Bab 11

Jaringan Syaraf

11.1 Pendahuluan

Jaringan saraf telah menggemparkan dunia dalam dekade terakhir. Namun, pekerjaan tersebut telah berlangsung sejak tahun 1940-an. Beberapa pekerjaan awal adalah memodelkan perilaku neuron biologis secara matematis. Frank Rosenblat pada tahun 1958 membangun sebuah mesin yang menunjukkan kemampuan untuk belajar berdasarkan gagasan matematis neuron. Proses membangun jaringan saraf disempurnakan lebih lanjut selama 4 dekade berikutnya.

Salah satu makalah terpenting yang memungkinkan pelatihan jaringan yang sangat kompleks muncul pada tahun 1986 dalam sebuah karya oleh David E. Rumelhart, Geoffrey Hinton, dan Ronald J. Williams [RHW86]. Makalah ini memperkenalkan kembali algoritma back-propagation yang merupakan andalan jaringan neural seperti yang digunakan saat ini. Dalam 2 dekade terakhir, karena tersedianya penyimpanan dan komputasi yang lebih murah, jaringan besar telah dibangun yang memecahkan masalah praktis yang signifikan. Hal ini telah membuat jaringan neural dan sepupunya seperti jaringan neural konvolusi, jaringan neural berulang, dll., menjadi nama-nama yang dikenal luas.

Dalam bab ini, kita akan memulai pembahasan dengan matematika di balik jaringan saraf, yang mencakup perambatan maju dan mundur. Selanjutnya, kita akan membahas visualisasi jaringan saraf. Terakhir, kita akan membahas pembuatan jaringan saraf menggunakan Keras, modul Python untuk pembelajaran mesin dan pembelajaran mendalam.

Pembaca yang tertarik disarankan untuk mengikuti diskusi di sumber berikut: [Dom15], [MTH], [GBC16], [Gro17].

11.2 Pendahuluan

Jaringan saraf adalah fungsi non-linier dengan banyak parameter. Kurva yang paling sederhana adalah garis dengan dua parameter: kemiringan dan intersep. Jaringan neural memiliki lebih banyak parameter, biasanya sekitar 10.000 atau lebih dan terkadang jutaan. Parameter ini dapat ditentukan melalui proses pengoptimalan fungsi kerugian yang menentukan kesesuaian.

11.3 Pemodelan Matematika

Kita akan memulai pembahasan matematika jaringan saraf dengan mencocokkan garis dan bidang. Kita kemudian dapat memperluasnya ke kurva sembarang.

11.3.1 Propagasi Maju

Persamaan garis didefinisikan sebagai

$$y1 = \text{Panjang } x + b \tag{11.1}$$

di mana x adalah variabel bebas, y1 adalah variabel terikat, W adalah kemiringan garis dan b adalah titik potong. Dalam dunia pembelajaran mesin, W disebut bobot dan b adalah bias.

Jika variabel bebas x adalah skalar, maka Persamaan 11.1 adalah garis, W adalah skalar dan b adalah skalar. Namun, jika x adalah vektor, maka Persamaan 11.1 adalah bidang, W adalah matriks dan b adalah vektor. Jika x sangat besar, Persamaan 11.1 disebut hiperbidang. Persamaan 11.1 adalah persamaan linier dan model terbaik yang dapat dibuat menggunakannya adalah model linier juga.

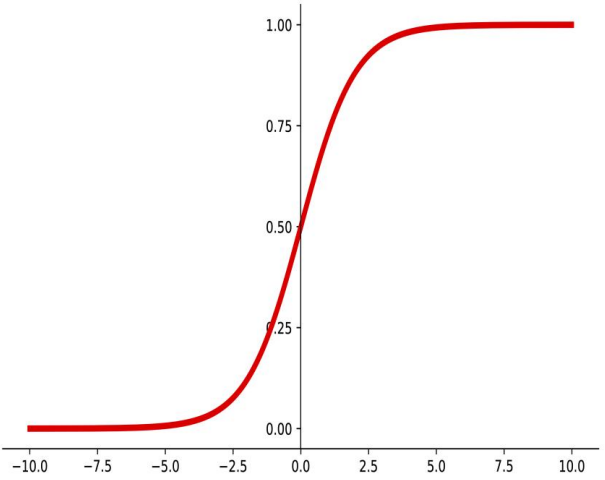
Untuk membuat model non-linier dalam jaringan saraf, kita menambahkan non-linieritas ke model linier ini. Kita akan membahas salah satu non-linieritas tersebut

disebut sigmoid. Namun dalam praktiknya, non-linieritas lain seperti tanh, rectified linear unit (RELU), dan leaky RELU juga digunakan.

Persamaan fungsi sigmoid adalah

$$\text{kamu} = \frac{1}{1 + e\ddot{y}x} \tag{11.2}$$

Bila x merupakan bilangan besar, nilai y secara asimtotik mencapai 1 (11.1), sedangkan untuk nilai x yang kecil, nilai y secara asimtotik mencapai 0. Di daerah sepanjang sumbu-x antara ÿ1 dan +1 secara kasar, kurvanya linear dan non-linear di tempat lain.



GAMBAR 11.1: Fungsi sigmoid.

Jika y1 dari 11.1 dilewatkan melalui fungsi sigmoid, kita akan memperoleh y1 baru,

$$y1 = \frac{1}{\text{Persamaan 1: } e\ddot{y}W1x\ddot{y}b1} \tag{11.3}$$

di mana W1 adalah bobot lapisan pertama dan b1 adalah bias lapisan pertama. y1 baru dalam Persamaan 11.3 adalah kurva non-linier. Persamaan tersebut dapat ditulis ulang sebagai,

$$\text{Persamaan } y1 = \ddot{y}(W1x + b1) \tag{11.4}$$

Dalam jaringan saraf sederhana di sini, kita akan menambahkan lapisan lain (yaitu, rangkaian W dan b yang lain) yang akan kita lewati y1 yang diperoleh dari Persamaan 11.4.

$$y = W2y1 + b2 \tag{11.5}$$

di mana W2 merupakan bobot lapisan kedua dan b2 merupakan bias lapisan kedua.

Jika kita mengganti 11,4 pada 11,5, kita memperoleh,

$$\text{Rumus untuk persamaan } y = W2\ddot{y}(W1x + b1) + b2 \tag{11.6}$$

Kita dapat mengulang proses ini dengan menambahkan lebih banyak lapisan dan membuat kom- kurva non-linier yang kompleks. Namun, demi kejelasan, kita akan membatasi diri pada 2 lapisan.

Pada Persamaan 11.6 terdapat 4 parameter yaitu W1, b1, W2, dan b2. Jika x adalah vektor, maka W1 dan W2 adalah matriks dan b1 dan b2 adalah matriks. vektor. Tujuan dari jaringan saraf adalah untuk menentukan nilai di dalam matriks dan vektor ini.

11.3.2 Propagasi Balik

Nilai dari 4 parameter dapat ditentukan dengan menggunakan proses back-propagation. Dalam proses ini, kita mulai dengan mengasumsikan nilai awal untuk parameter. Mereka dapat diberi nilai 0 atau beberapa nilai acak dapat digunakan.

Kami kemudian akan menentukan nilai awal y menggunakan Persamaan 11.6. Kita akan menunjukkan nilai ini sebagai ˆy. Nilai aktual y dan nilai prediksi y

Nilai ˆy tidak akan sama. Oleh karena itu akan ada kesalahan di antara keduanya.

Kita akan menyebut kesalahan ini sebagai “kerugian.”

$$L = (y \ddot{y} y)^2 \tag{11.7}$$

Tujuan kami adalah meminimalkan kerugian ini dengan menemukan nilai yang tepat untuk parameter Persamaan 11.6.

Menggunakan nilai parameter saat ini, nilai barunya secara berulang dapat dihitung dengan menggunakan,

$$Wnew = Wold \ddot{y} \frac{\ddot{y}L}{\ddot{y}W} \tag{11.8}$$

di mana W adalah parameter dan L adalah fungsi kerugian. Persamaan ini secara umum disebut 'persamaan pembaruan'.

Untuk menyederhanakan perhitungan turunan parsial seperti akan $\frac{\ddot{y}P}{\ddot{y}L}$, Kami menurunkannya dalam beberapa bagian dan menyusunnya menggunakan aturan

Kita akan mulai dengan menghitung $\frac{\ddot{y}L}{\ddot{y}\hat{y}}$ rantai. menggunakan Persamaan 11.7

$$\frac{\ddot{y}L}{} = 2 \ddot{y} (\hat{y} \ddot{y} y) \ddot{y}\hat{y} \tag{11.9}$$

Kemudian kita akan menghitung $\frac{\ddot{y}P}{\ddot{y}L2}$ menggunakan aturan rantai,

$$\frac{\ddot{y}P}{\ddot{y}L2} = \frac{\ddot{y}L}{\ddot{y}\hat{y}} * \frac{\ddot{y}\hat{y}}{\ddot{y}W2} \tag{11.10}$$

Jika kita mengganti ˆy dari Persamaan 11.5 dan kita $\frac{\ddot{y}L}{\ddot{y}\hat{y}}$ dari Persamaan 11.9, memperoleh,

$$\frac{\ddot{y}L}{} = 2 \ddot{y} (y \ddot{y} y) \ddot{y}\, y1\, \ddot{y}W2 \tag{11.11}$$

Turunan parsial kemudian dapat digunakan untuk memperbarui nilai W2 menggunakan nilai W2 yang ada dengan bantuan persamaan pembaruan. Perhitungan serupa (ditinggalkan sebagai latihan bagi pembaca) dapat ditunjukkan untuk b2 juga.

Selanjutnya kita akan menghitung nilai baru W1,

$$\frac{\ddot{y}L}{\ddot{y}W1} = \frac{\ddot{y}L}{\ddot{y}y1}\frac{\ddot{y}y\hat{}}{\ddot{y}y1}\frac{\ddot{y}y\hat{}}{\ddot{y}W1} \tag{11.12}$$

yang dapat dihitung menggunakan Persamaan 11.9, 11.5 dan 11.4. Jadi,

$$\frac{\ddot{y}L}{\ddot{y}W1} = 2(\hat{}y \ddot{y} y)(W2)(\ddot{y}(W1x + b1)(1 \ddot{y} \ddot{y}(W1x + b1))x) \tag{11.13}$$

yang dapat disederhanakan menjadi

$$\frac{\ddot{y}L}{\ddot{y}W1} = 2xW2(\hat{}y \ddot{y} y)\ddot{y}(W1x + b1)(1 \ddot{y} \ddot{y}(W1x + b1)) \tag{11.14}$$

Nilai baru W1 dapat dihitung menggunakan Persamaan 11.8 yang diperbarui. Perhitungan serupa (yang diberikan sebagai latihan kepada pembaca) dapat ditunjukkan untuk b1 juga.

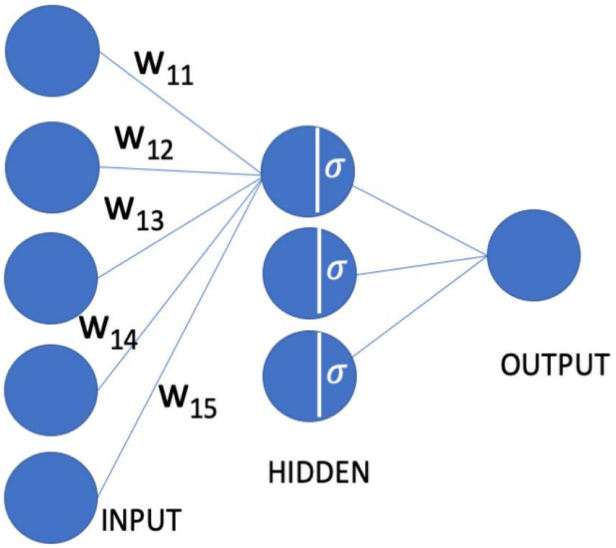
Untuk setiap titik data masukan atau sekumpulan titik data, kami melakukan propagasi maju, menentukan kerugian, lalu melakukan propagasi balik untuk memperbarui parameter (bobot dan bias) menggunakan persamaan pembaruan.
Proses ini diulang dengan semua data yang tersedia.

Singkatnya, proses back-propagation menemukan turunan parsial dari parameter sistem jaringan saraf dan menggunakan persamaan pembaruan untuk menemukan nilai yang lebih baik untuk parameter dengan meminimalkan
kerugian.

11.4 Representasi Grafis

Biasanya, jaringan saraf direpresentasikan seperti yang ditunjukkan pada Gambar 11.2. Lapisan kiri disebut lapisan masukan, lapisan tengah disebut lapisan tersembunyi, dan lapisan kanan disebut lapisan keluaran.

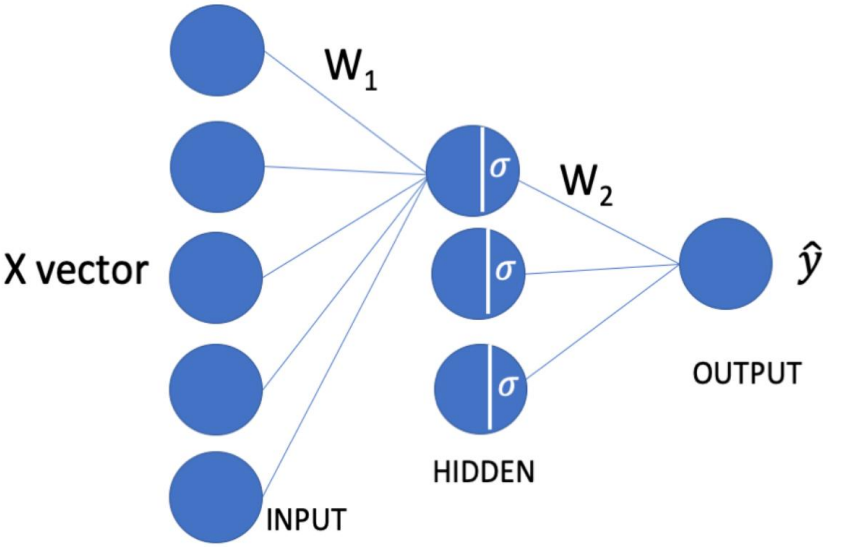
Node (lingkaran terisi) dalam lapisan tertentu terhubung ke semua node di lapisan berikutnya tetapi tidak terhubung ke node mana pun di lapisan tersebut. Pada Gambar 11.2, anak panah hanya digambar untuk berawal dari lapisan input ke node pertama di lapisan tersembunyi. Demi kejelasan, garis yang berakhir pada node lain dihilangkan. Nilai pada setiap node input dikalikan dengan bobot di garis antara node. Input yang terbobot kemudian ditambahkan di node di lapisan tersembunyi dan melewati fungsi sigmoid atau non-linearitas lainnya. Output dari fungsi sigmoid kemudian terbobot di lapisan berikutnya dan jumlah semua bobot tersebut akan menjadi output dari lapisan output (ˆy).



GAMBAR 11.2: Representasi grafis dari jaringan saraf.

Jika terdapat n node pada lapisan input dan m node pada lapisan tersembunyi, maka jumlah sisi yang terhubung dari lapisan input ke lapisan tersembunyi adalah n*m. Hal ini dapat direpresentasikan sebagai matriks berukuran [n, m]. Kemudian operasi yang dijelaskan pada paragraf sebelumnya akan berupa perkalian titik antara input x dan matriks yang diikuti dengan penerapan fungsi sigmoid yang dijelaskan pada Persamaan 11.4. Matriks ini adalah W1 yang telah dijelaskan sebelumnya.

Jika terdapat m node di lapisan tersembunyi dan k node di lapisan keluaran, maka jumlah sisi yang terhubung dari lapisan tersembunyi ke lapisan keluaran adalah m*k. Hal ini dapat direpresentasikan sebagai matriks berukuran [m, k] dan merupakan matriks W2 yang telah dijelaskan sebelumnya.



GAMBAR 11.3: Representasi grafis jaringan saraf sebagai matriks bobot.

Selama propagasi maju, nilai x digunakan sebagai input untuk memulai komputasi yang disebarkan dari sisi input ke output. Kerugian dihitung dengan membandingkan nilai prediksi dan aktual. Gradien kemudian dihitung secara terbalik dari lapisan output ke input dan parameter (bobot dan bias) diperbarui melalui propagasi balik.

Dalam pembahasan ini, kami berasumsi bahwa y adalah fungsi kontinu dan nilainya adalah bilangan riil. Jenis masalah ini disebut masalah regresi. Contoh masalah tersebut adalah prediksi harga suatu barang berdasarkan gambar.

11.5 Jaringan Syaraf Tiruan untuk Masalah Klasifikasi

Kelas masalah lainnya adalah masalah klasifikasi dimana variabel dependen y memiliki nilai diskrit. Contoh dari masalah tersebut adalah mengidentifikasi jenis kanker paru-paru tertentu berdasarkan gambar. Ada dua jenis utama: kanker paru-paru sel kecil (SCLC) dan kanker paru-paru non-sel kecil (SCLC). kanker paru-paru (NSCLC).

Dalam masalah klasifikasi, kita bertujuan untuk menggambar batas antara dua kelas titik seperti yang ditunjukkan pada Gambar 11.4. Dua kelas titik di gambarnya adalah lingkaran dan tanda plus. Batas linier (seperti garis atau bidang) yang mempunyai kesalahan paling rendah tidak dapat ditarik di antara kedua garis atau bidang tersebut. dua set titik. Jaringan saraf dapat digunakan untuk menggambar garis non-linier batas.

Salah satu fungsi kerugian umum untuk masalah klasifikasi adalah kerugian entropi silang. Hal ini didefinisikan sebagai

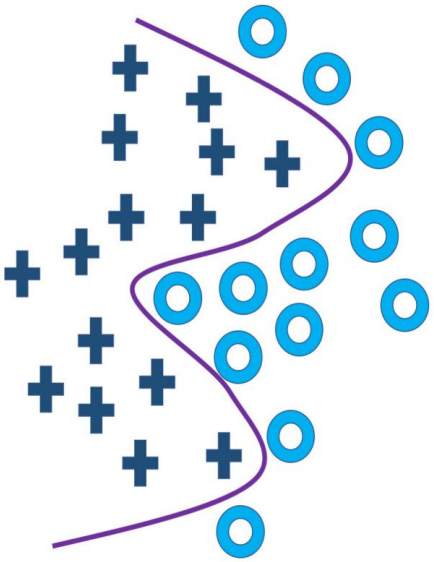
$$L = \ddot{y} \ylogi\hat{} \tag{11.15}$$

di mana y merupakan nilai aktual dan ˆy merupakan nilai prediksi.

Karena fungsi kerugian berbeda dibandingkan dengan probabilitas regresi, maka lem, turunan seperti yang $\frac{\ddot{y}L}{\ddot{y}W}$ akan menghasilkan persamaan yang berbeda. dipadankan dengan turunan untuk masalah regresi. Namun pendekatannya tetap sama.

11.6 Contoh Kode Jaringan Syaraf

Paket pembelajaran mendalam populer saat ini seperti Tensor-flow [ABC+16], Keras [C+20], dll., mengharuskan programmer untuk mendefinisikan perambatan maju sedangkan perambatan balik ditangani oleh kemasan.

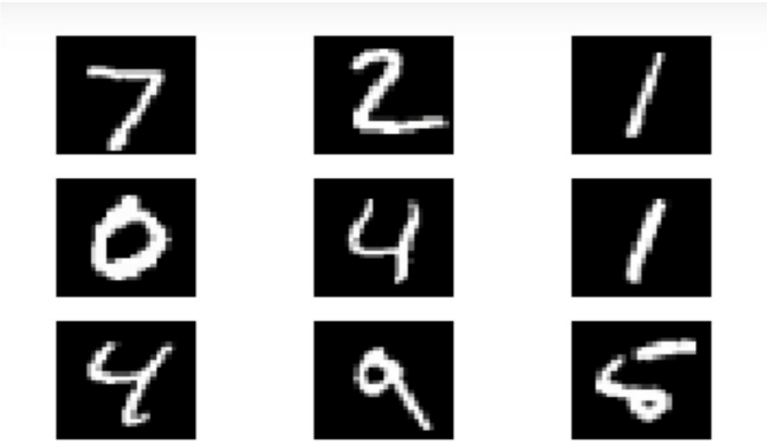


GAMBAR 11.4: Jaringan saraf untuk masalah klasifikasi menggambar batas non-linier antara dua kelas titik.

Dalam contoh di bawah ini, kami mendefinisikan jaringan saraf untuk memecahkan masalah mengidentifikasi digit tulisan tangan dari kumpulan data MNIST [LCB10], kumpulan data gambar populer untuk pembandingan aplikasi pembelajaran mesin dan pembelajaran mendalam. Gambar 11.5 menunjukkan beberapa gambar representatif dari kumpulan data MNIST. Setiap gambar berukuran 28 piksel kali 28 piksel. Jumlah total piksel = 28 x 28 = 784. Gambar-gambar tersebut berisi satu digit yang digambar dengan tangan. Seperti yang dapat dilihat pada gambar, dua angka mungkin tidak terlihat sama pada dua gambar yang berbeda. Tugasnya adalah mengidentifikasi digit pada gambar, dengan mempertimbangkan gambar itu sendiri. Gambar tersebut adalah input dan outputnya adalah salah satu dari 10 kelas (angka antara 0 dan 9).

Kita mulai dengan mengimpor semua modul yang diperlukan di Keras, khususnya Secara umum model Sequential dan lapisan Dense. Model Sequential memungkinkan pendefinisian sekumpulan lapisan. Dalam pembahasan matematis, kami mendefinisikan 2 lapisan. Dalam Keras, lapisan-lapisan ini dapat didefinisikan menggunakan kelas lapisan Dense. Tumpukan lapisan-lapisan ini membentuk lapisan Sequential.

Kami memuat dataset MNIST menggunakan fungsionalitas praktis (keras.datasets.mnist.load data) yang tersedia di Keras. Ini memuat dataset MNIST



GAMBAR 11.5: Beberapa contoh data dari dataset MNIST [LCB10].

data pelatihan dan data pengujian. Jumlah gambar dalam set data pelatihan adalah 60.000 dan jumlah gambar dalam set data pengujian adalah 10.000. Setiap gambar disimpan sebagai vektor sepanjang 784 piksel dengan presisi 8 bit (yaitu, nilai piksel berada di antara 0 dan 255). Y yang sesuai untuk setiap gambar adalah angka tunggal yang sesuai dengan digit dalam gambar tersebut.

Kemudian kami menormalkan gambar dengan membagi setiap nilai piksel dengan 255 dan menguranginya dengan 0,5. Dengan demikian, gambar yang dinormalisasi akan memiliki nilai piksel antara -0,5 dan +0,5.

Model dibangun dengan melewatkan 3 lapisan Padat ke kelas Berurutan. Lapisan pertama memiliki 64 node, lapisan kedua memiliki 64 node. 2 lapisan pertama menggunakan fungsi aktivasi Rectified Linear Unit (RELU) untuk non-linearitas. Lapisan terakhir menghasilkan vektor dengan panjang 10. Vektor ini dilewatkan melalui fungsi softmax (Persamaan 11.16). Output dari fungsi softmax adalah distribusi probabilitas karena setiap nilai sesuai dengan probabilitas digit tertentu dan juga jumlah semua nilai dalam vektor sama dengan 1. Setelah kita memperoleh vektor ini, menentukan digit yang sesuai dapat dilakukan dengan menemukan posisi dalam vektor dengan nilai probabilitas tertinggi.

$$\text{jika} = \frac{\text{dari e}^{xi}}{\text{contoh}} \tag{11.16}$$

Kami akan melewati model melalui proses optimasi dengan memanggil fungsi fit. Kami menjalankan model melalui 5 periode, di mana setiap periode didefinisikan sebagai kunjungan ke semua gambar dalam set data pelatihan. Biasanya kami memasukkan sekumpulan gambar untuk pelatihan, bukan satu gambar pada satu waktu. Dalam contoh ini, kami menggunakan sekumpulan 32 gambar, yang berarti dalam setiap pelatihan sekumpulan acak 32 gambar dan label yang sesuai dilewatkan.

```
impor numpy sebagai np dari
keras.models impor Sequential dari keras.layers impor
Dense dari keras.utils impor to_categorical dari
keras.datasets impor mnist


# Ambil data kereta dan uji.
(kereta_x, kereta_y), (uji_x, uji_y) = mnist.muat_data()

# Normalisasikan gambar sehingga semua nilai piksel
# berada antara -0,5 dan +0,5.
kereta_x = (kereta_x / 255) - 0,5
uji_x = (uji_x / 255) - 0,5

# Ubah bentuk kereta dan gambar uji ke ukuran vektor panjang 784. x_train = x_train.reshape((-1,
784)) x_test = x_test.reshape((-1, 784))


# Tentukan model jaringan saraf dengan 2 lapisan tersembunyi
# berukuran 64 node masing-masing.
model = Berurut([
        Padat(64, aktivasi='relu', bentuk_input=(784,)),
        Padat(64, aktivasi='relu'),
        Padat(10, aktivasi='softmax'),
])
```

```
# Kompilasi model menggunakan pengoptimal Adam dan gunakan
# kerugian entropi silang.
model.compile(optimizer='adam',
                    kerugian='kategori_crossentropi',
                    metrik=['akurasi'])

# Latih modelnya.
model.fit(kereta_x, ke_kategori(kereta_y), epoch=5,
                ukuran_batch=32)
```

Output berisi hasil pelatihan 5 epoch. Seperti yang dapat dilihat, nilai kerugian entropi silang menurun seiring berjalannya pelatihan. Dimulai pada 0,3501 dan akhirnya berakhir pada 0,0975. Demikian pula, akurasi meningkat seiring berjalannya pelatihan dari 0,8946 menjadi 0,9697.

```
Epoch 1/5
60000/60000 [===] - 3s 58us/langkah - kehilangan: 0,3501 - akurasi:
0.8946
Epoch 2/5
60000/60000 [===] - 3s 56us/langkah - kehilangan: 0,1790 - akurasi:
0.9457
Epoch 3/5
60000/60000 [===] - 3s 55us/langkah - kehilangan: 0,1357 - akurasi:
0,9576 tahun
Epoch 4/5
60000/60000 [===] - 3s 55us/langkah - kehilangan: 0,1129 - akurasi:
0,9649 tahun
Epoch 5/5
60000/60000 [===] - 3s 57us/langkah - kehilangan: 0,0975 - akurasi:
0,9697 tahun
```

Pembaca yang tertarik harus berkonsultasi dengan dokumentasi Keras untuk informasi lebih lanjut rincian.

11.7 Ringkasan

- Jaringan saraf adalah aproksimator fungsi universal. Dalam melatih jaringan saraf, kami menyesuaikan kurva non-linier menggunakan data yang tersedia.
- Untuk memperoleh non-linieritas dalam jaringan saraf, kami menggabungkan fungsi linier dengan fungsi non-linier seperti sigmoid, RELU, dll.
- Parameter kurva non-linier dipelajari melalui proses propagasi balik.
- Jaringan saraf dapat digunakan untuk regresi dan klasifikasi masalah.

11.8 Latihan

1. Anda diberi neuron yang melakukan penjumlahan y = x1*w1+x2*w2, di mana x1 dan x2 adalah input dan w1 dan w2 adalah bobot. Tuliskan persamaan back-propagation untuknya. Tuliskan juga persamaan update untuk w1 dan w2.
2. Dalam jaringan saraf, kita menggabungkan fungsi linear W x + b dengan fungsi non-linear. Kita menumpuk lapisan-lapisan ini bersama-sama untuk menghasilkan fungsi non-linear yang kompleksnya tak terbatas. Apa yang akan terjadi jika kita tidak menggunakan fungsi non-linear tetapi tetap menumpuk lapisan? Kurva seperti apa yang dapat kita buat?
3. Mengapa sigmoid tidak lagi populer sebagai fungsi aktivasi? Lakukan penelitian tentang topik ini.

Bab 12

Jaringan Syaraf Konvolusional

12.1 Pendahuluan

Jaringan saraf konvolusi (CNN) adalah model penglihatan matematis yang terinspirasi dari biologi. Perjalanan ini dimulai dengan sukses melalui karya David Hubel dan Torsten Wiesel yang memenangkan hadiah Nobel dalam bidang Fisiologi atau Kedokteran pada tahun 1981 untuk karya ini. Karya Hubel dan Weisel diringkas dengan baik oleh siaran pers komite Nobel ([ppr20]) dari tahun 1981. Paragraf berikut adalah reproduksi dari siaran pers

melepaskan:

"...analisis korteks visual terhadap pesan terkode dari retina berlangsung seolah-olah sel-sel tertentu membaca huruf-huruf sederhana dalam pesan dan menyusunnya menjadi suku kata yang kemudian dibaca oleh sel-sel lain, yang kemudian menyusun suku kata tersebut menjadi kata-kata, dan akhirnya dibaca oleh sel-sel lain yang menyusun kata-kata menjadi kalimat yang dikirim ke pusat-pusat yang lebih tinggi di otak, tempat kesan visual berasal dan memori gambar tersebut disimpan."

Seperti yang ditunjukkan kutipan tersebut, Hubel dan Wiesel menemukan bahwa otak memiliki serangkaian neuron. Neuron yang paling dekat dengan retina mendeteksi bentuk-bentuk sederhana. seperti garis-garis dalam orientasi yang berbeda. Neuron berikutnya mendeteksi bentuk-bentuk yang kompleks seperti kurva. Neuron-neuron di hilir mendeteksi bentuk-bentuk yang lebih kompleks seperti hidung, telinga, dll.

Pemahaman tentang korteks visual otak membuka jalan bagi pemodelan matematika jalur visual. Pekerjaan pertama yang berhasil dilakukan oleh Kunihiko Fukushima ([Fuk80]). Ia mendemonstrasikan

model hierarkis menggunakan konvolusi dan downsampling. Konvolusi memungkinkan tampilan hanya sebagian gambar atau video saat diproses. Downsampling dilakukan dengan cara merata-ratakan. Bertahun-tahun kemudian, metode berbeda yang disebut “maxpooling” diperkenalkan yang masih digunakan hari ini dan akan dibahas nanti dalam bab ini. Terobosan besar berikutnya adalah karya Yann Lecun [LBD+89] yang memperkenalkan pendekatan back-propagation untuk mempelajari parameter CNN.

Dengan tersedianya data dalam jumlah besar, penyimpanan lebih murah, kekuatan komputasi dan perangkat lunak, CNN telah menjadi alat yang tepat untuk memecahkan masalah pemrosesan gambar dan masalah visi komputer di semua bidang sains dan teknik.

12.2 Konvolusi

Kami membahas proses penerapan konvolusi pada gambar di Bab 4. Pada bagian ini, kita akan membahas konvolusi dari perspektif CNN. Dalam contoh di Bab 4, konvolusi adalah dilakukan dengan menggunakan filter 5-kali-5 dimana setiap elemen dalam filter memiliki nilai dari $\frac{1}{25}$. Dalam CNN, nilai dalam filter ditentukan oleh proses pembelajaran (yaitu, proses back-propagation).

Selain itu, tidak seperti contoh-contoh dari Bab 4, dalam CNN terdapat beberapa filter yang digunakan. Filter-filter ini disusun menjadi beberapa lapisan. Lapisan pertama dirancang untuk mendeteksi objek sederhana seperti garis. Karena ada banyak kemungkinan konfigurasi (kemiringan) untuk garis, lapisan pertama mungkin memiliki beberapa filter untuk mendeteksi garis pada semua orientasi ini. Lapisan kedua dirancang untuk mendeteksi kurva. Karena ada lebih banyak konfigurasi untuk kurva dibandingkan dengan garis, jumlah filter di lapisan kedua biasanya lebih banyak dari jumlah filter di lapisan pertama. Modern CNN1 biasanya memiliki lebih dari 2 lapisan.

1CNN baru berusia 40 tahun. Kami membedakan arsitektur CNN dari 10 tahun terakhir dari yang menggunakan kata modern.

12.3 Pengumpulan Maksimum

Maxpooling adalah teknik pengurangan dimensionalitas. Teknik ini mengambil input seperti gambar dan mengurangi ukurannya menggunakan nilai maksimum di antara tetangganya. Efek penggantian nilai piksel dengan nilai maksimum tetangganya menghasilkan representasi gambar yang abstrak. Kami akan menunjukkannya dengan sebuah contoh.

Mari kita pertimbangkan gambar kecil (Gambar 12.1(a)) berukuran 4x4 dan juga pertimbangkan filter maxpooling berukuran 2x2 yang ditempatkan di sudut kiri atas gambar. Dalam proses maxpooling, kita akan menemukan nilai maksimum dari wilayah 2x2 yang berisi nilai 10, 6, 8 dan 2. Kita akan membuat gambar baru (Gambar 12.1(b)) di mana kita akan menggunakan nilai maksimum (10) dari perhitungan sebelumnya. Kita kemudian akan memindahkan filter maxpooling ini dengan 2 langkah (disebut stride) di atas gambar dan menemukan wilayah berikutnya pada gambar yang terdiri dari nilai 4, 6, 12 dan 5. Nilai maksimumnya yaitu 12 akan digunakan untuk piksel berikutnya dalam gambar keluaran. Setelah satu baris operasi maxpooling selesai, kita pindah 2 baris di bawah dan melanjutkan proses ini.

(a) Gambar masukan. (b) Citra maxpooled.

GAMBAR 12.1: Contoh penerapan maxpooling pada sub-gambar.

Jika sebuah bayangan berukuran NxN dan kita bergerak dengan langkah 2x2, maka Gambar keluaran akan berukuran lebih kecil $\frac{N \times N}{\text{ukuran besar}} \rightarrow \frac{N}{2}$. Langkah yang lebih tinggi dapat digunakan untuk jika gambar diperkecil lebih lanjut.

Output dari lapisan konvolusi dan maxpooling akhirnya diteruskan ke pengklasifikasi atau regresor, yang biasanya dibangun menggunakan jaringan saraf seperti yang dibahas dalam bab sebelumnya. Hal ini disebabkan oleh fakta bahwa lapisan konvolusi dan maxpooling merupakan pengkondisi data yang menyiapkan data untuk pengklasifikasi akhir atau regresor.

12.4 Arsitektur LeNet

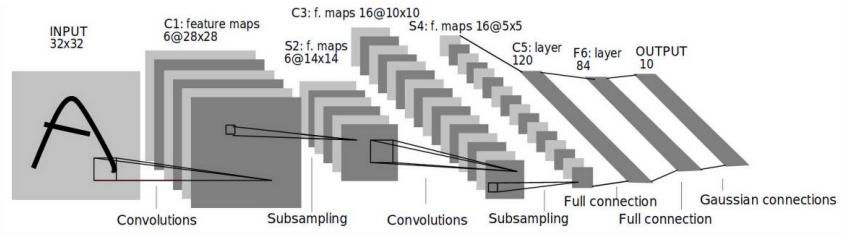
Kita akan menggunakan lapisan konvolusi dan maxpooling untuk membangun LeNet [LBBH98], salah satu CNN pertama yang merevolusi bidang visi komputer. Citra input (Gambar 12.2) dilewatkan ke serangkaian 6 konvolusi di lapisan pertama. Output dari lapisan konvolusi pertama disubsampel menggunakan maxpooling, dan kemudian dilewatkan ke lapisan konvolusi kedua, yang berisi 16 filter. Output dari lapisan konvolusi kedua dilewatkan ke lapisan maxpooling kedua. Output dari lapisan ini kemudian diratakan menjadi vektor dan dilewatkan ke pengklasifikasi atau regresor berbasis jaringan saraf.

Idealnya, pengklasifikasi apa pun seperti Support Vector Machine SVM atau regresi logistik dapat digunakan. Namun, jaringan saraf, seperti yang dibahas dalam bab terakhir, lebih disukai.

Pada bab terakhir, kita membahas bahwa parameter sistem dapat dipelajari menggunakan proses back-propagation. Hal yang sama berlaku untuk parameter (nilai dalam filter) CNN juga.

Untuk setiap gambar masukan atau sekumpulan gambar, kami melakukan propagasi maju melalui lapisan konvolusi, maxpooling, dan jaringan saraf. Kemudian kami menentukan kerugiannya. Kemudian kami melakukan propagasi balik melalui lapisan jaringan saraf diikuti oleh propagasi balik melalui lapisan konvolusi dan memperbarui parameter (bobot)

dan bias) menggunakan persamaan pembaruan. Proses ini diulang dengan semua data yang tersedia.



GAMBAR 12.2: Diagram arsitektur LeNet.

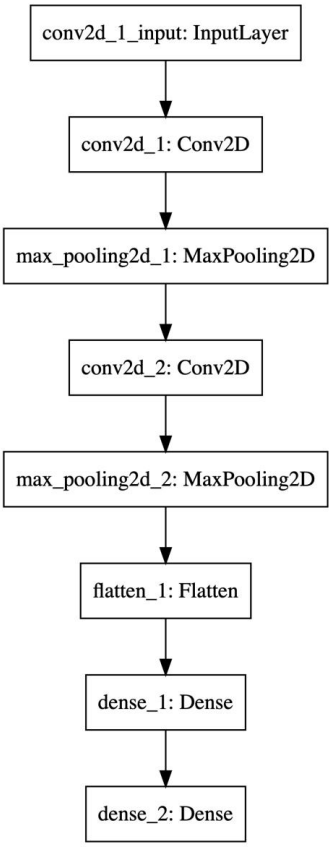
Dalam contoh di bawah, kami mendefinisikan LeNet CNN untuk memecahkan masalah mengidentifikasi digit tulisan tangan dari kumpulan data MNIST, yang juga kami gunakan dalam bab terakhir.

Kita mulai dengan mengimpor semua fungsi yang diperlukan dalam Keras, khususnya model Sequential, lapisan Dense, lapisan Conv2D, dan lapisan Max-Pooling2D. Model Sequential memungkinkan pendefinisian sekumpulan lapisan.
Daftar lapisan secara berurutan ditunjukkan pada Gambar 12.3.

Lapisan pertama adalah lapisan input yang mengambil gambar berukuran 28x28x1. Lapisan kedua adalah lapisan konvolusi dengan 32 filter. Lapisan ketiga 2. Lapisan keempat maxpooling yang mengurangi ukuran gambar sebesar dan lapisan kelima $\frac{1}{}$adalah lapisan adalah set kedua dari lapisan konvolusi dengan 64 filter dan lapisan maxpooling kedua yang mengurangi ukuran gambar sebesar 2. Gambar keluaran lapisan maxpooling kedua$\frac{1}{}$ diratakan dan dilewatkan melalui dua lapisan jaringan saraf untuk menghasilkan prediksi keluaran.

Dalam contoh ini, kami menggunakan gambar berukuran 28x28x1. Setelah melewati lapisan konvolusi pertama, kami akan memperoleh volume 3D berukuran 28x28x32. Lapisan maxpooling pertama mengurangi ukuran sebesar Volume ini melewati $\frac{-}{12}$ke 14x14x32. lapisan konvolusi kedua yang menghasilkan volume berukuran 14x14x64. Lapisan maxpooling kedua mengurangi gambar menjadi 7x7x64. Vektor yang diratakan akan berukuran 3136 yang merupakan hasil perkalian dari 7, 7, dan 64.

Dalam kode tersebut, kami memuat dataset pelatihan dan pengujian dengan menggunakan metode 'mnist.load data()'. Nilai x (piksel gambar) dinormalisasi



GAMBAR 12.3: Model LeNet Keras.

berada dalam kisaran [ÿ0,5, 0,5]. Mereka kemudian dibentuk ulang dari bentuk aslinya 50000x784 menjadi 50000x28x28x1.

Model sekuensial dibuat dan berbagai lapisan ditambahkan seperti yang dijelaskan sebelumnya. Di semua lapisan, non-linearitas RELU ditambahkan. Lapisan terakhir dilewatkan melalui softmax untuk memperoleh distribusi probabilitas yang dapat dievaluasi untuk kerugian entropi silang. Akhirnya, kami mengevaluasi model untuk akurasi menggunakan data uji.

```
impor numpy sebagai
np impor keras
dari keras.datasets impor mnist
```

```
dari keras.models impor Sequential dari keras.layers
impor Dense, Flatten dari keras.layers impor Conv2D,
MaxPooling2D dari keras.utils impor to_categorical


# Ukuran gambar adalah 28x28x1 saluran. input_shape
= (28, 28, 1)
ukuran_batch = 64
# jumlah kemungkinan hasil [0-9]
nkelas = 10
zaman = 3

# Ambil data kereta dan uji.
(kereta_x, kereta_y), (uji_x, uji_y) = mnist.muat_data()

# Normalisasikan gambar sehingga semua nilai piksel
# berada antara -0,5 dan +0,5.
kereta_x = (kereta_x / 255) - 0,5
uji_x = (uji_x / 255) - 0,5

# Ubah bentuk kereta dan gambar uji ke ukuran 28x28x1. x_train =
x_train.reshape((x_train.shape[0], *input_shape)) x_test = x_test.reshape((x_test.shape[0],
*input_shape))

# Tentukan model CNN dengan 2 lapisan konvolusi dan # 2 lapisan pengumpulan
maksimal diikuti oleh jaringan saraf # dengan 1 lapisan tersembunyi berukuran
128 node. model = Sequential() model.add(Conv2D(32,
kernel_size=(3, 3),
                                aktivasi='relu',
                                bentuk_input=bentuk_input))
model.tambahkan(MaxPooling2D(ukuran_pool=(2, 2)))
model.tambahkan(Conv2D(64, (3, 3), aktivasi='relu'))
```

```
model.tambahkan(MaxPooling2D(ukuran_kolam=(2, 2)))
model.tambahkan(Ratakan())
model.tambahkan(Padat(128, aktivasi='relu'))
model.tambahkan(Dense(nclasses, aktivasi='softmax'))

# Kompilasi model menggunakan pengoptimal Adam dan gunakan
# kerugian entropi silang.
model.compile(optimizer='adam',
                        kerugian='kategori_crossentropi',
                        metrik=['akurasi'])

# Latih modelnya.
model.fit(kereta_x, ke_kategori(kereta_y), periode=periode,
                    ukuran_batch=ukuran_batch)

# Evaluasi modelnya.
skor = model.evaluasi(uji_x, untuk_kategori(uji_y),
        bertele-tele=0)
cetak('Kerugian tes:', skor[0])
cetak('Keakuratan tes:', skor[1])
```

Dalam kasus CNN, dibandingkan dengan ujian jaringan saraf,
Dari bab sebelumnya, Anda akan melihat bahwa kami mencapai nilai akurasi yang tinggi
dalam lebih sedikit epoch meskipun setiap epoch membutuhkan waktu sedikit lebih lama.

```
Epoch 1/3
60000/60000 [====] - 33d 555us/langkah - kerugian: 0,1683 - akurasi:
0,9504 Epoch 2/3
60000/60000
[====] - 27d 444us/langkah - kerugian: 0,0493 - akurasi: 0,9847 Epoch
3/3 60000/60000
[====] - 48d
792us/langkah - kerugian: 0,0331 -
akurasi: 0.9898
```

Kerugian pengujian: 0,03353308427521261

Akurasi pengujian: 0.9894000291824341

12.5 Ringkasan

- CNN awalnya dikembangkan sebagai model matematika penglihatan.
  Oleh karena itu, mereka sangat cocok untuk memecahkan masalah visi komputer.
- CNN dibuat dengan komposisi lapisan konvolusi dan maxpooling diikuti oleh
  pengklasifikasi atau regresor, yang biasanya berupa jaringan saraf.
- Parameter lapisan konvolusi dipelajari menggunakan
  proses propagasi balik.

12.6 Latihan

1. Apa pengaruh penambahan jumlah lapisan konvolusi
   dalam jaringan saraf?
2. Ubah kode di atas dan jalankan dengan kumpulan data FashionMNIST yang tersedia di
   https://github.com/zalandoresearch/fashion-mnist . Kumpulan data ini juga memiliki 10
   kategori seperti celana panjang, sepatu, dll.