Seri Pengetahuan Penting MIT Press

Daftar lengkap judul dalam seri ini ada di bagian belakang buku ini.

Pembelajaran Mendalam

John D. Kelleher

MIT Press | Cambridge, Massachusetts | London, Inggris

Buku ini dibuat dalam Chaparral Pro oleh Toppan Best-set Premedia Limited. Dicetak dan dijilid di Amerika Serikat.

Library of Congress Katalogisasi-dalam-Data Publikasi

Nama: Kelleher, John D., 1974- penulis.

Judul: Pembelajaran Mendalam / John D. Kelleher.

Deskripsi: Cambridge, MA: The MIT Press, [2019] | Seri: Seri pengetahuan penting pers MIT | Termasuk referensi bibliografi dan indeks.

Pengidentifikasi: LCCN 2018059550 | ISBN 9780262537551 (pbk.: Alk. Paper)

Subjek: LCSH: Pembelajaran mesin. | Kecerdasan buatan.

Klasifikasi: LCC Q325.5 .K454 2019 | DDC 006.3 / 1 — data LC dc23 tersedia di https://lccn.loc.gov/2018059550

10 9 8 7 6 5 4 3 2 1

Isi

Seri Kata Pengantar

Seri MIT Press Essential Knowledge menawarkan buku-buku ukuran saku yang mudah diakses, ringkas, dan diproduksi dengan indah tentang topik-topik yang sedang menarik. Ditulis oleh para pemikir terkemuka, buku-buku dalam seri ini menyampaikan ikhtisar ahli dari berbagai subjek mulai dari budaya dan sejarah hingga ilmiah dan teknis.

Di era kepuasan informasi instan saat ini, kita memiliki akses ke opini, rasionalisasi, dan deskripsi yang dangkal. Jauh lebih sulit didapat adalah pengetahuan dasar yang menginformasikan pemahaman berprinsip tentang dunia.

Buku Pengetahuan Penting memenuhi kebutuhan itu. Menggabungkan materi pelajaran khusus untuk non-spesialis dan melibatkan topik kritis melalui dasar-dasar, masing-masing volume ringkas ini menawarkan pembaca titik akses ke ide-ide kompleks.

Bruce Tidor
Profesor Teknik Biologi dan Ilmu Komputer
Institut Teknologi Massachusetts

Kata pengantar

Pembelajaran mendalam memungkinkan inovasi dan perubahan di semua aspek kehidupan modern kita. Sebagian besar terobosan kecerdasan buatan yang Anda dengar di media didasarkan pada pembelajaran yang mendalam. Hasilnya, apakah Anda seorang pebisnis yang tertarik untuk meningkatkan efisiensi organisasi Anda, pembuat kebijakan yang peduli dengan etika dan privasi di dunia Big Data, peneliti yang bekerja dengan data kompleks, atau warga negara yang ingin tahu yang menginginkan pemahaman yang lebih baik tentang potensi kecerdasan buatan dan bagaimana hal itu akan mengubah hidup Anda, penting bagi Anda untuk memiliki pemahaman tentang pembelajaran yang mendalam.

Tujuan dari buku ini adalah untuk memungkinkan pembaca umum memperoleh pemahaman tentang apa itu deep learning, dari mana asalnya, bagaimana cara kerjanya, apa yang memungkinkan (dan apa yang tidak), dan bagaimana kemungkinan bidangnya. untuk berkembang dalam sepuluh tahun ke depan. Fakta bahwa pembelajaran mendalam adalah sekumpulan algoritme dan model berarti bahwa memahami pembelajaran dalam memerlukan pemahaman tentang bagaimana algoritme dan model ini memproses data. Alhasil, buku ini tidak semata-mata deskriptif dan definisi; itu juga termasuk penjelasan tentang algoritma. Saya telah mencoba menyajikan materi teknis dengan cara yang dapat diakses. Dari pengalaman mengajar saya, saya menemukan itu untuk topik teknis paling banyakpresentasi yang dapat diakses adalah menjelaskan konsep dasar secara bertahap. Jadi, meskipun saya telah mencoba untuk menjaga konten matematika seminimal mungkin, di mana saya merasa perlu untuk memasukkannya, saya telah berusaha untuk memandu Anda melalui persamaan matematika dengan cara yang sejelas dan langsung mungkin. Saya telah melengkapi penjelasan ini dengan contoh dan ilustrasi.

Apa yang benar-benar menakjubkan tentang pembelajaran mendalam bukanlah kompleksitas matematika yang menjadi dasarnya, melainkan, bahwa ia dapat melakukan beragam tugas yang menarik dan mengesankan menggunakan perhitungan sederhana seperti itu. Jangan kaget saat mendapati diri Anda berkata: "Hanya itu yang dilakukannya?" Faktanya, model deep learning sebenarnya

hanyalah banyak (memang, sangat banyak) perkalian dan penambahan dengan beberapa pemetaan nonlinier (yang akan saya jelaskan) ditambahkan. Namun, terlepas dari kesederhanaan ini, model ini dapat, di antara pencapaian lainnya. , kalahkan juara dunia Go, tentukan visi komputer dan terjemahan mesin tercanggih, dan kendarai mobil. Buku ini adalah teks pengantar tentang deep learning, tetapi saya berharap ini adalah pengantar yang memiliki kedalaman yang cukup sehingga Anda akan kembali ke buku saat kepercayaan diri Anda terhadap materi tersebut tumbuh.

Ucapan Terima Kasih

Buku ini tidak akan mungkin terwujud tanpa pengorbanan yang dilakukan oleh istri saya, Aphra, dan keluarga saya, khususnya orang tua saya John dan Betty Kelleher. Saya juga menerima banyak sekali dukungan dari teman-teman, terutama Alan McDonnell, Ionela Lungu, Simon Dobnik, Lorraine Byrne, Noel Fitzpatrick, dan Josef van Genabith.

Saya juga ingin mengucapkan terima kasih atas bantuan yang saya terima dari staf di MIT Press, dan dari sejumlah orang yang telah membaca bagian-bagian buku ini dan memberikan umpan balik. MIT Press mengatur tiga pengulas anonim yang membaca dan mengomentari draf buku tersebut. Saya berterima kasih kepada pengulas ini atas waktu dan masukannya yang bermanfaat. Juga sejumlah orang membaca draf bab dari buku ini dan saya ingin menggunakan kesempatan ini untuk mengakui bantuan mereka secara terbuka, jadi terima kasih saya kepada: Mike Dillinger, Magdalena Kacmajor, Elizabeth Kelleher, John Bernard Kelleher, Aphra Kerr, Filip Klubička, dan Abhijit Mahalunkar. Buku ini diinformasikan oleh banyak percakapan yang saya lakukan dengan kolega dan siswa tentang pembelajaran mendalam, khususnya dengan Robert Ross dan Giancarlo Salton.

Buku ini didedikasikan untuk saudara perempuan saya Elizabeth (Liz) Kelleher sebagai pengakuan atas cinta dan dukungannya, dan kesabarannya dengan seorang saudara lelaki yang tidak dapat berhenti menjelaskan banyak hal.

1

Pengantar Deep Learning

Pembelajaran mendalam adalah subbidang kecerdasan buatan yang berfokus pada pembuatan model jaringan saraf besar yang mampu membuat *keputusan berdasarkan data yang* akurat. Pembelajaran mendalam sangat cocok untuk konteks di mana datanya kompleks dan di mana tersedia kumpulan data yang

besar. Saat ini sebagian besar perusahaan online dan teknologi konsumen kelas atas menggunakan pembelajaran mendalam. Di antaranya, Facebook menggunakan pembelajaran mendalam untuk menganalisis teks dalam percakapan online. Google, Baidu, dan Microsoft semuanya menggunakan pembelajaran mendalam untuk pencarian gambar, dan juga untuk terjemahan mesin. Semua ponsel pintar modern memiliki sistem pembelajaran yang dalam; misalnya, pembelajaran mendalam sekarang menjadi teknologi standar untuk pengenalan suara, dan juga untuk deteksi wajah pada kamera digital. Di sektor perawatan kesehatan, pembelajaran mendalam digunakan untuk memproses gambar medis (pemindaian sinar-X, CT, dan MRI) dan mendiagnosis kondisi kesehatan. Pembelajaran mendalam juga merupakan inti dari mobil self-driving, di mana ia digunakanlokalisasi dan pemetaan, perencanaan gerak dan kemudi, dan persepsi lingkungan, serta melacak status pengemudi.

Mungkin contoh pembelajaran mendalam yang paling terkenal adalah AlphaGo DeepMind.[1] Go adalah permainan papan yang mirip dengan Catur. AlphaGo adalah program komputer pertama yang mengalahkan pemain Go profesional. Pada Maret 2016, ia mengalahkan profesional top Korea, Lee Sedol, dalam pertandingan yang ditonton lebih dari dua ratus juta orang. Tahun berikutnya, pada 2017, AlphaGo mengalahkan pemain peringkat 1 dunia, Ke Jie dari China.

Pada 2016, kesuksesan AlphaGo sangat mengejutkan. Pada saat itu, kebanyakan orang berharap dibutuhkan waktu bertahun-tahun untuk penelitian sebelum komputer dapat bersaing dengan pemain Go manusia tingkat atas. Sudah lama diketahui bahwa memprogram komputer untuk memainkan Go jauh lebih sulit daripada memprogramnya untuk bermain Catur. Ada lebih banyak kemungkinan konfigurasi papan di Go daripada di Catur. Ini karena Go memiliki papan yang lebih besar dan aturan yang lebih sederhana daripada Catur. Faktanya, ada lebih banyak kemungkinan konfigurasi papan di Go daripada jumlah atom di alam semesta. Ruang pencarian yang sangat besar dan faktor percabangan Go yang besar (jumlah konfigurasi papan yang dapat dicapai dalam satu gerakan) membuat Go menjadi game yang sangat menantang baik untuk manusia maupun komputer.

Salah satu cara untuk menggambarkan kesulitan relatif Go dan Catur yang disajikan ke program komputer adalah melalui perbandingan historis tentang bagaimana program Go dan Chess bersaing dengan pemain manusia. Pada tahun 1967, program Catur MacHack-6 MIT berhasil bersaing dengan manusia dan memiliki peringkat Elo[2] jauh di atas tingkat pemula, dan, pada Mei 1997, DeepBlue mampu mengalahkan juara dunia Catur Gary Kasparov. Sebagai perbandingan, program Go lengkap pertama tidak ditulis hingga tahun 1968 dan pemain manusia yang kuat masih dapat dengan mudah mengalahkan program Go terbaik pada tahun 1997.

Jeda waktu antara pengembangan program komputer Chess dan Go mencerminkan perbedaan kesulitan komputasi antara kedua game ini. Namun, perbandingan historis kedua antara Chess dan Go menggambarkan dampak revolusioner yang dimiliki deep learning pada kemampuan program komputer

untuk bersaing dengan manusia di Go. Perlu waktu tiga puluh tahun bagi program Catur untuk berkembang dari kompetensi tingkat manusia pada tahun 1967 menjadi tingkat juara dunia pada tahun 1997. Namun, dengan perkembangan pembelajaran yang mendalam, hanya dibutuhkan waktu tujuh tahun bagi program komputer Go untuk berkembang dari amatir tingkat lanjut menjadi juara dunia; baru-baru ini pada tahun 2009 program Go terbaik di dunia dinilai sebagai amatir tingkat rendah tingkat lanjut. Akselerasi kinerja melalui penggunaan pembelajaran mendalam ini luar biasa,

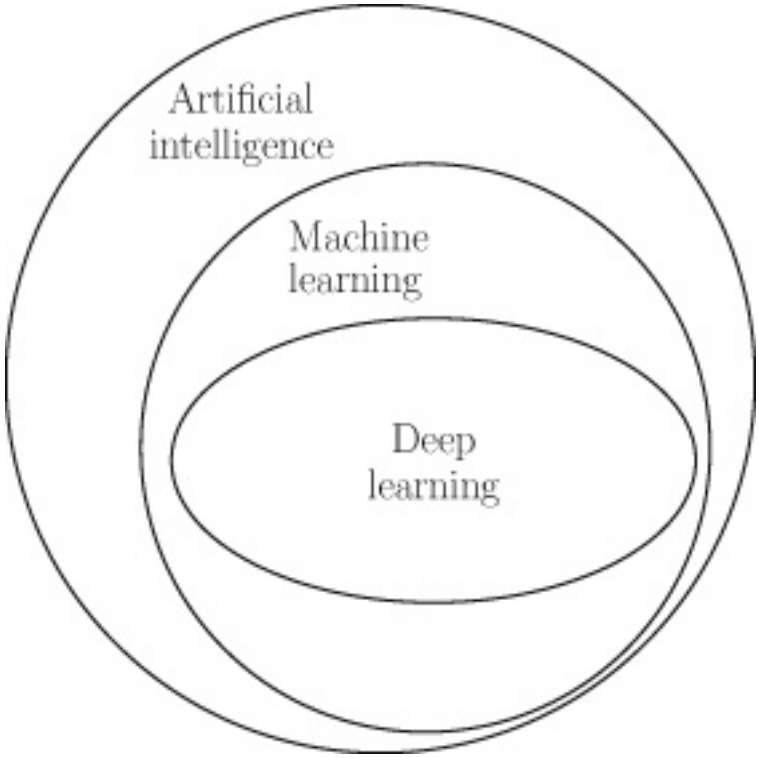
AlphaGo menggunakan pembelajaran mendalam untuk mengevaluasi konfigurasi papan dan untuk memutuskan langkah selanjutnya yang harus diambil. Fakta bahwa AlphaGo menggunakan pembelajaran mendalam untuk memutuskan langkah apa yang harus dilakukan selanjutnya adalah petunjuk untuk memahami mengapa pembelajaran mendalam berguna di banyak domain dan aplikasi yang berbeda. Pengambilan keputusan adalah bagian penting dalam hidup. Satu cara untuk membuat keputusan adalah dengan mendasarkannya pada "intuisi" atau "firasat" Anda. Namun, kebanyakan orang akan setuju bahwa cara terbaik untuk membuat keputusan adalah dengan mendasarkannya pada data yang relevan. Pembelajaran mendalam memungkinkan *keputusan* berdasarkan *data* dengan mengidentifikasi dan mengekstrak pola dari kumpulan data besar yang secara akurat memetakan dari kumpulan input kompleks ke hasil keputusan yang baik.

Kecerdasan Buatan, Pembelajaran Mesin, dan Pembelajaran Mendalam

Pembelajaran mendalam telah muncul dari penelitian dalam kecerdasan buatan dan pembelajaran mesin. Gambar 1.1 mengilustrasikan hubungan antara kecerdasan buatan, pembelajaran mesin, dan pembelajaran mendalam.

Pembelajaran mendalam memungkinkan *keputusan* berdasarkan *data* dengan mengidentifikasi dan mengekstrak pola dari kumpulan data besar yang secara akurat memetakan dari kumpulan input kompleks ke hasil keputusan yang baik.

Bidang kecerdasan buatan lahir pada lokakarya di Dartmouth College pada musim panas 1956. Penelitian tentang sejumlah topik dipresentasikan pada lokakarya tersebut antara lain pembuktian teorema matematika, pemrosesan bahasa alami, perencanaan untuk permainan, komputer program yang bisa belajar dari contoh, dan jaringan saraf. Bidang pembelajaran mesin modern mengacu pada dua topik terakhir: komputer yang dapat belajar dari contoh, dan penelitian jaringan saraf.



Gambar 1.1 Hubungan antara kecerdasan buatan, pembelajaran mesin, dan pembelajaran mendalam.

Pembelajaran mesin melibatkan pengembangan dan evaluasi algoritme yang memungkinkan komputer mengekstrak (atau mempelajari) fungsi dari kumpulan data (sekumpulan contoh). Untuk memahami apa yang dimaksud dengan pembelajaran mesin, kita perlu memahami tiga istilah: kumpulan data, algoritme, dan fungsi.

Dalam bentuk yang paling sederhana, dataset adalah tabel yang setiap barisnya berisi deskripsi satu contoh dari suatu domain, dan setiap kolom berisi informasi untuk salah satu fitur dalam domain. Misalnya, tabel 1.1 mengilustrasikan contoh dataset untuk domain aplikasi pinjaman. Dataset ini mencantumkan rincian empat contoh aplikasi pinjaman. Tidak termasuk fitur ID yang hanya untuk kemudahan referensi, setiap contoh dijelaskan menggunakan tiga fitur: pendapatan tahunan pemohon, hutang mereka saat ini, dan solvabilitas kredit mereka.

Tabel 1.1. Kumpulan data pemohon pinjaman dan peringkat solvabilitas kredit mereka yang diketahui

Indo	Pendapatan tahunan	Hutang Lancar	Solvabilitas Kredit
1	$ 150	- $ 100	100
2	$ 250	- $ 300	-50
3	$ 450	- $ 250	400

Indo	Pendapatan tahunan	Hutang Lancar	Solvabilitas Kredit
4	$ 200	- $ 350	-300

Algoritme adalah proses (atau resep, atau program) yang dapat diikuti oleh komputer. Dalam konteks pembelajaran mesin, algoritme menentukan proses untuk menganalisis kumpulan data dan mengidentifikasi pola berulang dalam data. Misalnya, algoritme mungkin menemukan pola yang menghubungkan pendapatan tahunan seseorang dan hutang saat ini dengan peringkat solvabilitas kreditnya. Dalam matematika, hubungan jenis ini disebut sebagai fungsi.

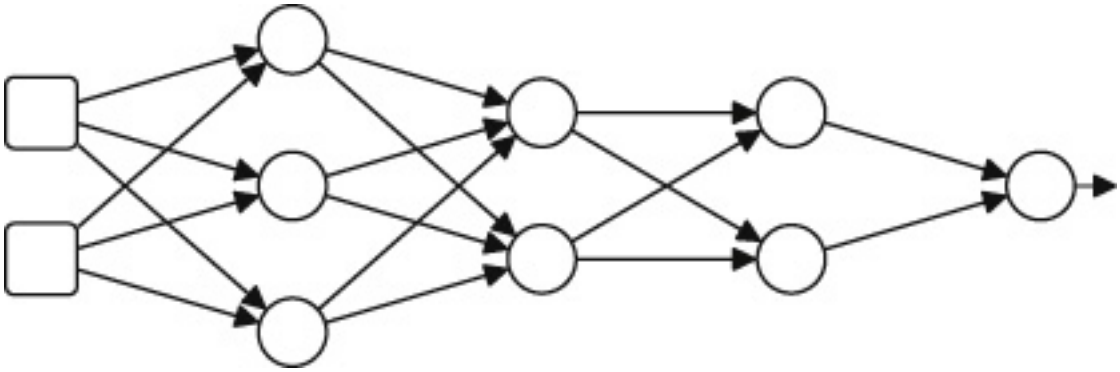
Fungsi adalah pemetaan deterministik dari sekumpulan nilai masukan ke satu atau lebih nilai keluaran. Fakta bahwapemetaan bersifat deterministik artinya untuk setiap rangkaian masukan tertentu suatu fungsi akan selalu mengembalikan keluaran yang sama. Misalnya, penjumlahan adalah pemetaan deterministik, sehingga 2 + 2 selalu sama dengan 4. Seperti yang akan kita bahas nanti, kita dapat membuat fungsi untuk domain yang lebih kompleks daripada aritmatika dasar, misalnya kita dapat mendefinisikan fungsi yang membutuhkan a pendapatan dan hutang seseorang sebagai input dan mengembalikan peringkat solvabilitas kredit mereka sebagai nilai output. Konsep suatu fungsi sangat penting untuk deep learning sehingga perlu mengulang definisi untuk penekanan: fungsi hanyalah pemetaan dari input ke output. Faktanya, tujuan pembelajaran mesin adalah mempelajari fungsi dari data. Suatu fungsi dapat direpresentasikan dengan berbagai cara: dapat sesederhana operasi aritmatika (mis.,*if-then-else* , atau dapat memiliki representasi yang jauh lebih kompleks.

Fungsi adalah pemetaan deterministik dari sekumpulan nilai masukan ke satu atau lebih nilai keluaran.

Salah satu cara untuk merepresentasikan suatu fungsi adalah dengan menggunakan jaringan saraf. Pembelajaran mendalam adalah subbidang pembelajaran mesin yang berfokus pada model jaringan saraf dalam. Faktanya, pola yang diekstrak oleh algoritme pembelajaran dalam dari kumpulan data adalah fungsi yang direpresentasikan sebagai jaringan saraf. Gambar 1.2 mengilustrasikan struktur jaringan saraf. Kotak di sebelah kiri gambar mewakili lokasi memori di mana input disajikan ke jaringan. Setiap lingkaran pada gambar ini disebut neurondan setiap neuron mengimplementasikan sebuah fungsi: ia mengambil sejumlah nilai sebagai masukan dan memetakannya menjadi nilai keluaran. Tanda panah dalam jaringan menunjukkan bagaimana keluaran dari setiap neuron diteruskan sebagai masukan ke neuron lain. Dalam jaringan ini, informasi mengalir dari kiri ke kanan. Misalnya, jika jaringan ini dilatih untuk memprediksi solvabilitas kredit seseorang, berdasarkan pendapatan dan utangnya, ia akan

menerima pendapatan dan utang sebagai input di sebelah kiri jaringan dan mengeluarkan skor solvabilitas kredit melalui neuron di sebelah kanan.

Jaringan saraf menggunakan strategi divide-and-conquer untuk mempelajari suatu fungsi: setiap neuron dalam jaringan mempelajari fungsi sederhana, dan fungsi keseluruhan (lebih kompleks), yang ditentukan oleh jaringan, dibuat dengan menggabungkan fungsi yang lebih sederhana ini. Bab 3 akan menjelaskan bagaimana jaringan saraf memproses informasi.



Gambar 1.2 Ilustrasi skema jaringan saraf.

Apa Itu Pembelajaran Mesin?

Algoritme pembelajaran mesin adalah proses pencarian yang dirancang untuk memilih fungsi terbaik, dari sekumpulan fungsi yang memungkinkan, untuk menjelaskan hubungan antar fitur dalam kumpulan data. Untuk mendapatkan pemahaman intuitif tentang apa yang terlibat dalam penggalian, atau pembelajaran, fungsi dari data, periksa sekumpulan input sampel berikut ke fungsi yang tidak diketahui dan output yang dikembalikannya. Dengan contoh-contoh ini, tentukan operasi aritmatika mana (penjumlahan, pengurangan, perkalian, atau pembagian) yang merupakan pilihan terbaik untuk menjelaskan pemetaan yang ditentukan oleh fungsi yang tidak diketahui antara input dan outputnya:

$$\text{function}(\text{Inputs}) = \text{Output}$$
$$\text{function}(5,5) = 25$$
$$\text{function}(2,6) = 12$$
$$\text{function}(4,4) = 16$$
$$\text{function}(2,2) = 04$$

Kebanyakan orang akan setuju bahwa perkalian adalah pilihan terbaik karena memberikan kecocokan terbaik untuk hubungan yang diamati, atau pemetaan, dari input hingga output:

$5 \times 5 = 25$
$2 \times 6 = 12$
$4 \times 7 = 28$
$2 \times 2 = 04$

Dalam contoh khusus ini, memilih fungsi terbaik relatif mudah, dan manusia dapat melakukannya tanpa bantuan komputer. Namun, karena jumlah input ke fungsi yang tidak diketahui meningkat (mungkin hingga ratusan atau ribuan input), dan variasi fungsi potensial untuk dipertimbangkan semakin besar, tugas menjadi jauh lebih sulit. Dalam konteks inilah memanfaatkan kekuatan pembelajaran mesin untuk mencari fungsi terbaik, untuk mencocokkan pola dalam kumpulan data, menjadi penting.

Pembelajaran mesin melibatkan proses dua langkah: pelatihan dan inferensi. Selama pelatihan, algoritme pembelajaran mesin memproses kumpulan data dan memilih fungsi yang paling cocok dengan pola dalam data. Fungsi yang diekstrak akan dikodekan dalam program komputer dalam bentuk tertentu (seperti aturan if-then-else atau parameter dari persamaan tertentu). Fungsi yang dikodekan dikenal sebagai model, dan analisis data untuk mengekstrak fungsi tersebut sering disebut sebagai pelatihan model.Pada dasarnya, model adalah fungsi yang dikodekan sebagai program komputer. Namun, dalam pembelajaran mesin, konsep fungsi dan model sangat erat kaitannya sehingga perbedaannya sering dilewati dan istilah tersebut bahkan dapat digunakan secara bergantian.

Dalam konteks pembelajaran yang mendalam, hubungan antara fungsi dan model adalah bahwa fungsi yang diekstrak dari dataset selama pelatihan direpresentasikan sebagai model jaringan saraf, dan sebaliknya model jaringan saraf mengkodekan fungsi sebagai program komputer. Proses standar yang digunakan untuk melatih jaringan saraf adalah memulai pelatihan dengan jaringan saraf di mana parameter jaringan diinisialisasi secara acak (kami akan menjelaskan parameter jaringan nanti; untuk saat ini anggap saja sebagai nilai yang mengontrol bagaimana fungsi yang dikodekan jaringan bekerja). Jaringan yang diinisialisasi secara acak ini akan sangat tidak akurat dalam hal kemampuannya untuk mencocokkan hubungan antara berbagai nilai masukan dan keluaran target untuk contoh dalam kumpulan data. Proses pelatihan kemudian dilanjutkan dengan melakukan iterasi melalui contoh-contoh dalam kumpulan data, dan, untuk setiap contoh, menyajikan nilai input ke jaringan dan kemudian menggunakan perbedaan antara output yang dikembalikan oleh jaringan dan output yang benar untuk contoh yang tercantum dalam dataset untuk memperbarui parameter jaringan sehingga lebih cocok dengan data. Setelah algoritme pembelajaran mesin menemukan fungsi yang cukup akurat (dalam hal keluaran yang dihasilkannyamencocokkan keluaran yang benar yang tercantum dalam kumpulan data) untuk masalah yang kita coba selesaikan, proses pelatihan selesai, dan model akhir dikembalikan oleh algoritme. Ini adalah titik di mana pembelajaran dalam pembelajaran mesin berhenti.

Setelah pelatihan selesai, model diperbaiki. Tahap kedua dalam pembelajaran mesin adalah inferensi. Ini adalah saat model diterapkan ke contoh baru — contoh yang nilai keluarannya tidak kita ketahui dengan benar, dan oleh karena itu kita ingin model tersebut menghasilkan perkiraan nilai ini untuk kita. Sebagian besar pekerjaan dalam pembelajaran mesin difokuskan pada cara melatih model yang akurat (yaitu, mengekstrak fungsi yang akurat dari data). Ini karena keterampilan dan metode yang diperlukan untuk menerapkan model pembelajaran mesin terlatih ke dalam produksi, untuk melakukan inferensi pada contoh baru dalam skala besar, berbeda dari yang dimiliki oleh data scientist pada umumnya. Ada pengakuan yang berkembang dalam industri tentang keterampilan khusus yang diperlukan untuk menerapkan sistem kecerdasan buatan dalam skala besar, dan ini tercermin dalam minat yang berkembang di bidang yang dikenal sebagai DevOps, istilah yang menjelaskan kebutuhan untuk kolaborasi antara tim pengembangan dan operasi (tim operasi menjadi tim yang bertanggung jawab untuk menerapkan sistem yang dikembangkan ke dalam produksi dan memastikan bahwa sistem ini stabil dan dapat diskalakan). Istilah MLOps, untuk operasi pembelajaran mesin, dan AIOps, untuk operasi kecerdasan buatan, juga digunakan untuk menggambarkan tantangan dalam menerapkan pelatihanmodel. Pertanyaan seputar penerapan model berada di luar cakupan buku ini, jadi kami akan fokus untuk mendeskripsikan apa itu pembelajaran mendalam, untuk apa ia dapat digunakan, bagaimana perkembangannya, dan bagaimana kita dapat melatih model pembelajaran mendalam yang akurat.

Satu pertanyaan relevan di sini adalah: mengapa mengekstrak fungsi dari data berguna? Alasannya adalah bahwa setelah suatu fungsi diekstraksi dari kumpulan data, fungsi tersebut dapat diterapkan ke data yang tidak terlihat, dan nilai yang dikembalikan oleh fungsi tersebut sebagai respons terhadap masukan baru ini dapat memberikan wawasan tentang keputusan yang tepat untuk masalah baru ini (yaitu, dapat digunakan untuk inferensi). Ingatlah bahwa fungsi hanyalah pemetaan deterministik dari input ke output. Namun, kesederhanaan definisi ini menyembunyikan keragaman yang ada di dalam rangkaian fungsi. Perhatikan contoh berikut:

Pemfilteran spam adalah fungsi yang mengambil email sebagai input dan mengembalikan nilai yang mengklasifikasikan email sebagai spam (atau bukan).

Pengenalan wajah adalah fungsi yang mengambil gambar sebagai input dan mengembalikan pelabelan piksel pada gambar yang membatasi wajah dalam gambar.

Prediksi gen adalah fungsi yang mengambil urutan DNA genom sebagai masukan dan mengembalikan daerah DNA yang menyandikan gen.

Pengenalan ucapan adalah fungsi yang mengambil sinyal suara audio sebagai input dan mengembalikan transkripsi teks dari ucapan tersebut.

Terjemahan mesin adalah fungsi yang mengambil kalimat dalam satu bahasa sebagai masukan dan mengembalikan terjemahan dari kalimat itu dalam bahasa lain.

Itu karena solusi untuk begitu banyak masalah di banyak domain dapat dibingkai sebagai fungsi sehingga pembelajaran mesin menjadi begitu penting dalam beberapa tahun terakhir.

Mengapa Pembelajaran Mesin Sulit?

Ada sejumlah faktor yang membuat tugas pembelajaran mesin menjadi sulit, bahkan dengan bantuan komputer. Pertama, sebagian besar kumpulan data akan menyertakan noise[3] dalam data, jadi mencari fungsi yang benar-benar cocok dengan data belum tentu merupakan strategi terbaik untuk diikuti, karena ini setara dengan mempelajari noise. Kedua, sering kali himpunan fungsi yang memungkinkan lebih besar daripada himpunan contoh dalam dataset. Ini berarti bahwa pembelajaran mesin adalah masalah yang tidak tepat: informasi yang diberikan dalam masalah tidak cukup untuk menemukan *satu* solusi terbaik; sebagai gantinya beberapa solusi yang mungkin akan cocok dengan data. Kita bisa menggunakan soal pemilihan operasi aritmatika (penjumlahan, pengurangan, perkalian, atau pembagian) yang terbaikcocok dengan sekumpulan contoh pemetaan input-output untuk fungsi yang tidak diketahui untuk menggambarkan konsep masalah yang tidak diharapkan. Berikut adalah contoh pemetaan untuk masalah pemilihan fungsi ini:

$$\text{function}(\text{Inputs}) = \text{Output}$$
$$\text{function}(1,1) = 1$$
$$\text{function}(2,1) = 2$$
$$\text{function}(3,1) = 3$$

Dengan contoh-contoh ini, perkalian dan pembagian lebih cocok untuk fungsi yang tidak diketahui daripada penjumlahan dan pengurangan. Namun, tidak mungkin untuk memutuskan apakah fungsi yang tidak diketahui sebenarnya adalah perkalian atau pembagian menggunakan sampel data ini, karena kedua operasi tersebut konsisten dengan semua contoh yang diberikan. Akibatnya, ini adalah masalah yang tidak diharapkan: tidak mungkin untuk memilih satu jawaban terbaik berdasarkan informasi yang diberikan dalam masalah tersebut.

Salah satu strategi untuk memecahkan masalah yang tidak diharapkan adalah mengumpulkan lebih banyak data (lebih banyak contoh) dengan harapan bahwa contoh baru akan membantu kita membedakan antara fungsi dasar yang benar dan alternatif yang tersisa. Namun, seringkali, strategi ini juga tidak memungkinkankarena data tambahan tidak tersedia atau terlalu mahal untuk dikumpulkan. Alih-alih, algoritme pembelajaran mesin mengatasi sifat buruk dari tugas pembelajaran mesin dengan melengkapi informasi yang diberikan oleh data

dengan sekumpulan asumsi tentang karakteristik dari fungsi terbaik, dan menggunakan asumsi ini untuk memengaruhi proses yang digunakan oleh algoritme itu. memilih fungsi (atau model) terbaik. Asumsi ini dikenal sebagai bias induktif dari algoritma karena dalam logika proses yang menyimpulkan aturan umum dari sekumpulan contoh spesifik dikenal sebagai penalaran induktif. Misalnya, jika semua angsa yang pernah Anda lihat dalam hidup Anda berwarna putih, dari contoh-contoh ini Anda mungkin mendapatkan aturan umum bahwa *semua angsa berwarna putih.*. Konsep penalaran induktif ini berkaitan dengan pembelajaran mesin karena algoritma pembelajaran mesin menginduksi (atau mengekstrak) aturan umum (fungsi) dari sekumpulan contoh spesifik (kumpulan data). Akibatnya, asumsi bahwa bias algoritme pembelajaran mesin, pada dasarnya, membiaskan proses penalaran induktif, dan inilah mengapa mereka dikenal sebagai bias induktif algoritme.

Jadi, algoritme pembelajaran mesin menggunakan dua sumber informasi untuk memilih fungsi terbaik: satu adalah kumpulan data, dan yang lainnya (bias induktif) adalah asumsi yang membiaskan algoritme untuk lebih memilih beberapa fungsi daripada yang lain, terlepas dari pola di Himpunan data. Bias induktif dari algoritme pembelajaran mesin dapat dipahami sebagai menyediakan algoritme dengan perspektif pada kumpulan data.Namun, seperti halnya di dunia nyata, di mana tidak ada satu pun perspektif terbaik yang berfungsi di semua situasi, tidak ada satu pun bias induktif terbaik yang berfungsi dengan baik untuk semua kumpulan data. Inilah mengapa ada begitu banyak algoritme pembelajaran mesin: setiap algoritme menyandikan bias induktif yang berbeda. Asumsi yang dikodekan dalam desain algoritme leanring mesin dapat bervariasi kekuatannya. Semakin kuat asumsi, semakin sedikit kebebasan yang diberikan algoritme dalam memilih fungsi yang sesuai dengan pola dalam dataset. Dalam arti tertentu, kumpulan data dan bias induktif saling mengimbangi: algoritme pembelajaran mesin yang memiliki bias induktif yang kuat kurang memperhatikan kumpulan data saat memilih fungsi. Misalnya, jika algoritme pembelajaran mesin dikodekan untuk memilih fungsi yang sangat sederhana, tidak peduli seberapa kompleks pola dalam datanya,

Dalam bab 2 kami akan menjelaskan bagaimana kita dapat menggunakan persamaan garis sebagai struktur template untuk mendefinisikan suatu fungsi. Persamaan garis adalah jenis fungsi matematika yang sangat sederhana. Algoritme pembelajaran mesin yang menggunakan persamaan garis sebagai struktur template untuk fungsi yang sesuai dengan kumpulan data membuat asumsi bahwa model yang mereka hasilkan harus menyandikan pemetaan linier sederhana dari masukan ke keluaran. Asumsi ini adalah contoh bias induktif. Faktanya, ini adalah contoh bias induktif yang kuat, tidak peduli seberapa kompleks (atau nonlinier)pola dalam data adalah algoritme akan dibatasi (atau bias) agar sesuai dengan model linier.

Salah satu dari dua hal bisa salah jika kita memilih algoritma pembelajaran mesin dengan bias yang salah. Pertama, jika bias induktif dari algoritme pembelajaran mesin terlalu kuat, algoritme akan mengabaikan informasi penting dalam data dan fungsi yang dikembalikan tidak akan menangkap nuansa pola

sebenarnya dalam data. Dengan kata lain, fungsi yang dikembalikan akan terlalu sederhana untuk domain tersebut, [4] dan keluaran yang dihasilkannya tidak akan akurat. Hasil ini dikenal sebagai fungsi underfitting data. Alternatifnya, jika bias terlalu lemah (atau permisif), algoritme diberi terlalu banyak kebebasan untuk menemukan fungsi yang sangat cocok dengan data. Dalam kasus ini, fungsi yang dikembalikan mungkin terlalu kompleks untuk domain tersebut, dan, yang lebih bermasalah, fungsi tersebut cenderung cocok dengan noise dalam sampel data yang dipasok ke algoritme selama pelatihan. Menyesuaikan kebisingan dalam data pelatihan akan mengurangi kemampuan fungsi untuk menggeneralisasi ke data baru (data yang tidak ada dalam sampel pelatihan). Hasil ini dikenal sebagai overfitting data.

Namun, dalam domain yang cukup kompleks untuk menjamin penggunaan pembelajaran mesin, tidak mungkin sebelumnya untuk mengetahui asumsi apa yang benar untuk digunakan untuk mencondongkan pemilihan model yang benar dari data. Akibatnya, data scientist harus menggunakan intuisinya (yaitu, membuat tebakan berdasarkan informasi) dan juga menggunakan eksperimen coba-coba untuk menemukan algoritme machine learning terbaik untuk digunakan dalam domain tertentu.

Jaringan saraf memiliki bias induktif yang relatif lemah. Akibatnya, secara umum, bahaya dengan deep learning adalah model jaringan neural akan overfit, bukan underfit, data. Itu karena jaringan saraf sangat memperhatikan data sehingga mereka paling cocok untuk konteks di mana ada kumpulan data yang sangat besar. Semakin besar kumpulan data, semakin banyak informasi yang disediakan data, dan oleh karena itu menjadi lebih bijaksana untuk lebih memperhatikan data. Memang, salah satu faktor terpenting yang mendorong munculnya pembelajaran mendalam selama dekade terakhir adalah munculnya Big Data. Kumpulan data besar yang telah tersedia melalui platform sosial online dan proliferasi sensor telah digabungkan untuk menyediakan data yang diperlukan untuk melatih model jaringan saraf guna mendukung aplikasi baru di berbagai domain.milik lebih dari empat ribu identitas (Taigman et al. 2014).

Bahan Utama Pembelajaran Mesin

Contoh di atas dalam menentukan operasi aritmatika mana yang paling baik menjelaskan hubungan antara input dan output dalam satu set data menggambarkan tiga bahan utama dalam pembelajaran mesin:

1. Data (sekumpulan contoh sejarah).
2. Serangkaian fungsi yang akan dicari oleh algoritme untuk menemukan yang paling cocok dengan data.
3. Beberapa ukuran kebugaran yang dapat digunakan untuk mengevaluasi seberapa cocok setiap fungsi kandidat dengan data.

Ketiga bahan ini harus benar jika proyek pembelajaran mesin ingin berhasil; di bawah ini kami menjelaskan masing-masing bahan tersebut secara lebih rinci.

Kami telah memperkenalkan konsep dataset sebagai tabel dua dimensi (atau matriks $n \times m$), 5 di mana setiap baris berisi informasi untuk satu contoh, dan setiap kolom berisi informasi untuk salah satu fitur di domain. Misalnya, tabel 1.2 menggambarkan bagaimana contoh masukan dan keluaran dari aritmatika pertama yang tidak diketahuimasalah fungsi dalam bab ini dapat direpresentasikan sebagai dataset. Dataset ini berisi empat contoh (juga dikenal sebagai instance), dan setiap contoh direpresentasikan menggunakan dua fitur masukan dan satu fitur keluaran (atau target). Merancang dan memilih fitur untuk mewakili contoh adalah langkah yang sangat penting dalam setiap proyek pembelajaran mesin.

Seperti yang sering terjadi dalam ilmu komputer dan pembelajaran mesin, ada kompromi dalam pemilihan fitur. Jika kami memilih untuk hanya menyertakan sejumlah kecil fitur dalam kumpulan data, maka kemungkinan fitur yang sangat informatif akan dikeluarkan dari data, dan fungsi yang dikembalikan oleh algoritme pembelajaran mesin tidak akan berfungsi dengan baik. Sebaliknya, jika kita memilih untuk menyertakan fitur sebanyak mungkin dalam domain, maka kemungkinan fitur yang tidak relevan atau redundan akan disertakan, dan ini juga akan mengakibatkan fungsi tidak berfungsi dengan baik. Salah satu alasannya adalah semakin banyak fitur redundan atau tidak relevan yang disertakan, semakin besar kemungkinan algoritme pembelajaran mesin untuk mengekstrak pola yang didasarkan pada korelasi palsu antara fitur-fitur ini. Dalam kasus-kasus ini,

Menemukan sekumpulan fitur yang tepat untuk disertakan dalam kumpulan data melibatkan keterlibatan dengan para ahli yang memahami domain, menggunakan analisis statistik dari distribusi fitur individu dan juga korelasi antara pasangan fitur, dan proses trial-and-error untuk membangun model dan memeriksa kinerja model ketika fitur tertentu disertakan atau dikecualikan. Proses desain kumpulan data ini adalah tugas padat karya yang sering menghabiskan sebagian besar waktu dan upaya yang dihabiskan untuk proyek pembelajaran mesin. Namun, ini adalah tugas penting jika proyek ingin berhasil. Memang, mengidentifikasi fitur mana yang informatif untuk tugas tertentu sering kali merupakan tempat munculnya nilai sebenarnya dari project machine learning.

Unsur kedua dalam proyek pembelajaran mesin adalah sekumpulan fungsi kandidat yang akan dianggap algoritme sebagai penjelasan potensial dari pola dalam data. Dalam skenario fungsi aritmatika yang tidak diketahui sebelumnya, himpunan fungsi yang dipertimbangkan secara eksplisit ditentukan dan dibatasi menjadi empat: *penjumlahan* , *pengurangan* , *perkalian* , atau *pembagian* . Secara lebih umum, himpunan fungsi secara implisit didefinisikan melalui bias induktif darialgoritma pembelajaran mesin dan representasi fungsi (atau model) yang sedang digunakan. Misalnya, model jaringan neural adalah representasi fungsi yang sangat fleksibel.

Tabel 1.2. Set data tabel sederhana

Masukan 1	Masukan 2	Target
5	5	25
2	6	12
4	4	16
2	2	04

Bahan ketiga dan terakhir untuk pembelajaran mesin adalah ukuran kebugaran. Pengukuran kesesuaian adalah fungsi yang mengambil keluaran dari fungsi kandidat, yang dihasilkan saat algoritme pembelajaran mesin menerapkan fungsi kandidat ke data, dan membandingkan keluaran ini dengan data, dalam beberapa cara. Hasil dari perbandingan ini adalah nilai yang menggambarkan kesesuaian fungsi kandidat relatif terhadap data. Fungsi kebugaran yang akan bekerja untuk skenario fungsi aritmatika kita yang tidak diketahui adalah menghitung berapa banyak contoh fungsi kandidat mengembalikan nilai yang sama persis dengan target yang ditentukan dalam data. Perkalian akan menghasilkan skor empat dari empat pada ukuran kebugaran ini, penjumlahan akan menghasilkan skor satu dari empat, dan pembagian serta pengurangan akan mendapat skor nol dari empat. Ada banyak variasi fungsi kebugaran yang dapat digunakan dalam pembelajaran mesin, dan pemilihan fungsi kebugaran yang tepat sangat penting untuk keberhasilan proyek pembelajaran mesin. Desain fungsi kebugaran baru adalah bidang penelitian yang kaya dalam pembelajaran mesin. Memvariasikan cara kumpulan data direpresentasikan, dan bagaimana fungsi kandidat dan fungsi kebugaran didefinisikan, menghasilkan tiga kategori pembelajaran mesin yang berbeda: diawasi, tidak diawasi, dan pembelajaran penguatan.

Pembelajaran Dibimbing, Tidak Diawasi, dan Penguatan

Pembelajaran mesin yang diawasi adalah jenis pembelajaran mesin yang paling umum. Dalam pembelajaran mesin yang diawasi, setiap contoh dalam kumpulan data diberi label dengan nilai keluaran (atau target) yang diharapkan. Misalnya, jika kita menggunakan dataset pada tabel 1.1 untuk mempelajari fungsi yang memetakan dari input pendapatan tahunan dan hutang ke skor solvabilitas kredit, fitur solvabilitas kredit dalam dataset akan menjadi fitur target. Untuk menggunakan pembelajaran mesin yang diawasi, kumpulan data kita harus mencantumkan nilai fitur target untuk setiap contoh dalam kumpulan data. Nilai

fitur target ini terkadang sangat sulit, dan mahal, untuk dikumpulkan. Dalam beberapa kasus, kita harus membayar tenaga ahli untuk memberi label pada setiap contoh dalam kumpulan data dengan nilai target yang benar. Namun, Manfaat memiliki nilai target ini dalam kumpulan data adalah algoritme pembelajaran mesin dapat menggunakan nilai-nilai ini untuk membantu proses pembelajaran. Hal ini dilakukan dengan membandingkan keluaran yang dihasilkan suatu fungsi dengan keluaran target yang ditentukan dalam kumpulan data, dan menggunakan perbedaan (atau kesalahan) untuk mengevaluasi kesesuaian fungsi kandidat, dan menggunakan evaluasi kesesuaian untuk memandu pencarian fungsi terbaik. Karena umpan balik dari label target dalam kumpulan data ke algoritme inilah jenis pembelajaran mesin ini dianggap diawasi. Ini adalah jenis pembelajaran mesin yang ditunjukkan oleh contoh dan menggunakan perbedaan (atau kesalahan) untuk mengevaluasi kesesuaian fungsi kandidat, dan menggunakan evaluasi kebugaran untuk memandu pencarian fungsi terbaik. Karena umpan balik dari label target dalam kumpulan data ke algoritme inilah jenis pembelajaran mesin ini dianggap diawasi. Ini adalah jenis pembelajaran mesin yang ditunjukkan oleh contoh dan menggunakan perbedaan (atau kesalahan) untuk mengevaluasi kesesuaian fungsi kandidat, dan menggunakan evaluasi kebugaran untuk memandu pencarian fungsi terbaik. Karena umpan balik dari label target dalam kumpulan data ke algoritme inilah jenis pembelajaran mesin ini dianggap diawasi. Ini adalah jenis pembelajaran mesin yang ditunjukkan oleh contohmemilih di antara fungsi aritmatika yang berbeda untuk menjelaskan perilaku fungsi yang tidak diketahui.

Pembelajaran mesin tanpa pengawasan umumnya digunakan untuk mengelompokkan data. Misalnya, jenis analisis data ini berguna untuk segmentasi pelanggan, di mana perusahaan ingin menyegmentasikan basis pelanggannya ke dalam kelompok yang koheren sehingga dapat menargetkan kampanye pemasaran dan / atau desain produk ke setiap kelompok. Dalam pembelajaran mesin tanpa pengawasan, tidak ada nilai target dalam kumpulan data. Akibatnya, algoritme tidak dapat secara langsung mengevaluasi kesesuaian fungsi kandidat terhadap nilai target dalam kumpulan data. Alih-alih, algoritme pembelajaran mesin mencoba mengidentifikasi fungsi yang memetakan contoh serupa ke dalam cluster, sedemikian rupa sehingga contoh dalam cluster lebih mirip dengan contoh lain dalam cluster yang sama daripada contoh di cluster lain. Perhatikan bahwa cluster tidak ditentukan sebelumnya, atau paling banyak pada awalnya sangat tidak ditentukan. Misalnya, data scientist dapat memberikan sejumlah target cluster pada algoritme, berdasarkan beberapa intuisi tentang domain, tanpa memberikan informasi eksplisit tentang ukuran relatif cluster atau tentang karakteristik contoh yang dimiliki dalam setiap cluster. Algoritme pembelajaran mesin yang tidak diawasi sering kali dimulai dengan menebak pengelompokan awal contoh dan kemudian menyesuaikan kluster secara berulang (dengan melepaskan instans dari satu kluster dan menambahkannya ke kluster lain) untuk meningkatkan kesesuaian set kluster. Fungsi kebugaran yang digunakan dalam Algoritme pembelajaran mesin yang tidak diawasi sering kali dimulai dengan menebak pengelompokan awal contoh dan kemudian menyesuaikan kluster secara berulang (dengan melepaskan instans dari satu kluster dan menambahkannya ke kluster lain) untuk

meningkatkan kesesuaian set kluster. Fungsi kebugaran yang digunakan dalam Algoritme pembelajaran mesin yang tidak diawasi sering kali dimulai dengan menebak pengelompokan awal contoh dan kemudian menyesuaikan kluster secara berulang (dengan melepaskan instans dari satu kluster dan menambahkannya ke kluster lain) untuk meningkatkan kesesuaian set kluster. Fungsi kebugaran yang digunakan dalampembelajaran mesin tanpa pengawasan umumnya menghargai fungsi kandidat yang menghasilkan kesamaan yang lebih tinggi dalam masing-masing cluster dan, juga, keragaman yang tinggi antar cluster.

Pembelajaran penguatan paling relevan untuk tugas kontrol online, seperti kontrol robot dan permainan game. Dalam skenario ini, agen perlu mempelajari kebijakan tentang bagaimana ia harus bertindak di lingkungan untuk mendapatkan penghargaan. Dalam pembelajaran penguatan, tujuan agen adalah untuk mempelajari pemetaan dari pengamatan lingkungan saat ini dan keadaan internalnya sendiri (ingatannya) ke tindakan apa yang harus diambil: misalnya, *apakah robot harus bergerak maju atau mundur* atau *haruskah robot bergerak maju atau mundur? program komputer memindahkan bidak atau mengambil ratu* . Keluaran dari kebijakan (fungsi) ini adalah tindakan yang harus diambil agen selanjutnya, mengingat konteks saat ini. Dalam jenis skenario ini, sulit untuk membuat kumpulan data historis, sehingga pembelajaran penguatan sering dilakukan *secara in situ*: agen dilepaskan ke lingkungan tempat ia bereksperimen dengan kebijakan yang berbeda (dimulai dengan kebijakan yang berpotensi acak) dan seiring waktu memperbarui kebijakannya sebagai tanggapan atas imbalan yang diterimanya dari lingkungan tersebut. Jika suatu tindakan menghasilkan reward positif, pemetaan dari observasi yang relevan dan menyatakan tindakan itu diperkuat dalam kebijakan, sedangkan jika tindakan menghasilkan reward negatif, pemetaan dilemahkan. Tidak seperti dalam pembelajaran mesin yang diawasi dan tidak diawasi, dalam pembelajaran penguatan, fakta bahwa pembelajaran dilakukan secara in situ berarti bahwa tahap pelatihan dan inferensi saling terkait dansedang berlangsung. Agen menyimpulkan tindakan apa yang harus dilakukan selanjutnya dan menggunakan umpan balik dari lingkungan untuk mempelajari cara memperbarui kebijakannya. Aspek khas dari pembelajaran penguatan adalah bahwa output target dari fungsi yang dipelajari (tindakan agen) dipisahkan dari mekanisme penghargaan. Imbalan mungkin bergantung pada beberapa tindakan dan mungkin tidak ada umpan balik imbalan, baik positif atau negatif, yang tersedia langsung setelah tindakan dilakukan. Misalnya, dalam skenario catur, hadiahnya mungkin +1 jika agen memenangkan permainan dan -1 jika agen kalah. Namun, umpan balik hadiah ini tidak akan tersedia sampai langkah terakhir permainan diselesaikan. Begitu, Salah satu tantangan dalam reinforcement learning adalah merancang mekanisme pelatihan yang dapat mendistribusikan reward secara tepat kembali melalui serangkaian tindakan sehingga kebijakan dapat diperbarui secara tepat. Teknologi DeepMind Google menghasilkan banyak minat dengan mendemonstrasikan bagaimana pembelajaran penguatan dapat digunakan untuk melatih model pembelajaran mendalam untuk mempelajari kebijakan kontrol untuk tujuh game komputer Atari yang berbeda (Mnih et al. 2013). Input ke sistem adalah nilai piksel mentah dari layar, dan kebijakan kontrol

menentukan tindakan joystick apa yang harus dilakukan agen di setiap titik dalam game. Lingkungan permainan komputer sangat cocok untuk pembelajaran penguatan karena agen dapat diizinkan untuk memainkan ribuan permainan melawan sistem permainan komputer untuk belajar dengan sukses. Teknologi DeepMind Google menghasilkan banyak minat dengan mendemonstrasikan bagaimana pembelajaran penguatan dapat digunakan untuk melatih model pembelajaran mendalam untuk mempelajari kebijakan kontrol untuk tujuh game komputer Atari yang berbeda (Mnih et al. 2013). Input ke sistem adalah nilai piksel mentah dari layar, dan kebijakan kontrol menentukan tindakan joystick apa yang harus dilakukan agen di setiap titik dalam game. Lingkungan permainan komputer sangat cocok untuk pembelajaran penguatan karena agen dapat diizinkan untuk memainkan ribuan permainan melawan sistem permainan komputer untuk belajar dengan sukses. Teknologi DeepMind Google menghasilkan banyak minat dengan mendemonstrasikan bagaimana pembelajaran penguatan dapat digunakan untuk melatih model pembelajaran mendalam untuk mempelajari kebijakan kontrol untuk tujuh game komputer Atari yang berbeda (Mnih et al. 2013). Input ke sistem adalah nilai piksel mentah dari layar, dan kebijakan kontrol menentukan tindakan joystick apa yang harus dilakukan agen di setiap titik dalam game. Lingkungan permainan komputer sangat cocok untuk pembelajaran penguatan karena agen dapat diizinkan untuk memainkan ribuan permainan melawan sistem permainan komputer untuk belajar dengan sukses. Input ke sistem adalah nilai piksel mentah dari layar, dan kebijakan kontrol menentukan tindakan joystick apa yang harus dilakukan agen di setiap titik dalam game. Lingkungan permainan komputer sangat cocok untuk pembelajaran penguatan karena agen dapat diizinkan untuk memainkan ribuan permainan melawan sistem permainan komputer untuk belajar dengan sukses. Input ke sistem adalah nilai piksel mentah dari layar, dan kebijakan kontrol menentukan tindakan joystick apa yang harus dilakukan agen di setiap titik dalam game. Lingkungan permainan komputer sangat cocok untuk pembelajaran penguatan karena agen dapat diizinkan untuk memainkan ribuan permainan melawan sistem permainan komputer untuk belajar dengan sukses.kebijakan, tanpa menimbulkan biaya pembuatan dan pelabelan sekumpulan besar contoh situasi dengan tindakan joystick yang benar. Sistem DeepMind menjadi sangat bagus dalam permainan sehingga mengungguli semua sistem komputer sebelumnya pada enam dari tujuh permainan, dan mengungguli ahli manusia pada tiga permainan.

Pembelajaran mendalam dapat diterapkan ke ketiga skenario pembelajaran mesin: diawasi, tidak diawasi, dan penguatan. Namun, pembelajaran mesin yang diawasi adalah jenis pembelajaran mesin yang paling umum. Akibatnya, sebagian besar buku ini akan fokus pada pembelajaran mendalam dalam konteks pembelajaran yang diawasi. Namun, sebagian besar perhatian dan prinsip pembelajaran mendalam yang diperkenalkan dalam konteks pembelajaran terbimbing juga berlaku untuk pembelajaran tanpa pengawasan dan penguatan.

Mengapa Deep Learning Sangat Sukses?

Dalam proses berbasis data apa pun, penentu utama kesuksesan adalah mengetahui apa yang harus diukur dan bagaimana mengukurnya. Inilah mengapa proses pemilihan fitur dan desain fitur sangat penting untuk pembelajaran mesin. Seperti yang dibahas di atas, tugas ini dapat memerlukan keahlian domain, analisis statistik data, dan iterasi eksperimen yang membangun model dengan kumpulan fitur yang berbeda. Akibatnya, desain dan persiapan dataset dapat memakan banyak halporsi waktu dan sumber daya yang dikeluarkan dalam proyek, dalam beberapa kasus mendekati hingga 80% dari total anggaran proyek (Kelleher dan Tierney 2018). Desain fitur adalah satu tugas di mana pembelajaran mendalam dapat memiliki keuntungan yang signifikan dibandingkan pembelajaran mesin tradisional. Dalam pembelajaran mesin tradisional, desain fitur sering kali membutuhkan banyak upaya manusia. Pembelajaran mendalam mengambil pendekatan berbeda untuk desain fitur, dengan mencoba mempelajari fitur yang paling berguna untuk tugas secara otomatis dari data mentah.

Dalam proses berbasis data apa pun, penentu utama kesuksesan adalah mengetahui apa yang harus diukur dan bagaimana mengukurnya.

Sebagai contoh desain fitur, indeks massa tubuh (BMI) seseorang adalah rasio berat badan seseorang (dalam kilogram) dibagi dengan tinggi badan (dalam meter kuadrat). Dalam pengaturan medis, BMI digunakan untuk mengkategorikan orang sebagai kurus, normal, kelebihan berat badan, atau obesitas. Mengelompokkan orang dengan cara ini dapat berguna untuk memprediksi kemungkinan seseorang mengembangkan kondisi medis terkait berat badan, seperti diabetes. BMI digunakan untuk kategorisasi ini karena memungkinkan dokter untuk mengkategorikan orang dengan cara yang relevan dengan kondisi medis terkait berat badan ini. Umumnya, semakin tinggi orang, mereka juga semakin berat. Namun, sebagian besar kondisi medis yang berhubungan dengan berat badan (seperti diabetes) tidak dipengaruhi oleh tinggi badan seseorang, melainkan jumlah kelebihan berat badan dibandingkan dengan orang lain dengan perawakan yang sama. BMI adalah fitur yang berguna untuk digunakan dalam kategorisasi medis berat badan seseorang karena mempertimbangkan pengaruh tinggi badan terhadap berat badan. BMIadalah contoh fitur yang diturunkan (atau dihitung) dari fitur mentah; dalam hal ini fitur mentahnya adalah berat dan tinggi. BMI juga merupakan contoh bagaimana fitur turunan bisa lebih berguna dalam membuat keputusan daripada fitur mentahnya. BMI adalah fitur yang dirancang dengan tangan: Adolphe Quetelet mendesainnya pada abad kedelapan belas.

Seperti yang disebutkan di atas, selama project machine learning, banyak waktu dan upaya dihabiskan untuk mengidentifikasi, atau merancang, fitur (turunan) yang berguna untuk tugas yang coba diselesaikan oleh project tersebut. Keuntungan deep learning adalah dapat mempelajari fitur turunan yang berguna dari data secara otomatis (kita akan membahas cara melakukannya di bab

selanjutnya). Memang, dengan kumpulan data yang cukup besar, pembelajaran mendalam telah terbukti sangat efektif dalam fitur pembelajaran sehingga model pembelajaran dalam sekarang lebih akurat daripada banyak model pembelajaran mesin lain yang menggunakan fitur rekayasa tangan. Ini juga mengapa pembelajaran mendalam sangat efektif dalam domain yang contoh-contohnya dijelaskan dengan fitur yang sangat banyak. Secara teknis, kumpulan data yang berisi banyak fitur disebut dimensi tinggi. Sebagai contoh, kumpulan foto dengan fitur untuk setiap piksel dalam foto akan berdimensi tinggi. Dalam domain dimensi tinggi yang kompleks, sangat sulit untuk merekayasa fitur secara manual: pertimbangkan tantangan fitur rekayasa tangan untuk pengenalan wajah atau terjemahan mesin. Jadi, dalam domain kompleks ini,mengadopsi strategi di mana fitur secara otomatis dipelajari dari kumpulan data yang besar masuk akal. Terkait dengan kemampuan untuk secara otomatis mempelajari fitur yang berguna, pembelajaran dalam juga memiliki kemampuan untuk mempelajari pemetaan nonlinier yang kompleks antara masukan dan keluaran; kami akan menjelaskan konsep pemetaan nonlinier di bab 3, dan di bab 6 kami akan menjelaskan bagaimana pemetaan ini dipelajari dari data.

Ringkasan dan Jalan ke Depan

Bab ini berfokus pada pemosisian deep learning dalam bidang machine learning yang lebih luas. Akibatnya, sebagian besar bab ini dikhususkan untuk memperkenalkan pembelajaran mesin. Secara khusus, konsep fungsi sebagai pemetaan deterministik dari input ke output diperkenalkan, dan tujuan pembelajaran mesin dijelaskan sebagai menemukan fungsi yang cocok dengan pemetaan dari fitur input ke fitur output yang diamati dalam contoh di Himpunan data.

Dalam konteks pembelajaran mesin ini, pembelajaran mendalam diperkenalkan sebagai subbidang pembelajaran mesin yang berfokus pada desain dan evaluasi algoritma pelatihan dan arsitektur model untuk jaringan saraf modern. Salah satu aspek berbeda dari pembelajaran mendalam dalam pembelajaran mesin adalah pendekatan yang diperlukandesain fitur. Di sebagian besar proyek pembelajaran mesin, desain fitur adalah tugas intensif manusia yang dapat memerlukan keahlian domain yang mendalam dan menghabiskan banyak waktu dan anggaran proyek. Model deep learning, di sisi lain, memiliki kemampuan untuk mempelajari fitur yang berguna dari data mentah level rendah, dan pemetaan nonlinier kompleks dari input ke output. Kemampuan ini bergantung pada ketersediaan kumpulan data yang besar; namun, jika kumpulan data tersebut tersedia, pembelajaran mendalam sering kali dapat mengungguli pendekatan pembelajaran mesin lainnya. Selain itu, kemampuan untuk mempelajari fitur yang berguna dari kumpulan data besar inilah yang menjadi alasan mengapa pembelajaran mendalam sering kali dapat menghasilkan model yang sangat akurat untuk domain yang kompleks, baik itu dalam terjemahan mesin, pemrosesan ucapan, atau pemrosesan gambar atau video.

Dalam arti tertentu, pembelajaran mendalam telah membuka potensi data besar. Dampak paling nyata dari perkembangan ini adalah integrasi model pembelajaran mendalam ke perangkat konsumen. Namun, fakta bahwa pembelajaran mendalam dapat digunakan untuk menganalisis kumpulan data yang sangat besar juga memiliki implikasi bagi privasi individu dan kebebasan sipil kita (Kelleher dan Tierney 2018). Inilah sebabnya mengapa memahami apa itu pembelajaran mendalam, bagaimana cara kerjanya, dan untuk apa ia bisa dan tidak bisa digunakan, sangat penting. Jalan di depan adalah sebagai berikut:

Bab 2 memperkenalkan beberapa konsep dasar deep learning, termasuk apa itu model, bagaimana parameter model dapat diatur menggunakan data, dan bagaimana kita dapat membuat model yang kompleks dengan menggabungkan model sederhana.

Bab 3 menjelaskan apa itu neural network, bagaimana mereka bekerja, dan apa yang kami maksud dengan deep neural network.

Bab 4 menyajikan sejarah pembelajaran yang mendalam. Sejarah ini berfokus pada terobosan konseptual dan teknis utama yang telah berkontribusi pada pengembangan bidang pembelajaran mesin. Secara khusus, ini memberikan konteks dan penjelasan mengapa pembelajaran mendalam telah melihat perkembangan pesat dalam beberapa tahun terakhir.

Bab 5 menjelaskan keadaan lapangan saat ini, dengan memperkenalkan dua arsitektur neural dalam yang paling populer saat ini: jaringan saraf konvolusional dan jaringan saraf berulang. Jaringan neural konvolusional cocok untuk memproses data gambar dan video. Jaringan neural berulang cocok untuk memproses data sekuensial seperti ucapan, teks, atau data deret waktu. Memahami perbedaan dan kesamaan di kedua arsitektur ini akan memberi Anda kesadaran tentang bagaimana jaringan saraf dalam dapat disesuaikan dengan karakteristik jenis data tertentu, dan juga apresiasi luasnya ruang desain dari arsitektur jaringan yang mungkin.

Bab 6 menjelaskan bagaimana model jaringan neural dalam dilatih, menggunakan algoritme penurunan gradien dan propagasi mundur. Memahami kedua algoritme ini akan memberi Anda wawasan nyata tentang keadaan kecerdasan buatan. Misalnya, ini akan membantu Anda untuk memahami mengapa, dengan data yang cukup, saat ini memungkinkan untuk melatih komputer untuk melakukan tugas tertentu dalam domain yang ditentukan dengan baik pada tingkat di luar kemampuan manusia, tetapi juga mengapa bentuk kecerdasan yang lebih umum masih merupakan tantangan penelitian terbuka untuk kecerdasan buatan.

Bab 7 melihat ke masa depan di bidang pembelajaran mendalam. Ini meninjau tren utama yang mendorong pengembangan pembelajaran mendalam saat ini, dan bagaimana mereka cenderung berkontribusi pada pengembangan bidang di tahun-tahun mendatang. Bab ini juga membahas beberapa tantangan yang dihadapi lapangan, khususnya tantangan untuk memahami dan menafsirkan cara kerja jaringan saraf dalam.

2

Landasan Konseptual

Bab ini memperkenalkan beberapa konsep dasar yang mendukung pembelajaran mendalam. Dasar dari bab ini adalah untuk memisahkan presentasi awal konsep-konsep ini dari terminologi teknis yang digunakan dalam pembelajaran mendalam, yang akan diperkenalkan pada bab-bab selanjutnya.

Jaringan pembelajaran yang mendalam adalah model matematika yang (secara longgar) diilhami oleh struktur otak. Akibatnya, untuk memahami pembelajaran mendalam, sangat membantu untuk memiliki pemahaman intuitif tentang apa itu model matematika, bagaimana parameter model dapat diatur, bagaimana kita dapat menggabungkan (atau menyusun) model, dan bagaimana kita dapat menggunakan geometri untuk memahami bagaimana model memproses informasi.

Apa Itu Model Matematika?

Dalam bentuknya yang paling sederhana, model matematika adalah persamaan yang menggambarkan bagaimana satu atau lebih variabel input terkait dengan variabel output. Dalam bentuk ini model matematika sama dengan fungsi: pemetaan dari masukan ke keluaran.

Dalam setiap diskusi yang berkaitan dengan model, penting untuk mengingat pernyataan George Box bahwa *semua model salah tetapi beberapa berguna!* Agar model berguna, model harus memiliki korespondensi dengan dunia nyata. Korespondensi ini paling jelas dalam hal makna yang dapat dikaitkan dengan variabel. Misalnya, dalam isolasi nilai seperti 78.000 tidak ada artinya karena tidak memiliki korespondensi dengan konsep di dunia nyata. Tetapi *pendapatan tahunan = $ 78.000* memberi tahu kita bagaimana angka tersebut menggambarkan aspek dunia nyata. Setelah variabel dalam model memiliki arti, kita dapat memahami model sebagai mendeskripsikan proses melalui berbagai aspek dunia yang berinteraksi dan menyebabkan peristiwa baru. Peristiwa baru kemudian dijelaskan oleh keluaran model.

Templat yang sangat sederhana untuk model adalah persamaan garis:

$$y = mx + c$$

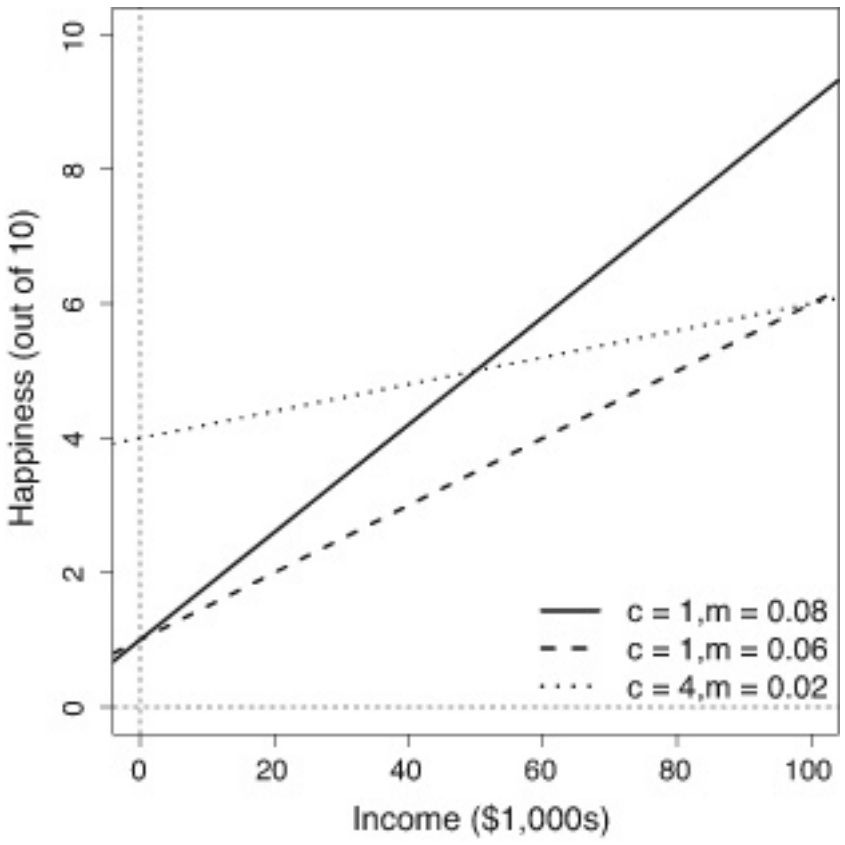
Dalam persamaan ini y adalah variabel keluaran, x merupakan variabel masukan, m dan c merupakan dua parameter modelyang dapat kita atur untuk menyesuaikan hubungan yang didefinisikan model antara input dan output.

Bayangkan kita memiliki hipotesis bahwa pendapatan tahunan memengaruhi kebahagiaan seseorang dan kami ingin menggambarkan hubungan antara kedua variabel ini.[1] Menggunakan persamaan garis, kita dapat mendefinisikan model untuk menggambarkan hubungan ini sebagai berikut:

$$\text{happiness} = m \times \text{income} + c$$

Model ini memiliki arti karena variabel-variabel dalam model (yang berbeda dengan parameter model) memiliki keterkaitan dengan konsep-konsep dari dunia nyata. Untuk melengkapi model kita, kita harus mengatur nilai parameter model: m dan c. Gambar 2.1 mengilustrasikan bagaimana memvariasikan nilai dari masing-masing parameter ini mengubah hubungan yang ditentukan oleh model antara *pendapatan* dan *kebahagiaan* .

Satu hal penting untuk diperhatikan dalam gambar ini adalah bahwa tidak peduli nilai apa yang kita tetapkan parameter model, hubungan yang didefinisikan oleh model antara input dan variabel output dapat diplot sebagai garis. Ini tidak mengherankan karena kami menggunakan persamaan garis sebagai templat untuk menentukan model kami, dan inilah mengapa model matematika yang didasarkan pada persamaan garis dikenal sebagai model linier. Hal penting lainnya yang perlu diperhatikan pada gambaradalah bagaimana mengubah parameter model mengubah hubungan antara pendapatan dan kebahagiaan.



Gambar 2.1 Tiga model linier berbeda tentang bagaimana pendapatan memengaruhi kebahagiaan.

Garis curam yang kokoh, dengan parameter $(c = 1, m = 0.08)$, adalah model dunia di mana orang dengan pendapatan nol memiliki tingkat kebahagiaan 1, dan peningkatan pendapatan berpengaruh signifikan terhadap kebahagiaan orang-orang. Garis putus-putus, dengan parameter $(c = 1, m = 0.06)$, adalah model di mana orang dengan pendapatan nol memiliki tingkat kebahagiaan 1 dan peningkatan pendapatan meningkatkan kebahagiaan, tetapi pada tingkat yang lebih lambat dibandingkan dengan dunia yang dimodelkan dengan garis padat. Akhirnya,Garis putus-putus, parameter $(c = 4, m = 0.02)$, adalah model dunia di mana tidak ada orang yang secara khusus tidak bahagia — bahkan orang dengan pendapatan nol memiliki kebahagiaan 4 dari 10 — dan meskipun peningkatan pendapatan memang memengaruhi kebahagiaan, efeknya sedang. Model ketiga ini mengasumsikan bahwa pendapatan memiliki pengaruh yang relatif lemah terhadap kebahagiaan.

Secara lebih umum, perbedaan antara ketiga model pada gambar 2.1 menunjukkan bagaimana melakukan perubahan pada parameter model linier mengubah model. Mengubah c menyebabkan garis naik dan selesai. Ini paling jelas terlihat jika kita fokus pada sumbu y: perhatikan bahwa garis yang ditentukan oleh model selalu melintasi (atau memotong) sumbu y pada nilai yang c ditetapkan. Inilah sebabnya mengapa c parameter dalam model linier dikenal sebagai intersep. Intersep dapat dipahami sebagai menentukan nilai variabel keluaran ketika variabel masukan adalah nol. Mengubah m parameter mengubah sudut (atau kemiringan) garis. Parameter kemiringan mengontrol seberapa cepat perubahan dalam efek pendapatan mengubah kebahagiaan. Dalam arti tertentu, nilai lereng adalah ukuran seberapa penting pendapatan bagi kebahagiaan. Jika pendapatan sangat penting (yaitu, jika perubahan kecil dalam pendapatan menghasilkan perubahan besar dalam kebahagiaan), maka parameter kemiringan model kita harus disetel ke nilai yang besar. Cara lain untuk memahami hal ini adalah dengan memikirkan parameter kemiringan model linier yang menggambarkan kepentingan, atau bobot, variabel masukan dalam menentukan nilai keluaran.

Model Linear dengan Banyak Input

Persamaan garis dapat digunakan sebagai template untuk model matematika yang memiliki lebih dari satu variabel input. Misalnya, bayangkan diri Anda dalam skenario di mana Anda telah dipekerjakan oleh lembaga keuangan untuk bertindak sebagai petugas pinjaman dan pekerjaan Anda melibatkan keputusan apakah pengajuan pinjaman harus diberikan atau tidak. Dari mewawancarai pakar domain, Anda mendapatkan hipotesis bahwa cara yang berguna untuk memodelkan solvabilitas kredit seseorang adalah dengan mempertimbangkan pendapatan tahunan dan utang mereka saat ini. Jika kita mengasumsikan bahwa terdapat

hubungan linier antara kedua variabel masukan ini dan solvabilitas kredit seseorang, maka model matematika yang sesuai, yang ditulis dalam bahasa Inggris adalah:

$$\begin{aligned}\text{solvency} = &(\text{income} \times \text{weight for income}) \\ &+ (\text{debt} \times \text{weight for debt}) + \text{intercept}\end{aligned}$$

Perhatikan bahwa dalam model ini m parameter telah diganti dengan bobot terpisah untuk setiap variabel masukan, dengan setiap bobot mewakili pentingnya masukan terkait dalam menentukan keluaran. Dalam notasi matematika model ini akan ditulis sebagai:

$$y = (\text{input}_1 \times \text{weight}_1) + (\text{input}_2 \times \text{weight}_2) + c$$

Dimana y mewakili output *solvabilitas kredit* , input_1 mewakili variabel *pendapatan* , input_2 mewakili variabel *hutang* , dan c mewakili intersep. Menggunakan ide untuk menambahkan bobot baru untuk setiap masukan baru ke model memungkinkan kita untuk menskalakan persamaan garis ke masukan sebanyak yang kita suka. Semua model yang didefinisikan dengan cara ini masih linier dalam dimensi yang ditentukan oleh jumlah input dan output. Artinya, model linier dengan dua masukan dan satu keluaran mendefinisikan bidang datar dan bukan garis karena seperti itulah garis dua dimensi yang telah diekstrusi menjadi tiga dimensi.

Menulis model matematika yang memiliki banyak input dapat menjadi membosankan, sehingga ahli matematika suka menulis sesuatu dalam bentuk sesingkat mungkin. Dengan pemikiran ini, persamaan di atas terkadang ditulis dalam bentuk singkat:

$$y = \sum_{i=1}^{n} (\text{input}_i \times \text{weight}_i) + c$$

Notasi ini memberitahu kita bahwa untuk menghitung variabel keluaran y kita harus terlebih dahulu melalui semua n masukan dan n mengalikan setiap masukan dengan bobot yang sesuai, kemudian kita harus menjumlahkan hasil perkalian ini , dan terakhir kita menambahkan c parameter intersep ke hasil penjumlahan. . The Σ simbol memberitahu kita bahwa kita menggunakan Selain untuk menggabungkan hasil dari perkalian, dan indeks i memberitahu kita bahwa kita mengalikan setiap masukan dengan bobot dengan indeks yang sama. Kita dapat membuat notasi kita lebih kompak dengan memperlakukan intersep sebagai

pemberat. Salah satu cara untuk melakukan ini adalah dengan mengasumsikan input_0 bahwa selalu sama dengan 1 dan memperlakukan intersep sebagai bobot pada input ini, yaitu weight_0,. Dengan melakukan ini, kami dapat menulis model sebagai berikut:

$$y = \sum_{i=0}^{n} (\text{input}_i \times \text{weight}_i)$$

Perhatikan bahwa indeks sekarang dimulai dari 0, bukan 1, karena kita sekarang mengasumsikan masukan tambahan $\text{input}_0 = 1$, dan kita telah memberi label ulang pada intersep weight_0.

Meskipun kita dapat menuliskan model linier dalam beberapa cara yang berbeda, inti dari model linier adalah bahwa keluaran dihitung sebagai *jumlah dari n nilai masukan dikalikan dengan bobot yang sesuai* . Akibatnya, model jenis ini mendefinisikan kalkulasi yang dikenal sebagai *penjumlahan terbobot* , karena kita memberi bobot pada setiap masukan dan menjumlahkan hasilnya. Meskipun penjumlahan terbobot mudah dihitung, namun ternyata sangat berguna dalam banyak situasi, dan ini adalah penghitungan dasar yang digunakan di setiap neuron dalam jaringan saraf.

Mengatur Parameter Model Linear

Mari kita kembali ke skenario kerja kita di mana kita ingin membuat model yang memungkinkan kita menghitung kredit solvabilitas individu yang telah mengajukan pinjaman keuangan. Untuk kesederhanaan dalam penyajian, kami akan mengabaikan parameter intersep dalam diskusi ini karena diperlakukan sama dengan parameter lainnya (yaitu bobot pada input). Jadi, dengan menghapus parameter intersep, kami memiliki model linier (atau jumlah tertimbang) berikut dari hubungan antara pendapatan dan hutang seseorang dengan solvabilitas kreditnya:

Perkalian input dengan bobot, diikuti dengan penjumlahan, dikenal sebagai *penjumlahan tertimbang.*

$$\begin{aligned}\text{solvency} = &(\text{income} \times \text{weight for income}) \\ &+ (\text{debt} \times \text{weight for debt})\end{aligned}$$

Untuk melengkapi model kita, kita perlu menentukan parameter model; artinya, kita perlu menentukan nilai bobot untuk setiap masukan. Salah satu cara untuk melakukannya adalah dengan menggunakan keahlian domain kami untuk menghasilkan nilai untuk setiap parameter.

Sebagai contoh, jika kita mengasumsikan bahwa peningkatan pendapatan seseorang memiliki dampak yang lebih besar pada solvabilitas kredit mereka daripada peningkatan yang sama pada hutang mereka, kita harus menetapkan bobot pendapatan menjadi lebih besar dari pada hutang. Model berikut mengkodekan asumsi ini; khususnya model ini menetapkan bahwa pendapatan tiga kali lebih penting dari hutang dalam menentukan solvabilitas kredit seseorang:

$$\text{solvency} = (\text{income} \times 3) + (\text{debt} \times 1)$$

Kekurangan dari penggunaan pengetahuan domain untuk mengatur parameter model adalah bahwa para ahli seringkali tidak setuju. Misalnya, Anda mungkin berpikir bahwa menimbang pendapatan tiga kali lebih penting daripada hutang tidaklah realistis; dalam hal ini model dapat disesuaikan dengan, misalnya, menetapkan pendapatan dan utang untuk memiliki bobot yang sama, yang akan setara dengan asumsi bahwa pendapatan dan utang sama pentingnya dalam menentukan solvabilitas kredit. Salah satu cara untuk menghindari pertengkaran antar pakar adalah dengan menggunakan data untuk mengatur parameter. Di sinilah pembelajaran mesin membantu. Pembelajaran yang dilakukan oleh pembelajaran mesin adalah menemukan parameter (atau bobot) model menggunakan dataset.

Parameter Model Pembelajaran dari Data

Nanti di buku ini kami akan menjelaskan algoritma standar yang digunakan untuk mempelajari bobot untuk model linier, yang dikenal sebagai algoritma penurunan gradien. Namun, kami dapat memberikan pratinjau singkat dari algoritme di sini. Kami mulai dengan kumpulan data yang berisi sekumpulan contoh yang kami memiliki nilai input (pendapatan dan hutang) dan nilai output (solvabilitas kredit). Tabel 2.1 mengilustrasikan kumpulan data tersebut dari skenario solvabilitas kredit kami.[2]

Pembelajaran yang dilakukan oleh pembelajaran mesin adalah menemukan parameter (atau bobot) model menggunakan dataset.

Kami kemudian memulai proses mempelajari bobot dengan menebak nilai awal untuk setiap bobot. Sangat mungkin bahwa model awal yang ditebak ini akan menjadi model yang sangat buruk. Initidak menjadi masalah, bagaimanapun,

karena kita akan menggunakan dataset untuk memperbarui bobot secara berulang sehingga model menjadi lebih baik dan lebih baik, dalam hal seberapa cocok model itu dengan data. Untuk tujuan contoh, kita akan menggunakan model yang dijelaskan di atas sebagai model awal (terkira) kita:

Tabel 2.1. Kumpulan aplikasi pinjaman dan peringkat solvabilitas kredit pemohon yang diketahui

Indo	Pendapatan tahunan	Hutang saat ini	Solvabilitas kredit
1	$ 150	- $ 100	100
2	$ 250	- $ 300	-50
3	$ 450	- $ 250	400
4	$ 200	- $ 350	-300

$$\text{solvency} = (\text{income} \times 3) + (\text{debt} \times 1)$$

Proses umum untuk meningkatkan bobot model adalah memilih contoh dari kumpulan data dan memasukkan nilai masukan dari contoh ke dalam model. Ini memungkinkan kita menghitung perkiraan nilai output untuk contoh. Setelah kita memiliki perkiraan keluaran ini, kita dapat menghitung kesalahan model pada contoh dengan mengurangkan keluaran yang diperkirakan dari keluaran yang benar untuk contoh yang tercantum dalam dataset. Dengan menggunakan kesalahan model pada contoh, kita dapat meningkatkan sebaik apa model tersebutmenyesuaikan data dengan memperbarui bobot dalam model menggunakan strategi berikut, atau aturan pembelajaran:

Jika errornya 0, maka bobot model tidak boleh diubah.
Jika kesalahannya positif, maka keluaran model terlalu rendah, jadi kita harus meningkatkan keluaran model untuk contoh ini dengan menambah bobot untuk semua masukan yang memiliki nilai positif sebagai contoh dan menurunkan bobot untuk semua masukan yang memiliki nilai negatif sebagai contoh.
Jika kesalahannya negatif, maka keluaran model terlalu tinggi, jadi kita harus menurunkan keluaran model untuk contoh ini dengan mengurangi bobot untuk semua masukan yang memiliki nilai positif untuk contoh dan menambah bobot untuk semua masukan yang memiliki nilai negatif sebagai contoh.

Untuk mengilustrasikan proses pembaruan bobot, kami akan menggunakan contoh 1 dari tabel 2.1 (pendapatan = 150, utang = -100, dan solvabilitas = 100) untuk menguji keakuratan model tebakan kami dan memperbarui bobot sesuai dengan kesalahan yang dihasilkan.

$$\begin{aligned}\text{solvency} &= (\text{income} \times 3) + (\text{debt} \times 1) \\ &= (150 \times 3) + (-100 \times 1) \\ &= 350\end{aligned}$$

Ketika nilai input contoh dilewatkan ke dalam model, estimasi solvabilitas kredit yang dikembalikan oleh model adalah 350. Ini lebih besar dari solvabilitas kredit yang dicantumkan untuk contoh ini di dataset, yaitu 100. Akibatnya, kesalahan modelnya negatif (100 - 350 = –250); Oleh karena itu, mengikuti aturan pembelajaran yang dijelaskan di atas, kita harus menurunkan keluaran model untuk contoh ini dengan mengurangi bobot untuk masukan positif dan meningkatkan bobot untuk masukan negatif. Untuk contoh ini, input pendapatan memiliki nilai positif dan input hutang memiliki nilai negatif. Jika kita menurunkan bobot pendapatan sebesar 1 dan menambah bobot utang sebanyak 1, kita berakhir dengan model berikut:

$$\text{solvency} = (\text{income} \times 2) + (\text{debt} \times 2)$$

Kita dapat menguji apakah pembaruan bobot ini telah meningkatkan model dengan memeriksa apakah model baru menghasilkan perkiraan yang lebih baik untuk contoh daripada model lama. Gambar berikut mengilustrasikan mendorong contoh yang sama melalui model baru:

$$\begin{aligned}\text{solvency} &= (\text{income} \times 2) + (\text{debt} \times 2) \\ &= (150 \times 2) + (-100 \times 2) \\ &= 100\end{aligned}$$

Kali ini estimasi solvabilitas kredit yang dihasilkan oleh model sesuai dengan nilai dalam dataset, menunjukkan bahwa model yang diperbarui lebih cocok dengan data daripada model asli. Faktanya, model baru ini menghasilkan keluaran yang benar untuk semua contoh dalam kumpulan data.

Dalam contoh ini, kami hanya perlu memperbarui bobot satu kali untuk menemukan satu set bobot yang membuat perilaku model konsisten dengan semua contoh dalam kumpulan data. Biasanya, bagaimanapun, dibutuhkan banyak iterasi untuk menyajikan contoh dan memperbarui bobot untuk mendapatkan model yang baik. Juga, dalam contoh ini, kami, demi kesederhanaan, mengasumsikan bahwa

bobot diperbarui dengan menambahkan atau mengurangi 1 dari bobot tersebut. Umumnya, dalam pembelajaran mesin, penghitungan berapa banyak untuk memperbarui setiap bobot lebih rumit dari ini. Namun, selain perbedaan ini, proses umum yang diuraikan di sini untuk memperbarui bobot (atau parameter) model agar sesuai dengan model ke dataset adalah proses pembelajaran pada inti dari pembelajaran yang mendalam.

Menggabungkan Model

Kami sekarang memahami bagaimana kami dapat menentukan model linier untuk memperkirakan solvabilitas kredit pemohon, dan bagaimana kami dapat memodifikasi parameter model untuk menyesuaikan model ke kumpulan data. Namun, sebagai petugas bagian pinjaman, tugas kita tidak hanya menghitung solvabilitas kredit pemohon; kami harus memutuskan apakah akan mengabulkan permohonan pinjaman atau tidak. Dengan kata lain, kita membutuhkan aturan yang akan memberikan pujianskor solvabilitas sebagai masukan dan pengembalian keputusan atas pengajuan pinjaman. Misalnya, kita mungkin menggunakan aturan keputusan bahwa *seseorang dengan solvabilitas kredit di atas 200 akan diberikan pinjaman* . Aturan keputusan ini juga merupakan model: memetakan variabel input, dalam hal ini *solvabilitas kredit* , ke variabel output, *keputusan pinjaman* .

Dengan menggunakan aturan keputusan ini, kita dapat memutuskan permohonan pinjaman dengan terlebih dahulu menggunakan model solvabilitas kredit untuk mengubah profil pemohon pinjaman (dijelaskan dalam hal pendapatan dan utang tahunan) menjadi skor solvabilitas kredit, dan kemudian meneruskan skor solvabilitas kredit yang dihasilkan. melalui model aturan keputusan kami untuk menghasilkan keputusan pinjaman. Kita dapat menulis proses ini dengan singkatan pseudomathematical sebagai berikut:

$$\begin{aligned}&\text{loan decision}\\&\quad=\text{decision rule}(\text{solvency}=(\text{income}\times 2)+(\text{debt}\times 2))\end{aligned}$$

Dengan menggunakan notasi ini, seluruh proses keputusan untuk mengadili pengajuan pinjaman misalnya 1 dari tabel 2.1 adalah:

$$\begin{aligned}&\text{loan decision}\\&\quad=\text{decision rule}(\text{solvency}=(\text{income}\times 2)+(\text{debt}\times 2))\\&=\text{decision rule}(\text{solvency}=(150\times 2)+(-100\times 2))\\&=\text{decision rule}(\text{solvency}=100)\\&=\text{reject}\end{aligned}$$

Kami sekarang berada dalam posisi di mana kami dapat menggunakan model (terdiri dari dua model yang lebih sederhana, aturan keputusan dan jumlah tertimbang) untuk menggambarkan bagaimana keputusan pinjaman dibuat. Terlebih lagi, jika kita menggunakan data dari aplikasi pinjaman sebelumnya untuk mengatur parameter (yaitu, bobot) model, model kita akan sesuai dengan bagaimana kita memproses aplikasi pinjaman sebelumnya. Ini berguna karena kita dapat menggunakan model ini untuk memproses aplikasi baru dengan cara yang konsisten dengan keputusan sebelumnya. Jika pengajuan pinjaman baru diajukan, kami cukup menggunakan model kami untuk memproses aplikasi dan menghasilkan keputusan. Kemampuan untuk menerapkan model matematika ke contoh-contoh baru inilah yang membuat pemodelan matematika sangat berguna.

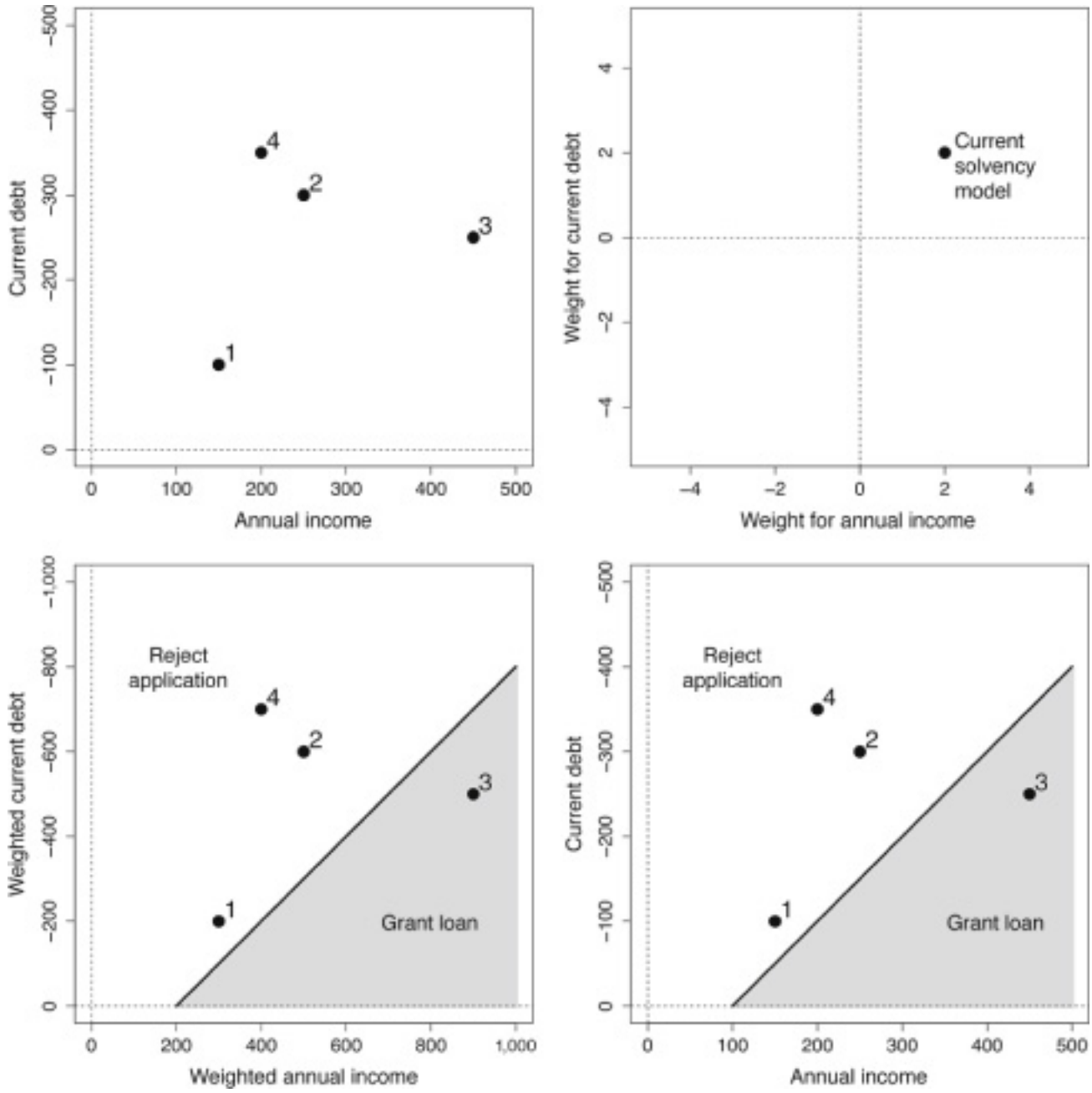
Saat kita menggunakan keluaran dari satu model sebagai masukan untuk model lain, kita membuat model ketiga dengan menggabungkan dua model. Strategi membangun model yang kompleks dengan menggabungkan model yang lebih kecil dan lebih sederhana ini merupakan inti dari jaringan pembelajaran yang mendalam. Seperti yang akan kita lihat, jaringan saraf terdiri dari sejumlah besar unit kecil yang disebut neuron. Masing-masing neuron ini adalah model sederhana dalam dirinya sendiri yang memetakan dari satu set input ke output. Model keseluruhan yang diterapkan oleh jaringan dibuat dengan memasukkan keluaran dari satu kelompok neuron sebagai masukan ke kelompok neuron kedua dan kemudian memberi masukan keluaran kelompok neuron kedua sebagai masukan ke kelompok neuron ketiga, begitu seterusnya, hingga hasil akhir model dihasilkan. Ide intinya adalah memberi makankeluaran dari beberapa neuron sebagai masukan ke neuron lain memungkinkan neuron berikutnya untuk belajar memecahkan bagian berbeda dari keseluruhan masalah yang coba dipecahkan jaringan dengan membangun solusi parsial yang diterapkan oleh neuron sebelumnya — dengan cara yang mirip dengan cara aturan keputusan menghasilkan putusan akhir untuk pengajuan pinjaman dengan membangun perhitungan model solvabilitas kredit. Kita akan kembali ke topik komposisi model ini di bab-bab selanjutnya.

Ruang Input, Ruang Bobot, dan Ruang Aktivasi

Meskipun model matematika dapat ditulis sebagai persamaan, sering kali berguna untuk memahami makna geometris dari suatu model. Misalnya, plot pada gambar 2.1 membantu kami memahami bagaimana perubahan parameter model linier mengubah hubungan antara variabel yang ditentukan model. Ada sejumlah ruang geometris yang berguna untuk dibedakan antara, dan dipahami, saat kita membahas jaringan saraf. Ini adalah ruang masukan, ruang bobot, dan ruang aktivasi neuron. Kita dapat menggunakan model keputusan untuk pengajuan pinjaman yang telah kita definisikan di bagian sebelumnya untuk menjelaskan ketiga jenis ruang yang berbeda ini.

Kami akan mulai dengan mendeskripsikan konsep ruang input. Model keputusan pinjaman kami mengambil dua masukan: thependapatan tahunan dan hutang pemohon saat ini. Tabel 2.1 mencantumkan nilai input ini untuk empat contoh aplikasi pinjaman. Kita dapat memplot ruang masukan model ini dengan memperlakukan setiap variabel masukan sebagai sumbu dari sistem koordinat. Ruang koordinat ini disebut sebagai ruang masukan karena setiap titik dalam ruang ini mendefinisikan kemungkinan kombinasi nilai masukan ke model. Misalnya, plot di kiri atas gambar 2.2 menunjukkan posisi masing-masing dari empat contoh aplikasi pinjaman dalam ruang input model.

Ruang bobot untuk model mendeskripsikan semesta dari kemungkinan kombinasi bobot yang mungkin digunakan model. Kita dapat memplot ruang bobot untuk suatu model dengan menentukan sistem koordinat dengan satu sumbu per bobot dalam model. Model keputusan pinjaman hanya memiliki dua bobot, satu bobot untuk input pendapatan tahunan, dan satu bobot untuk input utang saat ini. Alhasil, bobot ruang untuk model ini memiliki dua dimensi. Plot di kanan atas gambar 2.2 mengilustrasikan sebagian dari bobot ruang untuk model ini. Lokasi kombinasi bobot yang digunakan oleh model ^{2,2} disorot pada gambar ini. Setiap titik dalam sistem koordinat ini menggambarkan satu set bobot yang mungkin untuk model, dan oleh karena itu sesuai dengan fungsi penjumlahan tertimbang yang berbeda dalam model. Akibatnya, berpindah dari satu lokasi ke lokasi lain dalam ruang bobot ini setara dengan mengubah model karena mengubah pemetaan dari masukan ke keluaran yang ditentukan model.



Gambar 2.2 Ada empat ruang koordinat berbeda yang terkait dengan pemrosesan model keputusan pinjaman: ruang input di kiri atas plot; kanan atas memplot ruang bobot; kiri bawah memplot ruang aktivasi (atau keputusan); dan kanan bawah memplot ruang input dengan batas keputusan diplot.

Model linier memetakan sekumpulan nilai masukan ke titik di ruang baru dengan menerapkan penghitungan jumlah tertimbang ke masukan: kalikan setiap masukan dengan bobot, dan jumlahkan hasil perkalian. Dalam model keputusan pinjaman kami, di ruang inilah kami menerapkan aturan keputusan kami. Jadi, kita dapat menyebut ruang ini sebagai ruang keputusan, tetapi, untuk alasan yang akan menjadi jelas ketika kita mendeskripsikan struktur neuron di bab berikutnya, kita menyebut ruang ini sebagai ruang aktivasi. Sumbu ruang aktivasi model sesuai dengan input berbobot ke model. Akibatnya, setiap titik dalam ruang aktivasi mendefinisikan satu set input berbobot. Menerapkan aturan keputusan, seperti aturan kami bahwa *seseorang dengan solvabilitas kredit di atas 200 akan diberikan pinjaman*, ke setiap titik dalam ruang aktivasi ini, dan merekam hasil keputusan untuk setiap titik, memungkinkan kami untuk merencanakan batas keputusan model di ruang ini. Batas keputusan membagi titik-titik di ruang aktivasi yang melebihi ambang batas, dari titik-titik di ruang di bawah ambang

batas. Plot di kiri bawah gambar 2.2 mengilustrasikan ruang aktivasi untuk model keputusan pinjaman kami. Posisi dari empat contoh aplikasi pinjaman yang tercantum dalam tabel 2.1 saat diproyeksikan ke dalam ruang aktivasi ini ditampilkan. Garis hitam diagonal pada gambar ini menunjukkan batas keputusan. Dengan menggunakan ambang batas ini, permohonan pinjaman nomor tiga dikabulkan dan permohonan pinjaman lainnya ditolak. Kami dapat, jika ingin, memproyeksikan kembali batas keputusan ke ruang masukan asli denganmerekam untuk setiap lokasi di ruang masukan yang sisi mana dari batas keputusan dalam ruang aktivasi itu dipetakan oleh fungsi penjumlahan tertimbang. Plot di kanan bawah gambar 2.2 menunjukkan batas keputusan di ruang masukan asli (perhatikan perubahan nilai pada sumbu) dan dibuat menggunakan proses ini. Kami akan kembali ke konsep ruang bobot dan batas keputusan di bab berikutnya ketika kami menjelaskan bagaimana menyesuaikan parameter neuron mengubah himpunan kombinasi masukan yang menyebabkan neuron mengeluarkan aktivasi tinggi.

Ringkasan

Ide utama yang disajikan dalam bab ini adalah bahwa model matematika linier, baik itu dinyatakan sebagai persamaan atau diplot sebagai garis, menggambarkan hubungan antara sekumpulan input dan output. Ketahuilah bahwa tidak semua model matematika adalah model linier, dan kita akan menjumpai model nonlinier dalam buku ini. Namun, kalkulasi fundamental dari jumlah input yang dibobot menentukan model linier. Ide besar lainnya yang diperkenalkan dalam bab ini adalah bahwa model linier (penjumlahan terbobot) memiliki sekumpulan parameter, yaitu bobot yang digunakan dalam jumlah tertimbang. Dengan mengubah parameter ini kita dapat mengubah hubungan yang dijelaskan model antara input dan output. Jika ingin, kami dapat mengatur bobot ini dengan tangan menggunakan domain kamikeahlian; namun, kami juga dapat menggunakan pembelajaran mesin untuk menyetel bobot model sehingga perilaku model sesuai dengan pola yang ditemukan dalam kumpulan data. Ide besar terakhir yang diperkenalkan dalam bab ini adalah kita dapat membangun model yang kompleks dengan menggabungkan model yang lebih sederhana. Ini dilakukan dengan menggunakan keluaran dari satu (atau lebih) model sebagai masukan ke model lain. Kami menggunakan teknik ini untuk menentukan model gabungan kami untuk membuat keputusan pinjaman. Seperti yang akan kita lihat di bab selanjutnya, struktur neuron dalam jaringan saraf sangat mirip dengan struktur model keputusan pinjaman ini. Sama seperti model ini, neuron menghitung jumlah bobot inputnya dan kemudian memasukkan hasil perhitungan ini ke model kedua yang memutuskan apakah neuron aktif atau tidak.

Fokus bab ini adalah memperkenalkan beberapa konsep dasar sebelum kita memperkenalkan terminologi pembelajaran mesin dan pembelajaran mendalam. Untuk memberikan gambaran singkat tentang bagaimana konsep yang

diperkenalkan dalam bab ini dipetakan ke terminologi pembelajaran mesin, model keputusan pinjaman kami setara dengan neuron dua input yang menggunakan fungsi aktivasi ambang batas. Kedua indikator keuangan (pendapatan tahunan dan hutang saat ini) sejalan dengan input yang diterima neuron. Istilah vektor input atau vektor fitur kadang-kadang digunakan untuk merujuk pada serangkaian indikator yang menjelaskan satu contoh; dalam konteks ini contohnya adalah pemohon pinjaman tunggal, yang dijelaskan dalam dua fitur: pendapatan tahunan dan hutang lancar. Juga, sama sepertimodel keputusan pinjaman, neuron mengaitkan bobot dengan setiap masukan. Dan, sekali lagi, seperti model keputusan pinjaman, neuron mengalikan setiap masukan dengan bobot yang terkait dan menjumlahkan hasil perkalian ini untuk menghitung skor keseluruhan untuk masukan. Akhirnya, mirip dengan cara kami menerapkan ambang ke skor solvabilitas kredit untuk mengubahnya menjadi keputusan apakah akan mengabulkan atau menolak aplikasi pinjaman, neuron menerapkan fungsi (dikenal sebagai fungsi aktivasi) untuk mengubah skor keseluruhan dari masukan. Pada jenis neuron paling awal, fungsi aktivasi ini sebenarnya merupakan fungsi ambang batas yang bekerja persis sama dengan ambang batas skor yang digunakan dalam contoh penilaian kredit ini. Di jaringan saraf yang lebih baru, berbagai jenis fungsi aktivasi (misalnya, logistik, tanh, atau fungsi ULT) digunakan. Kami akan memperkenalkan fungsi aktivasi ini di bab berikutnya.

3

Jaringan Neural: Blok Bangunan Pembelajaran Mendalam

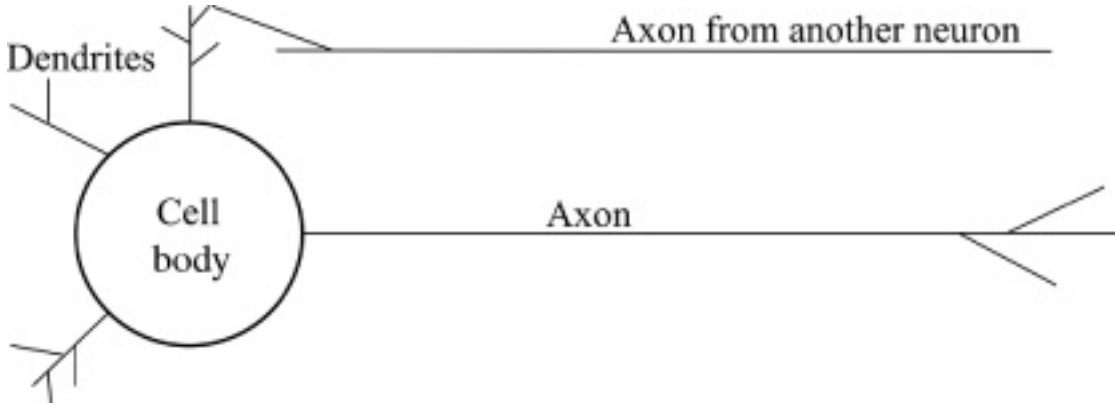
Istilah *pembelajaran mendalam* menggambarkan keluarga model *jaringan saraf* yang memiliki beberapa lapisan program pemrosesan informasi sederhana, yang dikenal sebagai neuron, dalam jaringan. Fokus bab ini adalah untuk memberikan pengantar yang jelas dan komprehensif tentang bagaimana neuron ini bekerja dan saling berhubungan dalam jaringan saraf tiruan. Di bab selanjutnya, kami akan menjelaskan bagaimana jaringan saraf dilatih menggunakan data.

Jaringan saraf adalah model komputasi yang terinspirasi oleh struktur otak manusia. Otak manusia terdiri dari sejumlah besar sel saraf, yang disebut neuron. Faktanya, beberapa perkiraan menyebutkan jumlah neuron di otak manusia mencapai seratus miliar (Herculano-Houzel 2009). Neuron memiliki struktur tiga bagian sederhana yang terdiri dari: badan sel, sekumpulan serat yang disebut dendrit, dan satu serat panjang yang disebut akson.Gambar 3.1 mengilustrasikan struktur sebuah neuron dan bagaimana ia terhubung ke neuron lain di otak. Dendrit dan batang akson dari badan sel, dan dendrit dari satu neuron terhubung ke akson neuron lain. Dendrit bertindak sebagai saluran masukan ke neuron dan menerima sinyal yang dikirim dari neuron lain di sepanjang aksonnya. Akson bertindak

sebagai saluran keluaran neuron, dan neuron lain, yang dendritnya terhubung ke akson, menerima sinyal yang dikirim sepanjang akson sebagai masukan.

Neuron bekerja dengan cara yang sangat sederhana. Jika rangsangan yang masuk cukup kuat, neuron mentransmisikan pulsa listrik, yang disebut potensial aksi, di sepanjang aksonnya ke neuron lain yang terhubung dengannya. Jadi, neuron bertindak sebagai sakelar semua atau tidak sama sekali, yang menerima satu set input dan mengeluarkan potensial aksi atau tidak ada output.

Penjelasan tentang otak manusia ini adalah penyederhanaan yang signifikan dari realitas biologis, tetapi penjelasannya dapat ditangkap poin utama yang diperlukan untuk memahami analogi antara struktur otak manusia dan model komputasi yang disebut jaringan saraf. Poin-poin analogi ini adalah: (1) otak terdiri dari sejumlah besar unit yang saling berhubungan dan sederhana yang disebut neuron; (2) fungsi otak dapat dipahami sebagai pemrosesan informasi, yang dikodekan sebagai sinyal listrik tinggi atau rendah, atau potensi aktivasi, yang menyebar ke seluruh jaringan neuron; dan (3) setiap neuron menerima satu set rangsangan dari tetangganya dan memetakan masukan ini ke keluaran bernilai tinggi atau rendah. Semua model komputasi jaringan saraf memiliki karakteristik ini.



Gambar 3.1 Struktur neuron di otak.

Jaringan Syaraf Tiruan

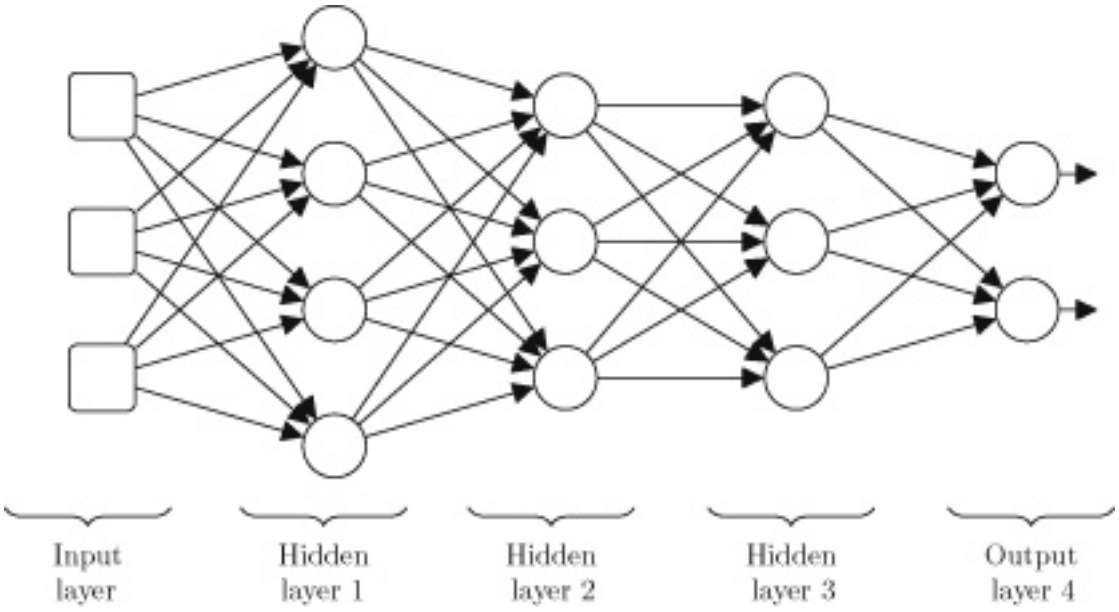
Jaringan saraf tiruan terdiri dari jaringan unit pemrosesan informasi sederhana, yang disebut neuron. Kekuatan jaringan saraf untuk memodelkan hubungan kompleks bukanlah hasil dari model matematika yang kompleks, melainkan muncul dari interaksi antara sekumpulan besar neuron sederhana.

Gambar 3.2 mengilustrasikan struktur jaringan saraf. Ini adalah standar untuk menganggap neuron dalam jaringan saraf diatur ke dalam lapisan. Jaringan yang digambarkan memiliki lima lapisan: satu lapisan masukan, tiga lapisan tersembunyi, dan satu lapisan keluaran. Lapisan tersembunyi hanyalah lapisan yang bukan keduanyalapisan masukan maupun keluaran. Jaringan pembelajaran dalam adalah jaringan saraf yang memiliki banyak lapisan neuron tersembunyi. Jumlah minimum hidden layer yang perlu dipertimbangkan dalam adalah dua. Namun, sebagian besar jaringan pembelajaran mendalam memiliki lebih dari dua

lapisan tersembunyi. Poin pentingnya adalah bahwa kedalaman jaringan diukur dengan jumlah lapisan tersembunyi, ditambah lapisan keluaran.

Jaringan pembelajaran dalam adalah jaringan saraf yang memiliki banyak lapisan neuron tersembunyi.

Pada gambar 3.2, kotak di lapisan masukan mewakili lokasi di memori yang digunakan untuk menyajikan masukan ke jaringan. Lokasi ini dapat dianggap sebagai neuron penginderaan. Tidak ada pemrosesan informasi dalam neuron penginderaan ini; keluaran dari masing-masing neuron ini hanyalah nilai dari data yang disimpan di lokasi memori.Lingkaran pada gambar mewakili neuron pemrosesan informasi di jaringan. Masing-masing neuron ini mengambil satu set nilai numerik sebagai masukan dan memetakannya ke satu nilai keluaran. Setiap input ke neuron pemrosesan adalah output dari neuron penginderaan atau output dari neuron pemrosesan lain.



Gambar 3.2 Ilustrasi topologi jaringan saraf sederhana.

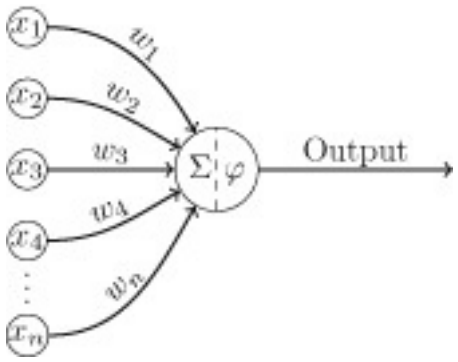
Panah pada gambar 3.2 mengilustrasikan bagaimana informasi mengalir melalui jaringan dari keluaran satu neuron ke masukan neuron lain. Setiap koneksi dalam jaringan menghubungkan dua neuron dan setiap koneksi diarahkan, yang berarti informasi yang dibawa sepanjang koneksi hanya mengalir dalam satu arah. Setiap koneksi dalam jaringan memiliki *bobot yang* terkait dengannya. Bobot koneksi hanyalah sebuah angka, tetapi bobot ini sangat penting. Bobot koneksi memengaruhi bagaimana neuron memproses informasi yang diterimanya sepanjang koneksi, dan, pada kenyataannya, melatih jaringan saraf tiruan, pada dasarnya, melibatkan pencarian set bobot terbaik (atau optimal).

Bagaimana Neuron Buatan Memproses Informasi

Pemrosesan informasi dalam neuron, yaitu pemetaan dari input ke output, sangat mirip dengan model keputusan pinjaman yang kami kembangkan di bab 2. Ingat bahwa model keputusan pinjaman pertama kali menghitung a jumlah tertimbang atas fitur input (pendapatan dan hutang). Bobot yang digunakan dalam penjumlahan tertimbang tersebut disesuaikan dengan menggunakan dataset sehingga hasil penghitungan jumlah tertimbang, dengan memberikan masukan pendapatan dan utang pemohon pinjaman, merupakan perkiraan yang akurat dari skor solvabilitas kredit pemohon. Proses tahap kedua dalam model keputusan pinjaman melibatkan penyampaian hasil perhitungan jumlah tertimbang (estimasi skor solvabilitas kredit) melalui aturan keputusan. Aturan keputusan ini merupakan fungsi yang memetakan skor solvabilitas kredit ke keputusan apakah pengajuan pinjaman dikabulkan atau ditolak.

Neuron juga mengimplementasikan proses dua tahap untuk memetakan input ke output. Tahap pertama pemrosesan melibatkan kalkulasi jumlah bobot masukan ke neuron. Kemudian hasil perhitungan jumlah tertimbang diteruskan melalui fungsi kedua yang memetakan hasil skor penjumlahan tertimbang ke nilai keluaran akhir neuron. Saat kita mendesain neuron, kita dapat menggunakan berbagai jenis fungsi untuk tahap atau pemrosesan kedua ini; mungkin sesederhana aturan keputusan yang kami gunakan untuk model keputusan pinjaman kami, atau mungkin lebih kompleks. Biasanya nilai keluaran neuron dikenal sebagai nilai aktivasi, sehingga fungsi kedua ini, yang memetakan dari hasil penjumlahan terbobot ke nilai aktivasi neuron, dikenal sebagai fungsi aktivasi.

Gambar 3.3 mengilustrasikan bagaimana tahapan pemrosesan ini tercermin dalam struktur neuron buatan. DiAngka 3.3, Σ simbol mewakili perhitungan jumlah tertimbang, dan φ simbol mewakili fungsi aktivasi memproses jumlah tertimbang dan menghasilkan output dari neuron.



Gambar 3.3 Struktur neuron buatan.

Neuron pada gambar 3.3 menerima *n* masukan $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ pada *n* sambungan masukan yang berbeda, dan setiap sambungan memiliki bobot yang terkait $[\mathbf{w}_1, \ldots, \mathbf{w}_n]$. Perhitungan jumlah tertimbang melibatkan perkalian input dengan bobot dan penjumlahan nilai yang dihasilkan. Secara matematis perhitungan ini ditulis sebagai:

$$z = (x_1 \times w_1) + (x_2 \times w_2) + \ldots + (x_n \times w_n)$$

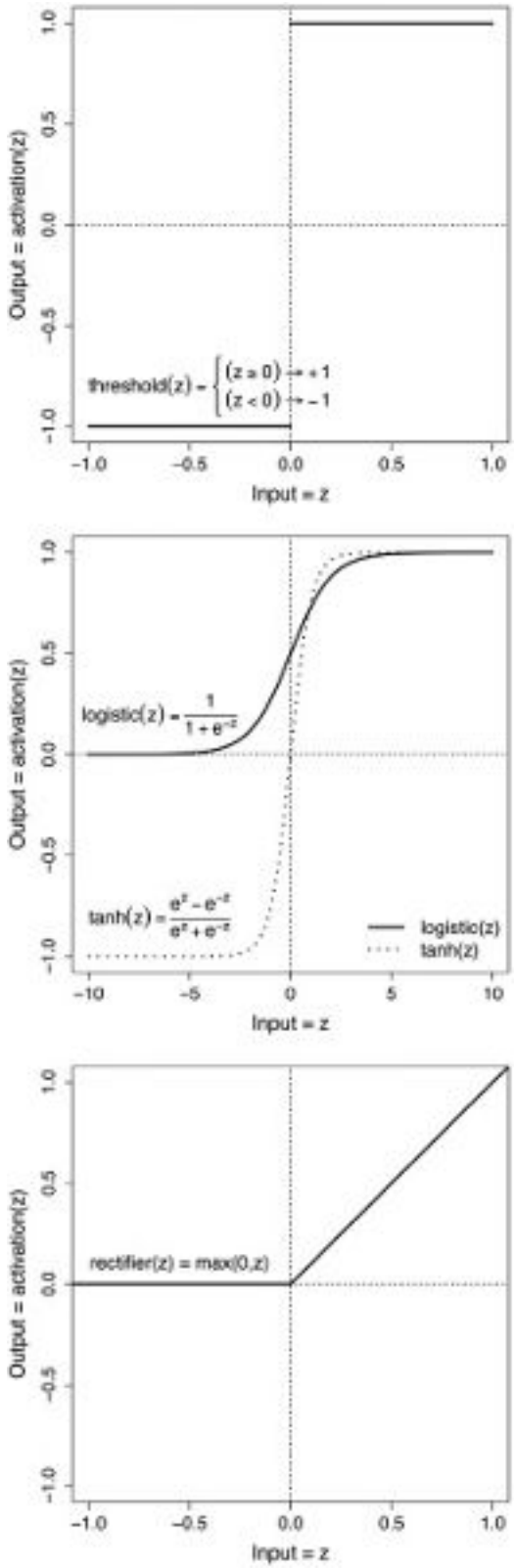
Perhitungan ini juga dapat ditulis dalam bentuk matematika yang lebih ringkas seperti:

$$z = \sum_{i=1}^{n} x_i \times w_i$$

Misalnya, dengan asumsi neuron menerima masukan $[x_1 = 3, x_2 = 9]$ dan memiliki bobot berikut $[w_1 = -3,\ w_2 = 1]$, perhitungan jumlah tertimbang adalah:

$$z = (3 \times -3) + (9 \times 1)$$
$$= 0$$

Tahap kedua dari pemrosesan dalam neuron adalah meneruskan hasil penjumlahan tertimbang, z nilainya, melalui *fungsi aktivasi* . Gambar 3.4 memplot bentuk sejumlah fungsi aktivasi yang mungkin, sebagai input untuk setiap fungsi, z, berkisar pada interval, baik [-1, ..., +1] atau [-10, ..., +10] tergantung pada interval mana yang paling menggambarkan bentuk fungsi. Gambar 3.4 (atas) memplot fungsi aktivasi ambang batas. Aturan keputusan yang kami gunakan dalam model keputusan pinjaman adalah contoh fungsi ambang batas; ambang batas yang digunakan dalam aturan keputusan itu adalah apakah skor solvabilitas kredit di atas 200. Aktivasi ambang umum terjadi dalam penelitian jaringan saraf awal. Gambar 3.4 (tengah) plot *logistik* dan *tanh*fungsi aktivasi. Unit yang menggunakan fungsi aktivasi ini populer di jaringan multilayer hingga baru-baru ini. Gambar 3.4 (bawah) memplot fungsi aktivasi penyearah (atau engsel, atau linier positif). Fungsi aktivasi ini sangat populer di jaringan pembelajaran mendalam modern; pada tahun 2011 fungsi aktivasi penyearah terbukti memungkinkan pelatihan yang lebih baik di jaringan dalam (Glorot et al. 2011). Faktanya,Seperti yang akan dibahas di bab 4, selama peninjauan sejarah pembelajaran mendalam, salah satu tren dalam penelitian jaringan saraf adalah pergeseran dari aktivasi ambang ke aktivasi logistik dan tanh, dan kemudian ke fungsi aktivasi penyearah.



Gambar 3.4 Atas: fungsi ambang batas; tengah: fungsi logistik dan tanh; bawah: fungsi linier yang diperbaiki.

Kembali ke contoh, hasil dari langkah penjumlahan berbobot adalah $z = 0$. Gambar 3.4 (plot tengah, garis penuh) menggambarkan fungsi logistik. Dengan asumsi bahwa neuron menggunakan fungsi aktivasi logistik, plot ini menunjukkan bagaimana hasil penjumlahan akan dipetakan ke aktivasi keluaran: $\text{logistic}(0) = 0.5$. Perhitungan aktivasi keluaran neuron ini dapat diringkas sebagai:

$$\text{Output} = \text{activation\_function}\left(z = \sum_{i=1}^{n} x_i \times w_i\right)$$
$$= \text{logistic}(z = (3 \times -3) + (9 \times 1))$$
$$= \text{logistic}(z = 0)$$
$$= 0.5$$

Perhatikan bahwa pemrosesan informasi di neuron ini hampir identik dengan pemrosesan informasi dalam model keputusan pinjaman yang kami kembangkan di bab sebelumnya. Perbedaan utamanya adalah bahwa kami telah mengganti aturan ambang batas keputusan yang memetakan skor jumlah tertimbang menjadi keluaran yang diterima atau ditolak dengan fungsi logistik yang memetakan skor jumlah tertimbang ke nilai antara 0 dan 1. Bergantung pada lokasi neuron ini di jaringan,aktivasi keluaran neuron, dalam hal ini $y = 0.5$, akan diteruskan sebagai masukan ke satu atau lebih neuron di lapisan berikutnya dalam jaringan, atau akan menjadi bagian dari keluaran keseluruhan jaringan. Jika sebuah neuron berada pada lapisan keluaran, interpretasi dari arti nilai keluarannya akan bergantung pada tugas yang dirancang untuk dimodelkan oleh neuron tersebut. Jika sebuah neuron berada di salah satu lapisan tersembunyi dari jaringan, maka tidak mungkin untuk menempatkan interpretasi yang berarti pada keluaran neuron selain dari interpretasi umum bahwa ia mewakili semacam fitur turunan (mirip dengan fitur BMI kita telah membahas dalam bab 1) bahwa jaringan telah menemukan manfaatnya dalam menghasilkan keluarannya. Kami akan kembali ke tantangan untuk menafsirkan makna aktivasi dalam jaringan saraf di bab 7.

Poin kunci yang perlu diingat dari bagian ini adalah bahwa neuron, blok bangunan fundamental dari jaringan saraf dan pembelajaran mendalam, ditentukan oleh urutan operasi dua langkah sederhana: menghitung jumlah tertimbang dan kemudian meneruskan hasilnya melalui fungsi aktivasi.

Gambar 3.4 mengilustrasikan bahwa baik tanh maupun fungsi logistik bukanlah fungsi linier. Faktanya, plot dari kedua fungsi ini memiliki profil berbentuk s (bukan linier) yang khas. Tidak semua fungsi aktivasi memiliki bentuk-s (misalnya, ambang batas dan penyearah tidak berbentuk s), tetapi semua fungsi aktivasi menerapkan pemetaan nonlinier ke keluaran jumlah berbobot. Faktanya,pengenalan pemetaan nonlinier ke dalam pemrosesan neuron itulah alasan mengapa fungsi aktivasi digunakan.

Mengapa Fungsi Aktivasi Diperlukan?

Untuk memahami mengapa pemetaan nonlinier diperlukan dalam neuron, pertama-tama perlu dipahami bahwa, pada dasarnya, semua jaringan saraf lakukan adalah menentukan pemetaan dari masukan ke keluaran, baik itu dari posisi permainan di

Go hingga evaluasi posisi itu. , atau dari X-ray hingga diagnosis pasien. Neuron adalah blok bangunan dasar jaringan saraf, dan oleh karena itu neuron adalah blok bangunan dasar dari pemetaan yang didefinisikan jaringan. Pemetaan keseluruhan dari input ke output yang didefinisikan jaringan terdiri dari pemetaan dari input ke output yang diimplementasikan oleh masing-masing neuron dalam jaringan. Implikasi dari hal ini adalah jika semua neuron dalam jaringan dibatasi untuk pemetaan linier (yaitu, penghitungan jumlah tertimbang), keseluruhan jaringan akan dibatasi pada pemetaan linier dari masukan ke keluaran. Namun, banyak hubungan di dunia yang mungkin ingin kita buat modelnya adalah nonlinier, dan jika kita mencoba memodelkan hubungan ini menggunakan model linier, model tersebut akan menjadi sangat tidak akurat. Mencoba memodelkan hubungan nonlinier dengan model linier akan menjadi contoh underfittingMasalah yang kita bahas di bab 1: underfitting terjadi ketika model yang digunakan untuk menyandikan pola dalam dataset terlalu sederhana dan akibatnya tidak akurat.

Hubungan linier terjadi antara dua hal ketika peningkatan satu selalu menghasilkan kenaikan atau penurunan yang lain dengan kecepatan konstan. Misalnya, jika seorang karyawan memiliki tarif per jam tetap, yang tidak bervariasi pada akhir pekan atau jika mereka melakukan lembur, maka ada hubungan linier antara jumlah jam mereka bekerja dan gaji mereka. Plot jam kerja mereka versus gaji mereka akan menghasilkan garis lurus; semakin curam garis, semakin tinggi tingkat gaji tetap mereka per jam. Namun, jika kita membuat sistem pembayaran untuk karyawan hipotetis kita sedikit lebih kompleks, dengan, misalnya, menaikkan tarif gaji per jam mereka saat mereka melakukan lembur atau bekerja di akhir pekan, maka hubungan antara jumlah jam kerja dan gaji mereka adalah tidak lagi linier. Jaringan saraf, dan khususnya jaringan pembelajaran mendalam, biasanya digunakan untuk memodelkan hubungan yang jauh lebih kompleks daripada gaji karyawan ini. Pemodelan hubungan ini secara akurat membutuhkan jaringan yang dapat mempelajari dan merepresentasikan pemetaan nonlinier yang kompleks. Jadi, untuk mengaktifkan jaringan saraf untuk mengimplementasikan pemetaan nonlinier, langkah nonlinier (fungsi aktivasi) harus disertakan dalam pemrosesan neuron di jaringan.

Pada prinsipnya, menggunakan fungsi nonlinier sebagai fungsi aktivasi memungkinkan jaringan saraf mempelajari nonlinier pemetaan dari masukan ke keluaran. Namun, seperti yang akan kita lihat nanti, sebagian besar fungsi aktivasi yang digambarkan dalam gambar 3.4 memiliki properti matematika bagus yang berguna saat melatih jaringan saraf, dan inilah mengapa fungsi tersebut sangat populer dalam penelitian jaringan saraf.

Fakta bahwa pengenalan nonlinier ke dalam pemrosesan neuron memungkinkan jaringan untuk mempelajari pemetaan nonlinier antara input dan output adalah ilustrasi lain dari fakta bahwa keseluruhan perilaku jaringan muncul dari interaksi pemrosesan yang dilakukan. keluar oleh neuron individu dalam jaringan. Jaringan neural memecahkan masalah menggunakan strategi divide-and-conquer: setiap neuron dalam jaringan memecahkan satu komponen dari masalah yang lebih besar, dan keseluruhan masalah diselesaikan dengan menggabungkan solusi komponen

ini. Aspek penting dari kekuatan jaringan saraf adalah bahwa selama pelatihan, karena bobot pada koneksi dalam jaringan ditetapkan, jaringan pada dasarnya mempelajari dekomposisi masalah yang lebih besar,

Dalam jaringan saraf, beberapa neuron mungkin menggunakan fungsi aktivasi yang berbeda dari neuron lain di jaringan. Secara umum, bagaimanapun, semua neuron dalam lapisan jaringan tertentu akan memiliki tipe yang sama (yaitu, mereka semua akan menggunakan fungsi aktivasi yang sama). Juga, terkadangneuron disebut sebagai unit, dengan perbedaan yang dibuat antara unit berdasarkan fungsi aktivasi yang digunakan unit: neuron yang menggunakan fungsi aktivasi ambang dikenal sebagai unit ambang, unit yang menggunakan fungsi aktivasi logistik dikenal sebagai unit logistik, dan neuron yang menggunakan fungsi aktivasi penyearah dikenal sebagai unit linier yang diperbaiki, atau ULT. Misalnya, jaringan mungkin memiliki lapisan ULT yang terhubung ke lapisan unit logistik. Keputusan mengenai fungsi aktivasi mana yang akan digunakan dalam neuron dalam jaringan dibuat oleh ilmuwan data yang merancang jaringan. Untuk membuat keputusan ini, data scientist dapat menjalankan sejumlah eksperimen untuk menguji fungsi aktivasi mana yang memberikan performa terbaik pada set data. Namun, sering kali data scientist default menggunakan fungsi aktivasi mana pun yang populer pada titik tertentu. Misalnya, saat ini ULT adalah jenis unit yang paling populer di jaringan neural, tetapi ini dapat berubah saat fungsi aktivasi baru dikembangkan dan diuji. Seperti yang akan kita bahas di akhir bab ini, elemen jaringan saraf yang diatur secara manual oleh data scientist sebelum proses pelatihan dikenal sebagai hyperparameter.

Jaringan saraf memecahkan masalah menggunakan strategi divide-and-conquer: setiap neuron dalam jaringan memecahkan satu komponen dari masalah yang lebih besar, dan keseluruhan masalah diselesaikan dengan menggabungkan solusi komponen ini.

Istilah hyperparameter digunakan untuk mendeskripsikan bagian model yang diperbaiki secara manual untuk membedakannya dari parameter model, yang merupakan bagian dari model yang diatur secara otomatis, oleh algoritme pembelajaran mesin, selama proses pelatihan. ItuParameter jaringan saraf adalah bobot yang digunakan dalam penghitungan jumlah bobot neuron di jaringan. Seperti yang telah kita bahas di bab 1 dan 2, proses pelatihan standar untuk menetapkan parameter jaringan saraf dimulai dengan menginisialisasi parameter (bobot jaringan) ke nilai acak, dan selama pelatihan menggunakan kinerja jaringan pada set data untuk menyesuaikan bobot ini secara perlahan untuk meningkatkan akurasi model pada data. Bab 6 menjelaskan dua algoritme yang paling umum digunakan untuk melatih jaringan saraf: algoritme penurunan gradien dan algoritme propagasi mundur. Apa yang akan kita fokuskan selanjutnya adalah memahami bagaimana mengubah parameter neuron memengaruhi bagaimana neuron merespons masukan yang diterimanya.

Bagaimana Mengubah Parameter Neuron Mempengaruhi Perilakunya?

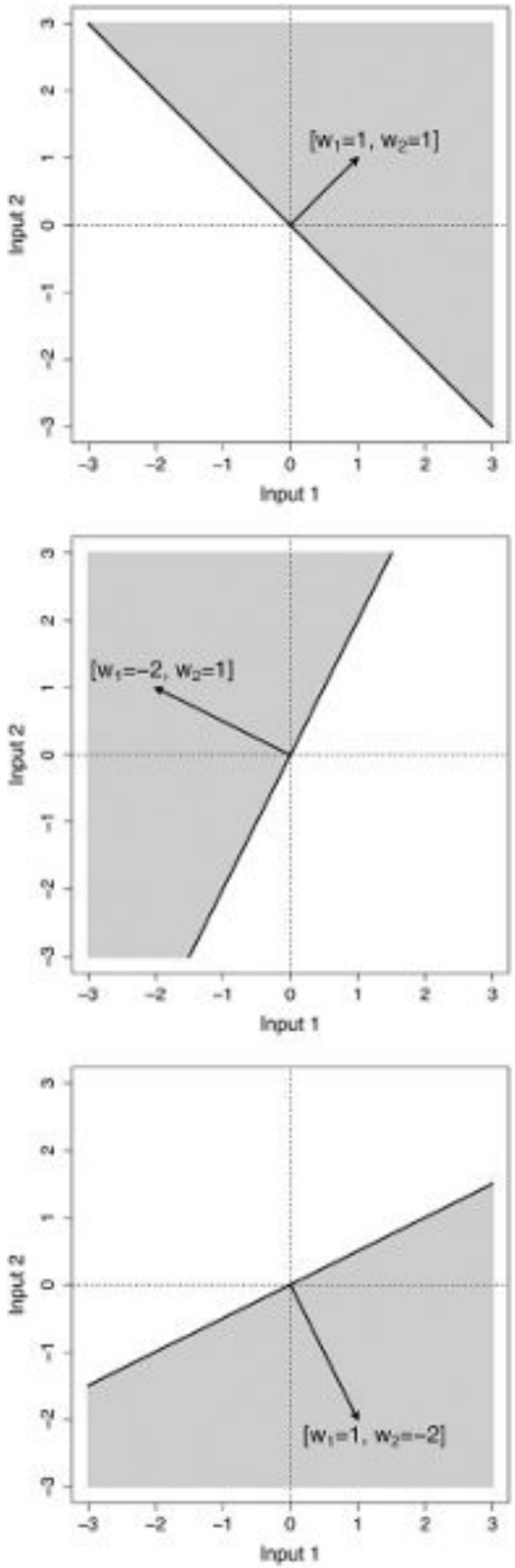
Parameter neuron adalah bobot yang digunakan neuron dalam penghitungan jumlah tertimbang. Meskipun penghitungan jumlah tertimbang dalam neuron adalah jumlah terbobot yang sama yang digunakan dalam model linier, dalam neuron hubungan antara bobot dan keluaran akhir neuron lebih kompleks karena hasil penjumlahan terbobot dilewatkan melalui fungsi aktivasi di memesan untukmenghasilkan hasil akhir. Untuk memahami bagaimana neuron membuat keputusan atas masukan yang diberikan, kita perlu memahami hubungan antara bobot neuron, masukan yang diterimanya, dan keluaran yang dihasilkannya sebagai respons.

Hubungan antara bobot neuron dan keluaran yang dihasilkannya untuk masukan tertentu paling mudah dipahami di neuron yang menggunakan fungsi aktivasi ambang batas. Sebuah neuron yang menggunakan jenis fungsi aktivasi ini setara dengan model keputusan pinjaman kami yang menggunakan aturan keputusan untuk mengklasifikasikan skor solvabilitas kredit, yang dihasilkan oleh penghitungan jumlah tertimbang, untuk menolak atau mengabulkan permohonan pinjaman. Pada akhir bab 2, kami memperkenalkan konsep ruang masukan, ruang bobot, dan ruang aktivasi (lihat gambar 2.2). Ruang masukan untuk model keputusan pinjaman dua masukan kami dapat divisualisasikan sebagai ruang dua dimensi, dengan satu masukan (pendapatan tahunan) diplot di sepanjang sumbu x, dan masukan lainnya (utang saat ini) pada sumbu y. Setiap titik dalam plot ini menentukan kombinasi input potensial ke model, dan himpunan titik dalam ruang masukan mendefinisikan himpunan masukan yang mungkin dapat diproses oleh model. Bobot yang digunakan dalam model keputusan pinjaman dapat dipahami dengan membagi ruang input menjadi dua wilayah: wilayah pertama berisi semua input yang mengakibatkan pengajuan pinjaman dikabulkan, dan wilayah lainnya berisi semua input yang menghasilkan pinjaman. aplikasi ditolak. Dalam skenario tersebut, mengubah bobot yang digunakan oleh model keputusan akan mengubah kumpulan aplikasi pinjaman dan daerah lain berisi semua masukan yang mengakibatkan pengajuan pinjaman ditolak. Dalam skenario tersebut, mengubah bobot yang digunakan oleh model keputusan akan mengubah kumpulan aplikasi pinjaman dan daerah lain berisi semua masukan yang mengakibatkan pengajuan pinjaman ditolak. Dalam skenario tersebut, mengubah bobot yang digunakan oleh model keputusan akan mengubah kumpulan aplikasi pinjamanyang diterima atau ditolak. Secara intuitif, ini masuk akal karena ini mengubah bobot yang kita berikan pada pendapatan pemohon relatif terhadap utangnya ketika kita memutuskan untuk memberikan pinjaman atau tidak.

Kita dapat menggeneralisasi analisis model keputusan pinjaman di atas ke neuron dalam jaringan saraf. Struktur neuron yang setara dengan model keputusan pinjaman adalah neuron dua masukan dengan fungsi aktivasi ambang batas. Ruang masukan untuk neuron semacam itu memiliki struktur yang mirip dengan ruang masukan untuk model keputusan pinjaman. Gambar 3.5 menyajikan tiga plot

ruang masukan untuk neuron dua masukan menggunakan fungsi ambang yang mengeluarkan aktivasi tinggi jika hasil penjumlahan terbobot lebih besar dari nol, dan sebaliknya aktivasi rendah. Perbedaan antara masing-masing plot pada gambar ini adalah bahwa neuron menentukan batas keputusan yang berbeda dalam setiap kasus. Di setiap plot, batas keputusan ditandai dengan garis hitam.

Masing-masing plot pada gambar 3.5 dibuat dengan terlebih dahulu memperbaiki bobot neuron dan kemudian untuk setiap titik di ruang masukan merekam apakah neuron mengembalikan aktivasi tinggi atau rendah ketika koordinat titik digunakan sebagai masukan ke neuron. . Titik masukan di mana neuron mengembalikan aktivasi tinggi diplot dengan warna abu-abu, dan titik lainnya diplot dengan warna putih. Satu-satunya perbedaan antara neuron yang digunakan untuk membuat plot ini adalah bobot yang digunakan dalam menghitung jumlah input yang dibobot. Panah di setiap plot menggambarkan bobotvektor yang digunakan oleh neuron untuk menghasilkan plot. Dalam konteks ini, vektor menggambarkan arah dan jarak suatu titik dari titik asal.[1] Seperti yang akan kita lihat, menafsirkan himpunan bobot yang digunakan oleh neuron sebagai mendefinisikan vektor (panah dari asal ke koordinat bobot) di ruang masukan neuron berguna dalam memahami bagaimana perubahan bobot mengubah batas keputusan dari neuron.



Gambar 3.5 Batas keputusan untuk neuron dua masukan. Atas: vektor bobot [w1 = 1, w2 = 1]; tengah: vektor berat [w1 = -2, w2 = 1]; bawah: vektor bobot [w1 = 1, w2 = -2].

Bobot yang digunakan untuk membuat setiap plot berubah dari satu plot ke plot berikutnya. Perubahan ini tercermin dalam arah panah (vektor bobot) di setiap plot. Secara khusus, mengubah bobot memutar vektor bobot di sekitar asalnya. Perhatikan bahwa batas keputusan di setiap plot peka terhadap arah vektor bobot: di semua plot, batas keputusan adalah ortogonal (yaitu, pada sudut kanan, atau 90 °) terhadap vektor bobot. Jadi, mengubah bobot tidak hanya memutar vektor bobot, tetapi juga memutar batas keputusan neuron. Rotasi ini mengubah

himpunan input yang dihasilkan neuron aktivasi tinggi sebagai respons (wilayah abu-abu).

Untuk memahami mengapa batas keputusan ini selalu ortogonal terhadap vektor bobot, kita harus menggeser perspektif kita, sejenak, ke aljabar linier. Ingatlah bahwa setiap titik dalam ruang masukan menentukan kombinasi potensial dari nilai masukan ke neuron. Sekarang, bayangkan setiap set nilai input ini sebagai mendefinisikan panah dari titik asal ke koordinat titik di ruang input.Ada satu panah untuk setiap titik di ruang masukan. Masing-masing panah ini sangat mirip dengan vektor bobot, hanya saja ia menunjuk ke koordinat input dan bukan ke koordinat bobot. Ketika kita memperlakukan sekumpulan input sebagai vektor, perhitungan jumlah tertimbang sama dengan mengalikan dua vektor, vektor input dengan vektor bobot. Dalam terminologi aljabar linier, mengalikan dua vektor dikenal sebagai operasi perkalian titik. Untuk keperluan pembahasan ini, yang perlu kita ketahui tentang perkalian titik adalah bahwa hasil dari operasi ini bergantung pada sudut antara dua vektor yang dikalikan. Jika sudut antara kedua vektor lebih kecil dari sudut siku-siku, maka hasilnya positif; jika tidak, hasilnya akan negatif. Begitu, mengalikan vektor bobot dengan vektor masukan akan mengembalikan nilai positif untuk semua vektor masukan pada sudut yang lebih kecil dari sudut siku-siku terhadap vektor bobot, dan nilai negatif untuk semua vektor lainnya. Fungsi aktivasi yang digunakan oleh neuron ini mengembalikan aktivasi tinggi ketika nilai positif dimasukkan dan aktivasi rendah ketika nilai negatif dimasukkan. Akibatnya, batas keputusan terletak pada sudut siku-siku terhadap vektor bobot karena semua masukan pada sudut yang kurang dari sudut siku-siku terhadap vektor bobot akan menghasilkan masukan positif ke fungsi aktivasi dan, oleh karena itu, memicu aktivasi keluaran tinggi. dari neuron; sebaliknya, semua masukan lainnya akan menghasilkan aktivasi keluaran rendah dari neuron. dan nilai negatif untuk semua vektor lainnya. Fungsi aktivasi yang digunakan oleh neuron ini mengembalikan aktivasi tinggi ketika nilai positif dimasukkan dan aktivasi rendah ketika nilai negatif dimasukkan. Akibatnya, batas keputusan terletak pada sudut siku-siku terhadap vektor bobot karena semua masukan pada sudut yang kurang dari sudut siku-siku terhadap vektor bobot akan menghasilkan masukan positif ke fungsi aktivasi dan, oleh karena itu, memicu aktivasi keluaran tinggi. dari neuron; sebaliknya, semua masukan lainnya akan menghasilkan aktivasi keluaran rendah dari neuron. dan nilai negatif untuk semua vektor lainnya. Fungsi aktivasi yang digunakan oleh neuron ini mengembalikan aktivasi tinggi ketika nilai positif dimasukkan dan aktivasi rendah ketika nilai negatif dimasukkan. Akibatnya, batas keputusan terletak pada sudut siku-siku terhadap vektor bobot karena semua masukan pada sudut yang kurang dari sudut siku-siku terhadap vektor bobot akan menghasilkan masukan positif ke fungsi aktivasi dan, oleh karena itu, memicu aktivasi keluaran tinggi. dari neuron; sebaliknya, semua masukan lainnya akan menghasilkan aktivasi keluaran rendah dari neuron. batas keputusan terletak pada sudut siku-siku ke vektor bobot karena semua masukan pada sudut kurang dari sudut siku-siku ke vektor bobot akan menghasilkan masukan positif ke fungsi aktivasi dan, oleh karena itu, memicu aktivasi keluaran tinggi dari neuron; sebaliknya, semua masukan lainnya akan

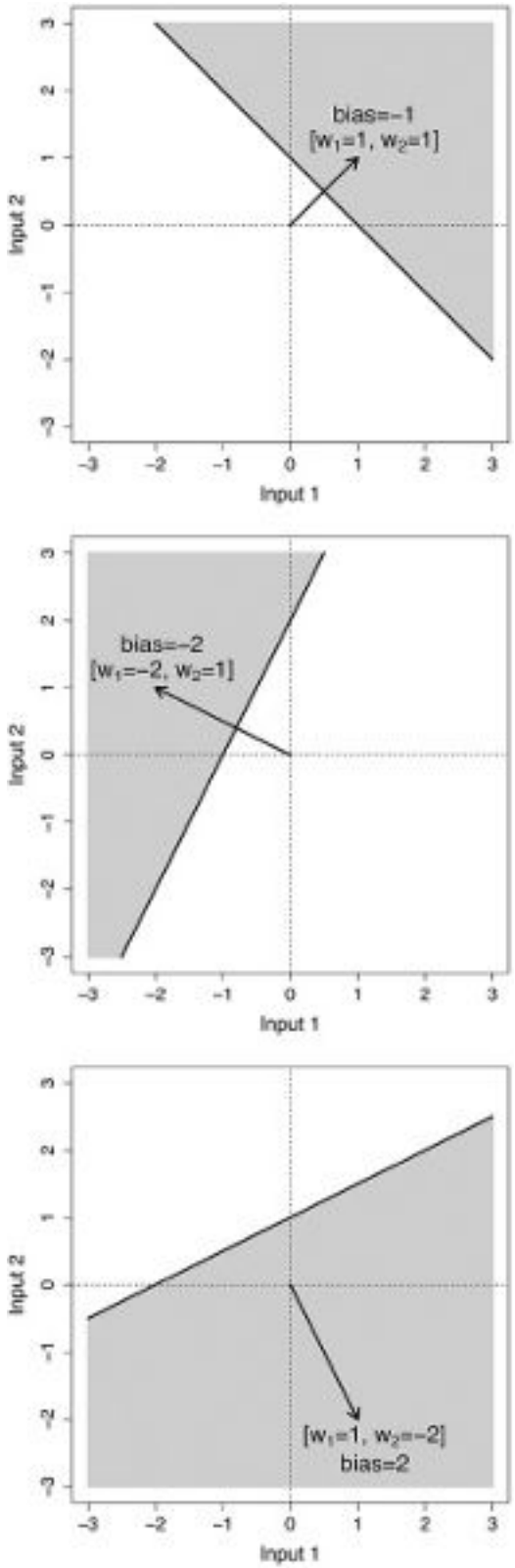
menghasilkan aktivasi keluaran rendah dari neuron. batas keputusan terletak pada sudut siku-siku terhadap vektor bobot karena semua masukan pada sudut kurang dari sudut siku-siku terhadap vektor bobot akan menghasilkan masukan positif ke fungsi aktivasi dan, oleh karena itu, memicu aktivasi keluaran tinggi dari neuron; sebaliknya, semua masukan lainnya akan menghasilkan aktivasi keluaran rendah dari neuron.

Beralih kembali ke plot pada gambar 3.5, meskipun batas keputusan di masing-masing plot berada pada sudut yang berbeda, semua batas keputusan melewati titik di ruang tempat vektor bobot berasal (yaitu, asal). Ini menggambarkan bahwa mengubah bobot neuron memutar batas keputusan neuron tetapi tidak menerjemahkannya. Menerjemahkan batas keputusan berarti memindahkan batas keputusan ke atas dan ke bawah bobot vektor, sehingga titik pertemuannya dengan vektor bukanlah titik asal. Pembatasan bahwa semua batas keputusan harus melewati asal membatasi perbedaan yang dapat dipelajari neuron di antara pola masukan. Cara standar untuk mengatasi batasan ini adalah dengan memperpanjang penghitungan jumlah tertimbang sehingga menyertakan elemen tambahan, yang dikenal sebagai *istilah bias*.. Istilah bias ini tidak sama dengan bias induktif yang kita bahas di bab 1. Ini lebih analog dengan parameter intersep dalam persamaan garis, yang menggerakkan garis ke atas dan ke bawah sumbu y. Tujuan istilah bias ini adalah untuk memindahkan (atau menerjemahkan) batasan keputusan menjauh dari asalnya.

Istilah bias hanyalah nilai tambahan yang dimasukkan dalam perhitungan jumlah tertimbang. Itu dimasukkan ke dalam neuron dengan menambahkan bias ke hasil penjumlahan tertimbang sebelum meneruskannya melalui fungsi aktivasi. Berikut adalah persamaan yang menjelaskan tahapan pemrosesan dalam neuron dengan istilah bias yang diwakili oleh istilah tersebut b:

$$\text{Output} = \text{activation\_function}\left(z = \underbrace{\left(\sum_{i=1}^{n} x_i \times w_i \right)}_{\text{weighted sum}} + \underbrace{b}_{\text{bias}} \right)$$

Gambar 3.6 mengilustrasikan bagaimana nilai istilah bias mempengaruhi batas keputusan neuron. Ketika istilah bias negatif, batas keputusan dipindahkan dari asalnya ke arah yang ditunjuk vektor bobot (seperti pada plot atas dan tengah pada gambar 3.6); ketika istilah biasnya positif, batas keputusan diterjemahkan ke arah yang berlawanan (lihat plot bawah gambar 3.6). Dalam kedua kasus, batas keputusan tetap ortogonal terhadap vektor bobot. Juga, ukuran istilah bias mempengaruhi jumlah batas keputusan dipindahkan dari asalnya; semakin besar nilai istilah biasnya, semakin banyak pula batas keputusan yang dipindahkan (bandingkan plot atas pada gambar 3.6 dengan plot tengah dan bawah).



Gambar 3.6 Plot batas keputusan untuk neuron dua masukan yang menggambarkan efek istilah bias pada batas keputusan. Atas: vektor bobot [w1 = 1, w2 = 1] dan bias sama dengan -1; tengah: vektor bobot [w1 = -2, w2 = 1] dan bias sama dengan -2; bawah: vektor bobot [w1 = 1, w2 = -2] dan bias sama dengan 2.

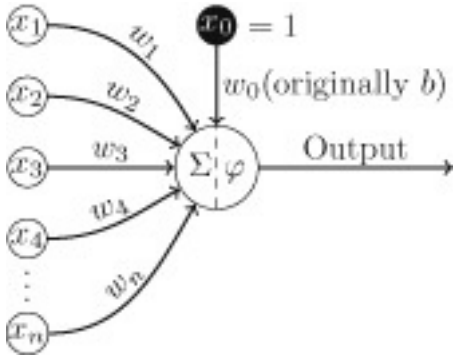
Alih-alih mengatur nilai istilah bias secara manual, lebih disukai membiarkan neuron mempelajari bias yang sesuai. Cara paling sederhana untuk melakukan ini adalah dengan memperlakukan istilah bias sebagai bobot dan memungkinkan neuron mempelajari istilah bias pada saat yang sama mempelajari bobot lainnya untuk inputnya. Semua yang diperlukan untuk mencapai ini adalah untuk menambah semua vektor input yang diterima neuron dengan input tambahan yang

selalu disetel ke 1. Menurut konvensi, input ini adalah input 0 ($x_0 = 1$), dan akibatnya, istilah bias ditentukan olehberat 0 (w_0). [2] Gambar 3.7 mengilustrasikan struktur neuron buatan ketika istilah bias telah diintegrasikan sebagai w_0.

Ketika istilah bias telah diintegrasikan ke dalam bobot neuron, persamaan yang menentukan pemetaan dari input ke aktivasi neuron dapat disederhanakan (setidaknya dari perspektif notasi) sebagai berikut:

$$\text{Output} = \text{activation\_function}\left(z = \sum_{i=0}^{n} x_i \times w_i \right)$$

Perhatikan bahwa dalam persamaan ini indeks i pergi dari 0 ke n, sehingga sekarang termasuk input tetap, $x_0 = 1$ dan istilah bias w_0; pada versi awal persamaan ini, indeks hanya berubah dari 1 menjadi n. Format baru ini berarti bahwa neuron dapat mempelajari istilah bias, hanya dengan mempelajari bobot yang sesuai w_0, menggunakan proses yang samayang digunakan untuk mempelajari bobot untuk input lain: pada awal pelatihan, istilah bias untuk setiap neuron dalam jaringan akan diinisialisasi ke nilai acak dan kemudian disesuaikan, bersama dengan bobot jaringan, sebagai respons terhadap kinerja jaringan pada dataset.



Gambar 3.7 Neuron buatan dengan istilah bias termasuk sebagai w_0.

Mempercepat Pelatihan Jaringan Neural Menggunakan GPU

Menggabungkan istilah bias lebih dari sekadar kenyamanan notasional; ini memungkinkan kita menggunakan perangkat keras khusus untuk mempercepat pelatihan jaringan saraf. Fakta bahwa suku bias dapat diperlakukan sama dengan bobot berarti bahwa perhitungan jumlah bobot input (termasuk penambahan suku

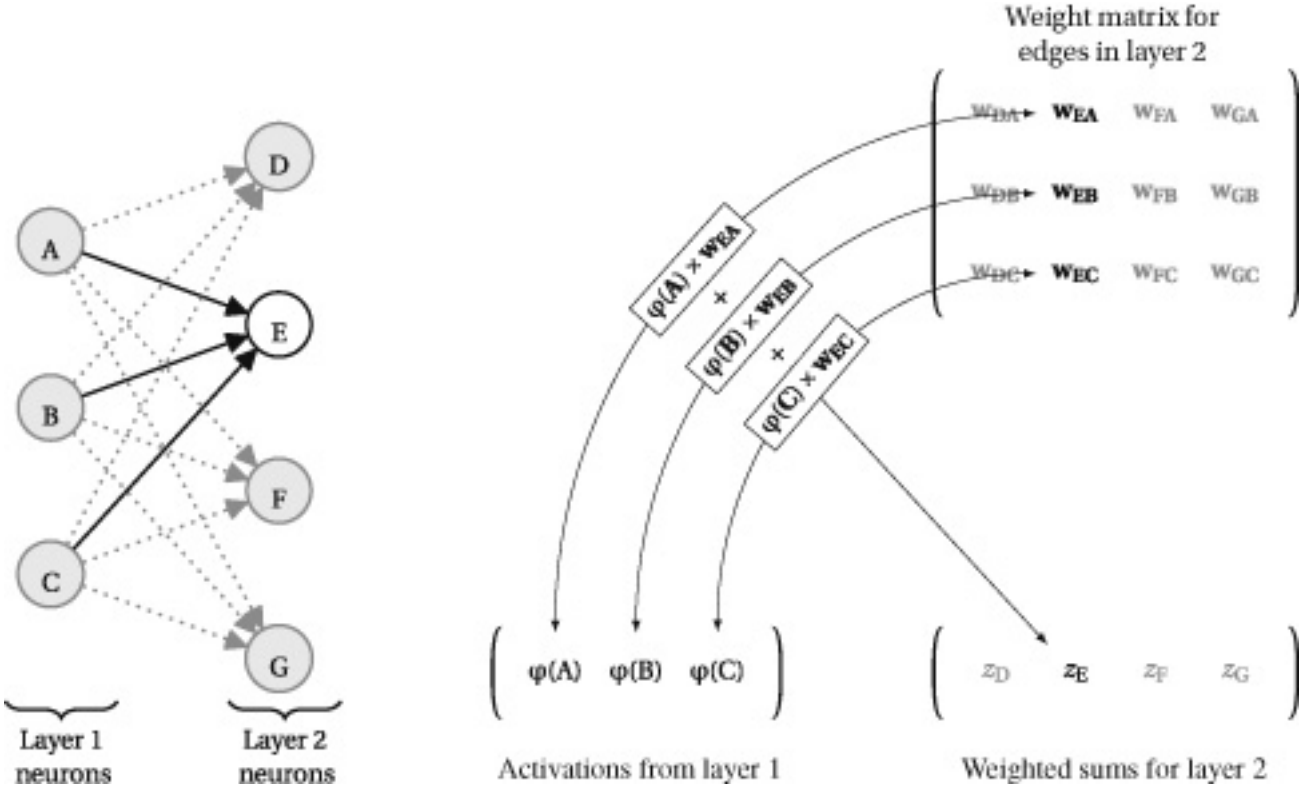
bias) dapat diperlakukan sebagai perkalian dua vektor. Seperti yang telah kita bahas sebelumnya, selama penjelasan mengapa batas keputusan ortogonal terhadap vektor bobot, kita dapat menganggap himpunan input sebagai vektor. Menyadari bahwa sebagian besar pemrosesan dalam jaringan saraf melibatkan perkalian vektor dan matriks, membuka kemungkinan menggunakan perangkat keras khusus untuk mempercepat penghitungan ini. Sebagai contoh,

Dalam jaringan feedforward standar, semua neuron dalam satu lapisan menerima semua keluaran (yaitu, aktivasi) dari semua neuron di lapisan sebelumnya. Ini berarti bahwa semua neuron dalam sebuah lapisan menerima masukan yang sama. Hasilnya, kita dapat menghitung kalkulasi jumlah tertimbang untuk semua neuron dalam sebuah lapisan hanya dengan menggunakan satu vektor dengan perkalian matriks. Melakukan ini jauh lebih cepat daripada menghitung jumlah tertimbang yang terpisah untuk setiap neuron di lapisan. Untuk melakukan penghitungan jumlah tertimbang ini untuk seluruh lapisan neuron dalam satu perkalian, kami menempatkan keluaran dari neuron di lapisan sebelumnya ke dalam vektor dan menyimpan semua bobot hubungan antara dua lapisan neuron dalam matriks. Kami kemudian mengalikan vektor dengan matriks, dan vektor yang dihasilkan berisi jumlah tertimbang untuk semua neuron.

Gambar 3.8 mengilustrasikan bagaimana penghitungan penjumlahan berbobot untuk semua neuron dalam sebuah lapisan dalam jaringan dapat dihitung menggunakan operasi perkalian matriks tunggal. Gambar ini terdiri dari dua grafik terpisah: grafik di sebelah kiri menggambarkan hubungan antara neuron dalam dua lapisan jaringan, dan grafik di sebelah kanan menggambarkan operasi matriks untuk menghitung jumlah bobot untuk neuron di lapisan kedua dari jaringan. Untuk membantu menjaga korespondensi antara dua grafik, koneksi ke neuron E disorot dalam grafik di sebelah kiri, dan kalkulasi jumlah tertimbang di neuron E disorot dalam grafik di sebelah kanan.

Berfokus pada grafik di sebelah kanan, 1×3 vektor (1 baris, 3 kolom) di kiri bawah grafik ini, menyimpan aktivasi untuk neuron di lapisan 1 jaringan; perhatikan bahwa aktivasi ini adalah keluaran dari fungsi aktivasi φ (fungsi aktivasi tertentu tidak ditentukan — bisa jadi fungsi ambang batas, tanh, fungsi logistik, atau unit linier yang diperbaiki / fungsi ULT). The 3×4 matriks (tiga baris dan empat kolom), di kanan atas grafik, memegang bobot untuk koneksi antara dua lapisan neuron. Dalam matriks ini, setiap kolom menyimpan bobot untuk koneksi yang masuk ke salah satu neuron di lapisan kedua jaringan. Kolom pertama menyimpan bobot untuk neuron D, kolom kedua untuk neuron E, dll.[3] Mengalikan 1×3 vektor aktivasi dari lapisan 1 dengan 3×4 matriks bobot menghasilkan 1×4 vektor yang sesuai dengan penjumlahan terbobot untuk empat neuron di lapisan 2 jaringan: z_D adalah jumlah masukan terbobot untuk neuron D, z_E untuk neuron E, dan seterusnya.

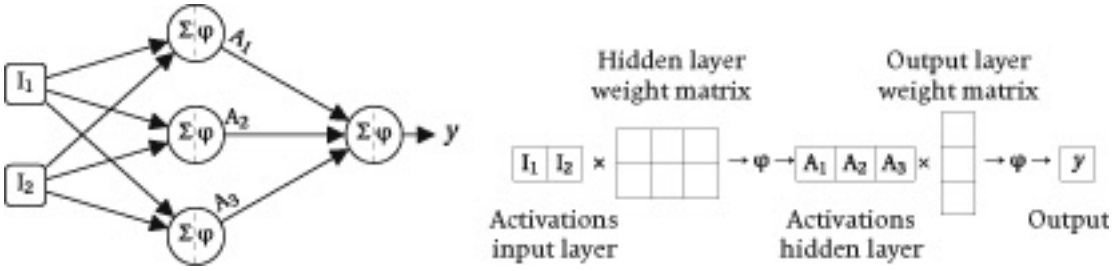
Untuk menghasilkan 1×4 vektor yang berisi penjumlahan berbobot untuk neuron di lapisan 2, vektor aktivasi dikalikan dengan setiap kolom dalam matriks secara bergantian. Ini dilakukan dengan mengalikan elemen pertama (paling kiri) dalam vektor dengan elemen pertama (paling atas) di kolom, kemudian mengalikan elemen kedua dalam vektor dengan elemen di baris kedua di kolom, dan seterusnya, hingga masing-masing elemen dalam vektor telah dikalikan dengan nyaelemen kolom yang sesuai. Setelah semua perkalian antara vektor dan kolom diselesaikan, hasilnya dijumlahkan dan disimpan dalam vektor keluaran. Gambar 3.8 mengilustrasikan perkalian vektor aktivasi dengan kolom kedua dalam matriks bobot (kolom yang berisi bobot untuk masukan ke neuron E) dan penyimpanan hasil penjumlahan perkalian ini dalam vektor keluaran sebagai nilainya z_E.



Gambar 3.8 Ilustrasi grafis dari koneksi topologi dari neuron E tertentu dalam jaringan, dan vektor yang sesuai dengan perkalian matriks yang menghitung penjumlahan berbobot dari input untuk neuron E, dan saudara kandungnya di lapisan yang sama.[5]

Memang, kalkulasi yang diterapkan oleh seluruh jaringan saraf dapat direpresentasikan sebagai rangkaian perkalian matriks, dengan penerapan fungsi aktivasi berdasarkan elemen ke hasil setiap perkalian. Gambar 3.9 mengilustrasikan bagaimana jaringan saraf dapat direpresentasikan dalam bentuk grafik (di sebelah kiri) dan sebagai urutan operasi matriks (di sebelah kanan). Dalam representasi matriks, $\times$ simbol merepresentasikan perkalian matriks standar (dijelaskan di atas) dan $\rightarrow \varphi \rightarrow$ Notasi merepresentasikan penerapan fungsi aktivasi ke setiap elemen dalam vektor yang dibuat oleh perkalian matriks sebelumnya. Keluaran dari penerapan fungsi aktivasi yang bijaksana elemen ini adalah vektor yang berisi aktivasi untuk neuron di lapisan jaringan. Untuk

membantu menunjukkan korespondensi antara dua representasi, baik angka menunjukkan input ke jaringan, I_1 dan I_2, yang aktivasi dari tiga unit tersembunyi, A_1, A_2, dan A_3, dan output keseluruhan jaringan, y.



Gambar 3.9 Sebuah representasi grafik dari jaringan syaraf (kiri), dan jaringan yang sama direpresentasikan sebagai urutan operasi matriks (kanan).[6]

Sebagai catatan tambahan, representasi matriks memberikan pandangan transparan tentang kedalaman jaringan; kedalaman jaringan dihitung sebagai jumlah lapisan yang memiliki matriks bobot yang terkait dengannya (atau setara, kedalaman jaringan adalah jumlah matriks bobot yang dibutuhkan oleh jaringan). Inilah sebabnya mengapa lapisan masukan tidak dihitung saat menghitung kedalaman jaringan: ia tidak memiliki matriks bobot yang terkait dengannya.

Seperti disebutkan di atas, fakta bahwa sebagian besar kalkulasi dalam jaringan neural dapat direpresentasikan sebagai urutan operasi matriks memiliki implikasi komputasi yang penting untuk pembelajaran yang mendalam. Jaringan saraf dapat berisi lebih dari satu juta neuron, dan tren saat ini adalah ukuran jaringan ini berlipat ganda setiap dua hingga tiga tahun.[4] Lebih lanjut, jaringan pembelajaran dalam dilatih dengan menjalankan jaringan secara berulang pada contoh-contoh yang diambil dari kumpulan data yang sangat besar dan kemudian memperbarui parameter jaringan (yaitu, bobot) untuk meningkatkan kinerja. Akibatnya, melatih jaringan pembelajaran yang dalam dapat memerlukan sejumlah besar jaringan berjalan, dengan setiap jaringan yang dijalankan memerlukan jutaan perhitungan. Inilah mengapa percepatan komputasi, seperti yang dapat dicapai dengan menggunakan GPU untuk melakukan perkalian matriks, menjadi sangat penting untuk pengembangan pembelajaran yang mendalam.

Hubungan antara GPU dan pembelajaran mendalam bukanlah satu arah. Pertumbuhan permintaan GPU yang dihasilkan oleh pembelajaran mendalam memiliki dampak yang signifikanProdusen GPU. Pembelajaran yang mendalam telah mengakibatkan perusahaan-perusahaan ini memfokuskan kembali bisnis mereka. Secara tradisional, perusahaan-perusahaan ini akan berfokus pada pasar game komputer, karena motivasi asli untuk mengembangkan chip GPU adalah untuk meningkatkan rendering grafis, dan ini merupakan aplikasi alami untuk game komputer. Namun, dalam beberapa tahun terakhir, perusahaan-perusahaan ini berfokus pada memposisikan GPU sebagai perangkat keras untuk pembelajaran mendalam dan aplikasi kecerdasan buatan. Selain itu, perusahaan GPU juga

berinvestasi untuk memastikan bahwa produk mereka mendukung kerangka kerja perangkat lunak pembelajaran mendalam teratas.

Ringkasan

Tema utama dalam bab ini adalah bahwa jaringan pembelajaran dalam terdiri dari sejumlah besar unit pemrosesan sederhana yang bekerja sama untuk mempelajari dan menerapkan pemetaan kompleks dari kumpulan data besar. Unit sederhana ini, neuron, menjalankan proses dua tahap: pertama, penjumlahan berbobot atas input ke neuron dihitung, dan kedua, hasil penjumlahan berbobot diteruskan melalui fungsi nonlinier, yang dikenal sebagai fungsi aktivasi. Fakta bahwa fungsi penjumlahan berbobot dapat dihitung secara efisien di seluruh lapisan neuron menggunakan operasi perkalian matriks tunggal adalah penting: ini berarti bahwa jaringan saraf dapat dipahami sebagai aurutan operasi matriks; hal ini memungkinkan penggunaan GPU, perangkat keras yang dioptimalkan untuk melakukan perkalian matriks dengan cepat, untuk mempercepat pelatihan jaringan, yang pada gilirannya memungkinkan ukuran jaringan untuk berkembang.

Sifat komposisi jaringan saraf berarti bahwa dimungkinkan untuk memahami pada tingkat yang sangat mendasar bagaimana jaringan saraf beroperasi. Memberikan gambaran yang komprehensif tentang tingkat pemrosesan ini telah menjadi fokus bab ini. Namun, sifat komposisi jaringan saraf juga menimbulkan pertanyaan yang berkaitan dengan bagaimana jaringan harus disusun untuk menyelesaikan tugas yang diberikan, misalnya:

Fungsi aktivasi apa yang harus digunakan neuron dalam jaringan?
Berapa banyak lapisan yang harus ada dalam jaringan?
Berapa banyak neuron yang harus ada di setiap lapisan?
Bagaimana neuron dihubungkan bersama?

Sayangnya, banyak dari pertanyaan-pertanyaan ini tidak dapat dijawab pada tingkat prinsip murni. Dalam terminologi pembelajaran mesin, jenis konsep yang dibahas dalam pertanyaan ini dikenal sebagai hyperparameter, yang berbeda dari parameter model. Parameter jaringan neural adalah bobot di tepinya, dan ini ditetapkan oleh pelatihanjaringan menggunakan kumpulan data besar. Sebaliknya, hyperparameter adalah parameter model (dalam kasus ini, parameter arsitektur jaringan neural) dan / atau algoritme pelatihan yang tidak dapat diperkirakan secara langsung dari data tetapi harus ditentukan oleh orang yang membuat model, baik melalui penggunaan aturan heuristik, intuisi, atau trial and error. Seringkali, banyak upaya yang dilakukan untuk menciptakan jaringan pembelajaran yang dalam melibatkan pekerjaan eksperimental untuk menjawab pertanyaan yang berkaitan dengan hyperparameter, dan proses ini dikenal sebagai penyetelan hyperparameter. Bab berikutnya akan mengulas sejarah dan evolusi pembelajaran mendalam, dan tantangan yang ditimbulkan oleh banyak pertanyaan ini adalah

tema-tema yang berjalan melalui tinjauan tersebut. Bab-bab selanjutnya dalam buku ini akan mengeksplorasi bagaimana menjawab pertanyaan-pertanyaan ini dengan cara yang berbeda dapat menciptakan jaringan dengan karakteristik yang sangat berbeda, masing-masing sesuai untuk jenis tugas yang berbeda. Misalnya, jaringan saraf berulang paling cocok untuk memproses data sekuensial / deret waktu, sedangkan jaringan saraf konvolusional awalnya dikembangkan untuk memproses gambar. Kedua jenis jaringan ini, bagaimanapun, dibangun menggunakan unit pemrosesan dasar yang sama, neuron buatan; perbedaan dalam perilaku dan kemampuan jaringan ini berasal dari bagaimana neuron ini disusun dan disusun. sedangkan jaringan saraf konvolusional pada awalnya dikembangkan untuk memproses gambar. Kedua jenis jaringan ini, bagaimanapun, dibangun menggunakan unit pemrosesan dasar yang sama, neuron buatan; perbedaan dalam perilaku dan kemampuan jaringan ini berasal dari bagaimana neuron ini disusun dan disusun. sedangkan jaringan saraf konvolusional pada awalnya dikembangkan untuk memproses gambar. Kedua jenis jaringan ini, bagaimanapun, dibangun menggunakan unit pemrosesan dasar yang sama, neuron buatan; perbedaan dalam perilaku dan kemampuan jaringan ini berasal dari bagaimana neuron ini disusun dan disusun.

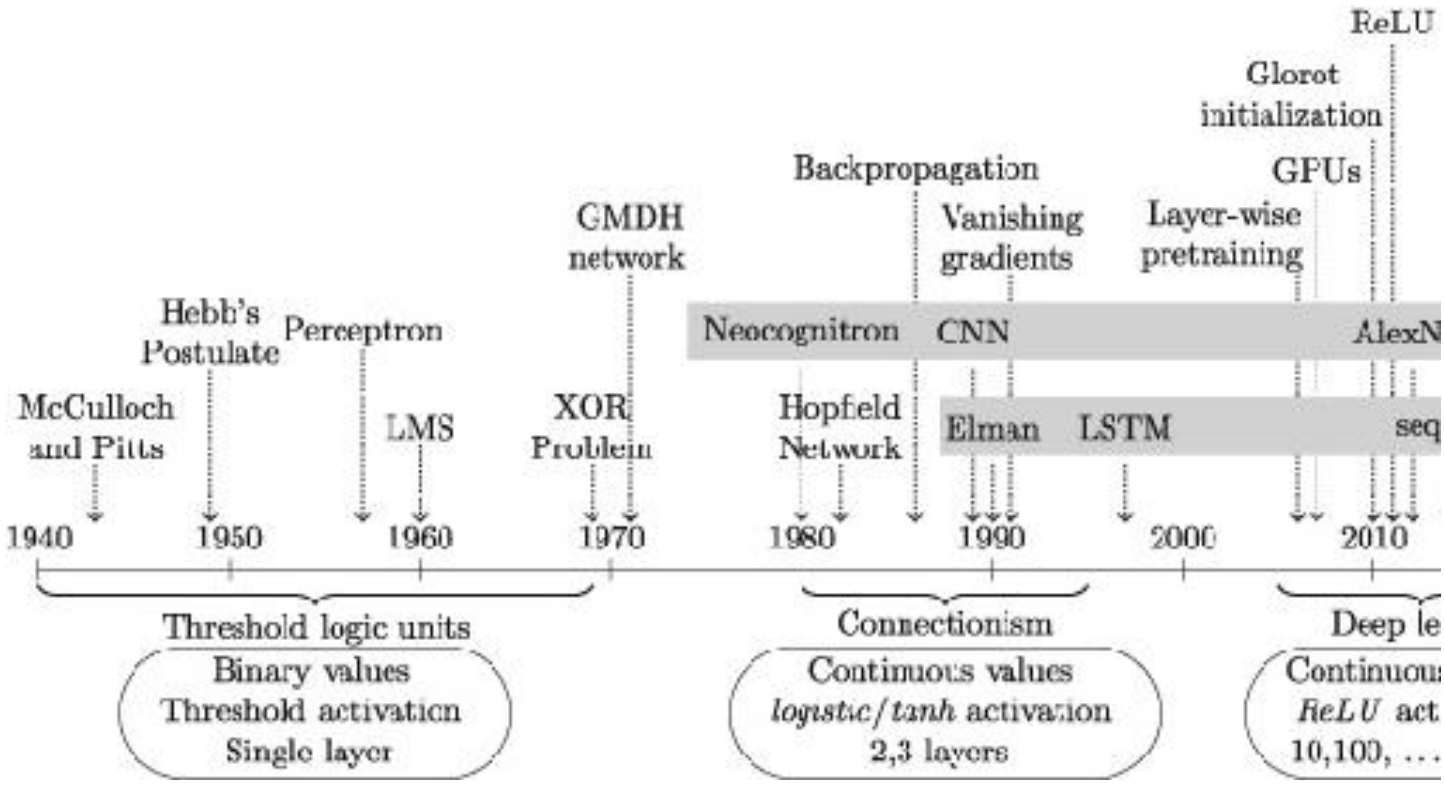
4

Sejarah Singkat Pembelajaran Mendalam

Sejarah pembelajaran mendalam dapat digambarkan sebagai tiga periode utama kegembiraan dan inovasi, diselingi dengan periode kekecewaan. Gambar 4.1 menunjukkan garis waktu dari sejarah ini, yang menyoroti periode penelitian utama ini: pada unit logika ambang (awal 1940-an hingga pertengahan 1960-an), koneksionisme (awal 1980-an hingga pertengahan 1990-an), dan pembelajaran mendalam (pertengahan 2000-an hingga saat ini) . Gambar 4.1 membedakan beberapa karakteristik utama dari jaringan yang dikembangkan di masing-masing dari tiga periode ini. Perubahan dalam karakteristik jaringan ini menyoroti beberapa tema utama dalam evolusi pembelajaran mendalam, termasuk: pergeseran dari nilai biner ke nilai berkelanjutan; perpindahan dari fungsi aktivasi ambang batas, ke aktivasi logistik dan tanh, dan kemudian ke aktivasi ULT; dan pendalaman progresif jaringan, dari satu lapisan,ke beberapa lapisan, lalu ke jaringan dalam. Akhirnya, bagian atas gambar 4.1 menyajikan beberapa terobosan konseptual penting, algoritme pelatihan, dan arsitektur model yang telah berkontribusi pada evolusi pembelajaran mendalam.

Gambar 4.1 memberikan peta struktur bab ini, dengan urutan konsep yang diperkenalkan dalam bab tersebut secara umum mengikuti kronologi garis waktu ini. Dua persegi panjang abu-abu pada gambar 4.1 mewakili pengembangan dari dua arsitektur jaringan pembelajaran dalam yang penting: jaringan saraf konvolusional (CNN), dan jaringan saraf berulang (RNN). Kami akan menjelaskan

evolusi dari dua arsitektur jaringan ini pada bab ini, dan bab 5 akan memberikan penjelasan yang lebih detail tentang cara kerja jaringan ini.



Gambar 4.1 Sejarah Deep Learning.

Penelitian Awal: Unit Logika Ambang

Dalam beberapa literatur tentang deep learning, penelitian jaringan saraf awal dikategorikan sebagai bagian dari sibernetika, bidang penelitian yang berkaitan dengan pengembangan model komputasi kontrol dan pembelajaran dalam unit biologis. Namun, pada gambar 4.1, mengikuti terminologi yang digunakan dalam Nilsson (1965), karya awal ini dikategorikan sebagai penelitian tentang unit logika ambang karena istilah ini secara transparan menggambarkan karakteristik utama darisistem yang dikembangkan selama periode ini. Sebagian besar model yang dikembangkan pada tahun 1940-an, 50-an, dan 60-an memproses input Boolean (true / false direpresentasikan sebagai + 1 / -1 atau 1/0) dan menghasilkan output Boolean. Mereka juga menggunakan fungsi aktivasi ambang (diperkenalkan di bab 3), dan dibatasi untuk jaringan lapisan tunggal; dengan kata lain, mereka dibatasi pada satu matriks bobot merdu. Seringkali, fokus penelitian awal ini adalah pada pemahaman apakah model komputasi yang didasarkan pada neuron buatan memiliki kapasitas untuk mempelajari hubungan logis, seperti konjungsi atau disjungsi.

Pada tahun 1943, Walter McCulloch dan Walter Pitts menerbitkan model komputasi neuron biologis yang berpengaruh dalam sebuah makalah berjudul: "Kalkulus Logis dari Ide yang Immanen dalam Aktivitas Saraf" (McCulloch dan Pitts 1943). Makalah ini menyoroti karakteristik semua atau tidak sama sekali dari aktivitas saraf di otak dan mulai menjelaskan secara matematis aktivitas saraf

dalam hal kalkulus logika proposisional. Dalam model McCulloch dan Pitts, semua input dan output ke neuron adalah 0 atau 1. Selanjutnya, setiap input adalah rangsang (memiliki bobot +1) atau penghambatan (memiliki bobot -1). Konsep utama yang diperkenalkan dalam model McCulloch dan Pitts adalah penjumlahan input diikuti dengan fungsi ambang yang diterapkan pada hasil penjumlahan. Dalam penjumlahan, jika input rangsang aktif, itu menambahkan 1;penjumlahan di atas ambang batas yang telah ditetapkan, maka keluaran dari neuron adalah 1; jika tidak, akan menghasilkan 0. Dalam makalah, McCulloch dan Pitts mendemonstrasikan bagaimana operasi logis (seperti konjungsi, disjungsi, dan negasi) dapat direpresentasikan menggunakan model sederhana ini. Model McCulloch dan Pitts mengintegrasikan sebagian besar elemen yang ada di neuron buatan yang diperkenalkan di bab 3. Dalam model ini, bagaimanapun, neuron telah diperbaiki; dengan kata lain, bobot dan ambang batas ditentukan oleh han.

Pada tahun 1949, Donald O. Hebb menerbitkan sebuah buku berjudul *The Organization of Behavior* , di mana ia menetapkan teori neuropsikologis (mengintegrasikan psikologi dan fisiologi otak) untuk menjelaskan perilaku manusia secara umum. Premis mendasar dari teori ini adalah bahwa perilaku muncul melalui tindakan dan interaksi neuron. Untuk penelitian jaringan saraf, ide terpenting dalam buku ini adalah dalil, yang sekarang dikenal sebagai dalil Hebb, yang menjelaskan penciptaan ingatan abadi pada hewan berdasarkan proses perubahan pada koneksi antar neuron:

> Ketika akson sel A cukup dekat untuk merangsang sel B dan berulang kali atau terus-menerus mengambil bagian dalam menembakkannya, beberapa proses pertumbuhan atau perubahan metabolisme terjadi di satu atau kedua sel sehingga efisiensi A, sebagai salah satu sel yang menembakkan B , meningkat. (Hebb 1949, hlm.62)

Postulat ini penting karena menegaskan bahwa informasi disimpan dalam koneksi antar neuron (yaitu, dalam bobot jaringan), dan lebih jauh lagi bahwa pembelajaran terjadi dengan mengubah koneksi ini berdasarkan pola aktivasi berulang (yaitu, pembelajaran dapat berlangsung dalam jaringan dengan mengubah bobot jaringan).

Aturan Pelatihan Perceptron Rosenblatt

Pada tahun-tahun setelah publikasi Hebb, sejumlah peneliti mengusulkan model komputasi aktivitas neuron yang mengintegrasikan unit aktivasi ambang batas Boolean dari McCulloch dan Pitts, dengan mekanisme pembelajaran berdasarkan penyesuaian bobot yang diterapkan pada input. Yang paling terkenal dari model ini adalah model perceptron Frank Rosenblatt (Rosenblatt 1958). Secara konseptual, model perceptron dapat dipahami sebagai jaringan saraf yang terdiri dari neuron buatan tunggal yang menggunakan unit aktivasi ambang batas. Yang penting, jaringan perceptron hanya memiliki satu lapisan bobot. Implementasi pertama dari

perceptron adalah implementasi perangkat lunak pada sistem IBM 704 (dan ini mungkin implementasi pertama dari semua jaringan saraf). Namun, Rosenblatt selalu menginginkan perceptron menjadi mesin fisik dan kemudian diterapkan di perangkat keras yang dibuat khusus yang dikenal sebagai "Mark 1 perceptron". Mark 1 perceptron menerima input dari kamera yang menghasilkan gambar 400-pikselditeruskan ke mesin melalui susunan 400 fotosel yang pada gilirannya terhubung ke neuron. Bobot pada koneksi ke neuron diimplementasikan menggunakan resistor listrik yang dapat disesuaikan yang dikenal sebagai potensiometer, dan penyesuaian bobot diterapkan dengan menggunakan motor listrik untuk mengatur potensiometer.

Rosenblatt mengusulkan prosedur pelatihan koreksi kesalahan untuk memperbarui bobot perceptron sehingga dapat belajar membedakan antara dua kelas masukan: masukan yang perceptron harus menghasilkan keluaran $y = +1$, dan masukan yang perceptron harus menghasilkan keluaran $y = -1$ (Rosenblatt 1960). Prosedur pelatihan mengasumsikan sekumpulan pola masukan yang dikodekan Boolean, masing-masing dengan keluaran target yang terkait. Pada awal pelatihan, bobot di perceptron diinisialisasi ke nilai acak. Pelatihan kemudian dilanjutkan dengan melakukan iterasi melalui contoh pelatihan, dan setelah setiap contoh disajikan ke jaringan, bobot jaringan diperbarui berdasarkan kesalahan antara keluaran yang dihasilkan oleh perceptron dan keluaran target yang ditentukan dalam data. Contoh pelatihan dapat disajikan ke jaringan dalam urutan apa pun dan contoh dapat disajikan beberapa kali sebelum pelatihan selesai. Pelatihan lengkap melewati sekumpulan contoh dikenal sebagai iterasi, dan pelatihan berakhir saat perceptron mengklasifikasikan semua contoh dengan benar dalam iterasi.

Rosenblatt mendefinisikan aturan pembelajaran (dikenal sebagai aturan pelatihan perceptron) untuk memperbarui setiap bobot dalam perceptron setelah contoh pelatihan diproses. Strategi aturan yang digunakan untuk memperbarui bobot sama dengan strategi tiga kondisi yang kami perkenalkan di bab 2 untuk menyesuaikan bobot dalam model keputusan pinjaman:

1. Jika keluaran model untuk sebuah contoh cocok dengan keluaran yang ditentukan untuk contoh tersebut dalam dataset, maka jangan perbarui bobotnya.
2. Jika keluaran model terlalu rendah untuk contoh saat ini, maka tingkatkan keluaran model tersebut dengan menambah bobot untuk masukan yang memiliki nilai positif sebagai contoh dan menurunkan bobot untuk masukan yang bernilai negatif untuk Contoh.
3. Jika keluaran model terlalu tinggi untuk contoh saat ini, maka kurangi keluaran model tersebut dengan cara mengurangi bobot masukan yang bernilai positif dan menambah bobot untuk masukan yang bernilai negatif misalnya .

Ditulis dalam persamaan, aturan pembelajaran Rosenblatt memperbarui weight i (w_i) sebagai:

$$w_i^{t+1} = w_i^t + (\eta \times (y^t - \hat{y}^t) \times x_i^t)$$

Dalam aturan ini, w_i^{t+1} adalah nilai bobot *i* setelah bobot jaringan diperbarui sebagai respons terhadap pemrosesan contoh *t* , w_i^t adalah nilai bobot *i yang* digunakan selama pemrosesan contoh *t* , η adalah konstanta positif preset (dikenal sebagai pembelajaran rate, dibahas di bawah), y^t adalah output yang diharapkan misalnya *t* seperti yang ditentukan dalam dataset pelatihan, $\hat{y}^t$ adalah output yang dihasilkan oleh perceptron misalnya *t* , dan x_i^t merupakan komponen input *t* yang dibobotkan w_i^t selama pemrosesan contoh.

Meskipun mungkin terlihat rumit, aturan pelatihan perceptron sebenarnya hanyalah spesifikasi matematis dari strategi pembaruan bobot tiga kondisi yang dijelaskan di atas. Bagian utama dari persamaan untuk memahami adalah perhitungan perbedaan antara output yang diharapkan dan apa perceptron sebenarnya diprediksi: $y^t - \hat{y}^t$. Hasil pengurangan ini memberi tahu kita yang mana dari tiga kondisi pembaruan yang kita hadapi. Dalam memahami cara kerja pengurangan ini, penting untuk diingat bahwa untuk model perceptron, keluaran yang diinginkan selalu $y = +1$ atau $y = -1$. Kondisi pertama adalah kapan $y^t - \hat{y}^t = 0$; maka output dari perceptron sudah benar dan bobot tidak berubah.

Kondisi pembaruan bobot kedua adalah ketika output perceptron terlalu besar. Kondisi ini hanya dapat terjadi jika keluaran yang benar misalnya t adalah $y^t = -1$ dan kondisi ini dipicu saat $y^t - \hat{y}^t < 0$. Dalam hal ini, jika perceptron output contohnya t adalah $\hat{y}^t = +1$, maka istilah kesalahannya negatif ($y^t - \hat{y}^t = -2$) dan bobot w_i diperbarui oleh $+(\eta \times -2 \times x_i^t)$. Dengan asumsi, untuk keperluan penjelasan ini, yang η disetel ke 0,5, maka pembaruan bobot ini disederhanakan menjadi $-x_i^t$. Dengan kata lain, ketika keluaran perceptron terlalu besar, aturan pembaruan bobot mengurangi nilai masukan dari bobot. Ini akan mengurangi bobot pada masukan dengan nilai positif sebagai contoh, dan menambah bobot pada masukan dengan nilai negatif sebagai contoh (mengurangi bilangan negatif sama dengan menjumlahkan bilangan positif).

Kondisi pembaruan bobot ketiga adalah ketika keluaran perceptron terlalu kecil. Kondisi pembaruan berat badan ini adalah kebalikan dari yang kedua. Itu hanya

dapat terjadi ketika $y^t = +1$ dan begitu juga dipicu ketika $y^t - \hat{y}^t > 0$. Dalam kasus ini ($y^t - \hat{y}^t = 2$), dan bobot diperbarui oleh $+(\eta \times 2 \times x_i^t)$. Sekali lagi dengan asumsi bahwa η diatur ke 0,5, maka pembaruan ini menyederhanakan $+x_i^t$, yang menyoroti bahwa ketika kesalahan perceptron positif, aturan memperbarui bobot dengan menambahkan input ke bobot. Ini memiliki efek menurunkan bobot pada input dengan nilai negatif sebagai contoh dan meningkatkan bobot pada input dengan nilai positif sebagai contoh.

Di sejumlah poin di paragraf sebelumnya, kami telah merujuk ke kecepatan pembelajaran η,. Tujuan kecepatan pemelajaran η,, adalah untuk mengontrol ukuran penyesuaian yang diterapkan pada bobot. Kecepatan pembelajaran adalah contoh hyperparameter yang telah disetel sebelumnya sebelum model dilatih. Ada kompromi dalam menyetel kecepatan pemelajaran:

Jika kecepatan pembelajaran terlalu kecil, mungkin diperlukan waktu yang sangat lama untuk proses pelatihan untuk menyatu pada satu set bobot yang sesuai.

Jika kecepatan pembelajaran terlalu besar, bobot jaringan dapat melompati ruang beban terlalu banyak dan pelatihan mungkin tidak menyatu sama sekali.

Salah satu strategi untuk menyetel kecepatan pembelajaran adalah menetapkannya ke nilai positif yang relatif kecil (mis., 0,01), dan strategi lainnya adalah menginisialisasi ke nilai yang lebih besar (mis., 1,0) tetapi secara sistematis menguranginya saat pelatihan berlangsung

(mis., $\eta^{t+1} = \eta^1 \times \frac{1}{t}$).

Untuk membuat diskusi tentang kecepatan pembelajaran ini lebih konkret, bayangkan Anda mencoba memecahkan teka-teki yang mengharuskan Anda memasukkan bola kecil untuk masuk ke dalam lubang. Anda dapat mengontrol arah dan kecepatan bola dengan memiringkan permukaan tempat bola menggelinding. Jika Anda memiringkan permukaan terlalu curam, bola akan bergerak sangat cepat dan kemungkinan besar akan melewati lubang, sehingga Anda harus menyesuaikan permukaannya lagi, dan jika Anda terlalu menyesuaikan, Anda mungkin akan berulang kali memiringkan permukaan. Sebaliknya, jika Anda hanya memiringkan permukaan sedikit, bola mungkin tidak bergerak sama sekali, atau mungkin bergerak sangat lambat membutuhkan waktu lama untuk mencapai lubang. Sekarang, dalam banyak hal, tantangan untuk memasukkan bola ke dalam hole serupa dengan masalah menemukan set bobot terbaik untuk jaringan. Pikirkan setiap poin di filepermukaan bola bergulir sebagai satu set bobot jaringan yang mungkin. Posisi bola di setiap titik waktu menentukan

set bobot jaringan saat ini. Posisi lubang menentukan kumpulan bobot jaringan yang optimal untuk tugas yang kami latih agar jaringan selesai. Dalam konteks ini, mengarahkan jaringan ke set bobot yang optimal dianalogikan dengan mengarahkan bola ke hole. Kecepatan pembelajaran memungkinkan kami untuk mengontrol seberapa cepat kami bergerak di seluruh permukaan saat kami mencari kumpulan bobot yang optimal. Jika kami menyetel kecepatan pemelajaran ke nilai tinggi, kami bergerak cepat di seluruh permukaan: kami mengizinkan update besar pada bobot di setiap iterasi, jadi ada perbedaan besar antara bobot jaringan dalam satu iterasi dan berikutnya. Atau, menggunakan analogi bola bergulir kami, bola bergerak sangat cepat, dan seperti dalam teka-teki ketika bola menggelinding terlalu cepat dan melewati lubang, proses pencarian kami mungkin berjalan sangat cepat sehingga melewatkan set bobot optimal. Sebaliknya, jika kami menyetel kecepatan pemelajaran ke nilai yang rendah, kami bergerak sangat lambat di seluruh permukaan: kami hanya mengizinkan pembaruan kecil pada bobot pada setiap iterasi; atau, dengan kata lain, kami hanya membiarkan bola bergerak sangat lambat. Dengan kecepatan pemelajaran yang rendah, kecil kemungkinan kita melewatkan kumpulan bobot yang optimal, tetapi mungkin perlu waktu yang sangat lama untuk mencapainya. Strategi memulai dengan kecepatan belajar tinggi dan kemudian menguranginya secara sistematis setara dengan memiringkan permukaan puzzle secara curam untuk membuat bola bergerak lalu mengurangi kemiringan untuk mengontrol bola saat mendekati lubang. proses pencarian kami mungkin berjalan sangat cepat sehingga melewatkan set bobot yang optimal. Sebaliknya, jika kami menyetel kecepatan pemelajaran ke nilai yang rendah, kami bergerak sangat lambat di seluruh permukaan: kami hanya mengizinkan pembaruan kecil pada bobot pada setiap iterasi; atau, dengan kata lain, kami hanya membiarkan bola bergerak sangat lambat. Dengan kecepatan pemelajaran yang rendah, kecil kemungkinan kita melewatkan kumpulan bobot yang optimal, tetapi mungkin perlu waktu yang sangat lama untuk mencapainya. Strategi memulai dengan kecepatan belajar tinggi dan kemudian menguranginya secara sistematis setara dengan memiringkan permukaan puzzle secara curam untuk membuat bola bergerak dan kemudian mengurangi kemiringan untuk mengontrol bola saat mendekati lubang. proses pencarian kami mungkin berjalan sangat cepat sehingga melewatkan set bobot yang optimal. Sebaliknya, jika kami menyetel kecepatan pemelajaran ke nilai yang rendah, kami bergerak sangat lambat di seluruh permukaan: kami hanya mengizinkan pembaruan kecil pada bobot pada setiap iterasi; atau, dengan kata lain, kami hanya membiarkan bola bergerak sangat lambat. Dengan kecepatan pemelajaran yang rendah, kecil kemungkinan kita melewatkan kumpulan bobot yang optimal, tetapi mungkin perlu waktu yang sangat lama untuk mendapatkannya. Strategi memulai dengan kecepatan belajar tinggi dan kemudian menguranginya secara sistematis setara dengan memiringkan permukaan puzzle secara curam untuk membuat bola bergerak dan kemudian mengurangi kemiringan untuk mengontrol bola saat mendekati lubang. kami hanya mengizinkan pembaruan kecil pada bobot di setiap iterasi; atau, dengan kata lain, kami hanya membiarkan bola bergerak sangat lambat. Dengan kecepatan pemelajaran yang rendah, kecil kemungkinan kita melewatkan kumpulan bobot

yang optimal, tetapi mungkin perlu waktu yang sangat lama untuk mendapatkannya. Strategi memulai dengan kecepatan belajar tinggi dan kemudian menguranginya secara sistematis setara dengan memiringkan permukaan puzzle secara curam agar bola bergerak dan kemudian mengurangi kemiringan untuk mengontrol bola saat mendekati lubang. kami hanya mengizinkan pembaruan kecil pada bobot di setiap iterasi; atau, dengan kata lain, kami hanya membiarkan bola bergerak sangat lambat. Dengan kecepatan pemelajaran yang rendah, kecil kemungkinan kita melewatkan kumpulan bobot yang optimal, tetapi mungkin perlu banyak waktu untuk mencapainya. Strategi memulai dengan kecepatan belajar tinggi dan kemudian menguranginya secara sistematis setara dengan memiringkan permukaan puzzle secara curam untuk membuat bola bergerak dan kemudian mengurangi kemiringan untuk mengontrol bola saat mendekati lubang.

Rosenblatt membuktikan bahwa jika ada satu set bobot yang memungkinkan perceptron mengklasifikasikan semua contoh pelatihan dengan benar, algoritme pelatihan perceptron pada akhirnya akan menyatu di set bobot ini. Temuan ini dikenal sebagai teorema konvergensi perceptron (Rosenblatt 1962). Kesulitan dalam melatih perceptron, bagaimanapun, adalah bahwa hal itu mungkin memerlukan sejumlah besar iterasi melalui data sebelum algoritme bertemu. Lebih lanjut, untuk banyak masalah tidak diketahui apakah satu set bobot yang sesuai sudah ada sebelumnya; akibatnya, jika pelatihan telah berlangsung dalam waktu lama, tidak mungkin untuk mengetahui apakah proses pelatihan hanya membutuhkan waktu lama untuk menyatu pada bobot dan berhenti, atau apakah tidak akan pernah berakhir.

Algoritma Kuadrat Rata-Rata Terkecil

Sekitar waktu yang sama ketika Rosenblatt mengembangkan perceptron, Bernard Widrow dan Marcian Hoff mengembangkan model yang sangat mirip yang disebut ADALINE (kependekan dari neuron linier adaptif), bersama dengan aturan pembelajaran yang disebut algoritma LMS (least mean square) (Widrow dan Hoff 1960). Jaringan ADALINE terdiri dari satu neuron yang sangat mirip dengan perceptron; satu-satunya perbedaan adalah bahwa jaringan ADALINE tidak menggunakan fungsi ambang batas. Faktanya, output dari jaringan ADALINE adalah jumlah input yang hanya berbobot. Inilah sebabnya mengapa ini dikenal sebagai neuron linier: penjumlahan tertimbang adalah fungsi linier (itumendefinisikan garis), sehingga jaringan ADALINE mengimplementasikan pemetaan linier dari input ke output. Aturan LMS hampir identik dengan aturan pembelajaran perceptron, kecuali bahwa output dari perceptron untuk contoh yang diberikan $\hat{y}^t$ diganti dengan jumlah input yang berbobot:

$$w_i^{t+1} = w_i^t + \left(\eta \times \left(y^t - \left(\sum_{i=0}^{n} w_i^t \times x_i^t \right) \right) \times x_i^t \right)$$

Logika aturan pembaruan LMS sama dengan logika aturan pelatihan perceptron. Jika keluaran terlalu besar, maka bobot yang diterapkan ke masukan positif menyebabkan keluaran menjadi lebih besar, dan bobot ini harus dikurangi, dan bobot yang diterapkan ke masukan negatif harus ditingkatkan, sehingga mengurangi keluaran di lain waktu pola masukan ini diterima. Dan, dengan logika yang sama, jika keluaran terlalu kecil, maka bobot yang diterapkan ke masukan positif akan dinaikkan dan bobot yang diterapkan ke masukan negatif harus dikurangi.

Jika keluaran model terlalu besar, maka bobot yang terkait dengan masukan positif harus dikurangi, sedangkan jika keluarannya terlalu kecil, maka bobot tersebut harus ditambah.

Salah satu aspek penting dari karya Widrow dan Hoff adalah untuk menunjukkan bahwa aturan LMS dapat digunakan untuk melatih jaringan untuk memprediksi sejumlah nilai apa pun, bukan hanya +1 atau -1. Aturan pembelajaran ini disebut algoritma kuadrat rata-rata terkecil karena menggunakan aturan LMS untuk menyesuaikan bobot secara iteratif dalam neuron sama dengan meminimalkan rata-rata kesalahan kuadrat pada set pelatihan. Saat ini, aturan pembelajaran LMS kadang-kadang disebut Widrow-Hoffaturan belajar, setelah penemu; Namun, ini lebih sering disebut aturan delta karena menggunakan perbedaan (atau delta) antara keluaran yang diinginkan dan keluaran aktual untuk menghitung penyesuaian bobot. Dengan kata lain, aturan LMS menetapkan bahwa bobot harus disesuaikan secara proporsional dengan perbedaan antara keluaran dari jaringan ADALINE dan keluaran yang diinginkan: jika neuron membuat kesalahan besar, maka bobot disesuaikan dengan jumlah yang besar, jika neuron membuat kesalahan kecil, kemudian bobot disesuaikan dengan jumlah kecil.

Saat ini, perceptron diakui sebagai tonggak penting dalam pengembangan jaringan saraf karena merupakan jaringan saraf pertama yang diimplementasikan. Namun, sebagian besar algoritme modern untuk melatih jaringan saraf lebih mirip dengan algoritme LMS. Algoritma LMS mencoba untuk meminimalkan kesalahan kuadrat rata-rata jaringan. Seperti yang akan dibahas dalam bab 6, secara teknis proses pengurangan kesalahan berulang ini melibatkan penurunan gradien ke permukaan kesalahan; dan, saat ini, hampir semua jaringan neural dilatih menggunakan beberapa varian penurunan gradien.

Masalah XOR

Keberhasilan Rosenblatt, Widrow dan Hoff, dan lainnya, dalam mendemonstrasikan bahwa model jaringan saraf dapat secara otomatis belajar untuk membedakan antara kumpulan pola yang berbeda, menghasilkan banyak kegembiraan seputar artifisial. intelijen dan penelitian jaringan saraf. Namun, pada tahun 1969, Marvin Minsky dan Seymour Papert menerbitkan sebuah buku berjudul *Perceptrons* , yang, dalam sejarah penelitian jaringan saraf, dikaitkan

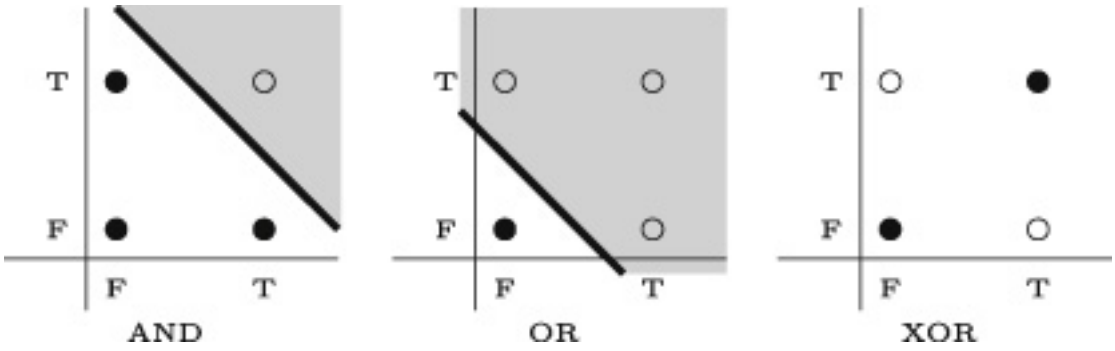
dengan penghancuran semangat dan optimisme awal ini sendirian (Minsky dan Papert 1969). Diakui, selama tahun 1960-an penelitian jaringan saraf telah menderita banyak hype, dan kurangnya keberhasilan dalam hal memenuhi harapan yang tinggi. Namun, buku Minsky dan Papert memberikan pandangan yang sangat negatif tentang kekuatan representasi jaringan saraf, dan setelah pendanaan publikasinya untuk penelitian jaringan saraf mengering.

Buku Minsky dan Papert terutama berfokus pada perceptron satu lapis. Ingatlah bahwa satu lapisan perceptron sama dengan satu neuron yang menggunakan fungsi aktivasi ambang batas, sehingga perceptron satu lapisan dibatasi untuk menerapkan batas keputusan linier (garis lurus).[1] Ini berarti bahwa satu lapisan perceptron hanya dapat belajar membedakan antara dua kelas masukan jika memungkinkan untuk menggambar garis lurus di ruang masukan yang memiliki semua contoh satu kelas di satu sisi garis dan semua contoh kelas lain di sisi lain garis. Minsky dan Papert menyoroti pembatasan ini sebagai kelemahan model-model ini.

Untuk memahami kritik Minsky dan Papert tentang perceptron lapisan tunggal, pertama-tama kita harus memahami konsep fungsi yang dapat dipisahkan secara linier. Kami akan menggunakan perbandinganantara fungsi logika AND dan OR dengan fungsi logika XOR untuk menjelaskan konsep fungsi yang dapat dipisahkan secara linier. Fungsi AND mengambil dua input, masing-masing bisa TRUE atau FALSE, dan mengembalikan TRUE jika kedua input TRUE. Plot di sebelah kiri gambar 4.4 menunjukkan ruang masukan untuk fungsi AND dan mengkategorikan masing-masing dari empat kemungkinan kombinasi masukan sebagai hasil dari nilai keluaran TRUE (diperlihatkan dalam gambar dengan menggunakan titik bening) atau SALAH (diperlihatkan di gambar dengan menggunakan titik hitam). Plot ini mengilustrasikan bahwa dimungkinkan untuk menggambar garis lurus antara input yang menghasilkan fungsi AND TRUE, (T, T), dan input yang menghasilkan FALSE, {(F, F), (F, T) , (T, F)}. Fungsi OR mirip dengan fungsi AND, kecuali fungsi OR mengembalikan TRUE jika salah satu atau kedua input adalah TRUE. Plot tengah pada gambar 4.4 menunjukkan bahwa dimungkinkan untuk menggambar garis yang memisahkan input yang diklasifikasikan oleh fungsi OR sebagai TRUE, {(F, T), (T, F), (T, T)}, dari input tersebut diklasifikasikan sebagai FALSE, (F, F). Karena kita dapat menggambar satu garis lurus dalam ruang masukan dari fungsi-fungsi ini yang membagi masukan yang termasuk dalam satu kategori keluaran dari masukan yang termasuk dalam kategori keluaran lainnya sehingga fungsi AND dan OR adalah fungsi yang dapat dipisahkan secara linier.

Fungsi XOR juga mirip strukturnya dengan fungsi AND dan OR; namun, ini hanya mengembalikan TRUE jika salah satu (tapi tidak keduanya) dari inputnya BENAR. Plot terus berlanjutbagian kanan gambar 4.2 menunjukkan ruang masukan untuk fungsi XOR dan mengkategorikan masing-masing dari empat kemungkinan kombinasi masukan sebagai mengembalikan TRUE (ditampilkan dalam gambar dengan menggunakan titik bening) atau SALAH (ditampilkan dalam gambar dengan menggunakan titik hitam). Melihat plot ini, Anda akan

melihat bahwa tidak mungkin menggambar garis lurus antara input yang diklasifikasikan fungsi XOR sebagai TRUE dan input yang diklasifikasikan sebagai FALSE. Karena kita tidak dapat menggunakan satu garis lurus untuk memisahkan input yang termasuk dalam kategori output yang berbeda untuk fungsi XOR, fungsi ini disebut sebagai fungsi yang dapat dipisahkan secara nonlinear. Fakta bahwa fungsi XOR dapat dipisahkan secara nonlinier tidak membuat fungsi tersebut unik, atau bahkan langka — ada banyak fungsi yang dapat dipisahkan secara nonlinier.

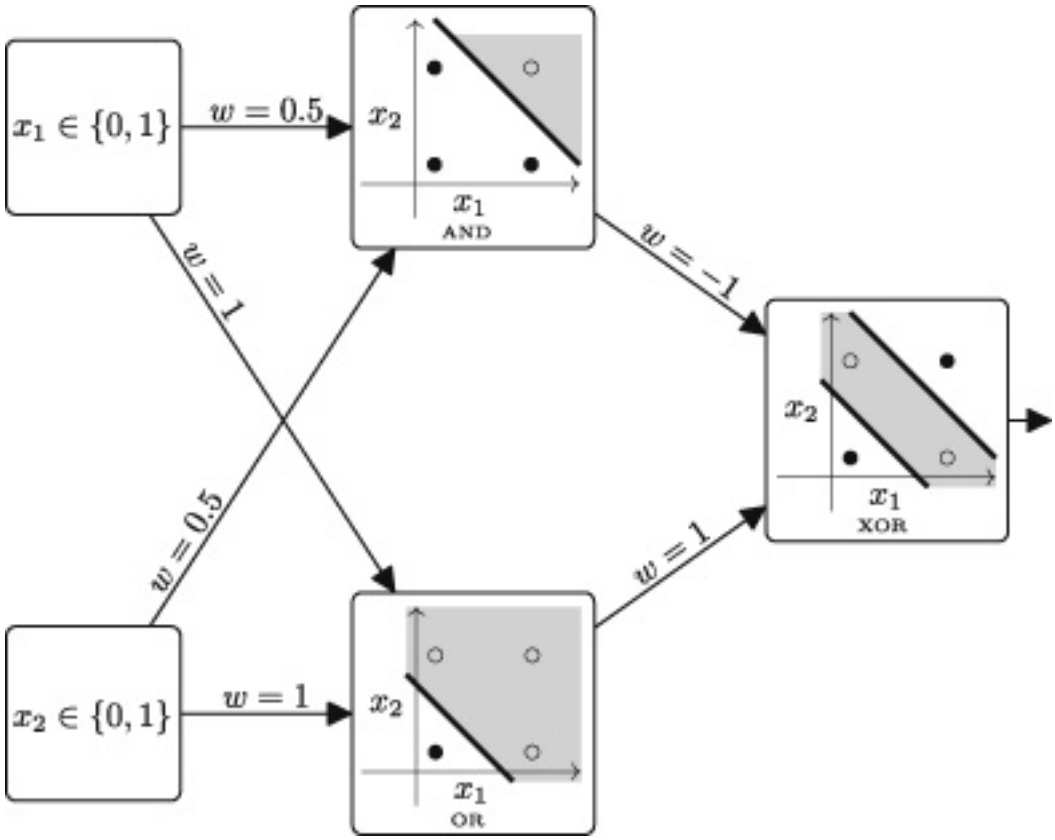


Gambar 4.2 Ilustrasi fungsi yang dapat dipisahkan secara linier. Di setiap gambar, titik hitam mewakili masukan yang dikembalikan fungsi FALSE, lingkaran mewakili masukan yang mengembalikan fungsi TRUE. (T berarti benar dan F berarti salah.)

Kritik utama yang dibuat Minsky dan Papert dari perceptrons satu lapis adalah bahwa model lapisan tunggal ini tidak dapat mempelajari fungsi yang dapat dipisahkan secara nonlinier, seperti fungsi XOR. Alasan untuk batasan ini adalah bahwa batas keputusan perceptron adalah linier sehingga perceptron satu lapis tidak dapat belajar membedakan antara input yang termasuk dalam kategori output dari fungsi yang dapat dipisahkan secara nonlinier dari input yang termasuk dalam kategori lain.

Diketahui pada saat publikasi Minsky dan Papert bahwa adalah mungkin untuk membangun jaringan saraf yang mendefinisikan batas keputusan nonlinier, dan dengan demikian mempelajari fungsi yang dapat dipisahkan secara nonlinier (seperti fungsi XOR). Kunci untuk menciptakan jaringan dengan batasan keputusan yang lebih kompleks (nonlinier) adalah memperluas jaringan agar memiliki banyak lapisan neuron. Misalnya, gambar 4.3 menunjukkan jaringan dua lapis yang mengimplementasikan fungsi XOR. Dalam jaringan ini, nilai logika TRUE dan FALSE dipetakan ke nilai numerik: nilai FALSE diwakili oleh 0, dan nilai TRUE diwakili oleh 1. Dalam jaringan ini, unit aktif (keluaran +1) jika jumlah bobot masukan adalah ≥ 1; jika tidak, mereka menghasilkan 0. Perhatikan bahwa unit di lapisan tersembunyi mengimplementasikan fungsi logika AND dan OR. Ini dapat dipahami sebagai langkah perantara untuk menyelesaikan tantangan XOR. Unit di lapisan keluaran mengimplementasikan XOR dengan menyusun keluaran dari lapisan tersembunyi ini. Dengan kata lain, unit di lapisan keluaran mengembalikan TRUE hanya ketika simpul AND mati (keluaran = 0) dan ORnode aktif (output = 1). Namun, pada saat itu tidak jelas bagaimana melatih jaringan dengan banyak lapisan. Juga, di akhir buku mereka, Minsky dan Papert

berpendapat bahwa "dalam penilaian mereka" penelitian tentang perluasan jaringan saraf ke beberapa lapisan adalah "steril" (Minsky dan Papert 1969, bagian 13.2 halaman 23).



Gambar 4.3 Jaringan yang mengimplementasikan fungsi XOR. Semua unit pemrosesan menggunakan fungsi aktivasi ambang dengan ambang batas ≥ 1.

Dalam twist sejarah yang agak ironis, sezaman dengan publikasi Minsky dan Papert, Alexey Ivakhnenko, seorang peneliti Ukraina, mengusulkan metode kelompok untuk penanganan data (GMDH), dan pada tahun 1971 diterbitkansebuah makalah yang menjelaskan bagaimana hal itu dapat digunakan untuk mempelajari jaringan saraf dengan delapan lapisan (Ivakhnenko 1971). Saat ini, jaringan GMDH Ivakhnenko pada 1971 dianggap sebagai contoh jaringan dalam yang pertama kali diterbitkan yang dilatih dari data (Schmidhuber 2015). Namun, selama bertahun-tahun, pencapaian Ivaknenko sebagian besar diabaikan oleh komunitas jaringan saraf yang lebih luas. Akibatnya, sangat sedikit pekerjaan saat ini dalam pembelajaran mendalam yang menggunakan metode GMDH untuk pelatihan: pada tahun-tahun berikutnya, algoritme pelatihan lain, seperti propagasi mundur (dijelaskan di bawah), menjadi standar di komunitas. Pada saat yang sama dengan pencapaian Ivakhnenko yang terabaikan, kritik Minsky dan Papert terbukti persuasif dan menandai akhir dari periode pertama penelitian signifikan tentang jaringan saraf.

Namun, periode pertama penelitian jaringan saraf ini meninggalkan warisan yang membentuk perkembangan bidang hingga hari ini. Struktur internal dasar dari neuron buatan didefinisikan: jumlah input yang diberikan melalui fungsi aktivasi. Konsep menyimpan informasi dalam bobot jaringan dikembangkan. Selanjutnya, algoritma pembelajaran yang didasarkan pada bobot adaptasi iteratif

diusulkan, bersama dengan aturan pembelajaran praktis, seperti aturan LMS. Secara khusus, pendekatan LMS, menyesuaikan bobot neuron secara proporsional dengan perbedaan antara keluaran neuron dan keluaran yang diinginkan, ada di sebagian besaralgoritma pelatihan modern. Akhirnya, ada pengakuan atas keterbatasan jaringan lapisan tunggal, dan pemahaman bahwa salah satu cara untuk mengatasi keterbatasan ini adalah dengan memperluas jaringan untuk memasukkan banyak lapisan neuron. Saat ini, bagaimanapun, tidak jelas bagaimana melatih jaringan dengan banyak lapisan. Memperbarui bobot membutuhkan pemahaman tentang bagaimana bobot memengaruhi kesalahan jaringan. Misalnya, dalam aturan LMS jika keluaran neuron terlalu besar, maka bobot yang diterapkan pada masukan positif menyebabkan keluaran tersebut meningkat. Oleh karena itu, mengurangi ukuran bobot ini akan mengurangi keluaran dan dengan demikian mengurangi kesalahan. Tapi, di akhir 1960-an, pertanyaan tentang bagaimana memodelkan hubungan antara bobot input ke neuron di lapisan tersembunyi jaringan dan kesalahan keseluruhan jaringan masih belum terjawab; dan, tanpa estimasi kontribusi bobot terhadap error ini, tidak mungkin menyesuaikan bobot di lapisan tersembunyi jaringan. Masalah menghubungkan (atau menugaskan) sejumlah kesalahan ke komponen dalam jaringan kadang-kadang disebut sebagai masalah penugasan kredit, atau sebagai masalah penetapan kesalahan.

Koneksionisme: Perceptrons Multilayer

Pada 1980-an, orang-orang mulai mengevaluasi kembali kritik pada akhir 1960-an sebagai terlalu parah. Dua perkembangan,khususnya, menghidupkan kembali bidang: (1) jaringan Hopfield; dan (2) algoritma propagasi mundur.

Pada tahun 1982, John Hopfield menerbitkan sebuah makalah di mana dia menggambarkan jaringan yang dapat berfungsi sebagai memori asosiatif (Hopfield 1982). Selama pelatihan, memori asosiatif mempelajari serangkaian pola masukan. Setelah jaringan memori asosiasi dilatih, kemudian, jika versi rusak dari salah satu pola masukan ditampilkan ke jaringan, jaringan dapat membuat ulang pola yang benar sepenuhnya. Memori asosiatif berguna untuk sejumlah tugas, termasuk penyelesaian pola dan koreksi kesalahan. Tabel 4.1 [2] mengilustrasikan tugas penyelesaian pola dan koreksi kesalahan menggunakan contoh memori asosiatif yang telah dilatih untuk menyimpan informasi pada hari ulang tahun orang. Dalam jaringan Hopfield, ingatan, atau pola masukan, dikodekan dalam string biner; dan,Dengan asumsi pola biner relatif berbeda satu sama lain, jaringan Hopfield dapat menyimpan hingga 0,138 N string ini, di mana N adalah jumlah neuron dalam jaringan. Jadi untuk menyimpan 10 pola berbeda membutuhkan jaringan Hopfield dengan 73 neuron, dan untuk menyimpan 14 pola berbeda membutuhkan 100 neuron.

Tabel 4.1. Ilustrasi penggunaan memori asosiasi untuk penyelesaian pola dan koreksi kesalahan

Pola pelatihan	Penyelesaian pola		
John**12May	Liz***?????	→	Liz***25Feb
Kerry*03Jan	???***10Mar	→	Des***10Mar
Liz***25Feb	**Koreksi kesalahan**		
Des***10Mar	Kerry*01Apr	→	Kerry*03Jan
Josef*13Dec	Jxsuf*13Dec	→	Josef*13Dec

Backpropagation dan Vanishing Gradients

Pada tahun 1986, sekelompok peneliti yang dikenal sebagai kelompok penelitian pemrosesan terdistribusi paralel (PDP) menerbitkan ikhtisar dua buku penelitian jaringan saraf (Rumelhart et al. 1986b, 1986c). Buku-buku ini terbukti sangat populer, dan bab 8 dalam volume satu menjelaskan tentang algoritma propagasi mundur (Rumelhart et al. 1986a). Algoritma propagasi mundur telah ditemukan beberapa kali,[3] tetapi bab ini oleh Rumelhart, Hinton, dan Williams, yang diterbitkan oleh PDP, yang mempopulerkan penggunaannya. Algoritma backpropagation merupakan solusi dari masalah credit assignment sehingga dapat digunakan untuk melatih jaringan saraf yang memiliki lapisan neuron tersembunyi. Algoritme propagasi mundur mungkin merupakan algoritme paling penting dalam pembelajaran mendalam. Namun, penjelasan yang jelas dan lengkap tentang algoritme propagasi mundur memerlukan penjelasan terlebih dahulu tentang konsep gradien kesalahan, lalu algoritme penurunan gradien. Konsekuensinya, penjelasan mendalam tentang propagasi mundur ditunda hingga bab 6 yang diawali dengan penjelasan.konsep yang diperlukan ini. Namun, struktur umum algoritme dapat dijelaskan dengan relatif cepat. Algoritme propagasi mundur dimulai dengan menetapkan bobot acak ke setiap koneksi di jaringan. Algoritme kemudian memperbarui bobot di jaringan secara berulang dengan menampilkan contoh pelatihan ke jaringan dan memperbarui bobot jaringan hingga jaringan berfungsi seperti yang diharapkan. Algoritma inti bekerja dalam proses dua tahap. Pada tahap pertama (dikenal sebagai forward pass), masukan disajikan ke jaringan dan aktivasi neuron dibiarkan mengalir maju melalui jaringan sampai keluaran dihasilkan. Tahap kedua (dikenal sebagai backward pass) dimulai pada lapisan keluaran dan bekerja mundur melalui jaringan sampai lapisan masukan tercapai.

Langkah mundur ini dimulai dengan menghitung kesalahan untuk setiap neuron di lapisan keluaran. Kesalahan ini kemudian digunakan untuk memperbarui bobot neuron keluaran ini. Kemudian kesalahan dari setiap neuron keluaran dibagikan kembali (backpropagated) ke neuron tersembunyi yang terhubung dengannya, sebanding dengan bobot pada koneksi antara neuron keluaran dan neuron tersembunyi. Setelah pembagian ini (atau tugas menyalahkan) telah diselesaikan untuk neuron tersembunyi, kesalahan total yang disebabkan oleh neuron tersembunyi itu dijumlahkan dan jumlah ini digunakan untuk memperbarui bobot pada neuron itu. Propagasi mundur (atau berbagi kembali) kesalahan kemudian diulangi untuk neuron yang belum menyalahkan yang dikaitkan dengan mereka. Proses penugasan menyalahkan dan Kemudian kesalahan dari setiap neuron keluaran dibagi kembali (backpropagated) ke neuron tersembunyi yang terhubung dengannya, sebanding dengan bobot pada koneksi antara neuron keluaran dan neuron tersembunyi. Setelah pembagian ini (atau tugas menyalahkan) telah diselesaikan untuk neuron tersembunyi, kesalahan total yang disebabkan oleh neuron tersembunyi itu dijumlahkan dan jumlah ini digunakan untuk memperbarui bobot pada neuron itu. Propagasi balik (atau berbagi kembali) kesalahan kemudian diulangi untuk neuron yang belum menyalahkan yang dikaitkan dengan mereka. Proses penugasan menyalahkan dan Kemudian kesalahan dari setiap neuron keluaran dibagi kembali (backpropagated) ke neuron tersembunyi yang terhubung dengannya, sebanding dengan bobot pada koneksi antara neuron keluaran dan neuron tersembunyi. Setelah pembagian ini (atau tugas menyalahkan) telah diselesaikan untuk neuron tersembunyi, kesalahan total yang disebabkan oleh neuron tersembunyi itu dijumlahkan dan jumlah ini digunakan untuk memperbarui bobot pada neuron itu. Propagasi mundur (atau berbagi kembali) kesalahan kemudian diulangi untuk neuron yang belum menyalahkan yang dikaitkan dengan mereka. Proses penugasan menyalahkan dan kesalahan total yang diakibatkan oleh neuron tersembunyi itu dijumlahkan dan jumlah ini digunakan untuk memperbarui bobot pada neuron itu. Propagasi mundur (atau berbagi kembali) kesalahan kemudian diulangi untuk neuron yang belum menyalahkan yang dikaitkan dengan mereka. Proses penugasan menyalahkan dan kesalahan total yang disebabkan oleh neuron tersembunyi itu dijumlahkan dan jumlah ini digunakan untuk memperbarui bobot pada neuron itu. Propagasi balik (atau berbagi kembali) kesalahan kemudian diulangi untuk neuron yang belum menyalahkan yang dikaitkan dengan mereka. Proses penugasan menyalahkan danpembaruan bobot berlanjut kembali melalui jaringan sampai semua bobot telah diperbarui.

Inovasi utama yang memungkinkan algoritme propagasi mundur bekerja adalah perubahan dalam fungsi aktivasi yang digunakan di neuron. Jaringan yang dikembangkan pada tahun-tahun awal penelitian jaringan saraf menggunakan fungsi aktivasi ambang batas. Algoritma propagasi mundur tidak bekerja dengan fungsi aktivasi ambang karena propagasi mundur mengharuskan fungsi aktivasi yang digunakan oleh neuron dalam jaringan dapat dibedakan. Fungsi aktivasi ambang tidak dapat dibedakan karena ada diskontinuitas dalam keluaran fungsi di ambang batas. Dengan kata lain, kemiringan fungsi ambang batas pada ambang batas tidak terbatas dan oleh karena itu tidak mungkin menghitung gradien fungsi

pada titik tersebut. Hal ini menyebabkan penggunaan fungsi aktivasi yang dapat dibedakan dalam jaringan saraf multilayer,

Namun, ada batasan inheren dengan menggunakan algoritme propagasi mundur untuk melatih jaringan dalam. Pada 1980-an, para peneliti menemukan bahwa propagasi mundur bekerja dengan baik dengan jaringan yang relatif dangkal (satu atau dua lapisan unit tersembunyi), tetapi ketika jaringan semakin dalam, jaringan tersebut membutuhkan banyak waktu untuk dilatih, atau mereka sepenuhnya gagal untuk melakukannya. berkumpul pada satu set bobot yang bagus. Pada tahun 1991, Sepp Hochreiter (bekerja dengan Jürgen Schmidhuber) mengidentifikasi penyebab masalah ini dalam bukunyatesis diploma (Hochreiter 1991). Masalahnya disebabkan oleh cara algoritma melakukan backpropagates error. Pada dasarnya algoritma backpropagation merupakan implementasi dari aturan rantai dari kalkulus. Aturan rantai melibatkan perkalian suku, dan melakukan propagasi mundur kesalahan dari satu neuron kembali ke neuron lain dapat melibatkan mengalikan kesalahan dengan suku bilangan dengan nilai kurang dari 1. Perkalian ini dengan nilai kurang dari 1 terjadi berulang kali saat sinyal kesalahan diteruskan kembali melalui jaringan. Ini menghasilkan sinyal kesalahan menjadi lebih kecil dan lebih kecil karena dipropagasi balik melalui jaringan. Memang, sinyal kesalahan sering berkurang secara eksponensial sehubungan dengan jarak dari lapisan keluaran. Efek dari kesalahan yang semakin berkurang ini adalah bahwa bobot di lapisan awal jaringan dalam sering kali disesuaikan hanya dengan jumlah kecil (atau nol) selama setiap iterasi pelatihan. Dengan kata lain, lapisan awal berlatih sangat, sangat lambat atau tidak bergerak sama sekali dari posisi awal acaknya. Namun, lapisan awal dalam jaringan saraf sangat penting untuk keberhasilan jaringan, karena neuron di lapisan inilah yang belajar mendeteksi fitur dalam masukan yang digunakan lapisan jaringan berikutnya sebagai blok bangunan dasar. representasi yang pada akhirnya menentukan keluaran jaringan. Untuk alasan teknis, yang akan dijelaskan pada bab 6, sinyal kesalahan yang di-propagasi balik melalui jaringan sebenarnya adalah gradien kesalahan jaringan, lapisan awal dalam jaringan saraf sangat penting untuk keberhasilan jaringan, karena neuron dalam lapisan inilah yang belajar mendeteksi fitur dalam masukan yang digunakan lapisan jaringan berikutnya sebagai blok bangunan dasar representasi. yang pada akhirnya menentukan keluaran jaringan. Untuk alasan teknis, yang akan dijelaskan pada bab 6, sinyal kesalahan yang di-propagasi balik melalui jaringan sebenarnya adalah gradien kesalahan jaringan, lapisan awal dalam jaringan saraf sangat penting untuk keberhasilan jaringan, karena neuron dalam lapisan inilah yang belajar mendeteksi fitur dalam masukan yang digunakan lapisan jaringan berikutnya sebagai blok bangunan dasar representasi. yang pada akhirnya menentukan keluaran jaringan. Untuk alasan teknis, yang akan dijelaskan pada bab 6, sinyal kesalahan yang di-propagasi balik melalui jaringan sebenarnya adalah gradien kesalahan jaringan,dan, akibatnya, masalah sinyal kesalahan ini dengan cepat berkurang mendekati nol dikenal sebagai masalah gradien lenyap.

Koneksionisme dan Representasi Lokal versus Terdistribusi

Terlepas dari masalah gradien yang menghilang, algoritme propagasi mundur membuka kemungkinan untuk melatih arsitektur jaringan saraf yang lebih kompleks (lebih dalam). Ini selaras dengan prinsip koneksionisme. Koneksionisme adalah gagasan bahwa perilaku cerdas dapat muncul dari interaksi antara sejumlah besar unit pemrosesan sederhana. Aspek lain dari koneksionisme adalah gagasan tentang representasi terdistribusi. Perbedaan dapat dibuat dalam representasi yang digunakan oleh jaringan saraf antara representasi lokalis dan terdistribusi. Dalam representasi lokalis terdapat korespondensi satu-ke-satu antara konsep dan neuron, sedangkan dalam representasi terdistribusi setiap konsep diwakili oleh pola aktivasi di seluruh rangkaian neuron. Karena itu,

Dalam representasi terdistribusi, setiap konsep diwakili oleh aktivasi beberapa neuron dan aktivasi setiap neuron berkontribusi pada representasi beberapa konsep.

Untuk mengilustrasikan perbedaan antara representasi lokalis dan terdistribusi, pertimbangkan skenario di mana (untuk beberapa alasan yang tidak ditentukan) satu set aktivasi neuron digunakan untuk mewakili ada atau tidaknya makanan yang berbeda. Selain itu, setiap makanan memiliki dua sifat, negara asal resep dan cita rasanya. Negara asal yang mungkin adalah: *Italia* , *Meksiko* , atau *Prancis* ; dan, set rasa yang mungkin adalah: *Manis* , *Asam* , atau *Pahit* . Jadi, total ada sembilan kemungkinan jenis makanan: *Italia* + *Manis* , *Italia* + *Asam* , *Italia* + *Pahit* , *Meksiko* + *Manis*, dll. Menggunakan representasi lokalis akan membutuhkan sembilan neuron, satu neuron per jenis makanan. Namun, ada sejumlah cara untuk menentukan representasi terdistribusi dari domain ini. Salah satu pendekatannya adalah dengan menetapkan bilangan biner untuk setiap kombinasi. Representasi ini hanya membutuhkan empat neuron, dengan pola aktivasi 0000 mewakili *Italia* + *Manis* , 0001 mewakili *Italia* + *Asam* , 0010 mewakili *Italia* + *Pahit* , dan seterusnya hingga 1000 mewakili *Prancis* + *Bitter* . Ini adalah representasi yang sangat kompak. Namun, perhatikan bahwa dalam representasi ini aktivasi setiap neuron dalam isolasi tidak memiliki interpretasi yang bermakna secara independen: neuron paling kanan akan aktif (*** 1) untuk *Italian* + *Sour* ,*Meksiko* + *Manis* , *Meksiko* + *Pahit* , dan *Prancis* + *Asam* , dan tanpa pengetahuan tentang aktivasi neuron lain, tidak mungkin mengetahui negara atau rasa apa yang diwakili. Namun, dalam jaringan yang dalam, kurangnya interpretabilitas semantik dari aktivasi unit tersembunyi tidak menjadi masalah, selama neuron di lapisan keluaran jaringan mampu menggabungkan representasi ini sedemikian rupa sehingga dapatmenghasilkan keluaran yang benar. Representasi lain yang lebih transparan dan terdistribusi dari domain makanan ini adalah menggunakan tiga neuron untuk mewakili negara dan tiga neuron untuk mewakili selera. Dalam representasi ini, pola aktivasi 100100 dapat mewakili *Italia* + *Manis* , 001100 dapat mewakili *Prancis* + *Manis* , dan 001001 dapat mewakili *Prancis* + *Pahit*. Dalam representasi ini, aktivasi setiap neuron dapat diinterpretasikan secara independen; namun distribusi aktivasi di seluruh

rangkaian neuron diperlukan untuk mendapatkan deskripsi lengkap dari makanan (negara + rasa). Perhatikan, bagaimanapun, bahwa kedua representasi terdistribusi ini lebih kompak daripada representasi lokalis. Kekompakan ini dapat secara signifikan mengurangi jumlah bobot yang dibutuhkan dalam jaringan, dan ini pada gilirannya dapat menghasilkan waktu pelatihan yang lebih cepat untuk jaringan tersebut.

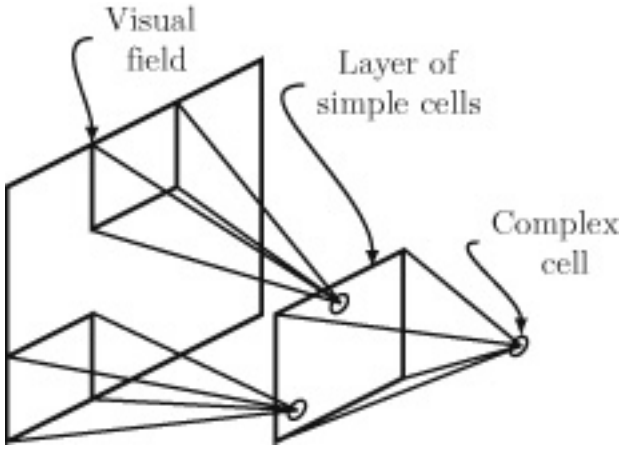
Konsep representasi terdistribusi sangat penting dalam pembelajaran yang mendalam. Memang, ada argumen yang bagus bahwa pembelajaran dalam mungkin lebih tepat dinamai pembelajaran representasi — argumennya adalah bahwa neuron di lapisan tersembunyi jaringan sedang mempelajari representasi terdistribusi dari input yang merupakan representasi perantara yang berguna dalam pemetaan dari input ke output. yang coba dipelajari jaringan. Tugas dari lapisan keluaran suatu jaringan adalah mempelajari bagaimana menggabungkan representasi perantara ini untuk menghasilkan keluaran yang diinginkan. Pertimbangkan lagi jaringan masukgambar 4.3 yang mengimplementasikan fungsi XOR. Unit tersembunyi dalam jaringan ini mempelajari representasi perantara dari input, yang dapat dipahami sebagai terdiri dari fungsi AND dan OR; lapisan keluaran kemudian menggabungkan representasi perantara ini untuk menghasilkan keluaran yang diperlukan. Dalam jaringan dalam dengan beberapa lapisan tersembunyi, setiap lapisan tersembunyi berikutnya dapat diartikan sebagai mempelajari representasi yang merupakan abstraksi atas keluaran dari lapisan sebelumnya. Abstraksi sekuensial inilah, melalui pembelajaran representasi perantara, yang memungkinkan jaringan dalam mempelajari pemetaan kompleks seperti itu dari masukan ke keluaran.

Arsitektur Jaringan: Jaringan Neural Konvolusional dan Berulang

Ada banyak cara di mana serangkaian neuron dapat dihubungkan bersama. Contoh jaringan yang disajikan sejauh ini dalam buku ini telah dihubungkan bersama dengan cara yang relatif tidak rumit: neuron diatur ke dalam lapisan dan setiap neuron dalam lapisan terhubung langsung ke semua neuron di lapisan jaringan berikutnya. Jaringan ini dikenal sebagai jaringan feedforward karena tidak ada loop di dalam sambungan jaringan: semua sambungan mengarah ke depan dari masukan ke keluaran. Lebih lanjut, semua contoh jaringan kami sejauh ini akan dianggap terhubung sepenuhnya, karena setiap neuron terhubung ke semuaneuron di lapisan berikutnya. Mungkin, dan seringkali berguna, untuk merancang dan melatih jaringan yang tidak feedforward dan / atau yang tidak sepenuhnya terhubung. Jika dilakukan dengan benar, menyesuaikan arsitektur jaringan dapat dipahami sebagai penyematan informasi arsitektur jaringan tentang properti masalah yang coba dipelajari oleh jaringan untuk dimodelkan.

Contoh yang sangat berhasil dalam menggabungkan pengetahuan domain ke dalam jaringan dengan menyesuaikan arsitektur jaringan adalah desain jaringan saraf konvolusional (CNN) untuk pengenalan objek dalam gambar. Pada 1960-an, Hubel dan Wiesel melakukan serangkaian percobaan pada korteks visual kucing (Hubel dan Wiesel 1962, 1965). Eksperimen ini menggunakan elektroda yang

dimasukkan ke dalam otak kucing yang dibius untuk mempelajari respons sel otak saat kucing diberikan rangsangan visual yang berbeda. Contoh rangsangan yang digunakan termasuk titik terang atau garis cahaya yang muncul di suatu lokasi di bidang visual, atau bergerak melintasi wilayah bidang visual. Percobaan menemukan bahwa sel yang berbeda menanggapi rangsangan yang berbeda di lokasi yang berbeda di bidang visual: efeknya, satu sel di korteks visual akan dihubungkan dengan tipe tertentu dari rangsangan visual yang terjadi dalam wilayah tertentu dari bidang visual. Wilayah bidang visual yang direspons oleh sel dikenal sebagai bidang reseptif sel. Hasil lain dari eksperimen ini adalah perbedaan antara dua jenis sel: "sederhana" dan"kompleks." Untuk sel sederhana, lokasi rangsangan sangat penting dengan sedikit perpindahan rangsangan yang mengakibatkan pengurangan yang signifikan dalam respons sel. Sel kompleks, bagaimanapun, menanggapi rangsangan target mereka terlepas dari di mana di bidang penglihatan rangsangan terjadi. Hubel dan Wiesel (1965) mengusulkan bahwa sel kompleks berperilaku seolah-olah mereka menerima proyeksi dari sejumlah besar sel sederhana yang semuanya merespon rangsangan visual yang sama tetapi berbeda dalam posisi bidang reseptifnya. Hierarki sel sederhana yang masuk ke dalam sel kompleks menghasilkan penyaluran rangsangan dari area luas bidang visual, melalui sekumpulan sel sederhana, ke dalam satu sel kompleks. Gambar 4.4 mengilustrasikan efek corong ini. Gambar ini menunjukkan lapisan sel sederhana yang masing-masing memantau bidang reseptif di lokasi yang berbeda dalam bidang visual. Bidang reseptif dari sel kompleks meliputi lapisan sel sederhana, dansel kompleks ini aktif jika salah satu sel sederhana di bidang reseptifnya aktif. Dengan cara ini, sel kompleks dapat merespons rangsangan visual jika terjadi di lokasi mana pun dalam bidang visual.



Gambar 4.4 Efek corong bidang reseptif yang dibuat oleh hierarki sel sederhana dan kompleks.

Pada akhir 1970-an dan awal 1980-an, Kunihiko Fukushima terinspirasi oleh analisis Hubel dan Wiesel tentang korteks visual dan mengembangkan arsitektur jaringan saraf untuk pengenalan pola visual yang disebut neokognitron (Fukushima 1980). Desain neokognitron didasarkan pada pengamatan bahwa jaringan pengenalan gambar harus dapat mengenali jika fitur visual hadir dalam gambar terlepas dari lokasinya di gambar — atau, secara teknis, jaringan harus dapat untuk melakukan deteksi fitur visual yang tidak berubah secara spasial.

Misalnya, jaringan pengenalan wajah harus dapat mengenali bentuk mata di mana pun dalam gambar itu muncul,

Fukushima menyadari bahwa fungsi sel sederhana dalam hierarki Hubel dan Wiesel dapat direplikasi dalam jaringan saraf menggunakan lapisan neuron yang semuanya menggunakan set bobot yang sama, tetapi dengan setiap neuron menerima masukan dari wilayah kecil tetap (bidang reseptif) di lokasi berbeda di bidang masukan. Untuk memahami hubungan antara neuron berbagi bobot dan spasial invarianDeteksi fitur visual, bayangkan neuron yang menerima sekumpulan nilai piksel, diambil sampelnya dari wilayah gambar, sebagai inputnya. Bobot yang diterapkan neuron ini ke nilai piksel ini menentukan fungsi deteksi fitur visual yang mengembalikan true (aktivasi tinggi) jika fitur visual (pola) tertentu terjadi dalam piksel input, dan false jika sebaliknya. Akibatnya, jika sekumpulan neuron semuanya menggunakan bobot yang sama, mereka semua akan menerapkan detektor fitur visual yang sama. Jika bidang reseptif neuron ini kemudian diatur sehingga bersama-sama menutupi seluruh gambar, maka jika fitur visual muncul di mana saja dalam gambar, setidaknya salah satu neuron dalam kelompok akan mengidentifikasinya dan mengaktifkannya.

Fukushima juga menyadari bahwa efek corong Hubel dan Wiesel (ke dalam sel kompleks) dapat diperoleh oleh neuron di lapisan selanjutnya yang juga menerima sebagai masukan keluaran dari satu set neuron tetap di wilayah kecil di lapisan sebelumnya. Dengan cara ini, neuron di lapisan terakhir jaringan masing-masing menerima masukan dari seluruh bidang masukan sehingga jaringan dapat mengidentifikasi keberadaan fitur visual di mana pun dalam masukan visual.

Beberapa bobot dalam neokognitron ditetapkan dengan tangan, dan lainnya ditetapkan menggunakan proses pelatihan tanpa pengawasan. Dalam proses pelatihan ini, setiap kali contoh disajikan ke jaringan, satu lapisan neuron yang berbagi bobot yang sama dipilih dari lapisan yang menghasilkan keluaran besar sebagai respons terhadap masukan. Bobot neuron di lapisan yang dipilih diperbarui untuk memperkuattanggapan mereka terhadap pola masukan dan bobot neuron yang tidak berada dalam lapisan tidak diperbarui. Pada tahun 1989 Yann LeCun mengembangkan arsitektur convolutional neural network (CNN) khusus untuk tugas pemrosesan gambar (LeCun 1989). Arsitektur CNN berbagi banyak fitur desain yang ditemukan di neocognitron; namun, LeCun menunjukkan bagaimana jenis jaringan ini dapat dilatih menggunakan propagasi mundur. CNN telah terbukti sangat sukses dalam pemrosesan gambar dan tugas lainnya. CNN yang sangat terkenal adalah jaringan AlexNet, yang memenangkan Tantangan Pengenalan Visual Skala Besar ImageNet (ILSVRC) pada tahun 2012 (Krizhevsky et al. 2012). Tujuan dari kompetisi ILSVRC ini adalah untuk mengidentifikasi objek-objek dalam foto. Keberhasilan AlexNet di kompetisi ILSVRC menghasilkan banyak kegembiraan tentang CNN, dan sejak AlexNet sejumlah arsitektur CNN lainnya telah memenangkan persaingan. CNN adalah salah satu jenis jaringan neural dalam yang paling populer, dan bab 5 akan memberikan penjelasan yang lebih mendetail tentangnya.

Jaringan saraf rekuren (RNN) adalah contoh lain dari arsitektur jaringan saraf yang telah disesuaikan dengan karakteristik tertentu dari suatu domain. RNN dirancang untuk memproses data sekuensial, seperti bahasa. Jaringan RNN memproses urutan data (seperti kalimat) satu input pada satu waktu. Sebuah RNN hanya memiliki satu lapisan tersembunyi. Namun, keluaran dari masing-masing neuron tersembunyi ini tidak hanya diumpankan ke neuron keluaran, itu jugajuga disimpan sementara dalam buffer dan kemudian diumpankan kembali ke semua neuron tersembunyi pada input berikutnya. Akibatnya, setiap kali jaringan memproses masukan, setiap neuron di lapisan tersembunyi menerima baik masukan saat ini maupun keluaran, lapisan tersembunyi yang dihasilkan sebagai respons terhadap masukan sebelumnya. Untuk memahami penjelasan ini, mungkin pada titik ini akan membantu untuk langsung beralih ke gambar 5.2 untuk melihat ilustrasi struktur RNN dan aliran informasi melalui jaringan. Perulangan berulang ini, aktivasi dari keluaran lapisan tersembunyi untuk satu masukan yang diumpankan kembali ke lapisan tersembunyi bersama masukan berikutnya, memberikan RNN sebuah memori yang memungkinkannya untuk memproses setiap masukan dalam konteks masukan sebelumnya yang telah diprosesnya .[4] RNN dianggap jaringan dalam karena memori yang berkembang ini dapat dianggap sedalam urutannya.

RNN terkenal awal adalah jaringan Elman. Pada tahun 1990, Jeffrey Locke Elman menerbitkan sebuah makalah yang menggambarkan RNN yang telah dilatih untuk memprediksi akhir dari ucapan dua dan tiga kata sederhana (Elman 1990). Model ini dilatih pada kumpulan data yang disintesis dari kalimat sederhana yang dihasilkan menggunakan tata bahasa buatan. Tata bahasanya dibangun menggunakan leksikon dua puluh tiga kata, dengan setiap kata ditetapkan ke satu kategori leksikal (misalnya, *pria* = KATA BENDA-HUM, *wanita* = KATA BENDA-HUM, *makan* = KATA KERJA-MAKAN, *kue* = KATA BENDA-MAKANAN, dll. .). Menggunakan leksikon ini, tata bahasa mendefinisikan pembuatan lima belas kalimattemplate (mis., NOUN-HUM + VERB-EAT + NOUN-FOOD yang akan menghasilkan kalimat seperti *man eat cookie*). Setelah dilatih, model tersebut dapat menghasilkan kelanjutan yang wajar untuk kalimat, seperti *wanita* + *makan* + *?* = *cookie* . Selanjutnya, setelah jaringan dimulai, ia dapat menghasilkan string yang lebih panjang yang terdiri dari beberapa kalimat, menggunakan konteks yang dihasilkannya sendiri sebagai masukan untuk kata berikutnya, seperti yang diilustrasikan oleh contoh tiga kalimat ini:

gadis makan roti buku gerakan anjing mouse bergerak mouse

Meskipun tugas pembuatan kalimat ini diterapkan pada domain yang sangat sederhana, kemampuan RNN untuk menghasilkan kalimat yang masuk akal diambil sebagai bukti bahwa jaringan saraf dapat memodelkan produktivitas linguistik tanpa memerlukan aturan tata bahasa yang eksplisit. Akibatnya, karya Elman berdampak besar pada psikolinguistik dan psikologi. Kutipan berikut, dari

Churchland 1996, mengilustrasikan pentingnya beberapa peneliti dikaitkan dengan pekerjaan Elman:

> Produktivitas jaringan ini tentu saja merupakan bagian yang lemah dari kapasitas besar yang diperintahkan oleh penutur bahasa Inggris biasa. Tetapi produktivitas adalah produktivitas, dan ternyata jaringan berulang dapat memilikinya. Demonstrasi Elman yang mencolok hampir tidak selesaimasalah antara pendekatan yang berpusat pada aturan untuk tata bahasa dan pendekatan jaringan. Itu akan menjadi waktu untuk bekerja sendiri. Tapi konfliknya sekarang menjadi satu. Saya tidak merahasiakan di mana taruhan saya akan ditempatkan. (Churchland 1996, hlm. 143)[5]

Meskipun RNN bekerja dengan baik dengan data sekuensial, masalah gradien menghilang sangat parah di jaringan ini. Pada tahun 1997, Sepp Hochreiter dan Jürgen Schmidhuber, para peneliti yang pada tahun 1991 telah mempresentasikan penjelasan tentang masalah gradien yang hilang, mengusulkan unit memori jangka pendek (LSTM) sebagai solusi untuk masalah ini di RNNs (Hochreiter dan Schmidhuber 1997). Nama unit ini mengacu pada perbedaan antara bagaimana jaringan saraf mengkodekan memori jangka panjang (dipahami sebagai konsep yang dipelajari selama periode waktu tertentu) melalui pelatihan dan memori jangka pendek (dipahami sebagai respons sistem terhadap rangsangan langsung.). Di jaringan saraf, memori jangka panjang dikodekan melalui penyesuaian bobot jaringan dan setelah dilatih, bobot ini tidak berubah. Memori jangka pendek dikodekan dalam jaringan melalui aktivasi yang mengalir melalui jaringan dan nilai aktivasi ini membusuk dengan cepat. Unit LSTM dirancang untuk memungkinkan memori jangka pendek (aktivasi) di jaringan untuk disebarkan dalam periode waktu yang lama (atau urutan input). Struktur internal LSTM relatif kompleks, dankami akan menjelaskannya di bab 5. Fakta bahwa LSTM dapat menyebarkan aktivasi dalam waktu lama memungkinkan mereka untuk memproses urutan yang mencakup ketergantungan jarak jauh (interaksi antar elemen dalam urutan yang dipisahkan oleh dua atau lebih posisi). Misalnya, ketergantungan antara subjek dan kata kerja dalam kalimat bahasa Inggris: *The **dog / dogs** in that house **is / are** agresif*. Hal ini membuat jaringan LSTM cocok untuk pemrosesan bahasa, dan selama beberapa tahun telah menjadi arsitektur jaringan neural default untuk banyak model pemrosesan bahasa alami, termasuk terjemahan mesin. Misalnya, arsitektur terjemahan mesin sequence-to-sequence (seq2seq) yang diperkenalkan pada tahun 2014 menghubungkan dua jaringan LSTM secara berurutan (Sutskever et al. 2014). Jaringan LSTM pertama, encoder, memproses urutan input satu input pada satu waktu, dan menghasilkan representasi terdistribusi dari input tersebut. Jaringan LSTM pertama disebut encoder karena menyandikan urutan kata menjadi representasi terdistribusi. Jaringan LSTM kedua, decoder, diinisialisasi dengan representasi input terdistribusi dan dilatih untuk menghasilkan urutan output satu elemen pada satu

waktu menggunakan loop umpan balik yang memasukkan elemen output terbaru yang dihasilkan oleh jaringan kembali sebagai input untuk langkah waktu berikutnya. Saat ini, arsitektur seq2seq ini adalah dasar untuk sebagian besar sistem terjemahan mesin modern, dan dijelaskan lebih detail di bab 5.

Pada akhir 1990-an, sebagian besar persyaratan konseptual untuk pembelajaran mendalam sudah ada, termasuk algoritme untuk melatih jaringan dengan banyak lapisan, dan arsitektur jaringan yang masih sangat populer saat ini (CNN dan RNN). Namun, masalah gradien yang menghilang masih menghambat pembuatan jaringan dalam. Juga, dari perspektif komersial, tahun 1990-an (mirip dengan 1960-an) mengalami gelombang hype berdasarkan jaringan saraf dan janji yang belum direalisasikan. Pada saat yang sama, sejumlah terobosan dalam bentuk model pembelajaran mesin lainnya, seperti pengembangan mesin vektor dukungan (SVM), mengalihkan fokus komunitas penelitian pembelajaran mesin dari jaringan saraf: pada saat SVM mencapai hal yang serupa akurasi model jaringan neural tetapi lebih mudah untuk dilatih.

Era Pembelajaran Mendalam

Penggunaan tercatat pertama dari istilah pembelajaran dalam dikreditkan ke Rina Dechter (1986), meskipun dalam makalah Dechter istilah itu tidak digunakan dalam kaitannya dengan jaringan saraf; dan penggunaan pertama istilah dalam kaitannya dengan jaringan saraf dikreditkan ke Aizenberg et al. (2000).[6] Pada pertengahan 2000-an, minat pada jaringan saraf mulai tumbuh, dan itu sudah adakali ini istilah pembelajaran dalam menjadi terkenal untuk menggambarkan jaringan saraf yang dalam. Istilah deep learning digunakan untuk menekankan fakta bahwa jaringan yang dilatih jauh lebih dalam daripada jaringan sebelumnya.

Salah satu keberhasilan awal dari era baru penelitian jaringan saraf ini adalah ketika Geoffrey Hinton dan rekan-rekannya mendemonstrasikan bahwa adalah mungkin untuk melatih jaringan saraf yang dalam menggunakan proses yang dikenal sebagai pelatihan awal bijaksana lapisan rakus. Pra-pelatihan bijak lapisan rakus dimulai dengan melatih satu lapisan neuron yang menerima masukan langsung dari masukan mentah. Ada sejumlah cara berbeda untuk melatih lapisan tunggal neuron ini, tetapi salah satu cara yang populer adalah menggunakan autoencoder. Autoencoder adalah jaringan neural dengan tiga lapisan: lapisan masukan, lapisan tersembunyi (pengodean), dan lapisan keluaran (pengodean). Jaringan dilatih untuk merekonstruksi masukan yang diterimanya di lapisan keluaran; dengan kata lain, jaringan dilatih untuk mengeluarkan nilai yang sama persis dengan yang diterima sebagai masukan. Fitur yang sangat penting dalam jaringan ini adalah bahwa mereka dirancang sedemikian rupa sehingga tidak mungkin jaringan hanya menyalin input ke output. Misalnya, autoencoder mungkin memiliki lebih sedikit neuron di lapisan tersembunyi daripada di lapisan masukan dan keluaran. Karena autoencoder mencoba untuk merekonstruksi

masukan pada lapisan keluaran, fakta bahwa informasi dari masukan harus melewati kemacetan ini di lapisan tersembunyi memaksa autoencoder untuk mempelajari pengkodean data masukan dilapisan tersembunyi yang hanya menangkap fitur terpenting dalam masukan, dan mengabaikan informasi yang berlebihan atau tidak berguna.[7]

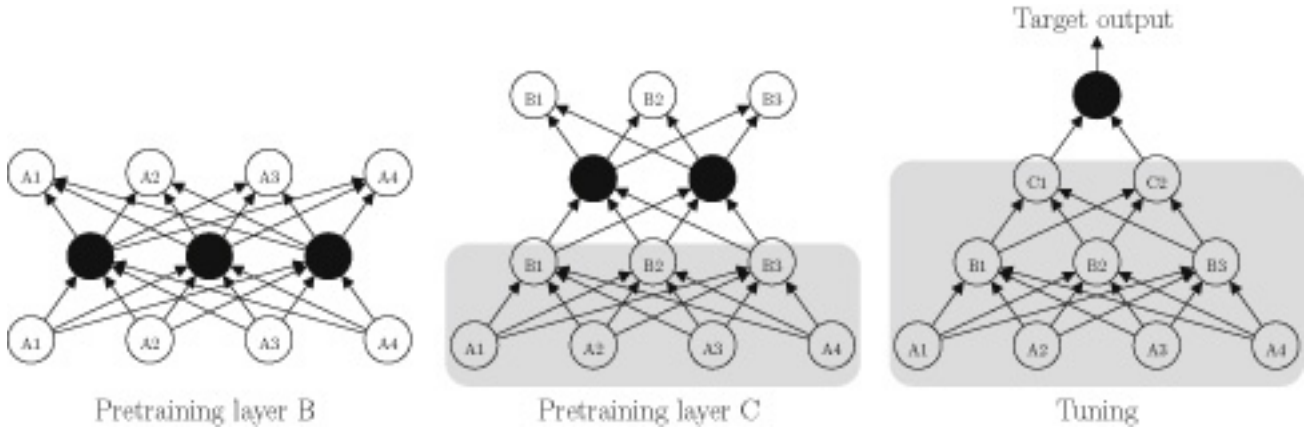
Pretraining Layer-Wise Menggunakan Autoencoders

Dalam pra-pelatihan berdasarkan lapisan, autoencoder awal mempelajari pengkodean untuk input mentah ke jaringan. Setelah pengkodean ini dipelajari, unit di lapisan pengkodean tersembunyi diperbaiki, dan lapisan keluaran (dekode) dibuang. Kemudian autoencoder kedua dilatih — tetapi autoencoder ini dilatih untuk merekonstruksi representasi data yang dihasilkan dengan melewatkannya melalui lapisan encoding dari autoencoder awal. Akibatnya, autoencoder kedua ini ditumpuk di atas lapisan encoding dari autoencoder pertama. Penumpukan lapisan pengkodean ini dianggap sebagai proses rakus karena setiap lapisan pengkodean dioptimalkan secara independen dari lapisan selanjutnya; dengan kata lain,

Setelah sejumlah [8] lapisan encoding telah dilatih, fase penyetelan dapat diterapkan. Dalam fase penyetelan, lapisan jaringan akhir dilatih untuk memprediksi keluaran target jaringan. Berbeda dengan pra-pelatihan dari lapisan jaringan sebelumnya, keluaran target untuk lapisan akhir berbeda dari vektor masukan dan ditentukandalam set data pelatihan. Penyetelan yang paling sederhana adalah tempat lapisan yang dilatih sebelumnya dibekukan (yaitu, bobot di lapisan yang dilatih sebelumnya tidak berubah selama penyetelan); namun, juga layak untuk melatih seluruh jaringan selama fase penyetelan. Jika seluruh jaringan dilatih selama penyetelan, maka pra-pelatihan berdasarkan lapisan paling baik dipahami sebagai menemukan bobot awal yang berguna untuk lapisan sebelumnya dalam jaringan. Selain itu, model prediksi akhir yang dilatih selama penyetelan tidak perlu menjadi jaringan neural. Sangat mungkin untuk mengambil representasi data yang dihasilkan oleh pelatihan awal berdasarkan lapisan dan menggunakannya sebagai representasi input untuk jenis algoritma pembelajaran mesin yang sama sekali berbeda, misalnya, mesin vektor dukungan atau algoritma tetangga terdekat. Skenario ini adalah contoh yang sangat transparan tentang bagaimana jaringan saraf mempelajari representasi data yang berguna sebelum tugas prediksi akhir dipelajari. Sebenarnya, istilah pra-pelatihan hanya menjelaskan pelatihan lapisan bijak dari autoencoder; namun, istilah ini sering digunakan untuk merujuk pada tahap pelatihan bijaksana lapisan dan tahap penyetelan model.

Gambar 4.5 menunjukkan tahapan dalam pra-pelatihan berlapis. Gambar di sebelah kiri mengilustrasikan pelatihan autoencoder awal di mana lapisan encoding (lingkaran hitam) dari tiga unit mencoba untuk mempelajari representasi yang berguna untuk tugas merekonstruksi vektor input dengan panjang 4. Gambar di tengah gambar 4.5 menunjukkan pelatihan autoencoder kedua yang ditumpuk di atas lapisan encodingdari autoencoder pertama. Dalam autoencoder ini, sebuah

lapisan tersembunyi dari dua unit mencoba mempelajari pengkodean untuk vektor input dengan panjang 3 (yang pada gilirannya merupakan pengkodean vektor dengan panjang 4). Latar belakang abu-abu di setiap gambar membatasi komponen dalam jaringan yang dibekukan selama tahap pelatihan ini. Gambar di sebelah kanan menunjukkan fase penyetelan tempat lapisan keluaran akhir dilatih untuk memprediksi fitur target model. Untuk contoh ini, dalam fase penyetelan, lapisan yang dilatih sebelumnya dalam jaringan telah dibekukan.



Gambar 4.5 Tahapan pra-pelatihan dan penyetelan dalam pra-pelatihan berdasarkan lapisan rakus. Lingkaran hitam mewakili neuron yang pelatihannya menjadi tujuan utama di setiap tahap pelatihan. Latar belakang abu-abu menandai komponen dalam jaringan yang dibekukan selama setiap tahap pelatihan.

Pra-pelatihan berdasarkan lapisan penting dalam evolusi pembelajaran mendalam karena ini adalah pendekatan pertama untuk melatih jaringan dalam yang diadopsi secara luas.[9] Namun, saat ini sebagian besar jaringan pembelajaran mendalam dilatihtanpa menggunakan pra-pelatihan berdasarkan lapisan. Pada pertengahan tahun 2000-an, para peneliti mulai memahami bahwa masalah gradien lenyap bukanlah batasan teoretis yang ketat, tetapi justru merupakan kendala praktis yang dapat diatasi. Masalah gradien menghilang tidak menyebabkan gradien kesalahan menghilang seluruhnya; masih ada gradien yang dipropagasi balik melalui lapisan awal jaringan, hanya saja mereka sangat kecil. Saat ini, ada sejumlah faktor yang telah diidentifikasi sebagai faktor penting dalam keberhasilan melatih jaringan dalam.

Pada pertengahan tahun 2000-an, para peneliti mulai memahami bahwa masalah gradien lenyap bukanlah batasan teoretis yang ketat, tetapi justru merupakan kendala praktis yang dapat diatasi.

Fungsi Inisialisasi Bobot dan Aktivasi ULT

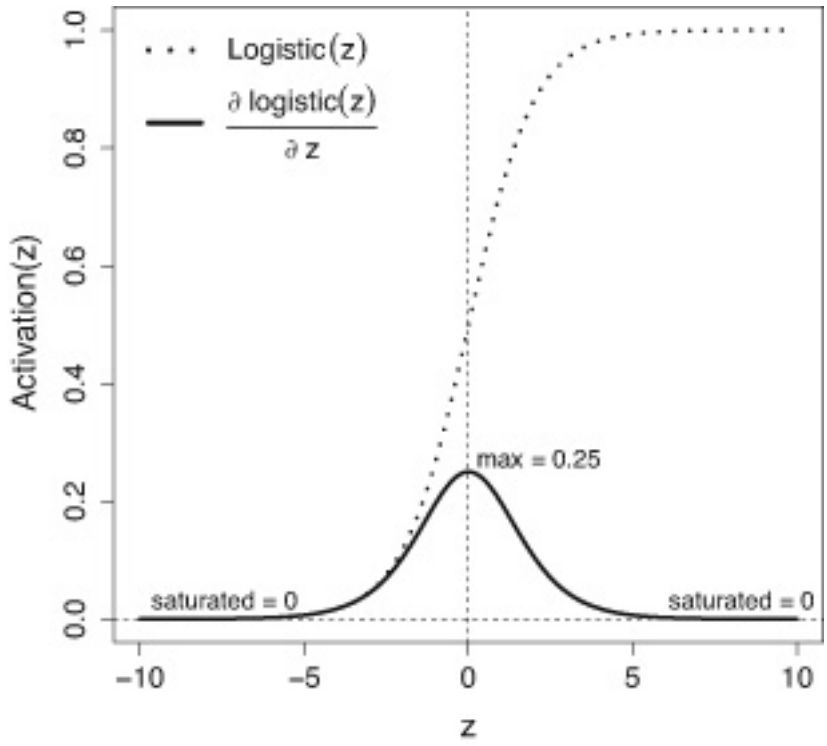
Salah satu faktor yang penting agar berhasil melatih jaringan dalam adalah bagaimana bobot jaringan diinisialisasi. Prinsip-prinsip yang mengontrol bagaimana inisialisasi beban mempengaruhi pelatihan jaringan masih belum jelas. Namun, ada prosedur inisialisasi beban yang secara empiris terbukti membantu

melatih jaringan yang dalam. Inisialisasi Glorot[10] adalah prosedur inisialisasi bobot yang sering digunakan untuk jaringan dalam. Ini didasarkan pada sejumlah asumsi tetapi memiliki keberhasilan empiris untuk mendukung penggunaannya. Untuk mendapatkan pemahaman intuitif tentang inisialisasi Glorot, pertimbangkan fakta bahwa biasanya ada hubungan antara besaran nilai dalam suatu himpunan dan varians himpunan: umumnya semakin besar nilai dalam suatu himpunan, semakin besar varians dari himpunan tersebut. Jadi, jika varian dihitung pada sekumpulan gradien yang disebarkan melalui lapisanpada satu titik di jaringan serupa dengan varian untuk kumpulan gradien yang disebarkan melalui lapisan lain dalam jaringan, kemungkinan besar gradien yang disebarkan melalui kedua lapisan ini juga akan serupa. Lebih lanjut, varian gradien dalam sebuah lapisan dapat dikaitkan dengan varian bobot dalam lapisan tersebut, sehingga strategi potensial untuk mempertahankan gradien yang mengalir melalui jaringan adalah dengan memastikan varian serupa di setiap lapisan dalam jaringan. Inisialisasi Glorot dirancang untuk menginisialisasi bobot dalam jaringan sedemikian rupa sehingga semua lapisan dalam jaringan akan memiliki varian yang sama dalam hal aktivasi forward pass dan gradien yang disebarkan selama backward pass di backpropagation. n_j adalah jumlah neuron dalam lapisan j, dan notasi w ~ U menunjukkan bahwa nilai w diambil sampel dari distribusi U)[11] :

$$w \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

Faktor lain yang berkontribusi terhadap keberhasilan atau kegagalan pelatihan jaringan dalam adalah pemilihan fungsi aktivasi yang digunakan di neuron. Mempropagasi gradien kesalahan melalui neuron melibatkan perkalian gradien dengan nilai turunan dari fungsi aktivasi pada nilai aktivasi neuron yang direkam selama forward pass. Turunan dari fungsi aktivasi logistik dan tanh memiliki sejumlah properti yang dapat memperburuk masalah gradien menghilang jika digunakan dalam langkah perkalian ini. Gambar 4.6 menyajikan plot dari fungsi logistik dan turunan dari fungsi logistik tersebut. Nilai maksimum turunannya adalah 0,25. Akibatnya, setelah gradien kesalahan dikalikandengan nilai turunan dari fungsi logistik pada aktivasi yang sesuai untuk neuron, nilai maksimum yang akan dimiliki gradien adalah seperempat gradien sebelum perkalian. Masalah lain dengan menggunakan fungsi logistik adalah bahwa ada sebagian besar domain dari fungsi di mana fungsinya *jenuh*(mengembalikan nilai yang sangat dekat dengan 0 atau 1), dan laju perubahan fungsi di wilayah ini mendekati nol; dengan demikian, turunan dari fungsinya mendekati 0. Ini adalah properti yang tidak diinginkan saat gradien kesalahan propagasi mundur karena gradien kesalahan akan dipaksa ke nol (atau mendekati nol) saat dipropagasi balik melalui neuron mana pun yang aktivasi berada dalam salah satu daerah jenuh ini. Pada tahun 2011 ditunjukkan bahwa

beralih ke fungsi aktivasi linier yang diperbaiki, $g(x) = \max(0, x)$, peningkatan pelatihan untuk jaringan saraf deep feedforward (Glorot et al. 2011). Neuron yang menggunakan fungsi aktivasi linier tersearah dikenal sebagai satuan linier tersearah (ULT). Salah satu keuntungan ULT adalah bahwa fungsi aktivasi linier untuk bagian positif dari domainnya dengan turunan sama dengan 1. Ini berarti gradien dapat mengalir dengan mudah melalui ULT yang memiliki aktivasi positif. Namun, kekurangan ULT adalah gradien fungsi untuk bagian negatif domainnya adalah nol, sehingga ULT tidak berlatih di bagian domain ini. Meskipun tidak diinginkan, ini tidak selalu merupakan kesalahan fatal untuk pembelajaran karena ketika melakukan propagasi mundur melalui lapisan ULT gradienmasih dapat mengalir melalui ULT di lapisan yang memiliki aktivasi positif. Selain itu, ada sejumlah varian dari ULT dasar yang memperkenalkan gradien di sisi negatif domain, varian yang umum digunakan adalah ULT bocor (Maas et al. 2013). Saat ini, ReLU (atau varian ReLU) adalah neuron yang paling sering digunakan dalam penelitian deep learning.



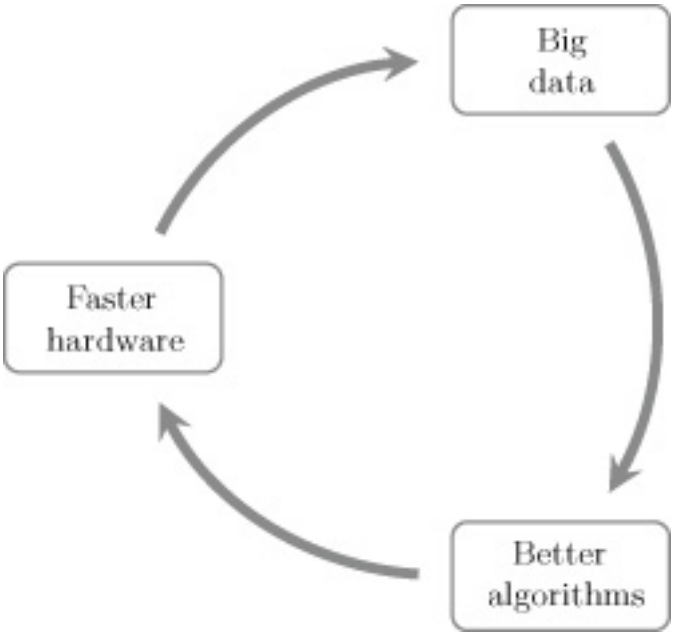
Gambar 4.6 Plot dari fungsi logistik dan turunan dari fungsi logistik.

Siklus Kebajikan: Algoritme Lebih Baik, Perangkat Keras Lebih Cepat, Data Lebih Besar

Meskipun metode inisialisasi bobot yang ditingkatkan dan fungsi aktivasi baru telah berkontribusi pada pertumbuhan pembelajaran mendalam, dalam beberapa tahun terakhir dua faktor terpenting yang mendorong pembelajaran mendalam adalah percepatan dalam kekuatan komputer dan peningkatan besar-besaran dalam ukuran kumpulan data. Dari perspektif komputasi, terobosan besar untuk pembelajaran mendalam terjadi pada akhir tahun 2000-an dengan adopsi unit pemrosesan grafis (GPU) oleh komunitas pembelajaran mendalam untuk mempercepat pelatihan. Jaringan neural dapat dipahami sebagai urutan perkalian

matriks yang diselingi dengan penerapan fungsi aktivasi nonlinier, dan GPU dioptimalkan untuk perkalian matriks yang sangat cepat. Akibatnya, GPU adalah perangkat keras yang ideal untuk mempercepat pelatihan jaringan saraf, dan penggunaannya telah memberikan kontribusi yang signifikan bagi perkembangan lapangan. Pada tahun 2004, Oh dan Jung melaporkan peningkatan kinerja dua puluh kali lipat menggunakan GPUimplementasi jaringan saraf (Oh dan Jung 2004), dan tahun berikutnya dua makalah lebih lanjut diterbitkan yang menunjukkan potensi GPU untuk mempercepat pelatihan jaringan saraf: Steinkraus et al. (2005) menggunakan GPU untuk melatih jaringan saraf dua lapis, dan Chellapilla et al. (2006) menggunakan GPU untuk melatih CNN. Namun, pada saat itu terdapat tantangan pemrograman yang signifikan untuk menggunakan GPU untuk jaringan pelatihan (algoritme pelatihan harus diterapkan sebagai urutan operasi grafis), sehingga adopsi awal GPU oleh peneliti jaringan saraf relatif lambat. Tantangan pemrograman ini berkurang secara signifikan pada tahun 2007 ketika NVIDIA (produsen GPU) merilis antarmuka pemrograman mirip-C untuk GPU yang disebut CUDA (komputasi arsitektur perangkat terpadu).[12] CUDA dirancang khusus untuk memfasilitasi penggunaan GPU untuk tugas komputasi umum. Pada tahun-tahun setelah rilis CUDA, penggunaan GPU untuk mempercepat pelatihan jaringan saraf menjadi standar.

Namun, bahkan dengan prosesor komputer yang lebih kuat ini, pembelajaran mendalam tidak akan mungkin dilakukan kecuali kumpulan data yang besar juga tersedia. Perkembangan platform internet dan media sosial, penyebaran smartphone dan sensor "internet of things", berarti bahwa jumlah data yang ditangkap telah tumbuh dengan kecepatan yang luar biasa selama sepuluh tahun terakhir. Ini mempermudah organisasi untuk mengumpulkan kumpulan data yang besar. Pertumbuhan data ini luar biasapenting untuk pembelajaran mendalam karena model jaringan neural dapat diskalakan dengan baik dengan data yang lebih besar (dan kenyataannya mereka dapat kesulitan dengan kumpulan data yang lebih kecil). Ini juga mendorong organisasi untuk mempertimbangkan bagaimana data ini dapat digunakan untuk mendorong pengembangan aplikasi dan inovasi baru. Hal ini pada gilirannya telah mendorong kebutuhan akan model komputasi baru (yang lebih kompleks) untuk menghasilkan aplikasi baru ini. Dan, kombinasi data yang besar dan algoritme yang lebih kompleks membutuhkan perangkat keras yang lebih cepat agar beban kerja komputasi yang diperlukan dapat diatur. Gambar 4.7 mengilustrasikan siklus baik antara big data, terobosan algoritmik (misalnya, inisialisasi bobot yang lebih baik, ULT, dll.),dan peningkatan perangkat keras yang mendorong revolusi pembelajaran mendalam.



Gambar 4.7 Siklus baik yang mendorong pembelajaran mendalam. Gambar terinspirasi oleh gambar 1.2 di Reagen et al. 2017.

Ringkasan

Sejarah pembelajaran mendalam mengungkapkan sejumlah tema yang mendasarinya. Telah terjadi pergeseran dari input biner sederhana ke input bernilai kontinu yang lebih kompleks. Tren ke arah input yang lebih kompleks ini akan terus berlanjut karena model pembelajaran mendalam paling berguna dalam domain dimensi tinggi, seperti pemrosesan gambar dan bahasa. Gambar sering kali memiliki ribuan piksel di dalamnya, dan pemrosesan bahasa memerlukan kemampuan untuk mewakili dan memproses ratusan ribu kata yang berbeda. Inilah sebabnya mengapa beberapa aplikasi pembelajaran mendalam yang paling terkenal ada di domain ini, misalnya, perangkat lunak pengenalan wajah Facebook, dan sistem terjemahan mesin saraf Google. Namun, ada semakin banyak domain baru tempat kumpulan data digital yang besar dan kompleks dikumpulkan.

Agak mengherankan, inti dari model yang kuat ini adalah unit pemrosesan informasi sederhana: neuron. Ide koneksionis bahwa perilaku kompleks yang berguna bisayang muncul dari interaksi antara sejumlah besar unit pemrosesan sederhana yang masih berlaku hingga saat ini. Perilaku yang muncul ini muncul melalui urutan lapisan dalam jaringan yang mempelajari abstraksi hierarkis dari fitur yang semakin kompleks. Abstraksi hierarkis ini dicapai oleh setiap neuron yang mempelajari transformasi sederhana dari masukan yang diterimanya. Jaringan secara keseluruhan kemudian menyusun urutan transformasi yang lebih kecil ini untuk menerapkan pemetaan nonlinier kompleks (sangat) ke input. Keluaran dari model kemudian dihasilkan oleh lapisan keluaran akhir neuron, berdasarkan representasi yang dipelajari yang dihasilkan melalui abstraksi hierarkis. Inilah sebabnya mengapa kedalaman merupakan faktor penting dalam jaringan saraf: semakin dalam jaringan, semakin kuat model tersebut dalam hal kemampuannya untuk mempelajari pemetaan nonlinier yang kompleks.

Pilihan desain penting dalam membuat jaringan saraf adalah memutuskan fungsi aktivasi mana yang akan digunakan dalam neuron dalam jaringan. Fungsi aktivasi dalam setiap neuron dalam jaringan adalah bagaimana nonlinier dimasukkan ke dalam jaringan, dan sebagai akibatnya ia menjadi komponen yang diperlukan jika jaringan mempelajari pemetaan nonlinier dari masukan ke keluaran. Seiring berkembangnya jaringan, demikian jugamemiliki fungsi aktivasi yang digunakan di dalamnya. Fungsi aktivasi baru telah muncul sepanjang sejarah deep learning, sering kali didorong oleh kebutuhan akan fungsi dengan properti yang lebih baik untuk propagasi error-gradien: faktor utama dalam pergeseran dari ambang batas ke fungsi aktivasi logistik dan tanh adalah kebutuhan akan fungsi yang dapat dibedakan dalam rangka untuk menerapkan propagasi mundur; Pergeseran yang lebih baru ke ULT juga didorong oleh kebutuhan untuk meningkatkan aliran gradien kesalahan melalui jaringan. Penelitian tentang fungsi aktivasi sedang berlangsung, dan fungsi baru akan dikembangkan dan diadopsi di tahun-tahun mendatang.

Pilihan desain penting lainnya dalam membuat jaringan neural adalah memutuskan struktur jaringan: misalnya, bagaimana neuron dalam jaringan harus dihubungkan bersama? Pada bab berikutnya, kita akan membahas dua jawaban yang sangat berbeda untuk pertanyaan ini: jaringan saraf konvolusi dan jaringan saraf berulang.

5

Jaringan Neural Konvolusional dan Berulang

Menyesuaikan struktur jaringan dengan karakteristik spesifik data dari domain tugas dapat mengurangi waktu pelatihan jaringan, dan meningkatkan akurasi jaringan. Penyesuaian dapat dilakukan dengan berbagai cara, seperti: membatasi koneksi antara neuron dalam lapisan yang berdekatan dengan subset (daripada memiliki lapisan yang terhubung sepenuhnya); memaksa neuron untuk berbagi beban; atau memperkenalkan koneksi mundur ke jaringan. Menyesuaikan dengan cara ini dapat dipahami sebagai membangun pengetahuan domain ke dalam jaringan. Perspektif lain, terkait, adalah membantu jaringan untuk belajar dengan membatasi serangkaian fungsi yang mungkin dapat dipelajari, dan dengan demikian memandu jaringan untuk menemukan solusi yang berguna. Tidak selalu jelas bagaimana menyesuaikan struktur jaringan dengan domain,data seperti gambar) ada arsitektur jaringan terkenal yang terbukti berhasil. Bab ini akan memperkenalkan dua arsitektur pembelajaran mendalam yang paling populer: jaringan saraf konvolusional dan jaringan saraf berulang.

Jaringan Neural Konvolusional

Jaringan saraf konvolusi (CNN) dirancang untuk tugas pengenalan gambar dan pada awalnya diterapkan untuk tantangan pengenalan digit tulisan tangan (Fukushima 1980; LeCun 1989). Tujuan desain dasar CNN adalah untuk membuat jaringan di mana neuron di lapisan awal jaringan akan mengekstrak fitur visual lokal, dan neuron di lapisan selanjutnya akan menggabungkan fitur ini untuk membentuk fitur tingkat tinggi. Fitur visual lokal adalah fitur yang jangkauannya terbatas pada patch kecil, sekumpulan piksel yang berdekatan, dalam sebuah gambar. Misalnya, ketika diterapkan pada tugas pengenalan wajah, neuron di lapisan awal CNN belajar untuk mengaktifkan sebagai respons terhadap fitur lokal sederhana (seperti garis pada sudut tertentu, atau segmen kurva),

Dengan menggunakan pendekatan ini, tugas mendasar dalam pengenalan gambar adalah mempelajari fungsi deteksi fitur yang dapat dengan kuat mengidentifikasi ada atau tidaknya fitur visual lokal dalam suatu gambar. Proses fungsi pembelajaran adalah inti dari jaringan saraf, dan dicapai dengan mempelajari kumpulan bobot yang sesuai untuk koneksi di jaringan. CNN mempelajari fungsi deteksi fitur untuk fitur visual lokal dengan cara ini. Namun, tantangan terkait adalah merancang arsitektur jaringan sehingga jaringan akan mengidentifikasi keberadaan fitur visual lokal pada citra terlepas dari di mana citra itu muncul. Dengan kata lain, fungsi deteksi fitur harus dapat bekerja dengan cara yang tidak berubah. Sebagai contoh, Sistem pengenalan wajah harus dapat mengenali bentuk mata pada suatu citra baik mata berada di tengah-tengah maupun di pojok kanan atas citra. Kebutuhan invariansi terjemahan ini telah menjadi prinsip desain utama CNN untuk pemrosesan gambar, seperti yang dinyatakan oleh Yann LeCun pada tahun 1989:

> Tampaknya berguna untuk memiliki sekumpulan detektor fitur yang dapat mendeteksi contoh tertentu dari fitur di mana saja pada bidang masukan. Karena lokasi yang tepat dari fitur tidak relevan dengan klasifikasi, kami dapat kehilangan beberapa informasi posisi dalam prosesnya. (LeCun 1989, hlm.14)

CNN mencapai invariansi terjemahan dari deteksi fitur visual lokal ini dengan menggunakan pembagian bobot antar neuron. Dalam pengaturan pengenalan gambar, fungsi yang diimplementasikan oleh neuron dapat dipahami sebagai pendeteksi fitur visual. Misalnya, neuron di lapisan tersembunyi pertama jaringan akan menerima sekumpulan nilai piksel sebagai input dan output aktivasi tinggi jika pola tertentu (fitur visual lokal) hadir dalam kumpulan piksel ini. Fakta bahwa fungsi yang diimplementasikan oleh neuron ditentukan oleh bobot yang digunakan neuron berarti bahwa jika dua neuron menggunakan set bobot yang sama maka keduanya menerapkan fungsi yang sama (detektor fitur). Dalam bab 4, kami memperkenalkan konsep bidang reseptif untuk mendeskripsikan area tempat neuron menerima inputnya. Jika dua neuron berbagi bobot yang sama tetapi memiliki bidang reseptif yang berbeda (mis , setiap neuron memeriksa area input yang berbeda), kemudian bersama-sama, neuron bertindak sebagai detektor fitur

yang aktif jika fitur tersebut terjadi di salah satu bidang reseptif. Akibatnya, dimungkinkan untuk merancang jaringan dengan deteksi fitur invarian terjemahan dengan membuat sekumpulan neuron yang berbagi bobot yang sama dan yang diatur sedemikian rupa sehingga: (1) setiap neuron memeriksa bagian gambar yang berbeda; dan (2) bersama-sama bidang reseptif dari neuron menutupi seluruh gambar. (1) setiap neuron memeriksa bagian gambar yang berbeda; dan (2) bersama-sama bidang reseptif dari neuron menutupi seluruh gambar. (1) setiap neuron memeriksa bagian gambar yang berbeda; dan (2) bersama-sama bidang reseptif dari neuron menutupi seluruh gambar.

Skenario pencarian gambar di ruangan gelap dengan senter yang memiliki pancaran sempit terkadang digunakan untuk menjelaskan bagaimana CNN mencari fitur lokal pada gambar.Setiap saat, Anda dapat mengarahkan senter ke suatu wilayah gambar dan memeriksa wilayah setempat itu. Dalam metafora senter ini, area gambar yang diterangi oleh senter setiap saat setara dengan bidang reseptif dari satu neuron, sehingga mengarahkan senter ke suatu lokasi setara dengan menerapkan fungsi deteksi fitur ke wilayah lokal tersebut. Namun, jika Anda ingin memastikan bahwa Anda memeriksa keseluruhan gambar, Anda mungkin memutuskan untuk lebih sistematis dalam mengarahkan senter. Misalnya, Anda dapat memulai dengan mengarahkan senter ke pojok kiri atas gambar dan memeriksa wilayah itu. Anda kemudian memindahkan senter ke kanan, melintasi gambar, memeriksa setiap lokasi baru saat terlihat, hingga Anda mencapai sisi kanan gambar. Anda kemudian mengarahkan senter kembali ke kiri gambar, tetapi tepat di bawah tempat Anda memulai, dan bergerak melintasi gambar lagi. Ulangi proses ini hingga Anda mencapai sudut kanan bawah gambar. Proses pencarian secara berurutan di seluruh gambar dan di setiap lokasi dalam pencarian yang menerapkan fungsi yang sama ke wilayah lokal (yang diterangi) adalah inti dari menggabungkan fungsi di seluruh gambar. Dalam CNN, pencarian sekuensial di seluruh gambar ini diimplementasikan menggunakan sekumpulan neuron yang berbagi bobot dan yang gabungan bidang reseptifnya mencakup seluruh gambar. Proses pencarian secara berurutan di seluruh gambar dan di setiap lokasi dalam pencarian yang menerapkan fungsi yang sama ke wilayah lokal (diterangi) adalah inti dari menggabungkan fungsi di seluruh gambar. Dalam CNN, pencarian sekuensial di seluruh gambar ini diimplementasikan menggunakan sekumpulan neuron yang berbagi bobot dan yang gabungan bidang reseptifnya mencakup seluruh gambar. Proses pencarian secara berurutan di seluruh gambar dan di setiap lokasi dalam pencarian yang menerapkan fungsi yang sama ke wilayah lokal (diterangi) adalah inti dari menggabungkan fungsi di seluruh gambar. Dalam CNN, pencarian sekuensial di seluruh gambar ini diimplementasikan menggunakan sekumpulan neuron yang berbagi bobot dan yang gabungan bidang reseptifnya mencakup seluruh gambar.

Gambar 5.1 mengilustrasikan berbagai tahapan pemrosesan yang sering ditemukan di CNN. The 6×6 matriks di sebelah kiri gambar mewakili gambar yang input keCNN. The 4×4 matriks segera di sebelah kanan input merupakan lapisan neuron yang bersama-sama mencari seluruh gambar untuk kehadiran fitur

lokal tertentu. Setiap neuron di lapisan ini terhubung ke 3×3 bidang reseptif yang berbeda (area) pada gambar, dan semuanya menerapkan matriks bobot yang sama ke inputnya:

$$\begin{bmatrix} w_0 & w_1 & w_2 \\ w_3 & w_4 & w_5 \\ w_6 & w_7 & w_8 \end{bmatrix}$$

Bidang reseptif neuron $[0,0]$ (kiri atas) di lapisan ini ditandai dengan kotak abu-abu yang menutupi 3×3 area di kiri atas gambar masukan. Panah putus-putus muncul dari masing-masing lokasi di area abu-abu ini mewakili masukan ke neuron $[0,0]$. Bidang reseptif dari neuron tetangga $[0,1]$ ditunjukkan oleh 3×3 persegi, dengan garis tebal pada gambar masukan. Perhatikan bahwa bidang reseptif dari kedua neuron ini tumpang tindih. Jumlah tumpang tindih bidang reseptif dikontrol oleh hyperparameter yang disebut panjang langkah. Dalam hal ini, panjang langkah adalah satu, artinya untuk setiap posisi yang digerakkan di lapisan bidang reseptif neuron diterjemahkan dengan jumlah yang sama pada input. Jika hyperparameter panjang langkah ditingkatkan, jumlah tumpang tindih antara bidang reseptif berkurang.

Bidang reseptif dari kedua neuron ini ($[0,0]$ dan $[0,1]$) adalah matriks nilai piksel dan bobot yang digunakan oleh neuron ini juga merupakan matriks. Dalam visi komputer, matriks bobot yang diterapkan ke input dikenal sebagai kernel (atau topeng konvolusi); operasi penerusan kernel secara berurutan di seluruh gambar dan di dalam setiap wilayah lokal, memberi bobot pada setiap masukan dan menambahkan hasilnya ke tetangga lokalnya, dikenal sebagai konvolusi. Perhatikan bahwa operasi konvolusi tidak menyertakan fungsi aktivasi nonlinier (ini diterapkan pada tahap pemrosesan selanjutnya). Kernel mendefinisikan fungsi deteksi fitur yang diterapkan oleh semua neuron dalam konvolusi. Menyatukan kernel di seluruh gambar sama dengan meneruskan detektor fitur visual lokal ke seluruh gambar dan merekam semua lokasi di gambar tempat fitur visual itu ada. Keluaran dari proses ini adalah peta dari semua lokasi pada gambar dimana fitur visual yang relevan terjadi. Untuk alasan ini, keluaran dari proses konvolusi terkadang dikenal sebagai peta fitur. Seperti disebutkan di atas, operasi konvolusi tidak mencakup fungsi aktivasi nonlinier (ini hanya melibatkan penjumlahan input berbobot). Akibatnya, merupakan standar untuk menerapkan operasi nonlinier ke peta fitur. Seringkali, ini dilakukan dengan menerapkan fungsi linier yang diperbaiki ke setiap posisi dalam peta fitur; fungsi aktivasi linier yang diperbaiki didefinisikan sebagai: operasi konvolusi tidak mencakup fungsi aktivasi nonlinier (ini hanya melibatkan penjumlahan input yang berbobot). Akibatnya, merupakan standar untuk menerapkan operasi nonlinier ke peta fitur. Seringkali, ini dilakukan dengan menerapkan fungsi linier yang diperbaiki ke setiap posisi dalam peta fitur;

fungsi aktivasi linier yang diperbaiki didefinisikan sebagai: operasi konvolusi tidak mencakup fungsi aktivasi nonlinier (ini hanya melibatkan penjumlahan input yang berbobot). Akibatnya, merupakan standar untuk menerapkan operasi nonlinier ke peta fitur. Seringkali, ini dilakukan dengan menerapkan fungsi linier yang diperbaiki ke setiap posisi dalam peta fitur; fungsi aktivasi linier yang diperbaiki didefinisikan sebagai: $\text{rectifier}(z) = \max(0, z)$. Meneruskan fungsi aktivasi linier yang diperbaiki melalui peta fitur akan mengubah semuanyanilai negatif menjadi 0. Pada gambar 5.1, proses pembaruan peta fitur dengan menerapkan fungsi aktivasi linier yang diperbaiki ke setiap elemennya diwakili oleh lapisan berlabel Nonlinier.

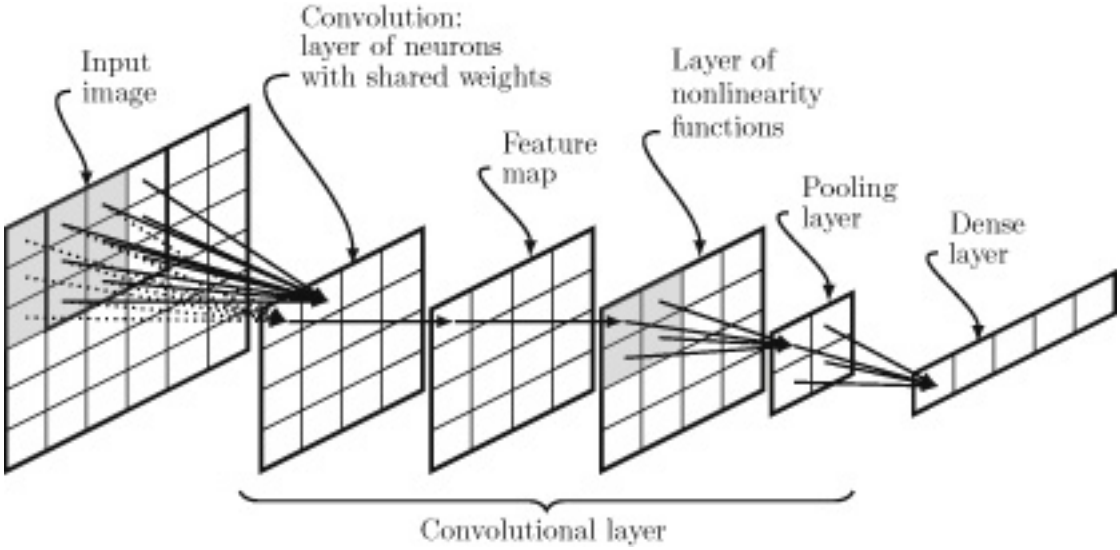
Kutipan dari Yann LeCun, di awal bagian ini, menyebutkan bahwa lokasi fitur yang tepat dalam sebuah gambar mungkin tidak relevan dengan tugas pemrosesan gambar. Dengan pemikiran ini, CNN sering membuang informasi lokasi demi menggeneralisasi kemampuan jaringan untuk melakukan klasifikasi gambar. Biasanya, ini dicapai dengan mengambil sampel peta fitur yang diperbarui menggunakan lapisan penggabungan. Dalam beberapa hal, penggabungan mirip dengan operasi konvolusi yang dijelaskan di atas, sejauh penggabungan melibatkan penerapan berulang kali fungsi yang sama di seluruh ruang masukan. Untuk penggabungan, ruang input sering kali berupa peta fitur yang elemennya telah diperbarui menggunakan fungsi linier yang diperbaiki. Selain itu, setiap operasi penggabungan memiliki bidang reseptif pada ruang masukan — meskipun, untuk penggabungan, bidang reseptif terkadang tidak tumpang tindih. Ada sejumlah fungsi penggabungan berbeda yang digunakan; yang paling umum disebut penggabungan maksimal, yang mengembalikan nilai maksimum dari setiap inputnya. Menghitung nilai rata-rata input juga digunakan sebagai fungsi penggabungan.

Menyatukan kernel di seluruh gambar sama dengan meneruskan detektor fitur visual lokal di seluruh gambar dan merekam semua lokasi di gambar tempat fitur visual itu ada.

Urutan operasi penerapan konvolusi, diikuti oleh nonlinier, ke peta fitur, dan kemudian pengambilan sampel menggunakan penggabungan, relatif standar di sebagian besar CNN. Seringkali ketiga operasi ini digabungkandianggap mendefinisikan lapisan konvolusional dalam jaringan, dan ini disajikan dalam gambar 5.1.

Fakta bahwa konvolusi mencari seluruh gambar berarti bahwa jika fitur visual (pola piksel) yang dideteksi oleh fungsi tersebut (ditentukan oleh kernel bersama) terjadi di mana saja dalam gambar, keberadaannya akan direkam di peta fitur (dan jika penggabungan digunakan , juga dalam keluaran berikutnya dari lapisan penggabungan). Dengan cara ini, CNN mendukung deteksi fitur visual invarian terjemahan. Namun, ini memiliki batasan bahwa konvolusi hanya dapat mengidentifikasi satu jenis fitur. CNN menggeneralisasi lebih dari satu fitur

dengan melatih beberapa lapisan konvolusional secara paralel (atau filter), dengan setiap filter mempelajari satumatriks kernel (fungsi deteksi fitur). Perhatikan lapisan konvolusi pada gambar 5.1 mengilustrasikan filter tunggal. Keluaran dari beberapa filter dapat diintegrasikan dengan berbagai cara. Salah satu cara untuk mengintegrasikan informasi dari filter yang berbeda adalah dengan mengambil peta fitur yang dihasilkan oleh filter terpisah dan menggabungkannya ke dalam satu peta fitur multifilter. Lapisan konvolusional berikutnya kemudian mengambil peta fitur multifilter ini sebagai masukan. Cara lain untuk mengintegrasikan informasi dari filter yang berbeda adalah dengan menggunakan lapisan neuron yang terhubung dengan rapat. Lapisan terakhir pada gambar 5.1 mengilustrasikan lapisan padat. Lapisan padat ini beroperasi dengan cara yang persis sama seperti lapisan standar dalam jaringan feedforward yang terhubung sepenuhnya. Setiap neuron di lapisan padat terhubung ke semua elemen yang dihasilkan oleh masing-masing filter, dan setiap neuron mempelajari satu set bobot yang unik untuk dirinya sendiri yang diterapkan pada input. Ini berarti bahwa setiap neuron dalam lapisan padat dapat mempelajari cara berbeda untuk mengintegrasikan informasi dari berbagai filter yang berbeda.



Gambar 5.1 Ilustrasi berbagai tahapan pemrosesan dalam lapisan konvolusional. Catatan dalam gambar ini Gambar dan Peta Fitur adalah struktur data; tahapan lainnya merepresentasikan operasi pada data.

CNN AlexNet, yang memenangkan Tantangan Pengenalan Visual Skala Besar ImageNet (ILSVRC) pada tahun 2012, memiliki lima lapisan konvolusional, diikuti oleh tiga lapisan padat. Lapisan konvolusional pertama memiliki sembilan puluh enam kernel (atau filter) yang berbeda dan termasuk nonlinier ReLU dan penggabungan. Lapisan konvolusi kedua memiliki 256 kernel dan juga termasuk nonlinier ReLU dan penggabungan. Ketiga,lapisan konvolusional keempat, dan kelima tidak menyertakan langkah nonlinier atau penggabungan, dan memiliki masing-masing 384, 384, dan 256 kernel. Mengikuti lapisan konvolusional kelima, jaringan memiliki tiga lapisan padat dengan masing-masing 4096 neuron. Secara total, AlexNet memiliki enam puluh juta bobot dan 650.000 neuron. Meskipun enam puluh juta bobot adalah jumlah yang besar, fakta bahwa banyak neuron yang

berbagi bobot sebenarnya mengurangi jumlah bobot dalam jaringan. Pengurangan jumlah bobot yang dibutuhkan ini merupakan salah satu keunggulan jaringan CNN. Pada 2015, Microsoft Research mengembangkan jaringan CNN yang disebut ResNet, yang memenangkan tantangan ILSVRC 2015 (He et al. 2016). Arsitektur ResNet memperluas arsitektur CNN standar menggunakan koneksi lewati. Lewati koneksi mengambil output dari satu lapisan di jaringan dan memasukkannya langsung ke lapisan yang mungkin jauh lebih dalam di jaringan. Menggunakan koneksi lewati memungkinkan untuk melatih jaringan yang sangat dalam. Faktanya, model ResNet yang dikembangkan oleh Microsoft Research memiliki kedalaman 152 lapisan.

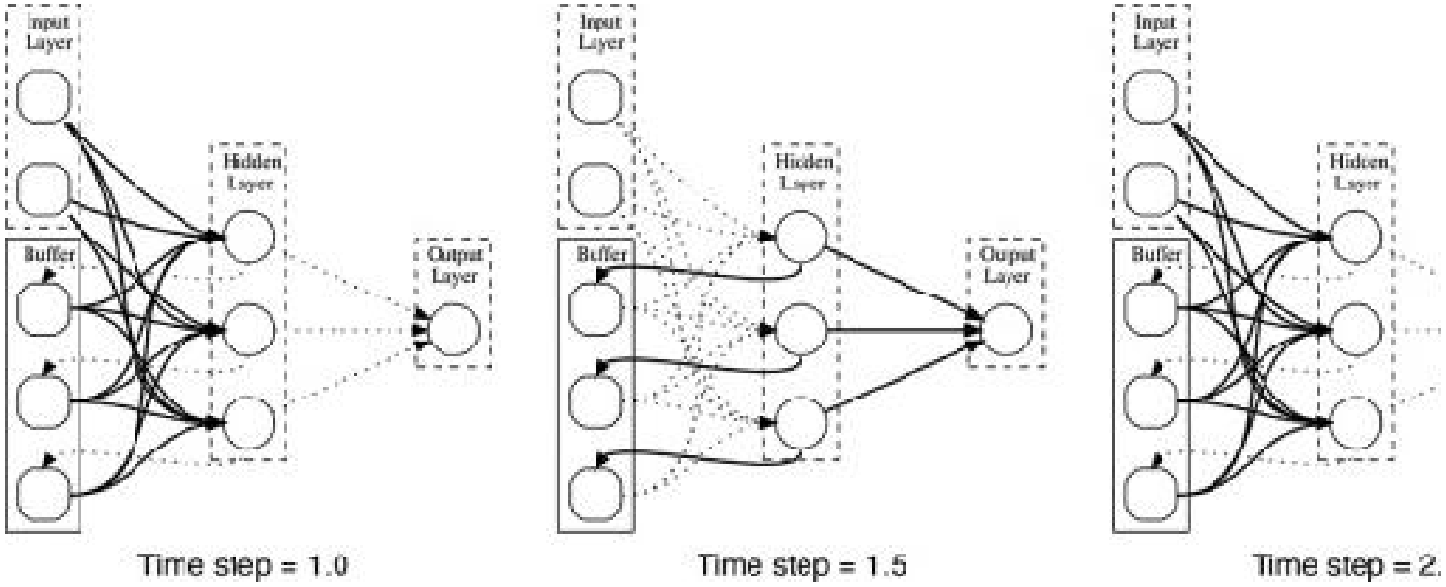
Jaringan Neural Berulang

Jaringan saraf berulang (RNN) disesuaikan untuk pemrosesan data sekuensial. Sebuah RNN memproses urutan data dengan memproses setiap elemen dalam urutan satu per satu. Jaringan RNN hanya memiliki satu jaringan tersembunyilapisan, tetapi juga memiliki penyangga memori yang menyimpan keluaran dari lapisan tersembunyi ini untuk satu masukan dan memasukkannya kembali ke lapisan tersembunyi bersama dengan masukan berikutnya dari urutan. Aliran informasi yang berulang ini berarti bahwa jaringan memproses setiap masukan dalam konteks yang dihasilkan dengan memproses masukan sebelumnya, yang selanjutnya diproses dalam konteks masukan sebelumnya. Dengan cara ini, informasi yang mengalir melalui loop berulang mengkodekan informasi kontekstual dari (berpotensi) semua input sebelumnya dalam urutan tersebut. Hal ini memungkinkan jaringan untuk mempertahankan memori dari apa yang telah dilihat sebelumnya dalam urutan untuk membantunya memutuskan apa yang harus dilakukan dengan masukan saat ini. Kedalaman RNN muncul dari fakta bahwa vektor memori disebarkan ke depan dan berkembang melalui setiap masukan dalam urutan;

Kedalaman RNN muncul dari fakta bahwa vektor memori disebarkan ke depan dan berkembang melalui setiap masukan dalam urutan; Akibatnya, jaringan RNN dianggap sedalam urutannya panjang.

Gambar 5.2 mengilustrasikan arsitektur RNN dan menunjukkan bagaimana informasi mengalir melalui jaringan saat memproses suatu urutan. Pada setiap langkah waktu, jaringan pada gambar ini menerima vektor yang mengandung dua elemen sebagai masukan. Skema di sebelah kiri gambar 5.2 (langkah waktu = 1.0) menunjukkan aliran informasi dalam jaringan ketika menerima masukan pertama dalam urutan. Vektor masukan ini dimasukkan ke dalam tiga neuron di lapisan tersembunyi jaringan. Pada saat yang sama neuron ini juga menerima informasi apa pun yang disimpan di dalam memoripenyangga. Karena ini adalah masukan

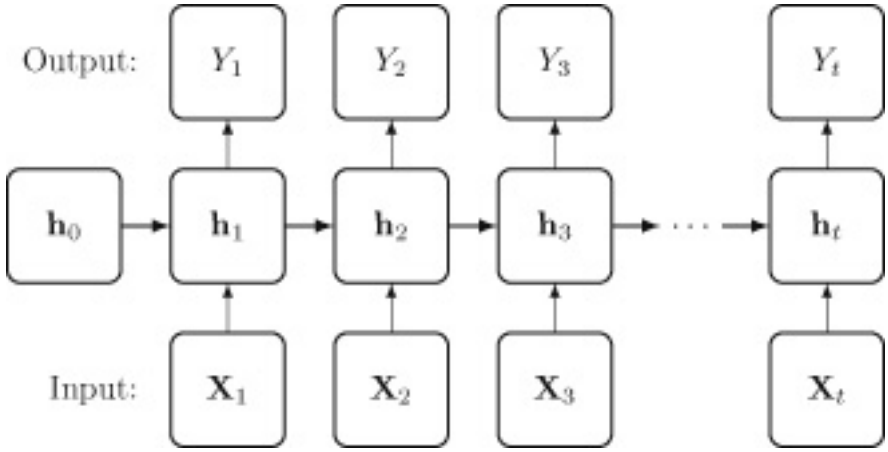
awal, buffer memori hanya akan berisi nilai inisialisasi default. Setiap neuron di lapisan tersembunyi akan memproses masukan dan menghasilkan aktivasi. Skema di tengah gambar 5.2 (langkah waktu = 1.5) menunjukkan bagaimana aktivasi ini mengalir melalui jaringan: aktivasi setiap neuron diteruskan ke lapisan keluaran di mana ia diproses untuk menghasilkan keluaran jaringan, dan itu terjadi. juga disimpan di buffer memori (menimpa informasi apa pun yang disimpan di sana). Unsur-unsur penyangga memori hanya menyimpan informasi yang tertulis padanya; mereka tidak mengubahnya dengan cara apa pun. Akibatnya, tidak ada bobot pada edge yang berpindah dari unit tersembunyi ke buffer. Namun, ada bobot di semua sisi lain dalam jaringan, termasuk yang berasal dari unit penyangga memori ke neuron di lapisan tersembunyi. Pada langkah waktu 2, jaringan menerima masukan berikutnya dari urutan, dan ini diteruskan ke neuron lapisan tersembunyi bersama dengan informasi yang disimpan di buffer. Kali ini buffer berisi aktivasi yang dihasilkan oleh neuron tersembunyi sebagai respons terhadap masukan pertama.



Gambar 5.2 Aliran informasi dalam RNN saat memproses urutan input. Panah yang dicetak tebal adalah jalur aktif arus informasi di setiap titik waktu; panah putus-putus menunjukkan koneksi yang tidak aktif pada saat itu.

Gambar 5.3 menunjukkan RNN yang telah dibuka gulungannya sepanjang waktu saat memproses urutan input $[X_1, X_2, \ldots, X_t]$. Setiap kotak pada gambar ini mewakili lapisan neuron. Kotak berlabel h_0 mewakili keadaan buffer memori saat jaringan diinisialisasi; kotak berlabel $[h_1, \ldots, h_t]$ mewakili lapisan tersembunyi darijaringan di setiap langkah waktu; dan kotak berlabel $[Y_1, \ldots, Y_t]$ mewakili lapisan keluaran jaringan pada setiap langkah waktu. Setiap panah pada gambar mewakili satu set koneksi antara satu lapisan dan lapisan lainnya. Misalnya, panah vertikal dari X_1 ke h_1 mewakili koneksi antara lapisan masukan dan lapisan tersembunyi pada langkah waktu 1. Demikian pula, panah horizontal yang menghubungkan lapisan tersembunyi mewakili penyimpanan aktivasi dari keadaan

tersembunyi pada satu langkah waktu dalam buffer memori (tidak ditampilkan) dan propagasi aktivasi ini ke lapisan tersembunyi pada langkah waktu berikutnya melalui koneksi dari buffer memori ke status tersembunyi. Pada setiap langkah waktu, masukan dari urutan disajikan ke jaringan dan diumpankan ke lapisan tersembunyi. Lapisan tersembunyi menghasilkan vektor aktivasi yang diteruskan ke lapisan keluaran dan juga disebarkan kelangkah selanjutnya sepanjang panah horizontal yang menghubungkan status tersembunyi.



Gambar 5.3 Jaringan RNN membuka gulungan sepanjang waktu saat memproses urutan input $[X_1, X_2, \ldots, X_t]$.

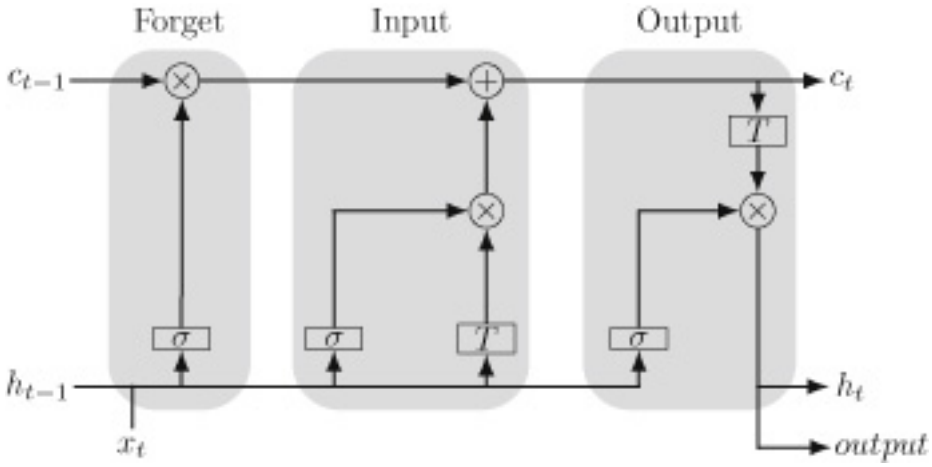
Meskipun RNN dapat memproses urutan input, RNN berjuang dengan masalah gradien yang hilang. Ini karena melatih RNN untuk memproses urutan input membutuhkan kesalahan untuk di-propagasi balik melalui seluruh panjang urutan. Misalnya, untuk jaringan pada gambar 5.3, kesalahan yang dihitung pada output Y_t harus di-propagasi balik melalui seluruh jaringan sehingga dapat digunakan untuk memperbarui bobot pada koneksi dari h_0 dan X_1 ke h_1. Ini memerlukan backpropagating kesalahan melalui semua lapisan tersembunyi, yang pada gilirannya melibatkan berulang kali mengalikan kesalahan dengan bobot pada koneksi yang memberi aktivasi dari satu lapisan tersembunyi ke lapisan tersembunyi berikutnya. Masalah khusus dengan proses ini adalah bahwa ini adalah kumpulan bobot yang sama yang digunakan pada semua koneksi antara lapisan tersembunyi: setiap panah horizontal mewakili himpunan koneksi yang sama antara buffer memori dan lapisan tersembunyi, dan bobot pada ini. koneksi tidak bergerak sepanjang waktu (yaitu, mereka tidak berubah dari satu langkah waktu ke langkah berikutnya selama pemrosesan urutan input tertentu). Akibatnya, backpropogating kesalahan melalui *k* langkah waktu melibatkan (di antara perkalian lainnya) mengalikan gradien kesalahan dengan set bobot yang sama*k* kali. Ini sama dengan mengalikan setiap gradien kesalahan dengan bobot yang dipangkatkan *k* . Jika berat inikurang dari 1, kemudian ketika dinaikkan ke suatu pangkat, ia berkurang pada tingkat eksponensial, dan akibatnya, gradien kesalahan juga cenderung berkurang pada tingkat eksponensial sehubungan dengan panjang urutan — dan menghilang.

Jaringan memori jangka pendek (LSTM) dirancang untuk mengurangi efek gradien yang hilang dengan menghilangkan perkalian berulang dengan vektor bobot yang sama selama propagasi mundur dalam RNN. Inti dari LSTM [1] unit adalah komponen yang disebut sel. Sel adalah tempat aktivasi (memori jangka pendek) disimpan dan disebarkan ke depan. Faktanya, sel sering mempertahankan vektor aktivasi. Propagasi aktivasi dalam sel melalui waktu dikendalikan oleh tiga komponen yang disebut gerbang: gerbang lupa, gerbang input, dan gerbang output. Gerbang lupa bertanggung jawab untuk menentukan aktivasi mana dalam sel yang harus dilupakan pada setiap langkah waktu, gerbang input mengontrol bagaimana aktivasi dalam sel harus diperbarui sebagai respons terhadap input baru, dan gerbang output mengontrol aktivasi apa yang harus digunakan untuk menghasilkan keluaran sebagai respons terhadap masukan saat ini. Setiap gerbang terdiri dari lapisan neuron standar, dengan satu neuron di lapisan per aktivasi dalam keadaan sel.

Gambar 5.4 mengilustrasikan struktur internal sel LSTM. Masing-masing panah pada gambar ini mewakili vektor aktivasi. Sel berada di sepanjang bagian atas gambar dari kiri (c_{t-1}) ke kanan (c_t). Aktivasi di dalam seldapat mengambil nilai dalam rentang -1 hingga +1. Melangkah melalui pemrosesan untuk satu masukan, vektor masukan x_t pertama-tama digabungkan dengan vektor keadaan tersembunyi yang telah disebarkan maju dari langkah waktu sebelumnya h_{t-1}. Bekerja dari kiri ke kanan melalui pemrosesan gerbang, gerbang lupa mengambil rangkaian input dan status tersembunyi dan melewati vektor ini melalui lapisan neuron yang menggunakan sigmoid (juga dikenal sebagai logistik) [2] fungsi aktivasi. Sebagai hasil dari neuron di lapisan lupa menggunakan fungsi aktivasi sigmoid, keluaran dari lapisan lupa ini adalah vektor nilai dalam kisaran 0 sampai 1. Status sel kemudian dikalikan dengan vektor lupa ini. Hasil dari perkalian ini adalah aktivasi dalam keadaan sel yang dikalikan dengan komponen pada vektor lupakan dengan nilai mendekati 0 akan terlupakan, dan aktivasi yang dikalikan dengan komponen vektor lupakan dengan nilai mendekati 1 akan diingat. Akibatnya, mengalikan keadaan sel dengan keluaran dari lapisan sigmoid bertindak sebagai filter pada keadaan sel.

Selanjutnya, gerbang input memutuskan informasi apa yang harus ditambahkan ke status sel. Pengolahan pada langkah ini dilakukan oleh komponen-komponen di blok tengah gambar 5.4 yang bertanda Input. Pemrosesan ini dipecah menjadi dua subbagian. Pertama, gerbang memutuskan elemen mana dalam status sel yang harus diperbarui, dan kedua memutuskan informasi apa yang harus disertakan dalam pembaruan. Keputusan mengenai elemen mana dalam status sel yang harus diperbarui diimplementasikan menggunakan mekanisme filter serupake gerbang lupa: masukan yang digabungkan x_t ditambah status tersembunyi h_{t-1} dilewatkan melalui lapisan unit sigmoid untuk menghasilkan vektor elemen, dengan lebar yang sama dengan sel, di mana setiap elemen dalam vektor berada dalam kisaran 0 hingga 1; nilai yang mendekati 0 menunjukkan bahwa elemen sel yang sesuai

tidak akan diperbarui, dan nilai yang mendekati 1 menunjukkan bahwa elemen sel yang sesuai akan diperbarui. Pada saat yang sama ketika vektor filter dibuat, input gabungan dan status tersembunyi juga dilewatkan melalui lapisan unit tanh (yaitu, neuron yang menggunakan fungsi aktivasi tanh). Sekali lagi, ada satu unit tanh untuk setiap aktivasi dalam sel LSTM. Vektor ini mewakili informasiyang dapat ditambahkan ke status sel. Unit tanh digunakan untuk menghasilkan vektor pembaruan ini karena nilai keluaran unit tanh dalam rentang -1 hingga +1, dan akibatnya nilai aktivasi dalam elemen sel dapat ditingkatkan dan diturunkan dengan pembaruan.[3] Setelah dua vektor ini dibuat, vektor pembaruan terakhir dihitung dengan mengalikan keluaran vektor dari lapisan tanh dengan vektor filter yang dihasilkan dari lapisan sigmoid. Vektor yang dihasilkan kemudian ditambahkan ke sel menggunakan penambahan vektor.



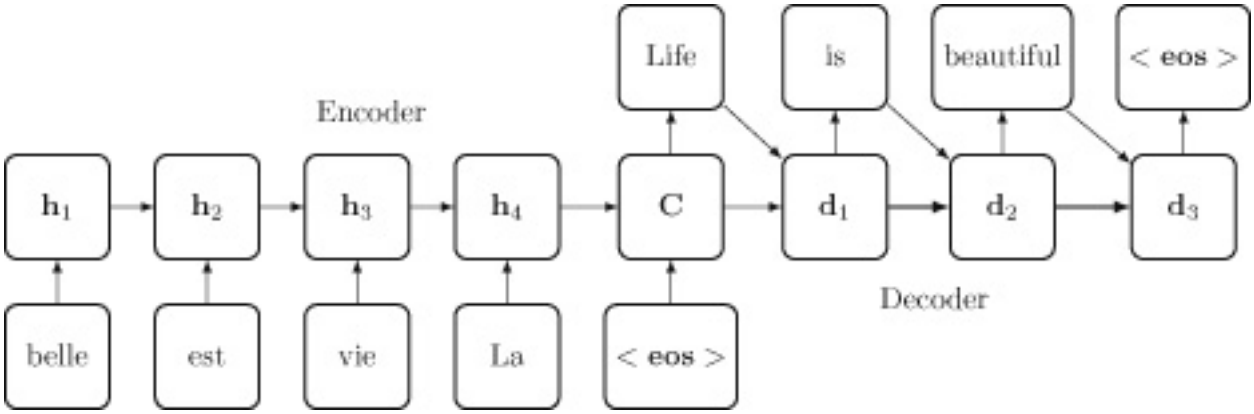
Gambar 5.4 Skema struktur internal unit LSTM: σ mewakili lapisan neuron dengan aktivasi sigmoid, $\mathbb{T}$ mewakili lapisan neuron dengan aktivasi tanh, × mewakili perkalian vektor, dan + mewakili penambahan vektor. Sosok tersebut terinspirasi oleh gambar oleh Christopher Olah yang tersedia di: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Tahap terakhir dari pemrosesan dalam LSTM adalah memutuskan elemen sel mana yang harus dikeluarkan sebagai respons terhadap masukan saat ini. Pemrosesan ini dilakukan oleh komponen-komponen di blok bertanda Output (di sebelah kanan gambar 5.4). Vektor keluaran kandidat dihasilkan dengan melewatkan sel melalui lapisan tanh. Pada saat yang sama, input gabungan dan vektor status tersembunyi yang disebarkan melewati lapisan unit sigmoid untuk membuat vektor filter lain. Vektor keluaran aktual kemudian dihitung dengan mengalikan vektor keluaran kandidat dengan vektor filter ini. Vektor yang dihasilkan kemudian diteruskan ke lapisan keluaran, dan juga disebarkan ke langkah waktu berikutnya sebagai status tersembunyi baru $\mathbf{h}_t$.

Fakta bahwa unit LSTM berisi banyak lapisan neuron berarti bahwa LSTM adalah jaringan itu sendiri. Namun, RNN dapat dibuat dengan memperlakukan LSTM sebagai lapisan tersembunyi di RNN. Dalam konfigurasi ini, fileUnit LSTM menerima masukan pada setiap langkah waktu dan menghasilkan keluaran untuk setiap masukan. RNN yang menggunakan unit LSTM sering disebut sebagai jaringan LSTM.

Jaringan LSTM cocok untuk pemrosesan bahasa alami (NLP). Tantangan utama dalam menggunakan jaringan saraf untuk melakukan pemrosesan bahasa alami adalah kata-kata dalam bahasa harus diubah menjadi vektor angka. Model word2vec, yang dibuat oleh Tomas Mikolov dan rekannya di penelitian Google, adalah salah satu cara paling populer untuk melakukan konversi ini (Mikolov et al. 2013). Model word2vec didasarkan pada gagasan bahwa kata-kata yang muncul dalam konteks yang mirip memiliki arti yang serupa. Definisi konteks di sini melingkupi kata-kata. Jadi misalnya, kata-kata *London* dan *Paris* semantik yang sama karena masing-masing dari mereka sering co-terjadi dengan kata-kata bahwa kata lain juga co-terjadi dengan, misalnya: *ibu* , *kota* , *Eropa*, *liburan* , *bandara* , dan sebagainya. Model word2vec adalah jaringan saraf yang mengimplementasikan gagasan kemiripan semantik ini dengan awalnya menetapkan vektor acak ke setiap kata dan kemudian menggunakan kejadian bersama dalam korpus untuk memperbarui vektor ini secara berulang sehingga kata-kata yang mirip secara semantik berakhir dengan vektor yang serupa. Vektor ini (dikenal sebagai embeddings kata) kemudian digunakan untuk merepresentasikan kata saat dimasukkan ke jaringan neural.

Salah satu area NLP tempat pembelajaran mendalam memiliki pengaruh besar adalah terjemahan mesin. Gambar 5.5 menyajikan skema tingkat tinggi dari seq2seq (atauencoder-decoder) untuk terjemahan mesin saraf (Sutskever et al. 2014). Arsitektur ini terdiri dari dua jaringan LSTM yang telah digabungkan. Jaringan LSTM pertama memproses kalimat masukan dengan cara kata demi kata. Dalam contoh ini, bahasa sumbernya adalah Prancis. Kata-kata tersebut dimasukkan ke dalam sistem dalam urutan terbalik karena ditemukan bahwa hal ini mengarah pada terjemahan yang lebih baik. Simbol ^{eos} adalah simbol akhir kalimat khusus. Saat setiap kata dimasukkan, pembuat enkode memperbarui status tersembunyi dan menyebarkannya ke langkah waktu berikutnya. Status tersembunyi yang dihasilkan oleh pembuat enkode sebagai tanggapan terhadap ^{eos} simbol diambil sebagai representasi vektor dari kalimat masukan. Vektor ini dilewatkan sebagai masukan awal ke decoder LSTM. Dekoder dilatih untuk mengeluarkan kalimat terjemahan kata demi kata, dan setelah setiap kata dihasilkan, kata ini dimasukkan kembali ke sistem sebagai masukan untuk langkah waktu berikutnya. Di sebuahcara, decoder berhalusinasi terjemahan karena menggunakan outputnya sendiri untuk mendorong proses pembuatannya sendiri. Proses ini berlanjut hingga decoder mengeluarkan ^{eos} simbol.



Gambar 5.5 Skema arsitektur seq2seq (atau encoder-decoder).

Ide menggunakan vektor angka untuk mewakili makna (interlingual) dari sebuah kalimat sangat kuat, dan konsep ini telah diperluas ke ide penggunaan vektor untuk merepresentasikan representasi intermodal / multimodal. Misalnya, perkembangan menarik dalam beberapa tahun terakhir adalah pengembangan sistem teks gambar otomatis. Sistem ini dapat mengambil gambar sebagai input dan menghasilkan deskripsi bahasa alami dari gambar tersebut. Struktur dasar sistem ini sangat mirip dengan arsitektur terjemahan mesin saraf yang ditunjukkan pada gambar 5.5. Perbedaan utamanya adalah jaringan encoder LSTM digantikan oleh arsitektur CNN yang memproses gambar input dan menghasilkan representasi vektor yang kemudian disebarkan ke decoder LSTM (Xu et al. 2015). Ini adalah contoh lain dari kekuatan pembelajaran mendalam yang muncul dari kemampuannya untuk mempelajari representasi informasi yang kompleks. Dalam hal ini, sistem mempelajari representasi antar moda yang memungkinkan informasi mengalir dari apa yang ada dalam gambar ke bahasa. Menggabungkan arsitektur CNN dan RNN menjadi semakin populer karena menawarkan potensi untuk mengintegrasikan keunggulan kedua sistem dan memungkinkan arsitektur pembelajaran mendalam untuk menangani data yang sangat kompleks.

Terlepas dari arsitektur jaringan yang kita gunakan, kita perlu menemukan bobot yang benar untuk jaringan jika kita ingin membuat model yang akurat. Bobot neuron menentukan transformasi yang diterapkan neuron ke inputnya. Jadi, bobot jaringanlah yang menentukan blok bangunan fundamental dari representasi yang dipelajari jaringan. Saat ini, metode standar untuk menemukan bobot ini adalah algoritme yang mulai terkenal pada 1980-an: propagasi mundur. Bab selanjutnya akan menyajikan pengantar komprehensif untuk algoritma ini.

6

Fungsi Pembelajaran

Model jaringan saraf, tidak peduli seberapa dalam atau kompleksnya, mengimplementasikan fungsi, pemetaan dari masukan ke keluaran. Fungsi yang diimplementasikan oleh jaringan ditentukan oleh bobot yang digunakan jaringan.

Jadi, melatih jaringan (mempelajari fungsi yang harus diterapkan oleh jaringan) pada data melibatkan pencarian kumpulan bobot yang paling memungkinkan jaringan untuk memodelkan pola dalam data. Algoritme yang paling umum digunakan untuk mempelajari pola dari data adalah algoritme penurunan gradien. Algoritme penurunan gradien sangat mirip dengan aturan pembelajaran perceptron dan algoritme LMS yang dijelaskan di bab 4: algoritme ini mendefinisikan aturan untuk memperbarui bobot yang digunakan dalam fungsi berdasarkan kesalahan fungsi. Dengan sendirinya, algoritma penurunan gradien dapat digunakan untuk melatih neuron keluaran tunggal. Namun, itu tidak dapat digunakan untuk melatih jaringan dalam dengan beberapa lapisan tersembunyi. Batasan ini karenaMasalah penugasan kredit: bagaimana seharusnya kesalahan yang disalahkan atas keseluruhan kesalahan jaringan dibagi di antara neuron yang berbeda (termasuk neuron tersembunyi) dalam jaringan? Akibatnya, melatih jaringan neural dalam melibatkan penggunaan algoritme penurunan gradien dan algoritme propagasi mundur secara bersamaan.

Proses yang digunakan untuk melatih jaringan neural dalam dapat dicirikan sebagai: menginisialisasi bobot jaringan secara acak, dan kemudian memperbarui bobot jaringan secara berulang, sebagai tanggapan atas kesalahan yang dibuat jaringan pada set data, hingga jaringan berfungsi sebagai diharapkan. Dalam kerangka kerja pelatihan ini, algoritme propagasi mundur menyelesaikan masalah penugasan kredit (atau kesalahan), dan algoritme penurunan gradien menentukan aturan pembelajaran yang benar-benar memperbarui bobot dalam jaringan.

Bab ini adalah bab paling matematika dalam buku ini. Namun, pada tingkat tinggi, yang perlu Anda ketahui tentang algoritme propagasi mundur dan algoritme penurunan gradien adalah bahwa algoritme tersebut dapat digunakan untuk melatih jaringan dalam. Jadi, jika Anda tidak punya waktu untuk membahas detail dalam bab ini, silakan membacanya. Namun, jika Anda ingin mendapatkan pemahaman yang lebih dalam tentang kedua algoritme ini, saya mendorong Anda untuk terlibat dengan materi. Algoritme ini merupakan inti dari pembelajaran mendalam dan memahami cara kerjanya, mungkin merupakan cara paling langsung untuk memahami potensi dan keterbatasannya. Saya telah mencoba menyajikan materi dalam bab ini dalam sebuahcara yang mudah diakses, jadi jika Anda mencari pengenalan yang relatif lembut namun masih komprehensif untuk algoritme ini, saya yakin ini akan menyediakannya untuk Anda. Bab ini dimulai dengan menjelaskan algoritme penurunan gradien, lalu menjelaskan bagaimana penurunan gradien dapat digunakan bersama dengan algoritme propagasi mundur untuk melatih jaringan neural.

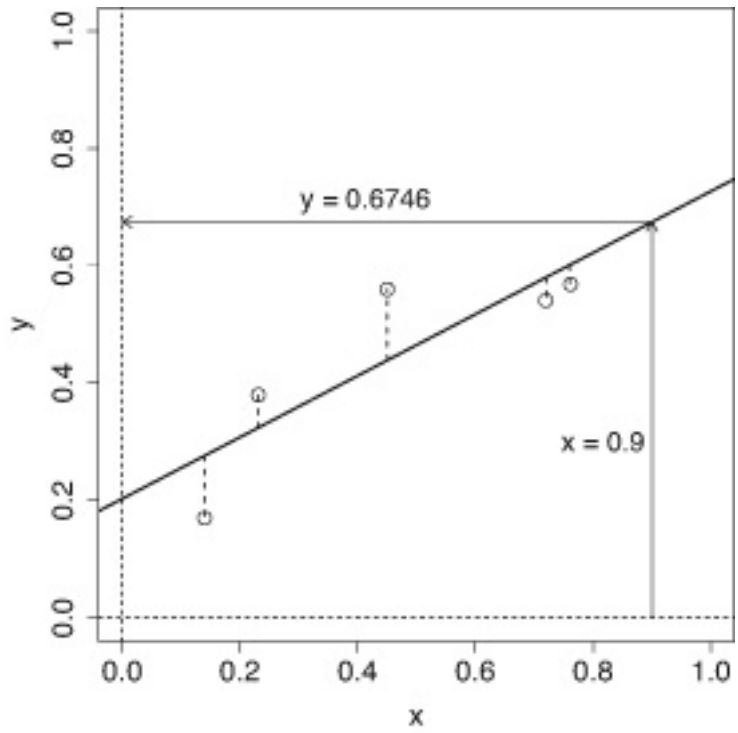
Penurunan Gradien

Jenis fungsi yang sangat sederhana adalah pemetaan linier dari satu masukan ke satu keluaran. Tabel 6.1 menyajikan kumpulan data dengan fitur masukan tunggal dan keluaran tunggal. Gambar 6.1 menyajikan diagram sebar dari data ini bersama

dengan plot garis yang paling sesuai dengan data ini. Garis ini dapat digunakan sebagai fungsi untuk memetakan dari nilai input ke prediksinilai keluaran. Misalnya, jika x = 0,9, maka respon yang dikembalikan oleh fungsi linier ini adalah y = 0,6746. Kesalahan (atau kerugian) menggunakan garis ini sebagai model untuk data ditunjukkan oleh garis putus-putus dari garis ke setiap datum.

Tabel 6.1. Contoh dataset dengan satu fitur masukan, x, dan fitur keluaran (target), y

X	Y
0.72	0,54
0.45	0,56
0.23	0.38
0.76	0,57
0.14	0.17



Gambar 6.1 Sebar data dengan garis "paling sesuai" dan kesalahan garis pada setiap contoh diplot sebagai segmen garis putus-putus vertikal. Gambar tersebut juga menunjukkan pemetaan yang ditentukan oleh garis untuk input x = 0.9 ke output y = 0.6746.

Dalam bab 2, kami menjelaskan bagaimana fungsi linier dapat direpresentasikan menggunakan persamaan garis:

$$y = mx + c$$

Dimana m adalah kemiringan garis, dan c merupakan titik potong dengan sumbu y, yang menentukan di mana garis memotong sumbu y. Untukbaris pada gambar 6.1, $c = 0.203$ dan $m = 0.524$; Inilah sebabnya mengapa fungsi mengembalikan nilai $y = 0.6746$ ketika $x = 0.9$, seperti berikut ini:

$$0.6746 = (0.524 \times 0.9) + 0.203$$

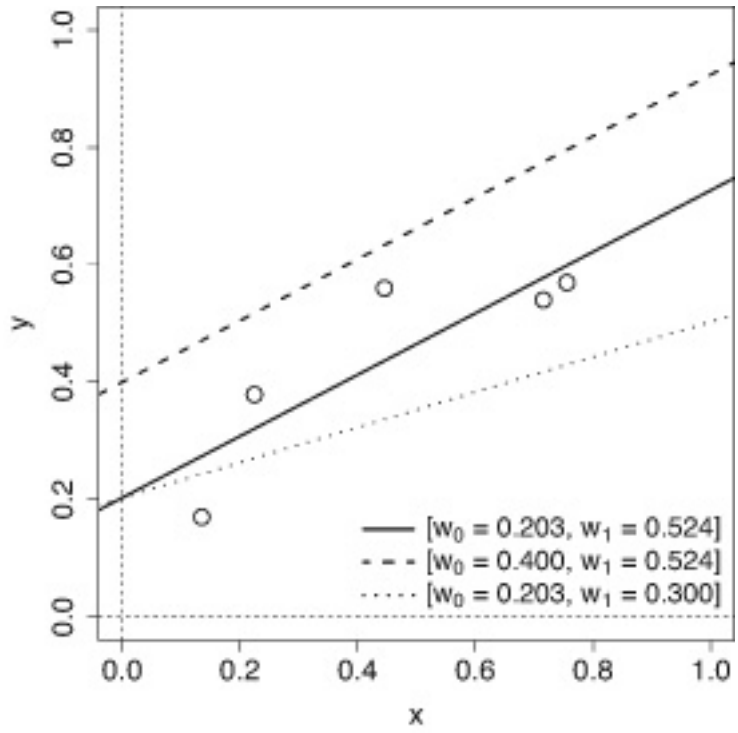
Kemiringan m dan perpotongan y c adalah parameter model ini, dan parameter ini dapat divariasikan agar sesuai dengan model dengan data.

Persamaan garis memiliki hubungan dekat dengan operasi penjumlahan terbobot yang digunakan dalam neuron. Ini menjadi jelas jika kita menulis ulang persamaan garis dengan parameter model ditulis ulang sebagai bobot ($c \rightarrow w_0, m \rightarrow w_1$):

$$y = (w_0 \times 1) + (w_1 \times x)$$

Garis yang berbeda (model linier yang berbeda untuk data) dapat dibuat dengan memvariasikan salah satu dari bobot ini (atau parameter model). Gambar 6.2 mengilustrasikan bagaimana suatu garis berubah karena perpotongan dan kemiringan garis bervariasi: garis putus-putus mengilustrasikan apa yang terjadi jika perpotongan y ditingkatkan, dan garis putus-putus menunjukkan apa yang terjadi jika kemiringan diturunkan. Mengubah perpotongan y w_0 secara vertikal menerjemahkan garis, sedangkan memodifikasi kemiringan w_1 memutar garis di sekitar titik $(x = 0, y = \text{intercept})$.

Masing-masing baris baru ini mendefinisikan fungsi yang berbeda, memetakan dari x ke y, dan setiap fungsi akan memilikinyakesalahan yang berbeda sehubungan dengan seberapa cocok data itu. Melihat gambar 6.2, kita dapat melihat bahwa garis penuh $[w_0 = 0.203, w_1 = 0.524]$,, lebih cocok dengan data daripada dua garis lainnya karena rata-rata melewati lebih dekat ke titik data. Dengan kata lain, rata-rata kesalahan baris ini untuk setiap titik data lebih kecil daripada dua baris lainnya. Total error model pada dataset dapat diukur dengan menjumlahkan error yang dibuat model pada setiap contoh dalam dataset. Cara standar untuk menghitung kesalahan total ini adalah dengan menggunakan persamaan yang dikenal sebagai jumlah kesalahan kuadrat (SSE):



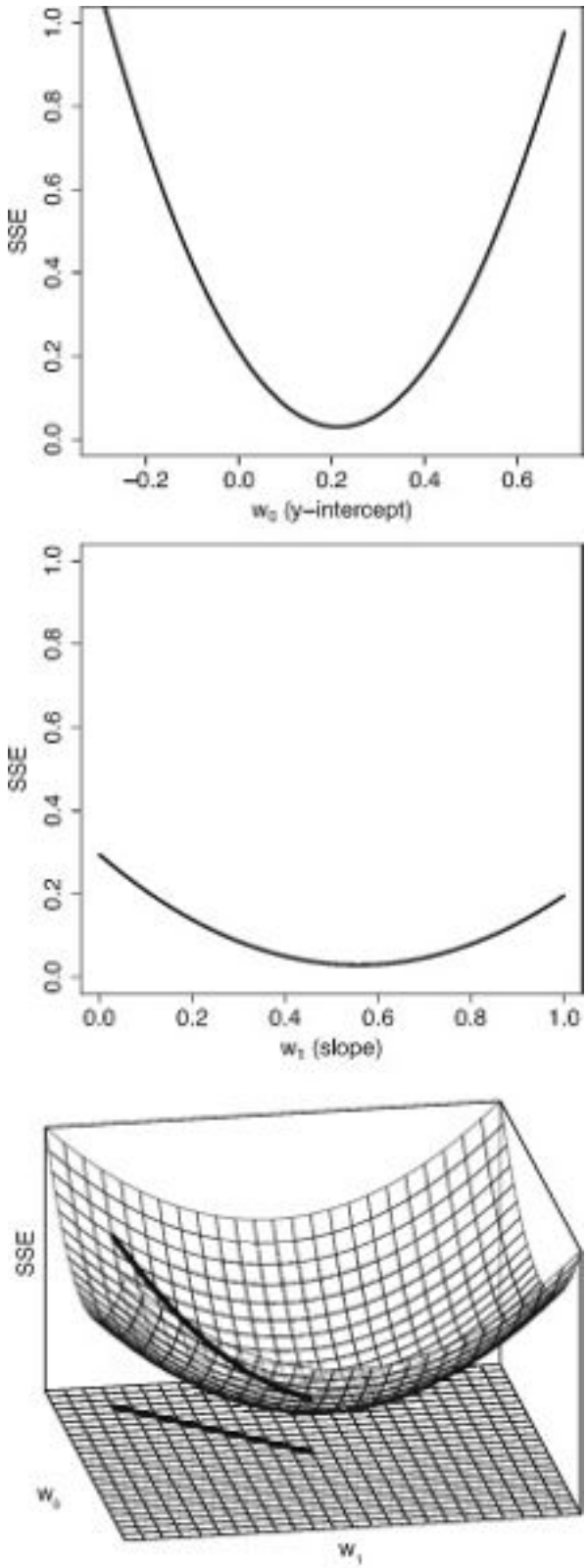
Gambar 6.2 Plot yang mengilustrasikan bagaimana suatu garis berubah karena divariasikan intersep (w_0) dan kemiringan (w_1).

$$SSE = \frac{1}{2}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2$$

Persamaan ini memberi tahu kita cara menambahkan kesalahan model pada dataset yang berisi *n* contoh. Persamaan ini menghitung untuk setiap n contoh dalam kumpulan data kesalahan model dengan mengurangi prediksi nilai target yang dikembalikan oleh model dari nilai target yang benar untuk contoh tersebut, seperti yang ditentukan dalam kumpulan data. Dalam persamaan ini y_j adalah nilai keluaran yang benar untuk fitur target yang tercantum dalam dataset misalnya *j* , dan $\hat{y}_j$ adalah perkiraan nilai target yang dikembalikan oleh model untuk contoh yang sama. Masing-masing kesalahan ini kemudian dikuadratkan dan kesalahan kuadrat ini kemudian dijumlahkan. Menguadratkan kesalahan memastikan bahwa semuanya positif, dan oleh karena itu dalam penjumlahan kesalahan untuk contoh di mana fungsi meremehkan target tidak membatalkan kesalahan pada contoh di mana itu melebih-lebihkan target. Perkalian dari penjumlahan kesalahan dengan $1/2$, meskipun tidak penting untuk pembahasan saat ini, akan berguna nanti. Semakin rendah SSE suatu fungsi, semakin baik fungsi tersebut memodelkan datanya. Akibatnya, jumlah kesalahan kuadrat dapat digunakan sebagai fungsi kesesuaian untuk mengevaluasi seberapa baik fungsi kandidat (dalam situasi ini model yang membuat sebuah garis) cocok dengan data.

Gambar 6.3 menunjukkan bagaimana kesalahan model linier bervariasi karena parameter model berubah. Plot ini menunjukkan SSE model linier pada contoh dataset single-input-single-output yang tercantum dalam tabel 6.1. Untuk setiap

parameter ada satu pengaturan terbaik dan ketika parameter menjauh dari pengaturan ini (di kedua arah) kesalahan model meningkat. Konsekuensi dari ini adalah profil kesalahan model karena setiap parameter bervariasi adalah cembung (berbentuk mangkuk). Bentuk cembung ini terlihat jelas pada plot atas dan tengah pada gambar 6.3, yang menunjukkan bahwa SSE model diminimalkan when $w_0 = 0.203$ (titik terendah kurva di plot atas), dan when $w_1 = 0.524$ (titik terendah kurva di tengah merencanakan).



Gambar 6.3 Plot perubahan error (SSE) model linier sebagai parameter perubahan model. Atas: profil SSE model linier dengan kemiringan tetap $w_1 = 0.524$ saat w_0 rentang melintasi interval 0,3 hingga 1. Tengah: profil SSE model linier dengan titik potong y ditetapkan pada $w_0 = 0.203$ saat w_1 rentang melintasi interval 0 hingga 1. Bawah: permukaan kesalahan model linier saat keduanya w_0 dan w_1 bervariasi.

Jika kita memplot kesalahan model karena kedua parameter bervariasi, kita menghasilkan permukaan berbentuk mangkuk cembung tiga dimensi, yang dikenal sebagai permukaan kesalahan. Mata jaring berbentuk mangkuk pada plot di bagian bawah gambar 6.3 mengilustrasikan permukaan kesalahan ini. Permukaan kesalahan ini dibuat dengan terlebih dahulu menentukan ruang berat. Bobot ruang

ini diwakili oleh grid datar di bagian bawah plot. Setiap koordinat dalam ruang bobot ini menentukan garis yang berbeda karena setiap koordinat menentukan titik potong (w_0 nilai) dan kemiringan (w_1 nilai). Akibatnya, bergerak melintasi ruang berat planar ini setara dengan bergerak di antara model yang berbeda. Langkah kedua dalam membangun permukaan kesalahan adalah mengaitkan ketinggian dengan setiap garis (yaitu,koordinat) di ruang bobot. Elevasi yang terkait dengan setiap koordinat ruang berat adalah SSE model yang ditentukan oleh koordinat tersebut; atau, secara lebih langsung, ketinggian permukaan kesalahan di atas bidang ruang berat adalah SSE dari model linier yang sesuai ketika digunakan sebagai model untuk kumpulan data. Koordinat ruang bobot yang sesuai dengan titik terendah dari permukaan kesalahan menentukan model linier yang memiliki SSE terendah pada dataset (yaitu, model linier yang paling sesuai dengan data).

Bentuk permukaan error pada plot di sebelah kanan gambar 6.3 menunjukkan bahwa hanya ada satu model linier terbaik untuk dataset ini karena terdapat satu titik di bagian bawah bowl yang memiliki elevasi lebih rendah (error lebih rendah) daripada titik lain di permukaan. Berpindah dari model terbaik ini (dengan memvariasikan bobot model) perlu melibatkan pemindahan ke model dengan SSE yang lebih tinggi. Perpindahan seperti itu setara dengan pindah ke koordinat baru di ruang berat, yang memiliki ketinggian lebih tinggi yang diasosiasikan dengannya di permukaan kesalahan. Permukaan kesalahan cembung atau berbentuk mangkuk sangat berguna untuk mempelajari fungsi linier untuk memodelkan kumpulan data karena ini berarti bahwa proses pembelajaran dapat dibingkai sebagai pencarian titik terendah pada permukaan kesalahan. Algoritme standar yang digunakan untuk menemukan titik terendah ini dikenal sebagai penurunan gradien.

Permukaan kesalahan cembung atau berbentuk mangkuk sangat berguna untuk mempelajari fungsi linier untuk memodelkan kumpulan data karena itu berarti proses pembelajaran dapat dibingkai sebagai pencarian titik terendah pada permukaan kesalahan.

Algoritme penurunan gradien dimulai dengan membuat model awal menggunakan sekumpulan bobot yang dipilih secara acak. Selanjutnya SSE dari model yang diinisialisasi secara acak inidihitung. Secara keseluruhan, kumpulan bobot yang ditebak dan SSE dari model yang sesuai menentukan titik awal awal pada permukaan error untuk pencarian. Sangat mungkin bahwa model yang diinisialisasi secara acak akan menjadi model yang buruk, sehingga sangat mungkin pencarian akan dimulai di lokasi yang memiliki elevasi tinggi pada permukaan kesalahan. Namun, awal yang buruk ini tidak menjadi masalah, karena setelah proses pencarian ditempatkan di permukaan kesalahan, proses dapat menemukan kumpulan bobot yang lebih baik hanya dengan mengikuti gradien permukaan kesalahan ke bawah hingga mencapai bagian bawah kesalahan.

permukaan (lokasi di mana bergerak ke segala arah menghasilkan peningkatan SSE). Inilah mengapa algoritme ini dikenal sebagai penurunan gradien:

Poin penting adalah bahwa pencarian tidak berkembang dari lokasi awal ke dasar lembah dalam satu kali pembaruan bobot. Sebaliknya, itu bergerak ke arah bawah permukaan kesalahan secara berulang, dan selama setiap iterasi, kumpulan bobot saat ini diperbarui untuk pindah ke lokasi terdekat di ruang bobot yang memiliki SSE lebih rendah. Mencapai bagian bawah permukaan kesalahan dapat membutuhkan banyak iterasi. Cara intuitif untuk memahami prosesnya adalah dengan membayangkan seorang pejalan kaki yang terjebak di sisi bukit ketika kabut tebal turun. Mobil mereka diparkir di dasar lembah; namun, karena kabut, mereka hanya dapat melihat beberapa meter ke segala arah. Asumsibahwa lembah memiliki bentuk cembung yang bagus, mereka masih dapat menemukan jalan ke mobil mereka, meskipun berkabut, dengan berulang kali mengambil langkah kecil yang menuruni bukit mengikuti kemiringan lokal di posisi mereka saat ini. Satu proses pencarian penurunan gradien diilustrasikan di plot bawah gambar 6.3. Kurva hitam yang diplot pada permukaan kesalahan menggambarkan jalur yang diikuti pencarian ke bawah permukaan, dan garis hitam pada ruang berat memplot pembaruan berat yang sesuai yang terjadi selama perjalanan menuruni permukaan kesalahan. Secara teknis, algoritma gradient descent dikenal sebagai algoritma optimasi karena tujuan dari algoritma tersebut adalah untuk mencari himpunan bobot yang optimal.

Komponen terpenting dari algoritme penurunan gradien adalah aturan yang menentukan bagaimana bobot diperbarui selama setiap iterasi algoritme. Untuk memahami bagaimana aturan ini didefinisikan, pertama-tama perlu dipahami bahwa permukaan kesalahan terdiri dari beberapa gradien kesalahan. Untuk contoh sederhana kami, permukaan kesalahan dibuat dengan menggabungkan dua kurva kesalahan. Satu kurva kesalahan didefinisikan oleh perubahan dalam SSE sebagai w_0 perubahan, yang ditunjukkan pada plot teratas gambar 6.3. Kurva kesalahan lainnya didefinisikan oleh perubahan dalam SSE sebagai w_1 perubahan, yang ditunjukkan pada plot di tengah gambar 6.3. Perhatikan bahwa gradien dari setiap kurva ini dapat bervariasi di sepanjang kurva, misalnya w_0 kurva kesalahan memiliki gradien yang curam di kiri dan kanan plot, tetapi gradien menjadiagak lebih dangkal di tengah kurva. Juga, gradien dari dua kurva yang berbeda dapat sangat bervariasi; dalam contoh khusus ini w_0 kurva kesalahan umumnya memiliki gradien yang lebih curam daripada w_1 kurva kesalahan.

Fakta bahwa permukaan kesalahan terdiri dari beberapa kurva, masing-masing dengan gradien yang berbeda, adalah penting karena algoritme penurunan gradien bergerak ke bawah permukaan kesalahan gabungan dengan secara independen memperbarui setiap bobot sehingga dapat bergerak ke bawah kurva kesalahan yang terkait dengan bobot tersebut. Dengan kata lain, selama iterasi tunggal dari algoritme penurunan gradien, w_0 diperbarui untuk bergerak ke bawah w_0 kurva

kesalahan dan w_1 diperbarui ke bawah w_1 kurva kesalahan. Selain itu, jumlah setiap bobot yang diperbarui dalam sebuah iterasi sebanding dengan kecuraman gradien kurva kesalahan bobot, dan gradien ini akan bervariasi dari satu iterasi ke berikutnya saat proses bergerak ke bawah kurva kesalahan. Sebagai contoh, w_0 akan diperbarui dengan jumlah yang relatif besar dalam iterasi di mana proses pencarian ditempatkan tinggi di kedua sisi w_0 kurva kesalahan, tetapi dengan jumlah yang lebih kecil dalam iterasi di mana proses pencarian lebih dekat ke bagian bawah w_0 kurva kesalahan.

Kurva kesalahan yang terkait dengan setiap bobot ditentukan oleh bagaimana SSE berubah sehubungan dengan perubahan nilai bobot. Kalkulus, dan khususnya diferensiasi, adalah bidang matematika yang berhubungan dengan laju perubahan. Misalnya, mengambil turunan dari suatu fungsi, $y = f(x)$, menghitung laju perubahan y (keluaran) untuk setiap perubahan unit dalam x (masukan). Lebih lanjut, jika suatu fungsi mengambil beberapa masukan [$y = f(x_1,\ldots,x_n)$] maka dimungkinkan untuk menghitung laju perubahan keluaran, y sehubungan dengan perubahan pada masing-masing masukan ini, x_i, dengan mengambil turunan parsial dari fungsi sehubungan dengan setiap masukan. Turunan parsial dari suatu fungsi yang terkait dengan input tertentu dihitung dengan terlebih dahulu mengasumsikan bahwa semua input lainnya tetap konstan (sehingga laju perubahannya adalah 0 dan menghilang dari perhitungan) dan kemudian mengambil turunan dari yang tersisa. Terakhir, laju perubahan fungsi untuk input tertentu juga dikenal sebagai gradien fungsi di lokasi pada kurva (ditentukan oleh fungsi) yang ditentukan oleh input. Akibatnya, turunan parsial SSE sehubungan dengan bobot menentukan bagaimana keluaran SSE berubah saat bobot tersebut berubah, dan karenanya menentukan gradien kurva kesalahan bobot tersebut. Inilah yang diperlukan untuk menentukan aturan pembaruan bobot penurunan gradien:

Turunan parsial dari suatu fungsi yang terkait dengan variabel tertentu adalah turunan fungsi whensemua variabel lainnya tetap konstan. Akibatnya, terdapat turunan parsial yang berbeda dari suatu fungsi terhadap setiap variabel, karena sekumpulan suku yang berbeda dianggap konstan dalam perhitungan masing-masing turunan parsial. Oleh karena itu, terdapat turunan parsial SSE yang berbeda untuk setiap bobot, meskipun semuanya memiliki bentuk yang serupa. Inilah sebabnya mengapa setiap bobot diperbarui secara independen dalam algoritme penurunan gradien: aturan pembaruan bobot bergantung pada turunan parsial SSE untuk setiap bobot, dan karena ada turunan parsial yang berbeda untuk setiap bobot, ada bobot terpisah perbarui aturan untuk setiap bobot. Sekali lagi, meskipun turunan parsial untuk setiap bobot berbeda, semua turunan ini memiliki bentuk yang sama, sehingga aturan pembaruan bobot untuk setiap bobot juga akan memiliki bentuk yang sama. Ini menyederhanakan definisi algoritma penurunan gradien. Faktor penyederhanaan lainnya adalah SSE didefinisikan relatif terhadap

kumpulan data dengan n contoh. Relevansi ini adalah bahwa satu-satunya variabel dalam SSE adalah bobot; output target y dan input x semuanya ditentukan oleh kumpulan data untuk setiap contoh, sehingga dapat dianggap konstanta. Akibatnya, saat menghitung turunan parsial SSE yang terkait dengan bobot, banyak istilah dalam persamaan yang tidak menyertakan bobot dapat dihapus karena dianggap konstanta.

Hubungan antara keluaran SSE dan bobot masing-masing menjadi lebih eksplisit jika definisi SSE ditulis ulang sehingga istilah tersebut $\hat{y}_j$, yang menunjukkan keluaran yang diprediksi oleh model, diganti dengan struktur model yang menghasilkan prediksi. Untuk model dengan satu masukan x_1 dan satu masukan palsu, $x_0 = 1$, versi SSE yang ditulis ulang ini adalah:

$$SSE = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \left(w_0 \times x_{j,0} + w_1 \times x_{j,1}\right)\right)^2$$

Persamaan ini menggunakan subskrip ganda pada input, subskrip pertama j mengidentifikasi contoh (atau baris dalam dataset) dan subskrip kedua menentukan fitur (atau kolom dalam dataset) dari input. Misalnya, $x_{j,1}$ mewakili fitur 1 dari contoh j. Definisi SSE ini dapat digeneralisasikan ke model dengan m input:

$$SSE = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \left(\sum_{i=0}^{m} w_i \times x_{j,i}\right)\right)^2$$

Menghitung turunan parsial SSE sehubungan dengan bobot tertentu melibatkan penerapan aturan rantai dari kalkulus dan sejumlah aturan diferensiasi standar. Hasil dari penurunan ini adalah persamaan berikut (untuk memudahkan penyajian kita beralih kembali ke notasi $\hat{y}_i$ untuk merepresentasikan keluaran dari model):

$$\frac{\partial\, SSE}{\partial\, w_i} = \sum_{j=1}^{n}\left(\underbrace{(y_j - \hat{y}_j)}_{\substack{\text{error of the}\\ \text{output of the}\\ \text{weighted sum}}} \times \underbrace{-x_{j,i}}_{\substack{\text{rate of change of}\\ \text{weighted sum}\\ \text{with respect to}\\ \text{change in } w_i}}\right)$$

Turunan parsial ini menentukan cara menghitung gradien kesalahan untuk bobot w_i set data di mana $x_{j,i}$ masukan terkait w_i untuk setiap contoh dalam kumpulan data. Perhitungan ini melibatkan perkalian dua istilah, kesalahan output dan laju perubahan output (yaitu, jumlah tertimbang) sehubungan dengan perubahan bobot. Salah satu cara untuk memahami perhitungan ini adalah jika perubahan bobot mengubah keluaran jumlah bobot dengan jumlah yang besar, maka gradien kesalahan sehubungan dengan bobot adalah besar (curam) karena perubahan bobot akan mengakibatkan perubahan besar pada kesalahannya. Namun, gradien ini adalah gradien menanjak, dan kami ingin memindahkan bobot untuk turun ke kurva kesalahan. Jadi dalam aturan pembaruan bobot penurunan gradien (ditampilkan di bawah) tanda "-" di depan masukan $x_{j,i}$ dihilangkan. Menggunakan t untuk mewakili iterasi algoritme (iterasi melibatkan satu lintasan melalui n contoh dalam kumpulan data), aturan pembaruan bobot penurunan gradien didefinisikan sebagai:

$$w_i^{t+1} = w_i^t + \left(\eta \times \underbrace{\sum_{j=1}^{n} \left(\left(y_j^t - \hat{y}_j^t \right) \times x_{j,i}^t \right)}_{\text{error gradient for } w_i} \right)$$

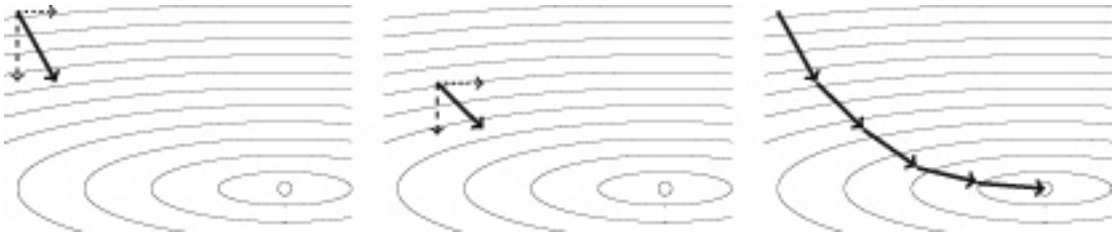
Ada sejumlah faktor penting tentang aturan pembaruan berat badan ini. Pertama, aturan tersebut menentukan bagaimana bobot w_i harus diperbarui setelah iterasi t melalui dataset. Pembaruan ini sebanding dengan gradien kurva kesalahan untuk bobot untuk iterasi itu (yaitu, istilah penjumlahan, yang sebenarnya mendefinisikan turunan parsial SSE untuk bobot tersebut). Kedua, aturan pembaruan bobot dapat digunakan untuk memperbarui bobot untuk fungsi dengan banyak input. Ini berarti bahwa algoritma penurunan gradien dapat digunakan untuk menurunkan permukaan kesalahan dengan lebih dari dua koordinat bobot. Tidak mungkin untuk memvisualisasikan permukaan kesalahan ini karena mereka akan memiliki lebih dari tiga dimensi, tetapi prinsip dasar penurunan permukaan kesalahan menggunakan gradien kesalahan menggeneralisasi fungsi pembelajaran dengan banyak masukan. Ketiga, meskipun aturan pembaruan bobot memiliki struktur yang serupa untuk setiap bobot, aturan tersebut menentukan pembaruan yang berbeda untuk setiap bobot selama setiap iterasi karena pembaruan bergantung pada input dalam contoh set data tempat bobot diterapkan. Keempat, penjumlahan dalam aturanmenunjukkan bahwa, dalam setiap iterasi algoritme penurunan gradien, model saat ini harus diterapkan ke semua n dari contoh dalam kumpulan data. Inilah salah satu alasan mengapa melatih jaringan pembelajaran yang dalam adalah tugas yang mahal secara komputasi. Biasanya untuk kumpulan data yang sangat besar, kumpulan data dibagi menjadi beberapa kumpulan contoh yang diambil sampelnya dari kumpulan data, dan setiap iterasi pelatihan

didasarkan pada kumpulan, bukan seluruh kumpulan data. Kelima, selain modifikasi yang diperlukan untuk memasukkan penjumlahan, aturan ini identik dengan aturan pembelajaran LMS (juga dikenal sebagai Widrow-Hoff atau delta) yang diperkenalkan di bab 4, dan aturan tersebut mengimplementasikan logika yang sama: jika output dari model terlalu besar, maka bobot yang terkait dengan masukan positif harus dikurangi; jika output terlalu kecil, maka bobot ini harus ditingkatkan. Apalagi tujuan dan fungsi hyperparameter kecepatan pembelajaran (η) sama dengan aturan LMS: menskalakan penyesuaian bobot untuk memastikan bahwa penyesuaian tidak terlalu besar sehingga algoritme melewatkan (atau melangkahi) kumpulan bobot terbaik. Dengan menggunakan aturan pembaruan bobot ini, algoritme penurunan gradien dapat diringkas sebagai berikut:

1. Buat model menggunakan satu set bobot awal.
2. Ulangi sampai kinerja model cukup baik.
    Sebuah. Terapkan model saat ini ke contoh dalam kumpulan data.
    b. Sesuaikan setiap berat menggunakan aturan pembaruan berat.
3. Kembalikan model terakhir.

Salah satu konsekuensi dari pembaruan independen bobot, dan fakta bahwa pembaruan bobot proporsional dengan gradien lokal pada kurva kesalahan terkait, adalah bahwa jalur yang diikuti algoritme penurunan gradien ke titik terendah pada permukaan kesalahan mungkin bukan garis lurus. . Hal ini karena gradien setiap kurva kesalahan komponen mungkin tidak sama di setiap lokasi pada permukaan kesalahan (gradien untuk salah satu bobot mungkin lebih curam daripada gradien untuk bobot lainnya). Akibatnya, satu bobot dapat diperbarui dengan jumlah yang lebih besar daripada bobot lain selama iterasi tertentu, dan dengan demikian penurunan ke dasar lembah mungkin tidak mengikuti rute langsung. Gambar 6.4 mengilustrasikan fenomena ini. Gambar 6.4 menyajikan sekumpulan tampilan atas-bawah dari sebagian plot kontur dari permukaan yang salah. Permukaan yang salah ini adalah lembah yang cukup panjang dan sempit dengan sisi yang lebih curam dan ujung yang lebih landai; kecuraman tercermin dari kedekatan kontur. Akibatnya, penelusuran awalnya bergerak melintasi lembah sebelum berbelok ke tengah lembah. Plot di sebelah kiri menggambarkan iterasi pertama dari algoritma penurunan gradien. Titik awal awal adalah lokasi pertemuan ketiga anak panah dalam plot ini. Panjang memenuhi. Panjang memenuhi. Panjangpanah putus-putus dan putus-putus masing-masing mewakili gradien lokal dari kurva w_0 dan w_1 kesalahan. Panah putus-putus lebih panjang dari panah putus-putus yang mencerminkan fakta bahwa gradien lokal dari w_0 kurva kesalahan lebih curam daripada w_1 kurva kesalahan. Dalam setiap iterasi, setiap bobot diperbarui secara proporsional dengan gradien kurva kesalahannya; jadi pada iterasi pertama,

pembaruan untuk w_0 lebih besar dari untuk w_1 dan oleh karena itu pergerakan keseluruhan lebih besar di sepanjang lembah daripada di sepanjang lembah. Panah hitam tebal mengilustrasikan pergerakan keseluruhan di ruang bobot yang mendasarinya, yang dihasilkan dari pembaruan bobot di iterasi pertama ini. Demikian pula, plot tengah mengilustrasikan gradien kesalahan dan pembaruan bobot keseluruhan untuk iterasi penurunan gradien berikutnya. Plot di sebelah kanan menunjukkan jalur lengkap penurunan yang diambil oleh proses pencarian dari lokasi awal ke minimum global (titik terendah di permukaan kesalahan).



Gambar 6.4 Tampilan atas-bawah dari bagian plot kontur permukaan error, yang menggambarkan jalur penurunan gradien di seluruh permukaan error. Setiap panah tebal menggambarkan pergerakan keseluruhan vektor bobot untuk satu iterasi algoritme penurunan gradien. Panjang panah putus-putus dan putus-putus mewakili gradien lokal dari kurva w_0 dan w_1 kesalahan, masing-masing, untuk iterasi itu. Plot di sebelah kanan menunjukkan jalur keseluruhan yang diambil ke minimum global permukaan kesalahan.

Relatif mudah untuk memetakan aturan pembaruan berat badan untuk melatih satu neuron. Dalam pemetaan ini, bobot w_0 adalah istilah bias untuk neuron, dan bobot lainnya dikaitkan dengan input lain ke neuron. Penurunan turunan parsial SSE bergantung pada struktur fungsi yang dihasilkan $\hat{y}$. Semakin kompleks fungsi ini, semakin kompleks pula turunan parsial. Fakta bahwa fungsi yang didefinisikan neuron mencakup penjumlahan berbobot dan fungsi aktivasi berarti bahwa turunan parsial SSE sehubungan dengan bobot dalam neuron lebih kompleks daripada turunan parsial yang diberikan di atas. Dimasukkannya fungsi aktivasi dalam neuron menghasilkan istilah tambahan dalam turunan parsial SSE. Suku ekstra ini adalah turunan dari fungsi aktivasi sehubungan dengan keluaran dari fungsi penjumlahan berbobot. Turunan dari fungsi aktivasi berkaitan dengan keluaran dari fungsi penjumlahan berbobot karena ini adalah masukan yang diterima oleh fungsi aktivasi. Fungsi aktivasi tidak menerima bobot secara langsung. Sebagai gantinya, Perubahan bobot hanya mempengaruhi keluaran dari fungsi aktivasi secara tidak langsung melalui pengaruh perubahan bobot ini terhadap keluaran penjumlahan berbobot. Alasan utama mengapa fungsi logistik menjadi fungsi aktivasi yang begitu populer di jaringan saraf begitu lama adalah karena ia memiliki turunan yang sangat langsung sehubungan dengan inputnya. ItuAturan pembaruan bobot penurunan gradien untuk neuron yang menggunakan fungsi logistik adalah sebagai berikut:

$$w_i^{t+1} = w_i^t + \left(\eta \times \underbrace{\sum_{j=1}^{n} \left((y_j^t - \hat{y}_j^t) \times \underbrace{(\hat{y}_j^t \times (1 - \hat{y}_j^t))}_{\substack{\text{derivative of the} \\ \text{logistic function} \\ \text{with respect to the} \\ \text{weighted summation}}} \times x_{j,i}^t \right)}_{\text{error gradient for } w_i} \right)$$

Fakta bahwa aturan pembaruan bobot mencakup turunan dari fungsi aktivasi berarti bahwa aturan pembaruan bobot akan berubah jika fungsi aktivasi neuron diubah. Namun, perubahan ini hanya akan melibatkan pembaruan turunan dari fungsi aktivasi; struktur keseluruhan aturan akan tetap sama.

Aturan pembaruan bobot yang diperpanjang ini berarti bahwa algoritme penurunan gradien dapat digunakan untuk melatih satu neuron. Namun, itu tidak dapat digunakan untuk melatih jaringan saraf dengan banyak lapisan neuron karena definisi gradien kesalahan untuk bobot bergantung pada kesalahan keluaran fungsi, istilah $y_j - \hat{y}_j$. Meskipun dimungkinkan untuk menghitung kesalahan keluaran neuron di lapisan keluaran jaringan dengan membandingkan secara langsungkeluaran dengan keluaran yang diharapkan, tidak mungkin menghitung istilah kesalahan ini secara langsung untuk neuron di lapisan tersembunyi jaringan, dan akibatnya tidak mungkin menghitung gradien kesalahan untuk setiap bobot. Algoritma propagasi mundur adalah solusi untuk masalah penghitungan gradien kesalahan untuk bobot di lapisan tersembunyi jaringan.

Melatih Jaringan Neural Menggunakan Backpropagation

Istilah propagasi mundur memiliki dua arti yang berbeda. Arti utamanya adalah bahwa ini adalah algoritma yang dapat digunakan untuk menghitung, untuk setiap neuron dalam jaringan, sensitivitas (gradien / laju perubahan) kesalahan jaringan terhadap perubahan bobot. Setelah gradien kesalahan untuk suatu bobot telah dihitung, bobot tersebut kemudian dapat disesuaikan untuk mengurangi keseluruhan error jaringan menggunakan aturan pembaruan bobot yang serupa dengan aturan pembaruan bobot penurunan gradien. Dalam pengertian ini, algoritma propagasi mundur adalah solusi untuk masalah penugasan kredit, yang diperkenalkan di bab 4. Arti kedua dari propagasi mundur adalah bahwa ini adalah algoritma lengkap untuk melatih jaringan saraf. Makna kedua ini mencakup

pengertian pertama, tetapi juga menyertakan aturan pembelajaran yang menentukan bagaimana gradien kesalahan bobot harus digunakan untuk memperbarui bobot dalam jaringan. Akibatnya, algoritmeyang dijelaskan dengan arti kedua ini melibatkan proses dua langkah: memecahkan masalah penetapan kredit, dan kemudian menggunakan gradien kesalahan bobot, dihitung selama penetapan kredit, untuk memperbarui bobot di jaringan. Berguna untuk membedakan kedua arti propagasi mundur ini karena ada sejumlah aturan pembelajaran berbeda yang dapat digunakan untuk memperbarui bobot, setelah masalah penugasan kredit diselesaikan. Aturan pembelajaran yang paling umum digunakan dengan propagasi mundur adalah algoritma penurunan gradien yang diperkenalkan sebelumnya. Uraian tentang algoritma propagasi mundur yang diberikan di sini berfokus pada makna pertama dari propagasi mundur, yaitu algoritma tersebut menjadi solusi untuk masalah penugasan kredit.

Propagasi Balik: Algoritma Dua Tahap

Algoritme propagasi mundur dimulai dengan menginisialisasi semua bobot jaringan menggunakan nilai acak. Perhatikan bahwa bahkan jaringan yang diinisialisasi secara acak masih dapat menghasilkan keluaran saat masukan disajikan ke jaringan, meskipun kemungkinan besar merupakan keluaran dengan kesalahan besar. Setelah bobot jaringan diinisialisasi, jaringan dapat dilatih dengan memperbarui bobot secara berulang untuk mengurangi kesalahan jaringan, di mana kesalahan jaringan dihitung dalam hal perbedaan antara keluaran yang dihasilkan oleh jaringan sebagai tanggapan. ke pola masukan, dan keluaran yang diharapkan untuk masukan itu, sebagaididefinisikan dalam set data pelatihan. Langkah penting dalam proses penyesuaian bobot berulang ini melibatkan penyelesaian masalah penetapan kredit, atau, dengan kata lain, menghitung gradien kesalahan untuk setiap bobot di jaringan. Algoritme propagasi mundur memecahkan masalah ini menggunakan proses dua tahap. Pada tahap pertama, yang dikenal sebagai forward pass, pola masukan disajikan ke jaringan, dan aktivasi neuron yang dihasilkan mengalir ke depan melalui jaringan sampai keluaran dihasilkan. Gambar 6.5 mengilustrasikan forward pass dari algoritma backpropagation. Dalam gambar ini, penjumlahan input yang dihitung pada setiap neuron (misalnya, z_1 merepresentasikan penjumlahan input yang dihitung untuk neuron 1) dan output (atau aktivasi, misalnya, a_1 mewakili aktivasi untuk neuron 1) dari setiap neuron ditampilkan. Alasan untuk membuat daftar nilai z_i dan a_i untuk setiap neuron pada gambar ini adalah untuk menyoroti fakta bahwa selama forward pass, kedua nilai ini, untuk setiap neuron, disimpan dalam memori. Alasan mereka disimpan dalam memori adalah karena mereka digunakan dalam algoritma backward pass. The z_i nilai neuron yang digunakan untuk menghitung update untuk bobot pada koneksi input ke neuron. The a_i nilai neuron yang digunakan untuk menghitung update untuk bobot pada koneksi output dari neuron. Secara

spesifik bagaimana nilai-nilai ini digunakan pada backward pass akan dijelaskan di bawah ini.

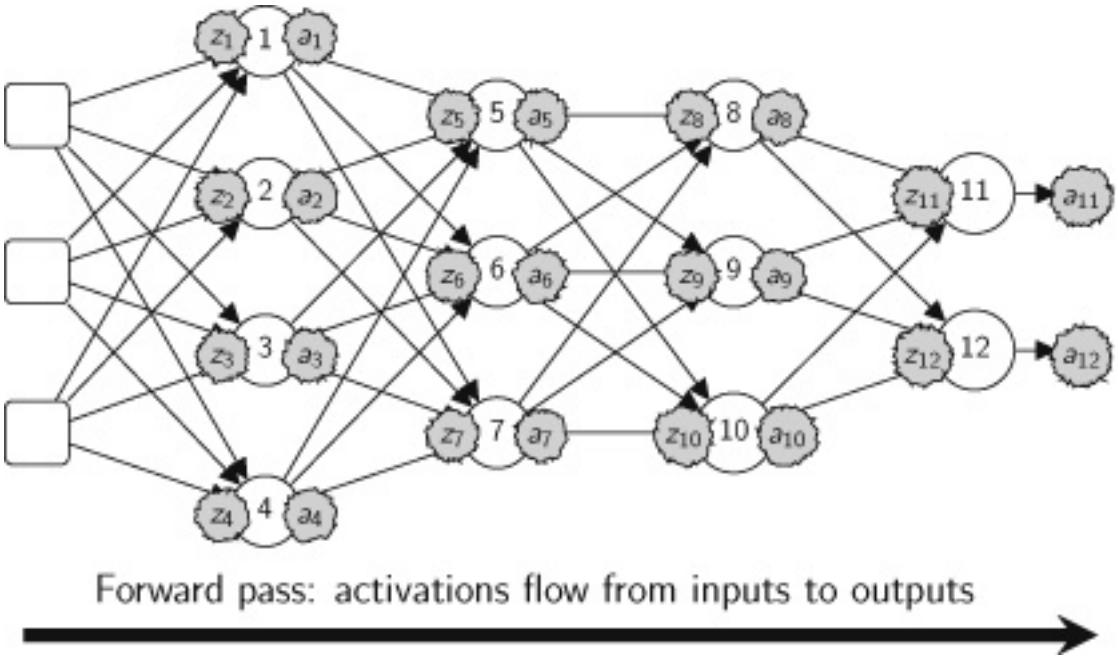
Tahap kedua, yang dikenal sebagai backward pass, dimulai dengan menghitung gradien kesalahan untuk setiap neuron masuk lapisan keluaran. Gradien kesalahan ini mewakili sensitivitas kesalahan jaringan terhadap perubahan dalam perhitungan penjumlahan berbobot neuron, dan mereka sering dilambangkan dengan notasi singkatan δ (diucapkan delta) dengan subskrip yang menunjukkan neuron. Misalnya, δ_k adalah gradien dari kesalahan jaringan sehubungan dengan perubahan kecil dalam perhitungan penjumlahan terbobot dari neuron k . Penting untuk diketahui bahwa ada dua gradien kesalahan berbeda yang dihitung dalam algoritma propagasi mundur:

1. Yang pertama adalah δ nilai untuk setiap neuron. Untuk δ setiap neuron adalah laju perubahan kesalahan jaringan sehubungan dengan perubahan dalam perhitungan penjumlahan berbobot dari neuron tersebut. Ada satu δ untuk setiap neuron. Inilah δ gradien kesalahan bahwa backpropagates algoritma.
2. Yang kedua adalah gradien kesalahan jaringan sehubungan dengan perubahan bobot jaringan. Ada salah satu dari gradien kesalahan ini untuk setiap bobot di jaringan. Ini adalah gradien kesalahan yang digunakan untuk memperbarui bobot di jaringan. Namun, pertama-tama perlu menghitung suku δ untuk setiap neuron (menggunakan propagasi mundur) untuk menghitung gradien kesalahan untuk bobot.

Perhatikan bahwa hanya ada satu δ per neuron, tetapi mungkin ada banyak bobot yang terkait dengan neuron itu, sehingga δ istilah untuk neuron dapat digunakan dalam perhitungan gradien kesalahan bobot ganda.

Setelah δ s untuk neuron keluaran telah dihitung, δ s untuk neuron di lapisan tersembunyi terakhir kemudian dihitung. Ini dilakukan dengan menetapkan sebagian δ dari setiap neuron keluaran ke setiap neuron tersembunyi yang terhubung langsung dengannya. Penetapan kesalahan ini, dari output neuron ke hidden neuron, bergantung pada berat koneksi antara neuron, dan aktivasi neuron tersembunyi selama forward pass (inilah mengapa aktivasi dicatat dalam memori selama forward pass) . Setelah penetapan kesalahan, dari lapisan keluaran, telah diselesaikan, δ untuk setiap neuron di lapisan tersembunyi terakhir dihitung dengan menjumlahkan bagian dari δ s ditugaskan ke neuron dari semua neuron keluaran yang terhubung dengannya. Proses yang sama dari tugas menyalahkan dan menjumlahkan kemudian diulangi untuk menyebarkan gradien kesalahan kembali dari lapisan terakhir neuron tersembunyi ke neuron di lapisan terakhir kedua, dan seterusnya, kembali ke lapisan masukan. Ini adalah propagasi mundur

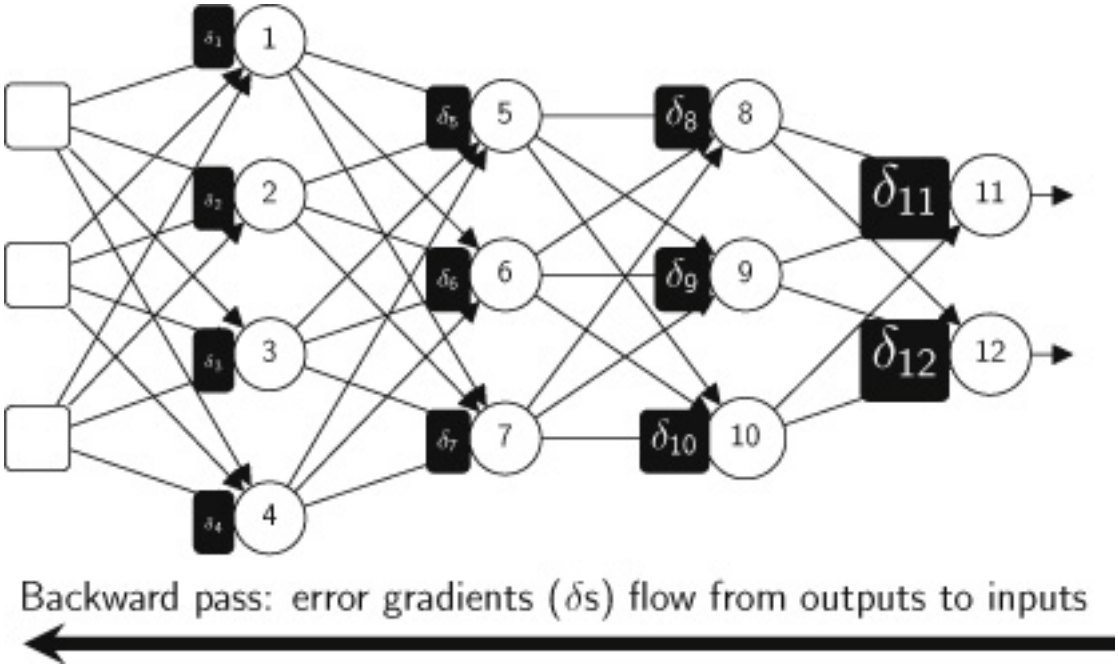
δs melalui jaringan yang memberi nama algoritma. Di akhir backward pass ini ada δpenghitungan untuk setiap neuron di jaringan (yaitu, masalah penetapan kredit telah diselesaikan) dan ini δkemudian dapat digunakan untuk memperbarui bobot di jaringan (menggunakan, misalnya,algoritma penurunan gradien yang diperkenalkan sebelumnya). Gambar 6.6 mengilustrasikan backward pass dari algoritma backpropagation. Dalam gambar ini, δs semakin kecil dan semakin kecil karena proses propagasi mundur semakin jauh dari lapisan keluaran. Hal ini mencerminkan masalah gradien menghilang yang dibahas dalam bab 4 yang memperlambat kecepatan pemelajaran lapisan awal jaringan.



Gambar 6.5 Algoritma forward pass dari algoritma propagasi mundur.

Singkatnya, langkah-langkah utama dalam setiap iterasi dari algoritma propagasi mundur adalah sebagai berikut:

1. Sajikan masukan ke jaringan dan biarkan aktivasi neuron mengalir maju melalui jaringan sampai keluaran dihasilkan. Catat jumlah tertimbang dan aktivasi setiap neuron.



Gambar 6.6 Lintasan mundur dari algoritma propagasi mundur.

2. Hitung δ gradien kesalahan (delta) untuk setiap neuron di lapisan keluaran.
3. Propagasi mundur δ gradien kesalahan untuk mendapatkan gradien kesalahan δ (delta) untuk setiap neuron dalam jaringan.
4. Gunakan δ gradien kesalahan dan algoritma pembaruan bobot, seperti penurunan gradien, untuk menghitung gradien kesalahan untuk bobot dan gunakan ini untuk memperbarui bobot di jaringan.

Algoritme terus melakukan iterasi melalui langkah-langkah ini sampai kesalahan jaringan dikurangi (atau digabungkan) ke tingkat yang dapat diterima.

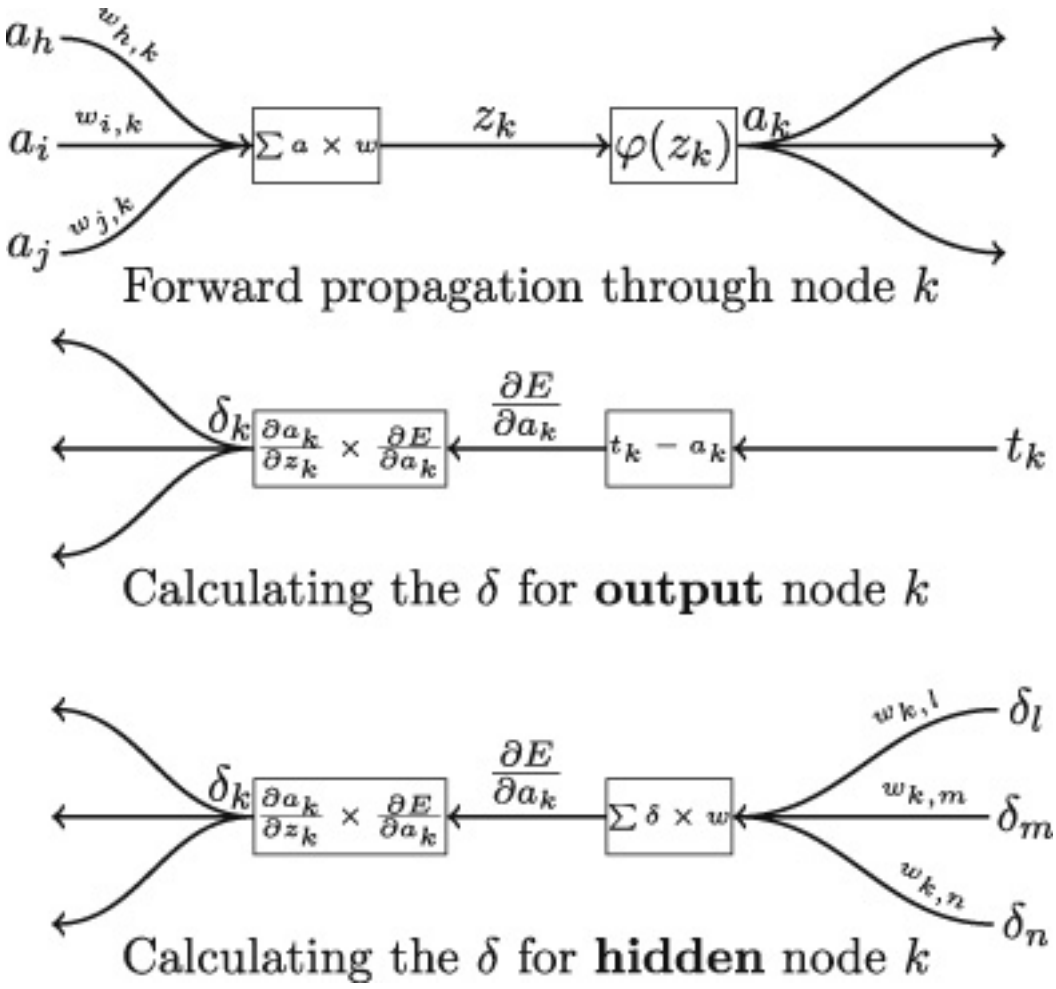
Propagasi mundur: Propagasi mundur δ s

Sebuah δ istilah neuron menggambarkan gradien kesalahan untuk jaringan sehubungan dengan perubahan dalam penjumlahan input yang dihitung oleh neuron. Untuk membantu membuat ini lebih konkret, gambar 6.7 (atas) membuka tahapan pemrosesan dalam neuron k dan menggunakan istilah tersebut z_k untuk menunjukkan hasil penjumlahan berbobot dalam neuron. Neuron dalam gambar ini menerima masukan (atau aktivasi) dari tiga neuron lain (h, i, j), dan z_k merupakan jumlah bobot aktivasi ini. Keluaran neuron,, a_k kemudian dihitung dengan melewatkan z_k fungsi aktivasi nonlinier φ, seperti fungsi logistik. Menggunakan notasi ini a δ untuk neuron k adalah tingkat perubahan kesalahan jaringan sehubungan dengan perubahan kecil dalam nilai z_k. Secara matematis, istilah ini adalah turunan parsial dari kesalahan jaringan sehubungan dengan z_k :

$$\delta_k = \frac{\partial \text{Error}}{\partial z_k}$$

Tidak peduli di mana dalam jaringan neuron berada (lapisan keluaran atau lapisan tersembunyi), δ untuk neuron dihitung sebagai produk dari dua istilah:

1. laju perubahan kesalahan jaringan sebagai respons terhadap perubahan aktivasi neuron (keluaran): $\partial E / \partial a_k$;



Gambar 6.7 Atas: propagasi maju dari aktivasi melalui penjumlahan tertimbang dan fungsi aktivasi dari sebuah neuron. Tengah: Perhitungan δ istilah untuk neuron keluaran (t_k adalah aktivasi yang diharapkan untuk neuron dan a_k merupakan aktivasi sebenarnya). Bawah: Perhitungan δ istilah untuk neuron tersembunyi. Angka ini secara longgar diilhami oleh gambar 5.2 dan gambar 5.3 dalam Reed dan Marks II 1999.

2. laju perubahan dari aktivasi neuron sehubungan dengan perubahan dalam jumlah tertimbang dari input ke neuron: $\partial a_k / \partial z_k$.

$$\delta_k = \frac{\partial E}{\partial a_k} \times \frac{\partial a_k}{\partial z_k}$$

Gambar 6.7 (tengah) mengilustrasikan bagaimana produk ini dihitung untuk neuron di lapisan keluaran jaringan. Langkah pertama adalah menghitung laju perubahan kesalahan jaringan sehubungan dengan keluaran neuron, istilahnya $\partial E / \partial a_k$. Secara intuitif, semakin besar perbedaan antara aktivasi neuron, a_k dan aktivasi yang diharapkan t_k, semakin cepat kesalahan dapat diubah dengan mengubah aktivasi neuron. Jadi tingkat perubahan kesalahan jaringan sehubungan dengan perubahan dalam aktivasi neuron keluaran k dapat dihitung dengan mengurangi aktivasi neuron (a_k) dari aktivasi yang diharapkan (t_k):

$$\frac{\partial E}{\partial a_k} = t_k - a_k$$

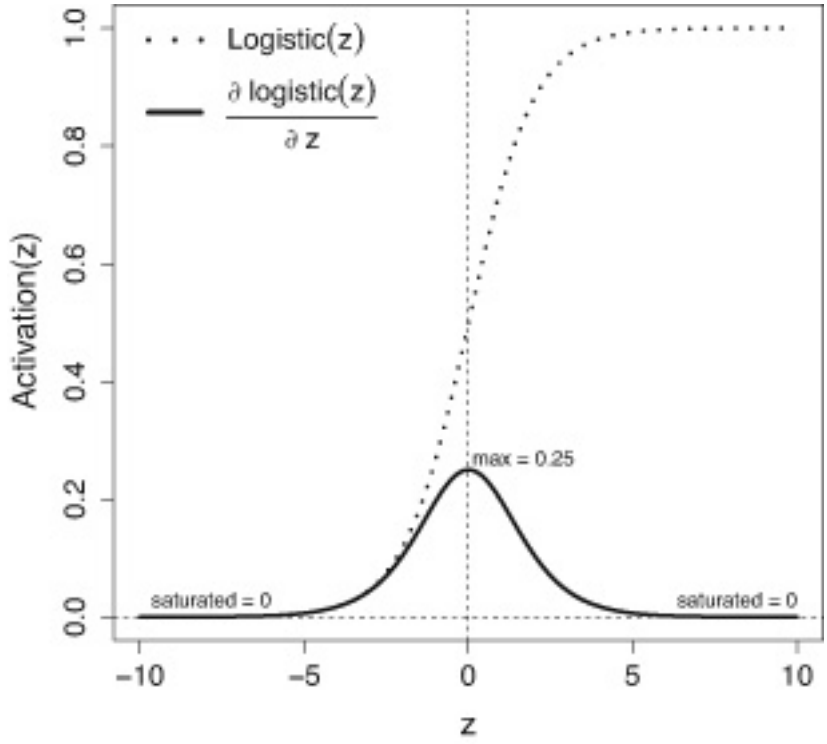
Istilah ini menghubungkan kesalahan jaringan ke keluaran neuron. Neuron δ, bagaimanapun, adalah tingkat perubahan kesalahan sehubungan dengan masukan ke fungsi aktivasi (z_k), bukan keluaran dari fungsi itu (a_k). Akibatnya, untuk menghitung δ untukneuron, $\partial E / \partial a_k$ nilainya harus disebarkan kembali melalui fungsi aktivasi untuk menghubungkannya ke input ke fungsi aktivasi. Ini dilakukan dengan mengalikan $\partial E / \partial a_k$ dengan laju perubahan fungsi aktivasi sehubungan dengan nilai input ke fungsi z_k ,. Pada gambar 6.7, tingkat perubahan dari fungsi aktivasi sehubungan dengan input dilambangkan dengan istilah: $\partial a_k / \partial z_k$. Istilah ini dihitung dengan memasukkan nilai z_k (disimpan dari forward pass melalui jaringan) ke dalam persamaan turunan fungsi aktivasi yang terkait dengan z_k. Misalnya, turunan dari fungsi logistik sehubungan dengan inputnya adalah:

$$\frac{\partial \text{logistic}(z)}{\partial z} = \text{logistic}(z) \times (1 - \text{logistic}(z))$$

Gambar 6.8 memplot fungsi ini dan menunjukkan bahwa memasukkan z_k nilai ke dalam persamaan ini akan menghasilkan nilai antara 0 dan 0,25. Misalnya, gambar 6.8 menunjukkan jika $z_k = 0$ kemudian $\partial a_k / \partial z_k = 0.25$. Inilah sebabnya mengapa nilai penjumlahan tertimbang untuk setiap neuron (z_k) disimpan selama penerusan algoritma.

Fakta [1] bahwa penghitungan neuron δ melibatkan produk yang mencakup turunan dari fungsi aktivasi neuron membuatnya perlu untuk dapat mengambil turunan dari fungsi aktivasi neuron. Tidak mungkin untuk mengambil turunan dari

aktivasi ambang batasberfungsi karena ada diskontinuitas dalam fungsi di ambang batas. Akibatnya, algoritme propagasi mundur tidak berfungsi untuk jaringan yang terdiri dari neuron yang menggunakan fungsi aktivasi ambang batas. Ini adalah salah satu alasan mengapa jaringan saraf menjauh dari aktivasi ambang dan mulai menggunakan fungsi aktivasi logistik dan tanh. Fungsi logistik dan tanh keduanya memiliki turunan yang sangat sederhana dan ini membuatnya sangat cocok untuk propagasi mundur.



Gambar 6.8 Plot dari fungsi logistik dan turunan dari fungsi logistik.

Gambar 6.7 (bawah) mengilustrasikan bagaimana δ neuron untuk sebuah lapisan tersembunyi dihitung. Ini melibatkan hal yang samaproduk istilah seperti yang digunakan untuk neuron di lapisan keluaran. Perbedaannya adalah bahwa penghitungannya $\partial E / \partial a_k$ lebih kompleks untuk unit tersembunyi. Untuk neuron tersembunyi, tidak mungkin untuk secara langsung menghubungkan keluaran neuron dengan kesalahan jaringan. Keluaran neuron tersembunyi hanya secara tidak langsung mempengaruhi keseluruhan kesalahan jaringan melalui variasi yang ditimbulkannya di neuron hilir yang menerima keluaran sebagai masukan, dan besarnya variasi ini bergantung pada bobot yang diterapkan masing-masing neuron hilir ini. hasil. Selain itu, efek tidak langsung pada kesalahan jaringan ini pada gilirannya bergantung pada sensitivitas kesalahan jaringan terhadap neuron-neuron selanjutnya ini, yaitu, neuron mereka. δ nilai-nilai. Akibatnya, sensitivitas kesalahan jaringan terhadap keluaran neuron tersembunyi dapat dihitung sebagai jumlah tertimbang dari δ nilai - nilai neuron yang berada tepat di hilir neuron:

$$\frac{\partial E}{\partial a_k} = \sum_{i=1}^{N} w_{k,i} \times \delta_i$$

Akibatnya, istilah kesalahan (δ nilai) untuk semua neuron hilir yang dilewati keluaran neuron di jalur maju harus dihitung sebelum $\partial E / \partial a_k$ for neuron *k* dapat dihitung. Ini, bagaimanapun, tidak menjadi masalah karena di backward pass algoritma bekerja mundur melalui jaringan dan akan dihitungyang δ istilah untuk neuron hilir sebelum mencapai neuron *k* .

Untuk neuron tersembunyi, istilah lain dalam δ produk,, $\partial a_k / \partial z_k$ dihitung dengan cara yang sama seperti yang dihitung untuk neuron keluaran: z_k nilai untuk neuron (penjumlahan masukan berbobot, disimpan selama penerusan melewati jaringan) dicolokkan ke turunan dari fungsi aktivasi neuron sehubungan dengan z_k .

Propagasi mundur: Memperbarui Bobot

Prinsip dasar algoritma backpropagation dalam menyesuaikan bobot dalam jaringan adalah bahwa setiap bobot dalam jaringan harus diperbarui secara proporsional dengan sensitivitas kesalahan jaringan secara keseluruhan terhadap perubahan bobot tersebut. Intinya adalah bahwa jika kesalahan keseluruhan jaringan tidak dipengaruhi oleh perubahan bobot, maka kesalahan jaringan tidak bergantung pada bobot itu, dan, oleh karena itu, bobot tidak berkontribusi pada kesalahan. Sensitivitas kesalahan jaringan terhadap perubahan bobot individu diukur dalam hal tingkat perubahan kesalahan jaringan dalam menanggapi perubahan bobot tersebut.

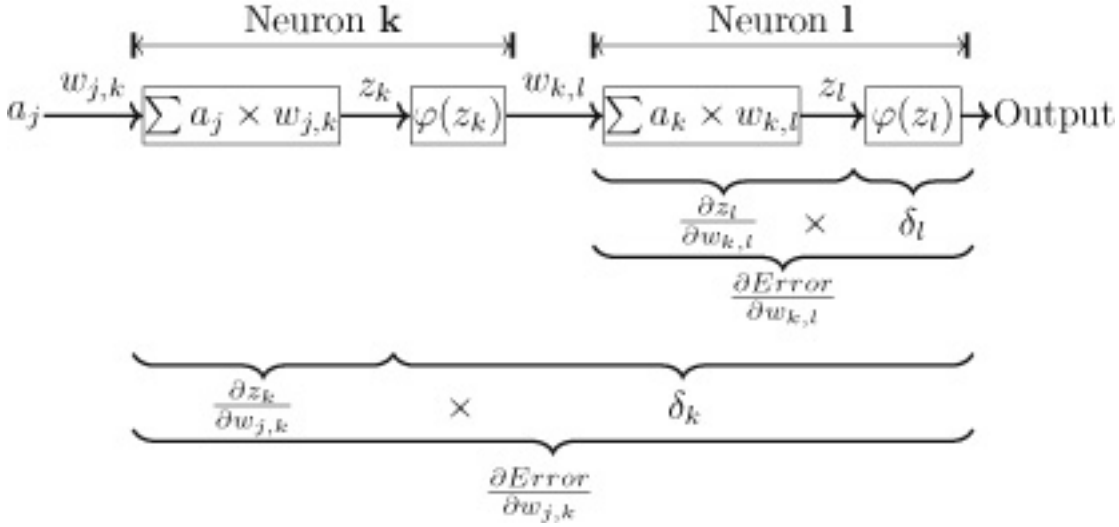
Prinsip dasar algoritma backpropagation dalam menyesuaikan bobot dalam jaringan adalah bahwa setiap bobot dalam jaringan harus diperbarui secara proporsional dengan sensitivitas kesalahan jaringan secara keseluruhan terhadap perubahan bobot tersebut.

Kesalahan keseluruhan jaringan adalah fungsi dengan banyak masukan: baik masukan ke jaringan dan semua bobot di jaringan. Jadi, laju perubahan kesalahan jaringan sebagai respons terhadap perubahan bobot jaringan tertentu dihitung dengan mengambil turunan parsial dari kesalahan jaringan sehubungan dengan bobot tersebut. Dalamalgoritma propagasi mundur, turunan parsial dari kesalahan jaringan untuk bobot tertentu dihitung menggunakan aturan rantai. Dengan menggunakan aturan rantai, turunan parsial dari kesalahan jaringan sehubungan dengan bobot $w_{j,k}$ pada hubungan antara neuron j dan neuron k dihitung sebagai produk dari dua istilah:

1. istilah pertama menjelaskan laju perubahan jumlah bobot masukan dalam neuron k sehubungan dengan perubahan bobot $\partial z_k / \partial w_{j,k}$;
2. dan istilah kedua menjelaskan tingkat perubahan kesalahan jaringan dalam menanggapi perubahan jumlah input yang dihitung oleh neuron k. (Istilah kedua ini δ_k untuk neuron k.)

Gambar 6.9 menunjukkan bagaimana produk dari kedua istilah ini menghubungkan bobot dengan kesalahan keluaran jaringan. Gambar tersebut menunjukkan pemrosesan dua neuron terakhir (k dan l) dalam jaringan dengan jalur aktivasi tunggal. Neuron k menerima satu masukan a_j dan keluaran dari neuron k adalah satu-satunya masukan ke neuron l. Keluaran neuron l adalah keluaran dari jaringan. Ada dua bobot di bagian jaringan ini, $w_{j,k}$ dan $w_{k,l}$.

Perhitungan yang ditunjukkan pada gambar 6.9 tampak rumit karena mengandung sejumlah komponen yang berbeda. Namun, seperti yang akan kita lihat, dengan melangkah melalui kalkulasi ini, masing-masing elemen individu sebenarnyamudah dihitung; itu hanya melacak semua elemen berbeda yang menimbulkan kesulitan.



Gambar 6.9 Ilustrasi bagaimana produk turunan menghubungkan bobot dalam jaringan ke kesalahan jaringan.

Berfokus pada $w_{k,l}$, bobot ini diterapkan ke input dari neuron keluaran jaringan. Ada dua tahap pemrosesan antara bobot ini dan keluaran jaringan (dan kesalahan): yang pertama adalah jumlah bobot yang dihitung dalam neuron l; yang kedua adalah fungsi nonlinier yang diterapkan pada penjumlahan terbobot ini dengan fungsi aktivasi neuron l. Bekerja mundur dari output, δ_l istilah dihitung menggunakan perhitungan yang ditunjukkan pada gambar tengah gambar 6.7: perbedaan antara aktivasi target untuk neuron dan aktivasi sebenarnya dihitung dan dikalikan dengan turunan parsial dari fungsi aktivasi neuron dengan

sehubungan dengan masukannya (jumlah tertimbang z_k) $\partial a_l / \partial z_l$,. Dengan asumsi bahwa fungsi aktivasi digunakanoleh neuron l adalah fungsi logistik, istilah $\partial a_l / \partial z_l$ ini dihitung dengan memasukkan nilai z_l (disimpan selama penerusan algoritme) ke dalam penurunan fungsi logistik:

$$\frac{\partial a_l}{\partial z_l} = \frac{\partial \text{logistic}(z_l)}{\partial z_l} = \text{logistic}(z_l) \times (1 - \text{logistic}(z_l))$$

Jadi perhitungan δ_l dengan asumsi neuron l menggunakan fungsi logistik adalah:

$$\delta_l = \text{logistic}(z_l) \times (1 - \text{logistic}(z_l)) \times (t_l - a_l)$$

The δ_l jangka menghubungkan kesalahan dari jaringan ke input ke fungsi aktivasi (jumlah tertimbang z_l). Namun, kami ingin menghubungkan kesalahan jaringan kembali ke beban $w_{k,l}$. Hal ini dilakukan dengan mengalikan δ_l istilah oleh turunan parsial dari fungsi penjumlahan tertimbang sehubungan dengan berat badan $w_{k,l}$: $\partial z_l / \partial w_{k,l}$. Turunan parsial ini menjelaskan bagaimana output dari fungsi penjumlahan terbobot z_l berubah seiring dengan $w_{k,l}$ perubahan bobot . Fakta bahwa fungsi penjumlahan berbobot adalah fungsi linier dari bobot dan aktivasi berarti bahwa dalam turunan parsial berkenaan dengan bobot tertentu, semua istilah dalam fungsi yang tidak melibatkan bobot spesifik menjadi nol (dianggap konstanta) dan turunan parsialmenyederhanakan menjadi hanya masukan yang terkait dengan bobot itu, dalam contoh ini masukan a_k .

$$\frac{\partial z_l}{\partial w_{k,l}} = a_k$$

Inilah sebabnya mengapa aktivasi untuk setiap neuron di jaringan disimpan di forward pass. Secara bersama-sama, kedua istilah ini, $\partial z_l / \partial w_{k,l}$ dan δ_l, menghubungkan bobot $w_{k,l}$ ke kesalahan jaringan dengan terlebih dahulu menghubungkan bobot ke z_l, dan kemudian menghubungkan z_l ke aktivasi neuron, dan dengan demikian ke kesalahan jaringan. Jadi, gradien kesalahan jaringan sehubungan dengan perubahan bobot $w_{k,l}$ dihitung sebagai:

$$\frac{\partial \text{Error}}{\partial w_{k,l}} = \frac{\partial z_l}{\partial w_{k,l}} \times \delta_l = a_k \times \delta_l$$

Bobot lain pada gambar 6.9 jaringan,, $w_{k,l}$ lebih dalam di jaringan, dan akibatnya, ada lebih banyak langkah pemrosesan antara jaringan dan keluaran jaringan (dan kesalahan). The δ istilah untuk neuron k dihitung, melalui backpropagation (seperti yang ditunjukkan di bagian bawah angka 6,7), menggunakan produk berikut istilah:

$$\delta_k = \frac{\partial a_k}{\partial z_k} \times (w_{k,l} \times \delta_l)$$

Dengan asumsi fungsi aktivasi yang digunakan oleh neuron k adalah fungsi logistik, maka istilah $\partial a_k / \partial z_k$ tersebut dihitung dengan cara yang mirip dengan $\partial a_l / \partial z_l$: nilai z_k dimasukkan ke dalam persamaan untuk turunan dari fungsi logistik. Jadi, dituliskan dalam bentuk panjang perhitungannya δ_k adalah:

$$\delta_k = \text{logistic}(z_k) \times (1 - \text{logistic}(z_k)) \times (w_{k,l} \times \delta_l)$$

Namun, dalam rangka untuk menghubungkan berat badan $w_{j,k}$ dengan kesalahan jaringan, istilah δ_k harus dikalikan dengan turunan parsial dari fungsi penjumlahan tertimbang sehubungan dengan berat: $\partial z_k / \partial w_{j,k}$. Seperti dijelaskan di atas, turunan parsial dari fungsi penjumlahan tertimbang sehubungan dengan suatu bobot direduksi menjadi masukan yang terkait dengan bobot $w_{j,k}$ (yaitu, a_j); dan gradien kesalahan jaringan sehubungan dengan berat tersembunyi $w_{j,k}$ dihitung dengan mengalikan a_j dengan δ_k. Konsekuensinya, produk dari istilah ($\partial z_k / \partial w_{j,k}$ dan δ_k) membentuk rantai yang menghubungkan bobot $w_{j,k}$ ke kesalahan jaringan. Untuk kelengkapan, produk istilah untuk $w_{j,k}$, dengan asumsi fungsi aktivasi logistik di neuron, adalah:

$$\frac{\partial \text{Error}}{\partial w_{j,k}} = \frac{\partial z_k}{\partial w_{j,k}} \times \delta_k = a_j \times \delta_k$$

Meskipun diskusi ini telah dibingkai dalam konteks jaringan yang sangat sederhana dengan hanya satu jalur koneksi, itu menggeneralisasi ke jaringan yang lebih kompleks karena perhitungan δ istilah untuk unit tersembunyi sudah memperhitungkan beberapa koneksi yang berasal dari neuron. Setelah gradien kesalahan jaringan sehubungan dengan suatu bobot telah dihitung ($\partial \text{Error} / w_{j,k} = \delta_k \times a_j$), bobot tersebut dapat disesuaikan untuk mengurangi bobot jaringan menggunakan aturan pembaruan bobot penurunan gradien. Berikut adalah aturan pembaruan bobot, ditentukan menggunakan notasi dari backpropagation, untuk bobot pada koneksi antara neuron j dan neuron k selama iterasi t algoritme:

$$w_{j,k}^{t+1} = w_{j,k}^{t} + (\eta \times \delta_k \times a_j)$$

Akhirnya, peringatan penting dalam melatih jaringan saraf dengan propagasi mundur dan penurunan gradien adalah bahwa permukaan kesalahan jaringan saraf jauh lebih kompleks daripada yang ada pada model linier. Gambar 6.3 mengilustrasikan permukaan kesalahan model linier sebagai mangkuk cembung mulus dengan minimum global tunggal (satu set bobot terbaik). Namun, permukaan error dari jaringan neural lebih seperti pegunungan dengan beberapa lembah dan puncak. Hal ini karena setiap neuron dalam jaringan menyertakan fungsi nonlinier dalam pemetaan input ke output, sehingga fungsi yang diimplementasikan oleh jaringan adalah fungsi nonlinier. Memasukkan nonlinier dalam neuron jaringan meningkatkan kekuatan ekspresif jaringandalam hal kemampuannya untuk mempelajari fungsi yang lebih kompleks. Namun, harga yang harus dibayar untuk ini adalah bahwa permukaan kesalahan menjadi lebih kompleks dan algoritme penurunan gradien tidak lagi dijamin untuk menemukan himpunan bobot yang menentukan minimum global pada permukaan kesalahan; malah mungkin macet dalam minima (minimum lokal). Untungnya, bagaimanapun, propagasi mundur dan penurunan gradien sering kali masih dapat menemukan kumpulan bobot yang menentukan model yang berguna, meskipun mencari model yang berguna mungkin memerlukan menjalankan proses pelatihan beberapa kali untuk menjelajahi bagian berbeda dari lanskap permukaan kesalahan.

7

Masa Depan Pembelajaran Mendalam

Pada 27 Maret 2019, Yoshua Bengio, Geoffrey Hinton, dan Yann LeCun bersama-sama menerima penghargaan ACM AM Turing. Penghargaan tersebut mengakui kontribusi yang telah mereka berikan pada pembelajaran mendalam yang menjadi

teknologi kunci yang mendorong revolusi kecerdasan buatan modern. Sering digambarkan sebagai "Hadiah Nobel untuk Komputasi," penghargaan ACM AM Turing membawa hadiah $ 1 juta. Kadang-kadang bekerja sama, dan di waktu lain bekerja secara mandiri atau bekerja sama dengan orang lain, ketiga peneliti ini, selama beberapa dekade bekerja, memberikan banyak kontribusi untuk pembelajaran mendalam, mulai dari mempopulerkan propagasi mundur pada 1980-an, hingga pengembangan jaringan neural konvolusional, embeddings kata, mekanisme perhatian dalam jaringan, dan jaringan adversarial generatif (untuk mencantumkan hanya beberapa contoh).bahwa pembelajaran mendalam telah mengarah pada visi komputer, robotika, pengenalan ucapan, dan pemrosesan bahasa alami, serta dampak mendalam yang ditimbulkan oleh teknologi ini terhadap masyarakat, dengan miliaran orang sekarang menggunakan kecerdasan buatan berbasis pembelajaran mendalam setiap hari melalui aplikasi ponsel pintar. Pengumuman itu juga menyoroti bagaimana pembelajaran mendalam telah memberi para ilmuwan alat-alat baru yang kuat yang menghasilkan terobosan ilmiah di berbagai bidang seperti kedokteran dan astronomi. Pemberian penghargaan ini kepada para peneliti ini mencerminkan pentingnya pembelajaran yang mendalam bagi sains dan masyarakat modern. Efek transformatif dari pembelajaran mendalam pada teknologi akan meningkat selama beberapa dekade mendatang dengan pengembangan dan adopsi pembelajaran mendalam yang terus didorong oleh siklus yang baik dari kumpulan data yang semakin besar, pengembangan algoritma baru, dan perangkat keras yang ditingkatkan. Tren ini tidak berhenti, dan bagaimana komunitas pembelajaran mendalam menanggapinya akan mendorong pertumbuhan dan inovasi dalam bidang ini selama beberapa tahun mendatang.

Inovasi Algoritmik Penggerak Big Data

Bab 1 memperkenalkan berbagai jenis machine learning: supervised, unsupervised, dan reinforcement learning. Sebagian besar buku ini berfokus pada pembelajaran yang diawasi,terutama karena ini adalah bentuk pembelajaran mesin yang paling populer. Namun, kesulitan dengan supervised learning adalah membutuhkan banyak uang dan waktu untuk membuat anotasi set data dengan label target yang diperlukan. Karena kumpulan data terus berkembang, biaya anotasi data menjadi penghalang untuk pengembangan aplikasi baru. Dataset ImageNet[1] memberikan contoh yang berguna dari skala tugas anotasi yang terlibat dalam proyek pembelajaran mendalam. Data ini dirilis pada tahun 2010, dan merupakan dasar untuk Tantangan Pengenalan Visual Skala Besar ImageNet (ILSVRC). Ini adalah tantangan yang dimenangkan oleh AlexNet CNN pada tahun 2012 dan dimenangkan oleh sistem ResNet pada tahun 2015. Seperti yang telah dibahas di bab 4, AlexNet memenangkan tantangan ILSVRC tahun 2012 menghasilkan banyak kegembiraan tentang model pembelajaran yang mendalam. Namun, kemenangan AlexNet tidak akan mungkin terjadi tanpa pembuatan set

data ImageNet. Kumpulan data ini berisi lebih dari empat belas juta gambar yang telah dianotasi secara manual untuk menunjukkan objek mana yang ada di setiap gambar; dan lebih dari satu juta gambar sebenarnya telah dianotasi dengan kotak pembatas objek pada gambar. Membuat anotasi data pada skala ini membutuhkan upaya dan anggaran penelitian yang signifikan, dan dicapai dengan menggunakan platform crowdsourcing. Tidaklah mungkin membuat set data beranotasi sebesar ini untuk setiap aplikasi.

Karena kumpulan data terus berkembang, biaya anotasi data menjadi penghalang untuk pengembangan aplikasi baru.

Salah satu tanggapan terhadap tantangan anotasi ini adalah minat yang semakin besar dalam pembelajaran tanpa pengawasan. Itumodel autoencoder yang digunakan dalam pra-pelatihan Hinton (lihat bab 4) adalah salah satu pendekatan jaringan saraf untuk pembelajaran tanpa pengawasan, dan dalam beberapa tahun terakhir berbagai jenis autoencoder telah diusulkan. Pendekatan lain untuk masalah ini adalah dengan melatih model generatif. Model generatif mencoba mempelajari distribusi data (atau, untuk memodelkan proses yang menghasilkan data). Mirip dengan autoencoder, model generatif sering digunakan untuk mempelajari representasi data yang berguna sebelum melatih model yang diawasi. Generative adversarial networks (GANs) adalah pendekatan untuk melatih model generatif yang telah mendapat banyak perhatian dalam beberapa tahun terakhir (Goodfellow et al. 2014). GAN terdiri dari dua jaringan neural, model generatif dan model diskriminatif, serta sampel data nyata. Model-model tersebut dilatih secara bermusuhan. Tugas dari model diskriminatif adalah belajar membedakan antara data nyata yang diambil sampelnya dari dataset, dan data palsu yang telah disintesis oleh generator. Tugas generator adalah belajar mensintesis data palsu yang dapat menipu model diskriminatif. Model generatif yang dilatih menggunakan GAN dapat belajar menyintesis gambar palsu yang meniru gaya artistik (Elgammal et al. 2017), dan juga untuk mensintesis gambar medis bersama dengan anotasi lesi (Frid-Adar et al. 2018). Belajar untuk mensintesis citra medis, bersama dengan segmentasi lesi pada citra yang disintesis, membuka kemungkinan secara otomatis menghasilkan set data berlabel besar yang dapat digunakan. Tugas generator adalah belajar mensintesis data palsu yang dapat menipu model diskriminatif. Model generatif yang dilatih menggunakan GAN dapat belajar menyintesis gambar palsu yang meniru gaya artistik (Elgammal et al. 2017), dan juga untuk mensintesis gambar medis bersama dengan anotasi lesi (Frid-Adar et al. 2018). Belajar untuk mensintesis citra medis, bersama dengan segmentasi lesi pada citra yang disintesis, membuka kemungkinan secara otomatis menghasilkan set data berlabel besar yang dapat digunakan. Tugas generator adalah belajar mensintesis data palsu yang dapat menipu model diskriminatif. Model generatif yang dilatih menggunakan GAN dapat belajar mensintesis gambar palsu yang meniru gaya artistik (Elgammal et al. 2017), dan juga untuk mensintesis gambar medis bersama dengan anotasi lesi (Frid-Adar et al. 2018). Belajar untuk

mensintesis citra medis, bersama dengan segmentasi lesi pada citra yang disintesis, membuka kemungkinan secara otomatis menghasilkan set data berlabel besar yang dapat digunakan.untuk pembelajaran yang diawasi. Aplikasi GAN yang lebih mengkhawatirkan adalah penggunaan jaringan ini untuk menghasilkan *pemalsuan yang mendalam* : *pemalsuan* yang mendalam adalah video palsu dari seseorang yang melakukan sesuatu yang tidak pernah mereka lakukan yang dibuat dengan menukar wajah mereka menjadi video orang lain. Pemalsuan mendalam sangat sulit dideteksi, dan telah digunakan dengan niat jahat dalam beberapa kesempatan untuk mempermalukan tokoh masyarakat, atau untuk menyebarkan berita palsu.

Solusi lain untuk kemacetan pelabelan data adalah bahwa daripada melatih model baru dari awal untuk setiap aplikasi baru, kami lebih memilih menggunakan ulang model yang telah dilatih pada tugas serupa. Pembelajaran transfer adalah tantangan pembelajaran mesin dalam menggunakan informasi (atau representasi) yang dipelajari pada satu tugas untuk membantu pembelajaran pada tugas lain. Agar pembelajaran transfer berfungsi, dua tugas harus dari domain terkait. Pemrosesan gambar adalah contoh dari domain di mana pembelajaran transfer sering digunakan untuk mempercepat pelatihan model di berbagai tugas. Pembelajaran transfer sesuai untuk tugas pemrosesan gambar karena fitur visual level rendah, seperti edge, relatif stabil dan berguna di hampir semua kategori visual. Selain itu, fakta bahwa model CNN mempelajari hierarki fitur visual, dengan lapisan awal dalam fungsi pembelajaran CNN yang mendeteksi fitur visual tingkat rendah ini dalam masukan, memungkinkan untuk menggunakan kembali lapisan awal CNN yang telah dilatih sebelumnya di beberapa proyek pemrosesan gambar. Misalnya, bayangkan skenario di mana proyek membutuhkan model klasifikasi gambar yang dapat mengidentifikasi objek dari spesialisasikategori yang tidak ada sampelnya dalam kumpulan data gambar umum, seperti ImageNet. Daripada melatih model CNN baru dari awal, sekarang relatif standar untuk terlebih dahulu mengunduh model mutakhir (seperti model Microsoft ResNet) yang telah dilatih di ImageNet, kemudian mengganti lapisan model yang lebih baru dengan sekumpulan lapisan baru, dan terakhir untuk melatih model hibrid baru ini pada kumpulan data yang relatif kecil yang telah diberi label dengan kategori yang sesuai untuk proyek tersebut. Lapisan selanjutnya dari model state-of-the-art (umum) diganti karena lapisan ini berisi fungsi yang menggabungkan fitur tingkat rendah ke dalam kategori tugas tertentu yang awalnya dilatih untuk diidentifikasi oleh model.

Meningkatnya minat dalam pembelajaran tanpa pengawasan, model generatif, dan pembelajaran transfer semuanya dapat dipahami sebagai respons terhadap tantangan membuat anotasi set data yang semakin besar.

Munculnya Model Baru

Tingkat kemunculan model pembelajaran dalam yang baru semakin cepat setiap tahun. Contoh terbaru adalah jaringan kapsul(Hinton et al.2018; Sabour et

al.2017). Jaringan kapsul dirancang untuk mengatasi beberapa keterbatasan CNN. Satu masalah dengan CNN, kadang-kadang dikenal sebagai masalah Picasso, adalah kenyataan bahwa CNN mengabaikan hubungan spasial yang tepat antara komponen tingkat tinggi dalam struktur objek. Artinya dalam praktiknya adalah bahwa CNN yang telah dilatih untuk mengidentifikasi wajah dapat belajar mengidentifikasi bentuk mata, hidung, dan mulut, tetapi tidak akan mempelajari hubungan spasial yang diperlukan antara bagian-bagian ini. Akibatnya, jaringan dapat tertipu dengan gambar yang berisi bagian tubuh ini, meskipun posisinya tidak relatif benar satu sama lain. Masalah ini muncul karena lapisan penggabungan di CNN yang membuang informasi posisi.

Inti dari jaringan kapsul adalah intuisi yang dipelajari otak manusia untuk mengidentifikasi jenis objek dalam sudut pandang yang tidak berubah. Intinya, untuk setiap tipe objek terdapat kelas objek yang memiliki sejumlah parameter instantiation. Kelas objek mengkodekan informasi seperti hubungan relatif dari bagian objek yang berbeda satu sama lain. Parameter instantiation mengontrol bagaimana deskripsi abstrak dari sebuah tipe objek dapat dipetakan ke instance spesifik dari objek yang sedang dilihat (misalnya, pose, skala, dll.).

Kapsul adalah sekumpulan neuron yang belajar untuk mengidentifikasi apakah jenis objek atau bagian objek tertentu ada di lokasi tertentu dalam gambar. Sebuah kapsul mengeluarkan filevektor aktivitas yang merepresentasikan parameter instantiation dari instance objek, jika ada di lokasi yang relevan. Kapsul tertanam di dalam lapisan konvolusional. Namun, jaringan kapsul menggantikan proses penggabungan, yang sering kali mendefinisikan antarmuka antara lapisan konvolusional, dengan proses yang disebut perutean dinamis. Ide di balik perutean dinamis adalah bahwa setiap kapsul dalam satu lapisan di jaringan belajar memprediksi kapsul mana di lapisan berikutnya yang merupakan kapsul paling relevan untuk meneruskan vektor keluarannya.

Pada saat atau saat penulisan, jaringan kapsul memiliki performa mutakhir pada set data pengenalan digit tulisan tangan MNIST tempat CNN asli dilatih. Namun, menurut standar saat ini, ini adalah kumpulan data yang relatif kecil, dan jaringan kapsul belum diskalakan ke kumpulan data yang lebih besar. Ini sebagian karena proses perutean dinamis memperlambat pelatihan jaringan kapsul. Namun, jika jaringan kapsul berhasil diskalakan, jaringan tersebut dapat memperkenalkan bentuk model baru yang penting yang memperluas kemampuan jaringan saraf untuk menganalisis gambar dengan cara yang lebih mendekati cara manusia.

Model terbaru lainnya yang telah menarik banyak perhatian adalah model transformator (Vaswani et al. 2017). Model transformator adalah contoh tren yang berkembang dalam pembelajaran mendalam di mana model dirancang untuk memiliki mekanisme perhatian internal yang canggih yang memungkinkan model untuk secara dinamis memilih subset dari input untuk difokuskanaktif saat menghasilkan keluaran. Model transformator telah mencapai kinerja mutakhir pada terjemahan mesin untuk beberapa pasangan bahasa, dan di masa depan arsitektur ini dapat menggantikan arsitektur encoder-decoder yang dijelaskan dalam bab 5. Model BERT (Representasi Encoder Bidirectional dari Transformers)

memiliki dibangun di atas arsitektur Transformer (Devlin et al. 2018). Pengembangan BERT sangat menarik karena pada intinya adalah gagasan pembelajaran transfer (seperti yang dibahas di atas dalam kaitannya dengan hambatan anotasi data). Pendekatan dasar untuk membuat model pemrosesan bahasa alami dengan BERT adalah untuk melatih model untuk bahasa tertentu menggunakan dataset besar yang tidak berlabel (fakta bahwa dataset tidak berlabel berarti relatif murah untuk dibuat). Model yang dilatih sebelumnya ini kemudian dapat digunakan sebagai dasar untuk membuat model untuk tugas-tugas tertentu untuk bahasa tersebut (seperti klasifikasi sentimen atau menjawab pertanyaan) dengan menyempurnakan model yang dilatih sebelumnya menggunakan pembelajaran yang diawasi dan kumpulan data beranotasi yang relatif kecil. Keberhasilan BERT telah menunjukkan pendekatan ini dapat ditelusuri dan efektif dalam mengembangkan sistem pemrosesan bahasa alami yang canggih.

Bentuk Perangkat Keras Baru

Pembelajaran mendalam hari ini didukung oleh unit pemrosesan grafis (GPU): perangkat keras khusus yang dioptimalkan untuk melakukan perkalian matriks secara cepat. Adopsi, pada akhir 2000-an, GPU komoditas untuk mempercepat pelatihan jaringan saraf merupakan faktor kunci dalam banyak terobosan yang membangun momentum di balik pembelajaran mendalam. Dalam sepuluh tahun terakhir, produsen perangkat keras telah menyadari pentingnya pasar pembelajaran mendalam dan telah mengembangkan serta merilis perangkat keras yang dirancang khusus untuk pembelajaran mendalam, dan yang mendukung pustaka pembelajaran mendalam, seperti TensorFlow dan PyTorch. Karena kumpulan data dan jaringan terus bertambah besar, permintaan akan perangkat keras yang lebih cepat terus berlanjut. Namun, pada saat yang sama, ada pengakuan yang semakin besar atas biaya energi yang terkait dengan pembelajaran mendalam, dan orang-orang mulai mencari solusi perangkat keras yang memiliki jejak energi yang berkurang.

Komputasi neuromorfik muncul pada akhir 1980-an dari karya Carver Mead.[2] Sebuah chip neuromorfik terdiri dari sirkuit terintegrasi skala sangat besar (VLSI), yang menghubungkan jutaan unit daya rendah yang berpotensi yang dikenal sebagai neuron spiking. Dibandingkan dengan neuron buatan yang digunakan dalam sistem pembelajaran mendalam standar, desain neuron spiking lebih dekat dengan perilaku neuron biologis. Secara khusus, neuron spiking tidak menyala sebagai respons terhadap rangkaian aktivasi input yang disebarkan padanya pada titik waktu tertentu. Sebaliknya, neuron spiking mempertahankan keadaan internal (atau potensi aktivasi) yang berubah seiring waktu saat menerima pulsa aktivasi. Potensi aktivasi meningkat ketika aktivasi baru diterima,dan membusuk seiring waktu tanpa adanya aktivasi yang masuk. Neuron menyala ketika potensi aktivasi melebihi ambang tertentu. Karena peluruhan temporal dari potensi aktivasi neuron,

neuron spiking hanya aktif jika menerima jumlah aktivasi input yang diperlukan dalam jendela waktu (pola spiking). Satu keuntungan dari pemrosesan berbasis temporal ini adalah bahwa spiking neuron tidak menyala pada setiap siklus propagasi, dan ini mengurangi jumlah energi yang dikonsumsi jaringan.

Dibandingkan dengan desain CPU tradisional, chip neuromorfik memiliki sejumlah karakteristik berbeda, termasuk:

1. Blok penyusun dasar: CPU tradisional dibangun menggunakan gerbang logika berbasis transistor (mis., Gerbang AND, OR, NAND), sedangkan chip neuromorfik dibuat menggunakan neuron spiking.
2. Chip neuromorfik memiliki aspek analog: dalam komputer digital tradisional, informasi dikirim dalam semburan listrik tinggi-rendah yang sinkron dengan jam pusat; dalam chip neuromorfik, informasi dikirim sebagai pola sinyal tinggi-rendah yang bervariasi sepanjang waktu.
3. Arsitektur: arsitektur CPU tradisional didasarkan pada arsitektur von Neumann, yang secara intrinsik terpusat dengan semua informasi yang melewati CPU. Chip neuromorfik dirancang untukmemungkinkan paralelisme masif arus informasi antara neuron spiking. Neuron spiking berkomunikasi langsung satu sama lain daripada melalui pusat pemrosesan informasi.
4. Representasi informasi didistribusikan melalui waktu: sinyal informasi disebarkan melalui chip neuromorfik menggunakan representasi terdistribusi, mirip dengan representasi terdistribusi yang dibahas dalam bab 4, dengan perbedaan bahwa dalam chip neuromorfik representasi ini juga didistribusikan melalui waktu. Representasi terdistribusi lebih tahan terhadap kehilangan informasi daripada representasi lokal, dan ini adalah properti yang berguna saat meneruskan informasi antara ratusan ribu, atau jutaan, komponen, beberapa di antaranya kemungkinan besar akan gagal.

Saat ini ada sejumlah proyek penelitian besar yang berfokus pada komputasi neuromorfik. Misalnya, pada 2013 Komisi Eropa mengalokasikan satu miliar euro dalam pendanaan untuk Proyek Otak Manusia selama sepuluh tahun.[3] Proyek ini secara langsung mempekerjakan lebih dari lima ratus ilmuwan, dan melibatkan penelitian dari lebih dari seratus pusat penelitian di seluruh Eropa. Salah satu tujuan utama proyek adalah pengembangan platform komputasi neuromorfik yang mampu menjalankan simulasi otak manusia yang lengkap. Sejumlah neuromorfik komersialchip juga telah dikembangkan. Pada tahun 2014, IBM meluncurkan chip TrueNorth, yang berisi lebih dari satu juta neuron yang dihubungkan bersama oleh lebih dari 286 juta sinapsis. Chip ini menggunakan sekitar 1/10.000 kekuatan mikroprosesor konvensional. Pada tahun 2018, Intel Labs mengumumkan chip neuromorfik Loihi (diucapkan *low-ee-hee*). Chip Loihi memiliki 131.072 neuron yang dihubungkan bersama oleh 130.000.000 sinapsis. Komputasi neuromorfik

memiliki potensi untuk merevolusi pembelajaran mendalam; Namun, masih menghadapi sejumlah tantangan, tidak terkecuali tantangan untuk mengembangkan algoritma dan pola perangkat lunak untuk memprogram perangkat keras paralel skala besar ini.

Akhirnya, dalam jangka waktu yang sedikit lebih lama, komputasi kuantum adalah aliran lain dari penelitian perangkat keras yang berpotensi merevolusi pembelajaran mendalam. Chip komputasi kuantum sudah ada; misalnya, Intel telah membuat chip uji kuantum 49 qubit, dengan kode bernama Tangle Lake. Qubit adalah padanan kuantum dari digit biner (bit) dalam komputasi tradisional. Sebuah qubit dapat menyimpan lebih dari satu bit informasi; namun, diperkirakan akan membutuhkan sistem dengan satu juta qubit atau lebih sebelum komputasi kuantum berguna untuk tujuan komersial. Perkiraan waktu saat ini untuk menskalakan chip kuantum ke level ini adalah sekitar tujuh tahun.

Tantangan Interpretabilitas

Pembelajaran mesin, dan pembelajaran mendalam, pada dasarnya adalah tentang membuat keputusan berdasarkan data. Meskipun pembelajaran mendalam menyediakan sekumpulan algoritme dan teknik yang kuat untuk melatih model yang dapat bersaing (dan dalam beberapa kasus mengungguli) manusia dalam berbagai tugas pengambilan keputusan, ada banyak situasi di mana keputusan itu sendiri tidak cukup. Seringkali, tidak hanya perlu memberikan keputusan tetapi juga alasan di balik suatu keputusan. Ini terutama benar ketika keputusan memengaruhi seseorang, baik itu diagnosis medis atau penilaian kredit. Kekhawatiran ini tercermin dalam peraturan privasi dan etika terkait dengan penggunaan data pribadi dan pengambilan keputusan algoritmik yang berkaitan dengan individu. Misalnya, Resital 71[4] Peraturan Perlindungan Data Umum (GDPR) menyatakan bahwa individu, yang terpengaruh oleh keputusan yang dibuat oleh proses pengambilan keputusan otomatis, memiliki hak untuk mendapatkan penjelasan terkait dengan bagaimana keputusan itu diambil.

Model pembelajaran mesin yang berbeda memberikan tingkat interpretabilitas yang berbeda sehubungan dengan cara mereka mencapai keputusan tertentu. Model pembelajaran mendalam, bagaimanapun, mungkin yang paling tidak dapat ditafsirkan. Pada satu tingkat deskripsi, model pembelajaran mendalam cukup sederhana: ia terdiri dari unit pemrosesan sederhana (neuron) yang terhubung bersama ke dalam jaringan. Namun, skala jaringan (dalam hal jumlah neuron dan koneksinyadi antara mereka), sifat representasi yang terdistribusi, dan transformasi berturut-turut dari data masukan saat informasi mengalir lebih dalam ke jaringan, membuatnya sangat sulit untuk ditafsirkan, dipahami, dan oleh karena itu menjelaskan, bagaimana jaringan menggunakan masukan untuk membuat sebuah keputusan.

Status hukum hak atas penjelasan dalam GDPR saat ini tidak jelas, dan implikasi spesifiknya terhadap pembelajaran mesin dan pembelajaran mendalam perlu diselesaikan di pengadilan. Namun, contoh ini menyoroti kebutuhan masyarakat akan pemahaman yang lebih baik tentang seberapa dalam model pembelajaran menggunakan data. Kemampuan untuk menafsirkan dan memahami cara kerja model pembelajaran yang dalam juga penting dari perspektif teknis. Misalnya, memahami bagaimana suatu model menggunakan data dapat mengungkapkan jika suatu model memiliki bias yang tidak diinginkan dalam cara mengambil keputusannya, dan juga mengungkapkan kasus-kasus sudut di mana model akan gagal. Pembelajaran mendalam dan komunitas penelitian kecerdasan buatan yang lebih luas sudah menanggapi tantangan ini. Saat ini,

Chis Olah dan rekan-rekannya merangkum teknik utama yang saat ini digunakan untuk memeriksa cara kerja model deep learning sebagai: visualisasi fitur, atribusi, dan reduksi dimensi (Olah et al. 2018). SatuCara untuk memahami bagaimana jaringan memproses informasi adalah dengan memahami input apa yang memicu perilaku tertentu dalam jaringan, seperti penembakan neuron. Memahami input spesifik yang memicu aktivasi neuron memungkinkan kita memahami apa yang dipelajari neuron untuk dideteksi dalam input tersebut. Tujuan dari visualisasi fitur adalah untuk menghasilkan dan memvisualisasikan input yang menyebabkan aktivitas tertentu dalam jaringan. Ternyata teknik pengoptimalan, seperti backpropogation, dapat digunakan untuk menghasilkan input ini. Prosesnya dimulai dengan masukan yang dihasilkan secara acak dan masukan tersebut kemudian diperbarui secara berulang hingga perilaku target dipicu. Setelah masukan yang diperlukan telah diisolasi, itu kemudian dapat divisualisasikan untuk memberikan pemahaman yang lebih baik tentang apa yang dideteksi jaringan dalam input ketika merespons dengan cara tertentu. Atribusi berfokus pada penjelasan hubungan antar neuron, misalnya, bagaimana keluaran neuron dalam satu lapisan jaringan berkontribusi pada keluaran keseluruhan jaringan. Hal ini dapat dilakukan dengan menghasilkan saliency (atau heat-map) untuk neuron dalam jaringan yang menangkap berapa banyak bobot yang diberikan jaringan pada keluaran neuron saat membuat keputusan tertentu. Akhirnya, banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi lebih besar Atribusi berfokus pada penjelasan hubungan antar neuron, misalnya, bagaimana keluaran neuron dalam satu lapisan jaringan berkontribusi pada keluaran keseluruhan jaringan. Ini dapat dilakukan dengan menghasilkan saliency (atau heat-map) untuk neuron dalam jaringan yang menangkap berapa banyak bobot yang diberikan jaringan pada keluaran neuron saat membuat keputusan tertentu. Akhirnya, banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi

lebih besar Atribusi berfokus pada penjelasan hubungan antar neuron, misalnya, bagaimana keluaran neuron dalam satu lapisan jaringan berkontribusi pada keluaran keseluruhan jaringan. Hal ini dapat dilakukan dengan menghasilkan saliency (atau heat-map) untuk neuron dalam jaringan yang menangkap berapa banyak bobot yang diberikan jaringan pada keluaran neuron saat membuat keputusan tertentu. Akhirnya, banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi lebih besar bagaimana output dari neuron dalam satu lapisan jaringan berkontribusi pada output jaringan secara keseluruhan. Ini dapat dilakukan dengan menghasilkan saliency (atau heat-map) untuk neuron dalam jaringan yang menangkap berapa banyak bobot yang diberikan jaringan pada keluaran neuron saat membuat keputusan tertentu. Akhirnya, banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi lebih besar bagaimana output dari neuron dalam satu lapisan jaringan berkontribusi pada output jaringan secara keseluruhan. Ini dapat dilakukan dengan menghasilkan saliency (atau heat-map) untuk neuron dalam jaringan yang menangkap berapa banyak bobot yang diberikan jaringan pada keluaran neuron saat membuat keputusan tertentu. Akhirnya, banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi lebih besar banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi lebih besar banyak aktivitas dalam jaringan pembelajaran yang dalam didasarkan pada pemrosesan vektor berdimensi tinggi. Memvisualisasikan data memungkinkan kita menggunakan korteks visual yang kuat untuk menafsirkan data dan hubungan di dalam data. Akan tetapi, sangat sulit untuk memvisualisasikan data yang memiliki dimensi lebih besardari tiga. Akibatnya, teknik visualisasi yang mampu secara sistematis mengurangi dimensi data berdimensi tinggi dan memvisualisasikan hasil adalah alat yang sangat berguna untuk menafsirkan aliran informasi dalam jaringan yang dalam. t-SNE[5] adalah teknik terkenal yang memvisualisasikan data dimensi tinggi dengan memproyeksikan setiap titik data ke dalam peta dua atau tiga dimensi (van der Maaten dan Hinton 2008). Penelitian tentang menafsirkan jaringan pembelajaran yang dalam masih dalam tahap awal, tetapi di tahun-tahun mendatang, baik untuk alasan sosial dan teknis, penelitian ini cenderung menjadi

perhatian yang lebih sentral bagi komunitas pembelajaran mendalam yang lebih luas.

Pikiran Akhir

Pembelajaran mendalam sangat cocok untuk aplikasi yang melibatkan kumpulan data besar dari data dimensi tinggi. Akibatnya, pembelajaran mendalam kemungkinan akan memberikan kontribusi yang signifikan terhadap beberapa tantangan ilmiah utama di zaman kita. Dalam dua dekade terakhir, terobosan dalam teknologi sekuensing biologis telah memungkinkan untuk menghasilkan sekuens DNA presisi tinggi. Data genetik ini berpotensi menjadi fondasi untuk generasi berikutnya dari pengobatan presisi yang dipersonalisasi. Pada saat yang sama, proyek penelitian internasional, seperti Large Hadron Collider dan teleskop orbit Bumi, menghasilkan jumlah yang sangat besardata setiap hari. Menganalisis data ini dapat membantu kita memahami fisika alam semesta kita pada skala terkecil dan terbesar. Menanggapi banjir data ini, para ilmuwan, yang jumlahnya terus meningkat, beralih ke pembelajaran mesin dan pembelajaran mendalam untuk memungkinkan mereka menganalisis data ini.

Salah satu cara untuk memahami bagaimana jaringan memproses informasi adalah dengan memahami input apa yang memicu perilaku tertentu dalam jaringan, seperti penembakan neuron.

Namun, pada tingkat yang lebih duniawi, pembelajaran mendalam sudah memengaruhi kehidupan kita secara langsung. Kemungkinan, selama beberapa tahun terakhir, Anda tanpa sadar telah menggunakan model pembelajaran mendalam setiap hari. Model pembelajaran mendalam mungkin dipanggil setiap kali Anda menggunakan mesin pencari internet, sistem terjemahan mesin, sistem pengenalan wajah di kamera atau situs media sosial, atau menggunakan antarmuka ucapan ke perangkat pintar. Apa yang berpotensi lebih mengkhawatirkan adalah jejak data dan metadata yang Anda tinggalkan saat Anda bergerak di dunia online juga sedang diproses dan dianalisis menggunakan model pembelajaran yang mendalam. Inilah sebabnya mengapa sangat penting untuk memahami apa itu pembelajaran mendalam, bagaimana cara kerjanya, apa kemampuannya, dan keterbatasannya saat ini.

Glosarium

ungsi Aktivasi

ebuah fungsi yang menerima masukan hasil penjumlahan bobot masukan ke neuron dan menerapkan pemetaan nonlinier ke penjumlahan pembobotan ini. Termasuk fungsi aktivasi dalam neuron jaringan memungkinkan jaringan untuk mempelajari pemetaan nonlinier. Contoh fungsi aktivasi yang umum digunakan meliputi: logistik, tanh, dan ULT.

ecerdasan buatan

idang penelitian yang difokuskan pada pengembangan sistem komputasi yang dapat melakukan tugas dan aktivitas yang biasanya dianggap membutuhkan kecerdasan manusia.

ropagasi mundur

ackpropagation adalah algoritma yang digunakan untuk melatih jaringan saraf dengan lapisan neuron tersembunyi. Selama pelatihan, bobot dalam jaringan diperbarui secara berulang untuk mengurangi kesalahan jaringan. Untuk memperbarui bobot pada tautan yang masuk ke neuron tertentu dalam jaringan, pertama-tama perlu dihitung perkiraan kontribusi keluaran neuron tersebut terhadap keseluruhan kesalahan jaringan. Algoritma propagasi mundur adalah solusi untuk menghitung perkiraan ini untuk setiap neuron di jaringan. Setelah perkiraan kesalahan ini dihitung untuk setiap neuron, bobot neuron dapat diperbarui menggunakan algoritme pengoptimalan seperti penurunan gradien. Propagasi mundur bekerja dalam dua fase: umpan maju dan umpan mundur. Di operan depan, sebuah contoh disajikan ke jaringan dan kesalahan keseluruhan jaringan dihitung pada lapisan keluaran jaringan dengan membandingkan keluaran jaringan dengan keluaran yang diharapkan untuk contoh yang ditentukan dalam dataset. Dalam backward pass, kesalahan jaringan dibagi kembali melalui jaringan dengan setiap neuron menerima sebagian dari kesalahan tersebut sebanding dengan sensitivitas kesalahan terhadap perubahan dalam output neuron tersebut. Proses berbagi kembali kesalahan melalui jaringan dikenal sebagai propagasi mundur kesalahan dan di sinilah algoritma mendapatkan namanya. kesalahan jaringan dibagi kembali melalui jaringan dengan masing-masing neuron menerima sebagian kesalahan untuk kesalahan sebanding dengan sensitivitas kesalahan terhadap perubahan dalam output neuron tersebut. Proses berbagi kembali kesalahan melalui jaringan dikenal sebagai propagasi mundur kesalahan dan di sinilah algoritma mendapatkan namanya. kesalahan jaringan dibagi kembali melalui jaringan dengan masing-masing neuron menerima sebagian dari kesalahan tersebut sebanding dengan sensitivitas kesalahan terhadap perubahan dalam keluaran neuron tersebut. Proses berbagi kembali kesalahan melalui jaringan dikenal sebagai propagasi mundur kesalahan dan di sinilah algoritma mendapatkan namanya.

aringan Neural Konvolusional

aringan neural konvolusional adalah jaringan yang memiliki setidaknya satu lapisan konvolusional di dalamnya. Lapisan konvolusi terdiri dari satu set neuron yang berbagi set bobot yang sama dan yang gabungan bidang reseptifnya mencakup seluruh input. Penyatuan keluaran dari satu set neuron dikenal sebagai peta fitur. Di banyak jaringan neural konvolusional, peta fitur dilewatkan melalui lapisan aktivasi ULT dan kemudian lapisan penggabungan.

limpunan data

umpulan contoh dengan setiap contoh yang dijelaskan dalam istilah sekumpulan fitur. Dalam bentuknya yang paling dasar, sebuah dataset disusun dalam matriks $n \times m$, di mana n adalah jumlah instance (baris) dan m adalah jumlah fitur (kolom).

embelajaran Mendalam

embelajaran mendalam adalah subbidang pembelajaran mesin yang merancang dan mengevaluasi algoritme dan arsitektur pelatihan untuk model jaringan neural modern. Jaringan saraf dalam adalah jaringan yang memiliki beberapa (misalnya,> 2) lapisan unit tersembunyi (atau neuron).

aringan Feedforward

aringan umpan maju adalah jaringan saraf di mana semua koneksi di jaringan mengarah ke depan ke neuron di lapisan berikutnya. Dengan kata lain, tidak ada tautan mundur dari keluaran neuron ke masukan neuron di lapisan sebelumnya.

ungsi

ungsi adalah pemetaan deterministik dari sekumpulan nilai masukan ke satu atau lebih nilai keluaran. Dalam konteks pembelajaran mesin, istilah fungsi sering digunakan secara bergantian dengan istilah model.

enurunan Gradien

radient descent adalah algoritma pengoptimalan untuk menemukan fungsi dengan kesalahan minimum sehubungan dengan pemodelan pola dalam sebuah dataset. Dalam konteks melatih jaringan saraf, penurunan gradien digunakan untuk menemukan himpunan bobot untuk neuron yang meminimalkan kesalahan keluaran neuron. Gradien yang diturunkan algoritme adalah gradien kesalahan neuron saat bobotnya diperbarui. Algoritme ini sering digunakan bersama dengan propagasi mundur untuk melatih jaringan saraf dengan lapisan neuron tersembunyi.

PU (Unit Pemrosesan Grafis)

erangkat keras khusus yang dioptimalkan untuk perkalian matriks cepat. Awalnya dirancang untuk meningkatkan kecepatan rendering grafis, tetapi juga terbukti

mempercepat pelatihan jaringan saraf.

STM (Memori Jangka Pendek Panjang)

aringan yang dirancang untuk mengatasi masalah hilangnya gradien di jaringan saraf berulang. Jaringan terdiri dari blok sel tempat aktivasi mengalir dari satu langkah waktu ke langkah berikutnya dan satu set gerbang pada blok sel yang mengontrol aliran aktivasi ini. Gerbang diimplementasikan menggunakan lapisan fungsi aktivasi sigmoid dan tanh. Arsitektur LSTM standar memiliki tiga gerbang: gerbang lupa, gerbang pembaruan, dan gerbang keluaran.

embelajaran Mesin (ML)

idang penelitian ilmu komputer yang berfokus pada pengembangan dan evaluasi algoritma yang memungkinkan komputer untuk belajar dari pengalaman. Umumnya konsep pengalaman direpresentasikan sebagai kumpulan data peristiwa bersejarah, dan pembelajaran melibatkan identifikasi dan penggalian pola yang berguna dari kumpulan data. Algoritme pembelajaran mesin mengambil kumpulan data sebagai masukan dan mengembalikan model yang menyandikan pola yang diekstrak (atau dipelajari) oleh algoritme dari data.

lgoritma Pembelajaran Mesin

roses yang menganalisis set data dan mengembalikan sebagai model (yaitu, instansiasi fungsi sebagai program komputer) yang cocok dengan pola dalam data.

lodel

alam pembelajaran mesin, model adalah program komputer yang menyandikan pola yang diekstrak oleh algoritma pembelajaran mesin dari kumpulan data. Ada banyak jenis model pembelajaran mesin; Namun, pembelajaran mendalam difokuskan pada pembuatan model jaringan saraf dengan banyak lapisan neuron tersembunyi. Model dibuat (atau dilatih) dengan menjalankan algoritme pembelajaran mesin pada kumpulan data. Setelah model dilatih, model tersebut kemudian dapat digunakan untuk menganalisis instance baru; istilah inferensi terkadang digunakan untuk mendeskripsikan proses analisis instance baru menggunakan model terlatih. Dalam konteks pembelajaran mesin, istilah model dan fungsi sering digunakan secara bergantian: model adalah instansiasi dari suatu fungsi sebagai program komputer.

omputasi Neuromorfik

hip neuromorfik terdiri dari set arsitektur neuron spiking yang sangat besar yang terhubung secara paralel secara masif.

aringan syaraf

Model pembelajaran mesin yang diimplementasikan sebagai jaringan unit pemrosesan informasi sederhana yang disebut neuron. Dimungkinkan untuk membuat berbagai jenis jaringan saraf dengan memodifikasi koneksi antara neuron dalam jaringan. Contoh jenis jaringan neural yang populer meliputi: feedforward, convolutional, dan jaringan berulang.

Neuron

Dalam konteks pembelajaran yang mendalam (sebagai lawan dari ilmu otak), neuron adalah algoritma pemrosesan informasi sederhana yang mengambil sejumlah nilai numerik sebagai masukan dan memetakan nilai-nilai ini ke aktivasi keluaran tinggi atau rendah. Pemetaan ini biasanya diterapkan dengan terlebih dahulu mengalikan setiap nilai masukan dengan bobot, lalu menjumlahkan hasil perkalian ini, dan terakhir meneruskan hasil penjumlahan berbobot melalui fungsi aktivasi.

Overfitting

Overfitting set data terjadi jika model yang dikembalikan oleh algoritme pembelajaran mesin sangat kompleks sehingga dapat memodelkan variasi kecil dalam data yang disebabkan oleh noise dalam sampel data.

Jaringan Neural Berulang

Jaringan saraf berulang memiliki satu lapisan neuron tersembunyi, yang keluarannya diumpankan kembali ke lapisan ini dengan masukan berikutnya. Umpan balik ini (atau pengulangan) dalam jaringan memberi jaringan memori yang memungkinkannya memproses setiap masukan dalam konteks apa yang telah diproses sebelumnya. Jaringan neural berulang cocok untuk memproses data sekuensial atau deret waktu.

Pembelajaran Penguatan

Tujuan dari pembelajaran penguatan adalah untuk memungkinkan agen mempelajari kebijakan tentang bagaimana seharusnya bertindak dalam lingkungan tertentu. Kebijakan adalah fungsi yang memetakan dari pengamatan terkini agen terhadap lingkungannya dan keadaan internalnya sendiri ke suatu tindakan. Biasanya digunakan untuk tugas kontrol online seperti kontrol robot dan bermain game.

Unit ULT

Unit ULT adalah neuron yang menggunakan fungsi linier yang diperbaiki sebagai fungsi aktivasinya.

Pembelajaran yang Diawasi

Suatu bentuk pembelajaran mesin yang tujuannya adalah mempelajari fungsi yang memetakan dari sekumpulan atribut input untuk sebuah instance ke perkiraan

akurat dari nilai yang hilang untuk atribut target dari instance yang sama.

tribut Target

alam pembelajaran mesin yang diawasi, atribut target adalah atribut tempat model dilatih untuk memperkirakan nilainya.

urang pas

ataset yang kurang pas terjadi jika model yang dikembalikan oleh algoritme pembelajaran mesin terlalu sederhana untuk menangkap kompleksitas nyata dari hubungan antara masukan dan keluaran dalam suatu domain.

embelajaran Tanpa Pengawasan

uatu bentuk pembelajaran mesin yang tujuannya adalah untuk mengidentifikasi keteraturan, seperti kelompok contoh serupa, dalam data. Tidak seperti pembelajaran yang diawasi, tidak ada atribut target dalam tugas pembelajaran tanpa pengawasan.

radien Menghilang

asalah gradien menghilang menjelaskan fakta bahwa semakin banyak lapisan yang ditambahkan ke jaringan, dibutuhkan waktu lebih lama untuk melatih jaringan. Masalah ini disebabkan oleh fakta bahwa ketika jaringan saraf dilatih menggunakan propagasi mundur dan penurunan gradien, pembaruan bobot pada tautan yang masuk ke neuron dalam jaringan bergantung pada gradien (atau sensitivitas) kesalahan jaringan sehubungan dengan keluaran neuron. Menggunakan propagasi mundur, proses membagikan kembali gradien kesalahan melalui neuron melibatkan urutan perkalian, seringkali dengan nilai kurang dari satu. Akibatnya, saat gradien kesalahan diteruskan kembali melalui jaringan, gradien kesalahan cenderung semakin kecil (mis., Menghilang). Sebagai konsekuensi langsung dari ini,

Catatan

Bab 1

. https://deepmind.com/research/alphago/.

. Sistem peringkat Elo adalah metode untuk menghitung tingkat keahlian pemain dalam permainan zero-sum, seperti Catur. Itu dinamai penemunya, Arpad Elo.

. Kebisingan dalam data mengacu pada data yang rusak atau tidak benar. Kebisingan pada data dapat disebabkan oleh sensor yang rusak, atau kesalahan dalam pemasukan data, dan lain sebagainya.

. Yang kami maksud dengan domain adalah masalah atau tugas yang kami coba selesaikan menggunakan pembelajaran mesin. Misalnya, dapat berupa pemfilteran spam, prediksi harga rumah, atau secara otomatis mengklasifikasikan sinar-X.

. Ada beberapa skenario yang memerlukan representasi set data yang lebih kompleks. Misalnya, untuk data deret waktu, set data mungkin memerlukan representasi tiga dimensi, yang terdiri dari rangkaian matriks dua dimensi, masing-masing menggambarkan keadaan sistem pada suatu titik waktu, dihubungkan bersama sepanjang waktu. Istilah *tensor* menggeneralisasi konsep *matriks* ke dimensi yang lebih tinggi.

Bab 2

. Ternyata hubungan antara pendapatan tahunan dan kebahagiaan adalah linier sampai titik tertentu, tetapi begitu pendapatan tahunan Anda melampaui titik ini, lebih banyak uang tidak akan membuat Anda lebih bahagia. Sebuah studi oleh Kahneman dan Deaton (2010) menemukan bahwa di AS pemutusan hubungan kerja secara umum, setelah peningkatan pendapatan tidak lagi meningkatkan kesejahteraan emosional, adalah sekitar $ 75.000.

. Ini adalah dataset yang sama yang muncul pada tabel 1.1 di bab 1; itu diulangi di sini untuk kenyamanan.

bagian 3

. Asal adalah lokasi dalam sistem koordinat tempat sumbu bersilangan. Dalam sistem koordinat dua dimensi, sumbu x dan sumbu y bersilangan — dengan kata lain letak pada koordinat x = 0, y = 0.

. Dalam bab 2, kami menggunakan pendekatan yang sama untuk menggabungkan parameter intersep dari model linier ke dalam bobot model.

. Untuk menyorot organisasi kolom ini, bobot telah diindeks kolom-baris, bukan baris-kolom.

. Untuk diskusi lebih lanjut tentang ukuran dan pertumbuhan jaringan, lihat halaman 23 dari Goodfellow et al. 2016.

Bab 4

. Gambar 3.6 dan 3.7 menunjukkan batas keputusan linier (garis lurus) dari neuron yang menggunakan fungsi aktivasi ambang batas.

. Ilustrasi penggunaan memori asosiatif untuk penyelesaian pola dan koreksi kesalahan ini terinspirasi dari contoh di bab 42 MacKay 2003.

. Sebagai contoh, tesis PhD Paul Werbos 1974 dikreditkan sebagai publikasi pertama yang menggambarkan penggunaan kesalahan propagasi mundur dalam pelatihan jaringan saraf tiruan (Werbos 1974).

. Arsitektur jaringan Hopfield, yang diperkenalkan pada awal bagian ini, juga menyertakan koneksi berulang (loop umpan balik antar neuron). Namun, desain arsitektur Hopfield sedemikian rupa sehingga jaringan Hopfield tidak dapat memproses urutan. Akibatnya, ini tidak dianggap sebagai arsitektur RNN penuh.

. Saya awalnya menemukan kutipan Churchland ini di Marcus 2003 (hlm. 25).

. Kritik atas makalah "Konspirasi Pembelajaran Mendalam" (*Nature* 521, p. 436), kritik diposting oleh Jürgen Schmidhuber, Juni 2015, tersedia di: http://people.idsia.ch/~juergen/deep-learning-conspiracy.html.

. Ada sejumlah cara lain yang dapat membatasi autoencoders untuk mencegah kemungkinan bahwa jaringan akan mempelajari pemetaan identitas yang tidak informatif dari masukan ke keluaran; misalnya, noise dapat dimasukkan ke dalam pola input dan jaringan dapat dilatih untuk merekonstruksi data yang tidak berisik. Atau, unit dalam lapisan tersembunyi (atau pengkodean) dapat dibatasi untuk memiliki nilai biner. Memang, Hinton dan rekan-rekannya awalnya menggunakan jaringan yang disebut Restricted Boltzman Machines (RBMs) dalam pekerjaan pra-pelatihan awal mereka, yang menggunakan unit biner di lapisan pengkodean.

. Jumlah lapisan yang dilatih selama pra-pelatihan adalah hyperparameter yang ditetapkan berdasarkan intuisi ilmuwan data dan eksperimen coba-dan-kesalahan.

. Pada awal tahun 1971, metode GMDH Alexey Ivakhnenko telah terbukti mampu melatih jaringan dalam (hingga delapan lapisan), tetapi metode ini sebagian besar telah diabaikan oleh komunitas peneliti.

0 . Inisialisasi Glorot juga dikenal sebagai inisialisasi Xavier. Kedua nama ini adalah referensi ke salah satu penulis (Xavier Glorot) makalah pertama yang memperkenalkan prosedur inisialisasi ini: Xavier Glorot dan Yoshua Bengio, "Memahami Kesulitan Pelatihan Jaringan Neural Deep Feedforward,"dalam *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics* (AISTATS), 2010, pp. 249-256.

1 . Inisialisasi glorot juga dapat didefinisikan sebagai pengambilan sampel bobot dari distribusi Gaussian dengan mean 0 dan deviasi standar ditetapkan ke akar kuadrat 2 dibagi dengan $n_j + n_{j+1}$. Namun, kedua definisi inisialisasi Glorot ini memiliki tujuan yang sama untuk memastikan varian yang sama dalam aktivasi dan gradien di seluruh lapisan dalam jaringan.

2 . https://developer.nvidia.com/cuda-zone.

Bab 5

. Penjelasan tentang unit LSTM yang disajikan di sini terinspirasi oleh entri blog yang sangat baik oleh Christopher Olah, yang menjelaskan LSTM dengan jelas dan rinci; posting tersedia di: http://colah.github.io/posts/2015–08-Understanding-LSTMs/.

. Fungsi sigmoid sebenarnya adalah kasus khusus dari fungsi logistik, dan untuk tujuan diskusi ini perbedaannya tidak relevan.

. Jika, misalnya, unit sigmoid dengan rentang keluaran 0 hingga 1 digunakan, aktivasi hanya dapat dipertahankan atau ditingkatkan pada setiap pembaruan dan pada akhirnya status sel akan dipenuhi dengan nilai maksimum.

Bab 6

. Gambar ini juga muncul di bab 4 tetapi diulangi di sini untuk memudahkan.

Bab 7

. http://www.image-net.org.

. https://en.wikipedia.org/wiki/Carver_Mead.

. https://www.humanbrainproject.eu/en/.

. Resital adalah bagian yang tidak mengikat secara hukum dari suatu peraturan yang berusaha untuk memperjelas arti dari teks hukum.

. Laurens van der Maaten dan Geoffrey Hinton, "Memvisualisasikan Data menggunakan t-SNE," *Jurnal Penelitian Pembelajaran Mesin* 9 (2008): 2579–2605.

Referensi

izenberg, IN, NN Aizenberg, dan J. Vandewalles. 2000. *Neuron Biner Multi-Nilai dan Universal: Teori, Pembelajaran dan Aplikasi* . Peloncat.

hellapilla, K., S. Puri, dan Patrice Simard. 2006. "Jaringan Neural Konvolusional Berkinerja Tinggi untuk Pemrosesan Dokumen." Dalam *Lokakarya Internasional Kesepuluh tentang Perbatasan dalam Pengenalan Tulisan Tangan.*

hurchland, PM 1996. *Mesin Nalar, Kursi Jiwa: Perjalanan Filosofis ke Otak.* MIT Press.

echter, R. 1986. "Belajar Sambil Mencari di Kendala-Kepuasan-Masalah." Dalam *Proceedings of the Fifth National Conference on Artificial Intelligence* (AAAI-86), hlm. 178–183.

evlin, J., MW Chang, K. Lee, dan K. Toutanova. 2018. “Bert: Pra-pelatihan transformator dua arah yang dalam untuk pemahaman bahasa.” arXiv pracetak arXiv: 1810.04805.

lgammal, A., B. Liu, M. Elhoseiny, dan M. Mazzone. 2017. “CAN: Creative Adversarial Networks, Menghasilkan 'Seni' dengan Mempelajari Gaya dan Menyimpang dari Norma Gaya.” arXiv: 1706.07068.

lman, JL 1990. “Menemukan Struktur dalam Waktu.” *Cogn. Sci.* 14: 179–211.

rid-Adar, M., I. Diamant, E. Klang, M. Amitai, J. Goldberger, dan H. Greenspan. 2018. "Augmentasi Gambar Medis Sintetis Berbasis GAN untuk Peningkatan Kinerja CNN dalam Klasifikasi Lesi Hati." arXiv: 1803.01229.

ukushima, K. 1980. “Neocognitron: Model jaringan saraf yang mengatur dirinya sendiri untuk mekanisme pengenalan pola yang tidak terpengaruh oleh pergeseran posisi.” *Biol. Cybern.* 36: 193–202.

lorot, X., dan Y. Bengio. 2010. “Memahami Kesulitan Melatih Jaringan Neural Deep Feedforward.” Dalam *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (AISTATS), pp. 249-256.

lorot, X., A. Bordes, dan Y. Bengio. 2011. "Jaringan Neural Penyearah Jarang Dalam". Dalam *Proceedings of the Fourteenth International Conference on*

Artificial Intelligence and Statistics (AISTATS), hlm. 315-323.

oodfellow, I., Y. Bengio, dan A. Courville. 2016. *Pembelajaran Mendalam.* MIT Press.

oodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, dan J. Bengio. 2014. "Jaring Permusuhan Generatif." Dalam *Kemajuan dalam Sistem Pemrosesan Informasi Neural* 27: 2672-2680.

ia, K., X. Zhang, S. Ren, dan J. Sun. 2016. "Pembelajaran Sisa Jauh untuk Pengenalan Gambar." Dalam *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR). IEEE, hlm. 770–778. https://doi.org/10.1109/CVPR.2016.90 .

ebb, DO 1949. *Organisasi Perilaku: Teori Neuropsikologis* . John Wiley & Sons.

erculano-Houzel, S. 2009. "Otak Manusia dalam Angka: Otak Primata yang Ditingkatkan Linier." *Depan. Bersenandung. Neurosci.* 3. https://doi.org/10.3389/neuro.09.031.2009.

inton, GE, S. Sabour, dan N. Frosst. 2018. "Kapsul Matriks dengan Perutean EM". Dalam *Prosiding Konferensi Internasional ke-7 tentang Representasi Pembelajaran* (ICLR).

ochreiter, S. 1991. Untersuchungen zu dynamischen neuronalen Netzen (Diploma). Technische Universität München.

ochreiter, S., Schmidhuber, J. 1997. "Memori Jangka Pendek yang Panjang." *Komputasi Neural.* 9: 1735–1780.

opfield, JJ 1982. "Jaringan Saraf dan Sistem Fisik dengan Kemampuan Komputasi Kolektif yang Muncul." *Proc. Natl. Acad. Sci.* 79: 2554–2558. https://doi.org/10.1073/pnas.79.8.2554.

ubel, DH, dan TN Wiesel. 1962. "Bidang Reseptif, Interaksi Teropong, dan Arsitektur Fungsional dalam Korteks Visual Kucing." *J. Physiol. Lond* . 160: 106–154.

ubel, DH, dan TN Wiesel. 1965. "Bidang Reseptif dan Fungsi Arsitektur di Dua Area Visual Nonstriate (18 dan 19) Kucing." *J. Neurophysiol.* 28: 229–289.

akhnenko, AG 1971. "Teori Polinomial Sistem Kompleks." *IEEE Trans. Syst. Man Cybern.* 4: 364–378.

elleher, JD, dan B. Tierney. 2018. *Ilmu Data.* MIT Press.

rizhevsky, A., I. Sutskever, dan GE Hinton. 2012. "Klasifikasi Imagenet dengan Deep Convolutional Neural Networks." Dalam *Kemajuan dalam Sistem Pemrosesan Informasi Neural* , hlm. 1097–1105.

eCun, Y. 1989. Generalisasi dan Strategi Desain Jaringan (Laporan Teknis No. CRG-TR-89-4). Kelompok Riset Koneksiis Universitas Toronto.

Iaas, AL, AY Hannun, dan AY Ng. 2013. "Nonlinier Penyearah Meningkatkan Model Akustik Jaringan Neural". Dalam *Proceedings of the Thirteenth International Conference on Machine Learning* (ICML) Workshop on Deep Learning for Audio, Speech and Language Processing, hal. 3.

IacKay, DJC 2003. *Teori Informasi, Inferensi, dan Algoritma Pembelajaran* . Cambridge University Press.

Iarcus, GF 2003. *Pikiran Aljabar: Mengintegrasikan Koneksionisme dan Ilmu Kognitif* . MIT Press.

IcCulloch, WS, dan W. Pitts. 1943. “Kalkulus Logis dari Ide-Ide yang Immanen dalam Aktivitas Saraf.” *Banteng. Matematika. Biofis.* 5: 115–133.

Iikolov, T., K. Chen, G. Corrado, dan J. Dean. 2013. "Estimasi Efisien Representasi Kata dalam Ruang Vektor." arXiv: 1301.3781.

Iinsky, M., dan S. Papert. 1969. *Perceptrons* . MIT Press.

Inih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, dan M. Riedmiller. 2013. "Bermain Atari dengan Pembelajaran Penguatan Mendalam." ArXiv13125602 Cs.

Iilsson, NJ 1965. *Mesin Pembelajaran: Dasar-dasar Sistem Klasifikasi Pola yang Dapat Dilatih, Seri dalam Ilmu Sistem* . McGraw-Hill.

)h, K.-S., dan K. Jung. 2004. "Penerapan GPU untuk Jaringan Neural". *Pengenalan Pola.* 36: 1311–1314.

)lah, C., A. Satyanarayan, I. Johnson, S. Carter, S. Ludwig, K. Ye, dan A. Mordvintsev. 2018. "The Building Block of Interpretability." Menyaring. https://doi.org/10.23915/distill.00010.

.eagen, B., R. Adolf, P. Whatmough, G.-Y. Wei, dan D. Brooks. 2017. “Pembelajaran Mendalam untuk Arsitek Komputer.” *Synth. Kuliah. Comput. Archit* . 12: 1–123. https://doi.org/10.2200/S00783ED1V01Y201706CAC041.

.eed, RD, dan RJ Marks II. 1999. *Neural Smithing: Pembelajaran yang Diawasi di Jaringan Syaraf Tiruan Feedforward* . MIT Press.

.osenblatt, F. 1960. Tentang Konvergensi Prosedur Penguatan dalam Perceptrons Sederhana (Proyek PARA). (Laporan No. VG-1196-G-4). Cornell Aeronautical Laboratory, Inc., Buffalo, NY.

.osenblatt, F. 1962. *Prinsip Neurodinamika: Perceptrons dan Teori Mekanisme Otak* . Buku Spartan.

.osenblatt, Frank, 1958. "The Perceptron: Model Probabilistik untuk Penyimpanan Informasi dan Organisasi di Otak." *Psikol. Wahyu* 65: 386–408. https://doi.org/10.1037/h0042519 .

.umelhart, DE, GE Hinton, dan RJ Williams. 1986a. "Mempelajari Representasi Internal dengan Propagasi Kesalahan". Dalam DE Rumelhart, JL McClelland, dan PDP Research Group, eds. *Pemrosesan Terdistribusi Paralel: Eksplorasi dalam Mikrostruktur Kognisi,* Vol. 1. MIT Press, hlm. 318–362.

.umelhart, DE, JL McClelland, Grup Riset PDP, eds. 1986b. *Pemrosesan Terdistribusi Paralel: Eksplorasi dalam Mikrostruktur Kognisi,* Vol. 1: *Yayasan* . MIT Press.

.umelhart, DE, JL McClelland, Grup Riset PDP, eds. 1986c. *Pemrosesan Terdistribusi Paralel: Eksplorasi dalam Mikrostruktur Kognisi,* Vol. 2: *Model Psikologis dan Biologis* . MIT Press.

abour, S., N. Frosst, dan GE Hinton. 2017. "Perutean Dinamis Antar Kapsul." Dalam *Proceedings of the 31st Conference on Neural Information Processing* (NIPS). hlm. 3856–3866.

chmidhuber, J. 2015. "Pembelajaran Mendalam di Jaringan Neural: Sebuah Tinjauan." *Jaringan Neural.* 61: 85–117.

teinkraus, D., Patrice Simard, dan I. Buck. 2005. “Menggunakan GPU untuk Algoritma Machine Learning.” Dalam *Konferensi Internasional Kedelapan tentang Analisis dan Pengakuan Dokumen* (ICDAR'05). IEEE. https://doi.org/10.1109/ICDAR.2005.251.

utskever, I., O. Vinyals, dan QV Le. 2014. "Sequence to Sequence Learning dengan Neural Networks." Dalam *Kemajuan dalam Sistem Pemrosesan Informasi Saraf* (NIPS), hlm. 3104-3112.

aigman, Y., M. Yang, M. Ranzato, dan L. Wolf. 2014. “DeepFace: Menutup Celah pada Kinerja Tingkat Manusia dalam Verifikasi Wajah.” Dipresentasikan pada Prosiding Konferensi IEEE tentang Visi Komputer dan Pengenalan Pola, hlm. 1701–1708.

an der Maaten, L., dan GE Hinton. 2008. “Memvisualisasikan Data Menggunakan t-SNE.” *J. Mach. Belajar. Res.* 9, 2579–2605.

'aswani, A., N. Shazer, N. Parmar, J. Uszkoreit, L. Jones, AN Gomez, L. Kaiser, dan I. Polosukhin. 2017. “Hanya Perhatian yang Anda Butuhkan”. Dalam *Proceedings of the 31st Conference on Neural Information Processing* (NIPS), hlm. 5998-6008.

/erbos, P. 1974. "Beyond Regression: Alat Baru untuk Prediksi dan Analisis dalam Ilmu Perilaku." PhD dis., Universitas Harvard.

Widrow, B., dan ME Hoff. 1960. Sirkuit Sakelar Adaptif (Laporan Teknis No. 1553-1). Laboratorium Elektronik Stanford, Universitas Stanford, Stanford, California.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y. 2015. “Tunjukkan, Hadir dan Beri Tahu: Citra Saraf Pembuatan Teks dengan Perhatian Visual. ” Dalam *Prosiding Konferensi Internasional ke-32 tentang Pembelajaran Mesin, Prosiding Penelitian Pembelajaran Mesin* . PMLR, hlm. 2048–2057.

Bacaan Lebih Lanjut

Buku tentang Deep Learning dan Neural Networks

Charniak, Eugene. 2018. *Pengantar Deep Learning* . MIT Press.

Goodfellow, Ian, Yoshua Bengio, dan Aaron Courville. 2016. *Pembelajaran Mendalam* . MIT Press.

Hagan, Martin T., Howard B. Demuth, Mark Hudson Beale, dan Orlando De Jesús. 2014. *Desain Jaringan Saraf* . Edisi ke-2.

Keagen, Brandon, Robert Adolf, Paul Whatmough, Gu-Yeon Wei, dan David Brooks. 2017. “Pembelajaran Mendalam untuk Arsitek Komputer.” *Kuliah Sintesis Arsitektur Komputer* 12 (4): 1–123.

Lejnowski, Terrence J. 2018. *Revolusi Pembelajaran Mendalam* . MIT Press.

Sumber Daya Online

Nielsen, Michael A. 2015. *Neural Networks dan Deep Learning* . Penentuan Pers. Tersedia di: http://neuralnetworksanddeeplearning.com.

Distill (jurnal akses terbuka dengan banyak artikel tentang pembelajaran mendalam dan pembelajaran mesin). Tersedia di: https://distill.pub.

Ikhtisar Artikel Jurnal

eCun, Yann, Yoshua Bengio, dan Geoffrey E. Hintron. 2015. “Pembelajaran Mendalam.” *Alam* 521: 436–444.

chmidhuber, Jürgen. 2015. “Pembelajaran Mendalam di Neural Networks: An Overview.” *Jaringan Neural* 61: 85–117.

Indeks

Seri Pengetahuan Penting MIT Press

Auctions , Timothy P. Hubbard dan Harry J. Paarsch
Buku , Amaranth Borsuk
Penangkapan Karbon , Howard J. Herzog
Komputasi Awan , Nayan B. Ruparelia
Pemikiran Komputasi , Peter J. Denning dan Matti Tedre
Komputasi: Sejarah Singkat , Paul E. Ceruzzi
Pikiran Sadar , Zoltan E. Torey
Crowdsourcing , Daren C. Brabham
Ilmu Data , John D. Kelleher dan Brendan Tierney
Pembelajaran Mendalam , John D. Kelleher
Ekstremisme , JM Berger
Makanan , Fabio Parasecoli
Kehendak Bebas , Mark Balaguer
Masa Depan , Nick Montfort
GPS , Paul E. Ceruzzi
Haptics , Lynette A. Jones
Informasi dan Masyarakat , Michael Buckland
Informasi dan Perusahaan Modern , James W. Cortada
Strategi Kekayaan Intelektual , John Palfrey
Internet of Things , Samuel Greengard

Pembelajaran Mesin: AI Baru , Ethem Alpaydin
Terjemahan Mesin , Thierry Poibeau
Meme dalam Budaya Digital , Limor Shifman
Metadata , Jeffrey Pomerantz
Masalah Pikiran-Tubuh , Jonathan Westphal
MOOCs , Jonathan Haber
Neuroplastisitas , Moheb Costandi
Nihilisme , Nolen Gertz
Akses Terbuka , Peter Suber
Paradoks , Margaret Cuonzo
Foto Otentikasi , Hany Farid
Pasca-Kebenaran , Lee McIntyre
Robot , John Jordan
Pilihan Sekolah , David R. Garcia
Pelacakan Diri , Gina Neff, dan Dawn Nafus
Persetujuan Seksual , Milena Popova
Penerbangan luar angkasa , Michael J. Neufeld
Keberlanjutan , Kent E. Portney
Sinestesia , Richard E. Cytowic
Singularitas Teknologi , Murray Shanahan
Pencetakan 3D , John Jordan
Memahami Keyakinan , Nils J. Nilsson
Ombak , Frederic Raichlen

John D. Kelleher adalah Profesor Ilmu Komputer dan Pemimpin Akademik dari lembaga penelitian Informasi, Komunikasi, dan Hiburan (ICE) di Technological University Dublin (TU Dublin). Dia memiliki pengalaman lebih dari dua puluh tahun dalam penelitian dan pengajaran di bidang kecerdasan buatan, pemrosesan bahasa alami, dan pembelajaran mesin. Dia telah menerbitkan lebih dari seratus artikel akademis di bidang ini, dan dua buku MIT Press: *Data Science* (2018) dan *Fundamentals of Machine Learning for Predictive Data Analytics* (2015). Risetnya didukung oleh ADAPT Research Center (https://www.adaptcentre.ie), yang didanai oleh Science Foundation Irlandia (Grant 13 / RC / 2106) dan didanai bersama oleh dana Pembangunan Regional Eropa, dan oleh proyek PRECISE4Q (https://precise4q.eu), yang didanai melalui penelitian dan inovasi program European Union's Horizon 2020 berdasarkan perjanjian hibah No. 777107.