Bab 1

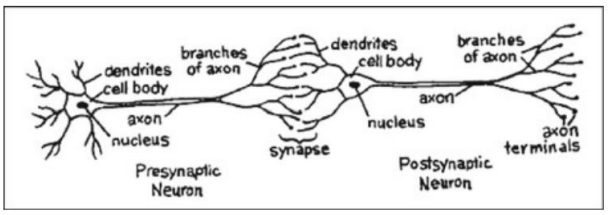
Pengantar Jaringan Syaraf

"Jangan membuat mesin untuk memalsukan pikiran manusia."—Frank Herbert

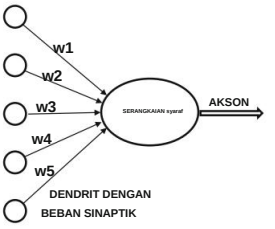
1.1 Pendahuluan

Jaringan saraf tiruan merupakan teknik pembelajaran mesin populer yang mensimulasikan mekanisme pembelajaran pada organisme biologis. Sistem saraf manusia mengandung sel-sel, yang disebut sebagai neuron. Neuron-neuron tersebut saling terhubung dengan menggunakan akson dan dendrit, dan daerah penghubung antara akson dan dendrit disebut sebagai sinapsis. Koneksi ini diilustrasikan pada Gambar 1.1(a). Kekuatan koneksi sinaptik sering berubah sebagai respons terhadap rangsangan eksternal. Perubahan ini merupakan cara pembelajaran berlangsung pada organisme hidup.

Mekanisme biologis ini disimulasikan dalam jaringan saraf tiruan, yang berisi unit komputasi yang disebut neuron. Dalam buku ini, kita akan menggunakan istilah "jaringan saraf" untuk merujuk pada jaringan saraf tiruan, bukan jaringan biologis. Unit komputasional saling terhubung melalui bobot, yang berfungsi sama



(a) Jaringan saraf biologis



(b) Jaringan syaraf tiruan

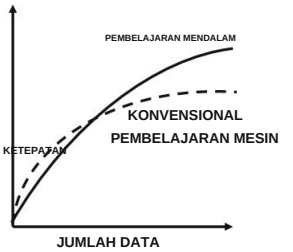
Gambar 1.1: Koneksi sinaptik antara neuron. Gambar pada (a) diambil dari "Otak: Memahami Neurobiologi Melalui Studi Kecanduan [598]." Hak cipta sekitar tahun 2000 oleh BSCS & Videodiscovery. Semua hak dilindungi undang-undang. Digunakan dengan izin.

CC Aggarwal, Jaringan Syaraf dan Pembelajaran
Mendalam, https://doi.org/10.1007/978-3-319-94463-0 1

peran sebagai kekuatan koneksi sinaptik dalam organisme biologis. Setiap masukan ke neuron diskalakan dengan bobot, yang memengaruhi fungsi yang dihitung pada unit tersebut. Arsitektur ini diilustrasikan dalam Gambar 1.1(b). Jaringan saraf tiruan menghitung fungsi masukan dengan menyebarkan nilai yang dihitung dari neuron masukan ke neuron keluaran dan menggunakan bobot sebagai parameter antara. Pembelajaran terjadi dengan mengubah bobot yang menghubungkan neuron. Sama seperti stimulus eksternal yang dibutuhkan untuk pembelajaran dalam organisme biologis, stimulus eksternal dalam jaringan saraf tiruan disediakan oleh data pelatihan yang berisi contoh pasangan masukan-keluaran dari fungsi yang akan dipelajari. Misalnya, data pelatihan mungkin berisi representasi piksel gambar (masukan) dan label beranotasi (misalnya, wortel, pisang) sebagai keluaran. Pasangan data pelatihan ini dimasukkan ke dalam jaringan saraf dengan menggunakan representasi masukan untuk membuat prediksi tentang label keluaran. Data pelatihan memberikan umpan balik terhadap kebenaran bobot dalam jaringan saraf tergantung pada seberapa baik keluaran yang diprediksi (misalnya, probabilitas wortel) untuk masukan tertentu cocok dengan label keluaran yang diberi anotasi dalam data pelatihan. Seseorang dapat melihat kesalahan yang dibuat oleh jaringan saraf dalam perhitungan suatu fungsi sebagai semacam umpan balik yang tidak menyenangkan dalam organisme biologis, yang mengarah pada penyesuaian kekuatan sinaptik. Demikian pula, bobot antara neuron disesuaikan dalam jaringan saraf sebagai respons terhadap kesalahan prediksi. Tujuan mengubah bobot adalah untuk memodifikasi fungsi yang dihitung agar prediksi lebih tepat dalam iterasi mendatang. Oleh karena itu, bobot diubah dengan hati-hati dengan cara yang dibenarkan secara matematis sehingga dapat mengurangi kesalahan dalam perhitungan pada contoh tersebut. Dengan menyesuaikan bobot antara neuron secara berurutan pada banyak pasangan masukan-keluaran, fungsi yang dihitung oleh jaringan saraf disempurnakan dari waktu ke waktu sehingga memberikan prediksi yang lebih akurat. Oleh karena itu, jika jaringan saraf dilatih dengan banyak gambar pisang yang berbeda, pada akhirnya ia akan dapat mengenali pisang dengan benar dalam gambar yang belum pernah dilihatnya sebelumnya. Kemampuan untuk menghitung fungsi input yang tidak terlihat secara akurat dengan melatih serangkaian pasangan input-output yang terbatas disebut sebagai generalisasi model. Kegunaan utama semua model pembelajaran mesin diperoleh dari kemampuannya untuk menggeneralisasi pembelajarannya dari data pelatihan yang terlihat ke contoh yang tidak terlihat.

Perbandingan biologis sering dikritik sebagai karikatur yang sangat buruk tentang cara kerja otak manusia; meskipun demikian, prinsip-prinsip ilmu saraf sering kali berguna dalam merancang arsitektur jaringan saraf. Pandangan yang berbeda adalah bahwa jaringan saraf dibangun sebagai abstraksi tingkat tinggi dari model klasik yang umum digunakan dalam pembelajaran mesin. Faktanya, unit komputasi paling dasar dalam jaringan saraf terinspirasi oleh algoritma pembelajaran mesin tradisional seperti regresi kuadrat terkecil dan regresi logistik.

Jaringan saraf memperoleh kekuatannya dengan menyatukan banyak unit dasar tersebut, dan mempelajari bobot dari berbagai unit secara bersamaan untuk meminimalkan kesalahan prediksi. Dari sudut pandang ini, jaringan saraf dapat dilihat sebagai grafik komputasional dari unit-unit dasar yang mana daya yang lebih besar diperoleh dengan menghubungkannya dengan cara-cara tertentu. Ketika jaringan saraf digunakan dalam bentuk yang paling dasar, tanpa menghubungkan beberapa unit, algoritma pembelajaran sering kali direduksi menjadi model pembelajaran mesin klasik (lihat Bab 2). Kekuatan sebenarnya dari model saraf atas metode klasik dilepaskan ketika unit komputasional dasar ini digabungkan, dan bobot dari model-model dasar dilatih menggunakan ketergantungan mereka satu sama lain. Dengan menggabungkan beberapa unit, seseorang meningkatkan kekuatan model untuk mempelajari fungsi data yang lebih rumit daripada yang melekat pada model-model dasar pembelajaran mesin dasar. Cara unit-unit ini digabungkan juga berperan dalam kekuatan arsitektur, dan memerlukan beberapa pemahaman dan wawasan dari analis. Lebih jauh, data pelatihan yang memadai juga diperlukan untuk mempelajari sejumlah besar bobot dalam grafik komputasional yang diperluas ini.



Gambar 1.2: Perbandingan ilustrasi keakuratan algoritme pembelajaran mesin yang umum dengan algoritme jaringan saraf besar. Pembelajar mendalam menjadi lebih menarik daripada metode konvensional terutama ketika data/daya komputasi yang memadai tersedia. Beberapa tahun terakhir telah terjadi peningkatan ketersediaan data dan daya komputasi, yang telah menyebabkan "ledakan Kambrium" dalam teknologi pembelajaran mendalam.

1.1.1 Manusia Versus Komputer: Melewati Batasan Kecerdasan Buatan

Manusia dan komputer pada dasarnya cocok untuk berbagai jenis tugas. Misalnya, menghitung akar pangkat tiga dari sejumlah besar sangat mudah bagi komputer, tetapi sangat sulit bagi manusia. Di sisi lain, tugas seperti mengenali objek dalam gambar adalah hal yang sederhana bagi manusia, tetapi secara tradisional sangat sulit bagi algoritma pembelajaran otomatis. Baru dalam beberapa tahun terakhir pembelajaran mendalam telah menunjukkan akurasi pada beberapa tugas ini yang melampaui manusia. Faktanya, hasil terbaru oleh algoritma pembelajaran mendalam yang melampaui kinerja manusia [184] dalam (beberapa tugas sempit) pengenalan gambar tidak akan dianggap mungkin oleh sebagian besar ahli visi komputer hingga 10 tahun yang lalu.

Banyak arsitektur pembelajaran mendalam yang telah menunjukkan kinerja luar biasa seperti itu tidak dibuat dengan menghubungkan unit komputasi tanpa pandang bulu. Kinerja superior dari jaringan saraf dalam mencerminkan fakta bahwa jaringan saraf biologis memperoleh banyak kekuatannya dari kedalaman juga. Lebih jauh lagi, jaringan biologis terhubung dengan cara yang tidak sepenuhnya kita pahami. Dalam beberapa kasus di mana struktur biologis dipahami pada beberapa tingkat, terobosan signifikan telah dicapai dengan merancang jaringan saraf buatan di sepanjang garis tersebut. Contoh klasik dari jenis arsitektur ini adalah penggunaan jaringan saraf konvolusional untuk pengenalan gambar. Arsitektur ini terinspirasi oleh eksperimen Hubel dan Wiesel [212] pada tahun 1959 tentang organisasi neuron di korteks visual kucing. Cikal bakal jaringan saraf konvolusional adalah neokognitron [127], yang secara langsung didasarkan pada hasil ini.

Struktur koneksi saraf manusia telah berevolusi selama jutaan tahun untuk mengoptimalkan kinerja yang didorong oleh kelangsungan hidup; kelangsungan hidup terkait erat dengan kemampuan kita untuk menggabungkan sensasi dan intuisi dengan cara yang saat ini tidak mungkin dilakukan dengan mesin. Ilmu saraf biologis [232] adalah bidang yang masih sangat baru, dan hanya sedikit yang diketahui tentang cara kerja otak yang sebenarnya. Oleh karena itu, wajar untuk menyarankan bahwa keberhasilan jaringan saraf konvolusional yang terinspirasi secara biologis dapat direplikasi dalam pengaturan lain, saat kita mempelajari lebih lanjut tentang cara kerja otak manusia [176]. Keuntungan utama jaringan saraf dibandingkan pembelajaran mesin tradisional adalah bahwa yang pertama menyediakan abstraksi tingkat tinggi untuk mengekspresikan wawasan semantik tentang domain data dengan pilihan desain arsitektur dalam grafik komputasi. Keuntungan kedua adalah bahwa jaringan saraf menyediakan cara sederhana untuk menyesuaikan

kompleksitas model dengan menambahkan atau menghapus neuron dari arsitektur sesuai dengan ketersediaan data pelatihan atau daya komputasi. Sebagian besar keberhasilan jaringan saraf baru-baru ini dijelaskan oleh fakta bahwa peningkatan ketersediaan data dan daya komputasi komputer modern telah melampaui batas algoritma pembelajaran mesin tradisional, yang gagal memanfaatkan sepenuhnya apa yang sekarang mungkin. Situasi ini diilustrasikan dalam Gambar 1.2. Kinerja pembelajaran mesin tradisional terkadang tetap lebih baik untuk set data yang lebih kecil karena lebih banyak pilihan, kemudahan interpretasi model yang lebih besar, dan kecenderungan untuk membuat fitur yang dapat ditafsirkan secara manual yang menggabungkan wawasan khusus domain. Dengan data yang terbatas, yang terbaik dari keragaman model yang sangat luas dalam pembelajaran mesin biasanya akan berkinerja lebih baik daripada satu kelas model (seperti jaringan saraf). Ini adalah salah satu alasan mengapa potensi jaringan saraf tidak terwujud pada tahun-tahun awal.

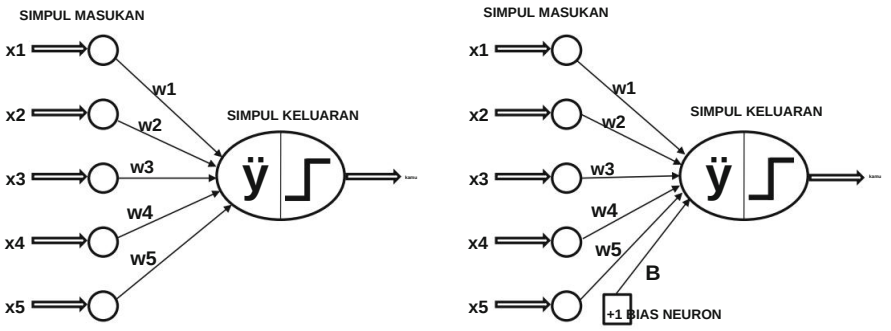
Era "big data" dimungkinkan oleh kemajuan dalam teknologi pengumpulan data; hampir semua yang kita lakukan saat ini, termasuk membeli barang, menggunakan telepon, atau mengklik situs, dikumpulkan dan disimpan di suatu tempat. Lebih jauh lagi, pengembangan Unit Prosesor Grafis (GPU) yang canggih telah memungkinkan pemrosesan yang semakin efisien pada kumpulan data yang begitu besar. Kemajuan ini sebagian besar menjelaskan keberhasilan pembelajaran mendalam terkini menggunakan algoritme yang hanya sedikit disesuaikan dari versi yang tersedia dua dekade lalu. Lebih jauh lagi, penyesuaian terbaru pada algoritme ini dimungkinkan oleh peningkatan kecepatan komputasi, karena waktu proses yang lebih singkat memungkinkan pengujian yang efisien (dan penyesuaian algoritme berikutnya). Jika diperlukan waktu satu bulan untuk menguji suatu algoritme, paling banyak dua belas variasi dapat diuji dalam setahun pada satu platform perangkat keras. Situasi ini secara historis telah membatasi eksperimen intensif yang diperlukan untuk menyempurnakan algoritme pembelajaran jaringan saraf. Kemajuan pesat yang terkait dengan tiga pilar peningkatan data, komputasi, dan eksperimen telah menghasilkan pandangan yang semakin optimis tentang masa depan pembelajaran mendalam. Pada akhir abad ini, diharapkan komputer akan memiliki kekuatan untuk melatih jaringan saraf dengan neuron sebanyak otak manusia. Meskipun sulit untuk memprediksi kemampuan sebenarnya dari kecerdasan buatan pada saat itu, pengalaman kita dengan visi komputer seharusnya mempersiapkan kita untuk mengharapkan hal yang tidak terduga.

Organisasi Bab

Bab ini disusun sebagai berikut. Bagian berikutnya memperkenalkan jaringan satu lapis dan banyak lapis. Berbagai jenis fungsi aktivasi, simpul keluaran, dan fungsi kerugian dibahas. Algoritma backpropagation diperkenalkan di Bagian 1.3. Masalah praktis dalam pelatihan jaringan neural dibahas di Bagian 1.4. Beberapa poin penting tentang bagaimana jaringan neural memperoleh kekuatannya dengan pilihan fungsi aktivasi tertentu dibahas di Bagian 1.5. Arsitektur umum yang digunakan dalam desain jaringan neural dibahas di Bagian 1.6. Topik lanjutan dalam pembelajaran mendalam dibahas di Bagian 1.7. Beberapa tolok ukur penting yang digunakan oleh komunitas pembelajaran mendalam dibahas di Bagian 1.8. Ringkasan disediakan di Bagian 1.9.

1.2 Arsitektur Dasar Jaringan Syaraf

Pada bagian ini, kami akan memperkenalkan jaringan saraf lapis tunggal dan lapis jamak. Pada jaringan lapis tunggal, sekumpulan masukan dipetakan secara langsung ke keluaran dengan menggunakan variasi umum dari fungsi linier. Instansiasi sederhana jaringan saraf ini juga disebut sebagai perceptron. Pada jaringan saraf lapis jamak, neuron disusun secara berlapis, di mana lapisan masukan dan keluaran dipisahkan oleh sekelompok lapisan tersembunyi. Arsitektur jaringan saraf lapis demi lapis ini juga disebut sebagai jaringan umpan maju. Bagian ini akan membahas jaringan lapis tunggal dan lapis jamak.



(a) Perseptron tanpa bias (b) Perseptron dengan bias

Gambar 1.3: Arsitektur dasar perceptron

1.2.1 Lapisan Komputasi Tunggal: Perceptron

Jaringan saraf yang paling sederhana disebut sebagai perceptron. Jaringan saraf ini mengandung satu lapisan input dan satu simpul output. Arsitektur dasar perceptron ditunjukkan pada Gambar 1.3(a). Pertimbangkan situasi di mana setiap contoh pelatihan berbentuk (X,y), di mana setiap $\overline{X} = [x1,...xd]$ berisi d variabel fitur, dan $y \ddot{y} \{\ddot{y}1, +1\}$ berisi nilai yang diamati dari variabel kelas biner. Dengan “nilai yang diamati” kita mengacu pada fakta bahwa itu diberikan kepada kita sebagai bagian dari data pelatihan, dan tujuan kita adalah untuk memprediksi variabel kelas untuk kasus-kasus yang tidak diperhatikan. Misalnya, dalam aplikasi deteksi penipuan kartu kredit, Fitur-fitur tersebut mungkin mewakili berbagai properti dari serangkaian transaksi kartu kredit (misalnya, jumlah dan frekuensi transaksi), dan variabel kelas mungkin mewakili apakah atau bukan rangkaian transaksi ini yang merupakan penipuan. Jelas, dalam jenis aplikasi ini, seseorang akan memiliki kasus historis di mana variabel kelas diamati, dan kasus lain (saat ini) di mana variabel kelas belum teramati tetapi perlu diprediksi.

Lapisan input berisi d node yang mengirimkan d fitur $\overline{X} = [x1 ...xd]$ dengan tepi bobot $\overline{W} = [w1 ...wd]$ ke simpul keluaran. Lapisan masukan tidak melakukan setiap perhitungan dalam haknya sendiri. Fungsi linier $W \cdot X = i = \overline{1}\ \overline{wixi}$ dihitung ^{D} pada simpul keluaran. Selanjutnya, tanda dari nilai riil ini digunakan untuk memprediksi variabel dependen X. Oleh karena itu, prediksi $\hat{y}$ dihitung sebagai berikut:

$$\hat{y} = \text{tanda}\{\overline{W} \cdot \overline{X}\} = \text{tanda}\{\sum_{j=1}^{D} wjxj\} \quad (1.1)$$

Fungsi tanda memetakan nilai riil ke +1 atau ÿ1, yang sesuai untuk biner klasifikasi. Perhatikan tanda sirkumfleks di atas variabel y untuk menunjukkan bahwa itu adalah prediksi nilai daripada nilai yang diamati. Oleh karena itu, kesalahan prediksi adalah $E(X) = y \ddot{y} \hat{y}$, yang merupakan salah satu nilai yang diambil dari himpunan {ÿ2, 0, +2}. Dalam kasus di mana nilai kesalahan E(X) bukan nol, bobot dalam jaringan saraf perlu diperbarui dalam (negatif) arah gradien kesalahan. Seperti yang akan kita lihat nanti, proses ini mirip dengan yang digunakan dalam berbagai jenis model linier dalam pembelajaran mesin. Meskipun ada kesamaan persepsi berkenaan dengan model pembelajaran mesin tradisional, interpretasinya sebagai komputasi unit sangat berguna karena memungkinkan kita untuk menyatukan beberapa unit untuk membuat yang jauh model yang lebih kuat daripada yang tersedia dalam pembelajaran mesin tradisional.

Arsitektur perceptron ditunjukkan pada Gambar 1.3(a), di mana satu lapisan input mentransmisikan fitur-fitur ke simpul output. Tepi dari input ke output berisi bobot w1 ...wd yang dengannya fitur-fitur dikalikan dan ditambahkan pada simpul output.
Selanjutnya, fungsi tanda diterapkan untuk mengubah nilai agregat menjadi label kelas. Fungsi tanda berperan sebagai fungsi aktivasi. Berbagai pilihan fungsi aktivasi dapat digunakan untuk mensimulasikan berbagai jenis model yang digunakan dalam pembelajaran mesin, seperti regresi kuadrat terkecil dengan target numerik, mesin vektor pendukung, atau pengklasifikasi regresi logistik. Sebagian besar model pembelajaran mesin dasar dapat dengan mudah direpresentasikan sebagai arsitektur jaringan saraf sederhana. Ini adalah latihan yang berguna untuk memodelkan teknik pembelajaran mesin tradisional sebagai arsitektur saraf, karena memberikan gambaran yang lebih jelas tentang bagaimana pembelajaran mendalam menggeneralisasi pembelajaran mesin tradisional. Sudut pandang ini dieksplorasi secara rinci dalam Bab 2. Perlu dicatat bahwa perceptron berisi dua lapisan, meskipun lapisan input tidak melakukan perhitungan apa pun dan hanya mengirimkan nilai fitur.
Lapisan masukan tidak termasuk dalam hitungan jumlah lapisan dalam jaringan saraf.
Karena perseptron berisi satu lapisan komputasi, ia dianggap sebagai jaringan satu lapisan.

Dalam banyak situasi, terdapat bagian invarian dari prediksi, yang disebut sebagai bias. Misalnya, pertimbangkan situasi di mana variabel fitur berpusat pada rata-rata, tetapi rata-rata prediksi kelas biner dari {ÿ1, +1} tidak 0. Hal ini cenderung terjadi dalam situasi di mana distribusi kelas biner sangat tidak seimbang. Dalam kasus seperti itu, pendekatan yang disebutkan di atas tidak cukup untuk prediksi. Kita perlu memasukkan variabel bias tambahan b yang menangkap bagian invarian dari prediksi ini:

$$\hat{y} = \text{tanda}\{\overline{W} \cdot \overline{X} + b\} = \text{tanda}\{\sum_{j=1}^{D} w_j x_j + b\} \tag{1.2}$$

Bias dapat dimasukkan sebagai bobot sisi dengan menggunakan neuron bias. Hal ini dicapai dengan menambahkan neuron yang selalu mengirimkan nilai 1 ke simpul keluaran. Bobot sisi yang menghubungkan neuron bias ke simpul keluaran menyediakan variabel bias.
Contoh neuron bias ditunjukkan pada Gambar 1.3(b). Pendekatan lain yang bekerja dengan baik dengan arsitektur lapisan tunggal adalah dengan menggunakan trik rekayasa fitur di mana fitur tambahan dibuat dengan nilai konstan 1. Koefisien fitur ini memberikan bias, dan seseorang kemudian dapat bekerja dengan Persamaan 1.1. Sepanjang buku ini, bias tidak akan digunakan secara eksplisit (untuk kesederhanaan dalam representasi arsitektur) karena bias dapat digabungkan dengan neuron bias. Rincian algoritme pelatihan tetap sama dengan hanya memperlakukan neuron bias seperti neuron lain dengan nilai aktivasi tetap 1. Oleh karena itu, berikut ini akan bekerja dengan asumsi prediktif Persamaan 1.1, yang tidak secara eksplisit menggunakan bias.

Pada saat algoritma perceptron diusulkan oleh Rosenblatt [405], optimasi ini dilakukan secara heuristik dengan sirkuit perangkat keras yang sebenarnya, dan tidak disajikan dalam bentuk gagasan formal optimasi dalam pembelajaran mesin (seperti yang umum saat ini). Namun, tujuannya selalu untuk meminimalkan kesalahan dalam prediksi, bahkan jika formulasi optimasi formal tidak disajikan. Oleh karena itu, algoritma perceptron dirancang secara heuristik untuk meminimalkan jumlah kesalahan klasifikasi, dan bukti konvergensi tersedia yang memberikan jaminan kebenaran algoritma pembelajaran dalam pengaturan yang disederhanakan. Oleh karena itu, kita masih dapat menulis tujuan (yang dimotivasi secara heuristik) dari algoritma perceptron dalam bentuk kuadrat terkecil sehubungan dengan semua contoh pelatihan dalam kumpulan data D yang berisi

berisi pasangan fitur-label:

$$\text{Memperkecil}_{\text{Kami}}\ L = \sum_{(X,Y)\in D\ (\text{Bahasa Indonesia})} (y - \hat{y})^2 = \sum_{(X,Y)\in D\ (\text{Bahasa Indonesia})} y \neq \text{tanda}\{W \cdot X\}^2$$

Fungsi tujuan minimisasi jenis ini juga disebut sebagai fungsi kerugian. Seperti yang kita ketahui, akan kita lihat nanti, hampir semua algoritma pembelajaran jaringan saraf diformulasikan dengan penggunaan dari fungsi kerugian. Seperti yang akan kita pelajari di Bab 2, fungsi kerugian ini sangat mirip dengan regresi kuadrat terkecil. Namun, yang terakhir didefinisikan untuk variabel target bernilai kontinu, dan kerugian yang sesuai adalah fungsi variabel yang halus dan berkelanjutan. Pada Di sisi lain, untuk bentuk kuadrat terkecil dari fungsi objektif, fungsi tanda tidak dapat dibedakan, dengan lompatan seperti langkah pada titik-titik tertentu. Lebih jauh, fungsi tanda mengambil pada nilai konstan di sebagian besar domain, dan oleh karena itu gradien yang tepat mengambil pada nilai nol pada titik-titik yang dapat dibedakan. Hal ini menghasilkan permukaan kerugian seperti tangga, yang tidak cocok untuk penurunan gradien. Algoritma perceptron (secara implisit) menggunakan perkiraan gradien fungsi objektif ini sehubungan dengan setiap contoh:

$$\neq L_{\text{halus}} = \sum_{(X,Y)\in D\ (\text{Bahasa Indonesia})} (y \neq \hat{y})X \tag{1.3}$$

Perhatikan bahwa gradien di atas bukanlah gradien sebenarnya dari permukaan seperti tangga dari fungsi objektif (heuristik), yang tidak memberikan gradien yang berguna. Oleh karena itu, tangga tersebut dihaluskan menjadi permukaan miring yang ditentukan oleh kriteria perceptron. Properti Kriteria perceptron akan dijelaskan pada Bagian 1.2.1.1. Perlu dicatat bahwa konsep seperti "kriteria perceptron" diusulkan lebih lambat dari makalah asli oleh Rosenblatt [405] untuk menjelaskan langkah-langkah penurunan gradien heuristik. Untuk saat ini, kita akan berasumsi bahwa Algoritma perceptron mengoptimalkan beberapa fungsi halus yang tidak diketahui dengan penggunaan gradien turun.

Meskipun fungsi tujuan di atas didefinisikan pada seluruh data pelatihan, algoritma pelatihan jaringan saraf bekerja dengan memasukkan setiap contoh data masukan X ke dalam jaringan satu per satu (atau dalam kelompok kecil) untuk membuat prediksi $\hat{y}$. Bobotnya kemudian diperbarui, berdasarkan nilai kesalahan $E(X)=(y \neq \hat{y})$. Secara khusus, ketika titik data X adalah dimasukkan ke dalam jaringan, vektor bobot W diperbarui sebagai berikut:

$$\text{Persamaan (x) dan (y) adalah persamaan yang menyatakan hubungan antara x dan y.} \tag{1.4}$$

Parameter $\neq$ mengatur laju pembelajaran jaringan saraf. Algoritma perceptron berulang kali berputar melalui semua contoh pelatihan dalam urutan acak dan menyesuaikan secara berulang bobot sampai konvergensi tercapai. Satu titik data pelatihan dapat didaur ulang berkali-kali. Setiap siklus tersebut disebut sebagai epoch. Kita juga dapat menuliskan pembaruan gradien-turun dalam bentuk kesalahan $E(X)=(y \neq \hat{y})$ sebagai berikut:

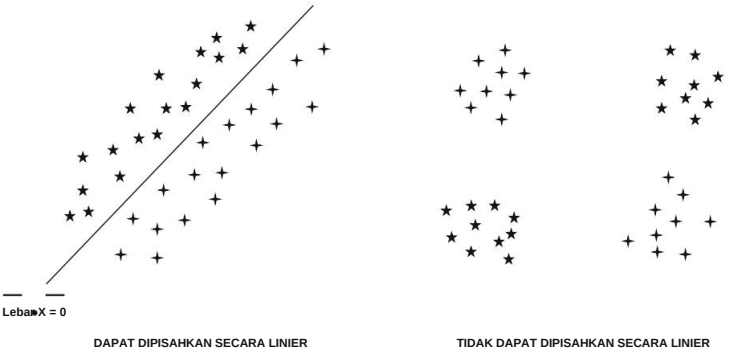
$$W \neq W + \neq E(X)X \tag{1.5}$$

Algoritma perceptron dasar dapat dianggap sebagai metode penurunan gradien stokastik, yang secara implisit meminimalkan kesalahan kuadrat prediksi dengan melakukan penurunan gradien pembaruan sehubungan dengan titik pelatihan yang dipilih secara acak. Asumsinya adalah bahwa saraf jaringan berputar melalui titik-titik dalam urutan acak selama pelatihan dan mengubah bobot dengan tujuan mengurangi kesalahan prediksi pada titik tersebut. Dari Persamaan 1.5, mudah untuk melihat bahwa pembaruan bukan nol dilakukan pada bobot hanya ketika $y = \hat{y}$, yang hanya terjadi

ketika terjadi kesalahan dalam prediksi. Dalam penurunan gradien stokastik mini-batch, pembaruan Persamaan 1.5 yang disebutkan di atas diimplementasikan pada subset titik pelatihan S yang dipilih secara acak:

$$\overline{\overline{P}\ \ddot{y}\ P} + \overline{\overline{\ddot{y}}} \qquad \text{Contoh: } \overline{X}(X)\overline{X} \qquad (1.6)$$

$$\overline{\overline{X}\ddot{y}S}$$



Gambar 1.4: Contoh data yang dapat dipisahkan secara linier dan tidak dapat dipisahkan dalam dua kelas

Keuntungan penggunaan penurunan gradien stokastik mini-batch dibahas di Bagian 3.2.8 dari Bab 3. Keunikan menarik dari perceptron adalah memungkinkan untuk mengatur laju pembelajaran ÿ menjadi 1, karena laju pembelajaran hanya menskalakan bobot.

Tipe model yang diusulkan dalam perceptron adalah model linear, di mana persamaan $W \cdot X = 0$ mendefinisikan hiperbidang linear. Di sini, W = (w1 ...wd) adalah vektor d-dimensi yang normal terhadap hiperbidang. Lebih jauh, nilai $W \cdot X$ positif untuk nilai X pada satu sisi hiperbidang, dan negatif untuk nilai X pada sisi lainnya. Tipe model ini berkinerja sangat baik ketika data dapat dipisahkan secara linear. Contoh data yang dapat dipisahkan secara linear dan tidak dapat dipisahkan ditunjukkan pada Gambar 1.4.

Algoritme perceptron bagus dalam mengklasifikasikan kumpulan data seperti yang ditunjukkan di sisi kiri Gambar 1.4, saat data dapat dipisahkan secara linear. Di sisi lain, algoritme ini cenderung berkinerja buruk pada kumpulan data seperti yang ditunjukkan di sisi kanan Gambar 1.4. Contoh ini menunjukkan keterbatasan pemodelan inheren dari sebuah perceptron, yang mengharuskan penggunaan arsitektur neural yang lebih kompleks.

Karena algoritma perceptron asli diusulkan sebagai minimalisasi heuristik dari kesalahan klasifikasi, sangat penting untuk menunjukkan bahwa algoritma tersebut konvergen ke solusi yang wajar dalam beberapa kasus khusus. Dalam konteks ini, ditunjukkan [405] bahwa algoritma perceptron selalu konvergen untuk memberikan kesalahan nol pada data pelatihan ketika data dapat dipisahkan secara linier. Namun, algoritma perceptron tidak dijamin untuk konvergen dalam contoh di mana data tidak dapat dipisahkan secara linier. Karena alasan yang dibahas di bagian berikutnya, perceptron terkadang dapat mencapai solusi yang sangat buruk dengan data yang tidak dapat dipisahkan secara linier (dibandingkan dengan banyak algoritma pembelajaran lainnya).

1.2.1.1 Fungsi Tujuan Apa yang Dioptimalkan oleh Perseptron?

Seperti yang dibahas sebelumnya dalam bab ini, makalah asli tentang perceptron oleh Rosenblatt [405] tidak secara formal mengusulkan fungsi kerugian. Pada tahun-tahun tersebut, implementasi ini dicapai dengan menggunakan rangkaian perangkat keras yang sebenarnya. Perceptron Mark I asli dimaksudkan untuk menjadi mesin daripada algoritma, dan perangkat keras yang dibuat khusus digunakan untuk membuatnya (lih. Gambar 1.5).

Tujuan umum adalah untuk meminimalkan jumlah kesalahan klasifikasi dengan pembaruan heuristik proses (dalam perangkat keras) yang mengubah bobot ke arah yang “benar” setiap kali terjadi kesalahan dibuat. Pembaruan heuristik ini sangat mirip dengan penurunan gradien tetapi tidak berasal dari sebagai metode penurunan gradien. Penurunan gradien hanya didefinisikan untuk fungsi kerugian halus dalam pengaturan algoritmik, sedangkan pendekatan yang berpusat pada perangkat keras dirancang dengan cara yang lebih

Gambar 1.5: Algoritma perceptron awalnya diimplementasikan menggunakan sirkuit perangkat keras. Gambar tersebut menggambarkan mesin perceptron Mark I yang dibuat pada tahun 1958. (Courtesy: Smithsonian) Lembaga)

cara heuristik dengan keluaran biner. Banyak prinsip biner dan sirkuit-sentris diwarisi dari model McCulloch-Pitts [321] neuron. Sayangnya, sinyal biner tidak rentan terhadap pengoptimalan berkelanjutan.

Bisakah kita menemukan fungsi kerugian halus, yang gradiennya ternyata adalah perceptron? pembaruan? Jumlah kesalahan klasifikasi dalam masalah klasifikasi biner dapat ditulis dalam bentuk fungsi kerugian 0/1 untuk titik data pelatihan (Xi, yi) sebagai berikut:

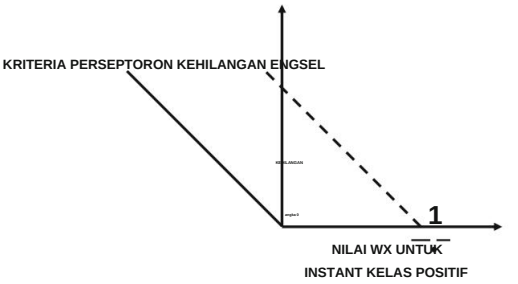
$$L_{(0/1)} = \frac{1}{2}(yi \ddot{y} \text{tanda}\{W \cdot Xi\})^2 = 1 \ddot{y} yi \cdot \text{tanda}\{W \cdot Xi\} \quad (1.7)$$

Penyederhanaan pada sisi kanan fungsi objektif di atas diperoleh dengan menetapkan y2 dan tanda{W ·Xi}2 ke 1, karena diperoleh dengan mengkuadratkan nilai yang diambil dari {ÿ1, +1}. Namun, fungsi objektif ini tidak dapat dibedakan, karena bentuknya seperti tangga, terutama jika ditambahkan pada beberapa titik. Perhatikan bahwa kerugian 0/1 di atas didominasi oleh istilah ÿyisign{W · Xi}, di mana fungsi tanda menyebabkan sebagian besar masalah yang berhubungan dengan non-diferensiabilitas. Karena jaringan saraf didefinisikan oleh optimasi berbasis gradien, kita perlu mendefinisikan fungsi objektif halus yang bertanggung jawab atas pembaruan persepsi. Dapat ditunjukkan [41] bahwa pembaruan persepsi mengoptimalkan kriteria perceptron secara implisit. Fungsi objektif ini didefinisikan dengan menghilangkan fungsi tanda di atas kehilangan 0/1 dan menetapkan nilai negatif ke 0 untuk mengobati semua prediksi yang benar dengan cara yang seragam dan tanpa kerugian:

$$Li = \text{maks}\{\ddot{y}yi(W \cdot Xi), 0\} \quad (1.8)$$

Pembaca didorong untuk menggunakan kalkulus untuk memverifikasi bahwa gradien fungsi objektif yang dihaluskan ini mengarah pada pembaruan persepsi, dan pembaruan persepsi pada dasarnya

W ÿ W ÿ ÿÿW Li. Fungsi kerugian yang dimodifikasi untuk memungkinkan komputasi gradien dari fungsi yang tidak dapat dibedakan juga disebut sebagai fungsi kerugian pengganti yang dihaluskan. Hampir semua metode pembelajaran berbasis pengoptimalan berkelanjutan (seperti jaringan saraf) dengan keluaran diskrit (seperti label kelas) menggunakan beberapa jenis fungsi kerugian pengganti yang dihaluskan.



Gambar 1.6: Kriteria perceptron versus kehilangan engsel

Meskipun kriteria perceptron yang disebutkan di atas direkayasa balik dengan bekerja mundur dari pembaruan perceptron, sifat fungsi kerugian ini memperlihatkan beberapa kelemahan pembaruan dalam algoritma asli. Pengamatan menarik tentang kriteria perceptron adalah bahwa seseorang dapat menetapkan W ke vektor nol terlepas dari set data pelatihan untuk memperoleh nilai kerugian optimal 0. Terlepas dari fakta ini, pembaruan perceptron terus menyatu ke pemisah yang jelas antara dua kelas dalam kasus yang dapat dipisahkan secara linier; lagipula, pemisah antara dua kelas juga memberikan nilai kerugian 0. Namun, perilaku untuk data yang tidak dapat dipisahkan secara linier agak arbitrer, dan solusi yang dihasilkan terkadang bahkan bukan pemisah kelas yang baik. Sensitivitas langsung kerugian terhadap besarnya vektor bobot dapat mengencerkan tujuan pemisahan kelas; pembaruan dapat memperburuk jumlah kesalahan klasifikasi secara signifikan sekaligus meningkatkan kerugian. Ini adalah contoh bagaimana fungsi kerugian pengganti terkadang tidak sepenuhnya mencapai tujuan yang dimaksudkan. Karena fakta ini, pendekatan ini tidak stabil dan dapat menghasilkan solusi dengan kualitas yang sangat bervariasi.

Oleh karena itu, beberapa variasi algoritma pembelajaran diusulkan untuk data yang tidak dapat dipisahkan, dan pendekatan alami adalah selalu melacak solusi terbaik dalam hal jumlah kesalahan klasifikasi [128]. Pendekatan untuk selalu menyimpan solusi terbaik dalam "saku" seseorang disebut sebagai algoritma saku. Varian berkinerja tinggi lainnya menggabungkan gagasan margin dalam fungsi kerugian, yang menciptakan algoritma yang identik dengan mesin vektor pendukung linier. Karena alasan ini, mesin vektor pendukung linier juga disebut sebagai perceptron stabilitas optimal.

1.2.1.2 Hubungan dengan Mesin Vektor Dukungan

Kriteria perceptron adalah versi pergeseran dari kerugian engsel yang digunakan dalam mesin vektor pendukung (lihat Bab 2). Kerugian engsel tampak lebih mirip dengan kriteria kerugian nol-satu dari Persamaan 1.7, dan didefinisikan sebagai berikut:

$$L_{svm} = \text{maks}\{1 \ddot{y} y_i(W \cdot X_i), 0\} \quad (1.9)$$

Perhatikan bahwa perseptrona tidak mempertahankan suku konstan 1 pada sisi kanan Persamaan 1.7, sedangkan kerugian engsel mempertahankan suku ini konstan dalam fungsi maksimisasi. Perubahan ini tidak mempengaruhi ekspresi aljabar untuk gradien, tetapi mengubah

titik mana yang tidak mengalami kerugian dan tidak menyebabkan pembaruan. Hubungan antara kriteria perceptron dan kerugian engsel ditunjukkan pada Gambar 1.6. Kesamaan ini menjadi sangat jelas ketika pembaruan perceptron Persamaan 1.6 ditulis ulang sebagai berikut:

$$\overline{P}\ \ddot{y}\ P + \overline{\ddot{y}} \qquad \overline{yx} \qquad (1.10)$$

$$(\overline{X},y)\ddot{y}S+$$

Di sini, S+ didefinisikan sebagai himpunan semua titik pelatihan yang salah diklasifikasikan X ÿ S yang memenuhi kondisi y(W ·X) < 0. Pembaruan ini tampaknya terlihat agak berbeda dari perceptron, karena perceptron menggunakan kesalahan E(X) untuk pembaruan, yang diganti dengan y dalam pembaruan di atas. Poin penting adalah bahwa nilai kesalahan (bilangan bulat) E(X)=(y ÿ sign{W · X}) ÿ {ÿ2, +2} tidak akan pernah menjadi 0 untuk titik yang salah diklasifikasikan di S+. Oleh karena itu, kita memiliki E(X)=2y untuk titik yang salah diklasifikasikan, dan E(X) dapat diganti dengan y dalam pembaruan setelah menyerap faktor 2 dalam laju pembelajaran. Pembaruan ini identik dengan yang digunakan oleh algoritma primal support vector machine (SVM) [448], kecuali bahwa pembaruan dilakukan hanya untuk titik-titik yang salah diklasifikasikan dalam perseptron, sedangkan SVM juga menggunakan titik-titik yang sedikit benar di dekat batas keputusan untuk pembaruan. Perhatikan bahwa SVM menggunakan kondisi y(W · X) < 1 [alih-alih menggunakan kondisi y(W · X) < 0] untuk mendefinisikan S+, yang merupakan salah satu perbedaan utama antara kedua algoritma tersebut. Hal ini menunjukkan bahwa perseptron pada dasarnya tidak jauh berbeda dari algoritma pembelajaran mesin yang terkenal seperti support vector machine meskipun asal-usulnya berbeda. Freund dan Schapire memberikan eksposisi yang indah tentang peran margin dalam meningkatkan stabilitas perseptron dan juga hubungannya dengan support vector machine [123]. Ternyata banyak model pembelajaran mesin tradisional dapat dilihat sebagai variasi minor dari arsitektur saraf dangkal seperti perseptron. Hubungan antara model pembelajaran mesin klasik dan jaringan saraf dangkal dijelaskan secara rinci dalam Bab 2.

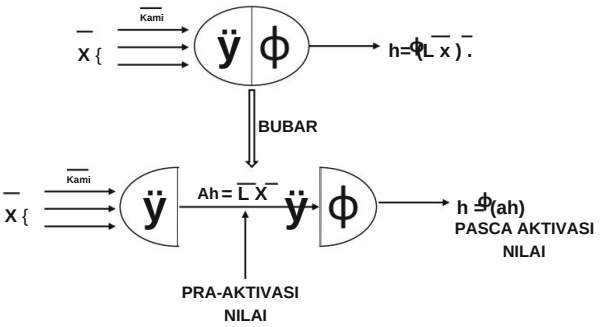
1.2.1.3 Pilihan Fungsi Aktivasi dan Kehilangan

Pemilihan fungsi aktivasi merupakan bagian penting dari desain jaringan saraf. Dalam kasus perceptron, pemilihan fungsi aktivasi tanda dimotivasi oleh fakta bahwa label kelas biner perlu diprediksi. Namun, ada kemungkinan untuk memiliki jenis situasi lain di mana variabel target yang berbeda dapat diprediksi. Misalnya, jika variabel target yang akan diprediksi adalah riil, maka masuk akal untuk menggunakan fungsi aktivasi identitas, dan algoritme yang dihasilkan sama dengan regresi kuadrat terkecil. Jika diinginkan untuk memprediksi probabilitas kelas biner, masuk akal untuk menggunakan fungsi sigmoid untuk mengaktifkan simpul keluaran, sehingga prediksi ˆy menunjukkan probabilitas bahwa nilai yang diamati, y, dari variabel dependen adalah 1. Logaritma negatif dari |y/2ÿ0.5+ˆy| digunakan sebagai kerugian, dengan asumsi bahwa y dikodekan dari {ÿ1, 1}. Jika ˆy adalah probabilitas bahwa y adalah 1, maka |y/2 ÿ 0.5+ˆy| adalah probabilitas bahwa nilai yang benar diprediksi. Pernyataan ini mudah diverifikasi dengan memeriksa dua kasus di mana y adalah 0 atau 1. Fungsi kerugian ini dapat ditunjukkan sebagai representasi dari log-likelihood negatif dari data pelatihan (lihat Bagian 2.2.3 dari Bab 2).

Pentingnya fungsi aktivasi nonlinier menjadi signifikan ketika seseorang beralih dari persepsi berlapis tunggal ke arsitektur berlapis ganda yang dibahas nanti dalam bab ini. Berbagai jenis fungsi nonlinier seperti tanda, sigmoid, atau garis singgung hiperbolik dapat digunakan dalam berbagai lapisan. Kami menggunakan notasi ÿ untuk menunjukkan fungsi aktivasi:

$$\hat{y} = \ddot{y}(\overline{W} \cdot \overline{X}) \quad (1.11)$$

Oleh karena itu, sebuah neuron benar-benar menghitung dua fungsi dalam simpul, itulah sebabnya kami telah memasukkan simbol penjumlahan ÿ dan juga simbol aktivasi ÿ dalam sebuah neuron. Pemecahan perhitungan neuron menjadi dua nilai terpisah ditunjukkan pada Gambar 1.7.



Gambar 1.7: Nilai pra-aktivasi dan pasca-aktivasi dalam neuron

Nilai yang dihitung sebelum menerapkan fungsi aktivasi ÿ(·) akan disebut sebagai nilai pra-aktivasi, sedangkan nilai yang dihitung setelah menerapkan fungsi aktivasi adalah disebut sebagai nilai pasca aktivasi. Output dari neuron selalu merupakan nilai pasca aktivasi. nilai, meskipun variabel pra-aktivasi sering digunakan dalam berbagai jenis analisis, seperti seperti perhitungan algoritma backpropagation yang dibahas kemudian dalam bab ini. Nilai pra-aktivasi dan pasca-aktivasi neuron ditunjukkan pada Gambar 1.7.

Fungsi aktivasi paling dasar ÿ(·) adalah identitas atau aktivasi linier, yang menyediakan tidak ada nonlinier:

Rumus untuk ÿ(v) = v

Fungsi aktivasi linier sering digunakan pada simpul keluaran, ketika targetnya adalah objek nyata. nilai. Bahkan digunakan untuk keluaran diskrit ketika fungsi kerugian pengganti yang dihaluskan perlu untuk disiapkan.

Fungsi aktivasi klasik yang digunakan pada awal pengembangan sistem saraf jaringan adalah fungsi tanda, sigmoid, dan fungsi tangen hiperbolik:

ÿ(v) = tanda(v) (fungsi tanda)

Persamaan $\frac{1}{(v) = 1 +}$ (fungsi sigmoid)

eÿv e2v ÿ 1

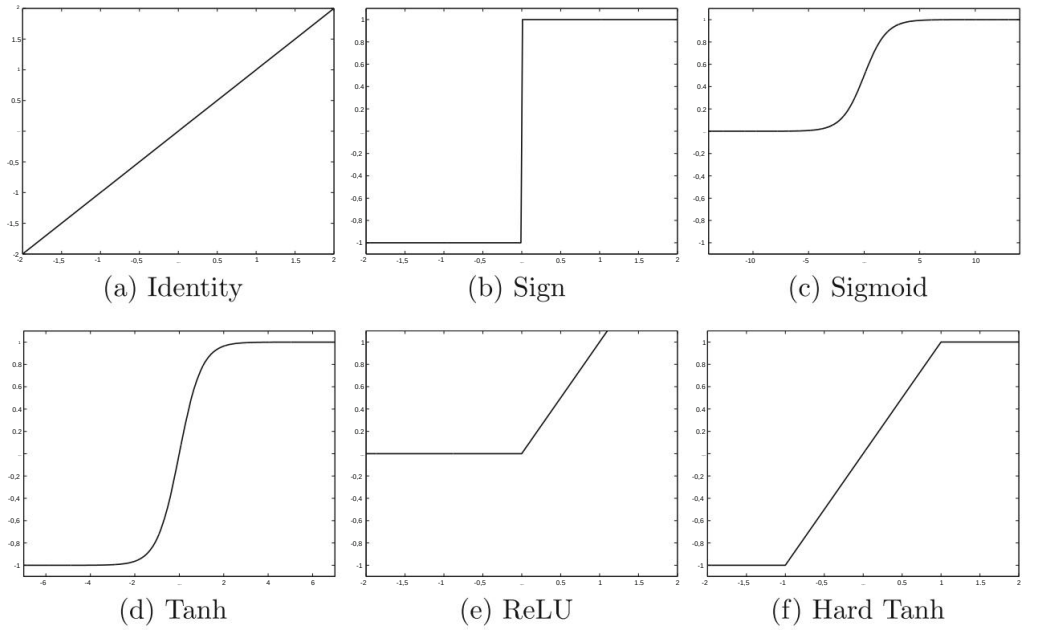
ÿ(v) = (fungsi tanh)

e2v + 1

Meskipun aktivasi tanda dapat digunakan untuk memetakan ke keluaran biner pada waktu prediksi, non-diferensiabilitas mencegah penggunaannya untuk membuat fungsi kerugian pada waktu pelatihan. Misalnya, sementara perceptron menggunakan fungsi tanda untuk prediksi, kriteria perceptron dalam pelatihan hanya memerlukan aktivasi linier. Aktivasi sigmoid menghasilkan nilai dalam (0, 1), yang berguna dalam melakukan perhitungan yang harus ditafsirkan sebagai probabilitas. Lebih jauh lagi, ini juga berguna dalam membuat keluaran probabilistik dan membangun kerugian fungsi yang berasal dari model kemungkinan maksimum. Fungsi tanh memiliki bentuk yang mirip dengan fungsi sigmoid, kecuali bahwa fungsi tersebut diskalakan ulang secara horizontal dan vertikal. diterjemahkan/diskalakan ulang ke [ÿ1, 1]. Fungsi tanh dan sigmoid terkait sebagai berikut (lihat Latihan 3):

$$\tanh(v)=2 \cdot \text{sigmoid}(2v) \ddot{y} 1$$

Fungsi tanh lebih disukai daripada fungsi sigmoid ketika keluaran perhitungan diinginkan positif dan negatif. Lebih jauh, pemusatan rata-rata dan gradien yang lebih besar



Gambar 1.8: Berbagai fungsi aktivasi

(karena peregangan) sehubungan dengan sigmoid membuatnya lebih mudah untuk dilatih. Sigmoid dan Fungsi tanh telah menjadi alat pilihan historis untuk menggabungkan nonlinieritas dalam jaringan saraf. Namun, dalam beberapa tahun terakhir, sejumlah fungsi aktivasi linier sepotong-sepotong menjadi lebih populer:

$$\ddot{y}(v) = \max\{v, 0\} \text{ (Satuan Linier Tersearahkan [ULT])}$$

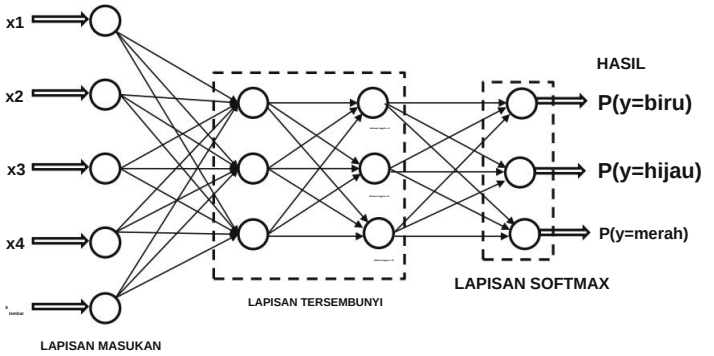
$$\ddot{y}(v) = \text{maks } \{\min [v, 1] , \ddot{y}1\} \text{ (tanh keras)}$$

Fungsi aktivasi ReLU dan hard tanh sebagian besar telah menggantikan sigmoid dan soft fungsi aktivasi tanh dalam jaringan saraf modern karena kemudahan dalam melatih jaringan saraf berlapis-lapis dengan fungsi aktivasi ini.

Representasi bergambar dari semua fungsi aktivasi yang disebutkan di atas diilustrasikan pada Gambar 1.8. Perlu dicatat bahwa semua fungsi aktivasi yang ditunjukkan di sini bersifat monotonik. Selain itu, selain fungsi aktivasi identitas, sebagian besar[1] fungsi aktivasi lainnya fungsi jenuh pada nilai absolut besar dari argumen di mana peningkatan lebih lanjut tidak tidak banyak mengubah aktivasi.

Seperti yang akan kita lihat nanti, fungsi aktivasi nonlinier seperti itu juga sangat berguna dalam multilayer jaringan, karena mereka membantu dalam menciptakan komposisi yang lebih kuat dari berbagai jenis fungsi. Banyak dari fungsi ini disebut sebagai fungsi squashing, karena mereka memetakan keluaran dari rentang sembarang ke keluaran terbatas. Penggunaan aktivasi nonlinier berperan peran mendasar dalam meningkatkan kekuatan pemodelan suatu jaringan. Jika suatu jaringan hanya digunakan aktivasi linier, itu tidak akan memberikan kekuatan pemodelan yang lebih baik daripada linier lapisan tunggal jaringan. Masalah ini dibahas di Bagian 1.5.

1ReLU menunjukkan saturasi asimetris.



Gambar 1.9: Contoh beberapa keluaran untuk klasifikasi kategoris dengan penggunaan lapisan softmax

1.2.1.4 Pilihan dan Jumlah Node Output

Pilihan dan jumlah node keluaran juga terkait dengan fungsi aktivasi, yang dalam gilirannya tergantung pada aplikasi yang ada. Misalnya, jika klasifikasi k-way dimaksudkan, k nilai keluaran dapat digunakan, dengan fungsi aktivasi softmax sehubungan dengan keluaran $\overline{v}$ = [v1,...,vk] pada node dalam lapisan tertentu. Secara khusus, fungsi aktivasi untuk lapisan i Output didefinisikan sebagai berikut:

$$\text{Rumus } \bar{y}(v)i = \frac{\exp(vi)}{\sum_{j=1}^{aku} \text{eksp}(vj)} \quad \text{Nilai k adalah } \ddot{y}i \ddot{y} \{1,...,k\} \tag{1.12}$$

Akan sangat membantu jika kita menganggap k nilai ini sebagai nilai yang dikeluarkan oleh k node, yang mana inputnya adalah v1 ...vk. Contoh fungsi softmax dengan tiga output diilustrasikan dalam Gambar 1.9, dan nilai v1, v2, dan v3 juga ditunjukkan pada gambar yang sama. Perhatikan bahwa tiga keluaran sesuai dengan probabilitas dari tiga kelas, dan mereka mengubah tiga keluaran dari lapisan tersembunyi terakhir menjadi probabilitas dengan fungsi softmax. Lapisan tersembunyi terakhir sering menggunakan aktivasi linear (identitas), saat dimasukkan ke dalam lapisan softmax. Lebih jauh lagi, tidak ada bobot yang dikaitkan dengan lapisan softmax, karena lapisan ini hanya mengubah keluaran bernilai riil menjadi probabilitas. Penggunaan softmax dengan satu lapisan tersembunyi aktivasi linier secara tepat mengimplementasikan sebuah model yang disebut logistik multinomial regresi [6]. Demikian pula, banyak variasi seperti SVM multi-kelas dapat dengan mudah diimplementasikan dengan jaringan saraf. Contoh lain dari kasus di mana beberapa simpul keluaran digunakan adalah autoencoder, di mana setiap titik data masukan direkonstruksi sepenuhnya oleh lapisan keluaran. Autoencoder dapat digunakan untuk mengimplementasikan metode faktorisasi matriks seperti nilai singular dekomposisi. Arsitektur ini akan dibahas secara rinci dalam Bab 2. Jaringan saraf paling sederhana yang mensimulasikan algoritma pembelajaran mesin dasar bersifat instruktif karena mereka terletak pada kontinum antara pembelajaran mesin tradisional dan jaringan dalam. Dengan mengeksplorasi Arsitektur ini, seseorang mendapatkan ide yang lebih baik tentang hubungan antara mesin tradisional pembelajaran dan jaringan saraf, serta keuntungan yang diberikan oleh jaringan saraf.

1.2.1.5 Pemilihan Fungsi Kerugian

Pemilihan fungsi kerugian sangat penting dalam mendefinisikan output dengan cara yang sensitif ke aplikasi yang sedang dikerjakan. Misalnya, regresi kuadrat terkecil dengan keluaran numerik

memerlukan kerugian kuadrat sederhana dalam bentuk (y ÿ yˆ)2 untuk satu contoh pelatihan dengan target y dan prediksi ˆy. Kita juga dapat menggunakan jenis kerugian lain seperti kerugian engsel untuk y ÿ {ÿ1, +1} dan prediksi bernilai riil ˆy (dengan aktivasi identitas):

$$L = \text{maks}\{0, 1 \ddot{y} y \cdot \hat{y}\} \tag{1.13}$$

Kehilangan engsel dapat digunakan untuk mengimplementasikan metode pembelajaran, yang disebut sebagai mesin vektor pendukung.

Untuk prediksi multiarah (seperti memprediksi pengenal kata atau salah satu dari beberapa kelas), keluaran softmax sangat berguna. Namun, keluaran softmax bersifat probabilistik, dan karenanya memerlukan jenis fungsi kerugian yang berbeda. Faktanya, untuk prediksi probabilistik, dua jenis fungsi kerugian yang berbeda digunakan, tergantung pada apakah prediksi tersebut biner atau multiarah:

1. Target biner (regresi logistik): Dalam kasus ini, diasumsikan bahwa nilai teramati y diambil dari {ÿ1, +1}, dan prediksi ˆy adalah nilai numerik sembarang dengan menggunakan fungsi aktivasi identitas. Dalam kasus seperti itu, fungsi kerugian untuk satu contoh dengan nilai teramati y dan prediksi bernilai riil ˆy (dengan aktivasi identitas) didefinisikan sebagai berikut:

$$L = \log(1 + \exp(\ddot{y}y \cdot \hat{y})) \tag{1.14}$$

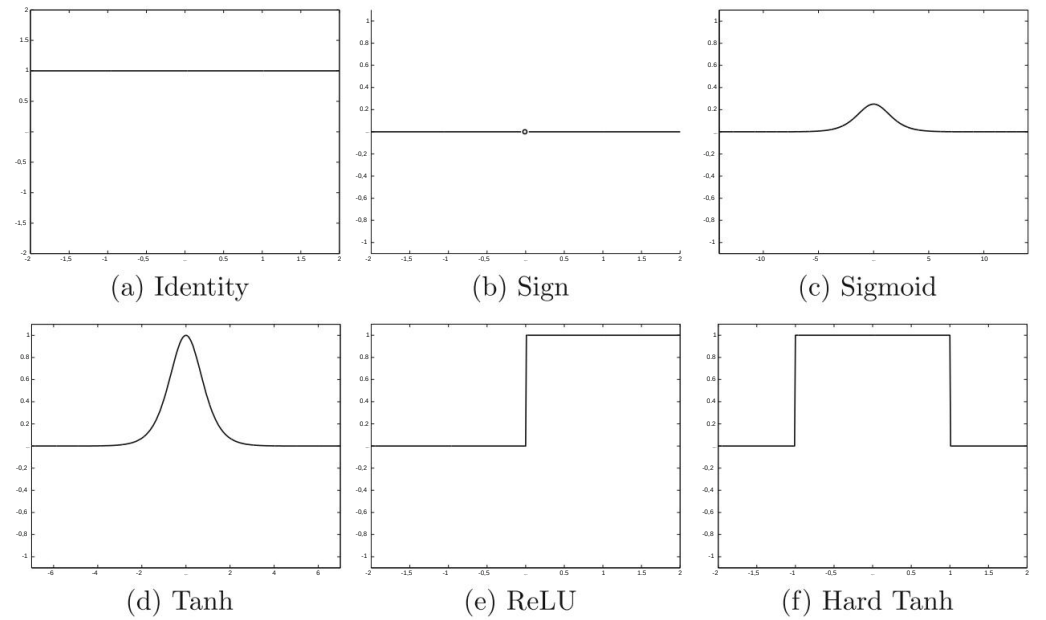
   Jenis fungsi kerugian ini menerapkan metode pembelajaran mesin fundamental, yang disebut regresi logistik. Atau, seseorang dapat menggunakan fungsi aktivasi sigmoid untuk menghasilkan ˆy ÿ (0, 1), yang menunjukkan probabilitas bahwa nilai y yang diamati adalah 1.
   Kemudian, logaritma negatif dari |y/2 ÿ 0.5+ˆy| memberikan kerugian, dengan asumsi bahwa y dikodekan dari {ÿ1, 1}. Hal ini karena |y/2 ÿ 0.5+ˆy| menunjukkan probabilitas bahwa prediksi tersebut benar. Pengamatan ini menggambarkan bahwa seseorang dapat menggunakan berbagai kombinasi fungsi aktivasi dan kerugian untuk mencapai hasil yang sama.

2. Target kategoris: Dalam kasus ini, jika ˆy1 ... yˆk adalah probabilitas kelas k (menggunakan aktivasi softmax Persamaan 1.9), dan kelas ke-r adalah kelas kebenaran dasar, maka fungsi kerugian untuk satu kejadian tunggal didefinisikan sebagai berikut:

$$L = \ddot{y}\log(\hat{y}_{tahun}) \tag{1.15}$$

   Jenis fungsi kerugian ini menerapkan regresi logistik multinomial, dan disebut sebagai kerugian entropi silang. Perhatikan bahwa regresi logistik biner identik dengan regresi logistik multinomial, ketika nilai k ditetapkan menjadi 2 pada regresi logistik multinomial.

Hal penting yang perlu diingat adalah bahwa sifat simpul keluaran, fungsi aktivasi, dan fungsi kerugian bergantung pada aplikasi yang digunakan. Lebih jauh, pilihan-pilihan ini juga saling bergantung. Meskipun perceptron sering kali disajikan sebagai representasi hakiki jaringan satu lapis, ia hanya merupakan satu representasi dari kemungkinan yang sangat besar. Dalam praktiknya, kriteria perceptron jarang digunakan sebagai fungsi kerugian. Untuk keluaran bernilai diskrit, aktivasi softmax dengan kerugian entropi silang biasanya digunakan. Untuk keluaran bernilai riil, aktivasi linear dengan kerugian kuadrat biasanya digunakan. Secara umum, kerugian entropi silang lebih mudah dioptimalkan daripada kerugian kuadrat.



Gambar 1.10: Turunan dari berbagai fungsi aktivasi

1.2.1.6 Beberapa Turunan Fungsi Aktivasi yang Berguna

Sebagian besar pembelajaran jaringan saraf terutama terkait dengan penurunan gradien dengan fungsi aktivasi. Oleh karena itu, turunan fungsi aktivasi ini digunakan berulang kali dalam buku ini, dan mengumpulkannya di satu tempat untuk referensi di masa mendatang sangatlah berguna. Bagian ini memberikan rincian tentang turunan fungsi kerugian ini. Bab-bab selanjutnya akan membahas secara ekstensif lihat hasil berikut ini.

1. Aktivasi linier dan tanda: Turunan dari fungsi aktivasi linier adalah 1 pada semua tempat. Turunan tanda (v) adalah 0 pada semua nilai v selain pada v = 0, di mana ia tidak berkesinambungan dan tidak dapat dibedakan. Karena gradien nol dan non-diferensiabilitas fungsi aktivasi ini, jarang digunakan dalam fungsi kerugian bahkan ketika digunakan untuk prediksi pada waktu pengujian. Turunan dari persamaan linier dan Aktivasi tanda diilustrasikan pada Gambar 1.10(a) dan (b), masing-masing.

2. Aktivasi sigmoid: Turunan dari aktivasi sigmoid sangatlah sederhana, ketika Hal ini diungkapkan dalam bentuk keluaran sigmoid, bukan masukan. Misalkan o adalah Output fungsi sigmoid dengan argumen v:

$$\text{atau} = \frac{1}{1 + \text{eksp}(\ddot{y}v)} \tag{1.16}$$

Selanjutnya, turunan aktivasi dapat dituliskan sebagai berikut:

$$\frac{\ddot{y}o}{\ddot{y}v} = \frac{\text{eksp}(\ddot{y}v)}{(1 + \text{eksp}(\ddot{y}v))2} \tag{1.17}$$

Hal yang penting adalah bahwa sigmoid ini dapat ditulis dengan lebih mudah dalam bentuk keluaran:

$$\frac{\ddot{y}o}{} = o(1 \ddot{y} o) \ddot{y}v \tag{1.18}$$

Turunan dari sigmoid sering digunakan sebagai fungsi output daripada fungsi input. Turunan fungsi aktivasi sigmoid diilustrasikan pada Gambar 1.10(c).

3. Aktivasi Tanh: Seperti halnya aktivasi sigmoid, aktivasi tanh sering terjadi digunakan sebagai fungsi keluaran o dan bukan masukan v:

$$\text{atau} = \frac{\exp(2v) \ddot{y} 1}{\text{eksp}(2v)+1} \tag{1.19}$$

Seseorang kemudian dapat menghitung gradien sebagai berikut:

$$\frac{\ddot{y}o}{\ddot{y}v} = \frac{4 \cdot \exp(2v)}{(\text{eksp}(2v) + 1)2} \tag{1.20}$$

Seseorang juga dapat menuliskan turunan ini dalam bentuk keluaran o:

$$\frac{\ddot{y}o}{\ddot{y}v} = 1 \ddot{y} o2 \tag{1.21}$$

Turunan dari aktivasi tanh diilustrasikan pada Gambar 1.10(d).

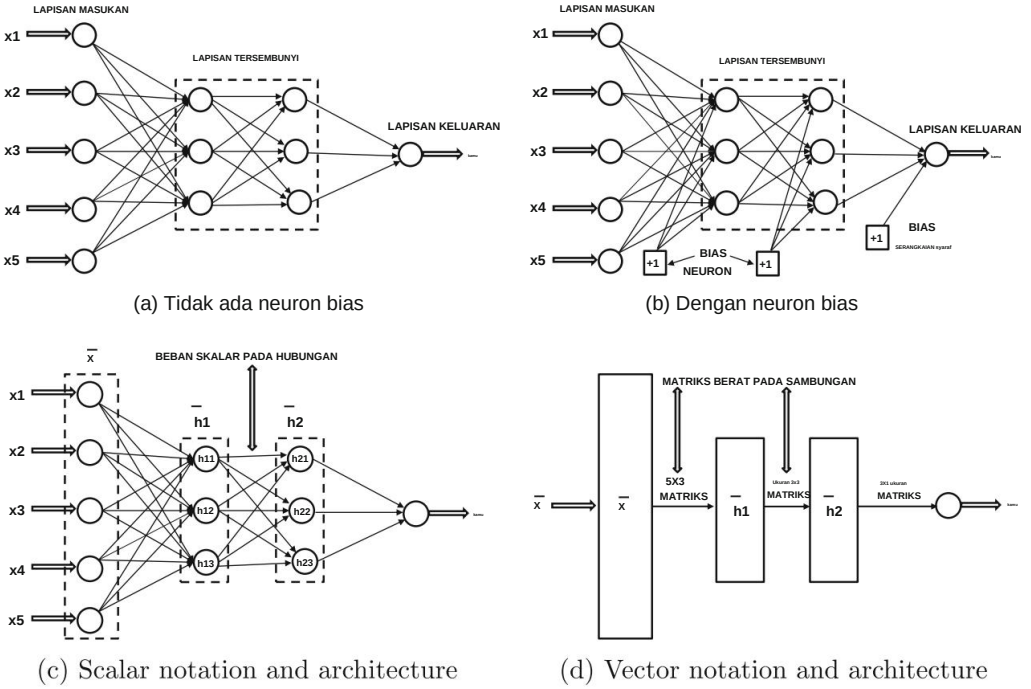
4. Aktivasi ReLU dan tanh keras: ReLU mengambil nilai turunan parsial sebesar 1 untuk nilai non-negatif dari argumennya, dan 0, jika tidak. Fungsi tanh keras mengambil nilai turunan parsial 1 untuk nilai argumen dalam [ÿ1, +1] dan 0, jika tidak. Turunan dari aktivasi ReLU dan tanh keras diilustrasikan dalam Gambar 1.10(e) dan (f), masing-masing.

1.2.2 Jaringan Syaraf Multilayer

Jaringan saraf multilayer mengandung lebih dari satu lapisan komputasi. Perceptron berisi lapisan masukan dan keluaran, yang mana lapisan keluaran merupakan satu-satunya lapisan yang melakukan komputasi. Lapisan masukan mengirimkan data ke lapisan keluaran, dan semua komputasi dapat dilihat sepenuhnya oleh pengguna. Jaringan saraf multilapis berisi beberapa lapisan komputasi; lapisan perantara tambahan (antara input dan output) adalah disebut lapisan tersembunyi karena perhitungan yang dilakukan tidak terlihat oleh pengguna. Arsitektur spesifik jaringan saraf multilayer disebut jaringan feed-forward, karena lapisan-lapisan yang berurutan saling memberi masukan ke lapisan lainnya dalam arah maju dari masukan. untuk keluaran. Arsitektur default jaringan feed-forward mengasumsikan bahwa semua node dalam satu lapisan terhubung ke lapisan berikutnya. Oleh karena itu, arsitektur jaringan saraf jaringan hampir sepenuhnya didefinisikan, setelah jumlah lapisan dan jumlah/jenis node di setiap lapisan telah didefinisikan. Satu-satunya detail yang tersisa adalah fungsi kerugian yang dioptimalkan di lapisan keluaran. Meskipun algoritma perceptron menggunakan kriteria perceptron, ini bukan satu-satunya pilihan. Sangat umum untuk menggunakan keluaran softmax dengan entropi silang kerugian untuk prediksi diskrit dan keluaran linear dengan kerugian kuadrat untuk prediksi bernilai riil.

Seperti halnya jaringan lapisan tunggal, neuron bias dapat digunakan baik dalam mode tersembunyi lapisan dan lapisan keluaran. Contoh jaringan multilayer dengan atau tanpa bias neuron ditunjukkan pada Gambar 1.11(a) dan (b), masing-masing. Dalam setiap kasus, jaringan saraf

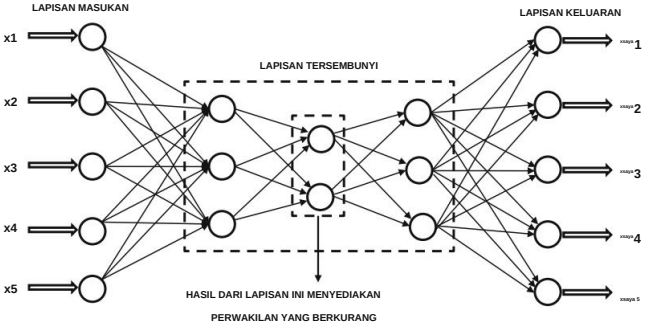
berisi tiga lapisan. Perhatikan bahwa lapisan input sering tidak dihitung, karena itu hanya mengirimkan data dan tidak ada perhitungan yang dilakukan di lapisan itu. Jika jaringan saraf berisi p1 ...pk unit di setiap k lapisannya, maka representasi vektor (kolom) dari keluaran ini, dilambangkan dengan $\overline{h1}$ ... $\overline{hk}$ memiliki dimensi p1 ... pk. Oleh karena itu, jumlah jumlah unit pada setiap lapisan disebut sebagai dimensionalitas lapisan tersebut.



(a) Tidak ada neuron bias (b) Dengan neuron bias

(c) Scalar notation and architecture (d) Vector notation and architecture

Gambar 1.11: Arsitektur dasar jaringan feed-forward dengan dua lapisan tersembunyi dan lapisan keluaran tunggal. Meskipun setiap unit berisi satu variabel skalar, seringkali satu mewakili semua unit dalam satu lapisan sebagai satu unit vektor. Unit vektor sering kali direpresentasikan sebagai persegi panjang dan memiliki matriks koneksi di antara keduanya.



Gambar 1.12: Contoh autoencoder dengan beberapa output

Bobot koneksi antara lapisan masukan dan lapisan tersembunyi pertama termuat dalam matriks W1 berukuran d × p1, sedangkan bobot antara lapisan tersembunyi ke-r dan lapisan tersembunyi ke-(r + 1) dilambangkan dengan matriks pr × pr+1 yang dilambangkan dengan Wr.
Jika lapisan keluaran berisi o simpul, maka matriks akhir Wk+1 berukuran pk × o. Vektor masukan d-dimensi x diubah menjadi keluaran menggunakan persamaan rekursif berikut:

$$\overline{h}_1 = \ddot{y}(W_T^1 \overline{x}) \quad \text{[Input ke Lapisan Tersembunyi]}$$

$$\overline{h}_{p+1} = \ddot{y}(W_{p+1}^T \overline{h}_p) \ddot{y}_p \ddot{y} \{1 \ldots k \ddot{y} 1\} o = \ddot{y}(W_{k+1}^T \overline{h}_k) \quad \text{[Tersembunyi ke Lapisan Tersembunyi]} \quad \text{[Tersembunyi di Lapisan Keluaran]}$$

Di sini, fungsi aktivasi seperti fungsi sigmoid diterapkan dalam mode elemen demi elemen pada argumen vektornya. Akan tetapi, beberapa fungsi aktivasi seperti softmax (yang biasanya digunakan dalam lapisan keluaran) secara alami memiliki argumen vektor. Meskipun setiap unit jaringan saraf berisi satu variabel, banyak diagram arsitektur menggabungkan unit-unit dalam satu lapisan untuk membuat satu unit vektor, yang direpresentasikan sebagai persegi panjang, bukan lingkaran. Misalnya, diagram arsitektur pada Gambar 1.11(c) (dengan unit skalar) telah diubah menjadi arsitektur saraf berbasis vektor pada Gambar 1.11(d). Perhatikan bahwa hubungan antara unit vektor sekarang menjadi matriks. Lebih jauh, asumsi implisit dalam arsitektur saraf berbasis vektor adalah bahwa semua unit dalam satu lapisan menggunakan fungsi aktivasi yang sama, yang diterapkan dalam mode elemen demi elemen pada lapisan tersebut. Kendala ini biasanya tidak menjadi masalah, karena sebagian besar arsitektur saraf menggunakan fungsi aktivasi yang sama di seluruh jalur komputasi, dengan satu-satunya penyimpangan yang disebabkan oleh sifat lapisan keluaran. Sepanjang buku ini, arsitektur saraf yang unitnya berisi variabel vektor akan digambarkan dengan unit persegi panjang, sedangkan variabel skalar akan sesuai dengan unit melingkar.

Perhatikan bahwa persamaan rekursif dan arsitektur vektor yang disebutkan di atas hanya berlaku untuk jaringan umpan maju per lapisan, dan tidak selalu dapat digunakan untuk desain arsitektur yang tidak konvensional. Dimungkinkan untuk memiliki semua jenis desain yang tidak konvensional di mana input dapat dimasukkan dalam lapisan antara, atau topologi dapat memungkinkan koneksi antara lapisan yang tidak berurutan. Lebih jauh, fungsi yang dihitung pada suatu simpul mungkin tidak selalu dalam bentuk kombinasi fungsi linier dan aktivasi. Dimungkinkan untuk memiliki semua jenis fungsi komputasional yang sewenang-wenang pada simpul.

Meskipun tipe arsitektur yang sangat klasik ditunjukkan pada Gambar 1.11, dimungkinkan untuk memvariasikannya dengan berbagai cara, seperti memungkinkan beberapa simpul keluaran. Pilihan ini sering ditentukan oleh tujuan aplikasi yang sedang digunakan (misalnya, klasifikasi atau pengurangan dimensionalitas). Contoh klasik dari pengaturan pengurangan dimensionalitas adalah autoencoder, yang membuat ulang keluaran dari masukan. Oleh karena itu, jumlah keluaran dan masukan sama, seperti yang ditunjukkan pada Gambar 1.12. Lapisan tersembunyi yang menyempit di tengah mengeluarkan representasi yang direduksi dari setiap contoh. Sebagai hasil dari penyempitan ini, ada beberapa kerugian dalam representasi, yang biasanya sesuai dengan gangguan dalam data. Keluaran dari lapisan tersembunyi sesuai dengan representasi data yang direduksi. Faktanya, varian dangkal dari skema ini dapat ditunjukkan secara matematis setara dengan metode pengurangan dimensionalitas yang terkenal yang dikenal sebagai dekomposisi nilai singular. Seperti yang akan kita pelajari di Bab 2, meningkatkan kedalaman jaringan menghasilkan pengurangan yang secara inheren lebih kuat.

Meskipun arsitektur yang terhubung sepenuhnya mampu bekerja dengan baik dalam banyak pengaturan, kinerja yang lebih baik sering kali dicapai dengan memangkas banyak koneksi atau membagikannya dengan cara yang berwawasan. Biasanya, wawasan ini diperoleh dengan menggunakan pemahaman data yang spesifik pada domain tertentu. Contoh klasik dari jenis pemangkasan dan pembagian bobot ini adalah

arsitektur jaringan saraf konvolusional (lih. Bab 8), yang mana arsitekturnya dirancang dengan cermat agar sesuai dengan sifat-sifat khas data gambar. Pendekatan semacam itu meminimalkan risiko overfitting dengan menggabungkan wawasan (atau bias) khusus domain.

Seperti yang akan kita bahas nanti dalam buku ini (lih. Bab 4), overfitting merupakan masalah yang meluas dalam desain jaringan saraf, sehingga jaringan sering kali berkinerja sangat baik pada data pelatihan, tetapi tidak dapat digeneralisasi dengan baik ke data uji yang tidak terlihat. Masalah ini terjadi ketika jumlah parameter bebas (yang biasanya sama dengan jumlah koneksi bobot) terlalu besar dibandingkan dengan ukuran data pelatihan. Dalam kasus seperti itu, sejumlah besar parameter mengingat nuansa spesifik dari data pelatihan, tetapi gagal mengenali pola yang signifikan secara statistik untuk mengklasifikasikan data uji yang tidak terlihat. Jelas, peningkatan jumlah simpul dalam jaringan saraf cenderung mendorong overfitting. Banyak penelitian terkini difokuskan pada arsitektur jaringan saraf serta pada perhitungan yang dilakukan dalam setiap simpul untuk meminimalkan overfitting. Lebih jauh, cara jaringan saraf dilatih juga berdampak pada kualitas solusi akhir. Banyak metode cerdas, seperti prapelatihan (lih. Bab 4), telah diusulkan dalam beberapa tahun terakhir untuk meningkatkan kualitas solusi yang dipelajari. Buku ini akan mengupas metode pelatihan lanjutan ini secara rinci.

1.2.3 Jaringan Multilayer sebagai Grafik Komputasional

Sangat membantu untuk melihat jaringan saraf sebagai grafik komputasional, yang dibangun dengan menyusun banyak model parametrik dasar. Jaringan saraf pada dasarnya lebih kuat daripada blok penyusunnya karena parameter model ini dipelajari bersama untuk menciptakan fungsi komposisi model ini yang sangat optimal. Penggunaan umum istilah "perceptron" untuk merujuk pada unit dasar jaringan saraf agak menyesatkan, karena ada banyak variasi unit dasar ini yang dimanfaatkan dalam pengaturan yang berbeda. Faktanya, jauh lebih umum untuk menggunakan unit logistik (dengan aktivasi sigmoid) dan unit linier sepotong-sepotong/penuh sebagai blok penyusun model ini.

Jaringan multilapis mengevaluasi komposisi fungsi yang dihitung pada node individual. Sebuah lintasan dengan panjang 2 dalam jaringan saraf di mana fungsi f(·) mengikuti g(·) dapat dianggap sebagai fungsi komposisi f(g(·)). Lebih jauh, jika g1(·), g2(·)...gk(·) adalah fungsi-fungsi yang dihitung dalam lapisan m, dan simpul lapisan-(m + 1) tertentu menghitung f(·), maka fungsi komposisi yang dihitung oleh simpul lapisan-(m + 1) dalam hal masukan lapisan-m adalah f(g1(·),...gk(·)). Penggunaan fungsi aktivasi nonlinier adalah kunci untuk meningkatkan daya beberapa lapisan. Jika semua lapisan menggunakan fungsi aktivasi identitas, maka jaringan multilapis dapat ditunjukkan untuk disederhanakan menjadi regresi linier. Telah ditunjukkan [208] bahwa jaringan dengan satu lapisan tersembunyi dari unit-unit nonlinier (dengan pilihan fungsi squashing yang luas seperti unit sigmoid) dan satu lapisan keluaran (linier) dapat menghitung hampir semua fungsi "wajar". Akibatnya, jaringan saraf sering disebut sebagai aproksimator fungsi universal, meskipun klaim teoritis ini tidak selalu mudah diterjemahkan ke dalam kegunaan praktis. Masalah utamanya adalah jumlah unit tersembunyi yang diperlukan untuk melakukannya agak besar, yang meningkatkan jumlah parameter yang harus dipelajari. Hal ini mengakibatkan masalah praktis dalam melatih jaringan dengan jumlah data yang terbatas. Faktanya, jaringan yang lebih dalam sering kali lebih disukai karena mengurangi jumlah unit tersembunyi di setiap lapisan serta jumlah parameter secara keseluruhan.

Deskripsi “blok bangunan” sangat tepat untuk jaringan saraf berlapis. Seringkali, perangkat lunak siap pakai untuk membangun jaringan saraf[2] menyediakan analis

[2] Contohnya termasuk Torch [572], Theano [573], dan TensorFlow [574].

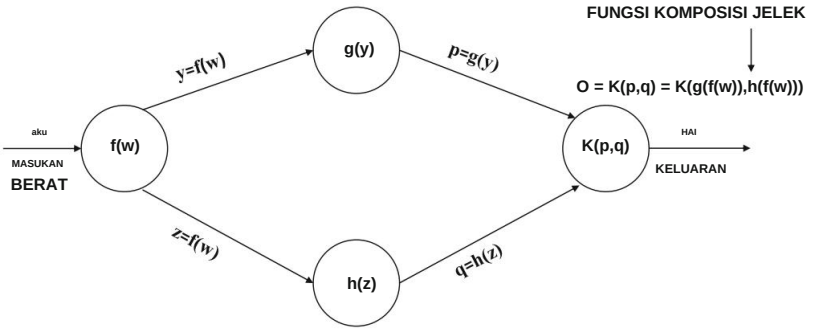
dengan akses ke blok-blok penyusun ini. Analis dapat menentukan jumlah dan jenis unit di setiap lapisan beserta fungsi kerugian yang siap pakai atau yang disesuaikan. Jaringan saraf dalam yang berisi puluhan lapisan sering kali dapat dijelaskan dalam beberapa ratus baris kode.
Semua pembelajaran bobot dilakukan secara otomatis oleh algoritma backpropagation yang menggunakan pemrograman dinamis untuk mengerjakan langkah-langkah pembaruan parameter yang rumit dari grafik komputasi yang mendasarinya. Analis tidak perlu menghabiskan waktu dan upaya untuk mengerjakan langkah-langkah ini secara eksplisit. Hal ini membuat proses mencoba berbagai jenis arsitektur relatif tidak menyakitkan bagi analis. Membangun jaringan saraf dengan banyak perangkat lunak siap pakai sering dibandingkan dengan seorang anak yang membuat mainan dari balok-balok bangunan yang saling cocok satu sama lain. Setiap balok seperti unit (atau lapisan unit) dengan jenis aktivasi tertentu. Sebagian besar kemudahan dalam melatih jaringan saraf ini disebabkan oleh algoritma backpropagation, yang melindungi analis dari mengerjakan langkah-langkah pembaruan parameter secara eksplisit dari apa yang sebenarnya merupakan masalah pengoptimalan yang sangat rumit.
Mengerjakan langkah-langkah ini sering kali merupakan bagian tersulit dari sebagian besar algoritma pembelajaran mesin, dan kontribusi penting dari paradigma jaringan saraf adalah menghadirkan pemikiran modular ke dalam pembelajaran mesin. Dengan kata lain, modularitas dalam desain jaringan saraf diterjemahkan menjadi modularitas dalam mempelajari parameternya; nama khusus untuk jenis modularitas yang terakhir adalah "backpropagation." Hal ini membuat desain jaringan saraf lebih merupakan tugas seorang insinyur (yang berpengalaman) daripada latihan matematika.

1.3 Melatih Jaringan Syaraf dengan Backpropagation

Dalam jaringan saraf satu lapis, proses pelatihan relatif mudah karena galat (atau fungsi kerugian) dapat dihitung sebagai fungsi langsung bobot, yang memungkinkan perhitungan gradien yang mudah. Dalam kasus jaringan multi lapis, masalahnya adalah kerugian merupakan fungsi komposisi rumit dari bobot di lapisan sebelumnya. Gradien fungsi komposisi dihitung menggunakan algoritma backpropagation. Algoritma backpropagation memanfaatkan aturan rantai kalkulus diferensial, yang menghitung gradien galat dalam bentuk penjumlahan produk gradien lokal di berbagai jalur dari simpul ke keluaran. Meskipun penjumlahan ini memiliki jumlah komponen (jalur) yang eksponensial, seseorang dapat menghitungnya secara efisien menggunakan pemrograman dinamis. Algoritma backpropagation merupakan aplikasi langsung dari pemrograman dinamis. Algoritma ini berisi dua fase utama, yang masing-masing disebut sebagai fase maju dan mundur. Fase maju diperlukan untuk menghitung nilai keluaran dan turunan lokal di berbagai simpul, dan fase mundur diperlukan untuk mengakumulasikan produk dari nilai-nilai lokal ini di semua jalur dari simpul ke keluaran:

1. Fase maju: Dalam fase ini, input untuk contoh pelatihan dimasukkan ke dalam jaringan saraf. Hal ini menghasilkan serangkaian komputasi maju di seluruh lapisan, menggunakan set bobot saat ini. Output akhir yang diprediksi dapat dibandingkan dengan contoh pelatihan dan turunan fungsi kerugian terhadap output dihitung. Turunan kerugian ini sekarang perlu dihitung terhadap bobot di semua lapisan dalam fase mundur.

2. Fase mundur: Sasaran utama dari fase mundur adalah mempelajari gradien fungsi kerugian sehubungan dengan bobot yang berbeda dengan menggunakan aturan rantai kalkulus diferensial. Gradien ini digunakan untuk memperbarui bobot. Karena gradien ini dipelajari dalam arah mundur, dimulai dari simpul keluaran, proses pembelajaran ini disebut sebagai fase mundur. Pertimbangkan urutan unit tersembunyi



$$\frac{\ddot{y}o}{\ddot{y}w} = \frac{\ddot{y}o}{\ddot{y}p}\frac{\ddot{y}p}{\ddot{y}w} + \frac{\ddot{y}o}{\ddot{y}q}\frac{\ddot{y}q}{\ddot{y}y} \quad \text{[Aturan Rantai Multivariabel]}$$

$$= \frac{\ddot{y}o}{\text{Tentukan}}\frac{\ddot{y}p}{\ddot{y}w \text{ dan } \ddot{y}q} + \frac{\ddot{y}o}{\ddot{y}z}\frac{\ddot{y}q}{\ddot{y}w}\frac{\ddot{y}z}{} \quad \text{[Aturan Rantai Univariat]}$$

$$= \underbrace{\frac{p \text{ dan } q}{\ddot{y}p} \cdot g'(y) \cdot f(w)'}_{\text{Jalur pertama}} + \underbrace{\frac{\text{Nilai p adalah k, k, dan k.}}{\ddot{y}q} \cdot h'(z) \cdot f(w)'}_{\text{Jalur kedua}}$$

Gambar 1.13: Ilustrasi aturan rantai dalam grafik komputasi: Hasil perkalian turunan parsial spesifik node sepanjang jalur dari bobot w ke output o digabungkan. nilai yang dihasilkan menghasilkan turunan dari output o terhadap bobot w. Hanya ada dua jalur antara masukan dan keluaran ada dalam contoh yang disederhanakan ini.

h1, h2,...,hk diikuti oleh output o, yang terhadapnya fungsi kerugian L dihitung. Lebih jauh, asumsikan bahwa bobot koneksi dari unit tersembunyi hr ke

hr+1 adalah w(hr,hr+1). Jadi, jika hanya ada satu jalur dari h1 ke o, kita dapat dapatkan gradien fungsi kerugian sehubungan dengan salah satu bobot tepi ini menggunakan Aturan rantai:

$$\frac{\ddot{y}L}{\ddot{y}w(jam\ddot{y}1,jam)} = \frac{\ddot{y}L}{\ddot{y}o}\frac{\ddot{y}o}{\ddot{y}hk}\prod_{aku=r}^{k-1}\frac{\ddot{y}hi+1}{\ddot{y}hai}\frac{\ddot{y}jam}{\ddot{y}w(jam\ddot{y}1,jam)} \quad \text{Nilai r adalah 1,56.} \qquad (1.22)$$

Ekspresi yang disebutkan di atas mengasumsikan bahwa hanya ada satu jalur dari h1 ke o di jaringan, sedangkan jumlah jalur eksponensial mungkin ada dalam kenyataan. Varian umum dari aturan rantai, disebut sebagai aturan rantai multivariabel, menghitung gradien dalam grafik komputasi, di mana lebih dari satu jalur mungkin ada. Ini adalah dicapai dengan menambahkan komposisi di sepanjang setiap jalur dari h1 ke o. Sebuah contoh aturan rantai dalam grafik komputasi dengan dua lintasan ditunjukkan pada Gambar 1.13. Oleh karena itu, kita dapat menggeneralisasikan ekspresi di atas ke kasus di mana sekumpulan P jalur ada dari hr ke o:

$$\frac{\ddot{y}L}{\ddot{y}w(jam\ddot{y}1,jam)} = \underbrace{\frac{\ddot{y}L}{\ddot{y}o}\ddot{y}}_{\ddot{y}\,[jam,jam+1,...hk,o]\ddot{y}P}\underbrace{\frac{\ddot{y}o}{\ddot{y}hk}\prod_{aku=r}^{k-1}\frac{\ddot{y}hi+1}{\ddot{y}hai}\ddot{y}}_{\ddot{y}}\frac{\ddot{y}jam}{\ddot{y}w(jam\ddot{y}1,jam)} \qquad (1.23)$$

$$\text{Propagasi mundur menghitung } \ddot{y}(jam, o) = \frac{\ddot{y}L}{\ddot{y}jam}$$

Perhitungan di sisi kanan mudah dan akan
akan dibahas di bawah (lih. Persamaan 1.27). Namun, istilah agregat jalur di atas
[diberi anotasi ÿ(hr, o) =] diagregasi selama peningkatan eksponensial
jalur (sehubungan dengan panjang jalur), yang tampaknya sulit diatasi pada pandangan pertama.
poin pentingnya adalah bahwa grafik komputasional jaringan saraf tidak memiliki siklus,
dan adalah mungkin untuk menghitung agregasi tersebut dengan cara yang berprinsip dalam metode mundur
arah dengan terlebih dahulu menghitung ÿ(hk, o) untuk node hk yang paling dekat dengan o, dan kemudian secara rekursif
menghitung nilai-nilai ini untuk simpul-simpul pada lapisan sebelumnya dalam kaitannya dengan simpul-simpul pada lapisan selanjutnya.
Selanjutnya, nilai ÿ(o, o) untuk setiap node keluaran diinisialisasi sebagai berikut:

$$\text{Apa itu } \ddot{y}(o, o)? \quad \frac{\ddot{y}L}{\ddot{y}o} \tag{1.24}$$

Teknik pemrograman dinamis jenis ini sering digunakan untuk menghitung secara efisien
semua jenis fungsi lintasan-sentris dalam grafik asiklik terarah, yang jika tidak akan
memerlukan sejumlah operasi eksponensial. Rekursi untuk ÿ(hr, o) dapat diturunkan
menggunakan aturan rantai multivariabel:

$$\ddot{y}(jam, o) = \frac{\ddot{y}L}{\ddot{y}jam} = \sum_{jam:jam\ddot{y}jam} \frac{\ddot{y}L}{\ddot{y}h}\frac{\ddot{y}h}{\ddot{y}jam} = \sum_{jam:jam\ddot{y}jam} \frac{\ddot{y}h}{\ddot{y}jam}\ddot{y}(h, o) \tag{1.25}$$

Karena setiap h berada di lapisan selanjutnya daripada hr, ÿ(h, o) telah dihitung sementara
mengevaluasi ÿ(hr, o). Namun, kita masih perlu mengevaluasi untuk menghitung Persamaan 1.25.
Pertimbangkan situasi di mana sisi yang menghubungkan hr ke h memiliki bobot w(hr,h),
dan biarkan ah menjadi nilai yang dihitung dalam unit tersembunyi h sebelum menerapkan aktivasi
fungsi ÿ(·). Dengan kata lain, kita memiliki h = ÿ(ah), di mana ah adalah kombinasi linier dari
masukannya dari unit lapisan sebelumnya yang terjadi pada h. Kemudian, dengan aturan rantai univariat,
ekspresi berikut untuk dapat diturunkan:

$$\frac{\ddot{y}h}{\ddot{y}jam} = \frac{\ddot{y}h}{\ddot{y}ah}\frac{\ddot{y}ah}{\ddot{y}jam} = \frac{\ddot{y}\ddot{y}(a)}{\ddot{y}ah} \cdot w(jam,h) = \ddot{y}\,(ah) \cdot w(jam,h)$$

Nilai ini digunakan dalam Persamaan 1.25, yang diulang secara rekursif dalam arah mundur, dimulai dengan
simpul keluaran. Pembaruan yang sesuai dalam
arah mundurnya adalah sebagai berikut:

$$\ddot{y}(jam, o) = \sum_{jam:jam\ddot{y}jam} \ddot{y}\,(ah) \cdot w(jam, jam) \cdot \ddot{y}(jam, o) \tag{1.26}$$

Oleh karena itu, gradien terakumulasi secara berurutan dalam arah mundur, dan
setiap node diproses tepat satu kali dalam lintasan mundur. Perhatikan bahwa perhitungan
Persamaan 1.25 (yang memerlukan operasi proporsional terhadap jumlah keluar
tepi) perlu diulang untuk setiap tepi yang masuk ke dalam simpul untuk menghitung gradien sehubungan
dengan semua bobot tepi. Akhirnya, Persamaan 1.23 memerlukan perhitungan
yang mana dapat dihitung dengan mudah sebagai berikut:

$$\frac{\ddot{y}jam}{\ddot{y}w(jam\ddot{y}1,jam)} = jam\ddot{y}1 \cdot \ddot{y}\,(ahr\,) \tag{1.27}$$

Di sini, gradien kunci yang dipropagasikan kembali adalah turunan terhadap aktivasi lapisan, dan gradien terhadap
bobot mudah dihitung untuk setiap insiden
tepi pada unit yang sesuai.

Perlu dicatat bahwa rekursi pemrograman dinamis Persamaan 1.26 dapat dihitung dengan beberapa cara, bergantung pada variabel mana yang digunakan untuk rantai perantara.
Semua rekursi ini setara dalam hal hasil akhir backpropagation. Berikut ini, kami memberikan versi alternatif dari rekursi pemrograman dinamis, yang lebih umum terlihat di buku teks. Perhatikan bahwa Persamaan 1.23 menggunakan variabel dalam lapisan tersembunyi sebagai variabel "rantai" untuk rekursi pemrograman dinamis. Seseorang juga dapat menggunakan nilai pra-aktivasi variabel untuk aturan rantai. Variabel pra-aktivasi dalam neuron diperoleh setelah menerapkan transformasi linier (tetapi sebelum menerapkan variabel aktivasi) sebagai variabel perantara. Nilai pra-aktivasi variabel tersembunyi h = ÿ(ah) adalah ah. Perbedaan antara nilai pra-aktivasi dan pasca-aktivasi dalam neuron ditunjukkan pada Gambar 1.7. Oleh karena itu, alih-alih Persamaan 1.23, seseorang dapat menggunakan aturan rantai berikut:

$$\frac{\ddot{y}L}{\ddot{y}w(jam\ddot{y}1,jam)} = \frac{\ddot{y}L}{\ddot{y}o} \cdot \ddot{y}'(a) \cdot \underbrace{\ddot{y}\ddot{y}[jam,jam+1,...hk,o]\ddot{y}P \sum_{aku=r}^{k-1} \frac{\ddot{y}ao}{\ddot{y}ahk} \frac{\ddot{y}ahi+1}{\ddot{y}ahi} \ddot{y}' \frac{\ddot{y}ahr}{\ddot{y}w(jam\ddot{y}1,jam)} \ddot{y}'}_{\text{Propagasi mundur menghitung } \ddot{y}(jam,o) = \ddot{y}ahr \frac{\ddot{y}L}{\ddot{y}L}} \quad (1.28)$$

Di sini, kami telah memperkenalkan notasi ÿ(hr, o) = bukan ÿ(hr, o) = ÿahr ÿhr ÿL diinisialisasi sebagai berikut: ÿao untuk pengaturan persamaan rekursif. Nilai ÿ(o, o) =

$$= \ddot{y}'(ao) \cdot \ddot{y}ao \quad \text{Tentukan nilai } \ddot{y}(o,o) = \frac{}{\ddot{y}o} \quad (1.29)$$

Kemudian, seseorang dapat menggunakan aturan rantai multivariabel untuk membuat rekursi serupa:

$$\ddot{y}(jam,o) = \frac{\ddot{y}L}{\ddot{y}ahr} = \sum_{jam:jam\ddot{y}jam} \frac{\ddot{y}(h,o)}{\ddot{y}L} \frac{\ddot{y}ah}{\ddot{y}ah} \frac{\ddot{y}ah}{\ddot{y}ahr} = \ddot{y}'(ahr) \sum_{jam:jam\ddot{y}jam} w(jam,h) \cdot \ddot{y}(h,o) \quad (1.30)$$

$$\ddot{y}(ahr)w(jam,h)$$

Kondisi rekursi ini lebih umum ditemukan dalam buku teks yang membahas backpropagation.
Turunan parsial dari kehilangan terhadap berat kemudian dihitung menggunakan ÿ(hr, o) sebagai berikut:

$$\frac{\ddot{y}L}{\ddot{y}w(jam\ddot{y}1,jam)} = \ddot{y}(jam,o) \cdot jam\ddot{y}1 \quad (1.31)$$

Seperti halnya jaringan satu lapis, proses pembaruan simpul diulang hingga mencapai konvergensi dengan berulang kali mengulang data pelatihan dalam periode waktu tertentu. Jaringan neural terkadang memerlukan ribuan periode waktu melalui data pelatihan untuk mempelajari bobot pada simpul yang berbeda. Uraian terperinci tentang algoritme backpropagation dan masalah terkait disediakan dalam Bab 3. Dalam bab ini, kami memberikan pembahasan singkat tentang masalah ini.

1.4 Masalah Praktis dalam Pelatihan Jaringan Syaraf

Meskipun jaringan saraf memiliki reputasi yang hebat sebagai aproksimator fungsi universal, masih ada tantangan besar dalam hal melatih jaringan saraf untuk memberikan tingkat kinerja ini. Tantangan ini terutama terkait dengan beberapa masalah praktis yang terkait dengan pelatihan, yang paling penting di antaranya adalah overfitting.

1.4.1 Masalah Overfitting

Masalah overfitting mengacu pada fakta bahwa pemasangan model pada set data pelatihan tertentu tidak menjamin bahwa model tersebut akan memberikan kinerja prediksi yang baik pada data uji yang tidak terlihat, bahkan jika model tersebut memprediksi target pada data pelatihan dengan sempurna. Dengan kata lain, selalu ada kesenjangan antara kinerja data pelatihan dan data uji, yang khususnya besar ketika modelnya kompleks dan set datanya kecil.

Untuk memahami hal ini, pertimbangkan jaringan saraf satu lapis sederhana pada set data dengan lima atribut, di mana kita menggunakan aktivasi identitas untuk mempelajari variabel target bernilai riil. Arsitektur ini hampir identik dengan Gambar 1.3, kecuali bahwa fungsi aktivasi identitas digunakan untuk memprediksi target bernilai riil. Oleh karena itu, jaringan mencoba mempelajari fungsi berikut:

$$\hat{y} = \sum_{saya=1}^{5} saya \cdot xi \tag{1.32}$$

Pertimbangkan situasi di mana nilai target yang diamati adalah nyata dan selalu dua kali lipat nilai atribut pertama, sedangkan atribut lainnya sama sekali tidak terkait dengan target. Akan tetapi, kita hanya memiliki empat contoh pelatihan, yang berarti satu kurang dari jumlah fitur (parameter bebas). Misalnya, contoh pelatihan dapat berupa sebagai berikut:

x1x2x3x4x5 tahun	
11000 2 20100 4 30010 6 40001 8	

Vektor parameter yang benar dalam kasus ini adalah W = [2, 0, 0, 0, 0] berdasarkan hubungan yang diketahui antara fitur pertama dan target. Data pelatihan juga memberikan kesalahan nol dengan solusi ini, meskipun hubungan tersebut perlu dipelajari dari contoh yang diberikan karena tidak diberikan kepada kita secara apriori. Namun, masalahnya adalah jumlah titik pelatihan lebih sedikit daripada jumlah parameter dan dimungkinkan untuk menemukan solusi dalam jumlah tak terbatas dengan kesalahan nol. Misalnya, set parameter [0, 2, 4, 6, 8] juga memberikan kesalahan nol pada data pelatihan. Namun, jika kita menggunakan solusi ini pada data uji yang tidak terlihat, kemungkinan akan memberikan kinerja yang sangat buruk karena parameter yang dipelajari disimpulkan secara salah dan tidak mungkin digeneralisasi dengan baik ke titik-titik baru di mana target dua kali lipat dari atribut pertama (dan atribut lainnya acak). Jenis inferensi palsu ini disebabkan oleh kelangkaan data pelatihan, di mana nuansa acak dikodekan ke dalam model. Akibatnya, solusinya tidak dapat digeneralisasi dengan baik ke data uji yang tidak terlihat. Situasi ini hampir mirip dengan pembelajaran dengan hafalan, yang sangat prediktif untuk data pelatihan tetapi tidak prediktif untuk data uji yang tidak terlihat. Menambah jumlah contoh pelatihan meningkatkan kekuatan generalisasi model, sedangkan menambah kompleksitas model mengurangi kekuatan generalisasinya. Pada saat yang sama, ketika banyak data pelatihan tersedia, model yang terlalu sederhana tidak mungkin menangkap hubungan yang kompleks antara fitur dan target. Aturan praktis yang baik adalah bahwa jumlah total titik data pelatihan harus setidaknya 2 hingga 3 kali lebih besar dari jumlah parameter dalam jaringan saraf, meskipun jumlah contoh data yang tepat bergantung pada model spesifik yang ada. Secara umum, model dengan jumlah parameter yang lebih besar dikatakan memiliki kapasitas tinggi, dan memerlukan jumlah data yang lebih besar untuk mendapatkan kekuatan generalisasi ke data uji yang tidak terlihat. Gagasan overfitting sering dipahami dalam trade-off antara bias dan varians dalam p

kesimpulan dari gagasan tentang trade-off bias-varians adalah bahwa seseorang tidak selalu menang dengan model yang lebih kuat (yaitu, kurang bias) ketika bekerja dengan data pelatihan yang terbatas, karena varians model-model ini lebih tinggi. Misalnya, jika kita mengubah data pelatihan dalam tabel di atas ke kumpulan empat titik yang berbeda, kita cenderung mempelajari kumpulan parameter yang sama sekali berbeda (dari nuansa acak titik-titik tersebut). Model baru ini cenderung menghasilkan prediksi yang sama sekali berbeda pada contoh pengujian yang sama dibandingkan dengan prediksi menggunakan kumpulan data pelatihan pertama. Jenis variasi dalam prediksi contoh pengujian yang sama menggunakan kumpulan data pelatihan yang berbeda merupakan manifestasi dari varians model, yang juga menambah kesalahan model; lagipula, kedua prediksi contoh pengujian yang sama tidak mungkin benar. Model yang lebih kompleks memiliki kelemahan dalam melihat pola yang salah dalam nuansa acak, terutama ketika data pelatihan tidak mencukupi. Seseorang harus berhati-hati untuk memilih titik optimal ketika memutuskan kompleksitas model. Gagasan ini dijelaskan secara rinci dalam Bab 4.

Jaringan saraf secara teoritis selalu dikenal cukup kuat untuk memperkirakan fungsi apa pun [208]. Akan tetapi, kurangnya ketersediaan data dapat mengakibatkan kinerja yang buruk; ini adalah salah satu alasan mengapa jaringan saraf baru-baru ini menjadi terkenal. Ketersediaan data yang lebih besar telah mengungkapkan keunggulan jaringan saraf dibandingkan pembelajaran mesin tradisional (lih. Gambar 1.2). Secara umum, jaringan saraf memerlukan desain yang cermat untuk meminimalkan efek berbahaya dari overfitting, bahkan ketika sejumlah besar data tersedia. Bagian ini memberikan gambaran umum tentang beberapa metode desain yang digunakan untuk mengurangi dampak overfitting.

1.4.1.1 Regularisasi

Karena jumlah parameter yang lebih besar menyebabkan overfitting, pendekatan alami adalah membatasi model untuk menggunakan lebih sedikit parameter bukan nol. Dalam contoh sebelumnya, jika kita membatasi vektor W untuk hanya memiliki satu komponen bukan nol dari lima komponen, maka akan diperoleh solusi [2, 0, 0, 0, 0] dengan benar. Nilai absolut parameter yang lebih kecil juga cenderung lebih sedikit melakukan overfitting. Karena sulit untuk membatasi nilai parameter, pendekatan yang lebih lunak dengan menambahkan penalti ÿ||W|| p ke fungsi kerugian digunakan. Nilai p biasanya ditetapkan ke 2, yang mengarah ke regularisasi Tikhonov. Secara umum, nilai kuadrat dari setiap parameter (dikalikan dengan parameter regularisasi ÿ > 0) ditambahkan ke fungsi tujuan.
Efek praktis dari perubahan ini adalah bahwa kuantitas yang proporsional terhadap ÿwi dikurangi dari pembaruan parameter wi. Contoh versi terregulasi Persamaan 1.6 untuk mini-batch S dan ukuran langkah pembaruan ÿ > 0 adalah sebagai berikut:

$$\underbrace{w \leftarrow w(1 - \alpha\lambda) + \alpha}_{\text{Bahasa Indonesia: W ÿ W(1 ÿ ÿÿ) + ÿ}} \underbrace{\sum_{(x,y)\in S} \nabla L(x, y, W)}_{\text{Contoh: X(X)X}} \quad (1.33)$$

Di sini, E[X] merepresentasikan kesalahan saat ini (y ÿ ŷ) antara nilai yang diamati dan diprediksi dari contoh pelatihan X. Seseorang dapat melihat jenis penalti ini sebagai semacam penurunan bobot selama pembaruan. Regularisasi sangat penting ketika jumlah data yang tersedia terbatas. Interpretasi biologis yang tepat dari regularisasi adalah bahwa hal itu sesuai dengan pelupaan bertahap, sebagai akibatnya pola yang "kurang penting" (yaitu, berisik) dihilangkan.
Secara umum, sering disarankan untuk menggunakan model yang lebih kompleks dengan regularisasi daripada model yang lebih sederhana tanpa regularisasi.

Sebagai catatan tambahan, bentuk umum Persamaan 1.33 digunakan oleh banyak model pembelajaran mesin yang terregulasi seperti regresi kuadrat terkecil (lih. Bab 2), di mana E(X) digantikan oleh fungsi kesalahan dari model spesifik tersebut. Menariknya, peluruhan bobot hanya digunakan secara terbatas dalam

single-layer perceptron3 karena terkadang dapat menyebabkan lupa yang terlalu cepat dengan sejumlah kecil titik pelatihan yang baru-baru ini salah diklasifikasikan yang mendominasi vektor bobot; masalah utamanya adalah bahwa kriteria perceptron sudah merupakan fungsi kerugian yang merosot dengan nilai minimum 0 pada $W = \overline{0}$ (tidak seperti sepupunya hinge-loss atau least-square). Keunikan ini merupakan warisan dari fakta bahwa single-layer perceptron awalnya didefinisikan dalam hal pembaruan yang terinspirasi secara biologis daripada dalam hal fungsi kerugian yang dipikirkan dengan saksama. Konvergensi ke solusi optimal tidak pernah dijamin selain dalam kasus yang dapat dipisahkan secara linier. Untuk single-layer perceptron, beberapa teknik regularisasi lainnya, yang dibahas di bawah ini, lebih umum digunakan.

1.4.1.2 Arsitektur Neural dan Pembagian Parameter

Cara paling efektif untuk membangun jaringan saraf adalah dengan membangun arsitektur jaringan saraf setelah memikirkan domain data yang mendasarinya. Misalnya, kata-kata yang berurutan dalam sebuah kalimat sering kali saling terkait, sedangkan piksel di dekatnya dalam sebuah gambar biasanya saling terkait. Jenis wawasan ini digunakan untuk membuat arsitektur khusus untuk data teks dan gambar dengan lebih sedikit parameter. Lebih jauh lagi, banyak parameter yang mungkin digunakan bersama. Misalnya, jaringan saraf konvolusional menggunakan serangkaian parameter yang sama untuk mempelajari karakteristik blok lokal gambar. Kemajuan terkini dalam penggunaan jaringan saraf seperti jaringan saraf berulang dan jaringan saraf konvolusional adalah contoh dari fenomena ini.

1.4.1.3 Penghentian Dini

Bentuk umum regularisasi lainnya adalah penghentian awal, di mana penurunan gradien diakhiri setelah hanya beberapa iterasi. Salah satu cara untuk menentukan titik penghentian adalah dengan menahan sebagian data pelatihan, lalu menguji kesalahan model pada set yang ditahan. Pendekatan penurunan gradien diakhiri saat kesalahan pada set yang ditahan mulai meningkat.
Penghentian dini pada dasarnya mengurangi ukuran ruang parameter ke lingkungan yang lebih kecil dalam nilai awal parameter. Dari sudut pandang ini, penghentian dini bertindak sebagai pengatur karena secara efektif membatasi ruang parameter.

1.4.1.4 Menukar Keluasan dengan Kedalaman

Seperti yang dibahas sebelumnya, jaringan saraf dua lapis dapat digunakan sebagai aproksimator fungsi universal [208], jika sejumlah besar unit tersembunyi digunakan dalam lapisan tersembunyi. Ternyata jaringan dengan lebih banyak lapisan (yaitu, kedalaman lebih besar) cenderung membutuhkan lebih sedikit unit per lapisan karena fungsi komposisi yang dibuat oleh lapisan yang berurutan membuat jaringan saraf lebih kuat. Peningkatan kedalaman adalah bentuk regularisasi, karena fitur-fitur di lapisan selanjutnya dipaksa untuk mematuhi jenis struktur tertentu yang diberlakukan oleh lapisan sebelumnya. Peningkatan kendala mengurangi kapasitas jaringan, yang berguna ketika ada batasan pada jumlah data yang tersedia. Penjelasan singkat tentang jenis perilaku ini diberikan di Bagian 1.5. Jumlah unit di setiap lapisan biasanya dapat dikurangi sedemikian rupa sehingga jaringan dalam sering kali memiliki parameter yang jauh lebih sedikit bahkan ketika ditambahkan pada jumlah lapisan yang lebih besar. Pengamatan ini telah menyebabkan ledakan dalam penelitian tentang topik pembelajaran mendalam.

3Peluruhan berat umumnya digunakan dengan fungsi kerugian lainnya dalam model lapisan tunggal dan dalam semua model multi-lapisan. model dengan sejumlah besar parameter.

Meskipun jaringan dalam memiliki lebih sedikit masalah terkait overfitting, jaringan dalam memiliki sejumlah masalah berbeda yang terkait dengan kemudahan pelatihan. Secara khusus, turunan kerugian terkait bobot di berbagai lapisan jaringan cenderung memiliki besaran yang sangat berbeda, yang menyebabkan tantangan dalam memilih ukuran langkah dengan tepat. Berbagai manifestasi perilaku yang tidak diinginkan ini disebut sebagai masalah gradien yang menghilang dan meledak. Lebih jauh, jaringan dalam sering kali membutuhkan waktu yang sangat lama untuk konvergen.
Masalah dan pilihan desain ini akan dibahas kemudian di bagian ini dan di beberapa tempat di seluruh buku ini.

1.4.1.5 Metode Ensemble

Berbagai metode ensemble seperti bagging digunakan untuk meningkatkan daya generalisasi model. Metode-metode ini tidak hanya berlaku untuk jaringan neural tetapi juga untuk semua jenis algoritma pembelajaran mesin. Namun, dalam beberapa tahun terakhir, sejumlah metode ensemble yang secara khusus difokuskan pada jaringan neural juga telah diusulkan. Dua metode tersebut meliputi Dropout dan Dropconnect. Metode-metode ini dapat dikombinasikan dengan banyak arsitektur jaringan neural untuk memperoleh peningkatan akurasi tambahan sekitar 2% dalam banyak pengaturan nyata. Namun, peningkatan yang tepat bergantung pada jenis data dan sifat pelatihan yang mendasarinya. Misalnya, menormalkan aktivasi dalam lapisan tersembunyi dapat mengurangi efektivitas metode Dropout, meskipun seseorang dapat memperoleh keuntungan dari normalisasi itu sendiri. Metode ensemble dibahas dalam Bab 4.

1.4.2 Masalah Gradien Hilang dan Meledak

Meskipun peningkatan kedalaman sering kali mengurangi jumlah parameter jaringan, hal itu mengarah pada berbagai jenis masalah praktis. Penyebaran mundur menggunakan aturan rantai memiliki kekurangan dalam jaringan dengan sejumlah besar lapisan dalam hal stabilitas pembaruan. Secara khusus, pembaruan di lapisan sebelumnya dapat sangat kecil (gradien menghilang) atau dapat semakin besar (gradien meledak) dalam jenis arsitektur jaringan saraf tertentu. Hal ini terutama disebabkan oleh komputasi produk seperti rantai dalam Persamaan 1.23, yang dapat meningkat atau menurun secara eksponensial sepanjang jalur. Untuk memahami hal ini, pertimbangkan situasi di mana kita memiliki jaringan multilapis dengan satu neuron di setiap lapisan. Setiap turunan lokal di sepanjang jalur dapat ditunjukkan sebagai hasil kali bobot dan turunan fungsi aktivasi. Turunan yang dipropagasikan mundur secara keseluruhan adalah hasil kali nilai-nilai ini. Jika setiap nilai tersebut didistribusikan secara acak, dan memiliki nilai yang diharapkan kurang dari 1, produk turunan ini dalam Persamaan 1.23 akan turun secara eksponensial dengan cepat seiring panjang lintasan. Jika nilai-nilai individual pada lintasan memiliki nilai yang diharapkan lebih besar dari 1, hal itu biasanya akan menyebabkan gradien meledak. Bahkan jika turunan lokal didistribusikan secara acak dengan nilai yang diharapkan tepat 1, turunan keseluruhan biasanya akan menunjukkan ketidakstabilan tergantung pada bagaimana nilai-nilai tersebut sebenarnya didistribusikan. Dengan kata lain, masalah gradien yang menghilang dan meledak cukup alami pada jaringan dalam, yang membuat proses pelatihannya tidak stabil.

Banyak solusi telah diusulkan untuk mengatasi masalah ini. Misalnya, aktivasi sigmoid sering kali mendorong masalah gradien yang menghilang, karena turunannya kurang dari 0,25 pada semua nilai argumennya (lihat Latihan 7), dan sangat kecil pada saturasi. Unit aktivasi ReLU diketahui lebih kecil kemungkinannya untuk menciptakan masalah gradien yang menghilang karena turunannya selalu 1 untuk nilai argumen yang positif. Pembahasan lebih lanjut tentang masalah ini disediakan dalam Bab 3. Selain penggunaan ReLU, sejumlah besar trik penurunan gradien digunakan untuk meningkatkan perilaku konvergensi masalah tersebut. Secara khusus, penggunaan

kecepatan belajar adaptif dan metode gradien konjugasi dapat membantu dalam banyak kasus. Lebih jauh lagi, teknik terbaru yang disebut normalisasi batch membantu dalam mengatasi beberapa masalah ini. Teknik-teknik ini dibahas dalam Bab 3.

1.4.3 Kesulitan dalam Konvergensi

Konvergensi yang cukup cepat dari proses optimasi sulit dicapai dengan jaringan yang sangat dalam, karena kedalaman menyebabkan peningkatan resistensi terhadap proses pelatihan dalam hal membiarkan gradien mengalir dengan lancar melalui jaringan. Masalah ini agak terkait dengan masalah gradien yang menghilang, tetapi memiliki karakteristik uniknya sendiri. Oleh karena itu, beberapa "trik" telah diusulkan dalam literatur untuk kasus-kasus ini, termasuk penggunaan jaringan gating dan jaringan residual [184]. Metode-metode ini dibahas masing-masing dalam Bab 7 dan 8.

1.4.4 Optima Lokal dan Palsu

Fungsi optimasi jaringan saraf sangat nonlinier, yang memiliki banyak optima lokal. Ketika ruang parameter besar, dan terdapat banyak optima lokal, masuk akal untuk meluangkan sedikit upaya dalam memilih titik inisialisasi yang baik. Salah satu metode untuk meningkatkan inisialisasi jaringan saraf disebut sebagai pra-pelatihan. Ide dasarnya adalah menggunakan pelatihan terbimbing atau tak terbimbing pada sub-jaringan dangkal dari jaringan asli untuk membuat bobot awal. Jenis pra-pelatihan ini dilakukan dengan cara serakah dan berlapis-lapis di mana satu lapisan jaringan dilatih pada satu waktu untuk mempelajari titik inisialisasi lapisan tersebut. Jenis pendekatan ini menyediakan titik inisialisasi yang mengabaikan bagian ruang parameter yang sangat tidak relevan untuk memulai. Lebih jauh, pra-pelatihan tak terbimbing sering cenderung menghindari masalah yang terkait dengan overfitting. Ide dasarnya di sini adalah bahwa beberapa minimum dalam fungsi kerugian adalah optima palsu karena hanya ditunjukkan dalam data pelatihan dan tidak dalam data uji. Penggunaan pra-pelatihan tanpa pengawasan cenderung memindahkan titik inisialisasi lebih dekat ke cekungan optima “baik” dalam data uji. Ini adalah masalah yang terkait dengan generalisasi model. Metode untuk pra-pelatihan dibahas di Bagian 4.7 dari Bab 4.

Menariknya, gagasan optima palsu sering kali dilihat dari sudut pandang generalisasi model dalam jaringan saraf. Ini adalah perspektif yang berbeda dari optimasi tradisional.
Dalam optimasi tradisional, seseorang tidak fokus pada perbedaan fungsi kerugian pada data pelatihan dan pengujian, tetapi pada bentuk fungsi kerugian pada data pelatihan saja.
Anehnya, masalah optima lokal (dari perspektif tradisional) merupakan masalah yang lebih kecil dalam jaringan saraf daripada yang biasanya diharapkan dari fungsi nonlinier tersebut. Sebagian besar waktu, nonlinieritas menyebabkan masalah selama proses pelatihan itu sendiri (misalnya, kegagalan untuk konvergen), daripada terjebak dalam minimum lokal.

1.4.5 Tantangan Komputasi

Tantangan signifikan dalam desain jaringan neural adalah waktu yang dibutuhkan untuk melatih jaringan. Tidak jarang diperlukan waktu berminggu-minggu untuk melatih jaringan neural dalam domain teks dan gambar. Dalam beberapa tahun terakhir, kemajuan dalam teknologi perangkat keras seperti Graphics Processor Units (GPU) telah membantu secara signifikan. GPU adalah prosesor perangkat keras khusus yang dapat mempercepat jenis operasi yang umum digunakan dalam jaringan neural secara signifikan. Dalam hal ini, beberapa kerangka kerja algoritmik seperti Torch sangat praktis karena memiliki dukungan GPU yang terintegrasi erat ke dalam platform.

Meskipun kemajuan algoritmik telah memainkan peran dalam kegembiraan terkini seputar pembelajaran mendalam, banyak keuntungan yang diperoleh dari fakta bahwa algoritme yang sama dapat melakukan lebih banyak hal pada perangkat keras modern. Perangkat keras yang lebih cepat juga mendukung pengembangan algoritmik, karena seseorang perlu menguji algoritme intensif komputasi secara berulang untuk memahami apa yang berhasil dan apa yang tidak. Misalnya, model saraf terkini seperti memori jangka pendek hanya berubah sedikit [150] sejak pertama kali diusulkan pada tahun 1997 [204]. Namun, potensi model ini baru-baru ini diakui karena kemajuan dalam daya komputasi mesin modern dan penyempurnaan algoritmik yang terkait dengan eksperimen yang lebih baik.

Salah satu sifat praktis dari sebagian besar model jaringan saraf adalah bahwa sebagian besar beban komputasi yang berat dimuat di awal selama fase pelatihan, dan fase prediksi sering kali efisien secara komputasi, karena memerlukan sejumlah kecil operasi (tergantung pada jumlah lapisan). Ini penting karena fase prediksi sering kali jauh lebih kritis terhadap waktu dibandingkan dengan fase pelatihan. Misalnya, jauh lebih penting untuk mengklasifikasikan gambar secara real time (dengan model yang telah dibuat sebelumnya), meskipun pembangunan model yang sebenarnya mungkin memerlukan waktu beberapa minggu untuk jutaan gambar. Metode juga telah dirancang untuk mengompresi jaringan yang telah dilatih agar dapat digunakan dalam pengaturan seluler dan terbatas ruang. Masalah ini dibahas dalam Bab 3.

1.5 Rahasia Kekuatan Komposisi Fungsi

Meskipun metafora biologis terdengar seperti cara yang menarik untuk secara intuitif membenarkan kekuatan komputasional jaringan saraf, metafora tersebut tidak memberikan gambaran lengkap tentang pengaturan di mana jaringan saraf bekerja dengan baik. Pada tingkat paling dasar, jaringan saraf adalah grafik komputasional yang melakukan komposisi fungsi yang lebih sederhana untuk memberikan fungsi yang lebih kompleks. Sebagian besar kekuatan pembelajaran mendalam muncul dari fakta bahwa komposisi berulang dari beberapa fungsi nonlinier memiliki kekuatan ekspresif yang signifikan. Meskipun karya dalam [208] menunjukkan bahwa komposisi tunggal dari sejumlah besar fungsi squashing dapat memperkirakan hampir semua fungsi, pendekatan ini akan memerlukan jumlah unit yang sangat besar (yaitu, parameter) jaringan. Hal ini meningkatkan kapasitas jaringan, yang menyebabkan overfitting kecuali kumpulan data sangat besar. Sebagian besar kekuatan pembelajaran mendalam muncul dari fakta bahwa komposisi berulang dari jenis fungsi tertentu meningkatkan kekuatan representasi jaringan, dan karena itu mengurangi ruang parameter yang diperlukan untuk pembelajaran.

Tidak semua fungsi dasar sama baiknya dalam mencapai tujuan ini. Faktanya, fungsi squashing nonlinier yang digunakan dalam jaringan saraf tidak dipilih secara sembarangan, tetapi dirancang dengan cermat karena jenis properti tertentu. Misalnya, bayangkan situasi di mana fungsi aktivasi identitas digunakan di setiap lapisan, sehingga hanya fungsi linier yang dihitung. Dalam kasus seperti itu, jaringan saraf yang dihasilkan tidak lebih kuat dari jaringan linier satu lapisan:

Teorema 1.5.1 Jaringan multilapis yang hanya menggunakan fungsi aktivasi identitas di semua lapisannya direduksi menjadi jaringan satu lapis yang melakukan regresi linier.

Bukti: Pertimbangkan jaringan yang berisi k lapisan tersembunyi, dan karenanya berisi total (k + 1) lapisan komputasional (termasuk lapisan keluaran). Matriks bobot (k + 1) yang sesuai antara lapisan yang berurutan dilambangkan dengan W1 ... Wk + 1. Misalkan x adalah vektor kolom d-dimensi yang sesuai dengan masukan, h1 ... hk adalah vektor kolom yang sesuai dengan lapisan tersembunyi, dan o adalah vektor kolom m-dimensi yang sesuai dengan keluaran.

Kemudian, kita memiliki kondisi pengulangan berikut untuk jaringan multi-lapis:

$$\overline{h1} = \ddot{y}(WT\ 1\ x) = \overline{WT\ hp+1}\ 1\ \overline{X}$$
$$= \ddot{y}(WT\ p+1hp) = \overline{WT\ p+1hp}\ \ddot{y}p\ \ddot{y}\ \{1\ ...k\ \ddot{y}\ 1\}\ o = \ddot{y}(WT\ k+1hk)$$
$$= WT\ k+1hk$$

Dalam semua kasus di atas, fungsi aktivasi ÿ(·) telah ditetapkan ke fungsi identitas.
Kemudian, dengan menghilangkan variabel lapisan tersembunyi, mudah untuk menunjukkan hal berikut:

$$\overline{o} = WT\ k+1WT\ k\ ...WT = \quad 1\ \overline{X}$$
$$(W1W2\ ...Wk+1) \qquad Tx$$

Perhatikan bahwa seseorang dapat mengganti matriks W1W2 ...Wk+1 dengan matriks d × m baru Wxo, dan mempelajari koefisien Wxo sebagai ganti semua matriks W1, W2 ...Wk+1, tanpa kehilangan ekspresivitas. Dengan kata lain, kita memiliki yang berikut:

$$\overline{o} = WT\ xox$$

Namun, kondisi ini persis sama dengan regresi linier dengan beberapa keluaran [6]. Faktanya, mempelajari matriks redundan W1 ...Wk+1 alih-alih Wxo adalah ide yang buruk, karena hal itu akan meningkatkan jumlah parameter yang harus dipelajari tanpa meningkatkan kekuatan model dengan cara apa pun. Oleh karena itu, jaringan saraf multilapis dengan aktivasi identitas tidak akan memperoleh keuntungan dari jaringan satu lapis dalam hal ekspresivitas. ■

Hasil yang disebutkan di atas adalah untuk kasus pemodelan regresi dengan variabel target numerik. Hasil serupa berlaku untuk variabel target biner. Dalam kasus khusus, di mana semua lapisan menggunakan aktivasi identitas dan lapisan terakhir menggunakan satu keluaran dengan aktivasi tanda untuk prediksi, jaringan saraf multilapis direduksi menjadi perceptron.

Lemma 1.5.1 Pertimbangkan jaringan multilapis di mana semua lapisan tersembunyi menggunakan aktivasi identitas dan simpul keluaran tunggal menggunakan kriteria perceptron sebagai fungsi kerugian dan aktivasi tanda untuk prediksi. Jaringan saraf ini direduksi menjadi perceptron satu lapis.

Bukti hasil ini hampir identik dengan yang dibahas di atas. Faktanya, selama lapisan tersembunyi bersifat linier, tidak ada yang diperoleh dengan menggunakan lapisan tambahan.

Hasil ini menunjukkan bahwa jaringan dalam sebagian besar masuk akal hanya ketika fungsi aktivasi di lapisan antara bersifat non-linier. Biasanya, fungsi seperti sigmoid dan tanh adalah fungsi squashing di mana output dibatasi dalam suatu interval, dan gradiennya adalah nilai mendekati nol terbesar. Untuk nilai absolut yang besar dari argumennya, fungsi-fungsi ini dikatakan mencapai saturasi di mana peningkatan nilai absolut argumen lebih lanjut tidak mengubah nilainya secara signifikan. Jenis fungsi ini di mana nilai-nilai tidak bervariasi secara signifikan pada nilai absolut yang besar dari argumennya dibagikan oleh keluarga fungsi lain, yang disebut sebagai kernel Gaussian, yang umumnya digunakan dalam estimasi kepadatan non-parametrik:

$$\text{Rumus untuk } \ddot{y}(v) = \exp(\ddot{y}v2/2) \tag{1.34}$$

Satu-satunya perbedaan adalah bahwa kernel Gaussian jenuh ke 0 pada nilai besar argumennya, sedangkan fungsi seperti sigmoid dan tanh juga dapat jenuh ke nilai +1 dan ÿ1. Sudah diketahui dalam literatur tentang estimasi densitas [451] bahwa jumlah banyak kernel Gaussian kecil dapat digunakan untuk memperkirakan fungsi densitas apa pun. Fungsi densitas memiliki fungsi khusus

Struktur nonnegatif di mana ekstrem dari distribusi data selalu jenuh ke kepadatan nol, dan oleh karena itu kernel yang mendasarinya juga menunjukkan perilaku yang sama. Prinsip serupa berlaku (lebih umum) untuk fungsi squashing di mana kombinasi linier dari banyak fungsi aktivasi kecil dapat digunakan untuk memperkirakan fungsi arbitrer; namun, fungsi squashing tidak jenuh ke nol untuk menangani perilaku arbitrer pada nilai ekstrem. Hasil perkiraan universal dari jaringan saraf [208] menyatakan bahwa kombinasi linier unit sigmoid (dan/atau sebagian besar fungsi squashing wajar lainnya) dalam satu lapisan tersembunyi dapat digunakan untuk memperkirakan fungsi apa pun dengan baik. Perhatikan bahwa kombinasi linier dapat dilakukan oleh satu simpul keluaran. Oleh karena itu, jaringan dua lapisan cukup selama jumlah unit tersembunyi cukup besar. Namun, beberapa jenis non-linieritas dasar dalam fungsi aktivasi selalu diperlukan untuk memodelkan belokan dan puntiran dalam fungsi arbitrer. Untuk memahami hal ini, perhatikan bahwa semua fungsi 1 dimensi dapat didekati sebagai jumlah fungsi langkah yang diskalakan/ditranslasikan dan sebagian besar fungsi aktivasi yang dibahas dalam bab ini (misalnya, sigmoid) tampak sangat mirip dengan fungsi langkah (lihat Gambar 1.8). Ide dasar ini adalah esensi dari teorema aproksimasi universal jaringan saraf. Faktanya, pembuktian kemampuan fungsi squashing untuk mengaproksimasi fungsi apa pun secara konseptual mirip dengan kernel setidaknya pada tingkat intuitif. Namun, jumlah fungsi dasar yang diperlukan untuk mencapai tingkat aproksimasi yang tinggi dapat sangat besar dalam kedua kasus, yang berpotensi meningkatkan persyaratan yang berpusat pada data ke tingkat yang tidak dapat dikelola. Karena alasan ini, jaringan dangkal menghadapi masalah overfitting yang terus-menerus. Teorema aproksimasi universal menegaskan kemampuan untuk mengaproksimasi fungsi yang tersirat dalam data pelatihan dengan baik, tetapi tidak memberikan jaminan tentang apakah fungsi tersebut dapat digeneralisasi ke data uji yang tidak terlihat.

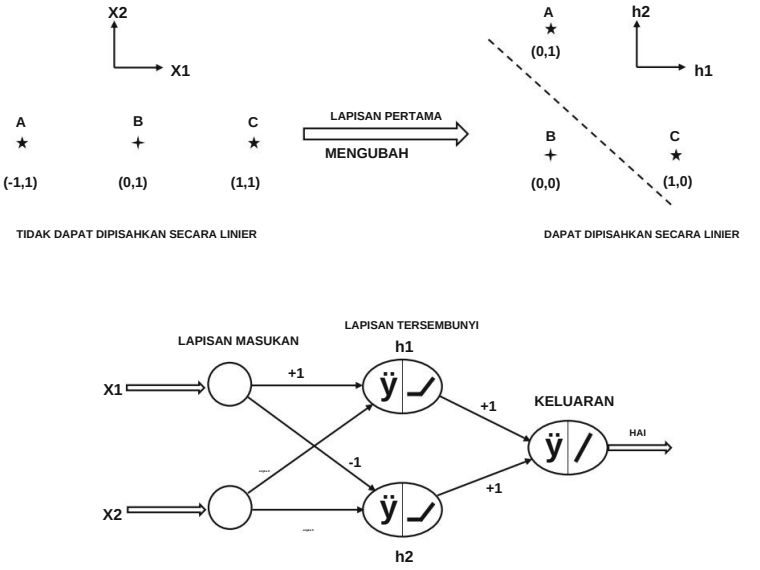
1.5.1 Pentingnya Aktivasi Nonlinier

Bagian sebelumnya memberikan bukti konkret dari fakta bahwa jaringan saraf dengan hanya aktivasi linier tidak memperoleh keuntungan dari peningkatan jumlah lapisan di dalamnya. Misalnya, perhatikan set data dua kelas yang diilustrasikan dalam Gambar 1.14, yang direpresentasikan dalam dua dimensi yang dilambangkan oleh x1 dan x2. Ada dua contoh, A dan B, dari kelas yang dilambangkan oleh '*' dengan koordinat (1, 1) dan (ÿ1, 1), masing-masing. Ada juga satu contoh B dari kelas yang dilambangkan oleh '+' dengan koordinat (0, 1). Jaringan saraf dengan hanya aktivasi linier tidak akan pernah dapat mengklasifikasikan data pelatihan dengan sempurna karena titik-titiknya tidak dapat dipisahkan secara linier.

Di sisi lain, pertimbangkan situasi di mana unit tersembunyi memiliki aktivasi ReLU, dan mereka mempelajari dua fitur baru h1 dan h2, yaitu sebagai berikut:

$$h1 = \text{maks}\{x1, 0\}$$
$$h2 = \text{maks}\{ÿx1, 0\}$$

Perhatikan bahwa tujuan ini dapat dicapai dengan menggunakan bobot yang sesuai dari masukan ke lapisan tersembunyi, dan juga menerapkan unit aktivasi ReLU. Yang terakhir mencapai tujuan ambang batas nilai negatif ke 0. Kami telah menunjukkan bobot yang sesuai dalam jaringan saraf yang ditunjukkan pada Gambar 1.14. Kami telah menunjukkan plot data dalam hal h1 dan h2 pada gambar yang sama. Koordinat dari tiga titik dalam lapisan tersembunyi 2 dimensi adalah {(1, 0),(0, 1),(0, 0)}. Segera terlihat jelas bahwa kedua kelas menjadi dapat dipisahkan secara linier dalam hal representasi tersembunyi yang baru. Dalam arti tertentu, tugas lapisan pertama adalah pembelajaran representasi untuk memungkinkan penyelesaian masalah dengan pengklasifikasi linier.
Oleh karena itu, jika kita menambahkan satu lapisan keluaran linier ke jaringan saraf, maka akan dapat

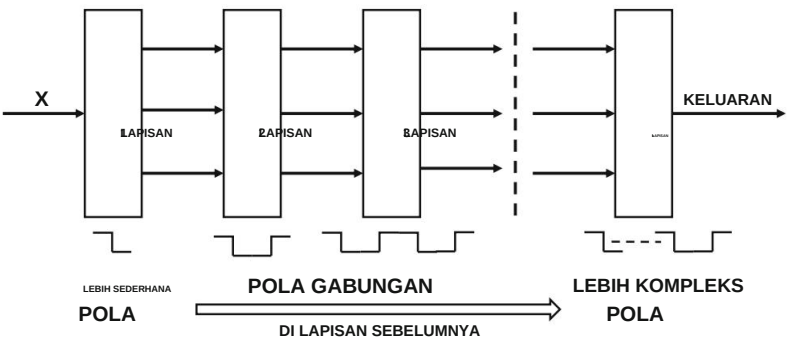


Gambar 1.14: Kekuatan fungsi aktivasi nonlinier dalam mengubah kumpulan data menjadi linier
sifat dpt dipisahkan

mengklasifikasikan contoh pelatihan ini dengan sempurna. Poin pentingnya adalah penggunaan nonlinier
Fungsi ReLU sangat penting dalam memastikan pemisahan linier ini. Fungsi aktivasi memungkinkan
pemetaan data nonlinier, sehingga titik-titik yang tertanam dapat dipisahkan secara linier.
Faktanya, jika kedua bobot dari lapisan tersembunyi ke lapisan keluaran diatur ke 1 dengan aktivasi linier
fungsi, output O akan didefinisikan sebagai berikut:

$$H1 = H2 \tag{1.35}$$

Fungsi linier sederhana ini memisahkan dua kelas karena selalu mengambil nilai
dari 1 untuk dua titik yang diberi label '*' dan mengambil 0 untuk titik yang diberi label '+'. Oleh karena itu,
sebagian besar kekuatan jaringan saraf tersembunyi dalam penggunaan fungsi aktivasi.
bobot yang ditunjukkan pada Gambar 1.14 perlu dipelajari dengan cara berbasis data, meskipun



Gambar 1.15: Jaringan yang lebih dalam dapat mempelajari fungsi yang lebih kompleks dengan menyusun fungsi
dipelajari pada lapisan sebelumnya.

ada banyak pilihan bobot alternatif yang dapat membuat representasi tersembunyi dapat dipisahkan secara linier. Oleh karena itu, bobot yang dipelajari mungkin berbeda dari yang ditunjukkan pada Gambar 1.14 jika pelatihan aktual dilakukan. Namun demikian, dalam kasus perceptron, tidak ada pilihan bobot yang dapat diharapkan untuk mengklasifikasikan set data pelatihan ini dengan sempurna karena set data tidak dapat dipisahkan secara linier di ruang asli. Dengan kata lain, fungsi aktivasi memungkinkan transformasi data nonlinier, yang menjadi semakin kuat dengan beberapa lapisan. Urutan aktivasi nonlinier memaksakan jenis struktur tertentu pada model yang dipelajari, yang kekuatannya meningkat seiring dengan kedalaman urutan (yaitu, jumlah lapisan dalam jaringan saraf).

Contoh klasik lainnya adalah fungsi XOR di mana dua titik {(0, 0),(1, 1)} termasuk dalam satu kelas, dan dua titik lainnya {(1, 0),(0, 1)} termasuk dalam kelas yang lain.
Aktivasi ReLU juga dapat digunakan untuk memisahkan kedua kelas ini, meskipun neuron bias akan dibutuhkan dalam kasus ini (lihat Latihan 1). Makalah backpropagation asli [409] membahas fungsi XOR, karena fungsi ini merupakan salah satu faktor pendorong untuk merancang jaringan multilayer dan kemampuan untuk melatihnya. Fungsi XOR dianggap sebagai uji lakmus untuk menentukan kelayakan dasar dari keluarga jaringan saraf tertentu untuk memprediksi kelas yang dapat dipisahkan secara nonlinier dengan tepat. Meskipun kami telah menggunakan fungsi aktivasi ReLU di atas demi kesederhanaan, sebagian besar fungsi aktivasi nonlinier lainnya dapat digunakan untuk mencapai tujuan yang sama.

1.5.2 Mengurangi Persyaratan Parameter dengan Kedalaman

Ide dasar pembelajaran mendalam adalah bahwa komposisi fungsi yang berulang sering kali dapat mengurangi persyaratan jumlah fungsi dasar (unit komputasi) dengan faktor yang secara eksponensial terkait dengan jumlah lapisan dalam jaringan. Oleh karena itu, meskipun jumlah lapisan dalam jaringan meningkat, jumlah parameter yang diperlukan untuk memperkirakan fungsi yang sama berkurang drastis. Hal ini meningkatkan daya generalisasi jaringan.

Gagasan di balik arsitektur yang lebih dalam adalah bahwa mereka dapat lebih baik memanfaatkan keteraturan berulang dalam pola data untuk mengurangi jumlah unit komputasi dan dengan demikian menggeneralisasi pembelajaran bahkan ke area ruang data di mana seseorang tidak memiliki contoh. Seringkali keteraturan yang berulang ini dipelajari oleh jaringan saraf dalam bobot sebagai vektor basis fitur hierarkis. Meskipun bukti terperinci [340] dari fakta ini berada di luar cakupan buku ini, kami memberikan contoh sederhana untuk menjelaskan hal ini. Pertimbangkan situasi di mana fungsi 1 dimensi didefinisikan oleh 1024 langkah berulang dengan ukuran dan tinggi yang sama. Jaringan dangkal dengan satu lapisan tersembunyi dan fungsi aktivasi langkah akan memerlukan setidaknya 1024 unit untuk memodelkan fungsi tersebut. Namun, jaringan multilapis akan memodelkan pola 1 langkah di lapisan pertama, 2 langkah di lapisan berikutnya, 4 langkah di lapisan ketiga, dan 2r langkah di lapisan ke-r. Situasi ini diilustrasikan dalam Gambar 1.15. Perhatikan bahwa pola 1 langkah adalah fitur paling sederhana karena diulang 1024 kali, sedangkan pola 2 langkah lebih kompleks. Oleh karena itu, fitur (dan fungsi yang dipelajari) di lapisan yang berurutan terkait secara hierarkis. Dalam kasus ini, total 10 lapisan diperlukan dan sejumlah kecil simpul konstan diperlukan di setiap lapisan untuk memodelkan penggabungan dua pola dari lapisan sebelumnya.

Cara lain untuk memahami hal ini adalah sebagai berikut. Pertimbangkan fungsi 1 dimensi yang mengambil satu nilai 1 dan ÿ1 dalam interval alternatif, dan nilai ini berganti 1024 kali pada interval reguler argumen. Satu-satunya cara untuk mensimulasikan fungsi ini dengan kombinasi linear fungsi aktivasi langkah (yang hanya berisi satu pergantian nilai) adalah dengan menggunakan 1024 di antaranya (atau faktor konstan kecil dari angka ini). Namun, jaringan saraf dengan 10 lapisan tersembunyi dan hanya 2 unit per lapisan memiliki 210 = 1024 jalur dari sumber

ke output. Selama fungsi yang akan dipelajari bersifat regular dalam beberapa hal, seringkali mungkin untuk mempelajari parameter untuk lapisan sehingga 1024 jalur ini mampu menangkap kompleksitas 1024 nilai yang berbeda dalam fungsi. Lapisan sebelumnya mempelajari lebih lanjut pola-pola yang terperinci, sedangkan lapisan-lapisan selanjutnya mempelajari pola-pola tingkat tinggi. Oleh karena itu, keseluruhan Jumlah node yang dibutuhkan adalah satu orde lebih kecil dari yang dibutuhkan dalam jaringan satu lapis. Ini berarti bahwa jumlah data yang dibutuhkan untuk pembelajaran juga merupakan satu orde lebih kecil. besarnya lebih kecil. Alasannya adalah karena jaringan multilayer secara implisit mencari keteraturan yang berulang dan mempelajarinya dengan lebih sedikit data, daripada mencoba mempelajarinya secara eksplisit setiap belokan dan putaran fungsi target. Saat menggunakan jaringan saraf konvolusional dengan data gambar, perilaku ini menjadi jelas secara intuitif di mana lapisan sebelumnya memodelkan sederhana fitur seperti garis, lapisan tengah mungkin memodelkan bentuk dasar, dan lapisan selanjutnya mungkin memodelkan bentuk yang rumit seperti wajah. Di sisi lain, satu lapisan akan sulit dalam memodelkan setiap liku-liku wajah. Ini memberikan model yang lebih dalam dengan lebih baik kekuatan generalisasi dan juga kemampuan untuk belajar dengan lebih sedikit data.

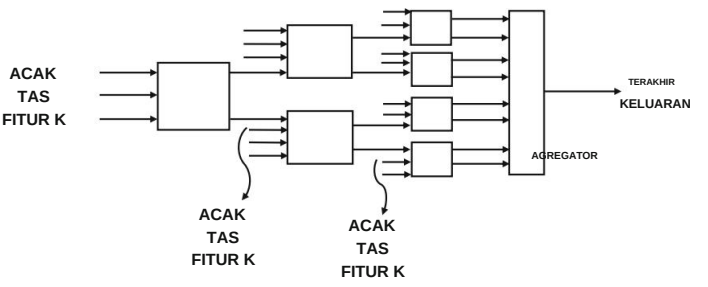
Namun, meningkatkan kedalaman jaringan bukan tanpa kekurangan. Jaringan seringkali lebih sulit untuk dilatih, dan mereka menunjukkan semua jenis perilaku tidak stabil seperti masalah gradien yang menghilang dan meledak. Jaringan dalam juga terkenal tidak stabil untuk pemilihan parameter. Masalah-masalah ini sering diatasi dengan desain fungsi yang cermat dihitung dalam node, serta penggunaan prosedur pra-pelatihan untuk meningkatkan kinerja.

1.5.3 Arsitektur Neural Non-Konvensional

Pembahasan yang disebutkan di atas memberikan gambaran umum mengenai cara-cara yang paling umum dilakukan operasi dan struktur jaringan saraf khas dibangun. Namun, ada banyak variasi dari tema umum ini. Berikut ini akan dibahas beberapa variasi tersebut.

1.5.3.1 Mengaburkan Perbedaan Antara Input, Tersembunyi, dan Lapisan Keluaran

Secara umum, terdapat penekanan yang kuat pada jaringan umpan maju berlapis-lapis dalam domain jaringan saraf dengan pengaturan berurutan antara lapisan masukan, tersembunyi, dan keluaran. dengan kata lain, semua node input masuk ke lapisan tersembunyi pertama, lapisan tersembunyi berikutnya saling memberi masukan, dan lapisan tersembunyi terakhir memberi masukan ke lapisan keluaran.



Gambar 1.16: Contoh arsitektur tidak konvensional di mana input terjadi pada lapisan selain lapisan tersembunyi pertama. Selama jaringan saraf bersifat asiklik (atau dapat diubah menjadi representasi asiklik), bobot grafik komputasi yang mendasarinya dapat dipelajari menggunakan pemrograman dinamis (backpropagation).

Unit-unit tasional sering didefinisikan dengan fungsi-fungsi squashing yang diterapkan pada kombinasi linear input. Lapisan tersembunyi umumnya tidak mengambil input, dan kerugian umumnya tidak dihitung atas nilai-nilai dalam lapisan tersembunyi. Karena fokus ini, mudah untuk melupakan bahwa jaringan saraf dapat didefinisikan sebagai jenis grafik komputasi berparameter apa pun, di mana pembatasan ini tidak diperlukan agar algoritma backpropagation berfungsi. Secara umum, dimungkinkan untuk memiliki komputasi input dan kerugian di lapisan perantara, meskipun ini kurang umum. Misalnya, jaringan saraf diusulkan dalam [515] yang terinspirasi oleh gagasan hutan acak [49], dan memungkinkan input di berbagai lapisan jaringan. Contoh dari jenis jaringan ini ditunjukkan pada Gambar 1.16. Dalam hal ini, jelas bahwa perbedaan antara lapisan input dan lapisan tersembunyi telah kabur.

Dalam variasi lain dari arsitektur umpan maju dasar, fungsi kerugian dihitung tidak hanya pada simpul keluaran, tetapi juga pada simpul tersembunyi. Kontribusi pada simpul tersembunyi sering kali dalam bentuk penalti yang bertindak sebagai pengatur. Misalnya, jenis metode ini digunakan untuk melakukan pembelajaran fitur jarang dengan mengenakan penalti pada simpul tersembunyi (lih. Bab 2 dan 4). Dalam kasus ini, perbedaan antara lapisan tersembunyi dan lapisan keluaran menjadi kabur.

Contoh lain pilihan desain terkini adalah penggunaan koneksi lewati [184] di mana masukan dari lapisan tertentu diizinkan untuk terhubung ke lapisan di luar lapisan berikutnya. Jenis pendekatan ini mengarah ke model yang benar-benar mendalam. Misalnya, arsitektur 152 lapisan, disebut sebagai ResNet [184], telah mencapai kinerja tingkat manusia dalam tugas pengenalan gambar. Meskipun arsitektur ini tidak mengaburkan perbedaan antara lapisan masukan, tersembunyi, dan keluaran, strukturnya berbeda dari jaringan umpan maju tradisional di mana koneksi hanya ditempatkan di antara lapisan yang berurutan. Jaringan ini memiliki pandangan iteratif tentang rekayasa fitur [161], di mana fitur-fitur di lapisan selanjutnya adalah penyempurnaan iteratif dari fitur-fitur di lapisan sebelumnya. Sebaliknya, pendekatan tradisional untuk rekayasa fitur bersifat hierarkis, di mana fitur-fitur di lapisan selanjutnya adalah representasi yang semakin abstrak yang diperoleh dari fitur-fitur di lapisan sebelumnya.

1.5.3.2 Operasi Non-Konvensional dan Jaringan Jumlah Produk

Beberapa jaringan saraf seperti memori jangka pendek dan jaringan saraf konvolusional mendefinisikan berbagai jenis operasi "melupakan" perkalian, konvolusi, dan penggabungan antara variabel yang tidak sepenuhnya dalam bentuk yang dibahas dalam bab ini. Faktanya, arsitektur ini sekarang digunakan secara luas dalam domain teks dan gambar sehingga tidak lagi dianggap tidak biasa.

Jenis arsitektur unik lainnya adalah jaringan jumlah-produk [383]. Dalam kasus ini, simpulnya adalah simpul penjumlahan atau simpul produk. Simpul penjumlahan mirip dengan transformasi linier tradisional dengan serangkaian tepi berbobot. Namun, bobotnya dibatasi menjadi positif. Simpul produk hanya mengalikan inputnya tanpa perlu bobot. Perlu dicatat bahwa ada banyak variasi dalam hal bagaimana produk dapat dihitung. Misalnya, jika inputnya adalah dua skalar, maka orang dapat menghitung produknya. Jika inputnya adalah dua vektor dengan panjang yang sama, orang dapat menghitung produk per elemennya. Beberapa pustaka pembelajaran mendalam mendukung jenis operasi produk ini. Wajar jika lapisan penjumlahan dan lapisan produk bergantian untuk memaksimalkan ekspresivitas.

Jaringan jumlah-produk cukup ekspresif, dan seringkali memungkinkan untuk membangun variasi mendalam dengan tingkat ekspresivitas yang tinggi [30, 93]. Poin penting adalah bahwa hampir semua fungsi matematika dapat ditulis secara kira-kira sebagai fungsi polinomial dari inputnya. Oleh karena itu, hampir semua fungsi dapat diekspresikan menggunakan arsitektur jumlah-produk, meskipun lebih dalam

Arsitektur memungkinkan pemodelan dengan struktur yang lebih baik. Tidak seperti jaringan saraf tradisional di mana nonlinieritas digabungkan dengan fungsi aktivasi, operasi produk adalah kunci nonlinieritas dalam jaringan jumlah produk.

Masalah Pelatihan

Sering kali membantu untuk bersikap fleksibel dalam menggunakan berbagai jenis operasi komputasi dalam node di luar transformasi dan fungsi aktivasi yang diketahui. Lebih jauh, koneksi antara node tidak perlu terstruktur dalam mode berlapis-lapis dan node dalam lapisan tersembunyi dapat disertakan dalam komputasi kerugian. Selama grafik komputasi yang mendasarinya bersifat asiklik, mudah untuk menggeneralisasi algoritma backpropagation ke semua jenis arsitektur dan operasi komputasi. Bagaimanapun, algoritma pemrograman dinamis (seperti backpropagation) dapat digunakan pada hampir semua jenis grafik asiklik terarah di mana beberapa node dapat digunakan untuk menginisialisasi rekursi pemrograman dinamis. Penting untuk diingat bahwa arsitektur yang dirancang dengan pemahaman khusus domain yang tepat sering kali dapat memberikan hasil yang lebih unggul daripada metode kotak hitam yang menggunakan jaringan umpan-maju yang terhubung sepenuhnya.

1.6 Arsitektur Neural Umum

Ada beberapa jenis arsitektur neural yang umum digunakan dalam berbagai aplikasi pembelajaran mesin. Bagian ini akan memberikan gambaran singkat mengenai beberapa arsitektur ini, yang akan dibahas lebih rinci dalam bab-bab selanjutnya.

1.6.1 Simulasi Pembelajaran Mesin Dasar dengan Model Dangkal

Sebagian besar model pembelajaran mesin dasar seperti regresi linier, klasifikasi, mesin vektor pendukung, regresi logistik, dekomposisi nilai singular, dan faktorisasi matriks dapat disimulasikan dengan jaringan saraf dangkal yang berisi tidak lebih dari satu atau dua lapisan.
Sangatlah bermanfaat untuk mengeksplorasi arsitektur dasar ini, karena secara tidak langsung hal ini menunjukkan kekuatan jaringan saraf; sebagian besar dari apa yang kita ketahui tentang pembelajaran mesin dapat disimulasikan dengan model yang relatif sederhana! Lebih jauh lagi, banyak model jaringan saraf dasar seperti model pembelajaran Widrow-Hoff secara langsung terkait dengan model pembelajaran mesin tradisional seperti diskriminan Fisher, meskipun model-model tersebut diusulkan secara independen. Pengamatan yang perlu diperhatikan adalah bahwa arsitektur yang lebih dalam sering kali dibuat dengan menumpuk model-model yang lebih sederhana ini dengan cara yang kreatif. Arsitektur saraf untuk model pembelajaran mesin dasar dibahas dalam Bab 2. Sejumlah aplikasi untuk penambangan teks, grafik, dan sistem rekomendasi juga akan dibahas dalam bab ini.

1.6.2 Jaringan Fungsi Basis Radial

Jaringan fungsi basis radial (RBF) merupakan arsitektur yang terlupakan dari sejarah panjang jaringan neural. Jaringan ini tidak umum digunakan di era modern, meskipun memiliki potensi signifikan untuk jenis masalah tertentu. Salah satu masalah yang membatasi adalah jaringan ini tidak mendalam, dan biasanya hanya menggunakan dua lapisan. Lapisan pertama dibangun dengan cara tanpa pengawasan, sedangkan lapisan kedua dilatih menggunakan metode yang diawasi. Jaringan ini pada dasarnya berbeda dari jaringan feed-forward, dan memperoleh kekuatannya dari jumlah node yang lebih banyak di lapisan tanpa pengawasan. Prinsip dasar penggunaan jaringan RBF pada dasarnya sangat berbeda dari jaringan feed-forward, dalam arti

bahwa yang pertama memperoleh kekuatannya dari perluasan ukuran ruang fitur daripada kedalaman. Pendekatan ini didasarkan pada teorema Cover tentang keterpisahan pola [84], yang menyatakan bahwa masalah klasifikasi pola lebih mungkin dapat dipisahkan secara linier ketika dilemparkan ke dalam ruang berdimensi tinggi dengan transformasi nonlinier. Lapisan kedua jaringan berisi prototipe di setiap simpul dan aktivasi ditentukan oleh kesamaan data masukan dengan prototipe. Aktivasi ini kemudian digabungkan dengan bobot terlatih dari lapisan berikutnya untuk membuat prediksi akhir. Pendekatan ini sangat mirip dengan pengklasifikasi tetangga terdekat, kecuali bahwa bobot di lapisan kedua memberikan tingkat pengawasan tambahan. Dengan kata lain, pendekatan ini adalah metode tetangga terdekat yang diawasi.

Khususnya, mesin vektor pendukung diketahui sebagai varian terbimbing dari pengklasifikasi tetangga terdekat di mana fungsi kernel digabungkan dengan bobot terbimbing untuk memberi bobot pada titik-titik tetangga dalam prediksi akhir [6]. Jaringan fungsi basis radial dapat digunakan untuk mensimulasikan metode kernel seperti mesin vektor pendukung. Untuk jenis masalah tertentu seperti klasifikasi, seseorang dapat menggunakan arsitektur ini secara lebih efektif daripada mesin vektor pendukung kernel yang siap pakai. Hal ini karena model ini lebih umum, menyediakan lebih banyak peluang untuk eksperimen daripada mesin vektor pendukung kernel. Lebih jauh lagi, terkadang dimungkinkan untuk memperoleh beberapa keuntungan dari peningkatan kedalaman pada lapisan terbimbing. Potensi penuh jaringan fungsi basis radial masih belum dieksplorasi dalam literatur, karena arsitektur ini sebagian besar telah dilupakan dengan meningkatnya fokus pada metode umpan maju yang biasa saja. Pembahasan tentang jaringan fungsi basis radial disediakan dalam Bab 5.

1.6.3 Mesin Boltzmann Terbatas

Mesin Boltzmann terbatas (RBM) menggunakan gagasan minimisasi energi untuk membuat arsitektur jaringan saraf guna memodelkan data tanpa pengawasan. Metode ini khususnya berguna untuk membuat model generatif data, dan terkait erat dengan model grafis probabilistik [251]. Mesin Boltzmann terbatas berasal dari penggunaan jaringan Hopfield [207], yang dapat digunakan untuk menyimpan memori.

Varian stokastik jaringan ini digeneralisasikan ke mesin Boltzmann, di mana lapisan tersembunyi memodelkan aspek generatif data.

Mesin Boltzmann terbatas sering digunakan untuk pemodelan tanpa pengawasan dan pengurangan dimensionalitas, meskipun mesin ini juga dapat digunakan untuk pemodelan dengan pengawasan. Akan tetapi, karena mesin ini tidak cocok untuk pemodelan dengan pengawasan, pelatihan dengan pengawasan sering didahului oleh fase tanpa pengawasan. Hal ini secara alami mengarah pada penemuan gagasan prapelatihan, yang ditemukan sangat bermanfaat untuk pembelajaran dengan pengawasan. RBM merupakan salah satu model pertama yang digunakan untuk pembelajaran mendalam, terutama dalam pengaturan tanpa pengawasan. Pendekatan prapelatihan akhirnya diadopsi oleh jenis model lainnya. Oleh karena itu, RBM juga memiliki signifikansi historis dalam hal memotivasi beberapa metodologi pelatihan untuk model mendalam.

Proses pelatihan mesin Boltzmann terbatas sangat berbeda dengan jaringan umpan maju. Secara khusus, model ini tidak dapat dilatih menggunakan backpropagation, dan memerlukan pengambilan sampel Monte Carlo untuk melakukan pelatihan. Algoritma khusus yang umum digunakan untuk melatih RBM adalah algoritma divergensi kontrastif.
Pembahasan tentang mesin Boltzmann terbatas disediakan dalam Bab 6.

1.6.4 Jaringan Syaraf Tiruan Berulang

Jaringan saraf berulang dirancang untuk data berurutan seperti kalimat teks, deret waktu, dan urutan diskrit lainnya seperti urutan biologis. Dalam kasus ini, inputnya adalah

bentuk x1 ... xn, di mana xt adalah titik berdimensi d yang diterima pada stempel waktu t. Misalnya, vektor xt mungkin berisi nilai d pada tanda waktu ke-t dari deret waktu multivariat (dengan d deret waktu yang berbeda). Dalam pengaturan teks, vektor xt akan berisi kata yang dikodekan one-hot pada stempel waktu ke-t. Dalam pengodean one-hot, kita memiliki vektor dengan panjang yang sama dengan ukuran leksikon, dan komponen untuk kata yang relevan memiliki nilai 1. Semua komponen lainnya adalah 0.

Poin penting tentang urutan adalah bahwa kata-kata yang berurutan saling bergantung satu sama lain. Oleh karena itu, akan sangat membantu jika menerima input tertentu xt hanya setelah input sebelumnya telah diterima dan diubah menjadi status tersembunyi. Jenis jaringan umpan maju tradisional di mana semua input masuk ke lapisan pertama tidak mencapai tujuan ini.

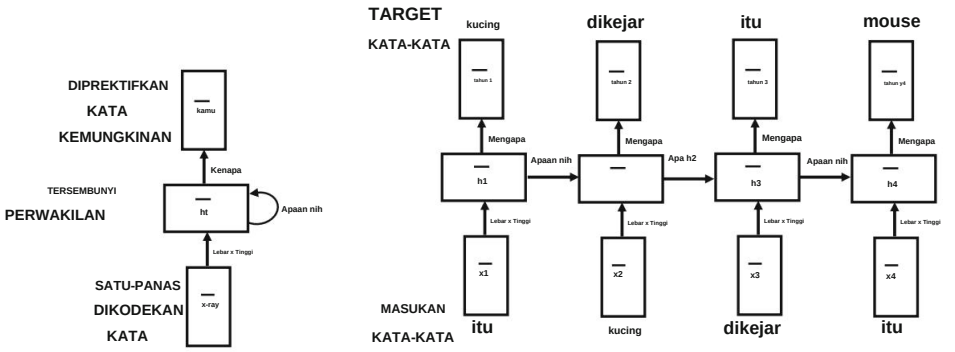
Oleh karena itu, jaringan saraf berulang memungkinkan input xt untuk berinteraksi langsung dengan status tersembunyi yang dibuat dari input pada cap waktu sebelumnya. Arsitektur dasar jaringan saraf berulang diilustrasikan dalam Gambar 1.17(a). Poin utamanya adalah bahwa ada input xt pada setiap cap waktu, dan status tersembunyi ht yang berubah pada setiap cap waktu saat titik data baru tiba. Setiap cap waktu juga memiliki nilai keluaran yt. Misalnya, dalam pengaturan deret waktu, keluaran yt mungkin merupakan prediksi yang diramalkan dari xt+1. Ketika digunakan dalam pengaturan teks untuk memprediksi kata berikutnya, pendekatan ini disebut sebagai pemodelan bahasa. Dalam beberapa aplikasi, kami tidak mengeluarkan yt pada setiap cap waktu, tetapi hanya pada akhir urutan. Misalnya, jika seseorang mencoba mengklasifikasikan sentimen kalimat sebagai "positif" atau "negatif," keluaran hanya akan terjadi pada cap waktu terakhir.

Keadaan tersembunyi pada waktu t diberikan oleh fungsi vektor input pada waktu t dan vektor tersembunyi pada waktu (t ÿ 1):

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{1.36}$$

Fungsi terpisah yt = g(ht) digunakan untuk mempelajari probabilitas keluaran dari status tersembunyi. Perhatikan bahwa fungsi f(·) dan g(·) sama pada setiap cap waktu. Asumsi implisitnya adalah bahwa deret waktu menunjukkan tingkat stasioneritas tertentu; properti yang mendasarinya tidak berubah seiring waktu. Meskipun properti ini tidak sepenuhnya benar dalam pengaturan nyata, ini adalah asumsi yang baik untuk digunakan untuk regularisasi.

Poin penting di sini adalah keberadaan self-loop pada Gambar 1.17(a), yang akan menyebabkan status tersembunyi jaringan saraf berubah setelah input setiap xt. Dalam praktiknya, seseorang hanya bekerja dengan urutan dengan panjang terbatas, dan masuk akal untuk menguraikan loop menjadi jaringan "berlapis waktu" yang lebih mirip jaringan umpan maju. Jaringan ini ditunjukkan pada Gambar 1.17(b). Perhatikan bahwa dalam kasus ini, kita memiliki simpul yang berbeda untuk status tersembunyi



(a) Recurrent neural network (b) Time-layered representation of (a)

Gambar 1.17: Jaringan saraf berulang dan representasinya berdasarkan lapisan waktu

status pada setiap cap waktu dan loop-diri telah diurai menjadi jaringan umpan-maju.
Representasi ini secara matematis setara dengan Gambar 1.17(a), tetapi jauh lebih mudah dipahami karena kemiripannya dengan jaringan tradisional. Perhatikan bahwa tidak seperti jaringan umpan maju tradisional, masukan juga terjadi pada lapisan perantara dalam jaringan yang tidak terurai ini. Matriks bobot koneksi dibagi oleh beberapa koneksi dalam jaringan berlapis waktu untuk memastikan bahwa fungsi yang sama digunakan pada setiap cap waktu. Pembagian ini adalah kunci wawasan khusus domain yang dipelajari oleh jaringan. Algoritme backpropagation memperhitungkan pembagian dan panjang temporal saat memperbarui bobot selama proses pembelajaran. Jenis khusus algoritme backpropagation ini disebut sebagai backpropagation through time (BPTT). Karena sifat rekursif Persamaan 1.36, jaringan berulang memiliki kemampuan untuk menghitung fungsi input dengan panjang variabel. Dengan kata lain, seseorang dapat memperluas rekursif Persamaan 1.36 untuk mendefinisikan fungsi untuk ht dalam hal input t. Misalnya, mulai dari h0, yang biasanya ditetapkan ke beberapa vektor konstan, kita memiliki h1 = f(h0, x1) dan h2 = f(f(h0, x1), x2). Perhatikan bahwa h1 adalah fungsi hanya x1, sedangkan h2 adalah fungsi dari x1 dan x2. Karena output yt merupakan fungsi dari ht, maka properti ini juga diwarisi oleh yt . Secara umum, kita dapat menuliskan yang berikut:

Persamaan (x1, x2, ...) dan (x3, x4) adalah persamaan yang menyatakan hubungan antara x1 dan x2. (1.37)

Perhatikan bahwa fungsi Ft(·) bervariasi dengan nilai t. Pendekatan semacam itu khususnya berguna untuk input dengan panjang variabel seperti kalimat teks. Rincian lebih lanjut tentang jaringan saraf berulang disediakan dalam Bab 7; bab ini juga akan membahas aplikasi jaringan saraf berulang dalam berbagai domain.

Sifat teoritis yang menarik dari jaringan saraf berulang adalah bahwa mereka adalah Turing lengkap [444]. Artinya, dengan data dan sumber daya komputasi yang cukup, jaringan saraf berulang dapat mensimulasikan algoritma apa pun. Namun, dalam praktiknya, sifat teoritis ini tidak berguna karena jaringan berulang memiliki masalah praktis yang signifikan dengan generalisasi untuk urutan yang panjang. Jumlah data dan ukuran status tersembunyi yang diperlukan untuk urutan yang lebih panjang meningkat dengan cara yang tidak realistis. Lebih jauh, ada masalah praktis dalam menemukan pilihan parameter yang optimal karena masalah gradien yang menghilang dan meledak. Akibatnya, varian khusus dari arsitektur jaringan saraf berulang telah diusulkan, seperti penggunaan memori jangka pendek yang panjang. Arsitektur canggih ini juga akan dibahas dalam Bab 7. Lebih jauh, beberapa varian canggih dari arsitektur berulang, seperti mesin Turing saraf, telah menunjukkan peningkatan atas jaringan saraf berulang dalam beberapa aplikasi.

1.6.5 Jaringan Syaraf Konvolusional

Jaringan saraf konvolusional adalah jaringan yang terinspirasi secara biologis yang digunakan dalam visi komputer untuk klasifikasi gambar dan deteksi objek. Motivasi dasar untuk jaringan saraf konvolusional diperoleh dari pemahaman Hubel dan Wiesel [212] tentang cara kerja korteks visual kucing, di mana bagian-bagian tertentu dari bidang visual tampaknya merangsang neuron tertentu. Prinsip yang lebih luas ini digunakan untuk merancang arsitektur yang jarang untuk jaringan saraf konvolusional. Arsitektur dasar pertama yang didasarkan pada inspirasi biologis ini adalah neokognitron, yang kemudian digeneralisasikan ke arsitektur LeNet-5 [279].
Dalam arsitektur jaringan saraf konvolusional, setiap lapisan jaringan adalah 3 dimensi, yang memiliki luas spasial dan kedalaman yang sesuai dengan jumlah fitur. Gagasan kedalaman satu lapisan dalam jaringan saraf konvolusional berbeda4 dari gagasan

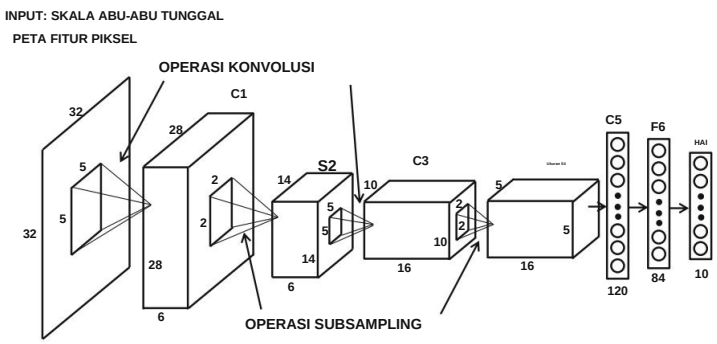
4Ini merupakan kelebihan muatan terminologi yang digunakan dalam jaringan saraf konvolusional. Arti kata “kedalaman” disimpulkan dari konteks penggunaannya.

kedalaman dalam hal jumlah lapisan. Pada lapisan masukan, fitur-fitur ini sesuai dengan saluran warna seperti RGB (yaitu, merah, hijau, biru), dan pada saluran tersembunyi, fitur-fitur ini mewakili peta fitur tersembunyi yang mengodekan berbagai jenis bentuk dalam gambar. Jika masukan dalam skala abu-abu (seperti LeNet-5), maka lapisan masukan akan memiliki kedalaman 1, tetapi lapisan selanjutnya akan tetap 3 dimensi. Arsitektur tersebut berisi dua jenis lapisan, yang masing-masing disebut sebagai lapisan konvolusi dan subsampling.

Untuk lapisan konvolusi, operasi konvolusi didefinisikan, di mana filter digunakan untuk memetakan aktivasi dari satu lapisan ke lapisan berikutnya. Operasi konvolusi menggunakan filter bobot 3 dimensi dengan kedalaman yang sama dengan lapisan saat ini tetapi dengan luas spasial yang lebih kecil. Produk titik antara semua bobot dalam filter dan pilihan wilayah spasial (dengan ukuran yang sama dengan filter) dalam suatu lapisan menentukan nilai status tersembunyi di lapisan berikutnya (setelah menerapkan fungsi aktivasi seperti ReLU). Operasi antara filter dan wilayah spasial dalam suatu lapisan dilakukan pada setiap posisi yang memungkinkan untuk menentukan lapisan berikutnya (di mana aktivasi mempertahankan hubungan spasialnya dari lapisan sebelumnya).

Koneksi dalam jaringan saraf konvolusional sangat jarang, karena setiap aktivasi dalam lapisan tertentu merupakan fungsi dari wilayah spasial kecil di lapisan sebelumnya.
Semua lapisan selain set terakhir dari dua atau tiga lapisan mempertahankan struktur spasialnya. Oleh karena itu, dimungkinkan untuk memvisualisasikan secara spasial bagian gambar mana yang memengaruhi bagian tertentu dari aktivasi dalam satu lapisan. Fitur dalam lapisan tingkat rendah menangkap garis atau bentuk primitif lainnya, sedangkan fitur dalam lapisan tingkat tinggi menangkap bentuk yang lebih kompleks seperti lingkaran (yang umumnya muncul dalam banyak digit). Oleh karena itu, lapisan selanjutnya dapat membuat digit dengan menyusun bentuk dalam fitur intuitif ini. Ini adalah contoh klasik tentang cara wawasan semantik tentang domain data tertentu digunakan untuk merancang arsitektur yang c
Selain itu, lapisan subsampling hanya merata-ratakan nilai-nilai di wilayah lokal berukuran 2×2 untuk mengompresi jejak spasial lapisan-lapisan tersebut dengan faktor 2. Ilustrasi arsitektur LeNet-5 ditunjukkan pada Gambar 1.18. Pada tahun-tahun awal, LeNet-5 digunakan oleh beberapa bank untuk mengenali angka-angka yang ditulis tangan pada cek.

Jaringan saraf konvolusional secara historis merupakan jenis jaringan saraf yang paling sukses. Jaringan ini digunakan secara luas untuk pengenalan gambar, deteksi/lokalisasi objek, dan bahkan pemrosesan teks. Kinerja jaringan ini baru-baru ini melampaui kinerja manusia dalam masalah klasifikasi gambar [184]. Jaringan saraf konvolusional memberikan contoh yang sangat baik tentang fakta bahwa pilihan desain arsitektur dalam jaringan saraf harus dilakukan dengan wawasan semantik tentang domain data yang ada. Dalam kasus tertentu



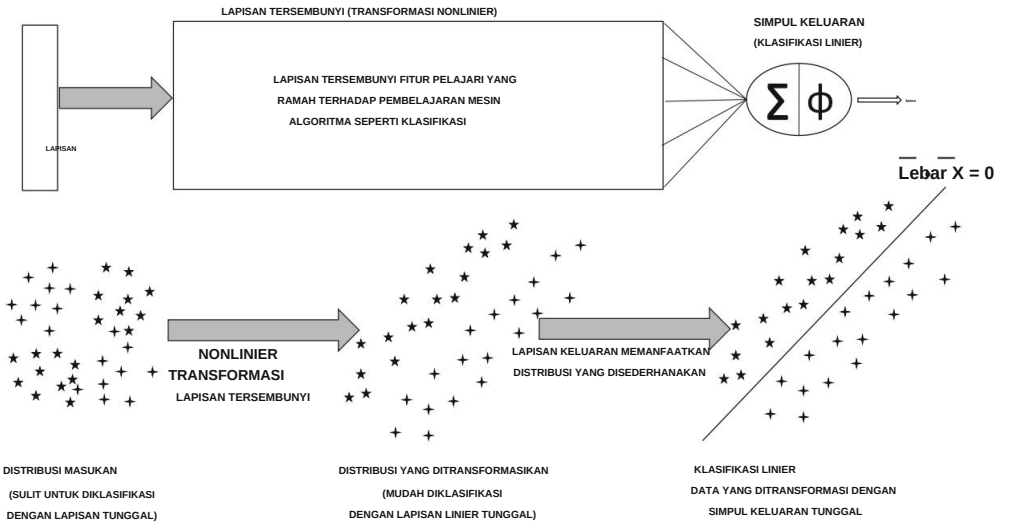
Gambar 1.18: LeNet-5: Salah satu jaringan saraf konvolusional paling awal.

dari jaringan saraf konvolusional, wawasan ini diperoleh dengan mengamati kerja biologis korteks visual kucing, dan banyak menggunakan hubungan spasial antarpiksel.
Fakta ini juga memberikan beberapa bukti bahwa pemahaman lebih lanjut tentang ilmu saraf mungkin juga membantu pengembangan metode dalam kecerdasan buatan.

Jaringan saraf konvolusional yang telah dilatih sebelumnya dari sumber daya yang tersedia untuk umum seperti ImageNet sering kali tersedia untuk digunakan langsung untuk aplikasi dan kumpulan data lainnya. Hal ini dicapai dengan menggunakan sebagian besar bobot yang telah dilatih sebelumnya dalam jaringan konvolusional tanpa perubahan apa pun kecuali untuk lapisan klasifikasi akhir. Bobot lapisan klasifikasi akhir dipelajari dari kumpulan data yang ada. Pelatihan lapisan akhir diperlukan karena label kelas dalam pengaturan tertentu mungkin berbeda dari label kelas ImageNet.
Meskipun demikian, bobot pada lapisan awal masih berguna karena mempelajari berbagai jenis bentuk dalam gambar yang dapat berguna untuk hampir semua jenis aplikasi klasifikasi.
Lebih jauh lagi, aktivasi fitur pada lapisan kedua terakhir bahkan dapat digunakan untuk aplikasi tanpa pengawasan. Misalnya, seseorang dapat membuat representasi multidimensi dari kumpulan data gambar sembarang dengan melewatkan setiap gambar melalui jaringan saraf konvolusional dan mengekstrak aktivasi lapisan kedua terakhir. Selanjutnya, semua jenis pengindeksan dapat diterapkan pada representasi ini untuk mengambil gambar yang mirip dengan gambar target tertentu. Pendekatan semacam itu sering kali memberikan hasil yang sangat baik dalam pengambilan gambar karena sifat semantik fitur yang dipelajari oleh jaringan. Perlu dicatat bahwa penggunaan jaringan konvolusional yang telah dilatih sebelumnya sangat populer sehingga pelatihan jarang dimulai dari awal. Jaringan saraf konvolusional dibahas secara rinci dalam Bab 8.

1.6.6 Rekayasa Fitur Hirarkis dan Model Pra-Terlatih

Banyak arsitektur yang lebih dalam dengan arsitektur feed-forward memiliki beberapa lapisan di mana transformasi input yang berurutan dari lapisan sebelumnya mengarah ke representasi data yang semakin canggih. Nilai setiap lapisan tersembunyi untuk input tertentu berisi representasi titik input yang ditransformasikan, yang menjadi semakin informatif tentang nilai target yang coba kita pelajari, saat lapisan semakin dekat ke simpul output. Seperti yang ditunjukkan di Bagian 1.5.1, representasi fitur yang ditransformasikan dengan tepat lebih sesuai dengan jenis prediksi sederhana di lapisan output. Kecanggihan ini merupakan hasil dari aktivasi nonlinier di lapisan perantara. Secara tradisional, aktivasi sigmoid dan tanh adalah pilihan paling populer di lapisan tersembunyi, tetapi aktivasi ReLU telah menjadi semakin populer dalam beberapa tahun terakhir karena properti yang diinginkan yaitu lebih baik dalam menghindari masalah gradien yang menghilang dan meledak (lih. Bagian 3.4.2 dari Bab 3). Untuk klasifikasi, lapisan akhir dapat dilihat sebagai lapisan prediksi yang relatif sederhana yang berisi satu neuron linier dalam kasus regresi, dan merupakan fungsi sigmoid/tanda dalam kasus klasifikasi biner. Keluaran yang lebih kompleks mungkin memerlukan beberapa simpul. Salah satu cara untuk melihat pembagian kerja antara lapisan tersembunyi dan lapisan prediksi akhir adalah bahwa lapisan awal membuat representasi fitur yang lebih sesuai dengan tugas yang sedang dikerjakan. Lapisan akhir kemudian memanfaatkan representasi fitur yang dipelajari ini. Pembagian kerja ini ditunjukkan pada Gambar 1.19. Poin utamanya adalah bahwa fitur yang dipelajari dalam lapisan tersembunyi sering kali (tetapi tidak selalu) dapat digeneralisasikan ke set data dan pengaturan masalah lain dalam domain yang sama (misalnya, teks, gambar, dan sebagainya). Properti ini dapat dimanfaatkan dengan berbagai cara hanya dengan mengganti simpul keluaran dari jaringan yang telah dilatih sebelumnya dengan lapisan keluaran khusus aplikasi yang berbeda (misalnya, lapisan regresi linier alih-alih lapisan klasifikasi sigmoid) untuk set data dan masalah yang sedang dikerjakan. Selanjutnya, hanya bobot lapisan keluaran baru yang diganti yang mungkin perlu dipelajari untuk kumpulan data dan aplikasi baru, sedangkan bobot lapisan lainnya tetap.



Gambar 1.19: Peran rekayasa fitur dari lapisan tersembunyi

Output dari setiap lapisan tersembunyi adalah representasi fitur data yang telah ditransformasikan, di mana dimensionalitas representasi ditentukan oleh jumlah unit dalam lapisan tersebut. Proses ini dapat dilihat sebagai semacam rekayasa fitur hierarkis di mana fitur-fitur pada lapisan sebelumnya mewakili karakteristik primitif data, sedangkan fitur-fitur pada lapisan selanjutnya mewakili karakteristik kompleks dengan signifikansi semantik pada label kelas.
Data yang direpresentasikan dalam bentuk fitur lapisan selanjutnya sering kali berperilaku lebih baik (misalnya, dapat dipisahkan secara linier) karena sifat semantik fitur yang dipelajari oleh transformasi. Jenis perilaku ini khususnya terbukti secara visual dalam beberapa domain seperti jaringan saraf konvolusional untuk data gambar. Dalam jaringan saraf konvolusional, fitur dalam lapisan sebelumnya menangkap bentuk yang mendetail tetapi primitif seperti garis atau tepi dari kumpulan data gambar. Di sisi lain, fitur dalam lapisan selanjutnya menangkap bentuk dengan kompleksitas yang lebih besar seperti segi enam, sarang lebah, dan sebagainya, tergantung pada jenis gambar yang disediakan sebagai data pelatihan. Perhatikan bahwa bentuk yang dapat ditafsirkan secara semantik tersebut sering kali memiliki korelasi yang lebih dekat dengan label kelas dalam domain gambar. Misalnya, hampir semua gambar akan berisi garis atau tepi, tetapi gambar yang termasuk dalam kelas tertentu akan lebih cenderung memiliki segi enam atau sarang lebah. Properti ini cenderung membuat representasi lapisan selanjutnya lebih mudah diklasifikasikan dengan model sederhana seperti pengklasifikasi linier. Proses ini diilustrasikan dalam Gambar 1.19. Fitur-fitur pada lapisan sebelumnya digunakan berulang kali sebagai blok penyusun untuk membuat fitur yang lebih kompleks. Prinsip umum "menyatukan" fitur-fitur sederhana untuk membuat fitur yang lebih kompleks ini merupakan inti dari keberhasilan yang dicapai dengan jaringan neural. Ternyata, properti ini juga berguna dalam memanfaatkan model yang telah dilatih sebelumnya dengan cara yang dikalibrasi dengan cermat. Praktik penggunaan model yang telah dilatih sebelumnya juga disebut sebagai pembelajaran transfer.

Jenis pembelajaran transfer tertentu, yang umum digunakan dalam jaringan saraf, adalah bahwa data dan struktur yang tersedia dalam set data tertentu digunakan untuk mempelajari fitur untuk seluruh domain tersebut. Contoh klasik dari pengaturan ini adalah data teks atau gambar. Dalam data teks, representasi kata-kata teks dibuat menggunakan set data tolok ukur standar seperti Wikipedia [594] dan model seperti word2vec. Ini dapat digunakan di hampir semua aplikasi teks, karena sifat data teks tidak banyak berubah dengan aplikasi tersebut.

Pendekatan ini sering digunakan untuk data gambar, di mana set data ImageNet (lih. Bagian 1.8.2) digunakan untuk melatih jaringan saraf konvolusional terlebih dahulu, dan menyediakan fitur yang siap digunakan. Seseorang dapat mengunduh model jaringan saraf konvolusional yang telah dilatih terlebih dahulu dan mengubah set data gambar apa pun menjadi representasi multidimensi dengan melewatkan gambar melalui jaringan yang telah dilatih terlebih dahulu. Lebih jauh, jika data khusus aplikasi tambahan tersedia, seseorang dapat mengatur tingkat pembelajaran transfer tergantung pada jumlah data yang tersedia. Hal ini dicapai dengan menyempurnakan subset lapisan dalam jaringan saraf yang telah dilatih terlebih dahulu dengan data tambahan ini. Jika sejumlah kecil data khusus aplikasi tersedia, seseorang dapat memperbaiki bobot lapisan awal ke nilai yang telah dilatih terlebih dahulu dan menyempurnakan hanya beberapa lapisan terakhir dari jaringan saraf. Lapisan awal sering kali berisi fitur primitif, yang lebih mudah digeneralisasikan ke aplikasi yang sewenang-wenang. Misalnya, dalam jaringan saraf konvolusional, lapisan awal mempelajari fitur primitif seperti tepi, yang berguna di berbagai gambar seperti truk atau wortel. Di sisi lain, lapisan-lapisan selanjutnya berisi fitur-fitur kompleks yang mungkin bergantung pada koleksi gambar yang ada (misalnya, roda truk versus bagian atas wortel). Penyetelan halus hanya pada bobot lapisan-lapisan selanjutnya masuk akal dalam kasus-kasus seperti itu. Jika sejumlah besar data khusus aplikasi tersedia, seseorang dapat menyempurnakan lebih banyak lapisan. Oleh karena itu, jaringan dalam memberikan fleksibilitas yang signifikan dalam hal bagaimana pembelajaran transfer dilakukan dengan model-model jaringan saraf yang telah dilatih sebelumnya.

1.7 Topik Lanjutan

Beberapa topik dalam pembelajaran mendalam semakin mendapat perhatian, dan telah meraih keberhasilan yang signifikan. Meskipun beberapa metode ini dibatasi oleh pertimbangan komputasional saat ini, potensinya cukup signifikan. Bagian ini akan membahas beberapa topik ini.

1.7.1 Pembelajaran Penguatan

Dalam bentuk umum kecerdasan buatan, jaringan saraf harus belajar mengambil tindakan dalam situasi yang terus berubah dan dinamis. Contohnya termasuk robot pembelajaran dan mobil yang dapat mengemudi sendiri. Dalam kasus ini, asumsi kritisnya adalah bahwa sistem pembelajaran tidak memiliki pengetahuan tentang urutan tindakan yang tepat di awal, dan ia belajar melalui penguatan berbasis hadiah saat mengambil berbagai tindakan. Jenis pembelajaran ini sesuai dengan urutan tindakan dinamis yang sulit dimodelkan menggunakan metode pembelajaran mesin tradisional. Asumsi utama di sini adalah bahwa sistem ini terlalu rumit untuk dimodelkan secara eksplisit, tetapi cukup sederhana untuk dievaluasi, sehingga nilai hadiah dapat ditetapkan untuk setiap tindakan pembelajar.

Bayangkan sebuah situasi di mana seseorang ingin melatih sistem pembelajaran untuk memainkan gim video dari awal tanpa pengetahuan sebelumnya tentang aturannya. Gim video adalah tempat uji coba yang sangat baik untuk metode pembelajaran penguatan karena gim video merupakan gambaran kecil dari menjalani "gim" kehidupan. Seperti dalam situasi dunia nyata, jumlah kemungkinan status (misalnya, posisi unik dalam gim) mungkin terlalu besar untuk dihitung, dan pilihan langkah yang optimal sangat bergantung pada pengetahuan tentang apa yang benar-benar penting untuk dimodelkan dari status tertentu. Lebih jauh, karena seseorang tidak memulai dengan pengetahuan apa pun tentang aturan, sistem pembelajaran perlu mengumpulkan data melalui tindakannya seperti tikus menjelajahi labirin untuk mempelajari strukturnya Oleh karena itu, data yang dikumpulkan sangat bias oleh tindakan pengguna, yang menyediakan lanskap yang sangat menantang untuk pembelajaran. Pelatihan metode pembelajaran penguatan yang berhasil merupakan gerbang penting untuk sistem pembelajaran mandiri, yang merupakan tujuan utama kecerdasan buatan. Meskipun bidang pembelajaran penguatan dikembangkan secara independen

bidang jaringan saraf, saling melengkapi yang kuat dari kedua bidang tersebut telah menyatukannya. Metode pembelajaran mendalam dapat berguna dalam mempelajari representasi fitur dari masukan sensorik berdimensi tinggi (misalnya, layar video piksel dalam gim video atau layar piksel dalam "penglihatan" robot). Lebih jauh, metode pembelajaran penguatan sering digunakan untuk mendukung berbagai jenis algoritma jaringan saraf seperti mekanisme perhatian.
Metode pembelajaran penguatan dibahas dalam Bab 9.

1.7.2 Memisahkan Penyimpanan Data dan Perhitungan

Aspek penting dari jaringan saraf adalah bahwa penyimpanan data dan komputasi terintegrasi dengan erat. Misalnya, status dalam jaringan saraf dapat dianggap sebagai jenis memori sementara, yang berperilaku seperti register yang selalu berubah dalam unit pemrosesan pusat komputer. Namun, bagaimana jika kita ingin membangun jaringan saraf tempat seseorang dapat mengontrol tempat membaca data, dan tempat menulis data. Sasaran ini tercapai dengan gagasan tentang perhatian dan memori eksternal. Mekanisme perhatian dapat digunakan dalam berbagai aplikasi seperti pemrosesan gambar tempat seseorang berfokus pada bagian kecil gambar untuk memperoleh wawasan yang berurutan. Teknik-teknik ini juga digunakan untuk penerjemahan mesin. Jaringan saraf yang dapat mengontrol akses dengan ketat dalam membaca dan menulis ke memori eksternal disebut sebagai mesin Turing saraf [158] atau jaringan memori [528]. Meskipun metode-metode ini merupakan varian lanjutan dari jaringan saraf berulang, metode-metode ini menunjukkan potensi yang jauh lebih baik daripada pendahulunya dalam hal jenis masalah yang dapat ditangani. Metode-metode ini dibahas dalam Bab 10.

1.7.3 Jaringan Adversarial Generatif

Jaringan adversarial generatif adalah model pembuatan data yang dapat membuat model generatif dari kumpulan data dasar dengan menggunakan permainan adversarial antara dua pemain. Kedua pemain tersebut berhubungan dengan generator dan diskriminator. Generator mengambil derau Gaussian sebagai input dan menghasilkan output, yang merupakan sampel yang dihasilkan seperti data dasar. Diskriminator biasanya berupa pengklasifikasi probabilistik seperti regresi logistik yang tugasnya adalah membedakan sampel nyata dari kumpulan data dasar dan sampel yang dihasilkan. Generator mencoba membuat sampel yang serealistis mungkin; tugasnya adalah mengelabui diskriminator, sedangkan tugas diskriminator adalah mengidentifikasi sampel palsu terlepas dari seberapa baik generator mencoba mengelabuinya. Masalah tersebut dapat dipahami sebagai permainan adversarial antara generator dan diskriminator, dan model optimasi formal adalah masalah pembelajaran minimax. Keseimbangan Nash dari permainan minimax ini menyediakan model akhir yang terlatih. Biasanya, titik keseimbangan ini adalah titik di mana diskriminator tidak dapat membedakan antara sampel nyata dan palsu.

Metode semacam itu dapat membuat sampel fantasi yang realistis menggunakan kumpulan data dasar, dan umumnya digunakan dalam domain gambar. Misalnya, jika pendekatan dilatih menggunakan kumpulan data yang berisi gambar kamar tidur, pendekatan tersebut akan menghasilkan kamar tidur yang tampak realistis yang sebenarnya bukan bagian dari data dasar. Oleh karena itu, pendekatan tersebut dapat digunakan untuk usaha artistik atau kreatif. Metode ini juga dapat dikondisikan pada jenis konteks tertentu, yang dapat berupa jenis objek apa pun seperti label, teks keterangan, atau gambar dengan detail yang hilang. Dalam kasus ini, pasangan objek pelatihan terkait digunakan. Pasangan yang umum dapat berupa keterangan (konteks) dan gambar (objek dasar). Demikian pula, seseorang mungkin memiliki pasangan yang sesuai dengan sketsa objek dan foto aktual. Oleh karena itu, dimulai dengan kumpulan data gambar yang diberi keterangan dari berbagai jenis hewan, dimungkinkan untuk membuat gambar fantasi yang bukan bagian dari data dasar dengan menggunakan keterangan kontekstual seperti "burung biru dengan cakar tajam." Demikian pula, dimulai dengan sketsa tas tangan seorang seniman, pendekatan tersebut dapat membuat gambar tas tangan yang realistis dan berwarna. Jaringan adversarial generatif dibahas dalam Bab 10.

1.8 Dua Tolok Ukur Penting

Tolok ukur yang digunakan dalam literatur jaringan saraf didominasi oleh data dari domain visi komputer. Meskipun kumpulan data pembelajaran mesin tradisional seperti repositori UCI [601] dapat digunakan untuk menguji jaringan saraf, tren umumnya adalah menggunakan kumpulan data dari domain data berorientasi persepsi yang dapat divisualisasikan dengan baik. Meskipun ada berbagai kumpulan data yang diambil dari domain teks dan gambar, dua di antaranya menonjol karena keberadaannya di mana-mana dalam makalah pembelajaran mendalam. Meskipun keduanya adalah kumpulan data yang diambil dari visi komputer, yang pertama cukup sederhana sehingga dapat juga digunakan untuk menguji aplikasi generik di luar bidang penglihatan. Berikut ini, kami memberikan ikhtisar singkat tentang kedua kumpulan data ini.

1.8.1 Basis Data Angka Tulisan Tangan MNIST

Basis data MNIST, yang merupakan singkatan dari Modified National Institute of Standards and Tech-nology database, adalah basis data besar digit tulisan tangan [281]. Seperti namanya, set data ini dibuat dengan memodifikasi basis data asli digit tulisan tangan yang disediakan oleh NIST. Set data berisi 60.000 gambar pelatihan dan 10.000 gambar pengujian. Setiap gambar adalah pindaian digit tulisan tangan dari 0 hingga 9, dan perbedaan antara gambar yang berbeda adalah hasil dari perbedaan tulisan tangan individu yang berbeda. Orang-orang ini adalah karyawan Biro Sensus Amerika dan siswa sekolah menengah Amerika. Gambar hitam putih asli dari NIST dinormalisasi ukurannya agar sesuai dalam kotak piksel 20 × 20 sambil mempertahankan rasio aspeknya dan dipusatkan dalam gambar 28 × 28 dengan menghitung pusat massa piksel. Gambar diterjemahkan untuk memposisikan titik ini di tengah bidang 28 × 28. Masing-masing nilai piksel 28×28 ini memiliki nilai dari 0 hingga 255, tergantung di mana letaknya dalam spektrum skala abu-abu. Label yang terkait dengan gambar sesuai dengan sepuluh nilai digit. Contoh digit dalam basis data MNIST diilustrasikan dalam Gambar 1.20. Ukuran kumpulan data agak kecil, dan hanya berisi objek sederhana yang sesuai dengan digit. Oleh karena itu, orang mungkin berpendapat bahwa basis data MNIST adalah data mainan.

Gambar 1.20: Contoh digit tulisan tangan dalam database MNIST

set. Namun, ukurannya yang kecil dan kesederhanaannya juga merupakan keuntungan karena dapat digunakan sebagai laboratorium untuk pengujian cepat algoritma pembelajaran mesin. Lebih jauh, penyederhanaan set data berdasarkan fakta bahwa digit-digitnya (secara kasar) dipusatkan membuatnya mudah digunakan untuk menguji algoritma di luar visi komputer. Algoritma visi komputer memerlukan asumsi khusus seperti invariansi translasi. Kesederhanaan set data ini membuat asumsi-asumsi ini tidak diperlukan. Telah dikemukakan oleh Geoff Hinton [600] bahwa MNIST

Basis data ini digunakan oleh para peneliti jaringan saraf dengan cara yang sama seperti ahli biologi menggunakan lalat buah untuk mendapatkan hasil awal dan cepat (sebelum pengujian serius pada organisme yang lebih kompleks).

Meskipun representasi matriks setiap gambar cocok untuk jaringan saraf konvolusional, seseorang juga dapat mengubahnya menjadi representasi multidimensi 28 × 28 = 784 dimensi. Konversi ini kehilangan beberapa informasi spasial dalam gambar, tetapi kehilangan ini tidak melemahkan (setidaknya dalam kasus set data MNIST) karena kesederhanaannya yang relatif. Faktanya, penggunaan mesin vektor pendukung sederhana pada representasi 784 dimensi dapat memberikan tingkat kesalahan yang mengesankan sekitar 0,56%. Jaringan saraf 2 lapis langsung pada representasi multidimensi (tanpa menggunakan struktur spasial dalam gambar) umumnya lebih buruk daripada mesin vektor pendukung di seluruh rentang pilihan parameter yang luas! Jaringan saraf dalam tanpa arsitektur konvolusional khusus dapat mencapai tingkat kesalahan 0,35% [72]. Jaringan saraf yang lebih dalam dan jaringan saraf konvolusional (yang menggunakan struktur spasial) dapat mengurangi tingkat kesalahan hingga serendah 0,21% dengan menggunakan ansambel lima jaringan konvolusional [402]. Oleh karena itu, bahkan pada kumpulan data sederhana ini, orang dapat melihat bahwa kinerja relatif jaringan saraf sehubungan dengan pembelajaran mesin tradisional peka terhadap arsitektur spesifik yang digunakan pada yang pertama.

Akhirnya, perlu dicatat bahwa representasi nonspasial 784 dimensi dari data MNIST digunakan untuk menguji semua jenis algoritma jaringan saraf di luar domain visi komputer. Meskipun penggunaan representasi 784 dimensi (yang diratakan) tidak sesuai untuk tugas penglihatan, representasi tersebut tetap berguna untuk menguji efektivitas umum algoritma jaringan saraf yang tidak berorientasi pada penglihatan (yaitu, generik). Misalnya, data MNIST sering digunakan untuk menguji autoencoder generik dan bukan hanya yang konvolusional. Bahkan ketika representasi nonspasial dari suatu gambar digunakan untuk merekonstruksinya dengan autoencoder, seseorang masih dapat memvisualisasikan hasilnya dengan posisi spasial asli dari piksel yang direkonstruksi untuk mendapatkan gambaran tentang apa yang dilakukan algoritma dengan data tersebut. Eksplorasi visual ini sering kali memberi peneliti beberapa wawasan yang tidak tersedia dengan set data arbitrer seperti yang diperoleh dari UCI Machine Learning Repository [601]. Dalam hal ini, set data MNIST cenderung memiliki kegunaan yang lebih luas daripada banyak jenis set data lainnya.

1.8.2 Basis Data ImageNet

Basis data ImageNet [581] adalah basis data besar yang berisi lebih dari 14 juta gambar yang diambil dari 1000 kategori berbeda. Cakupan kelasnya cukup lengkap sehingga mencakup sebagian besar jenis gambar yang akan ditemui dalam kehidupan sehari-hari. Basis data ini disusun menurut hierarki kata benda WordNet [329]. Basis data WordNet adalah kumpulan data yang berisi hubungan antarkata dalam bahasa Inggris menggunakan konsep synset. Hirarki WordNet telah berhasil digunakan untuk pembelajaran mesin dalam domain bahasa alami, dan oleh karena itu wajar untuk merancang kumpulan data gambar berdasarkan hubungan ini.

Basis data ImageNet terkenal karena fakta bahwa ImageNet Large Scale Visual Recognition Challenge (ILSVRC) tahunan [582] diadakan menggunakan kumpulan data ini. Kompetisi ini memiliki profil yang sangat tinggi di komunitas visi dan menerima entri dari sebagian besar kelompok penelitian utama dalam visi komputer. Entri untuk kompetisi ini telah menghasilkan banyak arsitektur pengenalan gambar canggih saat ini, termasuk metode yang telah melampaui kinerja manusia pada beberapa tugas sempit seperti klasifikasi gambar [184]. Karena ketersediaan luas hasil yang diketahui pada kumpulan data ini, ini merupakan alternatif populer untuk pembandingan. Kami akan membahas beberapa algoritme canggih yang diajukan ke kompetisi ImageNet di Bab 8 pada jaringan saraf konvolusional.

Makna penting lain dari kumpulan data ImageNet adalah bahwa kumpulan data tersebut cukup besar dan beragam untuk mewakili konsep visual utama dalam domain gambar. Oleh karena itu,

Jaringan saraf konvolusional sering dilatih pada set data ini; jaringan yang telah dilatih sebelumnya dapat digunakan untuk mengekstraksi fitur dari gambar yang sembarangan. Representasi gambar ini ditentukan oleh aktivasi tersembunyi di lapisan kedua terakhir dari jaringan saraf. Pendekatan semacam itu menciptakan representasi multidimensi baru dari set data gambar yang dapat digunakan dengan metode pembelajaran mesin tradisional. Seseorang dapat melihat pendekatan ini sebagai semacam pembelajaran transfer di mana konsep visual dalam set data ImageNet ditransfer ke objek data yang tidak terlihat untuk aplikasi lain.

1.9 Ringkasan

Meskipun jaringan saraf dapat dilihat sebagai simulasi proses pembelajaran pada organisme hidup, pemahaman yang lebih langsung tentang jaringan saraf adalah sebagai grafik komputasional. Grafik komputasional tersebut melakukan komposisi rekursif dari fungsi yang lebih sederhana untuk mempelajari fungsi yang lebih kompleks. Karena grafik komputasional ini diparameterisasi, masalahnya secara umum bermuara pada mempelajari parameter grafik untuk mengoptimalkan fungsi kerugian. Jenis jaringan saraf yang paling sederhana sering kali merupakan model pembelajaran mesin dasar seperti regresi kuadrat terkecil. Kekuatan jaringan saraf yang sebenarnya dilepaskan dengan menggunakan kombinasi yang lebih kompleks dari fungsi yang mendasarinya. Parameter jaringan tersebut dipelajari dengan menggunakan metode pemrograman dinamis, yang disebut sebagai backpropagation. Ada beberapa tantangan yang terkait dengan pembelajaran model jaringan saraf, seperti overfitting dan ketidakstabilan pelatihan. Dalam beberapa tahun terakhir, banyak kemajuan algoritmik telah mengurangi masalah ini. Desain metode pembelajaran mendalam dalam domain tertentu seperti teks dan gambar memerlukan arsitektur yang dibuat dengan cermat. Contoh arsitektur tersebut meliputi jaringan saraf berulang dan jaringan saraf konvolusional. Untuk pengaturan dinamis di mana serangkaian keputusan perlu dipelajari oleh suatu sistem, metode seperti pembelajaran penguatan berguna.

1.10 Catatan Bibliografi

Pemahaman yang tepat tentang desain jaringan saraf memerlukan pemahaman yang mendalam tentang algoritma pembelajaran mesin, dan khususnya model linier berdasarkan penurunan gradien. Pembaca disarankan untuk merujuk ke [2, 3, 40, 177] untuk pengetahuan dasar tentang metode pembelajaran mesin.
Banyak survei dan ikhtisar jaringan saraf dalam konteks yang berbeda dapat ditemukan di [27, 28, 198, 277, 345, 431]. Buku klasik tentang jaringan saraf untuk pengenalan pola dapat ditemukan di [41, 182], sedangkan perspektif yang lebih baru tentang pembelajaran mendalam dapat ditemukan di [147]. Buku penambangan teks baru-baru ini [6] juga membahas kemajuan terbaru dalam pembelajaran mendalam untuk analisis teks. Ikhtisar tentang hubungan antara pembelajaran mendalam dan ilmu saraf komputasional dapat ditemukan di [176, 239].

Algoritma perceptron diusulkan oleh Rosenblatt [405]. Untuk mengatasi masalah stabilitas, digunakan algoritma pocket [128], algoritma Maxover [523], dan metode berbasis margin lainnya [123]. Algoritma awal lainnya yang serupa termasuk algoritma Widrow-Hoff [531] dan algoritma Winnow [245]. Algoritma Winnow menggunakan pembaruan perkalian alih-alih pembaruan aditif, dan sangat berguna ketika banyak fitur tidak relevan. Ide awal backpropagation didasarkan pada ide diferensiasi komposisi fungsi seperti yang dikembangkan dalam teori kontrol [54, 237]. Penggunaan pemrograman dinamis untuk melakukan optimasi berbasis gradien dari variabel yang terkait melalui grafik asiklik terarah telah menjadi praktik standar sejak tahun enam puluhan. Namun, kemampuan untuk menggunakan metode ini untuk pelatihan jaringan saraf belum diamati pada saat itu. Pada tahun 1969, Minsky dan Papert

menerbitkan buku tentang perseptrona [330], yang sebagian besar negatif tentang potensi untuk dapat melatih jaringan saraf multilayer dengan benar. Buku tersebut menunjukkan bahwa perseptrona tunggal memiliki ekspresivitas terbatas, dan tidak seorang pun tahu cara melatih beberapa lapisan perseptrona. Minsky adalah tokoh berpengaruh dalam kecerdasan buatan, dan nada negatif bukunya berkontribusi pada musim dingin pertama di bidang jaringan saraf. Adaptasi metode pemrograman dinamis untuk backpropagation dalam jaringan saraf pertama kali diusulkan oleh Paul Werbos dalam tesis PhD-nya pada tahun 1974 [524]. Namun, karya Werbos tidak dapat mengatasi pandangan kuat terhadap jaringan saraf yang telah mengakar pada saat itu. Algoritma backpropagation diusulkan lagi oleh Rumelhart et al. pada tahun 1986 [408, 409]. Karya Rumelhart et al. penting karena keindahan presentasinya, dan mampu mengatasi setidaknya beberapa masalah yang dikemukakan sebelumnya oleh Minsky dan Papert.

Ini adalah salah satu alasan mengapa makalah Rumelhart et al. dianggap sangat berpengaruh dari perspektif backpropagation, meskipun makalah tersebut tentu saja bukan yang pertama kali mengusulkan metode tersebut. Pembahasan tentang sejarah algoritma backpropagation dapat ditemukan dalam buku karya Paul Werbos [525].

Pada titik ini, bidang jaringan saraf baru sebagian bangkit kembali, karena masih ada masalah dengan pelatihan jaringan saraf. Meskipun demikian, sejumlah peneliti terus bekerja di bidang tersebut, dan telah menyiapkan sebagian besar arsitektur saraf yang dikenal, seperti jaringan saraf konvolusi, jaringan saraf berulang, dan LSTM, sebelum tahun 2000.
Keakuratan metode ini masih cukup sederhana karena keterbatasan data dan komputasi. Lebih jauh, backpropagation ternyata kurang efektif dalam melatih jaringan yang lebih dalam karena masalah gradien yang menghilang dan meledak. Namun, saat ini, beberapa peneliti terkemuka telah berhipotesis bahwa algoritma yang ada akan menghasilkan peningkatan kinerja yang besar dengan peningkatan data, daya komputasi, dan eksperimen algoritmik. Penggabungan kerangka kerja big data dengan GPU ternyata menjadi keuntungan bagi penelitian jaringan saraf pada akhir tahun 2000-an. Dengan berkurangnya waktu siklus untuk eksperimen yang dimungkinkan oleh peningkatan daya komputasi, trik seperti prapelatihan mulai muncul pada akhir tahun 2000-an [198]. Kebangkitan jaringan saraf yang jelas bagi publik terjadi setelah tahun 2011 dengan kemenangan gemilang [255] jaringan saraf dalam kompetisi pembelajaran mendalam untuk klasifikasi gambar. Kemenangan algoritma pembelajaran mendalam yang konsisten dalam kompetisi ini meletakkan dasar bagi ledakan popularitas yang kita lihat saat ini. Khususnya, perbedaan arsitektur pemenang ini dari yang dikembangkan lebih dari dua dekade sebelumnya sederhana (tetapi penting).

Paul Werbos adalah pelopor jaringan saraf berulang, dan mengusulkan versi asli backpropagation melalui waktu [526]. Dasar-dasar jaringan saraf konvolusional diusulkan dalam konteks neokognitron dalam [127]. Ide ini kemudian digeneralisasi ke LeNet-5, yang merupakan salah satu jaringan saraf konvolusional pertama. Kemampuan jaringan saraf untuk melakukan perkiraan fungsi universal dibahas dalam [208]. Efek menguntungkan dari kedalaman dalam mengurangi jumlah parameter dibahas dalam [340].

Ekspresivitas teoritis jaringan saraf telah dikenali sejak awal dalam pengembangannya. Misalnya, penelitian awal mengenali bahwa jaringan saraf dengan satu lapisan tersembunyi dapat digunakan untuk memperkirakan fungsi apa pun [208]. Hasil lebih lanjut adalah bahwa arsitektur saraf tertentu seperti jaringan berulang bersifat Turing lengkap [444]. Yang terakhir berarti bahwa jaringan saraf berpotensi dapat mensimulasikan algoritma apa pun. Tentu saja, ada banyak masalah praktis yang terkait dengan pelatihan jaringan saraf, seperti mengapa hasil teoritis yang menarik ini tidak selalu diterjemahkan ke dalam kinerja dunia nyata. Masalah terpenting di antara semuanya adalah sifat arsitektur dangkal yang haus data, yang diperbaiki dengan peningkatan kedalaman.
Peningkatan kedalaman dapat dilihat sebagai bentuk regularisasi di mana seseorang memaksa jaringan saraf untuk mengidentifikasi dan mempelajari pola berulang dalam titik data. Namun, peningkatan kedalaman

membuat jaringan saraf lebih sulit dilatih dari sudut pandang optimasi. Pembahasan mengenai beberapa masalah ini dapat ditemukan di [41, 140, 147]. Evaluasi eksperimental yang menunjukkan keuntungan arsitektur yang lebih dalam disediakan di [267].

1.10.1 Kuliah Video

Pembelajaran mendalam memiliki sejumlah besar kuliah video gratis yang tersedia di sumber daya seperti YouTube dan Coursera. Dua sumber daya yang paling otoritatif termasuk kursus Geoff Hinton di Coursera [600]. Coursera memiliki banyak penawaran tentang pembelajaran mendalam, dan menawarkan sekelompok kursus terkait di bidang tersebut. Selama penulisan buku ini, kursus yang dapat diakses dari Andrew Ng juga ditambahkan ke penawaran. Kursus tentang jaringan saraf konvolusional dari Universitas Stanford tersedia gratis di YouTube [236]. Kelas Stanford oleh Karpathy, Johnson, dan Fei-Fei [236] adalah tentang jaringan saraf konvolusional, meskipun itu melakukan pekerjaan yang sangat baik dalam mencakup topik yang lebih luas dalam jaringan saraf. Bagian awal kursus membahas jaringan saraf vanilla dan metode pelatihan.

Banyak topik dalam pembelajaran mesin [89] dan pembelajaran mendalam [90] dibahas oleh Nando de Freitas dalam kuliah yang tersedia di YouTube. Kelas menarik lainnya tentang jaringan saraf tersedia dari Hugo Larochelle di Universite de Sherbrooke [262]. Kursus pembelajaran mendalam oleh Ali Ghodsi di University of Waterloo tersedia di [137]. Kuliah video oleh Christopher Manning tentang metode pemrosesan bahasa alami untuk pembelajaran mendalam dapat ditemukan di [312]. Kursus David Silver tentang pembelajaran penguatan tersedia di [619].

1.10.2 Sumber Daya Perangkat Lunak

Pembelajaran mendalam didukung oleh banyak kerangka kerja perangkat lunak seperti Caffe [571], Torch [572], Theano [573], dan TensorFlow [574]. Ekstensi Caffe untuk Python dan MATLAB tersedia. Caffe dikembangkan di University of California di Berkeley, dan ditulis dalam C++. Caffe menyediakan antarmuka tingkat tinggi di mana seseorang dapat menentukan arsitektur jaringan, dan memungkinkan konstruksi jaringan saraf dengan penulisan kode yang sangat sedikit dan skrip yang relatif sederhana. Kelemahan utama Caffe adalah dokumentasi yang tersedia relatif terbatas. Theano [35] berbasis Python, dan menyediakan paket tingkat tinggi seperti Keras [575] dan Lasagne [576] sebagai antarmuka. Theano didasarkan pada gagasan grafik komputasional, dan sebagian besar kemampuan yang disediakan di sekitarnya menggunakan abstraksi ini secara eksplisit.

TensorFlow [574] juga sangat berorientasi pada grafik komputasional, dan merupakan kerangka kerja yang diusulkan oleh Google. Torch [572] ditulis dalam bahasa tingkat tinggi yang disebut Lua, dan relatif mudah digunakan. Dalam beberapa tahun terakhir, Torch telah memperoleh beberapa kemajuan dibandingkan dengan kerangka kerja lainnya. Dukungan untuk GPU terintegrasi erat dalam Torch, yang membuatnya relatif mudah untuk menyebarkan aplikasi berbasis Torch pada GPU. Banyak dari kerangka kerja ini berisi model yang telah dilatih sebelumnya dari visi komputer dan penambangan teks, yang dapat digunakan untuk mengekstraksi fitur. Banyak alat siap pakai untuk pembelajaran mendalam tersedia dari repositori DeepLearning4j [590]. IBM memiliki platform PowerAI yang menawarkan banyak kerangka kerja pembelajaran mesin dan pembelajaran mendalam di atas IBM Power Systems [599]. Khususnya, pada saat penulisan buku ini, platform ini juga memiliki edisi gratis yang tersedia untuk penggunaan tertentu.

1.11 Latihan

1. Perhatikan kasus fungsi XOR di mana dua titik {(0, 0),(1, 1)} termasuk dalam satu kelas, dan dua titik lainnya {(1, 0),(0, 1)} termasuk dalam kelas lainnya. Tunjukkan bagaimana Anda dapat menggunakan fungsi aktivasi ReLU untuk memisahkan dua kelas dengan cara yang mirip dengan contoh pada Gambar 1.14.

2. Tunjukkan sifat-sifat fungsi aktivasi sigmoid dan tanh berikut (dilambangkan oleh ÿ(·) dalam setiap kasus):

   (a) Aktivasi sigmoid: ÿ(ÿv)=1 ÿ ÿ(v) (b) Aktivasi tanh:

   ÿ(ÿv) = ÿÿ(v) (c) Aktivasi tanh keras: ÿ(ÿv) =

   ÿÿ(v)

3. Buktikan bahwa fungsi tanh merupakan fungsi sigmoid yang diskalakan ulang dengan sumbu horizontal dan peregangan vertikal, serta translasi vertikal:

$$\tanh(v) = 2\text{sigmoid}(2v) \text{ ÿ } 1$$

4. Pertimbangkan himpunan data yang dua titiknya {(ÿ1, ÿ1),(1, 1)} termasuk dalam satu kelas, dan dua titik lainnya {(1, ÿ1),(ÿ1, 1)} termasuk dalam kelas lainnya. Mulailah dengan nilai parameter perceptron di (0, 0), dan kerjakan beberapa pembaruan penurunan gradien stokastik dengan ÿ = 1. Saat melakukan pembaruan penurunan gradien stokastik, lakukan siklus melalui titik pelatihan dalam urutan apa pun.

   (a) Apakah algoritmanya konvergen dalam artian bahwa perubahan fungsi objektif menjadi sangat kecil seiring berjalannya waktu? (b) Jelaskan

   mengapa situasi pada (a) terjadi.

5. Untuk himpunan data pada Latihan 4, di mana kedua fitur dilambangkan dengan (x1, x2), tentukan representasi 1 dimensi baru z yang dilambangkan dengan yang berikut:

$$z = x1 + x2$$

   Apakah kumpulan data dapat dipisahkan secara linear dalam hal representasi 1 dimensi yang sesuai dengan z? Jelaskan pentingnya transformasi nonlinier dalam masalah klasifikasi.

6. Implementasikan perceptron dalam bahasa pemrograman pilihan Anda.

7. Buktikan bahwa turunan fungsi aktivasi sigmoid paling banyak adalah 0,25, tanpa memperhatikan nilai argumennya. Pada nilai argumen berapakah fungsi aktivasi sigmoid mencapai nilai maksimumnya?

8. Buktikan bahwa turunan fungsi aktivasi tanh paling banyak adalah 1, tanpa memperhatikan nilai argumennya. Pada nilai argumen berapa aktivasi tanh mencapai nilai maksimumnya?

9. Pertimbangkan jaringan dengan dua masukan x1 dan x2. Jaringan ini memiliki dua lapisan tersembunyi, yang masing-masing berisi dua unit. Asumsikan bahwa bobot di setiap lapisan ditetapkan sehingga unit teratas di setiap lapisan menerapkan aktivasi sigmoid pada jumlah masukannya dan unit terbawah di setiap lapisan menerapkan aktivasi tanh pada jumlah masukannya. Akhirnya, keluaran tunggal

node menerapkan aktivasi ReLU pada jumlah kedua inputnya. Tulis output jaringan saraf ini dalam bentuk tertutup sebagai fungsi x1 dan x2. Latihan ini akan memberi Anda gambaran tentang kompleksitas fungsi yang dihitung oleh jaringan saraf.

10. Hitung turunan parsial dari bentuk tertutup yang dihitung dalam latihan sebelumnya terhadap x1. Apakah praktis untuk menghitung turunan untuk penurunan gradien dalam jaringan saraf dengan menggunakan ekspresi bentuk tertutup (seperti dalam pembelajaran mesin tradisional)?

11. Perhatikan himpunan data 2 dimensi yang semua titik dengan x1 > x2 termasuk dalam kelas positif, dan semua titik dengan x1 ÿ x2 termasuk dalam kelas negatif. Oleh karena itu, pemisah sebenarnya dari kedua kelas tersebut adalah hiperbidang linier (garis) yang didefinisikan oleh x1 ÿ x2 = 0.
Sekarang buat set data pelatihan dengan 20 titik yang dibuat secara acak di dalam kotak satuan di kuadran positif. Beri label pada setiap titik tergantung pada apakah koordinat pertama x1 lebih besar dari koordinat kedua x2.

    (a) Terapkan algoritma perceptron tanpa regularisasi, latih pada 20 titik di atas, dan uji akurasinya pada 1000 titik yang dihasilkan secara acak di dalam kotak satuan. Hasilkan titik uji menggunakan prosedur yang sama dengan titik pelatihan.

    (b) Ubah kriteria perceptron menjadi hinge-loss dalam implementasi Anda untuk pelatihan, dan ulangi perhitungan akurasi pada titik uji yang sama di atas. Regularisasi tidak digunakan.

    (c) Dalam hal manakah Anda memperoleh akurasi yang lebih baik dan

    mengapa? (d) Dalam hal manakah menurut Anda klasifikasi dari 1000 contoh pengujian yang sama tidak akan berubah secara signifikan dengan menggunakan serangkaian 20 titik pelatihan yang berbeda?