot yang sesuai. Itu menyelamatkan Anda dari menghitung beberapa data yang berlebihan. Ini berarti Anda dapat menghitung nilai untuk setiap node jaringan dan kemudian menggunakan Persamaan 13.94 hingga 13.99.

13.5 BENTUK MATRIKS PERHITUNGAN JARINGAN

Katakanlah ada dua record (dua titik) yang akan diproses oleh jaringan. Untuk jaringan yang sama, Anda dapat melakukan perhitungan menggunakan matriks. Misalnya, dengan memperkenalkan vektor Z, matriks W, dan vektor B, Anda bisa mendapatkan hasil perhitungan yang sama seperti yang Anda dapatkan saat menggunakan skalar. Lihat Gambar 13-5.

$$\begin{vmatrix} Z^1_1 \\ Z^1_2 \\ Z^1_3 \end{vmatrix} = \begin{Vmatrix} W^1_{11} & W^1_{12} \\ W^1_{21} & W^1_{22} \\ W^1_{21} & W^1_{22} \end{Vmatrix} \begin{vmatrix} I_1 \\ I_2 \end{vmatrix} + \begin{vmatrix} B1 \\ B1 \\ B1 \end{vmatrix} = \begin{Vmatrix} W^1 11*I_1 + W^1 12*I_2 + B_1 \\ W^1 11*I_1 + W^1 12*I_2 + B_1 \\ W^1 11*I_1 + W^1 12*I_2 + B_1 \end{Vmatrix}$$

Gambar 13.5 Bentuk matriks perhitungan jaringan

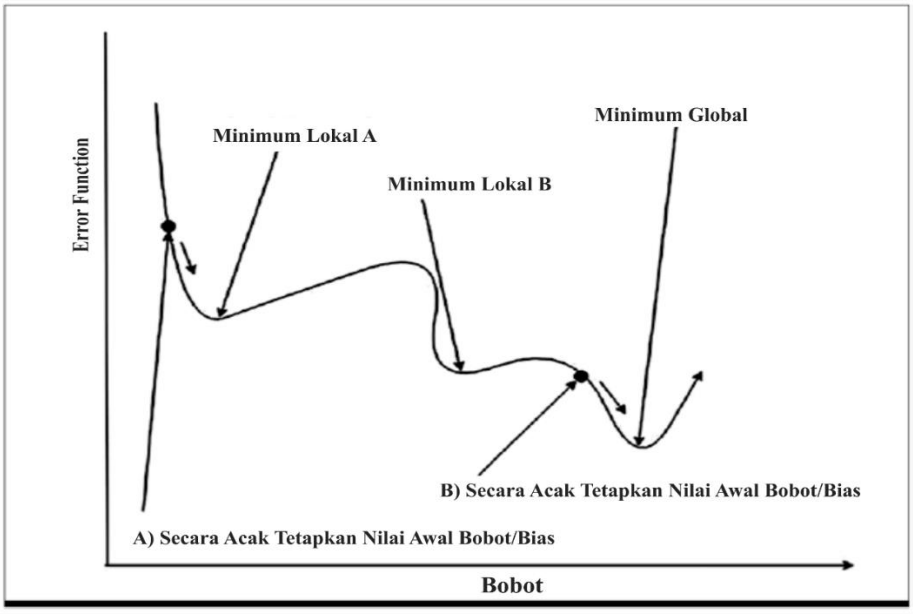
Menggunakan matriks versus perhitungan skalar adalah masalah preferensi. Menggunakan library matriks yang baik memberikan perhitungan yang cepat. Misalnya, library cuBLAS dapat memanfaatkan GPU dan FPGA. Sifat perampingan menggunakan matriks menyebabkan permintaan memori yang tinggi karena matriks harus disimpan dalam memori.

Menggali lebih dalam

Saat menggunakan jaringan saraf, Anda menetapkan batas kesalahan (secara khusus menunjukkan seberapa dekat hasil jaringan yang dilatih harus cocok dengan data target). Proses pelatihan bekerja secara iteratif dengan bergerak secara bertahap ke arah fungsi kesalahan minimum, sehingga mengurangi kesalahan. Proses berulang berhenti ketika perbedaan antara hasil jaringan yang dihitung dan hasil target kurang dari batas kesalahan yang telah ditetapkan.

Mungkinkah jaringan gagal mencapai batas kesalahan yang ditetapkan? Sayangnya ya. Mari kita bahas ini lebih detail. Tentu saja, kesalahan aproksimasi tergantung pada arsitektur jaringan yang dipilih (jumlah lapisan tersembunyi dan jumlah neuron dalam setiap lapisan tersembunyi). Namun, mari kita asumsikan bahwa arsitektur jaringan diatur dengan benar.

Kesalahan perkiraan juga tergantung pada topologi fungsi. Sekali lagi, mari kita asumsikan bahwa fungsinya monoton dan kontinu (kita akan membahas aproksimasi fungsi nonkontinyu nanti dalam buku ini). Namun, jaringan dapat gagal mencapai batas jaringan. Mengapa? Saya telah menyebutkan bahwa backpropagation adalah proses berulang yang mencari fungsi kesalahan minimum. Fungsi kesalahan biasanya merupakan fungsi dari banyak variabel, tetapi untuk kesederhanaan Gambar 13-6 menunjukkannya sebagai bagan ruang 2D.



Gambar 13.6 Fungsi kesalahan minimum lokal dan global

Tujuan dari proses pelatihan adalah untuk menemukan minimum dari fungsi error. Fungsi kesalahan tergantung pada parameter bobot/bias yang dikalibrasi selama proses pelatihan berulang. Nilai awal dari bobot/bias biasanya diatur secara acak, dan proses pelatihan menghitung kesalahan jaringan untuk pengaturan awal ini (titik). Mulai dari titik ini, proses pelatihan bergerak ke fungsi minimum.

Seperti yang ditunjukkan pada Gambar 13.6, fungsi error biasanya memiliki beberapa minimum. Yang terendah disebut minimum global, sedangkan sisanya disebut minimum lokal. Tergantung pada titik awal proses pelatihan, ia dapat menemukan beberapa minimum lokal yang dekat dengan titik awal. Setiap minimum lokal berfungsi sebagai jebakan untuk proses pelatihan karena setelah proses pelatihan mencapai minimum lokal, setiap langkah lebih lanjut akan menunjukkan nilai gradien yang berubah, dan proses iterasi akan berhenti, ditumpuk pada minimum lokal.

Pertimbangkan titik awal A dan B pada Gambar 13.6. Dalam kasus titik awal A, proses pelatihan akan menemukan minimum lokal A, yang menghasilkan kesalahan yang jauh lebih besar daripada kasus B. Itulah sebabnya menjalankan proses pelatihan yang sama beberapa kali selalu menghasilkan hasil kesalahan yang berbeda untuk setiap proses (karena untuk setiap proses , proses pelatihan dimulai pada titik awal acak).

Tips

bagaimana Anda mencapai hasil perkiraan terbaik? saat memprogram pemrosesan jaringan saraf, selalu atur logika untuk memulai proses pelatihan dalam satu lingkaran. Setelah setiap panggilan metode pelatihan, logika di dalam metode pelatihan harus memeriksa apakah kesalahan kurang dari batas kesalahan, dan jika tidak, itu harus keluar dari metode pelatihan dengan kode kesalahan bukan nol. kontrol akan dikembalikan ke kode yang memanggil metode pelatihan dalam satu lingkaran. kode memanggil metode pelatihan lagi jika kode yang dikembalikan bukan nol. logika melanjutkan loop sampai kesalahan yang dihitung menjadi kurang dari batas kesalahan. Ini akan keluar pada titik ini dengan kode pengembalian nol sehingga metode pelatihan tidak akan lagi dipanggil lagi. Jika ini tidak dilakukan, logika pelatihan hanya akan mengulang epoch, tidak dapat menghapus batas kesalahan. beberapa contoh kode pemrograman tersebut ditunjukkan pada bab-bab selanjutnya.

Mengapa terkadang sulit untuk melatih jaringan? Jaringan dianggap sebagai alat pendekatan fungsi universal. Namun, ada pengecualian untuk pernyataan ini. Jaringan hanya dapat memperkirakan fungsi kontinu dengan baik. Jika suatu fungsi tidak kontinu (membuat lompatan tajam ke atas dan ke bawah secara tiba-tiba), maka hasil aproksimasi untuk fungsi tersebut menunjukkan hasil berkualitas rendah (kesalahan besar) sehingga aproksimasi tersebut tidak berguna. Saya akan membahas masalah ini nanti dalam buku ini dan menunjukkan metode saya yang memungkinkan aproksimasi fungsi nonkontinyu dengan presisi tinggi.

Perhitungan yang ditunjukkan di sini dilakukan untuk satu titik fungsi. Ketika Anda perlu memperkirakan fungsi dari dua atau lebih variabel di banyak titik, volume perhitungan meningkat secara eksponensial. Proses intensif sumber daya seperti itu menempatkan permintaan tinggi pada sumber daya komputer (memori dan CPU). Itu sebabnya, seperti yang disebutkan dalam pendahuluan, upaya sebelumnya untuk menggunakan kecerdasan buatan tidak dapat memproses tugas yang serius. Baru kemudian, karena kekuatan komputasi yang meningkat secara dramatis, kecerdasan buatan mencapai kesuksesan besar.

13.6 *MINI-BATCHES* DAN GRADIEN STOKASTIK

Ketika kumpulan data input sangat besar (jutaan catatan), volume perhitungan menjadi sangat tinggi. Pemrosesan jaringan seperti itu membutuhkan waktu lama, dan pembelajaran jaringan menjadi sangat lambat, karena gradien harus dihitung untuk setiap catatan input.

Untuk mempercepat proses ini, Anda dapat memecah kumpulan data input besar menjadi beberapa bagian yang disebut mini-batch dan memproses setiap mini-batch secara independen. Memproses semua record dalam satu file mini-batch merupakan sebuah epoch, titik di mana penyesuaian bobot/bias dilakukan.

Karena ukuran file mini-batch yang jauh lebih kecil, pemrosesan semua mini-batch akan lebih cepat daripada pemrosesan seluruh kumpulan data sebagai satu file. Akhirnya, alih-alih menghitung gradien untuk setiap catatan dari seluruh kumpulan data, Anda menghitung di sini gradien stokastik, yang merupakan rata-rata gradien yang dihitung untuk setiap mini-batch.

Jika penyesuaian bobot yang diproses untuk neuron dalam file mini-batch m adalah Wnm , maka penyesuaian bobot untuk neuron tersebut untuk seluruh kumpulan data kira-kira sama dengan rata-rata penyesuaian yang dihitung secara independen untuk semua mini-batch.

Disesuaikan $W_s^k \approx W_s^k - \frac{n}{m}\sum_j^m \frac{\partial E}{\partial W_s^j}$ *, di mana m adalah jumlah mini-batch.*

Pemrosesan jaringan saraf untuk set data input besar sebagian besar dilakukan menggunakan mini-b atches.

13.7 RINGKASAN

Bab ini menunjukkan semua perhitungan jaringan saraf internal. Ini menjelaskan mengapa (bahkan untuk satu titik) volume perhitungan cukup tinggi. Bab ini memperkenalkan variabel , yang memungkinkan Anda untuk mengurangi volume kalkulasi. Di bagian “Menggali Lebih Dalam”, dijelaskan cara memanggil metode pelatihan untuk mencapai salah satu hasil aproksimasi terbaik. Pendekatan mini-batch juga dijelaskan di sini. Bab berikutnya menjelaskan cara mengkonfigurasi lingkungan Windows untuk menggunakan Java dan kerangka kerja pemrosesan jaringan Java.

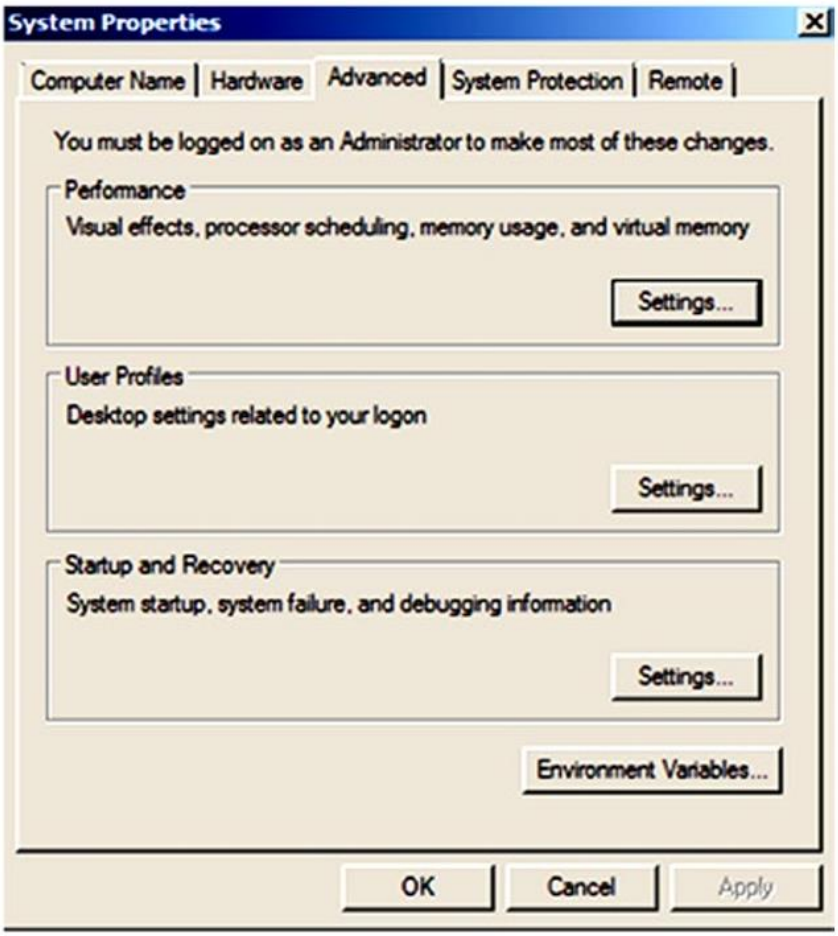
BAB 14
MENGKONFIGURASI LINGKUNGAN PENGEMBANGAN ANDA

Buku ini membahas tentang pemrosesan jaringan syaraf tiruan menggunakan Java. Sebelum Anda dapat mulai mengembangkan program jaringan saraf apa pun, Anda perlu mempelajari beberapa tool Java. Jika Anda seorang pengembang Java dan terbiasa dengan tool yang dibahas dalam bab ini, Anda dapat melewati bab ini. Pastikan saja bahwa semua tool yang diperlukan telah diinstal pada mesin Windows Anda.

14.1 MEMASANG LINGKUNGAN JAVA 11 DI MESIN WINDOWS ANDA

Semua contoh dalam buku ini bekerja dengan baik dengan Java versi 8–11. Berikut langkah-langkahnya:

1. Buka https://docs.Oracle.com/en/java/javase/11/install/installation-jdk-microsoft-windows-platforms.html#GUIDA740535E-9F97-448C-A141-B95BF1688E6F.
2. Unduh Java SE Development Kit terbaru untuk Windows. Klik dua kali file eksekusi yang diunduh.
3. Ikuti petunjuk penginstalan, dan lingkungan Java akan diinstal pada mesin Anda.
4. Pada desktop Anda, pilih Start ➤ Control Panel ➤ System and security ➤ System ➤ Advanced system setting. Layar yang ditunjukkan pada Gambar 14.1 akan muncul.

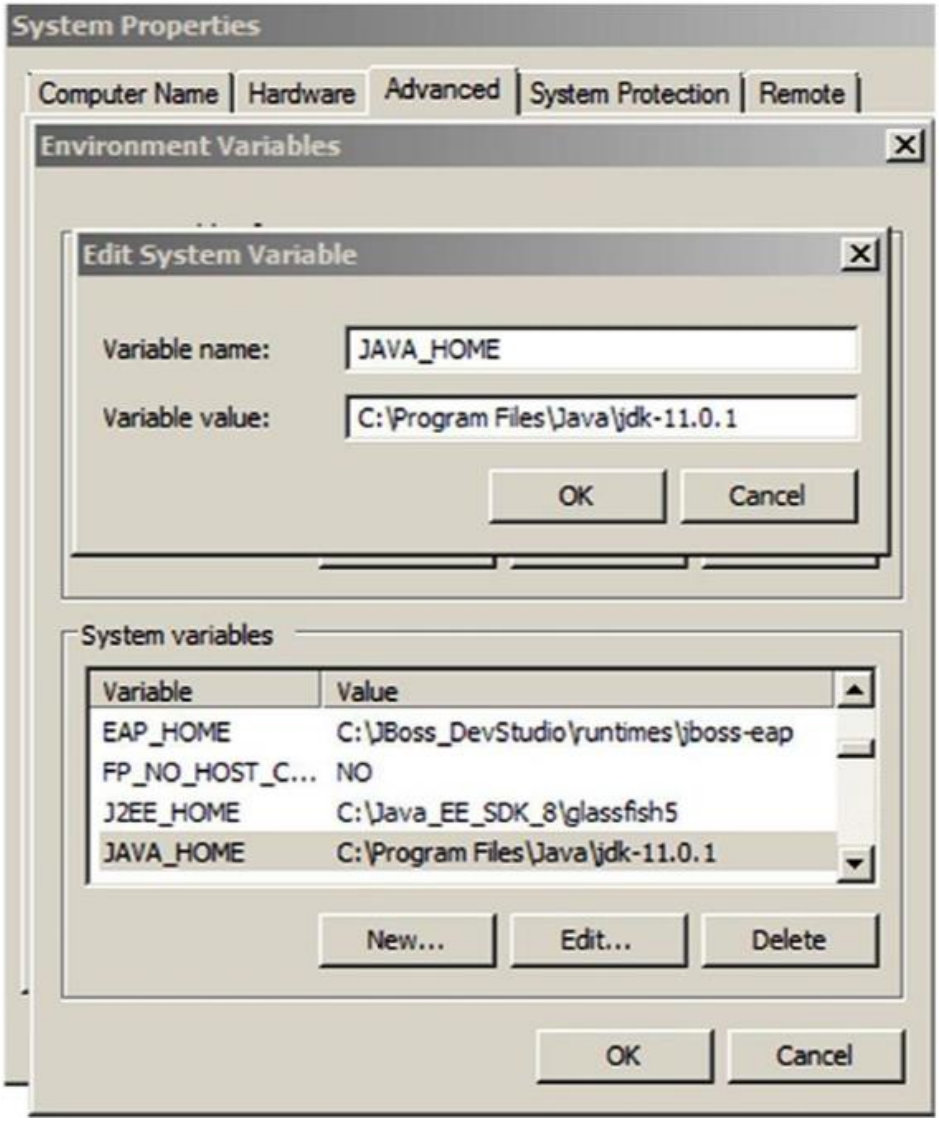


Gambar 14.1 System Properties dialog

5. Klik tombol Variabel Lingkungan pada tab Lanjutan.
6. Klik Baru untuk melihat dialog yang memungkinkan Anda memasukkan variabel lingkungan baru (Gambar 14.2).

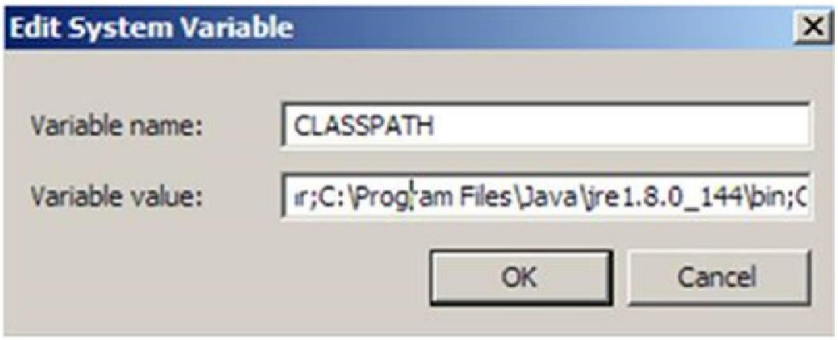
Gambar 14.2 Dialog Variabel Sistem Baru

7. Masukkan JAVA_HOME di bidang "Nama variabel".
8. Masukkan jalur ke lingkungan Java yang diinstal di bidang “Nilai variabel” (Gambar 14.3).
9.



Gambar 14.3 Dialog Variabel Sistem Baru, diisi

10. Klik Oke. Selanjutnya, pilih variabel lingkungan CLASSPATH dan klik Perbarui.
11. Tambahkan path ke direktori bin Java JDK, dan tambahkan file Java JAR ke bidang "Variable value" CLASSPATH (Gambar 14.4), seperti yang ditunjukkan di sini:
    C:\Program Files\Java\jre1.8.0_144\bin
    C:\Program Files\Java\jdk1.8.0_144\jre\bin\java.exe

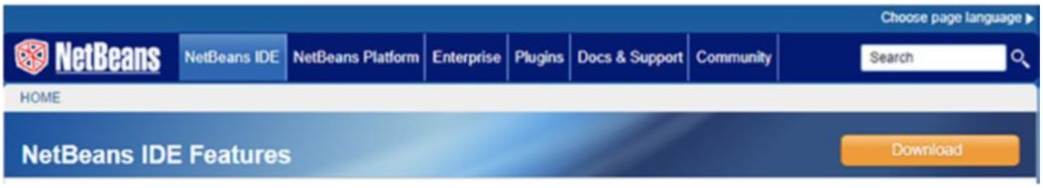


Gambar 14.4 Variabel sistem CLASSPATH yang diperbarui

12. Klik OK tiga kali.
13. Nyalakan ulang sistem. Lingkungan Java Anda telah diatur.

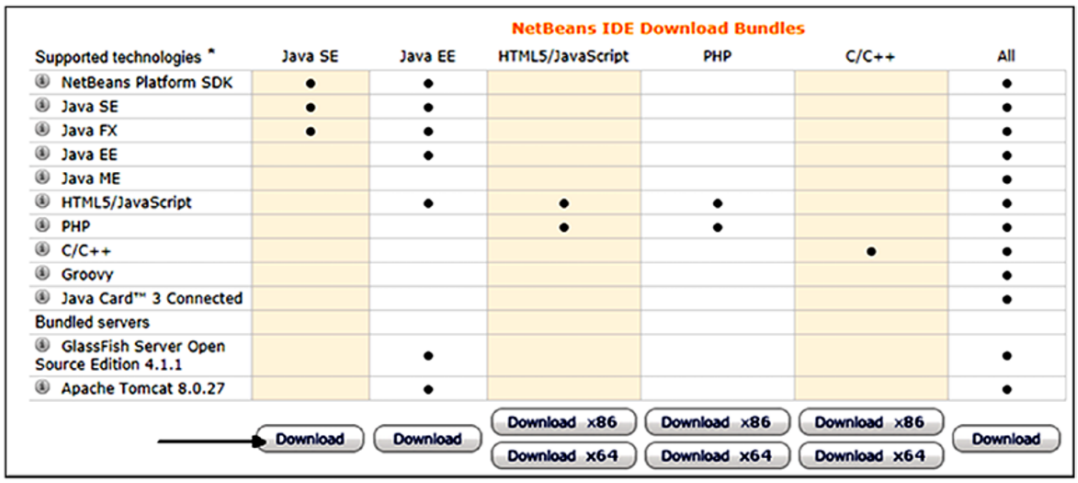
14.2 MENGINSTALL NETBEANS IDE

NetBeans adalah tool pengembangan Java standar yang saat ini dikelola oleh Oracle. Pada saat penulisan buku ini, versi NetBeans saat ini adalah 8.2, dan merupakan IDE resmi untuk platform Java 8. Untuk menginstal NetBeans, buka https://netbeans.org/features/index.html dan klik Download (Gambar 14.5).



Gambar 14.5 halaman utama NetBeans

Pada layar Unduh, klik tombol Unduh untuk Java SE (Gambar 14.6).



Gambar 14.6 halaman utama NetBeans

Klik dua kali file executable yang diunduh dan ikuti petunjuk penginstalan. NetBeans 8.2 akan diinstal pada mesin Windows Anda, dan ikonnya akan ditempatkan di desktop Anda.

14.3 MENGINSTALL KERANGKA KERJA JAVA ENCOG

Seperti yang dapat Anda lihat dari beberapa contoh pemrosesan jaringan saraf secara manual di Bab 3, bahkan pendekatan sederhana dari suatu fungsi pada satu titik melibatkan sejumlah besar perhitungan. Untuk pekerjaan serius apa pun, ada dua pilihan: otomatisasi proses ini sendiri atau gunakan salah satu kerangka kerja yang tersedia.
Beberapa kerangka kerja saat ini tersedia. Berikut adalah daftar kerangka kerja yang paling umum digunakan dan bahasa penulisannya:

- TensorFlow (Python, C++, and R)
- Caffe (C, C++, Python, and MATLAB)
- Torch (C++, Python)
- Keras (Python)
- Deep*learning*4j (Java)
- Encog (Java)
- Neurop (Java)

Kerangka kerja yang diimplementasikan di Java jauh lebih efisien dibandingkan dengan yang diimplementasikan dengan Python. Di sini kami tertarik pada kerangka kerja Java (karena alasan yang jelas untuk mengembangkan aplikasi pada satu mesin dan dapat menjalankannya di mana saja). Kami juga tertarik dengan kerangka kerja Java yang cepat dan yang nyaman digunakan. Setelah memeriksa beberapa kerangka kerja Java, saya memilih Encog sebagai kerangka kerja terbaik untuk pemrosesan jaringan saraf. Itulah kerangka kerja yang digunakan di seluruh buku ini. Kerangka pembelajaran mesin Encog dikembangkan oleh Heaton Research, dan gratis untuk digunakan. Semua dokumentasi Encog juga dapat ditemukan di situs web.

Untuk menginstal Encog, buka halaman web berikut: https://www.heatonresearch.com/encog. Gulir ke bawah ke bagian yang disebut Encog Java Links dan klik tautan Encog Java Download/Release. Pada layar berikutnya, pilih dua file ini untuk Encog rilis 4.3:
encog-core-3.4.jar
encog-java-examples.zip
Buka zip file kedua. Simpan file-file ini di direktori yang akan Anda ingat, dan tambahkan file berikut ke variabel lingkungan CLASSPATH (seperti yang Anda lakukan untuk instalasi Java): c:\encog-core-3.4\lib\encog-core-3.4.jar

14.4 MEMASANG PAKET XCHART

Selama persiapan data dan pengembangan/pengujian jaringan saraf, sangat berguna untuk dapat memetakan banyak hasil. Anda akan menggunakan library charting Java XChart dalam buku ini. Untuk mengunduh XChart, kunjungi situs web berikut: https://knowm.org/open-source/xchart/. Klik tombol Unduh. Layar yang ditunjukkan pada Gambar 14.7 akan muncul.

Gambar 14.7 Halaman beranda XChart

Buka zip file zip yang diunduh dan klik dua kali file instalasi yang dapat dieksekusi. Ikuti petunjuk penginstalan, dan paket XChart akan diinstal pada mesin Anda. Tambahkan dua file berikut ke variabel lingkungan CLASSPATH (seperti yang Anda lakukan untuk Java 8 sebelumnya):
c:\Download\XChart\xchart-3.5.0\xchart-3.5.0.jar
c:\Download\XChart\xchart-3.5.0\xchart-demo-3.5.0.jar
Terakhir, reboot sistem. Anda siap untuk pengembangan jaringan saraf!

14.5 RINGKASAN

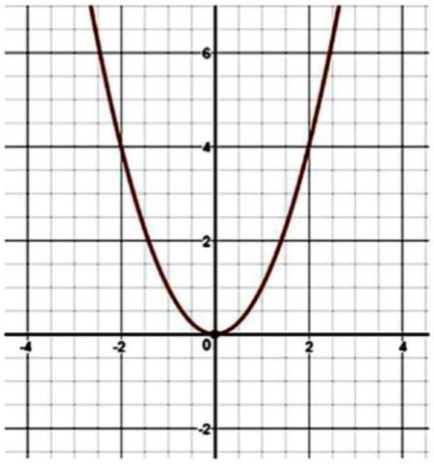
Bab ini memperkenalkan Anda ke lingkungan Java dan menjelaskan cara mengunduh dan menginstal seperangkat alat yang diperlukan untuk membangun, men-debug, menguji, dan menjalankan aplikasi jaringan saraf. Semua contoh pengembangan di sisa buku ini akan dibuat menggunakan lingkungan ini. Bab berikutnya menunjukkan bagaimana mengembangkan program jaringan saraf menggunakan kerangka Java Encog.

BAB 15
PENGEMBANGAN JARINGAN *NEURAL* DENGAN JAVA ENCOG *FRAMEWORK*

Untuk memfasilitasi pembelajaran Anda tentang pengembangan program jaringan menggunakan Java, Anda akan mengembangkan program sederhana pertama Anda menggunakan fungsi dari Contoh 1 di Bab 2.

Contoh 2: Pendekatan Fungsi Menggunakan Lingkungan Java

Gambar 15.1 menunjukkan fungsi yang diberikan kepada Anda dengan nilainya di sembilan titik.



Gambar 15.1 Fungsi yang akan didekati

Meskipun Encog dapat memproses sekumpulan format file yang termasuk dalam format BasicMLDataset, format file termudah yang dapat diproses Encog adalah format CSV. Format CSV adalah format file Excel yang disederhanakan yang menyertakan nilai yang dipisahkan koma di setiap catatan, dan file memiliki ekstensi .csv. Encog mengharapkan catatan pertama dalam file yang diproses menjadi catatan label; karenanya, Tabel 15.1 menunjukkan kumpulan data input (pelatihan) dengan nilai fungsi yang diberikan untuk contoh ini.

Tabel 15.1 Kumpulan Data Pelatihan

xPoint	Nilai Fungsi
0.15	0.0225
0.25	0.0625
0.5	0.25
0.75	0.5625
1	1
1.25	1.5625
1.5	2.25
1.75	3.0625
2	4

Berikutnya adalah kumpulan data untuk menguji jaringan terlatih, yang ditunjukkan pada Tabel 15.2. xPoints dari kumpulan data ini berbeda dari xPoints dalam kumpulan data

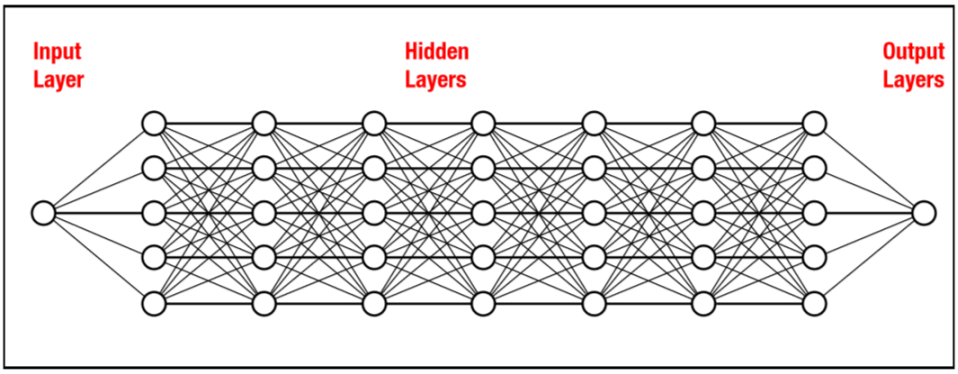
pelatihan karena file tersebut digunakan untuk menguji jaringan di xPoints yang tidak digunakan untuk pelatihan jaringan.

Tabel 15.2 Menguji Kumpulan Data

xPoint	Nilai Fungsi
0.2	0.04
0.3	0.09
0.4	0.16
0.7	0.49
0.96	0.9025
1.3	1.69
1.6	2.56
1.8	3.24
1.95	3.8025

15.1 ARSITEKTUR JARINGAN

Gambar 15.2 menunjukkan arsitektur jaringan sebagai contoh. Seperti yang telah disebutkan, arsitektur jaringan untuk setiap proyek ditentukan secara eksperimental, dengan mencoba berbagai konfigurasi dan memilih salah satu yang menghasilkan hasil terbaik. Jaringan diatur untuk memiliki lapisan input dengan satu neuron, tujuh lapisan tersembunyi (masing-masing memiliki lima neuron), dan lapisan output dengan satu neuron.



Gambar 15.2 Arsitektur jaringan

15.2 MENORMALKAN *INPUT DATA SET*

Baik set data pelatihan maupun pengujian perlu dinormalisasi pada interval [-1, 1]. Mari kita buat program Java yang menormalkan kumpulan data tersebut. Untuk menormalkan file, perlu diketahui nilai maks dan min untuk bidang yang dinormalisasi. Kolom pertama dari kumpulan data pelatihan memiliki nilai minimum 0,15 dan nilai maksimum 2,00. Kolom kedua dari kumpulan data pelatihan memiliki nilai minimum 0,0225 dan nilai maksimum 4,00. Kolom pertama dari kumpulan data pengujian memiliki nilai min 0.20 dan nilai max 1.95. Kolom kedua dari kumpulan data pengujian memiliki nilai minimum 0,04 dan nilai maksimum 3,8025. Oleh karena itu, untuk kesederhanaan, pilih nilai min dan maks untuk set data pelatihan dan pengujian sebagai berikut: min = 0,00 dan maks = 5,00. Rumus yang digunakan untuk menormalkan nilai pada interval [-1, 1] ditunjukkan di sini:

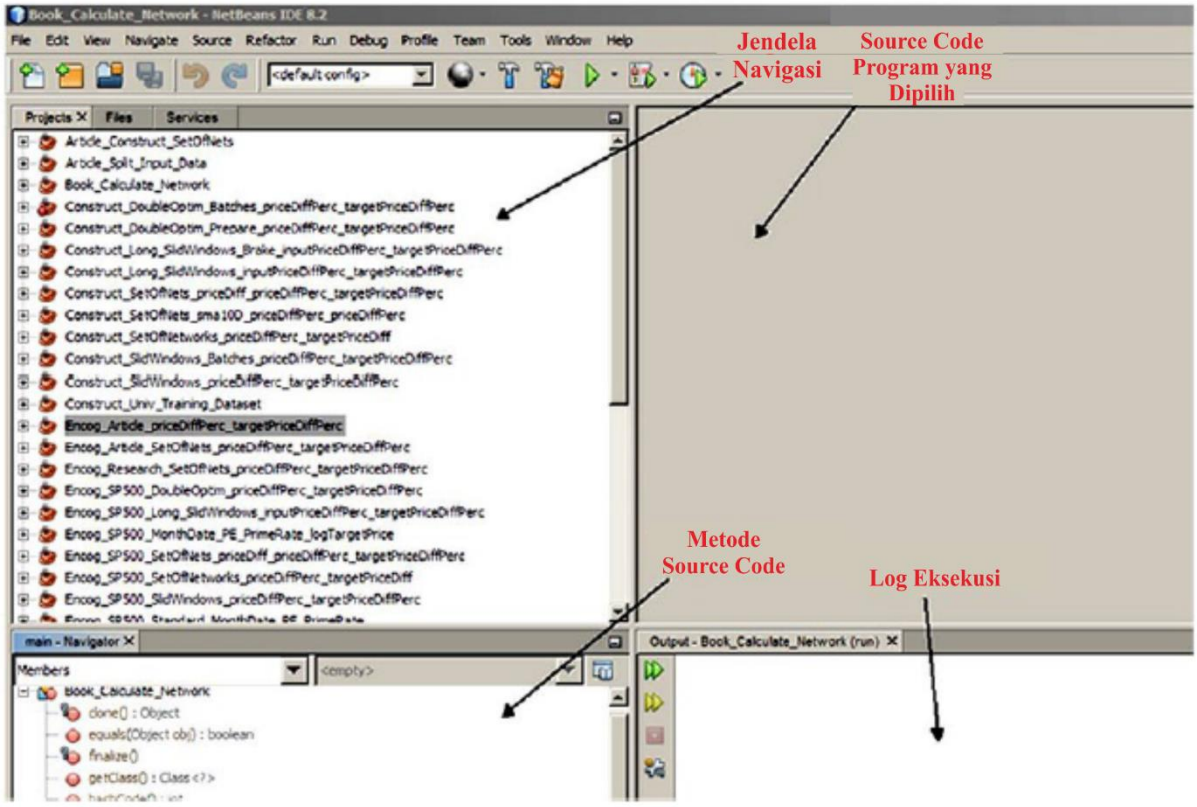
$$F(x) = ((x – DL) * (NH – NL)) / (DH – DL)+NL$$

di mana:

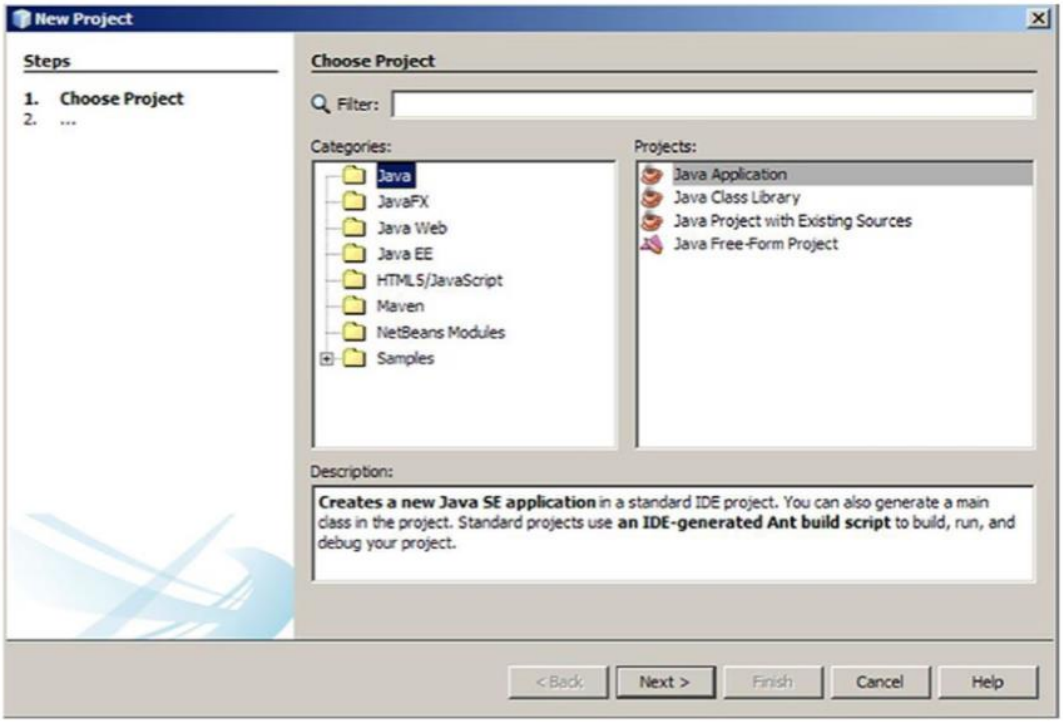
x : Input titik data
DL : Nilai minimum (terendah) x dalam kumpulan data input
DH : Nilai x maksimum (tertinggi) dalam set data input
NL : Bagian kiri dari interval ternormalisasi [-1, 1] = -1
NH : Bagian kanan dari interval ternormalisasi [-1, 1] = 1

Saat menggunakan rumus ini, pastikan bahwa nilai DL dan DH yang Anda gunakan benar-benar nilai fungsi terendah dan tertinggi pada interval yang diberikan; jika tidak, hasil optimasi tidak akan baik. Bab 6 membahas optimasi fungsi di luar rentang pelatihan.

Membangun Program Pemrosesan Jaringan Saraf



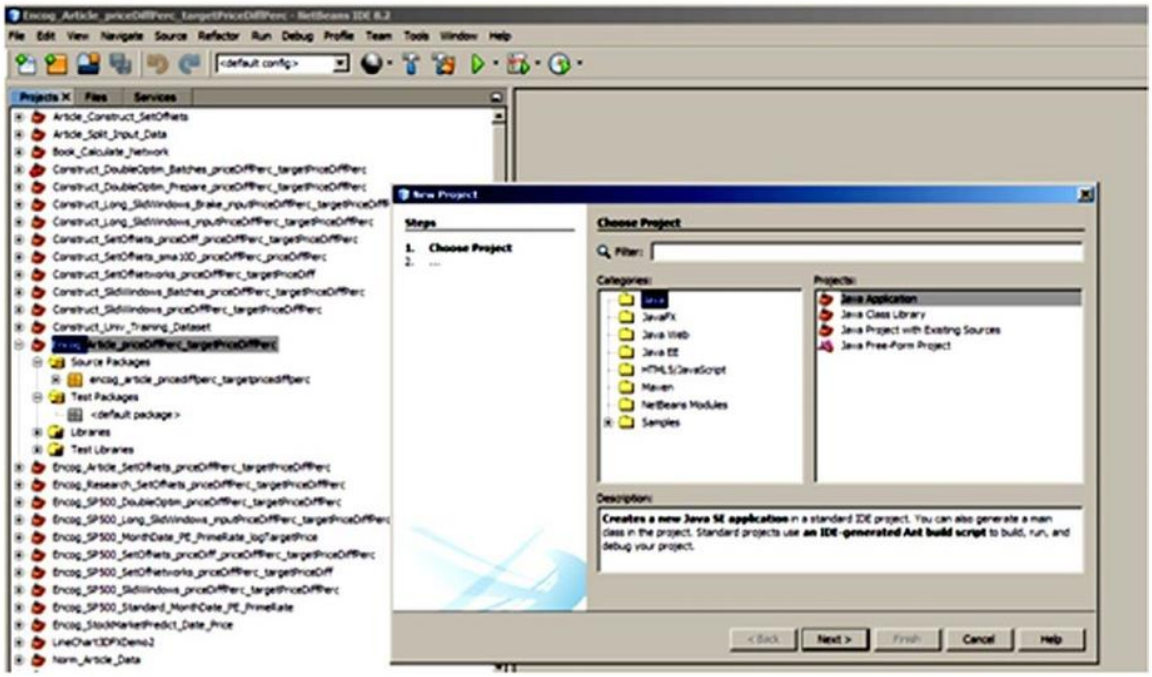
Gambar 15.3 IDE NetBeans



Gambar 15.4 Membuat proyek baru

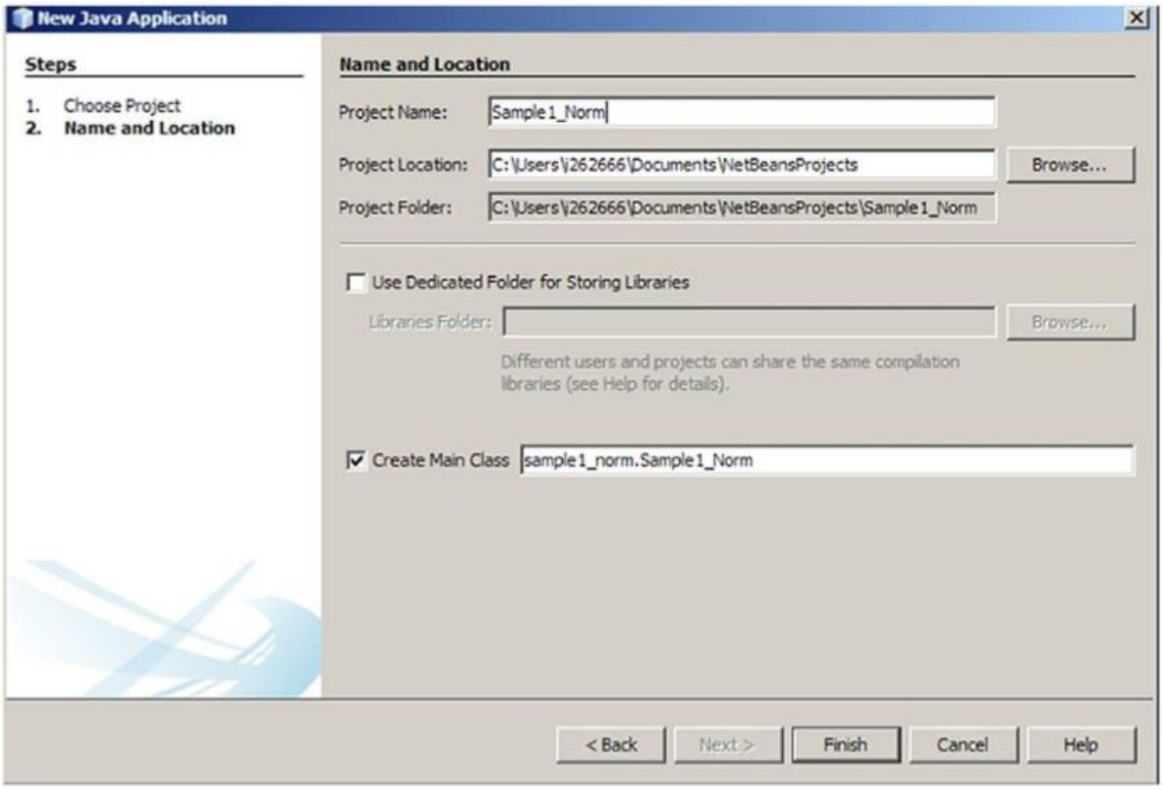
Klik ikon NetBeans di desktop Anda untuk membuka NetBeans IDE. Layar IDE dibagi menjadi beberapa jendela. Jendela Navigasi adalah tempat Anda melihat proyek Anda (Gambar 15.3). Mengklik ikon + di depan proyek menunjukkan komponen proyek: paket sumber, paket uji, dan pustaka.

Untuk membuat proyek baru, pilih File → New Project. Dialog yang ditunjukkan pada Gambar 15.4 muncul. Klik Berikutnya. Dialog yang ditunjukkan pada Gambar 15.5 akan muncul



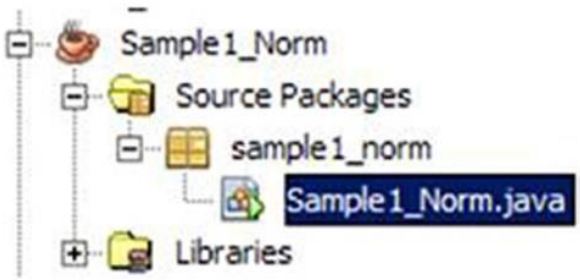
Gambar 15.5 Memberi nama proyek baru

Masukkan nama proyek **Sample1_Norm** dan klik tombol Finish. Gambar 15.6 menunjukkan dialog.



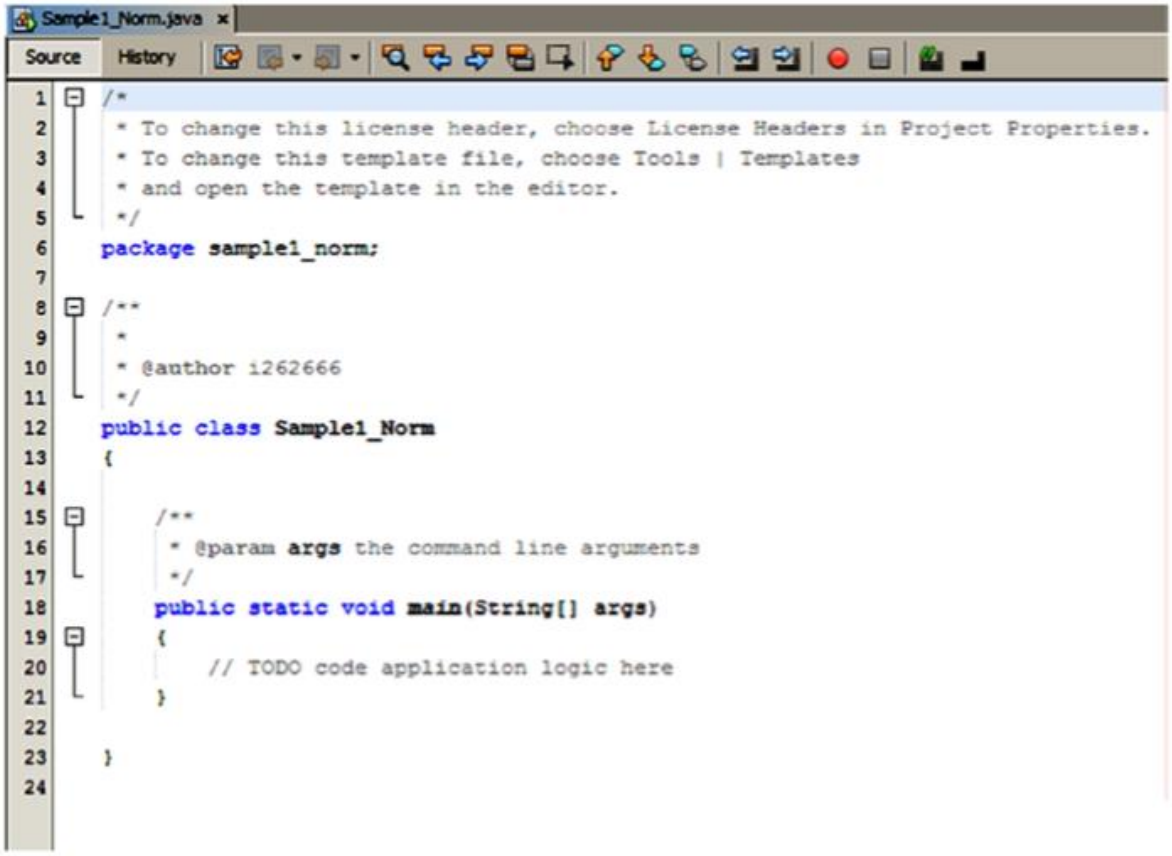
Gambar 15.6 Contoh Sample_Norm

Proyek yang dibuat ditampilkan di jendela Navigasi (Gambar 15.7).



Gambar 15.7 Proyek yang dibuat

Kode sumber menjadi terlihat di jendela kode sumber, seperti yang ditunjukkan pada Gambar 15.8.



Gambar 15.8 Kode sumber untuk proyek baru

Seperti yang Anda lihat, ini hanyalah kerangka program. Selanjutnya, tambahkan logika normalisasi ke program. Daftar 15-1 menunjukkan kode sumber untuk program normalisasi.

Daftar 15-1. Kode Program yang Menormalkan Kumpulan Data Pelatihan dan Pengujian

```
// ============================================================
// Normalisasikan semua kolom dari dataset CSV input dengan
// meletakkan hasilnya dalam file CSV output.
//
// Kolom pertama dari dataset input berisi nilai xPoint dan
// kolom kedua adalah nilai fungsi pada titik X.
// ============================================================
package sample2_norm;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader; import java.io.FileWriter;
import java.io.IOException; import java.nio.file.*;
public class Sample2_Norm
{
        // Interval to normalize
        static double Nh =  1;
        static double Nl = -1;
        // First column
        static double minXPointDl = 0.00;
```

```
static double maxXPointDh = 5.00;
// Second column - target data
static double minTargetValueDl = 0.00;
static double maxTargetValueDh = 5.00;
public static double normalize(double value, double Dh, double Dl)
{
        double normalizedValue = (value - Dl)*(Nh - Nl)/(Dh - Dl) + Nl;
        return normalizedValue;
}
public static void main(String[] args)
{
        // Config data (comment and uncomment the train or test config data)
        // Config for training
        //String inputFileName =
        "C:/My_Neural_Network_Book/Book_Examples/
        Sample2_Train_Real.csv";
        //String outputNormFileName =
        "C:/My_Neural_Network_Book/Book_Examples/Sample2_Train_Norm.csv";
        //Config for testing     String inputFileName =
        "C:/My_Neural_Network_Book/Book_Examples/
        Sample2_Test_Real.csv";
        String outputNormFileName =
        "C:/My_Neural_Network_Book/Book_Examples/
        Sample2_Test_Norm.csv";
        BufferedReader br = null;
        PrintWriter out = null;
        String line = "";
        String cvsSplitBy = ",";
        String strNormInputXPointValue;
        String strNormTargetXPointValue;
        String fullLine;
        double inputXPointValue;
        double targetXPointValue;
        double normInputXPointValue;
        double normTargetXPointValue;
        int i = -1;
        try
        {
        Files.deleteIfExists(Paths.get(outputNormFileName));
        br = new BufferedReader(new FileReader(inputFileName));
        out = new
        PrintWriter(new
        BufferedWriter(new FileWriter(outputNormFileName)));
        while ((line = br.readLine()) != null)
        {
                i++;
                if(i == 0)
                {
```

```
                    // Write the label line
                    out.println(line);
                }
                else
                {
                // Break the line using comma as separator
                String[] workFields = line.split(cvsSplitBy);
                inputXPointValue = Double.parseDouble(workFields[0]);
                targetXPointValue = Double.parseDouble( workFields[1]);
                // Normalize these fields
                normInputXPointValue =
                normalize(inputXPointValue, maxXPointDh, minXPointDl);
                normTargetXPointValue =
                normalize(targetXPointValue,          maxTargetValueDh,
                minTargetValueDl);
                // Convert normalized fields to string, so they can be  inserted
                //into  the  output  CSV  file          strNormInputXPointValue =
                Double.toString(normInputXPointValue);
                strNormTargetXPointValue                                    =
                Double.toString(normTargetXPointValue);
                // Concatenate these fields into a string line with
                //coma separator      fullLine  =      strNormInputXPointValue
                + "," + strNormTargetXPointValue;
                // Put fullLine into the output file      out.println(fullLine);
                }
                // End of IF Else
}   // End of WHILE
}
        // End of TRY
        catch (FileNotFoundException e)
        {
        e.printStackTrace();
        System.exit(1);
        }
        catch (IOException io)
        {
        io.printStackTrace();
        System.exit(2);
}
finally
{
if (br != null)
        {
                try
                {
                        br.close();
                                out.close();
                }
```

```
                    catch (IOException e1)
                    {
                                        e1.printStackTrace();
                                        System.exit(3);
                                }
                        }
                    }
                }
        }
// End of the class
```

Ini adalah program yang sederhana, dan tidak perlu banyak penjelasan. Pada dasarnya, Anda mengatur konfigurasi untuk menormalkan file pelatihan atau pengujian dengan memberi komentar dan menghapus komentar pada kalimat konfigurasi yang sesuai. Anda membaca baris file dalam satu lingkaran. Untuk setiap baris, bagi menjadi dua bidang dan normalkan. Selanjutnya, Anda mengubah kedua bidang kembali ke string, menggabungkannya menjadi satu baris, dan menulis baris ke file output. Tabel 15.3 menunjukkan set data pelatihan yang dinormalisasi.

Tabel 15.3 Kumpulan Data Pelatihan yang Dinormalisasi

xPoint	Nilai Aktual
-0.94	-0.991
-0.9	-0.975
-0.8	-0.9
-0.7	-0.775
-0.6	-0.6
-0.5	-0.375
-0.4	-0.1
-0.3	0.225
-0.2	0.6

Tabel 15.4 menunjukkan kumpulan data pengujian yang dinormalisasi.

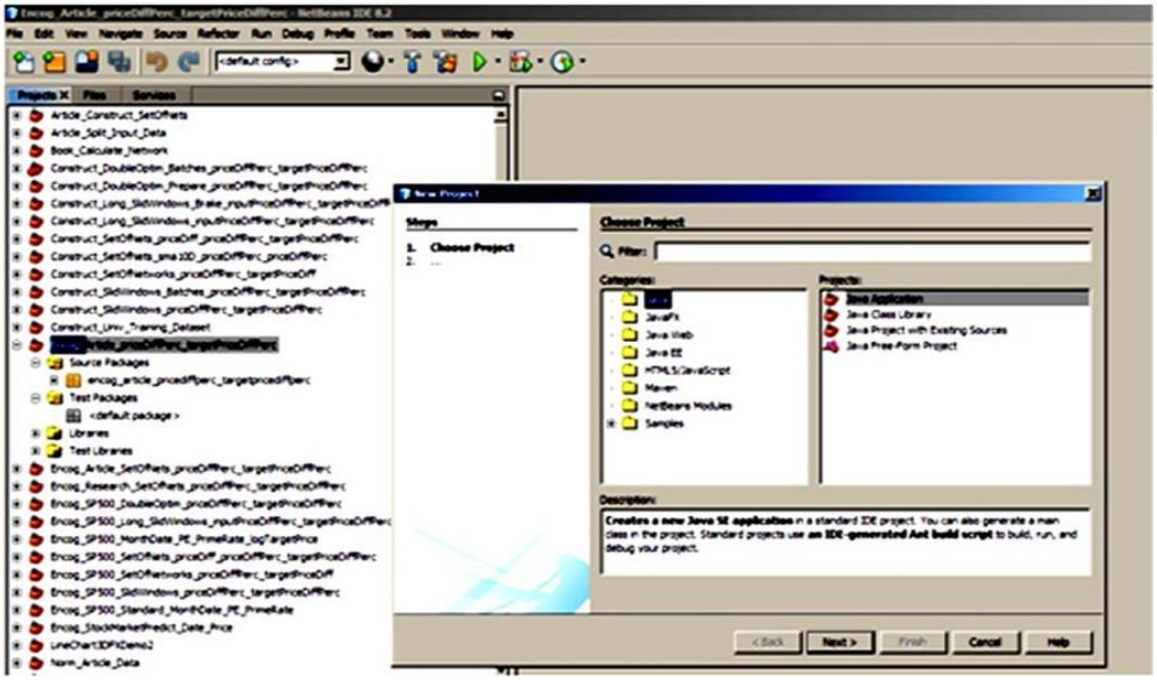
Tabel 15.4 Kumpulan Data Pengujian yang Dinormalisasi

xPoint	Nilai Aktual
-0.92	-0.984
-0.88	-0.964
-0.84	-0.936
-0.72	-0.804
-0.62	-0.639
-0.48	-0.324
-0.36	0.024
-0.28	0.296
-0.22	0.521

Anda akan menggunakan kumpulan data ini sebagai input untuk pelatihan dan pengujian jaringan.

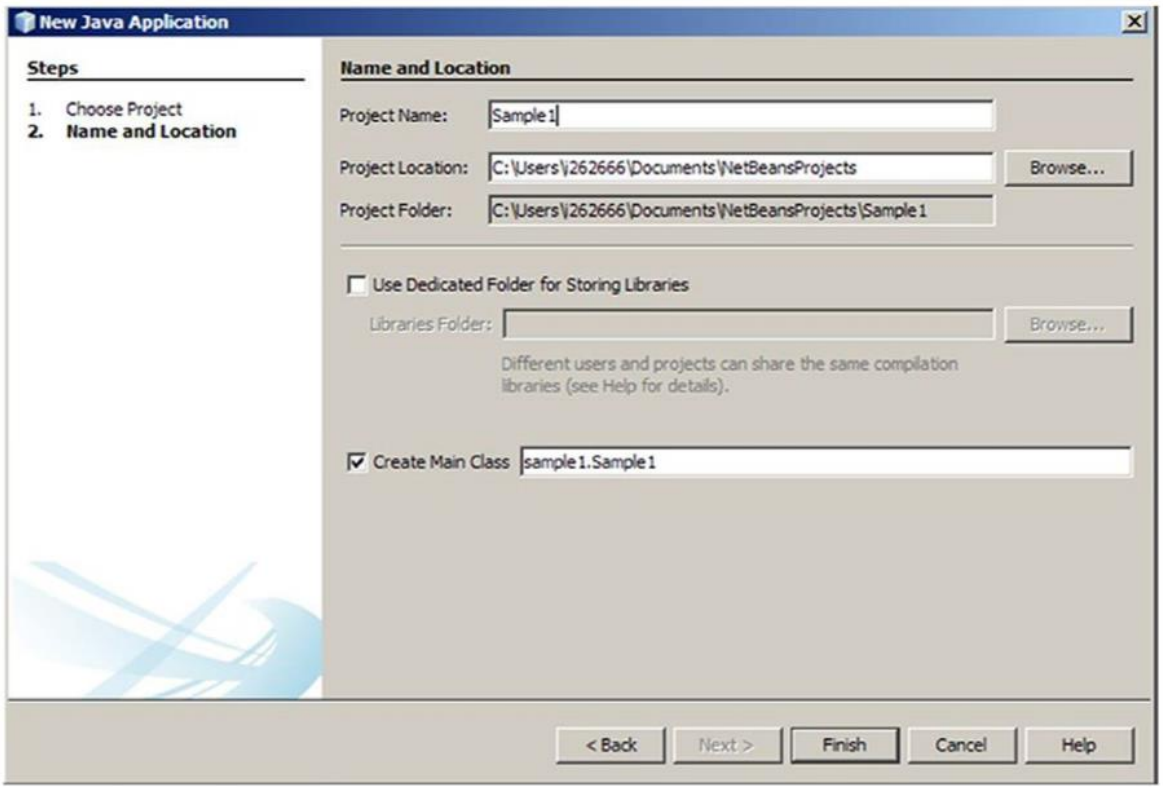
Membangun Program Pemrosesan Jaringan Saraf

Untuk membuat proyek baru, pilih File → New Project. Dialog yang ditunjukkan pada Gambar 15.9 muncul.



Gambar 15.9 NetBeans IDE, dengan dialog Proyek Baru terbuka

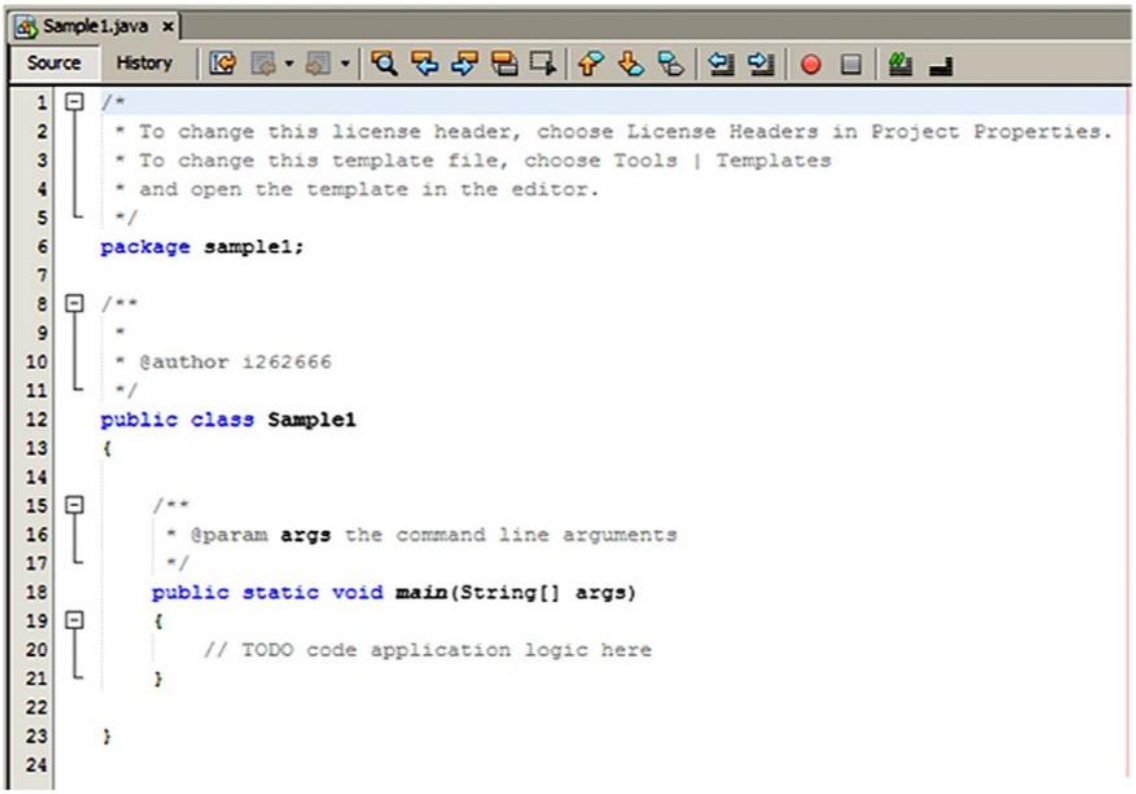
Klik Berikutnya. Pada layar berikutnya, yang ditunjukkan pada Gambar 15.10, masukkan nama proyek dan klik tombol Selesai.



Gambar 15.10 Proyek NetBeans baru

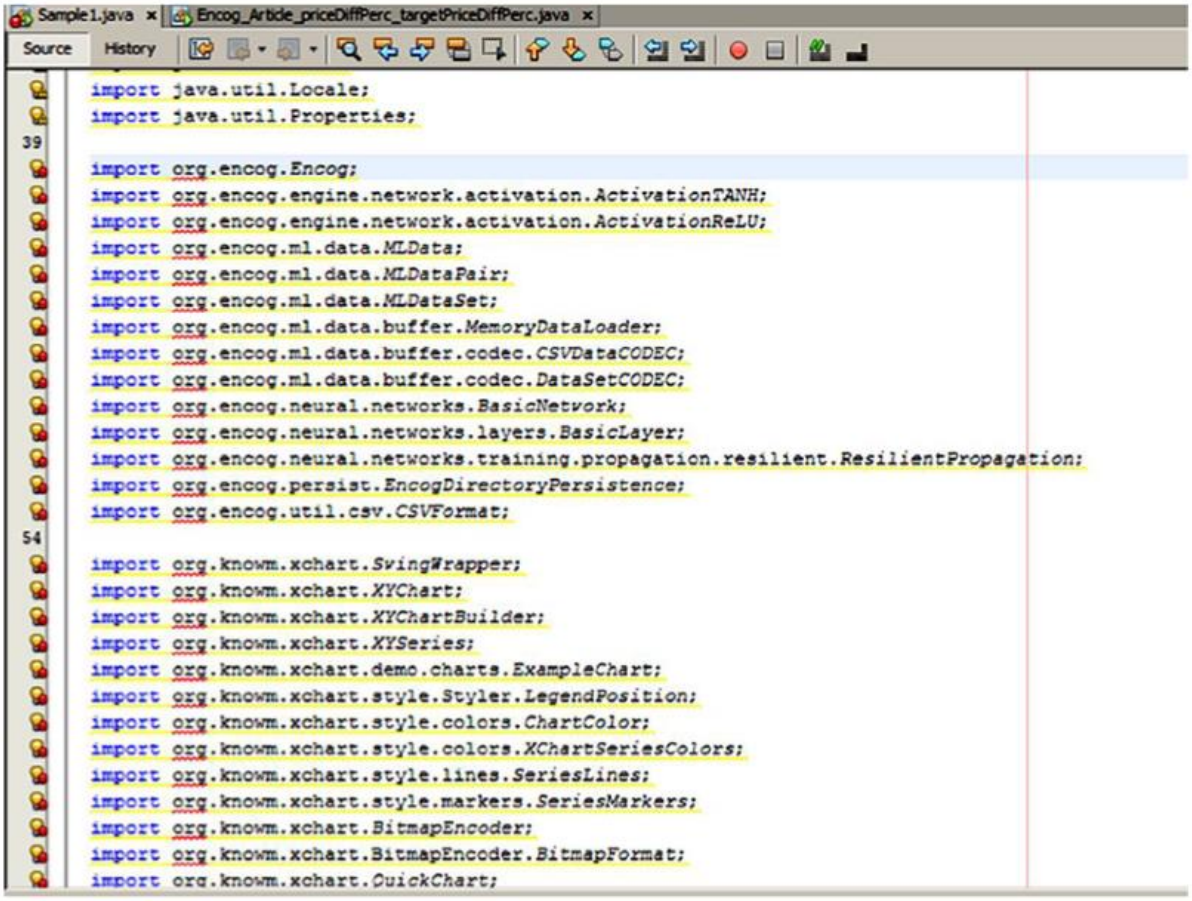
Proyek telah dibuat, dan Anda akan melihatnya di jendela Navigasi (Gambar 15.11). Kode sumber program muncul di jendela kode sumber (Gambar 15.12).

Gambar 15.11 Contoh Proyek1

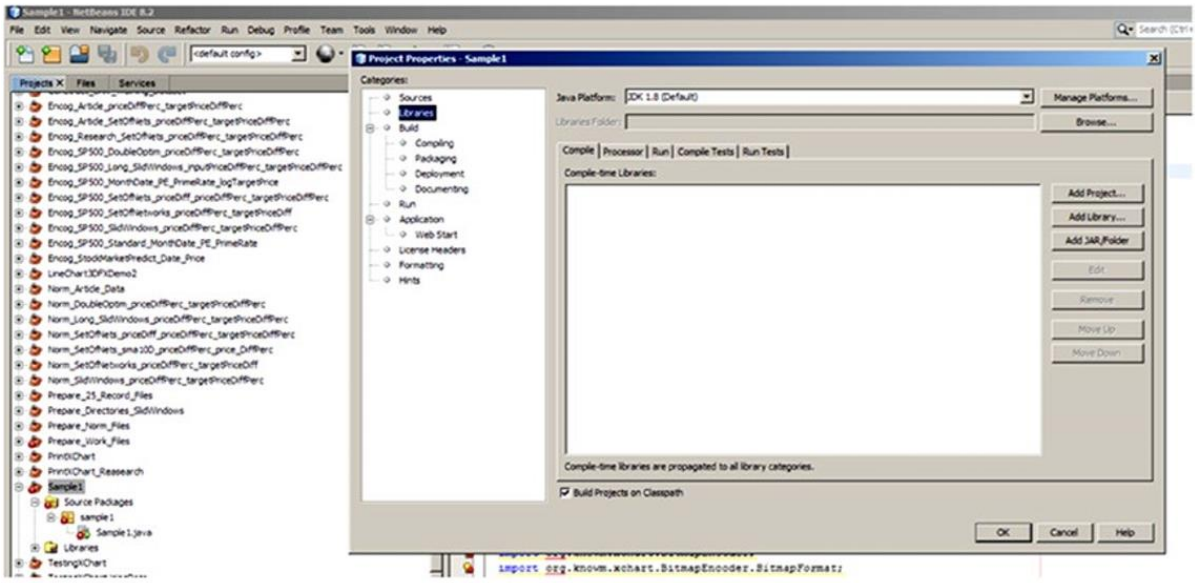


Gambar 15.12 Kode sumber untuk program Sample1.java1

Sekali lagi, ini hanyalah kerangka program Java yang dibuat secara otomatis. Mari tambahkan logika yang diperlukan di sini. Pertama, sertakan semua file impor yang diperlukan. Ada tiga kelompok pernyataan impor (impor Java, impor Encog, dan impor XChart), dan dua di antaranya (yang dimiliki Encog dan berikutnya milik XChart) ditandai sebagai kesalahan. Ini karena NetBeans IDE tidak dapat menemukannya (Gambar 15.13). Untuk memperbaikinya, klik kanan proyek dan pilih Properties. Dialog Properti Proyek akan muncul (Gambar 15.14).

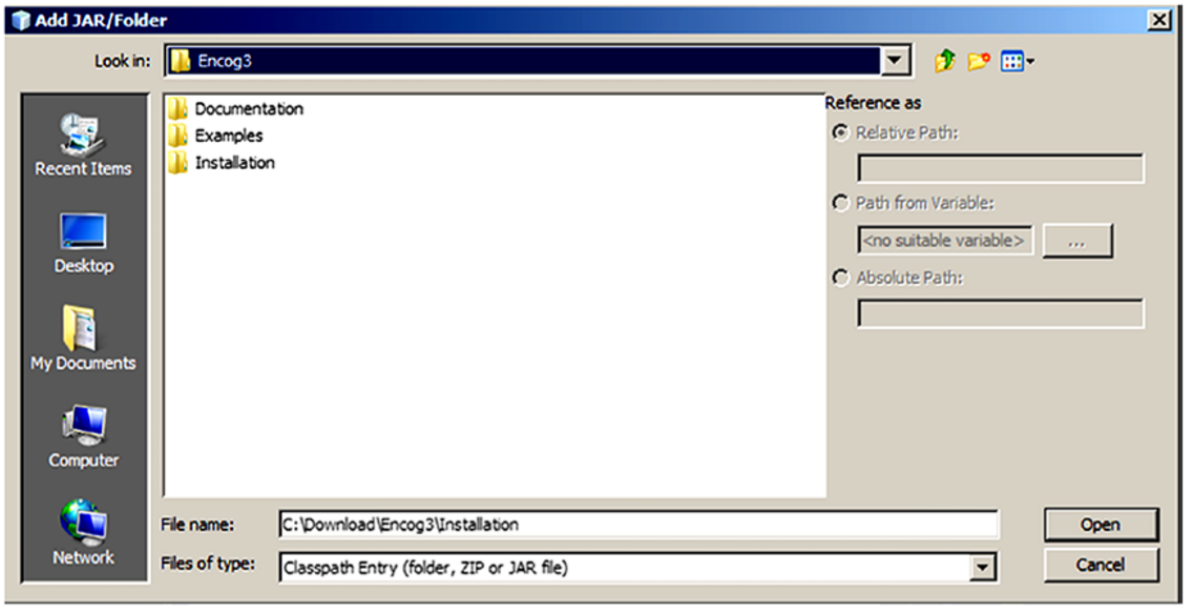


Gambar 15.13 Impor pernyataan yang ditandai sebagai kesalahan



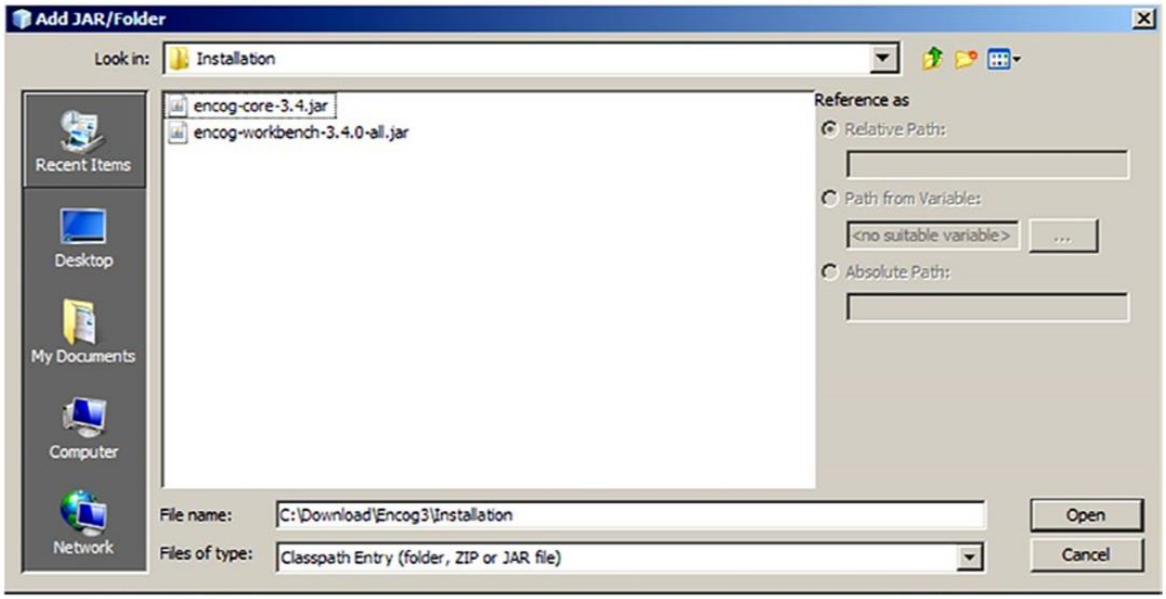
Gambar 15.14 Dialog Properti Proyek

Di kolom kiri dialog Properti Proyek, pilih Library. Klik tombol Add JAR/Folder di sebelah kanan dialog Project Properties. Klik panah bawah di bidang Java Platform (di bagian atas layar) dan pergi ke lokasi di mana paket Encog diinstal (Gambar 15.15).



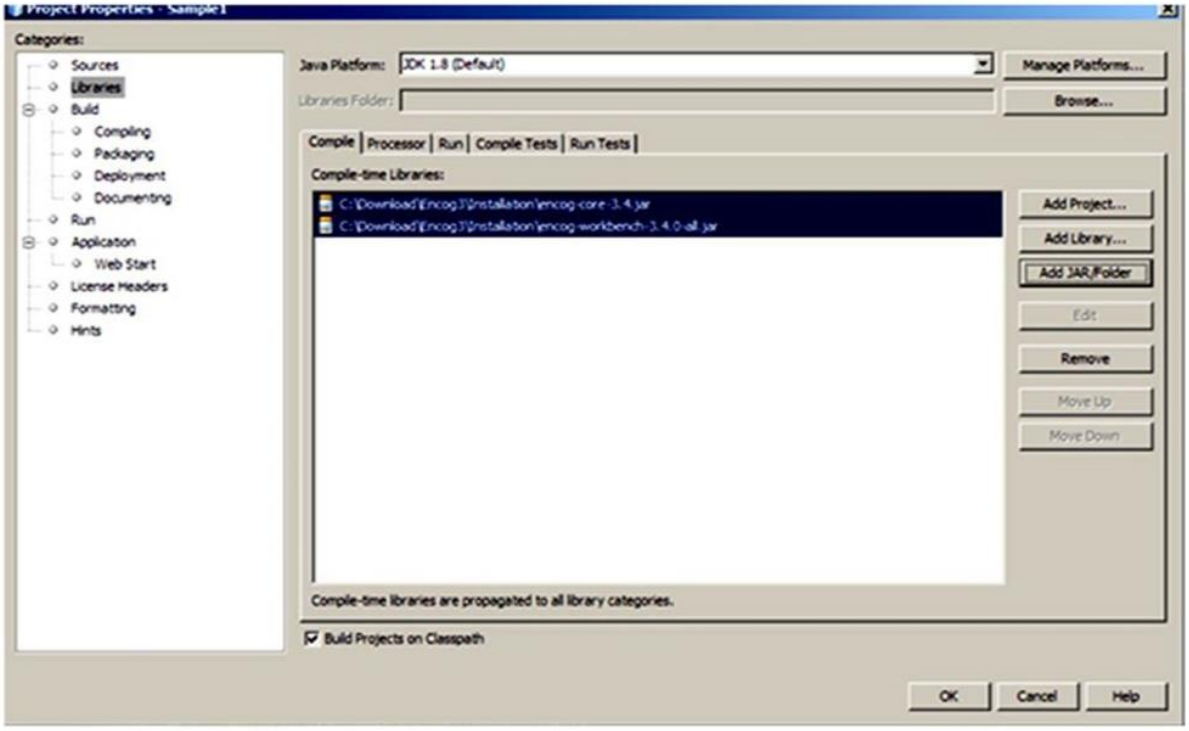
Gambar 15.15 Lokasi pemasangan Encog

Klik dua kali folder Instalasi, dan dua file JAR akan ditampilkan (Gambar 15.16).



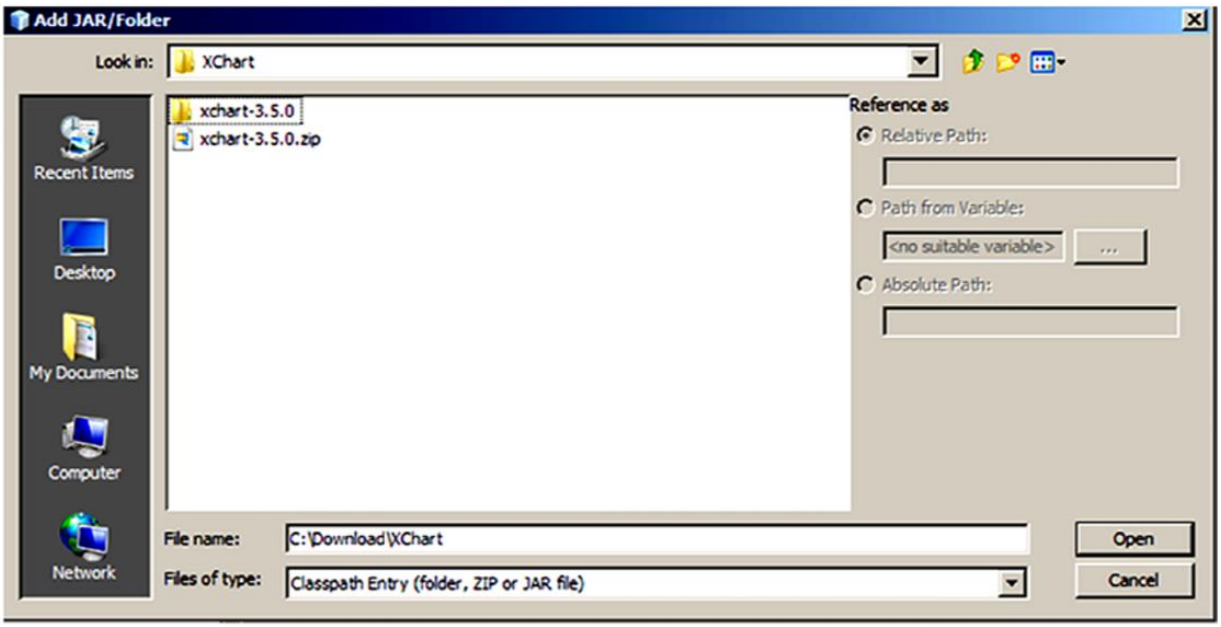
Gambar 15.16 Encog lokasi file JAR

Pilih kedua file JAR dan klik tombol Open. Mereka akan disertakan dalam daftar file JAR yang akan ditambahkan ke NetBeans IDE (Gambar 15.17).



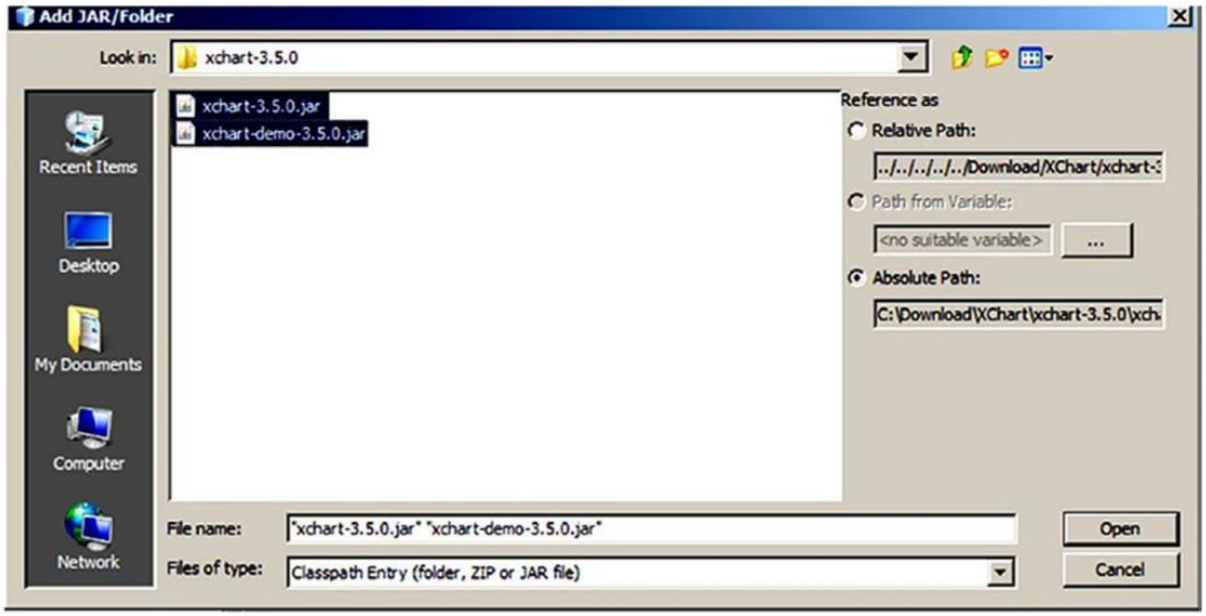
Gambar 15.17 Daftar file Encog JAR yang akan disertakan dalam NetBeans IDE

Klik tombol Add JAR/Folder lagi. Klik lagi panah bawah bidang Java Properties dan pergi ke lokasi di mana paket XChart diinstal (Gambar 15.18).



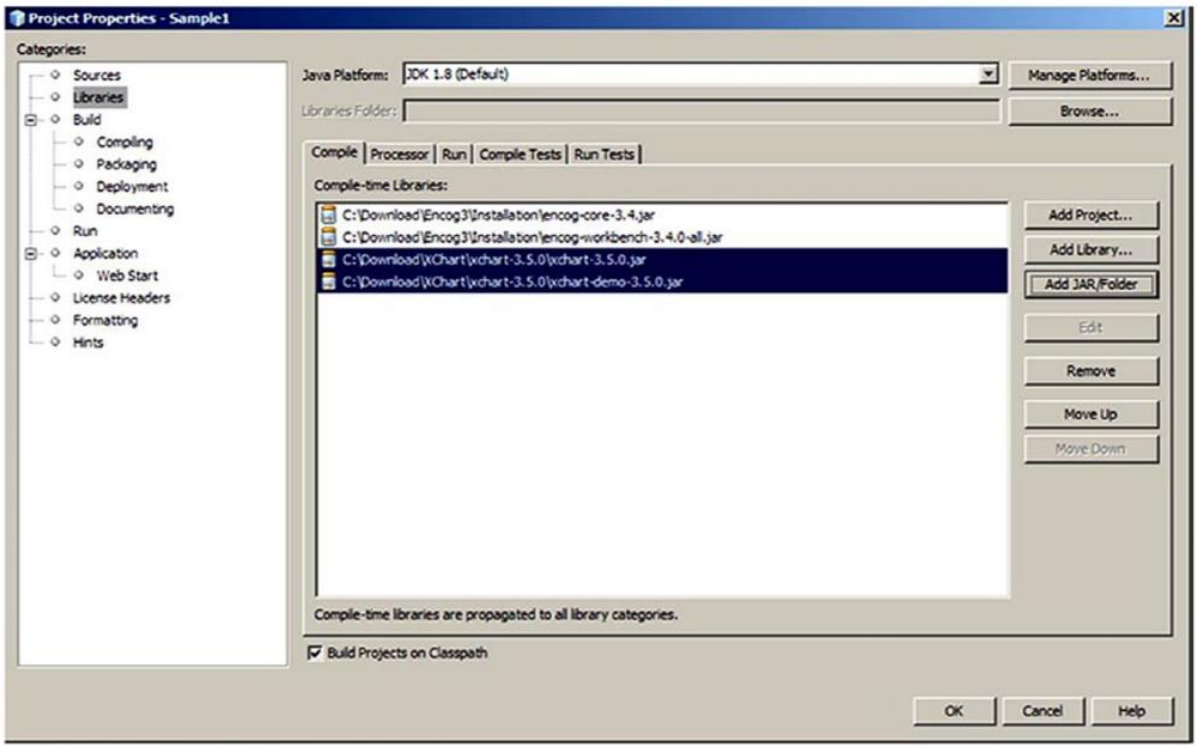
Gambar 15.18 Daftar file JAR XChart yang akan disertakan dalam NetBeans IDE

Klik dua kali folder XChart-3.5.0 dan pilih dua file XChart JAR (Gambar 15.19).



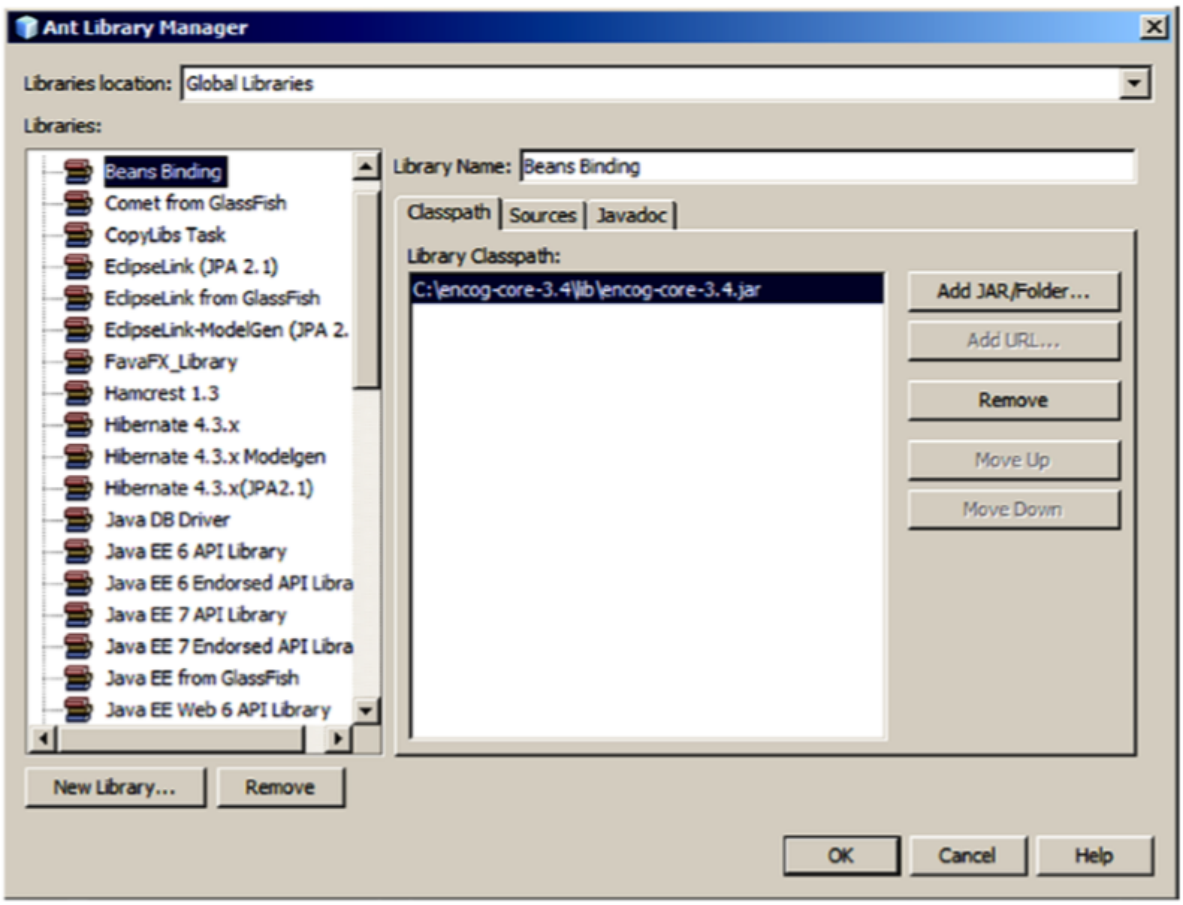
Gambar 15.19 Daftar file JAR XChart yang akan disertakan dalam NetBeans IDE

Klik tombol Open. Sekarang Anda memiliki daftar empat file JAR (dari Encog dan XChart) untuk disertakan dalam NetBeans IDE (Gambar 15.20).



Gambar 15.20 Daftar file JAR yang akan disertakan dalam NetBeans IDE

Terakhir, klik OK, dan semua kesalahan akan hilang. Alih-alih melakukan ini untuk setiap proyek baru, cara yang lebih baik adalah mengatur library global baru. Dari bilah utama, pilih Tool → Library. Dialog yang ditunjukkan pada Gambar 5.21 akan muncul.



Gambar 15.21 Membuat library global

Sekarang, Anda dapat mengulangi langkah-langkah yang sama di tingkat proyek. Lakukan ini dengan mengklik tombol Add JAR/Folder dua kali (untuk Encog dan XChart) dan tambahkan file JAR yang sesuai untuk paket Encog dan XChart.

15.3 KODE PROGRAM

Pada bagian ini, saya akan membahas semua bagian penting dari kode program menggunakan Encog. Ingatlah bahwa Anda dapat menemukan dokumentasi tentang semua API Encog dan banyak contoh pemrograman di situs web Encog. Lihat Daftar 15-2.

Daftar 15-2. Kode Program Pemrosesan Jaringan

```
// ========================================================
// Perkirakan fungsi variabel tunggal yang nilainya diberikan
// pada 9 titik.
// File train/test input dinormalisasi.
package sample2;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

```
import java.io.InputStream;
import java.nio.file.*;
 import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import  org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample2 implements ExampleChart<XYChart>
```

```
{
        // Interval to normalize
        static double Nh =  1;
        static double Nl = -1;
// First column   static double minXPointDl = 0.00;
static double maxXPointDh = 5.00;
// Second column - target data
static double minTargetValueDl = 0.00;
static double maxTargetValueDh = 5.00;
static double doublePointNumber = 0.00;
static int intPointNumber = 0;
static InputStream input = null;
static int intNumberOfRecordsInTrainFile;
static double[] arrPrices = new double[2500];
static double normInputXPointValue = 0.00;
static double normPredictXPointValue = 0.00;
static double normTargetXPointValue = 0.00;
static double normDifferencePerc = 0.00;
static double denormInputXPointValue = 0.00;
static double denormPredictXPointValue = 0.00;
static double denormTargetXPointValue = 0.00;
static double valueDifference = 0.00;
static int returnCode  = 0;
static int numberOfInputNeurons;
static int numberOfOutputNeurons;
static int intNumberOfRecordsInTestFile;
static String trainFileName;
static String priceFileName;
static String testFileName;
static String chartTrainFileName;
static String chartTestFileName;
static String networkFileName;
static int workingMode;
static String cvsSplitBy = ",";
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static XYChart Chart;
@Override
public XYChart getChart()
{
        // Create Chart
        Chart                                   =                                   new
        XYChartBuilder().width(900).height(500).title(getClass().getSimpleName()).xAxisTit
        le("x").yAxisTitle("y= f(x)").build();
        // Customize Chart          Chart.getStyler().setPlotBackgroundColor(ChartColor.
        getAWTColor(ChartColor.GREY));
        Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
```

```
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font. PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
// Set the workin mode the program should run (workingMode = 1 – training,
// workingMode = 2 – testing)
workingMode = 1;
try
        {
                )
                If (workingMode == 1)
                        {
                // Config for training the network
                workingMode = 1;
                intNumberOfRecordsInTrainFile = 10;
                trainFileName  =    "C:/My_Neural_Network_Book/Book_Examples/
                Sample2_Train_Norm.csv";
                chartTrainFileName = "Sample2_XYLine_Train_Results_Chart";
        }
        else
        {
                // Config for testing the trained network
                // workingMode = 2;
                // intNumberOfRecordsInTestFile = 10;
                // testFileName =  "C:/My_Neural_Network_Book/Book_Examples/
                Sample2_Test_Norm.csv";
                //  chartTestFileName = "XYLine_Test_Results_Chart";
                }
                // Common configuration data
                networkFileName                                                 =
                "C:/Book_Examples/Sample2_Saved_Network_File.csv";
                numberOfInputNeurons = 1;
                numberOfOutputNeurons = 1;
```

```
                    // Check the working mode to run
                    // Training mode.
                    if(workingMode == 1)
                    {
                            File file1 = new File(chartTrainFileName);
                            File file2 = new File(networkFileName);
                       if(file1.exists())
                            file1.delete();
                            f(file2.exists())
                            file2.delete();
                            returnCode = 0;
                            // Clear the return code variable
                            do
                            {
                            returnCode = trainValidateSaveNetwork();
                            } while (returnCode > 0);
                    }
                    // Test mode.
                    if(workingMode == 2)
                    {
                    // Test using the test dataset as input
                    loadAndTestNetwork();
                    }
                    }
                    catch (NumberFormatException e)
                    {
                    System.err.println("Problem parsing workingMode.  workingMode =
                    " + workingMode);
                    System.exit(1);
                    }
                    catch (Throwable t)
                    {         t.printStackTrace();
                    System.exit(1);
                    }
                    finally
                    {        Encog.getInstance().shutdown();
                    }       Encog.getInstance().shutdown();
                    return Chart;
            } // End of the method
/-------------------------------------------------------
// Load CSV to memory.
// @return The loaded dataset.
// -------------------------------------------------------
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal,
boolean headers,
CSVFormat       format, boolean significance)
{
```

```
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal,        significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);      MLDataSet dataset
        = load.external2Memory();
        return dataset;
        }
// ========================================================
// The main method.
// @param Command line arguments. No arguments are used.
// ========================================================
public static void main(String[] args)
{
ExampleChart<XYChart> exampleChart = new Sample2();
XYChart Chart = exampleChart.getChart();
new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ========================================================
// Metode pelatihan. Latih, validasi, dan simpan file jaringan
// yang terlatih
// ========================================================
static public int trainValidateSaveNetwork()
{
// Load the training CSV file in memory
MLDataSet = trainingSet
loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
// create a neural network     BasicNetwork network = new BasicNetwork();
// Input layer     network.addLayer(new BasicLayer(null,true,1));
// Hidden layer     network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
// Output layer     network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// Train the neural network                final ResilientPropagation train = new
ResilientPropagation(network, trainingSet);
int epoch = 1;
returnCode = 0;
do
{
        train.iteration();
        System.out.println("Epoch #" + epoch + " Error:" + train.getError());
        epoch++;
        if (epoch >= 500 && network.calculateError(trainingSet) > 0.000000031)
```

```
        {
        returnCode = 1;
        System.out.println("Try again");
        return returnCode;
}
} while (network.calculateError(trainingSet) > 0.00000003);
// Save the network file EncogDirectoryPersistence.saveObject(new
File(networkFileName),network);
System.out.println("Neural Network Results:");
double sumNormDifferencePerc = 0.00;
double averNormDifferencePerc = 0.00;
double maxNormDifferencePerc = 0.00;
int m = -1;
double xPointer = -1.00;
for(MLDataPair pair: trainingSet)
{
        m++;
        xPointer = xPointer + 2.00;
        //if(m == 0)
        // continue;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results normInputXPointValue =
        inputData.getData(0);
        normTargetXPointValue = actualData.getData(0);
        normPredictXPointValue = predictData.getData(0);
        denormInputXPointValue = ((minXPointDl
        maxXPointDh)*normInputXPointValue - Nh*minXPointDl + maxXPointDh
        *Nl)/(Nl - Nh);
        denormTargetXPointValue = ((minTargetValueDl - maxTargetValueDh)*
        normTargetXPointValue - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
        normPredictXPointValue - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        valueDifference = Math.abs(((denormTargetXPointValue
        denormPredictXPointValue)/denormTargetXPointValue)*100.00);
        System.out.println ("xPoint = " + denormTargetXPointValue + "
        denormPredictXPointValue = " + denormPredictXPointValue + "
        valueDifference = " + valueDifference);
        sumNormDifferencePerc = sumNormDifferencePerc + valueDifference;
        if (valueDifference > maxNormDifferencePerc) maxNormDifferencePerc =
        valueDifference;
        xData.add(denormInputXPointValue);
        yData1.add(denormTargetXPointValue);
        yData2.add(denormPredictXPointValue);
```

```
    }  // End for pair loop
    XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
    XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLUE);
    series2.setMarkerColor(Color.ORANGE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.SOLID);
    try
    {
    //Save the chart image BitmapEncoder.saveBitmapWithDPI(Chart,
    chartTrainFileName, BitmapFormat.JPG, 100);
    System.out.println ("Train Chart file has been saved") ;
    }
    catch (IOException ex)
    {
    ex.printStackTrace();
    System.exit(3);
    }
    // Finally, save this trained network EncogDirectoryPersistence.saveObject(new
    File(networkFileName),network);
    System.out.println ("Train Network has been saved") ;
    averNormDifferencePerc  = sumNormDifferencePerc/ intNumberOfRecordsInTrainFile;
    System.out.println(" ");
    System.out.println("maxErrorDifferencePerc = " + maxNormDifferencePerc + "
    averErrorDifferencePerc = " + averNormDifferencePerc);
    returnCode = 0;
    return returnCode;
    }  // End of the method
// ============================================================
// Muat dan uji jaringan terlatih pada titik yang tidak digunakan
// dalam pelatihan
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double targetToPredictPercent = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double maxGlobalIndex = 0;
        double normInputXPointValueFromRecord = 0.00;
        double normTargetXPointValueFromRecord = 0.00;
        double normPredictXPointValueFromRecord = 0.00;
        BufferedReader br4;
        BasicNetwork network;
```

```
int k1 = 0;    int k3 = 0;
maxGlobalResultDiff = 0.00;
averGlobalResultDiff = 0.00;
sumGlobalResultDiff = 0.00;
// Load the test dataset into memory MLDataSet testingSet =
loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutputNeuron
s,    true,CSVFormat.ENGLISH,false);
// Load the saved trained network network =
(BasicNetwork)EncogDirectoryPersistence.loadObject(new File(networkFileName));
int i = - 1;    double xPoint = -0.00;
for (MLDataPair pair:  testingSet)
{
        i++;        xPoint = xPoint + 2.00;
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // These values are Normalized as the whole input is
        normInputXPointValueFromRecord = inputData.getData(0);
        normTargetXPointValueFromRecord = actualData.getData(0);
        normPredictXPointValueFromRecord = predictData.getData(0);
        // De-normalize the obtained values denormInputXPointValue =
        ((minXPointDl - maxXPointDh)* normInputXPointValueFromRecord -
        Nh*minXPointDl + maxXPointDh*Nl)/(Nl - Nh);
        denormTargetXPointValue = ((minTargetValueDl - maxTargetValueDh)*
        normTargetXPointValueFromRecord - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
            denormPredictXPointValue =((minTargetValueDl -
        maxTargetValueDh)* normPredictXPointValueFromRecord -
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
        targetToPredictPercent = Math.abs((denormTargetXPointValue
        denormPredictXPointValue)/denormTargetXPointValue*100);
        System.out.println("xPoint = " + denormInputXPointValue + "
        denormTargetXPointValue = " + denormTargetXPointValue + "
        denormPredictXPointValue = " + denormPredictXPointValue + "
        targetToPredictPercent = " + targetToPredictPercent);
        if (targetToPredictPercent > maxGlobalResultDiff)
        maxGlobalResultDiff = targetToPredictPercent;
        sumGlobalResultDiff = sumGlobalResultDiff + targetToPredictPercent;
        // Populate chart elements xData.add(denormInputXPointValue);
        yData1.add(denormTargetXPointValue);
        yData2.add(denormPredictXPointValue);
} // End for pair loop
// Print the max and average results System.out.println(" ");
averGlobalResultDiff = sumGlobalResultDiff/intNumberOfRecordsInTestFile;
System.out.println("maxErrorDifferencePercent = " + maxGlobalResultDiff);
System.out.println("averErrorDifferencePercent = " + averGlobalResultDiff);
```

```
        // All testing batch files have been processed          XYSeries series1 =
        Chart.addSeries("Actual", xData, yData1);                XYSeries series2 =
        Chart.addSeries("Predicted", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image     try
        {
                BitmapEncoder.saveBitmapWithDPI(Chart,        chartTestFileName        ,
                BitmapFormat.JPG, 100);
                }
                catch (Exception bt)
                {
                bt.printStackTrace();
                }
                System.out.println ("The Chart has been saved");
                System.out.println("End of testing for test records");
        } // End of the method
} // End of the class
```

Di bagian atas, ada satu set instruksi yang diperlukan oleh paket XChart, dan mereka memungkinkan Anda untuk mengonfigurasi tampilan grafik (Daftar 15-3).

Daftar 15-3. Kumpulan Instruksi Yang Diperlukan oleh Paket XChart

```
static XYChart Chart;
        @Override
        public XYChart getChart()
        {
                // Create Chart                Chart = new
                XYChartBuilder().width(900).height(500).title(getClass().
                getSimpleName()).xAxisTitle("x").yAxisTitle("y= f(x)").build();
                // Customize Chart
                Chart.getStyler().setPlotBackgroundColor(ChartColor.
                getAWTColor(ChartColor.GREY));
                Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
                Chart.getStyler().setChartBackgroundColor(Color.WHITE);
                Chart.getStyler().setLegendBackgroundColor(Color.PINK);
                Chart.getStyler().setChartFontColor(Color.MAGENTA);
                Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
                Chart.getStyler().setChartTitleBoxVisible(true);
                Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
                Chart.getStyler().setPlotGridLinesVisible(true);
                Chart.getStyler().setAxisTickPadding(20);
                Chart.getStyler().setAxisTickMarkLength(15);
                Chart.getStyler().setPlotMargin(20);
                Chart.getStyler().setChartTitleVisible(false);
                Chart.getStyler().setChartTitleFont(new        Font(Font.MONOSPACED,
                Font.BOLD, 24));
                Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
```

```
Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC,
18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.PLAIN,
11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
```

Program dapat dijalankan dalam dua mode. Dalam mode pertama (pelatihan, workingMode = 1), program melatih jaringan, menyimpan jaringan yang dilatih pada disk, mencetak hasilnya, menampilkan hasil grafik, dan menyimpan grafik pada disk. Dalam mode kedua (pengujian, workingMode = 2), program memuat jaringan terlatih yang disimpan sebelumnya, menghitung nilai prediksi pada titik-titik yang tidak digunakan dalam pelatihan jaringan, mencetak hasilnya, menampilkan grafik, dan menyimpan grafik di disk. Program harus selalu dijalankan dalam mode pelatihan terlebih dahulu, karena mode kedua tergantung pada hasil pelatihan yang dihasilkan dalam mode pelatihan. Konfigurasi saat ini diatur untuk menjalankan program dalam mode pelatihan (lihat Daftar 15-4).

Daftar 15-4. Fragmen Kode dari Kode Metode Pelatihan

```
// Set the workin mode the program should run:     (workingMode = 1 – training,
workingMode = 2 – testing)
workingMode = 1;
try
        {
                If (workingMode == 1)
                {
                // Config for training the network          workingMode = 1;
                intNumberOfRecordsInTrainFile = 10;
                trainFileName =   "C:/My_Neural_Network_Book/Book_Examples/
                Sample2_Train_Norm.csv";
                chartTrainFileName = "Sample2_XYLine_Train_Results_Chart";
        }
        else
        {
                // Config for testing the trained network
                // workingMode = 2;
                // intNumberOfRecordsInTestFile = 10;
                // testFileName =  "C:/My_Neural_Network_Book/Book_Examples/
                Sample2_Test_Norm.csv";
                //  chartTestFileName = "XYLine_Test_Results_Chart";
                }
// Common configuration statements  (stays always uncommented)
networkFileName = "C:/Book_Examples/Saved_Network_File.csv";
numberOfInputNeurons = 1;
numberOfOutputNeurons = 1;
```

Karena workingMode saat ini disetel ke 1, program mengeksekusi metode pelatihan yang disebut trainValidateSaveNetwork(); jika tidak, ia akan memanggil metode pengujian yang disebut loadAndTestNetwork() (lihat Daftar 5-5).

Daftar 15-5. Memeriksa Nilai WorkingMode dan Menjalankan Metode yang Sesuai

```
// Check the working mode
if(workingMode == 1)
{
        // Training mode.
        File file1 = new File(chartTrainFileName);
        File file2 = new File(networkFileName);
        if(file1.exists())      file1.delete();
        if(file2.exists())      file2.delete();
        trainValidateSaveNetwork();
}
        if(workingMode == 2)
        {
                // Test using the test dataset as input      loadAndTestNetwork();
                }
}
catch (NumberFormatException e)
{
        System.err.println("Problem parsing workingMode. workingMode = " +
        workingMode);
        System.exit(1);
}
catch (Throwable t)
{
        t.printStackTrace();
        System.exit(1);
}
finally
{
Encog.getInstance().shutdown();
}
```

Daftar 15-6 menunjukkan logika metode pelatihan. Metode ini melatih jaringan, memvalidasinya, dan menyimpan file jaringan terlatih pada disk (untuk digunakan nanti oleh metode pengujian). Metode memuat kumpulan data pelatihan ke dalam memori. Parameter pertama adalah nama set data pelatihan input. Parameter kedua dan ketiga menunjukkan jumlah neuron input dan output dalam jaringan. Parameter keempat (benar) menunjukkan bahwa kumpulan data memiliki catatan label. Parameter yang tersisa menentukan format file dan bahasa.

Daftar 15-6. Fragmen Logika Pelatihan Jaringan

```
        MLDataSet trainingSet =
        loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeuro
        ns,true,CSVFormat.ENGLISH,false);
```

Setelah memuat kumpulan data pelatihan dalam memori, jaringan saraf baru dibangun dengan membuat jaringan dasar dan menambahkan lapisan input, tersembunyi, dan output ke dalamnya.

```
        // create a neural network BasicNetwork network = new BasicNetwork();
```

Berikut cara menambahkan lapisan input:

```
        network.addLayer(new BasicLayer(null,true,1));
```

Parameter pertama (null) menunjukkan bahwa ini adalah lapisan input (tidak ada fungsi aktivasi). Masukkan true sebagai parameter kedua untuk lapisan input dan tersembunyi, dan masukkan false untuk lapisan output. Parameter ketiga menunjukkan jumlah neuron pada lapisan. Selanjutnya Anda menambahkan lapisan tersembunyi.

```
network.addLayer(new BasicLayer(new ActivationTANH(),true,2));
```

Parameter pertama menentukan fungsi aktivasi yang akan digunakan (ActivationTANH()). Atau, fungsi aktivasi lain dapat digunakan seperti fungsi sigmoid yang disebut ActivationSigmoid(), fungsi logaritmik yang disebut ActivationLOG(), relai linier yang disebut ActivationReLU(), dan seterusnya. Parameter ketiga menentukan jumlah neuron di lapisan ini. Untuk menambahkan lapisan tersembunyi kedua, cukup ulangi pernyataan sebelumnya. Terakhir, tambahkan layer output, seperti yang ditunjukkan di sini:

```
network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
```

Parameter ketiga menentukan jumlah neuron di lapisan output. Dua pernyataan berikutnya menyelesaikan pembuatan jaringan:

```
network.getStructure().finalizeStructure();
network.reset();
```

Untuk melatih jaringan yang baru dibangun, Anda menentukan jenis propagasi balik. Di sini, Anda menentukan propagasi tangguh; ini adalah jenis propagasi paling canggih. Sebagai alternatif, tipe propagasi balik biasa dapat ditentukan di sini.

```
final ResilientPropagation train = new ResilientPropagation(network, trainingSet);
```

Saat jaringan dilatih, Anda mengulang jaringan. Pada setiap langkah loop, Anda mendapatkan nomor iterasi pelatihan berikutnya, menambah nomor epoch (lihat Bab sebelumnya untuk definisi epoch), dan periksa apakah kesalahan jaringan untuk iterasi saat ini dapat menghapus batas kesalahan yang disetel ke 0,00000003. Ketika kesalahan pada iterasi saat ini akhirnya menjadi kurang dari batas kesalahan, Anda keluar dari loop. Jaringan telah dilatih, dan Anda menyimpan jaringan terlatih pada disk. Jaringan juga tetap berada di memori.

```
int epoch = 1;
do
        {
                train.iteration();
                System.out.println("Epoch #" + epoch + " Error:" + train.getError());
                epoch++;
                }
                while (network.calculateError(trainingSet) > 0.00000046);
                // Save the network file
                EncogDirectoryPersistence.saveObject(new
                File(networkFileName),network);
```

Bagian kode berikutnya mengambil nilai input, aktual, dan prediksi untuk setiap record dalam kumpulan data pelatihan. Pertama, objek inputData, actualData, dan predictData dibuat.

```
MLData inputData = pair.getInput();
MLData actualData = pair.getIdeal();
MLData predictData = network.compute(inputData);
```

Setelah melakukannya, Anda mengulangi objek pasangan MLDataPair dengan menjalankan instruksi berikut:

```
normInputXPointValue = inputData.getData(0);
normTargetXPointValue = actualData.getData(0);
```

normPredictXPointValue = predictData.getData(0);

Satu bidang dalam objek inputData, actualData, dan predictData memiliki perpindahan nol. Dalam contoh ini, hanya ada satu field input dan satu field output dalam sebuah record. Jika record memiliki dua field input, Anda akan menggunakan pernyataan berikut untuk mengambil semua field input:

normInputXPointValue1 = inputData.getData(0);
normInputXPointValue2 = inputData.getData(1);

Sebaliknya, jika catatan memiliki dua bidang target, Anda akan menggunakan pernyataan serupa untuk mengambil semua bidang target, seperti yang ditunjukkan di sini:

normTargeValue1 = actualData.getData(0);
normTargeValue2 = actualData.getData(1);

Nilai prediksi diproses dengan cara yang sama. Nilai prediksi memprediksi nilai target untuk poin berikutnya. Nilai yang diambil dari jaringan dinormalisasi karena kumpulan data pelatihan yang proses jaringannya telah dinormalisasi. Setelah nilai tersebut diambil, Anda dapat mendenormalisasinya. Denormalisasi dilakukan dengan menggunakan rumus berikut:

$$F(x) = ((D_L – D_H) * x – N_H * D_L + D_H * N_L) / (N_L – N_H)$$

di mana:

x: Input titik data
DL: Nilai min (terendah) x dalam kumpulan data input
DH: Nilai maksimum (tertinggi) x dalam kumpulan data input
NL: Bagian kiri dari interval ternormalisasi [-1, 1]
NH: Bagian kanan dari interval ternormalisasi [-1, 1]

denormInputXPointValue = ((minXPointDl - maxXPointDh)*normInputXPointValue - Nh*minXPointDl + maxXPointDh *Nl)/(Nl - Nh);
denormTargetXPointValue = ((minTargetValueDl - maxTargetValueDh)*normTarget XPointValue – Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)* normPredictXPointValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);

Anda juga menghitung persentase kesalahan sebagai persentase perbedaan antara bidang denormTargetXPointValue dan denormPredictXPointValue. Anda dapat mencetak hasilnya, dan Anda juga dapat mengisi nilai denormTargetXPointValue dan denormPredictXPointValue sebagai elemen grafik untuk record xPointer yang saat ini diproses.

xData.add(denormInputXPointValue);
yData1.add(denormTargetXPointValue);
yData2.add(denormPredictXPointValue);
} // End for pair loop // End for the pair loop

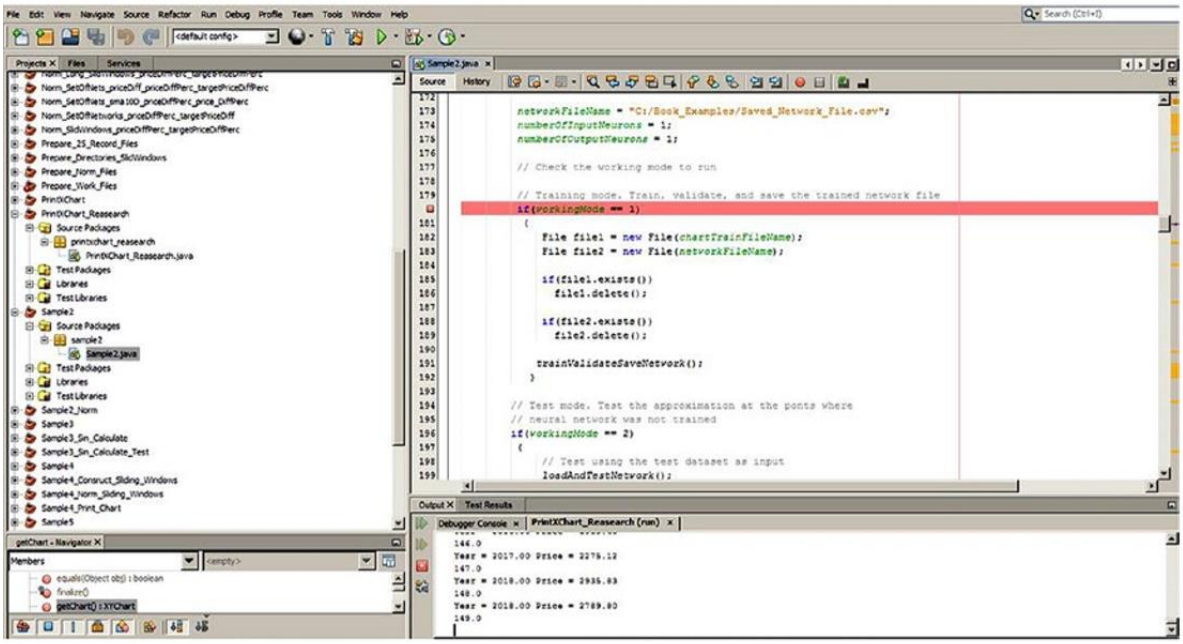
Sekarang simpan file bagan pada disk dan juga hitung selisih rata-rata dan persentase maksimum antara nilai aktual dan prediksi untuk semua rekaman yang diproses. Setelah keluar dari loop pasangan, Anda dapat menambahkan beberapa instruksi yang diperlukan oleh grafik untuk mencetak seri grafik dan menyimpan file grafik pada disk.

XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);

```
try
{
        //Save the chart image BitmapEncoder.saveBitmapWithDPI(Chart,
        chartTrainFileName, BitmapFormat.JPG, 100);
        System.out.println ("Train Chart file has been saved") ;
}
catch (IOException ex)
{
        ex.printStackTrace();
        System.exit(3);   }
// Finally, save this trained network
EncogDirectoryPersistence.saveObject(new File(networkFileName),network);
System.out.println ("Train Network has been saved") ;
averNormDifferencePerc  = sumNormDifferencePerc/4.00; System.out.println(" ");
System.out.println("maxErrorPerc = " + maxNormDifferencePerc +      "averErrorPerc = " +
averNormDifferencePerc);
} // End of the method
```

15.4 *DEBUGGING* DAN MENGEKSEKUSI PROGRAM

Ketika pengkodean program selesai, Anda dapat mencoba menjalankan proyek, tetapi jarang bekerja dengan benar. Anda perlu men-debug program. Untuk menyetel breakpoint, cukup klik nomor baris sumber program. Gambar 15.22 menunjukkan hasil mengklik garis 180. Garis merah mengonfirmasi bahwa titik henti sementara telah disetel. Jika Anda mengklik nomor yang sama lagi, breakpoint akan dihapus.



Gambar 15.22 Mengatur breakpoint

Di sini, Anda harus menyetel breakpoint pada logika yang memeriksa mode kerja mana yang akan dijalankan. Setelah mengatur breakpoint, pilih Debug Debug Project dari menu utama. Program mulai mengeksekusi dan kemudian berhenti di breakpoint. Di sini, jika Anda memindahkan kursor di atas variabel apa pun, nilainya akan ditampilkan di

jendela pop-up. Untuk memajukan eksekusi program, klik salah satu ikon panah menu, bergantung pada apakah Anda ingin memajukan eksekusi satu baris, masuk ke dalam metode eksekusi, keluar dari metode saat ini, dan seterusnya (lihat Gambar 15.23).

Gambar 15.23 Ikon untuk memajukan eksekusi saat debugging

Untuk menjalankan program, pilih Run → Run Project dari menu. Hasil eksekusi ditampilkan di jendela log.

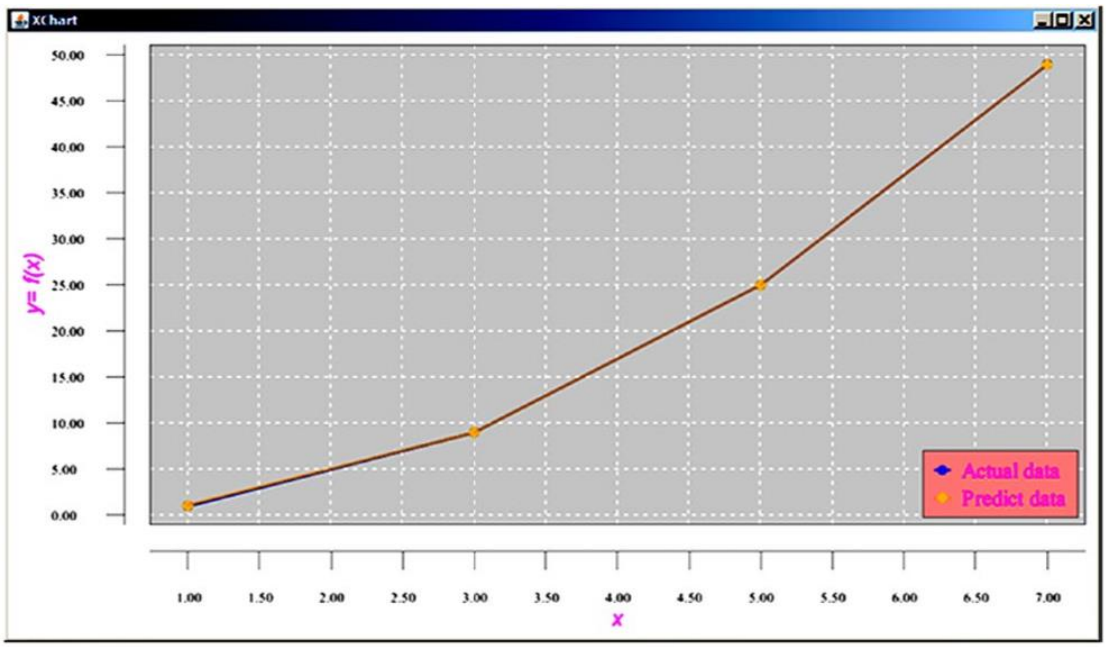
15.5 MEMPROSES HASIL UNTUK METODE PELATIHAN

Daftar 15-7 menunjukkan hasil pelatihan.

Daftar 15-7. Hasil Pemrosesan Pelatihan

RecordNumber = 0 TargetValue = 0.0224 PredictedValue = 0.022898 DiffPerc = 1.77
RecordNumber = 1 TargetValue = 0.0625 PredictedValue = 0.062009 DiffPerc = 0.79
RecordNumber = 2 TargetValue = 0.25 PredictedValue = 0.250359 DiffPerc = 0.14
RecordNumber = 3 TargetValue = 0.5625 PredictedValue = 0.562112 DiffPerc = 0.07
RecordNumber = 4 TargetValue = 1.0 PredictedValue = 0.999552 DiffPerc = 0.04
RecordNumber = 5 TargetValue = 1.5625 PredictedValue = 1.563148 DiffPerc = 0.04
RecordNumber = 6 TargetValue = 2.25 PredictedValue = 2.249499 DiffPerc = 0.02
RecordNumber = 7 TargetValue = 3.0625 PredictedValue = 3.062648 DiffPerc = 0.00
RecordNumber = 8 TargetValue = 4.0 PredictedValue = 3.999920 DiffPerc = 0.00
maxErrorPerc = 1.769902752691229
averErrorPerc = 0.2884023848904945

Rata-rata persen perbedaan kesalahan untuk semua catatan adalah 0,29 persen, dan persen perbedaan kesalahan maksimum untuk semua catatan adalah 1,77 persen. Diagram pada Gambar 15.24 menunjukkan hasil perkiraan di sembilan titik tempat jaringan dilatih.



Gambar 15.24 Bagan hasil pelatihan

Bagan aktual dan bagan yang diprediksi (perkiraan) praktis tumpang tindih di titik-titik di mana jaringan dilatih.

15.6 MENGUJI JARINGAN

Kumpulan data pengujian mencakup catatan yang tidak digunakan selama pelatihan jaringan. Untuk menguji jaringan, Anda perlu menyesuaikan pernyataan konfigurasi program untuk menjalankan program dalam mode uji. Untuk melakukannya, Anda mengomentari pernyataan konfigurasi untuk mode pelatihan dan menghapus komentar pernyataan konfigurasi untuk mode pengujian (Daftar 15-8).

Daftar 15-8. Konfigurasi untuk Menjalankan Program dalam Mode Uji

```
If (workingMode == 1)
        {// Config for training the network          workingMode = 1;
        intNumberOfRecordsInTrainFile = 10;
        trainFileName = "C:/My_Neural_Network_Book/Book_Examples/
        Sample2_Train_Norm.csv";
        chartTrainFileName = "Sample2_XYLine_Train_Results_Chart";
        }
        else
        {
                // Config for testing the trained network
                // workingMode = 2;
                // intNumberOfRecordsInTestFile = 10;
                // testFileName = "C:/My_Neural_Network_Book/Book_Examples/
                Sample2_Test_Norm.csv";
                //  chartTestFileName = "XYLine_Test_Results_Chart";
        }
// -------------------------------------------------------
// Common Configuration
// -------------------------------------------------------
networkFileName = "C:/Book_Examples/Saved_Network_File.csv";
numberOfInputNeurons = 1;
numberOfOutputNeurons = 1;
```

Logika pemrosesan metode pengujian mirip dengan metode pelatihan; namun, ada beberapa perbedaan. File input yang diproses metode sekarang adalah kumpulan data pengujian, dan metode tidak menyertakan logika pelatihan jaringan karena jaringan telah dilatih dan disimpan di disk selama eksekusi metode pelatihan. Sebagai gantinya, metode ini memuat file jaringan terlatih yang disimpan sebelumnya di memori (Daftar 15-9). Anda kemudian memuat kumpulan data pengujian dan file jaringan terlatih yang disimpan sebelumnya di memori.

Daftar 15-9. Fragmen dari Metode Pengujian

```
// Load the test dataset into memory
MLDataSet testingSet =
loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
// Load the saved trained network network =
(BasicNetwork)EncogDirectoryPersistence.loadObject(new File(networkFileName));
```

Anda mengulangi kumpulan data pasangan dan mendapatkan dari jaringan input yang dinormalisasi dan nilai aktual dan prediksi untuk setiap catatan. Selanjutnya, Anda

mendenormalisasi nilai-nilai tersebut dan menghitung persentase perbedaan rata-rata dan maksimum (antara nilai aktual dan prediksi yang didenormalisasi). Setelah mendapatkan nilai tersebut, Anda mencetaknya dan juga mengisi elemen bagan untuk setiap catatan. Terakhir, Anda menambahkan beberapa kode untuk mengontrol rangkaian grafik dan menyimpan grafik pada disk.

```
int i = - 1;
double xPoint = -0.00;
for (MLDataPair pair:  testingSet)
{
        i++;
        xPoint = xPoint + 2.00;
        // The chart accepts only double and Date variable types, not integer
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // These values are Normalized as the whole input is
        normInputXPointValueFromRecord = inputData.getData(0);
        normTargetXPointValueFromRecord = actualData.getData(0);
        normPredictXPointValueFromRecord = predictData.getData(0);
        denormInputXPointValue = ((minXPointDl - maxXPointDh)*
        normInputXPointValueFromRecord - Nh*minXPointDl +
        maxXPointDh*Nl)/(Nl - Nh);
        denormTargetXPointValue = ((minTargetValueDl - maxTargetValueDh)*
        normTargetXPointValueFromRecord - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
        normPredictXPointValueFromRecord - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        targetToPredictPercent = Math.abs((denormTargetXPointValue
        denormPredictXPointValue)/denormTargetXPointValue*100);
        System.out.println("xPoint = " + xPoint + " denormTargetXPointValue = " +
        denormTargetXPointValue + " denormPredictXPointValue = " +
        denormPredictXPointValue + " targetToPredictPercent = " +
        targetToPredictPercent);
        if (targetToPredictPercent > maxGlobalResultDiff) maxGlobalResultDiff =
        targetToPredictPercent;
            sumGlobalResultDiff = sumGlobalResultDiff + targetToPredictPercent;
        // Populate chart elements
        xData.add(denormInputXPointValue);
        yData1.add(denormTargetXPointValue);
        yData2.add(denormPredictXPointValue);
        }  // End for pair loop
        // Print the max and average results
        System.out.println("       "); averGlobalResultDiff =
        sumGlobalResultDiff/intNumberOfRecordsInTestFile;
        System.out.println("maxErrorPerc = " + maxGlobalResultDiff );
        System.out.println("averErrorPerc = " + averGlobalResultDiff);
```

```
// All testing batch files have been processed          XYSeries series1 =
Chart.addSeries("Actual", xData, yData1);
XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
// Save the chart image
try
{
        BitmapEncoder.saveBitmapWithDPI(Chart,          chartTestFileName,
        BitmapFormat.JPG, 100);
        }
        catch (Exception bt)
        {
        bt.printStackTrace();
        }
        System.out.println ("The Chart has been saved");
        System.out.println("End of testing for test records");
        } // End of the method
```

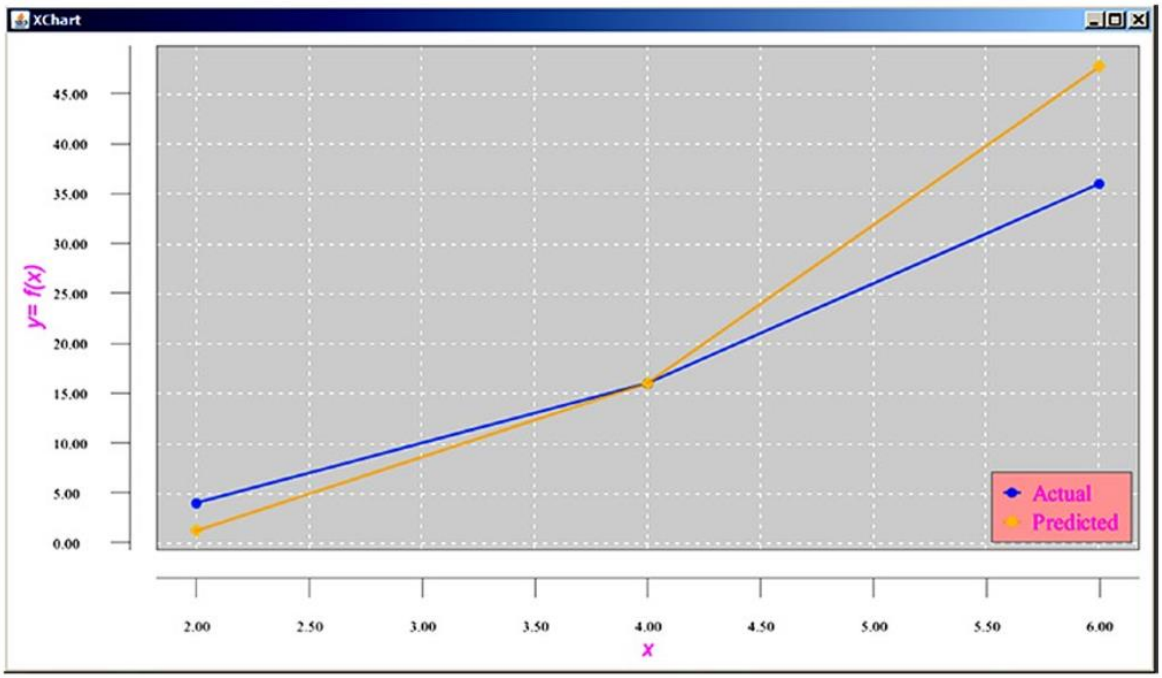
Hasil Pengujian

Listing 15-10 menunjukkan hasil pengujian.

Daftar 15-10. Hasil Pengujian

```
xPoint = 0.20 TargetValue = 0.04000 PredictedValue = 0.03785 targetToPredictDiffPerc = 5.37
xPoint = 0.30 TargetValue = 0.09000 PredictedValue = 0.09008 targetToPredictDiffPerc = 0.09
xPoint = 0.40 TargetValue = 0.16000 PredictedValue = 0.15798 targetToPredictDiffPerc = 1.26
xPoint = 0.70 TargetValue = 0.49000 PredictedValue = 0.48985 targetToPredictDiffPerc = 0.03
xPoint = 0.95 TargetValue = 0.90250 PredictedValue = 0.90208 targetToPredictDiffPerc = 0.05
xPoint = 1.30 TargetValue = 1.69000 PredictedValue = 1.69096 targetToPredictDiffPerc = 0.06
xPoint = 1.60 TargetValue = 2.56000 PredictedValue = 2.55464 targetToPredictDiffPerc = 0.21
xPoint = 1.80 TargetValue = 3.24000 PredictedValue = 3.25083 targetToPredictDiffPerc = 0.33
xPoint = 1.95 TargetValue = 3.80250 PredictedValue = 3.82933 targetToPredictDiffPerc = 0.71
maxErrorPerc = 5.369910680518282
averErrorPerc = 0.8098656579029523
```

Rata-rata error (perbedaan persen antara nilai aktual dan prediksi) adalah 5,37 persen. Kesalahan maksimum (perbedaan persen antara nilai aktual dan prediksi) adalah 0,81 persen. Gambar 15.25 menunjukkan diagram untuk hasil pengujian.



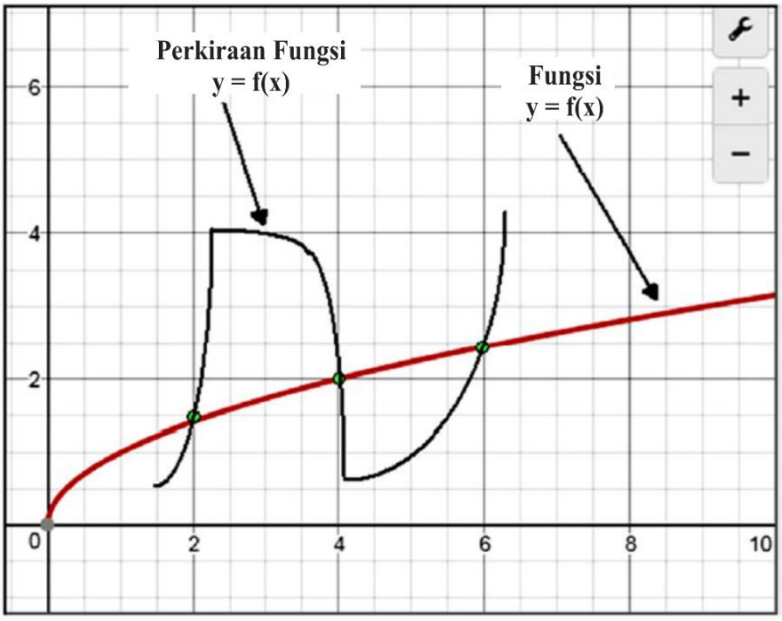
Gambar 15.25 Bagan perkiraan pada titik-titik di mana jaringan tidak dilatih

Perbedaan mencolok antara nilai aktual dan prediksi adalah karena perkiraan fungsi kasar. Anda biasanya dapat meningkatkan ketepatan aproksimasi dengan mengutak-atik arsitektur jaringan (jumlah lapisan tersembunyi, jumlah neuron dalam lapisan). Namun, masalah utama di sini adalah sejumlah kecil titik dan jarak yang relatif besar antara titik yang digunakan untuk melatih jaringan. Untuk mendapatkan hasil aproksimasi fungsi yang jauh lebih baik, Anda dapat menggunakan lebih banyak titik (dengan perbedaan yang jauh lebih kecil di antara keduanya) untuk memperkirakan fungsi ini.

Jika kumpulan data pelatihan mencakup lebih banyak titik (100; 1.000; atau bahkan 10.000) dan jarak yang jauh lebih kecil antara titik (0,01, 0,001, atau bahkan 0,0001), hasil aproksimasi akan jauh lebih tepat. Namun, itu bukan tujuan dari contoh sederhana pertama ini.

Menggali lebih dalam

Mengapa begitu banyak poin yang dibutuhkan untuk mendekati fungsi ini? Pendekatan fungsi mengontrol perilaku fungsi yang diperkirakan pada titik yang diproses selama pelatihan. Jaringan belajar untuk membuat hasil yang didekati sangat cocok dengan nilai fungsi aktual pada titik pelatihan, tetapi memiliki kontrol yang jauh lebih sedikit terhadap perilaku fungsi di antara titik pelatihan. Perhatikan Gambar 15.26.



Gambar 15.26 Fungsi asli dan perkiraan

Pada Gambar 15.26, nilai fungsi aproksimasi sangat cocok dengan nilai fungsi asli pada titik pelatihan, tetapi tidak di antara titik. Kesalahan untuk poin pengujian sengaja dibesar-besarkan untuk memperjelas poin. Jika lebih banyak titik pelatihan yang digunakan, maka titik pengujian akan selalu lebih dekat dengan salah satu titik pelatihan, dan hasil pengujian pada titik pengujian akan lebih dekat dengan nilai fungsi asli pada titik tersebut.

15.7 RINGKASAN

Bab ini menjelaskan cara mengembangkan aplikasi jaringan saraf tiruan menggunakan kerangka kerja Java Encog. Anda melihat pendekatan langkah demi langkah untuk mengkodekan aplikasi jaringan saraf menggunakan Encog. Semua contoh di sisa buku ini menggunakan kerangka Encog.

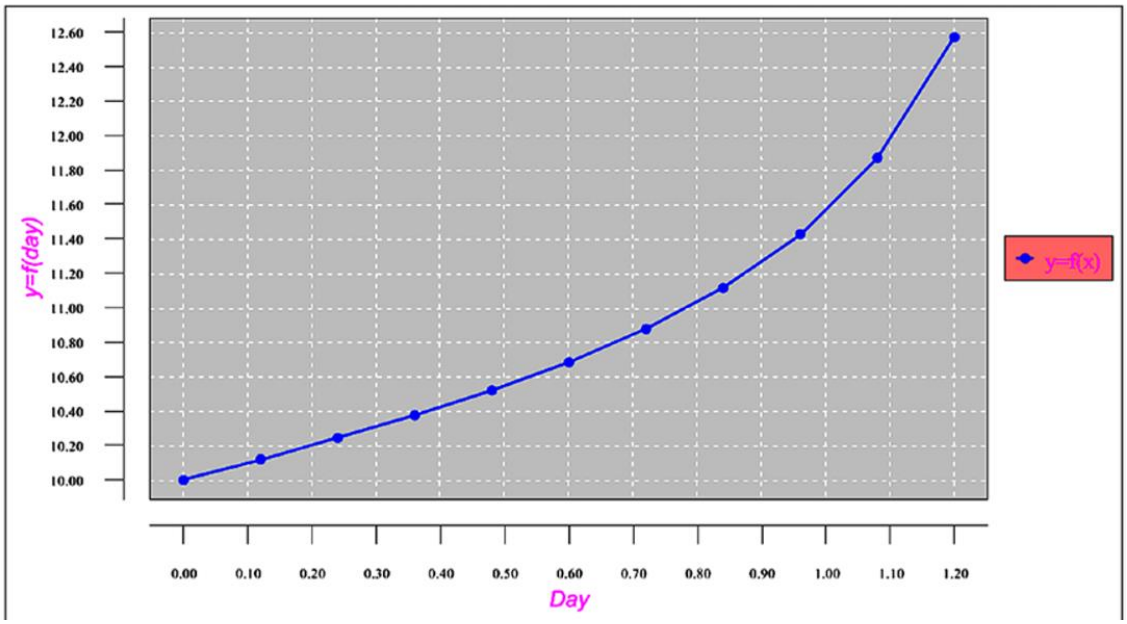
BAB 16
PREDIKSI JARINGAN *NEURAL* DI LUAR RENTANG PELATIHAN

Mempersiapkan data untuk pemrosesan jaringan saraf biasanya merupakan tugas yang paling sulit dan memakan waktu yang akan Anda hadapi saat bekerja dengan jaringan saraf. Selain volume data yang sangat besar yang dapat dengan mudah mencapai jutaan bahkan miliaran catatan, kesulitan utama adalah menyiapkan data dalam format yang benar untuk tugas yang bersangkutan. Dalam bab ini dan bab berikutnya, saya akan mendemonstrasikan beberapa teknik persiapan/transformasi data.

Tujuan dari contoh bab ini adalah untuk menunjukkan bagaimana melewati batasan utama untuk pendekatan jaringan saraf, yang menyatakan bahwa prediksi harus digunakan hanya di dalam interval pelatihan. Pembatasan ini ada untuk mekanisme aproksimasi fungsi apa pun (tidak hanya untuk aproksimasi oleh jaringan saraf tetapi juga untuk semua jenis aproksimasi seperti deret Taylor dan kalkulus aproksimasi Newtonian). Mendapatkan nilai fungsi di luar interval pelatihan disebut peramalan atau ekstrapolasi (bukan prediksi). Nilai fungsi peramalan didasarkan pada ekstrapolasi, sedangkan mekanisme pemrosesan jaringan saraf didasarkan pada mekanisme perkiraan. Mendapatkan nilai aproksimasi fungsi di luar interval pelatihan hanya menghasilkan hasil yang salah. Ini adalah salah satu konsep penting yang harus diperhatikan.

16.1 CONTOH 3A: MENAKSIR FUNGSI PERIODIK DI LUAR RENTANG LATIHAN

Untuk contoh ini, Anda akan menggunakan fungsi periodik tangen y = tan(x). Anggap saja Anda tidak tahu jenis fungsi periodik apa yang diberikan kepada Anda; fungsi diberikan kepada Anda dengan nilai-nilainya pada titik-titik tertentu. Tabel 16.1 menunjukkan nilai fungsi pada interval [0, 1.2]. Anda akan menggunakan data ini untuk pelatihan jaringan.



Gambar 16.1 menunjukkan grafik nilai fungsi pada interval [0, 1.2].

Tabel 16.1 Nilai Fungsi pada Interval [0, 1.2]

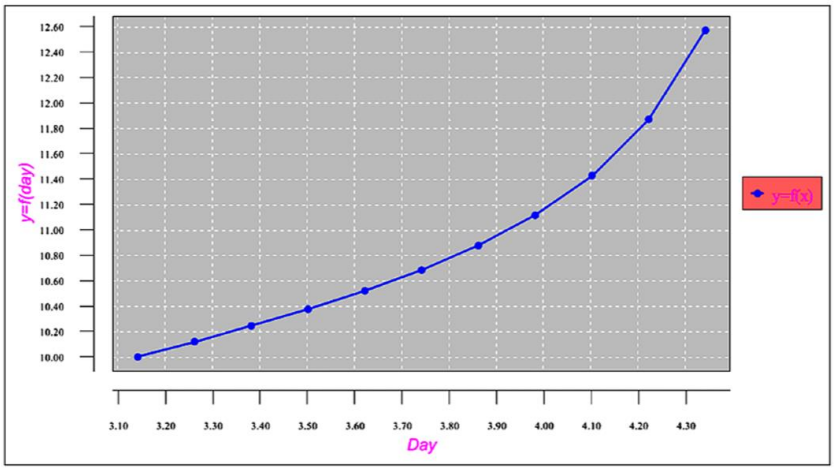
Poin x	y
0	10
0.12	10.12058
0.24	10.24472
0.36	10.3764
0.48	10.52061
0.6	10.68414
0.72	10.87707
0.84	11.11563
0.96	11.42836
1.08	11.87122
1.2	12.57215

Tabel 16.2 menunjukkan nilai fungsi pada interval [3.141592654, 4.341592654]. Anda akan menggunakan data ini untuk menguji jaringan terlatih.

Tabel 16.2 Nilai Fungsi pada Interval [3.141592654, 4.341592654]

Poin x	y
3.141593	10
3.261592	10.12058
3.381593	10.24472
3.501593	10.3764
3.621593	10.52061
3.741593	10.68414
3.861593	10.87707
3.981593	11.11563
4.101593	11.42836
4.221593	11.87122
4.341593	12.57215

Gambar 16.2 menunjukkan grafik nilai fungsi pada interval [3.141592654, 4.341592654].



Gambar 16.2 Bagan nilai fungsi pada interval [3.141592654, 4.341592654]

Tujuan dari contoh ini adalah untuk memperkirakan fungsi pada interval yang diberikan [0, 1.2] dan kemudian menggunakan jaringan terlatih untuk memprediksi nilai fungsi pada interval berikutnya, yaitu [3.141592654, 4.341592654].

Untuk Contoh 3a, Anda akan mencoba mendekati fungsi dengan cara konvensional, dengan menggunakan data yang diberikan apa adanya. Data ini perlu dinormalisasi pada interval [-1, 1]. Tabel 16.3 menunjukkan kumpulan data pelatihan yang dinormalisasi.

Tabel 16.3 Kumpulan Data Pelatihan yang Dinormalisasi

Poin x	**y**
-0.666666667	-0.5
-0.626666667	-0.43971033
-0.586666667	-0.37764165
-0.546666667	-0.311798575
-0.506666667	-0.23969458
-0.466666667	-0.157931595
-0.426666667	-0.06146605
-0.386666667	0.057816175
-0.346666667	0.214178745
-0.306666667	0.43560867
-0.266666667	0.78607581

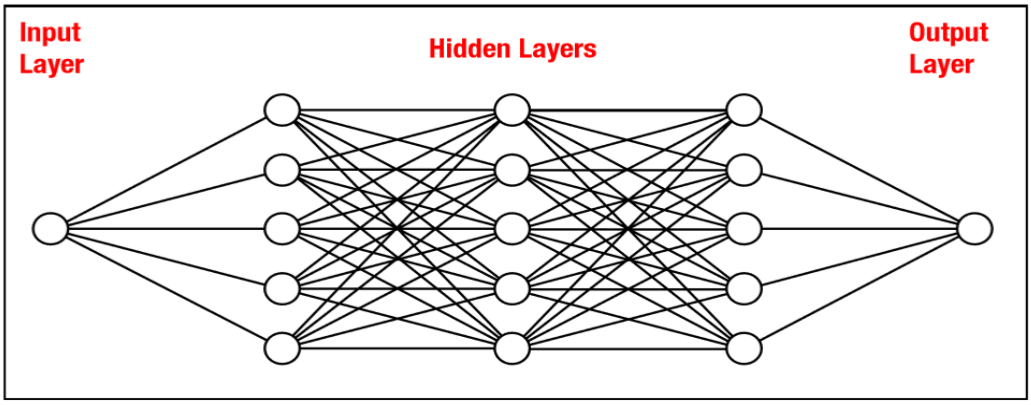
Tabel 16.4 menunjukkan kumpulan data pengujian yang dinormalisasi.

Tabel 16.4 Kumpulan Data Pengujian yang Dinormalisasi

Poin x	**y**
0.380530885	-0.5
0.420530885	-0.43971033
0.460530885	-0.37764165
0.500530885	-0.311798575
0.540530885	-0.23969458
0.580530885	-0.157931595
0.620530885	-0.06146605
0.660530885	0.057816175
0.700530885	0.214178745
0.740530885	0.43560867
0.780530885	0.78607581

16.2 ARSITEKTUR JARINGAN UNTUK CONTOH 3A

Gambar 16.3 menunjukkan arsitektur jaringan untuk contoh ini. Biasanya, arsitektur untuk proyek tertentu ditentukan secara eksperimental, dengan mencoba banyak dan memilih salah satu yang menghasilkan hasil aproksimasi terbaik. Jaringan terdiri dari satu neuron di lapisan input, tiga lapisan tersembunyi (masing-masing dengan lima neuron), dan satu neuron di lapisan output.



Gambar 16.3 Arsitektur jaringan

Kode Program untuk Contoh 3a

Daftar 16-1 menunjukkan kode program.

Daftar 16-1. Kode Program

```
// ============================================================
// Perkiraan fungsi periodik di luar rentang latihan.
//
// Input adalah file yang terdiri dari record dengan dua field:
// - Field pertama adalah nilai xPoint.
// - Kolom kedua adalah nilai fungsi target pada xPoint itu
// ============================================================
package sample3a;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date; import java.util.List;
import java.util.Locale; import java.util.Properties;
```

```
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample3a implements ExampleChart<XYChart>
{
        static double Nh =  1;
        static double Nl = -1;
        // First column
        static double maxXPointDh = 5.00;
        static double minXPointDl = -1.00;
        // Second column - target data
        static double maxTargetValueDh = 13.00;
        static double minTargetValueDl = 9.00;
        static double doublePointNumber = 0.00;
        static int intPointNumber = 0;
        static InputStream input = null;
        static double[] arrFunctionValue = new double[500];
        static double inputDiffValue = 0.00;
        static double predictDiffValue = 0.00;
        static double targetDiffValue = 0.00;
        static double valueDifferencePerc = 0.00;
        static String strFunctionValuesFileName;
        static int returnCode  = 0;
```

```
static int numberOfInputNeurons;
static int numberOfOutputNeurons;
static int numberOfRecordsInFile;
static int intNumberOfRecordsInTestFile;
static double realTargetValue;
static double realPredictValue;
static String functionValuesTrainFileName;
static String functionValuesTestFileName;
static String trainFileName;   static String priceFileName;
static String testFileName;
static String chartTrainFileName;
static String chartTestFileName;
static String networkFileName;
static int workingMode;
static String cvsSplitBy = ",";
static double denormTargetDiffPerc;
static double denormPredictDiffPerc;
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static XYChart Chart;
@Override
public XYChart getChart()
{
/ Create Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("x").yAxisTitle("y = f(x)").build();
// Customize Chart Chart.getStyler().setPlotBackgroundColor(ChartColor.
getAWTColor(ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font. PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
```

```
Chart.getStyler().setDecimalPattern("#0.00");
// Configuration
// Train
workingMode = 1;
trainFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample3a_Norm_ Tan_Train.csv";
unctionValuesTrainFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample3a_Tan_Calculate_
Train.csv";
chartTrainFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample3a_XYLine_Tan_Train_Cha
rt";
numberOfRecordsInFile = 12;
// Test the trained network at non-trained points
// workingMode = 2;
//    testFileName = "C:/My_Neural_Network_Book/Book_Examples/Sample3a_
Norm_Tan_Test.csv";
//                    functionValuesTestFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample3a_Tan_Calculate_
Test.csv";
//chartTestFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample3a_XYLine_Tan_Test_Char
t";
//numberOfRecordsInFile = 12;
//       Common configuration networkFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample3a_Saved_Tan_Network_
File.csv";
numberOfInputNeurons = 1;
numberOfOutputNeurons = 1;
try
{
        // Check the working mode to run
        if(workingMode == 1)
        {
        // Train mode
        loadFunctionValueTrainFileInMemory();
        File file1 = new File(chartTrainFileName);
        File file2 = new File(networkFileName);
        if(file1.exists())
        file1.delete();
        if(file2.exists())
        file2.delete();
        returnCode = 0;
        // Clear the return code variable
        do
        {
        returnCode = trainValidateSaveNetwork();
        } while (returnCode > 0);
```

```
                } // End the train logic
                else
                {
                // Testing mode.
                // Load testing file in memory          loadTestFileInMemory();
                File file1 = new File(chartTestFileName);
                if(file1.exists())
                file1.delete();
                loadAndTestNetwork();
                }
                }
                catch (Throwable t)
                {
                t.printStackTrace();
                System.exit(1);
                }
                finally
                {
                Encog.getInstance().shutdown();
                }
                   Encog.getInstance().shutdown();
                   return Chart;
                } // End of the method
// ============================
// Load CSV to memory.
// @return The loaded dataset.
// ============================
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal, boolean
headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// ====================================================
// The main method.
// @param Command line arguments. No arguments are used.
// ====================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample3a();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ====================================================
// Latih, validasi, dan simpan file jaringan yang terlatih
```

```
// =======================================================
static public int trainValidateSaveNetwork()
{
	double functionValue = 0.00;
	// Load the training CSV file in memory
	MLDataSet trainingSet =
	loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeuro
	ns,	true,CSVFormat.ENGLISH,false);
	// create a neural network	BasicNetwork network = new BasicNetwork();
	// Input layer	network.addLayer(new BasicLayer(null,true,1));
	// Hidden layer	network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
	network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
	network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
	// Output layer	network.addLayer(new BasicLayer(new
	ActivationTANH(),false,1));
	network.getStructure().finalizeStructure();
	network.reset();
	// train the neural network	final ResilientPropagation train = new
	ResilientPropagation(network, trainingSet);
	int epoch = 1;
	returnCode = 0;
	do
	{
		train.iteration();
		System.out.println("Epoch #" + epoch + " Error:" + train.getError());
		epoch++;
		if (epoch >= 500 && network.calculateError(trainingSet) > 0.000000061)
		{
		returnCode = 1;
		System.out.println("Try again");
		return returnCode;	}
	} while(train.getError() > 0.00000006);
	// Save the network file
	EncogDirectoryPersistence.saveObject(new
	File(networkFileName),network);
	System.out.println("Neural Network Results:");
	double sumDifferencePerc = 0.00;
	double averNormDifferencePerc = 0.00;
	double maxErrorPerc = 0.00;
	int m = -1;
	double xPoint_Initial = 0.00;
	double xPoint_Increment = 0.12;
	double xPoint = xPoint_Initial - xPoint_Increment;
	realTargetValue = 0.00;	realPredictValue = 0.00;
	for(MLDataPair pair: trainingSet)	{
		m++;
		xPoint = xPoint + xPoint_Increment;
		//if(xPoint >  3.14)
```

```
            //   break;
            final MLData output = network.compute(pair.getInput());
            MLData inputData = pair.getInput();
            MLData actualData = pair.getIdeal();
            MLData predictData = network.compute(inputData);
            // Calculate and print the results        inputDiffValue = inputData.getData(0);
            targetDiffValue = actualData.getData(0);
            predictDiffValue = predictData.getData(0);
            //De-normalize the values                denormTargetDiffPerc =
            ((minTargetValueDl - maxTargetValueDh)*  targetDiffValue -
            Nh*minTargetValueDl + maxTargetValueDh*Nl)/  (Nl - Nh);
            denormPredictDiffPerc =((minTargetValueDl - maxTargetValueDh)*
            predictDiffValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/  (Nl
            - Nh);
            valueDifferencePerc =                  Math.abs(((denormTargetDiffPerc -
            denormPredictDiffPerc)/ denormTargetDiffPerc)*100.00);
            System.out.println ("xPoint = " + xPoint + " realTargetValue = " +  d
            enormTargetDiffPerc + "  realPredictValue = " + denormPredictDiffPerc + "
            valueDifferencePerc = " + value  DifferencePerc);
            sumDifferencePerc = sumDifferencePerc + valueDifferencePerc;
            if (valueDifferencePerc > maxErrorPerc && m > 0)  maxErrorPerc =
            valueDifferencePerc;
            xData.add(xPoint);
            yData1.add(denormTargetDiffPerc);
            yData2.add(denormPredictDiffPerc);
    }  // End for pair loop
    XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
    XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLUE);
    series2.setMarkerColor(Color.ORANGE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.SOLID);
    try
    {
    //Save the chart image                   BitmapEncoder.saveBitmapWithDPI(Chart,
    chartTrainFileName, BitmapFormat.JPG, 100);
    System.out.println ("Train Chart file has been saved") ;
    }
    catch (IOException ex)
    {
    ex.printStackTrace();
    System.exit(3);
    }
    // Finally, save this trained network    EncogDirectoryPersistence.saveObject(new
    File(networkFileName),network);
    System.out.println ("Train Network has been saved") ;
    averNormDifferencePerc  = sumDifferencePerc/numberOfRecordsInFile;
    System.out.println(" ");
```

```
    System.out.println("maxErrorPerc = " + maxErrorPerc +       "  averNormDifferencePerc
    = " + averNormDifferencePerc);
    returnCode = 0;
    return returnCode;
}  // End of the method
// ============================================================
// This method load and test the trained network at the points not
// used for training.
// ============================================================
static public void loadAndTestNetwork()   {
        System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double sumDifferencePerc = 0.00;
        double maxErrorPerc = 0.00;
        double maxGlobalResultDiff = 0.00;
        double averErrorPerc = 0.00;
        double sumGlobalResultDiff = 0.00;
        double functionValue;
        BufferedReader br4;
        BasicNetwork network;
        int k1 = 0;
        // Process test records    maxGlobalResultDiff = 0.00;
        averErrorPerc = 0.00;
        sumGlobalResultDiff = 0.00;
        MLDataSet                                testingSet                                =
        loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutput
        Neurons,true,CSVFormat.ENGLISH,false);
        //    Load    the    saved    trained    network                network    =
        (BasicNetwork)EncogDirectoryPersistence.loadObject(new                File
        (networkFileName));
        int i = - 1; // Index of the current record    int m = -1;
        double xPoint_Initial = 3.141592654;    double xPoint_Increment = 0.12;
        double xPoint = xPoint_Initial - xPoint_Increment;
        realTargetValue = 0.00;
        realPredictValue = 0.00;
        for (MLDataPair pair:  testingSet)
        {
                m++;
                xPoint = xPoint + xPoint_Increment;
                //if(xPoint >  3.14)
                //   break;
                final MLData output = network.compute(pair.getInput());
                MLData inputData = pair.getInput();
                MLData actualData = pair.getIdeal();
                MLData predictData = network.compute(inputData);
                // Calculate and print the results        inputDiffValue = inputData.getData(0);
```

```
                targetDiffValue = actualData.getData(0);                predictDiffValue =
                predictData.getData(0);
                // De-normalize the values                              denormTargetDiffPerc =
                ((minTargetValueDl - maxTargetValueDh)* targetDiffValue -
                Nh*minTargetValueDl + maxTarget ValueDh*Nl)/(Nl - Nh);
                denormPredictDiffPerc =((minTargetValueDl - maxTargetValueDh)*
                predictDiffValue - Nh*minTargetValueDl + maxTargetValue Dh*Nl)/(Nl
                - Nh);
                valueDifferencePerc =                Math.abs(((denormTargetDiffPerc -
                denormPredictDiffPerc)/ denormTargetDiffPerc)*100.00);
                System.out.println ("xPoint = " + xPoint + " realTargetValue = " +          d
                enormTargetDiffPerc + "  realPredictValue = " + denormPredictDiffPerc + "
                valueDifferencePerc = " +              valueDifferencePerc);
                sumDifferencePerc = sumDifferencePerc + valueDifferencePerc;
                if (valueDifferencePerc > maxErrorPerc && m > 0)          maxErrorPerc =
                valueDifferencePerc;
                xData.add(xPoint);
                yData1.add(denormTargetDiffPerc);
                yData2.add(denormPredictDiffPerc);
    } // End for pair loop
        // Print max and average results
        System.out.println(" ");
    averErrorPerc = sumDifferencePerc/numberOfRecordsInFile;
    System.out.println("maxErrorPerc = " + maxErrorPerc);
    System.out.println("averErrorPerc = " + averErrorPerc);
    // All testing batch files have been processed          XYSeries series1 =
    Chart.addSeries("Actual", xData, yData1);
    XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLUE);
    series2.setMarkerColor(Color.ORANGE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image
    try
    {
    BitmapEncoder.saveBitmapWithDPI(Chart, chartTestFileName , BitmapFormat.JPG,
    100);
    }
    catch (Exception bt)
    {
    bt.printStackTrace();
    }
    System.out.println ("The Chart has been saved");
    } // End of the method
// ============================================
// Load Training Function Values file in memory
// ============================================
public static void loadFunctionValueTrainFileInMemory()
```

```
{
        BufferedReader br1 = null;
   String line = "";
        String cvsSplitBy = ",";
        double tempYFunctionValue = 0.00;
        try
        {
        br1 =  new BufferedReader(new FileReader(functionValuesTrain FileName));
   int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)        {          i++;          r++;
        // Skip the header line          if(i > 0)
        {
        // Break the line using comma as separator                    String[] workFields =
        line.split(cvsSplitBy);
        tempYFunctionValue = Double.parseDouble(workFields[1]);
        arrFunctionValue[r] = tempYFunctionValue;
        }
      } // end of the while loop
       br1.close();
     }
        catch (IOException ex)
        {       ex.printStackTrace();
                System.err.println("Error opening files = " + ex);
                System.exit(1);
        }
}
// ========================================
// Load testing Function Values file in memory
// ========================================
public static void loadTestFileInMemory()
{
BufferedReader br1 = null;
        String line = "";
        String cvsSplitBy = ",";
        double tempYFunctionValue = 0.00;
        try
        {
        br1 = new BufferedReader(new FileReader(functionValuesTestFileName));
        int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)
        {
                i++;
                r++;
        // Skip the header line
        if(i > 0)
        {
```

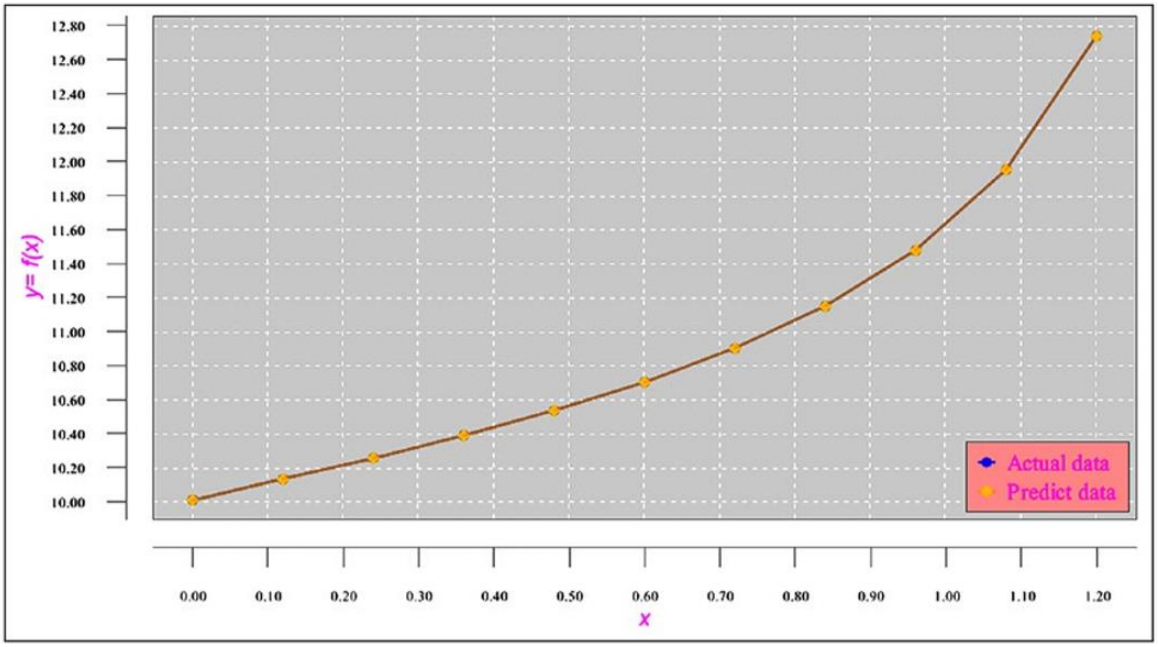
```
                // Break the line using comma as separator        String[] workFields =
                line.split(cvsSplitBy);
                tempYFunctionValue          =          Double.parseDouble(workFields[1]);
                arrFunctionValue[r] = tempYFunctionValue;
            }
            } // end of the while loop
            br1.close();
        }
        catch (IOException ex)
        {
        ex.printStackTrace();
        System.err.println("Error opening files = " + ex);
        System.exit(1);
        }
    }
} // End of the class
```

Kode ini mewakili pemrosesan jaringan saraf biasa dan tidak memerlukan penjelasan apa pun. Daftar 16-2 menunjukkan hasil pemrosesan pelatihan.

Daftar 16-2. Hasil Pemrosesan Pelatihan

```
xPoint = 0.00 TargetValue = 10.00000  PredictedValue = 10.00027 DiffPerc = 0.00274
xPoint = 0.12 TargetValue = 10.12058  PredictedValue = 10.12024 DiffPerc = 0.00336
xPoint = 0.24 TargetValue = 10.24471  PredictedValue = 10.24412 DiffPerc = 0.00580
xPoint = 0.36 TargetValue = 10.37640  PredictedValue = 10.37629 DiffPerc = 0.00102
xPoint = 0.48 TargetValue = 10.52061  PredictedValue = 10.52129 DiffPerc = 0.00651
xPoint = 0.60 TargetValue = 10.68414  PredictedValue = 10.68470 DiffPerc = 0.00530
xPoint = 0.72 TargetValue = 10.87707  PredictedValue = 10.87656 DiffPerc = 0.00467
xPoint = 0.84 TargetValue = 11.11563  PredictedValue = 11.11586 DiffPerc = 0.00209
xPoint = 0.96 TargetValue = 11.42835  PredictedValue = 11.42754 DiffPerc = 0.00712
xPoint = 1.08 TargetValue = 11.87121  PredictedValue = 11.87134 DiffPerc = 0.00104
xPoint = 1.20 TargetValue = 12.57215  PredictedValue = 12.57200 DiffPerc = 0.00119
maxErrorPerc = 0.007121086942321541
averErrorPerc = 0.0034047471040211954
```

Gambar 16.4 menunjukkan grafik hasil pelatihan.



Gambar 16.4 Bagan hasil latihan pada interval [0, 1.2]

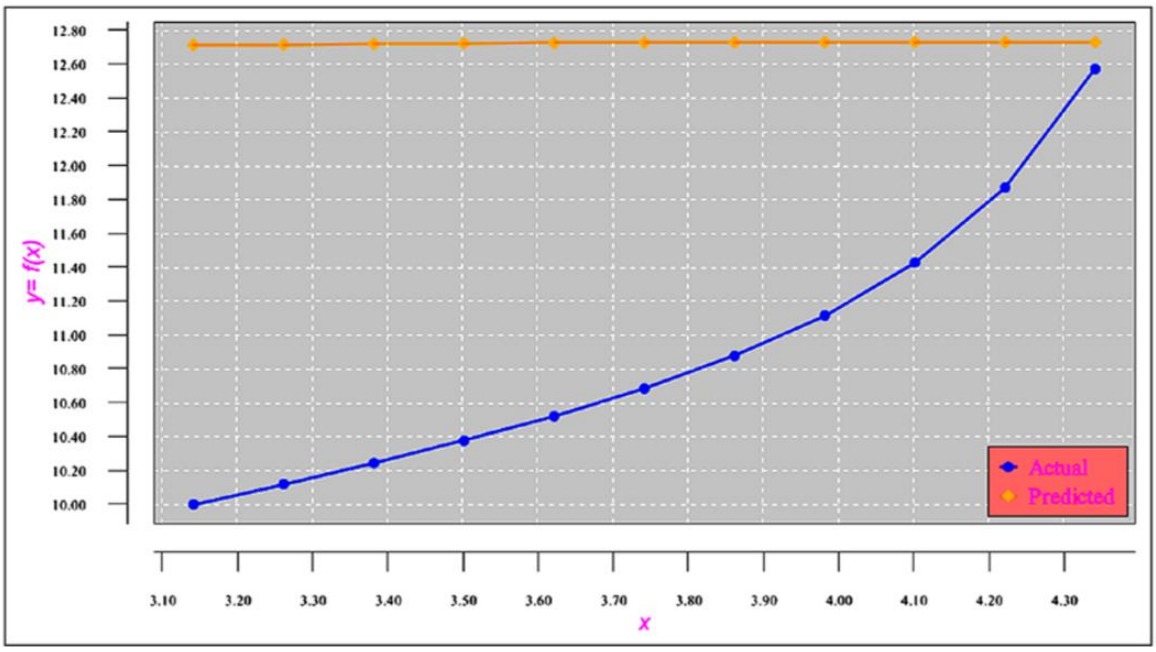
Menguji Jaringan

Saat memproses kumpulan data pengujian, Anda mengekstrak nilai xPoint (kolom 1) dari catatan, memasukkan nilai ini ke jaringan terlatih, memperoleh dari jaringan nilai fungsi yang diprediksi, dan membandingkan hasilnya dengan nilai fungsi yang kebetulan Anda ketahui (lihat Daftar 16-2, kolom 2). Daftar 16-3 menunjukkan hasil pemrosesan pengujian.

Daftar 16-3. Hasil Pemrosesan Tes

xPoint = 3.141594 TargetValue = 10.00000 PredictedValue = 12.71432 DiffPerc = 27.14318
xPoint = 3.261593 TargetValue = 10.12059 PredictedValue = 12.71777 DiffPerc = 25.66249
xPoint = 3.381593 TargetValue = 10.24471 PredictedValue = 12.72100 DiffPerc = 24.17133
xPoint = 3.501593 TargetValue = 10.37640 PredictedValue = 12.72392 DiffPerc = 22.62360
xPoint = 3.621593 TargetValue = 10.52061 PredictedValue = 12.72644 DiffPerc = 20.96674
xPoint = 3.741593 TargetValue = 10.68413 PredictedValue = 12.72849 DiffPerc = 19.13451
xPoint = 3.861593 TargetValue = 10.87706 PredictedValue = 12.73003 DiffPerc = 17.03549
xPoint = 3.981593 TargetValue = 11.11563 PredictedValue = 12.73102 DiffPerc = 14.53260
xPoint = 4.101593 TargetValue = 11.42835 PredictedValue = 12.73147 DiffPerc = 11.40249
xPoint = 4.221593 TargetValue = 11.87121 PredictedValue = 12.73141 DiffPerc = 7.246064
xPoint = 4.341593 TargetValue = 12.57215 PredictedValue = 12.73088 DiffPerc = 1.262565
maxErrorPerc = 25.662489243649677
averErrorPerc = 15.931756451553364

Gambar 16.5 menunjukkan grafik hasil pengolahan pengujian pada interval [3.141592654, 4.341592654].



Gambar 16.5 Bagan hasil pengujian pada interval [3.141592654, 4.341592654]

Perhatikan betapa berbedanya grafik yang diprediksi (garis atas) dibandingkan dengan grafik sebenarnya (garis lengkung). Kesalahan besar dari hasil pemrosesan pengujian (maxErrorPerc = 25,66% dan averErrorPerc > 15,93%), seperti yang ditunjukkan pada Daftar 16-7, dan bagan pada Gambar 16.5 menunjukkan bahwa pendekatan fungsi tersebut tidak berguna. Jaringan mengembalikan nilai-nilai tersebut ketika diumpankan nilai xPoints input dari catatan pengujian yang berada di luar rentang pelatihan. Dengan mengirimkan xPoints tersebut ke jaringan, Anda dapat mencoba memperkirakan nilai fungsi daripada memperkirakannya.

16.3 CONTOH 3B: MENAKSIR FUNGSI PERIODIK DI LUAR RENTANG PELATIHAN

Dalam contoh ini, Anda akan melihat bagaimana (dengan persiapan data khusus) memungkinkan fungsi periodik didekati dengan benar di luar rentang pelatihan jaringan. Seperti yang akan Anda lihat nanti, Anda juga dapat menggunakan teknik ini untuk fungsi periodik yang lebih kompleks dan bahkan beberapa fungsi nonperiodik.

Mempersiapkan Data Pelatihan

Sebagai pengingat, contoh ini perlu menggunakan jaringan yang dilatih pada interval [0, 1.2] untuk memprediksi hasil fungsi pada interval [3.141592654 – 4.341592654], yang berada di luar rentang latihan. Anda akan melihat di sini bagaimana menghindari pembatasan jaringan saraf ini untuk fungsi periodik. Untuk melakukan ini, pertama-tama Anda akan mengubah nilai fungsi yang diberikan ke kumpulan data dengan setiap record yang terdiri dari dua bidang.

- Kolom 1 adalah perbedaan antara nilai xPoint dari titik saat ini (rekam) dan titik pertama (rekam).
- Bidang 2 adalah perbedaan antara nilai fungsi pada titik berikutnya (rekam) dan titik saat ini (rekam).

Tips

saat menyatakan bidang pertama dari catatan sebagai perbedaan antara nilai xpoint dan bukan hanya nilai xpoint asli, Anda tidak lagi keluar dari interval latihan bahkan ketika Anda mencoba memprediksi nilai fungsi untuk interval berikutnya (dalam hal ini, [3.141592654 –

4.341592654]). dengan kata lain, selisih nilai xpoint pada interval berikutnya, yaitu [3.141592654 – 4.341592654], menjadi dalam rentang latihan.

Dengan menyusun kumpulan data input sedemikian rupa, pada dasarnya Anda mengajarkan jaringan untuk mempelajari bahwa ketika perbedaan nilai fungsi antara xPoints saat ini dan pertama sama dengan beberapa nilai "a", maka perbedaan nilai fungsi antara xPoints berikutnya dan titik saat ini harus sama dengan beberapa nilai "b." Itu memungkinkan jaringan untuk memprediksi nilai fungsi hari berikutnya dengan mengetahui nilai fungsi hari ini. Tabel 16.5 menunjukkan kumpulan data transformasi.

Tabel 16.5 Kumpulan Data Pelatihan yang Diubah

Point x	**y**
-0.12	9.879420663
0	10
0.12	10.12057934
0.24	10.2447167
0.36	10.37640285
0.48	10.52061084
0.6	10.68413681
0.72	10.8770679
0.84	11.11563235
0.96	11.42835749
1.08	11.87121734
1.2	12.57215162
1.32	13.90334779

Anda menormalkan set data pelatihan pada interval [-1,1]. Tabel 16.6 menunjukkan hasilnya.

Tabel 16.6 Kumpulan Data Pelatihan yang Dinormalisasi

xDiff	**fDiff**
- 0.968	- 0.967073056
- 0.776	- 0.961380224
- 0.584	- 0.94930216
- 0.392	- 0.929267216
- 0.2	- 0.898358448
- 0.008	- 0.851310256
0.184	0.77829688
0.376	0.659639776
0.568	- 0.45142424
0.76	- 0.038505152
0.952	0.969913872

Tabel 16.7 menunjukkan kumpulan data pengujian yang diubah.

Tabel 16.7 Kumpulan Data Pengujian yang Diubah

xPointDiff	yDiff
3.021592654	9.879420663
3.141592654	10
3.261592654	10.12057934
3.381592654	10.2447167
3.501592654	10.37640285
3.621592654	10.52061084
3.741592654	10.68413681
3.861592654	10.8770679
3.981592654	11.11563235
4.101592654	11.42835749
4.221592654	11.87121734
4.341592654	12.57215163
4.461592654	13.90334779

Tabel 16.8 menunjukkan kumpulan data pengujian yang dinormalisasi.

Tabel 16.8 Kumpulan Data Pengujian yang Dinormalisasi

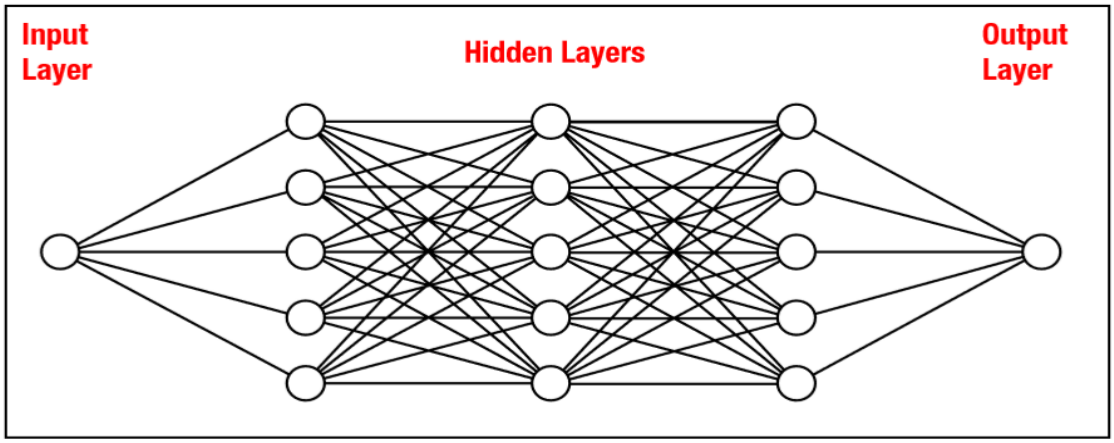
xDiff	fDiff
- 0.968	- 0.967073056
- 0.776	- 0.961380224
- 0.584	- 0.94930216
- 0.392	- 0.929267216
- 0.2	- 0.898358448
- 0.008	- 0.851310256
0.184	0.77829688
0.376	0.659639776
0.568	- 0.45142424
0.76	- 0.038505152
0.952	0.969913872

Anda sebenarnya tidak memerlukan kolom kedua dalam kumpulan data pengujian untuk diproses. Saya baru saja memasukkannya ke dalam kumpulan data pengujian untuk dapat membandingkan nilai yang diprediksi dengan nilai aktual secara terprogram. Mengumpankan perbedaan antara nilai xPoint pada titik saat ini dan sebelumnya (Bidang 1 dari catatan yang saat ini diproses) ke jaringan yang dilatih, Anda akan mendapatkan kembali perbedaan yang diprediksi antara nilai fungsi pada titik berikutnya dan titik saat ini. Oleh karena itu, nilai prediksi fungsi pada titik berikutnya sama dengan jumlah nilai fungsi target pada titik saat ini (rekam) dan nilai selisih prediksi jaringan.

16.4 ARSITEKTUR JARINGAN UNTUK CONTOH 3B

Untuk contoh ini, Anda akan menggunakan jaringan dengan lapisan output yang terdiri dari satu neuron, tiga lapisan tersembunyi (masing-masing menampung lima

neuron), dan lapisan output yang menampung satu neuron. Sekali lagi, saya datang dengan arsitektur ini secara eksperimental (dengan mencoba dan menguji). Gambar 16.6 menunjukkan arsitektur pelatihan.



Gambar 16.6 Arsitektur jaringan untuk contoh

Sekarang Anda siap untuk mengembangkan program pemrosesan jaringan dan menjalankan metode pelatihan dan pengujian.

Kode Program untuk Contoh 3b

Daftar 16-4 menunjukkan kode program.

Daftar 16-4. Kode Program

```
// ============================================================
// pproksimasi fungsi periodik di luar rentang latihan.
//
// Input adalah file yang terdiri dari record dengan dua field:
// - Field pertama menyimpan perbedaan antara nilai fungsi dari
// record saat ini dan record pertama.
// - Kolom kedua menyimpan perbedaan antara nilai fungsi dari
// record selanjutnya dan saat ini.
// ============================================================
package sample3b;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
```

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
/**
*
* @author i262666
*/
public class Sample3b implements ExampleChart<XYChart>
{
        static double Nh =  1;
        static double Nl = -1;
```

```
// First column   static double maxXPointDh = 1.35;
static double minXPointDl = 0.10;
// Second column - target data
static double maxTargetValueDh = 1.35;
static double minTargetValueDl = 0.10;
static double doublePointNumber = 0.00;
static int intPointNumber = 0;
static InputStream input = null;
static double[] arrFunctionValue = new double[500];
static double inputDiffValue = 0.00;
static double predictDiffValue = 0.00;
static double targetDiffValue = 0.00;
static double valueDifferencePerc = 0.00;
static String strFunctionValuesFileName;
static int returnCode  = 0;
static int numberOfInputNeurons;
static int numberOfOutputNeurons;
static int numberOfRecordsInFile;
static int intNumberOfRecordsInTestFile;
static double realTargetValue;
static double realPredictValue;
 static String functionValuesTrainFileName;
static String functionValuesTestFileName;
static String trainFileName;   static String priceFileName;
static String testFileName;
static String chartTrainFileName;
static String chartTestFileName;
static String networkFileName;
static int workingMode;   static String cvsSplitBy = ",";
static double denormTargetDiffPerc;
static double denormPredictDiffPerc;
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static XYChart Chart;
@Override
public XYChart getChart()
{
// Create Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("x").yAxisTitle("y=  f(x)").build();
// Customize Chart
Chart.getStyler().setPlotBackgroundColor(ChartColor.
getAWTColor(ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setChartFontColor(Color.MAGENTA);
```

```
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font. ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font. PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
// Configuration
// Set the mode of program run    workingMode = 1;
// Training mode
if (workingMode ==  1)
{
trainFileName     =     "C:/My_Neural_Network_Book/Book_Examples/Sample3b_
Norm_Tan_Train.csv";
functionValuesTrainFileName                                                    =
"C:/My_Neural_Network_Book/Book_Examples/Sample3b_Tan_Calculate_
Train.csv";
chartTrainFileName                                                             =
"C:/My_Neural_Network_Book/Book_Examples/Sample3b_XYLine_Tan_
Train_Chart";
numberOfRecordsInFile = 12;
}
else
{
//    Testing        mode                                          testFileName    =
"C:/My_Neural_Network_Book/Book_Examples/ Sample3b_Norm_Tan_Test.csv";
functionValuesTestFileName                                                     =
"C:/My_Neural_Network_Book/Book_Examples/Sample3b_Tan_Calculate_
Test.csv";
chartTestFileName                                                              =
"C:/My_Neural_Network_Book/Book_Examples/Sample3b_XYLine_Tan_
Test_Chart";
numberOfRecordsInFile = 12;      }
//        Common        configuration                    networkFileName        =
"C:/My_Neural_Network_Book/Book_Examples/Sample3b_Saved_Tan_
Network_File.csv";
numberOfInputNeurons = 1;
numberOfOutputNeurons = 1;
try
```

```
        {
        // Check the working mode to run
        if(workingMode == 1)
        {
        // Train mode
        loadFunctionValueTrainFileInMemory();
        File file1 = new File(chartTrainFileName);
        File file2 = new File(networkFileName);
        if(file1.exists())
        file1.delete();
        if(file2.exists())
        file2.delete();
        returnCode = 0;
        // Clear the return code variable
        do
        {
        returnCode = trainValidateSaveNetwork();
        }
        while (returnCode > 0);
        }  // End the train logic
        else
        {
        // Testing mode.
        // Load testing file in memory          loadTestFileInMemory();
        File file1 = new File(chartTestFileName);
        if(file1.exists())
        file1.delete();
        loadAndTestNetwork();
        }
        }
        catch (Throwable t)
        {
        t.printStackTrace();
        System.exit(1);
        }
        finally
        {
        Encog.getInstance().shutdown();
        }
        Encog.getInstance().shutdown();
        return Chart;
} // End of the method
// =======================================================
// Load CSV to memory.
// @return The loaded dataset.
// =======================================================
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal, boolean headers, CSVFormat format, boolean significance)
```

```
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
        }
// =============================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// =============================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample3b();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// =============================================================
// Latih, validasi, dan simpan file jaringan yang terlatih
// =============================================================
static public int trainValidateSaveNetwork()
{
        double functionValue = 0.00;
        // Load the training CSV file in memory          MLDataSet trainingSet =
        loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeuro
        ns,       true,CSVFormat.ENGLISH,false);
        // create a neural network
        BasicNetwork network = new BasicNetwork();
        // Input layer
        network.addLayer(new BasicLayer(null,true,1));
        // Hidden layer
        network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));
        network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));
        network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));
        //    Output    layer                network.addLayer(new    BasicLayer(new
        ActivationTANH(),false,1));
        network.getStructure().finalizeStructure();    network.reset();
        // train the neural network
        final ResilientPropagation train = new ResilientPropagation(network, trainingSet);
        int epoch = 1;
        returnCode = 0;
        do
        {
                train.iteration();
                System.out.println("Epoch #" + epoch + " Error:" + train.getError());
```

```
            epoch++;
            if (epoch >= 500 && network.calculateError(trainingSet) > 0.000000061)
            {
            returnCode = 1;
            System.out.println("Try again");
            return returnCode;
            }
      } while(train.getError() > 0.00000006);
      // Save the network file
      EncogDirectoryPersistence.saveObject(new File(networkFileName),network);
      System.out.println("Neural Network Results:");
      double sumDifferencePerc = 0.00;
      double averNormDifferencePerc = 0.00;
      double maxErrorPerc = 0.00;
      int m = -1;
      double xPoint_Initial = 0.00;
      double xPoint_Increment = 0.12;
      //double xPoint = xPoint_Initial - xPoint_Increment;    double xPoint = xPoint_Initial;
      realTargetValue = 0.00;
      realPredictValue = 0.00;
      for(MLDataPair pair: trainingSet)
      {
            m++;
            xPoint = xPoint + xPoint_Increment;
            final MLData output = network.compute(pair.getInput());
            MLData inputData = pair.getInput();
            MLData actualData = pair.getIdeal();
            MLData predictData = network.compute(inputData);
            // Calculate and print the results
            inputDiffValue = inputData.getData(0);
            targetDiffValue = actualData.getData(0);
            predictDiffValue = predictData.getData(0);
            // De-normalize the values          denormTargetDiffPerc = ((minXPointDl -
            maxXPointDh)*targetDiffValue      -      Nh*minXPointDl      +
            maxXPointDh*Nl)/(Nl  -  Nh);                  denormPredictDiffPerc
            =((minTargetValueDl    -    maxTargetValueDh)*    predictDiffValue    -
            Nh*minTargetValueDl + maxTarget        ValueDh*Nl)/(Nl - Nh);
            functionValue = arrFunctionValue[m+1];
            realTargetValue = functionValue + denormTargetDiffPerc;
            realPredictValue = functionValue + denormPredictDiffPerc;
            valueDifferencePerc  =                    Math.abs(((realTargetValue  -
            realPredictValue)/ realPredictValue)*100.00);
            System.out.println ("xPoint = " + xPoint + "  realTargetValue = " +          d
            enormTargetDiffPerc + "  realPredictValue = " + denormPredictDiffPerc + "
            valueDifferencePerc = " +  value          DifferencePerc);
            sumDifferencePerc = sumDifferencePerc + valueDifferencePerc;
            if (valueDifferencePerc > maxErrorPerc && m > 0)          maxErrorPerc =
            valueDifferencePerc;
```

```
                xData.add(xPoint);
                yData1.add(denormTargetDiffPerc);
                yData2.add(denormPredictDiffPerc);
        }  // End for pair loop
        XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        try
        {
        //Save the chart image BitmapEncoder.saveBitmapWithDPI(Chart,
        chartTrainFileName, BitmapFormat.JPG, 100);
        System.out.println ("Train Chart file has been saved") ;
        }
        catch (IOException ex)
        {
        ex.printStackTrace();
        System.exit(3);
        }
        // Finally, save this trained network EncogDirectoryPersistence.saveObject(new
        File(networkFileName),network);
        System.out.println ("Train Network has been saved") ;
        averNormDifferencePerc = sumDifferencePerc/numberOfRecordsInFile;
        System.out.println(" ");
        System.out.println("maxErrorPerc = " + maxErrorPerc + "
        averNormDifferencePerc = " + averNormDifferencePerc);
        returnCode = 0;
        return returnCode;
}  // End of the method
// ============================================================
// Metode ini memuat dan menguji jaringan yang dilatih pada titik-titik yang tidak
// digunakan untuk pelatihan.
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double sumDifferencePerc = 0.00;
        double maxErrorPerc = 0.00;
        double maxGlobalResultDiff = 0.00;
        double averErrorPerc = 0.00;
        double sumGlobalResultDiff = 0.00;
        double functionValue;
        BufferedReader br4;
```

```
BasicNetwork network;
int k1 = 0;
// Process test records
maxGlobalResultDiff = 0.00;
averErrorPerc = 0.00;
sumGlobalResultDiff = 0.00;
MLDataSet testingSet =
loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutput
Neurons,true,CSVFormat.ENGLISH,false);
int i = - 1; // Index of the current record
int m = -1;
double xPoint_Initial = 3.141592654;
double xPoint_Increment = 0.12;
double xPoint = xPoint_Initial;
realTargetValue = 0.00;
realPredictValue = 0.00;
for (MLDataPair pair:  testingSet)
{
        m++;
        xPoint = xPoint + xPoint_Increment;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results      inputDiffValue = inputData.getData(0);
        targetDiffValue = actualData.getData(0);
        predictDiffValue = predictData.getData(0);
        // De-normalize the values      denormTargetDiffPerc = ((minXPointDl -
        maxXPointDh)*targetDiffValue - Nh*minXPointDl +
        maxXPointDh*Nl)/(Nl - Nh);
        denormPredictDiffPerc =((minTargetValueDl - maxTargetValueDh)
        *predictDiffValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/ (Nl
        - Nh);
        functionValue = arrFunctionValue[m+1];
        realTargetValue = functionValue + denormTargetDiffPerc;
        realPredictValue = functionValue + denormPredictDiffPerc;
        valueDifferencePerc =                Math.abs(((realTargetValue -
        realPredictValue)/realPredictValue)*100.00);
        System.out.println ("xPoint = " + xPoint + "  realTargetValue = " +
        realTargetValue + "  realPredictValue = " + realPredictValue + "
        valueDifferencePerc = " + valueDifferencePerc);
        sumDifferencePerc = sumDifferencePerc + valueDifferencePerc;
        if (valueDifferencePerc > maxErrorPerc && m > 0)      maxErrorPerc =
        valueDifferencePerc;
        xData.add(xPoint);
        yData1.add(realTargetValue);
        yData2.add(realPredictValue);
} // End for pair loop
```

```
                // Print max and average results
                System.out.println(" ");
                averErrorPerc = sumDifferencePerc/numberOfRecordsInFile;
                System.out.println("maxErrorPerc = " + maxErrorPerc);
                System.out.println("averErrorPerc = " + averErrorPerc);
                // All testing batch files have been processed        XYSeries series1 =
                Chart.addSeries("Actual", xData, yData1);
                XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
                series1.setLineColor(XChartSeriesColors.BLUE);
                series2.setMarkerColor(Color.ORANGE);
                series1.setLineStyle(SeriesLines.SOLID);
                series2.setLineStyle(SeriesLines.SOLID);
                // Save the chart image     try
                {
                BitmapEncoder.saveBitmapWithDPI(Chart,         chartTestFileName         ,
                BitmapFormat.JPG, 100);
                }
                catch (Exception bt)
                {
                bt.printStackTrace();
                }
                System.out.println ("The Chart has been saved");
} // End of the method
// ============================================================
// Muat file Nilai Fungsi Pelatihan di memori
// ============================================================
public static void loadFunctionValueTrainFileInMemory()
{
        BufferedReader br1 = null;
        String line = "";     String cvsSplitBy = ",";
        double tempYFunctionValue = 0.00;
        try
        {
        br1 = new BufferedReader(new FileReader(functionValuesTrainFileName));
        int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)
        {
        i++;
        r++;
        // Skip the header line          if(i > 0)
        {
        // Break the line using comma as separator                    String[] workFields =
        line.split(cvsSplitBy);
        tempYFunctionValue = Double.parseDouble(workFields[1]);
        arrFunctionValue[r] = tempYFunctionValue;          }
        } // end of the while loop
        br1.close();
```

```
        }
        catch (IOException ex)
        {
        ex.printStackTrace();
        System.err.println("Error opening files = " + ex);
        System.exit(1);
    }
}
// ==========================================
// Memuat file Nilai Fungsi pengujian di memori
// ==========================================
public static void loadTestFileInMemory()
{
        BufferedReader br1 = null;
        String line = "";
        String cvsSplitBy = ",";
        double tempYFunctionValue = 0.00;
        try
        {
        br1 = new BufferedReader(new FileReader(functionValuesTestFileName));
        int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)         {
        i++;
        r++;
        // Skip the header line
        if(i > 0)
        {
        // Break the line using comma as separator                String[] workFields =
        line.split(cvsSplitBy);
        tempYFunctionValue = Double.parseDouble(workFields[1]);
        arrFunctionValue[r] = tempYFunctionValue;
        }
    }  // end of the while loop
    br1.close();
    }
    catch (IOException ex)
    {
    ex.printStackTrace();
    System.err.println("Error opening files = " + ex);
    System.exit(1);
    }
    }
} // End of the class
```

Seperti biasa, beberapa pernyataan lain-lain hadir di bagian atas program. Mereka diperlukan oleh paket XChart. Program dimulai dengan beberapa inisialisasi impor dan kode yang diperlukan oleh XCharts (lihat Daftar 16-5).

Daftar 16-5. Memanggil Metode Pelatihan secara Berulang

```
returnCode = 0;
// Clear the error Code
do
        {
                returnCode = trainValidateSaveNetwork();
        } while (returnCode > 0);
```

Logika ini memanggil metode pelatihan dan kemudian memeriksa nilai returnCode. Jika bidang returnCode tidak nol, metode pelatihan dipanggil lagi dalam satu lingkaran. Setiap kali metode dipanggil, parameter bobot/bias awal diberi nilai acak yang berbeda, yang membantu dalam memilih nilai terbaiknya saat metode berulang kali dipanggil dalam satu lingkaran.

Di dalam metode yang dipanggil, logika memeriksa nilai kesalahan setelah 500 iterasi. Jika kesalahan yang dihitung jaringan masih lebih besar dari batas kesalahan, metode akan keluar dengan nilai returnCode 1. Dan, seperti yang baru saja Anda lihat, metode akan dipanggil lagi. Akhirnya, ketika kesalahan yang dihitung menghapus batas kesalahan, metode keluar dengan nilai kode balik 0 dan tidak lagi dipanggil. Anda memilih nilai batas kesalahan secara eksperimental, sehingga menyulitkan jaringan untuk menghapus batas kode kesalahan tetapi tetap memastikan bahwa setelah iterasi yang cukup kesalahan akan melewati batas kesalahan.

Fragmen kode berikutnya (Daftar 16-6) menunjukkan awal dari metode pelatihan. Ini memuat kumpulan data pelatihan dalam memori dan menciptakan jaringan saraf yang terdiri dari lapisan input (dengan satu neuron), tiga lapisan tersembunyi (masing-masing dengan lima neuron), dan lapisan output (dengan satu neuron). Kemudian, Anda melatih jaringan menggunakan nilai ResilientPropagation yang paling efisien sebagai metode backpropagation.

Daftar 16-6. Memuat Kumpulan Data Pelatihan dan Membangun dan Melatih Jaringan

```
// Load the training CSV file in memory
MLDataSet trainingSet =
loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
// create a neural network BasicNetwork network = new BasicNetwork();
// Input layer
network.addLayer(new BasicLayer(null,true,1));
// Hidden layer (seven hidden layers are created
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
// Output layer
network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// train the neural network final ResilientPropagation train = new
ResilientPropagation(network, trainingSet);
```

Itu diikuti oleh fragmen yang melatih jaringan. Anda melatih jaringan dengan mengulang zaman. Pada setiap iterasi Anda memeriksa apakah kesalahan yang dihitung

kurang dari batas kesalahan yang ditetapkan (dalam hal ini, 0,00000006). Ketika kesalahan jaringan menjadi kurang dari batas kesalahan, Anda keluar dari loop. Jaringan dilatih dengan presisi yang diperlukan, sehingga Anda menyimpan jaringan terlatih pada disk.

```
int epoch = 1;
returnCode = 0;
do
{
        train.iteration();
        System.out.println("Epoch #" + epoch + " Error:" + train.getError());
        epoch++;
        if (epoch >= 10000 && network.calculateError(trainingSet) > 0.000000061)
                {
                        returnCode = 1;
                        System.out.println("Try again");
                        return returnCode;
                        }
                } while(train.getError()0.00000006);
        // Save the network file
        EncogDirectoryPersistence.saveObject(new File(networkFileName),network);
```

Perhatikan logika yang ditunjukkan pada Daftar 16-7 yang memeriksa apakah kesalahan jaringan menjadi kurang dari batas kesalahan.

Daftar 16-7. Memeriksa Kesalahan Jaringan

```
if (epoch >= 10000 && network.calculateError(trainingSet) > 0.00000006)
        {
                returnCode = 1;
                System.out.println("Try again");
                return returnCode;
        }
```

Kode ini memeriksa apakah setelah 500 iterasi kesalahan jaringan masih tidak kurang dari batas kesalahan. Jika demikian, nilai returnCode disetel ke 1, dan Anda keluar dari metode pelatihan, kembali ke titik di mana metode pelatihan dipanggil dalam satu lingkaran. Di sana, ia akan memanggil metode pelatihan lagi dengan serangkaian parameter bobot/bias acak yang baru. Tanpa kode itu, pengulangan akan berlanjut tanpa batas jika kesalahan jaringan yang dihitung tidak dapat menghapus batas kesalahan dengan set parameter bobot/bias awal yang dipilih secara acak. Ada dua API yang dapat memeriksa kesalahan jaringan yang dihitung. Hasilnya sedikit berbeda tergantung pada metode mana yang digunakan.

- train.getError(): Kesalahan dihitung sebelum pelatihan diterapkan.
- network.CalculateError(): Kesalahan dihitung setelah pelatihan diterapkan.

Fragmen kode berikutnya (ditunjukkan dalam Daftar 16-8) berulang di atas kumpulan data pasangan. xPoint dalam loop diatur pada interval [0, 1.2]. Untuk setiap record, ia mengambil nilai input, aktual, dan prediksi; mendenormalisasi mereka; dan dengan membuat nilai fungsi menghitung nilai realTargetValue dan realPredictValue, menambahkannya ke data bagan (bersama dengan nilai xPoint yang sesuai). Ini juga menghitung persentase perbedaan nilai maksimum dan rata-rata untuk semua catatan. Semua data ini dicetak sebagai log pelatihan. Akhirnya, jaringan terlatih dan file gambar

bagan disimpan di disk. Perhatikan bahwa kode pengembalian diatur ke nol pada saat itu, sebelum Anda kembali dari metode pelatihan, sehingga metode tersebut tidak akan dipanggil lagi.

Daftar 16-8. Mengulangi Kumpulan Data Pasangan

```
for(MLDataPair pair: trainingSet)
{
        m++;
        xPoint = xPoint + xPoint_Increment;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results
        inputDiffValue = inputData.getData(0);
        targetDiffValue = actualData.getData(0);
        predictDiffValue = predictData.getData(0);
        // De-normalize the values
        denormTargetDiffPerc = ((minXPointDl - maxXPointDh)*targetDiffValue -
        Nh*minXPointDl + maxXPointDh*Nl)/(Nl - Nh);
        denormPredictDiffPerc =((minTargetValueDl - maxTargetValueDh)
        *predictDiffValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/ (Nl - Nh);
        functionValue = arrFunctionValue[m];
        realTargetValue = functionValue + targetDiffValue;
        realPredictValue = functionValue + predictDiffValue;
        valueDifferencePerc = M ath.abs(((realTargetValue - realPredictValue)/
        realPredictValue)*100.00);
        System.out.println ("xPoint = " + xPoint + " realTargetValue = " +
        realTargetValue + "  realPredictValue = " + realPredictValue);
        sumDifferencePerc = sumDifferencePerc + valueDifferencePerc;
        if (valueDifferencePerc > maxDifferencePerc) maxDifferencePerc =
        valueDifferencePerc;
        xData.add(xPoint);
        yData1.add(realTargetValue);
        yData2.add(realPredictValue);
} // End for pair loop
        XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        try
        {
                //Save the chart image
                BitmapEncoder.saveBitmapWithDPI(Chart, chartTrainFileName,
                BitmapFormat.JPG, 100);
                System.out.println ("Train Chart file has been saved") ;
        }
```

```
        catch (IOException ex)
        {
                ex.printStackTrace();
                System.exit(3);
        }
        // Finally, save this trained network    EncogDirectoryPersistence.saveObject(new
        File(networkFileName),network);
        System.out.println ("Train Network has been saved") ;
        averNormDifferencePerc  = sumDifferencePerc/numberOfRecordsInFile;
        System.out.println(" ");
        System.out.println("maxDifferencePerc = " + maxDifferencePerc + "
        averNormDifferencePerc = " + averNormDifferencePerc);
        returnCode = 0;
        return returnCode;
}  // End of the method
```

Metode pengujian memiliki logika pemrosesan yang serupa dengan pengecualian membangun dan melatih jaringan. Alih-alih membangun dan melatih jaringan, ini memuat jaringan terlatih yang disimpan sebelumnya ke dalam memori. Itu juga memuat kumpulan data uji dalam memori. Dengan mengulang set data pasangan, ia mendapatkan nilai input, target, dan prediksi untuk setiap record. Nilai xPoint dalam loop diambil dari interval [3.141592654, 4.341592654].

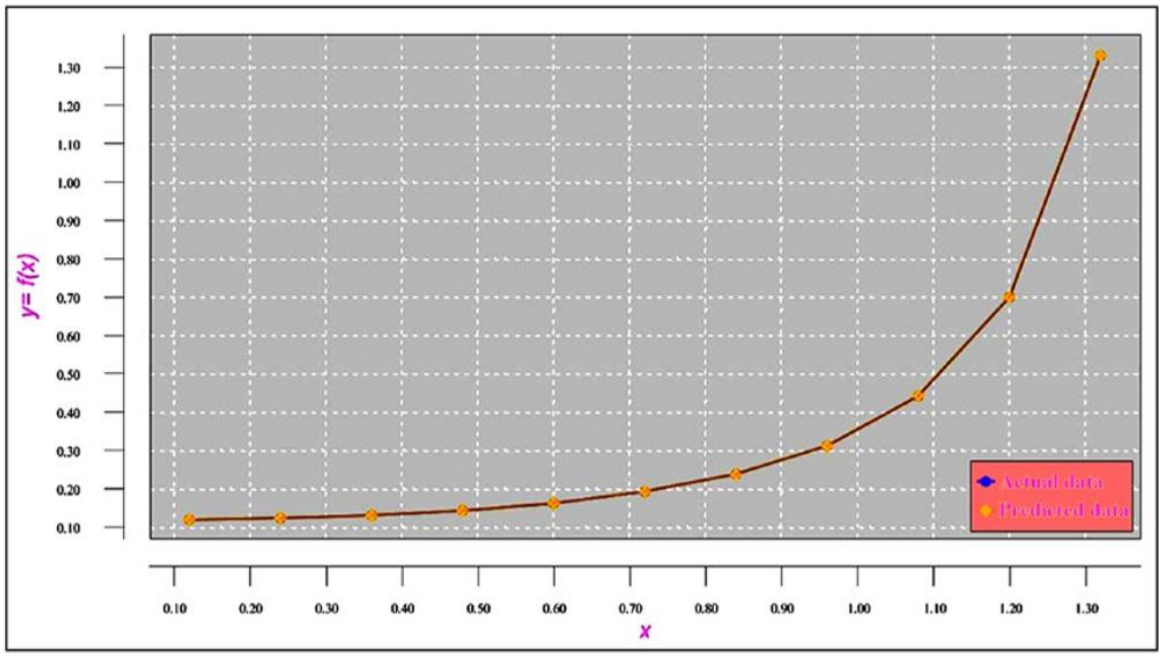
Hasil Pelatihan untuk Contoh 3b

Daftar 16-9 menunjukkan hasil pelatihan.

Daftar 16-9. Hasil Pelatihan

```
xPoint = 0.12 TargetValue = 0.12058 PredictedValue = 0.12072 DiffPerc = 0.00143
xPoint = 0.24 TargetValue = 0.12414 PredictedValue = 0.12427 DiffPerc = 0.00135
xPoint = 0.36 TargetValue = 0.13169 PredictedValue = 0.13157 DiffPerc = 9.6467E-4
xPoint = 0.48 TargetValue = 0.14421 PredictedValue = 0.14410 DiffPerc = 0.00100
xPoint = 0.60 TargetValue = 0.16353 PredictedValue = 0.16352 DiffPerc = 5.31138E-5
xPoint = 0.72 TargetValue = 0.19293 PredictedValue = 0.19326 DiffPerc = 0.00307
xPoint = 0.84 TargetValue = 0.23856 PredictedValue = 0.23842 DiffPerc = 0.00128
xPoint = 0.96 TargetValue = 0.31273 PredictedValue = 0.31258 DiffPerc = 0.00128
xPoint = 1.08 TargetValue = 0.44286 PredictedValue = 0.44296 DiffPerc = 8.16305E-4
xPoint = 1.20 TargetValue = 0.70093 PredictedValue = 0.70088 DiffPerc = 4.05989E-4
xPoint = 1.32 TargetValue = 1.33119 PredictedValue = 1.33123 DiffPerc = 2.74089E-4
maxErrorPerc = 0.0030734810314331077
averErrorPerc = 9.929718215067468E-4
```

Gambar 16.7 menunjukkan grafik nilai fungsi aktual versus hasil validasi.



Gambar 16.7 Bagan hasil pelatihan/validasi

Seperti yang ditunjukkan pada Gambar 16.7, kedua diagram praktis tumpang tindih.

Hasil Pengujian untuk Contoh 3b

Listing 16-10 menunjukkan hasil pengujian pada interval [3.141592654, 4.341592654].

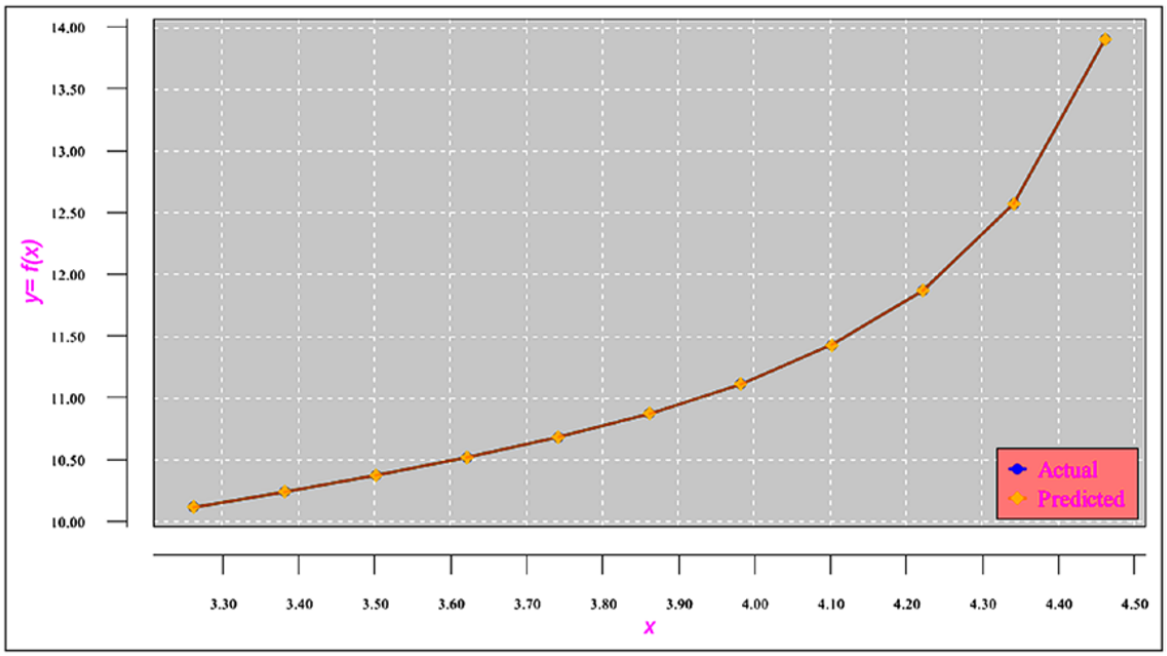
Daftar 16-10. Hasil Pengujian pada Interval [3.141592654, 4.341592654]

xPoint = 3.26159 TargetValue = 10.12058 PredictedValue = 10.12072 DiffPerc = 0.00143
xPoint = 3.38159 TargetValue = 10.24472 PredictedValue = 10.24485 DiffPerc = 0.00135
xPoint = 3.50159 TargetValue = 10.37640 PredictedValue = 10.37630 DiffPerc = 9.64667E-4
xPoint = 3.62159 TargetValue = 10.52061 PredictedValue = 10.52050 DiffPerc = 0.00100
xPoint = 3.74159 TargetValue = 10.68414 PredictedValue = 10.68413 DiffPerc = 5.31136E-5
xPoint = 3.86159 TargetValue = 10.87707 PredictedValue = 10.87740 DiffPerc = 0.00307
xPoint = 3.98159 TargetValue = 11.11563 PredictedValue = 11.11549 DiffPerc = 0.00127
xPoint = 4.10159 TargetValue = 11.42836 PredictedValue = 11.42821 DiffPerc = 0.00128
xPoint = 4.22159 TargetValue = 11.87122 PredictedValue = 11.87131 DiffPerc = 8.16306E-4
xPoint = 4.34159 TargetValue = 12.57215 PredictedValue = 12.57210 DiffPerc = 4.06070E-4
xPoint = 4.46159 TargetValue = 13.90335 PredictedValue = 13.90338 DiffPerc = 2.74161E-4

maxErrorPerc = 0.003073481240844822

averErrorPerc = 9.929844994337172E-4

Gambar 16.8 menunjukkan grafik hasil pengujian (nilai fungsi aktual versus nilai fungsi yang diprediksi) pada interval [3.141592654, 9.424777961].



Gambar 16.8 Bagan hasil pengujian

Baik grafik aktual dan prediksi praktis tumpang tindih.

16.5 RINGKASAN

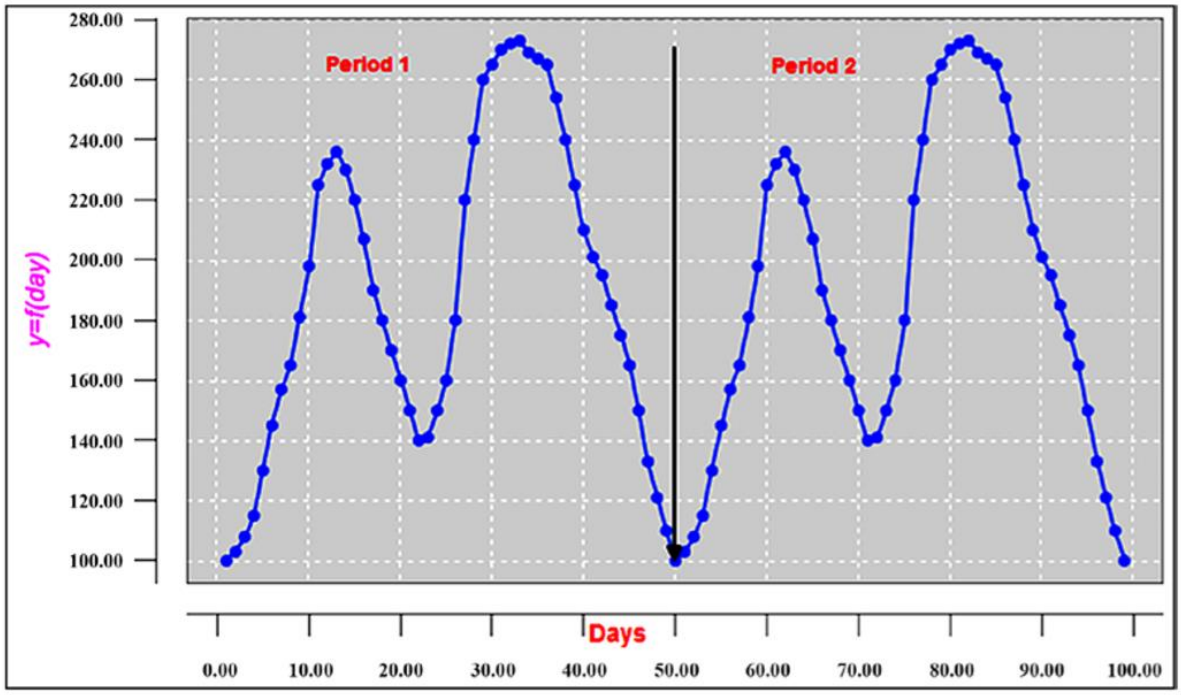
Sekali lagi, jaringan saraf adalah mekanisme pendekatan fungsi universal. Itu berarti bahwa setelah Anda memperkirakan fungsi pada beberapa interval, Anda dapat menggunakan jaringan saraf terlatih tersebut untuk memprediksi nilai fungsi pada titik mana pun dalam interval pelatihan. Namun, Anda tidak dapat menggunakan jaringan terlatih tersebut untuk memprediksi nilai fungsi di luar rentang pelatihan. Jaringan saraf bukanlah mekanisme ekstrapolasi fungsi. Bab ini menjelaskan bagaimana, untuk kelas fungsi tertentu (dalam hal ini, fungsi periodik), dimungkinkan untuk mendapatkan data yang diprediksi di luar rentang pelatihan. Anda akan terus mengeksplorasi konsep ini di bab berikutnya.

BAB 17
MEMPROSES FUNGSI PERIODIK KOMPLEKS

Bab ini melanjutkan diskusi tentang bagaimana memproses fungsi periodik, berkonsentrasi pada fungsi periodik yang lebih kompleks.

17.1 CONTOH 4: PERKIRAAN A FUNGSI DALAM DATA

Mari kita lihat diagram fungsi yang ditunjukkan pada Gambar 17.1. Fungsi tersebut mewakili beberapa data eksperimen yang diukur dalam hari (x adalah hari eksperimen berturut-turut). Ini adalah fungsi periodik dengan periode sama dengan 50 hari.



Gambar 17.1 Bagan fungsi periodik pada dua interval: 1–50 dan 51–100 hari

Tabel 17.1 menunjukkan nilai fungsi untuk dua periode (1–50 dan 50–100 hari).

Tabel 17.1 Nilai Fungsi pada Dua Periode

Hari	Fungsi Nilai (Periode 1)	Hari	Fungsi Nilai (Periode 2)
1	100	51	103
2	103	52	108
3	108	53	115
4	115	54	130
5	130	55	145
6	145	56	157
7	157	57	165
8	165	58	181
9	181	59	198
10	198	60	225
11	225	61	232

12	232	62	236
13	236	63	230
14	230	64	220
15	220	65	207
16	207	66	190
17	190	67	180
18	180	68	170
19	170	68	160
20	160	70	150
21	150	71	140
22	140	72	141
23	141	73	150
24	150	74	160
25	160	75	180
26	180	76	220
27	220	77	240
28	240	78	260
29	260	79	265
30	265	80	270
31	270	81	272
32	272	82	273
33	273	83	269
34	269	84	267
35	267	85	265
36	265	86	254
37	254	87	240
38	240	88	225
39	225	89	210
40	210	90	201
41	201	91	195
42	195	92	185
43	185	93	175
44	175	94	165
45	165	95	150
46	150	96	133
47	133	97	121
48	121	98	110
49	110	99	100
50	100	100	103

17.2 PERSIAPAN DATA

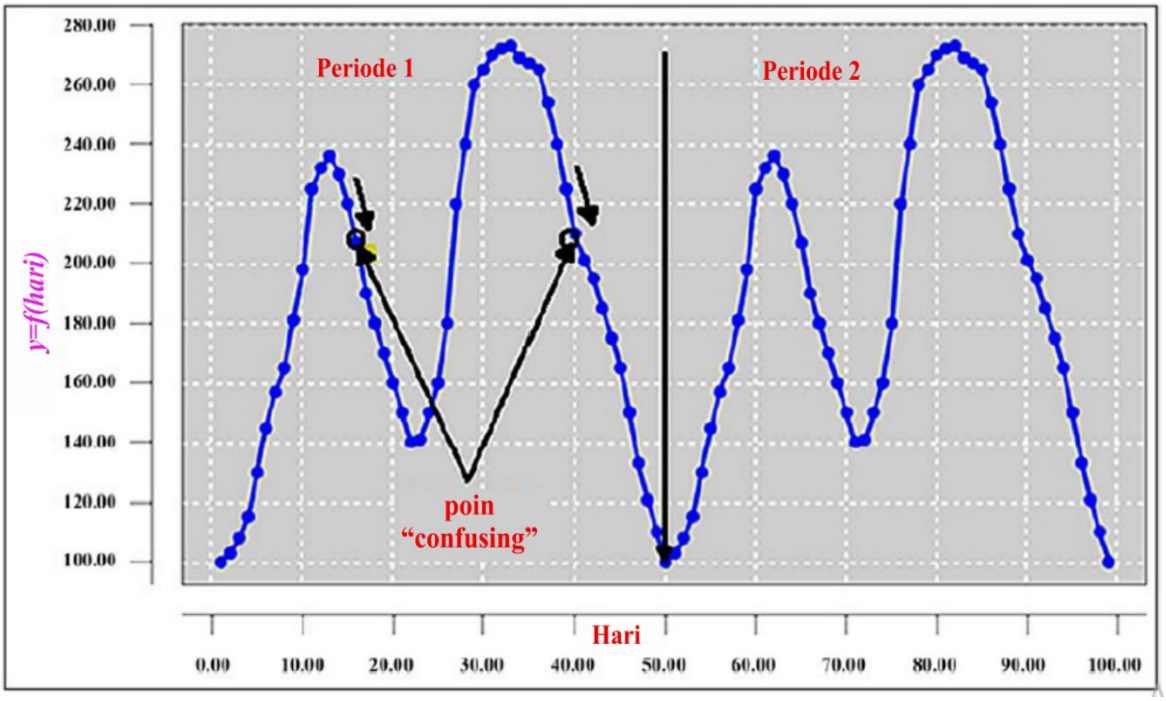
Untuk contoh ini, Anda akan melatih jaringan saraf menggunakan nilai fungsi pada interval pertama dan kemudian menguji jaringan dengan mendapatkan nilai fungsi yang diprediksi jaringan pada interval kedua. Seperti contoh sebelumnya, untuk dapat menentukan hasil aproksimasi fungsi di luar rentang pelatihan, kita akan menggunakan selisih antara nilai xPoint dan selisih antara nilai fungsi alih-alih xPoint dan nilai fungsi yang diberikan. Namun, dalam contoh ini, kami akan menggunakan perbedaan antara nilai

xPoint antara titik saat ini dan sebelumnya sebagai bidang 1 dan perbedaan antara nilai fungsi antara titik berikutnya dan saat ini sebagai bidang 2.

Dengan pengaturan seperti itu di file input, kami akan mengajarkan jaringan untuk mempelajari bahwa ketika perbedaan antara nilai xPoint sama dengan beberapa nilai "a", maka perbedaan nilai fungsi antara hari berikutnya dan hari ini harus sama dengan beberapa nilai "b." Ini memungkinkan jaringan untuk memprediksi nilai fungsi hari berikutnya dengan mengetahui nilai fungsi (catatan) hari ini.

Selama pengujian, kami akan menghitung nilai fungsi hari berikutnya dengan cara berikut. Dengan berada di titik x = 50, Anda ingin menghitung nilai prediksi fungsi pada titik berikutnya, x = 51. Mengumpankan perbedaan antara nilai xPoint pada titik saat ini dan sebelumnya (bidang 1) ke jaringan yang dilatih, kita akan mendapatkan kembali perbedaan yang diprediksi antara nilai fungsi pada titik berikutnya dan saat ini. Oleh karena itu, nilai fungsi prediksi pada titik berikutnya sama dengan jumlah nilai fungsi aktual pada titik saat ini dan selisih nilai prediksi yang diperoleh dari jaringan yang dilatih.

Namun, ini tidak akan berfungsi untuk contoh ini, hanya karena banyak bagian bagan mungkin memiliki perbedaan dan arah yang sama dalam nilai fungsi antara hari ini dan hari sebelumnya. Ini akan membingungkan proses pembelajaran jaringan saraf ketika mencoba menentukan bagian mana dari bagan yang dimiliki titik tersebut (lihat Gambar 17.2).



Gambar 17.2 Poin membingungkan pada bagan fungsi

17.3 MEREFLEKSIKAN TOPOLOGI FUNGSI DALAM DATA

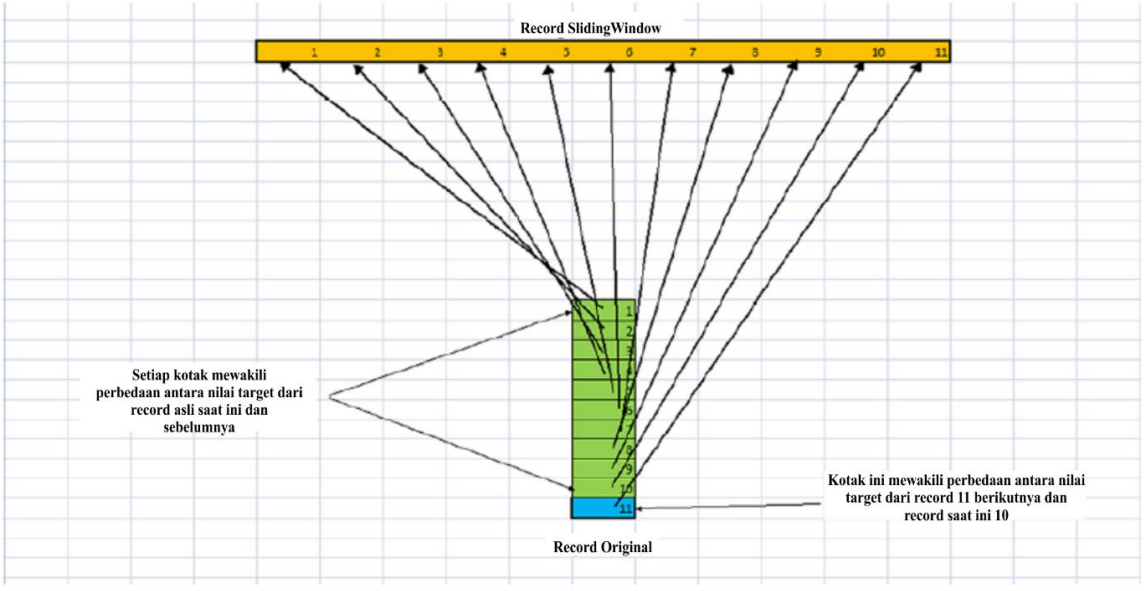
Untuk contoh ini, Anda perlu menggunakan trik tambahan. Anda akan menyertakan fungsi topologi dalam data untuk membantu jaringan membedakan antara titik-titik yang membingungkan. Secara khusus, file pelatihan Anda akan menggunakan jendela geser sebagai catatan input.

Setiap catatan jendela geser mencakup perbedaan nilai fungsi input (antara hari ini dan hari sebelumnya) dari sepuluh catatan sebelumnya. Nilai fungsi untuk sepuluh hari sebelumnya disertakan dalam jendela geser karena sepuluh hari sudah cukup untuk

membuat catatan yang membingungkan dapat dibedakan. Nilai fungsi target dari jendela geser adalah perbedaan nilai fungsi target (antara hari berikutnya dan saat ini) dari catatan asli 11.

Anda sedang membangun catatan jendela geser yang terdiri dari sepuluh bidang yang berisi nilai fungsi untuk sepuluh hari sebelumnya, karena sepuluh hari sudah cukup untuk membedakan titik-titik membingungkan pada bagan. Namun, lebih banyak hari dapat dimasukkan dalam catatan (katakanlah, 12 hari).

Pada dasarnya, dengan menggunakan format rekaman seperti itu, Anda mengajarkan jaringan untuk mempelajari kondisi berikut. Jika selisih nilai fungsi kesepuluh record sebelumnya sama dengan a1, a2, a3, a4, a5, a6, a7, a8, a9, dan a10, maka selisih nilai fungsi hari berikutnya dan nilai fungsi hari ini harus sama dengan nilai fungsi target dari record berikutnya (record 11). Gambar 17.3 menunjukkan contoh visual membuat rekaman jendela geser.



Gambar 17.3 Membangun catatan jendela geser

Tabel 17.2 menunjukkan set data pelatihan jendela geser. Setiap jendela geser (catatan) mencakup sepuluh bidang yang berasal dari sepuluh hari sebelumnya ditambah satu bidang tambahan dengan nilai yang diharapkan untuk diprediksi.

Tabel 17.2 Menggeser Kumpulan Data Windows pada Interval [1, 50]

Sliding Windows										
-9	-6	-10	-10	-10	-15	-17	-12	-11	-10	3
-6	-10	-10	-10	-15	-17	-12	-11	-10	3	5
-10	-10	-10	-15	-17	-12	-11	-10	3	3	7
-10	-10	-15	-17	-12	-11	-10	3	3	7	15
-10	-15	-17	-12	-11	-10	3	3	7	15	15
-15	-17	-12	-11	-10	3	3	7	15	15	12
-17	-12	-11	-10	3	3	7	15	15	12	8
-12	-11	-10	3	3	7	15	15	12	8	16
-11	-10	3	3	7	15	15	12	8	16	17
-10	3	3	7	15	15	12	8	16	17	27
3	3	7	15	15	12	8	16	17	27	7
3	7	15	15	12	8	16	17	27	7	4
7	15	15	12	8	16	17	27	7	4	-6
15	15	12	8	16	17	27	7	4	-6	-10
15	12	8	16	17	27	7	4	-6	-10	-13
12	8	16	17	27	7	4	-6	-10	-13	-17
8	16	17	27	7	4	-6	-10	-13	-17	-10
16	17	27	7	4	-6	-10	-13	-17	-10	-10
17	27	7	4	-6	-10	-13	-17	-10	-10	-10
27	7	4	-6	-10	-13	-17	-10	-10	-10	-10
7	4	-6	-10	-13	-17	-10	-10	-10	-10	-10
4	-6	-10	-13	-17	-10	-10	-10	-10	-10	1
-6	-10	-13	-17	-10	-10	-10	-10	-10	1	9
-10	-13	-17	-10	-10	-10	-10	-10	1	9	10
-13	-17	-10	-10	-10	-10	-10	1	9	10	20
-17	-10	-10	-10	-10	-10	1	9	10	20	40

-10	-10	-10	-10	-10	1	9	10	20	40	20
-10	-10	-10	-10	1	9	10	20	40	20	20
-10	-10	-10	1	9	10	20	40	20	20	5
-10	-10	1	9	10	20	40	20	20	5	5
-10	1	9	10	20	40	20	20	5	5	2
1	9	10	20	40	20	20	5	5	2	1
9	10	20	40	20	20	5	5	2	1	-4
10	20	40	20	20	5	5	2	1	-4	-2
20	40	20	20	5	5	2	1	-4	-2	-2
40	20	20	5	5	2	1	-4	-2	-2	-11
20	20	5	5	2	1	-4	-2	-2	-11	-14
20	5	5	2	1	-4	-2	-2	-11	-14	-15
5	5	2	1	-4	-2	-2	-11	-14	-15	-15
5	2	1	-4	-2	-2	-11	-14	-15	-15	-9
2	1	-4	-2	-2	-11	-14	-15	-15	-9	-6
1	-4	-2	-2	-11	-14	-15	-15	-9	-6	-10
-4	-2	-2	-11	-14	-15	-15	-9	-6	-10	-10
-2	-2	-11	-14	-15	-15	-9	-6	-10	-10	-10
-2	-11	-14	-15	-15	-9	-6	-10	-10	-10	-15
-11	-14	-15	-15	-9	-6	-10	-10	-10	-15	-17
-14	-15	-15	-9	-6	-10	-10	-10	-15	-17	-12
-15	-15	-9	-6	-10	-10	-10	-15	-17	-12	-11
-15	-9	-6	-10	-10	-10	-15	-17	-12	-11	-10
-9	-6	-10	-10	-10	-15	-17	-12	-11	-10	3

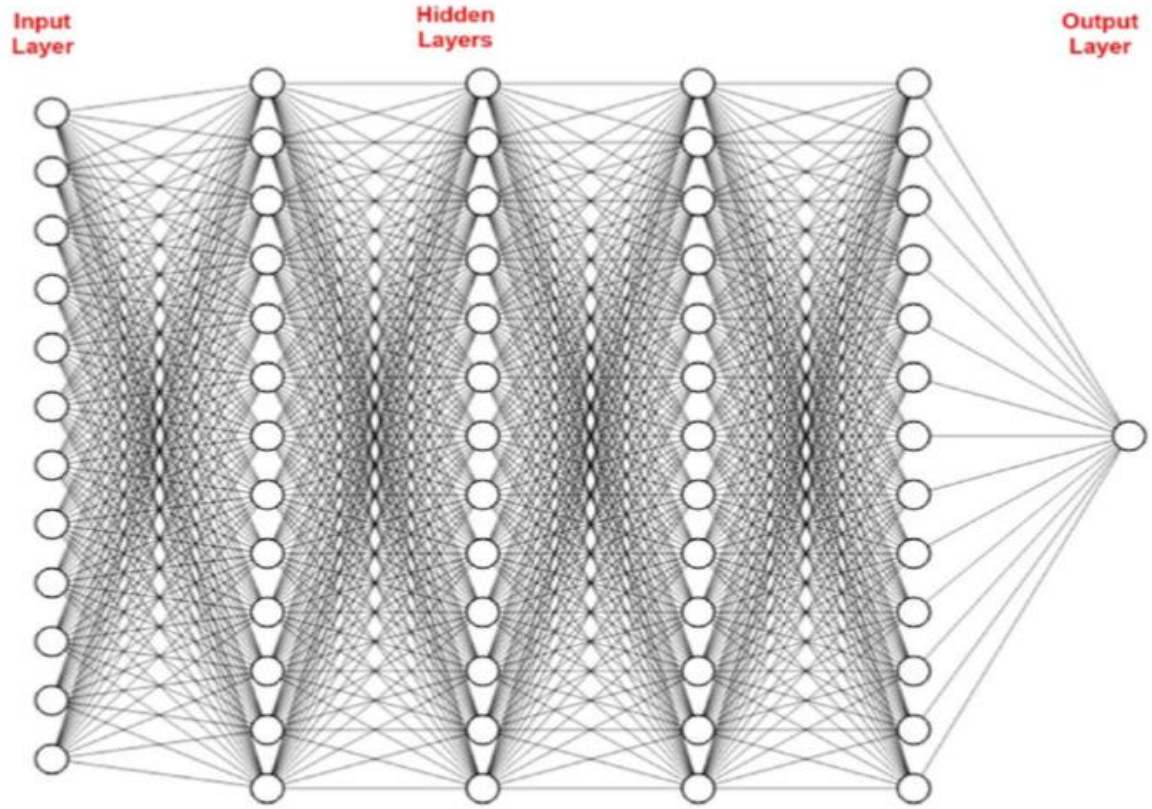
Normalized Sliding Windows

-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286
-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286	-0.28571
-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286	-0.34286	-0.22857
-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286	-0.34286	-0.22857	0
-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286	-0.34286	-0.22857	0	0
-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286	-0.34286	-0.22857	0	0	-0.08571
-0.91429	-0.77143	-0.74286	-0.71429	-0.34286	-0.34286	-0.22857	0	0	-0.08571	-0.2
-0.77143	-0.74286	-0.71429	-0.34286	-0.34286	-0.22857	0	0	-0.08571	-0.2	0.028571
-0.74286	-0.71429	-0.34286	-0.34286	-0.22857	0	0	-0.08571	-0.2	0.028571	0.057143
-0.71429	-0.34286	-0.34286	-0.22857	0	0	-0.08571	-0.2	0.028571	0.057143	0.342857
-0.34286	-0.34286	-0.22857	0	0	-0.08571	-0.2	0.028571	0.057143	0.342857	-0.22857
-0.34286	-0.22857	0	0	-0.08571	-0.2	0.028571	0.057143	0.342857	-0.22857	-0.31429
-0.22857	0	0	-0.08571	-0.2	0.028571	0.057143	0.342857	-0.22857	-0.31429	-0.6

0	0	-0.08571	-0.2	0.028571	0.057143	0.342857	-0.22857	-0.31429	-0.6	-0.71429
0	-0.08571	-0.2	0.028571	0.057143	0.342857	-0.22857	-0.31429	-0.6	-0.71429	-0.8
-0.08571	-0.2	0.028571	0.057143	0.342857	-0.22857	-0.31429	-0.6	-0.71429	-0.8	-0.91429
-0.2	0.028571	0.057143	0.342857	-0.22857	-0.31429	-0.6	-0.71429	-0.8	-0.91429	-0.71429
0.028571	0.057143	0.342857	-0.22857	-0.31429	-0.6	-0.71429	-0.8	-0.91429	-0.71429	-0.71429
0.057143	0.342857	-0.22857	-0.31429	-0.6	-0.71429	-0.8	-0.91429	-0.71429	-0.71429	-0.71429
0.342857	-0.22857	-0.31429	-0.6	-0.71429	-0.8	-0.91429	-0.71429	-0.71429	-0.71429	-0.71429
-0.22857	-0.31429	-0.6	-0.71429	-0.8	-0.91429	-0.71429	-0.71429	-0.71429	-0.71429	-0.71429
-0.31429	-0.6	-0.71429	-0.8	-0.91429	-0.71429	-0.71429	-0.71429	-0.71429	-0.71429	-0.4
-0.6	-0.71429	-0.8	-0.91429	-0.71429	-0.71429	-0.71429	-0.71429	-0.71429	-0.4	-0.17143
-0.71429	-0.8	-0.91429	-0.71429	-0.71429	-0.71429	-0.71429	-0.71429	-0.4	-0.17143	-0.14286
-0.8	-0.91429	-0.71429	-0.71429	-0.71429	-0.71429	-0.71429	-0.4	-0.17143	-0.14286	0.142857
-0.91429	-0.71429	-0.71429	-0.71429	-0.71429	-0.71429	-0.4	-0.17143	-0.14286	0.142857	0.714286
-0.71429	-0.71429	-0.71429	-0.71429	-0.71429	-0.4	-0.17143	-0.14286	0.142857	0.714286	0.142857
-0.71429	-0.71429	-0.71429	-0.71429	-0.4	-0.17143	-0.14286	0.142857	0.714286	0.142857	0.142857
-0.71429	-0.71429	-0.71429	-0.4	-0.17143	-0.14286	0.142857	0.714286	0.142857	0.142857	-0.28571
-0.71429	-0.71429	-0.4	-0.17143	-0.14286	0.142857	0.714286	0.142857	0.142857	-0.28571	-0.28571
-0.71429	-0.4	-0.17143	-0.14286	0.142857	0.714286	0.142857	0.142857	-0.28571	-0.28571	-0.37143
-0.4	-0.17143	-0.14286	0.142857	0.714286	0.142857	0.142857	-0.28571	-0.28571	-0.37143	-0.4
-0.17143	-0.14286	0.142857	0.714286	0.142857	0.142857	-0.28571	-0.28571	-0.37143	-0.4	-0.54286
-0.14286	0.142857	0.714286	0.142857	0.142857	-0.28571	-0.28571	-0.37143	-0.4	-0.54286	-0.48571
0.142857	0.714286	0.142857	0.142857	-0.28571	-0.28571	-0.37143	-0.4	-0.54286	-0.48571	-0.48571
0.714286	0.142857	0.142857	-0.28571	-0.28571	-0.37143	-0.4	-0.54286	-0.48571	-0.48571	-0.74286
0.142857	0.142857	-0.28571	-0.28571	-0.37143	-0.4	-0.54286	-0.48571	-0.48571	-0.74286	-0.82857
0.142857	-0.28571	-0.28571	-0.37143	-0.4	-0.54286	-0.48571	-0.48571	-0.74286	-0.82857	-0.85714
-0.28571	-0.28571	-0.37143	-0.4	-0.54286	-0.48571	-0.48571	-0.74286	-0.82857	-0.85714	-0.85714
-0.28571	-0.37143	-0.4	-0.54286	-0.48571	-0.48571	-0.74286	-0.82857	-0.85714	-0.85714	-0.68571
-0.37143	-0.4	-0.54286	-0.48571	-0.48571	-0.74286	-0.82857	-0.85714	-0.85714	-0.68571	-0.6
-0.4	-0.54286	-0.48571	-0.48571	-0.74286	-0.82857	-0.85714	-0.85714	-0.68571	-0.6	-0.71429
-0.54286	-0.48571	-0.48571	-0.74286	-0.82857	-0.85714	-0.85714	-0.68571	-0.6	-0.71429	-0.71429
-0.48571	-0.48571	-0.74286	-0.82857	-0.85714	-0.85714	-0.68571	-0.6	-0.71429	-0.71429	-0.71429
-0.48571	-0.74286	-0.82857	-0.85714	-0.85714	-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714
-0.74286	-0.82857	-0.85714	-0.85714	-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429
-0.82857	-0.85714	-0.85714	-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143
-0.85714	-0.85714	-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286
-0.85714	-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429
-0.68571	-0.6	-0.71429	-0.71429	-0.71429	-0.85714	-0.91429	-0.77143	-0.74286	-0.71429	-0.34286

17.4 ARSITEKTUR JARINGAN

Anda akan menggunakan jaringan (ditunjukkan pada Gambar 17.5) yang terdiri dari lapisan input (menampung sepuluh neuron), empat lapisan tersembunyi (masing-masing menampung 13 neuron), dan lapisan output (menampung satu neuron). Alasan memilih arsitektur ini telah dibahas sebelumnya.



Gambar 17.4 Arsitektur jaringan

Setiap record dalam set data pelatihan jendela geser berisi sepuluh bidang input dan satu bidang output. Anda siap untuk mengembangkan program pemrosesan jaringan saraf.

Kode Program

Daftar 17-1 menunjukkan kode program.

Daftar 17-1. Kode Program

```
// ============================================================
// Perkiraan fungsi periodik kompleks. Inputnya adalah
// file pelatihan
// atau pengujian dengan catatan yang dibuat sebagai
// jendela geser.Setiap catatan geser
// jendela berisi 11 bidang.
// 10 field pertama adalah nilai field1 dari 10 record
// asli ditambah
// nilai field2 dari record berikutnya, yang sebenarnya
// merupakan perbedaan antara nilai target dari record
// asli berikutnya (record 11) dan
// record 10.
// ============================================================
package sample4;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
```

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate; import java.time.Month;
import java.time.ZoneId; import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
```

```
public class Sample4 implements ExampleChart<XYChart>
{
	static double doublePointNumber = 0.00;
	static int intPointNumber = 0;
	static InputStream input = null;
	static double[] arrFunctionValue = new double[500];
	static double inputDiffValue = 0.00;
	static double targetDiffValue = 0.00;
	static double predictDiffValue = 0.00;
	static double valueDifferencePerc = 0.00;
	static String strFunctionValuesFileName;
	static int returnCode  = 0;
	static int numberOfInputNeurons;
	static int numberOfOutputNeurons;
	static int numberOfRecordsInFile;
	static int intNumberOfRecordsInTestFile;
	static double realTargetDiffValue;
	static double realPredictDiffValue;
	static String functionValuesTrainFileName;
	static String functionValuesTestFileName;
	static String trainFileName;
	static String priceFileName;
	static String testFileName;
	static String chartTrainFileName;
	static String chartTestFileName;
	static String networkFileName;
	static int workingMode;
	static String cvsSplitBy = ",";
	// De-normalization parameters
	static double Nh =  1;
	static double Nl = -1;
	static double Dh = 50.00;
	static double Dl = -20.00;
	static String inputtargetFileName ;
	static double lastFunctionValueForTraining = 0.00;
	static int tempIndexField;
	static double tempTargetField;
	static  int[] arrIndex = new int[100];
	static double[] arrTarget = new double[100];
	static List<Double> xData = new ArrayList<Double>();
	static List<Double> yData1 = new ArrayList<Double>();
	static List<Double> yData2 = new ArrayList<Double>();
	static XYChart Chart;
@Override
public XYChart getChart()
{
	// Create Chart
```

```
    Chart = new XYChartBuilder().width(900).height(500).title(getClass().
    getSimpleName()).xAxisTitle("Days").yAxisTitle("y= f(x)").build();
    // Customize Chart
    Chart.getStyler().setPlotBackgroundColor(ChartColor.
    getAWTColor(ChartColor.GREY));
    Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
    Chart.getStyler().setChartBackgroundColor(Color.WHITE);
    Chart.getStyler().setLegendBackgroundColor(Color.PINK);
    Chart.getStyler().setChartFontColor(Color.MAGENTA);
    Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
    Chart.getStyler().setChartTitleBoxVisible(true);
    Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
    Chart.getStyler().setPlotGridLinesVisible(true);
    Chart.getStyler().setAxisTickPadding(20);
    Chart.getStyler().setAxisTickMarkLength(15);
    Chart.getStyler().setPlotMargin(20);
    Chart.getStyler().setChartTitleVisible(false);
    Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,  Font.BOLD, 24));
    Chart.getStyler().setLegendFont(new
    Font(Font.SERIF, Font.PLAIN, 18));
    Chart.getStyler().setLegendPosition(LegendPosition.OutsideS);
    Chart.getStyler().setLegendSeriesLineLength(12);
    Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF,  Font.ITALIC, 18));
    Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.PLAIN, 11));
    Chart.getStyler().setDatePattern("yyyy-MM");
    Chart.getStyler().setDecimalPattern("#0.00");
    // Interval to normalize    double Nh =  1;    double Nl = -1;
    // Values in the sliding windows    double Dh = 50.00;    double Dl = -20.00;
Try
{
    // Configuration
    // Setting the mode of the program run        workingMode = 1;
    // Set to run the program in the training mode
    if (workingMode == 1)
    {
            // Configure the program to run in the training mode
            trainFileName = "C:/Book_Examples/Sample4_Norm_Train_Sliding_
            Windows_File.csv";
            functionValuesTrainFileName = "C:/Book_Examples/Sample4_
            Function_values_Period_1.csv";
            chartTrainFileName = "XYLine_Sample4_Train_Chart";
            numberOfRecordsInFile = 51;
    }
    else
            {
            // Configure the program to run in the testing mode
            trainFileName = "C:/Book_Examples/Sample4_Norm_Train_Sliding_
            Windows_File.csv";
```

```
                functionValuesTrainFileName = "C:/Book_Examples/Sample4_
                Function_values_Period_1.csv";
                chartTestFileName = "XYLine_Sample4_Test_Chart";
                numberOfRecordsInFile = 51;
                lastFunctionValueForTraining = 100.00;
    }
// -------------------------------------------------------
// konfigurasi umum
// -------------------------------------------------------
networkFileName = "C:/Book_Examples/Example4_Saved_Network_File.csv";
inputtargetFileName = "C:/Book_Examples/Sample4_Input_File.csv";
numberOfInputNeurons = 10;
numberOfOutputNeurons = 1;
// Check the working mode to run
// Training mode. Train, validate, and save the trained network file
if(workingMode == 1)    {
        // Load function values for training file in memory
        loadFunctionValueTrainFileInMemory();
        File file1 = new File(chartTrainFileName);
        File file2 = new File(networkFileName);
        if(file1.exists())
        file1.delete();
        if(file2.exists())
        file2.delete();
        returnCode = 0; // Clear the error Code
        do
        {
        returnCode = trainValidateSaveNetwork();
        } while (returnCode > 0);
    } // End the train logic
    Else
    {
        // Test mode. Test the approximation at the points where
        // neural network was not trained
        // Load function values for training file in memory
        loadInputTargetFileInMemory();
        //loadFunctionValueTrainFileInMemory();
        File file1 = new File(chartTestFileName);
        if(file1.exists())
        file1.delete();
        loadAndTestNetwork();
        }
    }
    catch (NumberFormatException e)
    {
        System.err.println("Problem parsing workingMode.workingMode = " +
        workingMode);
        System.exit(1);
```

```
        }
        catch (Throwable t)
        {
        t.printStackTrace();
        System.exit(1);
        }
        finally
        {
        Encog.getInstance().shutdown();
        }
        Encog.getInstance().shutdown();
        return Chart;
} // End of the method
// ==========================================================
// Metode ini melatih, memvalidasi, dan menyimpan file
// jaringan yang dilatih
// ==========================================================
public static MLDataSet loadCSV2Memory(String filename, int input,  int ideal, boolean headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers, input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// ==========================================================
// Metode Pengujian
// @param Command line arguments. No arguments are used.
// ==========================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample4();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ==========================================================
// Memuat nilai Fungsi untuk file pelatihan di memori
// ==========================================================
static public int trainValidateSaveNetwork()
{
        double functionValue = 0.00;
        double denormInputValueDiff = 0.00;
        double denormTargetValueDiff = 0.00;
        double denormTargetValueDiff_02 = 0.00;
        double denormPredictValueDiff = 0.00;
        double denormPredictValueDiff_02 = 0.00;
        // Load the training CSV file in memory
```

```
MLDataSet trainingSet =
loadCSV2Memory(trainFileName,numberOfInputNeurons,
numberOfOutputNeurons,true,CSVFormat.ENGLISH,false);
// create a neural network
BasicNetwork network = new BasicNetwork();
// Input layer
network.addLayer(new BasicLayer(null,true,10));
// Hidden layer
network.addLayer(new BasicLayer(new
ActivationTANH(),true,13));
network.addLayer(new BasicLayer(new
ActivationTANH(),true,13));
network.addLayer(new BasicLayer(new
ActivationTANH(),true,13));
network.addLayer(newBasicLayer(new
ActivationTANH(),true,13));
// Output layer network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// train the neural network final ResilientPropagation train = new
ResilientPropagation(network, trainingSet);
int epoch = 1;
returnCode = 0;
do
{
train.iteration();
System.out.println("Epoch #" + epoch + " Error:" + train.getError());
epoch++;
if (epoch >= 11000 && network.calculateError(trainingSet) > 0.00000119)
{
returnCode = 1;
System.out.println("Error = " +  network.calculateError (trainingSet));
System.out.println("Try again");
return returnCode;
}
} while(train.getError() > 0.000001187);
// Save the network file EncogDirectoryPersistence.saveObject(new
File(networkFileName),network);
double sumGlobalDifferencePerc = 0.00;
double sumGlobalDifferencePerc_02 = 0.00;
double averGlobalDifferencePerc = 0.00;
double maxGlobalDifferencePerc = 0.00;
double maxGlobalDifferencePerc_02 = 0.00;
int m = 0;
// Record number in the input file
double xPoint_Initial = 1.00;
double xPoint_Increment = 1.00;
double xPoint = xPoint_Initial - xPoint_Increment;
```

```
        realTargetDiffValue = 0.00;
        realPredictDiffValue = 0.00;
        for(MLDataPair pair: trainingSet)
        {
                m++;
                xPoint = xPoint + xPoint_Increment;
                if(xPoint >  50.00)
                break;
                final MLData output = network.compute(pair.getInput());
                MLData inputData = pair.getInput();
                MLData actualData = pair.getIdeal();
                MLData predictData = network.compute(inputData);
                inputDiffValue = inputData.getData(0);
                targetDiffValue = actualData.getData(0);
                predictDiffValue = predictData.getData(0);
                // De-normalize the values denormInputValueDiff =((Dl -
                Dh)*inputDiffValue - Nh*Dl + Dh*Nl)/(Nl - Nh);
                denormTargetValueDiff = ((Dl - Dh)*targetDiffValue - Nh*Dl + Dh*Nl)/(Nl -
                Nh);
                denormPredictValueDiff =((Dl - Dh)*predictDiffValue - Nh*Dl + Dh*Nl)/(Nl -
                Nh);
                functionValue = arrFunctionValue[m-1];
                realTargetDiffValue = functionValue + denormTargetValueDiff;
                realPredictDiffValue = functionValue + denormPredictValueDiff;
                valueDifferencePerc = Math.abs(((realTargetDiffValue -
                realPredictDiffValue)/ realPredictDiffValue)*100.00);
                System.out.println ("xPoint = " + xPoint + " realTargetDiffValue = " +
                realTargetDiffValue +" realPredictDiffValue = " + realPredictDiffValue);
                SumGlobalDifferencePerc = sumGlobalDifferencePerc +
                valueDifferencePerc;
                if (valueDifferencePerc > maxGlobalDifferencePerc)
                maxGlobalDifferencePerc = valueDifferencePerc;
                xData.add(xPoint);
                yData1.add(realTargetDiffValue);
                yData2.add(realPredictDiffValue);
} // End for pair loop
XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);       XYSeries series2 =
Chart.addSeries("Predict data", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
try
{
//Save the chart image BitmapEncoder.saveBitmapWithDPI(Chart, chartTrainFileName,
BitmapFormat.JPG, 100);
System.out.println ("Train Chart file has been saved") ;
}
```

```
catch (IOException ex)
{
ex.printStackTrace();
System.exit(3);
}
// Finally, save this trained network EncogDirectoryPersistence.saveObject(new
File(networkFileName),network);
System.out.println ("Train Network has been saved") ;
averGlobalDifferencePerc = sumGlobalDifferencePerc/numberOfRecordsInFile;
System.out.println(" ");
System.out.println("maxGlobalDifferencePerc = " + maxGlobalDifferencePerc + "
averGlobalDifferencePerc = " + averGlobalDifferencePerc);
returnCode = 0;
      return returnCode;
   }  // End of the method
// ============================================================
// Testing Method
// ============================================================
static public void loadAndTestNetwork()
{
         System.out.println("Testing the networks results");
         List<Double> xData = new ArrayList<Double>();
         List<Double> yData1 = new ArrayList<Double>();
         List<Double> yData2 = new ArrayList<Double>();
         int intStartingPriceIndexForBatch = 0;
         int intStartingDatesIndexForBatch = 0;
         double sumGlobalDifferencePerc = 0.00;
         double maxGlobalDifferencePerc = 0.00;
         double averGlobalDifferencePerc = 0.00;
         double targetToPredictPercent = 0;
         double maxGlobalResultDiff = 0.00;
         double averGlobalResultDiff = 0.00;
         double sumGlobalResultDiff = 0.00;
         double maxGlobalInputPrice = 0.00;
         double sumGlobalInputPrice = 0.00;
         double averGlobalInputPrice = 0.00;
         double maxGlobalIndex = 0;
         double inputDiffValueFromRecord = 0.00;
         double targetDiffValueFromRecord = 0.00;
         double predictDiffValueFromRecord = 0.00;
         double denormInputValueDiff     = 0.00;
         double denormTargetValueDiff = 0.00;
         double denormTargetValueDiff_02 = 0.00;
         double denormPredictValueDiff = 0.00;
         double denormPredictValueDiff_02 = 0.00;
         double normTargetPriceDiff;
         double normPredictPriceDiff;
         String tempLine;     String[] tempWorkFields;
```

```
double tempInputXPointValueFromRecord = 0.0;
double tempTargetXPointValueFromRecord = 0.00;
double tempValueDiffence = 0.00;
double functionValue;
double minXPointValue = 0.00;
double minTargetXPointValue = 0.00;
int tempMinIndex = 0;
double rTempTargetXPointValue = 0.00;
double rTempPriceDiffPercKey = 0.00;
double rTempPriceDiff = 0.00;
double rTempSumDiff = 0.00;
double r1 = 0.00;
double r2 = 0.00;
BufferedReader br4;
BasicNetwork network;
int k1 = 0;
// Process testing records
maxGlobalDifferencePerc  = 0.00;
averGlobalDifferencePerc = 0.00;
sumGlobalDifferencePerc = 0.00;
realTargetDiffValue = 0.00;
realPredictDiffValue = 0.00;
// Load the training dataset into memory MLDataSet trainingSet =
loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutput
Neurons,true,CSVFormat.ENGLISH,false);
// Load the saved trained network network =
(BasicNetwork)EncogDirectoryPersistence.loadObject(new File(networkFileName));
int m = 0;
// Index of the current record
// Record number in the input file  double xPoint_Initial = 51.00;
double xPoint_Increment = 1.00;
double xPoint = xPoint_Initial - xPoint_Increment;
for (MLDataPair pair:  trainingSet)
{
        m++;
        xPoint = xPoint + xPoint_Increment;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results  inputDiffValue = inputData.getData(0);
        targetDiffValue = actualData.getData(0); predictDiffValue =
        predictData.getData(0);
        if(m == 1)  functionValue = lastFunctionValueForTraining;
        else
        // De-normalize the values denormInputValueDiff =((Dl -
        Dh)*inputDiffValue - Nh*Dl + Dh*Nl)/(Nl - Nh);
```

```
                denormTargetValueDiff = ((Dl - Dh)*targetDiffValue - Nh*Dl + Dh*Nl)/(Nl -
                Nh);
                denormPredictValueDiff =((Dl - Dh)*predictDiffValue - Nh*Dl + Dh*Nl)/(Nl -
                Nh);
                realTargetDiffValue = functionValue + denormTargetValueDiff;
                realPredictDiffValue = functionValue + denormPredictValueDiff;
                valueDifferencePerc = Math.abs(((realTargetDiffValue -
                realPredictDiffValue)/ realPredictDiffValue)*100.00);
                System.out.println ("xPoint = " + xPoint + " realTargetDiffValue = " +
                realTargetDiffValue + " realPredictDiffValue = " + realPredictDiffValue);
                sumGlobalDifferencePerc = sumGlobalDifferencePerc +
                valueDifferencePerc;
                if (valueDifferencePerc > maxGlobalDifferencePerc)
                maxGlobalDifferencePerc = valueDifferencePerc;
                xData.add(xPoint);
                yData1.add(realTargetDiffValue);
                yData2.add(realPredictDiffValue);
                } // End for pair loop
                // Print the max and average results
                System.out.println(" ");
                averGlobalDifferencePerc =
                sumGlobalDifferencePerc/numberOfRecordsInFile;
                System.out.println("maxGlobalResultDiff = " + maxGlobalDifferencePerc);
                System.out.println("averGlobalResultDiff = " + averGlobalDifferencePerc);
                // All testing batch files have been processed XYSeries series1 =
                Chart.addSeries("Actual", xData, yData1);
                XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
                series1.setLineColor(XChartSeriesColors.BLUE);
                series2.setMarkerColor(Color.ORANGE);
                series1.setLineStyle(SeriesLines.SOLID);
                series2.setLineStyle(SeriesLines.SOLID);
                // Save the chart image try
                { BitmapEncoder.saveBitmapWithDPI(Chart, chartTestFileName,
                BitmapFormat.JPG, 100);
                }
                catch (Exception bt)
                {
                bt.printStackTrace();
                }
                System.out.println ("The Chart has been saved");
                System.out.println("End of testing for test records");
                } // End of the method
// ============================================================
// Muat Sample4_Input_File ke dalam 2 array di memori
// ============================================================
public static void loadFunctionValueTrainFileInMemory()
{
        BufferedReader br1 = null;
```

```
        String line = "";
        String cvsSplitBy = ",";
        String tempXPointValue = "";
        double tempYFunctionValue = 0.00;
        try
        {
                br1 = new BufferedReader(new FileReader(functionValuesTrainFileName));
                int i = -1;
                int r = -2;
                while ((line = br1.readLine()) != null)
                {
                i++;
                r++;
                // Skip the header line          if(i > 0)
                {
                // Break the line using comma as separator              String[] workFields =
                line.split(cvsSplitBy);
                tempYFunctionValue = Double.parseDouble(workFields[0]);
                arrFunctionValue[r] = tempYFunctionValue;
                //System.out.println("arrFunctionValue[r] = " + arrFunctionValue[r]);
                }
                } // end of the while loop
                br1.close();
                }
                catch (IOException ex)
                {
                ex.printStackTrace();
                System.err.println("Error opening files = " + ex);
                System.exit(1);
    }
}
// ============================================================
// Load Sample4_Input_File into 2 arrays in memory
// ============================================================
public static void loadInputTargetFileInMemory()
{
        BufferedReader br1 = null;
        String line = "";
        String cvsSplitBy = ",";
        String tempXPointValue = "";
        double tempYFunctionValue = 0.00;
        try
        {
        br1 = new BufferedReader(new FileReader(inputtargetFileName ));
        int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)
        {
```

```
i++;
r++;
// Skip the header line
if(i > 0)
{
// Break the line using comma as separator String[] workFields =
line.split(cvsSplitBy);

tempTargetField = Double.parseDouble(workFields[1]);
arrIndex[r] =  r;
arrTarget[r] = tempTargetField;
}
      } // end of the while loop
       br1.close();
     }
catch (IOException ex)
{
ex.printStackTrace();
System.err.println("Error opening files = " + ex);
System.exit(1);
}
   }
} // End of the class
```

Seperti biasa, Anda akan memuat file pelatihan ke dalam memori dan membangun jaringan. Jaringan yang dibangun memiliki lapisan input dengan sepuluh neuron, empat lapisan tersembunyi (masing-masing dengan 13 neuron), dan lapisan output dengan satu neuron. Setelah jaringan dibangun, Anda akan melatih jaringan dengan mengulang zaman sampai kesalahan jaringan menghapus batas kesalahan. Terakhir, Anda akan menyimpan jaringan terlatih pada disk (akan digunakan nanti oleh metode pengujian).

Perhatikan bahwa Anda memanggil metode pelatihan dalam satu lingkaran (seperti yang Anda lakukan pada contoh sebelumnya). Ketika setelah 11.000 iterasi kesalahan jaringan masih tidak kurang dari batas kesalahan, Anda keluar dari metode pelatihan dengan kode pengembalian 1. Itu akan memicu pemanggilan metode pelatihan lagi dengan set parameter bobot/bias yang baru (Daftar 17-2).

Daftar 17-2. Fragmen Kode di Awal Metode Pelatihan

```
// Load the training CSV file in memory MLDataSet trainingSet =
loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
// Create a neural network
BasicNetwork network = new BasicNetwork();
// Input layer network.addLayer(new BasicLayer(null,true,10));
// Hidden layer network.addLayer(new
BasicLayer(new ActivationTANH(),true,13));
network.addLayer(new BasicLayer(new ActivationTANH(),true,13));
network.addLayer(new BasicLayer(new ActivationTANH(),true,13));
network.addLayer(new BasicLayer(new ActivationTANH(),true,13)); // Output layer
network.addLayer(new
```

```
BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// train the neural network final ResilientPropagation train = new
ResilientPropagation(network, trainingSet);
int epoch = 1; returnCode = 0;
do
        {
        train.iteration();
        System.out.println("Epoch #" + epoch + " Error:" + train.getError());
        epoch++;
        if (epoch >= 11000 && network.calculateError(trainingSet) > 0.00000119)
        {
        // Exit the training method with the return code = 1
        returnCode = 1;
        System.out.println("Try again");
        return returnCode;
        }
        } while(train.getError() > 0.000001187);
        // Save the network file
    EncogDirectoryPersistence.saveObject(new File(networkFileName),network);
```

Selanjutnya, Anda mengulang set data pasangan, mendapatkan dari jaringan nilai input, aktual, dan prediksi untuk setiap record. Nilai record dinormalisasi, jadi Anda mendenormalisasi nilainya. Rumus berikut digunakan untuk denormalisasi.

$$f(x) = ((D_L - D_H) * x - N_H * D_L * N_L * D_H) / (N_L - N_H)$$

di mana:

- x: Input titik data
- DL: Nilai minimum (terendah) x dalam kumpulan data input
- DH: Nilai x maksimum (tertinggi) dalam kumpulan data input
- NL: Bagian kiri dari interval ternormalisasi [-1, 1] = -1
- NH: Bagian kanan dari interval ternormalisasi [-1, 1] = 1

Setelah denormalisasi, Anda menghitung bidang realTargetDiffValue dan realPredictDiffValue, mencetak nilainya di log pemrosesan, dan mengisi data bagan untuk rekaman saat ini. Terakhir, Anda menyimpan file grafik pada disk dan keluar dari metode pelatihan dengan kode pengembalian 0 (Daftar 17-3).

Daftar 17-3. Fragmen Kode di Akhir Metode Pelatihan

```
int m = 0;
double xPoint_Initial = 1.00;
double xPoint_Increment = 1.00;
double xPoint = xPoint_Initial - xPoint_Increment;
realTargetDiffValue = 0.00;
realPredictDiffValue = 0.00;
for(MLDataPair pair: trainingSet)
{
        m++;
        xPoint = xPoint + xPoint_Increment;
```

```
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results
        inputDiffValue = inputData.getData(0);
        targetDiffValue = actualData.getData(0);
        predictDiffValue = predictData.getData(0);
        // De-normalize the values denormInputValueDiff =((Dl - Dh)*inputDiffValue
        - Nh*Dl + Dh*Nl)/(Nl - Nh);
        denormTargetValueDiff = ((Dl - Dh)*targetDiffValue - Nh*Dl + Dh*Nl)/(Nl - Nh);
        denormPredictValueDiff =((Dl - Dh)*predictDiffValue - Nh*Dl + Dh*Nl)/(Nl - Nh);
        functionValue = arrFunctionValue[m-1];
        realTargetDiffValue = functionValue + denormTargetValueDiff;
        realPredictDiffValue = functionValue + denormPredictValueDiff;
        valueDifferencePerc = Math.abs(((realTargetDiffValue -
        realPredictDiffValue)/ realPredictDiffValue)*100.00);
        System.out.println ("xPoint = " + xPoint + " realTargetDiffValue = " +
        realTargetDiffValue + " realPredictDiffValue = " + realPredictDiffValue);
        sumDifferencePerc = sumDifferencePerc + valueDifferencePerc;
        if (valueDifferencePerc > maxDifferencePerc) maxDifferencePerc =
        valueDifferencePerc;
        xData.add(xPoint);
        yData1.add(realTargetDiffValue);
        yData2.add(realPredictDiffValue);
} // End for pair loop
YSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
try
        {
        //Save the chart image
        BitmapEncoder.saveBitmapWithDPI(Chart, chartTrainFileName,
        BitmapFormat.JPG, 100);
        System.out.println ("Train Chart file has been saved") ;
        }
        catch (IOException ex)
        {
        ex.printStackTrace();       System.exit(3);       }
        // Finally, save this trained network      EncogDirectoryPersistence.saveObject(new
        File(networkFileName),network);
        System.out.println ("Train Network has been saved") ;
        averNormDifferencePerc  = sumDifferencePerc/numberOfRecordsInFile;
        System.out.println(" ");
```

```
        System.out.println("maxDifferencePerc = " + maxDifferencePerc + "
        averNormDifferencePerc = " + averNormDifferencePerc);
        returnCode = 0;       return returnCode;
}  // End of the method
```

Sejauh ini, logika pemrosesan metode pelatihan hampir sama dengan contoh sebelumnya, dengan mengabaikan bahwa format kumpulan data pelatihan berbeda dan mencakup catatan jendela geser. Anda akan melihat perubahan substansial dalam logika metode pengujian.

17.5 MENGUJI JARINGAN

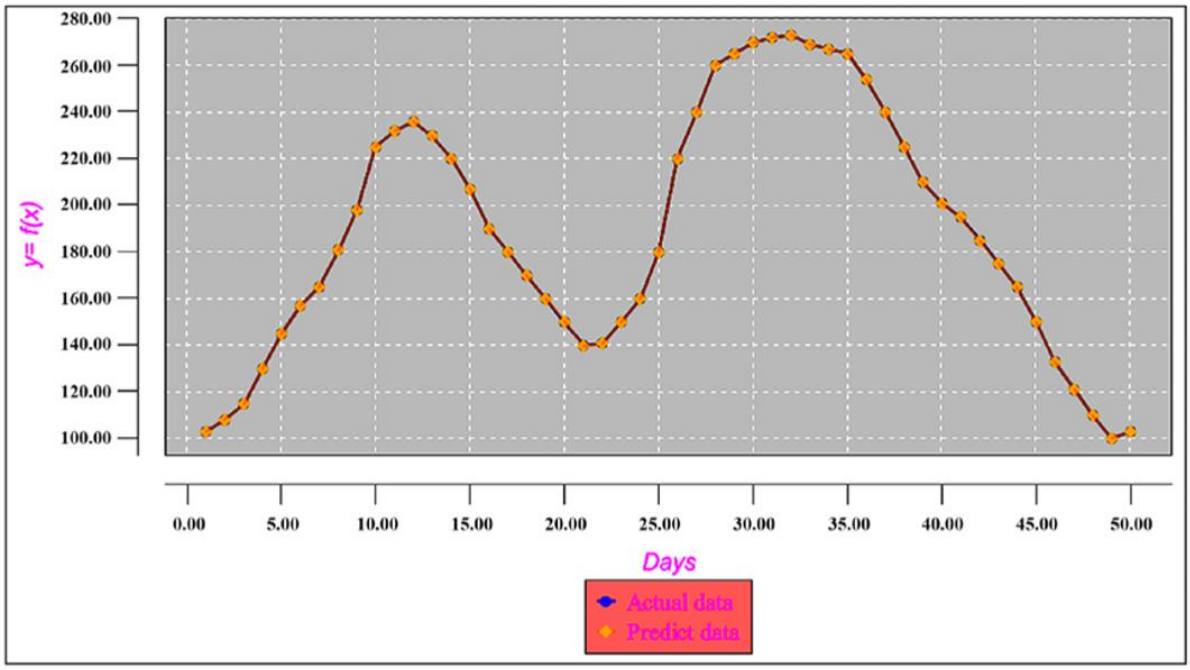
Daftar 17-4 menunjukkan hasil pemrosesan pelatihan.

Daftar 17-4. Hasil Pelatihan

```
xPoint =  1.0  TargetDiff = 102.99999  PredictDiff = 102.98510
xPoint =  2.0  TargetDiff = 107.99999  PredictDiff = 107.99950
xPoint =  3.0  TargetDiff = 114.99999  PredictDiff = 114.99861
xPoint =  4.0  TargetDiff = 130.0          PredictDiff = 130.00147
xPoint =  5.0  TargetDiff = 145.0          PredictDiff = 144.99901
xPoint =  6.0  TargetDiff = 156.99999  PredictDiff = 157.00011
xPoint =  7.0  TargetDiff = 165.0          PredictDiff = 164.99849
xPoint =  8.0  TargetDiff = 181.00000  PredictDiff = 181.00009
xPoint =  9.0  TargetDiff = 197.99999  PredictDiff = 197.99984
xPoint = 10.0  TargetDiff = 225.00000  PredictDiff = 224.99914
xPoint = 11.0  TargetDiff = 231.99999  PredictDiff = 231.99987
xPoint = 12.0  TargetDiff = 236.00000  PredictDiff = 235.99949
xPoint = 13.0  TargetDiff = 230.0          PredictDiff = 230.00122
xPoint = 14.0  TargetDiff = 220.00000  PredictDiff = 219.99767
xPoint = 15.0  TargetDiff = 207.0          PredictDiff = 206.99951
xPoint = 16.0  TargetDiff = 190.00000  PredictDiff = 190.00221
xPoint = 17.0  TargetDiff = 180.00000  PredictDiff = 180.00009
xPoint = 18.0  TargetDiff = 170.00000  PredictDiff = 169.99977
xPoint = 19.0  TargetDiff = 160.00000  PredictDiff = 159.98978
xPoint = 20.0  TargetDiff = 150.00000  PredictDiff = 150.07543
xPoint = 21.0  TargetDiff = 140.00000  PredictDiff = 139.89404
xPoint = 22.0  TargetDiff = 141.0          PredictDiff = 140.99714
xPoint = 23.0  TargetDiff = 150.00000  PredictDiff = 149.99875
xPoint = 24.0  TargetDiff = 159.99999  PredictDiff = 159.99929
xPoint = 25.0  TargetDiff = 180.00000  PredictDiff = 179.99896
xPoint = 26.0  TargetDiff = 219.99999  PredictDiff = 219.99909
xPoint = 27.0  TargetDiff = 240.00000  PredictDiff = 240.00141
xPoint = 28.0  TargetDiff = 260.00000  PredictDiff = 259.99865
xPoint = 29.0  TargetDiff = 264.99999  PredictDiff = 264.99938
xPoint = 30.0  TargetDiff = 269.99999  PredictDiff = 270.00068
xPoint = 31.0  TargetDiff = 272.00000  PredictDiff = 271.99931
xPoint = 32.0  TargetDiff = 273.0          PredictDiff = 272.99969
xPoint = 33.0  TargetDiff = 268.99999  PredictDiff = 268.99975
xPoint = 34.0  TargetDiff = 266.99999  PredictDiff = 266.99994
xPoint = 35.0  TargetDiff = 264.99999  PredictDiff = 264.99742
```

xPoint = 36.0 TargetDiff = 253.99999 PredictDiff = 254.00076
xPoint = 37.0 TargetDiff = 239.99999 PredictDiff = 240.02203
xPoint = 38.0 TargetDiff = 225.00000 PredictDiff = 225.00479
xPoint = 39.0 TargetDiff = 210.00000 PredictDiff = 210.03944
xPoint = 40.0 TargetDiff = 200.99999 PredictDiff = 200.86493
xPoint = 41.0 TargetDiff = 195.0 PredictDiff = 195.11291
xPoint = 42.0 TargetDiff = 185.00000 PredictDiff = 184.91010
xPoint = 43.0 TargetDiff = 175.00000 PredictDiff = 175.02804
xPoint = 44.0 TargetDiff = 165.00000 PredictDiff = 165.07052
xPoint = 45.0 TargetDiff = 150.00000 PredictDiff = 150.01101
xPoint = 46.0 TargetDiff = 133.00000 PredictDiff = 132.91352
xPoint = 47.0 TargetDiff = 121.00000 PredictDiff = 121.00125
xPoint = 48.0 TargetDiff = 109.99999 PredictDiff = 110.02157
xPoint = 49.0 TargetDiff = 100.00000 PredictDiff = 100.01322
xPoint = 50.0 TargetDiff = 102.99999 PredictDiff = 102.98510
maxErrorPerc = 0.07574160995391013
averErrorPerc = 0.01071011328541703

Gambar 17.5 menunjukkan grafik hasil pelatihan.



Gambar 17.5 Bagan hasil pelatihan/validasi

Pengujian Jaringan

Pertama, Anda mengubah data konfigurasi untuk memproses logika pengujian. Anda memuat jaringan terlatih yang disimpan sebelumnya. Perhatikan di sini bahwa Anda tidak memuat kumpulan data pengujian. Anda akan menentukan nilai fungsi saat ini dengan cara berikut. Pada langkah pertama loop, nilai fungsi saat ini sama dengan variabel lastFunctionValueForTraining, yang dihitung selama proses pelatihan. Variabel ini menyimpan nilai fungsi pada titik terakhir, yaitu 50. Pada semua langkah loop berikut, Anda

menetapkan nilai record saat ini ke nilai fungsi yang dihitung selama langkah loop sebelumnya.

Menjelang awal contoh ini saya menjelaskan bagaimana Anda akan menghitung nilai prediksi selama fase pengujian. Saya ulangi penjelasan ini di sini:

> *"Selama pengujian, Anda akan menghitung nilai fungsi hari berikutnya dengan cara berikut. Dengan berada di titik x = 50, Anda ingin menghitung nilai prediksi fungsi pada titik berikutnya, x = 51. Mengumpankan perbedaan antara nilai xPoint pada titik saat ini dan sebelumnya (bidang 1) ke jaringan yang dilatih, Anda akan mendapatkan kembali perbedaan yang diprediksi antara nilai fungsi pada titik berikutnya dan saat ini. Oleh karena itu, nilai fungsi yang diprediksi pada titik berikutnya sama dengan jumlah nilai fungsi aktual pada titik saat ini dan perbedaan nilai prediksi yang diperoleh dari jaringan yang dilatih.*

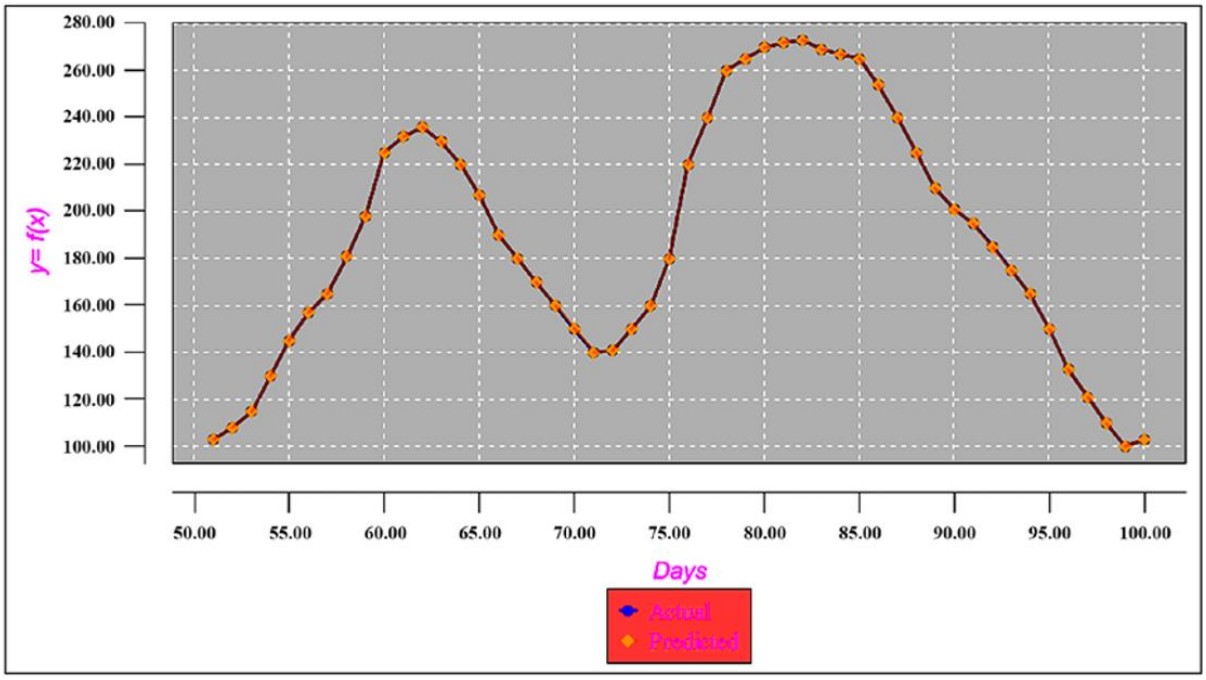
Selanjutnya, Anda mengulang set data pasangan mulai dari xPoint 51 (titik pertama dari interval pengujian). Pada setiap langkah loop, Anda mendapatkan nilai input, aktual, dan prediksi untuk setiap record, mendenormalisasi nilainya, dan menghitung realTargetDiffValue dan realPredictDiffValue untuk setiap record. Anda mencetak nilainya sebagai log pengujian dan mengisi elemen bagan dengan data untuk setiap rekaman. Terakhir, Anda menyimpan file grafik yang dihasilkan. Daftar 17-5 menunjukkan hasil pemrosesan pengujian.

Daftar 17-5. Hasil Pengujian

```
xPoint = 51.0  TargetDiff = 102.99999  PredictedDiff = 102.98510
xPoint = 52.0  TargetDiff = 107.98510  PredictedDiff = 107.98461
xPoint = 53.0  TargetDiff = 114.98461  PredictedDiff = 114.98322
xPoint = 54.0  TargetDiff = 129.98322  PredictedDiff = 129.98470
xPoint = 55.0  TargetDiff = 144.98469  PredictedDiff = 144.98371
xPoint = 56.0  TargetDiff = 156.98371  PredictedDiff = 156.98383
xPoint = 57.0  TargetDiff = 164.98383  PredictedDiff = 164.98232
xPoint = 58.0  TargetDiff = 180.98232  PredictedDiff = 180.98241
xPoint = 59.0  TargetDiff = 197.98241  PredictedDiff = 197.98225
xPoint = 60.0  TargetDiff = 224.98225  PredictedDiff = 224.98139
xPoint = 61.0  TargetDiff = 231.98139  PredictedDiff = 231.98127
xPoint = 62.0  TargetDiff = 235.98127  PredictedDiff = 235.98077
xPoint = 63.0  TargetDiff = 229.98077  PredictedDiff = 229.98199
xPoint = 64.0  TargetDiff = 219.98199  PredictedDiff = 219.97966
xPoint = 65.0  TargetDiff = 206.97966  PredictedDiff = 206.97917
xPoint = 66.0  TargetDiff = 189.97917  PredictedDiff = 189.98139
xPoint = 67.0  TargetDiff = 179.98139  PredictedDiff = 179.98147
xPoint = 68.0  TargetDiff = 169.98147  PredictedDiff = 169.98124
xPoint = 69.0  TargetDiff = 159.98124  PredictedDiff = 159.97102
xPoint = 70.0  TargetDiff = 149.97102  PredictedDiff = 150.04646
xPoint = 71.0  TargetDiff = 140.04646  PredictedDiff = 139.94050
xPoint = 72.0  TargetDiff = 140.94050  PredictedDiff = 140.93764
xPoint = 73.0  TargetDiff = 149.93764  PredictedDiff = 149.93640
xPoint = 74.0  TargetDiff = 159.93640  PredictedDiff = 159.93569
xPoint = 75.0  TargetDiff = 179.93573  PredictedDiff = 179.93465
xPoint = 76.0  TargetDiff = 219.93465  PredictedDiff = 219.93374
xPoint = 77.0  TargetDiff = 239.93374  PredictedDiff = 239.93515
xPoint = 78.0  TargetDiff = 259.93515  PredictedDiff = 259.93381
xPoint = 79.0  TargetDiff = 264.93381  PredictedDiff = 264.93318
xPoint = 80.0  TargetDiff = 269.93318  PredictedDiff = 269.93387
xPoint = 81.0  TargetDiff = 271.93387  PredictedDiff = 271.93319
```

xPoint = 82.0 TargetDiff = 272.93318 PredictedDiff = 272.93287
xPoint = 83.0 TargetDiff = 268.93287 PredictedDiff = 268.93262
xPoint = 84.0 TargetDiff = 266.93262 PredictedDiff = 266.93256
xPoint = 85.0 TargetDiff = 264.93256 PredictedDiff = 264.92998
xPoint = 86.0 TargetDiff = 253.92998 PredictedDiff = 253.93075
xPoint = 87.0 TargetDiff = 239.93075 PredictedDiff = 239.95277
xPoint = 88.0 TargetDiff = 224.95278 PredictedDiff = 224.95756
xPoint = 89.0 TargetDiff = 209.95756 PredictedDiff = 209.99701
xPoint = 90.0 TargetDiff = 200.99701 PredictedDiff = 200.86194
xPoint = 91.0 TargetDiff = 194.86194 PredictedDiff = 194.97485
maxGlobalResultDiff = 0.07571646804925916
averGlobalResultDiff = 0.01071236446121567

Gambar 17.6 menunjukkan grafik hasil pengujian.



Gambar 17.6 Bagan catatan hasil tes

Kedua grafik praktis tumpang tindih.

Menggali lebih dalam

Dalam contoh ini, Anda mempelajari bahwa dengan menggunakan beberapa teknik persiapan data khusus, Anda dapat menghitung nilai fungsi pada titik pertama dari interval berikutnya dengan mengetahui nilai fungsi pada titik terakhir dari interval pelatihan. Mengulangi proses ini untuk sisa poin dalam interval pengujian, Anda mendapatkan nilai fungsi untuk semua poin dari interval pengujian.

Mengapa saya selalu menyebutkan bahwa fungsinya harus periodik? Karena Anda menentukan nilai fungsi pada interval berikutnya berdasarkan hasil yang dihitung untuk interval pelatihan. Teknik ini juga dapat diterapkan pada fungsi nonperiodik. Satu-satunya persyaratan adalah bahwa nilai fungsi pada interval berikutnya dapat ditentukan dalam beberapa cara berdasarkan nilai dalam interval pelatihan. Misalnya, pertimbangkan fungsi di mana nilai pada interval berikutnya menggandakan nilai pada interval pelatihan. Fungsi seperti itu tidak periodik, tetapi teknik yang dibahas dalam bab ini akan berhasil. Juga, tidak

perlu bahwa setiap titik dalam interval berikutnya ditentukan oleh titik yang sesuai dalam interval latihan. Selama ada beberapa aturan untuk menentukan nilai fungsi di beberapa titik pada interval berikutnya berdasarkan nilai fungsi di beberapa titik pada interval pelatihan, teknik ini akan berhasil. Itu secara substansial meningkatkan kelas fungsi yang dapat diproses jaringan di luar interval pelatihan.

Tip

Bagaimana Anda mendapatkan batas kesalahan? di awal, cukup tebak nilai batas kesalahan dan jalankan proses pelatihan. jika Anda melihat saat mengulang zaman bahwa kesalahan jaringan dengan mudah menghapus batas kesalahan, kemudian kurangi batas kesalahan dan coba lagi. Terus turunkan nilai batas kesalahan sampai Anda melihat bahwa kesalahan jaringan dapat menghapus batas kesalahan; Namun, itu harus bekerja keras untuk melakukan ini.

Ketika Anda menemukan batas kesalahan seperti itu, cobalah bermain dengan arsitektur jaringan dengan mengubah jumlah lapisan tersembunyi dan jumlah neuron di dalam lapisan tersembunyi untuk melihat apakah mungkin untuk mengurangi batas kesalahan lebih banyak lagi. ingat bahwa untuk topologi fungsi yang lebih kompleks, menggunakan lebih banyak lapisan tersembunyi akan meningkatkan hasil. jika sambil meningkatkan jumlah lapisan tersembunyi dan jumlah neuron Anda mencapai titik ketika kesalahan jaringan menurun, hentikan proses ini dan kembali ke jumlah lapisan dan neuron sebelumnya.

17.6 RINGKASAN

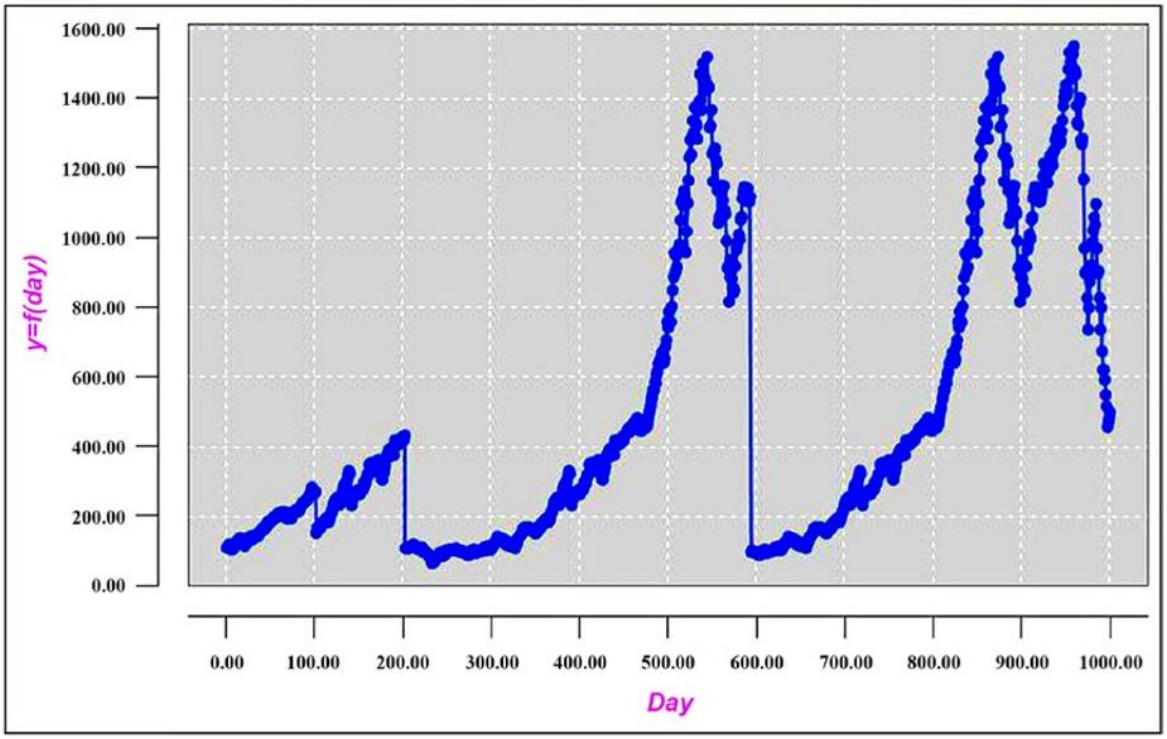
Dalam bab ini, Anda telah melihat cara memperkirakan fungsi periodik kompleks. Kumpulan data pelatihan dan pengujian ditransformasikan ke format catatan jendela geser untuk menambahkan informasi topologi fungsi ke data. Dalam bab berikutnya, saya akan membahas situasi yang lebih kompleks yang melibatkan aproksimasi fungsi nonkontinyu (yang saat ini merupakan masalah yang diketahui untuk aproksimasi jaringan saraf).

BAB 18
MENDEKATI FUNGSI TAK KONTINU

Bab ini akan membahas aproksimasi jaringan syaraf tiruan dari fungsi nonkontinyu. Saat ini, ini adalah area bermasalah untuk jaringan saraf karena pemrosesan jaringan didasarkan pada penghitungan turunan fungsi parsial (menggunakan algoritma penurunan gradien), dan menghitungnya untuk fungsi nonkontinyu pada titik di mana nilai fungsi tiba-tiba melompat atau turun menyebabkan hasil yang meragukan. Kami akan menggali lebih dalam masalah ini dalam bab ini. Bab ini juga mencakup metode yang saya kembangkan untuk memecahkan masalah ini.

18.1 CONTOH 5: MENDEKATI FUNGSI TAK KONTINU

Anda pertama-tama akan mencoba memperkirakan fungsi nonkontinyu (ditunjukkan pada Gambar 18.1) dengan menggunakan pemrosesan jaringan saraf konvensional sehingga Anda dapat melihat bahwa hasilnya berkualitas sangat rendah. Saya akan menjelaskan mengapa ini terjadi dan kemudian memperkenalkan metode yang memungkinkan Anda untuk memperkirakan fungsi tersebut dengan presisi yang baik.



Gambar 18.1 Bagan fungsi tak kontinu

Seperti yang Anda ingat dari bab sebelumnya, backpropagation jaringan saraf tiruan menggunakan turunan parsial dari fungsi kesalahan jaringan untuk mendistribusikan kembali kesalahan yang dihitung dari lapisan output ke semua neuron lapisan tersembunyi. Ini mengulangi proses iteratif ini dengan bergerak ke arah yang berlawanan dengan fungsi divergen untuk menemukan salah satu dari minimum fungsi kesalahan lokal (mungkin global). Karena masalah menghitung divergen/turunan untuk fungsi nonkontinu, pendekatan fungsi tersebut bermasalah.

Fungsi untuk contoh ini diberikan oleh nilainya pada 1.000 poin. Anda mencoba untuk memperkirakan fungsi ini menggunakan proses backpropagation jaringan saraf tradisional. Tabel 18.1 menunjukkan fragmen dari kumpulan data input.

Tabel 18.1 Fragmen dari Kumpulan Data Input

xPoint	yValue	xPoint	yValue	xPoint	yValue
26	146.568	56	200.337	86	236.364
27	140.847	57	197.229	87	236.272
28	139.791	58	201.805	88	238.42
29	131.033	59	206.756	89	241.18
30	136.216	60	205.89	90	242.341

Kumpulan data ini perlu dinormalisasi pada interval [-1,1]. Tabel 18.2 menunjukkan fragmen dari kumpulan data input yang dinormalisasi.

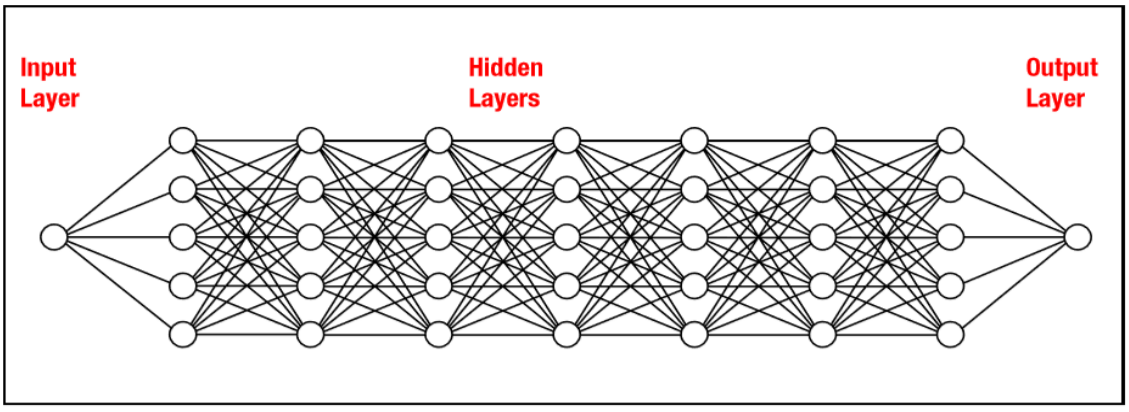
Tabel 18.2 Fragmen dari Kumpulan Data Input yang Dinormalisasi

xPoint	yValue	xPoint	yValue	xPoint	yValue
-1	-0.93846	-0.93994	-0.89879	-0.87988	-0.81883
-0.998	-0.93448	-0.93794	-0.89525	-0.87788	-0.80461
-0.996	-0.92605	-0.93594	-0.89082	-0.87588	-0.80936
-0.99399	-0.92381	-0.93393	-0.89452	-0.87387	-0.80708
-0.99199	-0.93644	-0.93193	-0.894	-0.87187	-0.80424
-0.98999	-0.94406	-0.92993	-0.8927	-0.86987	-0.80999
-0.98799	-0.93486	-0.92793	-0.88336	-0.86787	-0.8047
-0.98599	-0.94165	-0.92593	-0.88121	-0.86587	-0.82179
-0.98398	-0.92971	-0.92392	-0.87413	-0.86386	-0.82857
-0.98198	-0.92425	-0.92192	-0.87045	-0.86186	-0.80788
-0.97998	-0.9246	-0.91992	-0.86614	-0.85986	-0.80774
-0.97798	-0.91459	-0.91792	-0.86931	-0.85786	-0.81248
-0.97598	-0.91089	-0.91592	-0.859	-0.85586	-0.82594
-0.97397	-0.90556	-0.91391	-0.86585	-0.85385	-0.82698

-0.97197	-0.90574	-0.91191	-0.85954	-0.85185	-0.81042
-0.96997	-0.90083	-0.90991	-0.8498	-0.84985	-0.8097
-0.96797	-0.90322	-0.90791	-0.8434	-0.84785	-0.80559
-0.96597	-0.90641	-0.90591	-0.83788	-0.84585	-0.80534
-0.96396	-0.91009	-0.9039	-0.84642	-0.84384	-0.79564
-0.96196	-0.91962	-0.9019	-0.83644	-0.84184	-0.79598
-0.95996	-0.93098	-0.8999	-0.83476	-0.83984	-0.79633
-0.95796	-0.91516	-0.8979	-0.83325	-0.83784	-0.8018
-0.95596	-0.91598	-0.8959	-0.82811	-0.83584	-0.79236
-0.95395	-0.9146	-0.89389	-0.82335	-0.83383	-0.78716
-0.95195	-0.90748	-0.89189	-0.82719	-0.83183	-0.78196
-0.94995	-0.90056	-0.88989	-0.81774	-0.82983	-0.77096
-0.94795	-0.895	-0.88789	-0.82178	-0.82783	-0.77108
-0.94595	-0.89638	-0.88589	-0.81584	-0.82583	-0.76829
-0.94394	-0.90775	-0.88388	-0.80941	-0.82382	-0.7647
-0.94194	-0.90102	-0.88188	-0.81053	-0.82182	-0.76319

18.2 ARSITEKTUR JARINGAN

Jaringan untuk contoh ini terdiri dari lapisan input dengan satu neuron, tujuh lapisan tersembunyi (masing-masing dengan lima neuron), dan lapisan output dengan satu neuron. Lihat Gambar 18.2.



Gambar 18.2 Arsitektur jaringan

Kode Program

Daftar 18-1 menunjukkan kode program.

Daftar 18-1. Kode Program

```
// ============================================================
// Perkiraan fungsi non-kontinyu yang nilainya diberikan
// pada 999 poin. File input dinormalisasi.
// ============================================================
package sample5;
import java.io.BufferedReader;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate; import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList; import java.util.Calendar;
import java.util.Date; import java.util.List;
import java.util.Locale; import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
```

```
import org.knowm.xchart.SwingWrapper;
public class Sample5 implements ExampleChart<XYChart>
{
	// Interval to normalize
	static double Nh =  1;
	static double Nl = -1;
	// First column   static double minXPointDl = 1.00;
	static double maxXPointDh = 1000.00;
	// Second column - target data
	static double minTargetValueDl = 60.00;
	static double maxTargetValueDh = 1600.00;
	static double doublePointNumber = 0.00;
	static int intPointNumber = 0;
	static InputStream input = null;
	static double[] arrPrices = new double[2500];
	static double normInputXPointValue = 0.00;
	static double normPredictXPointValue = 0.00;
	static double normTargetXPointValue = 0.00;
	static double normDifferencePerc = 0.00;
	static double returnCode = 0.00;
	static double denormInputXPointValue = 0.00;
	static double denormPredictXPointValue = 0.00;
	static double denormTargetXPointValue = 0.00;
	static double valueDifference = 0.00;
	static int numberOfInputNeurons;
	static int numberOfOutputNeurons;
	static int intNumberOfRecordsInTestFile;
	static String trainFileName;
	static String priceFileName;
	static String testFileName;
	static String chartTrainFileName;
	static String chartTestFileName;
	static String networkFileName;
	static int workingMode;   static String cvsSplitBy = ",";
	static List<Double> xData = new ArrayList<Double>();
	static List<Double> yData1 = new ArrayList<Double>();
	static List<Double> yData2 = new ArrayList<Double>();
	static XYChart Chart;
	@Override  public XYChart getChart()
	{
	// Create Chart
	Chart = new XYChartBuilder().width(900).height(500).title(getClass().
	getSimpleName()).xAxisTitle("x").yAxisTitle("y= f(x)"). build();
	//                                  Customize                                  Chart
	Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor
	(ChartColor.GREY));
	Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
	Chart.getStyler().setChartBackgroundColor(Color.WHITE);
```

```
Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,  Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF,  Font.ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF,  Font.PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
//Chart.getStyler().setLocale(Locale.GERMAN);
Try
{
// Configuration
        // Set the mode to run the program
        workingModee = 1;
        // Training mode
        if( workingMode == 1)
        {
        // Training mode                                          trainFileName =
        "C:/Book_Examples/Sample5_Train_ Norm.csv";
        chartTrainFileName = "XYLine_Sample5_Train_Chart_Results";
        }
        else
        {
        // Testing mode                intNumberOfRecordsInTestFile = 3;
        testFileName          =          "C:/Book_Examples/Sample2_Norm.csv";
        chartTestFileName = "XYLine_Test_Results_Chart";
        }
        // Common part of config data                    networkFileName =
        "C:/Book_Examples/Sample5_Saved_Network_File.csv";
        numberOfInputNeurons = 1;       numberOfOutputNeurons = 1;
        // Check the working mode to run
        if(workingMode == 1)
        {
        // Training mode
        File file1 = new File(chartTrainFileName);
        File file2 = new File(networkFileName);
        if(file1.exists())
        file1.delete();
```

```
                if(file2.exists())
                file2.delete();
                returnCode = 0;
                // Clear the error Code
                do
                {
                returnCode = trainValidateSaveNetwork();
                }  while (returnCode > 0);
                }
                else
                {
                // Test mode
                loadAndTestNetwork();
                }
                }
                catch (Throwable t)
                {
                t.printStackTrace();
                System.exit(1);
                }
                finally
                {
                Encog.getInstance().shutdown();
                }
                Encog.getInstance().shutdown();
                return Chart;
                }  // End of the method
// ========================================================
// Load CSV to memory.
// @return The loaded dataset.
// ========================================================
public static MLDataSet loadCSV2Memory(String filename, int input,  int ideal, boolean
headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// ========================================================
// The main method.
// @param Command line arguments. No arguments are used.
// ========================================================
public static void main(String[] args)
{
ExampleChart<XYChart> exampleChart = new Sample5();
XYChart Chart = exampleChart.getChart();
```

```
new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ===========================================================
// Metode ini melatih, memvalidasi, dan menyimpan file
// jaringan yang terlatih
// ===========================================================
static public double trainValidateSaveNetwork()
{
        // Load the training CSV file in memory
        MLDataSet trainingSet =
        loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutput
        Neurons,true,CSVFormat.ENGLISH,false);
        // create a neural network     BasicNetwork network = new BasicNetwork();
        // Input layer
        network.addLayer(new BasicLayer(null,true,1));
        // Hidden layer
        network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
        network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));     network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));     network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));     network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));     network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));     network.addLayer(new
        BasicLayer(new ActivationTANH(),true,5));
        // Output layer                network.addLayer(new BasicLayer(new
        ActivationTANH(),false,1));
        network.getStructure().finalizeStructure();     network.reset();
        // train the neural network             final ResilientPropagation train = new
        ResilientPropagation (network, trainingSet);
        int epoch = 1;
        do
        {
        train.iteration();              System.out.println("Epoch #" + epoch + " Error:" +
        train.getError());
        epoch++;
        if (epoch >= 11000 && network.calculateError(trainingSet) > 0.00225)
        {
        returnCode = 1;
        System.out.println("Try again");
        return returnCode;
        }
        } while(train.getError() > 0.0022);
        // Save the network file             EncogDirectoryPersistence.saveObject(new
        File(networkFileName),network);
        System.out.println("Neural Network Results:");
        double sumNormDifferencePerc = 0.00;
        double averNormDifferencePerc = 0.00;
        double maxNormDifferencePerc = 0.00;
```

```
    int m = 0;
    double xPointer = 0.00;
for(MLDataPair pair: trainingSet)
{
    m++;
    xPointer++;
    final MLData output = network.compute(pair.getInput());
    MLData inputData = pair.getInput();
    MLData actualData = pair.getIdeal();
    MLData predictData = network.compute(inputData);
    // Calculate and print the results
    normInputXPointValue = inputData.getData(0);
    normTargetXPointValue = actualData.getData(0);
    normPredictXPointValue = predictData.getData(0);
    denormInputXPointValue = ((minXPointDl - maxXPointDh)* normInputXPointValue
    - Nh*minXPointDl + maxXPointDh *Nl)/ (Nl - Nh);
    denormTargetXPointValue =((minTargetValueDl - maxTargetValueDh)*
    normTargetXPointValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/ (Nl
    - Nh);
    denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
    normPredictXPointValue - Nh*minTargetValueDl + maxTargetValue Dh*Nl)/(Nl -
    Nh);
    valueDifference = Math.abs(((denormTargetXPointValue -
    denormPredictXPointValue)/ d enormTargetXPointValue)*100.00);
    S ystem.out.println ("RecordNumber = " + m + "  denormTargetX PointValue = " +
    denormTargetXPointValue + " denormPredictX PointValue = " +
    denormPredictXPointValue + "  value Difference = " + valueDifference);
    sumNormDifferencePerc = sumNormDifferencePerc + valueDifference;
    if (valueDifference > maxNormDifferencePerc) maxNormDifferencePerc =
    valueDifference;
    xData.add(xPointer);
    yData1.add(denormTargetXPointValue);
    yData2.add(denormPredictXPointValue);
}  // End for pair loop
XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
try
{
//Save the chart image BitmapEncoder.saveBitmapWithDPI(Chart,
chartTrainFileName, BitmapFormat.JPG, 100);
System.out.println ("Train Chart file has been saved") ;
}
catch (IOException ex)
{
```

```
    ex.printStackTrace();
    System.exit(3);
    }
    // Finally, save this trained network          EncogDirectoryPersistence.saveObject(new
    File(networkFileName), network);
    System.out.println ("Train Network has been saved") ;
    averNormDifferencePerc  = sumNormDifferencePerc/1000.00;
    System.out.println("  ");              System.out.println("maxNormDifferencePerc  =  " +
    maxNormDifference Perc + "  averNormDifferencePerc = " + averNormDifferencePerc);
    returnCode = 0.00;
    return returnCode;
}  // End of the method
// ============================================================
// This method load and test the trained network
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double targetToPredictPercent = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;    double maxGlobalIndex = 0;
        double normInputXPointValueFromRecord = 0.00;
        double normTargetXPointValueFromRecord = 0.00;
        double normPredictXPointValueFromRecord = 0.00;
        BasicNetwork network;
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        // Load the test dataset into memory
        MLDataSet  testingSet  =  loadCSV2Memory(testFileName,  numberOfInput
        Neurons,numberOfOutputNeurons,true,CSVFormat.ENGLISH,false);
        //  Load  the  saved  trained  network                network  =
        (BasicNetwork)EncogDirectoryPersistence.loadObject(new                File
        (networkFileName));
        int i = - 1;
        // Index of the current record    double xPoint = -0.00;
        for (MLDataPair pair:  testingSet)
        {
                i++;
                xPoint = xPoint + 2.00;
                MLData inputData = pair.getInput();
                MLData actualData = pair.getIdeal();
                MLData predictData = network.compute(inputData);
                normInputXPointValueFromRecord = inputData.getData(0);
```

```
            normTargetXPointValueFromRecord = actualData.getData(0);
            normPredictXPointValueFromRecord = predictData.getData(0);
            // De-normalize them
            denormInputXPointValue = ((minXPointDl - maxXPointDh)*
            normInputXPointValueFromRecord - Nh*minXPointDl +
            maxXPointDh*Nl)/(Nl - Nh);
            denormTargetXPointValue = ((minTargetValueDl - maxTargetValueDh)*
            normTargetXPointValueFromRecord - Nh*minTargetValueDl +
            maxTargetValueDh*Nl)/(Nl - Nh);
            denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
            normPredictXPointValueFromRecord - Nh*minTargetValueDl +
            maxTargetValueDh*Nl)/(Nl - Nh);
            targetToPredictPercent = Math.abs((denormTargetXPointValue -
            denormPredictXPointValue)/denormTargetXPointValue*100);
            System.out.println("xPoint = " + xPoint + "  denormTargetX PointValue = " +
            denormTargetXPointValue + "  denormPredictX PointValue = " +
            denormPredictXPointValue + "  targetToPredict Percent = " +
            targetToPredictPercent);
            if (targetToPredictPercent > maxGlobalResultDiff) maxGlobalResultDiff
            = targetToPredictPercent;
            sumGlobalResultDiff = sumGlobalResultDiff + targetToPredictPercent;
            // Populate chart elements
            xData.add(xPoint);
            yData1.add(denormTargetXPointValue);
            yData2.add(denormPredictXPointValue);
        } // End for pair loop
        // Print the max and average results     System.out.println(" ");
        averGlobalResultDiff = sumGlobalResultDiff/intNumberOfRecordsInTestFile;
        System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff + "  i = " +
        maxGlobalIndex);              System.out.println("averGlobalResultDiff = " +
        averGlobalResultDiff);
        // All testing batch files have been processed
        XYSeries series1 = Chart.addSeries("Actual", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image     try
        {
        BitmapEncoder.saveBitmapWithDPI(Chart, chartTestFileName, BitmapFormat.JPG,
        100);
        }
        catch (Exception bt)
        {
        bt.printStackTrace();
        }
        System.out.println ("The Chart has been saved");
```

```
        System.out.println("End of testing for test records");
    } // End of the method
} // End of the class
```

18.3 FRAGMEN KODE UNTUK PROSES PELATIHAN

Metode pelatihan dipanggil dalam satu lingkaran sampai berhasil menghapus batas kesalahan. Anda memuat file pelatihan yang dinormalisasi dan kemudian membuat jaringan dengan satu lapisan input (satu neuron), tujuh lapisan tersembunyi (masing-masing dengan lima neuron), dan lapisan output (satu neuron). Selanjutnya, Anda melatih jaringan dengan mengulang zaman sampai kesalahan jaringan menghapus batas kesalahan. Pada saat itu, Anda keluar dari loop. Jaringan dilatih, dan Anda menyimpannya di disk (ini akan digunakan oleh metode pengujian). Daftar 18-2 menunjukkan bagian dari metode pelatihan.

Daftar 18-2. Fragmen Kode Metode Pelatihan

```
// Load the training CSV file in memory
MLDataSet trainingSet = loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
// create a neural network
BasicNetwork network = new BasicNetwork();
// Input layer
network.addLayer(new BasicLayer(null,true,1));
// Hidden layer
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
// Output layer
network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// train the neural network final ResilientPropagation train = new ResilientPropagation(network, trainingSet);
int epoch = 1;
do
        {
                train.iteration();
                System.out.println("Epoch #" + epoch + " Error:" + train.getError());
                epoch++;
                if (epoch >= 11000 && network.calculateError(trainingSet) > 0.00225)
                // 0.0221   0.00008
                {
                returnCode = 1;
                System.out.println("Try again");
                return returnCode;
```

```
            }
            } while(train.getError() > 0.0022);
        // Save the network file
        EncogDirectoryPersistence.saveObject(new File(networkFileName),network);
```

Selanjutnya Anda mengulang set data pasangan dan mengambil dari jaringan nilai input, aktual, dan prediksi untuk setiap record. Anda kemudian mendenormalisasi nilai yang diambil, memasukkannya ke dalam log, dan mengisi data bagan.

```
int m = 0;
double xPointer = 0.00;
for(MLDataPair pair: trainingSet)
{
m++;
xPointer++;
final MLData output = network.compute(pair.getInput());
MLData inputData = pair.getInput();
MLData actualData = pair.getIdeal();
MLData predictData = network.compute(inputData);
        // Calculate and print the results
        normInputXPointValue = inputData.getData(0);
        normTargetXPointValue = actualData.getData(0);
        normPredictXPointValue = predictData.getData(0);
        denormInputXPointValue = ((minXPointDl - maxXPointDh)*normInputX PointValue
        - Nh*minXPointDl + maxXPointDh *Nl)/(Nl - Nh);
        denormTargetXPointValue =((minTargetValueDl - maxTargetValueDh)*
        normTargetXPointValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/ (Nl -
        Nh);
        denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
        normPredictXPointValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/ (Nl
        - Nh);
        valueDifference = Math.abs(((denormTargetXPointValue - denormPredictX
        PointValue)/denormTargetXPointValue)*100.00);
        System.out.println ("RecordNumber = " + m + "  denormTargetX PointValue = " +
        denormTargetXPointValue + "  denormPredictXPoint Value = " +
        denormPredictXPointValue + "  valueDifference = " + valueDifference);
        sumNormDifferencePerc = sumNormDifferencePerc + valueDifference;
        if (valueDifference > maxNormDifferencePerc) maxNormDifferencePerc =
        valueDifference;
        xData.add(xPointer);
        yData1.add(denormTargetXPointValue);
        yData2.add(denormPredictXPointValue);
}  // End for pair loop
```

Terakhir, Anda menghitung nilai rata-rata dan maksimum dari hasil dan menyimpan file bagan.

```
XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
```

```
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        try  {      //Save the chart image      B itmapEncoder.saveBitmapWithDPI(Chart,
        chartTrainFileName, BitmapFormat.JPG, 100);
        System.out.println ("Train Chart file has been saved") ;
        }
        catch (IOException ex)
        {
        ex.printStackTrace();
        System.exit(3);
        }
        // Save this trained network      EncogDirectoryPersistence.saveObject(new
        File(networkFileName),network);
        System.out.println ("Train Network has been saved") ;
        averNormDifferencePerc  = sumNormDifferencePerc/1000.00;
        System.out.println(" ");
        System.out.println("maxNormDifferencePerc = " + maxNormDifferencePerc + "
        averNormDifferencePerc = " + averNormDifferencePerc);
        returnCode = 0.00;
        return returnCode;
} // End of the method
```

Hasil Pelatihan Tidak Memuaskan

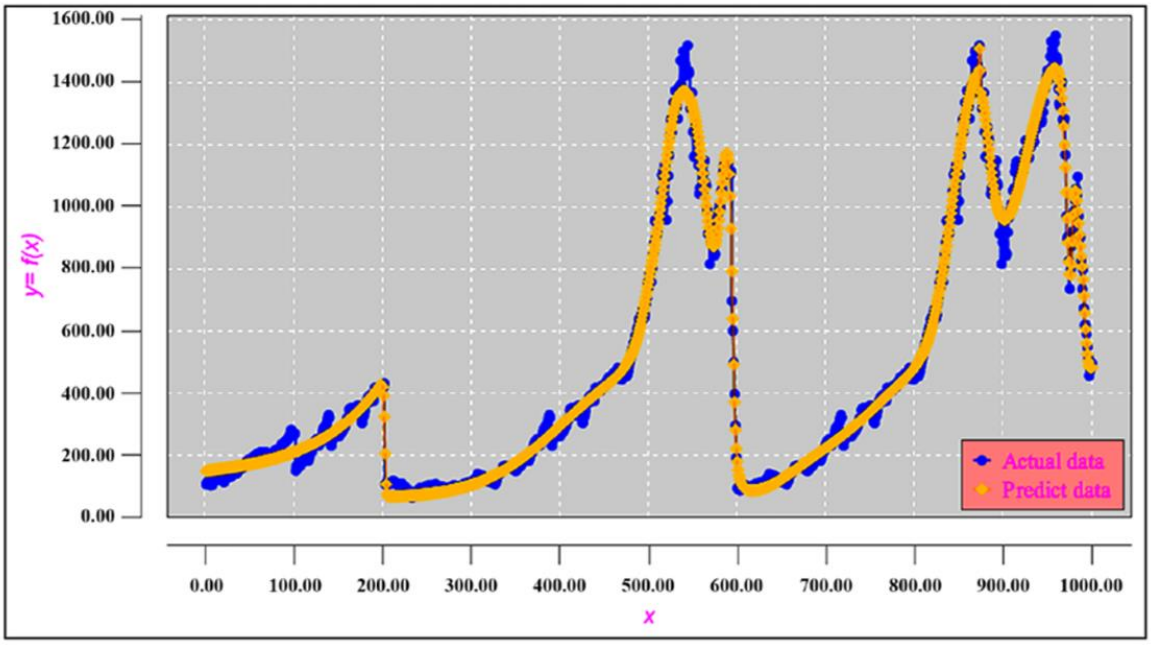
Daftar 18-3 menunjukkan bagian akhir dari hasil pelatihan.

Daftar 18-3. Mengakhiri Fragmen Hasil Pelatihan

```
RecordNumber =  983  TargetValue = 1036.19  PredictedValue = 930.03102    DiffPerc = 10.24513
RecordNumber =  984  TargetValue = 1095.63  PredictedValue = 915.36958    DiffPerc = 16.45267
RecordNumber =  985  TargetValue = 968.75   PredictedValue = 892.96942    DiffPerc = 7.822511
RecordNumber =  986  TargetValue = 896.24   PredictedValue = 863.64775    DiffPerc = 3.636554
RecordNumber =  987  TargetValue = 903.25   PredictedValue = 829.19287    DiffPerc = 8.198962
RecordNumber =  988  TargetValue = 825.88   PredictedValue = 791.96691    DiffPerc = 4.106298
RecordNumber =  989  TargetValue = 735.09   PredictedValue = 754.34279    DiffPerc = 2.619107
RecordNumber =  990  TargetValue = 797.87   PredictedValue = 718.23458    DiffPerc = 9.981002
RecordNumber =  991  TargetValue = 672.81   PredictedValue = 684.88576    DiffPerc = 1.794825
RecordNumber =  992  TargetValue = 619.14   PredictedValue = 654.90309    DiffPerc = 5.776254
RecordNumber =  993  TargetValue = 619.32   PredictedValue = 628.42044    DiffPerc = 1.469424
RecordNumber =  994  TargetValue = 590.47   PredictedValue = 605.28210    DiffPerc = 2.508528
RecordNumber =  995  TargetValue = 547.28   PredictedValue = 585.18808    DiffPerc = 6.926634
RecordNumber =  996  TargetValue = 514.62   PredictedValue = 567.78844    DiffPerc = 10.33159
RecordNumber =  997  TargetValue = 455.4    PredictedValue = 552.73603    DiffPerc = 21.37374
RecordNumber =  998  TargetValue = 470.43   PredictedValue = 539.71156    DiffPerc = 14.72728
RecordNumber =  999  TargetValue = 480.28   PredictedValue = 528.43269    DiffPerc = 10.02596
RecordNumber = 1000  TargetValue = 496.77   PredictedValue = 518.65485    DiffPerc = 4.405429
maxNormDifferencePerc = 97.69386964911284
averNormDifferencePerc = 7.232624870097155
```

Perkiraan ini berkualitas rendah. Bahkan dengan jaringan yang dioptimalkan dengan baik, kesalahan perkiraan rata-rata untuk semua catatan lebih dari 8 persen, dan kesalahan perkiraan maksimum (catatan perkiraan terburuk) lebih dari 97 persen.

Pendekatan fungsi seperti itu tentu tidak dapat digunakan. Gambar 18.3 menunjukkan diagram hasil perkiraan.



Gambar 18.3 Perkiraan fungsi berkualitas rendah

Saya tahu bahwa pendekatan ini tidak akan berhasil dan menyatakan ini di awal contoh. Namun, hal itu sengaja dilakukan untuk menunjukkan maksudnya. Sekarang, saya akan menunjukkan bagaimana fungsi nonkontinyu ini dapat berhasil didekati dengan menggunakan jaringan saraf.

Masalah dengan aproksimasi fungsi ini adalah topologi fungsi (yang memiliki lompatan tiba-tiba atau penurunan nilai fungsi pada titik-titik tertentu). Jadi, Anda akan memecah file input menjadi serangkaian file input satu record yang disebut micro-batch. Ini mirip dengan pelatihan batch, tetapi di sini Anda secara aktif mengontrol ukuran batch. Dengan melakukan ini, Anda menghilangkan dampak negatif dari topologi fungsi yang sulit. Setelah memecah kumpulan data, setiap catatan akan diisolasi dan tidak terkait dengan nilai fungsi sebelumnya atau berikutnya. Memecah file input menjadi mikro-batch menghasilkan 1.000 file input, yang diproses jaringan satu per satu. Anda menautkan setiap jaringan terlatih dengan catatan yang diwakilinya. Selama proses validasi dan pengujian, logika menemukan jaringan terlatih yang paling cocok dengan bidang pertama dari rekaman pengujian atau validasi yang sesuai.

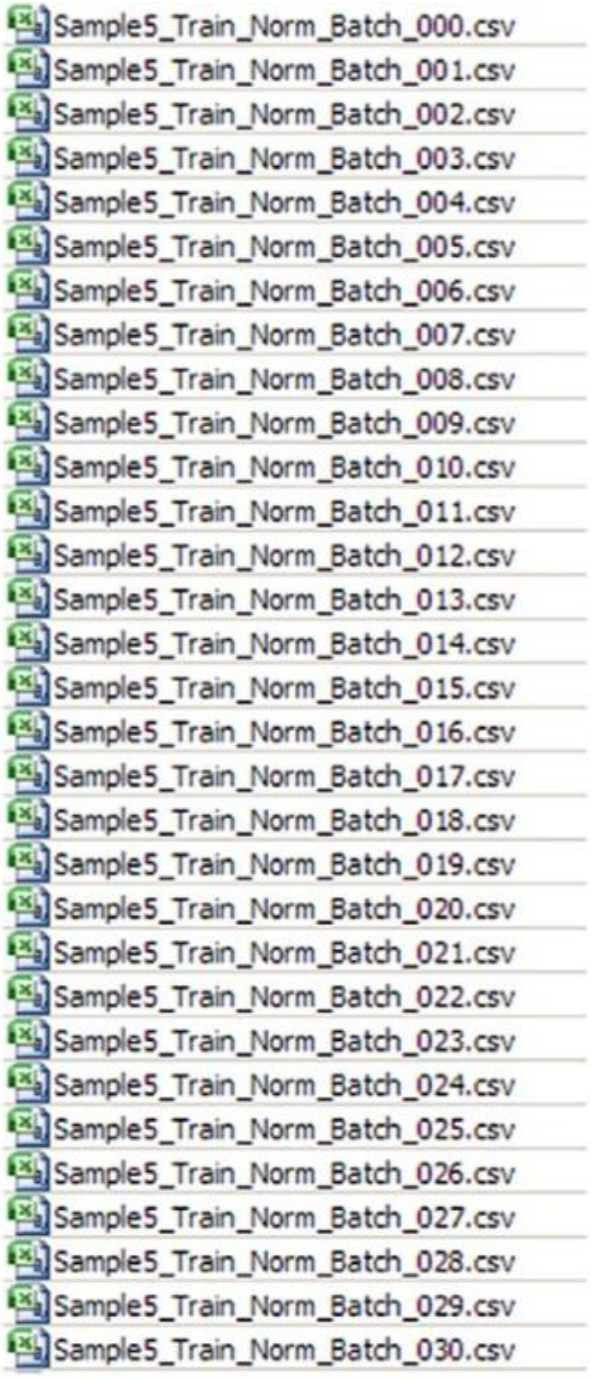
18.4 MENAKSIR FUNGSI TIDAK BERKELANJUTAN DENGAN METODE *MICRO-BATCH*

Mari kita pecahkan kumpulan data pelatihan yang dinormalisasi menjadi mikro-batch. Setiap kumpulan data mikro-batch harus berisi catatan label dan satu catatan dari file asli untuk diproses. Tabel 18.3 menunjukkan tampilan kumpulan data mikro.

Tabel 18.3 File Micro-Batch

xPoint	Nilai Funggsi
-1	-0.938458442

Di sini Anda menulis program sederhana untuk memecah kumpulan data pelatihan yang dinormalisasi menjadi mikro-batch. Sebagai hasil dari menjalankan program ini, Anda membuat 999 set data mikro-batch (dinomori dari 0 hingga 998). Gambar 18.4 menunjukkan fragmen daftar kumpulan data mikro.



Gambar 18.4 Fragmen daftar set data mikro-batch pelatihan yang dinormalisasi

Kumpulan kumpulan data mikro-batch ini sekarang menjadi input untuk melatih jaringan.

Kode Program untuk Pemrosesan *Micro-Batch*

Daftar 18-4 menunjukkan kode program.

Daftar 18-4. Kode Program

```
// ============================================================
// Perkiraan fungsi non-kontinyu menggunakan metode micro-batch.
// Input adalah kumpulan file mikro-batch yang dinormalisasi
```

```
// setiap mikro-batch
// menyertakan catatan satu hari).
// Setiap record terdiri dari: // - normDayValue
// - normTargetValue
//
// Jumlah neuron inputLayer adalah 12
// Jumlah neuron outputLayer adalah 1
//
// Perbedaan dari program ini adalah bahwa program ini secara mandiri melatih banyak
// jaringan satu hari. Itu memungkinkan pelatihan setiap
// jaringan harian menggunakan
// nilai parameter bobot/bias terbaik, sehingga mencapai hasil // optimalisasi terbaik untuk
setiap tahun.
//
// Setiap jaringan disimpan pada disk dan peta dibuat untuk
// menghubungkan setiap jaringan
// terlatih yang disimpan dengan file mikro-batch pelatihan
// yang sesuai.
// ============================================================
package sample5_microbatches;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
```

```
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.
resilient.  ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample5_Microbatches implements ExampleChart<XYChart>
{
        // Normalization parameters
        // Normalizing interval   static double Nh =  1;   static double Nl = -1;
        static double inputDayDh = 1000.00;
        static double inputDayDl = 1.00;
        static double targetFunctValueDiffPercDh = 1600.00;
        static double targetFunctValueDiffPercDl = 60.00;
        static String cvsSplitBy = ",";
        static Properties prop = null;
        static String strWorkingMode;
        static String strNumberOfBatchesToProcess;
        static String strTrainFileNameBase;
        static String strTestFileNameBase;
        static String strSaveTrainNetworkFileBase;
        static String strSaveTestNetworkFileBase;
        static String strValidateFileName;
        static String strTrainChartFileName;
        static String strTestChartFileName;
        static String strFunctValueTrainFile;
        static String strFunctValueTestFile;
        static int intDayNumber;
        static double doubleDayNumber;
        static int intWorkingMode;
        static int numberOfTrainBatchesToProcess;
```

```
static int numberOfTestBatchesToProcess;
static int intNumberOfRecordsInTrainFile;
static int intNumberOfRecordsInTestFile;
static int intNumberOfRowsInBatches;
static int intInputNeuronNumber;
static int intOutputNeuronNumber;
static String strOutputFileName;
static String strSaveNetworkFileName;
static String strDaysTrainFileName;
static XYChart Chart;
static String iString;
static double inputFunctValueFromFile;
static double targetToPredictFunctValueDiff;
static int[] returnCodes  = new int[3];
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static double[] DaysyearDayTraining = new double[1200];
static String[] strTrainingFileNames = new String[1200];
static String[] strTestingFileNames = new String[1200];
static String[] strSaveTrainNetworkFileNames = new String[1200];
static double[] linkToSaveNetworkDayKeys = new double[1200];
static double[] linkToSaveNetworkTargetFunctValueKeys = new double[1200];
static double[] arrTrainFunctValues = new double[1200];
static double[] arrTestFunctValues = new double[1200];
@Override
public XYChart getChart()
{
// Create Chart
        Chart = new XYChartBuilder().width(900).height(500).title(getClass().
        getSimpleName()).xAxisTitle("day").yAxisTitle("y=f(day)").build();
        // Customize Chart
        Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor(Chart
        Color.GREY));
        Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
        Chart.getStyler().setChartBackgroundColor(Color.WHITE);
        Chart.getStyler().setLegendBackgroundColor(Color.PINK);
        Chart.getStyler().setChartFontColor(Color.MAGENTA);
        Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
        Chart.getStyler().setChartTitleBoxVisible(true);
        Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
        Chart.getStyler().setPlotGridLinesVisible(true);
        Chart.getStyler().setAxisTickPadding(20);
        Chart.getStyler().setAxisTickMarkLength(15);
        Chart.getStyler().setPlotMargin(20);
        Chart.getStyler().setChartTitleVisible(false);
        Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,
        Font.BOLD, 24));
```

```
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.OutsideE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new  Font(Font.SANS_SERIF,  Font.ITALIC,
18));
Chart.getStyler().setAxisTickLabelsFont(new   Font(Font.SERIF,   Font.PLAIN,
11));
//Chart.getStyler().setDayPattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
// Config data
// Training mode
intWorkingMode = 0;
// Testing mode
numberOfTrainBatchesToProcess = 1000;
numberOfTestBatchesToProcess = 999;
intNumberOfRowsInBatches = 1;
intInputNeuronNumber = 1;
intOutputNeuronNumber = 1;
strTrainFileNameBase                                                    =
"C:/My_Neural_Network_Book/Temp_Files/Sample5_
Train_Norm_Batch_";
strTestFileNameBase                                                     =
"C:/My_Neural_Network_Book/Temp_Files/Sample5_ Test_Norm_Batch_";
strSaveTrainNetworkFileBase                                             =
"C:/Book_Examples/Sample5_Save_Network_    Batch_";
strTrainChartFileName   =   "C:/Book_Examples/Sample5_Chart_Train_File_
Microbatch.jpg";                                strTestChartFileName        =
"C:/Book_Examples/Sample5_Chart_Test_File_    Microbatch.jpg";
//  Generate training batch file names and the corresponding saveNetwork
file names    intDayNumber = -1;
// Day number for the chart
for (int i = 0;
i < numberOfTrainBatchesToProcess; i++)
{
intDayNumber++;       iString = Integer.toString(intDayNumber);
if (intDayNumber >= 10 & intDayNumber < 100  )
{
strOutputFileName = strTrainFileNameBase + "0" + iString + ".csv";
strSaveNetworkFileName = strSaveTrainNetworkFileBase + "0" +        iString
+ ".csv";
}
else
{
if (intDayNumber < 10)         {
strOutputFileName = strTrainFileNameBase + "00" + iString +             ".csv";
strSaveNetworkFileName    =    strSaveTrainNetworkFileBase    +    "00"    +
iString + ".csv";          }
else
```

```
                {
                strOutputFileName = strTrainFileNameBase + iString + “.csv”;
                strSaveNetworkFileName = strSaveTrainNetworkFileBase +          iString +
                ".csv";
                }
        }
        strTrainingFileNames[intDayNumber] = strOutputFileName;
        strSaveTrainNetworkFileNames[intDayNumber] =
        strSaveNetwork FileName;
} // End the FOR loop
// Build the array linkToSaveNetworkFunctValueDiffKeys
String tempLine;
double tempNormFunctValueDiff = 0.00;
double tempNormFunctValueDiffPerc = 0.00;
double tempNormTargetFunctValueDiffPerc = 0.00;
String[] tempWorkFields;     try
{
intDayNumber = -1;
// Day number for the chart
for (int m = 0; m < numberOfTrainBatchesToProcess; m++)
{
        intDayNumber++;
        BufferedReader      br3      =      new      BufferedReader(newFileReader
        (strTrainingFileNames[intDayNumber]));
        tempLine = br3.readLine();
        // Skip the label record and zero batch record          tempLine = br3.readLine();
        // Break the line using comma as separator                  tempWorkFields =
        tempLine.split(cvsSplitBy);
        tempNormFunctValueDiffPerc = Double.parseDouble (tempWorkFields[0]);
        tempNormTargetFunctValueDiffPerc = Double.parseDouble (tempWorkFields[1]);
        linkToSaveNetworkDayKeys[intDayNumber] = tempNormFunctValue DiffPerc;
        linkToSaveNetworkTargetFunctValueKeys[intDayNumber]                          =
        tempNormTargetFunctValueDiffPerc;
} // End the FOR loop
        // Generate testing batche file names       if(intWorkingMode == 1)
        {
        intDayNumber = -1;
        for (int i = 0;
        i < numberOfTestBatchesToProcess; i++)
        {
        intDayNumber++;
        iString = Integer.toString(intDayNumber);
        // Construct the testing batch names
        if (intDayNumber >= 10 & intDayNumber < 100  )
        {
        strOutputFileName = strTestFileNameBase + "0" +          iString + ".csv";
        }
        else
```

```
{
if (intDayNumber < 10)
{
strOutputFileName = strTestFileNameBase + "00" +                    iString + ".csv";
}
else
{
strOutputFileName = strTestFileNameBase +                    iString + ".csv";
}
}
strTestingFileNames[intDayNumber] = strOutputFileName;
}  // End the FOR loop
}   // End of IF
}    // End for try
catch (IOException io1)
{
io1.printStackTrace();
System.exit(1);
}
// Load, train, and test Function Values file in memory
//loadTrainFunctValueFileInMemory();
if(intWorkingMode == 0)
{
// Train mode          int paramErrorCode;
          int paramBatchNumber;
          int paramR;          int paramDayNumber;
          int paramS;
          File file1 = new File(strTrainChartFileName);
          if(file1.exists())            file1.delete();          returnCodes[0] = 0;
// Clear the error Code          returnCodes[1] = 0;
// Set the initial batch Number to 0;
returnCodes[2] = 0;
// Day number;         do
{
          paramErrorCode = returnCodes[0];
          paramBatchNumber = returnCodes[1];
          paramDayNumber = returnCodes[2];
          returnCodes      =          trainBatches(paramErrorCode,paramBatchNumber,
          paramDayNumber);
} while (returnCodes[0] > 0);
}
// End the train logic       else
{
// Testing mode          File file2 = new File(strTestChartFileName);
f(file2.exists())          file2.delete();
loadAndTestNetwork();
// End the test logic
}
```

```
        Encog.getInstance().shutdown();
        //System.exit(0);       return Chart;
    }  // End of method
// ========================================================
// Load CSV to memory.
// @return The loaded dataset.
// ========================================================
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal, boolean
headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();       return dataset;
}
// ========================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// ========================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample5_Microbatches();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ========================================================
// Metode ini melatih kumpulan sebagai jaringan1 individu
// menyimpannya dalam kumpulan data terlatih yang terpisah
// ========================================================
static public int[] trainBatches(int paramErrorCode,int paramBatch Number,int
paramDayNumber)
{
        int rBatchNumber;
        double targetToPredictFunctValueDiff = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double normInputFunctValueDiffPercFromRecord = 0.00;
        double normTargetFunctValue1 = 0.00;
        double normPredictFunctValue1 = 0.00;
        double denormInputDayFromRecord1;
        double denormInputFunctValueDiffPercFromRecord;
        double denormTargetFunctValue1 = 0.00;
        double denormAverPredictFunctValue11 = 0.00;
        BasicNetwork network1 = new BasicNetwork();
        // Input layer
        network1.addLayer(new BasicLayer(null,true,intInputNeuronNumber));
        // Hidden layer.
```

```
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
// Output layer
network1.addLayer(new BasicLayer(new ActivationTANH(),false,
intOutputNeuronNumber));
network1.getStructure().finalizeStructure();
network1.reset();
maxGlobalResultDiff = 0.00;
averGlobalResultDiff = 0.00;
sumGlobalResultDiff = 0.00;
// Loop over batches     intDayNumber = paramDayNumber;
// Day number for the chart
for (rBatchNumber = paramBatchNumber;
rBatchNumber < numberOfTrain BatchesToProcess; rBatchNumber++)
{
        intDayNumber++;
        // Load the training file in memory
        MLDataSet trainingSet = loadCSV2Memory(strTrainingFileNames
        [rBatchNumber],intInputNeuronNumber,intOutputNeuronNumber, true,
        CSVFormat.ENGLISH,false);
        // train the neural network1          ResilientPropagation train = new
        ResilientPropagation(network1, trainingSet);
        int epoch = 1;
        do
                {
        train.iteration();
        epoch++;
                for (MLDataPair pair11:  trainingSet)
                {
                MLData inputData1 = pair11.getInput();
                MLData actualData1 = pair11.getIdeal();
                MLData predictData1 = network1.compute(inputData1);
                // These values are Normalized as the whole input is
                normInputFunctValueDiffPercFromRecord =  inputData1.getData(0);
                normTargetFunctValue1 = actualData1.getData(0);
                normPredictFunctValue1 = predictData1.getData(0);
                denormInputFunctValueDiffPercFromRecord =((inputDayDl
                inputDayDh)*normInputFunctValueDiffPercFromRecord -
                Nh*inputDayDl + inputDayDh*Nl)/(Nl - Nh);
                denormTargetFunctValue1 = ((targetFunctValueDiffPercDl
                targetFunctValueDiffPercDh)*normTargetFunctValue1 - Nh*target
                FunctValueDiffPercDl + targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
```

```
                denormAverPredictFunctValue11 =((targetFunctValueDiffPercDl
                targetFunctValueDiffPercDh)*normPredictFunctValue1 - Nh*
                targetFunctValueDiffPercDl + targetFunctValueDiffPercDh*Nl)/
                (Nl - Nh);
                targetToPredictFunctValueDiff = (Math.abs(denormTarget
                FunctValue1 - denormAverPredictFunctValue11)/denormTarget
                FunctValue1)*100;
        }
        if (epoch >= 1000 && targetToPredictFunctValueDiff > 0.0000071)
                {
                returnCodes[0] = 1;
                returnCodes[1] =
                rBatchNumber;
                returnCodes[2] = intDayNumber-1;
                return returnCodes;
        }
} while(targetToPredictFunctValueDiff > 0.000007);
// This batch is optimized
// Save the network1 for the current batch
EncogDirectoryPersistence.saveObject(newFile(strSaveTrainNetwork
FileNames[rBatchNumber]),network1);
// Get the results after the network1 optimization     int i = - 1;
for (MLDataPair pair1:  trainingSet)
{
i++;
MLData inputData1 = pair1.getInput();
MLData actualData1 = pair1.getIdeal();
MLData predictData1 = network1.compute(inputData1);
// These values are Normalized as the whole input is
normInputFunctValueDiffPercFromRecord = inputData1.getData(0);
normTargetFunctValue1 = actualData1.getData(0);
normPredictFunctValue1 = predictData1.getData(0);
// De-normalize the obtained values
denormInputFunctValueDiffPercFromRecord =((inputDayDl - inputDayDh)*
normInputFunctValueDiffPercFromRecord - Nh*inputDayDl +
inputDayDh*Nl)/(Nl - Nh);
denormTargetFunctValue1 = ((targetFunctValueDiffPercDl - target
FunctValueDiffPercDh)*normTargetFunctValue1 - Nh*targetFunctValue
DiffPercDl + targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
denormAverPredictFunctValue11 =((targetFunctValueDiffPercDl - target
FunctValueDiffPercDh)*normPredictFunctValue1 - Nh*targetFunctValue
DiffPercDl + targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
//inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
targetToPredictFunctValueDiff = (Math.abs(denormTargetFunctValue1
denormAverPredictFunctValue11)/denormTargetFunctValue1)*100;
        System.out.println("intDayNumber = " + intDayNumber +  "  target
        FunctionValue = " + denormTargetFunctValue1 + "  predictFunction
```

```
                Value = " + denormAverPredictFunctValue11 + "    valurDiff = " +
                targetToPredictFunctValueDiff);
        if (targetToPredictFunctValueDiff > maxGlobalResultDiff)maxGlobal
        ResultDiff =targetToPredictFunctValueDiff;
        sumGlobalResultDiff = sumGlobalResultDiff +targetToPredictFunct ValueDiff;
        // Populate chart elements       doubleDayNumber = (double) rBatchNumber+1;
         xData.add(doubleDayNumber);
        yData1.add(denormTargetFunctValue1);
        yData2.add(denormAverPredictFunctValue11);
             }
        // End for FunctValue pair1 loop
} // End of the loop over batches
sumGlobalResultDiff = sumGlobalResultDiff +targetToPredictFunctValue Diff;
averGlobalResultDiff = sumGlobalResultDiff/numberOfTrainBatchesTo Process;
// Print the max and average results
System.out.println(" ");
System.out.println("    ");              System.out.println("maxGlobalResultDiff    =    "    +
maxGlobalResultDiff);
System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
XYSeries series1 = Chart.addSeries("Actual", xData, yData1);
XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
// Save the chart image    try
{
BitmapEncoder.saveBitmapWithDPI(Chart,    strTrainChartFileName,    BitmapFormat.JPG,
100);
}
catch (Exception bt)
{
bt.printStackTrace();
}
System.out.println ("The Chart has been saved");
        returnCodes[0] = 0;
        returnCodes[1] = 0;
        returnCodes[2] = 0;
        return returnCodes;
} // End of method
// ============================================================
// Muat jaringan 1 terlatih yang disimpan sebelumnya dan uji dengan
// memproses catatan Tes
// ============================================================
static public void loadAndTestNetwork()
{
System.out.println("Testing the network1s results");
List<Double> xData = new ArrayList<Double>();
```

```
List<Double> yData1 = new ArrayList<Double>();
List<Double> yData2 = new ArrayList<Double>();
double targetToPredictFunctValueDiff = 0;
double maxGlobalResultDiff = 0.00;
double averGlobalResultDiff = 0.00;
double sumGlobalResultDiff = 0.00;
double maxGlobalIndex = 0;
double normInputDayFromRecord1 = 0.00;
double normTargetFunctValue1 = 0.00;
double normPredictFunctValue1 = 0.00;
double denormInputDayFromRecord1 = 0.00;
double denormTargetFunctValue1 = 0.00;
double denormAverPredictFunctValue1 = 0.00;
double normInputDayFromRecord2 = 0.00;
double normTargetFunctValue2 = 0.00;
double normPredictFunctValue2 = 0.00;
double denormInputDayFromRecord2 = 0.00;
double denormTargetFunctValue2 = 0.00;
double denormAverPredictFunctValue2 = 0.00;
double normInputDayFromTestRecord = 0.00;
double denormInputDayFromTestRecord = 0.00;
double denormAverPredictFunctValue = 0.00;
double denormTargetFunctValueFromTestRecord = 0.00;
String tempLine;
String[] tempWorkFields;
double dayKeyFromTestRecord = 0.00;
double targetFunctValueFromTestRecord = 0.00;
double r1 = 0.00;
double r2 = 0.00;
BufferedReader br4;
BasicNetwork network1;
BasicNetwork network2;
int k1 = 0;
int k3 = 0;
try
{
        // Process testing records
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        for (k1 = 0; k1 < numberOfTestBatchesToProcess; k1++)
        {
                // Read the corresponding test micro-batch file.                br4 = new
                BufferedReader(new FileReader(strTestingFileNames[k1]));
                tempLine = br4.readLine();
                // Skip the label record          tempLine = br4.readLine();
                // Break the line using comma as separator                tempWorkFields =
                tempLine.split(cvsSplitBy);
```

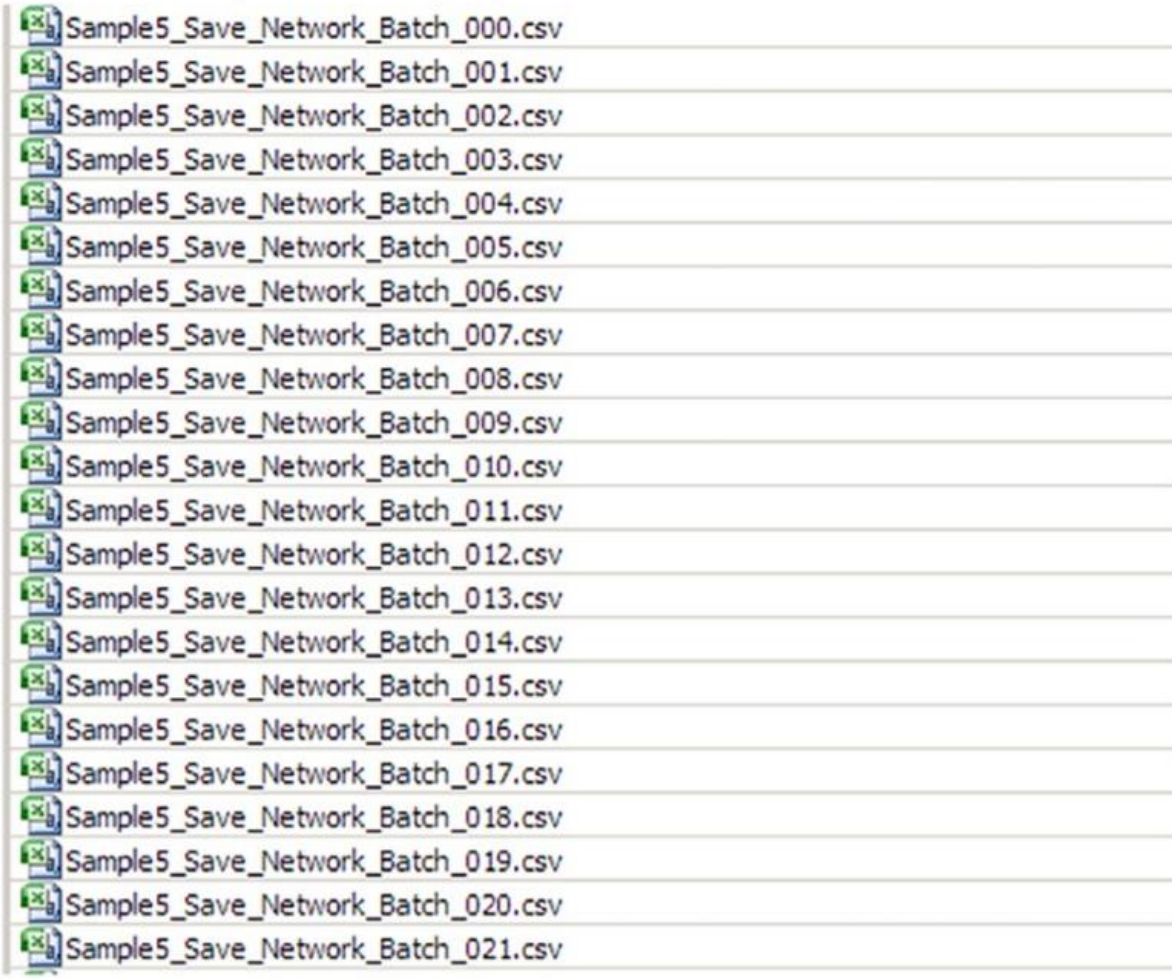
```
dayKeyFromTestRecord = Double.parseDouble(tempWorkFields[0]);
targetFunctValueFromTestRecord = Double.parseDouble(tempWork
Fields[1]);
// De-normalize the dayKeyFromTestRecord
denormInputDayFromTestRecord = ((inputDayDl - inputDayDh)*day
KeyFromTestRecord - Nh*inputDayDl + inputDayDh*Nl)/(Nl - Nh);
// De-normalize the targetFunctValueFromTestRecord
denormTargetFunctValueFromTestRecord = ((targetFunctValue DiffPercDl -
targetFunctValueDiffPercDh)*targetFunctValueFrom TestRecord -
Nh*targetFunctValueDiffPercDl + targetFunctValue DiffPercDh*Nl)/(Nl
- Nh);
// Load the corresponding training micro-batch dataset in memory
MLDataSet trainingSet1 = loadCSV2Memory(strTrainingFile
Names[k1],intInputNeuronNumber,intOutputNeuronNumber,true,
CSVFormat.ENGLISH,false); network1 =
(BasicNetwork)EncogDirectoryPersistence. loadObject(new
File(strSaveTrainNetworkFileNames[k1]));
// Get the results after the network1 optimization int iMax = 0;
int i = - 1;
// Index of the array to get results
for (MLDataPair pair1: trainingSet1)
{
        i++; iMax = i+1;
        MLData inputData1 = pair1.getInput();
        MLData actualData1 = pair1.getIdeal();
        MLData predictData1 = network1.compute(inputData1);
        // These values are Normalized
        normInputDayFromRecord1 = inputData1.getData(0);
        normTargetFunctValue1 = actualData1.getData(0);
        normPredictFunctValue1 = predictData1.getData(0);
        // De-normalize the obtained values
        denormInputDayFromRecord1 = ((inputDayDl - inputDayDh)*
        normInputDayFromRecord1 - Nh*inputDayDl + inputDayDh*Nl)/
        (Nl - Nh);
        denormTargetFunctValue1 = ((targetFunctValueDiffPercDl
        targetFunctValueDiffPercDh)*normTargetFunctValue1 - Nh*
        targetFunctValueDiffPercDl + targetFunctValueDiffPercDh*Nl)/
        (Nl - Nh);
        denormAverPredictFunctValue1 =((targetFunctValueDiffPercDl
        targetFunctValueDiffPercDh)*normPredictFunctValue1 - Nh* t
        argetFunctValueDiffPercDl + targetFunctValueDiffPercDh*Nl)/
        (Nl - Nh);
} // End for pair1
        // Now calculate everything again for the SaveNetwork (which
        // key is greater than dayKeyFromTestRecord value)in memory
        MLDataSet trainingSet2 = loadCSV2Memory(strTrainingFile
        Names[k1+1],intInputNeuronNumber,intOutputNeuronNumber,true,
        CSVFormat.ENGLISH,false);
```

```
network2 = (BasicNetwork)EncogDirectoryPersistence.loadObject
(new File(strSaveTrainNetworkFileNames[k1+1]));
// Get the results after the network1 optimization iMax = 0;
i = - 1;
for (MLDataPair pair2:  trainingSet2)
{
        i++;
        iMax = i+1;
        MLData inputData2 = pair2.getInput();
        MLData actualData2 = pair2.getIdeal();
        MLData predictData2 = network2.compute(inputData2);
        // These values are Normalized
        normInputDayFromRecord2 = inputData2.getData(0);
        normTargetFunctValue2 = actualData2.getData(0);
        normPredictFunctValue2 = predictData2.getData(0);
        // De-normalize the obtained values
        denormInputDayFromRecord2 = ((inputDayDl -
        inputDayDh)* normInputDayFromRecord2 -
        Nh*inputDayDl + inputDayDh*Nl)/ (Nl - Nh);
        denormTargetFunctValue2 = ((targetFunctValueDiffPercDl
        targetFunctValueDiffPercDh)*normTargetFunctValue2 -
        Nh*target FunctValueDiffPercDl +
        targetFunctValueDiffPercDh*Nl)/ (Nl - Nh);
        denormAverPredictFunctValue2
        =((targetFunctValueDiffPercDl
        targetFunctValueDiffPercDh)*normPredictFunctValue2
        Nh*targetFunctValueDiffPercDl +
        targetFunctValueDiffPercDh *Nl)/(Nl - Nh);
} // End for pair1 loop
// Get the average of the denormAverPredictFunctValue1 and
denormAverPredictFunctValue2 denormAverPredictFunctValue =
(denormAverPredictFunctValue1 + denormAverPredictFunctValue2)/2;
targetToPredictFunctValueDiff =
(Math.abs(denormTargetFunctValueFromTestRecord - denormAver
PredictFunctValue)/denormTargetFunctValueFromTestRecord)*100;
System.out.println("Record Number = " + k1 + "  DayNumber = " +
denormInputDayFromTestRecord + " denormTargetFunctValue
FromTestRecord = " + denormTargetFunctValueFromTestRecord + "
denormAverPredictFunctValue = " + denormAverPredict FunctValue + "
valurDiff = " + targetToPredictFunctValueDiff);
if (targetToPredictFunctValueDiff > maxGlobalResultDiff)
{
maxGlobalIndex = iMax; maxGlobalResultDiff
=targetToPredictFunctValueDiff; }
sumGlobalResultDiff = sumGlobalResultDiff + targetToPredict
FunctValueDiff;
// Populate chart elements
xData.add(denormInputDayFromTestRecord);
```

```
                yData1.add(denormTargetFunctValueFromTestRecord);
                yData2.add(denormAverPredictFunctValue);
}  // End of loop using k1
        // Print the max and average results
        System.out.println(" ");
        averGlobalResultDiff = sumGlobalResultDiff/numberOfTestBatchesToProcess;
        System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff +        "  i = " +
        maxGlobalIndex);
        System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
        }
        // End of TRY    catch (IOException e1)
        {
        e1.printStackTrace();
        }
        // All testing batch files have been processed          XYSeries series1 =
        Chart.addSeries("Actual", xData, yData1);
        XYSeries series2 = Chart.addSeries("Forecasted", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image     try
        {
        BitmapEncoder.saveBitmapWithDPI(Chart,                strTrainChartFileName,
        BitmapFormat.JPG, 100);
        }
        catch (Exception bt)
        {
        bt.printStackTrace();
        }
        System.out.println ("The Chart has been saved");      System.out.println("End of
        testing for mini-batches training");
    } // End of the method
} // End of the  Encog class
```

Logika pemrosesan sangat berbeda dalam program ini. Mari kita mulai dari metode getChart(). Terlepas dari pernyataan biasa yang diperlukan oleh paket XChart, di sini Anda membuat nama untuk kumpulan mikro pelatihan dan file jaringan penyimpanan. Nama file yang dihasilkan untuk mikro-batch harus cocok dengan nama file mikro-batch yang disiapkan pada disk saat Anda memecah file pelatihan yang dinormalisasi menjadi mikro-batch.

Nama untuk file jaringan yang disimpan memiliki struktur yang sesuai. Nama-nama yang dihasilkan ini akan digunakan oleh metode pelatihan untuk menyimpan jaringan terlatih yang sesuai dengan batch mikro pada disk. Nama yang dihasilkan disimpan dalam dua array yang disebut strTrainingFileNames[] dan strSaveTrainNetworkFileNames[]. Gambar 18.5 menunjukkan fragmen dari jaringan tersimpan yang dihasilkan.



Gambar 18.5 Fragmen file jaringan simpan yang dihasilkan

Selanjutnya, Anda membuat dan mengisi dua larik yang disebut linkToSaveNetworkDayKeys[] dan linkToSaveNetworkTargetFunctValueKeys[]. Untuk setiap hari berturut-turut, Anda mengisi larik linkToSaveNetworkDayKeys[] dengan nilai field1 dari catatan micro-batch pelatihan. Anda mengisi larik linkToSaveNetworkTargetFunctValueKeys[] dengan nama saveNetworkFiles yang sesuai pada disk. Oleh karena itu, kedua larik tersebut memiliki hubungan antara kumpulan data mikro-batch dan kumpulan data jaringan simpan yang sesuai.

Program ini juga menghasilkan nama file mikro-batch pengujian, mirip dengan nama yang dihasilkan untuk file mikro-batch pelatihan. Setelah semua ini selesai, Anda memanggil metode loadTrainFunctValueFileInMemory yang memuat nilai file pelatihan dalam memori.

18.5 KODE PROGRAM UNTUK METODE GETCHART()

Daftar 18.5 memperlihatkan kode program untuk metode getChart()

Daftar 18.5 Kode Metode getChart()

```
public XYChart getChart()
{
    // Create the Chart
    Chart = new XYChartBuilder().width(900).height(500).title(getClass().
    getSimpleName()).xAxisTitle("day").yAxisTitle("y=f(day)").build();
```

```
// Customize Chart
Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor
(ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.OutsideE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.PLAIN, 11));
//Chart.getStyler().setDayPattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
// Config data
// For training    //intWorkingMode = 0;
// For testing    intWorkingMode = 1;
// common config data
intNumberOfTrainBatchesToProcess = 1000;
intNumberOfTestBatchesToProcess = 1000;
intNumberOfRecordsInTestFile = 999;
intNumberOfRowsInBatches = 1;    intInputNeuronNumber = 1;
intOutputNeuronNumber      =      1;      strTrainFileNameBase      =
"C:/Book_Examples/Sample5_Train_Norm_Batch_";
strTestFileNameBase = "C:/Book_Examples/Sample5_Test_Norm_Batch_";
strSaveTrainNetworkFileBase    =    "C:/Book_Examples/Sample5_Save_Network_
Batch_";
strTrainChartFileName      =      "C:/Book_Examples/Sample5_Chart_Train_File_
Microbatch.jpg";
strTestChartFileName      =      "C:/Book_Examples/Sample5_Chart_Test_File_
Microbatch.jpg";
strFunctValueTrainFile = "C:/Book_Examples/Sample5_Train_Real.csv";
strFunctValueTestFile = "C:/Book_Examples/Sample5_Test_Real.csv";
//  Generate training micro-batch file names and the corresponding Save Network
file names
intDayNumber = -1;
// Day number for the chart
for (int i = 0;
i < intNumberOfTrainBatchesToProcess;
```

```
i++)
{
intDayNumber++;
iString = Integer.toString(intDayNumber);
if (intDayNumber >= 10 & intDayNumber < 100  )
{
strOutputFileName = strTrainFileNameBase + "0" + iString + ".csv";
strSaveNetworkFileName = strSaveTrainNetworkFileBase + "0" + iString + ".csv";
}
else
{
if (intDayNumber < 10)
{
strOutputFileName = strTrainFileNameBase + "00" + iString +          ".csv";
strSaveNetworkFileName = strSaveTrainNetworkFileBase + "00" + iString + ".csv";
}
else
{
strOutputFileName = strTrainFileNameBase + iString + ".csv";
strSaveNetworkFileName = strSaveTrainNetworkFileBase +  iString + ".csv";
}
}
strTrainingFileNames[intDayNumber] = strOutputFileName;
strSaveTrainNetworkFileNames[intDayNumber] =
strSaveNetwork FileName;
}
// End the FOR loop
// Build the array linkToSaveNetworkFunctValueDiffKeys
String tempLine;
double tempNormFunctValueDiff = 0.00;
double tempNormFunctValueDiffPerc = 0.00;
double tempNormTargetFunctValueDiffPerc = 0.00;
String[] tempWorkFields;
try
{
intDayNumber = -1;
// Day number for the chart
for (int m = 0;
m < intNumberOfTrainBatchesToProcess; m++)
{
intDayNumber++;
BufferedReader br3 = new BufferedReader(new
FileReader(strTrainingFileNames[intDayNumber]));
tempLine = br3.readLine();
// Skip the label record and zero batch record          tempLine = br3.readLine();
// Break the line using comma as separator                tempWorkFields =
tempLine.split(cvsSplitBy);
tempNormFunctValueDiffPerc = Double.parseDouble(tempWork Fields[0]);
```

```
tempNormTargetFunctValueDiffPerc = Double.parseDouble (tempWorkFields[1]);
linkToSaveNetworkDayKeys[intDayNumber] = tempNormFunctValue DiffPerc;
linkToSaveNetworkTargetFunctValueKeys[intDayNumber] =
tempNormTargetFunctValueDiffPerc;
}
// End the FOR loop
// Generate testing micro-batch file names
if(intWorkingMode == 1)
{
intDayNumber = -1;
for (int i = 0;
i < intNumberOfTestBatchesToProcess;
i++)
{
intDayNumber++;          iString =
Integer.toString(intDayNumber);
// Construct the testing batch names          if (intDayNumber >= 10 & intDayNumber
< 100)
{
strOutputFileName = strTestFileNameBase + "0" + iString +          ".csv";
}
else
{
if (intDayNumber < 10)
{
strOutputFileName = strTestFileNameBase + "00" +          iString + ".csv";
}
else
{
strOutputFileName = strTrainFileNameBase + iString +          ".csv";
}
}
strTestingFileNames[intDayNumber] = strOutputFileName;
}
// End the FOR loop
}
// End of IF
}
// End for try     catch (IOException io1)
{
io1.printStackTrace();
System.exit(1);
}
loadTrainFunctValueFileInMemory();
```

Ketika bagian itu selesai, logika akan memeriksa apakah akan menjalankan metode pelatihan atau pengujian. Saat bidang workingMode sama dengan 1, ia akan memanggil metode pelatihan dalam satu lingkaran (seperti yang Anda lakukan sebelumnya). Namun,

karena Anda sekarang memiliki banyak file pelatihan mikro-batch (bukan set data tunggal), Anda perlu memperluas larik errorCode untuk menampung satu nilai lagi: nomor mikro-batch.

Fragmen Kode 1 dari Metode Pelatihan

Di dalam file pelatihan, jika setelah banyak iterasi kesalahan jaringan tidak dapat menghapus batas kesalahan, Anda keluar dari metode pelatihan dengan nilai returnCode 1. Kontrol dikembalikan ke logika di dalam metode getChart() yang memanggil metode pelatihan dalam satu lingkaran. Pada saat itu, Anda perlu mengembalikan parameter yang dipanggil dengan metode microbatch. Daftar 18-6 menunjukkan fragmen kode 1 dari metode pelatihan.

Daftar 18.6 Kode Fragmen 1 pada Mode Training

```
if(intWorkingMode == 0)  {
        // Train batches and save the trained networks
        int paramErrorCode;
        int paramBatchNumber;
        int paramR;
        int paramDayNumber;
        int paramS;
        File file1 = new File(strTrainChartFileName);
        if(file1.exists())
        file1.delete();
        returnCodes[0] = 0;
        // Clear the error Code
        returnCodes[1] = 0;
        // Set the initial batch Number to 0;
        returnCodes[2] = 0;
        // Set the initial day number to 0;
        do
        {
                paramErrorCode = returnCodes[0];
                paramBatchNumber = returnCodes[1];
                paramDayNumber = returnCodes[2];
                returnCodes =
                trainBatches(paramErrorCode,paramBatchNumber,paramDayNumber);
                } while (returnCodes[0] > 0);
        }
        // End of the train logic else
        {
                // Load and test the network logic
                File file2 = new File(strTestChartFileName);
                if(file2.exists())
                file2.delete();
                loadAndTestNetwork();
                // End of the test logic
                }
        Encog.getInstance().shutdown();
        return Chart;
        }  // End of method
```

Fragmen Kode 2 dari Metode Pelatihan

Di sini, sebagian besar kode harus familier bagi Anda, kecuali logika yang terlibat dalam pemrosesan batch mikro. Anda membangun jaringan. Selanjutnya, Anda mengulang mikro-batch (ingat, ada banyak file pelatihan mikro-batch alih-alih satu set data pelatihan yang Anda proses sebelumnya). Di dalam loop, Anda memuat file micro-batch pelatihan di memori dan kemudian melatih jaringan menggunakan file micro-batch saat ini.

Saat jaringan dilatih, Anda menyimpannya di disk, menggunakan nama dari larik linkToSaveNetworkDayKeys yang sesuai dengan file batch mikro yang saat ini diproses. Mengulangi kumpulan data pasangan, Anda mengambil nilai input, aktual, dan prediksi untuk setiap batch mikro, mendenormalkannya, dan mencetak hasilnya sebagai log pelatihan.

Dalam loop kereta jaringan, ketika setelah banyak iterasi, kesalahan jaringan tidak dapat menghapus batas kesalahan, Anda menetapkan nilai returnCode ke 1 dan keluar dari metode pelatihan. Kontrol dikembalikan ke logika yang memanggil metode pelatihan dalam satu lingkaran. Saat Anda keluar dari metode pelatihan, Anda sekarang menetapkan tiga nilai returnCode: nilai returnCode, nomor micro-batch, dan nomor hari. Itu membantu logika yang memanggil metode pelatihan dalam satu lingkaran untuk tetap berada dalam batch mikro dan hari pemrosesan yang sama. Anda juga mengisi hasil untuk elemen bagan. Terakhir, Anda menambahkan data seri bagan, menghitung kesalahan rata-rata dan maksimum untuk semua mikro-batch, mencetak hasilnya sebagai file log, dan menyimpan file bagan. Daftar 18-7 menunjukkan fragmen kode 2 dari metode pelatihan.

Daftar 18-7. Fragmen Kode 2 dari Metode Pelatihan

```
// Build the network
BasicNetwork network = new BasicNetwork();
// Input layer
network.addLayer(new BasicLayer(null,true,intInputNeuronNumber));
// Hidden layer.
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
network.addLayer(new BasicLayer(new ActivationTANH(),true,5));
// Output layer
network.addLayer(new BasicLayer(new ActivationTANH(),false,
intOutputNeuronNumber));
network.getStructure().finalizeStructure(); network.reset();
maxGlobalResultDiff = 0.00; averGlobalResultDiff = 0.00; sumGlobalResultDiff = 0.00;
// Loop over micro-batches
intDayNumber = paramDayNumber;
// Day number for the chart
for (rBatchNumber = paramBatchNumber;
rBatchNumber < intNumberOfTrain BatchesToProcess; rBatchNumber++)
{
        intDayNumber++;
        // Day number for the chart
        // Load the training CVS file for the current batch in memory
```

```
        MLDataSet trainingSet =
        loadCSV2Memory(strTrainingFileNames[rBatchNumber],intInput
        NeuronNumber,intOutputNeuronNumber,true,CSVFormat.ENGLISH,false);
        // train the neural network
        ResilientPropagation train = new ResilientPropagation(network, trainingSet);
        int epoch = 1;  double tempLastErrorPerc = 0.00;
do
{
        train.iteration();
        epoch++;
        for (MLDataPair pair1:  trainingSet)
        {
                MLData inputData = pair1.getInput();
                MLData actualData = pair1.getIdeal();
                MLData predictData = network.compute(inputData);
                // These values are Normalized as the whole input is
                normInputFunctValueDiffPercFromRecord = inputData.getData(0);
                normTargetFunctValue = actualData.getData(0);
                normPredictFunctValue = predictData.getData(0);
                denormInputFunctValueDiffPercFromRecord =((inputDayDl -
                inputDayDh)*normInputFunctValueDiffPercFromRecord - Nh*inputDayDl +
                inputDayDh*Nl)/(Nl - Nh);
                denormTargetFunctValue = ((targetFunctValueDiffPercDl -
                targetFunctValueDiffPercDh)*normTargetFunctValue
                Nh*targetFunctValueDiffPercDl + targetFunctValueDiffPercDh*Nl)/ (Nl
                - Nh);
                denormPredictFunctValue = ((targetFunctValueDiffPercDl -
                targetFunctValueDiffPercDh)*normPredictFunctValue - Nh*target
                FunctValueDiffPercDl + targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
                inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
                targetToPredictFunctValueDiff = (Math.abs(denormTargetFunctValue
                denormPredictFunctValue)/denormTargetFunctValue)*100;
        }
        if (epoch >= 500 &&targetToPredictFunctValueDiff > 0.0002)
                {
                        returnCodes[0] = 1;
                        returnCodes[1] = rBatchNumber;
                        returnCodes[2] = intDayNumber-1;
                        return returnCodes;
                }
        } while(targetToPredictFunctValueDiff >  0.0002);
        // 0.00002
        // Save the network for the current batch
        EncogDirectoryPersistence.saveObject(newFile(strSaveTrainNetwork
        FileNames[rBatchNumber]),network);
        // Get the results after the network optimization
        int i = - 1;
        for (MLDataPair pair:  trainingSet)
```

```
        {
        i++;
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // These values are Normalized as the whole input is
        normInputFunctValueDiffPercFromRecord = inputData.getData(0);
        normTargetFunctValue = actualData.getData(0);
        normPredictFunctValue = predictData.getData(0);
        denormInputFunctValueDiffPercFromRecord = ((inputDayDl - inputDayDh)*
        normInputFunctValueDiffPercFromRecord - Nh*inputDayDl +
        inputDayDh*Nl)/(Nl - Nh);
        denormTargetFunctValue = ((targetFunctValueDiffPercDl - targetFunct
        ValueDiffPercDh)*normTargetFunctValue – Nh*targetFunctValueDiffPercDl +
        targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
        denormPredictFunctValue = ((targetFunctValueDiffPercDl - target
        FunctValueDiffPercDh)*normPredictFunctValue – Nh*targetFunctValue
        DiffPercDl + targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
        inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
        targetToPredictFunctValueDiff = (Math.abs(denormTargetFunctValue
        denormPredictFunctValue)/denormTargetFunctValue)*100;
        System.out.println("intDayNumber = " + intDayNumber + "  target FunctionValue =
        " + denormTargetFunctValue + " predictFunction Value = " +
        denormPredictFunctValue + "  valurDiff = " + targetTo PredictFunctValueDiff);
        if (targetToPredictFunctValueDiff > maxGlobalResultDiff) maxGlobalResultDiff
        =targetToPredictFunctValueDiff;
        sumGlobalResultDiff = sumGlobalResultDiff +targetToPredictFunct ValueDiff;
        // Populate chart elements doubleDayNumber = (double) rBatchNumber+1;
        xData.add(doubleDayNumber);
        yData1.add(denormTargetFunctValue);
        yData2.add(denormPredictFunctValue);
    } // End for the pair loop
} // End of the loop over batches
sumGlobalResultDiff = sumGlobalResultDiff +targetToPredictFunctValueDiff;
averGlobalResultDiff = sumGlobalResultDiff/intNumberOfTrainBatches ToProcess;
// Print the max and average results
System.out.println(" ");
System.out.println(" ");
System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff);
System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
XYSeries series1 = Chart.addSeries("Actual", xData, yData1);
XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLUE);
series2.setMarkerColor(Color.ORANGE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
// Save the chart image    try
{
```

```
BitmapEncoder.saveBitmapWithDPI(Chart, strTrainChartFileName,
BitmapFormat.JPG, 100);
}
catch (Exception bt)
{
bt.printStackTrace();
}
System.out.println ("The Chart has been saved");
returnCodes[0] = 0;
returnCodes[1] = 0;
returnCodes[2] = 0;
return returnCodes;
} // End of method
```

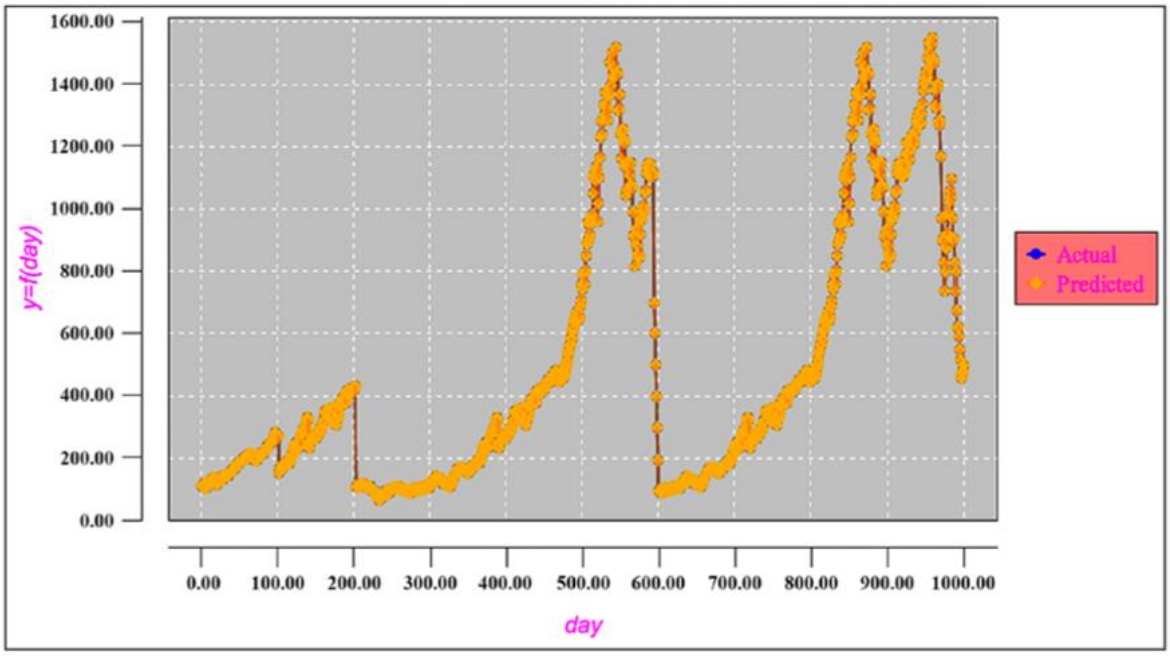
Hasil Pelatihan untuk Metode Micro-Batch

Daftar 18-8 menunjukkan bagian akhir dari hasil pelatihan.

Daftar 18-8. Hasil Pelatihan

```
DayNumber =  989  TargeValue = 735.09  PredictedValue = 735.09005    DiffPercf = 6.99834E-6
DayNumber =  990  TargeValue = 797.87  PredictedValue = 797.86995    DiffPercf = 6.13569E-6
DayNumber =  991  TargeValue = 672.81  PredictedValue = 672.80996    DiffPercf = 5.94874E-6
DayNumber =  992  TargeValue = 619.14  PredictedValue = 619.14003    DiffPercf = 5.53621E-6
DayNumber =  993  TargeValue = 619.32  PredictedValue = 619.32004    DiffPercf = 5.65663E-6
DayNumber =  994  TargeValue = 590.47  PredictedValue = 590.47004    DiffPercf = 6.40373E-6
DayNumber =  995  TargeValue = 547.28  PredictedValue = 547.27996    DiffPercf = 6.49734E-6
DayNumber =  996  TargeValue = 514.62  PredictedValue = 514.62002    DiffPercf = 3.39624E-6
DayNumber =  997  TargeValue = 455.4   PredictedValue = 455.40000    DiffPercf = 2.73780E-7
DayNumber =  998  TargeValue = 470.43  PredictedValue = 470.42999    DiffPercf = 4.35234E-7
DayNumber =  999  TargeValue = 480.28  PredictedValue = 480.28002    DiffPercf = 3.52857E-6
DayNumber = 1000 TargeValue = 496.77  PredictedValue = 496.76999    DiffPercf = 9.81900E-7
maxGlobalResultDiff = 9.819000149262707E-7
averGlobalResultDiff = 1.9638000298525415E-9
```

Saat ini, hasil pemrosesan pelatihan sudah cukup baik, terutama untuk aproksimasi fungsi nonkontinu. Kesalahan rata-rata adalah 0,000000001963800029852541 5, kesalahan maksimum (catatan yang dioptimalkan terburuk) adalah 0,0000009819000149262707, dan grafik tampak hebat. Gambar 18.6 menunjukkan diagram hasil pemrosesan pelatihan menggunakan batch mikro.



Gambar 18.6 Bagan hasil pelatihan menggunakan mikro-batch

Untuk pengujian, Anda akan membuat file dengan nilai di antara titik pelatihan. Misalnya, untuk dua catatan pelatihan 1 dan 2, Anda akan menghitung hari baru sebagai rata-rata dari dua hari pelatihan. Untuk nilai fungsi record, Anda akan menghitung rata-rata dari dua nilai fungsi pelatihan. Dengan cara ini, dua catatan pelatihan berturut-turut akan membuat satu catatan pengujian dengan nilai rata-rata dari catatan pelatihan. Tabel 18.4 menunjukkan tampilan data pengujian.

Tabel 18.4 Fragmen Kumpulan Data Pengujian

xPoint	yValue	xPoint	yValue	xPoint	yValue
1.5	108.918	31.5	139.295	61.5	204.9745
2.5	113.696	32.5	142.3625	62.5	208.6195
3.5	117.806	33.5	142.6415	63.5	207.67
4.5	113.805	34.5	141.417	64.5	209.645
5.5	106.006	35.5	142.1185	65.5	208.525
6.5	106.6155	36.5	146.215	66.5	208.3475
7.5	107.5465	37.5	150.6395	67.5	203.801
8.5	109.5265	38.5	154.1935	68.5	194.6105
9.5	116.223	39.5	158.338	69.5	199.9695
10.5	118.1905	40.5	161.4155	70.5	207.9885

11.5	121.9095	41.5	161.851	71.5	206.2175
12.5	127.188	42.5	164.6005	72.5	199.209
13.5	130.667	43.5	165.935	73.5	193.6235
14.5	132.6525	44.5	165.726	74.5	199.5985
15.5	134.472	45.5	171.9045	75.5	206.252
16.5	135.4405	46.5	178.1175	76.5	208.113
17.5	133.292	47.5	182.7085	77.5	209.791
18.5	130.646	48.5	181.5475	78.5	213.623
19.5	125.5585	49.5	182.102	79.5	217.2275
20.5	117.5155	50.5	186.5895	80.5	216.961
21.5	119.236	51.5	187.8145	81.5	214.721
22.5	125.013	52.5	190.376	82.5	216.248
23.5	125.228	53.5	194.19	83.5	221.882
24.5	128.5005	54.5	194.545	84.5	225.885
25.5	133.9045	55.5	196.702	85.5	232.1255
26.5	138.7075	56.5	198.783	86.5	236.318
27.5	140.319	57.5	199.517	87.5	237.346
28.5	135.412	58.5	204.2805	88.5	239.8
29.5	133.6245	59.5	206.323	89.5	241.7605
30.5	137.074	60.5	202.6945	90.5	244.6855

Tabel 18.5 Catatan Tes

1.5	108.918

Ini rata-rata dua catatan pelatihan, seperti yang ditunjukkan pada Tabel 18.5.

Tabel 18.6 Dua Catatan Pelatihan

1	107.387
2	110.449

Kumpulan data uji memiliki 998 catatan. Tabel 18.6 menunjukkan fragmen kumpulan data pengujian.

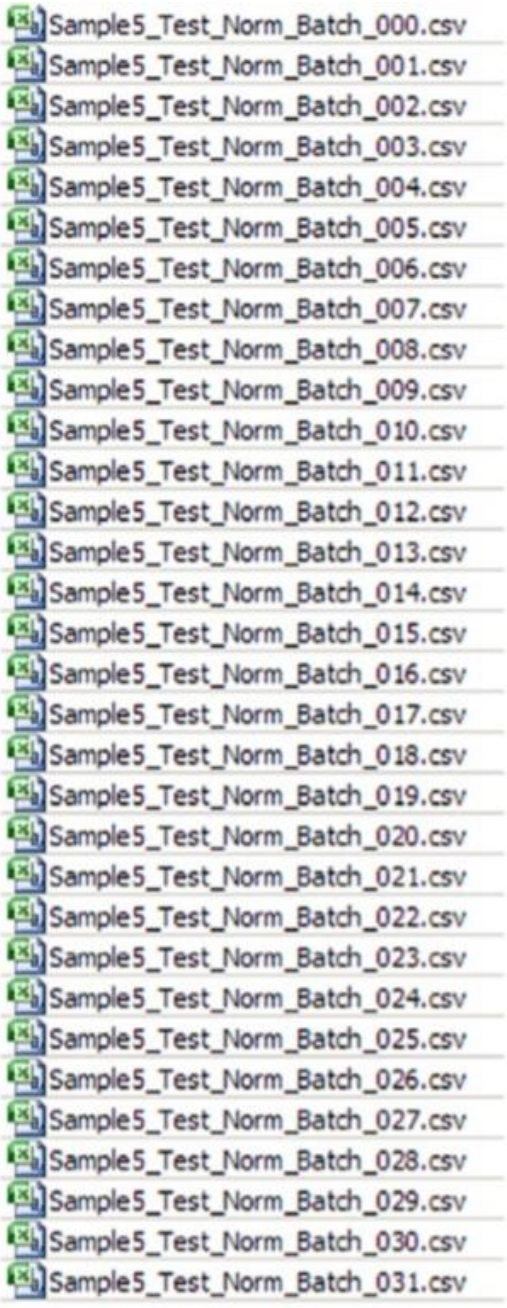
Tabel 18.7 menunjukkan sebuah fragmen dari kumpulan data pengujian yang dinormalisasi.

Tabel 18.7 Fragmen dari Kumpulan Data Pengujian yang Dinormalisasi

xPoint	y	xPoint	y	xPoint	y
-0.9990	-0.9365	-0.9389	-0.8970	-0.8789	-0.8117
-0.9970	-0.9303	-0.9369	-0.8930	-0.8769	-0.8070
-0.9950	-0.9249	-0.9349	-0.8927	-0.8749	-0.8082
-0.9930	-0.9301	-0.9329	-0.8943	-0.8729	-0.8057
-0.9910	-0.9403	-0.9309	-0.8934	-0.8709	-0.8071
-0.9890	-0.9395	-0.9289	-0.8880	-0.8689	-0.8073
-0.9870	-0.9383	-0.9269	-0.8823	-0.8669	-0.8132
-0.9850	-0.9357	-0.9249	-0.8777	-0.8649	-0.8252
-0.9830	-0.9270	-0.9229	-0.8723	-0.8629	-0.8182
-0.9810	-0.9244	-0.9209	-0.8683	-0.8609	-0.8078
-0.9790	-0.9196	-0.9189	-0.8677	-0.8589	-0.8101
-0.9770	-0.9127	-0.9169	-0.8642	-0.8569	-0.8192
-0.9750	-0.9082	-0.9149	-0.8624	-0.8549	-0.8265
-0.9730	-0.9056	-0.9129	-0.8627	-0.8529	-0.8187
-0.9710	-0.9033	-0.9109	-0.8547	-0.8509	-0.8101
-0.9690	-0.9020	-0.9089	-0.8466	-0.8488	-0.8076
-0.9670	-0.9048	-0.9069	-0.8406	-0.8468	-0.8055
-0.9650	-0.9083	-0.9049	-0.8421	-0.8448	-0.8005
-0.9630	-0.9149	-0.9029	-0.8414	-0.8428	-0.7958
-0.9610	-0.9253	-0.9009	-0.8356	-0.8408	-0.7962
-0.9590	-0.9231	-0.8989	-0.8340	-0.8388	-0.7991
-0.9570	-0.9156	-0.8969	-0.8307	-0.8368	-0.7971
-0.9550	-0.9153	-0.8949	-0.8257	-0.8348	-0.7898
-0.9530	-0.9110	-0.8929	-0.8253	-0.8328	-0.7846
-0.9510	-0.9040	-0.8909	-0.8225	-0.8308	-0.7765
-0.9489	-0.8978	-0.8889	-0.8198	-0.8288	-0.7710
-0.9469	-0.8957	-0.8869	-0.8188	-0.8268	-0.7697
-0.9449	-0.9021	-0.8849	-0.8126	-0.8248	-0.7665
-0.9429	-0.9044	-0.8829	-0.8100	-0.8228	-0.7639
-0.9409	-0.8999	-0.8809	-0.8147	-0.8208	-0.7601

Seperti halnya kumpulan data pelatihan yang dinormalisasi, Anda memecah kumpulan data pengujian yang dinormalisasi menjadi kumpulan mikro. Setiap kumpulan data mikro-batch harus berisi catatan label dan catatan dari file asli yang akan diproses.

Hasilnya, Anda akan mendapatkan 998 kumpulan data mikro-batch (diberi nomor dari 0 hingga 997). Gambar 18.7 menunjukkan fragmen daftar file mikro-batch pengujian yang dinormalisasi.



Gambar 18.7 Fragmen kumpulan data uji mikro-batch yang dinormalisasi

Kumpulan file ini sekarang menjadi input untuk proses pengujian jaringan saraf.

18.6 LOGIKA PEMROSESAN TES

Untuk logika pemrosesan pengujian, Anda mengulang mikro-batch. Untuk setiap batch mikro uji, Anda membaca catatannya, mengambil nilai catatan, dan mendenormalkannya. Selanjutnya, Anda memuat kumpulan data mikro-batch untuk titik 1 (yang merupakan titik terdekat dengan catatan pengujian tetapi kurang dari itu) dalam memori. Anda juga memuat file save-network yang sesuai di memori. Mengulangi

kumpulan data pasangan, Anda mengambil nilai input, aktif, dan prediksi untuk batch mikro dan mendenormalkannya.

Anda juga memuat kumpulan data mikro-batch untuk titik 2 (yang merupakan titik terdekat dengan catatan pengujian tetapi lebih besar dari itu) dalam memori, dan Anda memuat file jaringan simpan yang sesuai dalam memori. Mengulangi kumpulan data pasangan, Anda mengambil nilai input, aktif, dan prediksi untuk batch mikro dan mendenormalkannya.

Selanjutnya, Anda menghitung rata-rata nilai fungsi yang diprediksi untuk poin 1 dan poin 2. Terakhir, Anda menghitung persentase kesalahan dan mencetak hasilnya sebagai log pemrosesan. Sisanya hanya staf lain-lain. Daftar 18-9 menunjukkan kode program untuk metode pengujian.

Daftar 18-9. Kode Metode Pengujian

```
for (k1 = 0;
k1 < intNumberOfRecordsInTestFile; k1++)
{
        // Read the corresponding test micro-batch file.                br4 = new
        BufferedReader(new FileReader(strTestingFile Names[k1]));
        tempLine = br4.readLine();
        // Skip the label record          tempLine = br4.readLine();
        // Break the line using comma as separator                tempWorkFields =
        tempLine.split(cvsSplitBy);
        dayKeyFromRecord = Double.parseDouble(tempWorkFields[0]);
        targetFunctValueFromRecord = Double.parseDouble(tempWork Fields[1]);
        // Load the corresponding test micro-batch dataset in memory
        MLDataSet testingSet =
        loadCSV2Memory(strTestingFileNames[k1],intInputNeuronNumber,
        intOutputNeuronNumber,true,CSVFormat.ENGLISH,false);
        //  Load the corresponding save network for the currently processed micro-batch
        r1 = linkToSaveNetworkDayKeys[k1];
        network =              (BasicNetwork)EncogDirectoryPersistence.loadObject(new
        File(strSaveTrainNetworkFileNames[k1]));
        // Get the results after the network optimization          int iMax = 0;
        int i = - 1;
        // Index of the array to get results
        for (MLDataPair pair:  testingSet)
        {
        i++;
                iMax = i+1;
                MLData inputData = pair.getInput();
                MLData actualData = pair.getIdeal();
                MLData predictData = network.compute(inputData);
                // These values are Normalized as the whole input is
                normInputDayFromRecord = inputData.getData(0);
                normTargetFunctValue = actualData.getData(0);
                normPredictFunctValue = predictData.getData(0);
                denormInputDayFromRecord      =      ((inputDayDl      -      inputDayDh)*
                normInputDayFromRecord - Nh*inputDayDl + inputDayDh*Nl)/          (Nl
                - Nh);
```

```
                denormTargetFunctValue = ((targetFunctValueDiffPercDl
                targetFunctValueDiffPercDh)*normTargetFunctValue
                Nh*targetFunctValueDiffPercDl + targetFunctValue
                DiffPercDh*Nl)/(Nl - Nh);
                denormPredictFunctValue =((targetFunctValueDiffPercDl
                targetFunctValueDiffPercDh)*normPredictFunctValue - Nh*
                targetFunctValueDiffPercDl + targetFunctValueDiff PercDh*Nl)/(Nl
                - Nh);
                targetToPredictFunctValueDiff = (Math.abs(denormTarget
                FunctValue - denormPredictFunctValue)/denormTargetFunct
                Value)*100;
                System.out.println("DayNumber = " + denormInputDayFrom Record + "
                targetFunctionValue = " + denormTarget FunctValue + "
                predictFunctionValue = " + denormPredict FunctValue + " valurDiff
                = " + targetToPredictFunct ValueDiff);
                if (targetToPredictFunctValueDiff > maxGlobalResultDiff)
                {
                maxGlobalIndex = iMax; maxGlobalResultDiff
                =targetToPredictFunctValueDiff; }
                sumGlobalResultDiff = sumGlobalResultDiff + targetToPredict
                FunctValueDiff;
                // Populate chart elements
                xData.add(denormInputDayFromRecord);
                yData1.add(denormTargetFunctValue);
                yData2.add(denormPredictFunctValue);
        } // End for pair loop
} // End of loop using k1
// Print the max and average results
System.out.println(" ");
averGlobalResultDiff = sumGlobalResultDiff/intNumberOfRecords InTestFile;
System.out.println("maxErrorPerc = " + maxGlobalResultDiff);
System.out.println("averErroPerc = " + averGlobalResultDiff);
}
catch (IOException e1)
{
e1.printStackTrace();
}
        // All testing batch files have been processed
        XYSeries series1 = Chart.addSeries("Actual", xData, yData1);
        XYSeries series2 = Chart.addSeries("Forecasted", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image     try     {
                BitmapEncoder.saveBitmapWithDPI(Chart, strTrainChartFileName,
                BitmapFormat.JPG, 100);
        }
```

```
        catch (Exception bt)
        {
                bt.printStackTrace();
        }
        System.out.println ("The Chart has been saved");
        System.out.println("End of testing for mini-batches training");
} // End of the method
```

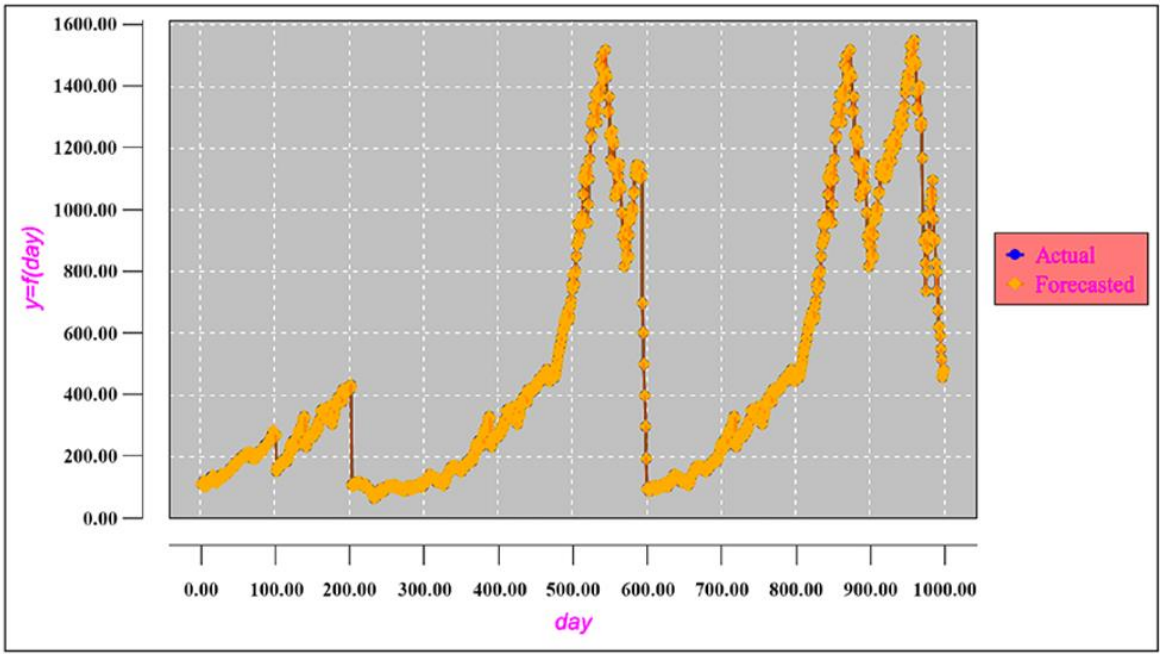
Hasil Pengujian untuk Metode Micro-Batch

Daftar 18-10 menunjukkan bagian akhir dari hasil pengujian.

Daftar 18-10. Fragmen Akhir Hasil Pengujian

```
DayNumber = 986.5  TargetValue = 899.745  AverPredictedValue = 899.74503    DiffPerc =
3.47964E-6
DayNumber = 987.5  TargetValue = 864.565  AverPredictedValue = 864.56503    DiffPerc =
3.58910E-6
DayNumber = 988.5  TargetValue = 780.485  AverPredictedValue = 780.48505    DiffPerc =
6.14256E-6
DayNumber = 989.5  TargetValue = 766.48   AverPredictedValue = 766.48000    DiffPerc =
1.62870E-7
DayNumber = 990.5  TargetValue = 735.34   AverPredictedValue = 735.33996    DiffPerc =
6.05935E-6
DayNumber = 991.5  TargetValue = 645.975  AverPredictedValue = 645.97500    DiffPerc =
4.53557E-7
DayNumber = 992.5  TargetValue = 619.23   AverPredictedValue = 619.23003    DiffPerc =
5.59670E-6
DayNumber = 993.5  TargetValue = 604.895  AverPredictedValue = 604.89504    DiffPerc =
6.02795E-6
DayNumber = 994.5  TargetValue = 568.875  AverPredictedValue = 568.87500    DiffPerc =
2.02687E-7
DayNumber = 995.5  TargetValue = 530.95   AverPredictedValue = 530.94999    DiffPerc =
1.71056E-6
DayNumber = 996.5  TargetValue = 485.01   AverPredictedValue = 485.01001    DiffPerc =
1.92301E-6
DayNumber = 997.5  TargetValue = 462.915  AverPredictedValue = 462.91499    DiffPerc =
7.96248E-8
DayNumber = 998.5  TargetValue = 475.355  AverPredictedValue = 475.35501    DiffPerc =
1.57186E-6
DayNumber = 999.5  TargetValue = 488.525  AverPredictedValue = 488.52501    DiffPerc =
1.23894E-6
maxErrorPerc = 6.840306081962611E-6
averErrorPerc  = 2.349685401959033E-6kim
```

Sekarang, hasil pengujiannya juga cukup bagus, mengingat fungsinya tidak kontinu. Bidang maxErrorPerc, yang merupakan kesalahan terburuk di antara semua catatan, kurang dari 0,0000068 persen, dan bidang averErrorPerc kurang dari 0,0000023 persen. Jika hari file uji mikro-batch saat ini tidak berada di tengah-tengah dua kunci jaringan simpan, hitung nilainya secara proporsional atau gunakan interpolasi untuk tujuan itu. Gambar 18.8 menunjukkan diagram hasil pengujian. Baik grafik yang sebenarnya dan yang diprediksi saling tumpang tindih.



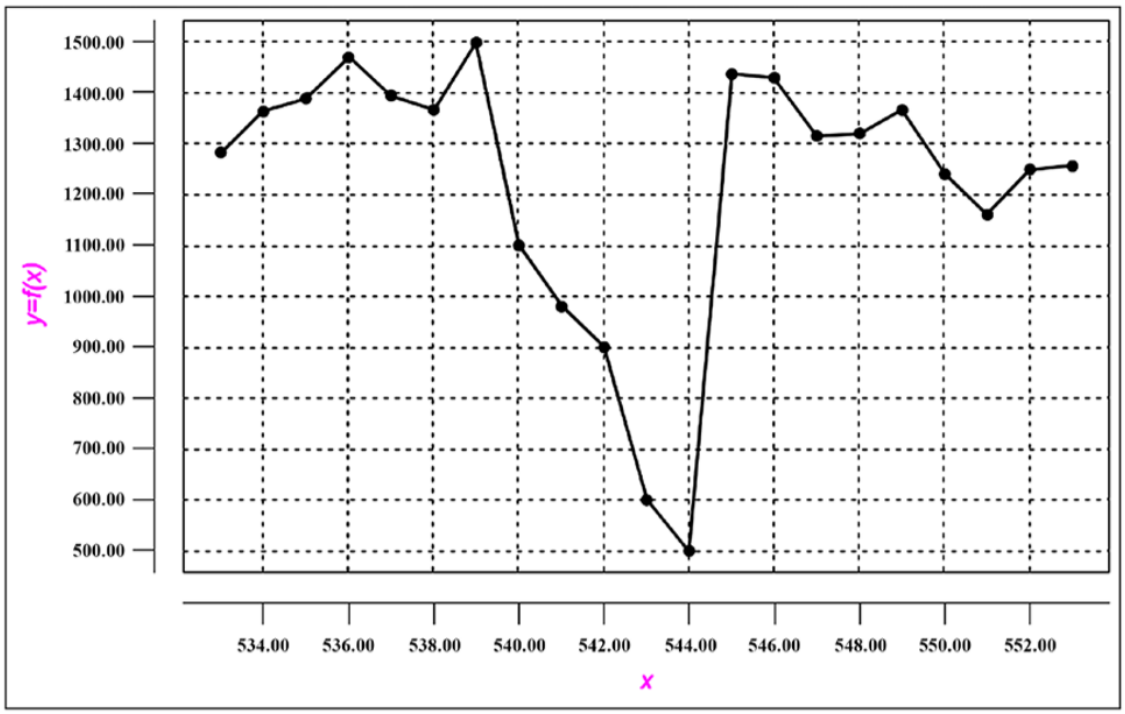
Gambar 18.8 Bagan hasil tes

Kedua grafik praktis identik dan saling tumpang tindih.

Menggali lebih dalam

Backpropagation jaringan saraf dianggap sebagai mekanisme pendekatan fungsi universal. Namun, ada batasan ketat untuk jenis fungsi yang dapat didekati oleh jaringan saraf: fungsi harus kontinu (teorema aproksimasi universal).

Mari kita bahas apa yang terjadi ketika jaringan mencoba mendekati fungsi nonkontinyu. Untuk meneliti pertanyaan ini, Anda menggunakan fungsi kecil tak kontinu yang diberikan oleh nilainya pada 20 poin. Titik-titik ini mengelilingi titik dari pola fungsi yang berubah dengan cepat. Gambar 18.9 menunjukkan diagram.



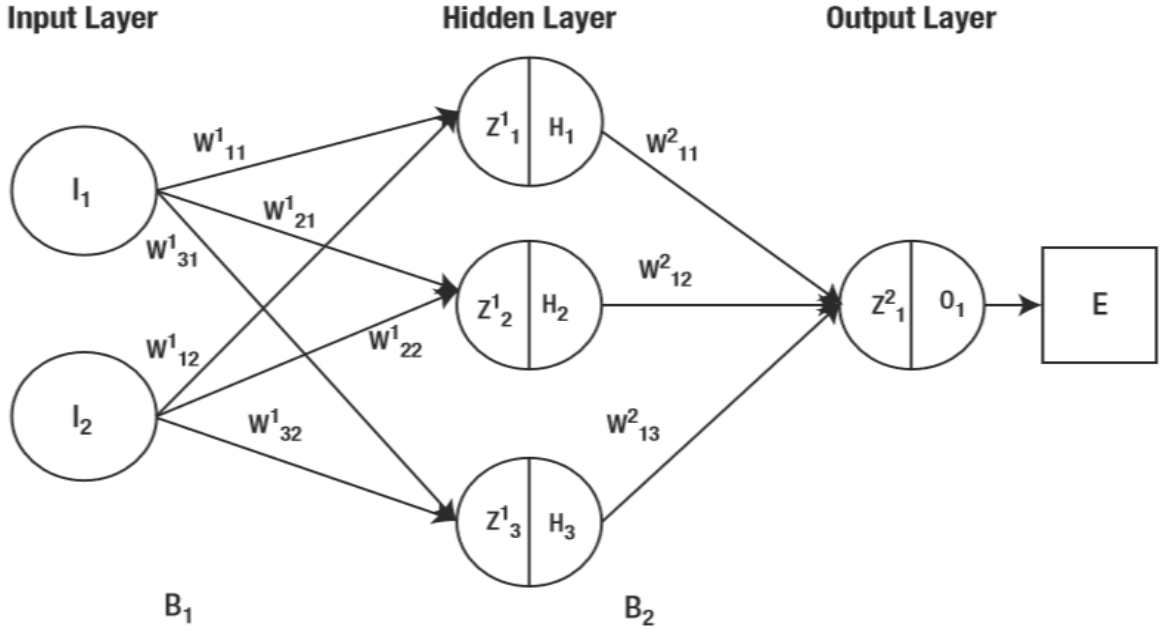
Gambar 18.9 Bagan fungsi dengan pola yang berubah dengan cepat

Tabel 18.8 menunjukkan nilai fungsi pada 20 titik.

Tabel 18.8 Nilai Fungsi

xPoint	Nilai Fungsi
533	1282.71
534	1362.93
535	1388.91
536	1469.25
537	1394.46
538	1366.42
539	1498.58
540	1100
541	980
542	900
543	600
544	500
545	1436.51
546	1429.4
547	1314.95
548	1320.28
549	1366.01
550	1239.94
551	1160.33
552	1249.46
553	1255.82

File ini dinormalisasi sebelum diproses. Gambar 18.10 menunjukkan arsitektur jaringan yang digunakan untuk mendekati fungsi ini

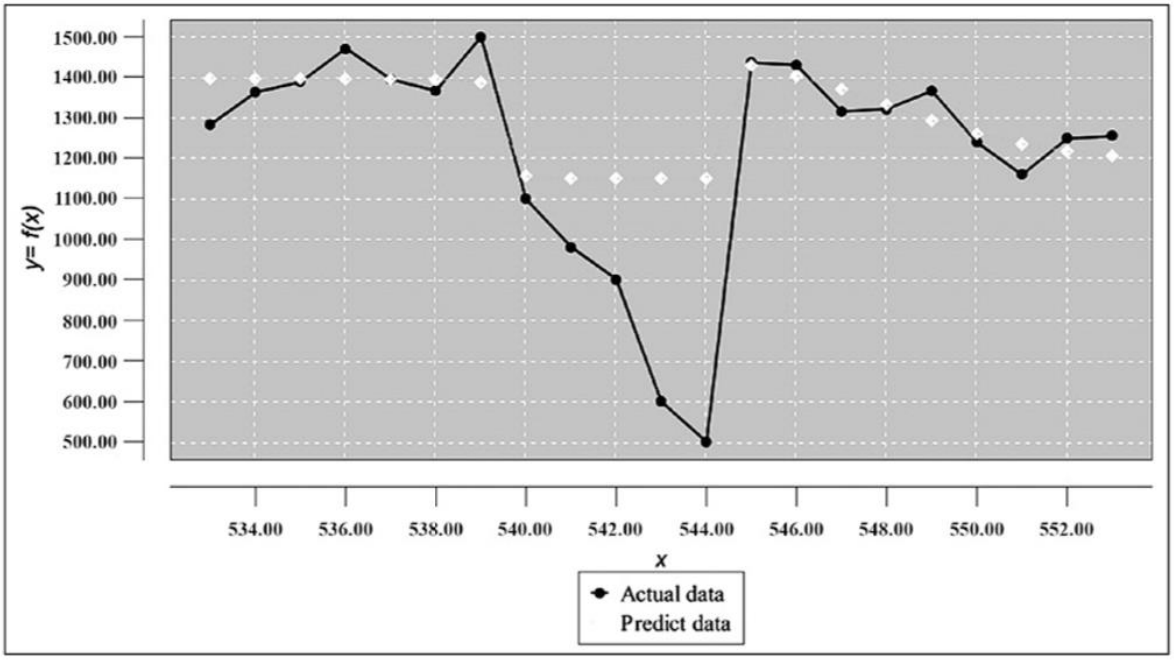


Gambar 18.10 Arsitektur jaringan

Menjalankan proses pelatihan menunjukkan hasil sebagai berikut:

- Persentase kesalahan maksimum (persentase maksimum perbedaan antara nilai fungsi yang sebenarnya dan yang diprediksi) lebih besar dari 130,06 persen.
- Rata-rata persen kesalahan (persen rata-rata perbedaan antara nilai fungsi yang sebenarnya dan yang diprediksi) lebih besar dari 16,25 persen.

Gambar 18.11 menunjukkan grafik hasil pengolahan.



Gambar 18.11 Bagan hasil pemrosesan

Tujuannya adalah untuk memahami apa yang terjadi selama proses aproksimasi fungsi nonkontinyu ini yang mengarah pada hasil yang buruk. Untuk meneliti ini, Anda akan menghitung hasil forward pass (kesalahan) untuk setiap record. Perhitungan untuk operan ke depan dilakukan dengan menggunakan Persamaan 18-1 sampai 18-5.

Neuron H_1

Persamaan 18.1

$$Z_1^1 = W_{11}^1 * I_1 + B_1 * I$$
$$H_1 = \sigma(Z_1^1)$$

Neuron H_2

Persamaan 18.2

$$Z_2^1 = W_{21}^1 * I_1 + B_1 * I$$
$$H_2 = \sigma(Z_2^1)$$

Neuron H_3

Persamaan 18.3

$$Z_3^1 = W_{31}^1 * I_1 + B_1 * I$$
$$H_3 = \sigma(Z_3^1)$$

Perhitungan ini memberikan output dari neuron H_1, H_2, dan H_3. Nilai-nilai tersebut digunakan saat memproses neuron di lapisan berikutnya (dalam hal ini, lapisan output). Lihat Persamaan 18-4 untuk neuron O_1.

$$Z_1^2 = W_{11}^2 * H_1 + W_{12}^2 * H_2 + W_{13}^2 * H_3 + B_1 * I$$
$$O_1 = \sigma(Z_1^2)$$

Persamaan 18-5 menunjukkan fungsi kesalahan.

$$E = 0.0 * (Nilai\ aktual\ untuk\ Catatan\ O_1)^2$$

Dalam Persamaan 18-1 sampai 18-3, adalah fungsi aktivasi, W adalah bobot, dan B adalah bias. Tabel 18.9 menunjukkan kesalahan yang dihitung untuk setiap catatan untuk umpan maju pertama.

Tabel 18.9 Rekam Kesalahan untuk Pass Pertama

Day	Function Value		
-0.76	-0.410177778		
-0.68	-0.053644444		
-0.6	0.061822222		
-0.52	0.418888889		
-0.44	0.086488889	**Max**	**0.202629155**
-0.36	-0.038133333	**Min**	**0.156038965**
-0.28	0.549244444		
-0.2	-1.222222222	**Difference Percent**	**29.86**
-0.12	-1.755555556		
-0.04	-2.111111111		
0.04	-3.444444444		
0.12	-3.888888889		
0.2	0.273377778		
0.28	0.241777778		
0.36	-0.266888889		
0.44	-0.2432		
0.52	-0.039955556		
0.6	-0.600266667		
0.68	-0.954088889		
0.76	-0.557955556		
0.84	-0.529688889		

Perbedaan antara nilai kesalahan maksimum dan minimum untuk semua catatan sangat besar dan sekitar 30 persen. Di situlah masalahnya ada. Ketika semua record diproses, titik ini adalah epoch. Pada saat itu, jaringan menghitung kesalahan rata-rata (untuk semua kesalahan yang diproses dalam epoch) dan kemudian memproses langkah backpropagation untuk mendistribusikan kesalahan rata-rata di antara semua neuron di lapisan output dan lapisan tersembunyi, menyesuaikan bobot dan nilai biasnya.

Kesalahan yang dihitung untuk semua catatan bergantung pada parameter bobot/bias awal (yang ditetapkan secara acak) yang ditetapkan untuk lintasan pertama ini. Ketika fungsi berisi nilai fungsi kontinu (monoton) yang secara bertahap diubah secara

berurutan, kesalahan yang dihitung untuk setiap catatan berdasarkan nilai bobot/bias awal cukup dekat, dan kesalahan rata-rata mendekati kesalahan yang dihitung untuk setiap catatan . Namun, ketika fungsinya tidak kontinu, polanya berubah dengan cepat di beberapa titik. Itu mengarah ke situasi ketika nilai bobot/bias awal yang dipilih secara acak tidak baik untuk semua catatan, yang mengarah ke perbedaan besar antara kesalahan catatan.

Selanjutnya, backpropagation menyesuaikan nilai bobot/bias awal dari neuron, tetapi masalah tetap ada: nilai yang disesuaikan tersebut tidak baik untuk semua record yang memiliki pola fungsi yang berbeda (topologi).

Tips

metode mikro-batch membutuhkan lebih banyak perhitungan daripada cara pemrosesan jaringan konvensional, sehingga hanya digunakan ketika metode konvensional tidak dapat memberikan hasil aproksimasi yang baik.

18.7 RINGKASAN

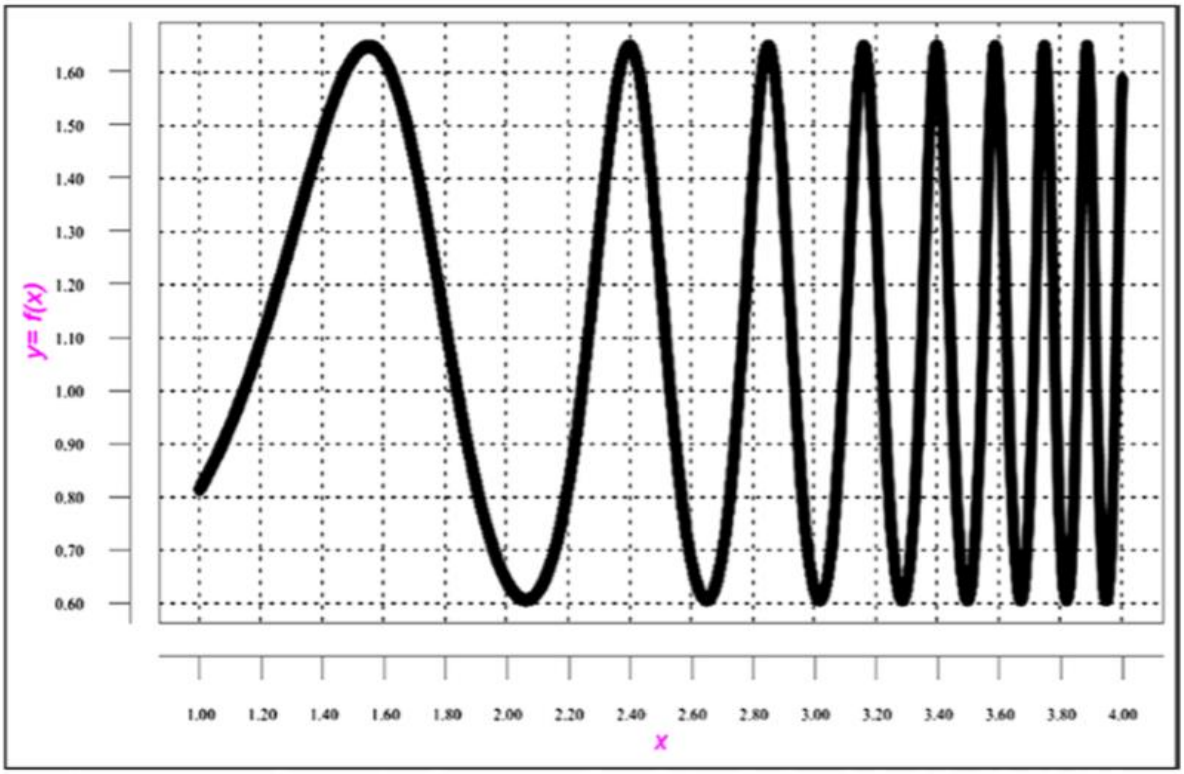
Pendekatan jaringan saraf fungsi noncontinuous adalah tugas yang sulit untuk jaringan saraf. Praktis tidak mungkin untuk mendapatkan perkiraan berkualitas baik untuk fungsi tersebut. Bab ini memperkenalkan metode mikro-batch, yang mampu mendekati fungsi non-kontinyu dengan hasil presisi tinggi. Bab berikutnya menunjukkan bagaimana metode mikro-batch secara substansial meningkatkan hasil aproksimasi untuk fungsi kontinu dengan topologi kompleks.

BAB 19
MENAKSIR FUNGSI KONTINU DENGAN TOPOLOGI KOMPLEKS

Bab ini menunjukkan bahwa metode mikro-batch secara substansial meningkatkan hasil aproksimasi fungsi kontinu dengan topologi kompleks.

19.1 CONTOH 5A: PERKIRAAN FUNGSI BERKELANJUTAN DENGAN TOPOLOGI KOMPLEKS

Gambar 19.1 menunjukkan salah satu fungsi tersebut. Fungsi tersebut memiliki rumus berikut: $Y = \sqrt{e^{-(\sin(e)^x))}}$. Namun, mari kita berpura-pura bahwa rumus fungsi tidak diketahui dan itu
fungsi diberikan kepada Anda dengan nilai-nilainya pada titik-titik tertentu.



Gambar 19.1 Bagan fungsi kontinu dengan topologi yang rumit

Sekali lagi, mari kita coba memperkirakan fungsi ini menggunakan proses jaringan saraf konvensional. Tabel 19.1 menunjukkan fragmen kumpulan data pelatihan.

Tabel 19.1 Fragmen dari Kumpulan Data Pelatihan

Poin x	Nilai Fungsi
1	0.81432914
1.0003	0.814632027
1.0006	0.814935228
1.0009	0.815238744
1.0012	0.815542575
1.0015	0.815846721

1.0018	0.816151183
1.0021	0.816455961
0.0024	0.816761055
1.0027	0.817066464
1.0030	0.817372191
1.0033	0.817678233
1.0036	0.817984593
1.0039	0.818291269
1.0042	0.818598262

Tabel 19.2 menunjukkan sebuah fragmen dari kumpulan data pengujian.

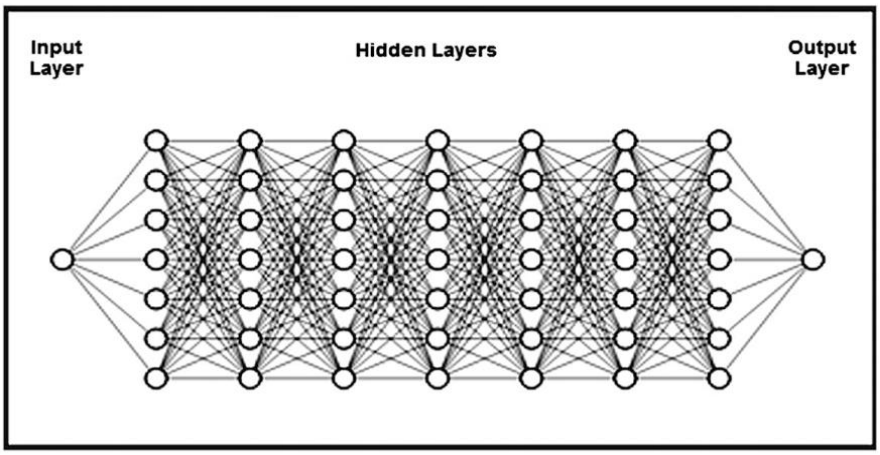
Tabel 19.2 Fragmen Kumpulan Data Pengujian

Poin x	Nilai Fungsi
1.000015	0.814344277
1.000315	0.814647179
1.000615	0.814950396
1.000915	0.815253928
1.001215	0.815557774
1.001515	0.815861937
1.001815	0.816166415
1.002115	0.816471208
1.002415	0.816776318
1.002715	0.817081743
1.003015	0.817387485
1.003315	0.817693544
1.003615	0.817999919
1.003915	0.818306611
1.004215	0.81861362

Baik set data pelatihan dan pengujian telah dinormalisasi sebelum diproses.

19.2 ARSITEKTUR JARINGAN UNTUK CONTOH 5A

Gambar 19.2 menunjukkan arsitektur jaringan.



Gambar 19.2 Arsitektur jaringan

Baik set data pelatihan dan pengujian telah dinormalisasi.

Kode Program untuk Contoh 5a

Daftar 19-1 menunjukkan kode program. Kode menunjukkan metode konvensional untuk pemrosesan jaringan saraf, yang mendekati fungsi dengan topologi kompleks (ditunjukkan pada Gambar 19.1). Seperti halnya pemrosesan jaringan saraf konvensional yang ditunjukkan pada bab-bab sebelumnya, Anda terlebih dahulu melatih jaringan tersebut.

Itu termasuk menormalkan data input pada interval [-1, 1] dan kemudian melakukan aproksimasi fungsi pada titik pelatihan. Selanjutnya, dalam mode pengujian, Anda menghitung (memprediksi) nilai fungsi pada titik yang tidak digunakan selama pelatihan jaringan. Terakhir, Anda menghitung perbedaan antara nilai fungsi aktual (yang Anda ketahui) dan nilai prediksi. Saya akan menunjukkan perbedaan antara grafik nilai aktual dan nilai prediksi.

Daftar 19-1. Kode Program

```
// ============================================================
// Perkiraan fungsi kompleks menggunakan pendekatan konvensional.
// Nilai fungsi kompleks diberikan pada 1000 titik.
//
// File input terdiri dari record dengan dua field:
// Field1 - nilai xPoint
// Field2 - Nilai fungsi pada xPoint
//
// File input dinormalisasi.
// ============================================================
package articleidi_complexformula_traditional;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
```

```
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class ArticleIDI_ComplexFormula_Traditional implements ExampleChart <XYChart>
{
        // Interval to normalize
        static double Nh =  1;
        static double Nl = -1;
        // First column
        static double minXPointDl = 0.95;
        static double maxXPointDh = 4.05;
        // Second column - target data
        static double minTargetValueDl = 0.60;
        static double maxTargetValueDh = 1.65;
        static double doublePointNumber = 0.00;
        static int intPointNumber = 0;
        static InputStream input = null;
        static double[] arrPrices = new double[2500];
        static double normInputXPointValue = 0.00;
```

```
static double normPredictXPointValue = 0.00;
static double normTargetXPointValue = 0.00;
static double normDifferencePerc = 0.00;
static double returnCode = 0.00;
static double denormInputXPointValue = 0.00;
static double denormPredictXPointValue = 0.00;
static double denormTargetXPointValue = 0.00;
static double valueDifference = 0.00;
static int numberOfInputNeurons;
static int numberOfOutputNeurons;
static int numberOfRecordsInFile;
static String trainFileName;
static String priceFileName;
static String testFileName;
static String chartTrainFileName;
static String chartTestFileName;
static String networkFileName;
static int workingMode;
static String cvsSplitBy = ",";
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static XYChart Chart;
@Override  public XYChart getChart()   {
// Create Chart
XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLACK);
        series2.setLineColor(XChartSeriesColors.YELLOW);
        series1.setMarkerColor(Color.BLACK);
        series2.setMarkerColor(Color.WHITE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.DASH_DASH);
        try     {       // Configuration
        // Set the mode the program should run       workingMode = 1;
        // Run the program in the training mode
        if(workingMode == 1)
        {
        // Training mode         numberOfRecordsInFile = 10001;         trainFileName
        =                                   "C:/Article_To_Publish/IGI_Global/Complex
        Formula_Calculate_Train_Norm.csv";
        chartTrainFileName       =       "C:/Article_To_Publish/IGI_Global/Complex
        Formula_Chart_Train_Results";
        }
        else
        {
                // Testing mode           numberOfRecordsInFile = 10001;
```

```
                testFileName = "C:/Article_To_Publish/IGI_Global/Complex
                Formula_Calculate_Test_Norm.csv";
                chartTestFileName = "C:/Article_To_Publish/IGI_Global/
                ComplexFormula_Chart_Test_Results";
            }
            // Common part of config data
            networkFileName = "C:/Article_To_Publish/IGI_Global/Complex
            Formula_Saved_Network_File.csv";
            numberOfInputNeurons = 1;
            numberOfOutputNeurons = 1;
            if(workingMode == 1)
            {
            // Training mode
            File file1 = new File(chartTrainFileName);
            File file2 = new File(networkFileName);
            if(file1.exists())
            file1.delete();
            if(file2.exists())
            file2.delete();
            returnCode = 0;
            // Clear the error Code
            do
            {
            returnCode = trainValidateSaveNetwork();
            }  while (returnCode > 0);
            }
            else
            {
            // Test mode          loadAndTestNetwork();
            }
            }
            catch (Throwable t)
            {
            t.printStackTrace();
            System.exit(1);
            }
            finally      {
    Encog.getInstance().shutdown();
    }
    Encog.getInstance().shutdown();
    return Chart;
}  // End of the method
// =============================================================
// Load CSV to memory.
// @return The loaded dataset.
// =============================================================
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal, boolean
headers, CSVFormat format, boolean significance)
```

```
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);      MLDataSet dataset
        = load.external2Memory();
        return dataset;
}
// =============================================================
// The main method.
// @param Command line arguments. No arguments are used.
// =============================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new ArticleIDI_ComplexFormula_
        Traditional();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// =============================================================
// Metode ini melatih, memvalidasi, dan menyimpan file jaringan
// yang terlatih
// =============================================================
static public double trainValidateSaveNetwork()
{
// Load the training CSV file in memory
MLDataSet trainingSet =
loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOf       OutputNeurons,
true,CSVFormat.ENGLISH,false);
// create a neural network
BasicNetwork network = new BasicNetwork();
/ Input layer     network.addLayer(new BasicLayer(null,true,1));
// Hidden layer
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
// Output layer
network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
//Train the neural network
final ResilientPropagation train = new ResilientPropagation (network, trainingSet);
int epoch = 1;
do
{
```

```
train.iteration();
System.out.println("Epoch #" + epoch + " Error:" + train.getError());
epoch++;
if (epoch >= 6000 && network.calculateError(trainingSet) > 0.101)
{
returnCode = 1;
System.out.println("Try again");
return returnCode;
}
} while(train.getError() > 0.10);
// Save the network file EncogDirectoryPersistence.saveObject(new
File(networkFileName), network);
System.out.println("Neural Network Results:");
double sumNormDifferencePerc = 0.00;
double averNormDifferencePerc = 0.00;
double maxNormDifferencePerc = 0.00;
int m = 0;
double stepValue = 0.00031;
double startingPoint = 1.00;
double xPoint = startingPoint - stepValue;
for(MLDataPair pair: trainingSet)
{
m++;
xPoint = xPoint + stepValue;
if(m == 0)
continue;
final MLData output = network.compute(pair.getInput());
MLData inputData = pair.getInput();
MLData actualData = pair.getIdeal();
MLData predictData = network.compute(inputData);
// Calculate and print the results
normInputXPointValue = inputData.getData(0);
normTargetXPointValue = actualData.getData(0);
normPredictXPointValue = predictData.getData(0);
denormInputXPointValue = ((minXPointDl - maxXPointDh)* normInputXPointValue -
Nh*minXPointDl + maxXPointDh *Nl)/ (Nl - Nh);
denormTargetXPointValue =((minTargetValueDl - maxTargetValueDh)*
normTargetXPointValue - Nh*minTargetValueDl + maxTarget ValueDh*Nl)/(Nl - Nh);
denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
normPredictXPointValue - Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
valueDifference = Math.abs(((denormTargetXPointValue -
denormPredictXPointValue)/ denormTargetXPointValue)*100.00);
System.out.println ("xPoint = " + xPoint + " denormTargetXPoint Value = " +
denormTargetXPointValue + "denormPredictXPointValue = " +
denormPredictXPointValue + " valueDifference = " + valueDifference);
sumNormDifferencePerc = sumNormDifferencePerc + valueDifference;
if (valueDifference > maxNormDifferencePerc) maxNormDifferencePerc =
valueDifference;
```

```
xData.add(xPoint);
yData1.add(denormTargetXPointValue);          yData2.add(denormPredictXPointValue);
}  // End for pair loop
        XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLACK);
        series2.setLineColor(XChartSeriesColors.YELLOW);
        series1.setMarkerColor(Color.BLACK);
        series2.setMarkerColor(Color.WHITE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.DASH_DASH);
        try
        {
        //Save the chart image                BitmapEncoder.saveBitmapWithDPI(Chart,
        chartTrainFileName,          BitmapFormat.JPG, 100);
        System.out.println ("Train Chart file has been saved") ;
        }
        catch (IOException ex)
        {
        ex.printStackTrace();
        System.exit(3);
        }
        // Finally, save this trained network      EncogDirectoryPersistence.saveObject(new
        File(networkFileName), network);
        System.out.println ("Train Network has been saved");
        averNormDifferencePerc  = sumNormDifferencePerc/(numberOfRecords InFile-1);
        System.out.println(" ");
        System.out.println("maxErrorDifferencePerc = " + maxNormDifference Perc + "
        averErrorDifferencePerc = " + averNormDifferencePerc);
        returnCode = 0.00;       return returnCode;
        }  // End of the method
// ============================================================
// Metode ini memuat dan menguji jaringan yang terlatih
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Menguji hasil jaringan");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double targetToPredictPercent = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double maxGlobalIndex = 0;
        double normInputXPointValueFromRecord = 0.00;
        double normTargetXPointValueFromRecord = 0.00;
        double normPredictXPointValueFromRecord = 0.00;
```

```
BasicNetwork network;
maxGlobalResultDiff = 0.00;
averGlobalResultDiff = 0.00;
sumGlobalResultDiff = 0.00;
// Load the test dataset into memory
MLDataSet testingSet = loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutput Neurons,true,CSVFormat.ENGLISH,false);
// Load the saved trained network network = (BasicNetwork)EncogDirectoryPersistence.loadObject(new File(network FileName));
int i = - 1;
// Index of the current record
double stepValue = 0.000298;
double startingPoint = 1.01;
double xPoint = startingPoint - stepValue;
for (MLDataPair pair:  testingSet)
{
i++;
xPoint = xPoint + stepValue;
MLData inputData = pair.getInput();
MLData actualData = pair.getIdeal();
MLData predictData = network.compute(inputData);
// These values are Normalized as the whole input is
normInputXPointValueFromRecord = inputData.getData(0);
normTargetXPointValueFromRecord = actualData.getData(0);
normPredictXPointValueFromRecord = predictData.getData(0);
denormInputXPointValue = ((minXPointDl - maxXPointDh)* normInputXPointValueFromRecord - Nh*minXPointDl + maxXPointDh*Nl)/(Nl - Nh);
de normTargetXPointValue = ((minTargetValueDl - maxTargetValueDh)* normTargetXPointValueFromRecord - Nh*minTargetValueDl + maxTarget ValueDh*Nl)/(Nl - Nh);
de normPredictXPointValue =((minTargetValueDl - maxTargetValueDh)* normPredictXPointValueFromRecord - Nh*minTargetValueDl + maxTarget ValueDh*Nl)/(Nl - Nh);
ta rgetToPredictPercent = Math.abs((denormTargetXPointValue - denormPredictXPointValue)/denormTargetXPointValue*100);
System.out.println("xPoint = " + xPoint + "  denormTargetX PointValue = " + denormTargetXPointValue + "  denormPredictX PointValue = " + denormPredictXPointValue +"  targetToPredict Percent = " + targetToPredictPercent);
if (targetToPredictPercent > maxGlobalResultDiff) maxGlobalResultDiff = targetToPredictPercent;
sumGlobalResultDiff = sumGlobalResultDiff + targetToPredict Percent;
// Populate chart elements
xData.add(xPoint);
yData1.add(denormTargetXPointValue);
yData2.add(denormPredictXPointValue);
```

```
    } // End for pair loop
    // Print the max and average results
    System.out.println(" ");
    averGlobalResultDiff = sumGlobalResultDiff/(numberOfRecordsInFile-1);
    System.out.println("maxErrorPerc = " + maxGlobalResultDiff);
    System.out.println("averErrorPerc = " + averGlobalResultDiff);
    // All testing batch files have been processed
    XYSeries series1 = Chart.addSeries("Actual", xData, yData1);
    XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLACK);
    series2.setLineColor(XChartSeriesColors.YELLOW);
    series1.setMarkerColor(Color.BLACK);
    series2.setMarkerColor(Color.WHITE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.DASH_DASH);
    // Save the chart image      try
    {
    BitmapEncoder.saveBitmapWithDPI(Chart, chartTestFileName , BitmapFormat.JPG,
    100);
    }
    catch (Exception bt)
    {
    bt.printStackTrace();
    }
    System.out.println ("The Chart has been saved");
    System.out.println("End of testing for test records");
    } // End of the method
} // End of the class
```

Hasil Pemrosesan Pelatihan untuk Contoh 5a

Daftar 19-2 menunjukkan bagian akhir dari hasil pemrosesan jaringan konvensional.

Daftar 19-2. Fragmen Akhir Hasil Latihan Konvensional

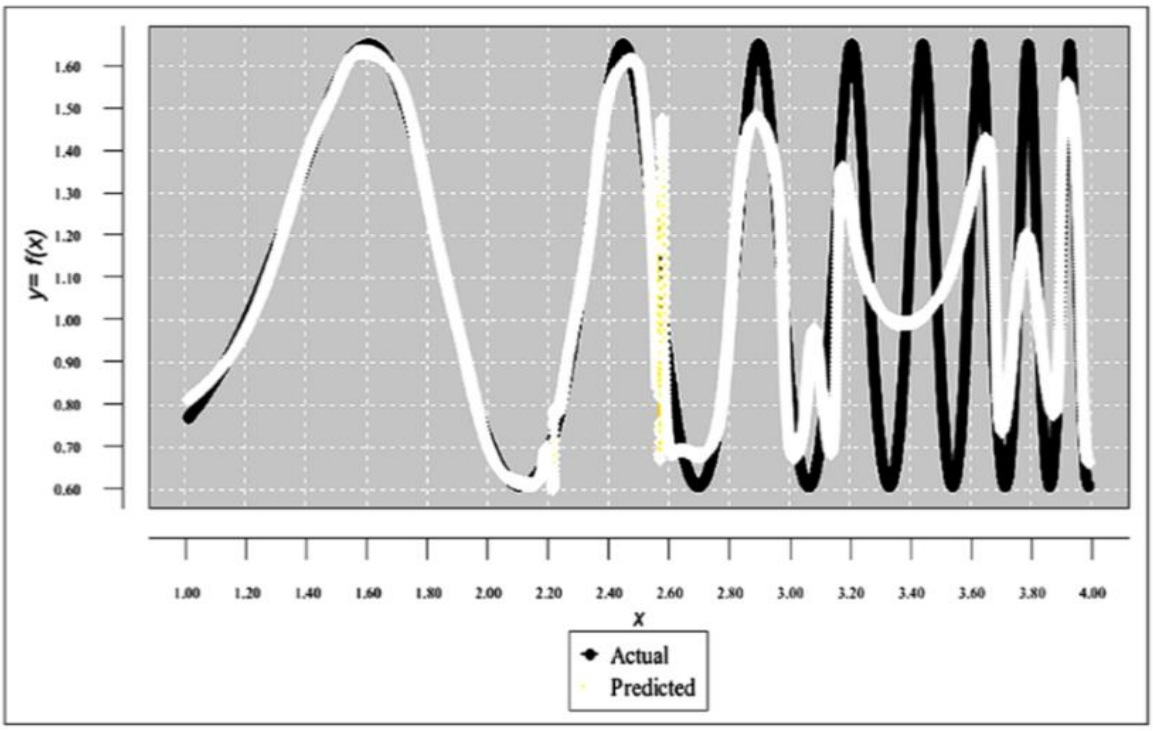
```
xPoint = 4.08605  TargetValue = 1.24795  PredictedValue = 1.15899    DifPerc = 7.12794
xPoint = 4.08636  TargetValue = 1.25699  PredictedValue = 1.16125    DifPerc = 7.61624
xPoint = 4.08667  TargetValue = 1.26602  PredictedValue = 1.16346    DifPerc = 8.10090
xPoint = 4.08698  TargetValue = 1.27504  PredictedValue = 1.16562    DifPerc = 8.58150
xPoint = 4.08729  TargetValue = 1.28404  PredictedValue = 1.16773    DifPerc = 9.05800
xPoint = 4.08760  TargetValue = 1.29303  PredictedValue = 1.16980    DifPerc = 9.53011
xPoint = 4.08791  TargetValue = 1.30199  PredictedValue = 1.17183    DifPerc = 9.99747
xPoint = 4.08822  TargetValue = 1.31093  PredictedValue = 1.17381    DifPerc = 10.4599
xPoint = 4.08853  TargetValue = 1.31984  PredictedValue = 1.17575    DifPerc = 10.9173
xPoint = 4.08884  TargetValue = 1.32871  PredictedValue = 1.17765    DifPerc = 11.3694
xPoint = 4.08915  TargetValue = 1.33755  PredictedValue = 1.17951    DifPerc = 11.8159
xPoint = 4.08946  TargetValue = 1.34635  PredictedValue = 1.18133    DifPerc = 12.25680
xPoint = 4.08978  TargetValue = 1.35510  PredictedValue = 1.18311    DifPerc = 12.69162
xPoint = 4.09008  TargetValue = 1.36380  PredictedValue = 1.18486    DifPerc = 13.12047
xPoint = 4.09039  TargetValue = 1.37244  PredictedValue = 1.18657    DifPerc = 13.54308
xPoint = 4.09070  TargetValue = 1.38103  PredictedValue = 1.18825    DifPerc = 13.95931
xPoint = 4.09101  TargetValue = 1.38956  PredictedValue = 1.18999    DifPerc = 14.36898
```

xPoint = 4.09132 TargetValue = 1.39802 PredictedValue = 1.19151 DifPerc = 14.77197
xPoint = 4.09164 TargetValue = 1.40642 PredictedValue = 1.19309 DifPerc = 15.16812
xPoint = 4.09194 TargetValue = 1.41473 PredictedValue = 1.19464 DifPerc = 15.55732
xPoint = 4.09225 TargetValue = 1.42297 PredictedValue = 1.19616 DifPerc = 15.93942
xPoint = 4.09256 TargetValue = 1.43113 PredictedValue = 1.19765 DifPerc = 16.31432
xPoint = 4.09287 TargetValue = 1.43919 PredictedValue = 1.19911 DifPerc = 16.68189
xPoint = 4.09318 TargetValue = 1.44717 PredictedValue = 1.20054 DifPerc = 17.04203
xPoint = 4.09349 TargetValue = 1.45505 PredictedValue = 1.20195 DifPerc = 17.39463
xPoint = 4.09380 TargetValue = 1.46283 PredictedValue = 1.20333 DifPerc = 17.73960
xPoint = 4.09411 TargetValue = 1.47051 PredictedValue = 1.20469 DifPerc = 18.07683
xPoint = 4.09442 TargetValue = 1.47808 PredictedValue = 1.20602 DifPerc = 18.40624
xPoint = 4.09473 TargetValue = 1.48553 PredictedValue = 1.20732 DifPerc = 18.72775
xPoint = 4.09504 TargetValue = 1.49287 PredictedValue = 1.20861 DifPerc = 19.04127
xPoint = 4.09535 TargetValue = 1.50009 PredictedValue = 1.20987 DifPerc = 19.34671
xPoint = 4.09566 TargetValue = 1.50718 PredictedValue = 1.21111 DifPerc = 19.64402
xPoint = 4.09597 TargetValue = 1.51414 PredictedValue = 1.21232 DifPerc = 19.93312
xPoint = 4.09628 TargetValue = 1.52097 PredictedValue = 1.21352 DifPerc = 20.21393
xPoint = 4.09659 TargetValue = 1.52766 PredictedValue = 1.21469 DifPerc = 20.48640
xPoint = 4.09690 TargetValue = 1.53420 PredictedValue = 1.21585 DifPerc = 20.75045
xPoint = 4.09721 TargetValue = 1.54060 PredictedValue = 1.21699 DifPerc = 21.00605
xPoint = 4.09752 TargetValue = 1.54686 PredictedValue = 1.21810 DifPerc = 21.25312
xPoint = 4.09783 TargetValue = 1.55296 PredictedValue = 1.21920 DifPerc = 21.49161
xPoint = 4.09814 TargetValue = 1.55890 PredictedValue = 1.22028 DifPerc = 21.72147
xPoint = 4.09845 TargetValue = 1.56468 PredictedValue = 1.22135 DifPerc = 21.94265
xPoint = 4.09876 TargetValue = 1.57030 PredictedValue = 1.22239 DifPerc = 22.15511
xPoint = 4.09907 TargetValue = 1.57574 PredictedValue = 1.22342 DifPerc = 22.35878
xPoint = 4.09938 TargetValue = 1.58101 PredictedValue = 1.22444 DifPerc = 22.55363
xPoint = 4.09969 TargetValue = 1.58611 PredictedValue = 1.22544 DifPerc = 22.73963
maxErrorPerc = 86.08183780343387
averErrorPerc = 10.116005438206885

Dengan proses konvensional, hasil aproksimasinya adalah sebagai berikut:

- Persentase kesalahan maksimum melebihi 86,08 persen.
- Rata-rata persen kesalahan melebihi 10. 11 persen.

Gambar 19.3 menunjukkan grafik hasil aproksimasi pelatihan menggunakan pemrosesan jaringan konvensional.



Gambar 19.3 Bagan hasil aproksimasi pelatihan menggunakan pemrosesan jaringan konvensional

Jelas, dengan perbedaan besar antara nilai aktual dan prediksi, perkiraan seperti itu tidak berguna.

19.3 FUNGSI BERKELANJUTAN DENGAN TOPOLOGI KOMPLEKS METODE MICRO-BATCH

Sekarang, mari kita perkirakan fungsi ini menggunakan metode micro-batch. Sekali lagi, set data pelatihan yang dinormalisasi dipecah menjadi satu set file pelatihan mikro, dan kemudian menjadi input untuk proses pelatihan. Daftar 19-3 menunjukkan fragmen akhir dari hasil pemrosesan pelatihan (menggunakan metode batch makro) setelah eksekusi.

Daftar 19-3. Fragmen Pengakhiran Hasil Pemrosesan Pelatihan (Menggunakan Metode Macro-Batch)

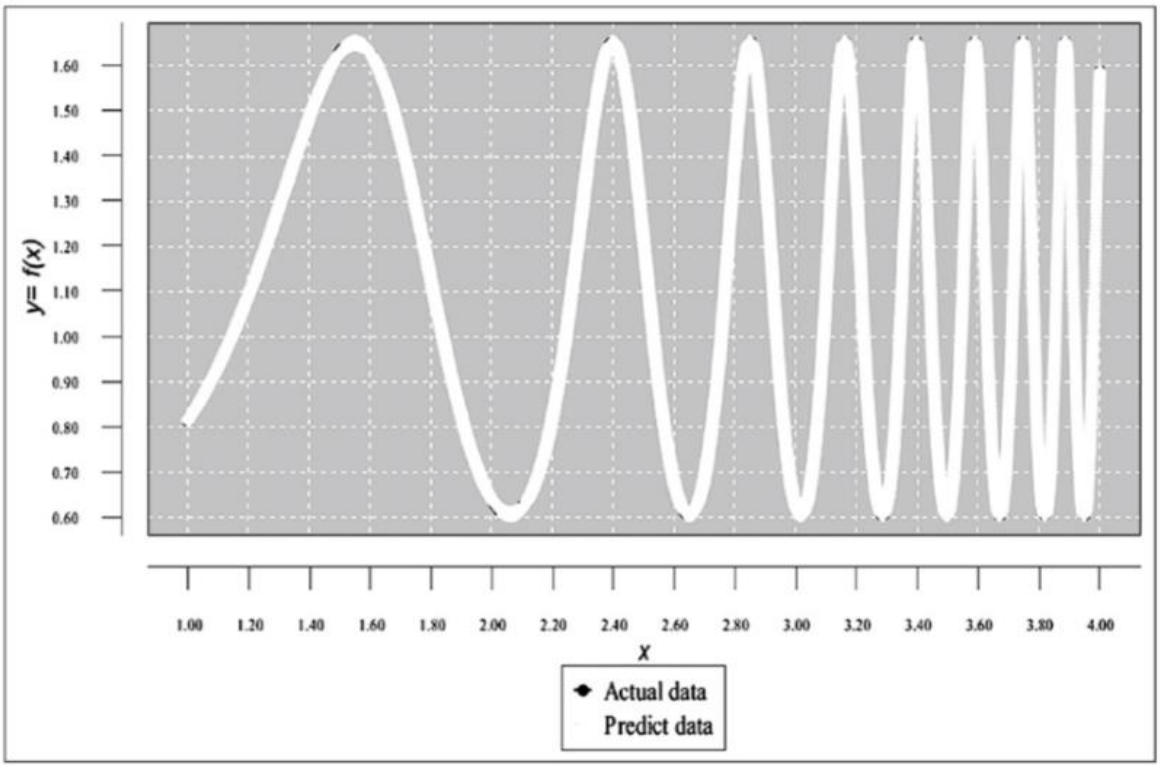
DayNumber = 9950 TargetValue = 1.19376 PredictedValue = 1.19376 DiffPerc = 4.66352E-6
DayNumber = 9951 TargetValue = 1.20277 PredictedValue = 1.20277 DiffPerc = 5.30417E-6
DayNumber = 9952 TargetValue = 1.21180 PredictedValue = 1.21180 DiffPerc = 4.79291E-6
DayNumber = 9953 TargetValue = 1.22083 PredictedValue = 1.22083 DiffPerc = 5.03070E-6
DayNumber = 9954 TargetValue = 1.22987 PredictedValue = 1.22987 DiffPerc = 3.79647E-6
DayNumber = 9955 TargetValue = 1.23891 PredictedValue = 1.23891 DiffPerc = 8.06431E-6
DayNumber = 9956 TargetValue = 1.24795 PredictedValue = 1.24795 DiffPerc = 7.19851E-6
DayNumber = 9957 TargetValue = 1.25699 PredictedValue = 1.25699 DiffPerc = 4.57148E-6
DayNumber = 9958 TargetValue = 1.26602 PredictedValue = 1.26602 DiffPerc = 5.88300E-6
DayNumber = 9959 TargetValue = 1.27504 PredictedValue = 1.27504 DiffPerc = 3.02448E-6
DayNumber = 9960 TargetValue = 1.28404 PredictedValue = 1.28404 DiffPerc = 7.04155E-6
DayNumber = 9961 TargetValue = 1.29303 PredictedValue = 1.29303 DiffPerc = 8.62206E-6
DayNumber = 9962 TargetValue = 1.30199 PredictedValue = 1.30199 DiffPerc = 9.16473E-8
DayNumber = 9963 TargetValue = 1.31093 PredictedValue = 1.31093 DiffPerc = 1.89459E-6
DayNumber = 9964 TargetValue = 1.31984 PredictedValue = 1.31984 DiffPerc = 4.16695E-6
DayNumber = 9965 TargetValue = 1.32871 PredictedValue = 1.32871 DiffPerc = 8.68118E-6
DayNumber = 9966 TargetValue = 1.33755 PredictedValue = 1.33755 DiffPerc = 4.55866E-6

DayNumber = 9967 TargetValue = 1.34635 PredictedValue = 1.34635 DiffPerc = 6.67697E-6
DayNumber = 9968 TargetValue = 1.35510 PredictedValue = 1.35510 DiffPerc = 4.80264E-6
DayNumber = 9969 TargetValue = 1.36378 PredictedValue = 1.36380 DiffPerc = 8.58688E-7
DayNumber = 9970 TargetValue = 1.37244 PredictedValue = 1.37245 DiffPerc = 5.19317E-6
DayNumber = 9971 TargetValue = 1.38103 PredictedValue = 1.38104 DiffPerc = 7.11052E-6
DayNumber = 9972 TargetValue = 1.38956 PredictedValue = 1.38956 DiffPerc = 5.15382E-6
DayNumber = 9973 TargetValue = 1.39802 PredictedValue = 1.39802 DiffPerc = 5.90734E-6
DayNumber = 9974 TargetValue = 1.40642 PredictedValue = 1.40642 DiffPerc = 6.20744E-7
DayNumber = 9975 TargetValue = 1.41473 PredictedValue = 1.41473 DiffPerc = 5.67234E-7
DayNumber = 9976 TargetValue = 1.42297 PredictedValue = 1.42297 DiffPerc = 5.54862E-6
DayNumber = 9977 TargetValue = 1.43113 PredictedValue = 1.43113 DiffPerc = 3.28318E-6
DayNumber = 9978 TargetValue = 1.43919 PredictedValue = 1.43919 DiffPerc = 7.84136E-6
DayNumber = 9979 TargetValue = 1.44717 PredictedValue = 1.44717 DiffPerc = 6.51767E-6
DayNumber = 9980 TargetValue = 1.45505 PredictedValue = 1.45505 DiffPerc = 6.59220E-6
DayNumber = 9981 TargetValue = 1.46283 PredictedValue = 1.46283 DiffPerc = 9.08060E-7
DayNumber = 9982 TargetValue = 1.47051 PredictedValue = 1.47051 DiffPerc = 8.59549E-6
DayNumber = 9983 TargetValue = 1.47808 PredictedValue = 1.47808 DiffPerc = 5.49575E-7
DayNumber = 9984 TargetValue = 1.48553 PredictedValue = 1.48553 DiffPerc = 1.07879E-6
DayNumber = 9985 TargetValue = 1.49287 PredictedValue = 1.49287 DiffPerc = 2.22734E-6
DayNumber = 9986 TargetValue = 1.50009 PredictedValue = 1.50009 DiffPerc = 1.28405E-6
DayNumber = 9987 TargetValue = 1.50718 PredictedValue = 1.50718 DiffPerc = 8.88272E-6
DayNumber = 9988 TargetValue = 1.51414 PredictedValue = 1.51414 DiffPerc = 4.91930E-6
DayNumber = 9989 TargetValue = 1.52097 PredictedValue = 1.52097 DiffPerc = 3.46714E-6
DayNumber = 9990 TargetValue = 1.52766 PredictedValue = 1.52766 DiffPerc = 7.67496E-6
DayNumber = 9991 TargetValue = 1.53420 PredictedValue = 1.53420 DiffPerc = 4.67918E-6
DayNumber = 9992 TargetValue = 1.54061 PredictedValue = 1.54061 DiffPerc = 2.20484E-6
DayNumber = 9993 TargetValue = 1.54686 PredictedValue = 1.54686 DiffPerc = 7.42466E-6
DayNumber = 9994 TargetValue = 1.55296 PredictedValue = 1.55296 DiffPerc = 3.86183E-6
DayNumber = 9995 TargetValue = 1.55890 PredictedValue = 1.55890 DiffPerc = 6.34568E-7
DayNumber = 9996 TargetValue = 1.56468 PredictedValue = 1.56468 DiffPerc = 6.23860E-6
DayNumber = 9997 TargetValue = 1.57029 PredictedValue = 1.57029 DiffPerc = 3.66380E-7
DayNumber = 9998 TargetValue = 1.57574 PredictedValue = 1.57574 DiffPerc = 4.45560E-6
DayNumber = 9999 TargetValue = 1.58101 PredictedValue = 1.58101 DiffPerc = 6.19952E-6
DayNumber = 10000 TargetValue = 1.5861 PredictedValue = 1.58611 DiffPerc = 1.34336E-6
maxGlobalResultDiff = 1.3433567671366473E-6
averGlobalResultDiff = 2.686713534273295E-10

Hasil pengolahan pelatihan (yang menggunakan metode micro-batch) adalah sebagai berikut:

1. Kesalahan maksimum kurang dari 0,00000134 persen.
2. Rata-rata kesalahan kurang dari 0,000000000269 persen.

Gambar 19.4 menunjukkan grafik hasil aproksimasi pelatihan (menggunakan metode micro-batch). Kedua grafik hampir sama (nilai sebenarnya berwarna hitam, dan nilai prediksi berwarna putih).



Gambar 19.4 Bagan hasil aproksimasi pelatihan (menggunakan metode micro-batch)

19.4 PEMROSESAN PENGUJIAN UNTUK CONTOH 5A

Seperti halnya normalisasi set data pelatihan, set data pengujian yang dinormalisasi dipecah menjadi satu set file mikro-batch yang sekarang menjadi input untuk proses pengujian.

Daftar 19-4 menunjukkan kode program.

Daftar 19-4. Kode Program

```
// ============================================================
/ Perkiraan fungsi kontinu dengan topologi kompleks
// menggunakan metode mikro-batch. Inputnya adalah kumpulan file
// mikro-batch yang dinormalisasi. Setiap micro-batch mencakup
// catatan satu hari
// yang berisi dua bidang:
// - normDayValue
// - normTargetValue
//
// Jumlah neuron inputLayer adalah 1
// Jumlah neuron outputLayer adalah 1
// ============================================================
package articleigi_complexformula_microbatchest;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
```

```
public class ArticleIGI_ComplexFormula_Microbatchest implements
ExampleChart<XYChart>
{
	// Normalization parameters
	// Normalizing interval
	static double Nh =  1;
	static double Nl = -1;
	// First 1
	static double minXPointDl = 0.95;
	static double maxXPointDh = 4.05;
	// Column 2
	static double minTargetValueDl = 0.60;
	static double maxTargetValueDh = 1.65;
	static String cvsSplitBy = ",";
	static Properties prop = null;
	static String strWorkingMode;
	static String strNumberOfBatchesToProcess;
	static String strTrainFileNameBase;
	static String strTestFileNameBase;
	static String strSaveTrainNetworkFileBase;
	static String strSaveTestNetworkFileBase;
	static String strValidateFileName;
	static String strTrainChartFileName;
	static String strTestChartFileName;
	static String strFunctValueTrainFile;
	static String strFunctValueTestFile;
	static int intDayNumber;
	static double doubleDayNumber;
	static int intWorkingMode;
	static int numberOfTrainBatchesToProcess;
	static int numberOfTestBatchesToProcess;
	static int intNumberOfRecordsInTrainFile;
	static int intNumberOfRecordsInTestFile;
	static int intNumberOfRowsInBatches;
	static int intInputNeuronNumber;
	static int intOutputNeuronNumber;
	static String strOutputFileName;
	static String strSaveNetworkFileName;
	static String strDaysTrainFileName;
	static XYChart Chart;
	static String iString;
	static double inputFunctValueFromFile;
	static double targetToPredictFunctValueDiff;
	static int[] returnCodes  = new int[3];
	static List<Double> xData = new ArrayList<Double>();
	static List<Double> yData1 = new ArrayList<Double>();
	static List<Double> yData2 = new ArrayList<Double>();
	static double[] DaysyearDayTraining = new double[10200];
```

```
        static String[] strTrainingFileNames = new String[10200];
        static String[] strTestingFileNames = new String[10200];
        static String[] strSaveTrainNetworkFileNames = new String[10200];
        static double[] linkToSaveNetworkDayKeys = new double[10200];
        static double[] linkToSaveNetworkTargetFunctValueKeys = new double[10200];
        static double[] arrTrainFunctValues = new double[10200];
        static double[] arrTestFunctValues = new double[10200];
@Override  public XYChart getChart()
{
// Create Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("day").yAxisTitle("y=f(day)").build();
// Customize Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass(). g
etSimpleName()).xAxisTitle("x").yAxisTitle("y=  f(x)").build();
// Customize Chart
Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor (ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
//Chart.getStyler().setPlotBackgroundColor(ChartColor.  getAWTColor(ChartColor.WHITE));
//Chart.getStyler().setPlotGridLinesColor(new Color(0, 0, 0));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
//Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setLegendBackgroundColor(Color.WHITE);
//Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartFontColor(Color.BLACK);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new
Font(Font.MONOSPACED,  Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
//Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendPosition(LegendPosition.OutsideS);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF,  Font.ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF,  Font.PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
// Config data
// Set the mode the program should run
intWorkingMode = 1;  // Training mode
if ( intWorkingMode == 1)
{
```

```
numberOfTrainBatchesToProcess = 10000;
numberOfTestBatchesToProcess = 9999;
intNumberOfRowsInBatches = 1;
intInputNeuronNumber = 1;
intOutputNeuronNumber = 1;
strTrainFileNameBase = "C:/Article_To_Publish/IGI_Global/Work_Files_
ComplexFormula/ComplexFormula_Train_Norm_Batch_";
strTestFileNameBase = "C:/Article_To_Publish/IGI_Global/Work_Files_
ComplexFormula/ComplexFormula_Test_Norm_Batch_";
strSaveTrainNetworkFileBase =
"C:/Article_To_Publish/IGI_Global/Work_Files_ComplexFormula/Save_
Network_MicroBatch_";
strTrainChartFileName =
"C:/Article_To_Publish/IGI_Global/Chart_Microbatch_Train_Results.jpg";
strTestChartFileName = "
C:/Article_To_Publish/IGI_Global/Chart_Microbatch_Test_MicroBatch.jpg";
// Generate training batch file names and the corresponding
// SaveNetwork file names
intDayNumber = -1;
// Day number for the chart
for (int i = 0;
i < numberOfTrainBatchesToProcess;
i++)
{
intDayNumber++;
        iString = Integer.toString(intDayNumber);
        strOutputFileName = strTrainFileNameBase + iString + ".csv";
        strSaveNetworkFileName = strSaveTrainNetworkFileBase + iString + ".csv";
        strTrainingFileNames[intDayNumber] = strOutputFileName;
        strSaveTrainNetworkFileNames[intDayNumber] = strSaveNetworkFileName;
    }  // End the FOR loop
    // Build the array linkToSaveNetworkFunctValueDiffKeys     String tempLine;
    double tempNormFunctValueDiff = 0.00;
    double tempNormFunctValueDiffPerc = 0.00;
    double tempNormTargetFunctValueDiffPerc = 0.00;
    String[] tempWorkFields;
    try
    {
        intDayNumber = -1;
        // Day number for the chart
        for (int m = 0;
        m < numberOfTrainBatchesToProcess;
        m++)
        {
        intDayNumber++;
        BufferedReader br3 = new BufferedReader(new
        FileReader(strTrainingFileNames[intDayNumber]));
        tempLine = br3.readLine();
```

```
        // Skip the label record and zero batch record          tempLine = br3.readLine();
        // Break the line using comma as separator                    tempWorkFields =
        tempLine.split(cvsSplitBy);
        tempNormFunctValueDiffPerc = Double.parseDouble(tempWork Fields[0]);
        tempNormTargetFunctValueDiffPerc =  Double.parseDouble (tempWorkFields[1]);
        linkToSaveNetworkDayKeys[intDayNumber] = tempNormFunctValue DiffPerc;
        linkToSaveNetworkTargetFunctValueKeys[intDayNumber]                    =
        tempNormTargetFunctValueDiffPerc;
    } // End the FOR loop
}
else
        {
        // Testing mode
        // Generate testing batch file names
        intDayNumber = -1;
        for (int i = 0;
        i < numberOfTestBatchesToProcess;
        i++)
        {           intDayNumber++;
        iString = Integer.toString(intDayNumber);
        // Construct the testing batch names                    strOutputFileName =
        strTestFileNameBase + iString + ".csv";
        strTestingFileNames[intDayNumber] = strOutputFileName;
        } // End the FOR loop
    }  // End of IF
}   // End for try
catch (IOException io1)
{
        io1.printStackTrace();
        System.exit(1);
        }
        if(intWorkingMode == 1)
        {
                // Training mode
                // Load, train, and test Function Values file in memory
                loadTrainFunctValueFileInMemory();
                int paramErrorCode;
                int paramBatchNumber;
                int paramR;
                int paramDayNumber;
                int paramS;
                File file1 = new File(strTrainChartFileName);
                if(file1.exists())
                file1.delete();
                returnCodes[0] = 0;
                // Clear the error Code         returnCodes[1] = 0;
                // Set the initial batch Number to 0;         returnCodes[2] = 0;
                // Day number;
```

```
                do
                {
                paramErrorCode = returnCodes[0];
                paramBatchNumber = returnCodes[1];
                paramDayNumber = returnCodes[2];
                returnCodes                                                    =
                trainBatches(paramErrorCode,paramBatchNumber,paramDayNumber);
                } while (returnCodes[0] > 0);
        }  // End the train logic
        else
                {
                // Testing mode
                File file2 = new File(strTestChartFileName);
                if(file2.exists())
                file2.delete();
                loadAndTestNetwork();
                // End the test logic
                }
                Encog.getInstance().shutdown();
                //System.exit(0);
                return Chart;
    }  // End of method
// ========================================================
// Load CSV to memory.
// @return The loaded dataset.
// ========================================================
public static MLDataSet loadCSV2Memory(String filename, int input,  int ideal, boolean
headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        inpu, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// ========================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// ========================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new     ArticleIGI_ComplexFormula_
        Microbatchest();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ========================================================
// Metode ini melatih kumpulan sebagai jaringan1 individu
```

```
// menyimpannya dalam kumpulan data terlatih yang terpisah
// ============================================================
static public int[] trainBatches(int paramErrorCode, int paramBatchNumber,
int paramDayNumber)
{
        int rBatchNumber;
        double targetToPredictFunctValueDiff = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;

        double sumGlobalResultDiff = 0.00;
        double normInputFunctValueDiffPercFromRecord = 0.00;
        double normTargetFunctValue1 = 0.00;
        double normPredictFunctValue1 = 0.00;
        double
        denormInputDayFromRecord1;
        double denormInputFunctValueDiffPercFromRecord;
        double denormTargetFunctValue1 = 0.00;
        double denormAverPredictFunctValue11 = 0.00;
        BasicNetwork network1 = new BasicNetwork();
        // Input layer
        network1.addLayer(new BasicLayer(null,true,intInputNeuronNumber));
        // Hidden layer.
        network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        // Output layer
        network1.addLayer(new BasicLayer(new ActivationTANH(),false, intOutput
        NeuronNumber));
        network1.getStructure().finalizeStructure();
        network1.reset();
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        // Loop over batches
        intDayNumber = paramDayNumber;
        // Day number for the chart
        for (rBatchNumber = paramBatchNumber;
        rBatchNumber < numberOfTrain BatchesToProcess;
        rBatchNumber++)
        {
                intDayNumber++;
                // Load the training file in memory
```

```
        MLDataSet trainingSet =
        loadCSV2Memory(strTrainingFileNames[rBatchNumber],intInput
        NeuronNumber,intOutputNeuronNumber,true,CSVFormat.ENGLISH,false);
        // train the neural network1 ResilientPropagation train = new
        ResilientPropagation(network1, trainingSet);
        int epoch = 1;
do
{
train.iteration();
epoch++;
for (MLDataPair pair11:  trainingSet)
{
        MLData inputData1 = pair11.getInput();
        MLData actualData1 = pair11.getIdeal();
        MLData predictData1 = network1.compute(inputData1);
        // These values are Normalized as the whole input is
        normInputFunctValueDiffPercFromRecord = inputData1. getData(0);
        normTargetFunctValue1 = actualData1.getData(0);
        normPredictFunctValue1 = predictData1.getData(0);
        denormInputFunctValueDiffPercFromRecord =((minXPointDl
        maxXPointDh)*normInputFunctValueDiffPercFromRecord - Nh*
        minXPointDl + maxXPointDh*Nl)/(Nl - Nh);
        denormTargetFunctValue1 = ((minTargetValueDl - maxTarget
        ValueDh)*normTargetFunctValue1 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormAverPredictFunctValue11 =((minTargetValueDl - maxTarget
        ValueDh)*normPredictFunctValue1 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        //inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
        targetToPredictFunctValueDiff = (Math.abs(denormTarget FunctValue1 -
        denormAverPredictFunctValue11)/denormTarget FunctValue1)*100;
}
if (epoch >= 1000 && targetToPredictFunctValueDiff > 0.0000091)
{
        returnCodes[0] = 1;
        returnCodes[1] = rBatchNumber;
        returnCodes[2] = intDayNumber-1;
        return returnCodes;
}
} while(targetToPredictFunctValueDiff > 0.000009);
// This batch is optimized
// Save the network1 for the cur rend batch
EncogDirectoryPersistence.saveObject(new
File(strSaveTrainNetworkFileNames[rBatchNumber]),network1);
// Get the results after the network1 optimization     int i = - 1;
for (MLDataPair pair1:  trainingSet)
{
i++;
```

```
        MLData inputData1 = pair1.getInput();
        MLData actualData1 = pair1.getIdeal();
        MLData predictData1 = network1.compute(inputData1);
        // These values are Normalized as the whole input is
        normInputFunctValueDiffPercFromRecord = inputData1.getData(0);
        normTargetFunctValue1 = actualData1.getData(0);
        normPredictFunctValue1 = predictData1.getData(0);
        // De-normalize the obtained values
        denormInputFunctValueDiffPercFromRecord =((minXPointDl - maxXPointDh)*
        normInputFunctValueDiffPercFromRecord - Nh*minXPointDl +
        maxXPointDh*Nl)/(Nl - Nh);
        denormTargetFunctValue1 = ((minTargetValueDl - maxTargetValueDh)*
        normTargetFunctValue1 - Nh*minTargetValueDl + maxTargetValueDh*Nl)/ (Nl
        - Nh);
        denormAverPredictFunctValue11 =((minTargetValueDl - maxTargetValueDh)*
        normPredictFunctValue1 - Nh*minTargetValueDl + maxTarget ValueDh*Nl)/(Nl
        - Nh);
        //inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
        targetToPredictFunctValueDiff = (Math.abs(denormTargetFunctValue1
        denormAverPredictFunctValue11)/denormTargetFunctValue1)*100;
        System.out.println("intDayNumber = " + intDayNumber + "  target FunctionValue =
        " + denormTargetFunctValue1 + "  predictFunctionValue = " +
        denormAverPredictFunctValue11 + "  valurDiff = " + target
        ToPredictFunctValueDiff);
        if (targetToPredictFunctValueDiff > maxGlobalResultDiff) maxGlobalResultDiff
        =targetToPredictFunctValueDiff;
        sumGlobalResultDiff = sumGlobalResultDiff +targetToPredictFunct ValueDiff;
        // Populate chart elements
        //doubleDayNumber = (double) rBatchNumber+1;
        xData.add(denormInputFunctValueDiffPercFromRecord);
        yData1.add(denormTargetFunctValue1);
        yData2.add(denormAverPredictFunctValue11);
    } // End for FunctValue pair1 loop
} // End of the loop over batches
        sumGlobalResultDiff = sumGlobalResultDiff  +targetToPredictFunct ValueDiff;
        averGlobalResultDiff = sumGlobalResultDiff/numberOfTrainBatchesTo Process;
        // Print the max and average results
        System.out.println(" ");
        System.out.println(" ");
        System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff);
        System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
        XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLACK);
        series2.setLineColor(XChartSeriesColors.YELLOW);
        series1.setMarkerColor(Color.BLACK);
        series2.setMarkerColor(Color.WHITE);
        series1.setLineStyle(SeriesLines.SOLID);
```

```
        series2.setLineStyle(SeriesLines.DASH_DASH);
        // Save the chart image    try
        {
        BitmapEncoder.saveBitmapWithDPI(Chart,                    strTrainChartFileName,
        BitmapFormat.JPG, 100);
        }
        catch (Exception bt)
        {
        bt.printStackTrace();
        }
        System.out.println ("The Chart has been saved");
        returnCodes[0] = 0;
        returnCodes[1] = 0;
        returnCodes[2] = 0;
        return returnCodes;
} // End of method
// ============================================================
// Muat jaringan 1 terlatih yang disimpan sebelumnya dan
// uji dengan
// memproses catatan Tes
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the network1s results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double targetToPredictFunctValueDiff = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double maxGlobalIndex = 0;
        double normInputDayFromRecord1 = 0.00;
        double normTargetFunctValue1 = 0.00;
        double normPredictFunctValue1 = 0.00;
        double denormInputDayFromRecord1 = 0.00;
        double denormTargetFunctValue1 = 0.00;
        double denormAverPredictFunctValue1 = 0.00;
        double normInputDayFromRecord2 = 0.00;
        double normTargetFunctValue2 = 0.00;
        double normPredictFunctValue2 = 0.00;
        double denormInputDayFromRecord2 = 0.00;
        double denormTargetFunctValue2 = 0.00;
        double denormAverPredictFunctValue2 = 0.00;
        double normInputDayFromTestRecord = 0.00;
        double denormInputDayFromTestRecord = 0.00;
        double denormAverPredictFunctValue = 0.00;
        double denormTargetFunctValueFromTestRecord = 0.00;
```

```
String tempLine;
String[] tempWorkFields;
double dayKeyFromTestRecord = 0.00;
double targetFunctValueFromTestRecord = 0.00;
double r1 = 0.00;
double r2 = 0.00;
BufferedReader br4;
BasicNetwork network1;
BasicNetwork network2;    int k1 = 0;    int k3 = 0;
try
{
        // Process testing records
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        for (k1 = 0; k1 < numberOfTestBatchesToProcess;
        k1++)
        {
        // if(k1 == 9998)
        //   k1 = k1;
        // Read the corresponding test micro-batch file.
        br4 = new BufferedReader(new FileReader(strTestingFileNames[k1]));
        tempLine = br4.readLine();
        // Skip the label record
        tempLine = br4.readLine();
        // Break the line using comma as separator          tempWorkFields =
        tempLine.split(cvsSplitBy);
        dayKeyFromTestRecord = Double.parseDouble(tempWorkFields[0]);
        targetFunctValueFromTestRecord          =          Double.parseDouble
        (tempWorkFields[1]);
        //          De-normalize          the          dayKeyFromTestRecord
        denormInputDayFromTestRecord = ((minXPointDl - maxXPointDh)*
        dayKeyFromTestRecord - Nh*minXPointDl + maxXPointDh*Nl)/          (Nl -
        Nh);
        //          De-normalize          the          targetFunctValueFromTestRecord
        denormTargetFunctValueFromTestRecord          =          ((minTargetValueDl
        maxTargetValueDh)*targetFunctValueFromTestRecord          -          Nh*
        minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
        / / Load the corresponding training micro-batch dataset in memory
        MLDataSet          trainingSet1          =          loadCSV2Memory(strTrainingFile
        Names[k1],intInputNeuronNumber,intOutputNeuronNumber,true,
        CSVFormat.ENGLISH,false);
        //MLDataSet testingSet =
        //    loadCSV2Memory(strTestingFileNames[k1], intInputNeuronNumber,
        //     intOutputNeuronNumber,true,CSVFormat.ENGLISH,false);
        network1          =                    (BasicNetwork)EncogDirectoryPersistence.
        loadObject(new File(strSaveTrainNetworkFileNames[k1]));
        // Get the results after the network1 optimization          int iMax = 0;
```

```
                int i = - 1;
                // Index of the array to get results
                for (MLDataPair pair1:  trainingSet1)
                {
                i++;
                iMax = i+1;
                MLData inputData1 = pair1.getInput();
                MLData actualData1 = pair1.getIdeal();
                MLData predictData1 = network1.compute(inputData1);
                // These values are Normalized normInputDayFromRecord1 =
                inputData1.getData(0);
                normTargetFunctValue1 = actualData1.getData(0);
                normPredictFunctValue1 = predictData1.getData(0);
                // De-normalize the obtained values denormInputDayFromRecord1
                = ((minXPointDl - maxXPointDh)* normInputDayFromRecord1 -
                Nh*minXPointDl + maxXPointDh*Nl)/(Nl - Nh);
                denormTargetFunctValue1 = ((minTargetValueDl - maxTarget
                ValueDh)*normTargetFunctValue1 - Nh*minTargetValueDl +
                maxTargetValueDh*Nl)/(Nl - Nh);
                denormAverPredictFunctValue1 =((minTargetValueDl
                maxTargetValueDh)*normPredictFunctValue1 - Nh*minTarget
                ValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
} // End for pair1
// ========================================================
// Sekarang hitung semuanya lagi untuk SaveNetwork (yang
// kuncinya lebih besar dari nilai dayKeyFromTestRecord)
// di memori
// ========================================================
MLDataSet trainingSet2 = loadCSV2Memory(strTrainingFileNames
[k1+1],intInputNeuronNumber,
intOutputNeuronNumber,true,CSVFormat.ENGLISH,false);
network2 = (BasicNetwork)EncogDirectoryPersistence.loadObject (new
File(strSaveTrainNetworkFileNames[k1+1]));
// Get the results after the network1 optimization
        iMax = 0;
        i = - 1; // Index of the array to get results
        for (MLDataPair pair2:  trainingSet2)
        {
                i++;
                iMax = i+1;
                MLData inputData2 = pair2.getInput();
                MLData actualData2 = pair2.getIdeal();
                MLData predictData2 = network2.compute(inputData2);
                // These values are Normalized
                normInputDayFromRecord2 = inputData2.getData(0);
                normTargetFunctValue2 = actualData2.getData(0);
                normPredictFunctValue2 = predictData2.getData(0);
```

```
                // De-normalize the obtained values          denormInputDayFromRecord2
                = ((minXPointDl - maxXPointDh)*              normInputDayFromRecord2 -
                Nh*minXPointDl + maxX          PointDh*Nl)/(Nl - Nh);
                denormTargetFunctValue2    =    ((minTargetValueDl    -    maxTarget
                ValueDh)*normTargetFunctValue2    -    Nh*minTargetValueDl    +
                maxTargetValueDh*Nl)/(Nl - Nh);
                denormAverPredictFunctValue2                    =((minTargetValueDl
                maxTargetValueDh)*normPredictFunctValue2 - Nh*minTarget ValueDl +
                maxTargetValueDh*Nl)/(Nl - Nh);
            } // End for pair1 loop
            //      Get    the    average    of    the    denormAverPredictFunctValue1    and
            denormAverPredictFunctValue2              denormAverPredictFunctValue =
            (denormAverPredictFunctValue1 +       denormAverPredictFunctValue2)/2;
            targetToPredictFunctValueDiff      =      (Math.abs(denormTargetFunct
            ValueFromTestRecord      -      enormAverPredictFunctValue)/
            ddenormTargetFunctValueFromTestRecord)*100;
            System.out.println("Record Number = " + k1 + "  DayNumber = " +
            denormInputDayFromTestRecord    +    "          denormTargetFunctValue
            FromTestRecord = " + denormTargetFunctValueFromTestRecord +          "
            denormAverPredictFunctValue = " + denormAverPredictFunct Value + "  valurDiff =
            " + targetToPredictFunctValueDiff);
            if (targetToPredictFunctValueDiff > maxGlobalResultDiff)
            {
            maxGlobalIndex = iMax;
            maxGlobalResultDiff =targetToPredictFunctValueDiff;
            }
            sumGlobalResultDiff = sumGlobalResultDiff + targetToPredict      FunctValueDiff;
            // Populate chart elements
            xData.add(denormInputDayFromTestRecord);
            yData1.add(denormTargetFunctValueFromTestRecord);
            yData2.add(denormAverPredictFunctValue);
        }  // End of loop using k1
// Print the max and average results
        System.out.println(" ");
        averGlobalResultDiff = sumGlobalResultDiff/numberOfTestBatches ToProcess;
        System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff +      "  i = " +
        maxGlobalIndex);                System.out.println("averGlobalResultDiff  =  "  +
        averGlobalResultDiff);
        }
        // End of TRY    catch (IOException e1)
        {
        e1.printStackTrace();
        }
        // All testing batch files have been processed
        XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLACK);
        series2.setLineColor(XChartSeriesColors.YELLOW);
```

```
    series1.setMarkerColor(Color.BLACK);
    series2.setMarkerColor(Color.WHITE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.DASH_DASH);
    // Save the chart image     try
    {
    BitmapEncoder.saveBitmapWithDPI(Chart,                    strTrainChartFileName,
    BitmapFormat.JPG, 100);
    }
    catch (Exception bt)
    {
    bt.printStackTrace();
    }
    System.out.println ("The Chart has been saved");
    System.out.println("End of testing for mini-batches training");
    } // End of the method
} // End of the  Encog class
```

Daftar 19-5 menunjukkan bagian akhir dari hasil pengujian setelah eksekusi.

Daftar 19-5. Fragmen Akhir dari Hasil Pemrosesan Pengujian

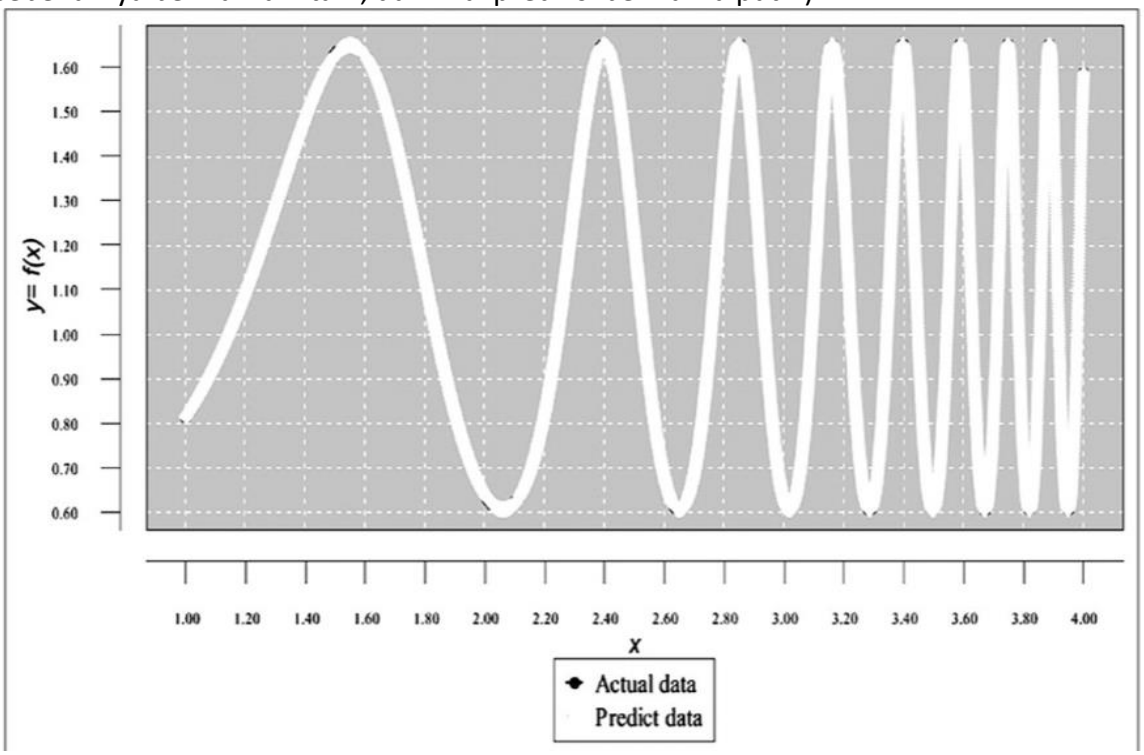
```
DayNumber = 3.98411  TargetValue = 1.17624  AverPredicedValue = 1.18028    DiffPerc = 0.34348
DayNumber = 3.98442  TargetValue = 1.18522  AverPredicedValue = 1.18927    DiffPerc = 0.34158
DayNumber = 3.98472  TargetValue = 1.19421  AverPredicedValue = 1.19827    DiffPerc = 0.33959
DayNumber = 3.98502  TargetValue = 1.20323  AverPredicedValue = 1.20729    DiffPerc = 0.33751
DayNumber = 3.98532  TargetValue = 1.21225  AverPredicedValue = 1.21631    DiffPerc = 0.33534
DayNumber = 3.98562  TargetValue = 1.22128  AverPredicedValue = 1.22535    DiffPerc = 0.33307
DayNumber = 3.98592  TargetValue = 1.23032  AverPredicedValue = 1.23439    DiffPerc = 0.33072
DayNumber = 3.98622  TargetValue = 1.23936  AverPredicedValue = 1.24343    DiffPerc = 0.32828
DayNumber = 3.98652  TargetValue = 1.24841  AverPredicedValue = 1.25247    DiffPerc = 0.32575
DayNumber = 3.98682  TargetValue = 1.25744  AverPredicedValue = 1.26151    DiffPerc = 0.32313
DayNumber = 3.98712  TargetValue = 1.26647  AverPredicedValue = 1.27053    DiffPerc = 0.32043
DayNumber = 3.98742  TargetValue = 1.27549  AverPredicedValue = 1.27954    DiffPerc = 0.31764
DayNumber = 3.98772  TargetValue = 1.28449  AverPredicedValue = 1.28854    DiffPerc = 0.31477
DayNumber = 3.98802  TargetValue = 1.29348  AverPredicedValue = 1.29751    DiffPerc = 0.31181
DayNumber = 3.98832  TargetValue = 1.30244  AverPredicedValue = 1.30646    DiffPerc = 0.30876
DayNumber = 3.98862  TargetValue = 1.31138  AverPredicedValue = 1.31538    DiffPerc = 0.30563
DayNumber = 3.98892  TargetValue = 1.32028  AverPredicedValue = 1.32428    DiffPerc = 0.30242
DayNumber = 3.98922  TargetValue = 1.32916  AverPredicedValue = 1.33313    DiffPerc = 0.29913
DayNumber = 3.98952  TargetValue = 1.33799  AverPredicedValue = 1.34195    DiffPerc = 0.29576
DayNumber = 3.98982  TargetValue = 1.34679  AverPredicedValue = 1.35072    DiffPerc = 0.29230
DayNumber = 3.99012  TargetValue = 1.35554  AverPredicedValue = 1.35945    DiffPerc = 0.28876
DayNumber = 3.99042  TargetValue = 1.36423  AverPredicedValue = 1.36812    DiffPerc = 0.28515
DayNumber = 3.99072  TargetValue = 1.37288  AverPredicedValue = 1.37674    DiffPerc = 0.28144
DayNumber = 3.99102  TargetValue = 1.38146  AverPredicedValue = 1.38530    DiffPerc = 0.27768
DayNumber = 3.99132  TargetValue = 1.38999  AverPredicedValue = 1.39379    DiffPerc = 0.27383
DayNumber = 3.99162  TargetValue = 1.39844  AverPredicedValue = 1.40222    DiffPerc = 0.26990
DayNumber = 3.99192  TargetValue = 1.40683  AverPredicedValue = 1.41057    DiffPerc = 0.26590
DayNumber = 3.99222  TargetValue = 1.41515  AverPredicedValue = 1.41885    DiffPerc = 0.26183
DayNumber = 3.99252  TargetValue = 1.42338  AverPredicedValue = 1.42705    DiffPerc = 0.25768
DayNumber = 3.99282  TargetValue = 1.43153  AverPredicedValue = 1.43516    DiffPerc = 0.25346
DayNumber = 3.99312  TargetValue = 1.43960  AverPredicedValue = 1.44318    DiffPerc = 0.24918
```

DayNumber = 3.99342 TargetValue = 1.44757 AverPredicedValue = 1.45111 DiffPerc = 0.24482
DayNumber = 3.99372 TargetValue = 1.45544 AverPredicedValue = 1.45894 DiffPerc = 0.24040
DayNumber = 3.99402 TargetValue = 1.46322 AverPredicedValue = 1.46667 DiffPerc = 0.23591
DayNumber = 3.99432 TargetValue = 1.47089 AverPredicedValue = 1.47429 DiffPerc = 0.23134
DayNumber = 3.99462 TargetValue = 1.47845 AverPredicedValue = 1.48180 DiffPerc = 0.22672
DayNumber = 3.99492 TargetValue = 1.48590 AverPredicedValue = 1.48920 DiffPerc = 0.22204
DayNumber = 3.99522 TargetValue = 1.49323 AverPredicedValue = 1.49648 DiffPerc = 0.21729
DayNumber = 3.99552 TargetValue = 1.50044 AverPredicedValue = 1.50363 DiffPerc = 0.21247
DayNumber = 3.99582 TargetValue = 1.50753 AverPredicedValue = 1.51066 DiffPerc = 0.20759
DayNumber = 3.99612 TargetValue = 1.51448 AverPredicedValue = 1.51755 DiffPerc = 0.20260
DayNumber = 3.99642 TargetValue = 1.52130 AverPredicedValue = 1.52431 DiffPerc = 0.19770
DayNumber = 3.99672 TargetValue = 1.52799 AverPredicedValue = 1.53093 DiffPerc = 0.19260
DayNumber = 3.99702 TargetValue = 1.53453 AverPredicedValue = 1.53740 DiffPerc = 0.18751
DayNumber = 3.99732 TargetValue = 1.54092 AverPredicedValue = 1.54373 DiffPerc = 0.18236
DayNumber = 3.99762 TargetValue = 1.54717 AverPredicedValue = 1.54991 DiffPerc = 0.17715
DayNumber = 3.99792 TargetValue = 1.55326 AverPredicedValue = 1.55593 DiffPerc = 0.17188
DayNumber = 3.99822 TargetValue = 1.55920 AverPredicedValue = 1.56179 DiffPerc = 0.16657
DayNumber = 3.99852 TargetValue = 1.56496 AverPredicedValue = 1.56749 DiffPerc = 0.16120
DayNumber = 3.99882 TargetValue = 1.57057 AverPredicedValue = 1.57302 DiffPerc = 0.15580
DayNumber = 3.99912 TargetValue = 1.57601 AverPredicedValue = 1.57838 DiffPerc = 0.15034
DayNumber = 3.99942 TargetValue = 1.58127 AverPredicedValue = 1.58356 DiffPerc = 0.14484
maxGlobalResultDiff = 0.3620154382225759
averGlobalResultDiff = 0.07501532301280595

Hasil pengolahan pengujian (menggunakan metode micro-batch) adalah sebagai berikut:

- Kesalahan maksimum kurang dari 0,36 persen.
- rata-rata kesalahan kurang dari 0,075 persen.

Gambar 19.5 menunjukkan grafik hasil pemrosesan pengujian (menggunakan metode micro-batch). Sekali lagi, kedua grafik sangat dekat dan praktis tumpang tindih (nilai sebenarnya berwarna hitam, dan nilai prediksi berwarna putih).



Gambar 19.5 Bagan hasil pemrosesan pengujian (menggunakan metode micro-batch)

19.5 CONTOH 5B: PERKIRAAN FUNGSI SEPERTI SPIRAL

Bagian ini melanjutkan diskusi tentang pendekatan fungsi dengan topologi yang sulit. Secara khusus, ini membahas sekelompok fungsi yang seperti spiral. Fungsi-fungsi ini memiliki properti umum; di beberapa titik, mereka memiliki beberapa nilai fungsi untuk satu titik x.

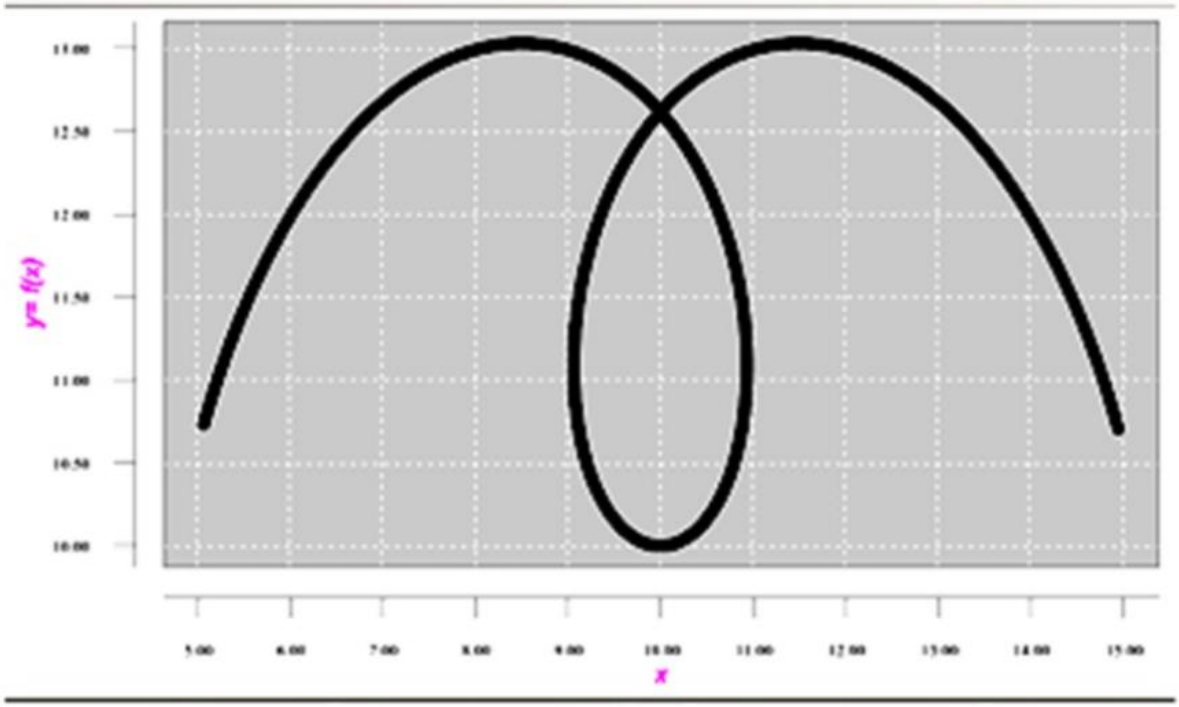
Fungsi dalam kelompok ini sangat sulit diperkirakan menggunakan jaringan saraf. Anda akan mencoba memperkirakan fungsi yang ditunjukkan pada Gambar 19.6 terlebih dahulu menggunakan metode konvensional (yang tidak berfungsi dengan baik) lalu menggunakan metode mikro-batch.

Fungsi dijelaskan oleh dua persamaan ini, di mana t adalah sudut:

$$x(t) = 10 + 0{,}5 * t * \text{Cos}(0{,}3 * t)$$

$$y(t) = 10 + 0{,}5 * t * \text{Sin}(0{,}3 * t)$$

Gambar 19.6 menunjukkan grafik yang dihasilkan dengan memplot nilai x dan y. Sekali lagi, kami berpura-pura bahwa rumus fungsi tidak diketahui dan fungsi tersebut diberikan kepada Anda dengan nilainya pada 1.000 poin. Seperti biasa, pertama-tama Anda akan mencoba memperkirakan fungsi ini dengan cara konvensional. Tabel 19.3 menunjukkan fragmen kumpulan data pelatihan.



Gambar 19.6 Fungsi dengan beberapa nilai untuk beberapa xPoints

Tabel 19.3 Fragmen dari Kumpulan Data Pelatihan

x	y
14.94996248	10.70560004
14.93574853	10.73381636
14.92137454	10.76188757
14.90684173	10.78981277
14.89215135	10.81759106
14.87730464	10.84522155
14.86230283	10.87270339

14.84714718	10.90003569
14.83183894	10.92721761
14.81637936	10.9542483
14.80076973	10.98112693
14.78501129	11.00785266
14.76910532	11.03442469
14.7530531	11.06084221
14.73685592	11.08710443
14.72051504	11.11321054
14.70403178	11.13915979
14.68740741	11.1649514
14.67064324	11.19058461
14.65374057	11.21605868
14.6367007	11.24137288
14.61952494	11.26652647
14.60221462	11.29151873
14.58477103	11.31634896
14.56719551	11.34101647
14.54948938	11.36552056
14.53165396	11.38986056
14.51369059	11.41403579
14.49560061	11.43804562
14.47738534	11.46188937
14.45904613	11.48556643

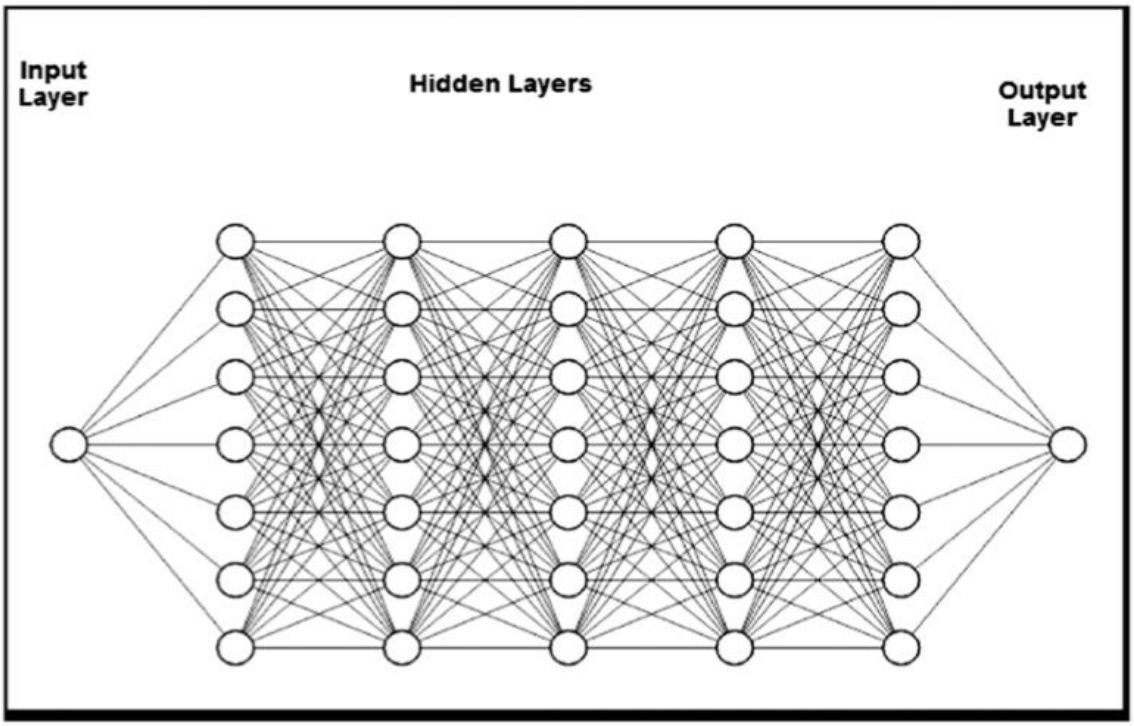
Kumpulan data pengujian disiapkan untuk poin yang tidak digunakan dalam pelatihan. Tabel 19.4 menunjukkan fragmen kumpulan data pengujian.

Tabel 19.4 Fragmen Kumpulan Data Pengujian

x	y
14.9499625	10.70560004
14.9357557	10.73380229
14.921389	10.76185957
14.9068637	10.78977099
14.8921809	10.81753565
14.8773419	10.84515266
14.8623481	10.87262116
14.8472005	10.89994029
14.8319005	10.92710918
14.8164493	10.954127
14.8008481	10.98099291
14.7850984	11.00770609
14.7692012	11.03426572
14.7531579	11.060671
14.7369698	11.08692113
14.7206381	11.11301533
14.7041642	11.13895282
14.6875493	11.16473284

14.6707947	11.19035462
14.6539018	11.21581743
14.6368718	11.24112053
14.619706	11.26626319
14.6024058	11.29124469
14.5849724	11.31606434
14.5674072	11.34072143
14.5497115	11.36521527
14.5318866	11.38954519
14.5139339	11.41371053
14.4958547	11.43771062
14.4776503	11.46154483
14.4593221	11.48521251

Baik set data pelatihan dan pengujian telah dinormalisasi sebelum diproses.



Gambar 19.7 Arsitektur jaringan

19.6 ARSITEKTUR JARINGAN UNTUK CONTOH 5B

Gambar 19.7 menunjukkan arsitektur jaringan.

Kode Program untuk Contoh 5b

Daftar 19-6 menunjukkan kode program dari aproksimasi menggunakan proses konvensional.

Daftar 19-6. Kode Program Proses Pendekatan Konvensional

```
// ============================================================
// Perkiraan fungsi seperti spiral menggunakan
// proses konvensional.
// File input dinormalisasi.
// ============================================================
package sample8;
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
```

```
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample8 implements ExampleChart<XYChart>
{
	/ Interval to normalize
	static double Nh =  1;
	static double Nl = -1;
	// First column
	static double minXPointDl = 1.00;
	static double maxXPointDh = 20.00;
	// Second column - target data
	static double minTargetValueDl = 1.00;
	static double maxTargetValueDh = 20.00;
	static double doublePointNumber = 0.00;
	static int intPointNumber = 0;
	static InputStream input = null;
	static double[] arrPrices = new double[2500];
	static double normInputXPointValue = 0.00;
	static double normPredictXPointValue = 0.00;
	static double normTargetXPointValue = 0.00;
	static double normDifferencePerc = 0.00;
	static double returnCode = 0.00;
	static double denormInputXPointValue = 0.00;
	static double denormPredictXPointValue = 0.00;
	static double denormTargetXPointValue = 0.00;
	static double valueDifference = 0.00;
	static int numberOfInputNeurons;
	static int numberOfOutputNeurons;
	static int intNumberOfRecordsInTestFile;
	static String trainFileName;
	static String priceFileName;
	static String testFileName;
	static String chartTrainFileName;
	static String chartTestFileName;
	static String networkFileName;
	static int workingMode;
	static String cvsSplitBy = ",";
	static int numberOfInputRecords = 0;
	static List<Double> xData = new ArrayList<Double>();
	static List<Double> yData1 = new ArrayList<Double>();
	static List<Double> yData2 = new ArrayList<Double>();
	static XYChart Chart;
	@Override  public XYChart getChart()
```

```
{
// Create Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("x").yAxisTitle("y= f(x)").build();
// Customize Chart Chart = new
XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("x").yAxisTitle("y= f(x)").build();
// Customize Chart
Chart.getStyler().setPlotBackgroundColor(ChartColor.
getAWTColor(ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
//Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor
(ChartColor.WHITE));
//Chart.getStyler().setPlotGridLinesColor(new Color(0, 0, 0));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
//Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setLegendBackgroundColor(Color.WHITE);
//Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartFontColor(Color.BLACK);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,  Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
//Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendPosition(LegendPosition.OutsideS);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font. ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font. PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
try
{
        // Configuration
        // Set the mode the program should run
        workingMode = 1;
        // Training mode
        if(workingMode == 1)
        {
        // Training mode          numberOfInputRecords = 1001;
        trainFileName                                                    =
        "/My_Neural_Network_Book/Book_Examples/Sample8_Calculate_
        Train_Norm.csv";
```

```
chartTrainFileName =
"C:/My_Neural_Network_Book/Book_Examples/
Sample8_Chart_ComplexFormula_Spiral_Train_Results.csv";
}
else
{
// Testing mode numberOfInputRecords = 1003;
intNumberOfRecordsInTestFile = 3;
testFileName = "C:/Book_Examples/Sample2_Norm.csv";
chartTestFileName = "XYLine_Test_Results_Chart";
}
// Common part of config data networkFileName =
"C:/My_Neural_Network_Book/Book_Examples/
Sample8_Saved_Network_File.csv";
numberOfInputNeurons = 1;
numberOfOutputNeurons = 1;
// Check the working mode to run
if(workingMode == 1)
{
// Training mode
File file1 = new File(chartTrainFileName);
File file2 = new File(networkFileName);
if(file1.exists())
file1.delete();
if(file2.exists())
file2.delete();
returnCode = 0;
// Clear the error Code
do
        {
        returnCode = trainValidateSaveNetwork();
        }
        while (returnCode > 0);
}
else
{
        // Test mode
        loadAndTestNetwork();
}
}
catch (Throwable t)
{
        t.printStackTrace();
        System.exit(1);
}
finally      {
Encog.getInstance().shutdown();
}
```

```
        Encog.getInstance().shutdown();
        return Chart;
    }  // End of the method
// ========================================================
// Load CSV to memory.
// @return The loaded dataset.
// ========================================================
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal, boolean
headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// ========================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// ========================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample8();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ========================================================
// Metode ini melatih, memvalidasi, dan menyimpan file
// jaringan yang dilatih
// ========================================================
static public double trainValidateSaveNetwork()
{
        // Load the training CSV file in memory
        MLDataSet                    trainingSet                    =
        loadCSV2Memory(trainFileName,numberOfInputNeurons,numberOf
        OutputNeurons,true,CSVFormat.ENGLISH,false);
        // create a neural network
        BasicNetwork network = new BasicNetwork();
        // Input layer
        network.addLayer(new BasicLayer(null,true,1));
        // Hidden layer
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,10));
```

```
// Output layer
//network.addLayer(new BasicLayer(new ActivationLOG(),false,1));
network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// train the neural network
final ResilientPropagation train = new ResilientPropagation(network, trainingSet);
int epoch = 1;
do
{
        train.iteration();
        System.out.println("Epoch #" + epoch + " Error:" +  train.getError());
        epoch++;
        if (epoch >= 11000 && network.calculateError(trainingSet) > 0.2251)
        {
        returnCode = 1;
        System.out.println("Try again");
        return returnCode;
        }
} while(train.getError() > 0.225);
// Save the network file
EncogDirectoryPersistence.saveObject(new
File(networkFileName), network);
System.out.println("Neural Network Results:");
double sumNormDifferencePerc = 0.00;
double averNormDifferencePerc = 0.00;
double maxNormDifferencePerc = 0.00;
int m = 0;
double xPointer = 0.00;
for(MLDataPair pair: trainingSet)
{
        m++;
        xPointer++;
        //if(m == 0)
        // continue;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results
        normInputXPointValue = inputData.getData(0);
        normTargetXPointValue = actualData.getData(0);
        normPredictXPointValue = predictData.getData(0);
        denormInputXPointValue = ((minXPointDl - maxXPointDh)*normInpu
        tXPointValue - Nh*minXPointDl + maxXPointDh *Nl)/(Nl - Nh);
        denormTargetXPointValue =((minTargetValueDl - maxTargetValueDh)*
        normTargetXPointValue - Nh*minTargetValueDl + maxTarget
        ValueDh*Nl)/(Nl - Nh);
```

```
                denormPredictXPointValue =((minTargetValueDl - maxTarget ValueDh)*
                normPredictXPointValue - Nh*minTargetValueDl + max
                TargetValueDh*Nl)/(Nl - Nh);
                valueDifference = Math.abs(((denormTargetXPointValue
                denormPredictXPointValue)/denormTargetXPointValue)*100.00);
                System.out.println ("Day = " + denormInputXPointValue + "
                denormTargetXPointValue = " + denormTargetXPointValue + "
                denormPredictXPointValue = " + denormPredictXPointValue + "
                valueDifference = " + valueDifference);
                //System.out.println("intPointNumber = " + intPointNumber);
                sumNormDifferencePerc = sumNormDifferencePerc + valueDifference;
                if (valueDifference > maxNormDifferencePerc)
                maxNormDifferencePerc = valueDifference;
                xData.add(denormInputXPointValue);
                yData1.add(denormTargetXPointValue);
                yData2.add(denormPredictXPointValue);
    }  // End for pair loop
    XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
    XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLACK);
    series2.setLineColor(XChartSeriesColors.LIGHT_GREY);
    series1.setMarkerColor(Color.BLACK);
    series2.setMarkerColor(Color.WHITE);
    series1.setLineStyle(SeriesLines.NONE);
    series2.setLineStyle(SeriesLines.SOLID);
    try
    {
    //Save the chart image
    BitmapEncoder.saveBitmapWithDPI(Chart, chartTrainFileName,
    BitmapFormat.JPG, 100);
    System.out.println ("Train Chart file has been saved") ;
    }
    catch (IOException ex)
    {
    ex.printStackTrace();
    System.exit(3);
    }
        // Finally, save this trained network
        EncogDirectoryPersistence.saveObject(new File(networkFileName), network);
        System.out.println ("Train Network has been saved") ;
        averNormDifferencePerc  = sumNormDifferencePerc/numberOfInput Records;
        System.out.println(" ");
        System.out.println("maxNormDifferencePerc = " + maxNormDifference Perc +
        averNormDifferencePerc = " + averNormDifferencePerc);"
        returnCode = 0.00;
        return returnCode;
    }  // End of the method
// ============================================================
```

```
// Metode ini memuat dan menguji jaringan yang dilatih
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();      List<Double> yData2 = new
        ArrayList<Double>();
        double targetToPredictPercent = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double maxGlobalIndex = 0;
        double normInputXPointValueFromRecord = 0.00;
        double normTargetXPointValueFromRecord = 0.00;
        double normPredictXPointValueFromRecord = 0.00;
        BasicNetwork network;
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        // Load the test dataset into memory
        MLDataSet testingSet =
        loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutput
        Neurons,true, CSVFormat.ENGLISH,false);
        // Load the saved trained network
        network =       (BasicNetwork)EncogDirectoryPersistence.loadObject(new
        File(network FileName));
        int i = - 1; // Index of the current record     double xPoint = -0.00;
        for (MLDataPair pair:  testingSet)
        {
                i++;
                xPoint = xPoint + 2.00;
                MLData inputData = pair.getInput();
                MLData actualData = pair.getIdeal();
                MLData predictData = network.compute(inputData);
                // These values are Normalized as the whole input is
                normInputXPointValueFromRecord = inputData.getData(0);
                normTargetXPointValueFromRecord = actualData.getData(0);
                normPredictXPointValueFromRecord = predictData.getData(0);
                denormInputXPointValue = ((minXPointDl - maxXPointDh)*
                normInputXPointValueFromRecord - Nh*minXPointDl + maxX
                PointDh*Nl)/(Nl - Nh);       denormTargetXPointValue = ((minTargetValueDl
                - maxTargetValueDh)*          n ormTargetXPointValueFromRecord -
                Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
                denormPredictXPointValue =((minTargetValueDl - maxTargetValueDh)*
                n ormPredictXPointValueFromRecord - Nh*minTargetValueDl +
                maxTargetValueDh*Nl)/(Nl - Nh);
```

```
                t argetToPredictPercent = Math.abs((denormTargetXPointValue -
                denormPredictXPointValue)/denormTargetXPointValue*100);
                System.out.println("xPoint = " + xPoint + "  denormTargetX PointValue = " +
                denormTargetXPointValue + "  denormPredictX PointValue = " +
                denormPredictXPointValue + " targetToPredict Percent = " +
                targetToPredictPercent);
                if (targetToPredictPercent > maxGlobalResultDiff)
                maxGlobalResultDiff = targetToPredictPercent;
                s umGlobalResultDiff = sumGlobalResultDiff + targetToPredict Percent;
                // Populate chart elements
                xData.add(xPoint);
                yData1.add(denormTargetXPointValue);
                yData2.add(denormPredictXPointValue);
        } // End for pair loop
                // Print the max and average results     System.out.println(" ");
                averGlobalResultDiff = sumGlobalResultDiff/numberOfInputRecords;
                System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff + "  i = " +
                maxGlobalIndex);
                System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
                // All testing batch files have been processed XYSeries series1 =
                Chart.addSeries("Actual", xData, yData1);
                XYSeries series2 = Chart.addSeries("Predicted", xData, yData2);
                series1.setLineColor(XChartSeriesColors.BLUE);
                series2.setMarkerColor(Color.ORANGE);
                series1.setLineStyle(SeriesLines.SOLID);
                series2.setLineStyle(SeriesLines.SOLID);
                // Save the chart image     try
                  {
                  BitmapEncoder.saveBitmapWithDPI(Chart, chartTestFileName ,
                  BitmapFormat.JPG, 100);
                }
                catch (Exception bt)
                {
                bt.printStackTrace();
                }
                System.out.println ("The Chart has been saved");
                System.out.println("End of testing for test records");
    } // End of the method
} // End of the class
```

Fungsi tersebut diperkirakan menggunakan pemrosesan jaringan konvensional. Daftar 19-7 menunjukkan bagian akhir dari hasil pengolahan konvensional.

Daftar 19-7. Fragmen Akhir Hasil Latihan Konvensional

```
Day = 5.57799  TargetValue = 11.53242  PredictedValue = 1.15068    DiffPerc = 90.02216
Day = 5.55941  TargetValue = 11.50907  PredictedValue = 1.15073    DiffPerc = 90.00153
Day = 5.54095  TargetValue = 11.48556  PredictedValue = 1.15077    DiffPerc = 89.98067
Day = 5.52261  TargetValue = 11.46188  PredictedValue = 1.15082    DiffPerc = 89.95958
Day = 5.50439  TargetValue = 11.43804  PredictedValue = 1.15086    DiffPerc = 89.93824
```

Day = 5.48630 TargetValue = 11.41403 PredictedValue = 1.15091 DiffPerc = 89.91667
Day = 5.46834 TargetValue = 11.38986 PredictedValue = 1.15096 DiffPerc = 89.89485
Day = 5.45051 TargetValue = 11.36552 PredictedValue = 1.15100 DiffPerc = 89.87280
Day = 5.43280 TargetValue = 11.34101 PredictedValue = 1.15105 DiffPerc = 89.85049
Day = 5.41522 TargetValue = 11.31634 PredictedValue = 1.15110 DiffPerc = 89.82794
Day = 5.39778 TargetValue = 11.29151 PredictedValue = 1.15115 DiffPerc = 89.80515
Day = 5.38047 TargetValue = 11.26652 PredictedValue = 1.15120 DiffPerc = 89.78210
Day = 5.36329 TargetValue = 11.24137 PredictedValue = 1.15125 DiffPerc = 89.75880
Day = 5.34625 TargetValue = 11.21605 PredictedValue = 1.15130 DiffPerc = 89.73525
Day = 5.32935 TargetValue = 11.19058 PredictedValue = 1.15134 DiffPerc = 89.71144
Day = 5.31259 TargetValue = 11.16495 PredictedValue = 1.15139 DiffPerc = 89.68737
Day = 5.29596 TargetValue = 11.13915 PredictedValue = 1.15144 DiffPerc = 89.66305
Day = 5.27948 TargetValue = 11.11321 PredictedValue = 1.15149 DiffPerc = 89.63846
Day = 5.26314 TargetValue = 11.08710 PredictedValue = 1.15154 DiffPerc = 89.61361
Day = 5.24694 TargetValue = 11.06084 PredictedValue = 1.15159 DiffPerc = 89.58850
Day = 5.23089 TargetValue = 11.03442 PredictedValue = 1.15165 DiffPerc = 89.56311
Day = 5.21498 TargetValue = 11.00785 PredictedValue = 1.15170 DiffPerc = 89.53746
Day = 5.19923 TargetValue = 10.98112 PredictedValue = 1.15175 DiffPerc = 89.51153
Day = 5.18362 TargetValue = 10.95424 PredictedValue = 1.15180 DiffPerc = 89.48534
Day = 5.16816 TargetValue = 10.92721 PredictedValue = 1.15185 DiffPerc = 89.45886
Day = 5.15285 TargetValue = 10.90003 PredictedValue = 1.15190 DiffPerc = 89.43211
Day = 5.13769 TargetValue = 10.87270 PredictedValue = 1.15195 DiffPerc = 89.40508
Day = 5.12269 TargetValue = 10.84522 PredictedValue = 1.15200 DiffPerc = 89.37776
Day = 5.10784 TargetValue = 10.81759 PredictedValue = 1.15205 DiffPerc = 89.35016
Day = 5.09315 TargetValue = 10.78981 PredictedValue = 1.15210 DiffPerc = 89.32228
Day = 5.07862 TargetValue = 10.76188 PredictedValue = 1.15215 DiffPerc = 89.29410
maxErrorPerc = 91.1677948809837
averErrorPerc = 90.04645291133258

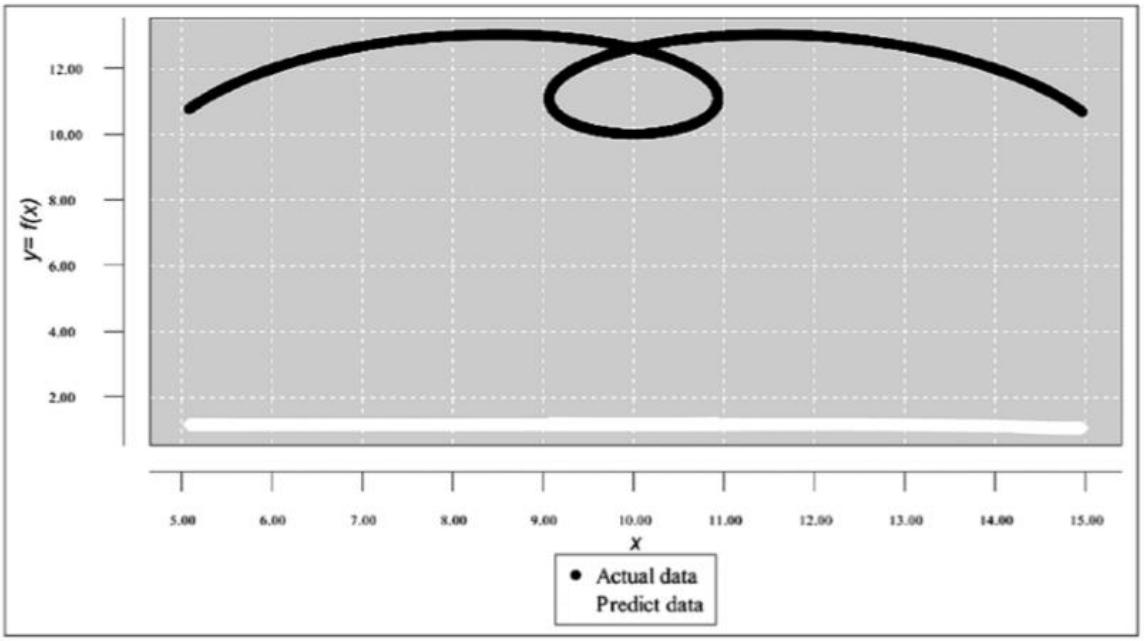
Dengan proses konvensional, hasil aproksimasinya adalah sebagai berikut:

- Persentase kesalahan maksimum melebihi 91,16 persen.
- Rata-rata persen kesalahan melebihi 90,0611 persen.

Gambar 19.8 menunjukkan grafik hasil aproksimasi pelatihan menggunakan pemrosesan jaringan konvensional.
Jelas, pendekatan seperti itu sama sekali tidak berguna.

19.7 PERKIRAAN FUNGSI YANG SAMA MENGGUNAKAN METODE *MICRO-BATCH*

Sekarang, mari kita perkirakan fungsi ini menggunakan metode micro-batch. Sekali lagi, set data pelatihan yang dinormalisasi dipecah menjadi satu set file pelatihan mikro, dan sekarang menjadi input untuk proses pelatihan. Daftar 19-8 menunjukkan kode program untuk metode pelatihan menggunakan proses mikro-batch



Gambar 19.8 Bagan hasil aproksimasi pelatihan menggunakan pemrosesan jaringan konvensional

Daftar 19-8. Kode Program untuk Metode Pelatihan Menggunakan Proses Micro-Batch

```
// ========================================================
// Perkiraan fungsi seperti spiral menggunakan metode
// micro-batch.
// Input adalah kumpulan file mikro-batch yang dinormalisasi
// (setiap mikro-batch
// menyertakan catatan satu hari). // Setiap record terdiri dari: // - normDayValue
// - normTargetValue
//
// Jumlah neuron inputLayer adalah 1
// Jumlah neuron outputLayer adalah 1
// Setiap jaringan disimpan pada disk dan peta
// dibuat untuk ditautkan setiap jaringan
// terlatih yang disimpan dengan file mikro-batch pelatihan
// yang sesuai.
// ========================================================
package sample8_microbatches;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
```

```
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient.  ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample8_Microbatches implements ExampleChart<XYChart>
{
        // Normalization parameters
        // Normalizing interval
        static double Nh =  1;
        static double Nl = -1;
        // inputFunctValueDiffPerc  static double inputDayDh = 20.00;
        static double inputDayDl = 1.00;
        // targetFunctValueDiffPerc
```

```
static double targetFunctValueDiffPercDh = 20.00;
static double targetFunctValueDiffPercDl = 1.00;
static String cvsSplitBy = ",";
static Properties prop = null;
static String strWorkingMode;
static String strNumberOfBatchesToProcess;
static String strTrainFileNameBase;
static String strTestFileNameBase;
static String strSaveTrainNetworkFileBase;
static String strSaveTestNetworkFileBase;
static String strValidateFileName;
static String strTrainChartFileName;
static String strTestChartFileName;
static String strFunctValueTrainFile;
static String strFunctValueTestFile;
static int intDayNumber;
static double doubleDayNumber;
static int intWorkingMode;
static int numberOfTrainBatchesToProcess;
static int numberOfTestBatchesToProcess;
static int intNumberOfRecordsInTrainFile;
static int intNumberOfRecordsInTestFile;
static int intNumberOfRowsInBatches;
static int intInputNeuronNumber;
static int intOutputNeuronNumber;
static String strOutputFileName;
static String strSaveNetworkFileName;
static String strDaysTrainFileName;
static XYChart Chart;
static String iString;
static double inputFunctValueFromFile;
static double targetToPredictFunctValueDiff;
static int[] returnCodes  = new int[3];
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static double[] DaysyearDayTraining = new double[1200];
static String[] strTrainingFileNames = new String[1200];
static String[] strTestingFileNames = new String[1200];
static String[] strSaveTrainNetworkFileNames = new String[1200];
static double[] linkToSaveNetworkDayKeys = new double[1200];
static double[] linkToSaveNetworkTargetFunctValueKeys = new double[1200];
static double[] arrTrainFunctValues = new double[1200];
static double[] arrTestFunctValues = new double[1200];
@Override  public XYChart getChart()
{
        // Create Chart
```

```
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("day").yAxisTitle("y=f(day)").build();
// Customize Chart
Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor
(ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,
Font.BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
// Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
Chart.getStyler().setLegendPosition(LegendPosition.OutsideE);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.ITALIC,
18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.PLAIN,
11));
//Chart.getStyler().setDayPattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
// Config data
// Set the mode the program should run    intWorkingMode = 1;
// Training mode
if(intWorkingMode == 1)
{
        numberOfTrainBatchesToProcess = 1000;
        numberOfTestBatchesToProcess = 999;
        intNumberOfRowsInBatches = 1;
        intInputNeuronNumber = 1;
        intOutputNeuronNumber = 1;
        strTrainFileNameBase =
        "C:/My_Neural_Network_Book/Book_Examples/
        Work_Files/Sample8_Microbatch_Train_";
        strTestFileNameBase =
        "C:/My_Neural_Network_Book/Book_Examples/
        Work_Files/Sample8_Microbatch_Test_";
        strSaveTrainNetworkFileBase =
        "C:/My_Neural_Network_Book/Book_
        Examples/Work_Files/Sample8_Save_Network_Batch_";
```

```
                    strTrainChartFileName =
                    "C:/Book_Examples/Sample8_Chart_Train_File_ Microbatch.jpg";
                    strTestChartFileName =
                    "C:/Book_Examples/Sample8_Chart_Test_File_ Microbatch.jpg";
                    // Generate training batches file names and the corresponding
                    // SaveNetwork file names
                    intDayNumber = -1;
                    // Day number for the chart
                    for (int i = 0;
                    i < numberOfTrainBatchesToProcess; i++)
                    {
                    intDayNumber++;
                    iString = Integer.toString(intDayNumber);
                    if (intDayNumber >= 10 & intDayNumber < 100  )
                    {
                    strOutputFileName = strTrainFileNameBase + "0" +  iString + ".csv";
                    strSaveNetworkFileName = strSaveTrainNetwork FileBase + "0" +
                    iString + ".csv";            }           else
                    {
                    if (intDayNumber < 10)
                    {
                    strOutputFileName = strTrainFileNameBase + "00" +           iString
                    + ".csv";
                    strSaveNetworkFileName = strSaveTrainNetworkFileBase + "00" +
                    iString + ".csv";
                    }
                    else
                    {
                    strOutputFileName = strTrainFileNameBase + iString + ".csv";
                    strSaveNetworkFileName = strSaveTrainNetworkFileBase +
                    iString + ".csv";
                    }
                    }
                    strTrainingFileNames[intDayNumber] = strOutputFileName;
                    strSaveTrainNetworkFileNames[intDayNumber] =
                    strSaveNetworkFileName;
} // End the FOR loop
// Build the array linkToSaveNetworkFunctValueDiffKeys     String tempLine;
double tempNormFunctValueDiff = 0.00;
double tempNormFunctValueDiffPerc = 0.00;
double tempNormTargetFunctValueDiffPerc = 0.00;
String[] tempWorkFields;
try
{
    intDayNumber = -1;
    // Day number for the chart
    for (int m = 0;
    m < numberOfTrainBatchesToProcess; m++)
```

```
    {
            intDayNumber++;
            BufferedReader br3 = new BufferedReader(new
            FileReader(strTrainingFileNames[intDayNumber]));
            tempLine = br3.readLine();
            // Skip the label record and zero batch record tempLine =
            br3.readLine();
            // Break the line using comma as separator tempWorkFields =
            tempLine.split(cvsSplitBy);
            tempNormFunctValueDiffPerc = Double.parseDouble (tempWorkFields[0]);
            tempNormTargetFunctValueDiffPerc = Double.parseDouble
            (tempWorkFields[1]);
            linkToSaveNetworkDayKeys[intDayNumber]= tempNormFunct
            ValueDiffPerc;
            linkToSaveNetworkTargetFunctValueKeys[intDayNumber] =
            tempNormTargetFunctValueDiffPerc;
    } // End the FOR loop
}
else
{
            // Testing mode. Generate testing batch file names intDayNumber = -1;
            for (int i = 0;
            i < numberOfTestBatchesToProcess;
            i++)
            {
            intDayNumber++;
            iString = Integer.toString(intDayNumber);
            // Construct the testing batch names
            if (intDayNumber >= 10 & intDayNumber < 100  )
            {
            strOutputFileName = strTestFileNameBase + "0" +  iString + ".csv";
            }
            else
            {
            if (intDayNumber < 10)
            {
            strOutputFileName = strTestFileNameBase + "00" + iString + ".csv";
            }
            else
            {
            strOutputFileName = strTestFileNameBase + iString + ".csv";
            }
            }
            strTestingFileNames[intDayNumber] = strOutputFileName;
            } // End the FOR loop
            } // End of IF
            } // End for try
            catch (IOException io1)
```

```
                {
                io1.printStackTrace();
                System.exit(1);
                }
        }
        else
        {
                // Train mode
                // Load, train, and test Function Values file in memory
                loadTrainFunctValueFileInMemory();
                int paramErrorCode;
                int paramBatchNumber;
                int paramR;
                int paramDayNumber;
                int paramS;
                File file1 = new File(strTrainChartFileName);
                if(file1.exists())
                file1.delete();
                returnCodes[0] = 0;
                // Clear the error Code
                returnCodes[1] = 0;
                // Set the initial batch Number to 0;
                returnCodes[2] = 0;
                // Day number;
                do
                {
                paramErrorCode = returnCodes[0];
                paramBatchNumber = returnCodes[1];
                paramDayNumber = returnCodes[2];
                returnCodes                                                                    =
                trainBatches(paramErrorCode,paramBatchNumber,paramDayNumber);
                }
                while (returnCodes[0] > 0);
                }
                // End the train logic       else
                {
                // Testing mode
                File file2 = new File(strTestChartFileName);
                if(file2.exists())
                file2.delete();
                loadAndTestNetwork();
                // End the test logic
                }
                Encog.getInstance().shutdown();
                //System.exit(0);
                return Chart;
    }  // End of method
// ============================================================
```

```
// Load CSV to memory.
// @return The loaded dataset.
// =========================================================
public static MLDataSet loadCSV2Memory(String filename, int input, int ideal, boolean headers, CSVFormat format, boolean significance)
{
	DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers, input, ideal, significance);
	MemoryDataLoader load = new MemoryDataLoader(codec);	MLDataSet dataset = load.external2Memory();
	return dataset;
}
// =========================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// =========================================================
static public int[] trainBatches(int paramErrorCode, int paramBatchNumber,int paramDayNumber)
{
	int rBatchNumber;
	double targetToPredictFunctValueDiff = 0;
	double maxGlobalResultDiff = 0.00;
	double averGlobalResultDiff = 0.00;
	double sumGlobalResultDiff = 0.00;
	double normInputFunctValueFromRecord = 0.00;
	double normTargetFunctValue1 = 0.00;
	double denormInputDayFromRecord;
	double denormInputFunctValueFromRecord = 0.00;
	double denormTargetFunctValue = 0.00;
	double denormPredictFunctValue1 = 0.00;
	BasicNetwork network1 = new BasicNetwork();
	// Input layer
	network1.addLayer(new BasicLayer(null,true,intInputNeuronNumber));
	// Hidden layer.
	network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
	network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
	network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
	network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
	network1.addLayer(new BasicLayer(new ActivationTANH(),true,7));
	// Output layer
	network1.addLayer(new BasicLayer(new ActivationTANH(),false, intOutputNeuronNumber));
	network1.getStructure().finalizeStructure();	network1.reset();
	maxGlobalResultDiff = 0.00;
	averGlobalResultDiff = 0.00;
	sumGlobalResultDiff = 0.00;
	// Loop over batches
	intDayNumber = paramDayNumber;
```

```
// Day number for the chart
for (rBatchNumber = paramBatchNumber;
rBatchNumber < numberOfTrain BatchesToProcess;       rBatchNumber++)
{
        intDayNumber++;
        //if(intDayNumber == 502)
        // rBatchNumber = rBatchNumber;
        // Load the training file in memory          MLDataSet trainingSet =
        loadCSV2Memory(strTrainingFileNames
        [rBatchNumber],intInputNeuronNumber,intOutputNeuronNumber,
        true,CSVFormat.ENGLISH,false);
        // train the neural network1          ResilientPropagation train = new
        ResilientPropagation(network1, trainingSet);
        int epoch = 1;
        do
        {
                train.iteration();
                epoch++;
                for (MLDataPair pair11:  trainingSet)
                {
                MLData inputData1 = pair11.getInput();
                MLData actualData1 = pair11.getIdeal();
                MLData predictData1 = network1.compute(inputData1);
                // These values are Normalized as the whole input is
                normInputFunctValueFromRecord = inputData1.getData(0);
                normTargetFunctValue1 = actualData1.getData(0);
                normPredictFunctValue1 = predictData1.getData(0);
                denormInputFunctValueFromRecord          =((inputDayDl
                inputDayDh)*normInputFunctValueFromRecord - Nh*inputDayDl +
                inputDayDh*Nl)/(Nl - Nh);
                denormTargetFunctValue      =      ((targetFunctValueDiffPercDl
                targetFunctValueDiffPercDh)*normTargetFunctValue1      -
                Nh*targetFunctValueDiffPercDl      +      targetFunctValue
                DiffPercDh*Nl)/(Nl - Nh);
                denormPredictFunctValue1      =((targetFunctValueDiffPercDl
                targetFunctValueDiffPercDh)*normPredictFunctValue1      -
                Nh*targetFunctValueDiffPercDl      +      targetFunctValueDiff
                PercDh*Nl)/(Nl - Nh);
                //inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
                targetToPredictFunctValueDiff      =      (Math.abs(denormTarget
                FunctValue      -      enormPredictFunctValue1)/ddenormTarget
                FunctValue)*100;
        }
        if (epoch >= 1000 && Math.abs(targetToPredictFunctValueDiff) > 0.0000091)
        {
        returnCodes[0] = 1;
        returnCodes[1] = rBatchNumber;
        returnCodes[2] = intDayNumber-1;
```

```
                return returnCodes;
                }
        //System.out.println("intDayNumber = " + intDayNumber);
        }
        while(Math.abs(targetToPredictFunctValueDiff) > 0.000009);
        // This batch is optimized
        // Save the network1 for the current batch
    EncogDirectoryPersistence.saveObject(newFile(strSaveTrainNetwork
        FileNames[rBatchNumber]),network1);
        // Get the results after the network1 optimization     int i = - 1;
        for (MLDataPair pair1:  trainingSet)
        {
            i++;
            MLData inputData1 = pair1.getInput();
            MLData actualData1 = pair1.getIdeal();
            MLData predictData1 = network1.compute(inputData1);
            // These values are Normalized as the whole input is
            normInputFunctValueFromRecord = inputData1.getData(0);
            normTargetFunctValue1 = actualData1.getData(0);
            normPredictFunctValue1 = predictData1.getData(0);
            // De-normalize the obtained values        denormInputFunctValueFromRecord
        =((inputDayDl - inputDayDh)*                normInputFunctValueFromRecord -
        Nh*inputDayDl + inputDayDh*Nl)/      (Nl - Nh);
            denormTargetFunctValue = ((targetFunctValueDiffPercDl - targetFunct
        ValueDiffPercDh)*normTargetFunctValue1 - DiffPercDl + target
        FunctValueDiffPercDh*Nl)/(Nl - Nh);
            denormPredictFunctValue1 =((targetFunctValueDiffPercDl - targetFunct
        ValueDiffPercDh)*normPredictFunctValue1 - Nh*targetFunctValue      DiffPercDl +
        targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
            //inputFunctValueFromFile = arrTrainFunctValues[rBatchNumber];
            targetToPredictFunctValueDiff       =       (Math.abs(denormTargetFunctValue
        denormPredictFunctValue1)/denormTargetFunctValue)*100;
            System.out.println("intDayNumber = " + intDayNumber + " target
        FunctionValue = " + denormTargetFunctValue + "  predictFunction       Value = " +
        denormPredictFunctValue1 + "  valurDiff = " +       targetToPredictFunctValueDiff);
            if          (targetToPredictFunctValueDiff          >          maxGlobalResultDiff)
        maxGlobalResultDiff =targetToPredictFunctValueDiff;
            sumGlobalResultDiff = sumGlobalResultDiff +targetToPredict FunctValueDiff;
            // Populate chart elements
            xData.add(denormInputFunctValueFromRecord);
            yData1.add(denormTargetFunctValue);
            yData2.add(denormPredictFunctValue1);
    }  // End for FunctValue pair1 loop
}  // End of the loop over batches
        sumGlobalResultDiff = sumGlobalResultDiff +targetToPredict FunctValueDiff;
        averGlobalResultDiff = sumGlobalResultDiff/numberOfTrainBatches ToProcess;
        // Print the max and average results
        System.out.println(" ");
```

```
        System.out.println(" ");
        System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff);
        System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
        XYSeries series1 = Chart.addSeries("Actual", xData, yData1);
        XYSeries series2 = Chart.addSeries("Forecasted", xData, yData2);
        series1.setMarkerColor(Color.BLACK);
        series2.setMarkerColor(Color.WHITE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image    try
        {
        BitmapEncoder.saveBitmapWithDPI(Chart,                    strTrainChartFileName,
        BitmapFormat.JPG, 100);
        }
        catch (Exception bt)
        {
        bt.printStackTrace();     }
        System.out.println ("The Chart has been saved");
        returnCodes[0] = 0;
        returnCodes[1] = 0;
        returnCodes[2] = 0;
        return returnCodes;
} // End of method
// ========================================================
// Muat jaringan 1 terlatih yang disimpan sebelumnya dan
// uji dengan
// memproses catatan Tes
// ========================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the network1s results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double targetToPredictFunctValueDiff = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double maxGlobalIndex = 0;
        double normInputDayFromRecord1 = 0.00;
        double normTargetFunctValue1 = 0.00;
        double normPredictFunctValue1 = 0.00;
        double denormInputDayFromRecord = 0.00;
        double denormTargetFunctValue = 0.00;
        double denormPredictFunctValue = 0.00;
        double normInputDayFromRecord2 = 0.00;
        double normTargetFunctValue2 = 0.00;
        double normPredictFunctValue2 = 0.00;
```

```
double denormInputDayFromRecord2 = 0.00;
double denormTargetFunctValue2 = 0.00;
double denormPredictFunctValue2 = 0.00;
double normInputDayFromTestRecord = 0.00;
double denormInputDayFromTestRecord = 0.00;
double denormTargetFunctValueFromTestRecord = 0.00;
String tempLine;
String[] tempWorkFields;
double dayKeyFromTestRecord = 0.00;
double targetFunctValueFromTestRecord = 0.00;
double r1 = 0.00;    double r2 = 0.00;
BufferedReader br4;
BasicNetwork network1;
BasicNetwork network2;
int k1 = 0;
int k3 = 0;
try
{
        // Process testing records
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        for (k1 = 0;
        k1 < numberOfTestBatchesToProcess; k1++)
        {
                if(k1 == 100)
                k1 = k1;
                // Read the corresponding test micro-batch file.          br4 = new
                BufferedReader(new FileReader(strTestingFileNames[k1]));
                tempLine = br4.readLine();
                // Skip the label record
                tempLine = br4.readLine();
                // Break the line using comma as separator          tempWorkFields =
                tempLine.split(cvsSplitBy);
                dayKeyFromTestRecord = Double.parseDouble(tempWorkFields[0]);
                targetFunctValueFromTestRecord = Double.parseDouble
                (tempWorkFields[1]);
                // De-normalize the dayKeyFromTestRecord
                denormInputDayFromTestRecord = ((inputDayDl -
                inputDayDh)*dayKeyFromTestRecord Nh*inputDayDl +
                inputDayDh*Nl)/(Nl - Nh);
                // De-normalize the targetFunctValueFromTestRecord
                denormTargetFunctValueFromTestRecord =
                ((targetFunctValueDiffPercDl - targetFunctValueDiffPercDh)*
                targetFunctValueFromTestRecord - Nh*targetFunctValueDiffPercDl
                + targetFunctValueDiffPercDh*Nl)/(Nl - Nh);
                // Load the corresponding training micro-batch dataset in memory
                MLDataSet trainingSet1 = loadCSV2Memory(strTrainingFileNames
```

```
        [k1],intInputNeuronNumber,intOutputNeuronNumber,
        true,CSVFormat.ENGLISH,false);
        //MLDataSet testingSet =
        //
        loadCSV2Memory(strTestingFileNames[k1],intInputNeuronNumber,
        //   intOutputNeuronNumber,true,CSVFormat.ENGLISH,false);
        network1 =              (BasicNetwork)EncogDirectoryPersistence.
        loadObject(new File(strSaveTrainNetworkFileNames[k1]));
        // Get the results after the network1 optimization          int iMax =
        0;
        int i = - 1;
        // Index of the array to get results
        for (MLDataPair pair1:  trainingSet1)          {
i++;
iMax = i+1;
MLData inputData1 = pair1.getInput();
MLData actualData1 = pair1.getIdeal();
MLData predictData1 = network1.compute(inputData1);
// These values are Normalized as the whole input is
normInputDayFromRecord1 = inputData1.getData(0);
normTargetFunctValue1 = actualData1.getData(0);
normPredictFunctValue1 = predictData1.getData(0);
denormInputDayFromRecord      =      ((inputDayDl      -      inputDayDh)*
normInputDayFromRecord1 - Nh*inputDayDl +          inputDayDh*Nl)/(Nl
- Nh);
        denormTargetFunctValue      =      ((targetFunctValueDiffPercDl
targetFunctValueDiffPercDh)*normTargetFunctValue1      -      Nh*
targetFunctValueDiffPercDl + targetFunctValue          DiffPercDh*Nl)/(Nl
- Nh);
        denormPredictFunctValue          =((targetFunctValueDiffPercDl
targetFunctValueDiffPercDh)*normPredictFunctValue1      -      Nh*
targetFunctValueDiffPercDl + targetFunctValue          DiffPercDh*Nl)/(Nl
- Nh);
targetToPredictFunctValueDiff = (Math.abs(denormTarget FunctValue -
denormPredictFunctValue)/denormTarget          FunctValue)*100;
System.out.println("Record Number = " + k1 + "  DayNumber = " +
denormInputDayFromTestRecord +                                    "
denormTargetFunctValueFromTestRecord      =      "      +      denormTarget
FunctValueFromTestRecord + "  denormPredictFunctValue = " +
denormPredictFunctValue + "  valurDiff = " + target
ToPredictFunctValueDiff);
if (targetToPredictFunctValueDiff > maxGlobalResultDiff)
{
        maxGlobalIndex = iMax;
        maxGlobalResultDiff =targetToPredictFunctValueDiff;
}
sumGlobalResultDiff          =          sumGlobalResultDiff          +
targetToPredictFunctValueDiff;
```

```
                // Populate chart elements
                xData.add(denormInputDayFromTestRecord);
                yData1.add(denormTargetFunctValueFromTestRecord);
                yData2.add(denormPredictFunctValue);
    } // End for pair2 loop
} // End of loop using k1
    // Print the max and average results
    System.out.println(" ");
    averGlobalResultDiff = sumGlobalResultDiff/numberOfTestBatches ToProcess;
    System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff + " i = " +
    maxGlobalIndex); System.out.println("averGlobalResultDiff = " +
    averGlobalResultDiff);
    }
    // End of TRY
        catch (FileNotFoundException nf) {
        nf.printStackTrace();
        }
        catch (IOException e1)
        {
        e1.printStackTrace();
        }
    // All testing batch files have been processed XYSeries series1 =
    Chart.addSeries("Actual", xData, yData1); XYSeries series2 =
    Chart.addSeries("Forecasted", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLACK);
    series2.setLineColor(XChartSeriesColors.LIGHT_GREY);
    series1.setMarkerColor(Color.BLACK);
    series2.setMarkerColor(Color.WHITE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.SOLID);
    // Save the chart image try
    {
    BitmapEncoder.saveBitmapWithDPI(Chart, strTrainChartFileName,
    BitmapFormat.JPG, 100);
    }
    catch (Exception bt)
    {
    bt.printStackTrace();
    }
    System.out.println ("The Chart has been saved");
    System.out.println("End of testing for mini-batches training");
    } // End of the method
} // End of the  Encog class
```

Daftar 19-9 menunjukkan fragmen akhir dari hasil pemrosesan pelatihan (menggunakan metode batch makro) setelah eksekusi.

Daftar 19-9. Fragmen Pengakhiran Hasil Pemrosesan Pelatihan (Menggunakan Metode Macro-Batch)

DayNumber = 947 targetFunctionValue = 12.02166
predictFunctionValue = 12.02166 valurDiff = 5.44438E-6
DayNumber = 948 targetFunctionValue = 12.00232
predictFunctionValue = 12.00232 valurDiff = 3.83830E-6
DayNumber = 949 targetFunctionValue = 11.98281
predictFunctionValue = 11.98281 valurDiff = 2.08931E-6
DayNumber = 950 targetFunctionValue = 11.96312
predictFunctionValue = 11.96312 valurDiff = 6.72376E-6
DayNumber = 951 targetFunctionValue = 11.94325
predictFunctionValue = 11.94325 valurDiff = 4.16461E-7
DayNumber = 952 targetFunctionValue = 11.92320
predictFunctionValue = 11.92320 valurDiff = 1.27943E-6
DayNumber = 953 targetFunctionValue = 11.90298
predictFunctionValue = 11.90298 valurDiff = 8.38334E-6
DayNumber = 954 targetFunctionValue = 11.88258
predictFunctionValue = 11.88258 valurDiff = 5.87549E-6
DayNumber = 955 targetFunctionValue = 11.86200
predictFunctionValue = 11.86200 valurDiff = 4.55675E-6
DayNumber = 956 targetFunctionValue = 11.84124
predictFunctionValue = 11.84124 valurDiff = 6.53477E-6
DayNumber = 957 targetFunctionValue = 11.82031
predictFunctionValue = 11.82031 valurDiff = 2.55647E-6
DayNumber = 958 targetFunctionValue = 11.79920
predictFunctionValue = 11.79920 valurDiff = 8.20278E-6
DayNumber = 959 targetFunctionValue = 11.77792
predictFunctionValue = 11.77792 valurDiff = 4.94157E-7
DayNumber = 960 targetFunctionValue = 11.75647
predictFunctionValue = 11.75647 valurDiff = 1.48410E-6
DayNumber = 961 targetFunctionValue = 11.73483
predictFunctionValue = 11.73484 valurDiff = 3.67970E-6
DayNumber = 962 targetFunctionValue = 11.71303
predictFunctionValue = 11.71303 valurDiff = 6.83684E-6
DayNumber = 963 targetFunctionValue = 11.69105
predictFunctionValue = 11.69105 valurDiff = 4.30269E-6
DayNumber = 964 targetFunctionValue = 11.66890
predictFunctionValue = 11.66890 valurDiff = 1.69128E-6
DayNumber = 965 targetFunctionValue = 11.64658
predictFunctionValue = 11.64658 valurDiff = 7.90340E-6
DayNumber = 966 targetFunctionValue = 11.62409
predictFunctionValue = 11.62409 valurDiff = 8.19566E-6
DayNumber = 967 targetFunctionValue = 11.60142
predictFunctionValue = 11.60143 valurDiff = 4.52810E-6
DayNumber = 968 targetFunctionValue = 11.57859
predictFunctionValue = 11.57859 valurDiff = 6.21339E-6
DayNumber = 969 targetFunctionValue = 11.55559
predictFunctionValue = 11.55559 valurDiff = 7.36500E-6
DayNumber = 970 targetFunctionValue = 11.53241
predictFunctionValue = 11.53241 valurDiff = 3.67611E-6
DayNumber = 971 targetFunctionValue = 11.50907
predictFunctionValue = 11.50907 valurDiff = 2.04084E-6
DayNumber = 972 targetFunctionValue = 11.48556
predictFunctionValue = 11.48556 valurDiff = 3.10021E-6

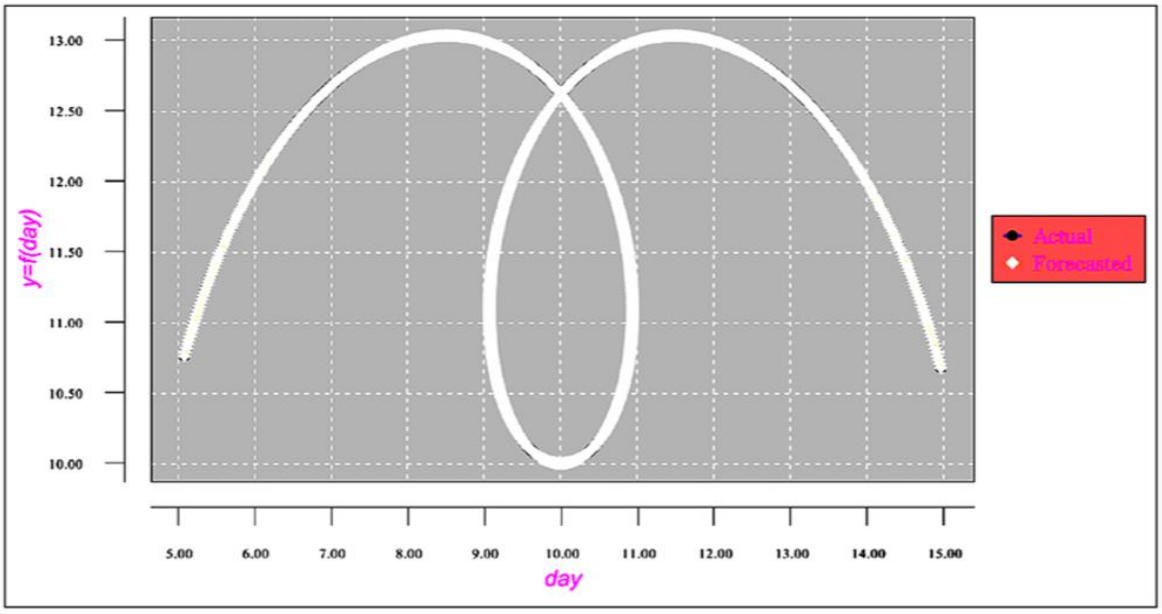
DayNumber = 973 targetFunctionValue = 11.46188
predictFunctionValue = 11.46188 valurDiff = 1.04282E-6
DayNumber = 974 targetFunctionValue = 11.43804
predictFunctionValue = 11.43804 valurDiff = 6.05919E-7
DayNumber = 975 targetFunctionValue = 11.41403
predictFunctionValue = 11.41403 valurDiff = 7.53612E-6
DayNumber = 976 targetFunctionValue = 11.38986
predictFunctionValue = 11.38986 valurDiff = 5.25148E-6
DayNumber = 977 targetFunctionValue = 11.36552
predictFunctionValue = 11.36551 valurDiff = 6.09695E-6
DayNumber = 978 targetFunctionValue = 11.34101
predictFunctionValue = 11.34101 valurDiff = 6.10243E-6
DayNumber = 979 targetFunctionValue = 11.31634
predictFunctionValue = 11.31634 valurDiff = 1.14757E-6
DayNumber = 980 targetFunctionValue = 11.29151
predictFunctionValue = 11.29151 valurDiff = 6.88624E-6
DayNumber = 981 targetFunctionValue = 11.26652
predictFunctionValue = 11.26652 valurDiff = 1.22488E-6
DayNumber = 982 targetFunctionValue = 11.24137
predictFunctionValue = 11.24137 valurDiff = 7.90076E-6
DayNumber = 983 targetFunctionValue = 11.21605
predictFunctionValue = 11.21605 valurDiff = 6.28815E-6
DayNumber = 984 targetFunctionValue = 11.19058
predictFunctionValue = 11.19058 valurDiff = 6.75453E-7
DayNumber = 985 targetFunctionValue = 11.16495
predictFunctionValue = 11.16495 valurDiff = 7.05756E-6
DayNumber = 986 targetFunctionValue = 11.13915
predictFunctionValue = 11.13915 valurDiff = 4.99135E-6
DayNumber = 987 targetFunctionValue = 11.11321
predictFunctionValue = 11.11321 valurDiff = 8.69072E-6
DayNumber = 988 targetFunctionValue = 11.08710
predictFunctionValue = 11.08710 valurDiff = 7.41462E-6
DayNumber = 989 targetFunctionValue = 11.06084
predictFunctionValue = 11.06084 valurDiff = 1.54419E-6
DayNumber = 990 targetFunctionValue = 11.03442
predictFunctionValue = 11.03442 valurDiff = 4.10382E-6
DayNumber = 991 targetFunctionValue = 11.00785
predictFunctionValue = 11.00785 valurDiff = 1.71356E-6
DayNumber = 992 targetFunctionValue = 10.98112
predictFunctionValue = 10.98112 valurDiff = 5.21117E-6
DayNumber = 993 targetFunctionValue = 10.95424
predictFunctionValue = 10.95424 valurDiff = 4.91220E-7
DayNumber = 994 targetFunctionValue = 10.92721
predictFunctionValue = 10.92721 valurDiff = 7.11803E-7
DayNumber = 995 targetFunctionValue = 10.90003
predictFunctionValue = 10.90003 valurDiff = 8.30447E-6
DayNumber = 996 targetFunctionValue = 10.87270
predictFunctionValue = 10.87270 valurDiff = 6.86302E-6
DayNumber = 997 targetFunctionValue = 10.84522
predictFunctionValue = 10.84522 valurDiff = 6.56004E-6
DayNumber = 998 targetFunctionValue = 10.81759
predictFunctionValue = 10.81759 valurDiff = 6.24024E-6

DayNumber = 999 targetFunctionValue = 10.78981
predictFunctionValue = 10.78981 valurDiff = 8.63897E-6
DayNumber = 1000 targetFunctionValue = 10.76181
predictFunctionValue = 10.76188 valurDiff = 7.69201E-6
maxErrorPerc = 1.482606020077711E-6
averErrorPerc = 2.965212040155422E-9

Hasil pengolahan pelatihan (yang menggunakan metode micro-batch) adalah sebagai berikut:

- Kesalahan maksimum kurang dari 0,00000148 persen.
- Rata-rata kesalahan kurang dari 0,00000000269 persen.

Gambar 19.9 menunjukkan grafik hasil aproksimasi pelatihan (menggunakan metode micro-batch). Kedua bagan praktis tumpang tindih (nilai aktual berwarna hitam, dan nilai prediksi berwarna putih).



Gambar 19.9 Bagan hasil aproksimasi pelatihan (menggunakan metode micro-batch)

Seperti halnya set data pelatihan yang dinormalisasi, set data pengujian yang dinormalisasi dipecah menjadi satu set file mikro-batch yang sekarang menjadi input untuk proses pengujian. Daftar 19-10 menunjukkan fragmen akhir dari hasil pengujian setelah eksekusi.

Daftar 19-10. Fragmen Akhir dari Hasil Pemrosesan Pengujian

DayNumber = 6.00372 TargettValue = 11.99207
PredictedValue = 12.00232 DiffPerc = 3.84430E-6
DayNumber = 5.98287 TargettValue = 11.97248
PredictedValue = 11.98281 DiffPerc = 2.09221E-6
DayNumber = 5.96212 TargettValue = 11.95270
PredictedValue = 11.96312 DiffPerc = 6.72750E-6
DayNumber = 5.94146 TargettValue = 11.93275
PredictedValue = 11.94325 DiffPerc = 4.20992E-7
DayNumber = 5.92089 TargettValue = 11.91262
PredictedValue = 11.92320 DiffPerc = 1.27514E-6
DayNumber = 5.90042 TargettValue = 11.89231
PredictedValue = 11.90298 DiffPerc = 8.38833E-6

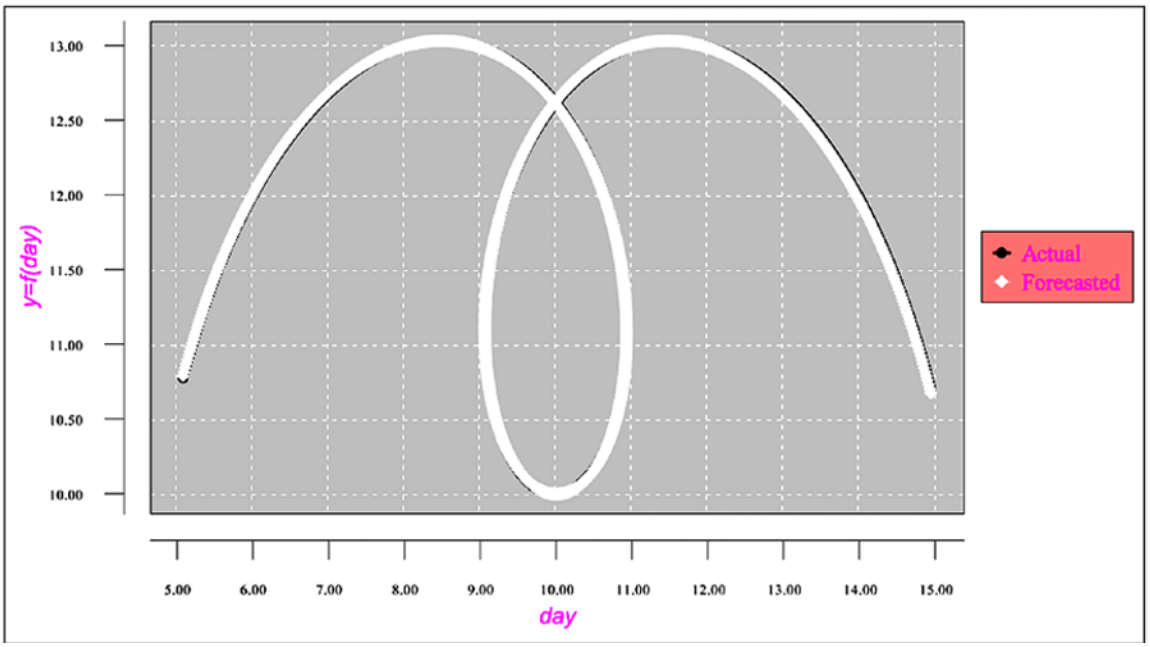
DayNumber = 5.88004 TargettValue = 11.87183
PredictedValue = 11.88258 DiffPerc = 5.88660E-6
DayNumber = 5.85977 TargettValue = 11.85116
PredictedValue = 11.86200 DiffPerc = 4.55256E-6
DayNumber = 5.83959 TargettValue = 11.83033
PredictedValue = 11.84124 DiffPerc = 6.53740E-6
DayNumber = 5.81952 TargettValue = 11.80932
PredictedValue = 11.82031 DiffPerc = 2.55227E-6
DayNumber = 5.79955 TargettValue = 11.78813
PredictedValue = 11.79920 DiffPerc = 8.20570E-6
DayNumber = 5.77968 TargettValue = 11.76676
PredictedValue = 11.77792 DiffPerc = 4.91208E-7
DayNumber = 5.75992 TargettValue = 11.74523
PredictedValue = 11.75647 DiffPerc = 1.48133E-6
DayNumber = 5.74026 TargettValue = 11.72352
PredictedValue = 11.73484 DiffPerc = 3.68852E-6
DayNumber = 5.72071 TargettValue = 11.70163
PredictedValue = 11.71303 DiffPerc = 6.82806E-6
DayNumber = 5.70128 TargettValue = 11.67958
PredictedValue = 11.69105 DiffPerc = 4.31230E-6
DayNumber = 5.68195 TargettValue = 11.65735
PredictedValue = 11.66890 DiffPerc = 1.70449E-6
DayNumber = 5.66274 TargettValue = 11.63495
PredictedValue = 11.64658 DiffPerc = 7.91193E-6
DayNumber = 5.64364 TargettValue = 11.61238
PredictedValue = 11.62409 DiffPerc = 8.20057E-6
DayNumber = 5.62465 TargettValue = 11.58964
PredictedValue = 11.60143 DiffPerc = 4.52651E-6
DayNumber = 5.60578 TargettValue = 11.56673
PredictedValue = 11.57859 DiffPerc = 6.20537E-6
DayNumber = 5.58703 TargettValue = 11.54365
PredictedValue = 11.55559 DiffPerc = 7.37190E-6
DayNumber = 5.56840 TargettValue = 11.52040
PredictedValue = 11.53241 DiffPerc = 3.68228E-6
DayNumber = 5.54989 TargettValue = 11.49698
PredictedValue = 11.50907 DiffPerc = 2.05114E-6
DayNumber = 5.53150 TargettValue = 11.47340
PredictedValue = 11.48556 DiffPerc = 3.10919E-6
DayNumber = 5.51323 TargettValue = 11.44965
PredictedValue = 11.46188 DiffPerc = 1.03517E-6
DayNumber = 5.49509 TargettValue = 11.42573
PredictedValue = 11.43804 DiffPerc = 6.10184E-7
DayNumber = 5.47707 TargettValue = 11.40165
PredictedValue = 11.41403 DiffPerc = 7.53367E-6
DayNumber = 5.45918 TargettValue = 11.37740
PredictedValue = 11.38986 DiffPerc = 5.25199E-6
DayNumber = 5.44142 TargettValue = 11.35299
PredictedValue = 11.36551 DiffPerc = 6.09026E-6
DayNumber = 5.42379 TargettValue = 11.32841
PredictedValue = 11.34101 DiffPerc = 6.09049E-6
DayNumber = 5.40629 TargettValue = 11.30368
PredictedValue = 11.31634 DiffPerc = 1.13713E-6

DayNumber = 5.38893 TargettValue = 11.27878
PredictedValue = 11.29151 DiffPerc = 6.88165E-6
DayNumber = 5.37169 TargettValue = 11.25371
PredictedValue = 11.26652 DiffPerc = 1.22300E-6
DayNumber = 5.35460 TargettValue = 11.22849
PredictedValue = 11.24137 DiffPerc = 7.89661E-6
DayNumber = 5.33763 TargettValue = 11.20311
PredictedValue = 11.21605 DiffPerc = 6.30025E-6
DayNumber = 5.32081 TargettValue = 11.17756
PredictedValue = 11.19058 DiffPerc = 6.76200E-7
DayNumber = 5.30412 TargettValue = 11.15186
PredictedValue = 11.16495 DiffPerc = 7.04606E-6
DayNumber = 5.28758 TargettValue = 11.12601
PredictedValue = 11.13915 DiffPerc = 4.98925E-6
DayNumber = 5.27118 TargettValue = 11.09999
PredictedValue = 11.11321 DiffPerc = 8.69060E-6
DayNumber = 5.25492 TargettValue = 11.07382
PredictedValue = 11.08710 DiffPerc = 7.41171E-6
DayNumber = 5.23880 TargettValue = 11.04749
PredictedValue = 11.06084 DiffPerc = 1.54138E-6
DayNumber = 5.22283 TargettValue = 11.02101
PredictedValue = 11.03442 DiffPerc = 4.09728E-6
DayNumber = 5.20701 TargettValue = 10.99437
PredictedValue = 11.00785 DiffPerc = 1.71899E-6
DayNumber = 5.19133 TargettValue = 10.96758
PredictedValue = 10.98112 DiffPerc = 5.21087E-6
DayNumber = 5.17581 TargettValue = 10.94064
PredictedValue = 10.95424 DiffPerc = 4.97273E-7
DayNumber = 5.16043 TargettValue = 10.91355
PredictedValue = 10.92721 DiffPerc = 7.21563E-7
DayNumber = 5.14521 TargettValue = 10.88630
PredictedValue = 10.90003 DiffPerc = 8.29551E-6
DayNumber = 5.13013 TargettValue = 10.85891
PredictedValue = 10.87270 DiffPerc = 6.86988E-6
DayNumber = 5.11522 TargettValue = 10.83136
PredictedValue = 10.84522 DiffPerc = 6.55538E-6
DayNumber = 5.10046 TargettValue = 10.80367
PredictedValue = 10.81759 DiffPerc = 6.24113E-6
DayNumber = 5.08585 TargettValue = 10.77584
PredictedValue = 10.78981 DiffPerc = 8.64007E-6
maxErrorPerc = 9.002677165459051E-6
averErrorPerc = 4.567068981414947E-6

Hasil pengolahan pengujian (menggunakan metode micro-batch) adalah sebagai berikut:

- Kesalahan maksimum kurang dari 0,00000900 persen.
- Rata-rata kesalahan kurang dari 0,00000457 persen.

Gambar 19.10 menunjukkan grafik hasil pemrosesan pengujian (menggunakan metode micro-batch). Sekali lagi, kedua grafik praktis tumpang tindih (nilai sebenarnya adalah hitam, dan nilai prediksi berwarna putih).



Gambar 19.10 Bagan hasil pemrosesan pengujian (menggunakan metode micro-batch)

19.8 RINGKASAN

Jaringan saraf memiliki masalah yang mendekati fungsi kontinu dengan topologi yang kompleks. Sulit untuk mendapatkan perkiraan berkualitas baik untuk fungsi tersebut. Bab ini menunjukkan bahwa metode micro-batch mampu mendekati fungsi tersebut dengan hasil presisi tinggi. Sampai sekarang Anda menggunakan jaringan saraf untuk menyelesaikan tugas regresi. Pada bab berikutnya, Anda akan mempelajari cara menggunakan jaringan saraf untuk klasifikasi objek.

BAB 20
JARINGAN SYARAF TIRUAN UNTUK MENGKLASIFIKASIKAN OBJEK

Dalam bab ini, Anda akan menggunakan jaringan saraf untuk mengklasifikasikan objek. Klasifikasi berarti mengenali berbagai objek dan menentukan kelas tempat objek tersebut berada. Seperti banyak bidang kecerdasan buatan, klasifikasi mudah dilakukan oleh manusia tetapi bisa sangat sulit bagi komputer.

20.1 CONTOH 6: KLASIFIKASI CATATAN

Untuk contoh ini, Anda disajikan dengan lima buku, dan setiap buku termasuk dalam bidang pengetahuan manusia yang berbeda: medis, pemrograman, teknik, listrik, atau musik. Anda juga diberikan tiga kata yang paling sering digunakan di setiap buku. Lihat Daftar 20-1.

Banyak catatan disediakan untuk contoh ini, dan setiap catatan mencakup tiga kata. Jika ketiga kata dalam sebuah record milik buku tertentu, maka program harus menentukan bahwa record tersebut milik buku itu. Jika sebuah catatan memiliki campuran kata-kata yang bukan milik salah satu dari lima buku, maka program harus mengklasifikasikan catatan itu sebagai milik buku yang tidak dikenal.

Contoh ini terlihat sederhana; pada kenyataannya, mungkin terlihat seperti tidak memerlukan jaringan saraf dan masalah tersebut dapat diselesaikan dengan menggunakan logika pemrograman biasa. Namun, ketika volume buku dan catatan menjadi jauh lebih besar, dengan sejumlah besar kombinasi kata yang tidak dapat diprediksi termasuk dalam setiap catatan dan dengan syarat bahwa hanya sejumlah kecil kata dari satu buku yang cukup untuk sebuah catatan milik sebuah buku tertentu, maka kecerdasan buatan diperlukan untuk menangani tugas tersebut.

Daftar 20-1. Daftar Lima Buku dengan Tiga Kata Paling Sering

Buku 1. Medis. operasi, darah, resep,
Buku 2. Pemrograman. file, java, debugging
Buku 3. Rekayasa. pembakaran, sekrup, mesin
Buku 4. Listrik. volt, solenoid, dioda
Buku 5. Musik. adagio, himne, opera,

Kata-kata tambahan. Kami akan menggunakan kata-kata dalam daftar ini untuk memasukkannya ke dalam kumpulan data pengujian. pelanggan, angin, rumput, kertas, kalkulator, bunga, printer, meja, foto, peta, pena, lantai.

Kata-kata tambahan. Kami akan menggunakan kata-kata dalam daftar ini untuk memasukkannya ke dalam kumpulan data pengujian. pelanggan, angin, rumput, kertas, kalkulator, bunga, printer, meja, foto, peta, pena, lantai.Extra words. We will use words in this list to include them in the test dataset. customer, wind, grass, paper, calculator, flower, printer ,desk, photo, map, pen, floor.

Untuk menyederhanakan pemrosesan, Anda menetapkan angka untuk semua kata dan menggunakan angka tersebut sebagai ganti kata saat membangun kumpulan data pelatihan dan pengujian. Tabel 20.1 menunjukkan referensi silang kata-ke-angka.

Tabel 20.1 Referensi Silang Kata-ke-Angka (word-to-number)

Word (Kata)	Assigned Number
surgery (operasi)	1
blood (darah)	2
prescription (resep)	3
file (file)	4
java (java)	5
debugging (debugging)	6
combustion (pembakaran)	7
screw (skrup/baut)	8
machine (mesin)	9
volt (volt)	10
solenoid (solenoida)	11
diode (dioda)	12
adagio (adagio)	13
hymn (himne)	14
opera (opeara)	15
customer (pelanggan)	16
wind (angin)	17
grass (rumput)	18
paper (kertas)	19
calculator (kalkulator)	20
flower (bunga)	21
printer (printer)	22
desk (meja)	23
photo (foto)	24
map (peta)	25
pen (pen)	26
floor (lantai)	27

20.2 KUMPULAN DATA SET

Setiap record dalam set data pelatihan terdiri dari tiga bidang yang menampung kata-kata dari daftar kata yang paling sering digunakan dalam buku. Juga termasuk dalam setiap catatan adalah lima bidang target, yang menunjukkan buku tempat catatan itu berada. Perhatikan bahwa ini adalah contoh pertama dalam buku di mana jaringan memiliki lima bidang target. Informasi ini digunakan untuk melatih jaringan. Misalnya, kombinasi 1, 0, 0, 0, 0 berarti buku #1; kombinasi 0, 1, 0, 0, 0 berarti buku #2; dan seterusnya. Juga, untuk setiap buku, Anda perlu membuat enam catatan dalam kumpulan data pelatihan, bukan satu. Keenam catatan ini mencakup semua kemungkinan permutasi kata dalam catatan. Tabel 20.2 menunjukkan semua kemungkinan permutasi kata di semua record. Saya menggunakan huruf miring untuk menyorot bagian dari setiap catatan yang menyimpan angka kata.

Tabel 20.2 Permutasi Kata di Semua Catatan

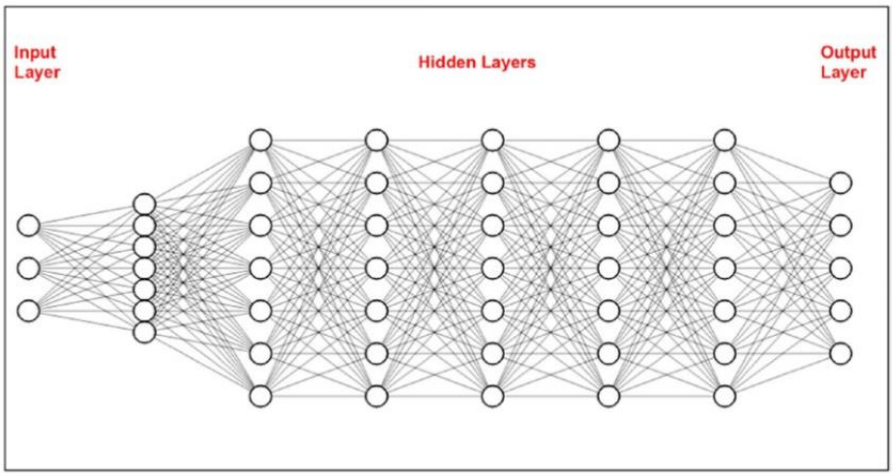
Records for Book 1							
1	*2*	*3*	1	0	0	0	0
1	*3*	*2*	1	0	0	0	0
2	*1*	*3*	1	0	0	0	0
2	*3*	*1*	1	0	0	0	0
3	*1*	*2*	1	0	0	0	0
3	*2*	*1*	1	0	0	0	0
Records for Book 2							
4	*5*	*6*	0	1	0	0	0
4	*6*	*5*	0	1	0	0	0
5	*4*	*6*	0	1	0	0	0
5	*6*	*4*	0	1	0	0	0
6	*4*	*5*	0	1	0	0	0
6	*5*	*4*	0	1	0	0	0
Records for Book 3							
7	*8*	*9*	0	0	1	0	0
7	*9*	*8*	0	0	1	0	0
8	*7*	*9*	0	0	1	0	0
8	*9*	*7*	0	0	1	0	0
9	*7*	*8*	0	0	1	0	0
9	*8*	*7*	0	0	1	0	0
Records for Book 4							
10	*11*	*12*	0	0	0	1	0
10	*12*	*11*	0	0	0	1	0
11	*10*	*12*	0	0	0	1	0
11	*12*	*10*	0	0	0	1	0
12	*10*	*11*	0	0	0	1	0
12	*11*	*10*	0	0	0	1	0
Records for Book 5							
13	*14*	*15*	0	0	0	0	1
13	*15*	*14*	0	0	0	0	1
14	*13*	*15*	0	0	0	0	1
14	*15*	*13*	0	0	0	0	1
15	*13*	*14*	0	0	0	0	1
15	*14*	*13*	0	0	0	0	1

Menyatukan semuanya, Tabel 20.3 menunjukkan kumpulan data pelatihan.

Tabel 20.3 Kumpulan Data Pelatihan

Word1	Word2	Word3	Target1	Target2	Target3	Target4	Target5
1	2	3	1	0	0	0	0
1	3	2	1	0	0	0	0
2	1	3	1	0	0	0	0
2	3	1	1	0	0	0	0
3	1	2	1	0	0	0	0
3	2	1	1	0	0	0	0
4	5	6	0	1	0	0	0
4	6	5	0	1	0	0	0
5	4	6	0	1	0	0	0
5	6	4	0	1	0	0	0
6	4	5	0	1	0	0	0
6	5	4	0	1	0	0	0
7	8	9	0	0	1	0	0
7	9	8	0	0	1	0	0
8	7	9	0	0	1	0	0
8	9	7	0	0	1	0	0
9	7	8	0	0	1	0	0
9	8	7	0	0	1	0	0
10	11	12	0	0	0	1	0
10	12	11	0	0	0	1	0
11	10	12	0	0	0	1	0
11	12	10	0	0	0	1	0
12	10	11	0	0	0	1	0
12	11	10	0	0	0	1	0
13	14	15	0	0	0	0	1
13	15	14	0	0	0	0	1
14	13	15	0	0	0	0	1
14	15	13	0	0	0	0	1
15	13	14	0	0	0	0	1
15	14	13	0	0	0	0	1

20.3 ARISTEKTUR JARINGAN



Gambar 20.1 Arsitektur jaringan

Jaringan memiliki lapisan input dengan tiga neuron input, enam lapisan tersembunyi dengan masing-masing tujuh neuron, dan lapisan output dengan lima neuron. Gambar 20.1 menunjukkan arsitektur jaringan.

20.4 MENGUJI KUMPULAN DATA

Kumpulan data pengujian terdiri dari catatan dengan kata/angka yang dimasukkan secara acak. Catatan-catatan ini bukan milik satu buku, meskipun beberapa di antaranya menyertakan satu atau dua kata dari daftar yang paling sering digunakan. Tabel 20.4 menunjukkan kumpulan data pengujian.

Tabel 20.4 Menguji Kumpulan Data

Word1	Word2	Word3	Target 1	Target 2	Target 3	Target 4	Target 5
1	2	16	0	0	0	0	0
4	17	5	0	0	0	0	0
8	9	18	0	0	0	0	0
19	10	11	0	0	0	0	0
15	20	13	0	0	0	0	0
27	1	26	0	0	0	0	0
14	23	22	0	0	0	0	0
21	20	18	0	0	0	0	0
25	23	24	0	0	0	0	0
11	9	6	0	0	0	0	0
3	5	8	0	0	0	0	0
6	10	15	0	0	0	0	0
16	17	18	0	0	0	0	0
19	1	8	0	0	0	0	0
27	23	17	0	0	0	0	0

Tidak perlu menyertakan kolom target dalam file pengujian; namun, mereka disertakan untuk kenyamanan (untuk membandingkan hasil yang diprediksi dan aktual). Kolom ini tidak digunakan untuk pemrosesan. Seperti biasa, Anda perlu menormalkan set data pelatihan dan pengujian pada interval [-1, 1]. Karena contoh ini menampilkan banyak neuron di lapisan input dan output, Anda memerlukan kode sumber normalisasi.

20.5 KODE PROGRAM UNTUK NORMALISASI DATA

Daftar 20-2 menunjukkan kode program yang menormalkan set data pelatihan dan pengujian.

Daftar 20-2. Kode Program untuk Normalisasi Data

```
// ============================================================
// Program ini menormalkan semua kolom dari dataset CSV input
// yang menempatkan hasil
// dalam file CSV output.
//
// Kolom pertama dari dataset input menyertakan nilai titik X
// dan kolom kedua dari input dataset menyertakan nilai
// fungsi di
// titik X.
// ============================================================
package sample5_norm;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.*;
public class Sample5_Norm
{
        // Interval to normalize
        static double Nh =  1;
        static double Nl = -1;
        // First column
        static double minXPointDl = 1.00;
        static double maxXPointDh = 1000.00;
        // Second column - target data
        static double minTargetValueDl = 60.00;
        static double maxTargetValueDh = 1600.00;
        public static double normalize(double value, double Dh, double Dl)
        {
        double normalizedValue = (value - Dl)*(Nh - Nl)/(Dh - Dl) + Nl;
        return normalizedValue;
        }
        public static void main(String[] args)
        {
        // Normalize train file      String inputFileName =
```

```
"C:/Book_Examples/Sample5_Train_Real.csv";
String outputNormFileName =
"C:/Book_Examples/Sample5_Train_Norm.csv";
// Normalize test file
// String inputFileName =
"C:/Book_Examples/Sample5_Test_Real.csv";
// String outputNormFileName = "C:/Book_Examples/Sample5_Test_Norm.csv";
BufferedReader br = null;
PrintWriter out = null;
String line = "";
String cvsSplitBy = ",";
double inputXPointValue;
double targetXPointValue;
double normInputXPointValue;
double normTargetXPointValue;
String strNormInputXPointValue;
String strNormTargetXPointValue;
String fullLine;
int i = -1;
try
{
Files.deleteIfExists(Paths.get(outputNormFileName));
br = new BufferedReader(new FileReader(inputFileName));
out = new PrintWriter(new BufferedWriter(new
FileWriter(outputNormFileName)));
while ((line = br.readLine()) != null)
{
        i++;
        if(i == 0)
        {
        // Write the label line        out.println(line);
        }
        else
        {
        // Break the line using comma as separator        String[] workFields =
        line.split(cvsSplitBy);
        inputXPointValue = Double.parseDouble(workFields[0]);
        targetXPointValue = Double.parseDouble( workFields[1]);
        // Normalize these fields normInputXPointValue =
        normalize(inputXPointValue, maxXPointDh, minXPointDl);
        normTargetXPointValue = normalize(targetXPointValue,
        maxTargetValueDh, minTargetValueDl);
        // Convert normalized fields to string, so they can be inserted
        //into the output CSV file strNormInputXPointValue =
        Double.toString(normInput XPointValue);
        strNormTargetXPointValue = Double.toString(normTarget XPointValue);
        // Concatenate these fields into a string line with
        //coma separator        fullLine =
```

```
                strNormInputXPointValue + "," +
                strNormTargetXPointValue;
                // Put fullLine into the output file          out.println(fullLine);
                } // End of IF Else
                        }
                        // end of while
                        }  // end of TRY
                        catch (FileNotFoundException e)
                        {
                        e.printStackTrace();
                        System.exit(1);
                        }
                        catch (IOException io)
                        {
                        io.printStackTrace();
                        }
                        finally
                        {
                        if (br != null)
                        {
                        try
                        {
                        br.close();
                        out.close();
                        }
                        catch (IOException e)
                        {
                        e.printStackTrace();
                        }
                }
        }
    }
}
```

Tabel 20.5 menunjukkan kumpulan data pelatihan yang dinormalisasi.

Tabel 20.5 Kumpulan Data Pelatihan yang Dinormalisasi

Word 1	Word 2	Word 3	Target 1	Target 2	Target 3	Target 4	Target 5
-1	-0.966101695	-0.93220339	1	-1	-1	-1	-1
-1	-0.93220339	-0.966101695	1	-1	-1	-1	-1
-0.966101695	-1	-0.93220339	1	-1	-1	-1	-1
-0.966101695	-0.93220339	-1	1	-1	-1	-1	-1
-0.93220339	-1	-0.966101695	1	-1	-1	-1	-1
-0.93220339	-0.966101695	-1	1	-1	-1	-1	-1
-0.898305085	-0.86440678	-0.830508475	-1	1	-1	-1	-1
-0.898305085	-0.830508475	-0.86440678	-1	1	-1	-1	-1
-0.86440678	-0.898305085	-0.830508475	-1	1	-1	-1	-1
-0.86440678	-0.830508475	-0.898305085	-1	1	-1	-1	-1
-0.830508475	-0.898305085	-0.86440678	-1	1	-1	-1	-1
-0.830508475	-0.86440678	-0.898305085	-1	1	-1	-1	-1
-0.796610169	-0.762711864	-0.728813559	-1	-1	1	-1	-1
-0.796610169	-0.728813559	-0.762711864	-1	-1	1	-1	-1
-0.762711864	-0.796610169	-0.728813559	-1	-1	1	-1	-1
-0.762711864	-0.728813559	-0.796610169	-1	-1	1	-1	-1
-0.728813559	-0.796610169	-0.762711864	-1	-1	1	-1	-1
-0.728813559	-0.762711864	-0.796610169	-1	-1	1	-1	-1
-0.694915254	-0.661016949	-0.627118644	-1	-1	-1	1	-1
-0.694915254	-0.627118644	-0.661016949	-1	-1	-1	1	-1
-0.661016949	-0.694915254	-0.627118644	-1	-1	-1	1	-1
-0.661016949	-0.627118644	-0.694915254	-1	-1	-1	1	-1
-0.627118644	-0.694915254	-0.661016949	-1	-1	-1	1	-1
-0.627118644	-0.661016949	-0.694915254	-1	-1	-1	1	-1
-0.593220339	-0.559322034	-0.525423729	-1	-1	-1	-1	1
-0.593220339	-0.525423729	-0.559322034	-1	-1	-1	-1	1
-0.559322034	-0.593220339	-0.525423729	-1	-1	-1	-1	1
-0.559322034	-0.525423729	-0.593220339	-1	-1	-1	-1	1
-0.525423729	-0.593220339	-0.559322034	-1	-1	-1	-1	1
-0.525423729	-0.559322034	-0.593220339	-1	-1	-1	-1	1

Tabel 20.6 menunjukkan kumpulan data pengujian yang dinormalisasi.

Tabel 20.6 Kumpulan Data Pengujian yang Dinormalisasi

Word 1	Word 2	Word 3	Target 1	Target 2	Target 3	Target 4	Target 5
-1	-0.966101695	-0.491525424	-1	-1	-1	-1	-1
-0.898305085	-0.457627119	-0.86440678	-1	-1	-1	-1	-1
-0.762711864	-0.728813559	-0.423728814	-1	-1	-1	-1	-1
-0.389830508	-0.694915254	-0.661016949	-1	-1	-1	-1	-1
-0.525423729	-0.355932203	-0.593220339	-1	-1	-1	-1	-1
-0.118644068	-1	0.152542373	-1	-1	-1	-1	-1
-0.559322034	-0.254237288	-0.288135593	-1	-1	-1	-1	-1
-0.322033898	-0.355932203	-0.423728814	-1	-1	-1	-1	-1
-0.186440678	-0.254237288	-0.220338983	-1	-1	-1	-1	-1
-0.661016949	-0.728813559	-0.830508475	-1	-1	-1	-1	-1
-0.93220339	-0.86440678	-0.762711864	-1	-1	-1	-1	-1
-0.830508475	-0.69491525	-0.525423729	-1	-1	-1	-1	-1
-0.491525424	-0.45762711	-0.423728814	-1	-1	-1	-1	-1
-0.389830508	-1	-0.762711864	-1	-1	-1	-1	-1
-0.118644068	-0.25423728	-0.457627119	-1	-1	-1	-1	-1

20.6 KODE PROGRAM UNTUK KLASIFIKASI

Listing 20-3 menunjukkan kode program klasifikasi.

Daftar 20-3. Kode Program Klasifikasi

```
// ============================================================
// Contoh penggunaan neural network untuk klasifikasi objek.
// File pelatihan/pengujian yang dinormalisasi terdiri dari
// record dengan format
// berikut: 3 field input (nomor kata) dan 5 field
// target (menunjukkan buku
// milik record).
// ============================================================
package sample6;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
```

```
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month; import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar; import java.util.Date;
import java.util.List; import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample6 implements ExampleChart<XYChart>
{
        // Interval to normalize data
        static double Nh;
        static double Nl;
        // Normalization parameters for workBook number
```

```
static double minWordNumberDl;
static double maxWordNumberDh;
// Normalization parameters for target values
static double minTargetValueDl;
static double maxTargetValueDh;
static double doublePointNumber = 0.00;
static int intPointNumber = 0;
static InputStream input = null;
static double[] arrPrices = new double[2500];
static double normInputWordNumber_01 = 0.00;
static double normInputWordNumber_02 = 0.00;
static double normInputWordNumber_03 = 0.00;
static double denormInputWordNumber_01 = 0.00;
static double denormInputWordNumber_02 = 0.00;
static double denormInputWordNumber_03 = 0.00;
static double normTargetBookNumber_01 = 0.00;
static double normTargetBookNumber_02 = 0.00;
static double normTargetBookNumber_03 = 0.00;
static double normTargetBookNumber_04 = 0.00;
static double normTargetBookNumber_05 = 0.00;
static double normPredictBookNumber_01 = 0.00;
static double normPredictBookNumber_02 = 0.00;
static double normPredictBookNumber_03 = 0.00;
static double normPredictBookNumber_04 = 0.00;
static double normPredictBookNumber_05 = 0.00;
static double denormTargetBookNumber_01 = 0.00;
static double denormTargetBookNumber_02 = 0.00;
static double denormTargetBookNumber_03 = 0.00;
static double denormTargetBookNumber_04 = 0.00;
static double denormTargetBookNumber_05 = 0.00;
static double denormPredictBookNumber_01 = 0.00;
static double denormPredictBookNumber_02 = 0.00;
static double denormPredictBookNumber_03 = 0.00;
static double denormPredictBookNumber_04 = 0.00;
static double denormPredictBookNumber_05 = 0.00;
static double normDifferencePerc = 0.00;
static double denormPredictXPointValue_01 = 0.00;
static double denormPredictXPointValue_02 = 0.00;
static double denormPredictXPointValue_03 = 0.00;
static double denormPredictXPointValue_04 = 0.00;
static double denormPredictXPointValue_05 = 0.00;
static double valueDifference = 0.00;
static int numberOfInputNeurons;
static int numberOfOutputNeurons;
static int intNumberOfRecordsInTestFile;
static String trainFileName;
static String priceFileName;
static String testFileName;
```

```
    static String chartTrainFileName;
    static String chartTestFileName;
    static String networkFileName;
    static int workingMode;
    static String cvsSplitBy = ",";
    static int returnCode;
    static List<Double> xData = new ArrayList<Double>();
    static List<Double> yData1 = new ArrayList<Double>();
    static List<Double> yData2 = new ArrayList<Double>();
    static XYChart Chart;
@Override  public XYChart getChart()
{
    // Create Chart
    Chart = new XYChartBuilder().width(900).height(500).title(getClass().g
    etSimpleName()).xAxisTitle("x").yAxisTitle("y= f(x)").build();
    // Customize Chart
    Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor
    (ChartColor.GREY));
    Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
    Chart.getStyler().setChartBackgroundColor(Color.WHITE);
    Chart.getStyler().setLegendBackgroundColor(Color.PINK);
    Chart.getStyler().setChartFontColor(Color.MAGENTA);
    Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
    Chart.getStyler().setChartTitleBoxVisible(true);
    Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
    Chart.getStyler().setPlotGridLinesVisible(true);
    Chart.getStyler().setAxisTickPadding(20);
    Chart.getStyler().setAxisTickMarkLength(15);
    Chart.getStyler().setPlotMargin(20);
    Chart.getStyler().setChartTitleVisible(false);
    Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,  Font.BOLD, 24));
    Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
    Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
    Chart.getStyler().setLegendSeriesLineLength(12);
    Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF,  Font.ITALIC, 18));
    Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF,  Font.PLAIN, 11));
    Chart.getStyler().setDatePattern("yyyy-MM");
    Chart.getStyler().setDecimalPattern("#0.00");
    // Interval to normalize data    Nh =  1;
    Nl = -1;
    // Normalization parameters for workBook number
    double minWordNumberDl = 1.00;
    double maxWordNumberDh = 60.00;
    // Normalization parameters for target values
    minTargetValueDl = 0.00;
    maxTargetValueDh = 1.00;
    // Configuration
    // Set the mode to run the program
```

```
        workingMode = 1;
        // Training mode
        if(workingMode == 1)
        {
        // Training mode
        intNumberOfRecordsInTestFile = 31;
        trainFileName = "C:/My_Neural_Network_Book/Book_Examples/Sample6_
        Norm_Train_File.csv";
        File file1 = new File(chartTrainFileName);    File file2 = new File(networkFileName);
        if(file1.exists())
        file1.delete();
        if(file2.exists())
        file2.delete();
        returnCode = 0;
        // Clear the return code variable
        do
        {
        returnCode = trainValidateSaveNetwork();
        }
        while (returnCode > 0);
        }
    // End the training mode
    else
    {
        // Testing mode
        intNumberOfRecordsInTestFile = 16;
        testFileName = "C:/My_Neural_Network_Book/Book_Examples/Sample6_
        Norm_Test_File.csv";
        networkFileName =
        "C:/My_Neural_Network_Book/Book_Examples/Sample6_Saved_Network_
        File.csv";
        numberOfInputNeurons = 3;
        numberOfOutputNeurons = 5;
        loadAndTestNetwork();
    }
    Encog.getInstance().shutdown();
    return Chart;
    }
// End of the method
// =============================================================
// Load CSV to memory.
// @return The loaded dataset.
// =============================================================
public static MLDataSet loadCSV2Memory(String filename, int input,  int ideal, boolean
headers, CSVFormat format, boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format,
        headers, input, ideal, significance);
```

```
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// =============================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// =============================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample6();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// =============================================================
// Metode ini melatih, memvalidasi, dan menyimpan file jaringan
// terlatih pada disk
// =============================================================
static public int trainValidateSaveNetwork()
{
        // Load the training CSV file in memory
        MLDataSet                                trainingSet                                =
        loadCSV2Memory(trainFileName,numberOfInputNeurons,
        numberOfOutputNeurons,true,CSVFormat.ENGLISH,false);
        // create a neural network     BasicNetwork network = new BasicNetwork();
        // Input layer     network.addLayer(new BasicLayer(null,true,3));
        // Hidden layer     network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new            BasicLayer(new            ActivationTANH(),true,7));
        network.addLayer(new            BasicLayer(new            ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        //     Output     layer                    network.addLayer(new     BasicLayer(new
        ActivationTANH(),false,5));
        network.getStructure().finalizeStructure();     network.reset();
        // train the neural network
        final ResilientPropagation train = new ResilientPropagation(network, trainingSet);
        int epoch = 1;
        do
        {
        train.iteration();
        System.out.println("Epoch #" + epoch + " Error:" +  train.getError());
        epoch++;
        if (epoch >= 1000 && network.calculateError(trainingSet) > 0.0000000000000012)
        {
        returnCode = 1;
        System.out.println("Try again");
        return returnCode;
```

```
}
}
while (network.calculateError(trainingSet) > 0.0000000000000011);
// Save the network file EncogDirectoryPersistence.saveObject(new
File(networkFileName), network);
System.out.println("Neural Network Results:");
int m = 0;
for(MLDataPair pair: trainingSet)
{
        m++;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results
        normInputWordNumber_01 = inputData.getData(0);
        normInputWordNumber_02 = inputData.getData(1);
        normInputWordNumber_03 = inputData.getData(2);
        normTargetBookNumber_01 = actualData.getData(0);
        normTargetBookNumber_02 = actualData.getData(1);
        normTargetBookNumber_03 = actualData.getData(2);
        normTargetBookNumber_04 = actualData.getData(3);
        normTargetBookNumber_05 = actualData.getData(4);
        normPredictBookNumber_01 = predictData.getData(0);
        normPredictBookNumber_02 = predictData.getData(1);
        normPredictBookNumber_03 = predictData.getData(2);
        normPredictBookNumber_04 = predictData.getData(3);
        normPredictBookNumber_05 = predictData.getData(4);
        // De-normalize the results denormInputWordNumber_01 =
        ((minWordNumberDl
        maxWordNumberDh)*normInputWordNumber_01 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormInputWordNumber_02 = ((minWordNumberDl
        maxWordNumberDh)*normInputWordNumber_02 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormInputWordNumber_03 = ((minWordNumberDl
        maxWordNumberDh)*normInputWordNumber_03 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormTargetBookNumber_01 = ((minTargetValueDl -
        maxTargetValueDh)*normTargetBookNumber_01
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
        denormTargetBookNumber_02 = ((minTargetValueDl -
        maxTargetValueDh)*normTargetBookNumber_02
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
        denormTargetBookNumber_03 = ((minTargetValueDl -
        maxTargetValueDh)*normTargetBookNumber_03
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
```

```
denormTargetBookNumber_04 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_04
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormTargetBookNumber_05 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_05
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_01 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_01
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_02 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_02
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_03 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_03
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_04 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_04
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_05 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_05
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
System.out.println ("RecordNumber = " + m);
System.out.println ("denormTargetBookNumber_01 = " +
denormTargetBookNumber_01 + "denormPredictBookNumber_01 = " +
denormPredictBookNumber_01);
System.out.println ("denormTargetBookNumber_02 = " +
denormTargetBookNumber_02 + "denormPredictBookNumber_02 = " +
denormPredictBookNumber_02);
System.out.println ("denormTargetBookNumber_03 = " +
denormTargetBookNumber_03 + "denormPredictBookNumber_03 = " +
denormPredictBookNumber_03);
System.out.println ("denormTargetBookNumber_04 = " +
denormTargetBookNumber_04 + "denormPredictBookNumber_04 = " +
denormPredictBookNumber_04);
System.out.println ("denormTargetBookNumber_05 = " +
denormTargetBookNumber_05 + "denormPredictBookNumber_05 = " +
denormPredictBookNumber_05);
//System.out.println (" ");
// Print the classification results
        if(Math.abs(denormPredictBookNumber_01) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) > 0.85  &
        Math.abs(denormPredictBookNumber_02) < 0.2   &
        Math.abs(denormPredictBookNumber_03) < 0.2   &
        Math.abs(denormPredictBookNumber_04) < 0.2   &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 1 belongs to book 1");
System.out.println (" ");
```

```
}
else
{             System.out.println ("Wrong results for record 1");
System.out.println (" ");
}
        if(Math.abs(denormPredictBookNumber_02) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) > 0.85 &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 2 belongs to book 2");
System.out.println (" ");
}
else
{
System.out.println ("Wrong results for record 2");
System.out.println (" ");
}
        if(Math.abs(denormPredictBookNumber_03) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) < 0.2 &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) > 0.85 &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 3 belongs to book 3");
System.out.println (" ");
}
else
{
System.out.println ("Wrong results for record 3");
System.out.println (" ");
}
        if(Math.abs(denormPredictBookNumber_04) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) > 0.85 &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 4 belongs to book 4");
System.out.println (" ");
}
else
{
System.out.println ("Wrong results for record 4");
```

```
                System.out.println (" ");
                }
                        if(Math.abs(denormPredictBookNumber_05) > 0.85)
                        if(Math.abs(denormPredictBookNumber_01) < 0.2  &
                        Math.abs(denormPredictBookNumber_02) < 0.2  &
                        Math.abs(denormPredictBookNumber_03) < 0.2  &
                        Math.abs(denormPredictBookNumber_04) < 0.2 &
                        Math.abs(denormPredictBookNumber_05) > 0.85)
                {
                System.out.println ("Record 5 belongs to book 5");
                System.out.println (" ");
                }
                else
                {
                System.out.println ("Wrong results for record 5");
                System.out.println (" ");
                }
    }  // End for pair loop
    returnCode = 0;
    return returnCode;
}  // End of the method
// =======================================================
// Muat dan uji jaringan yang terlatih pada titik yang
// tidak dapat dilatih
// =======================================================
static public void loadAndTestNetwork()
{
System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double targetToPredictPercent = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double normInputWordNumberFromRecord = 0.00;
        double normTargetBookNumberFromRecord = 0.00;
        double normPredictXPointValueFromRecord = 0.00;
        BasicNetwork network;    maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        // Load the test dataset into memory
        MLDataSet testingSet =    loadCSV2Memory(testFileName,numberOfInputNeurons,
        numberOfOutputNeurons,true,CSVFormat.ENGLISH,false);
        //    Load    the    saved    trained    network                network    =
        (BasicNetwork)EncogDirectoryPersistence.loadObject(new
        File(networkFileName));
        int i = 0;
```

```
for (MLDataPair pair:  testingSet)
{
        i++;
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // These values are Normalized as the whole input is
        normInputWordNumberFromRecord = inputData.getData(0);
        normTargetBookNumberFromRecord = actualData.getData(0);
        normPredictXPointValueFromRecord = predictData.getData(0);
        denormInputWordNumber_01 = ((minWordNumberDl -
        maxWordNumberDh)* normInputWordNumber_01 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormInputWordNumber_02 = ((minWordNumberDl -
        maxWordNumberDh)* normInputWordNumber_02 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormInputWordNumber_03 = ((minWordNumberDl
        maxWordNumberDh)*normInputWordNumber_03 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormTargetBookNumber_01 = ((minTargetValueDl -
        maxTargetValueDh)* normTargetBookNumber_01 -
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
        denormTargetBookNumber_02 = ((minTargetValueDl -
        maxTargetValueDh)*normTargetBookNumber_02
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
        denormTargetBookNumber_03 = ((minTargetValueDl -
        maxTargetValueDh)*normTargetBookNumber_03 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormTargetBookNumber_04 = ((minTargetValueDl - maxTarget
        ValueDh)*normTargetBookNumber_04 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormTargetBookNumber_05 = ((minTargetValueDl - maxTarget
        ValueDh)*normTargetBookNumber_05 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormPredictBookNumber_01 =((minTargetValueDl - maxTarget
        ValueDh)*normPredictBookNumber_01 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormPredictBookNumber_02 =((minTargetValueDl - maxTarget
        ValueDh)*normPredictBookNumber_02 -Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormPredictBookNumber_03 =((minTargetValueDl - maxTarget
        ValueDh)*normPredictBookNumber_03 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
        denormPredictBookNumber_04 =((minTargetValueDl - maxTarget
        ValueDh)*normPredictBookNumber_04 - Nh*minTargetValueDl +
        maxTargetValueDh*Nl)/(Nl - Nh);
```

```
denormPredictBookNumber_05 =((minTargetValueDl - maxTarget
ValueDh)*normPredictBookNumber_05 -Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
System.out.println ("RecordNumber = " + i);
System.out.println ("denormTargetBookNumber_01 = " +
denormTargetBookNumber_01 + "denormPredictBookNumber_01 = " +
denormPredictBookNumber_01);
System.out.println ("denormTargetBookNumber_02 = " +
denormTargetBookNumber_02 + "denormPredictBookNumber_02 = " +
denormPredictBookNumber_02);
System.out.println ("denormTargetBookNumber_03 = " +
denormTargetBookNumber_03 + "denormPredictBookNumber_03 = " +
denormPredictBookNumber_03);
System.out.println ("denormTargetBookNumber_04 = " +
denormTargetBookNumber_04 + "denormPredictBookNumber_04 = " +
denormPredictBookNumber_04);
System.out.println ("denormTargetBookNumber_05 = " +
denormTargetBookNumber_05 + "denormPredictBookNumber_05 = " +
denormPredictBookNumber_05);
//System.out.println (" ");
        if(Math.abs(denormPredictBookNumber_01) > 0.85  &
        Math.abs(denormPredictBookNumber_02) < 0.2   &
        Math.abs(denormPredictBookNumber_03) < 0.2   &
        Math.abs(denormPredictBookNumber_04) < 0.2   &
        Math.abs(denormPredictBookNumber_05) < 0.2
||
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) > 0.85 &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2
|
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) > 0.85 &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2        ||
        Math.abs(denormPredictBookNumber_01) < 0.2 &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) > 0.85 &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2
||
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) > 0.85 &
        Math.abs(denormPredictBookNumber_05) < 0.2
```

```
                ||
                        Math.abs(denormPredictBookNumber_01) < 0.2  &
                        Math.abs(denormPredictBookNumber_02) < 0.2  &
                        Math.abs(denormPredictBookNumber_03) < 0.2  &
                        Math.abs(denormPredictBookNumber_04) < 0.2 &
                        Math.abs(denormPredictBookNumber_05) > 0.85)
                {
                System.out.println ("Record belong to some book");
                System.out.println (" ");
                }
                else
                {
                System.out.println ("Unknown book");
                System.out.println (" ");
                }
        }  // End for pair loop
    } // End of the method
} // End of the class
```

Daftar 20-4 menunjukkan potongan kode dari metode pelatihan.

Daftar 20-4. Fragmen Kode Metode Pelatihan

```
static public int trainValidateSaveNetwork()
{
        // Load the training CSV file in memory
        MLDataSet                                   trainingSet                                   =
        loadCSV2Memory(trainFileName,numberOfInputNeurons,
        numberOfOutputNeurons,true,CSVFormat.ENGLISH,false);
        // create a neural network     BasicNetwork network = new BasicNetwork();
        // Input layer
        network.addLayer(new BasicLayer(null,true,3));
        // Hidden layer
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
        // Output layer
        network.addLayer(new BasicLayer(new ActivationTANH(),false,5));
        network.getStructure().finalizeStructure();
        network.reset();
        //Train the neural network
        final ResilientPropagation train = new ResilientPropagation(network, trainingSet);
        int epoch = 1;
        do
        {
        train.iteration();                  System.out.println("Epoch #" + epoch + " Error:" +
        train.getError());
```

```
epoch++;
if (epoch >= 1000 && network.calculateError(trainingSet) > 0.0000000000000012)
        {
        returnCode = 1;
        System.out.println("Try again");
        return returnCode;
        }
} while (network.calculateError(trainingSet) > 0.0000000000000011);
// Save the network file EncogDirectoryPersistence.saveObject(new
File(networkFileName),network);
System.out.println("Neural Network Results:");
double sumNormDifferencePerc = 0.00;
double averNormDifferencePerc = 0.00;
double maxNormDifferencePerc = 0.00;
int m = 0;
for(MLDataPair pair: trainingSet)
{
        m++;
        final MLData output = network.compute(pair.getInput());
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // Calculate and print the results
        normInputWordNumber_01 = inputData.getData(0);
        normInputWordNumber_02 = inputData.getData(1);
        normInputWordNumber_03 = inputData.getData(2);
        normTargetBookNumber_01 = actualData.getData(0);
        normTargetBookNumber_02 = actualData.getData(1);
        normTargetBookNumber_03 = actualData.getData(2);
        normTargetBookNumber_04 = actualData.getData(3);
        normTargetBookNumber_05 = actualData.getData(4);
        normPredictBookNumber_01 = predictData.getData(0);
        normPredictBookNumber_02 = predictData.getData(1);
        normPredictBookNumber_03 = predictData.getData(2);
        normPredictBookNumber_04 = predictData.getData(3);
        normPredictBookNumber_05 = predictData.getData(4);
        denormInputWordNumber_01 = ((minWordNumberDl -
        maxWordNumberDh)* normInputWordNumber_01 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormInputWordNumber_02 = ((minWordNumberDl -
        maxWordNumberDh)* normInputWordNumber_02 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormInputWordNumber_03 = ((minWordNumberDl -
        maxWordNumberDh)* normInputWordNumber_03 -
        Nh*minWordNumberDl + maxWordNumberDh *Nl)/(Nl - Nh);
        denormTargetBookNumber_01 = ((minTargetValueDl -
        maxTargetValueDh)*normTargetBookNumber_01
        Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
```

```
denormTargetBookNumber_02 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_02
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormTargetBookNumber_03 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_03
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormTargetBookNumber_04 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_04
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormTargetBookNumber_05 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_05
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_01 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_01
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_02 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_02
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_03 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_03
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_04 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_04
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_05 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_05
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
System.out.println ("RecordNumber = " + m);
System.out.println ("denormTargetBookNumber_01 = " +
denormTargetBookNumber_01 + "denormPredictBookNumber_01 = " +
denormPredictBookNumber_01);
System.out.println ("denormTargetBookNumber_02 = " +
denormTargetBookNumber_02 + "denormPredictBookNumber_02 = " +
denormPredictBookNumber_02);
System.out.println ("denormTargetBookNumber_03 = " +
denormTargetBookNumber_03 + "denormPredictBookNumber_03 = " +
denormPredictBookNumber_03);
System.out.println ("denormTargetBookNumber_04 = " +
denormTargetBookNumber_04 + "denormPredictBookNumber_04 = " +
denormPredictBookNumber_04);
System.out.println ("denormTargetBookNumber_05 = " +
denormTargetBookNumber_05 + "denormPredictBookNumber_05 = " +
denormPredictBookNumber_05);
//System.out.println (" ");
// Print the classification results in the log
      if(Math.abs(denormPredictBookNumber_01) > 0.85)
      if(Math.abs(denormPredictBookNumber_01) > 0.85  &
      Math.abs(denormPredictBookNumber_02) < 0.2   &
```

```
        Math.abs(denormPredictBookNumber_03) < 0.2   &
        Math.abs(denormPredictBookNumber_04) < 0.2   &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 1
belongs to book 1");
System.out.println (" ");
}
else
{
System.out.println ("Wrong results for record 1");
System.out.println (" ");
}
        if(Math.abs(denormPredictBookNumber_02) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) > 0.85 &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 2 belongs to book 2");
System.out.println (" ");
}
else
{
System.out.println ("Wrong results for record 2");
System.out.println (" ");
}
        if(Math.abs(denormPredictBookNumber_03) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) < 0.2 &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) > 0.85 &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2)
{
System.out.println ("Record 3 belongs to book 3");
System.out.println (" ");
}
else
{
System.out.println ("Wrong results for record 3");
System.out.println (" ");
}
        if(Math.abs(denormPredictBookNumber_04) > 0.85)
        if(Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) > 0.85 &
```

```
                Math.abs(denormPredictBookNumber_05) < 0.2)
                {
                System.out.println ("Record 4 belongs to book 4");
                System.out.println (" ");
                }
                else
                {
                System.out.println ("Wrong results for record 4");
                System.out.println (" ");
                }
                        if(Math.abs(denormPredictBookNumber_05) > 0.85)
                        if(Math.abs(denormPredictBookNumber_01) < 0.2  &
                        Math.abs(denormPredictBookNumber_02) < 0.2  &
                        Math.abs(denormPredictBookNumber_03) < 0.2  &
                        Math.abs(denormPredictBookNumber_04) < 0.2 &
                        Math.abs(denormPredictBookNumber_05) > 0.85)
                {
                System.out.println ("Record 5 belongs to book 5");
                System.out.println (" ");
                }
                else
                {
                System.out.println ("Wrong results for record 5");
                System.out.println (" ");
                }
    }  // End for pair loop
    returnCode = 0;
    return returnCode;
}  // End of the method
```

Daftar 20-5 menunjukkan fragmen kode dari metode pengujian. Di sini, Anda memuat kumpulan data pengujian dan jaringan terlatih yang disimpan sebelumnya di memori. Selanjutnya, Anda mengulang kumpulan data pasangan dan mengambil untuk setiap catatan tiga nomor buku input dan lima nomor buku target. Anda mendenormalisasi nilai yang diperoleh dan kemudian memeriksa apakah catatan tersebut milik salah satu dari lima buku.

Daftar 20-5. Fragmen Kode dari Metode Pengujian

```
// Load the test dataset into memory MLDataSet testingSet =
loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
 // Load the saved trained network network =
(BasicNetwork)EncogDirectoryPersistence.loadObject(new  File(networkFileName));
int i = 0;
for (MLDataPair pair:  testingSet)
{
        i++;
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
```

```
MLData predictData = network.compute(inputData);
// These values are Normalized as the whole input is
normInputWordNumberFromRecord = inputData.getData(0);
normTargetBookNumberFromRecord = actualData.getData(0);
normPredictXPointValueFromRecord = predictData.getData(0);
denormInputWordNumber_01 = ((minWordNumberDl -maxWordNumberDh)*
normInputWordNumber_01 - Nh*minWordNumberDl + maxWordNumberDh
*Nl)/(Nl - Nh);
denormInputWordNumber_02 = ((minWordNumberDl
maxWordNumberDh)*normInputWordNumber_02 - Nh*minWordNumberDl +
maxWordNumberDh *Nl)/(Nl - Nh);
denormInputWordNumber_03 = ((minWordNumberDl
maxWordNumberDh)*normInputWordNumber_03 - Nh*minWordNumberDl +
maxWordNumberDh *Nl)/(Nl - Nh);
denormTargetBookNumber_01 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_01 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
      denormTargetBookNumber_02 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_02 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
      denormTargetBookNumber_03 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_03 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormTargetBookNumber_04 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_04 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormTargetBookNumber_05 = ((minTargetValueDl -
maxTargetValueDh)*normTargetBookNumber_05 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_01 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_01 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_02 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_02 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_03 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_03 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_04 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_04 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictBookNumber_05 =((minTargetValueDl -
maxTargetValueDh)*normPredictBookNumber_05 Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
System.out.println ("RecordNumber = " + i);
      System.out.println ("denormTargetBookNumber_01 = " +
denormTargetBookNumber_01 + "denormPredictBookNumber_01 = " +
denormPredictBookNumber_01);
```

```
System.out.println ("denormTargetBookNumber_02 = " +
denormTargetBookNumber_02 + "denormPredictBookNumber_02 = " +
denormPredictBookNumber_02);
System.out.println ("denormTargetBookNumber_03 = " +
denormTargetBookNumber_03 + "denormPredictBookNumber_03 = " +
denormPredictBookNumber_03);
System.out.println ("denormTargetBookNumber_04 = " +
denormTargetBookNumber_04 + "denormPredictBookNumber_04 = " +
denormPredictBookNumber_04);
System.out.println ("denormTargetBookNumber_05 = " +
denormTargetBookNumber_05 + "denormPredictBookNumber_05 = " +
denormPredictBookNumber_05);
//System.out.println (" ");
        if(Math.abs(denormPredictBookNumber_01) > 0.85  &
        Math.abs(denormPredictBookNumber_02) < 0.2   &
        Math.abs(denormPredictBookNumber_03) < 0.2   &
        Math.abs(denormPredictBookNumber_04) < 0.2   &
        Math.abs(denormPredictBookNumber_05) < 0.2
|
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) > 0.85 &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2
|
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) > 0.85 &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2
|
        Math.abs(denormPredictBookNumber_01) < 0.2 &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) > 0.85 &
        Math.abs(denormPredictBookNumber_04) < 0.2  &
        Math.abs(denormPredictBookNumber_05) < 0.2
|
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) > 0.85 &
        Math.abs(denormPredictBookNumber_05) < 0.2
|
        Math.abs(denormPredictBookNumber_01) < 0.2  &
        Math.abs(denormPredictBookNumber_02) < 0.2  &
        Math.abs(denormPredictBookNumber_03) < 0.2  &
        Math.abs(denormPredictBookNumber_04) < 0.2 &
        Math.abs(denormPredictBookNumber_05) > 0.85)
```

```
        {
        System.out.println ("Record belong to some book");
        System.out.println (" ");
        }
        else
        {
        System.out.println ("Unknown book");          System.out.println (" ");
        }
    }  // End for pair loop
} // End of the method
```

20.7 HASIL PELATIHAN

Listing 20-6 menunjukkan hasil pelatihan/validasi.

Daftar 20-6. Hasil Pelatihan/Validasi

```
        RecordNumber = 1
denormTargetBookNumber_01 = 1.0
denormPredictBookNumber_01 = 1.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 3.6221384780432686E-9
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.3178162894256218E-8
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.220446049250313E-16
        Record 1 belongs to book 1
        RecordNumber = 2
denormTargetBookNumber_01 = 1.0
denormPredictBookNumber_01 = 1.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 3.6687665128098956E-9
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.0430401597982808E-8
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.220446049250313E-16
        Record 1 belongs to book 1
        RecordNumber = 3
denormTargetBookNumber_01 = 1.0
denormPredictBookNumber_01 = 1.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 4.35402175424926E-9
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 9.684705759571699E-9
denormTargetBookNumber_05 = -0.0
```

```
denormPredictBookNumber_05 = 2.220446049250313E-16
        Record 1 belongs to book 1
        RecordNumber = 4
denormTargetBookNumber_01 = 1.0
denormPredictBookNumber_01 = 1.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 6.477930192261283E-9
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 4.863816960298806E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.220446049250313E-16
        Record 1 belongs to book 1
        RecordNumber = 5
denormTargetBookNumber_01 = 1.0
denormPredictBookNumber_01 = 1.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 1.7098276960947345E-8
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 4.196660130517671E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.220446049250313E-16
        Record 1 belongs to book 1
        RecordNumber = 6
denormTargetBookNumber_01 = 1.0
denormPredictBookNumber_01 = 1.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 9.261896322110275E-8
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.6307949707593536E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.7755575615628914E-16
        Record 1 belongs to book 1
        RecordNumber = 7
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = 5.686340287525127E-12
denormTargetBookNumber_02 = 1.0
denormPredictBookNumber_02 = 0.9999999586267019
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.1329661653292078E-9
denormTargetBookNumber_05 = -0.0
```

```
denormPredictBookNumber_05 = 9.43689570931383E-16
        Record 2 belongs to book 2
        RecordNumber = 8
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = 1.0
denormPredictBookNumber_02 = 0.9999999999998506
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.091398971198032E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.6645352591003757E-15
        Record 2 belongs to book 2
        RecordNumber = 9
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = 1.0
denormPredictBookNumber_02 = 0.9999999999999962
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = -0.0
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.0686406759496947E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 3.7192471324942744E-15
        Record 2 belongs to book 2
        RecordNumber = 10
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = 1.0
denormPredictBookNumber_02 = 0.9999999999999798
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 2.2352120154778277E-12
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 7.627692921730045E-10
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 1.9817480989559044E-14
        Record 2 belongs to book 2
        RecordNumber = 11
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = 1.0
denormPredictBookNumber_02 = 0.9999999999999603
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 1.2451872866137137E-11
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 7.404629132068408E-10
denormTargetBookNumber_05 = -0.0
```

```
denormPredictBookNumber_05 = 2.298161660974074E-14
        Record 2 belongs to book 2
        RecordNumber = 12
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = 1.0
denormPredictBookNumber_02 = 0.9999999999856213
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 7.48775297876314E-8
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 6.947271091739537E-10
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 4.801714581503802E-14
        Record 2 belongs to book 2
        RecordNumber = 13
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 7.471272545078733E-9
denormTargetBookNumber_03 = 1.0
denormPredictBookNumber_03 = 0.9999999419988991
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.5249974888730264E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.027711332175386E-12
        Record 3 belongs to book 3
        RecordNumber = 14
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 2.295386103412511E-13
denormTargetBookNumber_03 = 1.0
denormPredictBookNumber_03 = 0.9999999999379154
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 4.873732140087128E-9
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 4.987454893523591E-12
        Record 3 belongs to book 3
        RecordNumber = 15
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = 2.692845946228317E-13
denormTargetBookNumber_03 = 1.0
denormPredictBookNumber_03 = 0.9999999998630087
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 4.701179112664988E-9
denormTargetBookNumber_05 = -0.0
```

```
denormPredictBookNumber_05 = 4.707678691318051E-12
        Record 3 belongs to book 3
        RecordNumber = 16
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = 1.0
denormPredictBookNumber_03 = 0.9999999999999996
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.0469307360215794E-8
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.843247859374287E-11
        Record 3 belongs to book 3
        RecordNumber = 17
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = 1.0
denormPredictBookNumber_03 = 0.9999999999999987
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.977055869017974E-8
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.68162714256448E-11
        Record 3 belongs to book 3
        RecordNumber = 18
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = 1.0
denormPredictBookNumber_03 = 0.9999999885142061
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.6820915488556807E-8
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 7.056188966458876E-12
        Record 3 belongs to book 3
        RecordNumber = 19
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 2.983344798979104E-8
denormTargetBookNumber_04 = 1.0
denormPredictBookNumber_04 = 0.9999999789933758
denormTargetBookNumber_05 = -0.0
```

```
denormPredictBookNumber_05 = 1.7987472622493783E-10
        Record 4 belongs to book 4
        RecordNumber = 20
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 1.0003242317813132E-7
denormTargetBookNumber_04 = 1.0
denormPredictBookNumber_04 = 0.9999999812213116
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.2566659652056842E-10
        Record 4 belongs to book 4
        RecordNumber = 21
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 1.4262971415046621E-8
denormTargetBookNumber_04 = 1.0
denormPredictBookNumber_04 = 0.9999999812440078
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.079504346497174E-10
        Record 4 belongs to book 4
        RecordNumber = 22
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 5.790115659154438E-8
denormTargetBookNumber_04 = 1.0
denormPredictBookNumber_04 = 0.9999999845075942
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 2.9504404475133583E-10
        Record 4 belongs to book 4
        RecordNumber = 23
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 6.890162551620449E-9
denormTargetBookNumber_04 = 1.0
denormPredictBookNumber_04 = 0.999999984526581
denormTargetBookNumber_05 = -0.0
```

```
denormPredictBookNumber_05 = 2.6966767707747863E-10
        Record 4 belongs to book 4
        RecordNumber = 24
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 9.975842318876715E-9
denormTargetBookNumber_04 = 1.0
denormPredictBookNumber_04 = 0.9999999856956441
denormTargetBookNumber_05 = -0.0
denormPredictBookNumber_05 = 3.077177401777931E-10
Record 4 belongs to book 4
RecordNumber = 25
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 3.569367024169878E-14
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.8838704707313525E-8
denormTargetBookNumber_05 = 1.0
denormPredictBookNumber_05 = 0.9999999996959972
        Record 5 belongs to book 5
        RecordNumber = 26
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 4.929390229335695E-14
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.943621164013365E-8
denormTargetBookNumber_05 = 1.0
denormPredictBookNumber_05 = 0.9999999997119369
        Record 5 belongs to book 5
        RecordNumber = 27
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 1.532107773982716E-14
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 1.926626319592728E-8
denormTargetBookNumber_05 = 1.0
```

```
denormPredictBookNumber_05 = 0.9999999996935514
        Record 5 belongs to book 5
        RecordNumber = 28
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 3.2862601528904634E-14
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.034116280968945E-8
denormTargetBookNumber_05 = 1.0
denormPredictBookNumber_05 = 0.9999999997226772
Record 5 belongs to book 5
RecordNumber = 29
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 1.27675647831893E-14
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.014738198496957E-8
denormTargetBookNumber_05 = 1.0
denormPredictBookNumber_05 = 0.9999999997076233
Record 5 belongs to book 5
RecordNumber = 30
denormTargetBookNumber_01 = -0.0
denormPredictBookNumber_01 = -0.0
denormTargetBookNumber_02 = -0.0
denormPredictBookNumber_02 = -0.0
denormTargetBookNumber_03 = -0.0
denormPredictBookNumber_03 = 2.0039525594484076E-14
denormTargetBookNumber_04 = -0.0
denormPredictBookNumber_04 = 2.0630209485172912E-8
denormTargetBookNumber_05 = 1.0
denormPredictBookNumber_05 = 0.9999999997212032
Record 5 belongs to book 5
```

Seperti yang ditunjukkan dalam log, program dengan benar mengidentifikasi nomor buku yang menjadi milik semua catatan .

20.8 HASIL PENGUJIAN

Listing 20-7 menunjukkan hasil pengujian.

Daftar 20-7. Hasil Pengujian

```
RecordNumber = 1
        Unknown book
RecordNumber = 2
```

```
        Unknown book
RecordNumber = 3
        Unknown book
RecordNumber = 4
        Unknown book
RecordNumber = 5
        Unknown book
RecordNumber = 6
        Unknown book
RecordNumber = 7
        Unknown book
RecordNumber = 8
        Unknown book
RecordNumber = 9
        Unknown book
RecordNumber = 10
        Unknown book
RecordNumber = 11
        Unknown book
RecordNumber = 12
        Unknown book
RecordNumber = 13
        Unknown book
RecordNumber = 14
        Unknown book
RecordNumber = 15
        Unknown book
```

Proses pengujian mengklasifikasikan objek dengan benar dengan menentukan bahwa semua catatan yang diproses bukan milik salah satu dari lima buku.

20.9 RINGKASAN

Bab ini menjelaskan cara menggunakan jaringan saraf untuk mengklasifikasikan objek. Secara khusus, contoh dalam bab ini menunjukkan bagaimana jaringan saraf dapat menentukan buku mana yang dimiliki oleh setiap catatan pengujian. Pada bab berikutnya, Anda akan mempelajari pentingnya memilih model pemrosesan yang benar.

BAB 21
PENTINGNYA MEMILIH MODEL YANG BENAR

Contoh yang dibahas dalam bab ini akan berakhir dengan hasil yang negatif. Namun, Anda bisa belajar banyak dari kesalahan seperti ini.

21.1 CONTOH 7: MEMPREDIKSI HARGA PASAR SAHAM BULAN DEPAN

Dalam contoh ini, Anda akan mencoba memprediksi harga dana yang diperdagangkan di bursa (ETF) SPY bulan depan; ini adalah ETF yang meniru indeks pasar saham S&P500. Alasan seseorang untuk mengembangkan proyek semacam itu bisa jadi seperti ini:

> *"Kami tahu bahwa harga pasar acak, naik turun setiap hari dan bereaksi terhadap berita yang berbeda. Namun, kami menggunakan harga bulanan, yang cenderung lebih stabil. Selain itu, pasar sering mengalami kondisi yang mirip dengan situasi masa lalu, sehingga orang (umumnya) harus bereaksi kurang lebih sama dengan kondisi yang sama. Oleh karena itu, dengan mengetahui bagaimana pasar bereaksi di masa lalu, kita harus dapat memprediksi dengan cermat perilaku pasar untuk bulan berikutnya."*

Dalam contoh ini, Anda akan menggunakan harga bulanan historis sepuluh tahun untuk ETF SPY dan akan mencoba memprediksi harga bulan depan. Tentu saja, menggunakan data SPY historis dari durasi yang lebih lama akan berkontribusi positif pada keakuratan prediksi; Namun, ini adalah sebuah contoh, jadi mari kita buat agar tetap masuk akal. Kumpulan data input berisi data selama sepuluh tahun (120 bulan), dari Januari 2000 hingga Januari 2009, dan Anda ingin memprediksi harga SPY pada akhir Februari 2009. Tabel 21.1 menunjukkan riwayat harga SPY bulanan untuk periode tersebut.

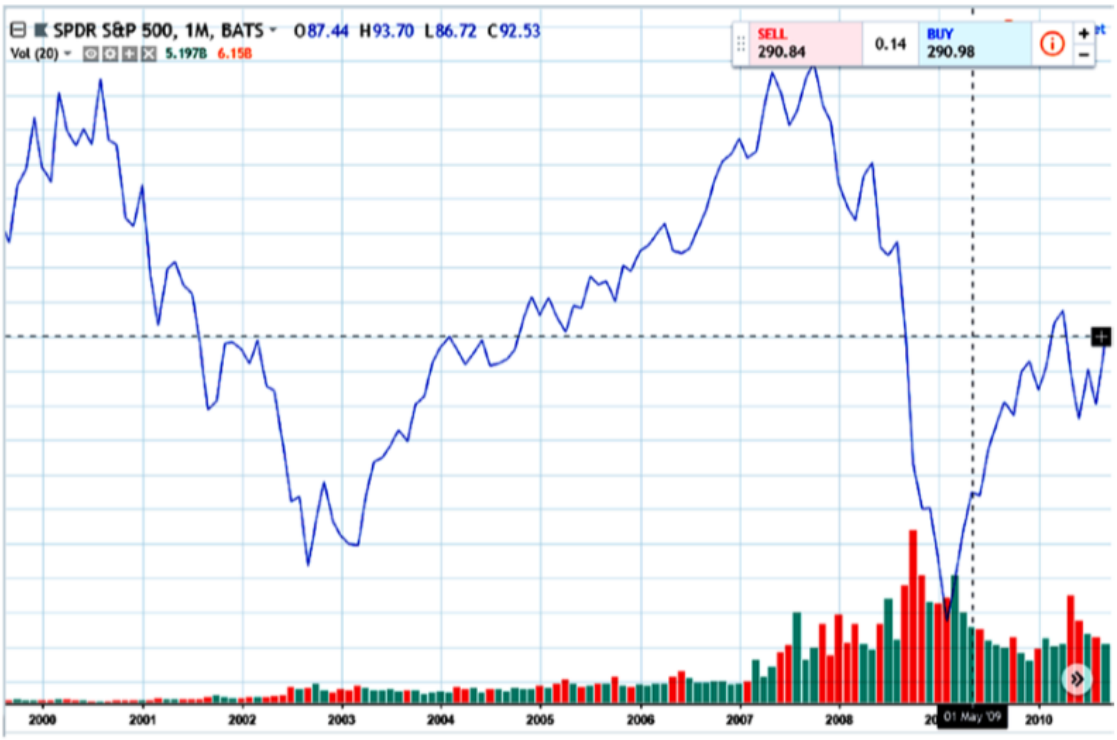
Tabel 21.1 Historis Harga ETF SPY Bulanan

Date	Price	Date	Price
200001	1394.46	200501	1181.27
200002	1366.42	200502	1203.6
200003	1498.58	200503	1180.59
200004	1452.43	200504	1156.85
200005	1420.6	200505	1191.5
200006	1454.6	200506	1191.33
200007	1430.83	200507	1234.18
200008	1517.68	200508	1220.33
200009	1436.51	200509	1228.81
200010	1429.4	200510	1207.01
200011	1314.95	200511	1249.48
200012	1320.28	200512	1248.29
200101	1366.01	200601	1280.08
200102	1239.94	200602	1280.66
200103	1160.33	200603	1294.87
200104	1249.46	200604	1310.61
200105	1255.82	200605	1270.09
200106	1224.38	200606	1270.2
200107	1211.23	200607	1276.66
200108	1133.58	200608	1303.82
200109	1040.94	200609	1335.85
200110	1059.78	200610	1377.94

200111	1139.45	200611	1400.63
200112	1148.08	200612	1418.3
200201	1130.2	200701	1438.24
200202	1106.73	200702	1406.82
200203	1147.39	200703	1420.86
200204	1076.92	200704	1482.37
200205	1067.14	200705	1530.62
200206	989.82	200706	1503.35
200207	911.62	200707	1455.27
200208	916.07	200708	1473.99
200209	815.28	200709	1526.75
200210	885.76	200710	1549.38
200211	936.31	200711	1481.14
200212	879.82	200712	1468.36
200301	855.7	200801	1378.55
200302	841.15	200802	1330.63
200303	848.18	200803	1322.7
200304	916.92	200804	1385.59
200305	963.59	200805	1400.38
200306	974.5	200806	1280
200307	990.31	200807	1267.38
200308	1008.01	200808	1282.83
200309	995.97	200809	1166.36
200310	1050.71	200810	968.75
200311	1058.2	200811	896.24

200312	1111.92	200812	903.25
200401	1131.13	200901	825.88
200402	1144.94	200902	735.09
200403	1126.21	200903	797.87
200404	1107.3	200904	872.81
200405	1120.68	200905	919.14
200406	1140.84	200906	919.32
200407	1101.72	200907	987.48
200408	1104.24	200908	1020.62
200409	1114.58	200909	1057.08
200410	1130.2	200910	1036.19
200411	1173.82	200911	1095.63
200412	1211.92	200912	1115.1

Gambar 21.1 menunjukkan grafik harga SPY bulanan historis.



Gambar 21.1 Bagan bulanan SPY untuk interval [2000/01 – 2009/01]

Perhatikan bahwa kumpulan data input mencakup harga pasar selama dua kehancuran pasar, sehingga jaringan harus dapat mempelajari tentang perilaku pasar selama kehancuran tersebut. Anda telah belajar dari contoh sebelumnya bahwa untuk membuat prediksi di luar rentang pelatihan, Anda perlu mengubah data asli ke format yang memungkinkan Anda melakukan ini. Jadi, sebagai bagian dari transformasi ini, Anda

membuat kumpulan data perbedaan harga dengan catatan yang menyertakan dua bidang ini:

- Field 1: Persentase perbedaan antara harga bulan ini dan sebelumnya
- Field 2: Persentase perbedaan antara harga bulan berikutnya dan saat ini

Tabel 21.2 menunjukkan fragmen dari kumpulan data perbedaan harga yang diubah

Tabel 21.2 Fragmen dari Kumpulan Data Selisih Harga

Field 1	Field 2		
priceDiffPerc	targetPriceDiffPerc	Date	InputPrice
-5.090352221	-2.010814222	200001	1394.46
-2.010814222	9.671989579	200002	1366.42
9.671989579	-3.079582004	200003	1498.58
-3.079582004	-2.191499762	200004	1452.43
-2.191499762	2.39335492	200005	1420.6
2.39335492	-1.63412622	200006	1454.6
-1.63412622	6.069903483	200007	1430.83
6.069903483	-5.348294766	200008	1517.68
-5.348294766	-0.494949565	200009	1436.51
-0.494949565	-8.006856024	200010	1429.4
-8.006856024	0.405338606	200011	1314.95
0.405338606	3.463659224	200012	1320.28
3.463659224	-9.229068601	200101	1366.01
-9.229068601	-6.420471958	200102	1239.94
-6.420471958	7.681435454	200103	1160.33
7.681435454	0.509019897	200104	1249.46
0.509019897	-2.503543501	200105	1255.82
-2.503543501	-1.07401297	200106	1224.38
-1.07401297	-6.410838569	200107	1211.23
-6.410838569	-8.172338962	200108	1133.58
-8.172338962	1.809902588	200109	1040.94
1.809902588	7.517597992	200110	1059.78
7.517597992	0.757382948	200111	1139.45
0.757382948	-1.557382761	200112	1148.08
-1.557382761	-2.076623606	200201	1130.2

-2.076623606	3.673886133	200202	1106.73
3.673886133	-6.141765224	200203	1147.39
-6.141765224	-0.908145452	200204	1076.92
-0.908145452	-7.245534794	200205	1067.14
-7.245534794	-7.90042634	200206	989.82
-7.90042634	0.488141989	200207	911.62
0.488141989	-11.00243431	200208	916.07
-11.00243431	8.64488274	200209	815.28
8.64488274	5.706963512	200210	885.76
8.64488274	5.706963512	200210	885.76
5.706963512	-6.033258216	200211	936.31
-6.033258216	-2.741469846	200212	879.82
-2.741469846	-1.700362276	200301	855.7
-1.700362276	0.835760566	200302	841.15
0.835760566	8.104411799	200303	848.18
8.104411799	5.089866073	200304	916.92
5.089866073	1.132224286	200305	963.59

Kolom 3 dan 4 dimasukkan untuk memudahkan perhitungan kolom 1 dan 2, tetapi diabaikan selama pemrosesan. Seperti biasa, Anda menormalkan kumpulan data ini pada interval [-1, 1]. Tabel 21.3 menunjukkan kumpulan data yang dinormalisasi.

Tabel 21.3 Fragmen dari Kumpulan Data Selisih Harga yang Dinormalisasi

priceDiffPerc	targetPriceDiffPerc	Date	inputPrice
-0.006023481	0.199279052	200001	1394.46
0.199279052	0.978132639	200002	1366.42
0.978132639	0.128027866	200003	1498.58
0.128027866	0.187233349	200004	1452.43
0.187233349	0.492890328	200005	1420.6
0.492890328	0.224391585	200006	1454.6
0.224391585	0.737993566	200007	1430.83
0.737993566	-0.023219651	200008	1517.68
-0.023219651	0.300336696	200009	1436.51
0.300336696	-0.200457068	200010	1429.4
-0.200457068	0.360355907	200011	1314.95
0.360355907	0.564243948	200012	1320.28
0.564243948	-0.281937907	200101	1366.01
-0.281937907	-0.094698131	200102	1239.94
-0.094698131	0.84542903	200103	1160.33
0.84542903	0.367267993	200104	1249.46
0.367267993	0.166430433	200105	1255.82
0.166430433	0.261732469	200106	1224.38
0.261732469	-0.094055905	200107	1211.23
-0.094055905	-0.211489264	200108	1133.58
-0.211489264	0.453993506	200109	1040.94
0.453993506	0.834506533	200110	1059.78
0.834506533	0.38382553	200111	1139.45
0.38382553	0.229507816	200112	1148.08
0.229507816	0.19489176	200201	1130.2

0.19489176	0.578259076	200202	1106.73
0.578259076	-0.076117682	200203	1147.39
-0.076117682	0.272790303	200204	1076.92
0.272790303	-0.14970232	200205	1067.14
-0.14970232	-0.193361756	200206	989.82
-0.193361756	0.365876133	200207	911.62
0.365876133	-0.400162287	200208	916.07
-0.400162287	0.909658849	200209	815.28
0.909658849	0.713797567	200210	885.76
0.713797567	-0.068883881	200211	936.31
-0.068883881	0.150568677	200212	879.82

Sekali lagi, abaikan kolom 3 dan 4. Kolom tersebut digunakan di sini untuk konvensi menyiapkan kumpulan data ini, tetapi kolom tersebut tidak diproses.

21.2 MENYERTAKAN TOPOLOGI FUNGSI DALAM KUMPULAN DATA

Selanjutnya, Anda akan menyertakan informasi tentang topologi fungsi dalam kumpulan data karena memungkinkan Anda untuk mencocokkan tidak hanya satu nilai Bidang 1 tetapi juga kumpulan 12 nilai Bidang 1 (yang berarti mencocokkan data senilai satu tahun). Untuk melakukan ini, Anda membuat file pelatihan dengan catatan jendela geser. Setiap catatan jendela geser terdiri dari 12 bidang inputPriceDiffPerc dari 12 catatan asli ditambah bidang targetPriceDiffPerc dari catatan asli berikutnya (catatan yang mengikuti 12 catatan asli). Tabel 21.4 menunjukkan fragmen kumpulan data yang dihasilkan.

Tabel 21.4 Fragmen Kumpulan Data Pelatihan yang Terdiri dari Rekaman Jendela Geser

Sliding Windows												
0.591	0.55	0.165	0.459	0.206	0.199	0.533	0.332	0.573	0.259	0.38	0.215	0.327
0.55	0.165	0.459	0.206	0.199	0.533	0.332	0.573	0.259	0.38	0.215	0.568	0.503
0.165	0.459	0.206	0.199	0.533	0.332	0.573	0.259	0.38	0.215	0.568	0.327	0.336
0.459	0.206	0.199	0.533	0.332	0.573	0.259	0.38	0.215	0.568	0.327	0.503	0.407
0.206	0.199	0.533	0.332	0.573	0.259	0.38	0.215	0.568	0.327	0.503	0.336	0.414
0.199	0.533	0.332	0.573	0.259	0.38	0.215	0.568	0.327	0.503	0.336	0.407	0.127
0.533	0.332	0.573	0.259	0.38	0.215	0.568	0.327	0.503	0.336	0.407	0.414	0.334
0.332	0.573	0.259	0.38	0.215	0.568	0.327	0.503	0.336	0.407	0.414	0.127	0.367
0.573	0.259	0.38	0.215	0.568	0.327	0.503	0.336	0.407	0.414	0.127	0.334	0.475
0.259	0.38	0.215	0.568	0.327	0.503	0.336	0.407	0.414	0.127	0.334	0.367	0.497
0.38	0.215	0.568	0.327	0.503	0.336	0.407	0.414	0.127	0.334	0.367	0.475	0.543
0.215	0.568	0.327	0.503	0.336	0.407	0.414	0.127	0.334	0.367	0.475	0.497	0.443
0.568	0.327	0.503	0.336	0.407	0.414	0.127	0.334	0.367	0.475	0.497	0.543	0.417
0.327	0.503	0.336	0.407	0.414	0.127	0.334	0.367	0.475	0.497	0.543	0.443	0.427
0.503	0.336	0.407	0.414	0.127	0.334	0.367	0.475	0.497	0.543	0.443	0.417	0.188
0.336	0.407	0.414	0.127	0.334	0.367	0.475	0.497	0.543	0.443	0.417	0.427	0.400
0.407	0.414	0.127	0.334	0.367	0.475	0.497	0.543	0.443	0.417	0.427	0.188	0.622
0.414	0.127	0.334	0.367	0.475	0.497	0.543	0.443	0.417	0.427	0.188	0.400	0.55
0.127	0.334	0.367	0.475	0.497	0.543	0.443	0.417	0.427	0.188	0.4	0.622	0.215
0.334	0.367	0.475	0.497	0.543	0.443	0.417	0.427	0.188	0.4	0.622	0.55	0.12
0.367	0.475	0.497	0.543	0.443	0.417	0.427	0.188	0.400	0.622	0.55	0.215	0.419
0.475	0.497	0.543	0.443	0.417	0.427	0.188	0.400	0.622	0.55	0.215	0.12	0.572
0.497	0.543	0.443	0.417	0.427	0.188	0.400	0.622	0.55	0.215	0.12	0.419	0.432
0.543	0.443	0.417	0.427	0.188	0.4	0.622	0.55	0.215	0.12	0.419	0.572	0.04
0.443	0.417	0.427	0.188	0.400	0.622	0.55	0.215	0.12	0.419	0.572	0.432	0.276
0.417	0.427	0.188	0.400	0.622	0.550	0.215	0.12	0.419	0.572	0.432	0.04	-0.074
0.427	0.188	0.400	0.622	0.55	0.215	0.12	0.419	0.572	0.432	0.040	0.276	0.102
0.188	0.400	0.622	0.55	0.215	0.12	0.419	0.572	0.432	0.04	0.276	-0.074	0.294
0.400	0.622	0.55	0.215	0.12	0.419	0.572	0.432	0.04	0.276	-0.07	0.102	0.650

Karena fungsinya tidak kontinu, Anda memecah kumpulan data ini menjadi mikro-batch (catatan satu bulan).

21.3 MEMBANGUN *FILE MICRO-BATCH*

Daftar 21-1 menunjukkan kode program yang membangun file mikro-batch dari kumpulan data jendela geser yang dinormalisasi.

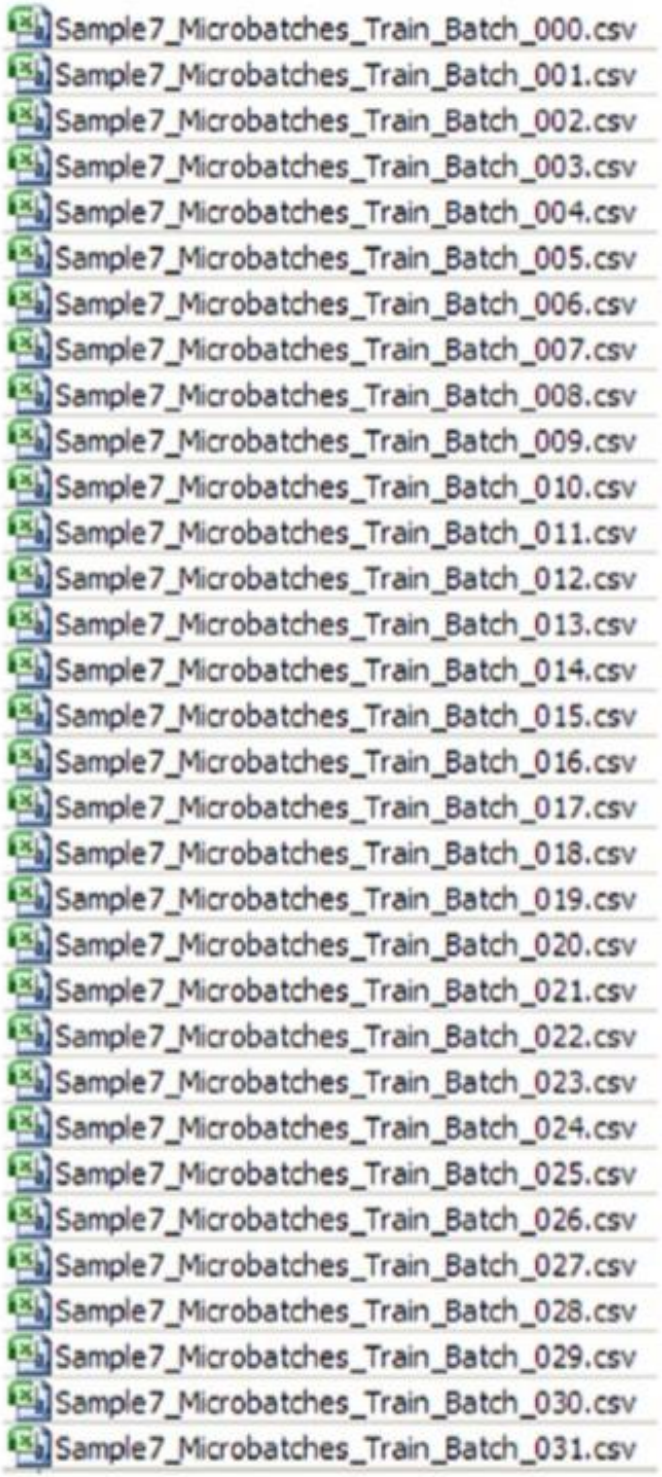
Daftar 21-1. Kode Program yang Membuat File Micro-Batch

```
// ============================================================
// Buat file mikro-batch dari file jendela geser yang
// dinormalisasi.
// Setiap kumpulan data mikro harus terdiri dari 12 bidang
// inputPriceDiffPerc
// diambil dari 12 catatan dalam file asli ditambah satu nilai
// targetPriceDiffPerc
// yang diambil dari catatan bulan berikutnya. Setiap
// mikro-batch menyertakan label record.
// ============================================================
package sample7_build_microbatches;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
public class Sample7_Build_MicroBatches
{
        // Config for Training
        static int numberOfRowsInInputFile = 121;
        static int numberOfRowsInBatch = 13;
        static String strInputFileName =
        "C:/My_Neural_Network_Book/Book_Examples/Sample7_SlidWindows_
        Train.csv";
        static String strOutputFileNameBase =
        "C:/My_Neural_Network_Book/Temp_Files/Sample7_Microbatches_
        Train_Batch_";
        // Config for Testing
        //static int numberOfRowsInInputFile = 122;
        //static int numberOfRowsInBatch = 13;
        //static String  strInputFileName =
        //          "C:/My_Neural_Network_Book/Book_Examples/Sample7_SlidWindows_
        Test.csv";   //static String  strOutputFileNameBase =
        //              "C:/My_Neural_Network_Book/Temp_Files/Sample7_Microbatches_
        Test_Batch_";
        static InputStream input = null;
// ========================================================
// Main method
// ========================================================
public static void main(String[] args)
```

```
{
        BufferedReader br;
        PrintWriter out;
        String cvsSplitBy = ",";
        String line = "";
        String lineLabel = "";
        String[] strOutputFileNames = new String[1070];
        String iString;
        String strOutputFileName;
        String[] strArrLine = new String[1086];
        int i;
        int r;
        // Read the original data and break it into batches
        try
        {
                // Delete all output file if they exist
                for (i = 0;
                i < numberOfRowsInInputFile; i++)
                {
                iString = Integer.toString(i);
                if(i < 10)              strOutputFileName = strOutputFileNameBase + "00" +
                iString + ".csv";
                else
                if (i >= 10 && i < 100)         strOutputFileName = strOutputFileNameBase
                + "0" + iString + ".csv";
        else
        strOutputFileName = strOutputFileNameBase + iString + ".csv";
        Files.deleteIfExists(Paths.get(strOutputFileName));
        }
        i = -1; // Input line number
        r = -2; // index to write in the memory
        br = new BufferedReader(new FileReader(strInputFileName));
        // Load all input recodes into memory
        while ((line = br.readLine()) != null)
        {
        i++;
        r++;
        if (i == 0)
        {
           // Save the label line
           lineLabel = line;
        }
        else
        {
           // Save the data in memory
           strArrLine[r] = line;
        }
        } // End of WHILE
```

```
        br.close();
        // Build batches
        br = new BufferedReader(new FileReader(strInputFileName));
        for (i = 0;
        i < numberOfRowsInInputFile - 1;
        i++)
        {
           iString = Integer.toString(i);
           // Construct the mini-batch
           if(i < 10)
                 strOutputFileName = strOutputFileNameBase + "00" +  iString + ".csv";
           else
        if (i >= 10 && i < 100)
        strOutputFileName = strOutputFileNameBase + "0" + iString + ".csv";
        else
        strOutputFileName = strOutputFileNameBase + iString + ".csv";
        out = new PrintWriter(new BufferedWriter(new FileWriter (strOutputFileName)));
        // write the header line as it is
        out.println(lineLabel);
        out.println(strArrLine[i]);
           out.close();
           }  // End of FOR i loop
           }  // End of TRY
           catch (IOException io)
           {
           io.printStackTrace();
           }
     }  // End of the Main method
}  // End of the class
```

Program ini memecah kumpulan data jendela geser menjadi file mikro-batch. Gambar 21.2 menunjukkan fragmen daftar file micro-batch.



Gambar 21.2 Fragmen daftar file mikro-batch

Daftar 21-2 menunjukkan bagaimana setiap kumpulan data mikro-batch terlihat saat dibuka.

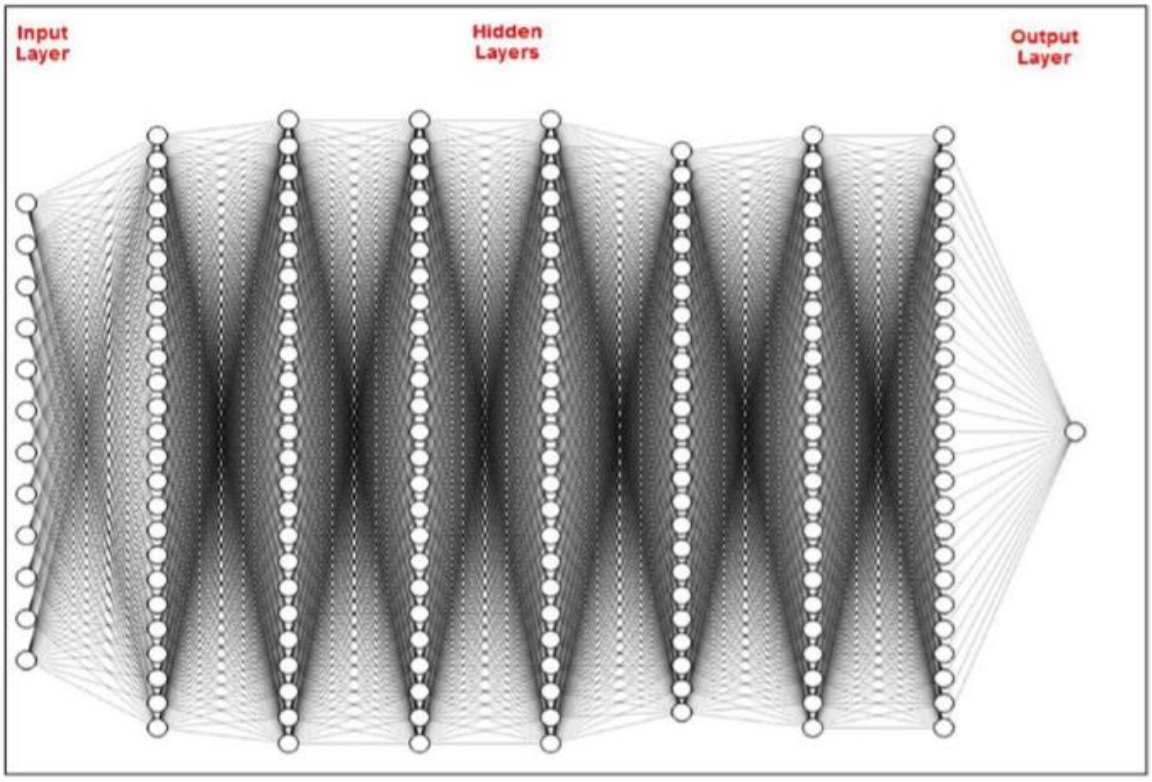
Daftar 21-2. Contoh File Micro-Batch

Sliding window micro-batch record
-0.006023481 0.199279052 0.978132639 0.128027866 0.187233349
0.492890328 0.224391585 0.737993566 -0.023219651 0.300336696 –
0.200457068 0.360355907 -0.281937907

Micro-batch files are the training files to be processed by the network.

21.4 ARSITEKTUR JARINGAN

Gambar 21.3 menunjukkan arsitektur jaringan untuk contoh ini. Jaringan memiliki 12 neuron input, tujuh lapisan tersembunyi (masing-masing dengan 25 neuron), dan lapisan output dengan satu neuron.



Gambar 21.3 Arsitektur jaringan

Sekarang Anda siap untuk membangun program pemrosesan jaringan.

Kode Program

Daftar 21-3 menunjukkan kode program.

Daftar 21-3. Kode Program Pemrosesan Jaringan Saraf

```
// ============================================================
// Perkirakan fungsi harga SPY menggunakan metode micro-batch.
// Setiap file mikro-batch menyertakan catatan label dan
// catatan data.
// Data record berisi 12 field inputPriceDiffPerc ditambah satu
// field targetPriceDiffPerc.
//
// Jumlah neuron Layer input adalah 12
// Jumlah neuron Layer output adalah 1
// ============================================================
package sample7;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
```

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient. ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
```

```
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample7 implements ExampleChart<XYChart>
{
	// Normalization parameters
	// Normalizing interval
	static double Nh =  1;
	static double Nl = -1;
	// inputPriceDiffPerc
	static double inputPriceDiffPercDh = 10.00;
	static double inputPriceDiffPercDl = -20.00;
	// targetPriceDiffPerc
	static double targetPriceDiffPercDh = 10.00;
	static double targetPriceDiffPercDl = -20.00;
	static String cvsSplitBy = ",";
	static Properties prop = null;
	static Date workDate = null;
	static int paramErrorCode = 0;
	static int paramBatchNumber = 0;
	static int paramDayNumber = 0;
	static String strWorkingMode;
	static String strNumberOfBatchesToProcess;
	static String strNumberOfRowsInInputFile;
	static String strNumberOfRowsInBatches;
	static String strIputNeuronNumber;
	static String strOutputNeuronNumber;
	static String strNumberOfRecordsInTestFile;
	static String strInputFileNameBase;
	static String strTestFileNameBase;
	static String strSaveNetworkFileNameBase;
	static String strTrainFileName;
	static String strValidateFileName;
	static String strChartFileName;
	static String strDatesTrainFileName;
	static String strPricesFileName;
	static int intWorkingMode;
	static int intNumberOfBatchesToProcess;
	static int intNumberOfRowsInBatches;
	static int intInputNeuronNumber;
	static int intOutputNeuronNumber;
	static String strOutputFileName;
	static String strSaveNetworkFileName;
	static String strNumberOfMonths;
	static String strYearMonth;
	static XYChart Chart;
	static String iString;
```

```
static double inputPriceFromFile;
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
// These arrays is where the two Date files are loaded
static Date[] yearDateTraining = new Date[150];
static String[] strTrainingFileNames = new String[150];
static String[] strTestingFileNames = new String[150];
static String[] strSaveNetworkFileNames = new String[150];
static BufferedReader br3;
static double recordNormInputPriceDiffPerc_00 = 0.00;
static double recordNormInputPriceDiffPerc_01 = 0.00;
static double recordNormInputPriceDiffPerc_02 = 0.00;
static double recordNormInputPriceDiffPerc_03 = 0.00;
static double recordNormInputPriceDiffPerc_04 = 0.00;
static double recordNormInputPriceDiffPerc_05 = 0.00;
static double recordNormInputPriceDiffPerc_06 = 0.00;
static double recordNormInputPriceDiffPerc_07 = 0.00;
static double recordNormInputPriceDiffPerc_08 = 0.00;
static double recordNormInputPriceDiffPerc_09 = 0.00;      static double
recordNormInputPriceDiffPerc_10 = 0.00;
static double recordNormInputPriceDiffPerc_11 = 0.00;
static double recordNormTargetPriceDiffPerc = 0.00;   static double tempMonth =
0.00;
static int intNumberOfSavedNetworks = 0;
static double[] linkToSaveInputPriceDiffPerc_00 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_01 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_02 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_03 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_04 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_05 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_06 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_07 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_08 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_09 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_10 = new double[150];
static double[] linkToSaveInputPriceDiffPerc_11 = new double[150];
static int[] returnCodes  = new int[3];
static int intDayNumber = 0;
static File file2 = null;
static double[] linkToSaveTargetPriceDiffPerc = new double[150];
static double[] arrPrices = new double[150];
@Override  public XYChart getChart()
{
// Create Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("Month").yAxisTitle("Price").build();
// Customize Chart
```

```
        Chart.getStyler().setPlotBackgroundColor(ChartColor.getAWTColor
        (ChartColor.GREY));
        Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
        Chart.getStyler().setChartBackgroundColor(Color.WHITE);
        Chart.getStyler().setLegendBackgroundColor(Color.PINK);
        Chart.getStyler().setChartFontColor(Color.MAGENTA);
        Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
        Chart.getStyler().setChartTitleBoxVisible(true);
        Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
        Chart.getStyler().setPlotGridLinesVisible(true);
        Chart.getStyler().setAxisTickPadding(20);
        Chart.getStyler().setAxisTickMarkLength(15);
        Chart.getStyler().setPlotMargin(20);
        Chart.getStyler().setChartTitleVisible(false);
        Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED,  Font.BOLD, 24));
        Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));    //
        Chart.getStyler().setLegendPosition(LegendPosition.InsideSE);
        Chart.getStyler().setLegendPosition(LegendPosition.OutsideE);
        Chart.getStyler().setLegendSeriesLineLength(12);
        Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF,  Font.ITALIC, 18));
        Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF,  Font.PLAIN, 11));
        Chart.getStyler().setDatePattern("yyyy-MM");
        Chart.getStyler().setDecimalPattern("#0.00");
        // Configuration
        // Set the mode of running this program
        intWorkingMode = 1;
        // Training mode
        if(intWorkingMode == 1)
        {
        // Training mode
        intNumberOfBatchesToProcess = 120;
        strInputFileNameBase                                                    =
        "C:/My_Neural_Network_Book/Temp_Files/Sample7_Microbatches_
        Train_Batch_";
        strSaveNetworkFileNameBase                                              =
        "C:/My_Neural_Network_Book/Temp_Files/Sample7_Save_Network_ Batch_";
        strChartFileName      =      "C:/My_Neural_Network_Book/Temp_Files/Sample7_
        XYLineChart_Train.jpg";
        strDatesTrainFileName                                                   =
        "C:/My_Neural_Network_Book/Book_Examples/Sample7_Dates_Real_
        SP500_3000.csv";
        strPricesFileName          =          "C:/My_Neural_Network_Book/Book_Examples/
        Sample7_InputPrice_SP500_200001_200901.csv";
    }
    else
    {
        // Testing mode        intNumberOfBatchesToProcess = 121;
        intNumberOfSavedNetworks = 120;
```

```
        strInputFileNameBase                                                    =
        "C:/My_Neural_Network_Book/Temp_Files/Sample7_Microbatches_
        Test_Batch_";
        strSaveNetworkFileNameBase                                              =
        "C:/My_Neural_Network_Book/Temp_Files/Sample7_Save_Network_ Batch_";
        strChartFileName                                                        =
        "C:/My_Neural_Network_Book/Book_Examples/Sample7_XYLineChart_ Test.jpg";
        strDatesTrainFileName                                                   =
        "C:/My_Neural_Network_Book/Book_Examples/Sample7_Dates_Real_
        SP500_3000.csv";
        trPricesFileName = "C:/My_Neural_Network_Book/Book_Examples/
        Sample7_InputPrice_SP500_200001_200901.csv";
        }
        // Common configuration
        intNumberOfRowsInBatches = 1;
        intInputNeuronNumber = 12;
        intOutputNeuronNumber = 1;
        //  Generate training batch file names and the corresponding Save Network file
        names and
        // save them arrays
        for (int i = 0;
        i < intNumberOfBatchesToProcess; i++)
        {
        iString = Integer.toString(i);
        // Construct the training batch names
        if (i < 10)
        {
        strOutputFileName = strInputFileNameBase + "00" + iString + ".csv";
        strSaveNetworkFileName = strSaveNetworkFileNameBase + "00" + iString + ".csv";
        }     else
        {
        if(i >=10 && i < 100)
        {
        strOutputFileName = strInputFileNameBase + "0" + iString + ".csv";
        strSaveNetworkFileName = strSaveNetworkFileNameBase + "0" + iString + ".csv";
        }
        else
        {
              strOutputFileName = strInputFileNameBase + iString + ".csv";
              strSaveNetworkFileName = strSaveNetworkFileNameBase + iString + ".csv";
    }
}
strSaveNetworkFileNames[i] = strSaveNetworkFileName;
if(intWorkingMode == 1)
{
strTrainingFileNames[i] = strOutputFileName;
File file1 = new File(strSaveNetworkFileNames[i]);
if(file1.exists())
```

```
file1.delete();
}
else
strTestingFileNames[i] = strOutputFileName;
}  // End the FOR loop
// Build the array linkToSaveInputPriceDiffPerc_01      String tempLine = null;      String[]
tempWorkFields = null;
recordNormInputPriceDiffPerc_00 = 0.00;
recordNormInputPriceDiffPerc_01 = 0.00;
recordNormInputPriceDiffPerc_02 = 0.00;
recordNormInputPriceDiffPerc_03 = 0.00;
recordNormInputPriceDiffPerc_04 = 0.00;
recordNormInputPriceDiffPerc_05 = 0.00;
recordNormInputPriceDiffPerc_06 = 0.00;
recordNormInputPriceDiffPerc_07 = 0.00;
recordNormInputPriceDiffPerc_08 = 0.00;
recordNormInputPriceDiffPerc_09 = 0.00;
recordNormInputPriceDiffPerc_10 = 0.00;
recordNormInputPriceDiffPerc_11 = 0.00;

double recordNormTargetPriceDiffPerc = 0.00;
try
{
for (int m = 0; m < intNumberOfBatchesToProcess; m++)
{
        if(intWorkingMode == 1)
        br3 = new BufferedReader(new FileReader(strTraining FileNames[m]));
        else
        br3 = new BufferedReader(new FileReader(strTesting FileNames[m]));
        // Skip the label record
        tempLine = br3.readLine();
        tempLine = br3.readLine();
        // Break the line using comma as separator
        tempWorkFields = tempLine.split(cvsSplitBy);
        recordNormInputPriceDiffPerc_00 = Double.parseDouble (tempWorkFields[0]);
        recordNormInputPriceDiffPerc_01 = Double.parseDouble (tempWorkFields[1]);
        recordNormInputPriceDiffPerc_02 = Double.parseDouble (tempWorkFields[2]);
        recordNormInputPriceDiffPerc_03 = Double.parseDouble (tempWorkFields[3]);
        recordNormInputPriceDiffPerc_04 = Double.parseDouble (tempWorkFields[4]);
        recordNormInputPriceDiffPerc_05 = Double.parseDouble (tempWorkFields[5]);
        recordNormInputPriceDiffPerc_06 = Double.parseDouble (tempWorkFields[6]);
        recordNormInputPriceDiffPerc_07 = Double.parseDouble (tempWorkFields[7]);
        recordNormInputPriceDiffPerc_08 = Double.parseDouble (tempWorkFields[8]);
        recordNormInputPriceDiffPerc_09 = Double.parseDouble (tempWorkFields[9]);
        recordNormInputPriceDiffPerc_10 = Double.parseDouble (tempWorkFields[10]);
        recordNormInputPriceDiffPerc_11 = Double.parseDouble (tempWorkFields[11]);
        recordNormTargetPriceDiffPerc =  Double.parseDouble (tempWorkFields[12]);
        linkToSaveInputPriceDiffPerc_00[m] = recordNormInputPrice DiffPerc_00;
```

```
            linkToSaveInputPriceDiffPerc_01[m] = recordNormInputPrice DiffPerc_01;
            linkToSaveInputPriceDiffPerc_02[m] = recordNormInputPrice DiffPerc_02;
            linkToSaveInputPriceDiffPerc_03[m] = recordNormInputPrice DiffPerc_03;
            linkToSaveInputPriceDiffPerc_04[m] = recordNormInputPrice DiffPerc_04;
            linkToSaveInputPriceDiffPerc_05[m] = recordNormInputPrice DiffPerc_05;
            linkToSaveInputPriceDiffPerc_06[m] = recordNormInputPrice DiffPerc_06;
            linkToSaveInputPriceDiffPerc_07[m] = recordNormInputPrice DiffPerc_07;
            linkToSaveInputPriceDiffPerc_08[m] = recordNormInputPrice DiffPerc_08;
            linkToSaveInputPriceDiffPerc_09[m] = recordNormInputPrice DiffPerc_09;
            linkToSaveInputPriceDiffPerc_10[m] = recordNormInputPrice DiffPerc_10;
            linkToSaveInputPriceDiffPerc_11[m] = recordNormInputPrice DiffPerc_11;
            linkToSaveTargetPriceDiffPerc[m] = recordNormTargetPrice DiffPerc;
        }  // End the FOR loop
    // Load dates into memory
    loadDatesInMemory();
            // Load Prices into memory
            loadPriceFileInMemory();
            file2 = new File(strChartFileName);
            if(file2.exists())
            file2.delete();
            // Test the working mode
            if(intWorkingMode == 1)
            {
            // Train batches and save the trained networks
            int  paramBatchNumber;
            returnCodes[0] = 0; // Clear the error Code
            returnCodes[1] = 0; // Set the initial batch Number to 1;
            returnCodes[2] = 0; // Set the initial day number;
            do
            {
                    paramErrorCode = returnCodes[0];
                    paramBatchNumber = returnCodes[1];
                    paramDayNumber = returnCodes[2];
            returnCodes                                                                                    =
            trainBatches(paramErrorCode,paramBatchNumber,paramDayNumber);
            } while (returnCodes[0] > 0);
            }   // End the train logic
            else
            {
                    // Load and test the network logic
                    loadAndTestNetwork();
            }  // End of ELSE
        } // End of Try
        catch (Exception e1)
        {
        e1.printStackTrace();
        }
        Encog.getInstance().shutdown();
```

```
    return Chart;
} // End of method
// ============================================================
// Load CSV to memory.
// @return The loaded dataset.
// ============================================================
public static MLDataSet loadCSV2Memory(String filename, int input,  int ideal,    Boolean
headers, CSVFormat format, Boolean significance)
{
        DataSetCODEC codec = new CSVDataCODEC(new File(filename), format, headers,
        input, ideal, significance);
        MemoryDataLoader load = new MemoryDataLoader(codec);
        MLDataSet dataset = load.external2Memory();
        return dataset;
}
// ============================================================
//  The main method.
//  @param Command line arguments. No arguments are used.
// ============================================================
public static void main(String[] args)
{
        ExampleChart<XYChart> exampleChart = new Sample7();
        XYChart Chart = exampleChart.getChart();
        new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
// ============================================================
// Mode 0. Latih kumpulan sebagai jaringan individual,
// simpan dalam file terpisah di disk.
// ============================================================
static public int[] trainBatches(int paramErrorCode,int paramBatch Number,
        int paramDayNumber)
{
        int rBatchNumber;
        double realDenormTargetToPredictPricePerc = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double normTargetPriceDiffPerc = 0.00;
        double normPredictPriceDiffPerc = 0.00;
        double normInputPriceDiffPercFromRecord = 0.00;
        double denormTargetPriceDiffPerc;
        double denormPredictPriceDiffPerc;
        double denormInputPriceDiffPercFromRecord;
        double workNormInputPrice;
        Date tempDate;
        double trainError;
        double realDenormPredictPrice;
        double realDenormTargetPrice;
```

```
// Build the network
BasicNetwork network = new BasicNetwork();
// Input layer
network.addLayer(new BasicLayer(null,true,intInputNeuronNumber));
// Hidden layer.
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
network.addLayer(new BasicLayer(new ActivationTANH(),true,25));
// Output layer
network.addLayer(new BasicLayer(new ActivationTANH(),false,intOutputN
euronNumber));
network.getStructure().finalizeStructure();     network.reset();
// Loop over batches
intDayNumber = paramDayNumber;
// Day number for the chart
for (rBatchNumber = paramBatchNumber;
rBatchNumber < intNumberOf BatchesToProcess;
rBatchNumber++)
{
intDayNumber++;
//if(rBatchNumber == 201)
// rBatchNumber = rBatchNumber;
// Load the training CVS file for the current batch in memory
MLDataSet trainingSet =
loadCSV2Memory(strTrainingFileNames                    [rBatchNumber],
intInputNeuronNumber,intOutputNeuronNumber,true,CSVFormat.  ENGLISH,false);
// train the neural network
ResilientPropagation train = new ResilientPropagation(network, trainingSet);
int epoch = 1;
double tempLastErrorPerc = 0.00;
do
        {
        train.iteration();
        epoch++;
        for (MLDataPair pair1:  trainingSet)
                {
                MLData inputData = pair1.getInput();
                MLData actualData = pair1.getIdeal();
                MLData predictData = network.compute(inputData);
                // These values are normalized
                normTargetPriceDiffPerc = actualData.getData(0);
                normPredictPriceDiffPerc = predictData.getData(0);
                // De-normalize these values
```

```
            denormTargetPriceDiffPerc = ((targetPriceDiffPercDl - target
            PriceDiffPercDh)*normTargetPriceDiffPerc - Nh*targetPrice
            DiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
            denormPredictPriceDiffPerc =((targetPriceDiffPercDl - target
            PriceDiffPercDh)*normPredictPriceDiffPerc - Nh*target
            PriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
            inputPriceFromFile = arrPrices[rBatchNumber+12];
            realDenormTargetPrice = inputPriceFromFile + inputPriceFrom
            File*denormTargetPriceDiffPerc/100;
            realDenormPredictPrice = inputPriceFromFile + inputPriceFrom
            File*denormPredictPriceDiffPerc/100;
            realDenormTargetToPredictPricePerc = (Math.abs(realDenorm
            TargetPrice - realDenormPredictPrice)/realDenormTarget
            Price)*100;
    }
    if (epoch >= 500 && realDenormTargetToPredictPricePerc > 0.00091)
    {
            returnCodes[0] = 1;
            returnCodes[1] = rBatchNumber;
            returnCodes[2] = intDayNumber-1;
            //System.out.println("Try again");
            return returnCodes;
    //System.out.println(realDenormTargetToPredictPricePerc);
    } while(realDenormTargetToPredictPricePerc >  0.0009);
    // This batch is optimized
    // Save the network for the current batch
    EncogDirectoryPersistence.saveObject(newFile(strSaveNetworkFileNames
    [rBatchNumber]),network);
    // Print the trained neural network results for the batch
    //System.out.println("Trained Neural Network Results");
    // Get the results after the network optimization     int i = - 1;
    // Index of the array to get results
    maxGlobalResultDiff = 0.00;
    averGlobalResultDiff = 0.00;
    sumGlobalResultDiff = 0.00;
    //if (rBatchNumber == 857)
    //    i = i;
    // Validation
    for (MLDataPair pair:  trainingSet)
    {
            i++;
            MLData inputData = pair.getInput();
            MLData actualData = pair.getIdeal();
            MLData predictData = network.compute(inputData);
            // These values are Normalized as the whole input is
            normTargetPriceDiffPerc = actualData.getData(0);
            normPredictPriceDiffPerc = predictData.getData(0);
            //normInputPriceDiffPercFromRecord[i] = inputData.getData(0);
```

```
                normInputPriceDiffPercFromRecord = inputData.getData(0);
                // De-normalize this data to show the real result value
                denormTargetPriceDiffPerc = ((targetPriceDiffPercDl - targetPrice
                DiffPercDh)*normTargetPriceDiffPerc - Nh*targetPriceDiffPercDl +
                targetPriceDiffPercDh*Nl)/(Nl - Nh);
                denormPredictPriceDiffPerc =((targetPriceDiffPercDl - targetPrice
                DiffPercDh)*normPredictPriceDiffPerc - Nh*targetPriceDiffPercDl +
                targetPriceDiffPercDh*Nl)/(Nl - Nh);
                denormInputPriceDiffPercFromRecord = ((inputPriceDiffPercDl -
                input PriceDiffPercDh)*normInputPriceDiffPercFromRecord -
                Nh*input PriceDiffPercDl + inputPriceDiffPercDh*Nl)/(Nl - Nh);
                // Get the price of the 12th element of the row inputPriceFromFile
                = arrPrices[rBatchNumber+12];
                // Convert denormPredictPriceDiffPerc and
                denormTargetPriceDiffPerc // to real de-normalized prices
                realDenormTargetPrice = inputPriceFromFile + inputPriceFromFile*
                (denormTargetPriceDiffPerc/100);
                realDenormPredictPrice = inputPriceFromFile + inputPriceFromFile*
                (denormPredictPriceDiffPerc/100);
                realDenormTargetToPredictPricePerc =
                (Math.abs(realDenormTarget Price -
                realDenormPredictPrice)/realDenormTargetPrice)*100;
                S ystem.out.println("Month = " + (rBatchNumber+1) + " targetPrice
                = " + realDenormTargetPrice + " predictPrice = " +
                realDenormPredict Price + " diff = " +
                realDenormTargetToPredictPricePerc);
                if (realDenormTargetToPredictPricePerc > maxGlobalResultDiff)
                {
                        maxGlobalResultDiff = realDenormTargetToPredictPricePerc;
                }
                        sumGlobalResultDiff = sumGlobalResultDiff +
                        realDenormTargetTo PredictPricePerc;
                        // Populate chart elements tempDate =
                        yearDateTraining[rBatchNumber+14];
                        //xData.add(tempDate);
                        tempMonth = (double) rBatchNumber+14;
                        xData.add(tempMonth);
                        yData1.add(realDenormTargetPrice);
                        yData2.add(realDenormPredictPrice);
                } // End for Price pair loop
        } // End of the loop over batches
        XYSeries series1 = Chart.addSeries("Actual price", xData, yData1);
        XYSeries series2 = Chart.addSeries("Predicted price", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Print the max and average results
```

```
                averGlobalResultDiff = sumGlobalResultDiff/intNumberOfBatchesToProcess;
                System.out.println(" ");
                System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff);
                System.out.println("averGlobalResultDiff = " + averGlobalResultDiff);
                System.out.println(" ");
                // Save the chart image
                try
                {
                        BitmapEncoder.saveBitmapWithDPI(Chart,          strChartFileName,
                        BitmapFormat.JPG, 100);
                }
                catch (Exception bt)
                {
                bt.printStackTrace();
                }
                System.out.println ("Chart and Network have been saved");
                System.out.println("End of validating batches for training");
                returnCodes[0] = 0;
                returnCodes[1] = 0;
                returnCodes[2] = 0;
                return returnCodes;
} // End of method
// ============================================================
// Mode 1. Muat jaringan terlatih yang disimpan sebelumnya dan
// proses uji coba mini-batch
// ============================================================
static public void loadAndTestNetwork()
{
        System.out.println("Testing the networks results");
        List<Double> xData = new ArrayList<Double>();
        List<Double> yData1 = new ArrayList<Double>();
        List<Double> yData2 = new ArrayList<Double>();
        double realDenormTargetToPredictPricePerc = 0;
        double maxGlobalResultDiff = 0.00;
        double averGlobalResultDiff = 0.00;
        double sumGlobalResultDiff = 0.00;
        double maxGlobalIndex = 0;
        recordNormInputPriceDiffPerc_00 = 0.00;
        recordNormInputPriceDiffPerc_01 = 0.00;
        recordNormInputPriceDiffPerc_02 = 0.00;
        recordNormInputPriceDiffPerc_03 = 0.00;
        recordNormInputPriceDiffPerc_04 = 0.00;
        recordNormInputPriceDiffPerc_05 = 0.00;
        recordNormInputPriceDiffPerc_06 = 0.00;
        recordNormInputPriceDiffPerc_07 = 0.00;
        recordNormInputPriceDiffPerc_08 = 0.00;
        recordNormInputPriceDiffPerc_09 = 0.00;
        recordNormInputPriceDiffPerc_10 = 0.00;
```

```
recordNormInputPriceDiffPerc_11 = 0.00;
double recordNormTargetPriceDiffPerc = 0.00;
double normTargetPriceDiffPerc;
double normPredictPriceDiffPerc;
double normInputPriceDiffPercFromRecord;
double denormTargetPriceDiffPerc;
double denormPredictPriceDiffPerc;
double denormInputPriceDiffPercFromRecord;
double realDenormTargetPrice = 0.00;
double realDenormPredictPrice = 0.00;
double minVectorValue = 0.00;
String tempLine;
String[] tempWorkFields;
int tempMinIndex = 0;
double rTempPriceDiffPerc = 0.00;
double rTempKey = 0.00;
double vectorForNetworkRecord = 0.00;
double r_00 = 0.00;
double r_01 = 0.00;
double r_02 = 0.00;
double r_03 = 0.00;
double r_04 = 0.00;
double r_05 = 0.00;
double r_06 = 0.00;
double r_07 = 0.00;
double r_08 = 0.00;
double r_09 = 0.00;
double r_10 = 0.00;
double r_11 = 0.00;
double vectorDiff;
double r1 = 0.00;
double r2 = 0.00;
double vectorForRecord = 0.00;
int k1 = 0;
int k3 = 0;
44BufferedReader br4;
BasicNetwork network;
try
{
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        for (k1 = 0;
        k1 < intNumberOfBatchesToProcess;
        k1++)
        {
                br4 = new BufferedReader(new
                FileReader(strTestingFileNames[k1]));
```

```
            tempLine = br4.readLine();
            // Skip the label record          tempLine = br4.readLine();
            // Break the line using comma as separator          tempWorkFields =
            tempLine.split(cvsSplitBy);
            recordNormInputPriceDiffPerc_00                                    =
            Double.parseDouble(tempWork Fields[0]);
            recordNormInputPriceDiffPerc_01                                    =
            Double.parseDouble(tempWork Fields[1]);
            recordNormInputPriceDiffPerc_02                                    =
            Double.parseDouble(tempWork Fields[2]);
            recordNormInputPriceDiffPerc_03                                    =
            Double.parseDouble(tempWork Fields[3]);
            recordNormInputPriceDiffPerc_04                                    =
            Double.parseDouble(tempWork Fields[4]);
            recordNormInputPriceDiffPerc_05                                    =
            Double.parseDouble(tempWork Fields[5]);
            recordNormInputPriceDiffPerc_06                                    =
            Double.parseDouble(tempWork Fields[6]);
            recordNormInputPriceDiffPerc_07                                    =
            Double.parseDouble(tempWork Fields[7]);
            recordNormInputPriceDiffPerc_08                                    =
            Double.parseDouble(tempWork Fields[8]);
            recordNormInputPriceDiffPerc_09                                    =
            Double.parseDouble(tempWork Fields[9]);
            recordNormInputPriceDiffPerc_10                                    =
            Double.parseDouble(tempWork Fields[10]);
            recordNormInputPriceDiffPerc_11                                    =
            Double.parseDouble(tempWork Fields[11]);
            recordNormTargetPriceDiffPerc = Double.parseDouble(tempWork
            Fields[12]);
            if(k1 < 120)
            {
// Load the network for the current record                    network =
(BasicNetwork)EncogDirectoryPersistence.loadObject
(newFile(strSaveNetworkFileNames[k1]));
// Load the training file record                    MLDataSet testingSet =
loadCSV2Memory(strTestingFileNames[k1],          intInputNeuronNumber,
intOutputNeuronNumber,true,          CSVFormat.ENGLISH,false);
// Get the results from the loaded previously saved networks          int i = -
1;
for (MLDataPair pair:  testingSet)
            {
            i++;
            MLData inputData = pair.getInput();
            MLData actualData = pair.getIdeal();
            MLData predictData = network.compute(inputData);
            // These values are Normalized as the whole input is
            normTargetPriceDiffPerc = actualData.getData(0);
```

```
                normPredictPriceDiffPerc = predictData.getData(0);
                normInputPriceDiffPercFromRecord = inputData.getData(11);
                // De-normalize this data denormTargetPriceDiffPerc =
                ((targetPriceDiffPercDl
                targetPriceDiffPercDh)*normTargetPriceDiffPerc - Nh*target
                PriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
                denormPredictPriceDiffPerc =((targetPriceDiffPercDl -
                targetPriceDiffPercDh)*normPredictPriceDiffPerc - Nh*
                targetPriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
                denormInputPriceDiffPercFromRecord = ((inputPriceDiff PercDl -
                inputPriceDiffPercDh)*normInputPriceDiffPercFrom Record
                - Nh*inputPriceDiffPercDl + inputPriceDiff PercDh*Nl)/(Nl -
                Nh);
                inputPriceFromFile = arrPrices[k1+12];
                // Convert denormPredictPriceDiffPerc and denormTarget
                PriceDiffPerc to a real // de-normalize price
                realDenormTargetPrice = inputPriceFromFile + inputPrice
                FromFile*(denormTargetPriceDiffPerc/100);
                realDenormPredictPrice = inputPriceFromFile + inputPrice
                FromFile*(denormPredictPriceDiffPerc/100);
                realDenormTargetToPredictPricePerc = (Math.abs(realDenorm
                TargetPrice –
                realDenormPredictPrice)/realDenormTarget Price)*100;
                S ystem.out.println("Month = " + (k1+1) + " targetPrice = " +
                realDenormTargetPrice + " predictPrice = " + real
                DenormPredictPrice + " diff = " + realDenormTargetTo
                PredictPricePerc);
        } // End for pair loop
} // End for IF
else
{
        vectorForRecord =
        Math.sqrt( Math.pow(recordNormInputPriceDiffPerc_00,2) +
        Math.pow(recordNormInputPriceDiffPerc_01,2) +
        Math.pow(recordNormInputPriceDiffPerc_02,2) +
        Math.pow(recordNormInputPriceDiffPerc_03,2) +
        Math.pow(recordNormInputPriceDiffPerc_04,2) +
        Math.pow(recordNormInputPriceDiffPerc_05,2) +
        Math.pow(recordNormInputPriceDiffPerc_06,2) +
        Math.pow(recordNormInputPriceDiffPerc_07,2) +
        Math.pow(recordNormInputPriceDiffPerc_08,2) +
        Math.pow(recordNormInputPriceDiffPerc_09,2) +
        Math.pow(recordNormInputPriceDiffPerc_10,2) +
        Math.pow(recordNormInputPriceDiffPerc_11,2));
        //  Look for the network of previous days that closely matches
        // the value of vectorForRecord
        minVectorValue = 999.99;
        for (k3 = 0; k3 < intNumberOfSavedNetworks; k3++)
```

```
            {
                r_00 = linkToSaveInputPriceDiffPerc_00[k3];
                r_01 = linkToSaveInputPriceDiffPerc_01[k3];
                r_02 = linkToSaveInputPriceDiffPerc_02[k3];
                r_03 = linkToSaveInputPriceDiffPerc_03[k3];
                r_04 = linkToSaveInputPriceDiffPerc_04[k3];
                r_05 = linkToSaveInputPriceDiffPerc_05[k3];
                r_06 = linkToSaveInputPriceDiffPerc_06[k3];
                r_07 = linkToSaveInputPriceDiffPerc_07[k3];
                r_08 = linkToSaveInputPriceDiffPerc_08[k3];
                r_09 = linkToSaveInputPriceDiffPerc_09[k3];
                r_10 = linkToSaveInputPriceDiffPerc_10[k3];
                r_11 = linkToSaveInputPriceDiffPerc_11[k3];
                r2 = linkToSaveTargetPriceDiffPerc[k3];
                vectorForNetworkRecord = Math.sqrt(
                Math.pow(r_00,2) +
                Math.pow(r_01,2) +
                Math.pow(r_02,2) +
                Math.pow(r_03,2) +
                Math.pow(r_04,2) +
                Math.pow(r_05,2) +
                Math.pow(r_06,2) +
                Math.pow(r_07,2) +
                Math.pow(r_08,2) +
                Math.pow(r_09,2) +
                Math.pow(r_10,2) +
                Math.pow(r_11,2));
                vectorDiff = Math.abs(vectorForRecord - vectorFor NetworkRecord);
                if(vectorDiff < minVectorValue)
                {
                minVectorValue = vectorDiff;
                // Save this network record attributes
                rTempKey = r_00;
                rTempPriceDiffPerc = r2;
                tempMinIndex = k3;
            }
    } // End  FOR k3 loop
        network = (BasicNetwork)EncogDirectoryPersistence.loadObject
        (newFile(strSaveNetworkFileNames[tempMinIndex]));
        // Now, tempMinIndex points to the corresponding saved network
        // Load this network MLDataSet testingSet =
        loadCSV2Memory(strTestingFileNames[k1],
        intInputNeuronNumber,intOutputNeuronNumber,true,CSVFormat.
        ENGLISH,false);
        //  Get the results from the previously saved and  now loaded network
        int i = - 1;
        for (MLDataPair pair:  testingSet)
        {
```

```
                i++;
                MLData inputData = pair.getInput();
                MLData actualData = pair.getIdeal();
                MLData predictData = network.compute(inputData);
                // These values are Normalized as the whole input is
                normTargetPriceDiffPerc = actualData.getData(0);
                normPredictPriceDiffPerc = predictData.getData(0);
                normInputPriceDiffPercFromRecord = inputData.getData(11);
                // Renormalize this data to show the real result value
                denormTargetPriceDiffPerc = ((targetPriceDiffPercDl -
                targetPriceDiffPercDh)*normTargetPriceDiffPerc - Nh*
                targetPriceDiffPercDl + targetPriceDiffPercDh*Nl)/ (Nl - Nh);
                denormPredictPriceDiffPerc =((targetPriceDiffPercDl -
                targetPriceDiffPercDh)*normPredictPriceDiffPerc - Nh*
                targetPriceDiffPercDl + targetPriceDiffPercDh*Nl)/ (Nl - Nh);
                denormInputPriceDiffPercFromRecord = ((inputPriceDiff PercDl -
                inputPriceDiffPercDh)*normInputPriceDiffPerc FromRecord -
                Nh*inputPriceDiffPercDl + inputPriceDiff PercDh*Nl)/(Nl - Nh);
                inputPriceFromFile = arrPrices[k1+12];
                //  Convert denormPredictPriceDiffPerc and denormTargetPriceDiffPerc to
                a real
                // demoralize prices realDenormTargetPrice = inputPriceFromFile
                + inputPriceFromFile*(denormTargetPriceDiffPerc/100);
                realDenormPredictPrice = inputPriceFromFile + inputPriceFrom
                File*(denormPredictPriceDiffPerc/100);
                realDenormTargetToPredictPricePerc = (Math.abs(realDenorm TargetPrice
                - realDenormPredictPrice)/realDenormTarget Price)*100;
                System.out.println("Month = " + (k1+1) + " targetPrice = " +
                realDenormTargetPrice + " predictPrice = " + real
                DenormPredictPrice + " diff = " + realDenormTargetTo
                PredictPricePerc);
                if (realDenormTargetToPredictPricePerc > maxGlobal ResultDiff)
                {  maxGlobalResultDiff = realDenormTargetToPredict PricePerc;
                }
                sumGlobalResultDiff = sumGlobalResultDiff + realDenorm
                TargetToPredictPricePerc;
            } // End of IF
        }  // End for pair loop
        // Populate chart elements
        tempMonth = (double) k1+14;
        xData.add(tempMonth);
        yData1.add(realDenormTargetPrice);
        yData2.add(realDenormPredictPrice);
    }  // End of loop K1
    // Print the max and average results
            System.out.println(" ");
            System.out.println(" ");
            System.out.println("Results of processing testing batches");
```

```
        averGlobalResultDiff = sumGlobalResultDiff/intNumberOfBatches ToProcess;
        System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff + "  i = " +
        maxGlobalIndex);
        System.out.println("averGlobalResultDiff = " + averGlobalResult Diff);
        System.out.println(" ");
        System.out.println(" ");
        }    // End of TRY
        catch (IOException e1)
        {
        e1.printStackTrace();
        }
        // All testing batch files have been processed          XYSeries series1 =
        Chart.addSeries("Actual Price", xData, yData1);
        XYSeries series2 = Chart.addSeries("Forecasted Price", xData, yData2);
        series1.setLineColor(XChartSeriesColors.BLUE);
        series2.setMarkerColor(Color.ORANGE);
        series1.setLineStyle(SeriesLines.SOLID);
        series2.setLineStyle(SeriesLines.SOLID);
        // Save the chart image     try
        {
        BitmapEncoder.saveBitmapWithDPI(Chart, strChartFileName, BitmapFormat.JPG,
        100);
        }
        catch (Exception bt)
        {
        bt.printStackTrace();
        }
        System.out.println ("The Chart has been saved");
        System.out.println("End of testing for mini-batches training");
    } // End of the method
// ============================================================
// Load training dates file in memory
// ============================================================
public static void loadDatesInMemory()
{
        BufferedReader br1 = null;
        DateFormat sdf = new SimpleDateFormat("yyyy-MM");
        Date dateTemporateDate = null;
        String strTempKeyorateDate;
        int intTemporateDate;
        String line = "";
        String cvsSplitBy = ",";
        try
        {
        br1 = new BufferedReader(new FileReader(strDatesTrainFileName));
        int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)         {
```

```
            i++;
            r++;
            // Skip the header line          if(i > 0)
            {
            // Break the line using comma as separator
            String[] workFields = line.split(cvsSplitBy);
            strTempKeyorateDate = workFields[0];
            intTemporateDate = Integer.parseInt(strTempKeyorateDate);
            try
            {
            dateTemporateDate = convertIntegerToDate(intTemporateDate);
            }
            catch (ParseException e)
            {
            e.printStackTrace();
            System.exit(1);
            }
            yearDateTraining[r] = dateTemporateDate;
            }
        }  // end of the while loop
            br1.close();
            }
            catch (IOException ex)
            {
            ex.printStackTrace();
            System.err.println("Error opening files = " + ex);
            System.exit(1);
        }
}
// ============================================================
// Ubah tanggal bulan sebagai bilangan bulat ke variabel Tanggal
// ============================================================
public static Date convertIntegerToDate(int denormInputDateI) throws ParseException
{
            int numberOfYears = denormInputDateI/12;
            int numberOfMonths = denormInputDateI - numberOfYears*12;
            if (numberOfMonths == 0)
            {
            numberOfYears = numberOfYears - 1;
            numberOfMonths = 12;
            }
            String strNumberOfYears = Integer.toString(numberOfYears);
            if(numberOfMonths < 10)
            {
            strNumberOfMonths = Integer.toString(numberOfMonths);
            strNumberOfMonths = "0" + strNumberOfMonths;
            }       else
            {
```

```
        strNumberOfMonths = Integer.toString(numberOfMonths);
        }
        / /strYearMonth = "01-" + strNumberOfMonths + "-" + strNumberOfYears +
        "T09:00:00.000Z";
        strYearMonth = strNumberOfYears + "-" + strNumberOfMonths;
        DateFormat sdf = new SimpleDateFormat("yyyy-MM");
        try
        {
        workDate = sdf.parse(strYearMonth);
        }
        catch (ParseException e)
        {
        e.printStackTrace();
        }
        return workDate;
    }  // End of method
// ============================================================
// Ubah tanggal bulan sebagai integer ke variabel string strDate
// ============================================================
public static String convertIntegerToString(int denormInputDateI)
{
        int numberOfYears = denormInputDateI/12;
        int numberOfMonths = denormInputDateI - numberOfYears*12;
        if (numberOfMonths == 0)
        {
        numberOfYears = numberOfYears - 1;
        numberOfMonths = 12;
        }
        String strNumberOfYears = Integer.toString(numberOfYears);
        if(numberOfMonths < 10)
        {
        strNumberOfMonths = Integer.toString(numberOfMonths);
        strNumberOfMonths = "0" + strNumberOfMonths;
        }
        else
        {
        strNumberOfMonths = Integer.toString(numberOfMonths);
        }
        strYearMonth = strNumberOfYears + "-" + strNumberOfMonths;
    return strYearMonth;
}  // End of method
// ============================================================
// Load Prices file in memory
// ============================================================
public static void loadPriceFileInMemory()
{
        BufferedReader br1 = null;
        String line = "";
```

```
        String cvsSplitBy = ",";
        String strTempKeyPrice = "";
        double tempPrice = 0.00;
        try
        {
        br1 = new BufferedReader(new FileReader(strPricesFileName));
        int i = -1;
        int r = -2;
        while ((line = br1.readLine()) != null)
        {
                i++;
                r++;
                // Skip the header line
                if(i > 0)
                {
                // Break the line using comma as separator          String[] workFields =
                line.split(cvsSplitBy);
                strTempKeyPrice = workFields[0];
                tempPrice = Double.parseDouble(strTempKeyPrice);
                arrPrices[r] = tempPrice;
        }
    } // end of the while loop
        br1.close();
}
        catch (IOException ex)
        {
        ex.printStackTrace();
        System.err.println("Error opening files = " + ex);
        System.exit(1);
        }
  }
} // End of the  Encog class
```

21.5 PROSES PELATIHAN

Untuk sebagian besar, logika metode pelatihan mirip dengan apa yang digunakan pada contoh sebelumnya, sehingga tidak memerlukan penjelasan apa pun, kecuali satu bagian yang akan saya bahas di sini.

Terkadang Anda harus berurusan dengan fungsi yang memiliki nilai yang sangat kecil, sehingga kesalahan yang dihitung menjadi lebih kecil. Misalnya, kesalahan jaringan dapat mencapai nilai mikroskopis seperti 14 atau lebih nol setelah titik, seperti pada 0,000000000000025. Ketika Anda mendapatkan kesalahan seperti itu, Anda akan mulai mempertanyakan keakuratan perhitungan. Dalam kode ini, saya telah menyertakan contoh bagaimana menangani situasi seperti itu.

Alih-alih hanya memanggil metode train.getError() untuk menentukan kesalahan jaringan, Anda menggunakan kumpulan data pasangan untuk mengambil nilai fungsi input, aktual, dan prediksi dari jaringan untuk setiap epoch; denormalisasi nilai-nilai itu; dan hitung perbedaan persen kesalahan antara nilai yang dihitung dan nilai aktual. Anda

kemudian keluar dari loop pasangan dengan nilai returnCode 0 ketika perbedaan ini kurang dari batas kesalahan. Hal ini ditunjukkan pada Daftar 21-4.

Daftar 21-4. Memeriksa Kesalahan Menggunakan Nilai Fungsi Sebenarnya

```
int epoch = 1;
double tempLastErrorPerc = 0.00;
do
        {
                train.iteration();
           epoch++;
                for (MLDataPair pair1:  trainingSet)
                {
                    MLData inputData = pair1.getInput();
                    MLData actualData = pair1.getIdeal();
                    MLData predictData = network.compute(inputData);
                    // These values are Normalized as the whole input is
                    normTargetPriceDiffPerc = actualData.getData(0);
                    normPredictPriceDiffPerc = predictData.getData(0);
                    denormTargetPriceDiffPerc = ((targetPriceDiffPercDl
                    targetPriceDiffPercDh)*normTargetPriceDiffPerc - Nh*target
                    PriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
                    denormPredictPriceDiffPerc =((targetPriceDiffPercDl
                    targetPriceDiffPercDh)*normPredictPriceDiffPerc - Nh*
                    targetPriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
                    inputPriceFromFile = arrPrices[rBatchNumber+12];
                    realDenormTargetPrice = inputPriceFromFile + inputPriceFrom
                    File*denormTargetPriceDiffPerc/100;
                    realDenormPredictPrice = inputPriceFromFile + inputPriceFrom
                    File*denormPredictPriceDiffPerc/100;
                    realDenormTargetToPredictPricePerc = (Math.abs(realDenorm
                    TargetPrice - realDenormPredictPrice)/realDenormTarget
                    Price)*100;
                }
            if (epoch >= 500 && realDenormTargetToPredictPricePerc > 0.00091)
                {
                        returnCodes[0] = 1;
                        returnCodes[1] = rBatchNumber;
                        returnCodes[2] = intDayNumber-1;
                        return returnCodes;
                }
            } while(realDenormTargetToPredictPricePerc >  0.0009);
```

Hasil Pelatihan

Listing 21-5 menunjukkan hasil pelatihan.

Daftar 21-5. Hasil Pelatihan

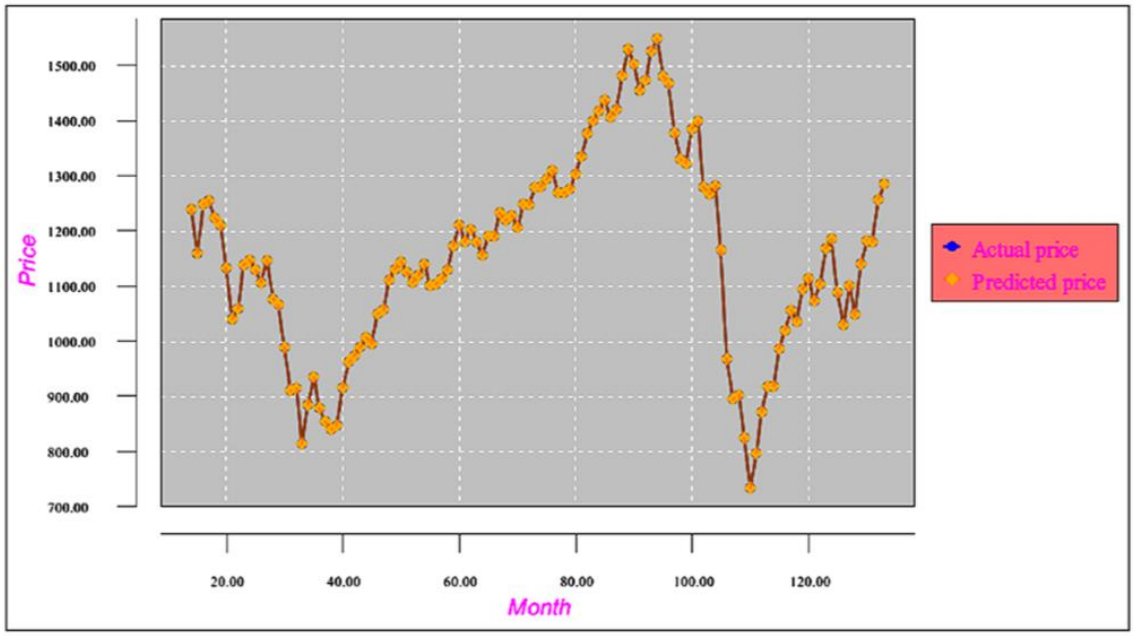
```
Month =  1  targetPrice = 1239.94000  predictPrice = 1239.93074 diff = 7.46675E-4
Month =  2  targetPrice = 1160.33000  predictPrice = 1160.32905 diff = 8.14930E-5
Month =  3  targetPrice = 1249.46000  predictPrice = 1249.44897 diff = 8.82808E-4
Month =  4  targetPrice = 1255.82000  predictPrice = 1255.81679 diff = 2.55914E-4
Month =  5  targetPrice = 1224.38000  predictPrice = 1224.37483 diff = 4.21901E-4
```

Month = 6 targetPrice = 1211.23000 predictPrice = 1211.23758 diff = 6.25530E-4
Month = 7 targetPrice = 1133.58000 predictPrice = 1133.59013 diff = 8.94046E-4
Month = 8 targetPrice = 1040.94000 predictPrice = 1040.94164 diff = 1.57184E-4
Month = 9 targetPrice = 1059.78000 predictPrice = 1059.78951 diff = 8.97819E-4
Month = 10 targetPrice = 1139.45000 predictPrice = 1139.45977 diff = 8.51147E-4
Month = 11 targetPrice = 1148.08000 predictPrice = 1148.07912 diff = 7.66679E-5
Month = 12 targetPrice = 1130.20000 predictPrice = 1130.20593 diff = 5.24564E-4
Month = 13 targetPrice = 1106.73000 predictPrice = 1106.72654 diff = 3.12787E-4
Month = 14 targetPrice = 1147.39000 predictPrice = 1147.39283 diff = 2.46409E-4
Month = 15 targetPrice = 1076.92000 predictPrice = 1076.92461 diff = 4.28291E-4
Month = 16 targetPrice = 1067.14000 predictPrice = 1067.14948 diff = 8.88156E-4
Month = 17 targetPrice = 989.819999 predictPrice = 989.811316 diff = 8.77328E-4
Month = 18 targetPrice = 911.620000 predictPrice = 911.625389 diff = 5.91142E-4
Month = 19 targetPrice = 916.070000 predictPrice = 916.071216 diff = 1.32725E-4
Month = 20 targetPrice = 815.280000 predictPrice = 815.286704 diff = 8.22304E-4
Month = 21 targetPrice = 885.760000 predictPrice = 885.767730 diff = 8.72729E-4
Month = 22 targetPrice = 936.310000 predictPrice = 936.307290 diff = 2.89468E-4
Month = 23 targetPrice = 879.820000 predictPrice = 879.812595 diff = 8.41647E-4
Month = 24 targetPrice = 855.700000 predictPrice = 855.700307 diff = 3.58321E-5
Month = 25 targetPrice = 841.150000 predictPrice = 841.157407 diff = 8.80559E-4
Month = 26 targetPrice = 848.180000 predictPrice = 848.177279 diff = 3.22296E-4
Month = 27 targetPrice = 916.920000 predictPrice = 916.914394 diff = 6.11352E-4
Month = 28 targetPrice = 963.590000 predictPrice = 963.591678 diff = 1.74172E-4
Month = 29 targetPrice = 974.500000 predictPrice = 974.505665 diff = 5.81287E-4
Month = 30 targetPrice = 990.310000 predictPrice = 990.302895 diff = 7.17406E-4
Month = 31 targetPrice = 1008.01000 predictPrice = 1008.00861 diff = 1.37856E-4
Month = 32 targetPrice = 995.970000 predictPrice = 995.961734 diff = 8.29902E-4
Month = 33 targetPrice = 1050.71000 predictPrice = 1050.70954 diff = 4.42062E-5
Month = 34 targetPrice = 1058.20000 predictPrice = 1058.19690 diff = 2.93192E-4
Month = 35 targetPrice = 1111.92000 predictPrice = 1111.91406 diff = 5.34581E-4
Month = 36 targetPrice = 1131.13000 predictPrice = 1131.12351 diff = 5.73549E-4
Month = 37 targetPrice = 1144.94000 predictPrice = 1144.94240 diff = 2.09638E-4
Month = 38 targetPrice = 1126.21000 predictPrice = 1126.21747 diff = 6.63273E-4
Month = 39 targetPrice = 1107.30000 predictPrice = 1107.30139 diff = 1.25932E-4
Month = 40 targetPrice = 1120.68000 predictPrice = 1120.67926 diff = 6.62989E-5
Month = 41 targetPrice = 1140.84000 predictPrice = 1140.83145 diff = 7.49212E-4
Month = 42 targetPrice = 1101.72000 predictPrice = 1101.72597 diff = 5.42328E-4
Month = 43 targetPrice = 1104.24000 predictPrice = 1104.23914 diff = 7.77377E-5
Month = 44 targetPrice = 1114.58000 predictPrice = 1114.58307 diff = 2.75127E-4
Month = 45 targetPrice = 1130.20000 predictPrice = 1130.19238 diff = 6.74391E-4
Month = 46 targetPrice = 1173.82000 predictPrice = 1173.82891 diff = 7.58801E-4
Month = 47 targetPrice = 1211.92000 predictPrice = 1211.92000 diff = 4.97593E-7
Month = 48 targetPrice = 1181.27000 predictPrice = 1181.27454 diff = 3.84576E-4
Month = 49 targetPrice = 1203.60000 predictPrice = 1203.60934 diff = 7.75922E-4
Month = 50 targetPrice = 1180.59000 predictPrice = 1180.60006 diff = 8.51986E-4
Month = 51 targetPrice = 1156.85000 predictPrice = 1156.85795 diff = 6.87168E-4
Month = 52 targetPrice = 1191.50000 predictPrice = 1191.50082 diff = 6.89121E-5
Month = 53 targetPrice = 1191.32000 predictPrice = 1191.32780 diff = 1.84938E-4
Month = 54 targetPrice = 1234.18000 predictPrice = 1234.18141 diff = 1.14272E-4
Month = 55 targetPrice = 1220.33000 predictPrice = 1220.33276 diff = 2.26146E-4
Month = 56 targetPrice = 1228.81000 predictPrice = 1228.80612 diff = 3.15986E-4
Month = 57 targetPrice = 1207.01000 predictPrice = 1207.00419 diff = 4.81617E-4

```
Month = 58  targetPrice = 1249.48000  predictPrice = 1249.48941 diff = 7.52722E-4
Month = 59  targetPrice = 1248.29000  predictPrice = 1248.28153 diff = 6.78199E-4
Month = 60  targetPrice = 1280.08000  predictPrice = 1280.07984 diff = 1.22483E-5
Month = 61  targetPrice = 1280.66000  predictPrice = 1280.66951 diff = 7.42312E-4
Month = 62  targetPrice = 1294.87000  predictPrice = 1294.86026 diff = 7.51869E-4
Month = 63  targetPrice = 1310.61000  predictPrice = 1310.60544 diff = 3.48001E-4
Month = 64  targetPrice = 1270.09000  predictPrice = 1270.08691 diff = 2.43538E-4
Month = 65  targetPrice = 1270.20000  predictPrice = 1270.19896 diff = 8.21560E-5
Month = 66  targetPrice = 1276.66000  predictPrice = 1276.66042 diff = 3.26854E-5
Month = 67  targetPrice = 1303.82000  predictPrice = 1303.82874 diff = 6.70418E-4
Month = 68  targetPrice = 1335.85000  predictPrice = 1335.84632 diff = 2.75638E-4
Month = 69  targetPrice = 1377.94000  predictPrice = 1377.94691 diff = 5.01556E-4
Month = 70  targetPrice = 1400.63000  predictPrice = 1400.63379 diff = 2.70408E-4
Month = 71  targetPrice = 1418.30000  predictPrice = 1418.31183 diff = 8.34099E-4
Month = 72  targetPrice = 1438.24000  predictPrice = 1438.24710 diff = 4.93547E-4
Month = 73  targetPrice = 1406.82000  predictPrice = 1406.81500 diff = 3.56083E-4
Month = 74  targetPrice = 1420.86000  predictPrice = 1420.86304 diff = 2.13861E-4
Month = 75  targetPrice = 1482.37000  predictPrice = 1482.37807 diff = 5.44135E-4
Month = 76  targetPrice = 1530.62000  predictPrice = 1530.60780 diff = 7.96965E-4
Month = 77  targetPrice = 1503.35000  predictPrice = 1503.35969 diff = 6.44500E-4
Month = 78  targetPrice = 1455.27000  predictPrice = 1455.25870 diff = 7.77012E-4
Month = 79  targetPrice = 1473.99000  predictPrice = 1474.00301 diff = 8.82764E-4
Month = 80  targetPrice = 1526.75000  predictPrice = 1526.74507 diff = 3.23149E-4
Month = 81  targetPrice = 1549.38000  predictPrice = 1549.38480 diff = 3.10035E-4
Month = 82  targetPrice = 1481.14000  predictPrice = 1481.14819 diff = 5.52989E-4
Month = 83  targetPrice = 1468.36000  predictPrice = 1468.34730 diff = 8.64876E-4
Month = 84  targetPrice = 1378.55000  predictPrice = 1378.53761 diff = 8.98605E-4
Month = 85  targetPrice = 1330.63000  predictPrice = 1330.64177 diff = 8.84310E-4
Month = 86  targetPrice = 1322.70000  predictPrice = 1322.71089 diff = 8.23113E-4
Month = 87  targetPrice = 1385.59000  predictPrice = 1385.58259 diff = 5.34831E-4
Month = 88  targetPrice = 1400.38000  predictPrice = 1400.36749 diff = 8.93019E-4
Month = 89  targetPrice = 1279.99999  predictPrice = 1279.98926 diff = 8.38844E-4
Month = 90  targetPrice = 1267.38       predictPrice = 1267.39112 diff = 8.77235E-4
Month = 91  targetPrice = 1282.83000  predictPrice = 1282.82564 diff = 3.40160E-4
Month = 92  targetPrice = 1166.36000  predictPrice = 1166.35838 diff = 1.38537E-4
Month = 93  targetPrice = 968.750000  predictPrice = 968.756639 diff = 6.85325E-4
Month = 94  targetPrice = 896.24000    predictPrice = 896.236238 diff = 4.19700E-4
Month = 95  targetPrice = 903.250006  predictPrice = 903.250891 diff = 9.86647E-5
Month = 96  targetPrice = 825.880000  predictPrice = 825.877467 diff = 3.06702E-4
Month = 97  targetPrice = 735.090000  predictPrice = 735.089888 diff = 1.51705E-5
Month = 98  targetPrice = 797.870000  predictPrice = 797.864377 diff = 7.04777E-4
Month = 99  targetPrice = 872.810000  predictPrice = 872.817137 diff = 8.17698E-4
Month = 100 targetPrice = 919.14000   predictPrice = 919.144707 diff = 5.12104E-4
Month = 101 targetPrice = 919.32000   predictPrice = 919.311948 diff = 8.75905E-4
Month = 102 targetPrice = 987.48000   predictPrice = 987.485732 diff = 5.80499E-4
Month = 103 targetPrice = 1020.6200   predictPrice = 1020.62163 diff = 1.60605E-4
Month = 104 targetPrice = 1057.0800   predictPrice = 1057.07122 diff = 8.30374E-4
Month = 105 targetPrice = 1036.1900   predictPrice = 1036.18940 diff = 5.79388E-5
Month = 106 targetPrice = 1095.6300   predictPrice = 1095.63936 diff = 8.54512E-4
Month = 107 targetPrice = 1115.1000   predictPrice = 1115.09792 diff = 1.86440E-4
Month = 108 targetPrice = 1073.8700   predictPrice = 1073.87962 diff = 8.95733E-4
Month = 109 targetPrice = 1104.4900   predictPrice = 1104.48105 diff = 8.10355E-4
```

Month = 110 targetPrice = 1169.4300 predictPrice = 1169.42384 diff = 5.26459E-4
Month = 111 targetPrice = 1186.6900 predictPrice = 1186.68972 diff = 2.39657E-5
Month = 112 targetPrice = 1089.4100 predictPrice = 1089.40111 diff = 8.16044E-4
Month = 113 targetPrice = 1030.7100 predictPrice = 1030.71574 diff = 5.57237E-4
Month = 114 targetPrice = 1101.6000 predictPrice = 1101.59105 diff = 8.12503E-4
Month = 115 targetPrice = 1049.3300 predictPrice = 1049.32154 diff = 8.06520E-4
Month = 116 targetPrice = 1141.2000 predictPrice = 1141.20704 diff = 6.1701E-4
Month = 117 targetPrice = 1183.2600 predictPrice = 1183.27030 diff = 8.705E-4
Month = 118 targetPrice = 1180.5500 predictPrice = 1180.54438 diff = 4.763E-4
Month = 119 targetPrice = 1257.6400 predictPrice = 1257.63292 diff = 5.628E-4
Month = 120 targetPrice = 1286.1200 predictPrice = 1286.11021 diff = 7.608E-4
maxErrorPerc = 7.607871107092592E-4
averErrorPerc = 6.339892589243827E-6

Log menunjukkan bahwa karena penggunaan metode micro-batch, hasil aproksimasi untuk fungsi nonkontinyu ini cukup baik.
maxErrorDifferencePerc < 0,000761% dan averErrorDifferencePerc < 0,00000634%
Gambar 21.4 menunjukkan grafik hasil pelatihan/validasi.



Gambar 21.4 Bagan hasil pelatihan

21.6 MENGUJI KUMPULAN DATA

Kumpulan data pengujian memiliki format yang sama dengan kumpulan data pelatihan. Seperti disebutkan di awal contoh ini, tujuannya adalah untuk memprediksi harga pasar untuk bulan berikutnya, berdasarkan data historis sepuluh tahun. Oleh karena itu, kumpulan data pengujian sama dengan kumpulan data pelatihan, tetapi harus menyertakan di akhir satu catatan mikro-batch tambahan, yang akan digunakan untuk prediksi harga bulan depan (di luar rentang pelatihan jaringan). Tabel 21.5 menunjukkan fragmen kumpulan data pengujian perbedaan harga.

Tabel 21.5 Fragmen dari Kumpulan Data Pengujian Selisih Harga

priceDiffPerc	targetPriceDiffPerc	Date	inputPrice
5.840553677	5.857688372	199704	801.34
5.857688372	4.345263356	199705	848.28
4.345263356	7.814583004	199706	885.14
7.814583004	-5.746560342	199707	954.31
-5.746560342	5.315352374	199708	899.47
5.315352374	-3.447766236	199709	947.28
-3.447766236	4.458682294	199710	914.62
4.458682294	1.573163073	199711	955.4
1.573163073	1.015013963	199712	970.43
1.015013963	7.04492594	199801	980.28
7.04492594	4.994568014	199802	1049.34
4.994568014	0.907646925	199803	1101.75
0.907646925	-1.882617495	199804	1111.75
-1.882617495	3.943822079	199805	1090.82
3.943822079	-1.161539547	199806	1133.84
-1.161539547	-14.57967109	199807	1120.67
-14.57967109	6.239553736	199808	957.28
6.239553736	8.029419573	199809	1017.01
8.029419573	5.91260342	199810	1098.67
5.91260342	5.63753083	199811	1163.63
5.63753083	4.10094124	199812	1229.23
4.10094124	-3.228251696	199901	1279.64
-3.228251696	3.879418249	199902	1238.33
3.879418249	3.79439819	199903	1286.37
3.79439819	-2.497041597	199904	1335.18
-2.497041597	5.443833344	199905	1301.84
5.443833344	-3.204609859	199906	1372.71
-3.204609859	-0.625413932	199907	1328.72
-0.625413932	-2.855173772	199908	1320.41
-2.855173772	6.253946722	199909	1282.71

Tabel 21.6 menunjukkan sebuah fragmen dari kumpulan data pengujian yang dinormalisasi.

Tabel 21.6 Fragmen dari Kumpulan Data Pengujian yang Dinormalisasi

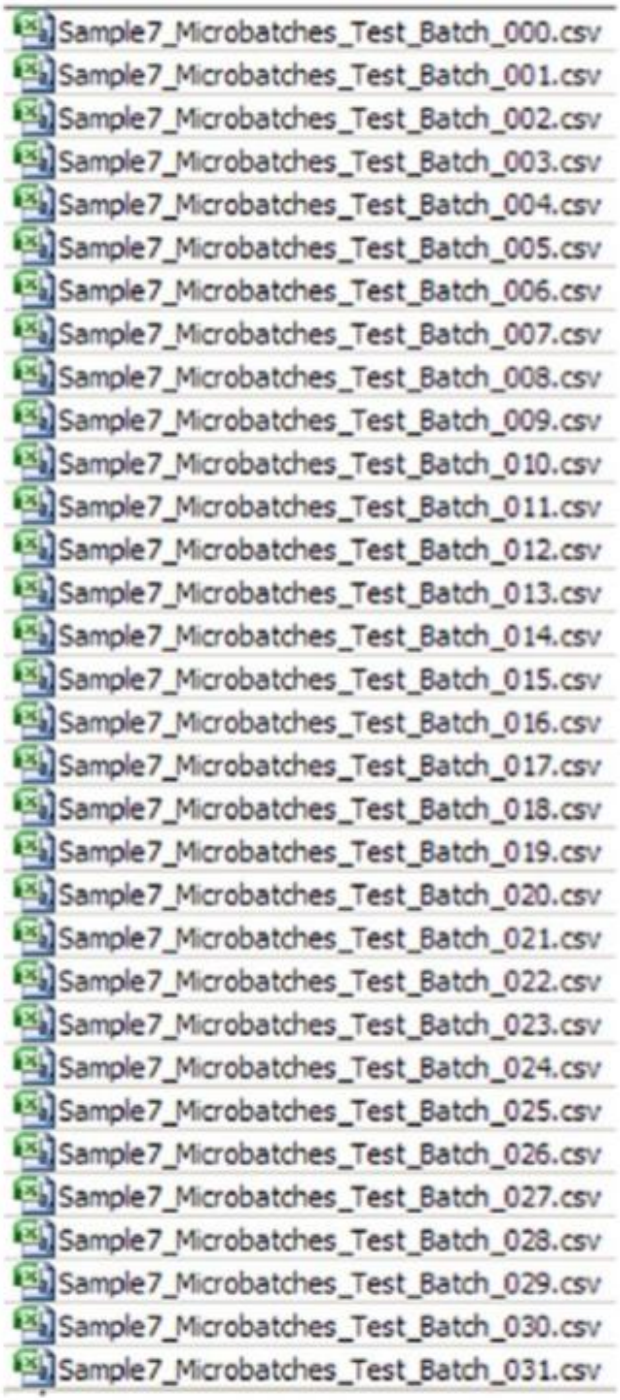
priceDiffPerc	targetPriceDiffPerc	Date	inputPrice
0.722703578	0.723845891	199704	801.34
0.723845891	0.623017557	199705	848.28
0.623017557	0.854305534	199706	885.14
0.854305534	-0.049770689	199707	954.31
-0.049770689	0.687690158	199708	899.47
0.687690158	0.103482251	199709	947.28
0.103482251	0.63057882	199710	914.62
0.63057882	0.438210872	199711	955.4
0.438210872	0.401000931	199712	970.43
0.401000931	0.802995063	199801	980.28
0.802995063	0.666304534	199802	1049.34
0.666304534	0.393843128	199803	1101.75
0.393843128	0.2078255	199804	1111.75
0.2078255	0.596254805	199805	1090.82
0.596254805	0.255897364	199806	1133.84
0.255897364	-0.638644739	199807	1120.67
-0.638644739	0.749303582	199808	957.28
0.749303582	0.868627972	199809	1017.01
0.868627972	0.727506895	199810	1098.67
0.727506895	0.709168722	199811	1163.63
0.709168722	0.606729416	199812	1229.23
0.606729416	0.118116554	199901	1279.64
0.118116554	0.591961217	199902	1238.33
0.591961217	0.586293213	199903	1286.37
0.586293213	0.166863894	199904	1335.18
0.166863894	0.696255556	199905	1301.84
0.696255556	0.119692676	199906	1372.71
0.119692676	0.291639071	199907	1328.72
0.291639071	0.142988415	199908	1320.41
0.142988415	0.750263115	199909	1282.71

Terakhir, Tabel 21.7 menunjukkan kumpulan data pengujian jendela geser. Ini adalah kumpulan data yang digunakan untuk menguji jaringan yang dilatih.

Tabel 21.7 Fragmen Kumpulan Data Pengujian Jendela Geser

Sliding Windows												
0.591	0.55	0.17	0.46	0.21	0.2	0.53	0.3	0.57	0.26	0.4	0.22	0.327
0.55	0.165	0.46	0.21	0.2	0.53	0.33	0.6	0.26	0.38	0.2	0.57	0.503
0.165	0.459	0.21	0.2	0.53	0.33	0.57	0.3	0.38	0.22	0.6	0.33	0.336
0.459	0.206	0.2	0.53	0.33	0.57	0.26	0.4	0.22	0.57	0.3	0.5	0.407
0.206	0.199	0.53	0.33	0.57	0.26	0.38	0.2	0.57	0.33	0.5	0.34	0.414
0.199	0.533	0.33	0.57	0.26	0.38	0.22	0.6	0.33	0.5	0.3	0.41	0.127
0.533	0.332	0.57	0.26	0.38	0.22	0.57	0.3	0.5	0.34	0.4	0.41	0.334
0.332	0.573	0.26	0.38	0.22	0.57	0.33	0.5	0.34	0.41	0.4	0.13	0.367
0.573	0.259	0.38	0.22	0.57	0.33	0.5	0.3	0.41	0.41	0.1	0.33	0.475
0.259	0.38	0.22	0.57	0.33	0.5	0.34	0.4	0.41	0.13	0.3	0.37	0.497
0.38	0.215	0.57	0.33	0.5	0.34	0.41	0.4	0.13	0.33	0.4	0.48	0.543
0.215	0.568	0.33	0.5	0.34	0.41	0.41	0.1	0.33	0.37	0.5	0.5	0.443
0.568	0.327	0.5	0.34	0.41	0.41	0.13	0.3	0.37	0.48	0.5	0.54	0.417
0.327	0.503	0.34	0.41	0.41	0.13	0.33	0.4	0.48	0.5	0.5	0.44	0.427
0.503	0.336	0.41	0.41	0.13	0.33	0.37	0.5	0.5	0.54	0.4	0.42	0.188
0.336	0.407	0.41	0.13	0.33	0.37	0.48	0.5	0.54	0.44	0.4	0.43	0.4
0.407	0.414	0.13	0.33	0.37	0.48	0.5	0.5	0.44	0.42	0.4	0.19	0.622
0.414	0.127	0.33	0.37	0.48	0.5	0.54	0.4	0.42	0.43	0.2	0.4	0.55
0.127	0.334	0.37	0.48	0.5	0.54	0.44	0.4	0.43	0.19	0.4	0.62	0.215
0.334	0.367	0.48	0.5	0.54	0.44	0.42	0.4	0.19	0.4	0.6	0.55	0.12
0.367	0.475	0.5	0.54	0.44	0.42	0.43	0.2	0.4	0.62	0.6	0.22	0.419
0.475	0.497	0.54	0.44	0.42	0.43	0.19	0.4	0.62	0.55	0.2	0.12	0.572
0.497	0.543	0.44	0.42	0.43	0.19	0.4	0.6	0.55	0.22	0.1	0.42	0.432
0.543	0.443	0.42	0.43	0.19	0.4	0.62	0.6	0.22	0.12	0.4	0.57	0.04
0.443	0.417	0.43	0.19	0.4	0.62	0.55	0.2	0.12	0.42	0.6	0.43	0.276
0.417	0.427	0.19	0.4	0.62	0.55	0.22	0.1	0.42	0.57	0.4	0.04	-0.074
0.427	0.188	0.4	0.62	0.55	0.22	0.12	0.4	0.57	0.43	0	0.28	0.102
0.188	0.4	0.62	0.55	0.22	0.12	0.42	0.6	0.43	0.04	0.3	-0.1	0.294
0.4	0.622	0.55	0.22	0.12	0.42	0.57	0.4	0.04	0.28	-0	0.1	0.65
0.622	0.55	0.22	0.12	0.42	0.57	0.43	0	0.28	-0.07	0.1	0.29	0.404

Kumpulan data pengujian jendela geser rusak dalam file mikro-batch. Gambar 21.5 menunjukkan fragmen daftar file pengujian mikro.



Gambar 21.5 Fragmen daftar pengujian set data mikro-batch

21.7 MENGUJI LOGIKA

Ada banyak fragmen pengkodean baru dalam metode ini, jadi mari kita bahas. Anda memuat kumpulan data micro-batch dan jaringan tersimpan yang sesuai dalam satu lingkaran di atas kumpulan pengujian kumpulan data micro-batch. Ingat, Anda tidak lagi memproses satu set data pengujian tetapi satu set set data pengujian mikro. Selanjutnya, Anda memperoleh nilai harga input, aktual, dan prediksi dari jaringan; normalkan mereka; dan menghitung harga aktual dan prediksi. Itu dilakukan untuk semua catatan pengujian yang ada catatan jaringan yang disimpan.

Namun, tidak ada file *save-network* untuk catatan *micro-batch* terakhir dalam kumpulan data uji, hanya karena jaringan tidak dilatih untuk titik itu. Untuk catatan ini,

Anda mengambil 12 bidang inputPriceDiffPerc, yang merupakan kunci yang digunakan selama pelatihan jaringan. Selanjutnya, Anda mencari kunci dari semua file jaringan tersimpan yang terletak di array memori yang disebut linkToSaveInputPriceDiffPerc_00, linkToSaveInputPriceDiffPerc_01, dan seterusnya.

Karena ada 12 kunci yang terkait dengan setiap jaringan yang disimpan, pencarian dilakukan dengan cara berikut. Untuk batch mikro yang akan diproses, Anda menghitung nilai vektor dalam ruang 12D menggunakan geometri Euclidean. Misalnya, untuk fungsi 12 variabel y = f(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12), nilai vektornya adalah akar kuadrat dari jumlah masing-masing nilai x diberdayakan ke 2 (lihat 10-1).
Persamaan 11.1

$$\sqrt{x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2 + x1^2}$$

Kemudian, untuk setiap set kunci jaringan yang disimpan dalam larik linkToSaveInputPriceDiffPerc, nilai vektor juga dihitung. Kunci jaringan yang sangat cocok dengan kumpulan kunci dari rekaman yang diproses dipilih dan dimuat ke dalam memori. Terakhir, Anda memperoleh dari jaringan itu nilai input, aktif, dan prediksi; denormalisasi mereka; dan menghitung nilai aktual dan prediksi. Daftar 21-6 menunjukkan kode untuk logika ini.

Daftar 21-6. Logika Memilih Rekaman Jaringan Tersimpan

```
static public void loadAndTestNetwork()
{
    List<Double> xData = new ArrayList<Double>();
    List<Double> yData1 = new ArrayList<Double>();
    List<Double> yData2 = new ArrayList<Double>();
    int k1 = 0;    int k3 = 0;
    BufferedReader br4;    BasicNetwork network;
    Try
    {
        // Process testing batches
        maxGlobalResultDiff = 0.00;
        averGlobalResultDiff = 0.00;
        sumGlobalResultDiff = 0.00;
        for (k1 = 0; k1 < intNumberOfBatchesToProcess; k1++)
        {
            br4 = new BufferedReader(new  FileReader(strTestingFile Names[k1]));
            tempLine = br4.readLine();
            // Skip the label record
            tempLine = br4.readLine();
            // Break the line using comma as separator
            tempWorkFields = tempLine.split(cvsSplitBy);
            recordNormInputPriceDiffPerc_00    =    Double.parseDouble(tempWork
            Fields[0]);
            recordNormInputPriceDiffPerc_01    =    Double.parseDouble(tempWork
            Fields[1]);
            recordNormInputPriceDiffPerc_02    =    Double.parseDouble(tempWork
            Fields[2]);
            recordNormInputPriceDiffPerc_03    =    Double.parseDouble(tempWork
            Fields[3]);
```

```
recordNormInputPriceDiffPerc_04 = Double.parseDouble(tempWork
Fields[4]);
recordNormInputPriceDiffPerc_05 = Double.parseDouble(tempWork
Fields[5]);
recordNormInputPriceDiffPerc_06 = Double.parseDouble(tempWork
Fields[6]);
recordNormInputPriceDiffPerc_07 = Double.parseDouble(tempWork
Fields[7]);
recordNormInputPriceDiffPerc_08 = Double.parseDouble(tempWork
Fields[8]);
recordNormInputPriceDiffPerc_09 = Double.parseDouble(tempWork
Fields[9]); recordNormInputPriceDiffPerc_10 =
Double.parseDouble(tempWork Fields[10]);
recordNormInputPriceDiffPerc_11 = Double.parseDouble(tempWork
Fields[11]);
recordNormTargetPriceDiffPerc = Double.parseDouble(tempWork
Fields[12]);
if(k1 < 120)
{
// Load the network for the current record network =
(BasicNetwork)EncogDirectoryPersistence. loadObject
(newFile(strSaveNetworkFileNames[k1]));
// Load the training file record MLDataSet testingSet =
loadCSV2Memory(strTestingFileNames[k1], intInputNeuronNumber,
intOutputNeuronNumber,true, CSVFormat.ENGLISH,false);
// Get the results from the loaded previously saved networks
int i = - 1;
// Index of the array to get results
for (MLDataPair pair:  testingSet)
{
        i++;
        MLData inputData = pair.getInput();
        MLData actualData = pair.getIdeal();
        MLData predictData = network.compute(inputData);
        // These values are Normalized as the whole input is
        normTargetPriceDiffPerc = actualData.getData(0);
        normPredictPriceDiffPerc = predictData.getData(0);
        normInputPriceDiffPercFromRecord = inputData.getData(11);
        // De-normalize this data to show the real result value
        denormTargetPriceDiffPerc = ((targetPriceDiffPercDl -
        targetPriceDiffPercDh)*normTargetPriceDiffPerc- Nh*target
        PriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
        denormPredictPriceDiffPerc =((targetPriceDiffPercDl -
        targetPriceDiffPercDh)*normPredictPriceDiffPerc - Nh*
        targetPriceDiffPercDl + targetPriceDiffPercDh*Nl)/(Nl - Nh);
                denormInputPriceDiffPercFromRecord = ((inputPriceDiff
        PercDl - inputPriceDiffPercDh)*normInputPriceDiffPerc
```

```
                    FromRecord - Nh*inputPriceDiffPercDl + inputPriceDiff
                    PercDh*Nl)/(Nl - Nh);
                    inputPriceFromFile = arrPrices[k1+12];
                    // Convert denormPredictPriceDiffPerc and denormTarget
                    PriceDiffPerc to real renormalized // price
                    realDenormTargetPrice = inputPriceFromFile +
                    inputPriceFromFile*(denormTargetPriceDiffPerc/100);
                    realDenormPredictPrice = inputPriceFromFile +
                    inputPriceFromFile*(denormPredictPriceDiffPerc/100);
                    realDenormTargetToPredictPricePerc = (Math.abs(realDenorm
                    TargetPrice - realDenormPredictPrice)/realDenorm
                    TargetPrice)*100;
                    System.out.println("Month = " + (k1+1) + " targetPrice = " +
                    realDenormTargetPrice + " predictPrice = " + real
                    DenormPredictPrice + " diff = " + realDenormTarget
                    ToPredictPricePerc);
            } // End of the for pair loop
    } // End for IF
else
{
    vectorForRecord =
    Math.sqrt( Math.pow(recordNormInputPriceDiffPerc_00,2) +
    Math.pow(recordNormInputPriceDiffPerc_01,2) +
    Math.pow(recordNormInputPriceDiffPerc_02,2) +
    Math.pow(recordNormInputPriceDiffPerc_03,2) +
    Math.pow(recordNormInputPriceDiffPerc_04,2) +
    Math.pow(recordNormInputPriceDiffPerc_05,2) +
    Math.pow(recordNormInputPriceDiffPerc_06,2) +
    Math.pow(recordNormInputPriceDiffPerc_07,2) +
    Math.pow(recordNormInputPriceDiffPerc_08,2) +
    Math.pow(recordNormInputPriceDiffPerc_09,2) +
    Math.pow(recordNormInputPriceDiffPerc_10,2) +
    Math.pow(recordNormInputPriceDiffPerc_11,2));
    //  Look for the network of previous months that closely match the
    // vectorForRecord value minVectorValue = 999.99;
    for (k3 = 0; k3 < intNumberOfSavedNetworks; k3++)
    {
            r_00 = linkToSaveInputPriceDiffPerc_00[k3];
            r_01 = linkToSaveInputPriceDiffPerc_01[k3];
            r_02 = linkToSaveInputPriceDiffPerc_02[k3];
            r_03 = linkToSaveInputPriceDiffPerc_03[k3];
            r_04 = linkToSaveInputPriceDiffPerc_04[k3];
            r_05 = linkToSaveInputPriceDiffPerc_05[k3];
            r_06 = linkToSaveInputPriceDiffPerc_06[k3];
            r_07 = linkToSaveInputPriceDiffPerc_07[k3];
            r_08 = linkToSaveInputPriceDiffPerc_08[k3];
            r_09 = linkToSaveInputPriceDiffPerc_09[k3];
            r_10 = linkToSaveInputPriceDiffPerc_10[k3];
```

```
                r_11 = linkToSaveInputPriceDiffPerc_11[k3];
                r2 = linkToSaveTargetPriceDiffPerc[k3];
                vectorForNetworkRecord =
                Math.sqrt(
                Math.pow(r_00,2) +
                Math.pow(r_01,2) +
                Math.pow(r_02,2) +
                Math.pow(r_03,2) +
                Math.pow(r_04,2) +
                Math.pow(r_05,2) +
                Math.pow(r_06,2) +
                Math.pow(r_07,2) +
                Math.pow(r_08,2) +
                Math.pow(r_09,2) +
                Math.pow(r_10,2) +
                Math.pow(r_11,2));
                vectorDiff = Math.abs(vectorForRecord - vectorFor NetworkRecord);
                if(vectorDiff < minVectorValue)
                {
                minVectorValue = vectorDiff;
                // Save this network record attributes
                        rTempKey = r_00;
                        rTempPriceDiffPerc = r2;
                        tempMinIndex = k3;
                }
} // End  FOR k3 loop
network =
(BasicNetwork)EncogDirectoryPersistence.loadObject(newFile
(strSaveNetworkFileNames[tempMinIndex]));
// Now, tempMinIndex points to the corresponding saved network
// Load this network in memory
MLDataSet testingSet = loadCSV2Memory(strTestingFileNames[k1],
intInputNeuronNumber,intOutputNeuronNumber,true,
CSVFormat.ENGLISH,false);
// Get the results from the loaded network          int i = - 1;
for (MLDataPair pair:  testingSet)
{
    i++;
    MLData inputData = pair.getInput();
    MLData actualData = pair.getIdeal();
    MLData predictData = network.compute(inputData);
    // These values are Normalized as the whole input is
    normTargetPriceDiffPerc = actualData.getData(0);
    normPredictPriceDiffPerc = predictData.getData(0);
    normInputPriceDiffPercFromRecord = inputData.getData(11);
    // Renormalize this data to show the real result value
    denormTargetPriceDiffPerc = ((targetPriceDiffPercDl
```

```
        targetPriceDiffPercDh)*normTargetPriceDiffPerc - Nh*target PriceDiffPercDl
        + targetPriceDiffPercDh*Nl)/(Nl - Nh);
        denormPredictPriceDiffPerc =((targetPriceDiffPercDl - targetPriceDiffPercDh)*
        normPredictPriceDiffPerc - Nh* targetPriceDiffPercDl +
        targetPriceDiffPercDh*Nl)/(Nl - Nh);
        denormInputPriceDiffPercFromRecord = ((inputPriceDiffPercDl -
        inputPriceDiffPercDh)*normInputPriceDiffPercFromRecord -
        Nh*inputPriceDiffPercDl + inputPriceDiffPercDh*Nl)/(Nl - Nh);
        inputPriceFromFile = arrPrices[k1+12];
        //  Convert denormPredictPriceDiffPerc and denormTarget PriceDiffPerc to a real
        //renormalized price realDenormTargetPrice = inputPriceFromFile +
        inputPrice FromFile*(denormTargetPriceDiffPerc/100);
        realDenormPredictPrice = inputPriceFromFile + inputPrice
        FromFile*(denormPredictPriceDiffPerc/100);
        realDenormTargetToPredictPricePerc = (Math.abs(realDenorm TargetPrice -
        realDenormPredictPrice)/realDenorm TargetPrice)*100;
        System.out.println("Month = " + (k1+1) + " targetPrice = " + realDenormTargetPrice
        + " predictPrice = " + realDenorm PredictPrice + " diff = " +
        realDenormTargetToPredict PricePerc);
        if (realDenormTargetToPredictPricePerc > maxGlobal ResultDiff)
        maxGlobalResultDiff = realDenormTargetToPredict PricePerc;
        sumGlobalResultDiff = sumGlobalResultDiff + realDenorm
        TargetToPredictPricePerc;
    } // End of IF
} // End for the pair loop
        // Populate chart elements
        tempMonth = (double) k1+14;
        xData.add(tempMonth);
        yData1.add(realDenormTargetPrice);
        yData2.add(realDenormPredictPrice);
    }  // End of loop K1
    // Print the max and average results
    System.out.println(" ");
    System.out.println(" ");
    System.out.println("Results of processing testing batches");
    averGlobalResultDiff = sumGlobalResultDiff/intNumberOfBatches ToProcess;
    System.out.println("maxGlobalResultDiff = " + maxGlobalResultDiff + "  i = " +
    maxGlobalIndex);
    System.out.println("averGlobalResultDiff = " + averGlobalResult Diff);
    System.out.println(" ");
    System.out.println(" ");
    }
    // End of TRY    catch (IOException e1)
    {
    e1.printStackTrace();
    }
    // All testing batch files have been processed
    XYSeries series1 = Chart.addSeries("Actual Price", xData, yData1);
```

```
    XYSeries series2 = Chart.addSeries("Forecasted Price", xData, yData2);
    series1.setLineColor(XChartSeriesColors.BLUE);
    series2.setMarkerColor(Color.ORANGE);
    series1.setLineStyle(SeriesLines.SOLID);
    series2.setLineStyle(SeriesLines.SOLID);
    // Save the chart image
    try
    {
    BitmapEncoder.saveBitmapWithDPI(Chart,
    strChartFileName, BitmapFormat.JPG, 100);
    }
    catch (Exception bt)
    {
    bt.printStackTrace();
    }
    System.out.println ("The Chart has been saved");
} // End of the method
```

21.8 HASIL PENGUJIAN

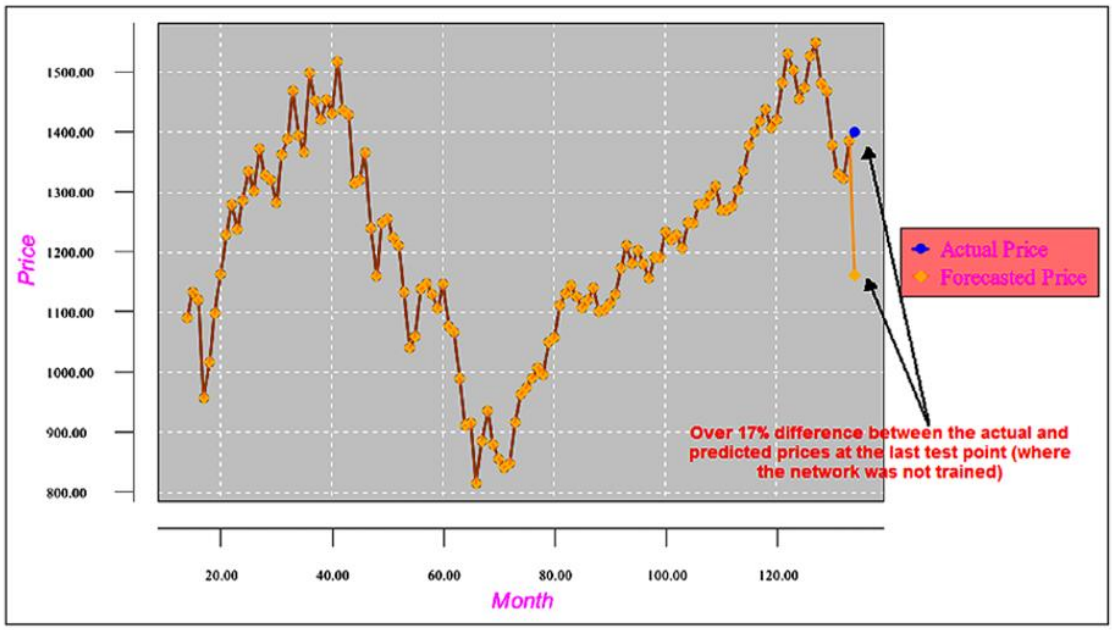
Daftar 21-7 menunjukkan log fragmen dari hasil pengujian.

Daftar 21-7. Hasil Pengujian

```
Month =  80  targetPrice = 1211.91999   predictPrice = 1211.91169 diff = 6.84919E-4
Month =  81  targetPrice = 1181.26999   predictPrice = 1181.26737 diff = 2.22043E-4
Month =  82  targetPrice = 1203.60000   predictPrice = 1203.60487 diff = 4.05172E-4
Month =  83  targetPrice = 1180.59000   predictPrice = 1180.59119 diff = 1.01641E-4
Month =  84  targetPrice = 1156.84999   predictPrice = 1156.84136 diff = 7.46683E-4
Month =  85  targetPrice = 1191.49999   predictPrice = 1191.49043 diff = 8.02666E-4
Month =  86  targetPrice = 1191.32999   predictPrice = 1191.31947 diff = 8.83502E-4
Month =  87  targetPrice = 1234.17999   predictPrice = 1234.17993 diff = 5.48814E-6
Month =  88  targetPrice = 1220.33000   predictPrice = 1220.31947 diff = 8.62680E-4
Month =  89  targetPrice = 1228.80999   predictPrice = 1228.82099 diff = 8.95176E-4
Month =  90  targetPrice = 1207.00999   predictPrice = 1207.00976 diff = 1.92764E-5
Month =  91  targetPrice = 1249.48000   predictPrice = 1249.48435 diff = 3.48523E-4
Month =  92  targetPrice = 1248.28999   predictPrice = 1248.27937 diff = 8.51313E-4
Month =  93  targetPrice = 1280.08000   predictPrice = 1280.08774 diff = 6.05221E-4
Month =  94  targetPrice = 1280.66000   predictPrice = 1280.66295 diff = 2.30633E-4
Month =  95  targetPrice = 1294.86999   predictPrice = 1294.85904 diff = 8.46250E-4
Month =  96  targetPrice = 1310.60999   predictPrice = 1310.61570 diff = 4.35072E-4
Month =  97  targetPrice = 1270.08999   predictPrice = 1270.08943 diff = 4.41920E-5
Month =  98  targetPrice = 1270.19999   predictPrice = 1270.21071 diff = 8.43473E-4
Month =  99  targetPrice = 1276.65999   predictPrice = 1276.65263 diff = 5.77178E-4
Month = 100  targetPrice = 1303.81999   predictPrice = 1303.82201 diff = 1.54506E-4
Month = 101  targetPrice = 1335.85000   predictPrice = 1335.83897 diff = 8.25569E-4
Month = 102  targetPrice = 1377.93999   predictPrice = 1377.94590 diff = 4.28478E-4
Month = 103  targetPrice = 1400.63000   predictPrice = 1400.62758 diff = 1.72417E-4
Month = 104  targetPrice = 1418.29999   predictPrice = 1418.31083 diff = 7.63732E-4
Month = 105  targetPrice = 1438.23999   predictPrice = 1438.23562 diff = 3.04495E-4
Month = 106  targetPrice = 1406.82000   predictPrice = 1406.83156 diff = 8.21893E-4
Month = 107  targetPrice = 1420.85999   predictPrice = 1420.86256 diff = 1.80566E-4
Month = 108  targetPrice = 1482.36999   predictPrice = 1482.35896 diff = 7.44717E-4
```

Month = 109 targetPrice = 1530.62000 predictPrice = 1530.62213 diff = 1.39221E-4
Month = 110 targetPrice = 1503.34999 predictPrice = 1503.33884 diff = 7.42204E-4
Month = 111 targetPrice = 1455.27000 predictPrice = 1455.27626 diff = 4.30791E-4
Month = 112 targetPrice = 1473.98999 predictPrice = 1473.97685 diff = 8.91560E-4
Month = 113 targetPrice = 1526.75000 predictPrice = 1526.76231 diff = 8.06578E-4
Month = 114 targetPrice = 1549.37999 predictPrice = 1549.39017 diff = 6.56917E-4
Month = 115 targetPrice = 1481.14000 predictPrice = 1481.15076 diff = 7.27101E-4
Month = 116 targetPrice = 1468.35999 predictPrice = 1468.35702 diff = 2.02886E-4
Month = 117 targetPrice = 1378.54999 predictPrice = 1378.55999 diff = 7.24775E-4
Month = 118 targetPrice = 1330.63000 predictPrice = 1330.61965 diff = 7.77501E-4
Month = 119 targetPrice = 1322.70000 predictPrice = 1322.69947 diff = 3.99053E-5
Month = 120 targetPrice = 1385.58999 predictPrice = 1385.60045 diff = 7.54811E-4
Month = 121 targetPrice = 1400.38000 predictPrice = 1162.09439 diff = 17.0157E-4
maxErrorPerc = 17.0157819794876
averErrorPerc = 0.14062629735113719

Gambar 21.6 menunjukkan grafik hasil pengujian.



Gambar 21.6 Bagan hasil pengujian

21.9 MENGANALISIS HASIL PENGUJIAN

Di semua titik di mana jaringan dilatih, harga yang diprediksi sangat cocok dengan harga sebenarnya (grafik kuning dan biru praktis tumpang tindih). Namun, pada titik bulan berikutnya (titik di mana jaringan tidak dilatih), harga yang diprediksi berbeda dari harga sebenarnya (yang kebetulan Anda ketahui) lebih dari 17 persen. Bahkan arah prediksi harga bulan depan (selisih dari bulan sebelumnya) salah. Harga sebenarnya sedikit meningkat, sementara harga yang diprediksi turun drastis.

Dengan harga pada titik-titik ini berada di sekitar 1200 hingga 1300, perbedaan 17 persen mewakili kesalahan lebih dari 200 poin. Ini tidak bisa dianggap sebagai prediksi sama sekali; hasilnya tidak berguna bagi trader/investor. Jadi, apa yang salah? Kami tidak melanggar batasan memprediksi nilai fungsi di luar rentang pelatihan (dengan mengubah fungsi harga menjadi bergantung pada perbedaan harga antar bulan, bukan bulan berurutan). Untuk menjawab pertanyaan ini, mari kita selidiki masalahnya.

Saat Anda memproses rekaman pengujian terakhir, Anda memperoleh nilai 12 bidang pertamanya dari 12 rekaman asli sebelumnya. Mereka mewakili persentase perbedaan harga antara bulan-bulan saat ini dan sebelumnya. Dan bidang terakhir dalam catatan adalah persen

Gambar perbedaan antara nilai harga pada bulan berikutnya (catatan 13) dan nilai harga pada bulan 12. Dengan semua bidang dinormalisasi, catatan tersebut ditunjukkan pada Persamaan 21-2.

Persamaan 21.2

0.621937887 0.550328191 0.214557935 0.12012062 0.419090615 0.571960009 0.4321489 0.039710508 0.275810074 -0.074423166 0.101592253 0.29360278 0.404494355

Dengan mengetahui harga untuk micro-batch record 12 (yaitu 1385,95) dan mendapatkan prediksi jaringan sebagai bidang targetPriceDiffPerc (yaitu persentase perbedaan antara harga bulan depan dan harga bulan ini), Anda dapat menghitung prediksi harga bulan berikutnya seperti yang ditunjukkan dalam Persamaan 21-3.

Persamaan 21.3

nextMonthPredictedPrice = record ActualPrice + record ActualPrice*predictedPriceDiffPerc /100

Untuk mendapatkan prediksi jaringan untuk catatan 13 (predictedPriceDiffPerc), Anda memberi makan jaringan terlatih nilai vektor dari 12 bidang inputPriceDiffPerc dari catatan yang sedang diproses (lihat 10-2). Jaringan mengembalikan -16.129995719. Menyatukan semuanya, Anda menerima prediksi harga untuk bulan berikutnya.

1385.59 - 1385.59 * 16.12999/100.00 = 1, 162 0943923170353

Untuk mendapatkan prediksi jaringan untuk catatan 13 (predictedPriceDiffPerc), Anda memberi makan jaringan terlatih nilai vektor dari 12 bidang inputPriceDiffPerc dari catatan yang sedang diproses (lihat 10-2). Jaringan mengembalikan -16.129995719. Menyatukan semuanya, Anda menerima prediksi harga untuk bulan berikutnya.

Bulan = 121 targetPrice = 1400.3800000016674

predictPrice = 1162.0943923170353 diff = 17.0157819794876

Hasil yang dihitung untuk harga bulan berikutnya secara matematis benar dan didasarkan pada jumlah harga pada titik pelatihan terakhir dan persentase perbedaan harga antara titik berikutnya dan saat ini yang dikembalikan oleh jaringan.

Masalahnya adalah bahwa harga pasar saham historis tidak berulang dalam kondisi yang sama atau serupa. Persentase selisih harga yang dikembalikan jaringan untuk vektor terhitung (10.1) dari catatan yang diproses terakhir tidak benar untuk menghitung

Bab memprediksi harga untuk bulan depan. Ini adalah masalah dengan model yang Anda gunakan dalam contoh ini, yang mengasumsikan bahwa persentase perbedaan harga untuk bulan mendatang serupa dengan persentase perbedaan harga yang dicatat untuk kondisi yang sama atau penutupan di masa lalu.

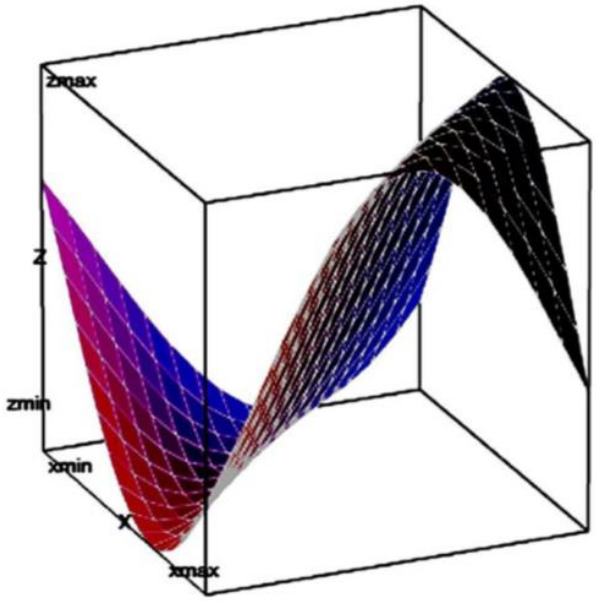
Ini adalah pelajaran penting untuk dipelajari. Jika modelnya salah, tidak ada yang akan berhasil. Sebelum melakukan pengembangan jaringan saraf, Anda perlu membuktikan bahwa model yang dipilih berfungsi dengan benar. Ini akan menghemat banyak waktu dan tenaga. Beberapa proses bersifat acak dan tidak dapat diprediksi menurut definisi. Jika pasar saham dapat diprediksi, itu akan hilang begitu saja karena premisnya didasarkan pada perbedaan pendapat. Jika semua orang tahu arah pasar masa depan, semua investor akan menjual, dan tidak ada yang akan membeli.

21.10 RINGKASAN

Bab ini menjelaskan pentingnya memilih model kerja yang benar untuk proyek tersebut. Anda harus membuktikan bahwa model berfungsi dengan benar untuk proyek Anda sebelum memulai pengembangan apa pun. Kegagalan untuk memilih model yang benar akan meninggalkan Anda dengan aplikasi yang salah bekerja. Jaringan juga akan menghasilkan hasil yang salah ketika digunakan untuk memprediksi hasil dari semua jenis permainan (judi, olahraga, dan sebagainya).

BAB 22
PERKIRAAN FUNGSI DALAM RUANG 3D

Bab ini membahas cara memperkirakan fungsi dalam ruang 3D. Nilai fungsi tersebut bergantung pada dua variabel (bukan satu variabel, yang telah dibahas dalam bab sebelumnya). Semua yang dibahas dalam bab ini juga benar untuk fungsi yang bergantung pada lebih dari dua variabel. Gambar 22.1 menunjukkan bagan fungsi 3D yang dibahas dalam bab ini.



Gambar 22.1 Bagan fungsi dalam ruang 3D

22.1 CONTOH 8: PERKIRAAN FUNGSI DALAM RUANG 3D

Rumus fungsinya adalah z(x, y) = 50,00 + sin(x*y), tetapi sekali lagi, mari kita anggap bahwa rumus fungsi tidak diketahui dan bahwa fungsi tersebut diberikan kepada Anda oleh nilainya pada titik-titik tertentu.

22.2 PERSIAPAN DATA

Nilai fungsi diberikan pada interval [3.00, 4.00] dengan nilai kenaikan 0,02 untuk kedua argumen fungsi x dan y. Titik awal untuk kumpulan data pelatihan adalah 3,00, dan titik awal untuk kumpulan data pengujian adalah 3,01. Kenaikan nilai x dan y adalah 0,02. Catatan kumpulan data pelatihan terdiri dari tiga bidang. Struktur record set data pelatihan adalah sebagai berikut:

Field/Bidang 1: Nilai argumen x
Field/Bidang 2: Nilai argumen y
Field/Bidang 3: Nilai fungsi

Tabel 22.1 menunjukkan sebuah fragmen dari kumpulan data pelatihan. Kumpulan data pelatihan mencakup catatan untuk semua kemungkinan kombinasi nilai x dan y.

Tabel 22.1 Fragmen dari Kumpulan Data Pelatihan

x	y	z
3	3	50.41211849
3	3.02	50.35674187
3	3.04	50.30008138
3	3.06	50.24234091
3	3.08	50.18372828
3	3.1	50.12445442
3	3.12	50.06473267
3	3.14	50.00477794
3	3.16	49.94480602
3	3.18	49.88503274
3	3.2	49.82567322
3	3.22	49.76694108
3	3.24	49.70904771
3	3.26	49.65220145
3	3.28	49.59660689
3	3.4	49.54246411
3	3.42	49.48996796
3	3.44	49.43930738
3	3.46	49.39066468
3	3.48	49.34421494
3	3.5	49.30012531
3	3.52	49.25855448
3	3.54	49.21965205
3	3.56	49.18355803
3	3.58	49.15040232
3	3.8	49.12030424
3	3.62	49.09337212
3	3.64	49.06970288
3	3.66	49.0493817
3	3.68	49.03248173
3	3.7	49.01906377
3	3.72	49.00917613
3	3.74	49.00285438
3	3.76	49.00012128
3	3.78	49.00098666
3	3.8	49.00544741
3	3.82	49.01348748
3	3.84	49.02507793
3	3.86	49.04017704
3	3.88	49.05873048
3	4	49.08067147
3.02	3	49.10592106
3.02	3.02	49.13438836
3.02	3.04	49.16597093
3.02	3.06	49.2005551
3.02	3.08	49.23801642

3.02	3.1	49.27822005
3.02	3.12	49.3210213
3.02	3.14	49.36626615
3.02	3.16	49.41379176
3.02	3.18	49.46342708
3.02	3.2	50.35674187
		50.29969979
		50.24156468
		50.18254857
		50.1228667
		50.06273673
		50.00237796

Tabel 22.2 menunjukkan fragmen dari kumpulan data pengujian. Ini memiliki struktur yang sama, tetapi mencakup titik x dan y yang tidak digunakan untuk pelatihan jaringan. Tabel 22.2 menunjukkan fragmen kumpulan data pengujian.

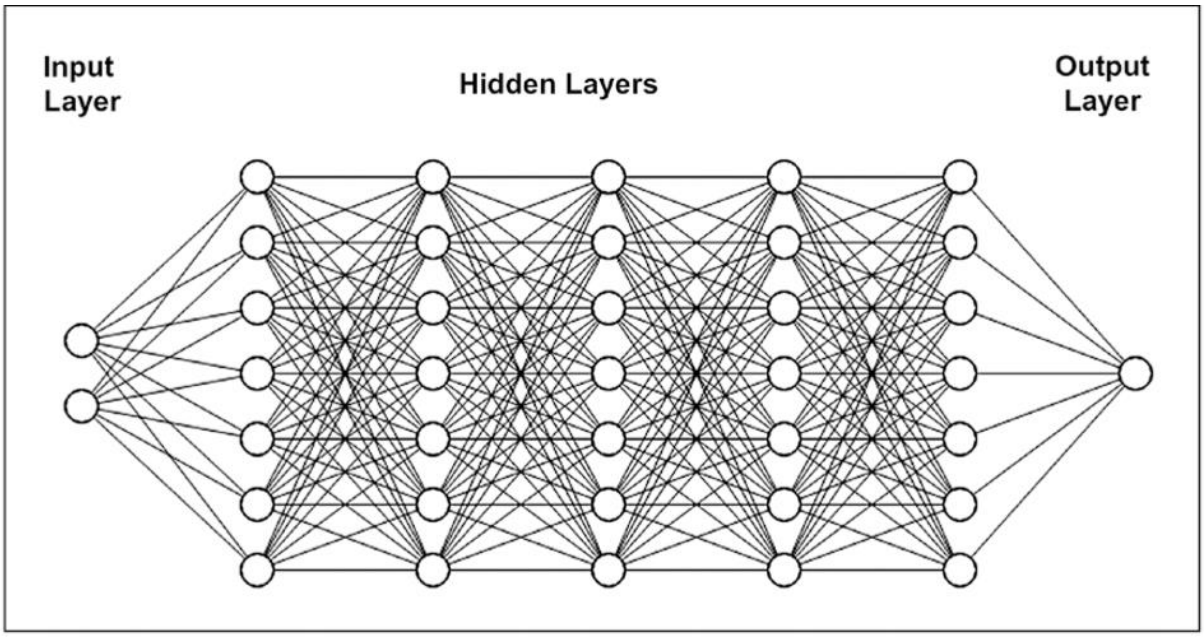
Tabel 22.2 Fragmen Kumpulan Data Pengujian

x	y	z
3.01	3.01	50.18304015
3.01	3.03	50.12356137
3.01	3.05	50.06363494
3.01	3.07	50.00347795
3.01	3.09	49.94330837
3.01	3.11	49.88334418
3.01	3.13	49.82380263
3.01	3.15	49.76489943
3.01	3.17	49.70684798
3.01	3.19	49.64985862
3.01	3.21	49.59413779
3.01	3.23	49.43657796
3.01	3.25	49.38789323
3.01	3.27	49.34142613
3.01	3.29	49.29734501
3.01	3.41	49.25580956
3.01	3.43	49.21697029
3.01	3.45	49.18096788
3.01	3.47	49.14793278
3.01	3.49	49.11798468
3.01	3.51	49.09123207
3.01	3.53	49.06777188
3.01	3.55	49.04768909
3.01	3.57	49.03105648
3.01	3.59	49.0179343
3.01	3.61	49.00837009
3.01	3.63	49.0023985
3.01	3.65	349.00004117
3.01	3.67	49.00130663

3.01	3.69	49.00619031
3.01	3.71	49.0146745
3.01	3.73	49.02672848
3.01	3.75	49.04230856
3.01	3.77	49.06135831
3.01	3.79	49.08380871
3.01	3.81	49.10957841
3.01	3.83	49.13857407
3.01	3.85	49.17069063
3.01	3.87	49.20581173
3.01	3.89	49.24381013
3.01	3.91	49.28454816
3.01	3.93	49.32787824
3.01	3.95	49.37364338
3.01	3.97	49.42167777
3.01	3.99	49.47180739
3.03	3.01	50.29979519
3.03	3.03	50.24146764
3.03	3.05	50.18225361

22.3 ARSITEKTUR JARINGAN

Gambar 22.2 menunjukkan arsitektur jaringan. Fungsi yang akan Anda proses memiliki dua input (x dan y); oleh karena itu, arsitektur jaringan memiliki dua input.



Gambar 22.2 Arsitektur jaringan

Baik set data pelatihan dan pengujian dinormalisasi sebelum diproses. Anda akan memperkirakan fungsi menggunakan proses jaringan konvensional. Berdasarkan hasil pemrosesan, Anda kemudian akan memutuskan apakah Anda perlu menggunakan metode mikro-batch.

Kode Program

Daftar 22-1 menunjukkan kode program.

Daftar 22-1. Kode Program

```
// ============================================================
// Approximation of the 3-D Function using conventional
// process.
// Input file di normalisasi.
// ============================================================
package sample9;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.*;
import java.util.Properties;
import java.time.YearMonth;
import java.awt.Color;
import java.awt.Font;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Properties;
import org.encog.Encog;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.engine.network.activation.ActivationReLU;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataPair;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.buffer.MemoryDataLoader;
import org.encog.ml.data.buffer.codec.CSVDataCODEC;
import org.encog.ml.data.buffer.codec.DataSetCODEC;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.resilient.
```

```
ResilientPropagation;
import org.encog.persist.EncogDirectoryPersistence;
import org.encog.util.csv.CSVFormat;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYChartBuilder;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.demo.charts.ExampleChart;
import org.knowm.xchart.style.Styler.LegendPosition;
import org.knowm.xchart.style.colors.ChartColor;
import org.knowm.xchart.style.colors.XChartSeriesColors;
import org.knowm.xchart.style.lines.SeriesLines;
import org.knowm.xchart.style.markers.SeriesMarkers;
import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
public class Sample9 implements ExampleChart<XYChart>
{
// Interval to normalize
static double Nh = 1;
static double Nl = -1;
// First column
static double minXPointDl = 2.00;
static double maxXPointDh = 6.00;
// Second column
static double minYPointDl = 2.00;
static double maxYPointDh = 6.00;
// Third column - target data
static double minTargetValueDl = 45.00;
static double maxTargetValueDh = 55.00;
static double doublePointNumber = 0.00;
static int intPointNumber = 0;
static InputStream input = null;
static double[] arrPrices = new double[2700];
static double normInputXPointValue = 0.00;
static double normInputYPointValue = 0.00;
static double normPredictValue = 0.00;
static double normTargetValue = 0.00;
static double normDifferencePerc = 0.00;
static double returnCode = 0.00;
static double denormInputXPointValue = 0.00;
static double denormInputYPointValue = 0.00;
static double denormPredictValue = 0.00;
static double denormTargetValue = 0.00;
static double valueDifference = 0.00;
static int numberOfInputNeurons;
static int numberOfOutputNeurons;
```

```
static int intNumberOfRecordsInTestFile;
static String trainFileName;
static String priceFileName;
static String testFileName;
static String chartTrainFileName;
static String chartTrainFileNameY;
static String chartTestFileName;
static String networkFileName;
static int workingMode;
static String cvsSplitBy = ",";
static int numberOfInputRecords = 0;
static List<Double> xData = new ArrayList<Double>();
static List<Double> yData1 = new ArrayList<Double>();
static List<Double> yData2 = new ArrayList<Double>();
static XYChart Chart;
@Override
public XYChart getChart()
{
// Create Chart
Chart = new XYChartBuilder().width(900).height(500).title(getClass().
getSimpleName()).xAxisTitle("x").yAxisTitle("y= f(x)").build();
// Customize Chart
//Chart = new XYChartBuilder().width(900).height(500).title(getClass().
// getSimpleName()).xAxisTitle("y").yAxisTitle("z= f(y)").build();
//Chart = new XYChartBuilder().width(900).height(500).title(getClass().
// getSimpleName()).xAxisTitle("y").yAxisTitle("z= f(y)").build();
// Customize Chart
Chart.getStyler().setPlotBackgroundColor(ChartColor.
getAWTColor(ChartColor.GREY));
Chart.getStyler().setPlotGridLinesColor(new Color(255, 255, 255));
//
Chart.getStyler().setPlotBackgroundColor(ChartColor.
getAWTColor(ChartColor.WHITE));
//Chart.getStyler().setPlotGridLinesColor(new Color(0, 0, 0));
Chart.getStyler().setChartBackgroundColor(Color.WHITE);
//Chart.getStyler().setLegendBackgroundColor(Color.PINK);
Chart.getStyler().setLegendBackgroundColor(Color.WHITE);
//Chart.getStyler().setChartFontColor(Color.MAGENTA);
Chart.getStyler().setChartFontColor(Color.BLACK);
Chart.getStyler().setChartTitleBoxBackgroundColor(new Color(0, 222, 0));
Chart.getStyler().setChartTitleBoxVisible(true);
Chart.getStyler().setChartTitleBoxBorderColor(Color.BLACK);
Chart.getStyler().setPlotGridLinesVisible(true);
Chart.getStyler().setAxisTickPadding(20);
Chart.getStyler().setAxisTickMarkLength(15);
Chart.getStyler().setPlotMargin(20);
Chart.getStyler().setChartTitleVisible(false);
Chart.getStyler().setChartTitleFont(new Font(Font.MONOSPACED, Font.
```

```
BOLD, 24));
Chart.getStyler().setLegendFont(new Font(Font.SERIF, Font.PLAIN, 18));
Chart.getStyler().setLegendPosition(LegendPosition.OutsideS);
Chart.getStyler().setLegendSeriesLineLength(12);
Chart.getStyler().setAxisTitleFont(new Font(Font.SANS_SERIF, Font.
ITALIC, 18));
Chart.getStyler().setAxisTickLabelsFont(new Font(Font.SERIF, Font.
PLAIN, 11));
Chart.getStyler().setDatePattern("yyyy-MM");
Chart.getStyler().setDecimalPattern("#0.00");
try
{
// Common part of config data
networkFileName =
"C:/My_Neural_Network_Book/Book_Examples/Sample9_Saved_Network_
File.csv";
numberOfInputNeurons = 2;
numberOfOutputNeurons = 1;
if(workingMode == 1)
{
// Training mode
numberOfInputRecords = 2602;
trainFileName = "C:/My_Neural_Network_Book/Book_Examples/
Sample9_Calculate_Train_Norm.csv";
chartTrainFileName = "C:/My_Neural_Network_Book/Book_Examples/
Sample9_Chart_X_Training_Results.csv";
chartTrainFileName = "C:/My_Neural_Network_Book/Book_Examples/
Sample9_Chart_Y_Training_Results.csv";
File file1 = new File(chartTrainFileName);
File file2 = new File(networkFileName);
if(file1.exists())
file1.delete();
if(file2.exists())
file2.delete();
returnCode = 0; // Clear the error Code
do
{
returnCode = trainValidateSaveNetwork();
} while (returnCode > 0);
}
else
{
// Testing mode
numberOfInputRecords = 2602;
testFileName = "C:/My_Neural_Network_Book/Book_Examples/
Sample9_Calculate_Test_Norm.csv";
chartTestFileName = "C:/My_Neural_Network_Book/Book_Examples/
Sample9_Chart_X_Testing_Results.csv";
```

```
chartTestFileName = "C:/My_Neural_Network_Book/Book_Examples/
Sample9_Chart_Y_Testing_Results.csv";
loadAndTestNetwork();
}
}
catch (Throwable t)
{
t.printStackTrace();
System.exit(1);
}
finally
{
Encog.getInstance().shutdown();
}
Encog.getInstance().shutdown();
return Chart;
} // End of the method
// =======================================================
// Load CSV to memory.
// @return The loaded dataset.
// =======================================================
public static MLDataSet loadCSV2Memory(String filename, int input,
int ideal, boolean headers,
CSVFormat format, boolean significance)
{
DataSetCODEC codec = new CSVDataCODEC(new File(filename), format,
headers, input, ideal, significance);
MemoryDataLoader load = new MemoryDataLoader(codec);
MLDataSet dataset = load.external2Memory();
return dataset;
}
// =======================================================
// The main method.
// @param Command line arguments. No arguments are used.
// =======================================================
public static void main(String[] args)
{
ExampleChart<XYChart> exampleChart = new Sample9();
XYChart Chart = exampleChart.getChart();
new SwingWrapper<XYChart>(Chart).displayChart();
} // End of the main method
//===============================================================
// This method trains, Validates, and saves the trained network file
//===============================================================
static public double trainValidateSaveNetwork()
{
// Load the training CSV file in memory
MLDataSet trainingSet = loadCSV2Memory(trainFileName,
```

```
numberOfInputNeurons, numberOfOutputNeurons,
true,CSVFormat.ENGLISH,false);
// create a neural network
BasicNetwork network = new BasicNetwork();
// Input layer
network.addLayer(new BasicLayer(null,true,numberOfInputNeurons));
// Hidden layer
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
network.addLayer(new BasicLayer(new ActivationTANH(),true,7));
// Output layer
network.addLayer(new BasicLayer(new ActivationTANH(),false,1));
network.getStructure().finalizeStructure();
network.reset();
// train the neural network
final ResilientPropagation train = new ResilientPropagation(network,
trainingSet);
int epoch = 1;
do
{
train.iteration();
System.out.println("Epoch #" + epoch + " Error:" + train.getError());
epoch++;
if (epoch >= 11000 && network.calculateError(trainingSet) >
0.00000091) // 0.00000371
{
returnCode = 1;
System.out.println("Try again");
return returnCode;
}
} while(train.getError() > 0.0000009); // 0.0000037
// Save the network file
EncogDirectoryPersistence.saveObject(new File(
networkFileName),network);
System.out.println("Neural Network Results:");
double sumNormDifferencePerc = 0.00;
double averNormDifferencePerc = 0.00;
double maxNormDifferencePerc = 0.00;
int m = 0; // Record number in the input file
double xPointer = 0.00;
for(MLDataPair pair: trainingSet)
{
m++;
xPointer++;
//if(m == 0)
// continue;
```

```
final MLData output = network.compute(pair.getInput());
MLData inputData = pair.getInput();
MLData actualData = pair.getIdeal();
MLData predictData = network.compute(inputData);
// Calculate and print the results
normInputXPointValue = inputData.getData(0);
normInputYPointValue = inputData.getData(1);
normTargetValue = actualData.getData(0);
normPredictValue = predictData.getData(0);
denormInputXPointValue = ((minXPointDl - maxXPointDh)*normInpu
tXPointValue -
Nh*minXPointDl + maxXPointDh *Nl)/(Nl - Nh);
denormInputYPointValue = ((minYPointDl - maxYPointDh)*normInpu
tYPointValue -
Nh*minYPointDl + maxYPointDh *Nl)/(Nl - Nh);
denormTargetValue =((minTargetValueDl - maxTargetValueDh)*
normTargetValue -
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictValue =((minTargetValueDl - maxTargetValueDh)*
normPredictValue -
Nh*minTargetValueDl + maxTargetValueDh*Nl)/(Nl - Nh);
valueDifference =
Math.abs(((denormTargetValue - denormPredictValue)/
denormTargetValue)*100.00);
System.out.println ("xPoint = " + denormInputXPointValue +
" yPoint = " +
denormInputYPointValue + " denormTargetValue = " +
denormTargetValue + " denormPredictValue = " +
denormPredictValue + " valueDifference = " +
valueDifference);
//System.out.println("intPointNumber = " + intPointNumber);
sumNormDifferencePerc = sumNormDifferencePerc + valueDifference;
if (valueDifference > maxNormDifferencePerc)
maxNormDifferencePerc = valueDifference;
xData.add(denormInputYPointValue);
//xData.add(denormInputYPointValue);
yData1.add(denormTargetValue);
yData2.add(denormPredictValue);
} // End for pair loop
XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLACK);
series2.setLineColor(XChartSeriesColors.LIGHT_GREY);
series1.setMarkerColor(Color.BLACK);
series2.setMarkerColor(Color.WHITE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
try
```

```
{
//Save the chart image
//BitmapEncoder.saveBitmapWithDPI(Chart, chartTrainFileName,
// BitmapFormat.JPG, 100);
BitmapEncoder.saveBitmapWithDPI(Chart,chartTrainFileName,BitmapF
ormat.JPG, 100);
System.out.println ("Train Chart file has been saved") ;
}
catch (IOException ex)
{
ex.printStackTrace();
System.exit(3);
}
// Finally, save this trained network
EncogDirectoryPersistence.saveObject(new File(networkFileName),net
work);
System.out.println ("Train Network has been saved") ;
averNormDifferencePerc = sumNormDifferencePerc/numberOfInputRecords;
System.out.println(" ");
System.out.println("maxErrorPerc = " + maxNormDifferencePerc +
" averErrorPerc = " + averNormDifferencePerc);
returnCode = 0.00;
return returnCode;
} // End of the method
//================================================
// This method load and test the trainrd network
//================================================
static public void loadAndTestNetwork()
{
System.out.println("Testing the networks results");
List<Double> xData = new ArrayList<Double>();
List<Double> yData1 = new ArrayList<Double>();
List<Double> yData2 = new ArrayList<Double>();
double targetToPredictPercent = 0;
double maxGlobalResultDiff = 0.00;
double averGlobalResultDiff = 0.00;
double sumGlobalResultDiff = 0.00;
double maxGlobalIndex = 0;
double normInputXPointValueFromRecord = 0.00;
double normInputYPointValueFromRecord = 0.00;
double normTargetValueFromRecord = 0.00;
double normPredictValueFromRecord = 0.00;
BasicNetwork network;
maxGlobalResultDiff = 0.00;
averGlobalResultDiff = 0.00;
sumGlobalResultDiff = 0.00;
// Load the test dataset into memory
MLDataSet testingSet =
```

```
loadCSV2Memory(testFileName,numberOfInputNeurons,numberOfOutputNeurons
,true,
CSVFormat.ENGLISH,false);
// Load the saved trained network
network =
(BasicNetwork)EncogDirectoryPersistence.loadObject(new
File(networkFileName));
int i = - 1; // Index of the current record
double xPoint = -0.00;
for (MLDataPair pair: testingSet)
{
i++;
xPoint = xPoint + 2.00;
MLData inputData = pair.getInput();
MLData actualData = pair.getIdeal();
MLData predictData = network.compute(inputData);
// These values are Normalized as the whole input is
normInputXPointValueFromRecord = inputData.getData(0);
normInputYPointValueFromRecord = inputData.getData(1);
normTargetValueFromRecord = actualData.getData(0);
normPredictValueFromRecord = predictData.getData(0);
denormInputXPointValue = ((minXPointDl - maxXPointDh)*
normInputXPointValueFromRecord - Nh*minXPointDl +
maxXPointDh*Nl)/(Nl - Nh);
denormInputYPointValue = ((minYPointDl - maxYPointDh)*
normInputYPointValueFromRecord - Nh*minYPointDl +
maxYPointDh*Nl)/(Nl - Nh);
denormTargetValue = ((minTargetValueDl - maxTargetValueDh)*
normTargetValueFromRecord - Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
denormPredictValue =((minTargetValueDl - maxTargetValueDh)*
normPredictValueFromRecord - Nh*minTargetValueDl +
maxTargetValueDh*Nl)/(Nl - Nh);
targetToPredictPercent = Math.abs((denormTargetValue -
denormPredictValue)/
denormTargetValue*100);
System.out.println("xPoint = " + denormInputXPointValue + "
yPoint = " +
denormInputYPointValue + " TargetValue = " +
denormTargetValue + " PredictValue = " +
denormPredictValue + " DiffPerc = " +
targetToPredictPercent);
if (targetToPredictPercent > maxGlobalResultDiff)
maxGlobalResultDiff = targetToPredictPercent;
sumGlobalResultDiff = sumGlobalResultDiff + targetToPredictPercent;
// Populate chart elements
xData.add(denormInputXPointValue);
yData1.add(denormTargetValue);
```

```
yData2.add(denormPredictValue);
} // End for pair loop
// Print the max and average results
System.out.println(" ");
averGlobalResultDiff = sumGlobalResultDiff/numberOfInputRecords;
System.out.println("maxErrorPerc = " + maxGlobalResultDiff);
System.out.println("averErrorPerc = " + averGlobalResultDiff);
// All testing batch files have been processed
XYSeries series1 = Chart.addSeries("Actual data", xData, yData1);
XYSeries series2 = Chart.addSeries("Predict data", xData, yData2);
series1.setLineColor(XChartSeriesColors.BLACK);
series2.setLineColor(XChartSeriesColors.LIGHT_GREY);
series1.setMarkerColor(Color.BLACK);
series2.setMarkerColor(Color.WHITE);
series1.setLineStyle(SeriesLines.SOLID);
series2.setLineStyle(SeriesLines.SOLID);
// Save the chart image
try
{
BitmapEncoder.saveBitmapWithDPI(Chart, chartTestFileName ,
BitmapFormat.JPG, 100);
}
catch (Exception bt)
{
bt.printStackTrace();
}
System.out.println ("The Chart has been saved");
System.out.println("End of testing for test records");
} // End of the method
} // End of the class
```

22.4 HASIL PEMROSESAN

Daftar 22.2 menampilkan fragmen akhir dari hasil proses percobaan

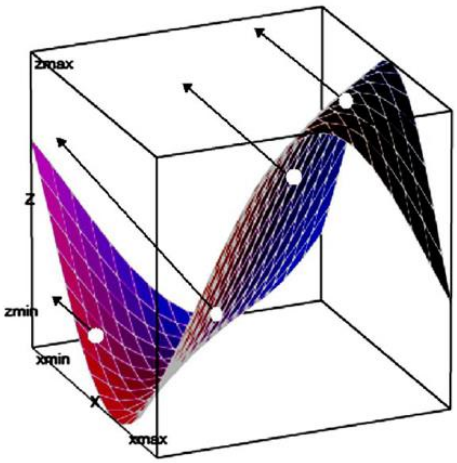
Daftar 22-2. fragmen akhir dari hasil proses percobaan

```
xPoint = 4.0 yPoint = 3.3 TargetValue = 50.59207
PredictedValue = 50.58836 DiffPerc = 0.00733
xPoint = 4.0 yPoint = 3.32 TargetValue = 50.65458
PredictedValue = 50.65049 DiffPerc = 0.00806
xPoint = 4.0 yPoint = 3.34 TargetValue = 50.71290
PredictedValue = 50.70897 DiffPerc = 0.00775
xPoint = 4.0 yPoint = 3.36 TargetValue = 50.76666
PredictedValue = 50.76331 DiffPerc = 0.00659
xPoint = 4.0 yPoint = 3.38 TargetValue = 50.81552
PredictedValue = 50.81303 DiffPerc = 0.00488
xPoint = 4.0 yPoint = 3.4 TargetValue = 50.85916
PredictedValue = 50.85764 DiffPerc = 0.00298
xPoint = 4.0 yPoint = 3.42 TargetValue = 50.89730
PredictedValue = 50.89665 DiffPerc = 0.00128
xPoint = 4.0 yPoint = 3.44 TargetValue = 50.92971
```

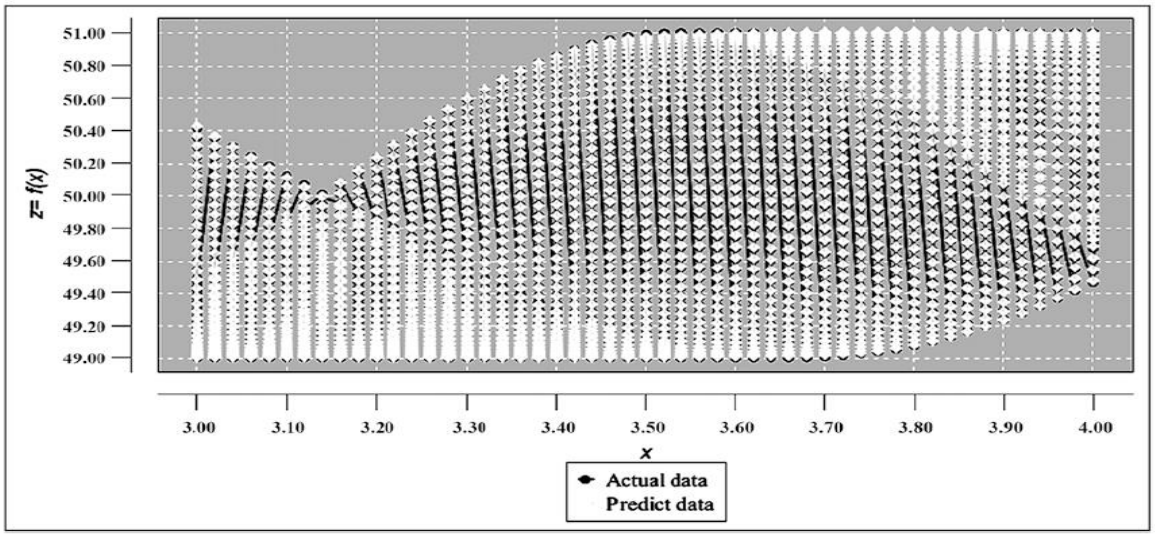
PredictedValue = 50.92964 DiffPerc = 1.31461
xPoint = 4.0 yPoint = 3.46 TargetValue = 50.95616
PredictedValue = 50.95626 DiffPerc = 1.79849
xPoint = 4.0 yPoint = 3.48 TargetValue = 50.97651
PredictedValue = 50.97624 DiffPerc = 5.15406
xPoint = 4.0 yPoint = 3.5 TargetValue = 50.99060
PredictedValue = 50.98946 DiffPerc = 0.00224
xPoint = 4.0 yPoint = 3.52 TargetValue = 50.99836
PredictedValue = 50.99587 DiffPerc = 0.00488
xPoint = 4.0 yPoint = 3.54 TargetValue = 50.99973
PredictedValue = 50.99556 DiffPerc = 0.00818
xPoint = 4.0 yPoint = 3.56 TargetValue = 50.99471
PredictedValue = 50.98869 DiffPerc = 0.01181
xPoint = 4.0 yPoint = 3.58 TargetValue = 50.98333
PredictedValue = 50.97548 DiffPerc = 0.01538
xPoint = 4.0 yPoint = 3.6 TargetValue = 50.96565
PredictedValue = 50.95619 DiffPerc = 0.01856
xPoint = 4.0 yPoint = 3.62 TargetValue = 50.94180
PredictedValue = 50.93108 DiffPerc = 0.02104
xPoint = 4.0 yPoint = 3.64 TargetValue = 50.91193
PredictedValue = 50.90038 DiffPerc = 0.02268
xPoint = 4.0 yPoint = 3.66 TargetValue = 50.87622
PredictedValue = 50.86429 DiffPerc = 0.02344
xPoint = 4.0 yPoint = 3.68 TargetValue = 50.83490
PredictedValue = 50.82299 DiffPerc = 0.02342
xPoint = 4.0 yPoint = 3.7 TargetValue = 50.78825
PredictedValue = 50.77664 DiffPerc = 0.02286
xPoint = 4.0 yPoint = 3.72 TargetValue = 50.73655
PredictedValue = 50.72537 DiffPerc = 0.02203
xPoint = 4.0 yPoint = 3.74 TargetValue = 50.68014
PredictedValue = 50.66938 DiffPerc = 0.02124
xPoint = 4.0 yPoint = 3.76 TargetValue = 50.61938
PredictedValue = 50.60888 DiffPerc = 0.02074
xPoint = 4.0 yPoint = 3.78 TargetValue = 50.55466
PredictedValue = 50.54420 DiffPerc = 0.02069
xPoint = 4.0 yPoint = 3.8 TargetValue = 50.48639
PredictedValue = 50.47576 DiffPerc = 0.02106
xPoint = 4.0 yPoint = 3.82 TargetValue = 50.41501
PredictedValue = 50.40407 DiffPerc = 0.02170
xPoint = 4.0 yPoint = 3.84 TargetValue = 50.34098
PredictedValue = 50.32979 DiffPerc = 0.02222
xPoint = 4.0 yPoint = 3.86 TargetValue = 50.26476
PredictedValue = 50.25363 DiffPerc = 0.02215
xPoint = 4.0 yPoint = 3.88 TargetValue = 50.18685
PredictedValue = 50.17637 DiffPerc = 0.02088
xPoint = 4.0 yPoint = 3.9 TargetValue = 50.10775
PredictedValue = 50.09883 DiffPerc = 0.01780
xPoint = 4.0 yPoint = 3.92 TargetValue = 50.02795
PredictedValue = 50.02177 DiffPerc = 0.01236
xPoint = 4.0 yPoint = 3.94 TargetValue = 49.94798
PredictedValue = 49.94594 DiffPerc = 0.00409
xPoint = 4.0 yPoint = 3.96 TargetValue = 49.86834

PredictedValue = 49.87197 DiffPerc = 0.00727
xPoint = 4.0 yPoint = 3.98 TargetValue = 49.78954
PredictedValue = 49.80041 DiffPerc = 0.02182
xPoint = 4.0 yPoint = 4.0 TargetValue = 49.71209
PredictedValue = 49.73170 DiffPerc = 0.03944
maxErrorPerc = 0.03944085774812906
averErrorPerc = 0.00738084715672128

Saya tidak akan menampilkan bagan hasil pelatihan di sini karena menggambar dua melintasi grafik 3D menjadi berantakan. Sebagai gantinya, saya akan memproyeksikan semua target dan nilai prediksi dari grafik pada satu panel sehingga dapat dengan mudah dibandingkan. Gambar 22.3 menunjukkan grafik dengan proyeksi nilai fungsi pada satu panel.



Gambar 22.3 Proyeksi nilai fungsi pada satu panel



Gambar 22.4 Bagan proyeksi hasil pelatihan

Daftar 22-3 menunjukkan hasil pengujian.

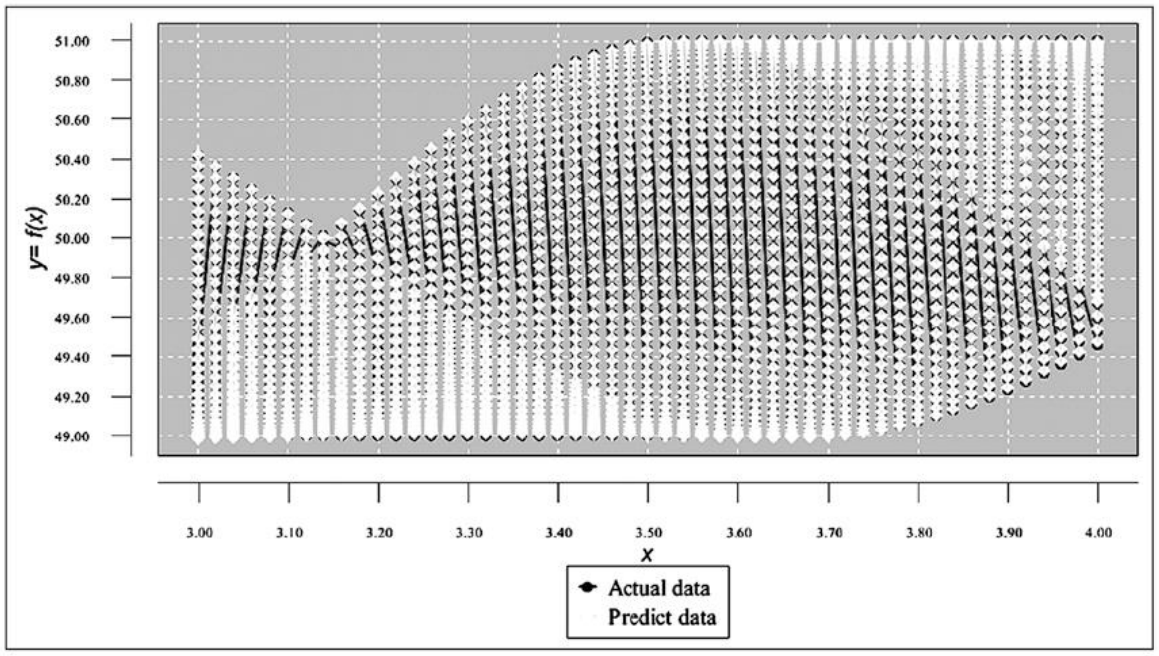
Daftar 22-3. Hasil Pengujian

xPoint = 3.99900 yPoint = 3.13900 TargetValue = 49.98649
PredictValue = 49.98797 DiffPerc = 0.00296
xPoint = 3.99900 yPoint = 3.15900 TargetValue = 50.06642

PredictValue = 50.06756 DiffPerc = 0.00227
xPoint = 3.99900 yPoint = 3.17900 TargetValue = 50.14592
PredictValue = 50.14716 DiffPerc = 0.00246
xPoint = 3.99900 yPoint = 3.19900 TargetValue = 50.22450
PredictValue = 50.22617 DiffPerc = 0.00333
xPoint = 3.99900 yPoint = 3.21900 TargetValue = 50.30163
PredictValue = 50.30396 DiffPerc = 0.00462
xPoint = 3.99900 yPoint = 3.23900 TargetValue = 50.37684
PredictValue = 50.37989 DiffPerc = 0.00605
xPoint = 3.99900 yPoint = 3.25900 TargetValue = 50.44964
PredictValue = 50.45333 DiffPerc = 0.00730
xPoint = 3.99900 yPoint = 3.27900 TargetValue = 50.51957
PredictValue = 50.52367 DiffPerc = 0.00812
xPoint = 3.99900 yPoint = 3.29900 TargetValue = 50.58617
PredictValue 50.59037 DiffPerc = 0.00829
xPoint = 3.99900 yPoint = 3.31900 TargetValue = 50.64903
PredictValue = 50.65291 DiffPerc = 0.00767
xPoint = 3.99900 yPoint = 3.33900 TargetValue = 50.70773
PredictValue = 50.71089 DiffPerc = 0.00621
xPoint = 3.99900 yPoint = 3.35900 TargetValue = 50.76191
PredictValue = 50.76392 DiffPerc = 0.00396
xPoint = 3.99900 yPoint = 3.37900 TargetValue = 50.81122
PredictValue = 50.81175 DiffPerc = 0.00103
xPoint = 3.99900 yPoint = 3.39900 TargetValue = 50.85535
PredictValue = 50.85415 DiffPerc = 0.00235
xPoint = 3.99900 yPoint = 3.41900 TargetValue = 50.89400
PredictValue = 50.89098 DiffPerc = 0.00594
xPoint = 3.99900 yPoint = 3.43900 TargetValue = 50.92694
PredictValue = 50.92213 DiffPerc = 0.00945
xPoint = 3.99900 yPoint = 3.45900 TargetValue = 50.95395
PredictValue = 50.94754 DiffPerc = 0.01258
xPoint = 3.99900 yPoint = 3.47900 TargetValue = 50.97487
PredictValue = 50.96719 DiffPerc = 0.01507
xPoint = 3.99900 yPoint = 3.49900 TargetValue = 50.98955
PredictValue = 50.98104 DiffPerc = 0.01669
xPoint = 3.99900 yPoint = 3.51900 TargetValue = 50.99790
PredictValue = 50.98907 DiffPerc = 0.01731
xPoint = 3.99900 yPoint = 3.53900 TargetValue = 50.99988
PredictValue = 50.99128 DiffPerc = 0.01686
xPoint = 3.99900 yPoint = 3.55900 TargetValue = 50.99546
PredictValue = 50.98762 DiffPerc = 0.01537
xPoint = 3.99900 yPoint = 3.57900 TargetValue = 50.98468
PredictValue = 50.97806 DiffPerc = 0.01297
xPoint = 3.99900 yPoint = 3.59900 TargetValue = 50.96760
PredictValue = 50.96257 DiffPerc = 0.00986
xPoint = 3.99900 yPoint = 3.61900 TargetValue = 50.94433
PredictValue = 50.94111 DiffPerc = 0.00632
xPoint = 3.99900 yPoint = 3.63900 TargetValue = 50.91503
PredictValue = 50.91368 DiffPerc = 0.00265
xPoint = 3.99900 yPoint = 3.65900 TargetValue = 50.87988
PredictValue = 50.88029 DiffPerc = 8.08563
xPoint = 3.99900 yPoint = 3.67900 TargetValue = 50.83910

PredictValue = 50.84103 DiffPerc = 0.00378
xPoint = 3.99900 yPoint = 3.69900 TargetValue = 50.79296
PredictValue = 50.79602 DiffPerc = 0.00601
xPoint = 3.99900 yPoint = 3.71900 TargetValue = 50.74175
PredictValue = 50.74548 DiffPerc = 0.00735
xPoint = 3.99900 yPoint = 3.73900 TargetValue = 50.68579
PredictValue = 50.68971 DiffPerc = 0.00773
xPoint = 3.99900 yPoint = 3.75900 TargetValue = 50.62546
PredictValue = 50.62910 DiffPerc = 0.00719
xPoint = 3.99900 yPoint = 3.77900 TargetValue = 50.56112
PredictValue = 50.56409 DiffPerc = 0.00588
xPoint = 3.99900 yPoint = 3.79900 TargetValue = 50.49319
PredictValue = 50.49522 DiffPerc = 0.00402
xPoint = 3.99900 yPoint = 3.81900 TargetValue = 50.42211
PredictValue = 50.42306 DiffPerc = 0.00188
xPoint = 3.99900 yPoint = 3.83900 TargetValue = 50.34834
PredictValue = 50.34821 DiffPerc = 2.51335
xPoint = 3.99900 yPoint = 3.85900 TargetValue = 50.27233
PredictValue = 50.27126 DiffPerc = 0.00213
xPoint = 3.99900 yPoint = 3.87900 TargetValue = 50.19459
PredictValue = 50.19279 DiffPerc = 0.00358
xPoint = 3.99900 yPoint = 3.89900 TargetValue = 50.11560
PredictValue = 50.11333 DiffPerc = 0.00452
xPoint = 3.99900 yPoint = 3.91900 TargetValue = 50.03587
PredictValue = 50.03337 DiffPerc = 0.00499
xPoint = 3.99900 yPoint = 3.93900 TargetValue = 49.95591
PredictValue = 49.95333 DiffPerc = 0.00517
xPoint = 3.99900 yPoint = 3.95900 TargetValue = 49.87624
PredictValue = 49.87355 DiffPerc = 0.00538
xPoint = 3.99900 yPoint = 3.97900 TargetValue = 49.79735
PredictValue = 49.79433 DiffPerc = 0.00607
xPoint = 3.99900 yPoint = 3.99900 TargetValue = 49.71976
PredictValue = 49.71588 DiffPerc = 0.00781
maxErrorPerc = 0.06317757842407223
averErrorPerc = 0.007356218626151153

Gambar 22.5 menunjukkan grafik proyeksi hasil pengujian.



Gambar 22.5 Bagan proyeksi hasil pengujian

Hasil perkiraan dapat diterima; oleh karena itu, tidak perlu menggunakan metode mikro-batch.

22.5 RINGKASAN

Bab ini membahas cara menggunakan jaringan saraf untuk memperkirakan fungsi dalam 3D ruang angkasa. Anda telah mempelajari bahwa aproksimasi fungsi dalam ruang 3D (berfungsi dengan dua variabel) dilakukan mirip dengan pendekatan fungsi dalam ruang 2D. Satu-satunya perbedaannya adalah bahwa arsitektur jaringan harus mencakup dua input, dan pelatihan dan kumpulan data pengujian harus mencakup catatan untuk semua kemungkinan kombinasi x dan y nilai-nilai. Hasil ini dapat diperluas untuk aproksimasi fungsi dengan lebih dari dua variabel.

DAFTAR PUSTAKA

A survey of robot learning from demonstration. Robotics and Autonomous Systems, 57:469–483, 2009

A. Batzill, Optimal route planning on mobile systems (Masterarbeit, Hochschule Ravensburg-Weingarten, 2016)

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, Handbook of Robotics, pages 1371–1394. Springer, 2008

A. Newell, H.A. Simon, Gps, a program that simulates human thought, in Lernende Automaten, ed. by H. Billing (Oldenbourg, München, 1961), pp. 109–124

A. Newell, J. C. Shaw, and H. A. Simon. Empirical explorations with the logic theory machine: A case study in heuristics. In J. Siekmann and G. Wrightson, editors, Automation of Reasoning 1: Classical Papers on Computational Logic 1957-1966, pages 49–73. Springer, Berlin, Heidelberg, 1983. Erstpublikation: 1957

A. Schwartz. SpamAssassin. O'Reilly, 2004. Spamassassin-Homepage: http://spamassassin.apache.org

A. Zell. Simulation Neuronaler Netze. Addison Wesley, 1994. Description of SNNS and JNNS: http://www-ra.informatik.uni-tuebingen.de/SNNS

A. Zielke, H. Sitter, T.A. Rampp, E. Sch''afer, C. Hasse, W. Lorenz, and M. Rothmund. Überprüfung eines diagnostischen Scoresystems (Ohmann-Score) für die akute Appendizitis. Chirurg 70, 777–783 (1999)

A.G. Barto, S. Mahadevan, Recent advances in hierarchical reinforcement learning. Discrete Event Systems, Special issue on reinforcement learning 13, 41 –77 (2003)

A.L. Samuel, Some Studies in Machine Learning Using the Game of Checkers. IBM Journal 1(3), 210–229 (1959)

A.L. Samuel, Some Studies in Machine Learning Using the Game of Checkers. II. IBM Journal 11(6), 601–617 (1967)

A.M. Turing, Computing Machinery and Intelligence. Mind 59, 433–460 (1950)

A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathemat. Society, 42(2), 1937

Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint http://arxiv.org/abs/1308.0850, 2013.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, Mai 2015. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

B. Brabec. Computergestützte regionale Lawinenprognose. PhD thesis, ETH Zürich, 2001

B. Fischer and J. Schumann. Setheo goes software engineering: Application of atp to software reuse. In Conference on Automated Deduction (CADE-14), volume 1249 of LNCS, pages 65–68. Springer, 1997. http://ase.arc.nasa.gov/people/schumann/publications/papers/cade97-reuse.html

B. Hontschik. Theorie und Praxis der Appendektomie. Mabuse Verlag, 1994

B. Staehle, S. Pfiffner, B. Reiner, W. Ertel, B. Weber-Fiori, and M. Winter. Marvin, ein Assistenzroboter für Menschen mit körperlicher Behinderung im praktischen Einsatz. In M.A. Pfannstiel, S. Krammer, and W. Swoboda, editors, Digitalisierung von Dienstleistungen im Gesundheitswesen. Springer Verlag, 2016. http://asrobe.hsweingarten.de

C. Beierle and G. Kern-Isberner. Methoden wissensbasierter Systeme. Vieweg, 2000

C. Elkan. The paradoxical success of fuzzy logic. In Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), pages 698–703. MIT Press, 1993

C. Goller and A. Küchler. Learning Task-Dependent Distributed Representations by Backpropagation Through Structure. In Proc. of the ICNN-96, volume 1, pages 347–352. IEEE, 1996

C. Goller and A. Küchler. Learning Task-Dependent Distributed Structure-Representations by Backpropagation Through Structure. AR-Report AR-95-02, Institut für Informatik, Technische UniversitätM ünchen, 1995. (a shortened version will appear in the Proc. of the ICNN-96)

C. Goller. A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems. PhD thesis, Fakultätf ür Informatik, Technische UniversitätM ünchen, 1997. (In preparation)

C. Goller. A Connectionist Control Component for the Theorem Prover SETHEO. In Proc. of the ECAI'94 Workshop W14: Combining Symbolic and Connectionist Processing, pages 99–93. ECAI in cooperation with AAAI and IJCAI, 1994

C. Ohmann, C. Franke, Q. Yang, M. Margulies, M. Chan, van P.J. Elk, F.T. de Dombal, and H.D. Röher. Diagnosescore für akute Appendizitis. Der Chirurg, 66:135–141, 1995

C. Ohmann, C. Platen, G. Belenky, Computerunterstütze Diagnose bei akuten Bauchschmerzen. Chirurg 63, 113–123 (1994)

C. Ohmann, Q. Yang, C. Franke, Diagnostic scores for Acute Appendicitis. Eur. J. Surg. 161, 273–281 (1995)

C. Ohmann, V. Moustakis, Q. Yang, K. Lang, Evaluation of automatic knowledge acquisition techniques in the diagnosis of acute abdominal pain. Art. Intelligence in Medicine 8, 23 –36 (1996)

C. Ohmann,M. Kraemer, S. Jaeger, H. Sitter, C. Pohl, B. Stadelmayer, P. Vietmeier, J. Wickers, L. Latzke, B. Koch, K. Thon, Akuter bauchschmerz - standardisierte befundung als diagnoseunterstützung. Chirurg 63, 113–123 (1992)

C. Szepesvari. Algorithms for Reinforcement Learning. Morgan & Claypool Publishers, 2010. draft available online: http://www.ualberta.ca/szepesva/RLBook.html

C.E. Rasmussen and C.K.I. Williams. Gaussian Processes for Machine Learning. Mit Press, 2006. Online version: http://www.gaussianprocess.org/gpml/chapters/

C.E. Shannon and W. Weaver. Mathematische Grundlagen der Informationstheorie. Oldenbourg Verlag, 1976

C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html, 1998

C.L. Chang, R.C. Lee, Symbolic Logic and Mechanical Theorem Proving (Academic Press, Orlando, Florida, 1973)

C.M. Bishop, Pattern recognition and machine learning (Springer, New York, 2006)

C.M. Bishop. Neural networks for pattern recognition. Oxford University Press, 2005

Ch. Kreitz, Formale methoden der künstlichen intelligenz. Künstliche Intelligenz 4, 22–28 (2006)

Ch. Suttner and W. Ertel. Automatic Acquisition of Search Guiding Heuristics. In 10th Int. Conf. on Automated Deduction, pages 470–484. Springer-Verlag, LNAI 449, 1990

D. Ferrucci, E. Nyberg, J. Allan, K. Barker, E. Brown, J. Chu-Carroll, A. Ciccolo, P. Duboue, J. Fan, D. Gondek et al. Towards the open advancement of question answer systems. IBM Technical Report RC24789, Yorktown Heights, NY, 2009. http://www.research.ibm.com/deepqa/question_answering.shtml

D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., Mastering the game of go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016)

D.E. Rumelhart, G.E. Hinton, and Williams R.J. Learning Internal Representations by Error Propagation. in [RM86

D.L. Meadows, D.H. Meadows, E. Zahn, and P. Milling. Die Grenzen des Wachstums. Bericht des Club of Rome zur Lage der Menschheit. Dt. Verl. Deutsche Verlagsanstalt, Stuttgart, 1972

D.M.J. Tax. One-class classification. PhD thesis, Delft University of Technology, 2001

D.W. Loveland. Automated Theorem Proving: a Logical Basis. North-Holland, 1978

Daniel Heckerman, Michael P. Wellman, Bayesian networks. Communications of the ACM 38(3), 27–30 (1995)

E. Eder. Relative Complexities of First Order Calculi. Vieweg Verlag, 1991

E. T. Jaynes. Information Theory and Statistical Mechanics. Physical Review, 1957

E.H. Shortliffe, Computer-based medical consultations (MYCIN. North-Holland, New York, 1976)

E.T. Jaynes, On the Rationale of Maximum Entropy Methods. Proc. of the IEEE 70 (9), 939–952 (1982)

E.T. Jaynes. Concentration of distributions at entropy maxima. In Rosenkrantz, editor, Papers on Probability, Statistics and statistical Physics. D. Reidel Publishing Company, 1982

E.T. Jaynes. Probability Theory: The Logic of Science. Cambridge University Press, 2003

E.T. Jaynes. The Well-Posed Problem. In R.D. Rosenkrantz, editor, E.T. Jaynes: Papers on Probability, Statistics and Statistical Physics, pages 133–148. Kluwer Academic Publishers, 1989

E.T. Jaynes. Where do we stand on Maximum Entropy? In R.D. Rosenkrantz, editor, Papers on Probability, Statistics and Statistical Physics, pages 210–314. Kluwer Academic Publishers, 1978

E.W. Adams. The Logic of Conditionals, volume 86 of Synthese Library. D. Reidel Publishing Company, 1975 [Alp04

Exploring Intelligence, volume 9 of Scientific American presents. Scientific American Inc., 1998

F. Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. Psychological Reviews, 65:386–408, 1958. Wiederabdruck in [AR88

F.T. de Dombal, D.J. Leaper, J.R. Staniland, A.P. McCann, J.C. Horrocks, Computer aided diagnosis of acute abdominal pain. British Medical Journal 2,9 –13 (1972)

F.T. de Dombal. Diagnosis of Acute Abdominal Pain. Churchill Livingstone, 1991

F.V. Jensen. Bayesian networks and decision graphs. Springer-Verlag, 2001

G. Görz, C.-R. Rollinger, and J. Schneeberger, editors. Handbuch der Künstlichen Intelligenz. Oldenbourg Verlag, 2003

G. Guerrerio, Spektrum der wissenschaft, spezial 1/2002: Kurt gödel (Spektrum Verlag, Heidelberg, 2002)

G. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets. Neural computation 18(7), 1527–1554 (2006)

G. Melancon, I. Dutour, and G. Bousque-Melou. Random generation of dags for graph drawing. Technical Report INS-R0005, Dutch Research Center for Mathematical and Computer Science (CWI), 2000. http://ftp.cwi.nl/CWIreports/INS/INSR0005.pdf

G. Palm, On associative memory. Biological Cybernetics 36, 19 –31 (1980)

G. Palm. Memory capacities of local rules for synaptic modification. Concepts in Neuroscience, 2(1):97–128, 1991. MPI Tübingen

G. Sutcliffe and C. Suttner. The State of CASC. AI Communications, 19(1):35–48, 2006. CASC-Homepage: http://www.cs.miami.edu/tptp/CASC

G. Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3), 1995. http://www.research.ibm.com/massive/tdl.html

George Pólya and S. Sloan. How to Solve It: A New Aspect of Mathematical Method. Ishi Press, 2009

Greg Bickerman, Sam Bosley, Peter Swire, and Robert Keller. Learning to create jazz melodies using deep belief nets. In First International Conference on Computational Creativity, 2010

H. Kimura, K. Miyazaki, and S. Kobayashi. Reinforcement Learning in POMDPs with Function Approximation. In 14th International Conference on Machine Learning, pages 152–160. Morgan Kaufmann Publishers, 1997. http://sysplan.nams.kyushu-u.ac.jp/gen/papers/JavaDemoML97/robodemo.html

H. Ritter, T. Martinez, and K. Schulten. Neural computation and self-organizing maps. Addison Wesley, 1992

I. Bratko. PROLOG Programming for Artificial Intelligence. Addison-Wesley, 4th edition, 2011

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Buch in Vorbereitung für MIT Press, http://www.deeplearningbook.org, 2016

I. Witten and E. Frank. Data mining. Hanser Verlag München, 2001. (DataMining Java Library WEKA: http://www.cs.waikato.ac.nz/ml/weka)

J. Anderson and E. Rosenfeld. Neurocomputing: Foundations of Research. MIT Press, Cambridge, MA, 1988. Collection of fundamental original papers

J. Anderson, A. Pellionisz, and E. Rosenfeld. Neurocomputing (vol. 2): directions for research. MIT Press, Cambridge, MA, USA, 1990

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In Advances in Neural Information Processing Systems, pages 2546–2554, 2011

J. Grahl and R. Kümmel. Das Loch im Fass – Energiesklaven, Arbeitsplätze und die Milderung des Wachstumszwangs. Wissenschaft und Umwelt Interdiziplinär, 13:195–212, 2009. http://www.fwu.at/assets/userFiles/Wissenschaft_Umwelt/13_2009/2009_13_wachstum_5.pdf

J. Hertz, A. Krogh, and R. Palmer. Introduction to the theory of neural computation. Addison Wesley, 1991

J. Huber. Monetäre Modernisierung, Zur Zukunft der Geldordnung: Vollgeld und Monetative. Metropolis Verlag, 2014. http://www.monetative.de

J. McDermott, R1: A rule-based configurer of computer systems. Artificial Intelligence 19, 39 –88 (1982)

J. Pearl, Heuristics (Addison-Wesley Publishing Company, Intelligent Search Strategies for Computer Problem Solving, 1984)

J. Pearl, Probabilistic Reasoning in Intelligent Systems (Morgan Kaufmann, Networks of Plausible Inference, 1988)

J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients. Neural Networks 21(4), 682–697 (2008)

J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement learning for humanoid robotics. In Humanoids2003, Third IEEE-RAS International Conference on Humanoid Robots, Karlsruhe, 2003 [Qui

J. Randers. 2052: A Global Forecast for the Next Forty Years. Chelsea Green Publishing, 2012

J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993. C4.5 Download: http://www.rulequest.com/Personal, C5.0 Bestellung: http://www.rulequest.com

J. Schumann. Automated Theorem Proving in Software Engineering. Springer Verlag, 2001

J. Siekmann and Ch. Benzmüller. Omega: Computer supported mathematics. In KI 2004: Advances in Artificial Intelligence, LNAI 3238, pages 3–28. Springer Verlag, 2004. http://www.ags.uni-sb.de/omega

J. Stewart. Multivariable Calculus. Brooks Cole, 2007

J. Weizenbaum, ELIZA–A Computer Program For the Study of Natural Language Communication Between Man and Machine. Communications of the ACM 9(1), 36–45 (1966)

J. Whittaker. Graphical models in applied multivariate statistics. Wiley, 1996 [Wie

J. Wielemaker. SWI-Prolog 5.4. Universität Amsterdam, 2004. http://www.swiprolog.org

J.A. Kalman. Automated Reasoning with OTTER. Rinton Press, 2001. http://wwwunix.mcs.anl.gov/AR/otter/index.html

J.A. Robinson, A machine-oriented logic based on the resolution principle. Journal of the ACM 12(1), 23–41 (1965)

J.B. Greenblatt, S. Saxena, Autonomous taxis could greatly reduce greenhouse-gas emissions of us light-duty vehicles. Nature Clim. Change 5(9), 860–863 (2015)

J.B. Paris, A. Vencovska, A Note on the Inevitability of Maximum Entropy. International Journal of Approximate Reasoning 3, 183–223 (1990)

J.C. Horrocks, A.P. McCann, J.R. Staniland, D.J. Leaper, F.T. de Dombal, Computer-aided diagnosis: Description of an adaptable system, and operational experience with 2.034 cases. British Medical Journal 2,5 –9 (1972)

J.J. Hopfield and D.W. Tank. “Neural” computation of decisions in optimization problems. Biological Cybernetics, 52(3):141–152, 1985. Springer

J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA, 79:2554–2558, April 1982. Wiederabdruck in [AR88

J.N. Kapur and H.K. Kesavan. Entropy Optimization Principles with Applications. Academic Press, 1992

K. Gödel, Diskussion zur Grundlegung der Mathematik, Erkenntnis 2. Monatsheft für Mathematik und Physik 32(1), 147–148 (1931)

K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatsheft für Mathematik und Physik 38(1), 173–198 (1931)

K. Schwab and R. Samans. The future of jobs – employment, skills and workforce strategy for the fourth industrial revolution. World Economic Forum, http://reports.weforum.org/future-of-jobs-2016 January 2016

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2015

Kai-Fu Lee, Sanjoy Mohajan, A Pattern Classification Approach to Evaluation Func tion Learning. Artificial Intelligence 36,1 –25 (1988)

L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Wadsworth, 1984 [Bib82

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In European Conference on Machine Learning(ECML) 2006, pages 282–293. Springer, 2006

L. Panait, S. Luke, Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11(3), 387–434 (2005)

L. v. Ahn. Games with a purpose. IEEE Computer Magazine, pages 96–98, Juni 2006. http://www.cs.cmu.edu/biglou/ieee-gwap.pdf

L.N. Kanal, On Pattern, Categories and Alternate Realities. Pattern Recognition Letters 14, 241–255 (1993)

L.P. Kaelbling, M.L. Littman, and A.P. Moore. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, 4:237–285, 1996. http://www2.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a.pdf

M. Berrondo. Fallgruben für Kopffüssler. Fischer Taschenbuch Nr. 8703, 1989

M. Breunig, H.P. Kriegel, R. Ng, J. Sander, Lof: identifying density-based local outliers. ACM sigmod record 29(2), 93–104 (2000)

M. Fitting. First-order logic and automated theorem proving. Springer, 1996

M. Greiner, G. Tinhofer, Stochastik für Studienanfänger der Informatik (Carl Hanser Verlag, München, Wien, 1996)

M. Greiner, Kölbl A, C. Kredler, and S. Wagenpfeil. Numerical Comparison of Standard SQP-Software with some Second Order Nonlinear Optimization Methods. Report 348, DFG-Schwerpunkt: Anwendungsbezogene Optimierung und Steuerung, 1991

M. Kennedy. Geld ohne Zinsen und Inflation. Ein Tauschmittel, das jedem dient. Goldmann Verlag, München, 2006

M. Lauer and M. Riedmiller. Generalisation in Reinforcement Learning and the Use of Obse rvation-Based Learning. In Gabriella Kokai and Jens Zeidler, editors, Proceedings of the FGML Workshop 2002, pages 100–107, 2002. http://amy.informatik.uos.de/riedmiller/publications/lauer.riedml.fgml02.ps.gz

M. Minsky, S. Papert, Perceptrons (MIT Press, Cambridge, MA, 1969)

M. Newborn. Automated Theorem Proving: Theory and Practice. Springer Verlag, 2000

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In Proceedings of the IEEE International Conference on Neural Networks, pages 586–591, 1993

M. Riedmiller, M. Montemerlo, and H. Dahlkamp. Learning to drive a real car in 20 minutes. In FBIT '07: Proceedings of the 2007 Frontiers in the Convergence of Bioscience and Information Technologies, pages 645–650, Washington, DC, USA, 2007. IEEE Computer Society

M. Riedmiller, T. Gabel, R. Hafner, S. Lange, M. Lauer, Die Brainstormers: Entwurfsprinzipien lernfähiger autonomer Roboter. Informatik-Spektrum 29(3), 175–190 (2006)

M. Schneider and W. Ertel. Robot Learning by Demonstration with Local Gaussian Process Regression. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10), 2010

M. Schneider, W. Ertel, and G. Palm. Expected similarity estimation for large-scale batch and streaming anomaly detection. Machine Learning, 2016. Accepted

M. Schramm and M. Greiner. Foundations: Indifference, Independence & Maxent. In J. Skilling, editor, Maximum Entropy and Bayesian Methods in Science and Engeneering (Proc. of the MaxEnt'94). Kluwer Academic Publishers, 1995

M. Schramm and W. Ertel. Reasoning with Probabilities and Maximum Entropy: The System PIT and its Application in LEXMED. In K. Inderfurth et al, editor, Operations Research Proceeedings (SOR'99), pages 274–280. Springer Verlag, 2000

M. Schramm. Indifferenz, Unabhängigkeit und maximale Entropie: Eine wahrscheinlichkeitstheoretische Semantik für Nicht-Monotones Schließen. Number 4 in Dissertationen zur Informatik. CS-Press, München, 1996

M. Tokic, W. Ertel, and J. Fessler. The crawler, a class room demonstrator for reinforcement learning. In In Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference (FLAIRS 09), Menlo Park, California, 2009. AAAI Press

M. Tokic. Entwicklung eines Lernfähigen Laufroboters. Diplomarbeit Hochschule Ravensburg-Weingarten, 2006. Inklusive Simulationssoftware verfügbar auf http://www.hs-weingarten.de/ertel/kibuch

Michael I. Jordan (ed.), Learning in graphical models (MIT Press, Cambridge, MA, USA, 1999)

Mysteries of the Mind. Speciel Issue. Scientific American Inc., 1997

N. Nilsson. Artificial Intelligence – A New Synthesis. Morgan Kaufmann, 1998

N. Paech, Befreiung vom Überfluss – Grundlagen einer Wirtschaft ohne Wachstum, Fromm Forum, volume 20 (Erich Fromm Gesellschaft, Tübingen, 2016), pp. 70–76

N.J. Nilsson, Probabilistic Logic. Artificial Intelligence 28(1), 71–87 (1986)

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115(3), 211–252 (2015)

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3156–3164, 2015

Oliver Obst and Markus Rollmann. SPARK – A Generic Simulator for Physical Multiagent Simulations. In Gabriela Lindemann, Jörg Denzinger, Ingo J. Timm, and Rainer Unland, editors, Multiagent System Technologies – Proceedings of the MATES 2004, volume 3187, pages 243–257. Springer, September 2004

P. Cheeseman. A method for computing generalised bayesian probability values for expert systems. In Proc. of the 8th Intl. Joint Conf. on Artificial Intelligence (IJCAI-83), 1983

P. Cheeseman. In defense of probability. In Proc. of the 9th Intl. Joint Conf. on Artificial Intelligence (IJCAI-85), 1985

P. Gao, R. Hensley, and A. Zielke. A road map to the future for the auto industry. McKinsey Quarterly, Oct, 2014

P. Hammerer and M. Lein. Stellenwert der PSA-Bestimmung zur Früherkennung des Prostatakarzinoms. Deutsches Ärzteblatt, 101(26):A–1892/B–1581/C–1517, 2004.

http://www.aerzteblatt.de/archiv/42497/Stellenwert-der-PSA-Bestimmungzur-Frueherkennung-des-Prostatakarzinoms

P. Stone, R.S. Sutton, and G. Kuhlmann. Reinforcement Learning for RoboCup-Soccer Keepaway. Adaptive Behavior, 2005. http://www.cs.utexas.edu/pstone/Papers/bib2html-links/AB05.pdf

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. 11, 3371–3408 (2010)

P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Computational and Applied Mathematics 20, 53 –65 (1987)

P.M. Murphy, D.W. Aha, UCI Repository of Machine Learning Databases (University of California at Irvine, Department of Information and Computer Science, 1994)

Peter Flach. Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press, 2012

R. Benenson. What is the class of this image? http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html, February 2016

R. Cubek, W. Ertel, and G. Palm. A critical review on the symbol grounding problem as an issue of autonomous agents. In Proceedings of the 38th German Conference on Artificial Intelligence (KI), Dresden, Germany, 2015

R. Cubek, W. Ertel, and G. Palm. High-level learning from demonstration with conceptual spaces and subspace clustering. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, 2015

R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In Experimental Algorithms, pages 319–333. Springer, 2008

R. Kowalski and A. Robert. Algorithmic = Logic + Control. Communications of the ACM, 22(7):424–436, Juli 1979

R. Kümmel. The second law of economics: Energy, entropy, and the origins of wealth. Springer Science & Business Media, 2011

R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. Journal of Automated Reasoning, 8(2):183–212, 1992. http://www4.informatik.tu-muenchen.de/letz/setheo

R. Rojas. Neural Networks: a Systematic Introduction. Springer, 1996

R. Sutton and A. Barto. Reinforcement Learning. MIT Press, 1998. http://www.cs.ualberta.ca/sutton/book/the-book.html

R. Tedrake. Learning control at intermediate reynolds numbers. In Workshop on: Robotics Challenges for Machine Learning II, International Conference on Intelligent Robots and Systems (IROS 2008), Nice, France, 2008

R. W. Robinson. Counting labeled acyclic digraphs. In F. Harary, editor, New Directions in the Theory of Graphs, pages 28–43. Academic Press, 1977

R.C. Barros, P.A. Jaskowiak, R. Cerri, A.C. de Carvalho, A framework for bottom-up induction of oblique decision trees. Neurocomputing 135,3 –12 (2014)

R.C. González and R.E. Woods. Digital Image Processing. Pearson/Prentice Hall, 2008

R.E. Bellman. Dynamic Programming. Princeton University Press, 1957

R.E. Neapolitan. Probabilistic Reasoning in Expert Systems. Wiley-Interscience. John Wiley & Sons, Inc., 1990

R.O. Duda and P.E. Hart. Pattern Classification and Scene Analysis. Wiley, 1973. Klassiker zur Bayes-Decision-Theorie

Robocup official site. http://www.robocup.org

S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. HUGIN - A Shell for Building Bayesian Belief Universes for Expert Systems. In Proc. of the 11th Intl. Joint Conf. on Artificial Intelligence (IJCAI-89), 1989

S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 3rd edition, 2010. 1st edition 1995, http://aima.cs.berkeley.edu

S. Schaal, C.G. Atkeson, Robot juggling: implementation of memory-based learning. IEEE Control Systems Magazine 14(1), 57–71 (1994)

S. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines (Optimization, and Beyond. MIT Press, Regularization, 2002)

S. Schulz. E – A Brainiac Theorem Prover. Journal of AI Communications, 15 (2/3):111–126, 2002. http://www4.informatik.tu-muenchen.de/schulz/WORK/eprover.html

S.L. Lauritzen. Graphical Models. Oxford Science Publications, 1996

S.M. Ross. Introduction to probability and statistics for engineers and scientists. Academic Press, 2009

Silvio Gesell and Philip Pye. The natural economic order. Owen, 1958

T. Cormen, Ch. Leiserson, R. Rivest, Introduction to Algorithms (MIT Press, Cambridge, Mass, 1990)

T. Gabel and M. Riedmiller. Learning a partial behavior for a competitive robotic soccer agent. Künstliche Intelligenz, 20(2), 2006. BöttcherIT Verlag

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data mining, Inference, and Prediction. Springer, Berlin, 3rd. edition, 2009. Online version: http://www-stat.stanford.edu/tibs/ElemStatLearn/

T. Nipkow, L.C. Paulson, and M. Wenzel. Isabelle/HOL — A Proof Assistant for Higher-Order Logic, volume 2283 of LNCS. Springer, 2002. http://www.cl.cam.ac.uk/Research/HVG/Isabelle

T. Piketty, Das Kapital im 21 (CH Beck Verlag, Jahrhundert, 2014)

T. Segaran, C. Evans, and J. Taylor. Programming the Semantic Web. O'Reilly, 2009

T.J. Sejnowski and C.R. Rosenberg. NETtalk: a parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, The John Hopkins University Electrical Engineering and Computer Science Technical Report, 1986. Wiederabdruck in [AR88

Th. Kane. Maximum entropy in nilsson's probabilistic logic. In Proc. of the 11th Intl. Joint Conf. on Artificial Intelligence (IJCAI-89), 1989

The DeepQA Project, 2011. http://www.research.ibm.com/deepqa/deepqa.shtml

V. Braitenberg. Vehicles – Experiments in Synthetic Psychology. MIT Press, 1984

V. Lifschitz. Benchmark problems for formal non-monotonic reasoning. In Reinfrank et al, editor, Non-Monotonic Reasoning: 2nd International Workshop, volume 346 of LNAI, pages 202–219. Springer, 1989

W. Ertel and M. Schramm. Combining Data and Knowledge by MaxEntOptimization of Probability Distributions. In PKDD'99 (3rd European Conference on Principles and Practice of Knowledge Discovery in Databases), volume 1704 of LNCS, pages 323–328, Prague, 1999. Springer Verlag

W. Ertel, J. Schumann, and Ch. Suttner. Learning Heuristics for a Theorem Prover using Back Propagation. In J. Retti and K. Leidlmair, editors, 5. Österreichische Artificial-Intelligence-Tagung, pages 87–95. Informatik-Fachberichte 208, Springer-Verlag, Berlin, Heidelberg, 1989

W. Ertel, M. Schneider, R. Cubek, and M. Tokic. The teaching-box: A universal robot learning framework. In Proceedings of the 14th International Conference on Advanced Robotics (ICAR 2009), 2009. http://www.servicerobotik.hs-weingarten.de/teachingbox

W. Ertel. Advanced mathematics for engineers. Lecture notes, Hochschule Ravensburg-Weingarten: http://www.hs-weingarten.de/ertel/vorlesungen/mae/matheng-skript-1516-v2.pdf, 2015

W. Ertel. Parallele Suche mit randomisiertem Wettbewerb in Inferenzsystemen, volume 25 of DISKI. In fix-Verlag, St. Augustin, 1993. Dissertation, Technische UniversitätM ünchen

W. McCune. Automated deduction systems and groups. www-unix.mcs.anl.gov/AR/others.html. see also http://www-formal.stanford.edu/clt/ARS/systems.html

W. Rautenberg. Einführung in die Mathematische Logik. Vieweg Verlag, 1996

W. Rödder and C.-H. Meyer. Coherent Knowledge Processing at Maximum Entropy by SPIRIT. In KI-96 (German national conference on AI), Dresden, 1996

W.C. Zimmerli, S. Wolf (eds.), Künstliche Intelligenz – Philosophische Probleme (Philipp Reclam, Stuttgart, 1994)

W.F. Clocksin, C.S. Mellish, Programming in Prolog, 4th edn. (Springer, Berlin, Heidelberg, New York, 1994)

W.S. Cleveland, Robust locally weighted regression and smoothing scatterplots. Journal of the American Statistical Association 74(368), 829–836 (1979)

Wikipedia, the free enzyclopedia. http://en.wikipedia.org, 2013 [Win

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. MNIST-Daten: http://yann.lecun.com/exdb/mnist

Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature 521(7553), 436–444 (2015)

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, volume 2011, page 4, 2011. SVHN-Daten: http://ufldl.stanford.edu/housenumbers

Y. Tian and Y. Zhu. Better computer go player with neural network and long-term prediction. arXiv preprint arXiv:1511.06410, 2016