

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

Supervisi Pembelajaran Kecerdasan buatan (AI)



YAYASAN PRIMA AGUS TEKNIK



Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

Supervisi Pembelajaran Kecerdasan buatan (AI)



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

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

ISBN 978-634-7227-30-0 (PDF)



9

786347

227300

Supervisi Pembelajaran Kecerdasan buatan (AI)

Penulis :

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

ISBN : 978-634-7227-30-0

Editor :

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

Penyunting :

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

Desain Sampul dan Tata Letak :

Irdha Yuniarto, S.Ds., M.Kom

Penebit :

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

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :

Jl. Majapahit no 605 Semarang

Telp. 08122925000

Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :

Universitas STEKOM

Jl. Majapahit no 605 Semarang

Telp. 08122925000

Fax. 024-6710144

Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin dari penulis

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga buku berjudul "***Supervisi Pembelajaran Kecerdasan Buatan (AI)***" ini dapat diselesaikan dengan baik. Buku ini disusun sebagai referensi komprehensif untuk memperdalam pemahaman tentang supervisi pembelajaran dalam kecerdasan buatan—teknologi yang menjadi kunci kemajuan di era digital.

Buku ini membahas berbagai aspek penting dalam pembelajaran AI secara terstruktur. Dimulai dari dasar-dasar pengetahuan dalam pembelajaran yang mencakup formulasi logis, pembelajaran berbasis penjelasan, hingga pemrograman logika induktif, yang membentuk kerangka konseptual kuat bagi pembaca. Selanjutnya, pembaca akan menemukan pembahasan mendalam terkait model probabilistik pembelajaran dan pendekatan pembelajaran penguatan, yang mencakup teori, algoritma, dan aplikasi praktisnya dalam pengembangan sistem AI adaptif.

Pemrosesan bahasa alami dan bahasa alami untuk komunikasi menjadi fokus bab berikutnya, membahas cara komputer memahami, mengklasifikasi, dan menginterpretasi bahasa manusia, yang sangat penting dalam interaksi antara manusia dan mesin. Bab tentang persepsi dan robotik menguraikan bagaimana AI dapat melihat, mengenali objek, dan mengambil tindakan nyata di dunia fisik melalui perangkat keras dan perangkat lunak robotik. Tidak kalah penting, buku ini membahas landasan filosofis AI, mengupas pertanyaan kritis tentang kecerdasan mesin, termasuk perdebatan antara AI lemah dan AI kuat, serta etika dan risiko dalam pengembangan teknologi ini. Bab terakhir membawa pembaca pada perspektif masa kini dan masa depan AI, dengan analisis tentang komponen agen dan arsitektur agen yang menjadi fondasi sistem AI modern.

Kami menyadari masih terdapat keterbatasan dalam penyusunan buku ini, oleh karena itu kami sangat mengharapkan kritik dan saran demi penyempurnaan. Ucapan terima kasih kami haturkan kepada semua pihak yang telah membantu terselesaikannya buku ini, semoga dapat memberikan manfaat besar bagi mahasiswa, peneliti, praktisi, dan pembaca luas dalam mengembangkan ilmu dan aplikasi kecerdasan buatan di Indonesia dan dunia.

Semarang, Agustus 2025

Penulis

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM.

DAFTAR ISI

Halaman judul	i
Kata Pengantar	ii
Daftar Isi	iii
BAB 1 PENGETAHUAN DALAM PEMBELAJARAN.....	1
1.1 Formulasi Logis Pembelajaran	1
1.2 Pengetahuan Dalam Pembelajaran.....	10
1.3 Pembelajaran Berbasis Penjelasan	14
1.4 Belajar Menggunakan Informasi Relevansi.....	19
1.5 Pemrograman Logika Induktif.....	23
1.6 Ringkasan	34
BAB 2 MODEL PROBABILISTIK PEMBELAJARAN	36
2.1 Pembelajaran Statistik	36
2.2 Pembelajaran Dengan Data Lengkap	40
2.3 Belajar Dengan Variabel Tersembunyi Algoritma Em	51
2.4 Ringkasan	61
BAB 3 PEMBALIAN PENGUATAN	63
3.1 Pendahuluan	63
3.2 Pembelajaran Penguatan Pasif	64
3.3 Pembelajaran Penguatan Aktif	72
3.4 Generalisasi Dalam Pembelajaran Penguatan	79
3.5 Pencarian Kebijakan.....	82
3.6 Aplikasi Pembelajaran Penguatan.....	85
3.7 Ringkasan	88
BAB 4 PEMROSESAN BAHASA ALAMI	90
4.1 Model Bahasa	90
4.2 Klasifikasi Teks.....	95
4.3 Pencarian Informasi	98
4.4 Ekstraksi Informasi	105
4.5 Ringkasan	115
BAB 5 BAHASA ALAMI UNTUK KOMUNIKASI	117
5.1 Tata Bahasa Struktur Frasa	117
5.2 Analisis Sintaktik (Parsing)	121
5.3 Tata Bahasa Yang Ditambahkan Dan Interpretasi Semantik	126
5.4 Terjemahan Mesin	138
5.5 Pengenalan Pidato	144
5.6 Ringkasan	150
BAB 6 PERSEPSI	152
6.1 Pembentukan Gambar	153
6.2 Operasi Pemrosesan Citra Awal	160
6.3 Pengenalan Objek Berdasarkan Penampilan	169

6.4	Merekonstruksi Dunia 3D	175
6.5	Pengenalan Objek Dari Informasi Struktural	186
6.6	Menggunakan Visi.....	190
BAB 7	ROBOTIK	196
7.1	Pendahuluan	196
7.2	Perangkat Keras Robot.....	198
7.3	Persepsi Robotik.....	203
7.4	Perencanaan Bergerak	212
7.5	Perencanaan Pergerakan Yang Tak Pasti	221
7.6	Bergerak	225
7.7	Arsitektur Perangkat Lunak Robotik	232
7.8	Domain Aplikasi	237
7.9	Ringkasan	241
BAB 8	LANDASAN FILSAFAT	243
8.1	AI Yang Lemah: Dapatkah Mesin Bertindak Secara Cerdas?	243
8.2	AI Yang Kuat: Dapatkah Mesin Benar-Benar Berpikir?	249
8.3	Etika Dan Risiko Pengembangan Kecerdasan Buatan	258
8.4	Ringkasan	266
BAB 9	AI: MASA KINI DAN MASA DEPAN	267
9.1	Komponen Agen	267
9.2	Arsitektur Agen	269
9.3	Apakah Kita Menuju Arah Yang Benar?	271
DAFTAR PUSTAKA		275

BAB 1

PENGETAHUAN DALAM PEMBELAJARAN

Dalam semua pendekatan pembelajaran yang dijelaskan di bab sebelumnya, idenya adalah membangun sebuah fungsi yang memiliki perilaku input-output yang teramati dalam data. Dalam setiap kasus, metode pembelajaran dapat dipahami sebagai pencarian ruang hipotesis untuk menemukan fungsi yang sesuai, dimulai dari asumsi yang sangat mendasar tentang bentuk fungsi tersebut, seperti "polinomial derajat dua" atau "pohon keputusan" dan mungkin preferensi untuk hipotesis yang lebih sederhana.

Melakukan hal ini sama saja dengan mengatakan bahwa sebelum Anda dapat mempelajari sesuatu yang baru, Anda harus terlebih dahulu melupakan (hampir) semua yang Anda ketahui. Dalam bab ini, kami mempelajari metode pembelajaran yang dapat memanfaatkan pengetahuan sebelumnya tentang dunia. Dalam kebanyakan kasus, pengetahuan sebelumnya direpresentasikan sebagai teori logika orde pertama yang umum; dengan demikian, untuk pertama kalinya, kami menggabungkan karya tentang representasi pengetahuan dan pembelajaran.

1.1 FORMULASI LOGIS PEMBELAJARAN

Pembelajaran induktif murni sebagai proses menemukan hipotesis yang sesuai dengan contoh-contoh yang diamati. Di sini, kami mengkhususkan definisi ini pada kasus di mana hipotesis direpresentasikan oleh serangkaian kalimat logis. Deskripsi dan klasifikasi contoh juga akan berupa kalimat logis, dan contoh baru dapat diklasifikasikan dengan menyimpulkan kalimat klasifikasi dari hipotesis dan deskripsi contoh tersebut.

Pendekatan ini memungkinkan konstruksi hipotesis secara bertahap, satu kalimat pada satu waktu. Pendekatan ini juga memungkinkan pengetahuan awal, karena kalimat-kalimat yang sudah diketahui dapat membantu dalam klasifikasi contoh-contoh baru. Formulasi logis pembelajaran mungkin tampak seperti pekerjaan tambahan yang banyak pada awalnya, tetapi ternyata dapat memperjelas banyak masalah dalam pembelajaran. Hal ini memungkinkan kita untuk melampaui metode pembelajaran sederhana di Bab 18 dengan memanfaatkan sepenuhnya kekuatan inferensi logis dalam pembelajaran.

Contoh dan Hipotesis

Ingat kembali Bab 18 tentang masalah pembelajaran restoran: mempelajari aturan untuk memutuskan apakah akan menunggu meja. Contoh-contoh tersebut dijelaskan oleh atribut seperti Alternatif, Bar, Jum/Sab, dan seterusnya. Dalam konteks logis, sebuah contoh dijelaskan oleh kalimat logis; atribut-atribut tersebut menjadi predikat unary. Mari kita sebut contoh ke- i secara umum sebagai X_i . Misalnya, contoh pertama dari Gambar 18.3 (halaman 700) dijelaskan oleh kalimat-kalimat

$$\text{Alternate}(X_1) \wedge \neg \text{Bar}(X_1) \wedge \neg \text{Fri/Sat}(X_1) \wedge \text{Hungry}(X_1) \wedge \dots$$

Kita akan menggunakan notasi $D_i(X_i)$ untuk merujuk pada deskripsi X_i , di mana D_i dapat berupa ekspresi logika apa pun yang hanya menggunakan satu argumen. Klasifikasi contoh ini diberikan oleh literal yang menggunakan predikat tujuan, dalam hal ini

$$WillWait(X_1) \text{ or } \neg WillWait(X_1) .$$

Dengan demikian, keseluruhan set pelatihan dapat dinyatakan sebagai gabungan dari semua deskripsi contoh dan literal tujuan. Tujuan pembelajaran induktif secara umum adalah untuk menemukan hipotesis yang mengklasifikasikan contoh dengan baik dan menggeneralisasi dengan baik ke contoh-contoh baru. Di sini kita membahas hipotesis yang dinyatakan dalam logika; setiap hipotesis h_j akan berbentuk

$$\forall x \text{ Goal}(x) \Leftrightarrow C_j(x) ,$$

di mana $C_j(x)$ adalah definisi kandidat suatu ekspresi yang melibatkan predikat atribut. Misalnya, pohon keputusan dapat diinterpretasikan sebagai ekspresi logis dalam bentuk ini. Dengan demikian, pohon pada Gambar 18.6 (halaman 702) mengekspresikan definisi logis berikut (yang akan kita sebut hr untuk referensi di masa mendatang):

$$\begin{aligned} \forall r \text{ WillWait}(r) \Leftrightarrow & Patrons(r, Some) \\ & \vee Patrons(r, Full) \wedge Hungry(r) \wedge Type(r, French) \\ & \vee Patrons(r, Full) \wedge Hungry(r) \wedge Type(r, Thai) \\ & \quad \wedge Fri/Sat(r) \\ & \vee Patrons(r, Full) \wedge Hungry(r) \wedge Type(r, Burger) . \end{aligned} \quad (1.1)$$

Setiap hipotesis memprediksi bahwa sekumpulan contoh tertentu yaitu, contoh-contoh yang memenuhi definisi kandidatnya akan menjadi contoh predikat tujuan. Kumpulan ini disebut perluasan predikat. Oleh karena itu, dua hipotesis dengan perluasan yang berbeda secara logis tidak konsisten satu sama lain, karena prediksi mereka tidak sependapat untuk setidaknya satu contoh. Jika keduanya memiliki perluasan yang sama, keduanya secara logis setara.

Ruang hipotesis H adalah himpunan semua hipotesis $\{h_1, \dots, h_n\}$ yang dirancang untuk dijalankan oleh algoritma pembelajaran. Misalnya, algoritma *Pembelajaran Pohon Keputusan* dapat menjalankan hipotesis pohon keputusan apa pun yang didefinisikan berdasarkan atribut yang diberikan; oleh karena itu, ruang hipotesisnya terdiri dari semua pohon keputusan ini. Agaknya, algoritma pembelajaran meyakini bahwa salah satu hipotesis itu benar; artinya, ia meyakini kalimat tersebut

$$h_1 \vee h_2 \vee h_3 \vee \dots \vee h_n . \quad (1.2)$$

Seiring munculnya contoh-contoh, hipotesis yang tidak konsisten dengan contoh-contoh tersebut dapat disingkirkan. Mari kita telaah konsep konsistensi ini lebih saksama. Tentu saja,

jika hipotesis h_j konsisten dengan seluruh set pelatihan, hipotesis tersebut harus konsisten dengan setiap contoh dalam set pelatihan. Apa artinya jika hipotesis tersebut tidak konsisten dengan sebuah contoh? Ada dua kemungkinan hal ini dapat terjadi:

- Sebuah contoh dapat menjadi negatif palsu untuk hipotesis, jika hipotesis menyatakan seharusnya negatif, tetapi kenyataannya positif. Misalnya, contoh baru X_{13} yang dijelaskan oleh Patrons ($X_{13}, Full$) $\wedge \neg Hungry(X_{13}) \wedge \dots \wedge WillWait(X_{13})$ akan menjadi negatif palsu untuk hipotesis hr yang diberikan sebelumnya. Dari hr dan deskripsi contoh, kita dapat menyimpulkan $WillWait(X_{13})$, yang merupakan apa yang dikatakan oleh contoh, dan $\neg WillWait(X_{13})$, yang merupakan apa yang diprediksi oleh hipotesis. Oleh karena itu, hipotesis dan contoh tersebut tidak konsisten secara logis.
- Sebuah contoh dapat menjadi positif palsu untuk hipotesis, jika hipotesis menyatakan seharusnya positif, tetapi kenyataannya negatif.

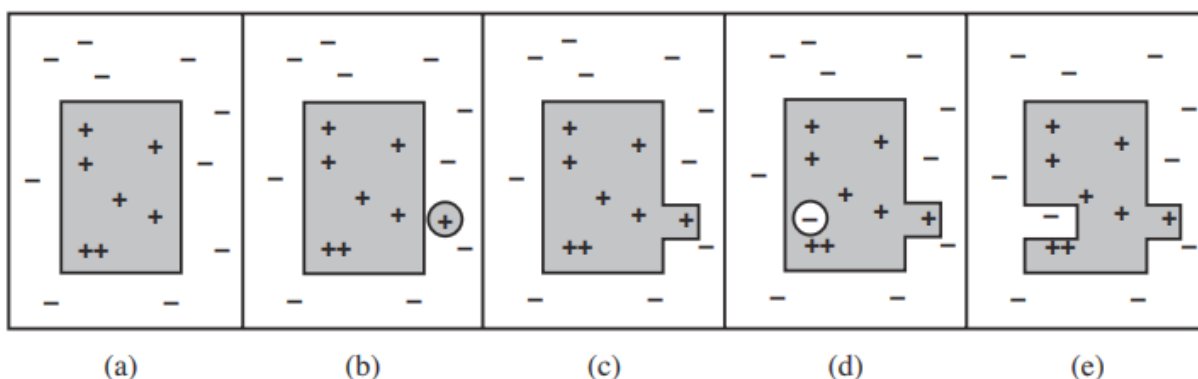
Jika sebuah contoh merupakan positif palsu atau negatif palsu untuk sebuah hipotesis, maka contoh dan hipotesis tersebut tidak konsisten secara logis satu sama lain. Dengan asumsi bahwa contoh tersebut merupakan observasi fakta yang benar, maka hipotesis dapat dikesampingkan. Secara logis, hal ini persis sama dengan aturan resolusi inferensi, di mana disjungsi hipotesis bersesuaian dengan sebuah klausa dan contoh tersebut bersesuaian dengan literal yang beresonansi dengan salah satu literal dalam klausa tersebut. Oleh karena itu, sistem inferensi logis biasa pada prinsipnya dapat belajar dari contoh tersebut dengan mengeliminasi satu atau lebih hipotesis. Misalkan, misalnya, contoh tersebut dilambangkan dengan kalimat I_1 , dan ruang hipotesisnya adalah $h_1 \vee h_2 \vee h_3 \vee h_4$. Kemudian, jika I_1 tidak konsisten dengan h_2 dan h_3 , sistem inferensi logis dapat menyimpulkan ruang hipotesis baru $h_1 \vee h_4$.

Oleh karena itu, kita dapat mengkarakterisasi pembelajaran induktif dalam lingkungan logis sebagai proses mengeliminasi hipotesis yang tidak konsisten dengan contoh secara bertahap, sehingga mempersempit kemungkinan. Karena ruang hipotesis biasanya luas (atau bahkan tak terhingga dalam kasus logika orde pertama), kami tidak menyarankan untuk mencoba membangun sistem pembelajaran menggunakan pembuktian teorema berbasis resolusi dan enumerasi lengkap ruang hipotesis. Sebaliknya, kami akan menjelaskan dua pendekatan yang menemukan hipotesis yang konsisten secara logis dengan upaya yang jauh lebih sedikit.

Pencarian hipotesis terbaik saat ini

Ide di balik pencarian hipotesis terbaik saat ini adalah mempertahankan satu hipotesis, dan menyesuaikannya ketika contoh-contoh baru muncul untuk menjaga konsistensi. Algoritme dasarnya dijelaskan oleh John Stuart Mill (1843), dan mungkin telah muncul lebih awal. Misalkan kita memiliki beberapa hipotesis seperti h_r , yang sudah cukup kita sukai. Selama setiap contoh baru konsisten, kita tidak perlu melakukan apa pun. Kemudian muncul contoh negatif palsu, X_{13} . Apa yang kita lakukan? Gambar 1.1(a) menunjukkan h_r secara skematis sebagai suatu daerah: segala sesuatu di dalam persegi panjang merupakan bagian dari perluasan hr . Contoh-contoh yang telah terlihat sejauh ini ditampilkan sebagai " + " atau

" – ", dan kita melihat bahwa h_r mengkategorikan semua contoh dengan tepat sebagai contoh positif atau negatif dari WillWait.



Gambar 1.1 (a) Hipotesis yang konsisten. (b) Negatif palsu. (c) Hipotesis digeneralisasi. (d) Positif palsu. (e) Hipotesis terspesialisasi.

```

function CURRENT – BEST – LEARNING(examples, h) returns a hypothesis or fail

  if examples is empty then
    return h
  e ← FIRST(examples)
  if e is consistent with h then
    return CURRENT – BEST – LEARNING(REST(examples), h)
  else if e is a false positive for h then
    for each h' in specializations of h consistent with examples seen so far do
      h'' ← CURRENT – BEST – LEARNING(REST(examples), h')
      if h'' ≠ fail then return h''
  else if e is a false negative for h then
    for each h' in generalizations of h consistent with examples seen so far do
      h'' ← CURRENT – BEST – LEARNING(REST(examples), h')
      if h'' = fail then return h''
  return fail
  
```

Gambar 1.2 Algoritma pembelajaran hipotesis terbaik saat ini. Algoritma ini mencari hipotesis konsisten yang sesuai dengan semua contoh dan melakukan backtrack ketika tidak ditemukan spesialisasi/generalisasi yang konsisten. Untuk memulai algoritma, hipotesis apa pun dapat dimasukkan; hipotesis tersebut akan dispesialisasikan atau digeneralisasikan sesuai kebutuhan.

Pada Gambar 1.1(b), sebuah contoh baru (yang dilingkari) merupakan negatif palsu: hipotesis menyatakan seharusnya negatif, tetapi sebenarnya positif. Perluasan hipotesis harus ditingkatkan untuk memasukkannya. Ini disebut generalisasi; satu kemungkinan generalisasi ditunjukkan pada Gambar 1.1(c). Kemudian pada Gambar 1.1(d), kita melihat positif palsu: hipotesis menyatakan contoh baru (yang dilingkari) seharusnya positif, tetapi sebenarnya negatif. Perluasan hipotesis harus dikurangi untuk mengecualikan contoh tersebut. Ini disebut spesialisasi; pada Gambar 1.1(e) kita melihat satu kemungkinan spesialisasi hipotesis.

Hubungan "lebih umum daripada" dan "lebih spesifik daripada" antar hipotesis menyediakan struktur logis pada ruang hipotesis yang memungkinkan pencarian yang efisien.

Kita sekarang dapat menentukan algoritma CURRENT-BEST-LEARNING, yang ditunjukkan pada Gambar 1.2. Perhatikan bahwa setiap kali kita mempertimbangkan untuk menggeneralisasi atau mengkhususkan hipotesis, kita harus memeriksa konsistensinya dengan contoh-contoh lain, karena peningkatan/penurunan yang sembarangan dalam perluasan tersebut dapat mencakup/mengecualikan contoh-contoh negatif/positif yang telah terlihat sebelumnya. Kita telah mendefinisikan generalisasi dan spesialisasi sebagai operasi yang mengubah perluasan suatu hipotesis. Sekarang kita perlu menentukan secara tepat bagaimana keduanya dapat diimplementasikan sebagai operasi sintaksis yang mengubah definisi kandidat yang terkait dengan hipotesis tersebut, sehingga program dapat menjalankannya. Hal ini dilakukan dengan terlebih dahulu mencatat bahwa generalisasi dan spesialisasi juga merupakan hubungan logis antar hipotesis. Jika hipotesis h_1 , dengan definisi C_1 , merupakan generalisasi dari hipotesis h_2 dengan definisi C_2 , maka kita harus memiliki

$$\forall x C_2(x) \Rightarrow C_1(x).$$

Oleh karena itu, untuk membangun generalisasi h_2 , kita hanya perlu menemukan definisi C_1 yang secara logis tersirat oleh C_2 . Ini mudah dilakukan. Misalnya, jika $C_2(x)$ adalah $Alternate(x) \wedge Patrons(x, some)$, maka satu generalisasi yang mungkin diberikan oleh $C_1(x) \equiv Patrons(x, Some)$. Ini disebut kondisi penurunan.

Secara intuitif, ini menghasilkan definisi yang lebih lemah dan oleh karena itu memungkinkan kumpulan contoh positif yang lebih besar. Ada sejumlah operasi generalisasi lainnya, tergantung pada bahasa yang dioperasikan. Demikian pula, kita dapat mengkhususkan hipotesis dengan menambahkan kondisi tambahan ke definisi kandidatnya atau dengan menghilangkan disjungsi dari definisi disjuntif.

- Contoh pertama, X_1 , positif. Atribut $Alternate(X_1)$ benar, jadi misalkan hipotesis awal adalah

$$h_1: \forall x WillWait(x) \Leftrightarrow Alternate(x)$$

- Contoh kedua, X_2 , negatif. h_1 memprediksinya positif, sehingga merupakan positif palsu. Oleh karena itu, kita perlu mengkhususkan h_1 . Hal ini dapat dilakukan dengan menambahkan kondisi tambahan yang akan menyingkirkan X_2 , sambil tetap mengklasifikasikan X_1 sebagai positif. Salah satu kemungkinannya adalah

$$h_2: \forall x WillWait(x) \Leftrightarrow Alternate(x) \Leftrightarrow \wedge Patrons(x, Some)$$

- Contoh ketiga, X_3 , positif. h_2 memprediksinya negatif, sehingga hasilnya negatif palsu. Oleh karena itu, kita perlu menggeneralisasi h_2 . Kita hilangkan kondisi Alternatif, menghasilkan

$$h_3: \forall x WillWait(x) \Leftrightarrow Patrons(x, Some)$$

- Contoh keempat, X_4 , positif. h_3 memprediksinya negatif, sehingga hasilnya negatif palsu. Oleh karena itu, kita perlu menggeneralisasi h_3 . Kita tidak dapat mengabaikan kondisi Patron, karena hal itu akan menghasilkan hipotesis inklusif-semua yang tidak konsisten dengan X_2 . Salah satu kemungkinannya adalah menambahkan disjungsi:

$$h_4 : \forall x \text{ WillWait}(x) \Leftrightarrow \text{Patrons}(x, \text{Some}) \vee (\text{Patrons}(x, \text{Full}) \wedge \text{Fri/Sat}(x))$$

Hipotesisnya sudah mulai terlihat masuk akal. Tentu saja, ada kemungkinan lain yang konsisten dengan empat contoh pertama; berikut dua di antaranya:

$$\begin{aligned} h'_3 : \forall x \text{ WillWait}(x) &\Leftrightarrow \neg \text{WaitEstimate}(x, 30 - 60) \\ h_4 : \forall x \text{ WillWait}(x) &\Leftrightarrow \text{patrons}(x, \text{Some}) \\ &\vee (\text{Patrons}(x, \text{Full}) \wedge \text{WaitEstimate}(x, 10 - 30)) \end{aligned}$$

Algoritma CURRENT – BEST – LEARNING dijelaskan secara nondeterministik, karena pada setiap titik, terdapat beberapa kemungkinan spesialisasi atau generalisasi yang dapat diterapkan. Pilihan yang dibuat belum tentu mengarah pada hipotesis yang paling sederhana, dan dapat mengarah pada situasi yang tidak dapat dipulihkan di mana tidak ada modifikasi sederhana dari hipotesis yang konsisten dengan semua data. Dalam kasus seperti itu, program harus melakukan backtracking ke titik pilihan sebelumnya.

Algoritma CURRENT – BEST – LEARNING dan variannya telah digunakan dalam banyak sistem pembelajaran mesin, dimulai dengan program "arch-learning" Patrick Winston (1970). Namun, dengan banyaknya contoh dan ruang yang besar, beberapa kesulitan muncul:

1. Memeriksa kembali semua contoh sebelumnya untuk setiap modifikasi sangat mahal.
2. Proses pencarian mungkin melibatkan banyak backtracking, ruang hipotesis dapat menjadi tempat yang luasnya berlipat ganda secara eksponensial.

```

function VERSION – SPACE – LEARNING(examples) returns a version space
  local variables:  $V$ , the version space: the set of all hypotheses

   $V \leftarrow$  the set of all hypotheses
  for each example  $e$  in examples do
    if  $V$  is not empty then  $V \leftarrow$  VERSION – SPACE – UPDATE( $V, e$ )
  return  $V$ 

function VERSION – SPACE – UPDATE( $V, e$ ) returns an updated version space
   $V \leftarrow \{h \in V : h \text{ is consistent with } e\}$ 

```

Gambar 1.3 Algoritma pembelajaran ruang versi. Algoritma ini menemukan subset V yang konsisten dengan semua contoh.

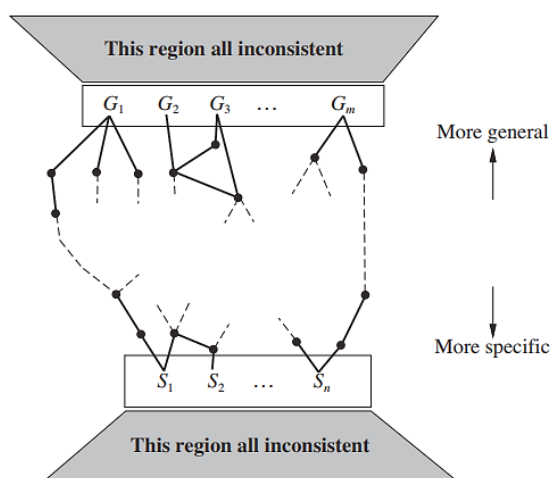
Pencarian dengan komitmen terendah

Penelusuran balik muncul karena pendekatan hipotesis terbaik saat ini harus memilih hipotesis tertentu sebagai tebakan terbaiknya meskipun belum memiliki cukup data untuk memastikan pilihan tersebut. Yang dapat kita lakukan adalah mempertahankan semua dan hanya hipotesis yang konsisten dengan semua data sejauh ini. Setiap contoh baru tidak akan berpengaruh atau akan menghilangkan beberapa hipotesis. Ingat bahwa ruang hipotesis awal dapat dipandang sebagai kalimat disjungtif.

$$h_1 \vee h_2 \vee h_3 \dots \vee h_n$$

Karena berbagai hipotesis ditemukan tidak konsisten dengan contoh-contoh, disjungsi ini menyusut, hanya mempertahankan hipotesis-hipotesis yang tidak dikesampingkan. Dengan asumsi bahwa ruang hipotesis asli memang berisi jawaban yang benar, disjungsi yang direduksi harus tetap berisi jawaban yang benar karena hanya hipotesis yang salah yang telah dihilangkan. Himpunan hipotesis yang tersisa disebut ruang versi, dan algoritma pembelajaran (digambarkan pada Gambar 1.3) disebut algoritma pembelajaran ruang versi (juga algoritma eliminasi kandidat).

Salah satu sifat penting dari pendekatan ini adalah sifatnya yang inkremental: seseorang tidak perlu kembali dan memeriksa ulang contoh-contoh lama. Semua hipotesis yang tersisa dijamin konsisten dengannya. Namun, ada masalah yang jelas. Kita telah mengatakan bahwa ruang hipotesis itu sangat besar, jadi bagaimana mungkin kita bisa menuliskan disjungsi yang sangat besar ini? Analogi sederhana berikut ini sangat membantu. Bagaimana Anda merepresentasikan semua bilangan riil antara 1 dan 2? Lagipula, jumlahnya tak terhingga! Jawabannya adalah dengan menggunakan representasi interval yang hanya menentukan batas himpunan: Cara ini berhasil karena kita memiliki pengurutan pada bilangan riil.



Gambar 1.4 Ruang versi berisi semua hipotesis yang konsisten dengan contoh.

Kita juga memiliki pengurutan pada ruang hipotesis, yaitu generalisasi/spesialisasi. Ini adalah pengurutan parsial, yang berarti bahwa setiap batas bukan merupakan sebuah titik melainkan sekumpulan hipotesis yang disebut himpunan batas. Hebatnya adalah kita dapat

merepresentasikan seluruh ruang versi hanya dengan menggunakan dua himpunan batas: batas paling umum (himpunan- G) dan batas paling spesifik (himpunan- S). Semua yang ada di antaranya dijamin konsisten dengan contoh-contoh. Sebelum kita membuktikannya, mari kita rekap:

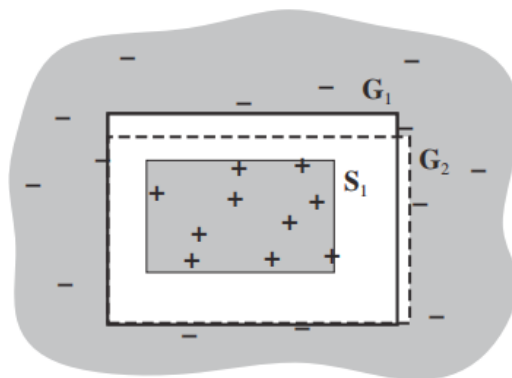
- ❖ Ruang versi saat ini adalah himpunan hipotesis yang konsisten dengan semua contoh sejauh ini. Ruang versi ini direpresentasikan oleh himpunan- S dan himpunan- G , yang masing-masing merupakan himpunan hipotesis.
- ❖ Setiap anggota himpunan- S konsisten dengan semua observasi sejauh ini, dan tidak ada hipotesis konsisten yang lebih spesifik.
- ❖ Setiap anggota G -set konsisten dengan semua pengamatan sejauh ini, dan tidak ada hipotesis konsisten yang lebih umum.

Kita ingin ruang versi awal (sebelum contoh apa pun terlihat) merepresentasikan semua hipotesis yang mungkin. Kita melakukan ini dengan menetapkan himpunan- G berisi Benar (hipotesis yang berisi segalanya), dan himpunan- S berisi Salah (hipotesis yang ekstensinya kosong).

Gambar 1.4 menunjukkan struktur umum representasi himpunan batas dari ruang versi. Untuk menunjukkan bahwa representasi tersebut memadai, kita memerlukan dua sifat berikut:

1. Setiap hipotesis yang konsisten (selain yang ada di himpunan batas) lebih spesifik daripada beberapa anggota himpunan- G , dan lebih umum daripada beberapa anggota himpunan- S . (Artinya, tidak ada "yang tertinggal" di luar.) Ini mengikuti langsung dari definisi S dan G . Jika ada yang tertinggal h , maka itu tidak boleh lebih spesifik daripada anggota G mana pun, yang dalam hal ini ia termasuk dalam G ; atau tidak lebih umum daripada anggota S mana pun, yang dalam hal ini termasuk dalam S .
2. Setiap hipotesis yang lebih spesifik daripada beberapa anggota himpunan G dan lebih umum daripada beberapa anggota himpunan S adalah hipotesis yang konsisten. (Artinya, tidak ada "lubang" di antara batas-batasnya.) Setiap h antara S dan G harus menolak semua contoh negatif yang ditolak oleh setiap anggota G (karena lebih spesifik), dan harus menerima semua contoh positif yang diterima oleh setiap anggota S (karena lebih umum). Dengan demikian, h harus sesuai dengan semua contoh, dan oleh karena itu tidak boleh tidak konsisten. Gambar 1.5 menunjukkan situasinya: tidak ada contoh yang diketahui di luar S tetapi di dalam G , sehingga setiap hipotesis dalam celah tersebut harus konsisten.

Oleh karena itu, kami telah menunjukkan bahwa jika S dan G dipertahankan sesuai definisinya, maka keduanya memberikan representasi ruang versi yang memuaskan. Satu-satunya masalah yang tersisa adalah bagaimana memperbarui S dan G untuk contoh baru (tugas fungsi VERSION – SPACE – UPDATE). Ini mungkin tampak agak rumit pada awalnya, tetapi dari definisi dan dengan bantuan Gambar 1.4, tidak terlalu sulit untuk merekonstruksi algoritmanya.



Gambar 1.5 Perluasan anggota G dan S . Tidak ada contoh yang diketahui terletak di antara dua set batas tersebut.

Kita perlu memperhatikan anggota S_i dan G_i dari himpunan S dan G . Untuk masing-masing anggota, contoh baru tersebut mungkin positif palsu atau negatif palsu.

1. Positif palsu untuk S_i : Ini berarti S_i terlalu umum, tetapi tidak ada spesialisasi S_i yang konsisten (menurut definisi), jadi kita membuangnya dari himpunan S .
2. Negatif palsu untuk S_i : Ini berarti S_i terlalu spesifik, jadi kita menggantinya dengan semua generalisasi langsungnya, asalkan lebih spesifik daripada beberapa anggota G .
3. Positif palsu untuk G_i : Ini berarti G_i terlalu umum, jadi kita menggantinya dengan semua spesialisasi langsungnya, asalkan lebih umum daripada beberapa anggota S .
4. Negatif palsu untuk G_i : Ini berarti G_i terlalu spesifik, tetapi tidak ada generalisasi G_i yang konsisten (menurut definisi) sehingga kami membuangnya dari himpunan G .

Kami melanjutkan operasi ini untuk setiap contoh baru hingga salah satu dari tiga hal terjadi:

1. Kami memiliki tepat satu hipotesis tersisa di ruang versi, yang dalam hal ini kami mengembalikannya sebagai hipotesis unik.
2. Ruang versi runtuh—baik S maupun G menjadi kosong, yang menunjukkan bahwa tidak ada hipotesis yang konsisten untuk himpunan pelatihan. Ini sama halnya dengan kegagalan versi sederhana dari algoritma pohon keputusan.
3. Kami kehabisan contoh dan memiliki beberapa hipotesis tersisa di ruang versi. Ini berarti ruang versi merepresentasikan disjungsi hipotesis. Untuk setiap contoh baru, jika semua disjungsi setuju, maka kami dapat mengembalikan klasifikasi contoh tersebut. Jika tidak setuju, salah satu kemungkinan adalah mengambil suara mayoritas.

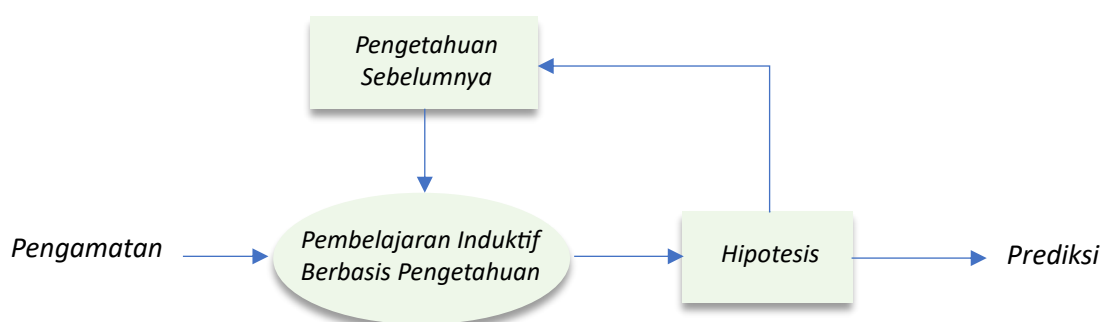
Kami meninggalkan penerapan algoritma VERSION – SPACE – LEARNING pada data restoran sebagai latihan. Terdapat dua kelemahan utama dari pendekatan ruang versi:

- ➡ Jika domain mengandung derau atau atribut yang tidak memadai untuk klasifikasi yang tepat, ruang versi akan selalu kolaps.
- ➡ Jika kita mengizinkan disjungsi tak terbatas dalam ruang hipotesis, himpunan- S akan selalu berisi satu hipotesis paling spesifik, yaitu disjungsi deskripsi contoh-contoh positif yang telah terlihat hingga saat ini. Demikian pula, himpunan- G hanya akan berisi negasi dari disjungsi deskripsi contoh-contoh negatif.

- Untuk beberapa ruang hipotesis, jumlah elemen dalam himpunan- S atau himpunan- G dapat bertambah secara eksponensial dalam jumlah atribut, meskipun terdapat algoritma pembelajaran yang efisien untuk ruang hipotesis tersebut.

Hingga saat ini, belum ditemukan solusi yang sepenuhnya berhasil untuk masalah derau. Masalah disjungsi dapat diatasi dengan hanya mengizinkan bentuk disjungsi yang terbatas atau dengan memasukkan hierarki generalisasi dari predikat yang lebih umum. Misalnya, alih-alih menggunakan disjungsi $WaitEstimate(x, 30 - 60) \vee WaitEstimate(x, > 60)$, kita dapat menggunakan $LongWait(x)$ literal tunggal. Rangkaian operasi generalisasi dan spesialisasi dapat dengan mudah diperluas untuk menangani hal ini.

Algoritma ruang versi murni pertama kali diterapkan dalam sistem Meta-DENDRAL, yang dirancang untuk mempelajari aturan-aturan dalam memprediksi bagaimana molekul akan terpecah menjadi beberapa bagian dalam spektrometer massa. Meta-DENDRAL mampu menghasilkan aturan-aturan yang cukup baru untuk menjamin publikasi dalam jurnal kimia analitik pengetahuan ilmiah nyata pertama yang dihasilkan oleh program komputer. Algoritma ini juga digunakan dalam sistem LEX yang elegan, yang mampu belajar memecahkan masalah integrasi simbolik dengan mempelajari keberhasilan dan kegagalannya sendiri. Meskipun metode ruang versi mungkin tidak praktis dalam sebagian besar masalah pembelajaran di dunia nyata, terutama karena adanya gangguan, metode ini memberikan banyak wawasan tentang struktur logis ruang hipotesis.



Gambar 1.6 Proses pembelajaran kumulatif menggunakan, dan menambah, stok pengetahuan latar belakangnya dari waktu ke waktu.

1.2 PENGETAHUAN DALAM PEMBELAJARAN

Bagian sebelumnya menjelaskan pengaturan paling sederhana untuk pembelajaran induktif. Untuk memahami peran pengetahuan awal, kita perlu membahas hubungan logis antara hipotesis, deskripsi contoh, dan klasifikasi. Misalkan Deskripsi menyatakan konjungsi semua deskripsi contoh dalam set pelatihan, dan Misalkan Klasifikasi menyatakan konjungsi semua klasifikasi contoh. Maka, Hipotesis yang "menjelaskan observasi" harus memenuhi sifat berikut (ingat bahwa $\models$ berarti "secara logis memerlukan"):

$$Hypothesis \wedge Descriptions \models Classification. \quad (1.3)$$

Kami menyebut hubungan semacam ini sebagai kendala entailmen, di mana Hipotesis adalah yang "tidak diketahui". Pembelajaran induktif murni berarti menyelesaikan kendala ini, di mana Hipotesis diambil dari beberapa ruang hipotesis yang telah ditentukan sebelumnya. Misalnya, jika kita menganggap pohon keputusan sebagai rumus logis (lihat Persamaan (1.1)), maka pohon keputusan yang konsisten dengan semua contoh akan memenuhi Persamaan (1.3). Jika kita tidak memberikan batasan pada bentuk logis hipotesis, tentu saja, maka Hipotesis = Klasifikasi juga memenuhi kendala tersebut. Pisau cukur Ockham memberi tahu kita untuk lebih menyukai hipotesis yang kecil dan konsisten, jadi kita mencoba melakukan yang lebih baik daripada sekadar menghafal contoh-contoh.

Gambaran sederhana tanpa pengetahuan tentang pembelajaran induktif ini bertahan hingga awal 1980-an. Pendekatan modern adalah merancang agen yang sudah mengetahui sesuatu dan mencoba mempelajari lebih lanjut. Ini mungkin tidak terdengar seperti wawasan yang sangat mendalam, tetapi ini membuat perbedaan yang cukup besar pada cara kita merancang agen. Hal ini mungkin juga relevan dengan teori kita tentang cara kerja sains itu sendiri. Gagasan umumnya ditunjukkan secara skematis pada Gambar 1.6.

Agan pembelajaran otonom yang menggunakan pengetahuan latar belakang harus memperoleh pengetahuan latar belakang tersebut terlebih dahulu, agar dapat digunakan dalam episode pembelajaran baru. Metode ini sendiri harus merupakan proses pembelajaran. Oleh karena itu, riwayat hidup agen akan dicirikan oleh perkembangan kumulatif, atau inkremental. Agaknya, agen dapat memulai dari nol, melakukan induksi dalam ruang hampa seperti program induksi murni yang sederhana. Namun, setelah ia memakan dari Pohon Pengetahuan, ia tidak dapat lagi mengejar spekulasi naif seperti itu dan harus menggunakan pengetahuan latar belakangnya untuk belajar lebih banyak dan lebih efektif. Pertanyaannya kemudian adalah bagaimana cara melakukannya.

Beberapa contoh sederhana

Mari kita pertimbangkan beberapa contoh pembelajaran yang masuk akal dengan pengetahuan latar belakang. Banyak kasus perilaku inferensial yang tampak rasional dalam menghadapi observasi jelas tidak mengikuti prinsip-prinsip sederhana induksi murni.

- ➡ Terkadang seseorang langsung mengambil kesimpulan umum setelah hanya satu observasi. Gary Larson pernah menggambar kartun di mana seorang manusia gua berkacamata, Zog, sedang memanggang kadalnya di ujung tongkat runcing. Ia disaksikan oleh kerumunan orang sezamannya yang kurang intelektual, yang telah menggunakan tangan kosong untuk memegang makanan mereka di atas api. Pengalaman yang mencerahkan ini cukup untuk meyakinkan para pengamat tentang prinsip umum memasak tanpa rasa sakit.
- ➡ Atau pertimbangkan kasus seorang pelancong ke Brasil yang bertemu dengan orang Brasil pertamanya. Ketika mendengarnya berbicara bahasa Portugis, ia langsung menyimpulkan bahwa orang Brasil berbicara bahasa Portugis, namun ketika mengetahui bahwa nama Zog adalah Fernando, ia tidak menyimpulkan bahwa semua orang Brasil disebut Fernando. Contoh serupa muncul dalam sains. Misalnya, ketika seorang mahasiswa fisika tingkat pertama mengukur densitas dan konduktansi sampel

tembaga pada suhu tertentu, ia cukup yakin untuk menggeneralisasi nilai-nilai tersebut ke semua keping tembaga. Namun, ketika ia mengukur massanya, ia bahkan tidak mempertimbangkan hipotesis bahwa semua keping tembaga memiliki massa tersebut. Di sisi lain, akan cukup masuk akal untuk membuat generalisasi semacam itu untuk semua koin.

- ➡ Terakhir, pertimbangkan kasus seorang mahasiswa kedokteran yang kurang pengetahuan farmakologis tetapi canggi dalam diagnostik yang mengamati sesi konsultasi antara seorang pasien dan seorang dokter spesialis penyakit dalam. Setelah serangkaian tanya jawab, ahli tersebut meminta pasien untuk mengonsumsi antibiotik tertentu. Mahasiswa kedokteran tersebut menyimpulkan aturan umum bahwa antibiotik tertentu efektif untuk jenis infeksi tertentu.

Ini semua adalah kasus di mana penggunaan pengetahuan latar belakang memungkinkan pembelajaran yang jauh lebih cepat daripada yang mungkin diharapkan dari program induksi murni.

Beberapa Skema Umum

Dalam setiap contoh sebelumnya, seseorang dapat mengacu pada pengetahuan sebelumnya untuk mencoba membenarkan generalisasi yang dipilih. Sekarang kita akan melihat jenis kendala entailment apa yang beroperasi dalam setiap kasus. Kendala tersebut akan melibatkan Pengetahuan Latar Belakang, selain Hipotesis dan Deskripsi serta Klasifikasi yang diamati.

Dalam kasus memanggang kadal, manusia gua menggeneralisasi dengan menjelaskan keberhasilan tongkat runcing: tongkat tersebut menopang kadal sambil menjauhkan tangan dari api. Dari penjelasan ini, mereka dapat menyimpulkan aturan umum: bahwa benda apa pun yang panjang, kaku, dan tajam dapat digunakan untuk memanggang makanan kecil bertubuh lunak. Proses generalisasi semacam ini disebut pembelajaran berbasis penjelasan, atau EBL. Perhatikan bahwa aturan umum tersebut mengikuti secara logis dari pengetahuan latar belakang yang dimiliki oleh manusia gua. Oleh karena itu, kendala entailment yang dipenuhi oleh EBL adalah sebagai berikut:

$$\begin{aligned} &Hypothesis \wedge Descriptions \mid = Classifications \\ &Background \models Hypothesis . \end{aligned}$$

Karena EBL menggunakan Persamaan (1.3), awalnya dianggap sebagai cara untuk belajar dari contoh. Namun karena mensyaratkan pengetahuan latar belakang yang cukup untuk menjelaskan Hipotesis, yang pada gilirannya menjelaskan pengamatan, agen tidak benar-benar mempelajari sesuatu yang baru secara faktual dari contoh tersebut. Agen bisa saja memperoleh contoh dari apa yang sudah diketahuinya, meskipun itu mungkin memerlukan jumlah komputasi yang tidak masuk akal. EBL sekarang dipandang sebagai metode untuk mengubah teori prinsip pertama menjadi pengetahuan yang berguna dan bertujuan khusus.

Situasi pelancong kami di Brasil sangat berbeda, karena dia tidak dapat serta merta menjelaskan mengapa Fernando berbicara seperti itu, kecuali dia mengetahui bulla

kepausannya. Selain itu, generalisasi yang sama akan muncul dari seorang pelancong yang sama sekali tidak mengetahui sejarah kolonial. Pengetahuan awal yang relevan dalam kasus ini adalah bahwa, di negara mana pun, kebanyakan orang cenderung berbicara dalam bahasa yang sama; Di sisi lain, Fernando tidak diasumsikan sebagai nama semua orang Brasil karena keteraturan semacam ini tidak berlaku untuk nama.

Demikian pula, mahasiswa fisika tingkat pertama juga akan kesulitan menjelaskan nilai-nilai spesifik yang ia temukan untuk konduktansi dan densitas tembaga. Namun, ia tahu bahwa material penyusun suatu objek dan suhunya bersama-sama menentukan konduktansinya. Dalam setiap kasus, pengetahuan awal Latar Belakang berkaitan dengan relevansi serangkaian fitur dengan predikat tujuan. Pengetahuan ini, bersama dengan pengamatan, memungkinkan agen untuk menyimpulkan aturan umum baru yang menjelaskan pengamatan:

$$\begin{aligned} &Hypothesis \wedge Descriptions \mid = Classifications, \\ &Background \wedge Descriptions \wedge Classifications \mid = Hypothesis. \end{aligned} \quad (1.4)$$

Kami menyebut generalisasi semacam ini sebagai pembelajaran berbasis relevansi, atau RBL (meskipun namanya tidak standar). Perhatikan bahwa meskipun RBL memanfaatkan isi observasi, RBL tidak menghasilkan hipotesis yang melampaui isi logis dari pengetahuan latar belakang dan observasi. RBL merupakan bentuk pembelajaran deduktif dan tidak dapat dengan sendirinya menjelaskan penciptaan pengetahuan baru yang dimulai dari awal.

Dalam kasus mahasiswa kedokteran yang mengamati pakar, kami berasumsi bahwa pengetahuan awal mahasiswa cukup untuk menyimpulkan penyakit pasien D dari gejalanya. Namun, hal ini tidak cukup untuk menjelaskan fakta bahwa dokter meresepkan obat tertentu M . Mahasiswa perlu mengusulkan aturan lain, yaitu bahwa M umumnya efektif melawan D . Dengan aturan ini dan pengetahuan awal mahasiswa, mahasiswa sekarang dapat menjelaskan mengapa pakar meresepkan M dalam kasus khusus ini. Kita dapat menggeneralisasi contoh ini untuk menghasilkan batasan entailmen.

$$Background \wedge Hypothesis \wedge Descriptions \models Classifications. \quad (1.5)$$

Artinya, pengetahuan latar belakang dan hipotesis baru digabungkan untuk menjelaskan contoh-contoh tersebut. Sebagaimana pembelajaran induktif murni, algoritma pembelajaran harus mengajukan hipotesis sesederhana mungkin, konsisten dengan batasan ini. Algoritma yang memenuhi batasan (1.5) disebut algoritma pembelajaran induktif berbasis pengetahuan, atau KBIL.

Algoritma KBIL, yang dijelaskan secara rinci di Bagian 19.5, telah dipelajari terutama di bidang pemrograman logika induktif, atau ILP. Dalam sistem ILP, pengetahuan awal memainkan dua peran kunci dalam mengurangi kompleksitas pembelajaran:

1. Karena setiap hipotesis yang dihasilkan harus konsisten dengan pengetahuan sebelumnya serta dengan observasi baru, ukuran ruang hipotesis efektif diperkecil untuk hanya mencakup teori-teori yang konsisten dengan apa yang sudah diketahui.

2. Untuk setiap rangkaian observasi tertentu, ukuran hipotesis yang diperlukan untuk membangun penjelasan atas observasi tersebut dapat dikurangi secara signifikan, karena pengetahuan sebelumnya akan tersedia untuk membantu aturan-aturan baru dalam menjelaskan observasi tersebut. Semakin kecil hipotesisnya, semakin mudah ditemukan.

Selain memungkinkan penggunaan pengetahuan sebelumnya dalam induksi, sistem ILP dapat merumuskan hipotesis dalam logika orde pertama umum. Ini berarti bahwa sistem tersebut dapat belajar di lingkungan yang tidak dapat dipahami oleh sistem yang lebih sederhana.

1.3 PEMBELAJARAN BERBASIS PENJELASAN

Pembelajaran berbasis penjelasan adalah metode untuk mengekstraksi aturan umum dari observasi individual. Sebagai contoh, perhatikan masalah diferensiasi dan penyederhanaan ekspresi aljabar. Jika kita mendiferensiasikan ekspresi seperti X^2 terhadap X , kita memperoleh $2X$. (Kita menggunakan huruf kapital untuk variabel aritmatika X yang tidak diketahui, untuk membedakannya dari variabel logika x .) Dalam sistem penalaran logis, tujuannya dapat dinyatakan sebagai $ASK(Derivative(X^2, X) = d, KB)$, dengan solusi $d = 2X$.

Siapa pun yang memahami kalkulus diferensial dapat melihat solusi ini "melalui inspeksi" sebagai hasil dari latihan dalam memecahkan masalah tersebut. Seorang mahasiswa yang baru pertama kali menghadapi masalah tersebut, atau program tanpa pengalaman, akan menghadapi pekerjaan yang jauh lebih sulit. Penerapan aturan diferensiasi standar pada akhirnya menghasilkan ekspresi $1 \times (2 \times (X(2 - 1)))$, dan akhirnya disederhanakan menjadi $2X$. Dalam implementasi pemrograman logika penulis, hal ini membutuhkan 136 langkah pembuktian, yang 99 di antaranya berada pada cabang buntu dalam pembuktian. Setelah pengalaman tersebut, kami ingin program tersebut menyelesaikan masalah yang sama jauh lebih cepat saat masalah tersebut muncul lagi.

Teknik memoisasi telah lama digunakan dalam ilmu komputer untuk mempercepat program dengan menyimpan hasil komputasi. Ide dasar fungsi memo adalah mengumpulkan basis data pasangan masukan-keluaran; ketika fungsi dipanggil, pertama-tama ia memeriksa basis data tersebut untuk melihat apakah ia dapat menghindari penyelesaian masalah dari awal. Pembelajaran berbasis penjelasan membawa hal ini lebih jauh, dengan menciptakan aturan umum yang mencakup seluruh kelas kasus.

Dalam kasus diferensiasi, memoisasi akan mengingat bahwa turunan X^2 terhadap X adalah $2X$, tetapi akan membiarkan agen menghitung turunan Z^2 terhadap Z dari awal. Kami ingin dapat mengekstrak aturan umum bahwa untuk setiap variabel aritmatika u yang tidak diketahui, turunan u^2 terhadap u adalah $2u$. (Aturan yang lebih umum untuk un juga dapat dibuat, tetapi contoh saat ini cukup untuk menjelaskan maksudnya.) Dalam istilah logis, hal ini diungkapkan oleh aturan

$$\text{Arithmetic Unknown } (u) \Rightarrow \text{Derivative } (u^2, u) = 2u$$

Jika basis pengetahuan memuat aturan semacam itu, maka setiap kasus baru yang merupakan contoh dari aturan ini dapat segera dipecahkan. Ini, tentu saja, hanyalah contoh sepele dari fenomena yang sangat umum. Setelah sesuatu dipahami, hal itu dapat digeneralisasi dan digunakan kembali dalam keadaan lain. Ini menjadi langkah yang "jelas" dan kemudian dapat digunakan sebagai blok bangunan dalam memecahkan masalah yang lebih kompleks. Alfred North Whitehead (1911), rekan penulis *Principia Mathematica* bersama Bertrand Russell, menulis, "Peradaban maju dengan memperluas jumlah operasi penting yang dapat kita lakukan tanpa memikirkannya," mungkin dirinya sendiri menerapkan EBL pada pemahamannya tentang peristiwa seperti penemuan Zog.

Jika Anda telah memahami ide dasar dari contoh diferensiasi, maka otak Anda sudah sibuk mencoba mengekstrak prinsip-prinsip umum pembelajaran berbasis penjelasan darinya. Perhatikan bahwa Anda belum menemukan EBL sebelum Anda melihat contohnya. Seperti manusia gua yang mengamati Zog, Anda (dan kita) membutuhkan contoh sebelum kita dapat menghasilkan prinsip-prinsip dasar. Hal ini karena menjelaskan mengapa sesuatu merupakan ide yang bagus jauh lebih mudah daripada memunculkan ide tersebut sejak awal.

Mengekstrak Aturan Umum dari Contoh

Ide dasar di balik EBL adalah pertama-tama membangun penjelasan observasi menggunakan pengetahuan sebelumnya, dan kemudian menetapkan definisi kelas kasus yang struktur penjelasannya dapat digunakan. Definisi ini memberikan dasar bagi aturan yang mencakup semua kasus dalam kelas tersebut. "Penjelasan" tersebut dapat berupa pembuktian logis, tetapi secara lebih umum, dapat berupa proses penalaran atau pemecahan masalah apa pun yang langkah-langkahnya terdefinisi dengan baik. Kuncinya adalah mampu mengidentifikasi kondisi yang diperlukan agar langkah-langkah yang sama tersebut dapat diterapkan pada kasus lain.

Kita akan menggunakan pembuktian teorema rantai mundur sederhana untuk sistem penalaran kita. Pohon pembuktian untuk $Derivative(X^2, X) = 2X$ terlalu besar untuk digunakan sebagai contoh, jadi kita akan menggunakan masalah yang lebih sederhana untuk mengilustrasikan metode generalisasi. Misalkan masalah kita adalah menyederhanakan $1 \times (0 + X)$. Basis pengetahuan mencakup aturan-aturan berikut:

$Rewrite(u, v) \wedge Simplify(v, w) \Rightarrow Simplify(u, w).$
 $Primitive(u) \Rightarrow Simplify(u, u).$
 $ArithmeticUnknown(u) \Rightarrow Primitive(u).$
 $Number(u) \Rightarrow Primitive(u).$
 $Rewrite(1 \times u, u).$
 $Rewrite(0 + u, u).$
 $\vdots$

Bukti bahwa jawabannya adalah X ditunjukkan pada bagian atas Gambar 1.7. Metode EBL sebenarnya membangun dua pohon pembuktian secara bersamaan. Pohon pembuktian kedua menggunakan tujuan yang divariasikan di mana konstanta dari tujuan awal digantikan oleh

variabel. Saat pembuktian awal berlangsung, pembuktian yang divariasikan berjalan secara bertahap, menggunakan penerapan aturan yang persis sama.

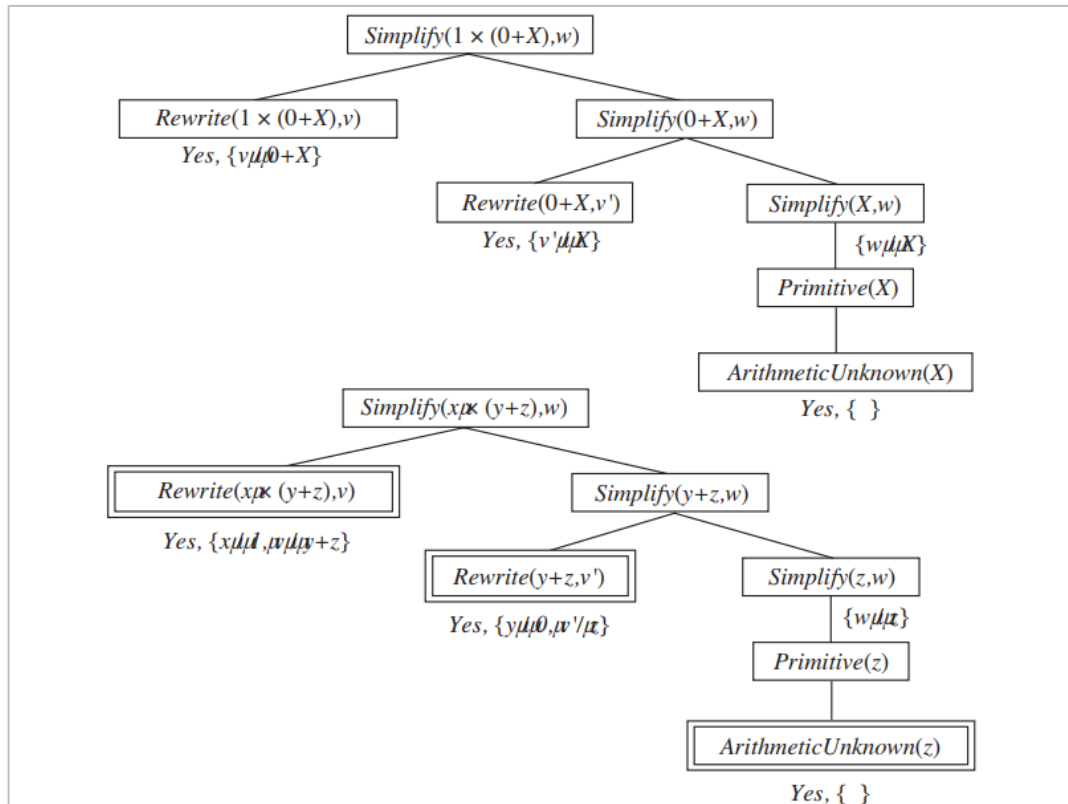
Hal ini dapat menyebabkan beberapa variabel menjadi terinstansiasi. Misalnya, untuk menggunakan aturan $Rewrite(1 \times u, u)$, variabel x dalam subtujuan $Rewrite(x \times (y + z), v)$ harus terikat ke 1. Demikian pula, y harus terikat ke 0 dalam subtujuan $Rewrite(y + z, v')$ agar dapat menggunakan aturan $Rewrite(0 + u, u)$. Setelah kita memiliki pohon pembuktian umum, kita mengambil daun-daunnya (dengan pengikatan yang diperlukan) dan membentuk aturan umum untuk predikat tujuan:

$$Rewrite(1 \times (0 + z), 0 + z) \wedge Rewrite(0 + z, z) \wedge ArithmeticUnknown(z) \\ \Rightarrow Simplify(1 \times (0 + z), z).$$

Pohon kedua menunjukkan pembuktian untuk contoh masalah dengan semua konstanta digantikan oleh variabel, yang darinya kita dapat memperoleh berbagai aturan lainnya. Perhatikan bahwa dua kondisi pertama di sisi kiri adalah benar terlepas dari nilai z . Oleh karena itu, kita dapat menghilangkannya dari aturan, menghasilkan

$$arithmeticUnknown(z) \Rightarrow Simplify(1 \times (0 + z), z)$$

Secara umum, kondisi dapat dihilangkan dari aturan akhir jika tidak memberlakukan batasan apa pun pada variabel di sisi kanan aturan, karena aturan yang dihasilkan akan tetap benar dan akan lebih efisien. Perhatikan bahwa kita tidak dapat menghilangkan kondisi $ArithmeticUnknown(z)$, karena tidak semua nilai z yang mungkin merupakan variabel aritmatika yang tidak diketahui. Nilai selain variabel aritmatika yang tidak diketahui mungkin memerlukan bentuk penyederhanaan yang berbeda: misalnya, jika z adalah 2×3 , maka penyederhanaan yang benar dari $1 \times (0 + (2 \times 3))$ adalah 6 dan bukan 2×3 .



Gambar 1.7 Pohon pembuktian untuk masalah penyederhanaan. Pohon pertama menunjukkan pembuktian untuk contoh masalah awal, yang darinya kita dapat memperoleh

Singkatnya, proses EBL dasar bekerja sebagai berikut:

1. Dengan sebuah contoh, buatlah bukti bahwa predikat tujuan berlaku untuk contoh tersebut menggunakan pengetahuan latar belakang yang tersedia.
2. Secara paralel, buatlah pohon bukti umum untuk tujuan yang divariasikan menggunakan langkah-langkah inferensi yang sama seperti pada bukti awal.
3. Buat aturan baru yang ruas kirinya terdiri dari daun-daun pohon pembuktian dan ruas kanannya adalah tujuan yang divariasikan (setelah menerapkan pengikatan yang diperlukan dari pembuktian umum).
4. Abaikan semua kondisi dari ruas kiri yang bernilai benar terlepas dari nilai variabel dalam tujuan.

Meningkatkan Efisiensi

Pohon pembuktian umum pada Gambar 1.7 sebenarnya menghasilkan lebih dari satu aturan umum. Misalnya, jika kita menghentikan, atau memangkas, pertumbuhan cabang kanan pada pohon pembuktian ketika mencapai langkah Primitif, kita mendapatkan aturan

$$Primitive(z) \Rightarrow Simplify(1 \times (0 + z), z)$$

Aturan ini sama validnya, tetapi lebih umum, daripada aturan yang menggunakan *ArithmeticUnknown*, karena mencakup kasus-kasus di mana z adalah sebuah angka. Kita dapat mengekstrak aturan yang lebih umum lagi dengan memangkas setelah langkah *Simplify(y + z, w)*, menghasilkan aturan

$$\text{Simplify}(y + z, w) \Rightarrow \text{Simplify}(1 \times (y + z), w)$$

Secara umum, sebuah aturan dapat diekstraksi dari setiap sub-pohon parsial dari pohon pembuktian umum. Sekarang kita memiliki masalah: aturan mana yang kita pilih?

Pilihan aturan mana yang akan dihasilkan bermuara pada pertanyaan efisiensi. Ada tiga faktor yang terlibat dalam analisis peningkatan efisiensi dari EBL:

1. Menambahkan sejumlah besar aturan dapat memperlambat proses penalaran, karena mekanisme inferensi tetap harus memeriksa aturan-aturan tersebut bahkan dalam kasus-kasus yang tidak menghasilkan solusi. Dengan kata lain, hal ini meningkatkan faktor percabangan dalam ruang pencarian.
2. Untuk mengkompensasi perlambatan dalam penalaran, aturan turunan harus menawarkan peningkatan kecepatan yang signifikan untuk kasus-kasus yang dicakupnya. Peningkatan ini terjadi terutama karena aturan turunan menghindari jalan buntu yang seharusnya diambil, tetapi juga karena aturan tersebut mempersingkat pembuktian itu sendiri.
3. Aturan turunan harus seumum mungkin, sehingga berlaku untuk kumpulan kasus terbesar yang memungkinkan.

Pendekatan umum untuk memastikan bahwa aturan turunan efisien adalah dengan menekankan operasionalitas setiap sub-tujuan dalam aturan tersebut. Subtujuan bersifat operasional jika "mudah" untuk dipecahkan. Misalnya, subtujuan *Primitif*(z) mudah dipecahkan, membutuhkan paling banyak dua langkah, sedangkan subtujuan *Sederhanakan* $\text{Simplify}(y + z, w)$ dapat menghasilkan jumlah inferensi yang sembarangan, bergantung pada nilai y dan z . Jika uji operasionalitas dilakukan pada setiap langkah dalam konstruksi pembuktian umum, maka kita dapat memangkas sisa cabang segera setelah subtujuan operasional ditemukan, dengan hanya mempertahankan subtujuan operasional sebagai konjungsi aturan baru.

Sayangnya, biasanya terdapat trade-off antara operasionalitas dan generalitas. Subtujuan yang lebih spesifik umumnya lebih mudah dipecahkan tetapi mencakup lebih sedikit kasus. Selain itu, operasionalitas adalah masalah derajat: satu atau dua langkah sudah pasti operasional, tetapi bagaimana dengan 10 atau 100? Akhirnya, biaya penyelesaian subtujuan tertentu bergantung pada aturan lain yang tersedia dalam basis pengetahuan. Biaya ini dapat naik atau turun seiring bertambahnya aturan. Dengan demikian, sistem EBL menghadapi masalah optimasi yang sangat kompleks dalam upaya memaksimalkan efisiensi basis pengetahuan awal yang diberikan. Terkadang dimungkinkan untuk memperoleh model matematika tentang pengaruh penambahan aturan tertentu terhadap efisiensi keseluruhan dan menggunakan model ini untuk memilih aturan terbaik yang akan ditambahkan.

Namun, analisisnya bisa menjadi sangat rumit, terutama ketika melibatkan aturan rekursif. Salah satu pendekatan yang menjanjikan adalah mengatasi masalah efisiensi secara empiris, cukup dengan menambahkan beberapa aturan dan melihat aturan mana yang bermanfaat dan benar-benar mempercepat proses. Analisis empiris efisiensi sebenarnya merupakan inti dari EBL. Apa yang selama ini kita sebut secara umum sebagai "efisiensi basis

pengetahuan tertentu" sebenarnya adalah kompleksitas kasus rata-rata pada distribusi masalah.

1.4 BELAJAR MENGGUNAKAN INFORMASI RELEVANSI

Wisatawan kami di Brasil tampaknya mampu membuat generalisasi yang meyakinkan mengenai bahasa yang digunakan oleh orang Brasil lainnya. Kesimpulan ini didukung oleh latar belakang pengetahuannya, yaitu bahwa orang-orang di suatu negara (biasanya) berbicara dalam bahasa yang sama. Kita dapat menyatakan hal ini dalam logika tingkat pertama sebagai berikut:

$$Nationality(x, n) \wedge Nationality(y, n) \wedge Language(x, l) \Rightarrow Language(y, l) \quad (1.6)$$

(Terjemahan harfiah: "Jika x dan y memiliki kewarganegaraan yang sama n dan x berbicara bahasa l , maka y juga berbicara bahasa itu.") Tidak sulit untuk menunjukkan bahwa, dari kalimat ini dan pengamatan bahwa

$$Nationality(Fernando, Brazil) \wedge Language(Fernando, Portuguese),$$

kesimpulan berikut ini:

$$Nationality(x, Brazil) \Rightarrow Language(x, Portuguese),$$

Kalimat-kalimat seperti (19.6) mengekspresikan bentuk relevansi yang ketat: berdasarkan kewarganegaraan, bahasa sepenuhnya ditentukan. (Dengan kata lain: bahasa adalah fungsi dari kewarganegaraan.) Kalimat-kalimat ini disebut dependensi atau determinasi fungsional. Kalimat-kalimat ini sangat umum muncul dalam jenis aplikasi tertentu (misalnya, mendefinisikan desain basis data) sehingga sintaksis khusus digunakan untuk menuliskannya. Kami mengadopsi notasi Davies (1985):

$$Nationality(x, n) \succ Language(x, l)$$

Seperti biasa, ini hanyalah pembubuhan gula sintaksis, tetapi memperjelas bahwa penentuannya sebenarnya merupakan hubungan antarpredikat: kebangsaan menentukan bahasa. Sifat-sifat relevan yang menentukan konduktansi dan densitas dapat diungkapkan dengan cara yang sama:

$$\begin{aligned} Material(x, m) \wedge Temperature(x, t) &\succ Conductance(x, \rho) \\ Material(x, m) \wedge Temperature(x, t) &\succ Density(x, d) \end{aligned}$$

Generalisasi yang sesuai mengikuti secara logis dari penentuan dan pengamatan.

Menentukan ruang hipotesis

Meskipun penentuan tersebut menyetujui kesimpulan umum mengenai semua orang Brasil, atau semua potongan tembaga pada suhu tertentu, penentuan tersebut tentu saja tidak dapat menghasilkan teori prediktif umum untuk semua kebangsaan, atau untuk semua suhu dan material, dari satu contoh. Efek utamanya dapat dilihat sebagai pembatasan ruang hipotesis yang perlu dipertimbangkan oleh agen pembelajaran.

Dalam memprediksi konduktansi, misalnya, seseorang hanya perlu mempertimbangkan material dan suhu dan dapat mengabaikan massa, kepemilikan, hari dalam seminggu, presiden saat ini, dan sebagainya. Hipotesis tentu saja dapat mencakup istilah-istilah yang pada gilirannya ditentukan oleh material dan suhu, seperti struktur molekul, energi termal, atau kerapatan elektron bebas.

Penentuan menentukan kosakata dasar yang memadai untuk membangun hipotesis mengenai predikat target. Pernyataan ini dapat dibuktikan dengan menunjukkan bahwa suatu penentuan yang diberikan secara logis ekuivalen dengan pernyataan bahwa definisi yang tepat dari predikat target adalah salah satu dari himpunan semua definisi yang dapat diekspresikan menggunakan predikat di ruas kiri penentuan tersebut.

Secara intuitif, jelas bahwa pengurangan ukuran ruang hipotesis akan mempermudah pembelajaran predikat target. Dengan menggunakan hasil dasar teori pembelajaran komputasional, kita dapat mengkuantifikasi kemungkinan keuntungan. Pertama, ingatlah bahwa untuk fungsi Boolean, contoh $\log(|\mathcal{H}|)$ diperlukan untuk konvergen ke hipotesis yang wajar, dengan $|\mathcal{H}|$ adalah ukuran ruang hipotesis. Jika pembelajar memiliki n fitur Boolean untuk membangun hipotesis, maka, tanpa adanya pembatasan lebih lanjut, $|\mathcal{H}| = O(2^{2^n})$, sehingga jumlah contoh adalah $O(2^n)$. Jika penentuan tersebut mengandung d predikat di ruas kiri, pembelajar hanya akan membutuhkan $O(2^d)$ contoh, yaitu pengurangan $O(2^{n-d})$.

Mempelajari dan Menggunakan Informasi Relevansi

Sebagaimana telah kami nyatakan dalam pendahuluan bab ini, pengetahuan awal memang berguna dalam pembelajaran; tetapi pengetahuan tersebut juga harus dipelajari. Untuk memberikan gambaran lengkap tentang pembelajaran berbasis relevansi, kami harus menyediakan algoritma pembelajaran untuk penentuan. Algoritma pembelajaran yang kami sajikan ini didasarkan pada upaya langsung untuk menemukan penentuan paling sederhana yang konsisten dengan observasi. Penentuan $P > Q$ menyatakan bahwa jika ada contoh yang cocok pada P , maka contoh-contoh tersebut juga harus cocok pada Q . Oleh karena itu, suatu penentuan konsisten dengan sekumpulan contoh jika setiap pasangan yang cocok pada predikat di ruas kiri juga cocok pada predikat tujuan. Sebagai contoh, misalkan kita memiliki contoh pengukuran konduktansi pada sampel material berikut:

```

function MINIMAL – CONSISTENT – DET( $E, A$ ) returns a set of attributes
  inputs:  $E$ , a set of examples
            $A$ , a set of attributes, of size  $n$ 

  for  $i = 0$  to  $n$  do
    for each subset  $A_i$  of  $A$  of size  $i$  do
      if CONSISTENT – DET? ( $A_i, E$ ) then return  $A_i$ 

function CONSISTENT – DET? ( $A, E$ ) returns a truth value
  inputs:  $A$ , a set of attributes
            $E$ , a set of examples
  local variables:  $H$ , a hash table

  for each example  $e$  in  $E$  do
    if some example in  $H$  has the same values as  $e$  for the attributes  $A$ 
      but a different classification then return false
    store the class of  $e$  in  $H$ , indexed by the values for attributes  $A$  of the example  $e$ 
  return true

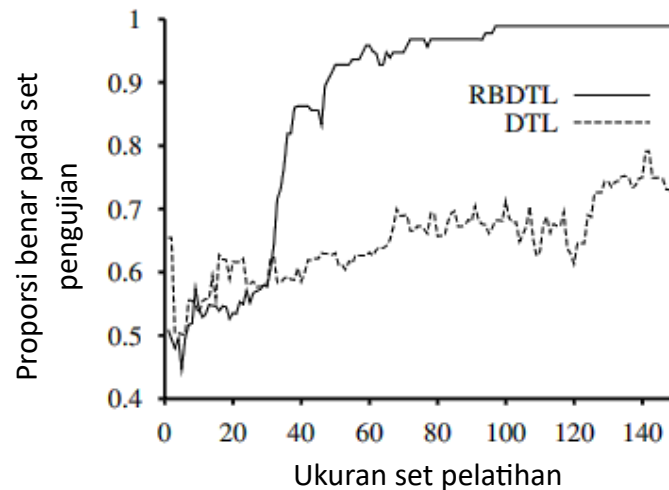
```

Gambar 1.8 Algoritma untuk menemukan penentuan minimal yang konsisten.

Sampel	Massa	Suhu (°C)	Material	Ukuran	Konduktivitas
S1	12	26	Tembaga	3	0.59
S1	12	100	Tembaga	3	0.57
S2	24	26	Tembaga	6	0.59
S3	12	26	Timbal	2	0.05
S3	12	100	Timbal	2	0.04
S4	24	26	Timbal	4	0.05

Penentuan minimal yang konsisten adalah $Material \wedge Suhu > Konduktansi$. Terdapat penentuan yang tidak minimal tetapi konsisten, yaitu, $Massa \wedge Ukuran \wedge Suhu > Konduktansi$. Hal ini konsisten dengan contoh-contoh karena massa dan ukuran menentukan densitas dan, dalam set data kami, kami tidak memiliki dua material berbeda dengan densitas yang sama. Seperti biasa, kami memerlukan set sampel yang lebih besar untuk mengeliminasi hipotesis yang hampir benar.

Ada beberapa kemungkinan algoritma untuk menemukan penentuan minimal yang konsisten. Pendekatan yang paling jelas adalah melakukan pencarian melalui ruang penentuan, memeriksa semua penentuan dengan satu predikat, dua predikat, dan seterusnya, hingga penentuan yang konsisten ditemukan. Penentuan d akan direpresentasikan oleh set atribut di sisi kiri, karena predikat target diasumsikan tetap. Algoritma dasar diuraikan dalam Gambar 1.8.



Gambar 1.9 Perbandingan kinerja antara DECISION-TREE-LEARNING dan RBDTL pada data yang dihasilkan secara acak untuk fungsi target yang hanya bergantung pada 5 dari 16 atribut.

Kompleksitas waktu dari algoritma ini bergantung pada ukuran penentuan konsisten terkecil. Misalkan penentuan ini memiliki p atribut dari total n atribut. Maka algoritma tidak akan menemukannya hingga mencari subset A berukuran p . Terdapat $\binom{n}{p} = O(n^p)$ subset seperti itu; oleh karena itu, algoritma bersifat eksponensial dalam ukuran penentuan minimal. Ternyata masalahnya adalah NP-lengkap, jadi kita tidak dapat berharap untuk melakukan yang lebih baik dalam kasus umum. Namun, di sebagian besar domain, akan ada struktur lokal yang cukup sehingga p akan kecil.

Dengan adanya algoritma untuk mempelajari penentuan, agen pembelajaran memiliki cara untuk membangun hipotesis minimal untuk mempelajari predikat target. Misalnya, kita dapat menggabungkan MINIMAL – CONSISTENT – DET dengan algoritma DECISION – TREE – LEARNING. Ini menghasilkan algoritma pembelajaran pohon keputusan berbasis relevansi (RBDTL) yang pertama-tama mengidentifikasi himpunan minimal atribut yang relevan dan kemudian meneruskan himpunan ini ke algoritma pohon keputusan untuk pembelajaran.

Berbeda dengan PEMBELAJARAN POHON KEPUTUSAN, RBDTL secara bersamaan mempelajari dan menggunakan informasi relevansi untuk meminimalkan ruang hipotesisnya. Kami memperkirakan RBDTL akan belajar lebih cepat daripada PEMBELAJARAN POHON KEPUTUSAN, dan memang demikianlah kenyataannya. Gambar 1.9 menunjukkan kinerja pembelajaran kedua algoritma pada data yang dihasilkan secara acak untuk suatu fungsi yang hanya bergantung pada 5 dari 16 atribut. Jelas, dalam kasus di mana semua atribut yang tersedia relevan, RBDTL tidak akan menunjukkan keunggulan.

Bagian ini baru menyentuh permukaan bidang bias deklaratif, yang bertujuan untuk memahami bagaimana pengetahuan awal dapat digunakan untuk mengidentifikasi ruang hipotesis yang tepat untuk mencari definisi target yang tepat. Masih banyak pertanyaan yang belum terjawab:

- Bagaimana algoritma ini dapat diperluas untuk menangani derau?
- Dapatkah kita menangani variabel bernilai kontinu?
- Bagaimana jenis pengetahuan awal lainnya dapat digunakan, selain penentuan?
- Bagaimana algoritme dapat digeneralisasi untuk mencakup teori orde pertama apa pun, alih-alih hanya representasi berbasis atribut?

Beberapa pertanyaan ini akan dibahas di bagian selanjutnya.

1.5 PEMROGRAMAN LOGIKA INDUKTIF

Pemrograman logika induktif (ILP) menggabungkan metode induktif dengan kekuatan representasi orde pertama, dengan fokus khusus pada representasi hipotesis sebagai program logika. Pemrograman ini mendapatkan popularitas karena tiga alasan. Pertama, ILP menawarkan pendekatan yang ketat terhadap masalah pembelajaran induktif berbasis pengetahuan umum. Kedua, ILP menawarkan algoritma lengkap untuk menginduksi teori orde pertama yang umum dari contoh, yang oleh karena itu dapat berhasil dipelajari dalam domain di mana algoritma berbasis atribut sulit diterapkan.

Contohnya adalah dalam mempelajari bagaimana struktur protein terlipat (Gambar 1.10). Konfigurasi tiga dimensi molekul protein tidak dapat direpresentasikan secara wajar oleh serangkaian atribut, karena konfigurasi tersebut secara inheren mengacu pada hubungan antar objek, bukan atribut dari satu objek. Logika orde pertama adalah bahasa yang tepat untuk menggambarkan hubungan tersebut.

Ketiga, pemrograman logika induktif menghasilkan hipotesis yang (relatif) mudah dipahami manusia. Sebagai contoh, terjemahan bahasa Inggris pada Gambar 1.10 dapat diteliti dan dikritik oleh para ahli biologi yang bekerja. Ini berarti bahwa sistem pemrograman logika induktif dapat berpartisipasi dalam siklus ilmiah eksperimen, pembuatan hipotesis, debat, dan sanggahan. Partisipasi semacam itu tidak akan mungkin terjadi pada sistem yang menghasilkan pengklasifikasi "kotak hitam", seperti jaringan saraf tiruan.

Sebuah Contoh

Ingat kembali dari Persamaan (1.5) bahwa masalah induksi berbasis pengetahuan umum adalah untuk "menyelesaikan" kendala konsekuensi.

$$\text{Background} \wedge \text{Hypothesis} \wedge \text{Description} \mid = \text{Classifications}$$

untuk Hipotesis yang tidak diketahui, mengingat pengetahuan Latar Belakang dan contoh-contoh yang dijelaskan oleh Deskripsi dan Klasifikasi. Untuk mengilustrasikan hal ini, kita akan menggunakan masalah mempelajari hubungan keluarga dari contoh-contoh. Deskripsi tersebut akan terdiri dari silsilah keluarga yang diperluas, yang dijelaskan dalam konteks hubungan Ibu, Ayah, dan Menikah serta properti Laki-laki dan Perempuan. Sebagai contoh, kita akan menggunakan silsilah keluarga yang ditunjukkan di sini pada Gambar 1.11. Deskripsi terkait adalah sebagai berikut:

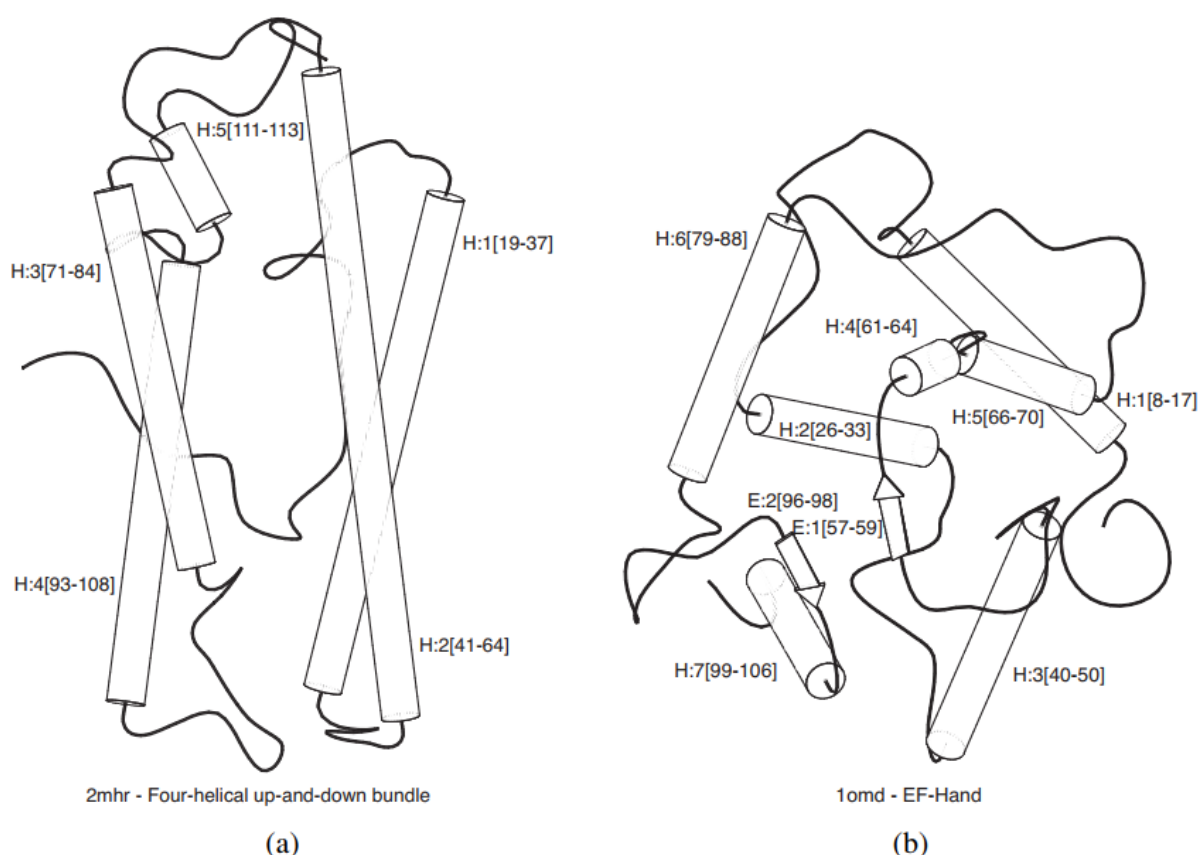
Father (Philip, Charles) Father (Philip, Anne) ...

<i>Mother (Mum, Margaret)</i>	<i>Mother (Mum, Elizabeth)</i>	...
<i>Married(Diana, Charles)</i>	<i>Married(Elizabeth, Philip)</i>	...
<i>Male(Philip)</i>	<i>Male(Charles)</i>	...
<i>Female(Beatrice)</i>	<i>Female(Margaret)</i>	...

Kalimat-kalimat dalam Klasifikasi (*Classifications*) bergantung pada konsep target yang sedang dipelajari. Kita mungkin ingin mempelajari Kakek/Nenek (*Grandparent*), Ipar/Kakak Ipar (*BrotherInLaw*), atau Leluhur/Nenek Moyang (*Ancestor*), misalnya. Untuk Kakek/Nenek, rangkaian lengkap Klasifikasi (*Classifications*) berisi $20 \times 20 = 400$ konjungsi dalam bentuk

Grandparent(Mum, Charles)Grandparent(Elizabeth, Beatrice) ...
 $\neg$ *Grandparent(Mum, Harry) \neg Grandparent(Spencer , Peter) ...*

Kita tentu saja bisa belajar dari sebagian dari himpunan lengkap ini.



Gambar 1.10 (a) dan (b) masing-masing menunjukkan contoh positif dan negatif dari konsep "bundel empat heliks naik-turun" dalam domain pelipatan protein. Setiap contoh struktur dikodekan ke dalam ekspresi logika sekitar 100 konjungsi seperti *TotalLength(D2mhr, 118) $\wedge$ NumberHelices(D2mhr, 6) $\wedge$...* Dari deskripsi ini dan dari klasifikasi seperti *Fold(FOUR – HELICAL – UP – AND – DOWN – BUNDLE, D2mhr)* sistem ILP PROGOL mempelajari aturan berikut:

$$\begin{aligned} &Fold(FOUR - HELICAL - UP - AND - DOWN - BUNDLE, p) \Leftarrow \\ &Helix(p, h_1) \wedge Length(h_1, \mathbf{HIGH}) \wedge Position(p, h_1, n) \\ &\wedge (1 \leq n \leq 3) \wedge Adjacent(p, h_1, h_2) \wedge Helix(p, h_2). \end{aligned}$$

Aturan semacam ini tidak dapat dipelajari, atau bahkan direpresentasikan, oleh mekanisme berbasis atribut seperti yang telah kita bahas di bab-bab sebelumnya. Aturan tersebut dapat diterjemahkan ke dalam bahasa Inggris sebagai "Protein p memiliki kelas lipatan "bundel naik-turun empat-heliks" jika mengandung heliks panjang h_1 pada posisi struktur sekunder antara 1 dan 3 dan h_1 berada di sebelah heliks kedua."

Tujuan program pembelajaran induktif adalah menghasilkan serangkaian kalimat untuk Hipotesis sedemikian rupa sehingga kendala konsekuensinya terpenuhi. Misalkan, untuk saat ini, agen tidak memiliki pengetahuan latar belakang: Latar belakang kosong. Maka salah satu solusi yang mungkin untuk Hipotesis adalah sebagai berikut:

$$\begin{aligned} Grandparent(x, y) \Leftrightarrow & [\exists z \text{ Mother}(x, z) \wedge \text{Mother}(z, y)] \\ & \vee [\exists z \text{ Mother}(x, z) \wedge \text{Father}(z, y)] \\ & \vee [\exists z \text{ Father}(x, z) \wedge \text{Mother}(z, y)] \\ & \vee [\exists z \text{ Father}(x, z) \wedge \text{Father}(z, y)]. \end{aligned}$$

Perhatikan bahwa algoritma pembelajaran berbasis atribut, seperti DECISION – TREE – LEARNING, tidak akan menyelesaikan masalah ini. Untuk menyatakan Kakek-Nenek sebagai atribut (yaitu, predikat unary), kita perlu menjadikan pasangan orang sebagai objek:

$$Grandparent(\langle Mum, Charles \rangle) \dots$$

Lalu kita terjebak dalam upaya merepresentasikan deskripsi contoh. Satu-satunya atribut yang mungkin adalah hal-hal mengerikan seperti

$$FirstElementIsMotherOfElizabeth(\langle Mum, Charles \rangle).$$

Definisi Kakek/Nenek berdasarkan atribut-atribut ini hanya menjadi disjungsi besar kasus-kasus spesifik yang tidak dapat digeneralisasi ke contoh-contoh baru sama sekali. Algoritma pembelajaran berbasis atribut tidak mampu mempelajari predikat relasional. Dengan demikian, salah satu keuntungan utama algoritma ILP adalah penerapannya pada rentang masalah yang jauh lebih luas, termasuk masalah relasional.

Pembaca tentu telah menyadari bahwa sedikit pengetahuan latar belakang akan membantu dalam representasi definisi Kakek/Nenek. Misalnya, jika Latar Belakang mencakup kalimat

$$Parent(x, y) \Leftrightarrow [Mother(x, y) \vee Father(x, y)],$$

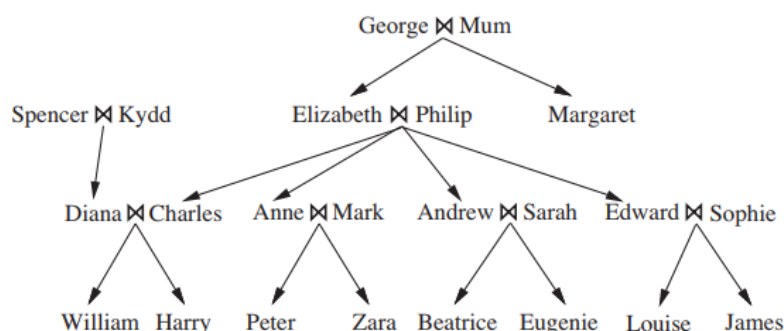
maka definisi Kakek Nenek akan disederhanakan menjadi

$$Grandparent(x, y) \Leftrightarrow [\exists z \text{ Parent}(x, z) \wedge \text{Parent}(z, y)] .$$

Hal ini menunjukkan bagaimana pengetahuan latar belakang dapat secara drastis mengurangi ukuran hipotesis yang diperlukan untuk menjelaskan observasi.

Algoritma ILP juga memungkinkan untuk membuat predikat baru guna memfasilitasi ekspresi hipotesis penjelasan. Mengingat contoh data yang ditunjukkan sebelumnya, sangatlah wajar bagi program ILP untuk mengusulkan predikat tambahan, yang akan kita sebut "Induk", untuk menyederhanakan definisi predikat target. Algoritma yang dapat menghasilkan predikat baru disebut algoritma induksi konstruktif. Jelas, induksi konstruktif merupakan bagian penting dari gambaran pembelajaran kumulatif. Ini telah menjadi salah satu masalah tersulit dalam pembelajaran mesin, tetapi beberapa teknik ILP menyediakan mekanisme yang efektif untuk mencapainya.

Di sisa bab ini, kita akan mempelajari dua pendekatan utama untuk ILP. Yang pertama menggunakan generalisasi metode pohon keputusan, dan yang kedua menggunakan teknik berdasarkan pembalikan bukti resolusi.



Gambar 1.11 Pohon keluarga yang umum.

Metode Pembelajaran Induktif Top-Down

Pendekatan pertama ILP bekerja dengan memulai dengan aturan yang sangat umum dan secara bertahap mengkhususkannya agar sesuai dengan data. Hal ini pada dasarnya terjadi dalam pembelajaran pohon keputusan, di mana pohon keputusan dikembangkan secara bertahap hingga konsisten dengan observasi. Untuk melakukan ILP, kami menggunakan literal orde pertama, bukan atribut, dan hipotesisnya adalah sekumpulan klausa, bukan pohon keputusan. Bagian ini menjelaskan FOIL, salah satu program ILP pertama.

Misalkan kita mencoba mempelajari definisi predikat *Grandfather* (x, y), menggunakan data keluarga yang sama seperti sebelumnya. Seperti halnya pembelajaran pohon keputusan, kita dapat membagi contoh menjadi contoh positif dan negatif. Contoh positif adalah

$\langle \text{George}, \text{Anne} \rangle, \langle \text{Philip}, \text{Peter} \rangle, \langle \text{Spencer}, \text{Harry} \rangle, \dots$

dan contoh negatifnya adalah

$\langle George, Elizabeth \rangle, \langle Harry, Zara \rangle, \langle Charles, Philip \rangle, \dots$

Perhatikan bahwa setiap contoh adalah sepasang objek, karena Kakek adalah predikat biner. Secara keseluruhan, terdapat 12 contoh positif dalam silsilah keluarga dan 388 contoh negatif (semua pasangan orang lainnya).

FOIL membangun serangkaian klausa, masing-masing dengan $Grandfather(x, y)$ sebagai kepala. Klausa-klausa tersebut harus mengklasifikasikan 12 contoh positif sebagai contoh dari hubungan $Grandfather(x, y)$ sambil mengesampingkan 388 contoh negatif. Klausa-klausa tersebut adalah klausa Horn, dengan perluasan bahwa literal yang dinegasikan diperbolehkan dalam badan klausa dan ditafsirkan menggunakan negasi sebagai kegagalan, seperti dalam Prolog. Klausa awal memiliki badan kosong:

$\Rightarrow Grandfather(x, y) .$

Klausa ini mengklasifikasikan setiap contoh sebagai positif, sehingga perlu dispesialisasikan. Kita melakukannya dengan menambahkan literal satu per satu di ruas kiri. Berikut tiga kemungkinan penambahan:

$$\begin{aligned} Father(x, y) &\Rightarrow Grandfather(x, y) . \\ Parent(x, z) &\Rightarrow Grandfather(x, y) . \\ Father(x, z) &\Rightarrow Grandfather(x, y) . \end{aligned}$$

(Perhatikan bahwa kita berasumsi bahwa klausa yang mendefinisikan Orang Tua sudah menjadi bagian dari pengetahuan latar belakang.) Klausa pertama dari ketiga klausa ini secara keliru mengklasifikasikan ke-12 contoh positif sebagai negatif dan karenanya dapat diabaikan. Klausa kedua dan ketiga sesuai dengan semua contoh positif, tetapi klausa kedua keliru pada sebagian besar contoh negatif dua kali lipatnya, karena memperbolehkan ibu maupun ayah. Oleh karena itu, kami lebih memilih klausa ketiga.

Sekarang kita perlu mengkhususkan klausa ini lebih lanjut, untuk mengesampingkan kasus-kasus di mana x adalah ayah dari beberapa z , tetapi z bukan orangtua dari y . Menambahkan satu literal $Parent(z, y)$ menghasilkan

$$Father(x, z) \wedge Parent(z, y) \Rightarrow Grandfather(x, y) ,$$

yang mengklasifikasikan semua contoh dengan benar. FOIL akan menemukan dan memilih literal ini, sehingga menyelesaikan tugas pembelajaran. Secara umum, solusinya adalah serangkaian klausa Horn, yang masing-masing menyiratkan predikat target. Misalnya, jika kita tidak memiliki predikat Induk dalam kosakata kita, maka solusinya mungkin

$$\begin{aligned} \text{Father}(x,z) \wedge \text{Father}(z,y) &\Rightarrow \text{Grandfather}(x,y) \\ \text{Father}(x,z) \wedge \text{Mother}(z,y) &\Rightarrow \text{Grandfather}(x,y) \end{aligned}$$

Perhatikan bahwa setiap klausa ini mencakup beberapa contoh positif, bahwa secara keseluruhan mencakup semua contoh positif, dan bahwa NEW – CLAUSE dirancang sedemikian rupa sehingga tidak ada klausa yang secara keliru mencakup contoh negatif. Secara umum, FOIL harus menelusuri banyak klausa yang tidak berhasil sebelum menemukan solusi yang tepat.

Contoh ini merupakan ilustrasi yang sangat sederhana tentang cara kerja FOIL. Sketsa algoritma lengkapnya ditunjukkan pada Gambar 1.12. Pada dasarnya, algoritma ini berulang kali membangun sebuah klausa, literal demi literal, hingga sesuai dengan beberapa subset contoh positif dan tidak ada satu pun contoh negatif. Kemudian, contoh positif yang dicakup oleh klausa tersebut dihapus dari set pelatihan, dan proses berlanjut hingga tidak ada contoh positif yang tersisa. Dua subrutin utama yang akan dijelaskan adalah NEW – LITERALS, yang membangun semua literal baru yang mungkin untuk ditambahkan ke klausa, dan CHOOSE – LITERAL, yang memilih literal untuk ditambahkan.

NEW – LITERALS mengambil sebuah klausa dan membangun semua literal "berguna" yang mungkin yang dapat ditambahkan ke klausa tersebut. Mari kita gunakan klausa sebagai contoh

$$\text{Father}(x,z) \Rightarrow \text{Grandfather}(x,y).$$

Ada tiga jenis literal yang dapat ditambahkan:

1. Literal yang menggunakan predikat: literal dapat dinegasikan atau tidak dinegasikan, predikat apa pun yang ada (termasuk predikat tujuan) dapat digunakan, dan semua argumen harus berupa variabel. Variabel apa pun dapat digunakan untuk argumen predikat apa pun, dengan satu batasan: setiap literal harus menyertakan setidaknya satu variabel dari literal sebelumnya atau dari kepala klausa. Literal seperti *Mother(z,u)*, *Married(z,z)*, $\neg \text{Male}(y)$, dan *Grandfather(v,x)* diperbolehkan, sedangkan *Menikah Married(u,v)* tidak diperbolehkan. Perhatikan bahwa penggunaan predikat dari kepala klausa memungkinkan FOIL mempelajari definisi rekursif.
2. Literal persamaan dan ketidaksetaraan: ini menghubungkan variabel yang sudah muncul dalam klausa. Misalnya, kita dapat menambahkan $z \neq x$. Literal ini juga dapat menyertakan konstanta yang ditentukan pengguna. Untuk mempelajari aritmatika, kita dapat menggunakan 0 dan 1, dan untuk mempelajari fungsi daftar, kita dapat menggunakan daftar kosong [].
3. Perbandingan aritmatika: ketika menangani fungsi variabel kontinu, literal seperti $x > y$ dan $y \leq z$ dapat ditambahkan. Seperti dalam pembelajaran pohon keputusan, nilai ambang batas konstan dapat dipilih untuk memaksimalkan daya diskriminatif pengujian.

```

function FOIL(examples, target) returns a set of Horn clauses
  inputs: examples, set of examples
           target, a literal for the goal predicate
  local variables: clauses, set of clauses, initially empty

  while examples contains positive examples do
    clause  $\leftarrow$  NEW – CLAUSE(examples, target)
    remove positive examples covered by clause from examples
    add clause to clauses
  return clauses

function NEW – CLAUSE(examples, target) returns a Horn clause
  local variables: clause, a clause with target as head and an empty body
                   l, a literal to be added to the clause
                   extended_examples, a set of examples with values for new variables

                   extended_examples  $\leftarrow$  examples
  while extended_examples contains negative examples do
    l  $\leftarrow$  CHOOSE – LITERAL(NEW – LITERALS(clause), extended_examples)
    append l to the body of clause
    extended_examples  $\leftarrow$  set of examples created by applying EXTEND – EXAMPLE
                           to each example in extended_examples
  return clause

function EXTEND – EXAMPLE(example, literal) returns a set of examples
  if example satisfies literal
    then return the set of examples created by extending example with
                each possible constant value for each new variable in literal
  else return the empty set

```

Gambar 1.12 Sketsa algoritma FOIL untuk mempelajari himpunan klausa Horn orde pertama dari contoh. NEW – LITERALS dan CHOOSE – LITERAL dijelaskan dalam teks.

Faktor percabangan yang dihasilkan dalam ruang pencarian ini sangat besar tetapi FOIL juga dapat menggunakan informasi tipe untuk mengurangnya. Misalnya, jika domain mencakup angka dan orang, pembatasan tipe akan mencegah NEW – LITERALS menghasilkan literal seperti *Parent(x, n)*, di mana *x* adalah orang dan *n* adalah angka.

CHOOSE – LITERAL menggunakan heuristik yang agak mirip dengan perolehan informasi untuk memutuskan literal mana yang akan ditambahkan. Detail pastinya tidak penting di sini, dan sejumlah variasi berbeda telah dicoba. Salah satu fitur tambahan FOIL yang menarik adalah penggunaan pisau cukur Ockham untuk mengeliminasi beberapa hipotesis. Jika sebuah klausa menjadi lebih panjang (menurut metrik tertentu) daripada panjang total contoh positif yang dijelaskan oleh klausa tersebut, klausa tersebut tidak dianggap sebagai hipotesis potensial. Teknik ini menyediakan cara untuk menghindari klausa yang terlalu kompleks yang dapat mengakomodasi noise dalam data.

FOIL dan program sejenisnya telah digunakan untuk mempelajari berbagai macam definisi. Salah satu demonstrasi yang paling mengesankan melibatkan penyelesaian

serangkaian latihan panjang tentang fungsi pemrosesan-daftar dari buku teks Prolog karya Bratko (1986). Dalam setiap kasus, program mampu mempelajari definisi fungsi yang tepat dari sekumpulan kecil contoh, dengan menggunakan fungsi yang telah dipelajari sebelumnya sebagai pengetahuan dasar.

Pembelajaran induktif dengan deduksi invers

Pendekatan utama kedua untuk ILP melibatkan pembalikan proses pembuktian deduktif normal. Resolusi invers didasarkan pada pengamatan bahwa jika Klasifikasi contoh mengikuti dari $Background \wedge Hypothesis \wedge Descriptions$, maka seseorang harus dapat membuktikan fakta ini melalui resolusi (karena resolusi bersifat tuntas). Jika kita dapat "menjalankan pembuktian mundur", maka kita dapat menemukan Hipotesis sedemikian rupa sehingga pembuktian tersebut berhasil.

Kuncinya adalah menemukan cara untuk membalikkan proses resolusi. Kami akan menunjukkan proses pembuktian mundur untuk resolusi invers yang terdiri dari langkah-langkah mundur individual. Ingatlah bahwa langkah resolusi biasa mengambil dua klausa C_1 dan C_2 dan menyelesaikannya untuk menghasilkan resolven C . Langkah resolusi invers mengambil resolven C dan menghasilkan dua klausa C_1 dan C_2 , sehingga C adalah hasil dari penyelesaian C_1 dan C_2 . Sebagai alternatif, langkah ini dapat mengambil resolven C dan klausa C_1 dan menghasilkan klausa C_2 sedemikian rupa sehingga C adalah hasil dari penyelesaian C_1 dan C_2 .

Langkah-langkah awal dalam proses resolusi invers ditunjukkan pada Gambar 1.13, di mana kami berfokus pada contoh positif $Grandparent(George, Anne)$. Proses ini dimulai di akhir pembuktian (ditunjukkan di bagian bawah gambar). Kami menganggap resolven C sebagai klausa kosong (yaitu kontradiksi) dan C_2 sebagai $\neg Grandparent(George, Anne)$, yang merupakan negasi dari contoh tujuan. Langkah invers pertama mengambil C dan C_2 dan menghasilkan klausa $Grandparent(George, Anne)$ untuk C_1 . Langkah selanjutnya mengambil klausa ini sebagai C dan klausa $Parent(Elizabeth, Anne)$ sebagai C_2 , dan menghasilkan klausa

$$\neg Parent(Elizabeth, y) \vee Grandparent(George, y)$$

sebagai C_1 . Langkah terakhir memperlakukan klausa ini sebagai resolven. Dengan $Parent(George, Elizabeth)$ induk (George, Elizabeth) sebagai C_2 , salah satu klausa C_1 yang mungkin adalah hipotesis

$$Parent(x, z) \wedge Parent(z, y) \Rightarrow Grandparent(x, y).$$

Sekarang kita memiliki bukti resolusi yang menyatakan bahwa hipotesis, deskripsi, dan pengetahuan latar belakang memerlukan klasifikasi $Grandparent(George, Anne)$.

Jelas, resolusi invers melibatkan pencarian. Setiap langkah resolusi invers bersifat nondeterministik, karena untuk setiap C , terdapat banyak atau bahkan tak terhingga jumlah klausa C_1 dan C_2 yang beresolusi ke C . Misalnya, alih-alih memilih $\neg Parent(Elizabeth, y) \vee$

$Grandparent(George, y)$ untuk C_1 pada langkah terakhir Gambar 1.13, langkah resolusi invers dapat memilih salah satu kalimat berikut:

- $\neg Parent(Elizabeth, Anne) \vee Grandparent(George, Anne) .$
- $\neg Parent(z, Anne) \vee Grandparent(George, Anne) .$
- $\neg Parent(z, y) \vee Grandparent(George, y) .$
- $\vdots$

Lebih lanjut, klausa yang berpartisipasi dalam setiap langkah dapat dipilih dari Pengetahuan Latar Belakang, dari Deskripsi contoh, dari Klasifikasi yang dinegasikan, atau dari klausa yang dihipotesiskan yang telah dihasilkan dalam pohon resolusi terbalik. Banyaknya kemungkinan berarti faktor percabangan yang besar (dan oleh karena itu pencarian yang tidak efisien) tanpa kontrol tambahan. Sejumlah pendekatan untuk menjinakkan pencarian telah dicoba dalam sistem ILP yang diimplementasikan:

1. Pilihan yang redundan dapat dihilangkan—misalnya, dengan hanya menghasilkan hipotesis yang paling spesifik dan dengan mengharuskan semua klausa yang dihipotesiskan konsisten satu sama lain, dan dengan pengamatan. Kriteria terakhir ini akan mengesampingkan klausa $\neg Parent(x, y) \vee Grandparent(George, y)$, yang tercantum sebelumnya.
2. Strategi pembuktian dapat dibatasi. Misalnya, resolusi linear adalah strategi yang lengkap dan terbatas. Resolusi linier menghasilkan pohon pembuktian yang memiliki struktur percabangan linier—seluruh pohon mengikuti satu baris, dengan hanya klausa tunggal yang bercabang dari baris tersebut (seperti pada Gambar 1.13).
3. Bahasa representasi dapat dibatasi, misalnya dengan menghilangkan simbol fungsi atau hanya mengizinkan klausa Horn. Misalnya, PROGOL beroperasi dengan klausa Horn menggunakan entailmen invers. Idenya adalah mengubah batasan entailmen.

$$Background \wedge Hypothesis \wedge Descriptions \models Classifications$$

ke bentuk yang setara secara logis

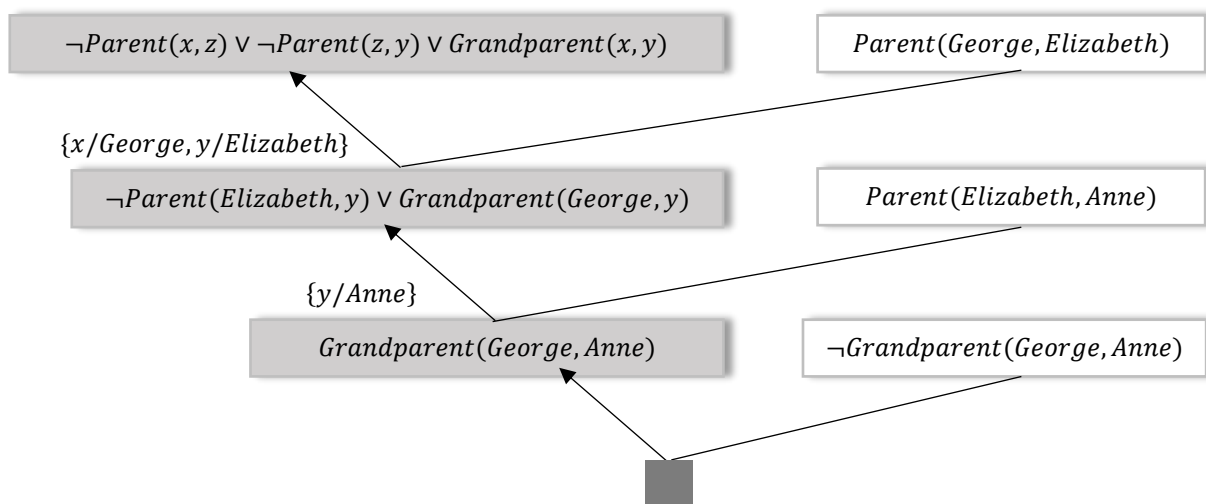
$$Background \wedge Descriptions \wedge \neg Classifications \models \neg Hypothesis.$$

Dari sini, seseorang dapat menggunakan proses yang mirip dengan deduksi klausa Horn Prolog normal, dengan negasi-sebagai-kegagalan untuk menurunkan *Hipotesis*. Karena terbatas pada klausa Horn, metode ini tidak lengkap, tetapi dapat lebih efisien daripada resolusi penuh. Inferensi lengkap dengan entailmen invers juga dapat diterapkan.

4. Inferensi dapat dilakukan dengan pengecekan model, alih-alih pembuktian teorema. Sistem PROGOL menggunakan suatu bentuk pengecekan model untuk membatasi

pencarian. Artinya, seperti pemrograman himpunan jawaban, sistem ini menghasilkan nilai-nilai yang memungkinkan untuk variabel logis, dan memeriksa konsistensi.

5. Inferensi dapat dilakukan dengan klausa proposisional dasar, alih-alih logika orde pertama. Sistem LINUS bekerja dengan menerjemahkan teori-teori orde pertama ke dalam logika proposisional, menyelesaikannya dengan sistem pembelajaran proposisional, lalu menerjemahkannya kembali. Bekerja dengan rumus proposisional dapat lebih efisien pada beberapa masalah.



Gambar 1.13 Langkah awal dalam proses resolusi invers. Klausa yang diarsir dihasilkan oleh langkah resolusi invers dari klausa di sebelah kanan dan klausa di bawahnya. Klausa yang tidak diarsir berasal dari Deskripsi dan Klasifikasi (termasuk Klasifikasi yang dinegasikan).

Membuat Penemuan dengan Pemrograman Logika Induktif

Prosedur resolusi invers yang membalikkan strategi resolusi lengkap, pada prinsipnya, merupakan algoritma lengkap untuk mempelajari teori orde pertama. Artinya, jika suatu Hipotesis yang tidak diketahui menghasilkan serangkaian contoh, maka prosedur resolusi invers dapat menghasilkan Hipotesis dari contoh-contoh tersebut. Pengamatan ini menunjukkan kemungkinan yang menarik: Misalkan contoh-contoh yang tersedia mencakup berbagai lintasan benda jatuh.

Apakah program resolusi invers secara teoritis mampu menyimpulkan hukum gravitasi? Jawabannya jelas ya, karena hukum gravitasi memungkinkan seseorang untuk menjelaskan contoh-contoh tersebut, dengan latar belakang matematika yang sesuai. Demikian pula, seseorang dapat membayangkan bahwa elektromagnetisme, mekanika kuantum, dan teori relativitas juga berada dalam cakupan program ILP. Tentu saja, mereka juga berada dalam cakupan monyet dengan mesin tik; kita masih membutuhkan heuristik yang lebih baik dan cara-cara baru untuk menyusun ruang pencarian.

Satu hal yang akan dilakukan sistem resolusi invers untuk Anda adalah menciptakan predikat baru. Kemampuan ini sering dianggap agak ajaib, karena komputer sering dianggap "hanya bekerja dengan apa yang diberikan kepada mereka." Faktanya, predikat baru jatuh langsung dari langkah resolusi invers. Kasus paling sederhana muncul dalam menghipotesiskan

dua klausa baru C_1 dan C_2 , dengan klausa C sebagai contoh. Resolusi C_1 dan C_2 menghilangkan literal yang dimiliki kedua klausa tersebut; oleh karena itu, sangat mungkin literal yang dihilangkan tersebut mengandung predikat yang tidak muncul dalam C . Jadi, ketika bekerja mundur, salah satu kemungkinan adalah menghasilkan predikat baru untuk merekonstruksi literal yang hilang.

Gambar 1.14 menunjukkan contoh di mana predikat baru P dihasilkan dalam proses mempelajari definisi untuk Ancestor. Setelah dihasilkan, P dapat digunakan dalam langkah resolusi invers selanjutnya. Misalnya, langkah selanjutnya mungkin menghipotesiskan bahwa $Mother(x, y) \Rightarrow P(x, y)$. Dengan demikian, predikat baru P memiliki makna yang dibatasi oleh pembangkitan hipotesis yang melibatkannya. Contoh lain mungkin mengarah pada kendala $Father(x, y) \Rightarrow P(x, y)$. Dengan kata lain, predikat P adalah apa yang biasa kita anggap sebagai hubungan Induk. Seperti yang telah disebutkan sebelumnya, penemuan predikat baru dapat secara signifikan mengurangi ukuran definisi predikat tujuan. Oleh karena itu, dengan memasukkan kemampuan untuk menciptakan predikat baru, sistem resolusi invers seringkali dapat memecahkan masalah pembelajaran yang tidak dapat diselesaikan dengan teknik lain.

Beberapa revolusi terdalam dalam sains berasal dari penemuan predikat dan fungsi baru misalnya, penemuan percepatan oleh Galileo atau penemuan energi termal oleh Joule. Setelah istilah-istilah ini tersedia, penemuan hukum-hukum baru menjadi (relatif) mudah. Bagian yang sulit terletak pada menyadari bahwa beberapa entitas baru, dengan hubungan spesifik dengan entitas-entitas yang sudah ada, akan memungkinkan seluruh rangkaian pengamatan dijelaskan dengan teori yang jauh lebih sederhana dan lebih elegan daripada yang ada sebelumnya.

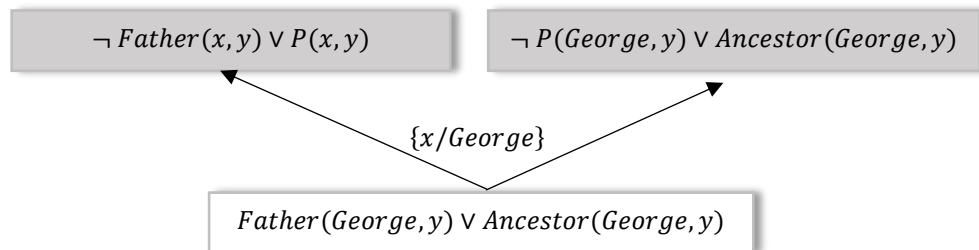
Hingga saat ini, sistem ILP belum membuat penemuan pada tingkat Galileo atau Joule, tetapi penemuan mereka telah dianggap dapat dipublikasikan dalam literatur ilmiah. Misalnya, dalam Journal of Molecular Biology, menggambarkan penemuan otomatis aturan untuk pelipatan protein oleh program ILP PROGOL. Banyak aturan yang ditemukan oleh PROGOL dapat diturunkan dari prinsip-prinsip yang diketahui, tetapi sebagian besar belum pernah dipublikasikan sebelumnya sebagai bagian dari basis data biologi standar. (Lihat Gambar 1.10 untuk contoh.).

Dalam karya terkait, Srinivasan et al. (1994) membahas masalah penemuan aturan berbasis struktur molekul untuk mutagenisitas senyawa nitroaromatik. Senyawa ini ditemukan dalam asap knalpot mobil. Untuk 80% senyawa dalam basis data standar, dimungkinkan untuk mengidentifikasi empat fitur penting, dan regresi linier pada fitur-fitur ini mengungguli ILP. Untuk 20% sisanya, fitur-fiturnya sendiri tidak prediktif, dan ILP mengidentifikasi hubungan yang memungkinkannya mengungguli regresi linier, jaringan saraf, dan pohon keputusan.

Yang paling mengesankan, King dkk. (2009) menganugerahi robot dengan kemampuan untuk melakukan eksperimen biologi molekuler dan memperluas teknik ILP untuk mencakup desain eksperimen, sehingga menciptakan ilmuwan otonom yang benar-benar menemukan pengetahuan baru tentang genomika fungsional ragi. Untuk semua contoh ini, tampaknya kemampuan untuk merepresentasikan hubungan dan menggunakan pengetahuan latar

belakang berkontribusi pada kinerja tinggi ILP. Fakta bahwa aturan yang ditemukan oleh ILP dapat ditafsirkan oleh manusia berkontribusi pada penerimaan teknik-teknik ini dalam jurnal biologi, bukan hanya jurnal ilmu komputer.

ILP telah memberikan kontribusi untuk ilmu-ilmu lain selain biologi. Salah satu yang paling penting adalah pemrosesan bahasa alami, di mana ILP telah digunakan untuk mengekstrak informasi relasional yang kompleks dari teks.



Gambar 1.14 Langkah resolusi invers yang menghasilkan predikat baru P.

1.6 RINGKASAN

Bab ini telah menyelidiki berbagai cara bagaimana pengetahuan awal dapat membantu agen belajar dari pengalaman baru. Karena banyak pengetahuan awal diekspresikan dalam bentuk model relasional, alih-alih model berbasis atribut, kami juga membahas sistem yang memungkinkan pembelajaran model relasional. Poin-poin pentingnya adalah:

- Penggunaan pengetahuan awal dalam pembelajaran mengarah pada gambaran pembelajaran kumulatif, di mana agen pembelajaran meningkatkan kemampuan belajar mereka seiring dengan perolehan pengetahuan yang lebih banyak.
- Pengetahuan awal membantu pembelajaran dengan menghilangkan hipotesis yang sebelumnya konsisten dan dengan "mengisi" penjelasan contoh, sehingga memungkinkan hipotesis yang lebih pendek. Kontribusi ini seringkali menghasilkan pembelajaran yang lebih cepat dari contoh yang lebih sedikit.
- Memahami berbagai peran logis yang dimainkan oleh pengetahuan awal, sebagaimana diungkapkan oleh batasan konsekuensi, membantu mendefinisikan berbagai teknik pembelajaran.
- Pembelajaran berbasis penjelasan (EBL) mengekstrak aturan umum dari contoh tunggal dengan menjelaskan contoh-contoh tersebut dan menggeneralisasi penjelasannya. Ini menyediakan metode deduktif untuk mengubah pengetahuan prinsip dasar menjadi keahlian yang bermanfaat, efisien, dan bertujuan khusus.
- Pembelajaran berbasis relevansi (RBL) menggunakan pengetahuan sebelumnya dalam bentuk penentuan untuk mengidentifikasi atribut yang relevan, sehingga menghasilkan ruang hipotesis yang lebih kecil dan mempercepat pembelajaran. RBL juga memungkinkan generalisasi deduktif dari contoh-contoh tunggal.
- Pembelajaran induktif berbasis pengetahuan (KBIL) menemukan hipotesis induktif yang menjelaskan serangkaian observasi dengan bantuan pengetahuan latar belakang.

- Teknik pemrograman logika induktif (ILP) melakukan KBIL pada pengetahuan yang diekspresikan dalam logika orde pertama. Metode ILP dapat mempelajari pengetahuan relasional yang tidak dapat diekspresikan dalam sistem berbasis atribut.
- ILP dapat dilakukan dengan pendekatan top-down untuk menyempurnakan aturan yang sangat umum atau melalui pendekatan bottom-up untuk membalikkan proses deduktif.
- Metode ILP secara alami menghasilkan predikat baru yang dengannya teori-teori baru yang ringkas dapat diekspresikan dan menunjukkan potensi sebagai sistem pembentukan teori ilmiah untuk tujuan umum.

BAB 2

MODEL PROBABILISTIK PEMBELAJARAN

Kita akan melihat bahwa pandangan Bayesian tentang pembelajaran sangatlah ampuh, memberikan solusi umum untuk masalah noise, overfitting, dan prediksi optimal. Pandangan ini juga memperhitungkan fakta bahwa agen yang kurang mahatahu tidak akan pernah bisa yakin tentang teori dunia mana yang benar, namun tetap harus membuat keputusan dengan menggunakan beberapa teori dunia. Kami menjelaskan metode untuk mempelajari model probabilitas—terutama jaringan Bayesian—di Bagian 2.2 dan 2.3. Beberapa materi dalam bab ini cukup matematis, meskipun pelajaran umumnya dapat dipahami tanpa membahas detailnya.

2.1 PEMBELAJARAN STATISTIK

Konsep kunci dalam bab ini, adalah data dan hipotesis. Di sini, data adalah bukti yaitu, perwujudan beberapa atau semua variabel acak yang menggambarkan domain. Hipotesis dalam bab ini adalah teori probabilistik tentang cara kerja domain, termasuk teori logika sebagai kasus khusus. Perhatikan contoh sederhana. Permen Kejutan favorit kami hadir dalam dua rasa: ceri (enak) dan jeruk nipis (ugh). Produsennya memiliki selera humor yang unik dan membungkus setiap permen dalam bungkus buram yang sama, apa pun rasanya. Permen ini dijual dalam kantong yang sangat besar, yang diketahui terdiri dari lima jenis sekali lagi, tidak dapat dibedakan dari luarnya:

h_1 : 100% *cherry*,
 h_2 : 75% *cherry* + 25% *lime*,
 h_3 : 50% *cherry* + 50% *lime*,
 h_4 : 25% *cherry* + 75% *lime*,
 h_5 : 100% *lime* .

Diberikan sekantong permen baru, variabel acak H (untuk hipotesis) menunjukkan jenis tas, dengan nilai yang mungkin h_1 hingga h_5 . H tidak dapat diamati secara langsung, tentu saja. Saat potongan permen dibuka dan diperiksa, data terungkap— $D_1, D_2, \dots, D_N$, di mana setiap D_i adalah variabel acak dengan nilai yang mungkin ceri dan jeruk nipis. Tugas dasar yang dihadapi oleh agen adalah untuk memprediksi rasa permen berikutnya. Meskipun tampak sepele, skenario ini berfungsi untuk memperkenalkan banyak masalah utama. Agen benar-benar perlu menyimpulkan teori dunianya, meskipun yang sangat sederhana.

Pembelajaran Bayesian hanya menghitung probabilitas setiap hipotesis, mengingat data, dan membuat prediksi berdasarkan itu. Artinya, prediksi dibuat dengan menggunakan semua hipotesis, yang diberi bobot oleh probabilitasnya, alih-alih hanya menggunakan satu hipotesis "terbaik". Dengan cara ini, pembelajaran direduksi menjadi inferensi probabilistik.

Biarkan D mewakili semua data, dengan nilai yang diamati d ; maka probabilitas masing-masing hipotesis diperoleh dengan aturan Bayes:

$$P(h_i|d) = \alpha P(d|h_i)P(h_i) \quad (2.1)$$

Sekarang, misalkan kita ingin membuat prediksi tentang kuantitas X yang tidak diketahui. Maka kita memiliki

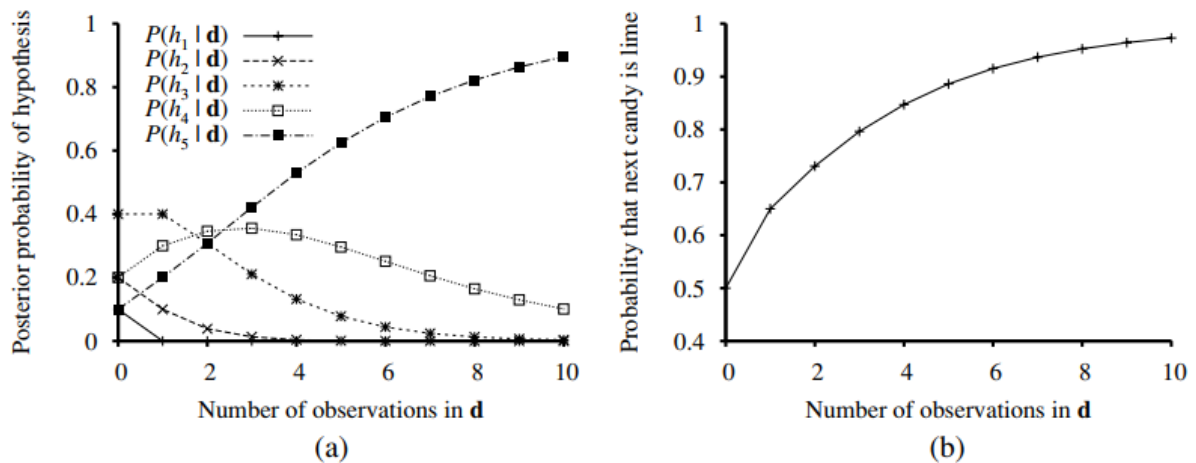
$$P(X|d) = \sum_i P(X|d, h_i) P(h_i|d) = \sum_i P(X|h_i) P(h_i|d) \quad (2.2)$$

di mana kita telah mengasumsikan bahwa setiap hipotesis menentukan distribusi probabilitas atas X . Persamaan ini menunjukkan bahwa prediksi adalah rata-rata tertimbang atas prediksi hipotesis individu. Hipotesis itu sendiri pada dasarnya adalah "perantara" antara data mentah dan prediksi. Kuantitas kunci dalam pendekatan Bayesian adalah hipotesis sebelumnya, $P(h_i)$, dan kemungkinan data di bawah setiap hipotesis, $P(d|h_i)$. Untuk contoh permen kita, kita akan berasumsi untuk saat ini bahwa distribusi sebelumnya atas $h_1, \dots, h_5$ diberikan oleh $\langle 0,1,0,2,0,4,0,2,0,1 \rangle$, seperti yang diiklankan oleh produsen. Kemungkinan data dihitung dengan asumsi bahwa pengamatan adalah **i.i.d.**, sehingga

$$P(X|d) = \prod_j P(d_j|h_i) \quad (2.3)$$

Misalnya, anggaplah tas itu benar-benar tas serba kapur (h_5) dan 10 permen pertama semuanya kapur; maka $P(d|h_3)$ adalah $0,5^{10}$, karena setengah dari permen dalam tas h_3 adalah kapur. Gambar 2.1(a) menunjukkan bagaimana probabilitas posterior dari lima hipotesis berubah ketika urutan 10 permen kapur diamati.

Perhatikan bahwa probabilitas dimulai dari nilai sebelumnya, jadi h_3 awalnya adalah pilihan yang paling mungkin dan tetap demikian setelah 1 permen kapur dibuka. Setelah 2 permen kapur dibuka, h_4 kemungkinan besar; setelah 3 atau lebih, h_5 (kantong serba kapur yang ditakuti) adalah yang paling mungkin. Setelah 10 berturut-turut, kita cukup yakin akan nasib kita. Gambar 2.1(b) menunjukkan probabilitas yang diprediksi bahwa permen berikutnya adalah kapur, berdasarkan Persamaan (2.2). Seperti yang kita duga, nilainya meningkat secara monoton menuju 1.



Gambar 2.1 (a) Probabilitas posterior $P(h_i | d_1, \dots, d_N)$ dari Persamaan (2.1). Jumlah observasi N berkisar antara 1 hingga 10, dan setiap observasi merupakan permen jeruk nipis. (b) Prediksi Bayesian $P(d_{N+1} = \text{lime} | d_1, \dots, d_N)$ dari Persamaan (2.2).

Contoh tersebut menunjukkan bahwa prediksi Bayesian pada akhirnya sesuai dengan hipotesis yang sebenarnya. Hal ini merupakan karakteristik pembelajaran Bayesian. Untuk setiap prior tetap yang tidak mengesampingkan hipotesis yang sebenarnya, probabilitas posterior dari setiap hipotesis yang salah, dalam kondisi teknis tertentu, pada akhirnya akan lenyap. Hal ini terjadi semata-mata karena probabilitas menghasilkan data yang "tidak khas" tanpa batas sangatlah kecil. Yang lebih penting, prediksi Bayesian optimal, baik kumpulan datanya kecil maupun besar. Dengan adanya hipotesis prior, prediksi lain diharapkan lebih jarang benar.

Optimalitas pembelajaran Bayesian tentu saja ada harganya. Untuk masalah pembelajaran nyata, ruang hipotesis biasanya sangat besar atau tak terhingga. Dalam beberapa kasus, penjumlahan dalam Persamaan (2.2) (atau integrasi, dalam kasus kontinu) dapat dilakukan dengan mudah, tetapi dalam kebanyakan kasus kita harus menggunakan metode perkiraan atau yang disederhanakan. Perkiraan yang sangat umum yang biasanya diadopsi dalam sains adalah membuat prediksi berdasarkan satu hipotesis paling mungkin yaitu, h_i yang memaksimalkan $P(h_i|d)$.

Ini sering disebut hipotesis a posteriori maksimum atau hipotesis MAP (diucapkan "em-ay-pee"). Prediksi yang dibuat berdasarkan hipotesis MAP h_{MAP} kira-kira bersifat Bayesian sejauh $P(X|d) \approx P(X|h_{MAP})$. Dalam contoh permen kita, $h_{MAP} = h_5$ setelah tiga permen jeruk nipis berturut-turut, sehingga pembelajar MAP kemudian memprediksi bahwa permen keempat adalah jeruk nipis dengan probabilitas 1,0 prediksi yang jauh lebih berbahaya daripada prediksi Bayesian sebesar 0,8 yang ditunjukkan pada Gambar 2.1(b). Seiring bertambahnya data yang masuk, prediksi MAP dan Bayesian menjadi semakin dekat, karena

pesaing hipotesis MAP menjadi semakin kecil kemungkinannya. Meskipun contoh kita tidak menunjukkannya, menemukan hipotesis MAP seringkali jauh lebih mudah daripada pembelajaran Bayesian, karena membutuhkan penyelesaian masalah optimasi, alih-alih masalah penjumlahan (atau integrasi) yang besar. Kita akan melihat contoh-contohnya nanti di bab ini. Baik dalam pembelajaran Bayesian maupun pembelajaran MAP , hipotesis prior

$P(h_i)$ memainkan peran penting. Kita telah melihat bahwa overfitting dapat terjadi ketika ruang hipotesis terlalu ekspresif, sehingga mengandung banyak hipotesis yang sesuai dengan kumpulan data. Alih-alih menetapkan batasan sembarangan pada hipotesis yang akan dipertimbangkan, metode pembelajaran Bayesian dan MAP menggunakan prior untuk mengurangi kompleksitas.

Biasanya, hipotesis yang lebih kompleks memiliki probabilitas prior yang lebih rendah sebagian karena biasanya terdapat lebih banyak hipotesis yang kompleks daripada hipotesis sederhana. Di sisi lain, hipotesis yang lebih kompleks memiliki kapasitas yang lebih besar untuk menyesuaikan data. (Dalam kasus ekstrem, tabel pencarian dapat mereproduksi data secara tepat dengan probabilitas 1.) Oleh karena itu, hipotesis prior mewujudkan tradeoff antara kompleksitas hipotesis dan tingkat kesesuaiannya dengan data.

Kita dapat melihat efek tradeoff ini paling jelas dalam kasus logis, di mana H hanya berisi hipotesis deterministik. Dalam kasus tersebut, $P(\mathbf{d}|h_i)$ bernilai 1 jika h_i konsisten dan 0 jika tidak. Dengan melihat Persamaan (2.1), kita melihat bahwa hMAP akan menjadi teori logika paling sederhana yang konsisten dengan data. Oleh karena itu, pembelajaran a posteriori maksimum memberikan perwujudan alami dari pisau cukur Ockham. Wawasan lain tentang tradeoff antara kompleksitas dan derajat kesesuaian diperoleh dengan mengambil logaritma Persamaan (2.1). Memilih h_{MAP} untuk memaksimalkan $P(\mathbf{d}|h_i)P(h_i)$ setara dengan meminimalkan

$$-\log_2 P(\mathbf{d}|h_i) - \log_2 P(h_i)$$

Dengan menggunakan hubungan antara pengkodean informasi dan probabilitas yang kami perkenalkan, kita melihat bahwa suku $-\log_2 P(h_i)$ sama dengan jumlah bit yang diperlukan untuk menentukan hipotesis h_i . Lebih lanjut, $-\log_2 P(\mathbf{d}|h_i)$ adalah jumlah bit tambahan yang diperlukan untuk menentukan data, mengingat hipotesis tersebut. (Untuk melihat ini, pertimbangkan bahwa tidak ada bit yang diperlukan jika hipotesis memprediksi data secara tepat seperti halnya dengan h_5 dan rangkaian permen jeruk nipis dan $\log_2 1 = 0$.)

Oleh karena itu, pembelajaran MAP adalah memilih hipotesis yang memberikan kompresi data maksimum. Tugas yang sama ditangani secara lebih langsung oleh metode pembelajaran panjang deskripsi minimum, atau MDL. Sementara pembelajaran MAP mengekspresikan kesederhanaan dengan menetapkan probabilitas yang lebih tinggi untuk hipotesis yang lebih sederhana, MDL mengekspresikannya secara langsung dengan menghitung bit dalam pengkodean biner hipotesis dan data.

Penyederhanaan terakhir disediakan dengan mengasumsikan prior seragam atas ruang hipotesis. Dalam kasus tersebut, pembelajaran MAP disederhanakan menjadi pemilihan h_i yang memaksimalkan $P(\mathbf{d}|h_i)$. Ini disebut hipotesis kemungkinan maksimum (ML), hML. Pembelajaran kemungkinan maksimum sangat umum dalam statistika, suatu disiplin ilmu di mana banyak peneliti meragukan sifat subjektif dari hipotesis sebelumnya.

Pendekatan ini masuk akal ketika tidak ada alasan untuk lebih memilih satu hipotesis daripada hipotesis lainnya secara apriori misalnya, ketika semua hipotesis sama kompleksnya. Pendekatan ini memberikan pendekatan yang baik untuk pembelajaran Bayesian dan MAP

ketika kumpulan data besar, karena data membanjiri distribusi prior atas hipotesis, tetapi pendekatan ini memiliki masalah (seperti yang akan kita lihat) dengan kumpulan data kecil.

2.2 PEMBELAJARAN DENGAN DATA LENGKAP

Tugas umum mempelajari model probabilitas, dengan data yang diasumsikan dihasilkan dari model tersebut, disebut estimasi densitas. (Istilah ini awalnya diterapkan pada fungsi densitas probabilitas untuk variabel kontinu, tetapi sekarang juga digunakan untuk distribusi diskrit.) Bagian ini membahas kasus paling sederhana, di mana kita memiliki data lengkap. Data dikatakan lengkap ketika setiap titik data berisi nilai untuk setiap variabel dalam model probabilitas yang sedang dipelajari. Kami berfokus pada pembelajaran parameter menemukan parameter numerik untuk model probabilitas yang strukturnya tetap. Misalnya, kita mungkin tertarik mempelajari probabilitas bersyarat dalam jaringan Bayesian dengan struktur tertentu. Kita juga akan melihat secara singkat masalah pembelajaran struktur dan estimasi densitas nonparametrik.

Pembelajaran parameter kemungkinan maksimum: Model diskrit

Misalkan kita membeli sekantong permen jeruk nipis dan ceri dari produsen baru yang proporsi jeruk nipis-cerinya sama sekali tidak diketahui; Fraksinya bisa berada di antara 0 dan 1. Dalam hal ini, kita memiliki kontinum hipotesis. Parameter dalam kasus ini, yang kita sebut θ , adalah proporsi permen ceri, dan hipotesisnya adalah h_θ . (Proporsi jeruk nipis hanya $1 - \theta$.) Jika kita berasumsi bahwa semua proporsi memiliki kemungkinan yang sama secara apriori, maka pendekatan kemungkinan maksimum masuk akal.

Jika kita memodelkan situasi dengan jaringan Bayesian, kita hanya perlu satu variabel acak, Rasa (rasa permen yang dipilih secara acak dari kantong). Variabel ini memiliki nilai ceri dan jeruk nipis, dengan probabilitas ceri adalah θ (lihat Gambar 2.2(a)). Sekarang, misalkan kita membuka bungkus permen N , yang c adalah ceri dan $\ell = N - c$ adalah jeruk nipis. Menurut Persamaan (2.3), kemungkinan dari kumpulan data khusus ini adalah

$$P(\mathbf{d}|h_\theta) \prod_{j=1}^N P(d_j|h_\theta) = \theta^c \cdot (1 - \theta)^\ell$$

Hipotesis kemungkinan maksimum diberikan oleh nilai θ yang memaksimalkan ekspresi ini. Nilai yang sama diperoleh dengan memaksimalkan logaritma kemungkinan,

$$L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = \sum_{j=1}^N \log P(d_j|h_\theta) = c \log \theta + \ell \log(1 - \theta)$$

(Dengan mengambil logaritma, kita menyederhanakan hasil perkalian menjadi jumlah data, yang biasanya lebih mudah dimaksimalkan.) Untuk menemukan nilai kemungkinan maksimum θ , kita bedakan L terhadap θ dan tetapkan ekspresi yang dihasilkan ke nol:

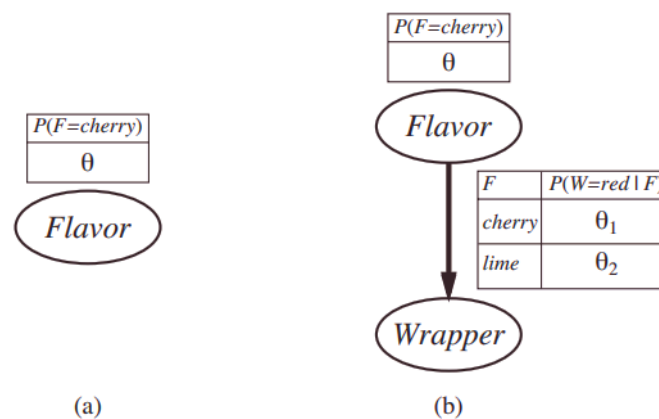
$$\frac{dL(\mathbf{d}|h_{\theta})}{d\theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \Rightarrow \theta = \frac{c}{c+\ell} = \frac{c}{N}$$

Dalam bahasa Inggris, hipotesis kemungkinan maksimum (H_{ML}) menyatakan bahwa proporsi ceri yang sebenarnya di dalam kantong sama dengan proporsi yang diamati pada permen yang belum dibungkus sejauh ini!

Tampaknya kita telah melakukan banyak upaya untuk menemukan hal yang jelas. Namun, faktanya, kita telah menyusun satu metode standar untuk pembelajaran parameter kemungkinan maksimum, sebuah metode dengan penerapan yang luas:

1. Tuliskan ekspresi untuk kemungkinan data sebagai fungsi parameter.
2. Tuliskan turunan dari log likelihood terhadap setiap parameter.
3. Temukan nilai parameter sehingga turunannya nol.

Langkah tersulit biasanya adalah yang terakhir. Dalam contoh kita, langkah ini sederhana, tetapi kita akan melihat bahwa dalam banyak kasus kita perlu menggunakan algoritma solusi iteratif atau teknik optimasi numerik lainnya. Contoh ini juga menggambarkan masalah signifikan dengan pembelajaran kemungkinan maksimum secara umum: ketika kumpulan data cukup kecil sehingga beberapa peristiwa belum teramati misalnya, tidak ada permen ceri hipotesis kemungkinan maksimum menetapkan probabilitas nol untuk peristiwa tersebut. Berbagai trik digunakan untuk menghindari masalah ini, seperti menginisialisasi hitungan untuk setiap peristiwa menjadi 1, bukan 0.



Gambar 2.2 (a) Model jaringan Bayesian untuk kasus permen dengan proporsi ceri dan jeruk nipis yang tidak diketahui. (b) Model untuk kasus di mana warna pembungkus bergantung (secara probabilistik) pada rasa permen.

Mari kita lihat contoh lain. Misalkan produsen permen baru ini ingin memberi sedikit petunjuk kepada konsumen dan menggunakan bungkus permen berwarna merah dan hijau. Pembungkus untuk setiap permen dipilih secara probabilistik, berdasarkan distribusi kondisional yang tidak diketahui, bergantung pada rasa. Model probabilitas yang sesuai ditunjukkan pada Gambar 2.2(b). Perhatikan bahwa model ini memiliki tiga parameter:

θ , θ_1 , dan θ_2 . Dengan parameter-parameter ini, probabilitas melihat, misalnya, permen ceri dalam pembungkus hijau dapat diperoleh dari semantik standar untuk jaringan Bayesian:

$$\begin{aligned} P(\text{Flavor} = \text{cherry}, \text{Wrapper} = \text{green} | h_{\theta, \theta_1, \theta_2}) \\ = P(\text{Flavor} = \text{cherry} | h_{\theta, \theta_1, \theta_2}) P(\text{Wrapper} = \text{green} | \text{Flavor} = \text{cherry}, h_{\theta, \theta_1, \theta_2}) \\ = \theta \cdot (1 - \theta_1). \end{aligned}$$

Sekarang kita buka bungkus permen N , yang c adalah ceri dan ℓ adalah jeruk nipis. Jumlah bungkusnya adalah sebagai berikut: r_c ceri memiliki bungkus merah dan g_c berwarna hijau, sedangkan r_ℓ jeruk nipis berwarna merah dan g_ℓ berwarna hijau. Kemungkinan data diberikan oleh

$$P(\mathbf{d} | h_{\theta, \theta_1, \theta_2}) = \theta^c (1 - \theta)^\ell \cdot \theta_1^{r_c} (1 - \theta_1)^{g_c} \cdot \theta_2^{r_\ell} (1 - \theta_2)^{g_\ell}$$

Ini terlihat sangat mengerikan, tetapi mengambil logaritma membantu:

$$L = [c \log \theta + \ell \log(1 - \theta)] + [r_c \log \theta_1 + g_c \log(1 - \theta_1)] + [r_\ell \log \theta_2 + g_\ell \log(1 - \theta_2)]$$

Manfaat menggunakan logaritma jelas: log likelihood adalah jumlah dari tiga suku, yang masing-masing mengandung satu parameter. Ketika kita mengambil turunan terhadap setiap parameter dan menetakannya ke nol, kita mendapatkan tiga persamaan independen, yang masing-masing hanya mengandung satu parameter:

$$\begin{aligned} \frac{\partial L}{\partial \theta} &= \frac{c}{\theta} - \frac{\ell}{1 - \theta} - 0 & \Rightarrow \theta &= \frac{c}{c + \ell} \\ \frac{\partial L}{\partial \theta_1} &= \frac{r_c}{\theta_1} - \frac{g_c}{1 - \theta_1} - 0 & \Rightarrow \theta_1 &= \frac{r_c}{r_c + g_c} \\ \frac{\partial L}{\partial \theta_2} &= \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1 - \theta_2} - 0 & \Rightarrow \theta_2 &= \frac{r_\ell}{r_\ell + g_\ell} \end{aligned}$$

Solusi untuk θ sama seperti sebelumnya. Solusi untuk θ_1 , probabilitas permen ceri berbungkus merah, adalah fraksi permen ceri berbungkus merah yang teramati, dan demikian pula untuk θ_2 . Hasil ini sangat meyakinkan, dan mudah dilihat bahwa hasil ini dapat diperluas ke jaringan Bayesian mana pun yang probabilitas kondisional direpresentasikan sebagai tabel.

Poin terpenting adalah, dengan data yang lengkap, masalah pembelajaran parameter kemungkinan maksimum untuk jaringan Bayesian terurai menjadi masalah pembelajaran terpisah, satu untuk setiap parameter. Poin kedua adalah bahwa nilai parameter untuk suatu variabel, mengingat induknya, hanyalah frekuensi teramati dari nilai variabel untuk setiap pengaturan nilai induk. Seperti sebelumnya, kita harus berhati-hati untuk menghindari angka nol ketika kumpulan data kecil.

Model Bayes Naif

Mungkin model jaringan Bayesian yang paling umum digunakan dalam pembelajaran mesin adalah model Naive Bayes. Dalam model ini, variabel "kelas" C (yang akan diprediksi) adalah akar dan variabel "atribut" X_i adalah daun. Model ini "naif" karena mengasumsikan

bahwa atribut-atributnya saling independen secara kondisional, mengingat kelasnya. (Model pada Gambar 2.2(b) adalah model Bayes Naif dengan kelas Flavor dan hanya satu atribut, Wrapper.) Dengan asumsi variabel Boolean, parameternya adalah

$$\theta = P(C = \text{true}), \theta_{i1} = P(X_i = \text{true} | C = \text{true}), \theta_{i2} = P(X_i = \text{true} | C = \text{false}).$$

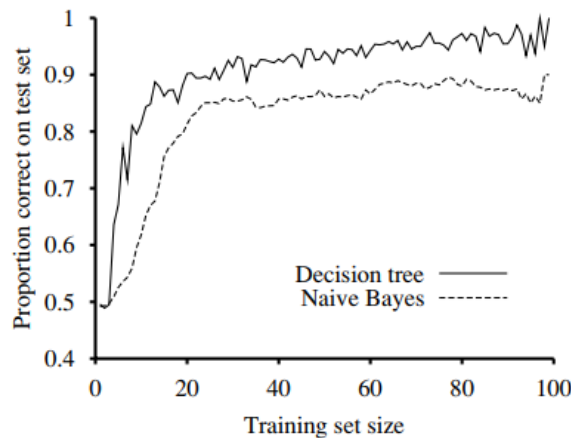
Nilai parameter kemungkinan maksimum ditemukan dengan cara yang persis sama seperti pada Gambar 2.2(b). Setelah model dilatih dengan cara ini, model dapat digunakan untuk mengklasifikasikan contoh-contoh baru yang variabel kelas C -nya tidak teramati. Dengan nilai atribut teramati $x_1, \dots, x_n$, probabilitas setiap kelas diberikan oleh

$$P(C | x_1, \dots, x_n) = \alpha P(C) \prod_i P(x_i | C)$$

Prediksi deterministik dapat diperoleh dengan memilih kelas yang paling mungkin. Gambar 2.3 menunjukkan kurva pembelajaran untuk metode ini ketika diterapkan pada masalah restoran.

Metode ini belajar cukup baik tetapi tidak sebaik pembelajaran pohon keputusan; ini mungkin karena hipotesis sebenarnya yang merupakan pohon keputusan tidak dapat direpresentasikan secara tepat menggunakan model Bayes naif. Pembelajaran Bayes naif ternyata bekerja dengan sangat baik dalam berbagai aplikasi; versi yang ditingkatkan adalah salah satu algoritma pembelajaran tujuan umum yang paling efektif.

Pembelajaran Bayes naif berskala baik untuk masalah yang sangat besar: dengan n atribut Boolean, hanya ada $2n + 1$ parameter, dan tidak diperlukan pencarian untuk menemukan h_{ML} , hipotesis Bayes naif kemungkinan maksimum. Akhirnya, sistem pembelajaran Bayes naif tidak memiliki kesulitan dengan data yang berisik atau hilang dan dapat memberikan prediksi probabilistik jika sesuai.



Gambar 2.3 Kurva pembelajaran untuk pembelajaran Bayes naif; kurva pembelajaran untuk pembelajaran pohon keputusan diperlihatkan untuk perbandingan.

Pembelajaran parameter probabilitas maksimum: Model kontinu

Model probabilitas kontinu seperti model Gaussian linear. Karena variabel kontinu terdapat di mana-mana dalam aplikasi dunia nyata, penting untuk mengetahui cara mempelajari parameter model kontinu dari data. Prinsip-prinsip pembelajaran probabilitas maksimum identik dalam kasus kontinu dan diskrit.

Mari kita mulai dengan kasus yang sangat sederhana: mempelajari parameter fungsi kepadatan Gaussian pada satu variabel. Artinya, data dihasilkan sebagai berikut:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

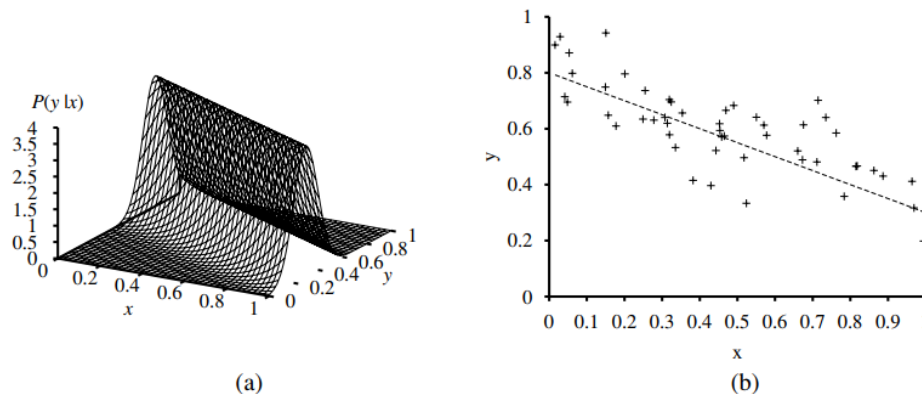
Parameter model ini adalah mean μ dan standar deviasi σ . (Perhatikan bahwa “konstanta” normalisasi bergantung pada σ , jadi kita tidak bisa mengabaikannya.) Misalkan nilai yang diamati adalah $x_1, \dots, x_N$. Maka log likelihoodnya adalah

$$L = \sum_{j=1}^N \text{Log} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_j-\mu)^2}{2\sigma^2}} = N(-\log \sqrt{2\pi} - \log \sigma) - \sum_{j=1}^N \frac{(x_j - \mu)^2}{2\sigma^2}$$

Dengan menetapkan turunannya ke nol seperti biasa, kita memperoleh

$$\begin{aligned} \frac{\partial L}{\partial \theta} &= \frac{1}{\sigma^2} \sum_{j=1}^N (x_j - \mu) = 0 & \Rightarrow \mu &= \frac{\sum_j x_j}{N} \\ \frac{\partial L}{\partial \theta} &= \frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{j=1}^N (x_j - \mu)^2 = 0 & \Rightarrow \sigma &= \sqrt{\frac{\sum_j (x_j - \mu)^2}{N}} \end{aligned} \quad (2.4)$$

Artinya, nilai kemungkinan maksimum dari rata-rata adalah rata-rata sampel dan nilai kemungkinan maksimum dari simpangan baku adalah akar kuadrat dari varians sampel. Sekali lagi, ini adalah hasil yang meyakinkan yang mengonfirmasi praktik "akal sehat".



Gambar 2.4 (a) Model Gaussian linear yang dideskripsikan sebagai $y = \theta_1 x + \theta_2$ ditambah derau Gaussian dengan varians tetap. (b) Satu set 50 titik data yang dihasilkan dari model ini.

Sekarang perhatikan model Gaussian linear dengan satu induk kontinu X dan satu anak kontinu Y . Sebagaimana dijelaskan, Y memiliki distribusi Gaussian yang rata-ratanya

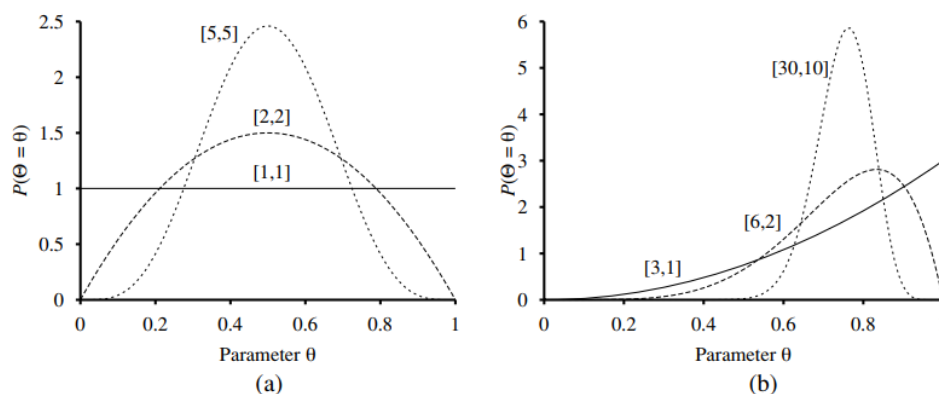
bergantung secara linear pada nilai X dan simpangan bakunya tetap. Untuk mempelajari distribusi kondisional $P(Y | X)$, kita dapat memaksimalkan kemungkinan kondisional.

$$P(y|x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-(\theta_1 x + \theta_2))^2}{2\sigma^2}} \quad (2.5)$$

Di sini, parameternya adalah θ_1 , θ_2 , dan σ . Data tersebut merupakan kumpulan pasangan (x_j, y_j) , seperti yang diilustrasikan pada Gambar 2.4. Dengan menggunakan metode yang biasa, kita dapat menemukan nilai kemungkinan maksimum dari parameter tersebut. Intinya di sini berbeda. Jika kita hanya mempertimbangkan parameter θ_1 dan θ_2 yang mendefinisikan hubungan linier antara x dan y , menjadi jelas bahwa memaksimalkan log likelihood terhadap parameter-parameter ini sama dengan meminimalkan pembilang $(y - (\theta_1 x + \theta_2))^2$ dalam eksponen Persamaan (2.5). Ini adalah kerugian L_2 , kuadrat kesalahan antara nilai aktual y dan prediksi $\theta_1 x + \theta_2$. Ini adalah kuantitas yang diminimalkan oleh prosedur regresi linier standar. Sekarang kita dapat memahami alasannya: meminimalkan jumlah galat kuadrat menghasilkan model garis lurus dengan probabilitas maksimum, asalkan data dihasilkan dengan derau Gaussian dengan varians tetap.

Pembelajaran Parameter Bayesian

Pembelajaran dengan probabilitas maksimum menghasilkan beberapa prosedur yang sangat sederhana, tetapi memiliki beberapa kekurangan serius pada set data yang kecil. Misalnya, setelah melihat satu permen ceri, hipotesis probabilitas maksimumnya adalah bahwa kantong tersebut 100% ceri (yaitu, $\theta = 1,0$). Kecuali hipotesis sebelumnya seseorang adalah bahwa kantong tersebut pasti seluruhnya ceri atau seluruhnya jeruk nipis, ini bukanlah kesimpulan yang masuk akal. Kemungkinan besar kantong tersebut merupakan campuran jeruk nipis dan ceri. Pendekatan Bayesian untuk pembelajaran parameter dimulai dengan mendefinisikan distribusi probabilitas sebelumnya atas hipotesis yang mungkin. Kami menyebutnya hipotesis sebelumnya. Kemudian, ketika data tiba, distribusi probabilitas posterior diperbarui.



Gambar 2.5 Contoh distribusi beta[a, b] untuk berbagai nilai [a, b].

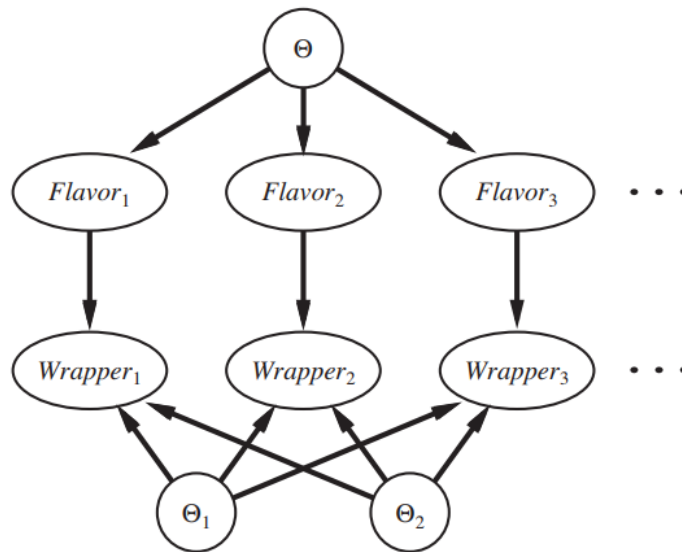
Contoh permen pada Gambar 2.2(a) memiliki satu parameter, θ : probabilitas bahwa sepotong permen yang dipilih secara acak beraroma ceri. Dalam pandangan Bayesian, θ adalah

nilai (yang tidak diketahui) dari variabel acak Θ yang mendefinisikan ruang hipotesis; prior hipotesis hanyalah distribusi prior $\mathbf{P}(\Theta)$. Dengan demikian, $P(\Theta = \theta)$ adalah probabilitas prior bahwa kantong permen tersebut memiliki fraksi θ permen ceri.

Jika parameter θ dapat berupa nilai apa pun antara 0 dan 1, maka $\mathbf{P}(\Theta)$ harus berupa distribusi kontinu yang bukan nol hanya antara 0 dan 1 dan yang terintegrasi ke 1. Kepadatan seragam $P(\theta) = \text{Uniform}[0, 1](\theta)$ adalah salah satu kandidat. Ternyata kepadatan seragam merupakan anggota dari keluarga distribusi beta. Setiap distribusi beta didefinisikan oleh dua *hiperparameter* a dan b sehingga

$$\text{beta}[a, b](\theta) = \alpha \theta^{a-1} (1 - \theta)^{b-1} \quad (2.6)$$

untuk θ dalam rentang $[0, 1]$. Konstanta normalisasi α , yang membuat distribusi terintegrasi ke 1, bergantung pada a dan b . Gambar 2.5 menunjukkan seperti apa distribusi untuk berbagai nilai a dan b . Nilai rata-rata distribusi adalah $a/(a + b)$, sehingga nilai a yang lebih besar menunjukkan keyakinan bahwa Θ lebih dekat ke 1 daripada ke 0. Nilai $a + b$ yang lebih besar membuat distribusi lebih memuncak, menunjukkan kepastian yang lebih besar tentang nilai Θ . Dengan demikian, keluarga beta menyediakan berbagai kemungkinan yang berguna untuk hipotesis sebelumnya.



Gambar 2.6 Jaringan Bayesian yang sesuai dengan proses pembelajaran Bayesian. Distribusi posterior untuk variabel parameter Θ , Θ_1 , dan Θ_2 dapat disimpulkan dari distribusi sebelumnya dan bukti dalam variabel $Flavor_i$ dan $Wrapper_i$.

Selain fleksibilitasnya, keluarga beta memiliki sifat luar biasa lainnya: jika Θ memiliki beta sebelumnya $[a, b]$, maka, setelah titik data diamati, distribusi posterior untuk Θ juga merupakan distribusi beta. Dengan kata lain, beta tertutup saat diperbarui. Keluarga beta disebut prior konjugat untuk keluarga distribusi variabel Boolean. Mari kita lihat cara kerjanya. Misalkan kita mengamati permen ceri; maka kita memiliki

$$\begin{aligned}
 P(\theta|D_1 = \text{cherry}) &= \alpha P(D_1 = \text{cherry}|\theta)P(\theta) \\
 &= \alpha' \theta \cdot \text{beta}[a, b](\theta) = \alpha' \theta \cdot \theta^{a-1}(1 - \theta)^{b-1} \\
 &= \alpha \theta^a (1 - \theta)^{b-1} = \text{beta}[a + 1, b](\theta)
 \end{aligned}$$

Dengan demikian, setelah melihat permen ceri, kita cukup menambah parameter a untuk mendapatkan posteriornya; demikian pula, setelah melihat permen jeruk nipis, kita menambah parameter b . Dengan demikian, kita dapat memandang hiperparameter a dan b sebagai hitungan virtual, dalam artian bahwa $\text{beta}[a, b]$ sebelumnya berperilaku persis seolah-olah kita memulai dengan $\text{beta}[1, 1]$ sebelumnya yang seragam dan melihat $a - 1$ permen ceri aktual dan $b - 1$ permen jeruk nipis aktual.

Dengan memeriksa sekuens distribusi beta untuk nilai a dan b yang meningkat, dengan menjaga proporsinya tetap, kita dapat melihat dengan jelas bagaimana distribusi posterior atas parameter θ berubah seiring dengan kedatangan data. Misalnya, anggaplah sekantong permen yang sebenarnya adalah 75% ceri. Gambar 2.5(b) menunjukkan sekuens $\text{beta}[3, 1]$, $\text{beta}[6, 2]$, $\text{beta}[30, 10]$. Jelas, distribusi tersebut konvergen ke puncak sempit di sekitar nilai θ yang sebenarnya.

Untuk set data yang besar, pembelajaran Bayesian (setidaknya dalam kasus ini) konvergen ke jawaban yang sama dengan pembelajaran kemungkinan maksimum. Sekarang mari kita pertimbangkan kasus yang lebih rumit. Jaringan pada Gambar 2.2(b) memiliki tiga parameter, θ , θ_1 , dan θ_2 , di mana θ_1 adalah probabilitas pembungkus merah pada permen ceri dan θ_2 adalah probabilitas pembungkus merah pada permen jeruk nipis. Hipotesis Bayesian sebelumnya harus mencakup ketiga parameter—yaitu, kita perlu menentukan $\mathbf{P}(\theta, \theta_1, \theta_2)$. Biasanya, kita mengasumsikan independensi parameter:

$$\mathbf{P}(\theta, \theta_1, \theta_2) = \mathbf{P}(\theta)\mathbf{P}(\theta_1)\mathbf{P}(\theta_2)$$

Dengan asumsi ini, setiap parameter dapat memiliki distribusi beta-nya sendiri yang diperbarui secara terpisah saat data tiba. Gambar 2.6 menunjukkan bagaimana kita dapat menggabungkan hipotesis sebelumnya dan data apa pun ke dalam satu jaringan Bayesian. Simpul θ , θ_1 , θ_2 tidak memiliki induk. Namun, setiap kali kita mengamati bungkus permen dan rasa permen yang sesuai, kita menambahkan simpul Flavor_i , yang bergantung pada parameter rasa θ :

$$P(\text{Flavor}_i = \text{cherry}|\theta = \theta) = \theta$$

Kami juga menambahkan simpul Wrapper_i , yang bergantung pada θ_1 dan θ_2 :

$$\begin{aligned}
 P(\text{Wrapper}_i = \text{red} | \text{Flavor}_i = \text{cherry}, \theta_1 = \theta_1) &= \theta_1 \\
 P(\text{Wrapper}_i = \text{red} | \text{Flavor}_i = \text{lime}, \theta_2 = \theta_2) &= \theta_2
 \end{aligned}$$

Kini, seluruh proses pembelajaran Bayesian dapat diformulasikan sebagai masalah inferensi. Kita menambahkan simpul bukti baru, lalu mengkueri simpul yang tidak diketahui (dalam kasus ini, $\theta, \theta_1, \theta_2$). Formulasi pembelajaran dan prediksi ini memperjelas bahwa

pembelajaran Bayesian tidak memerlukan "prinsip pembelajaran" tambahan. Lebih lanjut, pada dasarnya, hanya ada satu algoritma pembelajaran algoritma inferensi untuk jaringan Bayesian. Tentu saja, sifat jaringan ini agak berbeda karena potensi besarnya jumlah variabel bukti yang mewakili set pelatihan dan prevalensi variabel parameter bernilai kontinu.

Mempelajari Struktur Jaringan Bayes

Sejauh ini, kita berasumsi bahwa struktur jaringan Bayes telah diketahui dan kita hanya mencoba mempelajari parameternya. Struktur jaringan merepresentasikan pengetahuan kausal dasar tentang domain yang seringkali mudah diberikan oleh seorang pakar, atau bahkan pengguna awam. Namun, dalam beberapa kasus, model kausal mungkin tidak tersedia atau dapat diperdebatkan misalnya, beberapa perusahaan telah lama mengklaim bahwa merokok tidak menyebabkan kanker sehingga penting untuk memahami bagaimana struktur jaringan Bayes dapat dipelajari dari data. Bagian ini memberikan gambaran singkat tentang gagasan utamanya.

Pendekatan yang paling jelas adalah mencari model yang baik. Kita dapat memulai dengan model yang tidak mengandung tautan dan mulai menambahkan induk untuk setiap simpul, menyesuaikan parameter dengan metode yang baru saja kita bahas, dan mengukur akurasi model yang dihasilkan. Sebagai alternatif, kita dapat memulai dengan tebakan awal pada struktur dan menggunakan pencarian hill-climbing atau simulasi annealing untuk melakukan modifikasi, menyetel ulang parameter setelah setiap perubahan dalam struktur.

Modifikasi dapat mencakup pembalikan, penambahan, atau penghapusan tautan. Kita tidak boleh memasukkan siklus dalam prosesnya, sehingga banyak algoritma berasumsi bahwa urutan diberikan untuk variabel, dan bahwa sebuah simpul hanya dapat memiliki induk di antara simpul-simpul yang muncul lebih awal dalam urutan. Untuk generalisasi penuh, kita juga perlu menelusuri kemungkinan urutan. Ada dua metode alternatif untuk menentukan kapan suatu struktur yang baik telah ditemukan. Yang pertama adalah menguji apakah pernyataan independensi bersyarat yang tersirat dalam struktur tersebut benar-benar terpenuhi dalam data. Misalnya, penggunaan model Bayes naif untuk masalah restoran mengasumsikan bahwa

$$P(Fri/Sat, Bar | WillWait) = P(Fri/Sat | WillWait)P(Bar | WillWait)$$

dan kita dapat memeriksa dalam data bahwa persamaan yang sama berlaku antara frekuensi kondisional yang sesuai. Namun, meskipun struktur tersebut menggambarkan sifat kausal domain yang sebenarnya, fluktuasi statistik dalam kumpulan data berarti bahwa persamaan tersebut tidak akan pernah terpenuhi secara tepat, sehingga kita perlu melakukan uji statistik yang sesuai untuk melihat apakah ada cukup bukti bahwa hipotesis independensi dilanggar. Kompleksitas jaringan yang dihasilkan akan bergantung pada ambang batas yang digunakan untuk pengujian ini semakin ketat uji independensi, semakin banyak tautan yang akan ditambahkan dan semakin besar bahaya overfitting.

Pendekatan yang lebih konsisten dengan gagasan dalam bab ini adalah menilai sejauh mana model yang diusulkan menjelaskan data (dalam arti probabilistik). Namun, kita harus

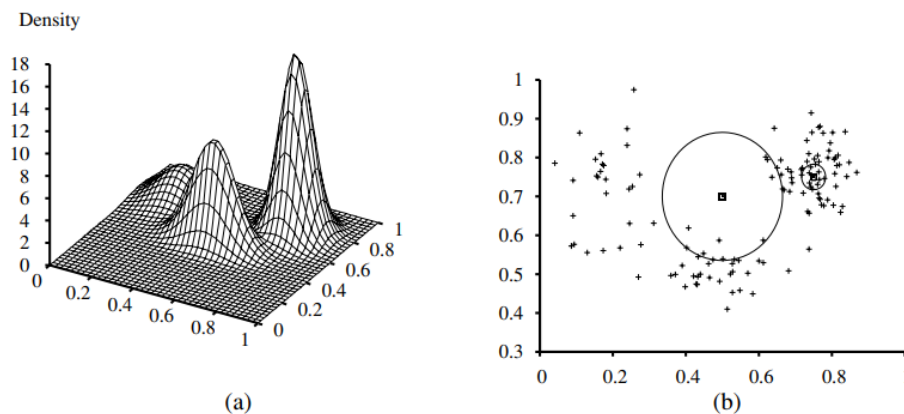
berhati-hati dalam mengukurnya. Jika kita hanya mencoba menemukan hipotesis kemungkinan maksimum, kita akan berakhir dengan jaringan yang terhubung penuh, karena menambahkan lebih banyak induk ke sebuah simpul tidak dapat mengurangi kemungkinan. Kita terpaksa memberikan penalti pada kompleksitas model dengan cara tertentu. Pendekatan MAP (atau MDL) hanya mengurangi penalti dari kemungkinan setiap struktur (setelah penyetelan parameter) sebelum membandingkan struktur yang berbeda. Pendekatan Bayesian menempatkan prior gabungan atas struktur dan parameter. Biasanya terdapat terlalu banyak struktur untuk dijumlahkan (jumlah variabelnya supereksponensial), sehingga sebagian besar praktisi menggunakan MCMC untuk mengambil sampel atas struktur.

Penalti kompleksitas (baik dengan metode MAP maupun Bayesian) memperkenalkan hubungan penting antara struktur optimal dan sifat representasi untuk distribusi kondisional dalam jaringan. Dengan distribusi tabular, penalti kompleksitas untuk distribusi simpul tumbuh secara eksponensial seiring dengan jumlah induknya, tetapi dengan, katakanlah, distribusi OR-bising, penalti tersebut hanya tumbuh secara linear. Ini berarti bahwa pembelajaran dengan model OR-bising (atau model berparameterisasi kompak lainnya) cenderung menghasilkan struktur yang dipelajari dengan lebih banyak induk daripada pembelajaran dengan distribusi tabular.

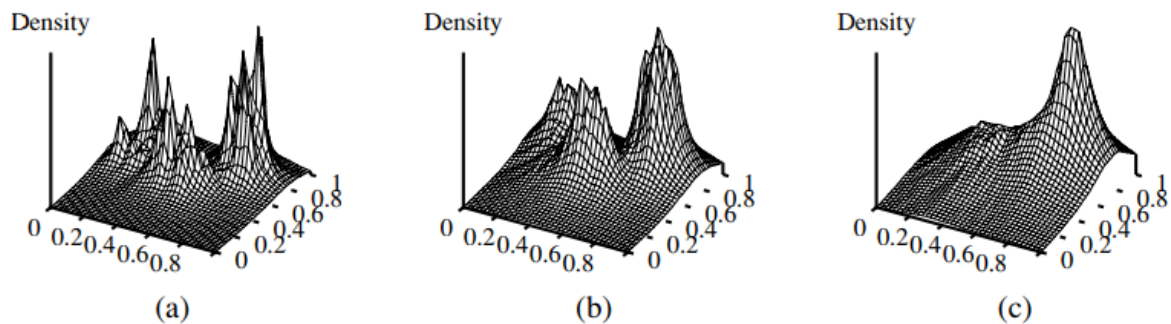
Estimasi Densitas dengan Model Nonparametrik

Metode nonparametrik pada Bagian 18.8 memungkinkan untuk mempelajari model probabilitas tanpa membuat asumsi apa pun tentang struktur dan parameterisasinya. Estimasi densitas nonparametrik biasanya dilakukan dalam domain kontinu, seperti yang ditunjukkan pada Gambar 2.7(a). Gambar tersebut menunjukkan fungsi densitas probabilitas pada ruang yang didefinisikan oleh dua variabel kontinu. Pada Gambar 2.7(b), kita melihat sampel titik data dari fungsi densitas ini. Pertanyaannya adalah, dapatkah kita memulihkan model dari sampel tersebut?

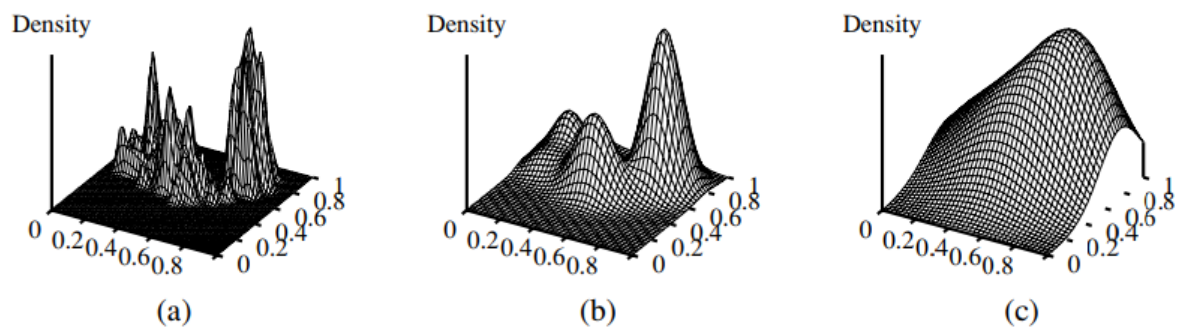
Pertama, kita akan mempertimbangkan model k -tetangga-terdekat. (Pada Bab 18, kita melihat model tetangga-terdekat untuk klasifikasi dan regresi; di sini kita melihatnya untuk estimasi densitas.) Dengan sampel titik data, untuk menaksir densitas probabilitas yang tidak diketahui pada titik kueri x , kita cukup mengukur densitas titik data di sekitar x . Gambar 2.7(b) menunjukkan dua titik kueri (kotak kecil). Untuk setiap titik kueri, kami telah menggambar lingkaran terkecil yang melingkupi 10 tetangga 10 tetangga terdekat. Kita dapat melihat bahwa lingkaran pusat berukuran besar, yang berarti terdapat kepadatan rendah di sana, dan lingkaran di sebelah kanan berukuran kecil, yang berarti terdapat kepadatan tinggi di sana. Pada Gambar 2.8, kami menunjukkan tiga plot estimasi kepadatan menggunakan k —tetangga terdekat, untuk nilai k yang berbeda. Tampak jelas bahwa (b) hampir tepat, sementara (a) terlalu runcing (k terlalu kecil) dan (c) terlalu halus (k terlalu besar).



Gambar 2.7 (a) Plot 3D campuran Gaussian dari Gambar 2.11(a). (b) Sampel 128 titik dari campuran, bersama dengan dua titik kueri (kotak kecil) dan 10 lingkungan terdekatnya (lingkaran sedang dan besar).



Gambar 2.8 Estimasi kepadatan menggunakan k-nearest-neighbors, diterapkan pada data pada Gambar 2.7(b), untuk $k = 3, 10$, dan 40 secara berurutan. Nilai $k = 3$ terlalu runcing, 40 terlalu halus, dan 10 hampir tepat. Nilai terbaik untuk k dapat dipilih melalui validasi silang.



Gambar 2.9 Estimasi kepadatan kernel untuk data pada Gambar 2.7(b), menggunakan kernel Gaussian dengan $w = 0.02, 0.07$, dan 0.20 berturut-turut. $w = 0.07$ sudah mendekati benar.

Kemungkinan lain adalah menggunakan fungsi kernel, seperti yang kami lakukan untuk regresi terbobot lokal. Untuk menerapkan model kernel pada estimasi densitas, asumsikan bahwa

setiap titik data menghasilkan fungsi densitas kecilnya sendiri, menggunakan kernel Gaussian. Densitas yang diestimasi pada titik kueri x kemudian menjadi densitas rata-rata yang diberikan oleh setiap fungsi kernel:

$$P(x) = \frac{1}{N} \sum_{j=1}^N K(x, x_j)$$

Kita akan mengasumsikan Gaussian berbentuk bola dengan deviasi standar w sepanjang setiap sumbu:

$$K(x, x_j) = \frac{1}{(w^2 \sqrt{2\pi})^d} e^{-\frac{D(x, x_j)^2}{2w^2}}$$

dengan d adalah jumlah dimensi dalam x dan D adalah fungsi jarak Euclidean. Kita masih menghadapi masalah dalam memilih nilai yang sesuai untuk lebar kernel w ; Gambar 2.9 menunjukkan nilai-nilai yang terlalu kecil, tepat, dan terlalu besar. Nilai w yang baik dapat dipilih dengan menggunakan validasi silang.

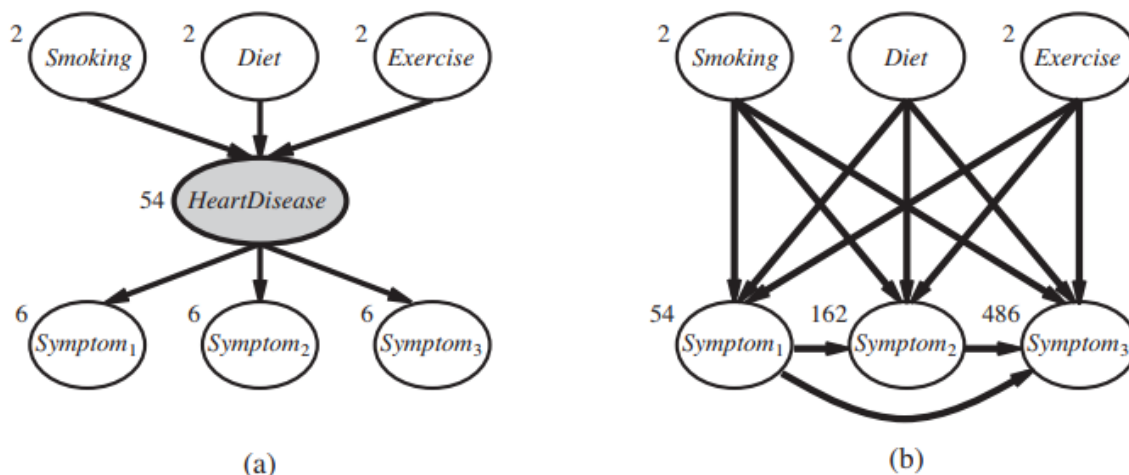
2.3 BELAJAR DENGAN VARIABEL TERSEMBUNYI ALGORITMA EM

Bagian sebelumnya membahas kasus yang sepenuhnya dapat diamati. Banyak masalah dunia nyata memiliki variabel tersembunyi (kadang-kadang disebut variabel laten), yang tidak dapat diamati dalam data yang tersedia untuk pembelajaran. Misalnya, catatan medis sering kali mencakup gejala yang diamati, diagnosis dokter, perawatan yang diterapkan, dan mungkin hasil perawatan, tetapi jarang mengandung pengamatan langsung dari penyakit itu sendiri! (Perhatikan bahwa diagnosis bukanlah penyakit; itu adalah konsekuensi kausal dari gejala yang diamati, yang pada gilirannya disebabkan oleh penyakit.)

Seseorang mungkin bertanya, "Jika penyakitnya tidak diamati, mengapa tidak membangun model tanpanya?" Jawabannya muncul pada Gambar 2.10, yang menunjukkan model diagnostik kecil dan fiktif untuk penyakit jantung. Ada tiga faktor predisposisi yang dapat diamati dan tiga gejala yang dapat diamati (yang terlalu menyedihkan untuk disebutkan). Asumsikan bahwa setiap variabel memiliki tiga nilai yang mungkin (misalnya, tidak ada, sedang, dan parah). Menghapus variabel tersembunyi dari jaringan dalam (a) menghasilkan jaringan dalam (b); Jumlah total parameter meningkat dari 78 menjadi 708. Dengan demikian, variabel laten dapat secara drastis mengurangi jumlah parameter yang diperlukan untuk menentukan jaringan Bayesian. Hal ini, pada gilirannya, dapat secara drastis mengurangi jumlah data yang diperlukan untuk mempelajari parameter.

Variabel tersembunyi memang penting, tetapi mereka memang mempersulit masalah pembelajaran. Pada Gambar 2.10(a), misalnya, tidak jelas bagaimana mempelajari distribusi kondisional untuk HeartDisease, mengingat induknya, karena kita tidak mengetahui nilai HeartDisease dalam setiap kasus; masalah yang sama muncul dalam mempelajari distribusi untuk gejala. Bagian ini menjelaskan algoritma yang disebut ekspektasi-maksimisasi, atau EM, yang memecahkan masalah ini dengan cara yang sangat umum. Kami akan menunjukkan tiga

contoh dan kemudian memberikan deskripsi umum. Algoritma ini tampak seperti sihir pada awalnya, tetapi setelah intuisi dikembangkan, seseorang dapat menemukan aplikasi untuk EM dalam berbagai macam masalah pembelajaran.



Gambar 2.10 (a) Jaringan diagnostik sederhana untuk penyakit jantung, yang diasumsikan sebagai variabel tersembunyi. Setiap variabel memiliki tiga kemungkinan nilai dan diberi label dengan jumlah parameter independen dalam distribusi kondisionalnya; jumlah totalnya adalah 78. (b) Jaringan ekuivalen dengan HeartDisease dihapus. Perhatikan bahwa variabel gejala tidak lagi independen secara kondisional mengingat induknya. Jaringan ini membutuhkan 708 parameter.

Pengelompokan Tanpa Supervisi: Mempelajari Campuran Gaussian

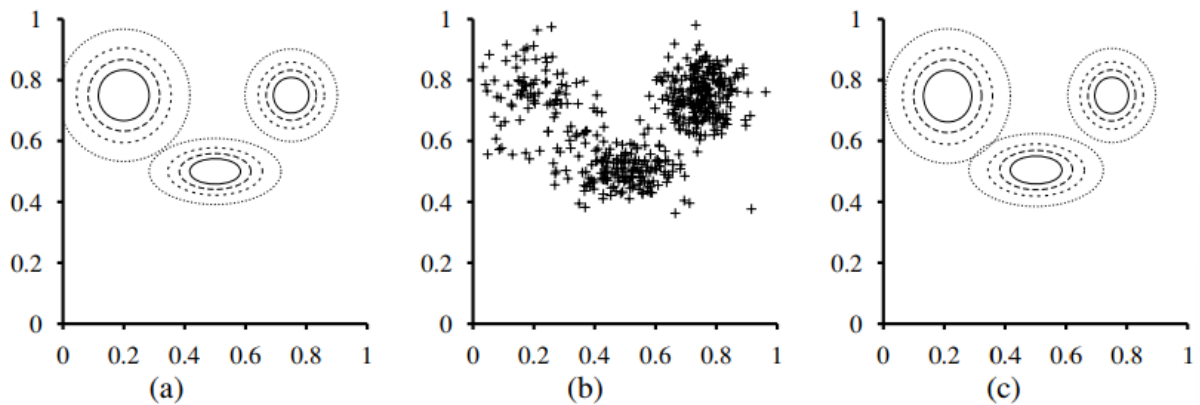
Pengelompokan tanpa supervisi adalah masalah dalam membedakan beberapa kategori dalam sekumpulan objek. Masalah ini disebut tanpa supervisi karena label kategori tidak diberikan. Misalnya, misalkan kita merekam spektrum seratus ribu bintang; adakah jenis bintang yang berbeda yang terungkap oleh spektrum tersebut, dan, jika ya, berapa banyak jenisnya dan apa karakteristiknya? Kita semua akrab dengan istilah seperti "raksasa merah" dan "katai putih", tetapi bintang-bintang tidak memiliki label ini di topi mereka—para astronom harus melakukan pengelompokan tanpa supervisi untuk mengidentifikasi kategori-kategori ini. Contoh lain termasuk identifikasi spesies, genera, ordo, dan sebagainya dalam taksonomi Linnæan dan penciptaan jenis alami untuk objek biasa.

Pengelompokan tanpa supervisi dimulai dengan data. Gambar 2.11(b) menunjukkan 500 titik data, yang masing-masing menentukan nilai dari dua atribut kontinu. Titik-titik data mungkin bersesuaian dengan bintang, dan atribut-atributnya mungkin bersesuaian dengan intensitas spektral pada dua frekuensi tertentu. Selanjutnya, kita perlu memahami jenis distribusi probabilitas yang mungkin menghasilkan data tersebut. Pengelompokan mengasumsikan bahwa data dihasilkan dari distribusi campuran, P . Distribusi tersebut memiliki k komponen, yang masing-masing merupakan distribusi tersendiri. Titik data dihasilkan dengan terlebih dahulu memilih sebuah komponen dan kemudian menghasilkan

sampel dari komponen tersebut. Misalkan variabel acak C menyatakan komponen tersebut, dengan nilai $1, \dots, k$; maka distribusi campuran diberikan oleh

$$P(x) = \sum_{i=1}^k P(C = i) P(x|C = i)$$

di mana x merujuk pada nilai atribut untuk suatu titik data. Untuk data kontinu, pilihan alami untuk distribusi komponen adalah Gaussian multivariat, yang menghasilkan apa yang disebut keluarga distribusi campuran Gaussian. Parameter campuran Gaussian adalah $w_i = P(C = i)$ (bobot setiap komponen), μ_i (rata-rata setiap komponen), dan Σ_i (kovarians setiap komponen). Gambar 2.11(a) menunjukkan campuran tiga Gaussian; campuran ini sebenarnya merupakan sumber data pada (b) sekaligus model yang ditunjukkan pada Gambar 2.7(a)



Gambar 2.11 (a) Model campuran Gaussian dengan tiga komponen; bobot (kiri ke kanan) adalah 0,2, 0,3, dan 0,5. (b) 500 titik data diambil sampelnya dari model pada (a). (c) Model yang direkonstruksi oleh EM dari data pada (b).

Masalah pengelompokan tanpa pengawasan, dengan demikian, adalah memulihkan model campuran seperti yang ada pada Gambar 2.11(a) dari data mentah seperti yang ada pada Gambar 2.11(b). Jelas, jika kita tahu komponen mana yang menghasilkan setiap titik data, maka akan mudah untuk memulihkan komponen Gaussian: kita bisa memilih semua titik data dari komponen yang diberikan dan kemudian menerapkan (versi multivariat dari) Persamaan (2.4) untuk menyesuaikan parameter Gaussian ke satu set data. Di sisi lain, jika kita tahu parameter setiap komponen, maka kita bisa, setidaknya dalam arti probabilistik, menetapkan setiap titik data ke suatu komponen. Masalahnya adalah kita tidak tahu penugasan maupun parameternya.

Ide dasar EM dalam konteks ini adalah untuk berpura-pura bahwa kita tahu parameter model dan kemudian menyimpulkan probabilitas bahwa setiap titik data termasuk dalam setiap komponen. Setelah itu, kita menyesuaikan kembali komponen-komponen tersebut ke data, di mana setiap komponen disesuaikan dengan seluruh set data dengan setiap titik diberi bobot oleh probabilitas bahwa titik tersebut termasuk dalam komponen tersebut. Proses ini

berulang hingga konvergensi. Pada dasarnya, kami "melengkapi" data dengan menyimpulkan distribusi probabilitas atas variabel tersembunyi komponen yang menjadi tempat setiap titik data berada berdasarkan model saat ini. Untuk campuran Gaussian, kami menginisialisasi parameter model campuran secara acak, lalu mengulangi dua langkah berikut:

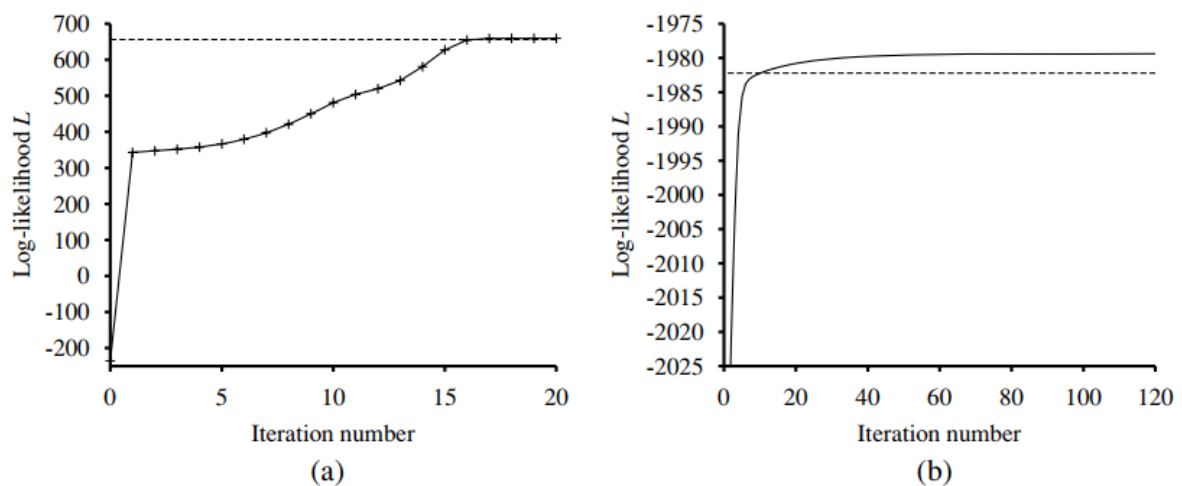
1. **Langkah-E (E-Step):** Hitung probabilitas $p_{ij} = P(C = i|x_j)$, probabilitas bahwa datum x_j dihasilkan oleh komponen i . Berdasarkan aturan Bayes, kami memperoleh $p_{ij} = \alpha P(x_j|C = i)P(C = i)$. Suku $P(x_j|C = i)$ hanyalah probabilitas pada x_j dari Gaussian ke- i , dan suku $P(C = i)$ hanyalah parameter bobot untuk Gaussian ke- i . Tentukan $n_i = \sum_j p_{ij}$, jumlah efektif titik data yang saat ini ditetapkan untuk komponen i .
2. **Langkah-M (M-Step):** Hitung rerata, kovarians, dan bobot komponen baru menggunakan langkah-langkah berikut secara berurutan:

$$\begin{aligned}\mu_i &\leftarrow \sum_j p_{ij} x_j / n_i \\ \Sigma_i &\leftarrow \sum_j p_{ij} (x_j - \mu_i)(x_j - \mu_i)^\top / n_i \\ w_i &\leftarrow n_i / N\end{aligned}$$

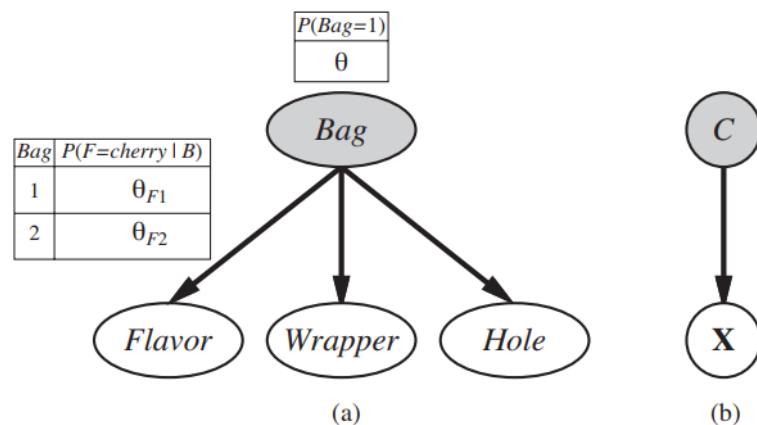
di mana N adalah jumlah total titik data. Langkah-E, atau langkah ekspektasi, dapat dipandang sebagai penghitungan nilai ekspektasi p_{ij} dari variabel indikator tersembunyi Z_{ij} , di mana Z_{ij} bernilai 1 jika datum x_j dihasilkan oleh komponen ke- i dan 0 jika tidak. Langkah-M, atau langkah maksimisasi, menemukan nilai-nilai baru dari parameter yang memaksimalkan log likelihood data, dengan nilai ekspektasi variabel indikator tersembunyi.

Model akhir yang dipelajari EM ketika diterapkan pada data pada Gambar 2.11(a) ditunjukkan pada Gambar 2.11(c); model ini hampir tidak dapat dibedakan dari model asli tempat data dihasilkan. Gambar 2.12(a) memplot log likelihood data menurut model saat ini seiring perkembangan EM. Ada dua hal yang perlu diperhatikan. Pertama, log likelihood untuk model akhir yang dipelajari sedikit melebihi model asli, tempat data dihasilkan. Hal ini mungkin tampak mengejutkan, tetapi hal ini mencerminkan fakta bahwa data dihasilkan secara acak dan mungkin tidak memberikan gambaran yang tepat dari model yang mendasarinya. Poin kedua adalah bahwa EM meningkatkan logaritma data pada setiap iterasi.

Fakta ini dapat dibuktikan secara umum. Lebih lanjut, dalam kondisi tertentu (yang berlaku dalam sebagian besar kasus), EM dapat dibuktikan mencapai maksimum lokal dalam hal kemungkinan. (Dalam kasus yang jarang terjadi, EM dapat mencapai titik pelana atau bahkan minimum lokal.) Dalam hal ini, EM menyerupai algoritma pendakian bukit berbasis gradien, tetapi perlu diperhatikan bahwa algoritma ini tidak memiliki parameter "ukuran langkah".



Gambar 2.12 Grafik yang menunjukkan log likelihood data, L , sebagai fungsi iterasi EM. Garis horizontal menunjukkan log likelihood menurut model sebenarnya. (a) Grafik untuk model campuran Gaussian pada Gambar 2.11. (b) Grafik untuk jaringan Bayesian pada Gambar 2.13(a).



Gambar 2.13 (a) Model campuran untuk permen. Proporsi rasa, pembungkus, dan keberadaan lubang yang berbeda bergantung pada jenis kantong, yang tidak teramati. (b) Jaringan Bayesian untuk campuran Gaussian. Rata-rata dan kovariansi variabel teramati X bergantung pada komponen C .

Segala sesuatunya tidak selalu berjalan sebaik yang ditunjukkan Gambar 2.12(a). Misalnya, bisa saja satu komponen Gaussian menyusut sehingga hanya mencakup satu titik data. Kemudian, variansnya akan menjadi nol dan kemungkinannya akan mencapai tak terhingga! Masalah lainnya adalah dua komponen dapat "bergabung", memperoleh rata-rata dan varians yang identik, serta berbagi titik data mereka. Maksima lokal yang terdegenerasi seperti ini merupakan masalah serius, terutama dalam dimensi tinggi. Salah satu solusinya adalah menempatkan prior pada parameter model dan menerapkan versi MAP dari EM. Solusi lainnya adalah memulai ulang komponen dengan parameter acak baru jika ukurannya terlalu kecil atau terlalu dekat dengan komponen lain. Inisialisasi yang masuk akal juga membantu.

Mempelajari jaringan Bayesian dengan variabel tersembunyi

Untuk mempelajari jaringan Bayesian dengan variabel tersembunyi, kami menerapkan wawasan yang sama yang berhasil untuk campuran Gaussian. Gambar 2.13 merepresentasikan situasi di mana terdapat dua kantong permen yang telah dicampur. Permen dijelaskan oleh tiga fitur: selain Rasa dan Pembungkus, beberapa permen memiliki Lubang di tengah dan beberapa tidak. Distribusi permen di setiap kantong dijelaskan oleh model Bayes naif: fitur-fiturnya independen, mengingat kantongnya, tetapi distribusi probabilitas bersyarat untuk setiap fitur bergantung pada kantongnya.

Parameternya adalah sebagai berikut: θ adalah probabilitas sebelumnya bahwa permen berasal dari Kantong 1; θ_{F1} dan θ_{F2} adalah probabilitas bahwa rasanya adalah ceri, mengingat permen tersebut masing-masing berasal dari Kantong 1 atau Kantong 2; θ_{W1} dan θ_{W2} memberikan probabilitas bahwa pembungkusnya berwarna merah; dan θ_{H1} dan θ_{H2} memberikan probabilitas bahwa permen tersebut memiliki lubang.

Perhatikan bahwa model keseluruhan adalah model campuran. (Faktanya, kita juga dapat memodelkan campuran Gaussian sebagai jaringan Bayesian, seperti yang ditunjukkan pada Gambar 2.13(b).) Pada gambar tersebut, kantong merupakan variabel tersembunyi karena, setelah permen dicampur, kita tidak lagi tahu dari kantong mana masing-masing permen berasal. Dalam kasus seperti itu, dapatkah kita memulihkan deskripsi kedua kantong dengan mengamati permen dari campuran tersebut? Mari kita kerjakan melalui iterasi EM untuk masalah ini. Pertama, mari kita lihat datanya. Kita menghasilkan 1000 sampel dari model yang parameter sebenarnya adalah sebagai berikut:

$$\theta = 0.5, \theta_{F1} = \theta_{W1} = \theta_{H1} = 0.8, \theta_{F2} = \theta_{W2} = \theta_{H2} = 0.3. \quad (2.7)$$

Artinya, permen-permen tersebut kemungkinan besar berasal dari kedua kantong; kantong pertama sebagian besar berisi ceri dengan bungkus merah dan berlubang; kantong kedua sebagian besar berisi jeruk nipis dengan bungkus hijau dan tanpa lubang. Jumlah delapan jenis permen yang mungkin adalah sebagai berikut:

Warna (W)	W= merah		W= hijau	
	H=1	H=0	H=1	H=0
Buah = Cherry	273	93	104	90
Buah = Lime	79	100	94	167

Kita mulai dengan menginisialisasi parameter. Untuk menyederhanakan numerik, kita pilih 5 secara acak

$$\theta^{(0)} = 0.6, \theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6, \theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4. \quad (2.8)$$

Pertama, mari kita bahas parameter θ . Dalam kasus yang sepenuhnya teramati, kita akan mengestimasi langsung dari jumlah permen yang teramati dari kantong 1 dan 2. Karena

kantong tersebut merupakan variabel tersembunyi, kita hitung jumlah yang diharapkan. Jumlah yang diharapkan $\hat{N}(Bag = 1)$ adalah jumlah, atas semua permen, dari probabilitas permen tersebut berasal dari kantong 1:

$$\theta^{(1)} = \hat{N}(Bag = 1)/N = \sum_{j=1}^N P(Bag = 1 | flavor_j, wrapper_j, holes_j)/N.$$

Probabilitas ini dapat dihitung dengan algoritma inferensi apa pun untuk jaringan Bayesian. Untuk model Bayes naif seperti dalam contoh kita, kita dapat melakukan inferensi "secara manual", menggunakan aturan Bayes dan menerapkan independensi bersyarat:

$$\theta^{(1)} = \frac{1}{N} \sum_{j=1}^N \frac{P(flavor_j | Bag = 1)P(wrapper_j | Bag = 1)P(holes_j | Bag = 1)P(Bag = 1)}{\sum_i P(flavor_j | Bag = i)P(wrapper_j | Bag = i)P(holes_j | Bag = i)P(Bag = i)}.$$

Dengan menerapkan rumus ini pada, katakanlah, 273 permen ceri yang dibungkus merah dan berlubang, kita mendapatkan kontribusi sebesar:

$$\frac{273}{1000} \cdot \frac{\theta_{F1}^{(0)} \theta_{W2}^{(0)} \theta_{H1}^{(0)} \theta^{(0)}}{\theta_{F1}^{(0)} \theta_{W2}^{(0)} \theta_{H1}^{(0)} \theta^{(0)} + \theta_{F2}^{(0)} \theta_{W2}^{(0)} \theta_{H2}^{(0)} \theta^{(0)}} \approx 0.22797$$

Melanjutkan tujuh jenis permen lainnya dalam tabel jumlah, kita memperoleh $\theta^{(1)} = 0,6124$. Sekarang mari kita pertimbangkan parameter lainnya, seperti θ_{F1} . Dalam kasus yang sepenuhnya teramati, kita akan memperkirakan ini langsung dari jumlah permen ceri dan jeruk nipis yang teramati dari kantong 1.

Jumlah permen ceri yang diharapkan dari kantong 1 diberikan oleh

$$\sum_{j: Flavor_j = Cherry} P(Bag = 1 | Flavor_j = cherry, wrapper_j, holes_j)$$

Sekali lagi, probabilitas ini dapat dihitung dengan algoritma jaring Bayes apa pun. Dengan menyelesaikan proses ini, kita memperoleh nilai baru dari semua parameter:

$$\begin{aligned} \theta^{(1)} &= 0.6124, \theta_{F1}^{(1)} = 0.6684, \theta_{W1}^{(1)} = 0.6483, \theta_{H1}^{(1)} = 0.6558, \\ \theta_{F2}^{(1)} &= 0.3887, \theta_{W2}^{(1)} = 0.3817, \theta_{H2}^{(1)} = 0.3827. \end{aligned} \quad (2.9)$$

Log likelihood data meningkat dari sekitar -2044 pada awalnya menjadi sekitar -2021 setelah iterasi pertama, seperti yang ditunjukkan pada Gambar 2.12(b). Artinya, pembaruan meningkatkan likelihood itu sendiri dengan faktor sekitar $e^{23} \approx 10^{10}$. Pada iterasi kesepuluh, model yang dipelajari lebih cocok daripada model asli ($L = -1982.214$). Setelah itu,

kemajuan menjadi sangat lambat. Hal ini tidak jarang terjadi pada EM, dan banyak sistem praktis menggabungkan EM dengan algoritma berbasis gradien seperti Newton–Raphson untuk fase pembelajaran terakhir.

Pelajaran umum dari contoh ini adalah bahwa pembaruan parameter untuk pembelajaran jaringan Bayesian dengan variabel tersembunyi tersedia langsung dari hasil inferensi pada setiap contoh. Selain itu, hanya probabilitas posterior lokal yang diperlukan untuk setiap parameter. Di sini, "lokal" berarti bahwa CPT untuk setiap variabel X_i dapat dipelajari dari probabilitas posterior yang hanya melibatkan X_i dan induknya U_i . Dengan mendefinisikan θ_{ijk} sebagai parameter CPT $P(X_i = x_{ij} | U_i = u_{ik})$, pembaruan diberikan oleh hitungan yang diharapkan dinormalisasi sebagai berikut:

$$\theta_{ijk} \leftarrow \hat{N}(X_i = x_{ij}, U_i = u_{ik}) / \hat{N}(U_i = u_{ik})$$

Jumlah yang diharapkan diperoleh dengan menjumlahkan contoh-contoh, menghitung probabilitas $P(X_i = x_{ij}, U_i = u_{ik})$ untuk masing-masing dengan menggunakan algoritma inferensi jaring Bayes apa pun. Untuk algoritma yang tepat—termasuk eliminasi variabel—semua probabilitas ini dapat diperoleh secara langsung sebagai produk sampingan dari inferensi standar, tanpa memerlukan komputasi tambahan khusus untuk pembelajaran. Selain itu, informasi yang dibutuhkan untuk pembelajaran tersedia secara lokal untuk setiap parameter.

Mempelajari Model Markov Tersembunyi

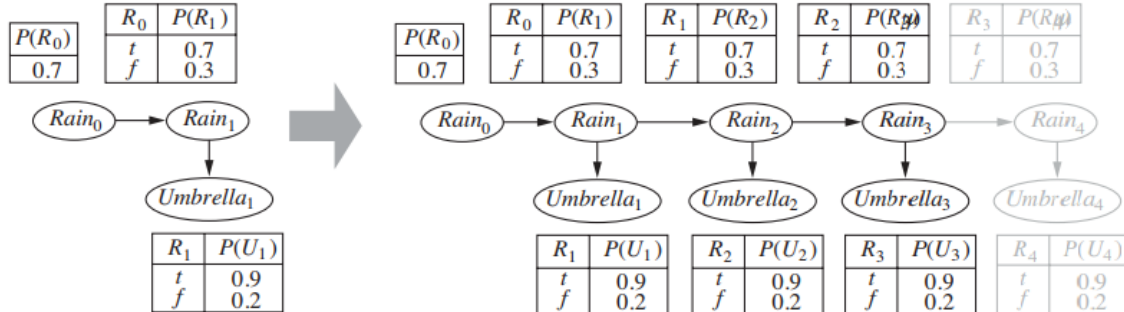
Aplikasi terakhir EM kita melibatkan pembelajaran probabilitas transisi dalam model Markov tersembunyi (HMM). Model Markov tersembunyi dapat direpresentasikan oleh jaring Bayes dinamis dengan satu variabel keadaan diskrit, seperti yang diilustrasikan pada Gambar 2.14. Setiap titik data terdiri dari sekuens observasi dengan panjang terhingga, sehingga permasalahannya adalah mempelajari probabilitas transisi dari sekumpulan sekuens observasi (atau hanya dari satu sekuens panjang).

Kita telah mengetahui cara mempelajari jaring Bayes, tetapi ada satu komplikasi: dalam jaring Bayes, setiap parameter bersifat unik; di sisi lain, dalam model Markov tersembunyi, probabilitas transisi individual dari keadaan i ke keadaan j pada waktu t , $\theta_{ijt} = P(X_{t+1} = j | X_t = i)$, berulang sepanjang waktu—yaitu, $\theta_{ijt} = \theta_{ij}$ untuk semua t . Untuk memperkirakan probabilitas transisi dari keadaan i ke keadaan j , kita cukup menghitung proporsi ekspektasi frekuensi sistem mengalami transisi ke keadaan j ketika berada dalam keadaan i :

$$\theta_{ij} \leftarrow \sum_t \hat{N}(X_{t+1} = j, X_t = i) / \sum_t \hat{N}(X_t = i).$$

Jumlah harapan dihitung dengan algoritma inferensi HMM. Algoritma maju-mundur dapat dimodifikasi dengan sangat mudah untuk menghitung probabilitas yang diperlukan. Satu poin penting adalah bahwa probabilitas yang diperlukan diperoleh dengan penghalusan, alih-alih

penyaringan; artinya, kita perlu memperhatikan bukti selanjutnya dalam memperkirakan probabilitas terjadinya transisi tertentu. Bukti dalam kasus pembunuhan biasanya diperoleh setelah kejahatan (yaitu, transisi dari keadaan i ke keadaan j) terjadi.



Gambar 2.14 Jaringan Bayesian dinamis yang tidak digulirkan yang merepresentasikan model Markov tersembunyi.

Bentuk umum algoritma EM

Kita telah melihat beberapa contoh algoritma EM. Masing-masing melibatkan penghitungan nilai harapan variabel tersembunyi untuk setiap contoh dan kemudian penghitungan ulang parameternya, menggunakan nilai harapan seolah-olah nilai tersebut adalah nilai teramati. Misalkan $\mathbf{x}$ adalah semua nilai teramati dalam semua contoh, misalkan $\mathbf{Z}$ menyatakan semua variabel tersembunyi untuk semua contoh, dan misalkan $\boldsymbol{\theta}$ adalah semua parameter untuk model probabilitas. Maka algoritma EM adalah

$$\boldsymbol{\theta}^{(i+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\mathbf{z}} P(\mathbf{Z} = \mathbf{z} | \mathbf{x}, \boldsymbol{\theta}^{(i)}) L(\mathbf{x}, \mathbf{Z} = \mathbf{z} | \boldsymbol{\theta})$$

Persamaan ini merupakan inti dari algoritma EM. Langkah-E adalah komputasi penjumlahan, yang merupakan ekspektasi log likelihood dari data "lengkap" terhadap distribusi $P(\mathbf{Z} = \mathbf{z} | \mathbf{x}, \boldsymbol{\theta}^{(i)})$, yang merupakan posterior atas variabel tersembunyi, berdasarkan data tersebut. Langkah-M adalah maksimalisasi log likelihood yang diharapkan ini terhadap parameter. Untuk campuran Gaussian, variabel tersembunyi adalah Z_{ij} , dengan Z_{ij} adalah 1 jika contoh j dihasilkan oleh komponen i . Untuk jaring Bayes, Z_{ij} adalah nilai variabel tak teramati X_i dalam contoh j . Untuk HMM, Z_{jt} adalah keadaan deret dalam contoh j pada waktu t . Berangkat dari bentuk umum, dimungkinkan untuk menurunkan algoritma EM untuk aplikasi spesifik setelah variabel tersembunyi yang sesuai telah diidentifikasi. Begitu kita memahami gagasan umum EM, akan mudah untuk menurunkan berbagai varian dan penyempurnaan. Misalnya, dalam banyak kasus, langkah-E—perhitungan posterior atas variabel tersembunyi tidak mudah diatasi, seperti dalam jaring Bayes yang besar. Ternyata, seseorang dapat menggunakan langkah-E perkiraan dan tetap memperoleh algoritma pembelajaran yang efektif.

Dengan algoritma pengambilan sampel seperti MCMC, proses pembelajaran sangat intuitif: setiap keadaan (konfigurasi variabel tersembunyi dan yang diamati) yang dikunjungi oleh MCMC diperlakukan persis seolah-olah merupakan pengamatan lengkap. Dengan

demikian, parameter dapat diperbarui langsung setelah setiap transisi MCMC. Bentuk lain dari inferensi perkiraan, seperti metode variasional dan loopy, juga terbukti efektif untuk mempelajari jaringan yang sangat besar.

Mempelajari struktur jaringan Bayes dengan variabel tersembunyi

Pada Bagian sebelumnya, kami membahas masalah mempelajari struktur jaringan Bayes dengan data lengkap. Ketika variabel yang tidak teramati mungkin memengaruhi data yang teramati, hal-hal menjadi lebih rumit. Dalam kasus yang paling sederhana, seorang pakar manusia mungkin memberi tahu algoritma pembelajaran bahwa terdapat variabel tersembunyi tertentu, dan membiarkan algoritma tersebut menemukan tempatnya dalam struktur jaringan. Misalnya, suatu algoritma mungkin mencoba mempelajari struktur yang ditunjukkan pada Gambar 2.10(a) dengan informasi bahwa *HeartDisease* (variabel bernilai tiga) harus dimasukkan dalam model. Seperti dalam kasus data lengkap, keseluruhan algoritma memiliki loop luar yang menelusuri struktur dan loop dalam yang sesuai dengan parameter jaringan berdasarkan struktur tersebut.

Jika algoritma pembelajaran tidak diberi tahu variabel tersembunyi mana yang ada, maka ada dua pilihan: berpura-pura bahwa data tersebut benar-benar lengkap yang mungkin memaksa algoritma untuk mempelajari model yang intensif parameter seperti pada Gambar 2.10(b) atau menciptakan variabel tersembunyi baru untuk menyederhanakan model. Pendekatan terakhir dapat diimplementasikan dengan memasukkan pilihan modifikasi baru dalam pencarian struktur: selain memodifikasi tautan, algoritma dapat menambah atau menghapus variabel tersembunyi atau mengubah aritasnya.

Tentu saja, algoritma tidak akan tahu bahwa variabel baru yang diciptakannya bernama *HeartDisease*; juga tidak akan memiliki nama yang bermakna untuk nilai-nilai tersebut. Untungnya, variabel tersembunyi yang baru diciptakan biasanya akan terhubung ke variabel yang sudah ada sebelumnya, sehingga seorang pakar manusia sering kali dapat memeriksa distribusi kondisional lokal yang melibatkan variabel baru dan memastikan maknanya. Seperti dalam kasus data lengkap, pembelajaran struktur kemungkinan maksimum murni akan menghasilkan jaringan yang terhubung sepenuhnya (terlebih lagi, jaringan tanpa variabel tersembunyi), sehingga diperlukan beberapa bentuk penalti kompleksitas. Kita juga dapat menerapkan MCMC untuk mengambil sampel dari banyak kemungkinan struktur jaringan, sehingga mengaproksimasi pembelajaran Bayesian. Misalnya, kita dapat mempelajari campuran Gaussian dengan jumlah komponen yang tidak diketahui dengan mengambil sampel dari jumlah tersebut; distribusi posterior aproksimasi untuk jumlah Gaussian diberikan oleh frekuensi pengambilan sampel dari proses MCMC.

Untuk kasus data lengkap, loop dalam untuk mempelajari parameter sangat cepat hanya masalah mengekstraksi frekuensi kondisional dari kumpulan data. Ketika terdapat variabel tersembunyi, loop dalam dapat melibatkan banyak iterasi EM atau algoritma berbasis gradien, dan setiap iterasi melibatkan kalkulasi posterior dalam jaring Bayes, yang merupakan masalah NP-hard. Hingga saat ini, pendekatan ini terbukti tidak praktis untuk mempelajari model yang kompleks. Salah satu kemungkinan peningkatan adalah apa yang disebut algoritma EM struktural, yang beroperasi dengan cara yang hampir sama seperti EM biasa

(parametrik) kecuali bahwa algoritma tersebut dapat memperbarui struktur serta parameternya.

Sebagaimana EM biasa menggunakan parameter saat ini untuk menghitung jumlah ekspektasi pada langkah E dan kemudian menerapkan jumlah tersebut pada langkah M untuk memilih parameter baru, EM struktural menggunakan struktur saat ini untuk menghitung jumlah ekspektasi dan kemudian menerapkan jumlah tersebut pada langkah M untuk mengevaluasi kemungkinan munculnya struktur baru yang potensial. (Hal ini berbeda dengan metode lingkaran luar/lingkaran dalam, yang menghitung jumlah ekspektasi baru untuk setiap struktur potensial.) Dengan cara ini, EM struktural dapat membuat beberapa perubahan struktural pada jaringan tanpa perlu menghitung ulang jumlah ekspektasi, dan mampu mempelajari struktur jaringan Bayes yang non-trivial. Meskipun demikian, masih banyak pekerjaan yang harus dilakukan sebelum kita dapat mengatakan bahwa masalah pembelajaran struktur telah terpecahkan.

2.4 RINGKASAN

Metode pembelajaran statistik berkisar dari perhitungan rata-rata sederhana hingga konstruksi model kompleks seperti jaringan Bayesian. Metode ini memiliki aplikasi di seluruh ilmu komputer, teknik, biologi komputasional, ilmu saraf, psikologi, dan fisika. Bab ini telah menyajikan beberapa gagasan dasar dan memberikan gambaran tentang dasar-dasar matematika. Poin-poin utamanya adalah sebagai berikut:

- ❖ Metode pembelajaran Bayesian merumuskan pembelajaran sebagai bentuk inferensi probabilistik, menggunakan observasi untuk memperbarui distribusi prior atas hipotesis. Pendekatan ini menyediakan cara yang baik untuk mengimplementasikan pisau cukur Ockham, tetapi dengan cepat menjadi rumit untuk ruang hipotesis yang kompleks.
- ❖ Pembelajaran a posteriori maksimum (MAP) memilih satu hipotesis yang paling mungkin berdasarkan data. Hipotesis prior masih digunakan dan metode ini seringkali lebih mudah dipahami daripada pembelajaran Bayesian penuh.
- ❖ Pembelajaran kemungkinan maksimum hanya memilih hipotesis yang memaksimalkan kemungkinan data; ini setara dengan pembelajaran MAP dengan prior seragam. Dalam kasus sederhana seperti regresi linier dan jaringan Bayesian yang sepenuhnya teramati, solusi kemungkinan maksimum dapat ditemukan dengan mudah dalam bentuk tertutup. Pembelajaran Bayes Naif merupakan teknik yang sangat efektif dan berskala baik.
- ❖ Ketika beberapa variabel tersembunyi, solusi kemungkinan maksimum lokal dapat ditemukan menggunakan algoritma EM. Aplikasinya meliputi pengelompokan menggunakan campuran Gaussian, pembelajaran jaringan Bayesian, dan pembelajaran model Markov tersembunyi.
- ❖ Mempelajari struktur jaringan Bayesian merupakan contoh pemilihan model. Hal ini biasanya melibatkan pencarian diskrit dalam ruang struktur. Beberapa metode diperlukan untuk membandingkan kompleksitas model dengan derajat kesesuaian.

- ❖ Model nonparametrik merepresentasikan suatu distribusi menggunakan kumpulan titik data. Dengan demikian, jumlah parameter bertambah seiring dengan set pelatihan. Metode tetangga terdekat melihat contoh-contoh yang paling dekat dengan titik yang dimaksud, sedangkan metode kernel membentuk kombinasi semua contoh yang terbobot jaraknya.

Pembelajaran statistik terus menjadi bidang penelitian yang sangat aktif. Kemajuan pesat telah dicapai baik dalam teori maupun praktik, hingga memungkinkan untuk mempelajari hampir semua model yang memungkinkan inferensi eksak maupun aproksimasi.

BAB 3

PEMBALIAN PENGUATAN

3.1 PENDAHULUAN

Bab sebelumnya membahas metode yang mempelajari fungsi, teori logika, dan model probabilitas dari contoh. Dalam bab ini, kami akan mempelajari bagaimana agen dapat mempelajari apa yang harus dilakukan tanpa adanya contoh berlabel tentang apa yang harus dilakukan. Pertimbangkan, misalnya, masalah belajar bermain catur. Agen pembelajaran yang diawasi perlu diberi tahu langkah yang benar untuk setiap posisi yang dihadapinya, tetapi umpan balik semacam itu jarang tersedia. Tanpa umpan balik dari guru, agen dapat mempelajari model transisi untuk langkahnya sendiri dan mungkin dapat belajar memprediksi langkah lawan, tetapi tanpa umpan balik tentang apa yang baik dan apa yang buruk, agen tidak akan memiliki dasar untuk memutuskan langkah mana yang akan diambil.

Agen perlu tahu bahwa sesuatu yang baik telah terjadi ketika (secara tidak sengaja) melakukan skakmat terhadap lawan, dan bahwa sesuatu yang buruk telah terjadi ketika skakmat atau sebaliknya, jika permainannya adalah catur bunuh diri. Umpan balik semacam ini disebut imbalan, atau penguatan. Dalam permainan seperti catur, penguatan hanya diterima di akhir permainan. Di lingkungan lain, imbalan datang lebih sering. Dalam pingpong, setiap poin yang dicetak dapat dianggap sebagai imbalan; ketika belajar merangkak, setiap gerakan maju merupakan sebuah pencapaian. Kerangka kerja kami untuk agen menganggap imbalan sebagai bagian dari persepsi masukan, tetapi agen harus "terprogram" untuk mengenali bagian itu sebagai imbalan, bukan sekadar masukan sensorik lainnya. Dengan demikian, hewan tampaknya terprogram untuk mengenali rasa sakit dan lapar sebagai imbalan negatif dan kesenangan serta asupan makanan sebagai imbalan positif. Penguatan telah dipelajari dengan cermat oleh para psikolog hewan selama lebih dari 60 tahun.

Imbalan/Pembalihan berfungsi untuk mendefinisikan kebijakan optimal dalam proses keputusan Markov (MDP). Kebijakan optimal adalah kebijakan yang memaksimalkan total imbalan yang diharapkan. Tugas pembelajaran penguatan adalah menggunakan imbalan yang teramati untuk mempelajari kebijakan optimal (atau hampir optimal) bagi lingkungan. Sementara Model lingkungan yang lengkap dan mengetahui fungsi imbalan, di sini kita berasumsi tidak memiliki pengetahuan sebelumnya tentang keduanya. Bayangkan bermain permainan baru yang aturannya tidak Anda ketahui; setelah sekitar seratus langkah, lawan Anda mengumumkan, "Anda kalah." Singkatnya, inilah pembelajaran penguatan.

Dalam banyak domain yang kompleks, pembelajaran penguatan adalah satu-satunya cara yang layak untuk melatih program agar berkinerja pada tingkat tinggi. Misalnya, dalam permainan, sangat sulit bagi manusia untuk memberikan evaluasi yang akurat dan konsisten terhadap sejumlah besar posisi, yang diperlukan untuk melatih fungsi evaluasi langsung dari contoh. Sebaliknya, program dapat diberi tahu kapan ia menang atau kalah, dan dapat menggunakan informasi ini untuk mempelajari fungsi evaluasi yang memberikan estimasi probabilitas menang yang cukup akurat dari posisi mana pun.

Demikian pula, sangat sulit untuk memprogram agen untuk menerbangkan helikopter; namun, jika diberi imbalan negatif yang sesuai karena menabrak, bergoyang, atau menyimpang dari jalur yang ditentukan, agen dapat belajar terbang sendiri. Pembelajaran penguatan dapat dianggap mencakup seluruh AI: seorang agen ditempatkan di suatu lingkungan dan harus belajar berperilaku dengan sukses di dalamnya. Agar bab ini mudah dipahami, kita akan berfokus pada lingkungan sederhana dan desain agen sederhana.

Sebagian besar, kita akan mengasumsikan lingkungan yang sepenuhnya dapat diamati, sehingga keadaan saat ini diberikan oleh setiap persepsi. Di sisi lain, kita akan mengasumsikan bahwa agen tidak tahu bagaimana lingkungan tersebut bekerja atau apa yang dilakukan tindakannya, dan kita akan memperhitungkan hasil tindakan probabilistik. Dengan demikian, agen menghadapi proses keputusan Markov yang tidak diketahui. Kita akan mempertimbangkan tiga desain agen yang pertama kali diperkenalkan di Bab 2:

- ❖ Agen berbasis utilitas mempelajari fungsi utilitas pada status dan menggunakannya untuk memilih tindakan yang memaksimalkan utilitas hasil yang diharapkan.
- ❖ Agen Q-learning mempelajari fungsi utilitas tindakan, atau fungsi Q, yang memberikan utilitas yang diharapkan dari melakukan tindakan tertentu dalam status tertentu.
- ❖ Agen refleks mempelajari kebijakan yang memetakan langsung dari status ke tindakan.

Agen berbasis utilitas juga harus memiliki model lingkungan agar dapat mengambil keputusan, karena harus mengetahui status yang akan dituju oleh tindakannya. Misalnya, untuk memanfaatkan fungsi evaluasi backgammon, program backgammon harus mengetahui langkah hukumnya dan bagaimana langkah tersebut memengaruhi posisi papan. Hanya dengan cara ini ia dapat menerapkan fungsi utilitas pada status hasil. Di sisi lain, agen Q-learning dapat membandingkan utilitas yang diharapkan untuk pilihan yang tersedia tanpa perlu mengetahui hasilnya, sehingga tidak memerlukan model lingkungan. Di sisi lain, karena mereka tidak tahu ke mana tindakan mereka mengarah, agen Q-learning tidak dapat melihat ke depan; hal ini dapat secara serius membatasi kemampuan mereka untuk belajar, seperti yang akan kita lihat.

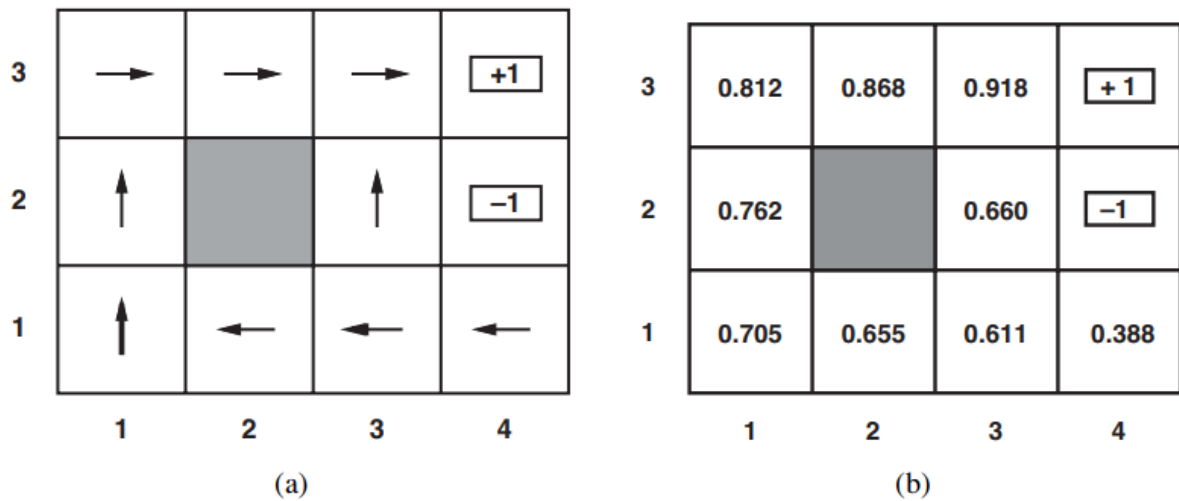
Kita mulai di Bagian 3.2 dengan pembelajaran pasif, di mana kebijakan agen telah ditetapkan dan tugasnya adalah mempelajari utilitas keadaan (atau pasangan keadaan-tindakan); ini juga dapat melibatkan pembelajaran model lingkungan. Bagian 3.3 membahas pembelajaran aktif, di mana agen juga harus mempelajari apa yang harus dilakukan. Isu utamanya adalah eksplorasi: seorang agen harus mengalami lingkungannya sebanyak mungkin untuk mempelajari bagaimana berperilaku di dalamnya. Bagian 3.4 membahas bagaimana seorang agen dapat menggunakan pembelajaran induktif untuk belajar lebih cepat dari pengalamannya. Bagian 3.5 membahas metode untuk mempelajari representasi kebijakan langsung pada agen refleks. Pemahaman tentang proses keputusan Markov sangat penting untuk bab ini.

3.2 PEMBELAJARAN PENGUATAN PASIF

Untuk menyederhanakannya, kita mulai dengan kasus agen pembelajaran pasif yang menggunakan representasi berbasis keadaan dalam lingkungan yang sepenuhnya dapat

diamati. Dalam pembelajaran pasif, kebijakan agen π bersifat tetap: dalam keadaan s , agen selalu mengeksekusi tindakan $\pi(s)$. Tujuannya hanyalah untuk mempelajari seberapa baik kebijakan tersebut yaitu, mempelajari fungsi utilitas $U^\pi(s)$. Kita akan menggunakan dunia 4×3 sebagai contoh yang diperkenalkan pada Bab 17.

Gambar 3.1 menunjukkan kebijakan untuk dunia tersebut dan utilitas yang sesuai. Jelas, tugas pembelajaran pasif serupa dengan tugas evaluasi kebijakan. Perbedaan utamanya adalah agen pembelajaran pasif tidak mengetahui model transisi $P(s' | s, a)$, yang menentukan probabilitas mencapai keadaan s' dari keadaan s setelah melakukan tindakan a ; juga tidak mengetahui fungsi imbalan $R(s)$, yang menentukan imbalan untuk setiap keadaan.



Gambar 3.1 (a) Kebijakan π untuk dunia 4×3 ; kebijakan ini kebetulan optimal dengan imbalan $R(s) = -0,04$ di negara-negara nonterminal dan tanpa diskonto. (b) Utilitas negara-negara di dunia 4×3 , dengan kebijakan π yang diberikan.

Agan menjalankan serangkaian uji coba di lingkungan menggunakan kebijakannya π . Dalam setiap uji coba, agen memulai dari keadaan (1,1) dan mengalami serangkaian transisi keadaan hingga mencapai salah satu keadaan terminal, (4,2) atau (4,3). Persepsinya memberikan informasi tentang keadaan saat ini dan imbalan yang diterima dalam keadaan tersebut. Uji coba yang umum mungkin terlihat seperti ini:

$(1,1)_{-0.04} \rightsquigarrow (1,2)_{-0.04} \rightsquigarrow (1,3)_{-0.04} \rightsquigarrow (1,2)_{-0.04} \rightsquigarrow (1,3)_{-0.04} \rightsquigarrow (2,3)_{-0.04} \rightsquigarrow (3,3)_{-0.04} \rightsquigarrow (4,3)_{+1}$
 $(1,1)_{-0.04} \rightsquigarrow (1,2)_{-0.04} \rightsquigarrow (1,3)_{-0.04} \rightsquigarrow (2,3)_{-0.04} \rightsquigarrow (3,3)_{-0.04} \rightsquigarrow (3,2)_{-0.04} \rightsquigarrow (3,3)_{-0.04} \rightsquigarrow (4,3)_{+1}$
 $(1,1)_{-0.04} \rightsquigarrow (2,1)_{-0.04} \rightsquigarrow (3,1)_{-0.04} \rightsquigarrow (3,2)_{-0.04} \rightsquigarrow (4,2)_{-1}$

Perhatikan bahwa setiap persepsi keadaan disubskripsikan dengan imbalan yang diterima. Tujuannya adalah menggunakan informasi tentang imbalan untuk mempelajari utilitas yang diharapkan $U^\pi(s)$ yang terkait dengan setiap keadaan nonterminal s . Utilitas didefinisikan sebagai jumlah imbalan (diskonto) yang diharapkan yang diperoleh jika kebijakan π diikuti.

$$U^\pi(s) = E \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right] \quad (3.1)$$

di mana $R(s)$ adalah imbalan untuk suatu keadaan, S_t (variabel acak) adalah keadaan yang dicapai pada waktu t ketika menjalankan kebijakan π , dan $S_0 = s$. Kita akan memasukkan faktor diskonto γ dalam semua persamaan kita, tetapi untuk dunia 4×3 kita akan menetapkan $\gamma = 1$.

Estimasi Utilitas Langsung

Sebuah metode sederhana untuk estimasi utilitas langsung ditemukan pada akhir tahun 1950-an dalam bidang teori kendali adaptif oleh Widrow dan Hoff (1960). Idennya adalah bahwa utilitas suatu keadaan adalah total imbalan yang diharapkan dari keadaan tersebut dan seterusnya (disebut imbalan yang diharapkan untuk pergi), dan setiap percobaan memberikan sampel dengan kuantitas ini untuk setiap keadaan yang dikunjungi.

Misalnya, percobaan pertama dalam rangkaian tiga percobaan yang diberikan sebelumnya memberikan total imbalan sampel sebesar 0,72 untuk keadaan (1,1), dua sampel sebesar 0,76 dan 0,84 untuk (1,2), dua sampel sebesar 0,80 dan 0,88 untuk (1,3), dan seterusnya. Dengan demikian, di akhir setiap rangkaian, algoritma menghitung imbalan-untuk-pergi yang teramati untuk setiap keadaan dan memperbarui utilitas yang diestimasi untuk keadaan tersebut, hanya dengan menyimpan rata-rata berjalan untuk setiap keadaan dalam sebuah tabel. Dalam limit percobaan yang tak terhingga banyaknya, rata-rata sampel akan konvergen ke ekspektasi sebenarnya dalam Persamaan (3.1).

Jelas bahwa estimasi utilitas langsung hanyalah sebuah contoh pembelajaran terawasi di mana setiap contoh memiliki keadaan sebagai masukan dan imbalan-untuk-pergi yang teramati sebagai keluaran. Ini berarti bahwa kita telah mereduksi pembelajaran penguatan menjadi masalah pembelajaran induktif standar. Bagian 3.4 membahas penggunaan jenis representasi yang lebih kuat untuk fungsi utilitas. Teknik pembelajaran untuk representasi tersebut dapat diterapkan langsung pada data yang diamati.

Estimasi utilitas langsung berhasil mereduksi masalah pembelajaran penguatan menjadi masalah pembelajaran induktif, yang telah banyak diketahui. Sayangnya, estimasi ini melewatkan sumber informasi yang sangat penting, yaitu fakta bahwa utilitas suatu keadaan tidaklah independen! Utilitas setiap keadaan sama dengan imbalannya sendiri ditambah utilitas yang diharapkan dari keadaan-keadaan berikutnya. Artinya, nilai utilitas mematuhi persamaan Bellman untuk kebijakan tetap:

$$U^\pi(s) = R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) U^\pi(s') \quad (3.2)$$

Dengan mengabaikan hubungan antar-keadaan, estimasi utilitas langsung kehilangan peluang pembelajaran. Misalnya, percobaan kedua dari tiga percobaan yang diberikan sebelumnya mencapai keadaan (3,2), yang belum pernah dikunjungi sebelumnya. Transisi berikutnya mencapai (3,3), yang diketahui dari percobaan pertama memiliki utilitas yang tinggi.

Persamaan Bellman langsung menunjukkan bahwa (3,2) juga cenderung memiliki utilitas yang tinggi, karena mengarah ke (3,3), tetapi estimasi utilitas langsung tidak mempelajari apa pun hingga akhir percobaan. Secara lebih luas, kita dapat memandang estimasi utilitas langsung sebagai pencarian U dalam ruang hipotesis yang jauh lebih besar daripada yang seharusnya, karena mencakup banyak fungsi yang melanggar persamaan Bellman. Karena alasan ini, algoritma seringkali konvergen sangat lambat.

```

function PASSIVE – ADP – AGENT(percept) returns an action
  inputs: percept, a percept indicating the current state  $s'$  and reward signal  $r'$ 
  persistent:  $\pi$ , a fixed policy
               mdp, an MDP with model  $P$ , rewards  $R$ , discount  $\gamma$ 
                $U$ , a table of utilities, initially empty
                $N_{sa}$ , a table of frequencies for state – action pairs, initially zero
                $N_{s|sa}$ , a table of outcome frequencies given state – action pairs, initially zero
                $s, a$ , the previous state and action, initially null

  if  $s'$  is new then  $U[s'] \leftarrow r'; R[s'] \leftarrow r'$ 
  if  $s$  is not null then
    increment  $N_{sa}[s, a]$  and  $N_{s'|sa}[s', s, a]$ 
    for each  $t$  such that  $N_{s'|sa}[t, s, a]$  is nonzero do
       $P(t | s, a) \leftarrow N_{s'|sa}[t, s, a] / N_{sa}[s, a]$ 
   $U \leftarrow \text{POLICY – EVALUATION}(\pi, U, \text{mdp})$ 
  if  $s'$ . TERMINAL? then  $s, a \leftarrow \text{null}$  else  $s, a \leftarrow s', \pi[s']$ 
  return  $a$ 

```

Gambar 3.2 Agen pembelajaran penguatan pasif berbasis pemrograman dinamis adaptif. Fungsi POLICY-EVALUATION menyelesaikan persamaan Bellman dengan kebijakan tetap.

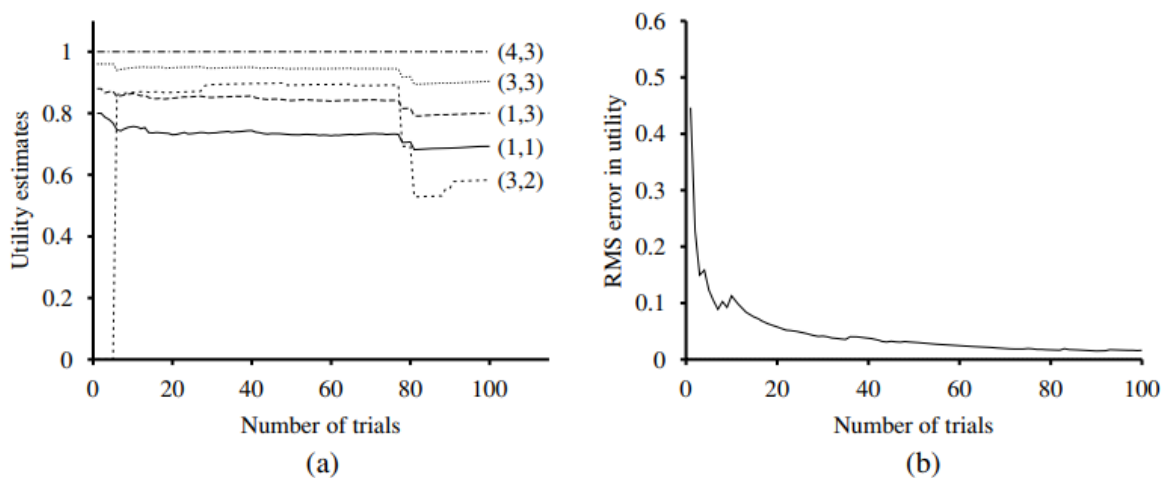
Pemrograman Dinamis Adaptif

Agen pemrograman dinamis adaptif (atau ADP) memanfaatkan kendala di antara utilitas keadaan dengan mempelajari model transisi yang menghubungkannya dan menyelesaikan proses keputusan Markov yang sesuai menggunakan metode pemrograman dinamis. Untuk agen pembelajaran pasif, ini berarti memasukkan model transisi yang dipelajari $P(s'|s, \pi(s))$ dan imbalan yang teramati $R(s)$ ke dalam persamaan Bellman (21.2) untuk menghitung utilitas keadaan. Sebagaimana telah kami sampaikan dalam pembahasan tentang iterasi kebijakan, persamaan-persamaan ini bersifat linear (tidak melibatkan maksimisasi) sehingga dapat diselesaikan menggunakan paket aljabar linear apa pun.

Sebagai alternatif, kita dapat mengadopsi pendekatan iterasi kebijakan yang dimodifikasi menggunakan proses iterasi nilai yang disederhanakan untuk memperbarui estimasi utilitas setelah setiap perubahan pada model yang dipelajari. Karena model biasanya hanya berubah sedikit pada setiap observasi, proses iterasi nilai dapat menggunakan estimasi utilitas sebelumnya sebagai nilai awal dan seharusnya konvergen dengan cukup cepat. Proses pembelajaran model itu sendiri mudah, karena lingkungannya sepenuhnya dapat diamati. Ini

berarti bahwa kita memiliki tugas pembelajaran terawasi di mana inputnya adalah pasangan keadaan-tindakan dan outputnya adalah keadaan yang dihasilkan.

Dalam kasus yang paling sederhana, kita dapat merepresentasikan model transisi sebagai tabel probabilitas. Kita melacak seberapa sering setiap hasil tindakan terjadi dan memperkirakan probabilitas transisi $P(s'|s, a)$ dari frekuensi tercapainya s' saat mengeksekusi a dalam s . Misalnya, dalam tiga percobaan, Kanan dieksekusi tiga kali dalam (1,3) dan dua dari tiga kali keadaan yang dihasilkan adalah (2,3), jadi $P((2,3) | (1,3), \text{Kanan})$ diperkirakan menjadi $2/3$.



Gambar 3.3 Kurva pembelajaran ADP pasif untuk dunia 4×3 , dengan kebijakan optimal yang ditunjukkan pada Gambar 3.1. (a) Estimasi utilitas untuk subset keadaan terpilih, sebagai fungsi dari jumlah percobaan. Perhatikan perubahan besar yang terjadi di sekitar percobaan ke-78—ini adalah pertama kalinya agen mencapai keadaan terminal -1 pada (4,2). (b) Kesalahan akar kuadrat rata-rata dalam estimasi untuk $U(1, 1)$, dirata-ratakan selama 20 kali percobaan, masing-masing 100 kali percobaan.

Program agen lengkap untuk agen ADP pasif ditunjukkan pada Gambar 3.2. Performanya di dunia 4×3 ditunjukkan pada Gambar 3.3. Dalam hal seberapa cepat estimasi nilainya meningkat, agen ADP hanya dibatasi oleh kemampuannya untuk mempelajari model transisi.

Dalam hal ini, ia menyediakan standar untuk mengukur algoritma pembelajaran penguatan lainnya. Namun, ia sulit untuk ruang keadaan yang besar. Dalam backgammon, misalnya, ia melibatkan penyelesaian sekitar 10^{50} persamaan dalam 10^{50} variabel yang tidak diketahui. Pembaca yang familier dengan gagasan pembelajaran Bayesian di Bab 20 akan menyadari bahwa algoritma pada Gambar 3.2 menggunakan estimasi kemungkinan maksimum untuk mempelajari model transisi; lebih lanjut, dengan memilih kebijakan hanya berdasarkan model yang diestimasi, ia bertindak seolah-olah model tersebut benar. Ini belum tentu merupakan ide yang bagus! Misalnya, seorang agen taksi yang tidak tahu bagaimana lampu lalu lintas dapat mengabaikan lampu merah sekali atau dua kali tanpa efek buruk, kemudian merumuskan kebijakan untuk mengabaikan lampu merah selanjutnya.

Sebaliknya, mungkin merupakan ide yang baik untuk memilih kebijakan yang, meskipun tidak optimal untuk model yang diestimasi dengan kemungkinan maksimum, bekerja cukup baik untuk seluruh rentang model yang memiliki peluang wajar untuk menjadi model yang sebenarnya. Ada dua pendekatan matematika yang memiliki ciri ini. Pendekatan pertama, pembelajaran penguatan Bayesian, mengasumsikan probabilitas sebelumnya $P(h)$ untuk setiap hipotesis h tentang model yang sebenarnya; probabilitas posterior $P(h|e)$ diperoleh dengan cara biasa menggunakan aturan Bayes berdasarkan pengamatan hingga saat ini. Kemudian, jika agen telah memutuskan untuk berhenti belajar, kebijakan optimal adalah kebijakan yang memberikan utilitas harapan tertinggi. Misalkan u_h^π adalah utilitas harapan, dirata-ratakan dari semua kemungkinan keadaan awal, yang diperoleh dengan mengeksekusi kebijakan π dalam model h . Maka kita memiliki

$$\pi^* = \operatorname{argmax}_{\pi} \sum_h P(h|e) u_h^\pi$$

Dalam beberapa kasus khusus, kebijakan ini bahkan dapat dihitung! Namun, jika agen akan terus belajar di masa mendatang, menemukan kebijakan optimal menjadi jauh lebih sulit, karena agen harus mempertimbangkan dampak observasi di masa mendatang terhadap keyakinannya tentang model transisi. Permasalahan ini menjadi POMDP yang status keyakinannya merupakan distribusi atas model. Konsep ini memberikan landasan analitis untuk memahami permasalahan eksplorasi yang dijelaskan di Bagian 3.3.

Pendekatan kedua, yang diturunkan dari teori kendali robust, memungkinkan serangkaian model $\mathcal{H}$ yang memungkinkan dan mendefinisikan kebijakan robust optimal sebagai kebijakan yang memberikan hasil terbaik dalam kasus terburuk atas $\mathcal{H}$:

$$\pi^* = \operatorname{argmax}_{\pi} \min_h u_h^\pi$$

Seringkali, himpunan $\mathcal{H}$ merupakan himpunan model yang melampaui ambang batas kemungkinan tertentu pada $P(h|e)$, sehingga pendekatan robust dan Bayesian saling berkaitan. Terkadang, solusi robust dapat dihitung secara efisien. Selain itu, terdapat algoritma pembelajaran penguatan yang cenderung menghasilkan solusi robust, meskipun kami tidak membahasnya di sini.

Pembelajaran Selisih Temporal

Menyelesaikan MDP yang mendasari seperti pada bagian sebelumnya bukanlah satu-satunya cara untuk menerapkan persamaan Bellman pada permasalahan pembelajaran. Cara lain adalah dengan menggunakan transisi yang teramati untuk menyesuaikan utilitas dari keadaan yang teramati agar sesuai dengan persamaan kendala. Perhatikan, misalnya, transisi dari (1,3) ke (2,3) pada percobaan kedua. Misalkan, sebagai hasil dari percobaan pertama, estimasi utilitasnya adalah $U^\pi(1,3) = 0,84$ dan $U^\pi(2,3) = 0,92$. Nah, jika transisi ini terjadi sepanjang waktu, kita akan mengharapkan utilitasnya mematuhi persamaan

$$U^\pi(1, 3) = -0.04 + U^\pi(2, 3)$$

Jadi, $U^\pi(1, 3)$ akan menjadi 0,88. Dengan demikian, estimasi saat ini sebesar 0,84 mungkin agak rendah dan perlu ditingkatkan. Secara umum, ketika terjadi transisi dari keadaan s ke keadaan s' , kami menerapkan pembaruan berikut pada $U^\pi(s)$:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha((R(s) + \gamma U^\pi(s')) - U^\pi(s)) \quad (3.3)$$

Di sini, α adalah parameter laju pembelajaran. Karena aturan pembaruan ini menggunakan selisih utilitas antara keadaan yang berurutan, aturan ini sering disebut persamaan selisih temporal, atau TD. Semua metode selisih temporal bekerja dengan menyesuaikan estimasi utilitas menuju ekuilibrium ideal yang berlaku secara lokal ketika estimasi utilitas tersebut benar. Dalam kasus pembelajaran pasif, ekuilibrium diberikan oleh Persamaan (3.2). Persamaan (3.3) memang menyebabkan agen mencapai ekuilibrium yang diberikan oleh Persamaan (3.2), tetapi ada beberapa kehalusan yang terlibat.

Pertama, perhatikan bahwa pembaruan hanya melibatkan penerus s' yang teramati, sedangkan kondisi ekuilibrium aktual melibatkan semua kemungkinan keadaan selanjutnya. Kita mungkin berpikir bahwa hal ini menyebabkan perubahan $U^\pi(s)$ yang sangat besar ketika transisi yang sangat jarang terjadi; tetapi, pada kenyataannya, karena transisi yang jarang terjadi hanya jarang terjadi, nilai rata-rata $U^\pi(s)$ akan konvergen ke nilai yang benar. Lebih lanjut, jika kita mengubah α dari parameter tetap menjadi fungsi yang menurun seiring bertambahnya jumlah kunjungan suatu keadaan, maka $U^\pi(s)$ sendiri akan konvergen ke nilai yang benar.¹ Ini memberi kita program agen yang ditunjukkan pada Gambar 3.4. Gambar 3.5 mengilustrasikan kinerja agen TD pasif di dunia 4×3 . Ia tidak belajar secepat agen ADP dan menunjukkan variabilitas yang jauh lebih tinggi, tetapi jauh lebih sederhana dan membutuhkan lebih sedikit komputasi per observasi. Perhatikan bahwa TD tidak memerlukan model transisi untuk melakukan pembaruannya. Lingkungan menyediakan koneksi antara keadaan tetangga dalam bentuk transisi yang diamati.

Pendekatan ADP dan pendekatan TD sebenarnya terkait erat. Keduanya mencoba melakukan penyesuaian lokal pada estimasi utilitas untuk membuat setiap keadaan "setuju" dengan penerusnya. Satu perbedaannya adalah TD menyesuaikan suatu keadaan agar sesuai dengan penerusnya yang teramati (Persamaan (3.3)), sedangkan ADP menyesuaikan keadaan agar sesuai dengan semua penerus yang mungkin terjadi, yang dibobot dengan probabilitasnya (Persamaan (3.2)). Perbedaan ini menghilang ketika efek penyesuaian TD dirata-ratakan pada sejumlah besar transisi, karena frekuensi setiap penerus dalam rangkaian transisi kira-kira proporsional dengan probabilitasnya. Perbedaan yang lebih penting adalah bahwa sementara TD membuat satu penyesuaian per transisi yang teramati, ADP membuat sebanyak yang diperlukan untuk memulihkan konsistensi antara estimasi utilitas U dan model lingkungan P . Meskipun transisi yang teramati hanya membuat perubahan lokal pada P , efeknya mungkin perlu disebarkan ke seluruh U . Dengan demikian, TD dapat dipandang sebagai perkiraan pertama yang kasar tetapi efisien untuk ADP.

Setiap penyesuaian yang dilakukan oleh ADP dapat dilihat, dari sudut pandang TD, sebagai hasil dari "pseudoexperience" yang dihasilkan dengan mensimulasikan model lingkungan saat ini. Pendekatan TD dapat diperluas dengan menggunakan model lingkungan untuk menghasilkan beberapa pseudoexperience—transisi yang dapat dibayangkan oleh agen TD, mengingat modelnya saat ini. Untuk setiap transisi yang diamati, agen TD dapat menghasilkan sejumlah besar transisi imajiner. Dengan cara ini, estimasi utilitas yang dihasilkan akan semakin mendekati estimasi utilitas ADP—tentu saja, dengan mengorbankan waktu komputasi yang lebih lama. Dengan cara yang sama, kita dapat menghasilkan versi ADP yang lebih efisien dengan secara langsung mengaproksimasi algoritma untuk iterasi nilai atau iterasi kebijakan. Meskipun algoritma iterasi nilai efisien, algoritma ini sulit diterapkan jika kita memiliki, katakanlah, 10^{100} status. Namun, banyak penyesuaian yang diperlukan terhadap nilai status pada setiap iterasi akan sangat kecil.

Salah satu pendekatan yang memungkinkan untuk menghasilkan jawaban yang cukup baik dengan cepat adalah dengan membatasi jumlah penyesuaian yang dilakukan setelah setiap transisi yang diamati. Kita juga dapat menggunakan heuristik untuk memeringkat kemungkinan penyesuaian sehingga hanya melakukan penyesuaian yang paling signifikan. Heuristik sapuan prioritas lebih suka melakukan penyesuaian pada keadaan yang kemungkinan penerusnya baru saja mengalami penyesuaian besar dalam estimasi utilitas mereka sendiri.

Dengan menggunakan heuristik seperti ini, algoritma ADP perkiraan biasanya dapat belajar kira-kira secepat ADP penuh, dalam hal jumlah urutan pelatihan, tetapi dapat beberapa kali lipat lebih efisien dalam hal komputasi. Hal ini memungkinkan mereka untuk menangani ruang keadaan yang terlalu besar untuk ADP penuh. Algoritma ADP perkiraan memiliki keuntungan tambahan: pada tahap awal mempelajari lingkungan baru, model lingkungan P sering kali jauh dari benar, sehingga tidak ada gunanya menghitung fungsi utilitas yang tepat untuk mencocokkannya. Algoritma perkiraan dapat menggunakan ukuran penyesuaian minimum yang berkurang saat model lingkungan menjadi lebih akurat. Ini menghilangkan iterasi nilai yang sangat panjang yang dapat terjadi di awal pembelajaran karena perubahan besar dalam model.

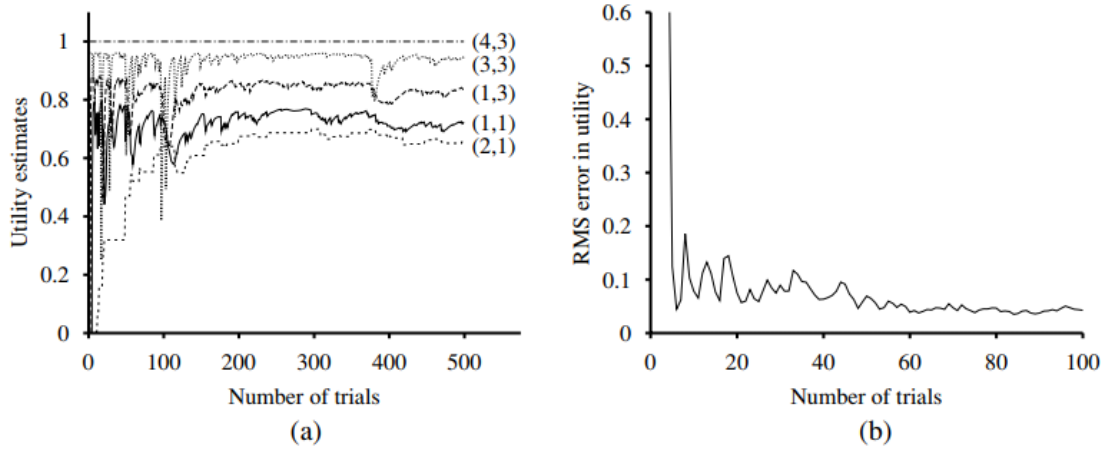
```

function PASSIVE – TD – AGENT(percept) returns an action
  inputs: percept, a percept indicating the current state  $s'$  and reward signal  $r'$ 
  persistent:  $\pi$ , a fixed policy
                $U$ , a table of utilities, initially empty
                $N_s$ , a table of frequencies for states, initially zero
                $s, a, r$ , the previous state, action, and reward, initially null

  if  $s'$  is new then  $U[s'] \leftarrow r'$ 
  if  $s$  is not null then
    increment  $N_s[s]$ 
     $U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma U[s'] - U[s])$ 
  if  $s'$ . TERMINAL? then  $s, a, r \leftarrow \text{null}$  else  $s, a, r \leftarrow s', \pi[s'], r'$ 
  return  $a$ 

```

Gambar 3.4 Agen pembelajaran penguatan pasif yang mempelajari estimasi utilitas menggunakan perbedaan temporal. Fungsi ukuran langkah $\alpha(n)$ dipilih untuk memastikan konvergensi, seperti yang dijelaskan dalam teks.



Gambar 3.5 Kurva pembelajaran TD untuk dunia 4×3 . (a) Estimasi utilitas untuk subset keadaan terpilih, sebagai fungsi dari jumlah percobaan. (b) Kesalahan akar kuadrat rata-rata dalam estimasi untuk $U(1, 1)$, dirata-ratakan selama 20 kali percobaan, masing-masing sebanyak 500 kali percobaan. Hanya 100 percobaan pertama yang ditampilkan agar dapat dibandingkan dengan Gambar 3.3.

3.3 PEMBELAJARAN PENGUATAN AKTIF

Agen pembelajaran pasif memiliki kebijakan tetap yang menentukan perilakunya. Agen aktif harus memutuskan tindakan apa yang akan diambil. Mari kita mulai dengan agen pemrograman dinamis adaptif dan pertimbangkan bagaimana agen tersebut harus dimodifikasi untuk menangani kebebasan baru ini. Pertama, agen perlu mempelajari model lengkap dengan probabilitas hasil untuk semua tindakan, alih-alih hanya model untuk kebijakan tetap. Mekanisme pembelajaran sederhana yang digunakan oleh PASSIVE-ADP-AGENT akan berfungsi dengan baik untuk ini. Selanjutnya, kita perlu memperhitungkan fakta bahwa agen memiliki pilihan tindakan. Utilitas yang perlu dipelajari adalah yang didefinisikan oleh kebijakan optimal; utilitas tersebut mematuhi persamaan Bellman, yang kami ulangi di sini untuk kenyamanan:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U(s') \quad (3.4)$$

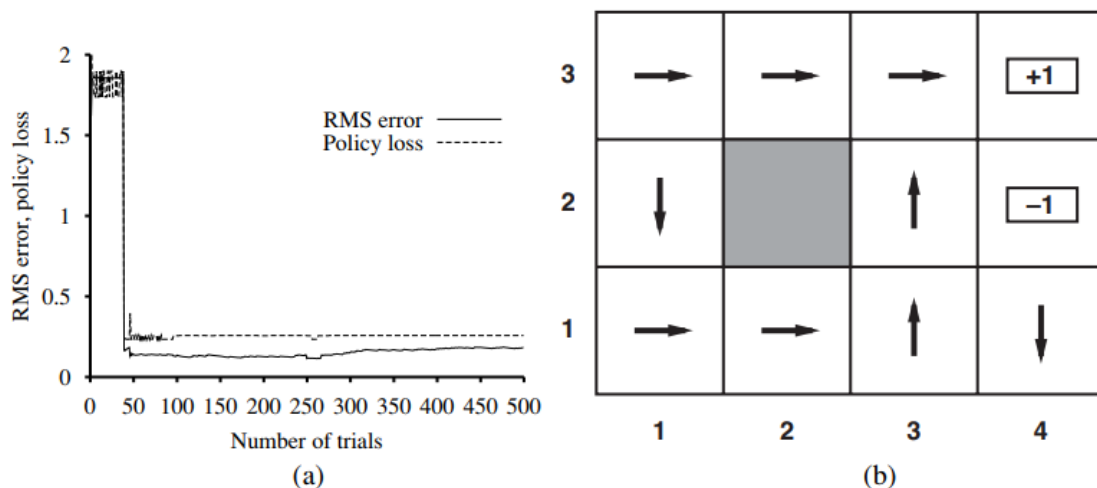
Persamaan-persamaan ini dapat diselesaikan untuk mendapatkan fungsi utilitas U menggunakan algoritma iterasi nilai atau iterasi kebijakan. Persoalan terakhir adalah apa yang harus dilakukan di setiap langkah. Setelah mendapatkan fungsi utilitas U yang optimal untuk model yang dipelajari, agen dapat mengekstrak tindakan optimal dengan melihat ke depan satu langkah untuk memaksimalkan utilitas yang diharapkan; sebagai alternatif, jika menggunakan iterasi kebijakan, kebijakan optimal sudah tersedia, sehingga agen cukup

menjalankan tindakan yang direkomendasikan oleh kebijakan optimal tersebut. Atau haruskah demikian?

Eksplorasi

Gambar 3.6 menunjukkan hasil dari satu rangkaian percobaan untuk agen ADP yang mengikuti rekomendasi kebijakan optimal untuk model yang dipelajari di setiap langkah. Agen tidak mempelajari utilitas yang sebenarnya atau kebijakan optimal yang sebenarnya! Yang terjadi justru, pada percobaan ke-39, agen menemukan kebijakan yang mencapai imbalan +1 di sepanjang rute yang lebih rendah melalui (2,1), (3,1), (3,2), dan (3,3). (Lihat Gambar 3.6(b).) Setelah bereksperimen dengan variasi kecil, mulai percobaan ke-276 dan seterusnya, ia tetap berpegang pada kebijakan tersebut, tidak pernah mempelajari utilitas keadaan lain dan tidak pernah menemukan rute optimal melalui (1,2), (1,3), dan (2,3). Kami menyebut agen ini sebagai agen greedy. Eksperimen berulang menunjukkan bahwa agen greedy sangat jarang konvergen ke kebijakan optimal untuk lingkungan ini dan terkadang konvergen ke kebijakan yang sangat buruk.

Bagaimana mungkin memilih tindakan optimal menghasilkan hasil yang suboptimal? Jawabannya adalah model yang dipelajari tidak sama dengan lingkungan yang sebenarnya; apa yang optimal dalam model yang dipelajari oleh karena itu bisa jadi suboptimal dalam lingkungan yang sebenarnya. Sayangnya, agen tidak mengetahui lingkungan yang sebenarnya, sehingga tidak dapat menghitung tindakan optimal untuk lingkungan yang sebenarnya. Lalu, apa yang harus dilakukan?



Gambar 3.6 Kinerja agen ADP rakus yang mengeksekusi tindakan yang direkomendasikan oleh kebijakan optimal untuk model yang dipelajari. (a) Kesalahan RMS dalam estimasi utilitas yang dirata-ratakan pada sembilan kotak nonterminal. (b) Kebijakan suboptimal yang menjadi titik konvergensi agen rakus dalam rangkaian percobaan tertentu ini.

Apa yang diabaikan oleh agen greedy adalah bahwa tindakan melakukan lebih dari sekadar memberikan imbalan menurut model yang dipelajari saat ini; tindakan juga berkontribusi untuk mempelajari model yang sebenarnya dengan memengaruhi persepsi yang diterima. Dengan meningkatkan model, agen akan menerima imbalan yang lebih besar di masa

mendatang. Oleh karena itu, seorang agen harus membuat pilihan antara eksploitasi untuk memaksimalkan imbalannya sebagaimana tercermin dalam estimasi utilitasnya saat ini dan eksplorasi untuk memaksimalkan kesejahteraan jangka panjangnya. Eksploitasi murni berisiko terjebak dalam kebiasaan. Eksplorasi murni untuk meningkatkan pengetahuan tidak akan berguna jika seseorang tidak pernah mempraktikkan pengetahuan tersebut. Di dunia nyata, seseorang harus terus-menerus memutuskan antara melanjutkan hidup yang nyaman dan menjelajahi hal yang tidak diketahui dengan harapan menemukan kehidupan yang baru dan lebih baik. Dengan pemahaman yang lebih besar, eksplorasi yang diperlukan lebih sedikit.

Bisakah kita sedikit lebih tepat dari ini? Apakah ada kebijakan eksplorasi yang optimal? Pertanyaan ini telah dipelajari secara mendalam dalam subbidang teori keputusan statistik yang membahas apa yang disebut masalah bandit. (Lihat bilah sisi.) Meskipun masalah bandit sangat sulit dipecahkan secara tepat untuk mendapatkan metode eksplorasi yang optimal, tetap dimungkinkan untuk menghasilkan skema yang masuk akal yang pada akhirnya akan mengarah pada perilaku optimal oleh agen. Secara teknis, skema semacam itu harus bersifat rakus dalam batas eksplorasi tak terbatas, atau GLIE. Skema GLIE harus mencoba setiap tindakan di setiap status berkali-kali tanpa batas untuk menghindari probabilitas terbatas bahwa tindakan optimal terlewatkan karena serangkaian hasil yang sangat buruk. Agen ADP yang menggunakan skema semacam itu pada akhirnya akan mempelajari model lingkungan yang sebenarnya. Skema GLIE juga harus pada akhirnya menjadi serakah, sehingga tindakan agen menjadi optimal sehubungan dengan model yang dipelajari (dan karenanya model yang sebenarnya).

Ada beberapa skema GLIE; salah satu yang paling sederhana adalah dengan meminta agen memilih tindakan acak sebagian kecil $1/t$ dari waktu dan mengikuti kebijakan greedy jika tidak. Meskipun hal ini pada akhirnya akan menyatu menjadi kebijakan optimal, hal ini bisa sangat lambat. Pendekatan yang lebih masuk akal akan memberikan beberapa bobot pada tindakan yang belum sering dicoba oleh agen, sementara cenderung menghindari tindakan yang diyakini memiliki utilitas rendah. Hal ini dapat diimplementasikan dengan mengubah persamaan kendala (3.4) sehingga memberikan estimasi utilitas yang lebih tinggi pada pasangan keadaan-tindakan yang relatif belum dieksplorasi. Pada dasarnya, hal ini sama dengan prior optimis atas lingkungan yang memungkinkan dan menyebabkan agen berperilaku awalnya seolah-olah ada imbalan yang luar biasa tersebar di semua tempat. Mari kita gunakan $U^+(s)$ untuk menunjukkan estimasi optimis dari utilitas (yaitu, imbalan yang diharapkan) dari keadaan s , dan misalkan $N(s, a)$ adalah jumlah kali tindakan a telah dicoba dalam keadaan s . Misalkan kita menggunakan iterasi nilai dalam agen pembelajaran ADP; maka kita perlu menulis ulang persamaan pembaruan untuk memasukkan estimasi optimis. Persamaan berikut melakukan hal ini:

$$U^+(s) \leftarrow R(s) + \gamma \max_a f \left(\sum_{s'} P(s'|s, a) U^+(s'), N(s, a) \right) \quad (3.5)$$

Di sini, $f(u, n)$ disebut fungsi eksplorasi. Fungsi ini menentukan bagaimana keserakahan (preferensi terhadap nilai u yang tinggi) dipertukarkan dengan rasa ingin tahu (preferensi

terhadap tindakan yang belum sering dicoba dan memiliki n yang rendah). Fungsi $f(u, n)$ seharusnya meningkat pada u dan menurun pada n . Jelas, ada banyak kemungkinan fungsi yang sesuai dengan kondisi ini. Salah satu definisi yang sangat sederhana adalah

$$f(u, n) = f(x) = \begin{cases} R^+, & \text{jika } n < N_e \\ u, & \text{Lainnya} \end{cases}$$

di mana R^+ adalah estimasi optimistis dari imbalan terbaik yang mungkin diperoleh dalam keadaan apa pun dan N_e adalah parameter tetap. Hal ini akan membuat agen mencoba setiap pasangan tindakan-keadaan setidaknya N_e kali. Fakta bahwa U^+ , alih-alih U , muncul di ruas kanan Persamaan (3.5), sangat penting. Seiring eksplorasi berlangsung, keadaan dan tindakan yang mendekati keadaan awal kemungkinan besar akan dicoba berkali-kali. Jika kita menggunakan U , estimasi utilitas yang lebih pesimistis, maka agen akan segera enggan untuk mengeksplorasi lebih jauh. Penggunaan U^+ , berarti bahwa manfaat eksplorasi disebarkan kembali dari tepi wilayah yang belum dieksplorasi, sehingga tindakan yang mengarah ke wilayah yang belum dieksplorasi diberi bobot lebih tinggi, alih-alih hanya tindakan yang tidak familiar. Efek dari kebijakan eksplorasi ini dapat dilihat dengan jelas pada Gambar 3.7, yang menunjukkan konvergensi cepat menuju kinerja optimal, tidak seperti pendekatan greedy. Kebijakan yang hampir optimal ditemukan hanya setelah 18 percobaan. Perhatikan bahwa estimasi utilitas itu sendiri tidak konvergen secepat itu. Hal ini karena agen segera berhenti menjelajahi bagian-bagian ruang keadaan yang tidak menguntungkan, dan baru mengunjunginya "secara tidak sengaja" setelahnya. Namun, masuk akal jika agen tidak peduli dengan utilitas pasti dari keadaan yang diketahuinya tidak diinginkan dan dapat dihindari.

Eksplorasi Dan Bandit

Di Las Vegas, bandit berlengan satu adalah mesin slot. Seorang penjudi dapat memasukkan koin, menarik tuas, dan mengumpulkan kemenangan (jika ada). Bandit berlengan n memiliki tuas n . Penjudi harus memilih tuas mana yang akan dimainkan pada setiap koin berikutnya—yang paling menguntungkan, atau mungkin yang belum pernah dicoba? Masalah bandit berlengan n adalah model formal untuk masalah nyata di banyak bidang yang sangat penting, seperti menentukan anggaran tahunan untuk penelitian dan pengembangan AI. Setiap lengan berhubungan dengan suatu tindakan (misalnya, mengalokasikan Rp. 325.800.000.000 untuk pengembangan buku teks AI baru), dan hasil dari menarik lengan tersebut berhubungan dengan manfaat yang diperoleh dari tindakan tersebut (sangat besar). Eksplorasi, baik itu eksplorasi bidang penelitian baru atau eksplorasi pusat perbelanjaan baru, berisiko, mahal, dan memiliki hasil yang tidak pasti; di sisi lain, kegagalan untuk mengeksplorasi sama sekali berarti seseorang tidak pernah menemukan tindakan yang bermanfaat.

Untuk merumuskan masalah bandit dengan tepat, seseorang harus mendefinisikan dengan tepat apa yang dimaksud dengan perilaku optimal. Sebagian besar definisi dalam literatur berasumsi bahwa tujuannya adalah untuk memaksimalkan total imbalan yang diharapkan diperoleh selama masa hidup agen. Definisi-definisi ini mengharuskan ekspektasi

diambil alih dunia-dunia yang mungkin di mana agen bisa berada, serta atas kemungkinan hasil dari setiap urutan tindakan di dunia mana pun. Di sini, "dunia" didefinisikan oleh model transisi $P(s' | s, a)$. Dengan demikian, untuk bertindak secara optimal, agen membutuhkan distribusi sebelumnya atas model-model yang mungkin. Masalah optimasi yang dihasilkan biasanya sangat sulit diatasi.

Dalam beberapa kasus misalnya, ketika hasil setiap mesin bersifat independen dan imbalan yang didiskontokan digunakan dimungkinkan untuk menghitung indeks Gittins untuk setiap mesin slot (Gittins, 1989). Indeks tersebut hanya merupakan fungsi dari berapa kali mesin slot telah dimainkan dan berapa banyak yang telah dibayarkan. Indeks untuk setiap mesin menunjukkan seberapa berharga untuk berinvestasi lebih banyak; Secara umum, semakin tinggi imbal hasil yang diharapkan dan semakin tinggi ketidakpastian utilitas suatu pilihan, semakin baik. Memilih mesin dengan nilai indeks tertinggi akan menghasilkan kebijakan eksplorasi yang optimal. Sayangnya, belum ditemukan cara untuk memperluas indeks Gittins ke permasalahan keputusan sekuensial.

Teori bandit berlengan n dapat digunakan untuk mendukung kewajaran strategi seleksi dalam algoritma genetika. Jika setiap lengan dalam permasalahan bandit berlengan n dianggap sebagai rangkaian gen yang mungkin, dan investasi koin pada salah satu lengan merupakan reproduksi gen-gen tersebut, maka dapat dibuktikan bahwa algoritma genetika mengalokasikan koin secara optimal, dengan asumsi independensi yang sesuai.

Mempelajari fungsi utilitas aksi

Setelah kita memiliki agen ADP aktif, mari kita pertimbangkan cara membangun agen pembelajaran selisih temporal aktif. Perubahan paling jelas dari kasus pasif adalah agen tersebut tidak lagi dilengkapi dengan kebijakan tetap. Jadi, jika mempelajari fungsi utilitas U , agen tersebut perlu mempelajari model agar dapat memilih tindakan berdasarkan U melalui one-step look-ahead. Masalah akuisisi model untuk agen TD identik dengan masalah untuk agen ADP. Bagaimana dengan aturan pembaruan TD itu sendiri? Mungkin mengejutkan, aturan pemsamaan (21.3) tetap tidak berubah. Hal ini mungkin tampak aneh, karena alasan berikut:

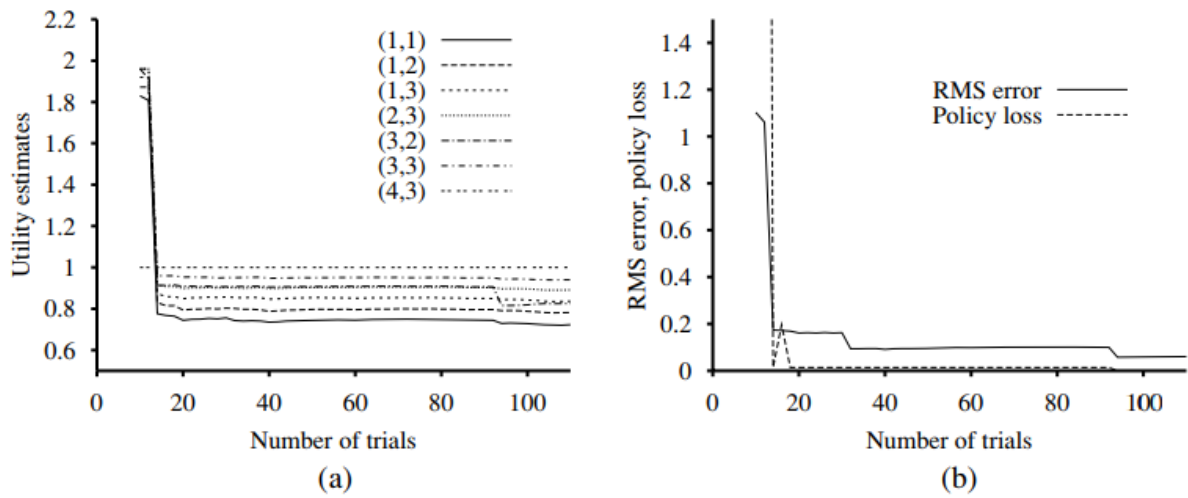
Misalkan agen mengambil langkah yang biasanya mengarah ke tujuan yang baik, tetapi karena nondeterminisme di lingkungan, agen tersebut berakhir dalam keadaan katastrofik. Aturan pembaruan TD akan menganggap hal ini seserius seolah-olah hasilnya adalah hasil normal dari tindakan tersebut, sementara orang mungkin berasumsi bahwa, karena hasilnya adalah kebetulan, agen tidak perlu terlalu mengkhawatirkannya. Faktanya, tentu saja, hasil yang tidak mungkin terjadi hanya akan jarang terjadi dalam serangkaian besar urutan pelatihan; oleh karena itu, dalam jangka panjang, efeknya akan terbobot secara proporsional dengan probabilitasnya, seperti yang kita harapkan. Sekali lagi, dapat ditunjukkan bahwa algoritma TD akan konvergen ke nilai yang sama dengan ADP seiring dengan meningkatnya jumlah urutan pelatihan hingga tak terhingga.

Ada metode TD alternatif, yang disebut pembelajaran- Q , yang mempelajari representasi utilitas-tindakan, alih-alih mempelajari utilitas. Kita akan menggunakan notasi $Q(s, a)$ untuk menyatakan nilai dari melakukan tindakan a dalam keadaan s . Nilai- Q berhubungan langsung dengan nilai utilitas sebagai berikut:

$$U(s) = \max_a Q(s, a) \quad (3.6)$$

Fungsi- Q mungkin tampak seperti cara lain untuk menyimpan informasi utilitas, tetapi fungsi- Q memiliki sifat yang sangat penting: agen TD yang mempelajari fungsi- Q tidak memerlukan model berbentuk $P(s' | s, a)$, baik untuk pembelajaran maupun pemilihan tindakan. Karena alasan ini, pembelajaran- Q disebut metode bebas-model. Seperti halnya utilitas, kita dapat menulis persamaan kendala yang harus berada pada kesetimbangan ketika nilai- Q benar:

$$Q(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a') \quad (3.7)$$



Gambar 3.7 Kinerja agen ADP eksploratif, menggunakan $R^+ = 2$ dan $N_e = 5$. (a) Estimasi utilitas untuk negara bagian terpilih dari waktu ke waktu. (b) Kesalahan RMS dalam nilai utilitas dan kerugian kebijakan terkait.

```

function Q – LEARNING – AGENT(percept) returns an action
  inputs: percept, a percept indicating the current state  $s'$  and reward signal  $r'$ 
  persistent:  $Q$ , a table of action values indexed by state and action, initially zero
                $N_{sa}$ , a table of frequencies for state – action pairs, initially zero
                $s, a, r$ , the previous state, action, and reward, initially null

  if TERMINAL? ( $s$ ) then  $Q[s, None] \leftarrow r'$ 
  if  $s$  is not null then
    increment  $N_{sa}[s, a]$ 
     $Q[s, a] \leftarrow Q[s, a] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$ 
     $s, a, r \leftarrow s', \operatorname{argmax}_a f(Q[s', a'], N_{sa}[s', a']), r'$ 
  return  $a$ 

```

Gambar 3.8 Agen Q-learning eksploratori. Agen ini merupakan pembelajar aktif yang mempelajari nilai $Q(s, a)$ dari setiap tindakan dalam setiap situasi. Agen ini menggunakan fungsi eksplorasi f yang sama dengan agen ADP eksploratori, tetapi menghindari keharusan

mempelajari model transisi karena nilai Q suatu keadaan dapat dihubungkan langsung dengan nilai Q tetangganya.

Seperti pada agen pembelajaran ADP, kita dapat menggunakan persamaan ini secara langsung sebagai persamaan pembaruan untuk proses iterasi yang menghitung *nilai* $-Q$ eksak, dengan model yang diestimasi. Namun, hal ini mengharuskan model juga dipelajari, karena persamaan tersebut menggunakan $P(s'|s, a)$. Di sisi lain, pendekatan perbedaan temporal tidak memerlukan model transisi keadaan—yang dibutuhkan hanyalah *nilai* $- nilai$ Q . Persamaan pembaruan untuk *pembelajaran* $- Q$ TD adalah

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a)) \quad (3.8)$$

yang dihitung setiap kali tindakan a dieksekusi dalam keadaan s yang mengarah ke keadaan s' . Desain agen lengkap untuk *agen* Q -learning eksploratori menggunakan TD ditunjukkan pada Gambar 3.8. Perhatikan bahwa ia menggunakan fungsi eksplorasi f yang sama persis dengan yang digunakan oleh agen ADP eksploratori oleh karena itu, perlu menyimpan statistik tentang tindakan yang diambil (tabel N). Jika kebijakan eksplorasi yang lebih sederhana digunakan misalnya, bertindak secara acak pada beberapa fraksi langkah, di mana fraksi tersebut menurun seiring waktu maka kita dapat mengabaikan statistik tersebut. $Q - learning$ memiliki kerabat dekat yang disebut SARSA (untuk State-Action-Reward-State-Action). Aturan pembaruan untuk SARSA sangat mirip dengan Persamaan (3.8):

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R(s) + \gamma Q(s', a') - Q(s, a)) \quad (3.9)$$

di mana a' adalah tindakan yang sebenarnya diambil dalam keadaan s' . Aturan ini diterapkan di akhir setiap quintuplet s, a, r, s', a' —maka dinamakan demikian. Perbedaan dari Q -learning cukup halus: sementara $Q - learning$ mencadangkan nilai- Q terbaik dari keadaan yang dicapai dalam transisi yang diamati, SARSA menunggu hingga suatu tindakan benar-benar diambil dan mencadangkan nilai- Q untuk tindakan itu. Sekarang, untuk agen greedy yang selalu mengambil tindakan dengan nilai- Q terbaik, kedua algoritma itu identik.

Namun, ketika eksplorasi terjadi, keduanya berbeda secara signifikan. Karena Q -learning menggunakan nilai- Q terbaik, ia tidak memperhatikan kebijakan yang sebenarnya diikuti ia adalah algoritma pembelajaran di luar kebijakan, sedangkan SARSA adalah algoritma sesuai kebijakan. Q -learning lebih fleksibel daripada SARSA, dalam arti bahwa agen Q -learning dapat belajar bagaimana berperilaku baik bahkan ketika dipandu oleh kebijakan eksplorasi acak atau adversarial. Di sisi lain, SARSA lebih realistis: misalnya, jika kebijakan keseluruhan, bahkan sebagian, dikendalikan oleh agen lain, lebih baik mempelajari fungsi Q untuk apa yang sebenarnya akan terjadi daripada apa yang diinginkan agen terjadi.

Baik Q -learning maupun SARSA mempelajari kebijakan optimal untuk dunia 4×3 , tetapi melakukannya dengan kecepatan yang jauh lebih lambat daripada agen ADP. Hal ini karena pembaruan lokal tidak menegakkan konsistensi di antara semua nilai- Q melalui model.

Perbandingan ini menimbulkan pertanyaan umum: apakah lebih baik mempelajari model dan fungsi utilitas atau mempelajari fungsi utilitas aksi tanpa model? Dengan kata lain, apa cara terbaik untuk merepresentasikan fungsi agen? Ini adalah isu yang mendasari kecerdasan buatan. Sebagaimana telah kami nyatakan di Bab 1, salah satu karakteristik historis utama dari sebagian besar penelitian AI adalah kepatuhannya (yang seringkali tidak disebutkan) pada pendekatan berbasis pengetahuan. Hal ini mengarah pada asumsi bahwa cara terbaik untuk merepresentasikan fungsi agen adalah dengan membangun representasi dari beberapa aspek lingkungan tempat agen tersebut berada.

Beberapa peneliti, baik di dalam maupun di luar AI, telah mengklaim bahwa ketersediaan metode bebas model seperti Q-learning berarti bahwa pendekatan berbasis pengetahuan tidak diperlukan. Namun, hanya ada sedikit dasar selain intuisi. Intuisi kami, sejujurnya, adalah bahwa seiring lingkungan menjadi lebih kompleks, keunggulan pendekatan berbasis pengetahuan menjadi lebih nyata. Hal ini terbukti bahkan dalam permainan seperti catur, dam (draught), dan backgammon (lihat bagian selanjutnya), di mana upaya untuk mempelajari fungsi evaluasi melalui model telah lebih berhasil daripada metode Q-learning.

3.4 GENERALISASI DALAM PEMBELAJARAN PENGUATAN

Sejauh ini, kami berasumsi bahwa fungsi utilitas dan fungsi-Q yang dipelajari oleh agen direpresentasikan dalam bentuk tabel dengan satu nilai keluaran untuk setiap tupel masukan. Pendekatan semacam itu bekerja cukup baik untuk ruang keadaan kecil, tetapi waktu konvergensi dan (untuk ADP) waktu per iterasi meningkat pesat seiring bertambahnya ruang. Dengan metode ADP aproksimasi yang dikontrol secara cermat, dimungkinkan untuk menangani 10.000 keadaan atau lebih. Ini cukup untuk lingkungan seperti labirin dua dimensi, tetapi dunia yang lebih realistis mustahil. Backgammon dan catur adalah subset kecil dari dunia nyata, namun ruang keadaan mereka masing-masing berisi sekitar 10^{20} dan 10^{40} keadaan. Akan absurd jika menganggap bahwa seseorang harus mengunjungi semua keadaan ini berkali-kali untuk mempelajari cara bermainnya!

Salah satu cara untuk menangani masalah tersebut adalah dengan menggunakan aproksimasi fungsi, yang berarti menggunakan representasi apa pun untuk fungsi-Q selain tabel pencarian. Representasi ini dianggap perkiraan karena fungsi utilitas atau fungsi-Q yang sebenarnya mungkin tidak dapat direpresentasikan dalam bentuk yang dipilih. Misalnya, penulis menjelaskan fungsi evaluasi untuk catur yang direpresentasikan sebagai fungsi linear terbobot dari sekumpulan fitur (atau fungsi basis) $f_1, \dots, f_n$:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \dots + \theta_n f_n(s)$$

Algoritma pembelajaran penguatan dapat mempelajari nilai untuk parameter $\theta = \theta_1, \dots, \theta_n$ sehingga fungsi evaluasi $\hat{U}_\theta$ mengaproksimasi fungsi utilitas yang sebenarnya. Alih-alih, katakanlah, 10^{40} nilai dalam tabel, aproksimator fungsi ini dicirikan oleh, katakanlah, $n = 20$ parameter kompresi yang sangat besar. Meskipun tidak seorang pun mengetahui fungsi utilitas

yang sebenarnya untuk catur, tidak seorang pun percaya bahwa fungsi tersebut dapat direpresentasikan secara tepat dalam 20 angka.

Namun, jika aproksimasinya cukup baik, agen tersebut mungkin masih bermain catur dengan sangat baik. Aproksimasi fungsi membuatnya praktis untuk merepresentasikan fungsi utilitas untuk ruang keadaan yang sangat besar, tetapi itu bukanlah manfaat utamanya. Kompresi yang dicapai oleh aproksimator fungsi memungkinkan agen pembelajaran untuk menggeneralisasi dari keadaan yang telah dikunjungi ke keadaan yang belum dikunjungi. Artinya, aspek terpenting dari aproksimasi fungsi bukanlah membutuhkan lebih sedikit ruang, tetapi memungkinkan generalisasi induktif atas keadaan masukan.

Untuk memberi Anda gambaran tentang kekuatan efek ini: dengan hanya memeriksa satu dari setiap 10^{12} kemungkinan keadaan backgammon, dimungkinkan untuk mempelajari fungsi utilitas yang memungkinkan program bermain sebaik manusia mana pun. Di sisi lain, tentu saja, terdapat masalah bahwa mungkin tidak ada fungsi dalam ruang hipotesis yang dipilih yang dapat mengaproksimasi fungsi utilitas sebenarnya dengan cukup baik. Seperti dalam semua pembelajaran induktif, terdapat tradeoff antara ukuran ruang hipotesis dan waktu yang dibutuhkan untuk mempelajari fungsi tersebut. Ruang hipotesis yang lebih besar meningkatkan kemungkinan ditemukannya aproksimasi yang baik, tetapi juga berarti konvergensi cenderung tertunda. Mari kita mulai dengan kasus paling sederhana, yaitu estimasi utilitas langsung.

Dengan aproksimasi fungsi, ini merupakan contoh pembelajaran terbimbing. Sebagai contoh, misalkan kita merepresentasikan utilitas untuk dunia 4×3 menggunakan fungsi linear sederhana. Fitur-fitur kotak hanyalah koordinat x dan y -nya, jadi kita memiliki

$$\hat{U}_{\theta}(x, y) = \theta_0 + \theta_1 x + \theta_2 y \quad (3.10)$$

Dengan demikian, jika $(\theta_0, \theta_1, \theta_2) = (0,5, 0,2, 0,1)$, maka $\hat{U}_{\theta}(1, 1) = 0,8$. Diberikan sekumpulan percobaan, kita memperoleh sekumpulan nilai sampel $\hat{U}_{\theta}(x, y)$, dan kita dapat menemukan kecocokan terbaik, dalam arti meminimalkan galat kuadrat, menggunakan regresi linier standar. Untuk pembelajaran penguatan, lebih masuk akal untuk menggunakan algoritma pembelajaran daring yang memperbarui parameter setelah setiap percobaan. Misalkan kita menjalankan percobaan dan total imbalan yang diperoleh mulai dari $(1,1)$ adalah 0,4. Ini menunjukkan bahwa $\hat{U}_{\theta}(1, 1)$, saat ini 0,8, terlalu besar dan harus dikurangi.

Bagaimana parameter harus disesuaikan untuk mencapai hal ini? Seperti halnya pembelajaran jaringan saraf, kita menulis fungsi galat dan menghitung gradiennya sehubungan dengan parameter. Jika $u_j(s)$ adalah total reward yang teramati dari keadaan s dan seterusnya pada percobaan ke- j , maka galat didefinisikan sebagai (setengah) selisih kuadrat antara total prediksi dan total aktual: $E_j(s) = (\hat{U}_{\theta}(s) - u_j(s))^2/2$. Laju perubahan galat terhadap setiap parameter θ_i adalah $\partial E_j / \partial \theta_i$, sehingga untuk menggerakkan parameter ke arah penurunan galat, kita perlu

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial E_j(s)}{\partial \theta_j} = \theta_i + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i} \quad (3.11)$$

Ini disebut aturan Widrow–Hoff, atau aturan delta, untuk kuadrat terkecil daring. Untuk aproksimator fungsi linear $\hat{U}_\theta(s)$ dalam Persamaan (3.10), kita mendapatkan tiga aturan pembaruan sederhana:

$$\begin{aligned} \theta_0 &\leftarrow \theta_0 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right), \\ \theta_1 &\leftarrow \theta_1 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) x, \\ \theta_2 &\leftarrow \theta_2 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) y, \end{aligned}$$

Kita dapat menerapkan aturan-aturan ini pada contoh di mana $\hat{U}_\theta(1,1)$ adalah 0,8 dan $u_j(1,1)$ adalah 0,4. θ_0 , θ_1 , dan θ_2 semuanya berkurang sebesar $0,4\alpha$, yang mengurangi kesalahan untuk (1,1). Perhatikan bahwa mengubah parameter θ sebagai respons terhadap transisi yang diamati antara dua keadaan juga mengubah nilai $\hat{U}_\theta$ untuk setiap keadaan lainnya! Inilah yang kita maksud dengan mengatakan bahwa aproksimasi fungsi memungkinkan pembelajar penguatan untuk menggeneralisasi dari pengalamannya.

Kami berharap bahwa agen akan belajar lebih cepat jika menggunakan aproksimator fungsi, asalkan ruang hipotesis tidak terlalu besar, tetapi mencakup beberapa fungsi yang cukup sesuai dengan fungsi utilitas sebenarnya. Peningkatan dalam dunia 4×3 terlihat tetapi tidak dramatis, karena ini adalah ruang keadaan yang sangat kecil untuk memulai. Peningkatannya jauh lebih besar di dunia 10×10 dengan imbalan +1 di (10,10).

Dunia ini sangat cocok untuk fungsi utilitas linear karena fungsi utilitas sebenarnya halus dan hampir linear. Jika kita menempatkan imbalan +1 di (5,5), utilitas sebenarnya lebih seperti piramida dan aproksimator fungsi dalam Persamaan (3.10) akan gagal total. Namun, belum semuanya hilang! Ingat bahwa yang penting untuk aproksimasi fungsi linear adalah bahwa fungsinya harus linear dalam parameter fiturnya sendiri dapat berupa fungsi nonlinier arbitrer dari variabel keadaan. Oleh karena itu, kita dapat memasukkan suku seperti $\theta_3 f_3(x, y) = \theta_3 \sqrt{(x - x_g)^2 + (y - y_g)^2}$ yang mengukur jarak ke tujuan.

Kita dapat menerapkan ide-ide ini dengan baik pada pembelajar dengan perbedaan temporal. Yang perlu kita lakukan hanyalah menyesuaikan parameter untuk mencoba mengurangi perbedaan temporal antara keadaan yang berurutan. Versi baru persamaan TD dan Q-learning diberikan oleh

$$\theta_i \leftarrow \theta_i - \alpha [R(s) + \gamma \hat{U}_\theta(s) - \hat{U}_\theta(s)] \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i} \quad (3.12)$$

untuk utilitas dan

$$\theta_i \leftarrow \theta_i - \alpha \left[R(s) + \gamma \max_{a'} \hat{Q}_\theta(s', a') - \hat{Q}_\theta(s, a) \right] \frac{\partial \hat{Q}_\theta(s, a)}{\partial \theta_i} \quad (3.13)$$

untuk nilai- Q . Untuk pembelajaran TD pasif, aturan pembaruan dapat ditunjukkan konvergen ke aproksimasi terdekat yang mungkin terhadap fungsi sebenarnya ketika aproksimator fungsi bersifat linear dalam parameternya. Dengan pembelajaran aktif dan fungsi nonlinier seperti jaringan saraf, semua taruhan dibatalkan: Ada beberapa kasus yang sangat sederhana di mana parameter dapat mencapai tak terhingga meskipun terdapat solusi yang baik dalam ruang hipotesis. Ada algoritma yang lebih canggih yang dapat menghindari masalah ini, tetapi saat ini pembelajaran penguatan dengan aproksimator fungsi umum masih merupakan seni yang rumit.

Aproksimasi fungsi juga dapat sangat membantu untuk mempelajari model lingkungan. Ingatlah bahwa mempelajari model untuk lingkungan yang dapat diamati adalah masalah pembelajaran terawasi, karena persepsi berikutnya memberikan keadaan hasil. Metode pembelajaran terawasi apa pun dapat digunakan, dengan penyesuaian yang sesuai untuk fakta bahwa kita perlu memprediksi deskripsi keadaan yang lengkap, alih-alih hanya klasifikasi Boolean atau nilai riil tunggal.

Untuk lingkungan yang dapat diamati sebagian, masalah pembelajaran jauh lebih sulit. Jika kita mengetahui variabel tersembunyi dan bagaimana hubungan kausalnya satu sama lain dan dengan variabel teramati, maka kita dapat memperbaiki struktur jaringan Bayesian dinamis dan menggunakan algoritma EM untuk mempelajari parameternya, seperti yang dijelaskan di Bab 2. Menemukan variabel tersembunyi dan mempelajari struktur model masih merupakan masalah yang belum terpecahkan. Beberapa contoh praktis dijelaskan di Bagian 3.6.

3.5 PENCARIAN KEBIJAKAN

Pendekatan terakhir yang akan kita pertimbangkan untuk masalah pembelajaran penguatan disebut pencarian kebijakan. Dalam beberapa hal, pencarian kebijakan adalah yang paling sederhana dari semua metode dalam bab ini: idenya adalah terus mengutak-atik kebijakan selama kinerjanya membaik, lalu berhenti. Mari kita mulai dengan kebijakan itu sendiri. Ingat bahwa kebijakan π adalah fungsi yang memetakan status ke tindakan. Kita terutama tertarik pada representasi π berparameter yang memiliki parameter jauh lebih sedikit daripada status dalam ruang status (seperti pada bagian sebelumnya). Misalnya, kita dapat merepresentasikan π dengan sekumpulan fungsi- Q berparameter, satu untuk setiap tindakan, dan mengambil tindakan dengan nilai prediksi tertinggi:

$$\pi(s) = \max_a \hat{Q}_\theta(s, a). \quad (3.14)$$

Setiap fungsi Q bisa menjadi fungsi linear dari parameter θ , seperti pada Persamaan (3.10), atau bisa menjadi fungsi nonlinier seperti jaringan saraf. Pencarian kebijakan kemudian akan menyesuaikan parameter θ untuk meningkatkan kebijakan. Perhatikan bahwa jika kebijakan direpresentasikan oleh fungsi Q , maka pencarian kebijakan menghasilkan proses yang mempelajari fungsi Q . Proses ini tidak sama dengan Q -learning! Dalam Q -learning dengan

perkiraan fungsi, algoritma menemukan nilai θ sehingga $\hat{Q}_\theta$ "dekat" dengan Q^* , fungsi Q yang optimal.

Pencarian kebijakan, di sisi lain, menemukan nilai θ yang menghasilkan kinerja yang baik; nilai yang ditemukan oleh kedua metode tersebut mungkin berbeda secara substansial. (Misalnya, fungsi Q perkiraan yang didefinisikan oleh $\hat{Q}_\theta(s, a) = Q^*(s, a)/10$ memberikan kinerja optimal, meskipun tidak mendekati Q^* sama sekali.) Contoh lain yang jelas dari perbedaan ini adalah kasus ketika $\pi(s)$ dihitung menggunakan, katakanlah, pencarian look-ahead depth-10 dengan fungsi utilitas perkiraan $\hat{U}_\theta$. Nilai θ yang memberikan hasil yang baik mungkin masih jauh dari membuat $\hat{U}_\theta$ menyerupai fungsi utilitas yang sebenarnya.

Salah satu masalah dengan representasi kebijakan seperti yang diberikan dalam Persamaan (3.14) adalah bahwa kebijakan tersebut merupakan fungsi diskontinu dari parameter ketika tindakannya diskrit. (Untuk ruang tindakan kontinu, kebijakan dapat menjadi fungsi halus dari parameter.) Artinya, akan ada nilai θ sedemikian rupa sehingga perubahan infinitesimal dalam θ menyebabkan kebijakan beralih dari satu tindakan ke tindakan lainnya. Ini berarti bahwa nilai kebijakan juga dapat berubah secara diskontinu, yang membuat pencarian berbasis gradien menjadi sulit. Oleh karena itu, metode pencarian kebijakan sering kali menggunakan representasi kebijakan stokastik $\pi_\theta(s, a)$, yang menentukan probabilitas pemilihan tindakan a dalam keadaan s . Salah satu representasi yang populer adalah fungsi softmax:

$$\pi_\theta(s, a) = e^{\hat{Q}_\theta(s, a)} / \sum_{a'} e^{\hat{Q}_\theta(s, a')}$$

Softmax menjadi hampir deterministik jika satu tindakan jauh lebih baik daripada yang lain, tetapi selalu menghasilkan fungsi diferensiabel θ ; oleh karena itu, nilai kebijakan (yang bergantung secara kontinu pada probabilitas pemilihan tindakan) adalah fungsi diferensiabel θ . Softmax adalah generalisasi fungsi logistik ke beberapa variabel.

Sekarang mari kita lihat metode untuk meningkatkan kebijakan. Kita mulai dengan kasus paling sederhana: kebijakan deterministik dan lingkungan deterministik. Misalkan $\rho(\theta)$ adalah nilai kebijakan, yaitu, ekspektasi reward-to-go ketika π_θ dieksekusi. Jika kita dapat menurunkan ekspresi untuk $\rho(\theta)$ dalam bentuk tertutup, maka kita memiliki masalah optimasi standar. Kita dapat mengikuti vektor gradien kebijakan $\nabla_{\theta} \rho(\theta)$ asalkan $\rho(\theta)$ terdiferensiabel. Alternatifnya, jika $\rho(\theta)$ tidak tersedia dalam bentuk tertutup, kita dapat mengevaluasi π_θ hanya dengan mengeksekusinya dan mengamati akumulasi imbalannya. Kita dapat mengikuti gradien empiris dengan mendaki bukit yaitu, mengevaluasi perubahan nilai kebijakan untuk kenaikan kecil pada setiap parameter. Dengan peringatan yang biasa, proses ini akan konvergen ke optimum lokal dalam ruang kebijakan.

Ketika lingkungan (atau kebijakan) bersifat stokastik, segalanya menjadi lebih rumit. Misalkan kita mencoba melakukan pendakian bukit, yang memerlukan perbandingan $\rho(\theta)$ dan $\rho(\theta + \Delta\theta)$ untuk beberapa $\Delta\theta$ kecil. Masalahnya adalah total imbalan pada setiap percobaan dapat sangat bervariasi, sehingga estimasi nilai kebijakan dari sejumlah kecil

percobaan akan sangat tidak andal; mencoba membandingkan dua estimasi tersebut akan menjadi lebih tidak andal lagi.

Salah satu solusinya adalah dengan menjalankan banyak percobaan, mengukur varians sampel, dan menggunakannya untuk menentukan bahwa cukup banyak percobaan yang telah dijalankan untuk mendapatkan indikasi yang andal tentang arah perbaikan untuk $\rho(\theta)$. Sayangnya, hal ini tidak praktis untuk banyak permasalahan nyata di mana setiap percobaan mungkin mahal, memakan waktu, dan bahkan mungkin berbahaya. Untuk kasus kebijakan stokastik $\pi_\theta(s, a)$, estimasi gradien yang tidak bias pada θ , $\nabla_\theta \rho(\theta)$, dapat diperoleh langsung dari hasil percobaan yang dilakukan pada θ . Untuk menyederhanakannya, kita akan menurunkan estimasi ini untuk kasus sederhana lingkungan nonsekuensial di mana imbalan $R(a)$ diperoleh segera setelah melakukan tindakan a pada keadaan awal s_0 . Dalam kasus ini, nilai kebijakan hanyalah nilai ekspektasi imbalan, dan kita memiliki

$$\nabla_\theta \rho(\theta) = \nabla_\theta \rho \sum_a \pi_\theta(s_0, a) R(a) = \sum_a (\nabla_\theta \pi_\theta(s_0, a)) R(a)$$

Sekarang kita melakukan trik sederhana agar penjumlahan ini dapat didekati dengan sampel yang dihasilkan dari distribusi probabilitas yang didefinisikan oleh $\pi_\theta(s_0, a)$. Misalkan kita memiliki N percobaan secara keseluruhan dan tindakan yang diambil pada percobaan ke- j adalah a_j . Maka

$$\nabla_\theta \rho(\theta) = \sum_a \pi_\theta(s_0, a) \cdot \frac{(\nabla_\theta \pi_\theta(s_0, a)) R(a)}{\pi_\theta(s_0, a)} \approx \frac{1}{N} \sum_{j=1}^N \frac{(\nabla_\theta \pi_\theta(s_0, a_j)) R(a_j)}{\pi_\theta(s_0, a_j)}$$

Dengan demikian, gradien sebenarnya dari nilai kebijakan didekati dengan jumlah suku yang melibatkan gradien probabilitas pemilihan tindakan dalam setiap percobaan. Untuk kasus sekuensial, hal ini digeneralisasi menjadi

$$\nabla_\theta \rho(\theta) \approx \frac{1}{N} \sum_{j=1}^N \frac{(\nabla_\theta \pi_\theta(s_0, a_j)) R_j(s)}{\pi_\theta(s_0, a_j)}$$

untuk setiap keadaan s yang dikunjungi, di mana a_j dieksekusi di s pada percobaan ke- j dan $R_j(s)$ adalah total imbalan yang diterima dari keadaan s dan seterusnya pada percobaan ke- j . Algoritma yang dihasilkan disebut REINFORCE; algoritma ini biasanya jauh lebih efektif daripada mendaki bukit dengan banyak percobaan pada setiap nilai θ . Namun, algoritma ini masih jauh lebih lambat dari yang seharusnya.

Pertimbangkan tugas berikut: diberikan dua program blackjack, tentukan mana yang terbaik. Salah satu cara untuk melakukannya adalah dengan meminta masing-masing program bermain melawan "dealer" standar untuk sejumlah putaran tertentu dan kemudian mengukur kemenangan masing-masing. Masalahnya, seperti yang telah kita lihat, adalah bahwa kemenangan setiap program sangat berfluktuasi tergantung pada apakah program tersebut

menerima kartu bagus atau buruk. Solusi yang jelas adalah dengan menghasilkan sejumlah putaran tertentu terlebih dahulu dan meminta setiap program memainkan set putaran yang sama. Dengan cara ini, kita menghilangkan kesalahan pengukuran akibat perbedaan kartu yang diterima. Ide ini, yang disebut pengambilan sampel berkorelasi, mendasari algoritma pencarian kebijakan yang disebut PEGASUS.

Algoritma ini berlaku untuk domain yang menyediakan simulator sehingga hasil "acak" dari tindakan dapat diulang. Algoritma ini bekerja dengan menghasilkan N urutan angka acak terlebih dahulu, yang masing-masing dapat digunakan untuk menjalankan uji coba kebijakan apa pun. Pencarian kebijakan dilakukan dengan mengevaluasi setiap kandidat kebijakan menggunakan serangkaian urutan acak yang sama untuk menentukan hasil tindakan. Dapat ditunjukkan bahwa jumlah sekuens acak yang diperlukan untuk memastikan nilai setiap kebijakan diestimasi dengan baik hanya bergantung pada kompleksitas ruang kebijakan, dan sama sekali tidak bergantung pada kompleksitas domain yang mendasarinya.

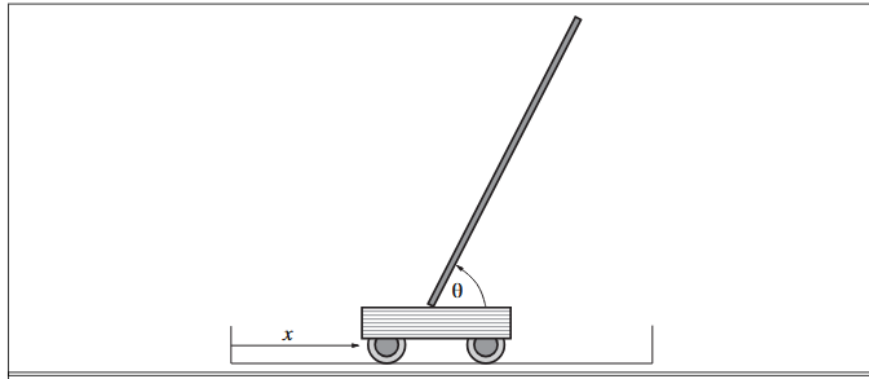
3.6 APLIKASI PEMBELAJARAN PENGUATAN

Sekarang kita beralih ke contoh aplikasi pembelajaran penguatan skala besar. Kita mempertimbangkan aplikasi dalam permainan gim, di mana model transisi diketahui dan tujuannya adalah mempelajari fungsi utilitas, dan dalam robotika, di mana modelnya biasanya tidak diketahui.

Aplikasi dalam Permainan Gim

Aplikasi signifikan pertama dari pembelajaran penguatan juga merupakan program pembelajaran signifikan pertama program catur yang ditulis oleh Arthur Samuel (1959, 1967). Samuel pertama kali menggunakan fungsi linear terbobot untuk evaluasi posisi, menggunakan hingga 16 suku pada satu waktu. Ia menerapkan versi Persamaan (3.12) untuk memperbarui bobot. Namun, terdapat beberapa perbedaan signifikan antara programnya dan metode saat ini. Pertama, ia memperbarui bobot menggunakan selisih antara keadaan saat ini dan nilai cadangan yang dihasilkan oleh pencarian penuh di pohon pencarian. Ini berfungsi dengan baik, karena sama saja dengan melihat ruang keadaan pada tingkat ketelitian yang berbeda.

Perbedaan kedua adalah program tersebut tidak menggunakan imbalan yang teramat! Artinya, nilai-nilai status terminal yang dicapai dalam permainan mandiri diabaikan. Ini berarti secara teoretis, program Samuel mungkin tidak konvergen, atau konvergen pada strategi yang dirancang untuk kalah, alih-alih menang. Ia berhasil menghindari nasib ini dengan menegaskan bahwa bobot keuntungan materi harus selalu positif. Hebatnya, hal ini cukup untuk mengarahkan program ke area ruang bobot yang sesuai dengan permainan catur yang baik.



Gambar 3.9 Pengaturan untuk masalah penyeimbangan tiang panjang di atas kereta yang sedang bergerak. Kereta dapat digerakkan ke kiri atau ke kanan oleh pengontrol yang mengamati x , θ , $\dot{x}$, dan $\dot{\theta}$.

Program backgammon Gerry Tesauro, TD-GAMMON (1992), dengan tegas menggambarkan potensi teknik pembelajaran penguatan. Dalam karya sebelumnya, Tesauro mencoba mempelajari representasi jaringan saraf dari $Q(s, a)$ langsung dari contoh langkah yang diberi label dengan nilai relatif oleh seorang pakar manusia. Pendekatan ini terbukti sangat membosankan bagi pakar.

Hasilnya adalah sebuah program, yang disebut NEUROGAMMON, yang kuat menurut standar komputer, tetapi tidak kompetitif dengan pakar manusia. Proyek TD-GAMMON merupakan upaya untuk belajar dari permainan mandiri saja. Sinyal hadiah hanya diberikan di akhir setiap permainan. Fungsi evaluasi direpresentasikan oleh jaringan saraf tiruan yang terhubung penuh dengan satu lapisan tersembunyi yang berisi 40 simpul.

Hanya dengan penerapan Persamaan (3.12) berulang kali, TD-GAMMON belajar bermain jauh lebih baik daripada NEUROGAMMON, meskipun representasi masukan hanya berisi posisi papan mentah tanpa fitur komputasi. Hal ini membutuhkan sekitar 200.000 permainan latihan dan dua minggu waktu komputasi. Meskipun jumlah tersebut mungkin tampak banyak, itu hanyalah sebagian kecil dari ruang keadaan.

Ketika fitur prakomputasi ditambahkan ke representasi masukan, sebuah jaringan dengan 80 simpul tersembunyi mampu, setelah 300.000 permainan latihan, mencapai standar permainan yang sebanding dengan tiga pemain manusia teratas di dunia. Kit Woolsey, seorang pemain dan analis papan atas, mengatakan bahwa "Tidak diragukan lagi dalam benak saya bahwa penilaian posisinya jauh lebih baik daripada saya."

Aplikasi pada Kontrol Robot

Pengaturan untuk masalah keseimbangan kereta-tiang yang terkenal, juga dikenal sebagai pendulum terbalik, ditunjukkan pada Gambar 3.9. Masalahnya adalah mengendalikan posisi x kereta agar tiang tetap tegak lurus ($\theta \approx \pi/2$), sambil tetap berada dalam batas lintasan kereta seperti yang ditunjukkan. Ribuan makalah dalam pembelajaran penguatan dan teori kontrol telah diterbitkan mengenai masalah yang tampaknya sederhana ini. Masalah kereta-tiang berbeda dari masalah yang dijelaskan sebelumnya karena variabel keadaan

x , θ , $\dot{x}$, dan $\dot{\theta}$. bersifat kontinu. Aksi-aksinya biasanya diskrit: sentakan ke kiri atau sentakan ke kanan, yang disebut rezim kontrol bang-bang.

Penelitian paling awal tentang pembelajaran untuk masalah ini dilakukan oleh Michie dan Chambers (1968). Algoritma BOXES mereka mampu menyeimbangkan tiang selama lebih dari satu jam hanya setelah sekitar 30 percobaan. Lebih lanjut, tidak seperti banyak sistem selanjutnya, BOXES diimplementasikan dengan kereta dan tiang sungguhan, bukan simulasi. Algoritme ini pertama-tama mendiskritkan ruang keadaan empat dimensi ke dalam kotak-kotak—karena itulah namanya.



Gambar 3.10 Gambar time-lapse tumpang tindih dari sebuah helikopter otonom yang melakukan manuver "nose-in-circle" yang sangat sulit. Helikopter tersebut berada di bawah kendali kebijakan yang dikembangkan oleh algoritma pencarian kebijakan PEGASUS. Sebuah model simulator dikembangkan dengan mengamati efek berbagai manipulasi kendali pada helikopter sungguhan; kemudian algoritma tersebut dijalankan pada model simulator semalaman. Berbagai pengontrol dikembangkan untuk berbagai manuver. Dalam semua kasus, kinerjanya jauh melampaui kinerja pilot manusia ahli yang menggunakan kendali jarak jauh.

Kemudian, ia menjalankan uji coba hingga tiang roboh atau kereta mencapai ujung lintasan. Penguatan negatif dikaitkan dengan tindakan terakhir di kotak terakhir dan kemudian disebarkan kembali melalui urutan tersebut. Ditemukan bahwa diskritisasi menyebabkan beberapa masalah ketika peralatan diinisialisasi pada posisi yang berbeda dari yang digunakan dalam pelatihan, menunjukkan bahwa generalisasi tidak sempurna.

Generalisasi yang lebih baik dan pembelajaran yang lebih cepat dapat diperoleh dengan menggunakan algoritme yang secara adaptif mempartisi ruang keadaan sesuai dengan variasi yang diamati dalam imbalan, atau dengan menggunakan aproksimator fungsi nonlinier keadaan kontinu seperti jaringan saraf tiruan. Saat ini, menyeimbangkan pendulum terbalik rangkap tiga adalah latihan yang umum suatu prestasi yang jauh melampaui kemampuan kebanyakan manusia.

Yang lebih mengesankan lagi adalah penerapan pembelajaran penguatan pada penerbangan helikopter (Gambar 3.10). Penelitian ini umumnya menggunakan pencarian kebijakan serta algoritma PEGASUS dengan simulasi berdasarkan model transisi yang dipelajari.

3.7 RINGKASAN

Bab ini telah mengkaji permasalahan pembelajaran penguatan: bagaimana sebuah agen dapat menjadi mahir dalam lingkungan yang tidak dikenal, hanya dengan persepsi dan imbalan yang sesekali diberikan. Pembelajaran penguatan dapat dipandang sebagai gambaran kecil dari keseluruhan permasalahan AI, tetapi pembelajaran ini dipelajari dalam sejumlah pengaturan yang disederhanakan untuk memfasilitasi kemajuan. Poin-poin utamanya adalah:

- Desain agen secara keseluruhan menentukan jenis informasi yang harus dipelajari. Tiga desain utama yang kami bahas adalah desain berbasis model, menggunakan model P dan fungsi utilitas U ; desain bebas model, menggunakan fungsi utilitas aksi Q ; dan desain refleksi, menggunakan kebijakan π .
- Utilitas dapat dipelajari menggunakan tiga pendekatan:
 1. Estimasi utilitas langsung menggunakan total imbalan yang diamati untuk suatu keadaan tertentu sebagai bukti langsung untuk mempelajari utilitasnya.
 2. Pemrograman dinamis adaptif (ADP) mempelajari model dan fungsi imbalan dari observasi, lalu menggunakan iterasi nilai atau kebijakan untuk mendapatkan utilitas atau kebijakan optimal. ADP memanfaatkan kendala lokal pada utilitas keadaan yang diberlakukan melalui struktur lingkungan sekitar secara optimal.
 3. Metode perbedaan temporal (TD) memperbarui estimasi utilitas agar sesuai dengan estimasi keadaan penerus. Metode ini dapat dipandang sebagai aproksimasi sederhana untuk pendekatan ADP yang dapat belajar tanpa memerlukan model transisi. Namun, menggunakan model yang dipelajari untuk menghasilkan pseudoexperience dapat menghasilkan pembelajaran yang lebih cepat.
- Fungsi utilitas aksi, atau fungsi Q , dapat dipelajari dengan pendekatan ADP atau pendekatan TD. Dengan TD, pembelajaran Q tidak memerlukan model baik dalam fase pembelajaran maupun pemilihan tindakan. Hal ini menyederhanakan masalah pembelajaran tetapi berpotensi membatasi kemampuan untuk belajar di lingkungan yang kompleks, karena agen tidak dapat mensimulasikan hasil dari kemungkinan tindakan.
- Ketika agen pembelajaran bertanggung jawab untuk memilih tindakan saat belajar, ia harus mempertimbangkan nilai estimasi tindakan tersebut dengan potensi untuk mempelajari informasi baru yang bermanfaat. Solusi eksak untuk masalah eksplorasi tidak mungkin dilakukan, tetapi beberapa heuristik sederhana cukup memadai.
- Dalam ruang keadaan yang besar, algoritma pembelajaran penguatan harus menggunakan representasi fungsional aproksimasi agar dapat melakukan generalisasi atas keadaan-keadaan tersebut. Sinyal perbedaan temporal dapat digunakan secara

langsung untuk memperbarui parameter dalam representasi seperti jaringan saraf tiruan.

- Metode pencarian kebijakan beroperasi secara langsung pada representasi kebijakan, mencoba memperbaikinya berdasarkan kinerja yang diamati. Variasi kinerja dalam domain stokastik merupakan masalah serius; untuk domain simulasi, hal ini dapat diatasi dengan memperbaiki keacakan terlebih dahulu.

Karena potensinya untuk menghilangkan pengkodean manual pada strategi kontrol, pembelajaran penguatan terus menjadi salah satu bidang penelitian pembelajaran mesin yang paling aktif. Aplikasi dalam robotika menjanjikan nilai yang sangat berharga; hal ini membutuhkan metode untuk menangani lingkungan yang berkelanjutan, berdimensi tinggi, dan dapat diamati sebagian, di mana perilaku yang berhasil dapat terdiri dari ribuan atau bahkan jutaan tindakan primitif.

BAB 4

PEMROSESAN BAHASA ALAMI

Homo sapiens dibedakan dari spesies lain oleh kapasitas bahasanya. Sekitar 100.000 tahun yang lalu, manusia belajar berbicara, dan sekitar 7.000 tahun yang lalu belajar menulis. Meskipun simpanse, lumba-lumba, dan hewan lainnya telah menunjukkan kosakata ratusan isyarat, hanya manusia yang dapat dengan andal mengomunikasikan pesan yang berbeda secara kualitatif dalam jumlah tak terbatas tentang topik apa pun menggunakan isyarat-isyarat yang terpisah.

Tentu saja, ada atribut lain yang unik bagi manusia: tidak ada spesies lain yang mengenakan pakaian, menciptakan karya seni representasional, atau menonton televisi tiga jam sehari. Namun, ketika Alan Turing mengusulkan Tesnya, ia mendasarkannya pada bahasa, bukan seni atau TV. Ada dua alasan utama mengapa kita ingin agen komputer kita mampu memproses bahasa alami: pertama, untuk berkomunikasi dengan manusia, topik yang kita bahas di Bab 5, dan kedua, untuk memperoleh informasi dari bahasa tulis, fokus bab ini.

Ada lebih dari satu triliun halaman informasi di Web, hampir semuanya dalam bahasa alami. Agen yang ingin melakukan akuisisi pengetahuan perlu memahami (setidaknya sebagian) bahasa ambigu dan rumit yang digunakan manusia. Kami mengkaji masalah ini dari sudut pandang tugas-tugas pencarian informasi spesifik: klasifikasi teks, pengambilan informasi, dan ekstraksi informasi. Salah satu faktor umum dalam menangani tugas-tugas ini adalah penggunaan model bahasa: model yang memprediksi distribusi probabilitas ekspresi bahasa.

4.1 MODEL BAHASA

Bahasa formal, seperti bahasa pemrograman Java atau Python, memiliki model bahasa yang didefinisikan secara tepat. Suatu bahasa dapat didefinisikan sebagai sekumpulan string; "print(2 + 2)" adalah program legal dalam bahasa Python, sedangkan "2)+(2 print" tidak. Karena terdapat jumlah program legal yang tak terhingga, program-program tersebut tidak dapat dienumerasi; melainkan ditentukan oleh seperangkat aturan yang disebut tata bahasa. Bahasa formal juga memiliki aturan yang mendefinisikan makna atau semantik suatu program; misalnya, aturan tersebut menyatakan bahwa "makna" dari "2 + 2" adalah 4, dan makna dari "1/0" adalah bahwa kesalahan telah ditandai.

Bahasa alami, seperti bahasa Inggris atau Spanyol, tidak dapat dikarakterisasikan sebagai rangkaian kalimat yang definitif. Semua orang sepakat bahwa "Not to be invitation is sad" adalah kalimat dalam bahasa Inggris, tetapi orang-orang tidak sependapat tentang gramatikalitas "To be not invitation is sad." Oleh karena itu, akan lebih bermanfaat untuk mendefinisikan model bahasa alami sebagai distribusi probabilitas atas kalimat-kalimat daripada rangkaian definitif. Artinya, daripada menanyakan apakah serangkaian kata merupakan anggota rangkaian yang mendefinisikan bahasa tersebut atau bukan, kita justru menanyakan $P(S = \text{words})$ berapa probabilitas sebuah kalimat acak akan berupa kata-kata.

Bahasa alami juga ambigu. "He saw her duck" dapat berarti bahwa ia melihat seekor unggas air miliknya, atau bahwa ia melihat bebek itu bergerak untuk menghindari sesuatu. Jadi, sekali lagi, kita tidak dapat berbicara tentang makna tunggal untuk sebuah kalimat, melainkan distribusi probabilitas atas kemungkinan makna. Terakhir, bahasa alami sulit untuk dipahami karena ukurannya yang sangat besar, dan terus berubah. Dengan demikian, model bahasa kita, paling banter, hanyalah sebuah perkiraan. Kita mulai dengan perkiraan yang paling sederhana dan melanjutkan dari sana.

Model karakter N-gram

Pada akhirnya, teks tertulis terdiri dari karakter huruf, angka, tanda baca, dan spasi dalam bahasa Inggris (dan karakter yang lebih eksotis dalam beberapa bahasa lain). Dengan demikian, salah satu model bahasa yang paling sederhana adalah distribusi probabilitas atas rangkaian karakter. Penulis menulis $P(c_{1:N})$ untuk probabilitas rangkaian karakter N , c_1 hingga c_N . Dalam satu koleksi Web, $P("the") = 0,027$ dan $P("zgq") = 0,000000002$.

Rangkaian simbol tertulis dengan panjang n disebut n -gram (dari akar kata Yunani untuk tulisan atau huruf), dengan kasus khusus "unigram" untuk 1-gram, "bigram" untuk 2-gram, dan "trigram" untuk 3-gram. Model distribusi probabilitas rangkaian n -huruf dengan demikian disebut model n -gram. (Namun hati-hati: kita dapat memiliki model n -gram atas rangkaian kata, suku kata, atau unit lain; bukan hanya atas karakter.) Model n -gram didefinisikan sebagai rantai Markov dengan orde $n - 1$. Probabilitas karakter c_i hanya bergantung pada karakter yang mendahuluinya, bukan pada karakter lain. Jadi, dalam model trigram (rantai Markov dengan orde 2), kita memiliki

$$P(c_i | c_{1:i-1}) = P(c_i | c_{i-2:i-1})$$

Kita dapat mendefinisikan probabilitas urutan karakter $P(c_{1:N})$ di bawah model trigram dengan terlebih dahulu melakukan faktorisasi dengan aturan rantai dan kemudian menggunakan asumsi Markov:

$$P(c_{1:N}) = \prod_{i=1}^N P(c_i | c_{1:i-1}) = \prod_{i=1}^N P(c_i | c_{i-2:i-1})$$

Untuk model karakter trigram dalam bahasa dengan 100 karakter, $P(c_i | c_{i-2:i-1})$ memiliki satu juta entri, dan dapat diperkirakan secara akurat dengan menghitung urutan karakter dalam teks yang terdiri dari 10 juta karakter atau lebih. Kita menyebut teks sebagai korpus (jamaknya corpora), dari kata Latin yang berarti tubuh.

Apa yang bisa kita lakukan dengan model karakter n -gram? Salah satu tugas yang paling sesuai adalah identifikasi bahasa: dengan sebuah teks, tentukan bahasa alami penulisannya. Ini tugas yang relatif mudah; bahkan dengan teks pendek seperti "Hello, world" atau "Wie geht es dir," mudah untuk mengidentifikasi yang pertama sebagai bahasa Inggris dan yang kedua sebagai bahasa Jerman. Sistem komputer mengidentifikasi bahasa dengan akurasi lebih dari 99%; terkadang, bahasa yang berkerabat dekat, seperti bahasa Swedia dan Norwegia, tertukar.

Salah satu pendekatan untuk identifikasi bahasa adalah dengan terlebih dahulu membangun model karakter trigram untuk setiap kandidat bahasa, $P(C_i | C_{i-2:i-1}, \ell)$ dengan variabel ℓ berkisar di antara bahasa-bahasa tersebut. Untuk setiap ℓ , model dibangun dengan menghitung trigram dalam korpus bahasa tersebut. (Diperlukan sekitar 100.000 karakter untuk setiap bahasa.) Hal ini memberi kita model $P(\text{Teks} | \text{Language})$, tetapi kita ingin memilih bahasa yang paling mungkin berdasarkan teks tersebut, jadi kita menerapkan aturan Bayes diikuti dengan asumsi Markov untuk mendapatkan bahasa yang paling mungkin:

$$\begin{aligned} l^* &= \operatorname{argmax}_l P(\ell | c_{1:N}) \\ &= \operatorname{argmax}_l P(\ell) P(c_{1:N} | \ell) \\ &= \operatorname{argmax}_l P(\ell) \prod_{i=1}^N P(c_i | c_{i-2:i-1}, \ell) \end{aligned}$$

Model trigram dapat dipelajari dari korpus, tetapi bagaimana dengan probabilitas prior $P(\ell)$? Kita mungkin memiliki beberapa estimasi untuk nilai-nilai ini; misalnya, jika kita memilih halaman web acak, kita tahu bahwa bahasa Inggris adalah bahasa yang paling mungkin dan probabilitas bahasa Makedonia akan kurang dari 1%. Jumlah pasti yang kita pilih untuk prior ini tidaklah penting karena model trigram biasanya memilih satu bahasa yang beberapa kali lipat lebih mungkin daripada bahasa lainnya.

Tugas lain untuk model karakter meliputi koreksi ejaan, klasifikasi genre, dan pengenalan entitas bernama. Klasifikasi genre berarti memutuskan apakah suatu teks merupakan berita, dokumen hukum, artikel ilmiah, dll. Meskipun banyak fitur membantu dalam klasifikasi ini, jumlah tanda baca dan fitur n-gram karakter lainnya sangat berperan.

Pengenalan entitas bernama adalah tugas untuk menemukan nama-nama benda dalam suatu dokumen dan memutuskan kelasnya. Misalnya, dalam teks "Mr. Sopersteen diberi resep aciphex," kita harus mengenali bahwa "Mr. Sopersteen" adalah nama orang dan "aciphex" adalah nama obat. Model tingkat karakter baik untuk tugas ini karena dapat mengaitkan urutan karakter "ex_" ("ex" diikuti spasi) dengan nama obat dan "steen_" dengan nama orang, dan dengan demikian mengidentifikasi kata-kata yang belum pernah dilihat sebelumnya.

Pemulusan Model N-gram

Komplikasi utama model n-gram adalah korpus pelatihan hanya memberikan estimasi distribusi probabilitas yang sebenarnya. Untuk urutan karakter umum seperti "_th", korpus bahasa Inggris mana pun akan memberikan estimasi yang baik: sekitar 1,5% dari semua trigram. Di sisi lain, "_ht" sangat jarang tidak ada kata dalam kamus yang dimulai dengan ht. Kemungkinan urutan tersebut akan memiliki hitungan nol dalam korpus pelatihan bahasa Inggris standar. Apakah itu berarti kita harus menetapkan $P("_th") = 0$? Jika kita melakukannya, maka teks "Program mengeluarkan permintaan http" akan memiliki probabilitas bahasa Inggris nol, yang tampaknya salah.

Kita memiliki masalah dalam generalisasi: kita ingin model bahasa kita menggeneralisasi dengan baik ke teks yang belum mereka lihat. Hanya karena kita belum pernah melihat "http" sebelumnya tidak berarti bahwa model kita harus mengklaim bahwa itu tidak mungkin. Dengan demikian, kita akan menyesuaikan model bahasa kita sehingga urutan yang memiliki hitungan nol dalam korpus pelatihan akan diberi probabilitas bukan nol kecil (dan hitungan lainnya akan disesuaikan sedikit ke bawah sehingga probabilitasnya masih berjumlah 1). Proses penyesuaian probabilitas hitungan frekuensi rendah disebut penghalusan.

Jenis penghalusan paling sederhana disarankan oleh Pierre-Simon Laplace pada abad ke-18: ia mengatakan bahwa, karena kurangnya informasi lebih lanjut, jika variabel Boolean acak X salah dalam semua n pengamatan sejauh ini, maka estimasi untuk $P(X = \text{True})$ seharusnya $1/(n + 2)$. Artinya, ia berasumsi bahwa dengan dua percobaan lagi, satu mungkin benar dan satu salah.

Penghalusan Laplace (juga disebut penghalusan tambah-satu) adalah langkah ke arah yang benar, tetapi kinerjanya relatif buruk. Pendekatan yang lebih baik adalah model backoff, di mana kita mulai dengan memperkirakan jumlah n -gram, tetapi untuk setiap urutan tertentu yang memiliki jumlah rendah (atau nol), kita mundur ke $(n - 1)$ -gram. Penghalusan interpolasi linier adalah model backoff yang menggabungkan model trigram, bigram, dan unigram dengan interpolasi linier. Ini mendefinisikan estimasi probabilitas sebagai:

$$\hat{P}(c_i | c_{i-2:i-1}) = \lambda_3 P(c_i | c_{i-2:i-1}) + \lambda_2 P(c_i | c_{i-1}) + \lambda_1 P(c_i)$$

di mana $\lambda_3 + \lambda_2 + \lambda_1 = 1$. Nilai parameter λ_i dapat ditetapkan, atau dapat dilatih dengan algoritma ekspektasi-maksimisasi. Dimungkinkan juga untuk membuat nilai λ_i bergantung pada jumlah: jika kita memiliki jumlah trigram yang tinggi, maka kita memberi bobot yang relatif lebih; jika hanya jumlah yang rendah, maka kita memberi bobot lebih pada model bigram dan unigram. Satu kelompok peneliti telah mengembangkan model penghalusan yang semakin canggih, sementara kelompok lain menyarankan untuk mengumpulkan korpus yang lebih besar sehingga model penghalusan yang sederhana pun berfungsi dengan baik. Keduanya mencapai tujuan yang sama: mengurangi varians dalam model bahasa.

Satu komplikasi: perhatikan bahwa ekspresi $P(c_i | c_{i-2:i-1})$ meminta $P(c_1 | c_{-1:0})$ ketika $i = 1$, tetapi tidak ada karakter sebelum c_1 . Kita dapat memperkenalkan karakter buatan, misalnya, mendefinisikan c_0 sebagai karakter spasi atau karakter "mulai teks" khusus. Atau kita dapat kembali menggunakan model Markov orde rendah, yang pada dasarnya mendefinisikan $c_{1:0}$ sebagai deret kosong dan dengan demikian $P(c_1 | c_{-1:0}) = P(c_1)$

Evaluasi Model

Dengan begitu banyak kemungkinan model n -gram—unigram, bigram, trigram, pemulusan interpolasi dengan nilai λ yang berbeda, dll.—bagaimana kita tahu model mana yang harus dipilih? Kita dapat mengevaluasi model dengan validasi silang. Membagi korpus menjadi korpus pelatihan dan korpus validasi. Menentukan parameter model dari data pelatihan. Kemudian mengevaluasi model pada korpus validasi.

Evaluasi dapat berupa metrik khusus tugas, seperti mengukur akurasi identifikasi bahasa. Sebagai alternatif, kita dapat memiliki model kualitas bahasa yang independen terhadap tugas: menghitung probabilitas yang ditetapkan ke korpus validasi oleh model; semakin tinggi probabilitasnya, semakin baik. Metrik ini kurang praktis karena probabilitas korpus yang besar akan menjadi angka yang sangat kecil, dan underflow floating-point menjadi masalah. Cara lain untuk menggambarkan probabilitas suatu deret adalah dengan ukuran yang disebut perplexity, yang didefinisikan sebagai:

$$\text{Perplexity}(c_{1:N}) = P(c_{1:N})^{-\frac{1}{N}}$$

Perplexity dapat dianggap sebagai kebalikan dari probabilitas, yang dinormalisasi berdasarkan panjang sekuens. Perplexity juga dapat dianggap sebagai faktor percabangan rata-rata tertimbang dari suatu model. Misalkan terdapat 100 karakter dalam bahasa kita, dan model kita menyatakan bahwa semuanya memiliki kemungkinan yang sama. Maka, untuk sekuens dengan panjang berapa pun, perplexity-nya akan menjadi 100. Jika beberapa karakter memiliki kemungkinan lebih besar daripada yang lain, dan model mencerminkan hal tersebut, maka model akan memiliki perplexity kurang dari 100.

Model kata N-gram

Sekarang kita beralih ke model n-gram untuk kata, alih-alih karakter. Mekanisme yang sama berlaku untuk model kata dan karakter. Perbedaan utamanya adalah kosakata himpunan simbol yang membentuk korpus dan model lebih besar. Hanya terdapat sekitar 100 karakter di sebagian besar bahasa, dan terkadang kita membangun model karakter yang bahkan lebih terbatas, misalnya dengan memperlakukan "A" dan "a" sebagai simbol yang sama atau dengan memperlakukan semua tanda baca sebagai simbol yang sama.

Namun, dengan model kata, kita memiliki setidaknya puluhan ribu simbol, dan terkadang jutaan. Rentang yang luas ini disebabkan oleh ketidakjelasan apa yang membentuk sebuah kata. Dalam bahasa Inggris, rangkaian huruf yang dikelilingi spasi merupakan sebuah kata, tetapi dalam beberapa bahasa, seperti bahasa Mandarin, kata-kata tidak dipisahkan oleh spasi, dan bahkan dalam bahasa Inggris, banyak keputusan harus dibuat untuk memiliki kebijakan yang jelas tentang batasan kata: berapa banyak kata yang berada dalam "ne'er-do-well"? Atau dalam "[Tel:1-800-960-5660x123](tel:1-800-960-5660x123)"?

Model n-gram kata perlu menangani kata-kata yang berada di luar kosakata. Dengan model karakter, kita tidak perlu khawatir tentang seseorang yang menciptakan huruf alfabet baru.¹ Namun, dengan model kata, selalu ada kemungkinan kata baru yang tidak terlihat dalam korpus pelatihan, jadi kita perlu memodelkannya secara eksplisit dalam model bahasa kita. Ini dapat dilakukan dengan menambahkan hanya satu kata baru ke dalam kosakata: <UNK>, yang mewakili kata yang tidak diketahui.

Kita dapat memperkirakan jumlah n-gram untuk <UNK> dengan trik ini: telusuri korpus pelatihan, dan saat pertama kali sebuah kata muncul, kata tersebut sebelumnya tidak diketahui, jadi gantilah dengan simbol <UNK>. Semua kemunculan kata selanjutnya tetap tidak berubah. Kemudian, hitung jumlah n-gram untuk korpus seperti biasa, perlakukan <UNK>

seperti kata lainnya. Kemudian, ketika sebuah kata yang tidak diketahui muncul dalam set pengujian, kita cari probabilitasnya di bawah <UNK>. Terkadang beberapa simbol kata yang tidak diketahui digunakan, untuk kelas yang berbeda. Misalnya, rangkaian angka apa pun dapat diganti dengan <NUM>, atau alamat email apa pun dengan <EMAIL>.

Untuk mendapatkan gambaran tentang apa yang dapat dilakukan oleh model kata, kami membangun model unigram, bigram, dan trigram berdasarkan kata-kata dalam buku ini, lalu mengambil sampel urutan kata secara acak dari model tersebut. Hasilnya adalah

- ➡ *Unigram*: logika adalah seperti halnya kebingungan, suatu mungkin benar mencoba agen tujuan adalah ...
- ➡ *Bigram*: sistem sangat mirip dengan pendekatan komputasional yang akan direpresentasikan ...
- ➡ *Trigram*: perencanaan dan penjadwalan terintegrasi. Keberhasilan model Bayes Naif adalah ...

Bahkan dengan sampel kecil ini, seharusnya jelas bahwa model unigram merupakan perkiraan yang buruk untuk Bahasa Inggris atau isi buku teks AI, dan bahwa model bigram dan trigram jauh lebih baik. Model-model tersebut sesuai dengan penilaian ini: kebingungan adalah 891 untuk model unigram, 142 untuk model bigram, dan 91 untuk model trigram. Dengan dasar-dasar model n-gram baik berbasis karakter maupun kata yang telah mapan, kita sekarang dapat beralih ke beberapa tugas bahasa.

4.2 KLASIFIKASI TEKS

Sekarang kita akan membahas secara mendalam tugas klasifikasi teks, yang juga dikenal sebagai kategorisasi: jika diberikan suatu teks, kita akan menentukan jenis teks tersebut dari serangkaian kelas yang telah ditentukan sebelumnya. Identifikasi bahasa dan klasifikasi genre adalah contoh klasifikasi teks, begitu pula analisis sentimen (mengklasifikasikan film atau ulasan produk sebagai positif atau negatif) dan deteksi spam (mengklasifikasikan pesan email sebagai spam atau bukan spam). Karena "bukan spam" terdengar aneh, para peneliti menciptakan istilah "ham" untuk bukan spam. Kita dapat menganggap deteksi spam sebagai masalah dalam pembelajaran terawasi. Kumpulan pelatihan tersedia: contoh positif (spam) ada di folder spam saya, contoh negatif (ham) ada di kotak masuk saya. Berikut kutipannya:

- ➡ Spam: Jam Tangan Fashion Grosir -57% hari ini. Jam tangan desainer murah...
- ➡ Spam: Anda dapat membeli **produk ini** mulai dari Rp30.000. Semua obat tersedia dengan harga yang tak tertandingi!...
- ➡ Spam: KAMI BISA MENGOBATI APA PUN YANG ANDA DERITA, PERCAYALAH PADA KAMI ...
- ➡ Spam: Mulailah mendapatkan gaji yang pantas Anda dapatkan dengan mendapatkan kredensial yang tepat!
- ❖ Ham: Signifikansi praktis lebar hypertree dalam mengidentifikasi lebih banyak ...
- ❖ Ham: Abstrak: Kami akan memotivasi masalah pengelompokan identitas sosial: ...
- ❖ Ham: Senang bertemu denganmu, temanku. Hai Peter, senang mendengar kabarmu. ...

- ❖ Ham: PDS menyiratkan konveksitas dari masalah optimasi yang dihasilkan (Kernel Ridge ...

Dari kutipan ini, kita dapat mulai mendapatkan gambaran tentang fitur-fitur apa saja yang mungkin baik untuk disertakan dalam model pembelajaran terbimbing. N-gram kata seperti "murah" dan "Anda dapat membeli" tampaknya menjadi indikator spam (meskipun keduanya juga memiliki probabilitas bukan nol dalam ham).

Fitur tingkat karakter juga tampaknya penting: spam lebih cenderung menggunakan huruf kapital semua dan memiliki tanda baca yang tertanam dalam kata-kata. Rupanya, para pengirim spam berpikir bahwa kata bigram "Anda pantas" akan terlalu menunjukkan spam, sehingga mereka menulis "yo,u d-eserve" sebagai gantinya. Sebuah model karakter seharusnya dapat mendeteksi hal ini. Kita dapat membuat model n-gram karakter lengkap dari spam dan ham, atau kita dapat membuat fitur-fitur seperti "jumlah tanda baca yang tertanam dalam kata-kata."

Perhatikan bahwa kita memiliki dua cara yang saling melengkapi untuk membahas klasifikasi. Dalam pendekatan pemodelan bahasa, kita mendefinisikan satu model bahasa n-gram untuk $P(\text{Message}|\text{spam})$ dengan melatih pada folder spam, dan satu model untuk $P(\text{Message}|\text{ham})$ dengan pelatihan di kotak masuk. Kemudian, kita dapat mengklasifikasikan pesan baru dengan penerapan aturan Bayes:

$$\operatorname{argmax}_{c \in (\text{spam}, \text{ham})} P(c|\text{message}) = \operatorname{argmax}_{c \in (\text{spam}, \text{ham})} P(\text{message}|c) P(c)$$

di mana $P(c)$ diperkirakan hanya dengan menghitung jumlah total pesan spam dan ham. Pendekatan ini efektif untuk deteksi spam, sama seperti untuk identifikasi bahasa.

Dalam pendekatan pembelajaran mesin, kami merepresentasikan pesan sebagai sekumpulan pasangan fitur/nilai dan menerapkan algoritma klasifikasi h ke vektor fitur X . Kita dapat membuat pendekatan pemodelan bahasa dan pembelajaran mesin kompatibel dengan menganggap n-gram sebagai fitur. Ini paling mudah dilihat dengan model unigram. Fitur adalah kata-kata dalam kosakata: "a," "aardvark," .. ., dan nilainya adalah berapa kali setiap kata muncul dalam pesan. Itu membuat vektor fitur besar dan jarang.

Jika ada 100.000 kata dalam model bahasa, maka vektor fitur memiliki panjang 100.000, tetapi untuk pesan email pendek hampir semua fitur akan memiliki hitungan nol. Representasi unigram ini disebut model kantong kata. Anda dapat menganggap model tersebut seperti memasukkan kata-kata dari korpus pelatihan ke dalam kantong dan kemudian memilih kata satu per satu. Gagasan urutan kata hilang; model unigram memberikan probabilitas yang sama untuk setiap permutasi teks. Model n-gram orde tinggi mempertahankan beberapa gagasan lokal tentang urutan kata.

Dengan bigram dan trigram, jumlah fitur dikuadratkan atau dipangkatkan, dan kita dapat menambahkan fitur non-n-gram lainnya: waktu pengiriman pesan, apakah URL atau gambar merupakan bagian dari pesan, nomor ID pengirim pesan, jumlah pesan spam dan ham sebelumnya dari pengirim, dan sebagainya. Pemilihan fitur merupakan bagian terpenting

dalam menciptakan detektor spam yang baik—lebih penting daripada pemilihan algoritma untuk memproses fitur. Hal ini sebagian disebabkan oleh banyaknya data pelatihan, sehingga jika kita dapat mengusulkan suatu fitur, data tersebut dapat secara akurat menentukan apakah fitur tersebut baik atau tidak. Pembaruan fitur perlu dilakukan secara terus-menerus, karena deteksi spam merupakan tugas yang bersifat adversarial; spammer memodifikasi spam mereka sebagai respons terhadap perubahan detektor spam.

Menjalankan algoritma pada vektor fitur yang sangat besar dapat memakan biaya besar, sehingga seringkali proses pemilihan fitur digunakan untuk hanya mempertahankan fitur yang paling baik dalam membedakan antara spam dan ham. Misalnya, bigram "of the" sering muncul dalam bahasa Inggris, dan mungkin sama seringnya muncul dalam spam dan ham, jadi tidak masuk akal untuk menghitungnya. Seringkali, sekitar seratus fitur teratas mampu membedakan kelas dengan baik.

Setelah memilih serangkaian fitur, kita dapat menerapkan salah satu teknik pembelajaran terawasi yang telah kita lihat; teknik populer untuk kategorisasi teks meliputi k-nearest-neighbors, mesin vektor pendukung, pohon keputusan, Bayes Naif, dan regresi logistik. Semua ini telah diterapkan untuk deteksi spam, biasanya dengan akurasi dalam kisaran 98%–99%. Dengan serangkaian fitur yang dirancang dengan cermat, akurasi dapat melebihi 99,9%.

Klasifikasi dengan kompresi data

Cara lain untuk memandang klasifikasi adalah sebagai masalah dalam kompresi data. Algoritma kompresi lossless mengambil serangkaian simbol, mendeteksi pola berulang di dalamnya, dan menulis deskripsi rangkaian yang lebih ringkas daripada aslinya. Misalnya, teks "0.142857142857142857" dapat dikompresi menjadi "0.[142857]*3." Algoritma kompresi bekerja dengan membangun kamus subsekuens teks, lalu merujuk ke entri dalam kamus. Contoh di sini hanya memiliki satu entri kamus, "142857."

Sebenarnya, algoritma kompresi menciptakan sebuah model bahasa. Algoritma LZW khususnya secara langsung memodelkan distribusi probabilitas entropi maksimum. Untuk melakukan klasifikasi dengan kompresi, pertama-tama kita mengelompokkan semua pesan pelatihan spam dan mengompresnya sebagai satu unit. Kita melakukan hal yang sama untuk ham. Kemudian, ketika diberikan pesan baru untuk diklasifikasikan, kita menambahkannya ke pesan spam dan mengompres hasilnya. Kita juga menambahkannya ke ham dan mengompresnya. Kelas mana pun yang terkompresi lebih baik menambahkan lebih sedikit byte tambahan untuk pesan baru adalah kelas yang diprediksi. Idennya adalah bahwa pesan spam akan cenderung berbagi entri kamus dengan pesan spam lainnya dan dengan demikian akan terkompresi lebih baik ketika ditambahkan ke koleksi yang sudah berisi kamus spam.

Eksperimen klasifikasi berbasis kompresi pada beberapa korpus standar untuk klasifikasi teks set data 20-Newsgroups, Korpus Reuters-10, dan korpus Sektor Industri menunjukkan bahwa meskipun menjalankan algoritma kompresi siap pakai seperti gzip, RAR, dan LZW bisa sangat lambat, akurasinya sebanding dengan algoritma klasifikasi tradisional. Hal ini menarik, dan juga menunjukkan bahwa terdapat potensi untuk algoritma yang

menggunakan n-gram karakter secara langsung tanpa pra-pemrosesan teks atau pemilihan fitur: algoritma tersebut tampaknya menangkap beberapa pola nyata.

4.3 PENCARIAN INFORMASI

Pencarian informasi adalah tugas menemukan dokumen yang relevan dengan kebutuhan informasi pengguna. Contoh sistem pencarian informasi yang paling terkenal adalah mesin pencari di World Wide Web. Pengguna web dapat mengetikkan kueri seperti [AI book] ke dalam mesin pencari dan melihat daftar halaman yang relevan. Di bagian ini, kita akan melihat bagaimana sistem tersebut dibangun. Sistem pencarian informasi (selanjutnya disebut IR) dapat dicirikan oleh

1. **Korpus dokumen.** Setiap sistem harus memutuskan apa yang ingin diperlakukan sebagai dokumen: paragraf, halaman, atau teks multihalaman.
2. **Kueri yang diajukan dalam bahasa kueri.** Kueri menentukan apa yang ingin diketahui pengguna. Bahasa kueri dapat berupa daftar kata, seperti [AI book]; atau dapat menentukan frasa kata yang harus berdekatan, seperti dalam ["AI book"]; dapat berisi operator Boolean seperti dalam [AI AND book]; dapat mencakup operator non-Boolean seperti [AI NEAR book] atau [AI book site:www.aaai.org].
3. **Kumpulan hasil.** Ini adalah subkumpulan dokumen yang dinilai relevan oleh sistem IR dengan kueri. Yang kami maksud dengan relevan adalah kemungkinan berguna bagi orang yang mengajukan kueri, untuk kebutuhan informasi tertentu yang diungkapkan dalam kueri tersebut.
4. **Presentasi kumpulan hasil.** Ini bisa sesederhana daftar judul dokumen yang diurutkan atau serumit peta warna berputar dari kumpulan hasil yang diproyeksikan ke ruang tiga dimensi, yang ditampilkan sebagai tampilan dua dimensi.

Sistem IR paling awal bekerja pada model kata kunci Boolean. Setiap kata dalam kumpulan dokumen diperlakukan sebagai fitur Boolean yang bernilai true untuk suatu dokumen jika kata tersebut muncul dalam dokumen dan false jika tidak muncul. Jadi, "pengambilan" fitur bernilai true untuk bab saat ini tetapi false. Bahasa kueri adalah bahasa ekspresi Boolean atas fitur. Suatu dokumen hanya relevan jika ekspresi tersebut bernilai true. Misalnya, kueri [informasi DAN pengambilan] bernilai benar.

Model ini memiliki keuntungan karena mudah dijelaskan dan diimplementasikan. Namun, model ini juga memiliki beberapa kelemahan. Pertama, tingkat relevansi suatu dokumen hanya satu bit, sehingga tidak ada panduan tentang cara mengurutkan dokumen yang relevan untuk presentasi. Kedua, ekspresi Boolean kurang familiar bagi pengguna yang bukan programmer atau ahli logika. Pengguna merasa kurang intuitif ketika ingin mengetahui tentang pertanian di negara bagian Kansas dan Nebraska, mereka perlu mengeluarkan kueri [pertanian (Kansas ATAU Nebraska)]. Ketiga, merumuskan kueri yang tepat bisa jadi sulit, bahkan bagi pengguna yang terampil. Misalkan kita mencoba [informasi DAN pengambilan DAN model DAN optimasi] dan mendapatkan hasil yang kosong. Kita bisa mencoba [informasi ATAU pengambilan ATAU model ATAU optimasi], tetapi jika hasilnya terlalu banyak, sulit untuk mengetahui apa yang harus dicoba selanjutnya.

Fungsi Penilaian IR

Sebagian besar sistem IR telah meninggalkan model Boolean dan menggunakan model yang berbasis pada statistik jumlah kata. Kami menjelaskan fungsi penilaian BM25, yang berasal dari proyek Okapi Stephen Robertson dan Karen Sparck Jones di City College London, dan telah digunakan di mesin pencari seperti proyek Lucene sumber terbuka. Fungsi penilaian mengambil dokumen dan kueri dan mengembalikan skor numerik; dokumen yang paling relevan memiliki skor tertinggi. Dalam fungsi BM25, skor merupakan kombinasi skor terbobot linear untuk setiap kata yang membentuk kueri. Tiga faktor memengaruhi bobot istilah kueri: Pertama, frekuensi kemunculan istilah kueri dalam dokumen (juga dikenal sebagai TF untuk frekuensi istilah).

Untuk kueri [bertani di Kansas], dokumen yang sering menyebutkan "bertani" akan memiliki skor yang lebih tinggi. Kedua, frekuensi dokumen terbalik dari istilah tersebut, atau IDF. Kata "in" muncul di hampir setiap dokumen, sehingga memiliki frekuensi dokumen yang tinggi, dan dengan demikian frekuensi dokumen terbalik yang rendah, sehingga tidak sepenting "farming" atau "Kansas" dalam kueri. Ketiga, panjang dokumen. Dokumen sejuta kata mungkin akan menyebutkan semua kata kueri, tetapi mungkin tidak benar-benar membahas kueri tersebut. Dokumen pendek yang menyebutkan semua kata adalah kandidat yang jauh lebih baik.

Fungsi BM25 memperhitungkan ketiga hal ini. Kami berasumsi bahwa kami telah membuat indeks dari N dokumen dalam korpus sehingga kami dapat mencari $TF(q_i, d_j)$, jumlah kemunculan kata q_i dalam dokumen d_j . Kami juga mengasumsikan tabel jumlah frekuensi dokumen, $DF(q_i)$, yang memberikan jumlah dokumen yang berisi kata q_i . Kemudian, dengan diberikan dokumen d_j dan kueri yang terdiri dari kata $q_{1:N}$, kami memiliki

$$BM25(d_{j,q_{1:N}}) = \sum_{i=1}^N IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k + 1)}{TF(q_i, d_j) + k \cdot \left(1 - b + b \cdot \frac{|d_j|}{L}\right)}$$

di mana $|d_j|$ adalah panjang dokumen d_j dalam kata, dan L adalah rata-rata panjang dokumen dalam korpus: $L = \sum_i |d_i| / N$. Kami memiliki dua parameter, k dan b , yang dapat disetel melalui validasi silang; nilai tipikal adalah $k = 2,0$ dan $b = 0,75$. $IDF(q_i)$ adalah frekuensi dokumen invers dari kata q_i , yang diberikan oleh

$$IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

Tentu saja, menerapkan fungsi penilaian BM25 pada setiap dokumen dalam korpus tidaklah praktis. Sebagai gantinya, sistem membuat indeks terlebih dahulu yang mencantumkan, untuk setiap kosakata, dokumen-dokumen yang mengandung kata tersebut. Indeks ini disebut daftar hit untuk kata tersebut. Kemudian, ketika diberikan kueri, kami memotong daftar hit kata-kata

kueri tersebut dan hanya menilai dokumen-dokumen yang berada di dalam perpotongan tersebut.

Evaluasi Sistem IR

Bagaimana kita mengetahui apakah sistem IR berkinerja baik? Kami melakukan eksperimen di mana sistem diberikan serangkaian kueri dan rangkaian hasilnya dinilai berdasarkan penilaian relevansi manusia. Secara tradisional, terdapat dua ukuran yang digunakan dalam penilaian: recall dan precision. Kami menjelaskannya dengan bantuan sebuah contoh. Bayangkan sebuah sistem IR telah mengembalikan rangkaian hasil untuk satu kueri, yang mana kita tahu dokumen mana yang relevan dan tidak, dari korpus yang berisi 100 dokumen. Jumlah dokumen dalam setiap kategori diberikan dalam tabel berikut:

	Dalam Himpunan Hasil	Tidak dalam Himpunan Hasil
<i>Relevan</i>	30	20
<i>Tidak Relevan</i>	10	40

Presisi mengukur proporsi dokumen dalam kumpulan hasil yang benar-benar relevan. Dalam contoh kita, presisinya adalah $30/(30 + 10) = 0,75$. Rasio positif palsu adalah $1 - 0,75 = 0,25$. Recall mengukur proporsi semua dokumen relevan dalam kumpulan yang ada dalam kumpulan hasil. Dalam contoh kita, recall adalah $30/(30 + 20) = 0,60$. Rasio negatif palsu adalah $1 - 0,60 = 0,40$.

Dalam kumpulan dokumen yang sangat besar, seperti World Wide Web, recall sulit dihitung karena tidak ada cara mudah untuk memeriksa relevansi setiap halaman di Web. Yang bisa kita lakukan hanyalah memperkirakan recall dengan mengambil sampel atau mengabaikan recall sepenuhnya dan hanya menilai presisi. Dalam kasus mesin pencari web, mungkin terdapat ribuan dokumen dalam kumpulan hasil, sehingga lebih masuk akal untuk mengukur presisi untuk beberapa ukuran berbeda, seperti "P@10" (presisi dalam 10 hasil teratas) atau "P@50", daripada memperkirakan presisi di seluruh kumpulan hasil.

Presisi dapat dipertukarkan dengan perolehan kembali (recall) dengan memvariasikan ukuran kumpulan hasil yang dikembalikan. Dalam kasus ekstrem, sistem yang mengembalikan setiap dokumen dalam kumpulan dokumen dijamin perolehan kembali 100%, tetapi presisinya akan rendah. Sebagai alternatif, sistem dapat mengembalikan satu dokumen dan memiliki perolehan kembali yang rendah, tetapi peluang presisi 100% cukup besar. Ringkasan dari kedua ukuran tersebut adalah skor F_1 , sebuah angka tunggal yang merupakan rata-rata harmonis presisi dan perolehan kembali, $2PR/(P + R)$.

Penyempurnaan IR

Ada banyak kemungkinan penyempurnaan untuk sistem yang dijelaskan di sini, dan memang mesin pencari web terus memperbarui algoritma mereka seiring ditemukannya pendekatan baru dan seiring pertumbuhan serta perubahan web. Salah satu penyempurnaan umum adalah model yang lebih baik tentang pengaruh panjang dokumen terhadap relevansi. Singhal dkk. (1996) mengamati bahwa skema normalisasi panjang dokumen sederhana cenderung terlalu mengutamakan dokumen pendek dan kurang mengutamakan dokumen

panjang. Mereka mengusulkan skema normalisasi panjang dokumen yang dipivot; idenya adalah bahwa pivot adalah panjang dokumen di mana normalisasi gaya lama sudah tepat; dokumen yang lebih pendek dari itu mendapatkan peningkatan dan yang lebih panjang mendapatkan penalti.

Fungsi penilaian BM25 menggunakan model kata yang memperlakukan semua kata sebagai sepenuhnya independen, tetapi kita tahu bahwa beberapa kata berkorelasi: "couch" berkaitan erat dengan "couches" dan "sofa." Banyak sistem IR mencoba memperhitungkan korelasi ini. Misalnya, jika kuerinya adalah [couch], akan sangat disayangkan untuk mengecualikan dari kumpulan hasil dokumen-dokumen yang menyebutkan "COUCH" atau "couches" tetapi bukan "couch." Sebagian besar sistem IR melakukan pelipatan huruf besar-kecil dari "COUCH" menjadi "couch," dan beberapa menggunakan algoritma stemming untuk mereduksi "couches" menjadi bentuk stem "couch," baik dalam kueri maupun dokumen. Ini biasanya menghasilkan sedikit peningkatan dalam recall (sekitar 2% untuk bahasa Inggris).

Namun, hal ini dapat merusak presisi. Misalnya, stemming "stocking" menjadi "stock" akan cenderung menurunkan presisi untuk kueri tentang penutup kaki atau instrumen keuangan, meskipun dapat meningkatkan recall untuk kueri tentang pergudangan. Algoritma stemming berdasarkan aturan (misalnya, hapus "-ing") tidak dapat menghindari masalah ini, tetapi algoritma berdasarkan kamus (jangan hapus "-ing" jika kata tersebut sudah tercantum dalam kamus) dapat.

Sementara stemming memiliki efek kecil dalam bahasa Inggris, itu lebih penting dalam bahasa lain. Dalam bahasa Jerman, misalnya, tidak jarang melihat kata-kata seperti "Lebensversicherungsgesellschaftsangestellter" (karyawan perusahaan asuransi jiwa). Bahasa seperti Finlandia, Turki, Inuit, dan Yupik memiliki aturan morfologi rekursif yang pada prinsipnya menghasilkan kata-kata dengan panjang yang tidak terbatas. Langkah selanjutnya adalah mengenali sinonim, seperti "sofa" untuk "sofa". Seperti halnya stemming, hal ini berpotensi meningkatkan ingatan, tetapi dapat mengurangi presisi. Pengguna yang mengajukan kueri [Tim Couch] ingin melihat hasil tentang pemain sepak bola, bukan sofa. Masalahnya adalah "bahasa membenci sinonim absolut sebagaimana alam membenci kekosongan". Artinya, setiap kali ada dua kata yang memiliki arti yang sama, penutur bahasa tersebut berkonspirasi untuk mengembangkan maknanya guna menghilangkan kebingungan.

Kata-kata terkait yang bukan sinonim juga memainkan peran penting dalam pemeringkatan—istilah seperti "kulit", "kayu", atau "modern" dapat berfungsi untuk mengonfirmasi bahwa dokumen tersebut benar-benar membahas "sofa". Sinonim dan kata-kata terkait dapat ditemukan di kamus atau dengan mencari korelasi dalam dokumen atau kueri—jika kami menemukan bahwa banyak pengguna yang mengajukan kueri [sofa baru] menindaklanjutinya dengan kueri [sofa baru], kami dapat mengubah [sofa baru] menjadi [sofa baru ATAU sofa baru] di kemudian hari. Sebagai penyempurnaan terakhir, IR dapat ditingkatkan dengan mempertimbangkan metadata—data di luar teks dokumen. Contohnya termasuk kata kunci yang disediakan manusia dan data publikasi. Di Web, tautan hiperteks antar dokumen merupakan sumber informasi yang krusial.

Algoritma PageRank

PageRank3 adalah salah satu dari dua ide awal yang membedakan pencarian Google dari mesin pencari web lainnya ketika diperkenalkan pada tahun 1997. (Inovasi lainnya adalah penggunaan teks jangkar teks yang digarisbawahi dalam hipertaut untuk mengindeks halaman, meskipun teks jangkar tersebut berada di halaman yang berbeda dari halaman yang diindeks.) PageRank diciptakan untuk memecahkan masalah tirani skor TF: jika kuerinya adalah [IBM], bagaimana kita memastikan bahwa halaman beranda IBM, *ibm.com*, adalah hasil pertama, meskipun halaman lain menyebutkan istilah "IBM" lebih sering? Idenya adalah *ibm.com* memiliki banyak tautan masuk (tautan ke halaman), sehingga peringkatnya seharusnya lebih tinggi: setiap tautan masuk merupakan suara untuk kualitas halaman yang ditautkan.

```
function HITS(query) returns pages with hub and authority numbers

pages ← EXPAND – PAGES(RELEVANT – PAGES(query))
for each p in pages do
  p.AUTHORITY ← 1
  p.HUB ← 1
  repeat until convergence do
    for each p in pages do
      p.AUTHORITY ←  $\sum_i \text{INLINK}_i(p). \text{HUB}$ 
      p.HUB ←  $\sum_i \text{OUTLINK}_i(p). \text{AUTHORITY}$ 
    NORMALIZE(pages)
  return pages
```

Gambar 4.1 Algoritma HITS untuk menghitung hub dan otoritas terkait suatu kueri.

RELEVANT-PAGES mengambil halaman yang sesuai dengan kueri, dan EXPAND-PAGES menambahkan setiap halaman yang tertaut atau tertaut dari salah satu halaman yang relevan. NORMALIZE membagi skor setiap halaman dengan jumlah kuadrat skor semua halaman (terpisah untuk skor otoritas dan hub).

Namun, jika kita hanya menghitung tautan masuk, maka spammer web dapat membuat jaringan halaman dan mengarahkan semuanya ke halaman pilihannya, sehingga meningkatkan skor halaman tersebut. Oleh karena itu, algoritma PageRank dirancang untuk memberi bobot lebih besar pada tautan dari situs berkualitas tinggi. Apa itu situs berkualitas tinggi? Situs yang ditautkan oleh situs berkualitas tinggi lainnya. Definisinya rekursif, tetapi kita akan melihat bahwa rekursi tersebut mencapai titik terendah dengan tepat. PageRank untuk halaman p didefinisikan sebagai:

$$PR(p) = \frac{1-d}{N} + d \sum_i \frac{PR(in_i)}{C(in_i)}$$

di mana $PR(p)$ adalah PageRank dari halaman p , N adalah jumlah total halaman dalam korpus, ini adalah halaman yang tertaut ke p , dan $C(in_i)$ adalah jumlah total tautan keluar pada halaman in_i . Konstanta d adalah faktor peredam. Hal ini dapat dipahami melalui model peselancar acak: bayangkan seorang peselancar Web yang memulai di beberapa halaman acak dan mulai menjelajah.

Dengan probabilitas d (kita asumsikan $d = 0,85$) peselancar mengklik salah satu tautan di halaman (memilih secara seragam di antara mereka), dan dengan probabilitas $1 - d$ dia bosan dengan halaman tersebut dan memulai kembali pada halaman acak di mana saja di Web. PageRank dari halaman p adalah probabilitas bahwa peselancar acak akan berada di halaman p pada suatu titik waktu. PageRank dapat dihitung dengan prosedur berulang: mulai dengan semua halaman yang memiliki $PR(p) = 1$, dan ulangi algoritma, perbarui peringkat hingga konvergen.

Algoritma HITS

Algoritma Pencarian Topik yang Diinduksi Hyperlink, juga dikenal sebagai "Hub dan Otoritas" atau HITS, adalah algoritma analisis tautan berpengaruh lainnya HITS berbeda dari PageRank dalam beberapa hal. Pertama, HITS merupakan ukuran yang bergantung pada kueri: HITS memeringkat halaman berdasarkan kueri tersebut. Artinya, HITS harus dihitung ulang untuk setiap kueri—beban komputasi yang tidak ditanggung oleh sebagian besar mesin pencari.

Berdasarkan suatu kueri, HITS pertama-tama menemukan sekumpulan halaman yang relevan dengan kueri tersebut. HITS melakukannya dengan memotong daftar kata kunci kueri, lalu menambahkan halaman di sekitar tautan halaman-halaman ini halaman yang tertaut ke atau ditautkan dari salah satu halaman dalam kumpulan relevan asli. Setiap halaman dalam kumpulan ini dianggap sebagai otoritas pada kueri tersebut sejauh halaman lain dalam kumpulan relevan tersebut mengarah ke halaman tersebut. Suatu halaman dianggap sebagai hub sejauh halaman tersebut mengarah ke halaman otoritatif lain dalam kumpulan relevan tersebut. Layaknya PageRank, kami tidak hanya ingin menghitung jumlah tautan; kami ingin memberikan nilai lebih kepada hub dan otoritas berkualitas tinggi.

Oleh karena itu, seperti halnya PageRank, kami mengulangi proses yang memperbarui skor otoritas suatu halaman menjadi jumlah skor hub dari halaman-halaman yang mengarah ke halaman tersebut, dan skor hub menjadi jumlah skor otoritas dari halaman-halaman yang ditujunya. Jika kami kemudian menormalkan skor dan mengulanginya sebanyak k kali, prosesnya akan konvergen. PageRank dan HITS memainkan peran penting dalam mengembangkan pemahaman kita tentang pengambilan informasi web. Algoritme ini dan ekstensinya digunakan untuk memeringkat miliaran kueri setiap hari seiring mesin pencari terus mengembangkan cara yang lebih baik untuk mengekstraksi sinyal relevansi pencarian yang lebih akurat.

Menjawab pertanyaan

Pencarian informasi adalah tugas menemukan dokumen yang relevan dengan suatu kueri, yang mana kueri tersebut dapat berupa pertanyaan, atau sekadar area topik atau konsep. Menjawab pertanyaan adalah tugas yang agak berbeda, di mana kueri benar-benar

sebuah pertanyaan, dan jawabannya bukanlah daftar dokumen yang diurutkan, melainkan respons singkat sebuah kalimat, atau bahkan sekadar frasa. Sistem NLP (pemrosesan bahasa alami) untuk menjawab pertanyaan telah ada sejak tahun 1960-an, tetapi baru sejak tahun 2001 sistem tersebut menggunakan pengambilan informasi Web untuk secara radikal meningkatkan cakupannya.

Sistem ASKMSR adalah sistem tanya jawab berbasis Web yang umum. Sistem ini didasarkan pada intuisi bahwa sebagian besar pertanyaan akan dijawab berkali-kali di Web, sehingga menjawab pertanyaan harus dianggap sebagai masalah presisi, bukan ingatan. Kita tidak harus berurusan dengan semua cara berbeda untuk merumuskan jawaban kita hanya perlu menemukan salah satunya. Misalnya, perhatikan kueri [Siapa yang membunuh Abraham Lincoln?]. Misalkan suatu sistem harus menjawab pertanyaan itu hanya dengan akses ke satu ensiklopedia, yang entri tentang Lincoln-nya mengatakan John Wilkes Booth mengubah sejarah dengan peluru. Ia akan selamanya dikenal sebagai orang yang mengakhiri hidup Abraham Lincoln.

Untuk menggunakan bagian ini guna menjawab pertanyaan, sistem harus mengetahui bahwa mengakhiri hidup seseorang bisa dianggap sebagai pembunuhan, bahwa "Dia" merujuk pada Booth, dan beberapa fakta linguistik dan semantik lainnya. ASKMSR tidak mencoba kecanggihan semacam ini ia tidak tahu apa-apa tentang referensi kata ganti, atau tentang membunuh, atau kata kerja lainnya. Ia mengetahui 15 jenis pertanyaan yang berbeda, dan bagaimana mereka dapat ditulis ulang sebagai kueri ke mesin pencari. Ia tahu bahwa [Siapa yang membunuh Abraham Lincoln] dapat ditulis ulang sebagai kueri [* membunuh Abraham Lincoln] dan sebagai [Abraham Lincoln dibunuh oleh *]. Ia mengeluarkan kueri yang ditulis ulang ini dan memeriksa hasil yang kembali—bukan halaman Web lengkap, hanya ringkasan teks pendek yang muncul di dekat istilah kueri.

Hasilnya dipecah menjadi 1-, 2-, dan 3-gram dan dihitung untuk frekuensi dalam set hasil dan untuk bobot: n-gram yang kembali dari penulisan ulang kueri yang sangat spesifik (seperti kueri pencocokan frasa yang tepat ["Abraham Lincoln dibunuh oleh *"]) akan mendapatkan bobot lebih dari satu dari penulisan ulang kueri umum, seperti [Abraham ATAU Lincoln ATAU dibunuh]. Kami berharap "John Wilkes Booth" akan termasuk di antara n-gram berperingkat tinggi yang diambil, tetapi begitu pula "Abraham Lincoln", "pembunuhan" dan "Teater Ford". Setelah n-gram dinilai, n-gram tersebut difilter berdasarkan tipe yang diharapkan. Jika kueri awal dimulai dengan "siapa", maka kami memfilter berdasarkan nama orang; untuk "berapa banyak", kami memfilter berdasarkan angka, untuk "kapan", berdasarkan tanggal atau waktu. Ada juga filter yang menyatakan bahwa jawaban tersebut tidak boleh menjadi bagian dari pertanyaan; bersama-sama, filter ini akan memungkinkan kami untuk mengembalikan "John Wilkes Booth" (dan bukan "Abraham Lincoln") sebagai respons dengan skor tertinggi.

Dalam beberapa kasus, jawabannya akan lebih panjang dari tiga kata; karena komponen respons hanya mencapai 3-gram, respons yang lebih panjang harus disusun dari potongan-potongan yang lebih pendek. Misalnya, dalam sistem yang hanya menggunakan bigram, jawaban "John Wilkes Booth" dapat disusun dari potongan-potongan dengan skor

tinggi "John Wilkes" dan "Wilkes Booth". Pada Konferensi Evaluasi Pengambilan Teks (TREC), ASKMSR dinilai sebagai salah satu sistem terbaik, mengalahkan pesaingnya dengan kemampuan pemahaman bahasa yang jauh lebih kompleks. ASKMSR mengandalkan keluasan konten di Web, alih-alih kedalaman pemahamannya sendiri. Sistem ini tidak akan mampu menangani pola inferensi yang kompleks seperti mengaitkan "siapa yang membunuh" dengan "mengakhiri kehidupan". Namun, sistem ini tahu bahwa Web begitu luas sehingga mampu mengabaikan bagian-bagian seperti itu dan menunggu bagian sederhana yang dapat ditanganinya.

4.4 EKSTRAKSI INFORMASI

Ekstraksi informasi adalah proses memperoleh pengetahuan dengan membaca sekilas teks dan mencari kemunculan kelas objek tertentu dan hubungan antar objek. Tugas yang umum adalah mengekstrak contoh alamat dari halaman Web, dengan bidang basis data untuk jalan, kota, negara bagian, dan kode pos; atau contoh badai dari laporan cuaca, dengan bidang untuk suhu, kecepatan angin, dan curah hujan.

Dalam domain terbatas, ini dapat dilakukan dengan akurasi tinggi. Karena domain menjadi lebih umum, model linguistik yang lebih kompleks dan teknik pembelajaran yang lebih kompleks diperlukan. Kita akan melihat bagaimana mendefinisikan model bahasa kompleks dari struktur frasa (frasa nomina dan frasa verba) bahasa Inggris. Namun sejauh ini tidak ada model lengkap seperti ini, jadi untuk kebutuhan terbatas ekstraksi informasi, kami mendefinisikan model terbatas yang mendekati model bahasa Inggris lengkap, dan berkonsentrasi hanya pada bagian-bagian yang diperlukan untuk tugas yang ada. Model-model yang kami jelaskan di bagian ini merupakan aproksimasi, sama seperti model logika 1-CNF sederhana merupakan aproksimasi dari model logika penuh yang bergelombang. Di bagian ini, kami menjelaskan enam pendekatan berbeda untuk ekstraksi informasi, berdasarkan peningkatan kompleksitas pada beberapa dimensi: deterministik ke stokastik, spesifik domain ke umum, buatan tangan ke dipelajari, dan skala kecil ke skala besar.

Automata keadaan-hingga untuk ekstraksi informasi

Jenis sistem ekstraksi informasi yang paling sederhana adalah sistem ekstraksi berbasis atribut yang mengasumsikan bahwa keseluruhan teks merujuk pada satu objek dan tugasnya adalah mengekstrak atribut objek tersebut. Sebagai contoh, kami telah menyebutkan di Bagian 12.7 masalah mengekstrak dari teks "IBM ThinkBook 970. Harga kami: Rp.3.990.000" sekumpulan atribut {Produsen = IBM, Model = ThinkBook970, Harga = Rp. 3.990.000}. Kita dapat mengatasi masalah ini dengan mendefinisikan templat (juga dikenal sebagai pola) untuk setiap atribut yang ingin kita ekstrak. Templat didefinisikan oleh automata keadaan-hingga, contoh paling sederhana adalah ekspresi reguler, atau regex.

Ekspresi reguler digunakan dalam perintah Unix seperti grep, dalam bahasa pemrograman seperti Perl, dan dalam pengolah kata seperti Microsoft Word. Detailnya sedikit berbeda dari satu alat ke alat lainnya, sehingga sebaiknya dipelajari dari manual yang sesuai. Namun, di sini kami menunjukkan cara membuat templat ekspresi reguler untuk harga dalam dolar:

$[0-9]$ cocok dengan digit apa pun dari 0 hingga 9
 $[0-9]^+$ cocok dengan satu digit atau lebih
 $[.][0-9][0-9]$ cocok dengan titik yang diikuti dua digit
 $([.][0-9][0-9])^?$ cocok dengan titik yang diikuti dua digit, atau tidak sama sekali
 $[\$][0-9]^+([.][0-9][0-9])^?$ cocok dengan Rp.2.490.099 atau Rp.100.230 atau Rp.1.000.000 atau . . .

Templat sering kali didefinisikan dengan tiga bagian: regex awalan, regex target, dan regex akhiran. Untuk harga, regex target seperti di atas, awalan akan mencari string seperti "price:" dan akhiran bisa kosong. Idennya adalah bahwa beberapa petunjuk tentang suatu atribut berasal dari nilai atribut itu sendiri dan beberapa berasal dari teks di sekitarnya.

Jika ekspresi reguler untuk suatu atribut cocok dengan teks tepat satu kali, maka kita dapat menarik bagian teks yang merupakan nilai atribut tersebut. Jika tidak ada kecocokan, yang bisa kita lakukan hanyalah memberikan nilai default atau membiarkan atribut tersebut kosong; tetapi jika ada beberapa kecocokan, kita memerlukan proses untuk memilih di antara kecocokan tersebut. Salah satu strateginya adalah memiliki beberapa templat untuk setiap atribut, diurutkan berdasarkan prioritas.

Jadi, misalnya, templat prioritas utama untuk harga mungkin mencari awalan "harga kami: "; jika tidak ditemukan, kita mencari awalan "harga:" dan jika tidak ditemukan, awalan kosong. Strategi lain adalah mengambil semua kecocokan dan menemukan cara untuk memilih di antara kecocokan tersebut. Misalnya, kita dapat mengambil harga terendah yang berada dalam kisaran 50% dari harga tertinggi. Itu akan memilih Rp.78.000 sebagai target dari teks "Harga daftar Rp.99.000, harga obral khusus Rp.78.000, ongkos kirim Rp.30.000." Sistem ekstraksi relasional, yang menangani beberapa objek dan relasi di antara objek-objek tersebut, selangkah lebih maju dari sistem ekstraksi berbasis atribut.

Oleh karena itu, ketika sistem ini melihat teks "Rp.249.990", mereka perlu menentukan bukan hanya bahwa itu adalah sebuah harga, tetapi juga objek mana yang memiliki harga tersebut. Sistem ekstraksi berbasis relasional yang umum adalah FASTUS, yang menangani berita tentang merger dan akuisisi perusahaan. Sistem ini dapat membaca berita tersebut Bridgestone Sports Co. mengatakan pada hari Jumat bahwa mereka telah mendirikan usaha patungan di Taiwan dengan perusahaan lokal dan sebuah rumah dagang Jepang untuk memproduksi stik golf yang akan dikirim ke Jepang.
dan ekstrak hubungannya:

$$\begin{aligned}
 e \in \text{JointVentures} \wedge \text{Product}(e, \text{"golf clubs"}) \wedge \text{Date}(e, \text{"Friday"}) \\
 \wedge \text{Member}(e, \text{"Bridgestone Sports Co"}) \wedge \text{Member}(e, \text{"a local concern"}) \\
 \wedge \text{Member}(e, \text{"a Japanese trading house"}).
 \end{aligned}$$

Sistem ekstraksi relasional dapat dibangun sebagai serangkaian transduser finite-state yang berjenjang. Artinya, sistem ini terdiri dari serangkaian *finite-state automata* (FSA) yang kecil dan efisien, di mana setiap automaton menerima teks sebagai masukan, mentransduksi teks

tersebut ke dalam format yang berbeda, dan meneruskannya ke automaton berikutnya. FASTUS terdiri dari lima tahap:

1. Tokenisasi
2. Penanganan kata kompleks
3. Penanganan grup dasar
4. Penanganan frasa kompleks
5. Penggabungan struktur

Tahap pertama FASTUS adalah tokenisasi, yang membagi aliran karakter menjadi token (kata, angka, dan tanda baca). Untuk bahasa Inggris, tokenisasi bisa cukup sederhana; cukup memisahkan karakter dengan spasi atau tanda baca saja sudah cukup baik. Beberapa tokenizer juga menangani bahasa markup seperti HTML, SGML, dan XML.

Tahap kedua menangani kata kompleks, termasuk kolokasi seperti "set up" dan "joint venture," serta nama diri seperti "Bridgestone Sports Co." Ini dikenali melalui kombinasi entri leksikal dan aturan tata bahasa keadaan terbatas. Misalnya, nama perusahaan mungkin dikenali melalui aturan

CapitalizedWord + ("Company" | "Co" | "Inc" | "Ltd")

Tahap ketiga menangani kelompok-kelompok dasar, yaitu kelompok nomina dan kelompok verba. Idennya adalah untuk membagi kelompok-kelompok ini menjadi unit-unit yang akan dikelola pada tahap-tahap selanjutnya.

Kita akan melihat cara menulis deskripsi kompleks frasa nomina dan verba di Bab selanjutnya, tetapi di sini kita memiliki aturan sederhana yang hanya mendekati kompleksitas bahasa Inggris, tetapi memiliki keuntungan karena dapat direpresentasikan oleh otomata keadaan terbatas. Kalimat contoh akan muncul dari tahap ini sebagai urutan kelompok yang ditandai berikut:

- | | | | | | |
|---|-----------|-------------------------------|----|-----------|---------------------------------|
| 1 | NG | <i>Bridgestone Sports Co.</i> | 10 | NG | <i>a local concern</i> |
| 2 | VG | <i>said</i> | 11 | CJ | <i>and</i> |
| 3 | NG | <i>Friday</i> | 12 | NG | <i>a Japanese trading house</i> |
| 4 | NG | <i>it</i> | 13 | VG | <i>to produce</i> |
| 5 | VG | <i>had set up</i> | 14 | NG | <i>golf clubs</i> |
| 6 | NG | <i>a joint venture</i> | 15 | VG | <i>to be shipped</i> |
| 7 | PR | <i>in</i> | 16 | PR | <i>to</i> |
| 8 | NG | <i>Taiwan</i> | 17 | NG | <i>Japan</i> |
| 9 | PR | <i>with</i> | | | |

Di sini, NG berarti kelompok nomina, VG adalah kelompok verba, PR adalah preposisi, dan CJ adalah konjungsi. Tahap keempat menggabungkan kelompok-kelompok dasar menjadi frasa kompleks. Sekali lagi, tujuannya adalah untuk memiliki aturan yang berhingga sehingga dapat diproses dengan cepat, dan menghasilkan frasa keluaran yang tidak ambigu (atau hampir tidak

ambigu). Salah satu jenis aturan kombinasi berkaitan dengan peristiwa spesifik domain. Misalnya, aturan

Company + SetUp JointVenture ("with" Company+)?

Bahasa Indonesia: menangkap satu cara untuk menggambarkan pembentukan usaha patungan. Tahap ini adalah yang pertama dalam kaskade di mana keluaran ditempatkan ke dalam templat basis data serta ditempatkan di aliran keluaran. Tahap terakhir menggabungkan struktur yang dibangun pada langkah sebelumnya. Jika kalimat berikutnya mengatakan "Usaha patungan akan mulai berproduksi pada bulan Januari," maka langkah ini akan melihat bahwa ada dua referensi ke usaha patungan, dan bahwa keduanya harus digabungkan menjadi satu.

Secara umum, ekstraksi informasi berbasis templat keadaan-hingga bekerja dengan baik untuk domain terbatas di mana dimungkinkan untuk menentukan terlebih dahulu subjek apa yang akan dibahas, dan bagaimana mereka akan disebutkan. Model transduser berjenjang membantu memodulasi pengetahuan yang diperlukan, memudahkan konstruksi sistem. Sistem ini bekerja dengan sangat baik ketika mereka merekayasa balik teks yang telah dihasilkan oleh suatu program.

Misalnya, situs belanja di Web dihasilkan oleh suatu program yang mengambil entri basis data dan memformatnya menjadi halaman Web; Ekstraktor berbasis templat kemudian memulihkan basis data asli. Ekstraksi informasi keadaan-hingga kurang berhasil dalam memulihkan informasi dalam format yang sangat bervariasi, seperti teks yang ditulis oleh manusia tentang berbagai subjek.

Model probabilistik untuk ekstraksi informasi

Ketika ekstraksi informasi harus dicoba dari masukan yang bising atau bervariasi, pendekatan keadaan-hingga yang sederhana kurang berhasil. Terlalu sulit untuk mendapatkan semua aturan dan prioritasnya dengan benar; lebih baik menggunakan model probabilistik daripada model berbasis aturan. Model probabilistik paling sederhana untuk deret dengan keadaan tersembunyi adalah model Markov tersembunyi, atau HMM. HMM memodelkan suatu progresi melalui deret keadaan tersembunyi, x_t , dengan observasi e_t pada setiap langkah. Untuk menerapkan HMM pada ekstraksi informasi, kita dapat membangun satu HMM besar untuk semua atribut atau membangun HMM terpisah untuk setiap atribut.

Kita akan melakukan yang kedua. Observasi adalah kata-kata dalam teks, dan status tersembunyi adalah apakah kita berada di bagian target, awalan, atau akhiran dari templat atribut, atau di latar belakang (bukan bagian dari templat). Sebagai contoh, berikut adalah teks singkat dan jalur (Viterbi) yang paling mungkin untuk teks tersebut untuk dua HMM, satu dilatih untuk mengenali pembicara dalam pengumuman ceramah, dan satu lagi dilatih untuk mengenali tanggal. Tanda "-" menunjukkan status latar belakang:

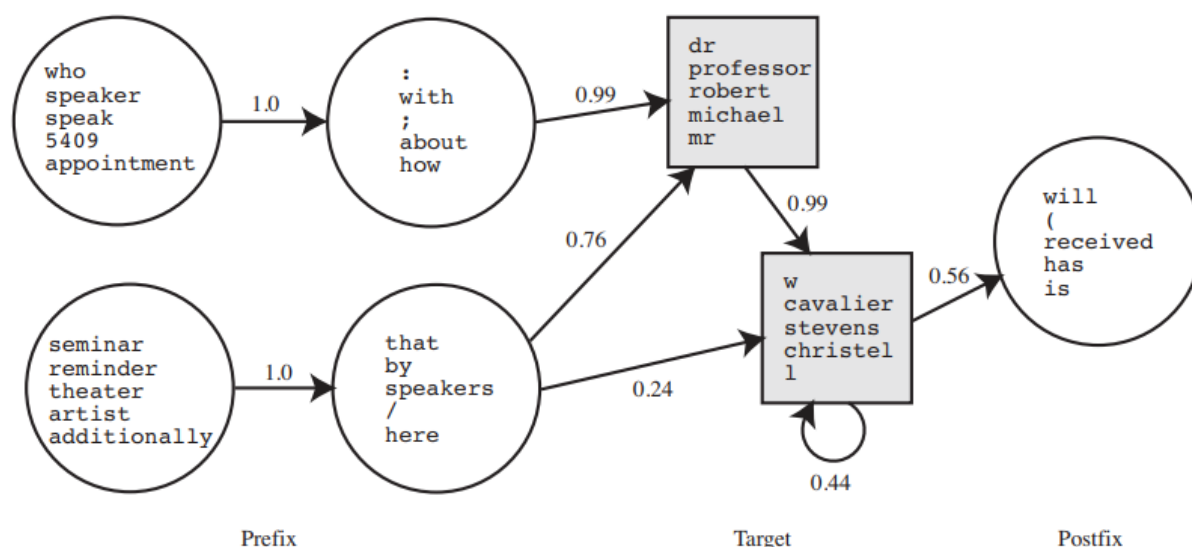
Text	There	will	be	a	seminar	by	Dr.	Andrew	McCallum	on	Friday
Speaker	-	-	-	-	PRE	PRE	TARGET	TARGET	TARGET	POST	-
Date	-	-	-	-	-	-	-	-	-	PRE	TARGET

HMM memiliki dua keunggulan besar dibandingkan FSA untuk ekstraksi. Pertama, HMM bersifat probabilistik, sehingga toleran terhadap derau. Dalam ekspresi reguler, jika satu karakter yang diharapkan hilang, regex gagal untuk dicocokkan; dengan HMM terdapat degradasi yang anggun dengan karakter/kata yang hilang, dan kita mendapatkan probabilitas yang menunjukkan tingkat kecocokan, bukan hanya kecocokan/kegagalan Boolean. Kedua, HMM dapat dilatih dari data; HMM tidak memerlukan rekayasa templat yang rumit, sehingga dapat lebih mudah diperbarui seiring perubahan teks seiring waktu.

Perhatikan bahwa kami telah mengasumsikan tingkat struktur tertentu dalam templat HMM kami: semuanya terdiri dari satu atau lebih status target, dan status awalan apa pun harus mendahului target, status akhiran harus mengikuti target, dan status lainnya harus berada di latar belakang. Struktur ini memudahkan untuk mempelajari HMM dari contoh. Dengan struktur yang ditentukan sebagian, algoritma maju-mundur dapat digunakan untuk mempelajari probabilitas transisi $P(X_t | X_{t-1})$ antar status dan model observasi, $P(E_t | X_t)$, yang menunjukkan seberapa besar kemungkinan setiap kata berada di setiap status. Misalnya, kata "Jumat" akan memiliki probabilitas tinggi di satu atau lebih status target HMM tanggal, dan probabilitas lebih rendah di tempat lain.

Dengan data pelatihan yang memadai, HMM secara otomatis mempelajari struktur tanggal yang kita anggap intuitif: HMM tanggal mungkin memiliki satu status target dengan kata-kata probabilitas tinggi seperti "Senin," "Selasa," dst., dan yang memiliki transisi probabilitas tinggi ke status target dengan kata-kata "Jan," "Januari," "Feb," dst. HMM untuk pembicara pengumuman ceramah, sebagaimana dipelajari dari data. Awalan tersebut mencakup ekspresi seperti "Pembicara:" dan "seminar oleh", dan target memiliki satu status yang mencakup gelar dan nama depan, serta status lain yang mencakup inisial dan nama belakang.

Setelah HMM dipelajari, kita dapat menerapkannya pada teks, menggunakan algoritma Viterbi untuk menemukan jalur yang paling mungkin melalui status HMM. Salah satu pendekatannya adalah menerapkan setiap atribut HMM secara terpisah; dalam hal ini, Anda akan mengharapkan sebagian besar HMM menghabiskan sebagian besar waktunya dalam status latar belakang. Hal ini sesuai ketika ekstraksi bersifat sparse ketika jumlah kata yang diekstraksi kecil dibandingkan dengan panjang teks.



Gambar 4.2 Model Markov Tersembunyi untuk pembicara dalam pengumuman ceramah. Dua status perseg adalah target (perhatikan bahwa status target kedua memiliki perulangan otomatis, sehingga target dapat mencocokkan string dengan panjang berapa pun), empat lingkaran di sebelah kiri adalah awalan, dan satu di sebelah kanan adalah akhiran. Untuk setiap status, hanya beberapa kata **dengan probabilitas tinggi yang ditampilkan**.

Pendekatan lainnya adalah menggabungkan semua atribut individual ke dalam satu HMM besar, yang kemudian akan menemukan jalur yang menelusuri berbagai atribut target, pertama-tama menemukan target pembicara, lalu target tanggal, dst. HMM terpisah lebih baik ketika kita hanya mengharapkan satu atribut dari setiap teks, dan satu HMM besar lebih baik ketika teks lebih bebas dan padat dengan atribut.

Dengan kedua pendekatan tersebut, pada akhirnya kita memiliki kumpulan observasi atribut target, dan harus memutuskan apa yang harus dilakukan dengannya. Jika setiap atribut yang diharapkan memiliki satu pengisi target, maka keputusannya mudah: kita memiliki contoh relasi yang diinginkan. Jika terdapat beberapa pengisi, kita perlu memutuskan mana yang akan dipilih, seperti yang telah kita bahas dengan sistem berbasis templat. HMM memiliki keuntungan menyediakan angka probabilitas yang dapat membantu dalam pengambilan keputusan. Jika beberapa target hilang, kita perlu memutuskan apakah ini merupakan contoh relasi yang diinginkan, atau apakah target yang ditemukan merupakan positif palsu. Algoritma pembelajaran mesin dapat dilatih untuk membuat pilihan ini.

Medan Acak Bersyarat untuk Ekstraksi Informasi

Salah satu masalah dengan HMM untuk tugas ekstraksi informasi adalah ia memodelkan banyak probabilitas yang sebenarnya tidak kita perlukan. HMM adalah model generatif; ia memodelkan probabilitas gabungan penuh dari observasi dan status tersembunyi, sehingga dapat digunakan untuk menghasilkan sampel. Artinya, kita dapat menggunakan model HMM tidak hanya untuk mengurai teks dan memulihkan pembicara dan tanggal, tetapi juga untuk menghasilkan contoh acak dari teks yang berisi pembicara dan tanggal. Karena kita tidak tertarik dengan tugas tersebut, wajar untuk bertanya apakah kita mungkin lebih baik menggunakan model yang tidak repot-repot memodelkan kemungkinan tersebut. Yang kita

butuhkan untuk memahami sebuah teks adalah model diskriminatif, model yang memodelkan probabilitas bersyarat dari atribut tersembunyi yang diberikan observasi (teks). Diberikan sebuah teks $e_{1:N}$, model bersyarat menemukan deret status tersembunyi $X_{1:N}$ yang memaksimalkan $P(X_{1:N} | e_{1:N})$.

Memodelkan hal ini secara langsung memberi kita kebebasan. Kita tidak memerlukan asumsi independensi model Markov kita dapat memiliki x_t yang bergantung pada x_1 . Kerangka kerja untuk model jenis ini adalah medan acak bersyarat, atau CRF, yang memodelkan distribusi probabilitas bersyarat dari sekumpulan variabel target yang diberikan sekumpulan variabel teramati. Seperti jaringan Bayesian, CRF dapat merepresentasikan berbagai struktur dependensi antar variabel. Salah satu struktur umum adalah medan acak bersyarat rantai linier untuk merepresentasikan dependensi Markov antar variabel dalam urutan temporal. Dengan demikian, HMM adalah versi temporal dari model Bayes naif, dan CRF rantai linier adalah versi temporal dari regresi logistik, di mana target yang diprediksi adalah keseluruhan urutan keadaan, alih-alih satu variabel biner.

Misalkan $e_{1:N}$ adalah observasi (misalnya, kata-kata dalam dokumen), dan $x_{1:N}$ adalah urutan keadaan tersembunyi (misalnya, keadaan awalan, target, dan akhiran). Medan acak bersyarat rantai linier mendefinisikan distribusi probabilitas bersyarat:

$$P(x_{1:N} | e_{1:N}) = \alpha e^{\sum_{i=1}^N F(x_{i-1}, x_i, e, i)}$$

di mana α adalah faktor normalisasi (untuk memastikan probabilitas berjumlah 1), dan F adalah fungsi fitur yang didefinisikan sebagai jumlah tertimbang dari kumpulan fungsi fitur komponen k :

$$F(x_{i-1}, x_i, e, i) = \sum_k \lambda_k f_k(x_{i-1}, x_i, e, i)$$

Nilai parameter λ_k dipelajari dengan prosedur estimasi MAP (maksimum a posteriori) yang memaksimalkan kemungkinan kondisional dari data pelatihan. Fungsi fitur merupakan komponen kunci dari CRF. Fungsi f_k memiliki akses ke sepasang status yang berdekatan, x_{i-1} dan x_i , tetapi juga seluruh urutan observasi (kata) e , dan posisi saat ini dalam urutan temporal, i . Hal ini memberi kita banyak fleksibilitas dalam mendefinisikan fitur. Kita dapat mendefinisikan fungsi fitur sederhana, misalnya yang menghasilkan nilai 1 jika kata saat ini adalah ANDREW dan status saat ini adalah SPEAKER:

$$f_1(x_{i-1}, x_i, e, i) = f(x) = \begin{cases} 1, & \text{jika } x_i = \text{SPEAKER dan } e_i = \text{ANDREW} \\ 0, & \text{lainnya} \end{cases}$$

Fitur ini benar jika status saat ini adalah SPEAKER dan kata berikutnya adalah "said". Oleh karena itu, kita akan mengharapkan nilai λ_2 positif untuk fitur tersebut. Lebih menariknya, perhatikan bahwa f_1 dan f_2 dapat berlaku secara bersamaan untuk kalimat seperti "Andrew

said" Dalam kasus ini, kedua fitur saling tumpang tindih dan keduanya meningkatkan keyakinan pada $x_1 = \text{SPEAKER}$. Karena asumsi independensi, HMM tidak dapat menggunakan fitur yang tumpang tindih; CRF dapat.

Lebih lanjut, fitur dalam CRF dapat menggunakan bagian mana pun dari deret $e_{1:N}$. Fitur juga dapat didefinisikan melalui transisi antar status. Fitur yang kami definisikan di sini bersifat biner, tetapi secara umum, fungsi fitur dapat berupa fungsi bernilai riil apa pun. Untuk domain tempat kita memiliki pengetahuan tentang jenis fitur yang ingin kita sertakan, formalisme CRF memberi kita banyak fleksibilitas dalam mendefinisikannya. Fleksibilitas ini dapat menghasilkan akurasi yang lebih tinggi dibandingkan dengan model yang kurang fleksibel seperti HMM.

Ekstraksi ontologi dari korpus besar

Sejauh ini kita telah menganggap ekstraksi informasi sebagai pencarian serangkaian relasi spesifik (misalnya, pembicara, waktu, lokasi) dalam teks tertentu (misalnya, pengumuman ceramah). Aplikasi lain dari teknologi ekstraksi adalah membangun basis pengetahuan atau ontologi fakta yang besar dari sebuah korpus. Hal ini berbeda dalam tiga hal: Pertama, bersifat terbuka kita ingin memperoleh fakta tentang semua jenis domain, bukan hanya satu domain spesifik. Kedua, dengan korpus besar, tugas ini didominasi oleh presisi, bukan ingatan sama seperti menjawab pertanyaan di Web. Ketiga, hasilnya dapat berupa agregat statistik yang dikumpulkan dari berbagai sumber, alih-alih diekstraksi dari satu teks spesifik.

Misalnya, Hearst (1992) meneliti masalah mempelajari ontologi kategori dan subkategori konsep dari korpus besar. (Pada tahun 1992, korpus besar adalah ensiklopedia 1000 halaman; sekarang korpus web 100 juta halaman.) Pekerjaan difokuskan pada templat yang sangat umum (tidak terikat pada domain tertentu) dan memiliki presisi tinggi (hampir selalu benar jika cocok) tetapi daya ingatnya rendah (tidak selalu cocok). Berikut adalah salah satu templat yang paling produktif:

NP such as NP(, NP) (,)? ((dan |atau) NP)*

Di sini, kata-kata tebal dan koma harus muncul secara harfiah di dalam teks, tetapi tanda kurung untuk pengelompokan, tanda bintang berarti pengulangan nol atau lebih, dan tanda tanya bersifat opsional. NP adalah variabel yang mewakili frasa nomina; Bab 5 menjelaskan cara mengidentifikasi frasa nomina; untuk saat ini, asumsikan saja kita tahu beberapa kata adalah nomina dan kata lain (seperti verba) yang dapat kita asumsikan dengan andal bukan bagian dari frasa nomina sederhana. Templat ini cocok dengan teks "penyakit seperti rabies memengaruhi anjing Anda" dan "mendukung protokol jaringan seperti DNS," yang menyimpulkan bahwa rabies adalah penyakit dan DNS adalah protokol jaringan. Templat serupa dapat dibuat dengan kata kunci "termasuk," "terutama," dan "atau lainnya."

Tentu saja templat ini akan gagal untuk mencocokkan banyak bagian yang relevan, seperti "Rabies adalah penyakit." Hal itu disengaja. Templat "*NP* adalah *NP*" memang terkadang menunjukkan hubungan subkategori, tetapi seringkali berarti sesuatu yang lain,

seperti dalam "Ada Tuhan" atau "Dia sedikit lelah." Dengan korpus yang besar, kita bisa lebih selektif; hanya menggunakan templat berpresisi tinggi. Kita akan kehilangan banyak pernyataan tentang hubungan subkategori, tetapi kemungkinan besar kita akan menemukan parafrase pernyataan tersebut di tempat lain dalam korpus dalam bentuk yang dapat kita gunakan.

Konstruksi templat otomatis

Relasi subkategori begitu mendasar sehingga perlu dibuat beberapa templat secara manual untuk membantu mengidentifikasi kejadian kemunculannya dalam teks bahasa alami. Namun, bagaimana dengan ribuan relasi lain di dunia? Jumlah mahasiswa pascasarjana AI di dunia tidak cukup untuk membuat dan men-debug templat untuk semuanya. Untungnya, kita dapat mempelajari templat dari beberapa contoh, lalu menggunakan templat tersebut untuk mempelajari lebih banyak contoh, yang darinya lebih banyak templat dapat dipelajari, dan seterusnya. Dalam salah satu eksperimen pertama semacam ini, Brin (1999) memulai dengan kumpulan data yang hanya terdiri dari lima contoh:

(“Isaac Asimov”, “The Robots of Dawn”)

(“David Brin”, “Startide Rising”)

(“James Gleick”, “Chaos—Making a New Science”)

(“Charles Dickens”, “Great Expectations”)

(“William Shakespeare”, “The Comedy of Errors”)

Jelas ini adalah contoh hubungan penulis-judul, tetapi sistem pembelajaran tidak memiliki pengetahuan tentang penulis atau judul. Kata-kata dalam contoh ini digunakan dalam pencarian melalui korpus Web, menghasilkan 199 kecocokan. Setiap kecocokan didefinisikan sebagai tupel tujuh string,

(Author, Title, Order, Prefix, Middle, Postfix, URL)

di mana Order bernilai true jika penulis muncul pertama dan false jika judul muncul pertama, Middle adalah karakter antara penulis dan judul, Prefix adalah 10 karakter sebelum pencocokan, Suffix adalah 10 karakter setelah pencocokan, dan URL adalah alamat web tempat pencocokan dibuat. Dengan sekumpulan pencocokan, skema pembangkitan templat sederhana dapat menemukan templat untuk menjelaskan pencocokan tersebut. Bahasa templat dirancang agar memiliki pemetaan yang dekat dengan pencocokan itu sendiri, agar dapat menerima pembelajaran otomatis, dan untuk menekankan presisi tinggi (mungkin dengan risiko ingatan yang lebih rendah).

Setiap templat memiliki tujuh komponen yang sama dengan pencocokan. Author dan Title adalah regex yang terdiri dari karakter apa pun (tetapi dimulai dan diakhiri dengan huruf) dan dibatasi untuk memiliki panjang dari setengah panjang minimum contoh hingga dua kali panjang maksimum. Prefix, middle, dan postfix dibatasi untuk string literal, bukan regex. Middle adalah yang paling mudah dipelajari: setiap string tengah yang berbeda dalam sekumpulan pencocokan adalah templat kandidat yang berbeda. Untuk setiap kandidat

tersebut, Awalan templat kemudian didefinisikan sebagai sufiks umum terpanjang dari semua awalan dalam kecocokan, dan Postfix didefinisikan sebagai awalan umum terpanjang dari semua postfix dalam kecocokan. Jika salah satu dari keduanya memiliki panjang nol, maka templat ditolak. URL templat didefinisikan sebagai awalan terpanjang dari URL dalam kecocokan. Dalam percobaan yang dijalankan oleh Brin, 199 kecocokan pertama menghasilkan tiga templat. Templat yang paling produktif adalah

Title by Author (
URL: www.sff.net/locus/c

Ketiga templat tersebut kemudian digunakan untuk mengambil 4.047 contoh lagi (penulis, judul). Contoh-contoh tersebut kemudian digunakan untuk menghasilkan lebih banyak templat, dan seterusnya, yang akhirnya menghasilkan lebih dari 15.000 judul. Dengan sekumpulan templat yang baik, sistem dapat mengumpulkan sekumpulan contoh yang baik. Dengan sekumpulan contoh yang baik, sistem dapat membangun sekumpulan templat yang baik.

Kelemahan terbesar dalam pendekatan ini adalah sensitivitas terhadap derau. Jika salah satu dari beberapa templat pertama salah, kesalahan dapat menyebar dengan cepat. Salah satu cara untuk membatasi masalah ini adalah dengan tidak menerima contoh baru kecuali jika diverifikasi oleh beberapa templat, dan tidak menerima templat baru kecuali jika menemukan beberapa contoh yang juga ditemukan oleh templat lain.

Pembacaan Mesin

Konstruksi templat otomatis merupakan langkah maju yang besar dari konstruksi templat manual, tetapi masih memerlukan beberapa contoh berlabel dari setiap relasi untuk memulai. Untuk membangun ontologi besar dengan ribuan relasi, bahkan pekerjaan sebanyak itu pun akan sangat berat; Kami ingin memiliki sistem ekstraksi tanpa masukan manusia apa pun sistem yang dapat membaca sendiri dan membangun basis datanya sendiri. Sistem semacam itu akan independen terhadap relasi; akan berfungsi untuk relasi apa pun. Dalam praktiknya, sistem ini bekerja pada semua relasi secara paralel, karena tuntutan I/O dari korpus yang besar. Sistem ini berperilaku kurang seperti sistem ekstraksi informasi tradisional yang ditargetkan pada beberapa relasi dan lebih seperti pembaca manusia yang belajar dari teks itu sendiri; karena itu, bidang ini disebut pembacaan mesin.

Sistem pembacaan mesin yang representatif adalah TEXTRUNNER. TEXTRUNNER menggunakan cotraining untuk meningkatkan kinerjanya, tetapi membutuhkan sesuatu untuk melakukan bootstrap. Dalam kasus Hearst (1992), pola-pola spesifik (misalnya, seperti) menyediakan bootstrap, dan untuk Brin (1998), itu adalah sekumpulan lima pasangan penulis-judul. Untuk TEXTRUNNER, inspirasi awalnya adalah taksonomi delapan templat sintaksis yang sangat umum, seperti yang ditunjukkan pada Gambar 4.3. Dirasa bahwa sejumlah kecil templat seperti ini dapat mencakup sebagian besar cara hubungan diungkapkan dalam bahasa Inggris. Bootstrapping yang sebenarnya dimulai dari serangkaian contoh berlabel yang diekstrak dari Penn Treebank, sebuah korpus kalimat yang telah diurai. Misalnya, dari

penguraian kalimat "Einstein menerima Hadiah Nobel pada tahun 1921," TEXTRUNNER dapat mengekstrak relasinya ("Einstein," "menerima," "Hadiah Nobel").

Dengan serangkaian contoh berlabel jenis ini, TEXTRUNNER melatih CRF rantai linear untuk mengekstrak contoh lebih lanjut dari teks yang tidak berlabel. Fitur-fitur dalam CRF mencakup kata fungsi seperti "to" dan "of" dan "the," tetapi tidak termasuk kata benda dan kata kerja (dan bukan frasa kata benda atau frasa kata kerja). Karena TEXTRUNNER bersifat domain-independen, ia tidak dapat bergantung pada daftar kata benda dan kerja yang telah ditetapkan sebelumnya. TEXTRUNNER mencapai presisi 88% dan recall 45% (F1 60%) pada korpus web yang besar. TEXTRUNNER telah mengekstrak ratusan juta fakta dari korpus berisi setengah miliar halaman web. Misalnya, meskipun tidak memiliki pengetahuan medis yang telah ditentukan sebelumnya, TEXTRUNNER telah mengekstrak lebih dari 2000 jawaban untuk pertanyaan [apa yang membunuh bakteri]; jawaban yang benar meliputi antibiotik, ozon, klorin, Cipro, dan kecambah brokoli. Jawaban yang dipertanyakan meliputi "air," yang berasal dari kalimat "Air mendidih selama minimal 10 menit akan membunuh bakteri." Akan lebih baik untuk mengaitkan hal ini dengan "air mendidih" daripada hanya "air."

Type	Template	Example	Frequency
Verb	NP_1 Verb NP_2	X established Y	38%
Noun–Prep	NP_1 NP Prep NP_2	X settlement with Y	23%
Verb–Prep	NP_1 Verb Prep NP_2	X moved to Y	16%
Infinitive	NP_1 to Verb NP_2	X plans to acquire Y	9%
Modifier	NP_1 Verb NP_2 Noun	X is Y winner	5%
Noun-Coordinate	* NP_1 (, and - :) NP_2 NP	X-Y deal	2%
Verb-Coordinate	NP_1 (, and) NP_2 Verb	X, Y merge	1%
Appositive	* NP_1 NP (: ;) ? NP_2 *	X hometown : Y	1%

Gambar 4.3 Delapan templat umum yang mencakup sekitar 95% cara hubungan diungkapkan dalam bahasa Inggris.

Dengan teknik yang diuraikan dalam bab ini dan penemuan baru yang terus-menerus, kita mulai mendekati tujuan pembacaan mesin.

4.5 RINGKASAN

Poin-poin utama bab ini adalah sebagai berikut:

- Model bahasa probabilistik berbasis n-gram memulihkan sejumlah besar informasi tentang suatu bahasa. Model ini dapat bekerja dengan baik pada beragam tugas seperti identifikasi bahasa, koreksi ejaan, klasifikasi genre, dan pengenalan entitas bernama.
- Model bahasa ini dapat memiliki jutaan fitur, sehingga pemilihan fitur dan prapemrosesan data untuk mengurangi gangguan menjadi penting.
- Klasifikasi teks dapat dilakukan dengan model n-gram Bayes naif atau dengan algoritma klasifikasi apa pun yang telah kita bahas sebelumnya. Klasifikasi juga dapat dilihat sebagai masalah dalam kompresi data.

- Sistem temu kembali informasi menggunakan model bahasa yang sangat sederhana berdasarkan kumpulan kata, namun tetap berkinerja baik dalam hal mengingat dan presisi pada korpus teks yang sangat besar. Pada korpus web, algoritma analisis tautan meningkatkan kinerja.
- Penjawaban pertanyaan dapat ditangani dengan pendekatan berbasis temu kembali informasi, untuk pertanyaan yang memiliki banyak jawaban dalam korpus. Ketika lebih banyak jawaban tersedia dalam korpus, kita dapat menggunakan teknik yang lebih menekankan presisi daripada mengingat.
- Sistem ekstraksi informasi menggunakan model yang lebih kompleks yang mencakup gagasan sintaksis dan semantik yang terbatas dalam bentuk templat. Model ini dapat dibangun dari otomata keadaan-hingga, HMM, atau medan acak bersyarat, dan dapat dipelajari dari contoh.
- Dalam membangun sistem bahasa statistik, sebaiknya merancang model yang dapat memanfaatkan data yang tersedia dengan baik, meskipun model tersebut tampak terlalu sederhana.

BAB 5

BAHASA ALAMI UNTUK KOMUNIKASI

5.1 TATA BAHASA STRUKTUR FRASA

Model bahasa n -gram pada Bab 4 didasarkan pada urutan kata. Masalah besar bagi model ini adalah kelangkaan data—dengan kosakata, katakanlah, 10^5 kata, terdapat 10^{15} probabilitas trigram yang perlu diestimasi, sehingga korpus yang berisi bahkan satu triliun kata pun tidak akan mampu memberikan estimasi yang andal untuk semuanya. Kita dapat mengatasi masalah kelangkaan ini melalui generalisasi. Dari fakta bahwa "black dog" lebih sering muncul daripada "dog black" dan pengamatan serupa, kita dapat membentuk generalisasi bahwa kata sifat cenderung muncul sebelum kata benda dalam bahasa Inggris (sedangkan kata sifat cenderung muncul setelah kata benda dalam bahasa Prancis: "chien noir" lebih sering muncul). Tentu saja selalu ada pengecualian; "galore" adalah kata sifat yang muncul setelah kata benda yang dimodifikasinya.

Meskipun ada pengecualian, gagasan tentang kategori leksikal (juga dikenal sebagai jenis kata) seperti kata benda atau kata sifat merupakan generalisasi yang berguna—berguna dalam dirinya sendiri, tetapi lebih berguna lagi ketika kita merangkai kategori leksikal untuk membentuk kategori sintaksis seperti frasa nomina atau frasa verba, dan menggabungkan kategori sintaksis ini menjadi pohon yang mewakili struktur frasa kalimat: frasa bersarang, masing-masing ditandai dengan sebuah kategori.

Kapasitas Generatif

Formalitas gramatikal dapat diklasifikasikan berdasarkan kapasitas generatifnya: himpunan bahasa yang dapat diwakilinya. Chomsky menjelaskan empat kelas formalisme gramatikal yang hanya berbeda dalam bentuk aturan penulisan ulang. Kelas-kelas tersebut dapat disusun dalam hierarki, di mana setiap kelas dapat digunakan untuk mendeskripsikan semua bahasa yang dapat dideskripsikan oleh kelas yang kurang kuat, serta beberapa bahasa tambahan. Di sini kami mencantumkan hierarkinya, dengan kelas yang paling kuat terlebih dahulu:

Tata bahasa yang dapat dihitung secara rekursif menggunakan aturan tak terbatas: kedua sisi aturan penulisan ulang dapat memiliki sejumlah simbol terminal dan nonterminal, seperti dalam aturan $A B C \rightarrow D E$. Tata bahasa ini setara dengan mesin Turing dalam hal daya ekspresifnya.

Tata bahasa peka konteks dibatasi hanya karena ruas kanan harus berisi setidaknya simbol sebanyak ruas kiri. Nama "peka konteks" berasal dari fakta bahwa aturan seperti $A X B \rightarrow A Y B$ menyatakan bahwa sebuah X dapat ditulis ulang sebagai sebuah Y dalam konteks A sebelumnya dan B setelahnya. Tata bahasa peka konteks dapat merepresentasikan bahasa seperti $a^n b^n c^n$ (urutan n salinan a diikuti oleh jumlah b s yang sama dan kemudian c s).

Dalam tata bahasa bebas konteks (atau CFG), ruas kiri terdiri dari satu simbol nonterminal. Dengan demikian, setiap aturan mengizinkan penulisan ulang nonterminal

sebagai ruas kanan dalam konteks apa pun. CFG populer untuk tata bahasa bahasa alami dan bahasa pemrograman, meskipun sekarang diterima secara luas bahwa setidaknya beberapa bahasa alami memiliki konstruksi yang tidak bebas konteks. Tata bahasa bebas konteks dapat merepresentasikan $a^n b^n$, tetapi bukan $a^n b^n c^n$.

Tata bahasa reguler adalah kelas yang paling terbatas. Setiap aturan memiliki satu non-terminal di sisi kiri dan simbol terminal yang opsional diikuti oleh non-terminal di sisi kanan. Tata bahasa reguler setara kekuatannya dengan mesin berhingga. Tata bahasa ini kurang cocok untuk bahasa pemrograman, karena tidak dapat merepresentasikan konstruksi seperti tanda kurung buka dan tutup yang seimbang (variasi dari bahasa $a^n b^n$). Yang paling mendekati adalah merepresentasikan $a^* b^*$, deret bilangan berapa pun yang diikuti oleh bilangan b berapa pun.

Tata bahasa yang lebih tinggi dalam hierarki memiliki daya ekspresif yang lebih besar, tetapi algoritma untuk menanganinya kurang efisien. Hingga tahun 1980an, para ahli bahasa berfokus pada bahasa bebas konteks dan peka konteks. Sejak saat itu, minat terhadap tata bahasa reguler kembali muncul, didorong oleh kebutuhan untuk memproses dan mempelajari teks daring berukuran gigabita atau terabita dengan sangat cepat, bahkan dengan mengorbankan analisis yang kurang lengkap. Seperti yang dikatakan Fernando Pereira, "Semakin tua saya, semakin jauh saya menuruni hierarki Chomsky." Untuk melihat maksudnya, bandingkan Pereira dan Warren dengan Mohri, Pereira, dan Riley (dan perhatikan bahwa ketiga penulis ini sekarang bekerja pada korpus teks besar di Google).

Ada banyak model bahasa yang bersaing berdasarkan gagasan struktur frasa; kami akan menjelaskan model populer yang disebut tata bahasa bebas konteks probabilistik, atau PCFG. Tata bahasa adalah kumpulan aturan yang mendefinisikan suatu bahasa sebagai serangkaian rangkaian kata yang diizinkan. "Bebas konteks" dan "probabilistik" berarti bahwa tata bahasa menetapkan probabilitas untuk setiap rangkaian. Berikut adalah aturan PCFG:

$$VP \rightarrow Verb [0.70] \\ | VP NP [0.30] .$$

Di sini, VP (*verb phrase*) dan NP (*noun phrase*) adalah simbol non-terminal. Tata bahasa juga merujuk pada kata-kata aktual, yang disebut simbol terminal. Aturan ini menyatakan bahwa dengan probabilitas 0,70, sebuah frasa verba hanya terdiri dari verba, dan dengan probabilitas 0,30, frasa tersebut adalah VP yang diikuti oleh NP . Lampiran B menjelaskan tata bahasa bebas konteks non-probabilistik.

Kami sekarang mendefinisikan tata bahasa untuk fragmen kecil bahasa Inggris yang cocok untuk komunikasi antar agen yang menjelajahi dunia wumpus. Kami menyebut bahasa ini ε_0 . Bagian selanjutnya menyempurnakan ε_0 agar sedikit lebih dekat dengan bahasa Inggris yang sebenarnya. Kami tidak mungkin pernah merancang tata bahasa yang lengkap untuk bahasa Inggris, karena tidak ada dua orang yang sepenuhnya sepakat tentang apa yang merupakan bahasa Inggris yang valid.

Leksikon ε_0

Pertama, kami mendefinisikan leksikon, atau daftar kata yang diperbolehkan. Kata-kata tersebut dikelompokkan ke dalam kategori leksikal yang familiar bagi pengguna kamus: nomina, pronomina, dan nama untuk menunjukkan sesuatu; kata kerja untuk menunjukkan peristiwa; kata sifat untuk menerangkan kata benda; kata keterangan untuk menerangkan kata kerja; dan kata fungsi: artikel (seperti *the*), preposisi (*in*), dan konjungsi (*and*). Gambar 5.1 menunjukkan leksikon kecil untuk bahasa ε_0 .

Setiap kategori diakhiri dengan ... untuk menunjukkan bahwa ada kata lain dalam kategori tersebut. Untuk kata benda, nama, kata kerja, kata sifat, dan kata keterangan, bahkan pada prinsipnya mustahil untuk mencantumkan semua kata. Tidak hanya ada puluhan ribu anggota di setiap kelas, tetapi yang baru—seperti *iPod* atau *biodiesel*—terus bertambah. Kelima kategori ini disebut kelas terbuka.

Untuk kategori kata ganti, kata ganti relatif, artikel, preposisi, dan konjungsi, kita bisa mencantumkan semua kata dengan sedikit usaha. Ini disebut kelas tertutup; mereka memiliki jumlah kata yang sedikit (sekitar selusin). Kelas tertutup berubah selama berabad-abad, bukan bulan. Misalnya, "thee" dan "thou" merupakan kata ganti yang umum digunakan pada abad ke-17, mengalami penurunan pada abad ke-19, dan kini hanya terlihat dalam puisi dan beberapa dialek daerah.

Tata Bahasa ε_0

Langkah selanjutnya adalah menggabungkan kata-kata tersebut menjadi frasa. Gambar 5.2 menunjukkan tata bahasa untuk ε_0 , dengan aturan untuk masing-masing dari enam kategori sintaksis dan contoh untuk setiap aturan penulisan ulang. Gambar 5.3 menunjukkan pohon parse untuk kalimat "Every wumpus smells." Pohon parse ini memberikan bukti konstruktif bahwa rangkaian kata tersebut memang merupakan kalimat menurut aturan ε_0 . Tata bahasa ε_0 menghasilkan beragam kalimat bahasa Inggris, seperti berikut:

John ada di dalam lubang

Wumpus yang baunya menyengat ada di 2 2

Mary ada di Boston dan wumpusnya dekat 3 2

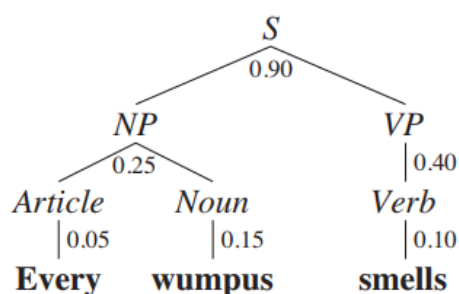
Sayangnya, tata bahasanya menghasilkan terlalu banyak kalimat: artinya, ia menghasilkan kalimat-kalimat yang tidak gramatikal, seperti "Me go Boston" dan "I smell pits wumpus John." Ia juga menghasilkan terlalu sedikit kalimat: ada banyak kalimat bahasa Inggris yang ditolaknya, seperti "I think the wumpus is smelly." Kita akan melihat cara mempelajari tata bahasa yang lebih baik nanti; untuk saat ini, kita akan fokus pada apa yang bisa kita lakukan dengan tata bahasa yang kita miliki.

<i>Noun</i>	→	stench [0.05] breeze [0.10] wumpus [0.15] pits [0.05] ...
<i>Verb</i>	→	is [0.10] feel [0.10] smells [0.10] stinks [0.05] ...
<i>Adjective</i>	→	right [0.10] dead [0.05] smelly [0.02] breezy [0.02] ...
<i>Adverb</i>	→	here [0.05] ahead [0.05] nearby [0.02] ...
<i>Pronoun</i>	→	me [0.10] you [0.03] I [0.10] it [0.10] ...
<i>RelPro</i>	→	that [0.40] which [0.15] who [0.20] whom [0.02] ∨ ...
<i>Name</i>	→	John [0.01] Mary [0.01] Boston [0.01] ...
<i>Article</i>	→	the [0.40] a [0.30] an [0.10] every [0.05] ...
<i>Prep</i>	→	to [0.20] in [0.10] on [0.05] near [0.10] ...
<i>Conj</i>	→	and [0.50] or [0.10] but [0.20] yet [0.02] ∨ ...
<i>Digit</i>	→	0 [0.20] 1 [0.20] 2 [0.20] 3 [0.20] 4 [0.20] ...

Gambar 5.1 Leksikon untuk ε_0 . *RelPro* adalah singkatan untuk kata ganti relatif, *Prep* untuk preposisi, dan *Conj* untuk konjungsi. Jumlah probabilitas untuk setiap kategori adalah 1.

ε_0 :	<i>S</i>	→	<i>NP VP</i>	[0.90]	I + feel a breeze
			<i>S Conj S</i>	[0.10]	I feel a breeze + and + It stinks
	<i>NP</i>	→	<i>Pronoun</i>	[0.30]	I
			<i>Name</i>	[0.10]	John
			<i>Noun</i>	[0.10]	pits
			<i>Article Noun</i>	[0.25]	the + wumpus
			<i>Article Adjs Noun</i>	[0.05]	the + smelly dead + wumpus
			<i>Digit Digit</i>	[0.05]	3 4
			<i>NP PP</i>	[0.10]	the wumpus + in 1 3
			<i>NP RelClause</i>	[0.05]	the wumpus + that is smelly
	<i>VP</i>	→	<i>Verb</i>	[0.40]	stinks
			<i>VP NP</i>	[0.35]	feel + a breeze
			<i>VP Adjective</i>	[0.05]	smells + dead
			<i>VP PP</i>	[0.10]	is + in 1 3
			<i>VP Adverb</i>	[0.10]	go + ahead
	<i>Adjs</i>	→	<i>Adjective</i>	[0.80]	smelly
			<i>Adjective Adjs</i>	[0.20]	smelly + dead
	<i>PP</i>	→	<i>Prep NP</i>	[1.00]	to + the east
	<i>RelClause</i>	→	<i>RelPro VP</i>	[1.00]	that + is smelly

Gambar 5.2 Tata bahasa untuk ε_0 , dengan contoh frasa untuk setiap aturan. Kategori sintaksisnya adalah sentence (*S*), nounphrase (*NP*), verb phrase (*VP*), list of adjectives (*Adjs*), prepositional phrase (*PP*), dan relative clause (*RelClause*).



Gambar 5.3 Pohon parse untuk kalimat "Every wumpus smells" menurut tata bahasa ε_0 .

Setiap simpul interior pohon diberi label probabilitasnya. Probabilitas pohon secara keseluruhan adalah $0,9 \times 0,25 \times 0,05 \times 0,15 \times 0,40 \times 0,10 = 0,0000675$. Karena pohon ini merupakan satu-satunya parse kalimat, angka tersebut juga merupakan probabilitas kalimat tersebut. Pohon ini juga dapat ditulis dalam bentuk linear sebagai $[S [NP [Article\ every] [Noun\ wumpus]] [VP [Verb\ smells]]]$.

5.2 ANALISIS SINTAKTIK (PARSING)

Parsing adalah proses menganalisis serangkaian kata untuk mengungkap struktur frasanya, sesuai dengan aturan tata bahasa. Gambar 5.4 menunjukkan bahwa kita dapat memulai dengan simbol S dan mencari dari atas ke bawah untuk pohon yang memiliki kata-kata tersebut sebagai daunnya, atau kita dapat memulai dengan kata-kata tersebut dan mencari dari bawah ke atas untuk pohon yang berpuncak pada S . Namun, baik parsing atas-bawah maupun bawah-atas bisa jadi tidak efisien, karena keduanya dapat berakhir dengan pengulangan upaya di area ruang pencarian yang berujung pada jalan buntu. Perhatikan dua kalimat berikut:

Mintalah siswa di bagian 2 Ilmu Komputer 101 untuk mengikuti ujian.

Apakah siswa di bagian 2 Ilmu Komputer 101 untuk mengikuti ujian?

Meskipun 10 kata pertama mereka sama, kalimat-kalimat ini memiliki parsing yang sangat berbeda, karena yang pertama adalah perintah dan yang kedua adalah pertanyaan. Algoritma penguraian kiri-ke-kanan harus menebak apakah kata pertama merupakan bagian dari perintah atau pertanyaan, dan tidak akan dapat memastikan tebakan tersebut benar hingga setidaknya kata kesebelas, "take" atau "taken". Jika tebakan algoritma salah, ia harus kembali ke kata pertama dan menganalisis ulang seluruh kalimat dengan interpretasi yang berbeda.

<i>List of items</i>	<i>Rule</i>
S	
$NP\ VP$	$S \rightarrow NP\ VP$
$NP\ VP\ Adjective$	$VP \rightarrow VP\ Adjective$
$NP\ Verb\ Adjective$	$VP \rightarrow Verb$
$NP\ Verb\ \mathbf{dead}$	$Adjective \rightarrow \mathbf{dead}$
$NP\ \mathbf{is\ dead}$	$Verb \rightarrow \mathbf{is}$
$Article\ Noun\ \mathbf{is\ dead}$	$NP \rightarrow Article\ Noun$
$Article\ \mathbf{wumpus\ is\ dead}$	$Noun \rightarrow \mathbf{wumpus}$
$\mathbf{the\ wumpus\ is\ dead}$	$Article \rightarrow \mathbf{the}$

Gambar 5.4 Jejak proses pencarian parse untuk string "The wumpus is dead" sebagai kalimat, menurut tata bahasa ε_0 . Dilihat sebagai parse top-down, kita mulai dengan daftar item yang berupa S dan, pada setiap langkah, mencocokkan item X dengan aturan dalam bentuk ($X \rightarrow \dots$) dan mengganti X dalam daftar item dengan ($\dots$). Dilihat sebagai parse bottom-up, kita mulai dengan daftar item yang berupa kata-kata kalimat, dan, pada setiap

langkah, mencocokkan string token (. . .) dalam daftar dengan aturan dalam bentuk ($X \rightarrow \dots$) dan mengganti (. ..) dengan X .

Untuk menghindari sumber inefisiensi ini, kita dapat menggunakan pemrograman dinamis: setiap kali kita menganalisis sub-string, simpan hasilnya agar kita tidak perlu menganalisisnya lagi nanti. Misalnya, setelah kita menemukan bahwa "mahasiswa di bagian 2 Ilmu Komputer 101" adalah NP , kita dapat mencatat hasil tersebut dalam struktur data yang dikenal sebagai bagan. Algoritma yang melakukan hal ini disebut pengurai bagan. Karena kita berurusan dengan tata bahasa bebas konteks, frasa apa pun yang ditemukan dalam konteks satu cabang ruang pencarian dapat berfungsi dengan baik di cabang ruang pencarian lainnya. Ada banyak jenis pengurai bagan; kami menjelaskan versi bottom-up yang disebut algoritma CYK, berdasarkan penemunya, John Cocke, Daniel Younger, dan Tadeo Kasami.

Algoritma CYK ditunjukkan pada Gambar 5.5. Perhatikan bahwa hal ini memerlukan tata bahasa dengan semua aturan dalam salah satu dari dua format yang sangat spesifik: aturan leksikal dalam bentuk $X \rightarrow \text{kata}$, dan aturan sintaksis dalam bentuk $X \rightarrow YZ$. Format tata bahasa ini, yang disebut Bentuk Normal Chomsky, mungkin tampak membatasi, tetapi sebenarnya tidak: tata bahasa bebas konteks apa pun dapat secara otomatis diubah menjadi Bentuk Normal Chomsky.

```
function CYK – PARSE(words, grammar) returns  $P$ , a table of probabilities
   $N \leftarrow \text{LENGTH}(\textit{words})$ 
   $M \leftarrow$  the number of nonterminal symbols in grammar
   $P \leftarrow$  an array of size  $[M, N, N]$ , initially all 0
  /* Insert lexical rules for each word */
  for  $i = 1$  to  $N$  do
    for each rule of form ( $X \rightarrow \textit{words}_i [p]$ ) do
       $P[X, i, 1] \leftarrow p$ 
  /* Combine first and second parts of right – hand sides of rules, from short to long */
  for  $\textit{length} = 2$  to  $N$  do
    for  $\textit{start} = 1$  to  $N - \textit{length} + 1$  do
      for  $\textit{len1} = 1$  to  $N - 1$  do
         $\textit{len2} \leftarrow \textit{length} - \textit{len1}$ 
        for each rule of the form ( $X \rightarrow YZ[p]$ ) do
           $P[X, \textit{start}, \textit{length}] \leftarrow \text{MAX}(P[X, \textit{start}, \textit{length}],$ 
             $P[Y, \textit{start}, \textit{len1}] \times P[Z, \textit{start} + \textit{len1}, \textit{len2}] \times p)$ 
  return  $P$ 
```

Gambar 5.5 Algoritma CYK untuk penguraian. Diberikan sederet kata, algoritma ini menemukan derivasi yang paling mungkin untuk seluruh sederet dan untuk setiap subsederet. Algoritma ini mengembalikan seluruh tabel, P , dengan entri $P[X, \textit{start}, \textit{len}]$ yang merupakan probabilitas X yang paling mungkin dengan panjang $\textit{len}$ yang dimulai dari posisi $\textit{start}$. Jika tidak ada X dengan ukuran tersebut di lokasi tersebut, probabilitasnya adalah 0.

Algoritma CYK menggunakan ruang $O(n^2m)$ untuk tabel P , di mana n adalah jumlah kata dalam kalimat, dan m adalah jumlah simbol nonterminal dalam tata bahasa, dan membutuhkan waktu $O(n^3m)$. (Karena m konstan untuk tata bahasa tertentu, ini umumnya digambarkan sebagai $O(n^3)$.) Tidak ada algoritma yang dapat bekerja lebih baik untuk tata bahasa bebas konteks umum, meskipun ada algoritma yang lebih cepat pada tata bahasa yang lebih terbatas. Faktanya, ini merupakan trik yang cukup sulit bagi algoritma untuk menyelesaikannya dalam waktu $O(n^3)$, mengingat bahwa sebuah kalimat mungkin memiliki jumlah pohon parse yang eksponensial. Perhatikan kalimat

"Daun musim gugur berguguran" dan "daun musim semi berguguran".

Kalimat ini ambigu karena setiap kata (kecuali "dan") dapat berupa kata benda atau kata kerja, dan "musim gugur" serta "musim semi" juga dapat berupa kata sifat. (Misalnya, salah satu arti dari "Daun musim gugur berguguran" setara dengan "Musim gugur meninggalkan musim gugur".) Dengan ε_0 , kalimat tersebut memiliki empat penguraian:

$[S [S [NP \textit{Fall leaves}] \textit{fall}] \textit{and} [S [NP \textit{spring leaves}] \textit{spring}]]$
 $[S [S [NP \textit{Fall leaves}] \textit{fall}] \textit{and} [S \textit{spring} [VP \textit{leaves spring}]]]$
 $[S [S \textit{Fall} [VP \textit{leaves fall}]]] \textit{and} [S [NP \textit{spring leaves}] \textit{spring}]$
 $[S [S \textit{Fall} [VP \textit{leaves fall}]]] \textit{and} [S \textit{spring} [VP \textit{leaves spring}]] .$

Jika kita memiliki c subkalimat gabungan ambigu dua arah, kita akan memiliki 2^c cara memilih parse untuk subkalimat tersebut. Bagaimana algoritma CYK memproses 2^c pohon parse ini dalam waktu $O(c^3)$? Jawabannya adalah algoritma ini tidak memeriksa semua pohon parse; yang harus dilakukan hanyalah menghitung probabilitas pohon yang paling mungkin. Semua sub-pohon direpresentasikan dalam tabel P , dan dengan sedikit usaha, kita dapat mencacah semuanya (dalam waktu eksponensial), tetapi keunggulan algoritma CYK adalah kita tidak perlu mencacahnya kecuali kita mau.

Dalam praktiknya, kita biasanya tidak tertarik pada semua parse; hanya satu atau beberapa parse terbaik. Anggaplah algoritma CYK mendefinisikan ruang keadaan lengkap yang didefinisikan oleh operator "terapkan aturan tata bahasa". Dimungkinkan untuk mencari hanya sebagian dari ruang ini menggunakan pencarian A^* . Setiap status dalam ruang ini adalah daftar item (kata atau kategori), seperti yang ditunjukkan pada tabel parse bottom-up. Status awal adalah daftar kata, dan status tujuan adalah item tunggal S .

Biaya suatu status adalah kebalikan dari probabilitasnya sebagaimana didefinisikan oleh aturan yang diterapkan sejauh ini, dan terdapat berbagai heuristik untuk memperkirakan jarak yang tersisa ke tujuan; heuristik terbaik berasal dari pembelajaran mesin yang diterapkan pada korpus kalimat. Dengan algoritma A^* , kita tidak perlu mencari seluruh ruang status, dan kita dijamin bahwa parse pertama yang ditemukan akan menjadi yang paling mungkin.

```

[ [S [NP – SBJ – 2 Her eyes]
  [VP were
    [VP glazed
      [NP * –2]
      [SBAR – ADV as if
        [S [NP – SBJ she]
          [VP did n't
            [VP [VP hear [NP * –1]]
              or
              [VP [ADVP even] see [NP * –1]]
              [NP – 1 him]]]]]]]]]]
  .]

```

Gambar 5.6 Pohon anotasi untuk kalimat "Her eyes were glazed as if she Didn't hear or even see him" dari Penn Treebank. Perhatikan bahwa dalam tata bahasa ini terdapat perbedaan antara frasa nomina objek (*NP*) dan frasa nomina subjek (*NP – SBJ*). Perhatikan juga fenomena tata bahasa yang belum kita bahas: perpindahan frasa dari satu bagian pohon ke bagian lain. Pohon ini menganalisis frasa "hear or even see him" yang terdiri dari dua VP penyusun, [VP hear [NP *-1]] dan [VP [ADVP even] see [NP *-1]], yang keduanya memiliki objek yang hilang, dilambangkan *-1, yang merujuk pada *NP* yang diberi label di tempat lain dalam pohon sebagai [NP-1 him].

Probabilitas Pembelajaran untuk PCFG

PCFG memiliki banyak aturan, dengan probabilitas untuk setiap aturan. Hal ini menunjukkan bahwa mempelajari tata bahasa dari data mungkin lebih baik daripada pendekatan rekayasa pengetahuan. Pembelajaran paling mudah jika kita diberikan korpus kalimat yang diurai dengan benar, yang biasa disebut treebank. Penn Treebank adalah yang paling terkenal; terdiri dari 3 juta kata yang telah dianotasi dengan kelas kata dan struktur pohon parse, menggunakan tenaga manusia yang dibantu oleh beberapa alat otomatis. Gambar 5.6 menunjukkan pohon beranotasi dari Penn Treebank.

Diberikan korpus pohon, kita dapat membuat PCFG hanya dengan menghitung (dan menghaluskan). Dalam contoh di atas, terdapat dua simpul dengan bentuk $[S [NP \dots][VP \dots]]$. Kita akan menghitung simpul-simpul ini, dan semua sub-pohon lainnya dengan akar *S* di dalam korpus. Bila terdapat 100.000 simpul *S* yang 60.000 diantaranya berbentuk seperti ini, maka kita buat aturan:

$$S \rightarrow NP VP [0.60].$$

Bagaimana jika treebank tidak tersedia, tetapi kita memiliki korpus kalimat mentah tanpa label? Mempelajari tata bahasa dari korpus semacam itu masih mungkin, tetapi lebih sulit. Pertama-tama, kita sebenarnya memiliki dua masalah: mempelajari struktur aturan tata

bahasa dan mempelajari probabilitas yang terkait dengan setiap aturan. (Kita memiliki perbedaan yang sama dalam mempelajari jaringan Bayes.)

Kita akan berasumsi bahwa kita diberi nama kategori leksikal dan sintaksis. (Jika tidak, kita dapat mengasumsikan kategori $X_1, \dots, X_n$ dan menggunakan validasi silang untuk memilih nilai n terbaik.) Kita kemudian dapat mengasumsikan bahwa tata bahasa mencakup setiap kemungkinan aturan ($X \rightarrow YZ$) atau ($X \rightarrow \text{word}$), meskipun banyak dari aturan ini akan memiliki probabilitas 0 atau mendekati 0.

Kita kemudian dapat menggunakan pendekatan ekspektasi-maksimisasi (EM), seperti yang kita lakukan dalam mempelajari HMM. Parameter yang kita coba pelajari adalah probabilitas aturan; kita memulainya dengan nilai acak atau seragam. Variabel tersembunyi adalah pohon parse: kita tidak tahu apakah string kata $w_i \dots w_j$ dihasilkan atau tidak oleh suatu aturan ($X \rightarrow \dots$). Langkah E memperkirakan probabilitas bahwa setiap suburutan dihasilkan oleh setiap aturan. Langkah M kemudian memperkirakan probabilitas setiap aturan. Seluruh komputasi dapat dilakukan dalam mode pemrograman dinamis dengan algoritma yang disebut algoritma dalam-luar yang analog dengan algoritma maju-mundur untuk HMM. Algoritma dalam-luar tampak ajaib karena menginduksi tata bahasa dari teks yang tidak diurai. Namun, algoritma ini memiliki beberapa kelemahan. Pertama, parse yang ditetapkan oleh tata bahasa yang diinduksi seringkali sulit dipahami dan tidak memuaskan bagi ahli bahasa.

Hal ini menyulitkan untuk menggabungkan pengetahuan buatan tangan dengan induksi otomatis. Kedua, algoritma ini lambat: $O(n^3 m^3)$, di mana n adalah jumlah kata dalam kalimat dan m adalah jumlah kategori tata bahasa. Ketiga, ruang lingkup penugasan probabilitas sangat luas, dan secara empiris tampaknya terjebak dalam maksima lokal merupakan masalah yang serius. Alternatif seperti simulasi anil dapat mendekati maksimum global, dengan biaya komputasi yang lebih besar. Lari dan Young menyimpulkan bahwa rasio inside-outside "secara komputasional sulit untuk permasalahan realistis." Namun, kemajuan dapat dicapai jika kita bersedia melangkah keluar dari batasan pembelajaran hanya dari teks yang belum diurai. Salah satu pendekatannya adalah belajar dari prototipe: memulai proses dengan selusin atau dua aturan, serupa dengan aturan di ε_1 .

Dari sana, aturan yang lebih kompleks dapat dipelajari dengan lebih mudah, dan tata bahasa yang dihasilkan mengurai bahasa Inggris dengan tingkat ingatan dan presisi kalimat keseluruhan sekitar 80%. Pendekatan lain adalah menggunakan treebank, tetapi selain mempelajari aturan PCFG langsung dari tanda kurung, juga mempelajari perbedaan yang tidak terdapat di treebank. Misalnya, bukan berarti pohon pada Gambar 5.6 membedakan antara NP dan $NP - SBJ$. $NP - SBJ$ digunakan untuk kata ganti "dia", sedangkan NP digunakan untuk kata ganti "dia".

Kita akan membahas masalah ini di Bagian 23.6; untuk saat ini, mari kita katakan saja bahwa ada banyak cara yang berguna untuk membagi kategori seperti NP —sistem induksi tata bahasa yang menggunakan treebank tetapi secara otomatis membagi kategori lebih baik daripada sistem yang tetap menggunakan set kategori asli. Tingkat kesalahan untuk tata bahasa yang dipelajari secara otomatis masih sekitar 50% lebih tinggi daripada tata bahasa yang dibangun secara manual, tetapi kesenjangannya semakin mengecil.

Membandingkan Model Bebas Konteks dan Model Markov

Masalah dengan PCFG adalah bahwa keduanya bebas konteks. Artinya, perbedaan antara $P(\text{"eat a banana"})$ dan $P(\text{"eat a bandana"})$ hanya bergantung pada $P(\text{Noun} \rightarrow \text{"banana"})$ versus $P(\text{Noun} \rightarrow \text{"bandana"})$, dan bukan pada hubungan antara "eat" dan objek masing-masing. Model Markov orde dua atau lebih, dengan korpus yang cukup besar, akan mengetahui bahwa "eat a banana" lebih mungkin.

Kita dapat menggabungkan PCFG dan model Markov untuk mendapatkan yang terbaik dari keduanya. Pendekatan paling sederhana adalah memperkirakan probabilitas sebuah kalimat dengan rata-rata geometrik probabilitas yang dihitung oleh kedua model. Dengan demikian, kita akan mengetahui bahwa "eat a banana" mungkin dari sudut pandang gramatikal dan leksikal. Namun, model tersebut tetap tidak dapat menangkap hubungan antara "eat" dan "banana" dalam "makan pisang yang agak tua tetapi masih enak" karena di sini hubungannya berjarak lebih dari dua kata. Meningkatkan urutan model Markov tidak akan mencapai hubungan tersebut secara tepat; untuk melakukannya, kita dapat menggunakan PCFG yang dileksikalisasi, seperti yang dijelaskan di bagian selanjutnya.

Masalah lain dengan PCFG adalah mereka cenderung terlalu menyukai kalimat yang lebih pendek. Dalam korpus seperti *Wall Street Journal*, panjang rata-rata sebuah kalimat adalah sekitar 25 kata. Namun, PCFG biasanya akan menetapkan probabilitas yang cukup tinggi untuk banyak kalimat pendek, seperti "Dia tidur," sedangkan dalam Jurnal, kita lebih cenderung melihat sesuatu seperti "Telah dilaporkan oleh sumber yang dapat dipercaya bahwa tuduhan bahwa dia tidur dapat dipercaya." Tampaknya frasa-frasa dalam Jurnal sebenarnya tidak bebas konteks; sebaliknya, penulis memiliki gambaran tentang panjang kalimat yang diharapkan dan menggunakan panjang tersebut sebagai batasan global yang lunak pada kalimat mereka. Hal ini sulit tercermin dalam PCFG.

5.3 TATA BAHASA YANG DITAMBAHKAN DAN INTERPRETASI SEMANTIK

Di bagian ini, kita akan melihat cara memperluas tata bahasa bebas konteks—misalnya, tidak setiap *NP* independen dari konteks, melainkan, *NP* tertentu lebih mungkin muncul dalam satu konteks, dan yang lainnya dalam konteks lain.

PCFG Leksikal

Untuk memahami hubungan antara verba "eat" dan nomina "banana" versus "bandanna", kita dapat menggunakan PCFG leksikal, yang probabilitasnya untuk suatu aturan bergantung pada hubungan antarkata dalam pohon parse, bukan hanya pada kedekatan kata dalam kalimat. Tentu saja, kita tidak dapat membuat probabilitas bergantung pada setiap kata dalam pohon parse, karena kita tidak akan memiliki cukup data pelatihan untuk memperkirakan semua probabilitas tersebut. Akan berguna untuk memperkenalkan gagasan kepala frasa—kata yang paling penting.

Dengan demikian, "eat" adalah kepala dari *VP* "eat a banana" dan "banana" adalah kepala *NP* "banana". Notasi $VP(v)$ digunakan untuk menunjukkan frasa dengan kategori *VP* yang kata utamanya adalah v . Kita katakan bahwa kategori *VP* tersebut di-augmentasi dengan

variabel utama v . Berikut adalah tata bahasa augmentasi yang menggambarkan relasi verba-objek:

$$\begin{array}{ll}
 VP(v) \rightarrow Verb(v) NP(n) & [P_1(v, n)] \\
 VP(v) \rightarrow Verb(v) & [P_2(v)] \\
 NP(n) \rightarrow Article(a) Adjs(j) Noun(n) & [P_3(n, a)] \\
 Noun(banana) \rightarrow banana & [p_n] \\
 \dots & \dots
 \end{array}$$

Di sini, probabilitas $P_1(v, n)$ bergantung pada kata kepala v dan n . Kita akan menetapkan probabilitas ini relatif tinggi ketika v adalah "eat" dan n adalah "banana," dan rendah ketika n adalah "bandanna."

Perhatikan bahwa karena kita hanya mempertimbangkan kepala, perbedaan antara "eat a banana" dan "eat a ranchid banana" tidak akan tertangkap oleh probabilitas ini. Masalah lain dengan pendekatan ini adalah bahwa, dalam kosakata dengan, katakanlah, 20.000 kata benda dan 5.000 kata kerja, P_1 membutuhkan 100 juta estimasi probabilitas. Hanya beberapa persen dari ini yang dapat berasal dari korpus; sisanya harus berasal dari penghalusan. Misalnya, kita dapat memperkirakan $P_1(v, n)$ untuk pasangan (v, n) yang belum sering kita lihat (atau sama sekali) dengan mundur ke model yang hanya bergantung pada v . Probabilitas tanpa objek ini masih sangat berguna; mereka dapat menangkap perbedaan antara kata kerja transitif seperti "eat" - yang akan memiliki nilai tinggi untuk P_1 dan nilai rendah untuk P_2 - dan kata kerja intransitif seperti "sleep," yang akan memiliki kebalikannya. Sangat layak untuk mempelajari probabilitas ini dari treebank.

Definisi Formal Aturan Tata Bahasa Augmented

Aturan augmented itu rumit, jadi kami akan memberikan definisi formalnya dengan menunjukkan bagaimana aturan augmented dapat diterjemahkan menjadi kalimat yang logis. Kalimat tersebut akan berbentuk klausa pasti, sehingga hasilnya disebut tata bahasa klausa pasti, atau DCG. Sebagai contoh, kami akan menggunakan versi aturan dari tata bahasa leksikal untuk NP dengan satu notasi baru:

$$NP(n) \rightarrow Article(a) Adjs(j) Noun(n) \{Compatible(j, n)\}.$$

Aspek baru di sini adalah notasi $\{constraint\}$ untuk menunjukkan kendala logis pada beberapa variabel; aturan ini hanya berlaku jika kendala tersebut benar. Di sini, predikat $Compatible(j, n)$ dimaksudkan untuk menguji apakah kata sifat j dan kata benda n kompatibel; predikat ini akan didefinisikan oleh serangkaian pernyataan seperti $Compatible(black, dog)$.

Kita dapat mengubah aturan tata bahasa ini menjadi klausa pasti dengan (1) membalik urutan ruas kanan dan kiri, (2) membuat konjungsi semua konstituen dan kendala, (3) menambahkan variabel s_i ke daftar argumen untuk setiap konstituen untuk mewakili urutan

kata yang direntang oleh konstituen tersebut, (4) menambahkan istilah untuk penggabungan kata, $Append(s_1, \dots)$, ke daftar argumen untuk akar pohon. Hal ini memberi kita

$$Article(a, s_1) \wedge Adjs(j, s_2) \wedge Noun(n, s_3) \wedge Compatible(j, n) \\ \Rightarrow NP(n, Append(s_1, s_2, s_3)) .$$

Klausa pasti ini menyatakan bahwa jika predikat *Article* berlaku untuk kata utama a dan string s_1 , dan *Adjs* berlaku pula untuk kata utama j dan string s_2 , dan *Noun* berlaku untuk kata utama n dan string s_3 , dan jika j dan n kompatibel, maka predikat *NP* berlaku untuk kata utama n dan merupakan hasil penambahan string s_1 , s_2 , dan s_3 .

Translasi DCG menghilangkan probabilitas, tetapi kita dapat mengembalikannya: cukup tambahkan setiap konstituen dengan satu variabel lagi yang mewakili probabilitas konstituen, dan tambahkan akar dengan variabel yang merupakan hasil perkalian probabilitas konstituen dengan probabilitas aturan.

Translasi dari aturan tata bahasa ke klausa pasti memungkinkan kita untuk membicarakan penguraian sebagai inferensi logis. Hal ini memungkinkan penalaran tentang bahasa dan string dengan berbagai cara. Misalnya, ini berarti kita dapat melakukan penguraian bottom-up menggunakan forward chaining atau penguraian top-down menggunakan backward chaining. Faktanya, penguraian bahasa alami dengan DCG merupakan salah satu aplikasi pertama bahasa pemrograman logika Prolog. Terkadang, proses ini dapat dijalankan secara terbalik dan menghasilkan bahasa sekaligus melakukan penguraian. Misalnya, dengan melompat ke Gambar 5.10, sebuah program logika dapat diberi bentuk semantik $Loves(John, Mary)$ dan menerapkan aturan klausa pasti untuk menyimpulkan

$$S(Loves(John, Mary), [John, loves, Mary]) .$$

Hal ini berfungsi untuk contoh mainan, tetapi sistem pembangkitan bahasa yang serius memerlukan kontrol lebih besar atas proses tersebut dibandingkan yang diberikan oleh aturan DCG saja.

$\mathcal{E}_1 :$	$S \rightarrow NP_S VP \mid \dots$
	$NP_S \rightarrow Pronoun_S \mid Name \mid Noun \mid \dots$
	$NP_O \rightarrow Pronoun_O \mid Name \mid Noun \mid \dots$
	$VP \rightarrow VP NP_O \mid \dots$
	$PP \rightarrow Prep NP_O$
	$Pronoun_S \rightarrow \mathbf{I} \mid \mathbf{you} \mid \mathbf{he} \mid \mathbf{she} \mid \mathbf{it} \mid \dots$
	$Pronoun_O \rightarrow \mathbf{me} \mid \mathbf{you} \mid \mathbf{him} \mid \mathbf{her} \mid \mathbf{it} \mid \dots$
	$\dots$
$\mathcal{E}_2 :$	$S(head) \rightarrow NP(Sbj, pn, h) VP(pn, head) \mid \dots$
	$NP(c, pn, head) \rightarrow Pronoun(c, pn, head) \mid Noun(c, pn, head) \mid \dots$
	$VP(pn, head) \rightarrow VP(pn, head) NP(Obj, p, h) \mid \dots$
	$PP(head) \rightarrow Prep(head) NP(Obj, pn, h)$
	$Pronoun(Sbj, 1S, \mathbf{I}) \rightarrow \mathbf{I}$
	$Pronoun(Sbj, 1P, \mathbf{we}) \rightarrow \mathbf{we}$
	$Pronoun(Obj, 1S, \mathbf{me}) \rightarrow \mathbf{me}$
	$Pronoun(Obj, 3P, \mathbf{them}) \rightarrow \mathbf{them}$
	$\dots$

Gambar 5.7 Atas: bagian tata bahasa untuk bahasa $\mathcal{E}_1$, yang menangani kasus subjektif dan objektif dalam frasa nomina sehingga tidak menghasilkan kata kerja secara berlebihan seburuk $\mathcal{E}_0$. Bagian yang identik dengan $\mathcal{E}_0$ telah dihilangkan. Bawah: bagian tata bahasa ditambah untuk $\mathcal{E}_2$, dengan tiga penambahan: kesesuaian kasus, kesesuaian subjek-verba, dan kata dasar. *Sbj*, *Obj*, *1S*, *1P*, dan *3P* adalah konstanta, dan nama dengan huruf kecil adalah variabel.

Kesesuaian Kasus dan Kesesuaian Subjek-Verba

Kita telah melihat bahwa tata bahasa sederhana untuk $\mathcal{E}_0$ menghasilkan kata ganti yang berlebihan, menghasilkan kata ganti yang tidak sesuai seperti "Saya mencium bau busuk." Untuk menghindari masalah ini, tata bahasa kita harus mengetahui bahwa "saya" bukanlah *NP* yang valid ketika menjadi subjek kalimat.

Para ahli bahasa mengatakan bahwa kata ganti "Saya" berada dalam kasus subjektif, dan "saya" berada dalam kasus objektif. Kita dapat menjelaskan hal ini dengan membagi *NP* menjadi dua kategori, NP_S dan NP_O , yang masing-masing mewakili frasa nomina dalam kasus subjektif dan objektif. Kita juga perlu membagi kategori *Pronoun* menjadi dua kategori, *Pronoun_S* (yang mencakup "Saya") dan *Pronoun_O* (yang mencakup "saya"). Bagian atas Gambar 5.7 menunjukkan tata bahasa untuk kesesuaian kasus; kita menyebut bahasa yang dihasilkan $\mathcal{E}_1$. Perhatikan bahwa semua aturan *NP* harus diduplikasi, sekali untuk NP_S dan sekali untuk NP_O .

Sayangnya, $\mathcal{E}_1$ masih menghasilkan kata ganti yang berlebihan. Bahasa Inggris mensyaratkan kesesuaian subjek-verba untuk orang dan bilangan dari subjek dan verba utama sebuah kalimat. Misalnya, jika "I" adalah subjeknya, maka "I smell" bersifat gramatikal, tetapi "I smells" tidak. Jika "it" adalah subjeknya, kita mendapatkan kebalikannya. Dalam bahasa

Inggris, perbedaan kesesuaiannya minimal: kebanyakan verba memiliki satu bentuk untuk subjek orang ketiga tunggal (he, she, atau it), dan bentuk kedua untuk semua kombinasi orang dan bilangan lainnya.

Ada satu pengecualian: verb "to be" memiliki tiga bentuk, "I am / you are / he is." Jadi, satu perbedaan (kasus) membagi *NP* menjadi dua cara, perbedaan lain (orang dan bilangan) membagi *NP* menjadi tiga cara, dan ketika kita menemukan perbedaan lain, kita akan mendapatkan jumlah bentuk *NP* yang bersubskrip secara eksponensial jika kita mengambil pendekatan ε_1 . Augmentasi adalah pendekatan yang lebih baik: augmentasi dapat merepresentasikan jumlah bentuk yang eksponensial sebagai satu aturan.

Di bagian bawah Gambar 5.7, kita melihat (sebagian dari) tata bahasa augmented untuk bahasa ε_2 , yang menangani kesesuaian kasus, kesesuaian subjek-verba, dan kata-kata utama. Kita hanya memiliki satu kategori *NP*, tetapi $NP(c, pn, head)$ memiliki tiga augmentasi: *c* adalah parameter untuk kasus, *pn* adalah parameter untuk orang dan angka, dan *head* adalah parameter untuk kata utama frasa. Kategori-kategori lain juga augmented dengan heads dan argumen lainnya. Mari kita bahas satu aturan secara detail:

$$S(head) \rightarrow NP(Sbj, pn, h) VP(pn, head).$$

Aturan ini paling mudah dipahami dari kanan ke kiri: ketika sebuah *NP* dan sebuah *VP* dikonjungsi, keduanya membentuk sebuah *S*, tetapi hanya jika *NP* tersebut memiliki kasus subjektif (*Sbj*) dan orang serta nomor (*pn*) dari *NP* dan *VP* identik. Jika hal ini berlaku, maka kita memiliki sebuah *S* yang kepalanya sama dengan kepala *VP*. Perhatikan bahwa kepala *NP*, yang dilambangkan dengan variabel dummy *h*, bukan bagian dari augmentasi *S*. Aturan leksikal untuk E_2 mengisi nilai parameter dan juga paling baik dibaca dari kanan ke kiri. Misalnya, aturan

$$Pronoun(Sbj, 1S, I) \rightarrow I$$

menyatakan bahwa "I" dapat diartikan sebagai *Pronoun* dalam kasus subjektif, orang pertama tunggal, dengan kepala "I." Untuk penyederhanaan, kami telah menghilangkan probabilitas untuk aturan-aturan ini, tetapi augmentasi memang berfungsi dengan probabilitas. Augmentasi juga dapat bekerja dengan mekanisme pembelajaran otomatis. Petrov dan Klein (2007c) menunjukkan bagaimana algoritma pembelajaran dapat secara otomatis membagi kategori *NP* menjadi NP_S dan NP_O .

Interpretasi Semantik

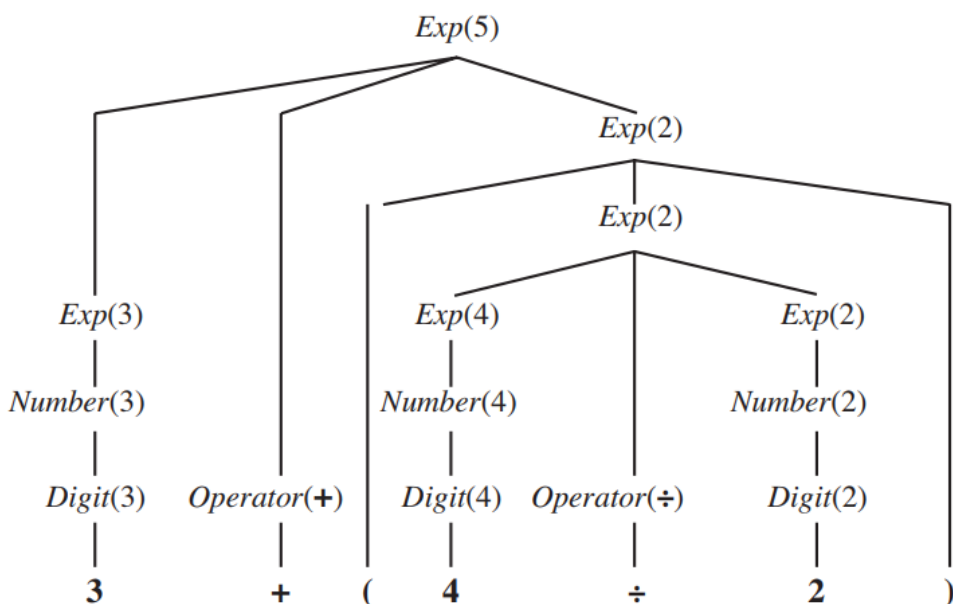
Untuk menunjukkan cara menambahkan semantik ke tata bahasa, kita mulai dengan contoh yang lebih sederhana daripada bahasa Inggris: semantik ekspresi aritmatika. Gambar 5.8 menunjukkan tata bahasa untuk ekspresi aritmatika, di mana setiap aturan ditambah dengan variabel yang menunjukkan interpretasi semantik frasa tersebut. Semantik digit seperti "3" adalah digit itu sendiri. Semantik ekspresi seperti "3 + 4" adalah operator "+" yang

diterapkan pada semantik frasa "3" dan frasa "4." Aturan-aturan tersebut mematuhi prinsip semantik komposisi—semantik sebuah frasa merupakan fungsi dari semantik subfrasa. Gambar 5.9 menunjukkan pohon parse untuk $3 + (4 \div 2)$ menurut tata bahasa ini. Akar dari pohon parse adalah $Exp(5)$, sebuah ekspresi yang interpretasi semantiknya adalah 5.

$Exp(x) \rightarrow Exp(x_1)Operator(op)Exp(x_2)\{x = Apply(op, x_1, x_2)\}$
 $Exp(x) \rightarrow (Exp(x))$
 $Exp(x) \rightarrow Number(x)$
 $Number(x) \rightarrow Digit(x)$
 $Number(x) \rightarrow Number(x_1)Digit(x_2)\{x = 10 \times x_1 + x_2\}$
 $Digit(x) \rightarrow x \{0 \leq x \leq 9\}$
 $Operator(x) \rightarrow x \{x \in \{+, -, \div, \times\}\}$

Gambar 5.8 Tata bahasa untuk ekspresi aritmatika, ditambah dengan semantik. Setiap variabel x_i merepresentasikan semantik suatu konstituen. Perhatikan penggunaan notasi $\{test\}$ untuk mendefinisikan predikat logika yang harus dipenuhi, tetapi bukan merupakan konstituen.

Sekarang mari kita beralih ke semantik bahasa Inggris, atau setidaknya ε_0 . Kita mulai dengan menentukan representasi semantik apa yang ingin kita kaitkan dengan frasa apa. Kita menggunakan kalimat contoh sederhana "John loves Mary." *NP* "John" harus memiliki interpretasi semantiknya berupa istilah logis *John*, dan kalimat secara keseluruhan harus memiliki interpretasinya berupa kalimat logis *Loves(John, Mary)*. Itu tampak jelas. Bagian yang rumit adalah *VP* "loves Mary."



Gambar 5.9 Pohon parse dengan interpretasi semantik untuk string " $3 + (4 \div 2)$ ".

Interpretasi semantik dari frasa ini bukanlah istilah logis maupun kalimat logis yang lengkap. Secara intuitif, "loves Mary" adalah deskripsi yang mungkin berlaku atau tidak berlaku untuk

orang tertentu. Artinya, "mencintai Maria" adalah predikat yang, jika digabungkan dengan istilah yang mewakili seseorang (orang yang mencintai), menghasilkan kalimat logis yang lengkap. Dengan menggunakan notasi λ , kita dapat merepresentasikan "mencintai Maria" sebagai predikat.

$$\lambda x \text{ Loves}(x, \text{Mary}) .$$

Sekarang kita memerlukan aturan yang menyatakan “*NP* dengan semantik *obj* diikuti oleh *VP* dengan semantik *pred* menghasilkan kalimat yang semantiknya merupakan hasil penerapan *pred* ke *obj* :”

$$S(\text{pred}(\text{obj})) \rightarrow NP(\text{obj}) VP(\text{pred}) .$$

Aturan tersebut memberi tahu kita bahwa interpretasi semantik dari “John mencintai Mary” adalah

$$(\lambda x \text{ Loves}(x, \text{Mary}))(\text{John}) ,$$

yang setara dengan $\text{Loves}(\text{John}, \text{Mary})$.

Sisa semantik mengikuti secara lugas pilihan yang telah kita buat sejauh ini. Karena *VP* direpresentasikan sebagai predikat, sebaiknya konsisten dan merepresentasikan verba sebagai predikat juga. Verba "loves" direpresentasikan sebagai $\lambda y \lambda x \text{ Loves}(x, y)$, predikat yang, ketika diberikan argumen *Mary*, mengembalikan predikat $\lambda x \text{ Loves}(x, \text{Mary})$. Kita berakhir dengan tata bahasa yang ditunjukkan pada Gambar 5.10 dan pohon parse yang ditunjukkan pada Gambar 5.11. Kita bisa saja menambahkan semantik ke ε_2 ; kita memilih untuk bekerja dengan ε_0 agar pembaca dapat fokus pada satu jenis augmentasi pada satu waktu.

Menambahkan augmentasi semantik ke tata bahasa secara manual itu melelahkan dan rawan kesalahan. Oleh karena itu, ada beberapa proyek untuk mempelajari augmentasi semantik dari contoh. CHILL adalah program pemrograman logika induktif (ILP) yang mempelajari tata bahasa dan parser khusus untuk tata bahasa tersebut dari contoh-contoh. Domain targetnya adalah kueri basis data bahasa alami. Contoh-contoh pelatihan terdiri dari pasangan rangkaian kata dan bentuk semantik yang sesuai—misalnya;

Apa ibu kota negara bagian dengan populasi terbesar?

$$\text{Answer}(c, \text{Capital}(s, c) \wedge \text{Largest}(p, \text{State}(s) \wedge \text{Population}(s, p)))$$

Tugas CHILL adalah mempelajari predikat $\text{Parse}(\text{words}, \text{semantics})$ yang konsisten dengan contoh dan, diharapkan, dapat digeneralisasi dengan baik ke contoh lain. Menerapkan ILP secara langsung untuk mempelajari predikat ini menghasilkan kinerja yang buruk: parser yang diinduksi hanya memiliki akurasi sekitar 20%.

Untungnya, pembelajar ILP dapat meningkatkan kinerja dengan menambahkan pengetahuan. Dalam kasus ini, sebagian besar predikat *Parse* didefinisikan sebagai program logika, dan tugas CHILL direduksi menjadi menginduksi aturan kontrol yang memandu parser untuk memilih satu parse di atas parse lainnya. Dengan pengetahuan latar belakang tambahan ini, CHILL dapat belajar mencapai akurasi 70% hingga 85% pada berbagai tugas kueri basis data.

Komplikasi

Tata bahasa Inggris yang sebenarnya sangatlah kompleks. Kami akan membahas beberapa contoh secara singkat. Waktu dan kala: Misalkan kita ingin merepresentasikan perbedaan antara "John loves Mary" dan "John loved Mary." Bahasa Inggris menggunakan kala verba (lampau, sekarang, dan masa depan) untuk menunjukkan waktu relatif suatu peristiwa. Salah satu pilihan yang baik untuk merepresentasikan waktu kejadian adalah notasi kalkulus kejadian. Dalam kalkulus kejadian, kita memiliki

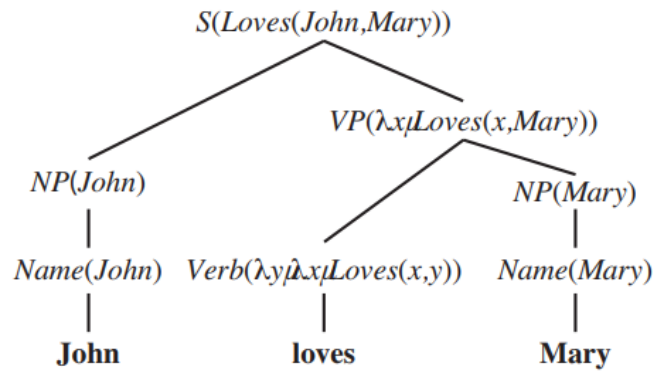
$$\begin{aligned}\text{John loves mary: } E_1 &\in \text{Loves}(\text{John}, \text{Mary}) \wedge \text{During}(\text{Now}, \text{Extent}(E_1)) \\ \text{John loved mary: } E_2 &\in \text{Loves}(\text{John}, \text{Mary}) \wedge \text{After}(\text{Now}, \text{Extent}(E_2)) .\end{aligned}$$

Hal ini menunjukkan bahwa dua aturan leksikal kita untuk kata "mencintai" dan "dicintai" seharusnya adalah sebagai berikut:

$$\begin{aligned}\text{Verb}(\lambda y \lambda x e \in \text{Loves}(x, y) \wedge \text{During}(\text{Now}, e)) &\rightarrow \text{loves} \\ \text{Verb}(\lambda y \lambda x e \in \text{Loves}(x, y) \wedge \text{After}(\text{Now}, e)) &\rightarrow \text{loved} .\end{aligned}$$

$$\begin{aligned}S(\text{pred}(\text{obj})) &\rightarrow \text{NP}(\text{obj})\text{VP}(\text{pred}) \\ \text{VP}(\text{pred}(\text{obj})) &\rightarrow \text{Verb}(\text{pred})\text{NP}(\text{obj}) \\ \text{NP}(\text{obj}) &\rightarrow \text{Name}(\text{obj}) \\ \text{Name}(\text{John}) &\rightarrow \mathbf{John} \\ \text{Name}(\text{Mary}) &\rightarrow \mathbf{Mary} \\ \text{Verb}(\lambda y \lambda x \text{Loves}(x, y)) &\rightarrow \mathbf{loves}\end{aligned}$$

Gambar 5.10 Tata bahasa yang dapat menghasilkan pohon parse dan interpretasi semantik untuk "John loves Mary" (dan tiga kalimat lainnya). Setiap kategori dilengkapi dengan satu argumen yang mewakili semantiknya.



Gambar 5.11 Pohon parse dengan interpretasi semantik untuk string “John loves Mary”.

Bahasa Indonesia: Selain perubahan ini, semua hal lain tentang tata bahasa tetap sama, yang merupakan berita yang menggembirakan; ini menunjukkan bahwa kita berada di jalur yang benar jika kita dapat dengan mudah menambahkan komplikasi seperti tense kata kerja (meskipun kita baru saja menggores permukaan tata bahasa yang lengkap untuk waktu dan tense).

Hal yang menggembirakan juga adalah bahwa perbedaan antara proses dan peristiwa diskrit yang kita buat dalam diskusi kita tentang representasi pengetahuan sebenarnya tercermin dalam penggunaan bahasa. Kita dapat mengatakan “John banyak tidur tadi malam,” di mana *Sleeping* adalah kategori proses, tetapi aneh untuk mengatakan “John banyak menemukan unicorn tadi malam,” di mana *Finding* adalah kategori peristiwa diskrit. Tata bahasa akan mencerminkan fakta itu dengan memiliki probabilitas rendah untuk menambahkan frasa adverbial “banyak” ke peristiwa diskrit.

Kuantifikasi: Perhatikan kalimat “Setiap agen terasa mudah.” Kalimat tersebut hanya memiliki satu parse sintaksis di bawah ε_0 , tetapi sebenarnya ambigu secara semantik; makna yang lebih disukai adalah “Untuk setiap agen ada angin sepoi-sepoi yang dirasakan oleh agen tersebut,” namun makna alternatif yang dapat diterima adalah “Ada angin sepoi-sepoi yang dirasakan oleh setiap agen.” Kedua interpretasi tersebut dapat direpresentasikan sebagai

$$\begin{aligned} \forall a \, a \in \text{Agents} &\Rightarrow \\ &\exists b \, b \in \text{Breezes} \wedge \exists e \, e \in \text{Feel}(a, b) \wedge \text{During}(\text{Now}, e); \\ \exists b \, b \in \text{Breezes} \forall a \, a \in \text{Agents} &\Rightarrow \\ &\exists e \, e \in \text{Feel}(a, b) \wedge \text{During}(\text{Now}, e). \end{aligned}$$

Pendekatan standar untuk kuantifikasi adalah tata bahasa mendefinisikan bukan kalimat semantik logis yang sebenarnya, melainkan bentuk kuasi-logis yang kemudian diubah menjadi kalimat logis oleh algoritma di luar proses penguraian. Algoritma tersebut dapat memiliki aturan preferensi untuk lebih memilih satu cakupan kuantifier daripada yang lain—preferensi yang tidak perlu tercermin langsung dalam tata bahasa.

Pragmatik: Kami telah menunjukkan bagaimana seorang agen dapat memahami serangkaian kata dan menggunakan tata bahasa untuk mendapatkan serangkaian

kemungkinan interpretasi semantik. Sekarang kami membahas masalah melengkapi interpretasi dengan menambahkan informasi yang bergantung pada konteks tentang situasi saat ini. Kebutuhan paling jelas akan informasi pragmatik adalah dalam menyelesaikan makna indeksikal, yaitu frasa yang merujuk langsung ke situasi saat ini.

Misalnya, dalam kalimat "Saya di Boston hari ini," baik "Saya" maupun "hari ini" adalah indeksikal. Kata "Saya" akan diwakili oleh *Speaker* yang fasih, dan terserah kepada pendengar untuk menyelesaikan makna dari fasih tersebut—hal itu tidak dianggap sebagai bagian dari tata bahasa melainkan merupakan masalah pragmatik; penggunaan konteks situasi terkini untuk menafsirkan fasih.

Bagian lain dari pragmatik adalah menafsirkan maksud pembicara. Tindakan pembicara dianggap sebagai tindak tutur, dan pendengarlah yang menentukan jenis tindakannya—pertanyaan, pernyataan, janji, peringatan, perintah, dan sebagainya. Perintah seperti "pergi ke 2 2" secara implisit merujuk pada pendengar. Sejauh ini, tata bahasa kita untuk *S* hanya mencakup kalimat deklaratif. Kita dapat dengan mudah memperluasnya untuk mencakup perintah. Perintah dapat dibentuk dari sebuah *VP*, di mana subjeknya secara implisit adalah pendengar. Kita perlu membedakan perintah dari pernyataan, jadi kita mengubah aturan untuk *S* agar mencakup jenis tindak tutur:

$$\begin{aligned} S(\text{Statement}(\text{Speaker}, \text{pred}(\text{obj}))) &\rightarrow NP(\text{obj}) VP(\text{pred}) \\ S(\text{Command}(\text{Speaker}, \text{pred}(\text{Hearer}))) &\rightarrow VP(\text{pred}) . \end{aligned}$$

Ketergantungan jarak jauh: Pertanyaan memperkenalkan kompleksitas tata bahasa yang baru. Dalam kalimat "Kepada siapa agen itu meminta Anda memberikan emas itu?", kata terakhir "kepada" harus diurai sebagai [*PP to* —], di mana "—" menunjukkan celah atau jejak di mana sebuah *NP* hilang; *NP* yang hilang ditandai dengan kata pertama kalimat, "siapa."

Sistem augmentasi yang kompleks digunakan untuk memastikan bahwa *NP* yang hilang cocok dengan kata-kata yang diberi lisensi dengan cara yang tepat, dan mencegah celah di tempat yang salah. Misalnya, Anda tidak dapat memiliki celah di salah satu cabang konjungsi *NP*: "Apa yang dia mainkan [*NP Dungeons and* _]?" tidak gramatikal. Namun, Anda dapat memiliki celah yang sama di kedua cabang konjungsi *VP*: "Apa yang Anda [*VP* [*VP smell* —] dan [*VP tembakkan panah ke* —]]?"

Ambiguitas: Dalam beberapa kasus, pendengar secara sadar menyadari ambiguitas dalam sebuah tuturan. Berikut beberapa contoh yang diambil dari tajuk utama surat kabar:

Tim membantu korban gigitan anjing.

Polisi memulai kampanye untuk menindak pejalan kaki yang menyeberang jalan sembarangan.

Helikopter bertenaga lalat manusia.

Industri popok kain yang dulu lesu terselamatkan oleh tempat pembuangan sampah yang penuh.

Toilet portabel dibom; polisi tidak punya bukti apa pun.

Guru memukul anak-anak yang malas.

Libatkan anak-anak Anda saat membuat kue.

Rumah sakit dituntut oleh dokter setinggi 7 kaki.

Peminum susu beralih ke susu bubuk.

Pakar keselamatan mengatakan penumpang bus sekolah harus diikat dengan sabuk pengaman.

Namun, sebagian besar bahasa yang kita dengar tampak tidak ambigu. Jadi, ketika para peneliti pertama kali mulai menggunakan komputer untuk menganalisis bahasa pada tahun 1960an, mereka cukup terkejut mengetahui bahwa hampir setiap ucapan sangat ambigu, meskipun interpretasi alternatifnya mungkin tidak terlihat oleh penutur asli.

Sistem dengan tata bahasa dan leksikon yang luas mungkin menemukan ribuan interpretasi untuk kalimat yang sangat biasa. Ambiguitas leksikal, di mana sebuah kata memiliki lebih dari satu makna, cukup umum; "Kembali" bisa berupa kata keterangan (kembali), kata sifat (pintu belakang), kata benda (bagian belakang ruangan), atau kata kerja (cadangkan berkas Anda). "Jack" bisa berupa nama, kata benda (kartu remi, bidak permainan logam berujung enam, bendera bahari, ikan, soket, atau alat untuk mengangkat benda berat), atau kata kerja (mendongkrak mobil, berburu dengan senter, atau memukul bola bisbol dengan keras). Ambiguitas sintaksis mengacu pada frasa yang memiliki beberapa penguraian: "I smelled a wumpus in 2,2" memiliki dua penguraian: satu di mana frasa preposisional "in 2,2" menerangkan kata benda dan satu di mana frasa tersebut menerangkan kata kerja. Ambiguitas sintaksis ini menyebabkan ambiguitas semantik, karena satu penguraian berarti wumpus berada di 2,2 dan yang lainnya berarti bau busuk berada di 2,2. Dalam hal ini, salah tafsir bisa menjadi kesalahan fatal bagi pelaku.

Akhirnya, bisa terjadi ambiguitas antara makna harfiah dan kiasan. Majas penting dalam puisi, tetapi secara mengejutkan juga umum dalam percakapan sehari-hari. Metonimi adalah majas di mana satu objek digunakan untuk mewakili objek lain. Ketika kita mendengar "Chrysler mengumumkan model baru," kita tidak menafsirkannya sebagai pernyataan bahwa perusahaan dapat berbicara; melainkan, kita memahami bahwa seorang juru bicara yang mewakili perusahaan tersebut membuat pengumuman tersebut. Metonimi umum dan sering ditafsirkan secara tidak sadar oleh pendengar manusia.

Sayangnya, tata bahasa kita sebagaimana tertulis tidaklah semudah itu. Untuk menangani semantik metonimi dengan tepat, kita perlu memperkenalkan tingkat ambiguitas yang sama sekali baru. Kita melakukan ini dengan menyediakan dua objek untuk interpretasi semantik setiap frasa dalam kalimat: satu untuk objek yang secara harfiah dirujuk oleh frasa tersebut (Chrysler) dan satu untuk referensi metonimik (juru bicara). Kita kemudian harus mengatakan bahwa ada hubungan antara keduanya. Dalam tata bahasa kita saat ini, "Chrysler mengumumkan" ditafsirkan sebagai

$$x = \text{Chrysler} \wedge e \in \text{Announce}(x) \wedge \text{After}(\text{Now}, \text{Extent}(e)) .$$

Kita perlu mengubahnya menjadi

$$x = \text{Chrysler} \wedge e \in \text{Announce}(m) \wedge \text{After}(\text{Now}, \text{Extent}(e)) \\ \wedge \text{Metonymy}(m, x).$$

Ini menyatakan bahwa terdapat satu entitas x yang setara dengan Chrysler, dan entitas lain m yang melakukan pengumuman, dan keduanya berada dalam relasi metonimi. Langkah selanjutnya adalah mendefinisikan jenis relasi metonimi apa saja yang dapat terjadi. Kasus paling sederhana adalah ketika tidak ada metonimi sama sekali—objek literal x dan objek metonimi m identik:

$$\forall m, x (m = x) \Rightarrow \text{Metonymy}(m, x).$$

Untuk contoh Chrysler, generalisasi yang masuk akal adalah bahwa suatu organisasi dapat digunakan untuk mewakili juru bicara organisasi tersebut:

$$\forall m, x (x \in \text{Organizations} \wedge \text{Spokesperson}(m, x) \Rightarrow \text{Metonymy}(m, x)).$$

Metonimi lainnya mencakup pengarang untuk karya-karyanya (Saya membaca *Shakespeare*) atau lebih umum, produser untuk produknya (Saya mengendarai *Honda*) dan bagian untuk keseluruhannya (The Red Sox butuh lengan yang kuat). Beberapa contoh metonimi, seperti "Roti lapis ham di Meja 4 ingin bir lagi," lebih baru dan ditafsirkan berdasarkan suatu situasi. Metafora adalah majas lain, di mana frasa dengan satu makna literal digunakan untuk menunjukkan makna yang berbeda melalui analogi. Dengan demikian, metafora dapat dipandang sebagai sejenis metonimi yang hubungannya berupa kesamaan.

Disambiguasi adalah proses menemukan kembali makna yang paling mungkin dari suatu ujaran. Dalam satu pengertian, kita sudah memiliki kerangka kerja untuk memecahkan masalah ini: setiap aturan memiliki probabilitas yang terkait dengannya, sehingga probabilitas suatu interpretasi adalah hasil kali dari probabilitas aturan-aturan yang mengarah pada interpretasi tersebut. Sayangnya, probabilitas tersebut mencerminkan seberapa umum frasa-frasa tersebut dalam korpus tempat tata bahasa dipelajari, dan dengan demikian mencerminkan pengetahuan umum, bukan pengetahuan khusus tentang situasi saat ini. Untuk melakukan disambiguasi dengan tepat, kita perlu menggabungkan empat model:

1. Model dunia: kemungkinan suatu proposisi muncul di dunia. Mengingat apa yang kita ketahui tentang dunia, kemungkinan besar seorang penutur yang mengatakan "Saya mati" berarti "Saya dalam masalah besar" daripada "Hidup saya berakhir, tetapi saya masih bisa bicara."
2. Model mental: kemungkinan penutur memiliki niat untuk mengomunikasikan fakta tertentu kepada pendengar. Pendekatan ini menggabungkan model-model tentang apa yang diyakini penutur, apa yang diyakini penutur akan diyakini pendengar, dan seterusnya. Misalnya, ketika seorang politisi mengatakan, "Saya bukan penjahat," model dunia mungkin menetapkan probabilitas hanya 50% untuk proposisi bahwa politisi tersebut bukan penjahat, dan 99,999% untuk proposisi bahwa ia bukan

tongkat gembala berkait. Namun demikian, kami memilih interpretasi pertama karena lebih mungkin untuk dikatakan.

3. Model bahasa: kemungkinan terpilihnya serangkaian kata tertentu, mengingat pembicara memiliki niat untuk mengomunikasikan fakta tertentu.
4. Model akustik: untuk komunikasi lisan, kemungkinan dihasilkannya serangkaian bunyi tertentu, mengingat pembicara telah memilih serangkaian kata tertentu.

5.4 TERJEMAHAN MESIN

Penerjemahan mesin adalah penerjemahan teks secara otomatis dari satu bahasa alami (sumber) ke bahasa lain (target). Ini adalah salah satu area aplikasi pertama yang dibayangkan untuk komputer, tetapi baru dalam dekade terakhir teknologi ini digunakan secara luas. Berikut kutipan dari halaman 1 buku ini:

AI adalah salah satu bidang terbaru dalam sains dan teknik. Pekerjaan dimulai dengan sungguh-sungguh segera setelah Perang Dunia II, dan namanya sendiri diciptakan pada tahun 1956. Bersama dengan biologi molekuler, AI sering disebut sebagai "bidang yang paling ingin saya tekuni" oleh para ilmuwan di disiplin ilmu lain.

Dan berikut terjemahannya dari bahasa Inggris ke bahasa Denmark oleh alat daring, Google Translate:

AI adalah salah satu bidang terbaru dalam sains dan teknik. Pekerjaan dimulai tepat setelah Perang Dunia II, dan namanya sendiri ditemukan pada tahun 1956. Bersama dengan biologi molekuler, AI tampaknya dianggap sebagai "saya merasa paling tepat berada di" dalam penelitian dan disiplin ilmu lainnya.

Bagi mereka yang tidak bisa membaca bahasa Denmark, berikut terjemahan bahasa Denmark kembali ke bahasa Inggris. Kata-kata yang keluar berbeda dicetak miring:

AI adalah salah satu bidang sains dan teknik terbaru. Pekerjaan dimulai dengan sungguh-sungguh tepat setelah Perang Dunia Kedua, dan namanya sendiri diciptakan pada tahun 1956. Bersama dengan biologi molekuler, AI sering disebut sebagai — "bidang yang paling ingin saya geluti" oleh para peneliti di disiplin ilmu lain.

Semua perbedaan tersebut merupakan parafrase yang wajar, seperti yang sering disebutkan untuk dikutip secara teratur. Satu-satunya kesalahan nyata adalah penghilangan artikel *the*, yang dilambangkan dengan simbol $\varnothing$. Ini adalah akurasi yang umum: dari dua kalimat, satu kalimat memiliki kesalahan yang tidak akan dilakukan oleh penutur asli, namun maknanya tersampaikan dengan jelas.

Secara historis, terdapat tiga aplikasi utama penerjemahan mesin. Penerjemahan kasar, seperti yang disediakan oleh layanan daring gratis, memberikan "inti" dari kalimat atau dokumen asing, tetapi mengandung kesalahan. Penerjemahan pra-sunting digunakan oleh perusahaan untuk menerbitkan dokumentasi dan materi penjualan mereka dalam berbagai bahasa. Teks sumber asli ditulis dalam bahasa terbatas yang lebih mudah diterjemahkan secara otomatis, dan hasilnya biasanya disunting oleh manusia untuk memperbaiki kesalahan. Penerjemahan sumber terbatas bekerja sepenuhnya otomatis, tetapi hanya pada bahasa yang sangat stereotip, seperti laporan cuaca.

Penerjemahan sulit karena, dalam kasus yang sepenuhnya umum, diperlukan pemahaman teks yang mendalam. Hal ini berlaku bahkan untuk teks yang sangat sederhana—bahkan "texts" yang terdiri dari satu kata. Perhatikan kata "Open" di pintu sebuah toko. Kata tersebut mengomunikasikan gagasan bahwa toko tersebut sedang menerima pelanggan saat ini.

Sekarang perhatikan kata yang sama, "Open", pada spanduk besar di luar sebuah toko yang baru dibangun. Artinya, toko tersebut kini beroperasi setiap hari, tetapi pembaca tanda ini tidak akan merasa tertipu jika toko tersebut tutup pada malam hari tanpa melepas spanduknya. Kedua tanda tersebut menggunakan kata yang sama untuk menyampaikan makna yang berbeda. Dalam bahasa Jerman, tanda di pintu bertuliskan "Offen", sementara spanduknya bertuliskan "Neu Eröffnet".

Masalahnya adalah bahasa yang berbeda mengkategorikan dunia secara berbeda. Misalnya, kata "doux" dalam bahasa Prancis mencakup beragam makna yang kira-kira setara dengan kata "soft" dalam bahasa Inggris, "sweet" dalam bahasa Inggris, dan "gentle" dalam bahasa Inggris. Demikian pula, kata "hard" dalam bahasa Inggris mencakup hampir semua penggunaan kata "hart" dalam bahasa Jerman (keras kepala, kejam) dan beberapa penggunaan kata "schwierig" (sulit). Oleh karena itu, merepresentasikan makna sebuah kalimat lebih sulit untuk diterjemahkan daripada untuk dipahami dalam satu bahasa. Sistem penguraian bahasa Inggris dapat menggunakan predikat seperti $Open(x)$, tetapi untuk menerjemahkan, bahasa representasi harus membuat lebih banyak perbedaan, mungkin dengan $Open_1(x)$ mewakili makna "Offen" dan $Open_2(x)$ mewakili makna "Neu Eröffnet". Bahasa representasi yang membuat semua perbedaan yang diperlukan untuk sekumpulan bahasa disebut interlingua.

Seorang penerjemah (manusia atau mesin) sering kali perlu memahami situasi aktual yang dijelaskan dalam sumber, bukan hanya kata-kata individualnya. Misalnya, untuk menerjemahkan kata bahasa Inggris "him" ke dalam bahasa Korea, pilihan harus dibuat antara bentuk sederhana dan bentuk hormat, pilihan yang bergantung pada hubungan sosial antara pembicara dan referen "him".

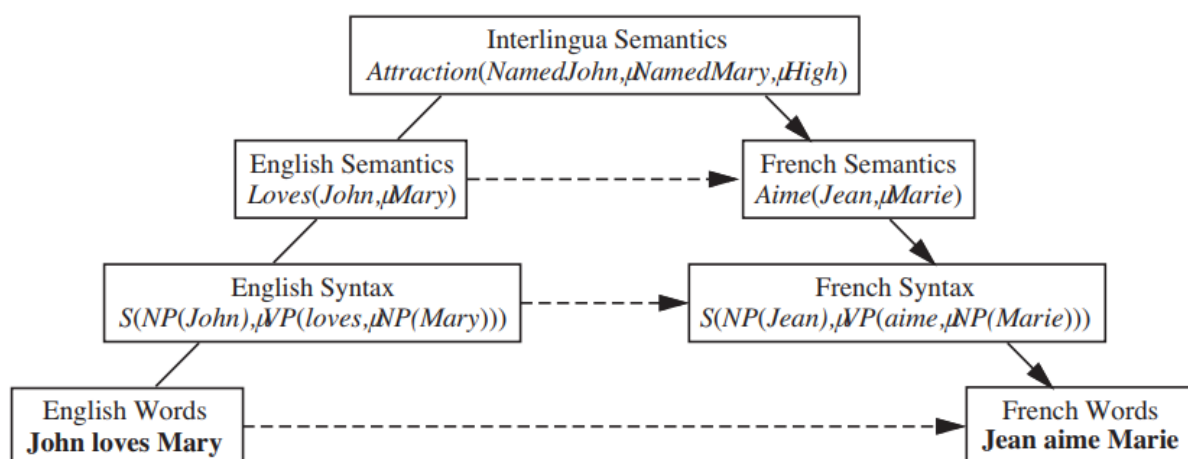
Dalam bahasa Jepang, bentuk hormat bersifat relatif, sehingga pilihannya bergantung pada hubungan sosial antara pembicara, referen, dan pendengar. Penerjemah (baik mesin maupun manusia) terkadang merasa sulit untuk membuat pilihan ini. Sebagai contoh lain, untuk menerjemahkan "Bola bisbol itu mengenai jendela. Bola itu pecah." ke dalam bahasa Prancis, kita harus memilih bentuk feminin "elle" atau bentuk maskulin "il" untuk "it", jadi kita harus memutuskan apakah "it" merujuk pada bola bisbol atau jendela. Untuk mendapatkan terjemahan yang tepat, seseorang harus memahami fisika sekaligus bahasa.

Terkadang tidak ada pilihan yang dapat menghasilkan terjemahan yang sepenuhnya memuaskan. Misalnya, sebuah puisi cinta Italia yang menggunakan bentuk maskulin "il sole" (matahari) dan bentuk feminin "la luna" (bulan) untuk melambangkan dua kekasih pasti akan berubah ketika diterjemahkan ke dalam bahasa Jerman, di mana gendernya dibalik, dan akan berubah lebih lanjut ketika diterjemahkan ke dalam bahasa yang gendernya sama.

Sistem Penerjemahan Mesin

Semua sistem penerjemahan harus memodelkan bahasa sumber dan bahasa target, tetapi sistem bervariasi dalam jenis model yang digunakan. Beberapa sistem mencoba menganalisis teks bahasa sumber hingga ke representasi pengetahuan interlingua dan kemudian menghasilkan kalimat dalam bahasa target dari representasi tersebut. Hal ini sulit karena melibatkan tiga masalah yang belum terpecahkan: menciptakan representasi pengetahuan yang lengkap dari segala hal; mengurai ke dalam representasi tersebut; dan menghasilkan kalimat dari representasi tersebut.

Sistem lain didasarkan pada model transfer. Sistem ini menyimpan basis data aturan penerjemahan (atau contoh), dan kapan pun aturan (atau contoh) tersebut cocok, sistem tersebut langsung menerjemahkan. Transfer dapat terjadi pada tingkat leksikal, sintaksis, atau semantik. Misalnya, aturan sintaksis yang ketat memetakan bahasa Inggris [*Adjective Noun*] ke bahasa Prancis [*Noun Adjective*]. Aturan sintaksis dan leksikal campuran memetakan bahasa Prancis [S_1 "et puis" S_2] ke bahasa Inggris [S_1 "and then" S_2]. Gambar 5.12 menunjukkan diagram berbagai titik transfer.



Gambar 5.12 Segitiga Vauquois: diagram skematis pilihan untuk sistem terjemahan mesin. Kita mulai dengan teks bahasa Inggris di bagian atas. Sistem berbasis interlingua mengikuti garis padat, mengurai bahasa Inggris terlebih dahulu menjadi bentuk sintaksis, kemudian menjadi representasi semantik dan representasi interlingua, dan kemudian melalui pembangkitan menjadi bentuk semantik, sintaksis, dan leksikal dalam bahasa Prancis. Sistem berbasis transfer menggunakan garis putus-putus sebagai jalan pintas. Sistem yang berbeda melakukan transfer pada titik yang berbeda; beberapa melakukannya di beberapa titik.

Penerjemahan Mesin Statistik

Setelah kita melihat betapa rumitnya tugas penerjemahan, tidak mengherankan jika sistem penerjemahan mesin yang paling sukses dibangun dengan melatih model probabilistik menggunakan statistik yang dikumpulkan dari korpus teks yang besar. Pendekatan ini tidak memerlukan ontologi konsep interlingua yang kompleks, juga tidak memerlukan tata bahasa yang dibuat sendiri dari bahasa sumber dan bahasa target, atau treebank yang diberi label secara manual. Yang dibutuhkan hanyalah data—contoh terjemahan yang darinya model

penerjemahan dapat dipelajari. Untuk menerjemahkan kalimat dalam, katakanlah, bahasa Inggris (e) ke bahasa Prancis (f), kita menemukan rangkaian kata f^* yang memaksimalkan

$$f^* = \underset{f}{\operatorname{argmax}} P(f|e) = \operatorname{argmax} P(e|f)P(f).$$

Di sini faktor $P(f)$ adalah model bahasa target untuk bahasa Prancis; model ini menyatakan seberapa besar kemungkinan kalimat tertentu dalam bahasa Prancis. $P(e|f)$ adalah model penerjemahan; model ini menyatakan seberapa besar kemungkinan kalimat bahasa Inggris sebagai penerjemahan untuk kalimat bahasa Prancis tertentu. Demikian pula, $P(f|e)$ adalah model penerjemahan dari bahasa Inggris ke bahasa Prancis.

Haruskah kita bekerja langsung pada $P(f|e)$, atau menerapkan aturan Bayes dan mengerjakan $P(e|f)P(f)$? Dalam aplikasi diagnostik seperti kedokteran, lebih mudah untuk memodelkan domain dalam arah kausal: $P(\text{symptoms}|\text{disease})$ daripada $P(\text{disease}|\text{symptoms})$. Namun dalam penerjemahan, kedua arah sama mudahnya.

Pekerjaan paling awal dalam penerjemahan mesin statistik menerapkan aturan Bayes—sebagian karena para peneliti memiliki model bahasa yang baik, $P(f)$, dan ingin memanfaatkannya, dan sebagian karena mereka berasal dari latar belakang pengenalan ucapan, yang merupakan masalah diagnostik. Kami mengikuti jejak mereka dalam bab ini, tetapi kami mencatat bahwa karya terkini dalam penerjemahan mesin statistik sering mengoptimalkan $P(f|e)$ secara langsung, menggunakan model yang lebih canggih yang memperhitungkan banyak fitur dari model bahasa.

Model bahasa, $P(f)$, dapat menangani level apa pun di sisi kanan Gambar 5.12, tetapi pendekatan termudah dan paling umum adalah membangun model n -gram dari korpus bahasa Prancis, seperti yang telah kita lihat sebelumnya. Ini hanya menangkap sebagian, ide lokal dari kalimat-kalimat bahasa Prancis; namun, itu seringkali cukup untuk penerjemahan kasar.

Model penerjemahan dipelajari dari korpus dwibahasa—kumpulan teks paralel, masing-masing pasangan bahasa Inggris/Prancis. Sekarang, jika kita memiliki korpus yang sangat besar, maka menerjemahkan kalimat hanya akan menjadi tugas pencarian: kita akan melihat kalimat bahasa Inggris sebelumnya di korpus, jadi kita bisa mengembalikan kalimat bahasa Prancis yang dipasangkan. Tetapi tentu saja sumber daya kita terbatas, dan sebagian besar kalimat yang akan diminta untuk kita terjemahkan akan menjadi baru. Namun, kalimat-kalimat itu akan terdiri dari frasa-frasa yang telah kita lihat sebelumnya (bahkan jika beberapa frasa sesingkat satu kata). Misalnya, dalam buku ini, frasa umum meliputi "dalam latihan ini kita akan", "ukuran ruang keadaan", "sebagai fungsi dari", dan "catatan di akhir bab". Jika diminta menerjemahkan kalimat baru "Dalam latihan ini kita akan menghitung ukuran ruang keadaan sebagai fungsi dari jumlah tindakan." ke dalam bahasa Prancis, kita seharusnya dapat memecah kalimat menjadi frasa-frasa, menemukan frasa-frasa tersebut dalam korpus bahasa Inggris (buku ini), menemukan frasa-frasa Prancis yang sesuai (dari terjemahan bahasa Prancis buku tersebut), lalu menyusun kembali frasa-frasa Prancis tersebut ke dalam urutan yang

masuk akal dalam bahasa Prancis. Dengan kata lain, jika diberikan kalimat bahasa Inggris sumber, e , menemukan terjemahan bahasa Prancis f terdiri dari tiga langkah:

1. Pecah kalimat bahasa Inggris menjadi frasa $e_1, \dots, e_n$.
2. Untuk setiap frasa e_i , pilih frasa bahasa Prancis yang sesuai f_i . Kita menggunakan notasi $P(f_i | e_i)$ untuk probabilitas frasa bahwa f_i merupakan translasi dari e_i .
3. Pilih permutasi frasa $f_1, \dots, f_n$. Kita akan menentukan permutasi ini dengan cara yang tampaknya agak rumit, tetapi dirancang untuk memiliki distribusi probabilitas yang sederhana: Untuk setiap f_i , kita memilih distorsi d_i , yang merupakan jumlah kata yang telah dipindahkan oleh frasa f_i terhadap f_{i-1} ; positif untuk bergerak ke kanan, negatif untuk bergerak ke kiri, dan nol jika f_i langsung mengikuti f_{i-1} .

Gambar 5.13 menunjukkan contoh prosesnya. Di bagian atas, kalimat "There is a smelly wumpus sleeping in 2 2" dipecah menjadi lima frasa, $e_1, \dots, e_n$. Masing-masing frasa tersebut diterjemahkan menjadi frasa f_i yang bersesuaian, dan kemudian frasa-frasa ini dipermutasi ke dalam urutan f_1, f_3, f_4, f_2, f_5 .

Kita menentukan permutasi dalam hal distorsi di dari setiap frasa bahasa Prancis, yang didefinisikan sebagai

$$d_i = \text{START}(f_i) - \text{END}(f_{i-1}) - 1,$$

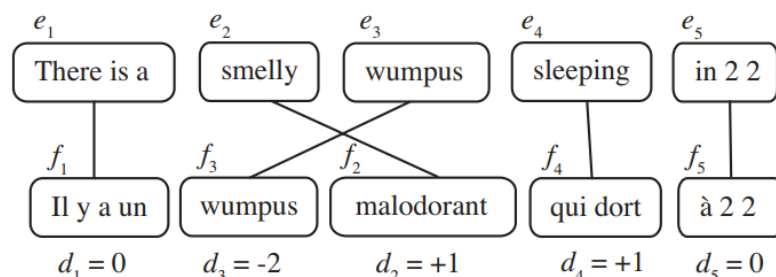
di mana $\text{START}(f_i)$ adalah nomor urut kata pertama frasa f_i dalam kalimat bahasa Prancis, dan $\text{END}(f_{i-1})$ adalah nomor urut kata terakhir frasa f_{i-1} . Pada Gambar 5.13, kita melihat bahwa f_5 , "a` 2 2," langsung mengikuti f_4 , "qui dort," sehingga $d_5 = 0$. Namun, frasa f_2 telah bergeser satu kata ke kanan dari f_1 , sehingga $d_2 = 1$. Sebagai kasus khusus, kita memiliki $d_1 = 0$, karena f_1 dimulai pada posisi 1 dan $\text{END}(f_0)$ didefinisikan sebagai 0 (meskipun f_0 tidak ada).

Setelah kita mendefinisikan distorsi, d_i , kita dapat mendefinisikan distribusi probabilitas untuk distorsi, $P(d_i)$. Perhatikan bahwa untuk kalimat yang dibatasi oleh panjang n , kita memiliki $|d_i| \leq n$, sehingga distribusi probabilitas penuh $P(d_i)$ hanya memiliki $2n + 1$ elemen, jauh lebih sedikit angka yang harus dipelajari daripada jumlah permutasi, $n!$. Itulah sebabnya kami mendefinisikan permutasi dengan cara yang berliku-liku ini. Tentu saja, ini adalah model distorsi yang agak miskin.

Model ini tidak menyatakan bahwa kata sifat biasanya terdistorsi sehingga muncul setelah kata benda ketika kita menerjemahkan dari bahasa Inggris ke bahasa Prancis—fakta itu direpresentasikan dalam model bahasa Prancis, $P(f)$. Probabilitas distorsi sepenuhnya independen dari kata-kata dalam frasa—ia hanya bergantung pada nilai integer d_i . Distribusi probabilitas memberikan ringkasan volatilitas permutasi; seberapa besar kemungkinan distorsi $P(d = 2)$, dibandingkan dengan $P(d = 0)$, misalnya.

Kita sekarang siap untuk menyatukan semuanya: kita dapat mendefinisikan $P(f, d|e)$, probabilitas bahwa deret frasa f dengan distorsi d merupakan translasi dari deret frasa e . Kita berasumsi bahwa setiap translasi frasa dan setiap distorsi independen satu sama lain, sehingga kita dapat memfaktorkan ekspresi tersebut sebagai

$$P(f, d|e) = \prod_i P(f_i | e_i) P(d_i)$$



Gambar 5.13 Kandidat frasa bahasa Prancis untuk setiap frasa kalimat bahasa Inggris, dengan nilai distorsi (d) untuk setiap frasa bahasa Prancis.

Hal ini memberi kita cara untuk menghitung probabilitas $P(f, d|e)$ untuk kandidat translasi f dan distorsi d . Namun, untuk menemukan f dan d terbaik, kita tidak bisa hanya menghitung kalimat; dengan mungkin 100 frasa bahasa Prancis untuk setiap frasa bahasa Inggris dalam korpus, terdapat 100^5 translasi 5-frasa yang berbeda, dan $5!$ penataan ulang untuk masing-masingnya. Kita harus mencari solusi yang baik. Pencarian berkas lokal dengan heuristik yang memperkirakan probabilitas telah terbukti efektif dalam menemukan translasi yang hampir paling mungkin.

Yang tersisa hanyalah mempelajari probabilitas frasa dan distorsi. Kita akan menguraikan prosedurnya; lihat catatan di akhir bab untuk detailnya.

1. Temukan teks paralel: Pertama, kumpulkan korpus dwibahasa paralel. Misalnya, Hansard adalah rekaman debat parlemen. Kanada, Hong Kong, dan negara-negara lain menghasilkan Hansard dwibahasa, Uni Eropa menerbitkan dokumen resminya dalam 11 bahasa, dan Perserikatan Bangsa-Bangsa menerbitkan dokumen multibahasa. Teks dwibahasa juga tersedia daring; beberapa situs web menerbitkan konten paralel dengan URL paralel, misalnya, /en/ untuk halaman bahasa Inggris dan /fr/ untuk halaman bahasa Prancis terkait. Sistem penerjemahan statistik terkemuka melatih ratusan juta kata teks paralel dan miliaran kata teks monolingual.
2. Segmentasikan ke dalam kalimat: Unit penerjemahan adalah kalimat, jadi kita harus membagi korpus menjadi kalimat-kalimat. Titik merupakan indikator kuat akhir kalimat, tetapi perhatikan "Dr. J. R. Smith dari Rodeo Dr. membayar Rp 299.900 pada 9.9.09."; hanya titik terakhir yang mengakhiri kalimat. Salah satu cara untuk menentukan apakah titik mengakhiri kalimat adalah dengan melatih model yang mengambil fitur kata-kata di sekitarnya dan jenis katanya. Pendekatan ini mencapai akurasi sekitar 98%.
3. Sejajarkan kalimat: Untuk setiap kalimat dalam versi bahasa Inggris, tentukan kalimat mana yang sesuai dengan versi bahasa Prancis. Biasanya, kalimat bahasa Inggris berikutnya sesuai dengan kalimat bahasa Prancis berikutnya dalam kecocokan 1:1, tetapi terkadang ada variasi: satu kalimat dalam satu bahasa akan dipecah menjadi

kecocokan 2:1, atau urutan dua kalimat akan ditukar, menghasilkan kecocokan 2:2. Dengan melihat panjang kalimat saja (yaitu kalimat pendek harus sejajar dengan kalimat pendek), dimungkinkan untuk menyelaraskannya (1:1, 1:2, atau 2:2, dll.) dengan akurasi dalam kisaran 90% hingga 99% menggunakan variasi pada algoritma Viterbi. Penyejarasan yang lebih baik dapat dicapai dengan menggunakan landmark yang umum untuk kedua bahasa, seperti angka, tanggal, nama diri, atau kata-kata yang kita tahu dari kamus dwibahasa memiliki terjemahan yang tidak ambigu. Misalnya, jika kalimat ke-3 dalam bahasa Inggris dan ke-4 dalam bahasa Prancis mengandung string "1989" dan kalimat-kalimat di sebelahnya tidak, hal itu merupakan bukti kuat bahwa kalimat-kalimat tersebut seharusnya disejajarkan.

4. Menyejajarkan frasa: Di dalam sebuah kalimat, frasa dapat disejajarkan melalui proses yang serupa dengan yang digunakan untuk penyejajaran kalimat, tetapi memerlukan perbaikan berulang. Saat kita memulai, kita tidak dapat mengetahui bahwa "qui dort" sejajar dengan "sleeping", tetapi kita dapat mencapai penyejajaran tersebut melalui proses agregasi bukti. Dari semua contoh kalimat yang telah kita lihat, kita melihat bahwa "qui dort" dan "sleeping" muncul bersamaan dengan frekuensi tinggi, dan bahwa dalam pasangan kalimat yang disejajarkan, tidak ada frasa selain "qui dort" yang muncul bersamaan dengan "sleeping" sesering itu dalam kalimat lain. Penyejajaran frasa yang lengkap pada korpus kita memberikan kita probabilitas frasa (setelah penghalusan yang sesuai).
5. Mengekstrak distorsi: Setelah kita memiliki penyejajaran frasa, kita dapat menentukan probabilitas distorsi. Hitung saja seberapa sering distorsi terjadi dalam korpus untuk setiap jarak $d = 0, \pm 1, \pm 2, \dots$, dan terapkan pemulusan.
6. Perbaiki estimasi dengan EM: Gunakan ekspektasi-maksimisasi untuk memperbaiki estimasi nilai $P(f|e)$ dan $P(d)$. Kami menghitung keselarasan terbaik dengan nilai parameter saat ini pada langkah E, lalu memperbarui estimasi pada langkah M dan mengulangi proses hingga konvergensi.

5.5 PENGENALAN PIDATO

Pengenalan pidato adalah tugas mengidentifikasi rangkaian kata yang diucapkan oleh seorang pembicara, berdasarkan sinyal akustik. Hal ini telah menjadi salah satu aplikasi utama AI—jutaan orang berinteraksi dengan sistem pengenalan suara setiap hari untuk menavigasi sistem pesan suara, menelusuri web dari ponsel, dan aplikasi lainnya. Ucapan merupakan pilihan yang menarik ketika pengoperasian tanpa menggunakan tangan diperlukan, seperti saat mengoperasikan mesin.

Pengenalan suara sulit karena suara yang dihasilkan oleh seorang pembicara ambigu dan, ya, berisik. Sebagai contoh yang terkenal, frasa "recognize speech" terdengar hampir sama dengan "wreck a nice beach" ketika diucapkan dengan cepat. Bahkan contoh singkat ini menunjukkan beberapa masalah yang membuat ucapan menjadi problematis. Pertama, segmentasi: kata-kata tertulis dalam bahasa Inggris memiliki spasi di antaranya, tetapi dalam ucapan cepat tidak ada jeda dalam "wreck a nice" yang akan membedakannya sebagai frasa

multikata, bukan kata tunggal "recognize." Kedua, koartikulasi: ketika berbicara cepat, bunyi "s" di akhir "nice" menyatu dengan bunyi "b" di awal "beach", menghasilkan sesuatu yang mirip dengan "sp". Masalah lain yang tidak muncul dalam contoh ini adalah homofon—kata-kata seperti "to", "too", dan "two" yang bunyinya sama tetapi berbeda makna.

Kita dapat memandang pengenalan suara sebagai masalah dalam penjelasan urutan yang paling mungkin. Ini adalah masalah menghitung urutan variabel keadaan yang paling mungkin, $x_{1:t}$, dengan urutan observasi $e_{1:t}$. Dalam hal ini, variabel keadaan adalah kata-kata, dan observasi adalah bunyi. Lebih tepatnya, observasi adalah vektor fitur yang diekstraksi dari sinyal audio. Seperti biasa, urutan yang paling mungkin dapat dihitung dengan bantuan aturan Bayes sebagai berikut:

$$\underset{word_{1:t}}{\operatorname{argmax}} P(word_{1:t} | sound_{1:t}) = \underset{word_{1:t}}{\operatorname{argmax}} P(sound_{1:t} | word_{1:t}) P(word_{1:t}).$$

Di sini $P(sound_{1:t} | word_{1:t})$ adalah model akustik. Model ini menggambarkan bunyi kata—bahwa "ceiling" dimulai dengan "c" lembut dan berbunyi sama dengan "sealing". $P(word_{1:t})$ dikenal sebagai model bahasa. Model ini menentukan probabilitas sebelumnya dari setiap ujaran—misalnya, "ceiling fan" sekitar 500 kali lebih mungkin sebagai rangkaian kata daripada "sealing fan".

Vowels		Consonants B–N		Consonants P–Z	
Phone	Example	Phone	Example	Phone	Example
[iy]	beat	[b]	bet	[p]	pet
[ih]	b <u>i</u> t	[ch]	<u>Ch</u> et	[r]	<u>r</u> at
[eh]	be <u>t</u>	[d]	<u>d</u> ebt	[s]	<u>s</u> et
[æ]	ba <u>t</u>	[f]	<u>f</u> at	[sh]	<u>sh</u> oe
[ah]	bu <u>t</u>	[g]	<u>g</u> et	[t]	<u>t</u> en
[ao]	bo <u>u</u> ght	[hh]	<u>h</u> at	[th]	<u>th</u> ick
[ow]	bo <u>o</u> t	[hv]	<u>h</u> igh	[dh]	<u>th</u> at
[uh]	bo <u>o</u> k	[jh]	<u>j</u> et	[dx]	bu <u>tt</u> er
[ey]	ba <u>i</u> t	[k]	<u>k</u> ick	[v]	<u>v</u> et
[er]	Be <u>r</u> t	[l]	<u>l</u> et	[w]	<u>w</u> et
[ay]	bu <u>y</u>	[el]	bo <u>tt</u> le	[wh]	<u>wh</u> ich
[oy]	bo <u>y</u>	[m]	<u>m</u> et	[y]	<u>y</u> et
[axr]	di <u>n</u> er	[em]	bo <u>tt</u> om	[z]	<u>z</u> oo
[aw]	do <u>w</u> n	[n]	<u>n</u> et	[zh]	mea <u>s</u> ure
[ax]	a <u>b</u> out	[en]	bu <u>tt</u> on		
[ix]	ro <u>s</u> es	[ng]	si <u>n</u> g		
[aa]	co <u>t</u>	[eng]	wa <u>sh</u> ing	[-]	<i>silence</i>

Gambar 5.14 Alfabet fonetik ARPA, atau ARPabet, mencantumkan semua fonetik yang digunakan dalam Bahasa Inggris Amerika. Terdapat beberapa notasi alternatif, termasuk Alfabet Fonetik Internasional (IPA), yang memuat fonetik dalam semua bahasa yang dikenal.

Pendekatan ini dinamai model kanal bising oleh Claude Shannon. Ia menggambarkan situasi di mana pesan asli (kata-kata dalam contoh kita) ditransmisikan melalui kanal bising (seperti

saluran telepon) sehingga pesan yang rusak (bunyi dalam contoh kita) diterima di ujung yang lain. Shannon menunjukkan bahwa betapa pun bisingnya kanal tersebut, pesan asli dapat dipulihkan dengan kesalahan yang sangat kecil, jika kita mengodekan pesan asli dengan cara yang cukup redundan. Pendekatan kanal bising telah diterapkan pada pengenalan suara, penerjemahan mesin, koreksi ejaan, dan tugas-tugas lainnya.

Setelah kita mendefinisikan model akustik dan bahasa, kita dapat memecahkan urutan kata yang paling mungkin menggunakan algoritma Viterbi. Sebagian besar sistem pengenalan ucapan menggunakan model bahasa yang menggunakan asumsi Markov—bahwa keadaan saat ini $Word_t$ hanya bergantung pada sejumlah tetap n keadaan sebelumnya—dan merepresentasikan $Word_t$ sebagai variabel acak tunggal yang memiliki sekumpulan nilai terbatas, yang menjadikannya Model Markov Tersembunyi (HMM). Dengan demikian, pengenalan ucapan menjadi aplikasi sederhana dari metodologi HMM, sederhana, setelah kita mendefinisikan model akustik dan bahasa. Kita akan membahasnya selanjutnya.

Model Akustik

Gelombang suara adalah perubahan tekanan periodik yang merambat melalui udara. Ketika gelombang ini mengenai diafragma mikrofon, gerakan maju mundur menghasilkan arus listrik. Konverter analog-ke-digital mengukur besarnya arus—yang mendekati amplitudo gelombang suara—pada interval diskrit yang disebut laju pengambilan sampel.

Suara ucapan, yang sebagian besar berada dalam rentang 100 Hz (100 siklus per detik) hingga 1000 Hz, biasanya diambil sampelnya pada laju 8 kHz. (CD dan berkas mp3 diambil sampelnya pada 44,1 kHz.) Presisi setiap pengukuran ditentukan oleh faktor kuantisasi; pengenalan ucapan biasanya menyimpan 8 hingga 12 bit. Artinya, sistem kelas bawah, yang mengambil sampel pada 8 kHz dengan kuantisasi 8-bit, akan membutuhkan hampir setengah megabita per menit ucapan. Karena kita hanya ingin tahu kata-kata yang diucapkan, bukan persis seperti apa bunyinya, kita tidak perlu menyimpan semua informasi itu.

Kita hanya perlu membedakan berbagai bunyi ujaran. Para ahli bahasa telah mengidentifikasi sekitar 100 bunyi ujaran, atau fonem, yang dapat dikomposisi untuk membentuk semua kata dalam semua bahasa manusia yang dikenal. Secara kasar, fonem adalah bunyi yang sesuai dengan satu vokal atau konsonan, tetapi ada beberapa komplikasi: kombinasi huruf, seperti "th" dan "ng" menghasilkan fonem tunggal, dan beberapa huruf menghasilkan fonem yang berbeda dalam konteks yang berbeda (misalnya, "a" dalam rat dan rate).

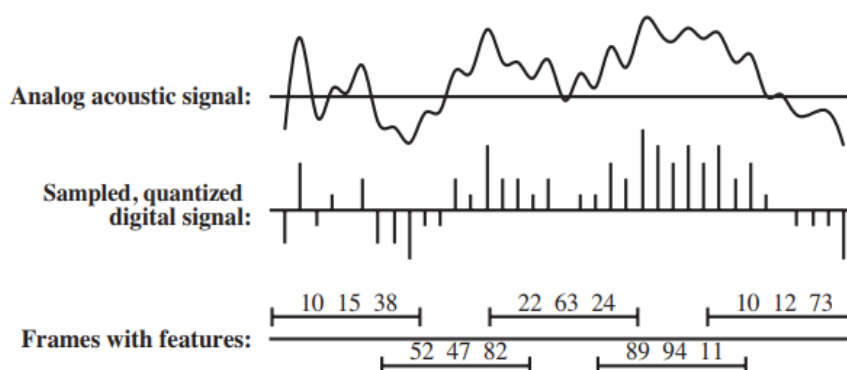
Gambar 5.14 mencantumkan semua fonem yang digunakan dalam bahasa Inggris, dengan contoh masing-masing. Fonem adalah unit bunyi terkecil yang memiliki arti berbeda bagi penutur bahasa tertentu. Misalnya, "t" dalam "stick" terdengar cukup mirip dengan "t" dalam "tick" sehingga penutur bahasa Inggris menganggapnya sebagai fonem yang sama. Tetapi perbedaannya signifikan dalam bahasa Thailand, jadi ada dua fonem. Untuk merepresentasikan bahasa Inggris lisan, kita menginginkan representasi yang dapat membedakan antara fonem yang berbeda, tetapi yang tidak perlu membedakan variasi nonfonemis dalam bunyi: keras atau lembut, cepat atau lambat, suara pria atau wanita, dll.

Pertama, kita mengamati bahwa meskipun frekuensi bunyi dalam ucapan mungkin beberapa kHz, perubahan dalam konten Sinyal yang tumpang tindih terjadi jauh lebih jarang, mungkin tidak lebih dari 100 Hz. Oleh karena itu, sistem ucapan merangkum sifat-sifat sinyal dalam irisan waktu yang disebut frame. Panjang frame sekitar 10 milidetik (yaitu, 80 sampel pada 8 kHz) cukup pendek untuk memastikan hanya sedikit fenomena berdurasi pendek yang terlewat. Frame yang tumpang tindih digunakan untuk memastikan kita tidak melewatkan sinyal hanya karena kebetulan berada di batas frame.

Setiap frame diringkas oleh vektor fitur. Memilih fitur dari sinyal suara seperti mendengarkan orkestra dan berkata, "Di sini, terompet Prancis dimainkan dengan keras dan biola dimainkan dengan lembut." Kami akan memberikan gambaran singkat tentang fitur-fitur dalam sistem yang umum. Pertama, transformasi Fourier digunakan untuk menentukan jumlah energi akustik pada sekitar selusin frekuensi. Kemudian, kami menghitung ukuran yang disebut koefisien cepstral frekuensi mel (MFCC) atau MFCC untuk setiap frekuensi.

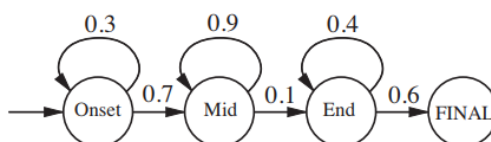
Kami juga menghitung energi total dalam frame. Hasilnya adalah tiga belas fitur; untuk setiap fitur, kami menghitung selisih antara frame ini dan frame sebelumnya, dan selisih antar perbedaan, dengan total 39 fitur. Fitur-fitur ini bernilai kontinu; cara termudah untuk memasukkannya ke dalam kerangka kerja HMM adalah dengan mendiskritisasi nilainya. (Model HMM juga dapat diperluas untuk menangani campuran Gaussian yang kontinu.) Gambar 5.15 menunjukkan urutan transformasi dari suara mentah ke urutan frame dengan fitur diskrit. Kita telah melihat cara berpindah dari sinyal akustik mentah ke serangkaian observasi, dst. Sekarang kita harus mendeskripsikan status HMM dan mendefinisikan model transisi, $P(X_t | X_{t-1})$, dan model sensor, $P(E_t | X_t)$.

Model transisi dapat dibagi menjadi dua level: kata dan telepon. Kita akan mulai dari bawah: model telepon mendeskripsikan telepon sebagai tiga status, awal, tengah, dan akhir. Misalnya, telepon [t] memiliki awal yang senyap, ledakan suara kecil di tengah, dan (biasanya) desisan di akhir. Gambar 5.16 menunjukkan contoh untuk telepon [m]. Perhatikan bahwa dalam ucapan normal, telepon rata-rata memiliki durasi 50–100 milidetik, atau 5–10 bingkai. Self-loop di setiap status memungkinkan variasi durasi ini. Dengan mengambil banyak self-loop (terutama di status tengah), kita dapat merepresentasikan suara "mmmmmmmmmmmm" yang panjang. Melewati self-loop menghasilkan bunyi "m" yang pendek.



Gambar 5.15 Menerjemahkan sinyal akustik ke dalam serangkaian frame. Dalam diagram ini, setiap frame dideskripsikan oleh nilai diskretisasi dari tiga fitur akustik; sistem nyata akan memiliki lusinan fitur.

Phone HMM for [m]:



Output probabilities for the phone HMM:

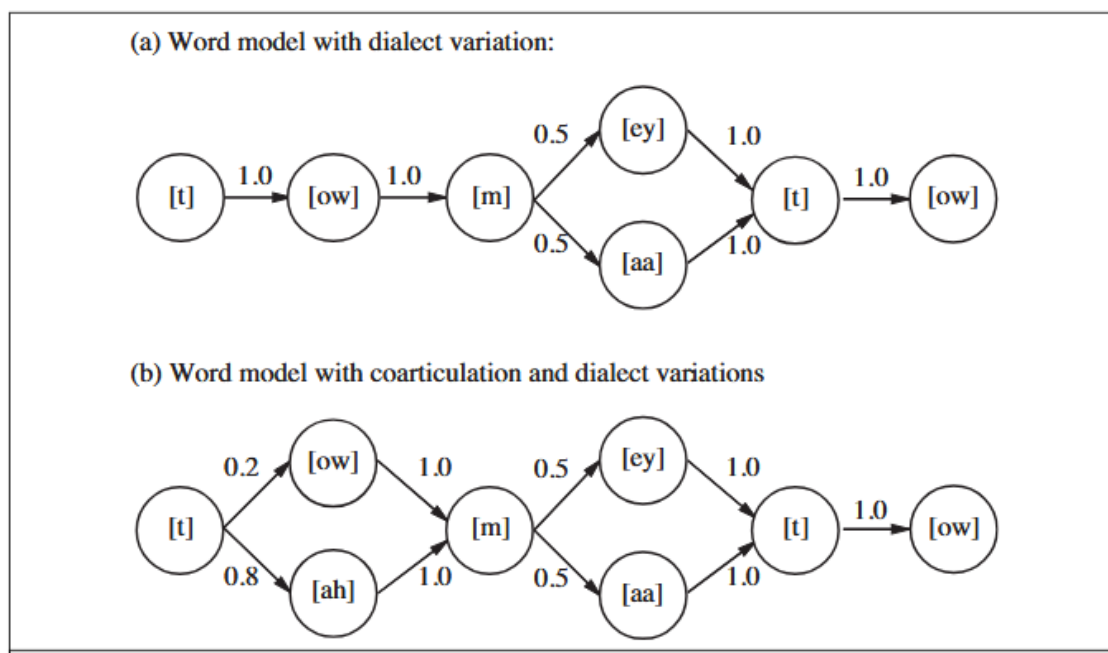
Onset:	Mid:	End:
C_1 : 0.5	C_3 : 0.2	C_4 : 0.1
C_2 : 0.2	C_4 : 0.7	C_6 : 0.5
C_3 : 0.3	C_5 : 0.1	C_7 : 0.4

Gambar 5.16 HMM untuk telepon tiga status [m]. Setiap status memiliki beberapa kemungkinan keluaran, masing-masing dengan probabilitasnya sendiri. Label fitur MFCC C_1 hingga C_7 bersifat arbitrer, mewakili beberapa kombinasi nilai fitur.

Pada Gambar 5.17, model-model fon dirangkai bersama untuk membentuk model pelafalan sebuah kata. Menurut Gershwin, Anda mengucapkan [t ow m ey t ow] dan saya mengucapkan [t ow m aa t ow]. Gambar 5.17(a) menunjukkan model transisi yang menyediakan variasi dialek ini. Setiap lingkaran dalam diagram ini merepresentasikan model fon seperti pada Gambar 5.16.

Selain variasi dialek, kata-kata dapat memiliki variasi koartikulasi. Misalnya, fon [t] dihasilkan dengan lidah di bagian atas mulut, sedangkan [ow] memiliki lidah di dekat bagian bawah. Ketika berbicara cepat, lidah tidak punya waktu untuk berada pada posisi [ow], dan kita berakhir dengan [t ah], alih-alih [t ow]. Gambar 5.17(b) memberikan model untuk "tomat" yang memperhitungkan efek koartikulasi ini. Model telepon yang lebih canggih mempertimbangkan konteks telepon di sekitarnya.

Terdapat variasi pelafalan yang substansial untuk sebuah kata. Pelafalan "because" yang paling umum adalah [b iy k ah z], tetapi itu hanya mencakup sekitar seperempat dari keseluruhan penggunaan. Seperempat lainnya (kurang lebih) mengganti [ix], [ih], atau [ax] untuk vokal pertama, dan sisanya mengganti [ax] atau [aa] untuk vokal kedua, [zh] atau [s] untuk [z] terakhir, atau menghilangkan "be" sepenuhnya, menyisakan "cuz".



Gambar 5.17 Dua model pelafalan kata "tomat". Setiap model ditampilkan sebagai diagram transisi dengan keadaan berupa lingkaran dan panah yang menunjukkan transisi yang diizinkan beserta probabilitasnya. (a) Model yang memperhitungkan perbedaan dialek. Angka 0,5 merupakan estimasi berdasarkan pelafalan yang disukai kedua penulis. (b) Model dengan efek koartikulasi pada vokal pertama, yang memperbolehkan fonem [ow] atau [ah].

Model Bahasa

Untuk pengenalan ucapan untuk keperluan umum, model bahasa dapat berupa model n -gram teks yang dipelajari dari korpus kalimat tertulis. Namun, bahasa lisan memiliki karakteristik yang berbeda dengan bahasa tertulis, sehingga lebih baik mendapatkan korpus transkrip bahasa lisan. Untuk pengenalan ucapan yang spesifik untuk tugas tertentu, korpus harus spesifik untuk tugas tertentu: untuk membangun sistem reservasi maskapai penerbangan Anda, dapatkan transkrip panggilan sebelumnya. Memiliki kosakata yang spesifik untuk tugas tertentu juga membantu, seperti daftar semua bandara dan kota yang dilayani, serta semua nomor penerbangan.

Bagian dari desain antarmuka pengguna suara adalah memaksa pengguna untuk mengatakan sesuatu dari serangkaian pilihan yang terbatas, sehingga pengenalan ucapan akan memiliki distribusi probabilitas yang lebih ketat untuk ditangani. Misalnya, bertanya "Kota mana yang ingin Anda kunjungi?" akan memunculkan respons dengan model bahasa yang sangat terbatas, sementara bertanya "Apa yang bisa saya bantu?" tidak.

Membangun Pengenal Ucapan

Kualitas sistem pengenalan ucapan bergantung pada kualitas semua komponennya—model bahasa, model pengucapan kata, model telepon, dan algoritma pemrosesan sinyal yang digunakan untuk mengekstraksi fitur spektral dari sinyal akustik. Kita telah membahas bagaimana model bahasa dapat dibangun dari korpus teks tertulis, dan kita serahkan detail pemrosesan sinyal ke buku teks lain.

Kita hanya membahas model pengucapan dan telepon. Struktur model pengucapan seperti model tomat pada Gambar 5.17 biasanya dikembangkan secara manual. Kamus pengucapan yang besar kini tersedia untuk bahasa Inggris dan bahasa lainnya, meskipun akurasi sangat bervariasi. Struktur model telepon tiga-keadaan sama untuk semua telepon, seperti yang ditunjukkan pada Gambar 5.16. Hal ini menyisakan probabilitas itu sendiri.

Seperti biasa, kita akan memperoleh probabilitas dari sebuah korpus, kali ini korpus ucapan. Jenis korpus yang paling umum diperoleh adalah korpus yang menyertakan sinyal ucapan untuk setiap kalimat yang dipasangkan dengan transkrip kata-kata. Membangun model dari korpus ini lebih sulit daripada membangun model n -gram teks, karena kita harus membangun model Markov tersembunyi urutan telepon untuk setiap kata dan status telepon untuk setiap kerangka waktu merupakan variabel tersembunyi.

Pada awal-awal pengenalan suara, variabel tersembunyi disediakan melalui pelabelan spektrogram secara manual yang rumit. Sistem terkini menggunakan ekspektasi-maksimisasi untuk secara otomatis menyediakan data yang hilang. Idenya sederhana: dengan HMM dan urutan observasi, kita dapat menggunakan algoritma penghalusan untuk menghitung probabilitas setiap status pada setiap langkah waktu dan, dengan perluasan sederhana, probabilitas setiap pasangan status-status pada langkah waktu yang berurutan. Probabilitas ini dapat dipandang sebagai label tak tentu.

Dari label tak tentu, kita dapat memperkirakan probabilitas transisi dan sensor baru, dan prosedur EM berulang. Metode ini dijamin akan meningkatkan kesesuaian antara model dan data pada setiap iterasi, dan umumnya konvergen ke kumpulan nilai parameter yang jauh lebih baik daripada yang disediakan oleh estimasi awal yang diberi label secara manual. Sistem dengan akurasi tertinggi bekerja dengan melatih model yang berbeda untuk setiap penutur, sehingga menangkap perbedaan dialek serta variasi pria/wanita dan lainnya. Pelatihan ini dapat memerlukan beberapa jam interaksi dengan penutur, sehingga sistem dengan adopsi paling luas tidak menciptakan model yang spesifik untuk penutur.

Akurasi suatu sistem bergantung pada sejumlah faktor. Pertama, kualitas sinyal penting: mikrofon terarah berkualitas tinggi yang diarahkan ke mulut yang diam di ruangan berbantalan akan jauh lebih baik daripada mikrofon murah yang mentransmisikan sinyal melalui saluran telepon dari mobil yang sedang macet dengan radio menyala. Ukuran kosakata penting: ketika mengenali rangkaian digit dengan kosakata 11 kata (1-9 ditambah "oh" dan "nol"), tingkat kesalahan kata akan di bawah 0,5%, sementara itu meningkat menjadi sekitar 10% pada berita dengan kosakata 20.000 kata, dan 20% pada korpus dengan kosakata 64.000 kata. Tugasnya juga penting: ketika sistem mencoba menyelesaikan tugas tertentu—memesan tiket pesawat atau memberikan petunjuk arah ke restoran—tugas tersebut sering kali dapat diselesaikan dengan sempurna bahkan dengan tingkat kesalahan kata sebesar 10% atau lebih.

5.6 RINGKASAN

Pemahaman bahasa alami merupakan salah satu subbidang AI yang paling penting. Tidak seperti kebanyakan bidang AI lainnya, pemahaman bahasa alami memerlukan investigasi empiris terhadap perilaku manusia yang sebenarnya—yang ternyata kompleks dan menarik.

- Teori bahasa formal dan tata bahasa struktur frasa (dan khususnya, tata bahasa bebas konteks) merupakan alat yang berguna untuk menangani beberapa aspek bahasa alami. Formalisme tata bahasa bebas konteks probabilistik (PCFG) banyak digunakan.
- Kalimat dalam bahasa bebas konteks dapat diurai dalam waktu $O(n^3)$ oleh pengurai bagan seperti algoritma CYK, yang mengharuskan aturan tata bahasa berada dalam Bentuk Normal Chomsky.
- Treebank dapat digunakan untuk mempelajari tata bahasa. Mempelajari tata bahasa dari korpus kalimat yang belum diurai juga dimungkinkan, tetapi cara ini kurang berhasil.
- PCFG yang terleksikalisasi memungkinkan kita untuk merepresentasikan bahwa beberapa hubungan antar kata lebih umum daripada yang lain.
- Memperkaya tata bahasa dapat dilakukan dengan mudah untuk menangani masalah seperti kesesuaian subjek-verba dan kasus pronomina. Tata bahasa klausa pasti (DCG) adalah formalisme yang memungkinkan augmentasi. Dengan DCG, penguraian dan interpretasi semantik (bahkan pembangkitan) dapat dilakukan menggunakan inferensi logis.
- Interpretasi semantik juga dapat ditangani oleh tata bahasa yang diperbesar.
- Ambiguitas merupakan masalah yang sangat penting dalam pemahaman bahasa alami; Sebagian besar kalimat memiliki banyak kemungkinan interpretasi, tetapi biasanya hanya satu yang tepat. Disambiguasi bergantung pada pengetahuan tentang dunia, tentang situasi terkini, dan tentang penggunaan bahasa.
- Sistem penerjemahan mesin telah diimplementasikan menggunakan berbagai teknik, mulai dari analisis sintaksis dan semantik lengkap hingga teknik statistik berdasarkan frekuensi frasa. Saat ini, model statistik adalah yang paling populer dan paling berhasil.
- Sistem pengenalan ucapan juga terutama didasarkan pada prinsip-prinsip statistik. Sistem ucapan populer dan bermanfaat, meskipun tidak sempurna.
- Bersama-sama, penerjemahan mesin dan pengenalan ucapan merupakan dua kesuksesan besar teknologi bahasa alami. Salah satu alasan mengapa model-model ini berkinerja baik adalah karena tersedianya korpus yang besar—baik penerjemahan maupun ucapan adalah tugas yang dilakukan "di alam bebas" oleh orang-orang setiap hari. Sebaliknya, tugas-tugas seperti penguraian kalimat kurang berhasil, sebagian karena tidak ada korpus besar kalimat yang telah diurai yang tersedia "di alam bebas" dan sebagian karena penguraian itu sendiri tidak bermanfaat.

BAB 6

PERSEPSI

Persepsi memberi agen informasi tentang dunia yang mereka huni dengan menafsirkan respons sensor. **Sensor** mengukur beberapa aspek lingkungan dalam bentuk yang dapat digunakan sebagai input oleh program agen. Sensor bisa sesederhana sakelar, yang memberikan satu bit yang memberi tahu apakah sakelar menyala atau mati, atau serumit mata. Berbagai modalitas sensorik tersedia untuk agen buatan. Modalitas yang mereka miliki bersama manusia meliputi penglihatan, pendengaran, dan sentuhan. Modalitas yang tidak tersedia bagi manusia tanpa bantuan meliputi sinyal radio, inframerah, GPS, dan nirkabel. Beberapa robot melakukan penginderaan aktif, artinya mereka mengirimkan sinyal, seperti radar atau ultrasound, dan merasakan pantulan sinyal ini dari lingkungan. Alih-alih mencoba membahas semua ini, bab ini akan membahas satu modalitas secara mendalam: penglihatan.

Kita telah melihat dalam deskripsi POMDP bahwa agen teoretis keputusan berbasis model dalam lingkungan yang dapat diamati sebagian memiliki **model sensor**—distribusi probabilitas $P(E|S)$ atas bukti yang diberikan sensornya, mengingat keadaan dunia. Aturan Bayes kemudian dapat digunakan untuk memperbarui estimasi keadaan tersebut.

Untuk penglihatan, model sensor dapat dipecah menjadi dua komponen: **Model objek** menggambarkan objek yang menghuni dunia visual—orang, bangunan, pohon, mobil, dll. Model objek dapat mencakup model geometris 3D presisi yang diambil dari sistem desain berbantuan komputer (CAD), atau dapat berupa batasan yang samar, seperti fakta bahwa mata manusia biasanya berjarak 5 hingga 7 cm. **Model rendering** menggambarkan proses fisik, geometris, dan statistik yang menghasilkan stimulus dari dunia. Model rendering cukup akurat, tetapi ambigu. Misalnya, objek putih dalam cahaya redup mungkin tampak berwarna sama dengan objek hitam dalam cahaya intens. Objek kecil di dekat mungkin tampak sama dengan objek besar di jauh. Tanpa bukti tambahan, kita tidak dapat membedakan apakah gambar yang memenuhi bingkai adalah Godzilla mainan atau monster sungguhan.

Ambiguitas dapat diatasi dengan pengetahuan sebelumnya—kita tahu Godzilla tidak nyata, jadi gambar tersebut pasti mainan—atau dengan memilih untuk mengabaikan ambiguitas tersebut secara selektif. Misalnya, sistem penglihatan untuk mobil otonom mungkin tidak dapat menginterpretasikan objek yang jauh, tetapi agen dapat memilih untuk mengabaikan masalah tersebut, karena kecil kemungkinannya untuk menabrak objek yang jaraknya bermil-mil jauhnya.

Agen teoretis-keputusan bukanlah satu-satunya arsitektur yang dapat memanfaatkan sensor penglihatan. Misalnya, lalat buah (*Drosophila*) sebagian

merupakan agen refleks: mereka memiliki serat servikal raksasa yang membentuk jalur langsung dari sistem visual mereka ke otot-otot sayap yang memicu respons melarikan diri—reaksi langsung, tanpa pertimbangan. Lalat dan banyak hewan terbang lainnya memanfaatkan arsitektur kontrol loop tertutup untuk mendarat di suatu objek. Sistem visual mengekstrak perkiraan jarak ke objek, dan sistem kontrol menyesuaikan otot-otot sayapnya, memungkinkan perubahan arah yang sangat cepat, tanpa memerlukan model objek yang detail.

Dibandingkan dengan data dari sensor lain (seperti bit tunggal yang memberi tahu robot vakum bahwa ia telah menabrak dinding), observasi visual sangat kaya, baik dari segi detail yang dapat diungkapkan maupun jumlah data yang dihasilkannya. Kamera video untuk aplikasi robotik dapat menghasilkan satu juta piksel 24-bit pada 60 Hz; kecepatan 10 GB per menit. Permasalahan bagi agen yang berkemampuan penglihatan adalah: *Aspek mana dari stimulus visual yang kaya yang harus dipertimbangkan untuk membantu agen membuat pilihan tindakan yang baik, dan aspek mana yang harus diabaikan?* Visi—dan semua persepsi—berfungsi untuk memajukan tujuan agen, bukan sebagai tujuan akhir.

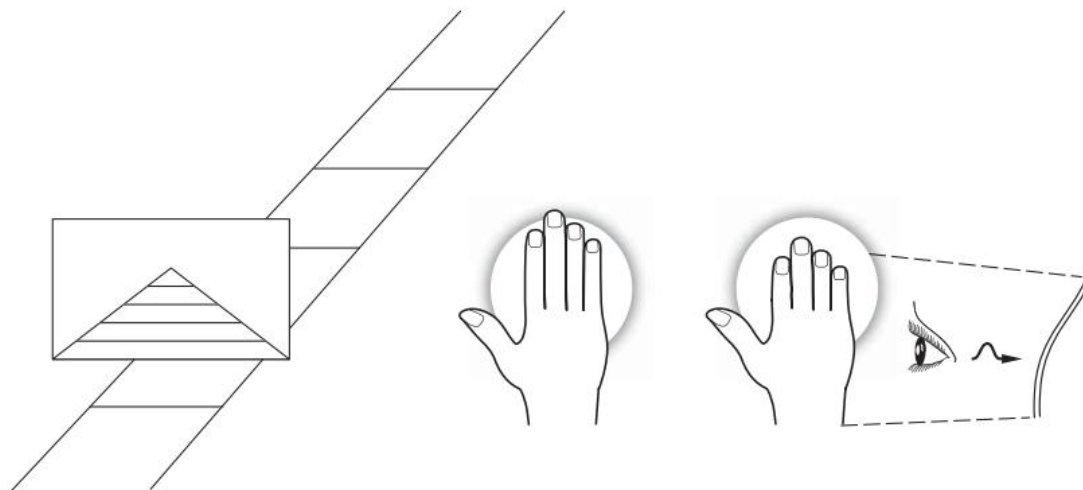
Kita dapat menggolongkan tiga pendekatan umum untuk masalah ini. Pendekatan **ekstraksi fitur**, seperti yang ditunjukkan oleh *Drosophila*, menekankan komputasi sederhana yang diterapkan langsung pada observasi sensor. Dalam **pengakuan** pengenalan, agen membedakan objek-objek yang ditemuinya berdasarkan informasi visual dan informasi lainnya. Pengenalan dapat berarti memberi label ya atau tidak pada setiap gambar, apakah gambar tersebut berisi makanan yang harus kita cari, atau berisi wajah Nenek. Terakhir, dalam pendekatan **rekonstruksi**, agen membangun model geometris dunia dari sebuah gambar atau serangkaian gambar. Tiga puluh tahun terakhir penelitian telah menghasilkan perangkat dan metode yang ampuh untuk menangani pendekatan-pendekatan ini. Memahami metode-metode ini membutuhkan pemahaman tentang proses pembentukan gambar. Oleh karena itu, kita sekarang membahas fenomena fisika dan statistik yang terjadi dalam produksi gambar.

6.1 PEMBENTUKAN GAMBAR

Pencitraan mendistorsi tampilan objek. Misalnya, gambar yang diambil dengan melihat ke bawah pada rangkaian rel kereta api yang panjang dan lurus akan menunjukkan bahwa rel-rel tersebut bertemu dan menyatu. Contoh lain, jika Anda meletakkan tangan di depan mata, Anda dapat menghalangi bulan, yang tidak lebih kecil dari tangan Anda. Saat Anda menggerakkan tangan maju mundur atau memiringkannya, tangan Anda akan tampak mengecil dan membesar *dalam gambar*, tetapi kenyataannya tidak demikian (Gambar 6.1). Model efek ini penting untuk pengenalan dan rekonstruksi.

Gambar tanpa lensa: Kamera lubang jarum

Sensor gambar mengumpulkan cahaya yang tersebar dari objek dalam suatu **pemandangan** dan menciptakan **gambar** dua dimensi. Di mata, gambar terbentuk di retina, yang terdiri dari dua jenis sel: sekitar 100 juta sel batang, yang sensitif terhadap cahaya pada rentang panjang gelombang yang luas, dan 5 juta sel kerucut.



Gambar 6.1 Pencitraan mendistorsi geometri. Garis-garis paralel tampak bertemu di kejauhan, seperti pada gambar rel kereta api di sebelah kiri. Di tengah, sebuah tangan kecil menutupi sebagian besar bulan yang besar. Di sebelah kanan terdapat efek pemendekan: tangan dimiringkan menjauhi mata, membuatnya tampak lebih pendek daripada di gambar tengah.

Sel kerucut, yang penting untuk penglihatan warna, terdiri dari tiga jenis utama, yang masing-masing sensitif terhadap serangkaian panjang gelombang yang berbeda. Pada kamera, gambar dibentuk pada bidang gambar, yang dapat berupa selembur film berlapis perak halida atau kisi persegi panjang yang terdiri dari beberapa juta **piksel** fotosensitif, masing-masing berupa semikonduktor oksida logam komplementer (CMOS) atau perangkat penggandeng muatan (CCD). Setiap foton yang tiba di sensor menghasilkan efek, yang kekuatannya bergantung pada panjang gelombang foton. Keluaran sensor adalah jumlah semua efek akibat foton yang diamati dalam suatu rentang waktu, yang berarti bahwa sensor gambar melaporkan rata-rata tertimbang dari intensitas cahaya yang tiba di sensor.

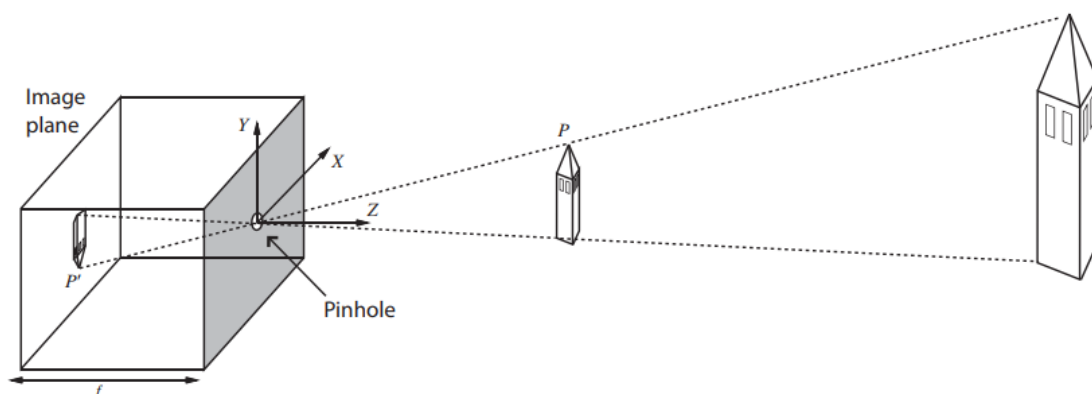
Untuk melihat gambar yang terfokus, kita harus memastikan bahwa semua foton dari titik yang kira-kira sama dalam pemandangan tiba di titik yang kira-kira sama pada bidang gambar. Cara paling sederhana untuk membentuk gambar terfokus adalah dengan melihat objek diam menggunakan **kamera lubang jarum**, yang terdiri dari lubang jarum, *O*, di bagian depan kotak, dan bidang gambar di bagian belakang kotak (Gambar 6.2). Foton dari pemandangan harus melewati lubang jarum, jadi jika lubang jarum

cukup kecil, foton di dekatnya akan berada di dekat bidang gambar, dan gambar akan terfokus.

Geometri pemandangan dan gambar paling mudah dipahami dengan kamera lubang jarum. Kami menggunakan sistem koordinat tiga dimensi dengan titik asal di lubang jarum, dan mempertimbangkan titik P di pemandangan, dengan koordinat (X, Y, Z) . P diproyeksikan ke titik P' di bidang gambar dengan koordinat (x, y, z) . Jika f adalah jarak dari lubang jarum ke bidang gambar, maka dengan segitiga sebangun, kita dapat memperoleh persamaan berikut:

$$\frac{-x}{f} = \frac{X}{Z'} \quad \frac{-y}{f} = \frac{Y}{Z} \Rightarrow x = \frac{-fX}{Z}, \quad y = \frac{-fY}{Z}$$

Persamaan-persamaan ini mendefinisikan proses pembentukan gambar yang dikenal sebagai **proyeksi perspektif**. Perhatikan bahwa Z dalam penyebut berarti semakin jauh suatu objek, semakin kecil bayangannya. Perhatikan juga bahwa tanda minus berarti bayangan *terbalik*, baik kiri-kanan maupun atas-bawah, dibandingkan dengan pemandangan.



Gambar 6.2 Setiap elemen peka cahaya pada bidang gambar di bagian belakang kamera lubang jarum menerima cahaya dari rentang arah yang sempit yang melewati lubang jarum. Jika lubang jarum cukup kecil, hasilnya adalah gambar terfokus di bagian belakang lubang jarum. Proses proyeksi berarti objek besar yang jauh terlihat sama dengan objek kecil yang dekat. Perhatikan bahwa gambar diproyeksikan terbalik.

Dalam proyeksi perspektif, objek yang jauh tampak kecil. Inilah yang memungkinkan Anda menutupi bulan dengan tangan Anda (Gambar 6.1). Hasil penting dari efek ini adalah garis-garis paralel bertemu di satu titik di cakrawala. (Bayangkan rel kereta api, Gambar 6.1.) Sebuah garis dalam pemandangan ke arah (U, V, W) dan melewati titik (X_0, Y_0, Z_0) dapat digambarkan sebagai himpunan titik-titik $(X_0 +$

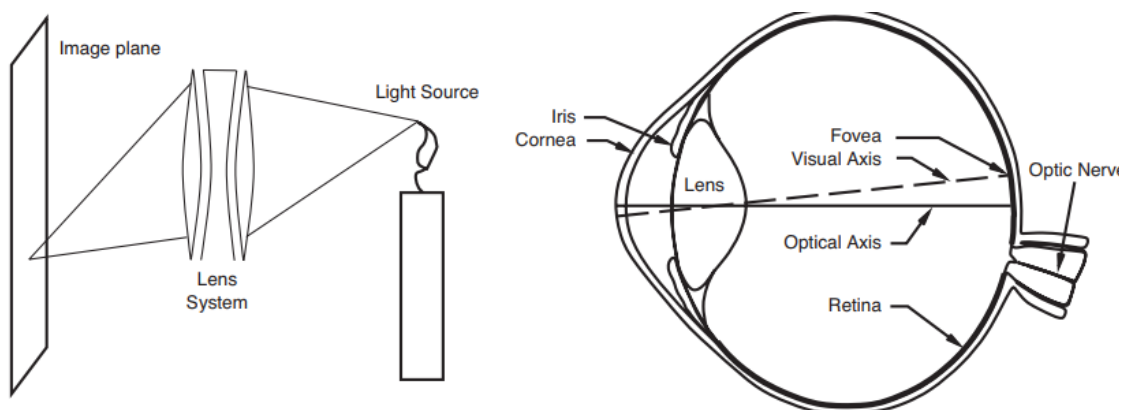
$\lambda U, Y_0 + \lambda V, Z_0 + \lambda W$), dengan λ bervariasi antara $-\infty$ dan $+\infty$. Pilihan yang berbeda dari (X_0, Y_0, Z_0) menghasilkan garis-garis yang berbeda yang sejajar satu sama lain. Proyeksi titik P_λ dari garis ini ke bidang gambar diberikan oleh

$$\left(f \frac{X_0 + \lambda U}{Z_0 + \lambda W}, f \frac{Y_0 + \lambda V}{Z_0 + \lambda W} \right)$$

Karena $\lambda \rightarrow \infty$ atau $\lambda \rightarrow -\infty$, ini menjadi $p_\infty = (fU/W, fV/W)$ jika $W \neq 0$. Ini berarti bahwa dua garis sejajar yang meninggalkan titik-titik berbeda di ruang angkasa akan bertemu di dalam gambar—untuk λ yang besar, titik-titik gambar hampir sama, berapa pun nilai (X_0, Y_0, Z_0) (sekali lagi, bayangkan rel kereta api, Gambar 6.1). Kita menyebut p_∞ **titik hilang** yang terkait dengan keluarga garis lurus dengan arah (U, V, W) . Garis-garis dengan arah yang sama memiliki titik hilang yang sama.

Sistem Lensa

Kelemahan kamera lubang jarum adalah kita membutuhkan lubang jarum kecil untuk menjaga gambar tetap fokus. Namun, semakin kecil lubang jarum, semakin sedikit foton yang masuk, yang berarti gambar akan menjadi gelap. Kita bisa mengumpulkan lebih banyak foton dengan membiarkan lubang jarum terbuka lebih lama, tetapi kemudian kita akan mendapatkan efek **kabur gerak**—objek dalam pemandangan yang bergerak akan tampak kabur karena mengirimkan foton ke beberapa lokasi pada bidang gambar. Jika kita tidak bisa membiarkan lubang jarum terbuka lebih lama, kita bisa mencoba memperbesarnya. Lebih banyak cahaya akan masuk, tetapi cahaya dari sepetak kecil objek dalam pemandangan sekarang akan tersebar ke sepetak kecil pada bidang gambar, menyebabkan gambar menjadi kabur.



Gambar 6.3 Lensa mengumpulkan cahaya yang meninggalkan titik pemandangan ke berbagai arah, dan mengarahkannya hingga tiba di satu titik pada bidang gambar.

Pemfokusan berfungsi untuk titik-titik yang terletak dekat dengan bidang fokus di ruang angkasa; titik-titik lain tidak akan terfokus dengan baik. Pada kamera, elemen-

elemen sistem lensa bergerak untuk mengubah bidang fokus, sedangkan pada mata, bentuk lensa diubah oleh otot-otot khusus.

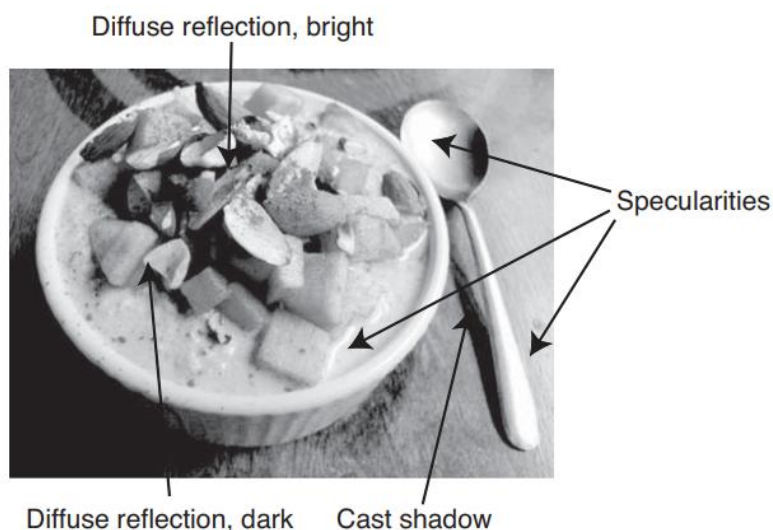
Mata vertebrata dan kamera modern menggunakan sistem **lensa** untuk mengumpulkan cahaya yang cukup sekaligus menjaga fokus gambar. Sebuah bukaan besar ditutupi lensa yang memfokuskan cahaya dari lokasi objek terdekat ke lokasi terdekat lainnya pada bidang gambar. Namun, sistem lensa memiliki **kedalaman bidang** yang terbatas: sistem ini hanya dapat memfokuskan cahaya dari titik-titik yang berada dalam rentang kedalaman tertentu (berpusat di sekitar **bidang fokus**). Objek di luar rentang ini akan berada di luar fokus gambar. Untuk menggerakkan bidang fokus, lensa pada mata dapat berubah bentuk (Gambar 6.3); pada kamera, lensa bergerak maju mundur.

Proyeksi ortografis berskala

Efek perspektif tidak selalu terlihat jelas. Misalnya, bintik-bintik pada macan tutul yang jauh mungkin terlihat kecil karena macan tutul tersebut berada jauh, tetapi dua bintik yang bersebelahan akan memiliki ukuran yang kurang lebih sama. Hal ini karena perbedaan jarak ke bintik-bintik tersebut kecil dibandingkan dengan jarak ke bintik-bintik tersebut, sehingga kita dapat menyederhanakan model proyeksi. Model yang tepat adalah **proyeksi ortografis berskala**. Idennya adalah sebagai berikut: Jika kedalaman Z titik-titik pada objek bervariasi dalam rentang tertentu $Z_0 \pm \Delta Z$, dengan $\Delta Z \ll Z_0$, maka faktor skala perspektif f/Z dapat didekati dengan konstanta $s = f/Z_0$. Persamaan proyeksi dari koordinat pemandangan (X, Y, Z) ke bidang gambar menjadi $x = sX$ dan $y = sY$. Proyeksi ortografi berskala adalah perkiraan yang hanya berlaku untuk bagian-bagian pemandangan dengan variasi kedalaman internal yang tidak banyak. Misalnya, proyeksi ortografi berskala dapat menjadi model yang baik untuk fitur-fitur di bagian depan bangunan yang jauh.

Cahaya dan Bayangan

Kecerahan piksel dalam gambar merupakan fungsi dari kecerahan patch permukaan dalam pemandangan yang diproyeksikan ke piksel tersebut. Kami akan mengasumsikan model linier (kamera saat ini memiliki non-linieritas pada ekstrem terang dan gelap, tetapi linier di tengah). Kecerahan gambar merupakan isyarat yang kuat, meskipun ambigu, terhadap bentuk suatu objek, dan dari sana, terhadap identitasnya.

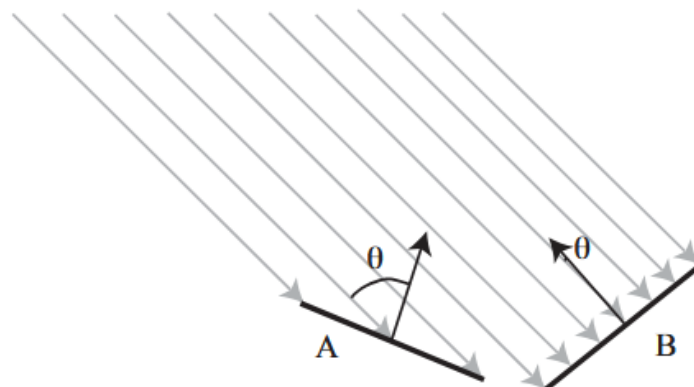


Gambar 6.4 Berbagai efek iluminasi. Terdapat spekularitas pada sendok logam dan susu. Permukaan difusi yang terang tampak terang karena menghadap arah cahaya. Permukaan difusi yang gelap tampak gelap karena bersinggungan dengan arah iluminasi. Bayangan muncul di titik-titik permukaan yang tidak dapat melihat sumber cahaya.

Orang biasanya dapat membedakan tiga penyebab utama variasi kecerahan dan merekayasa balik sifat-sifat objek. Penyebab pertama adalah **intensitas cahaya secara keseluruhan**. Meskipun objek putih dalam bayangan mungkin kurang terang dibandingkan objek hitam di bawah sinar matahari langsung, mata dapat membedakan kecerahan relatif dengan baik, dan menganggap objek putih sebagai putih. Kedua, titik-titik yang berbeda dalam pemandangan dapat **memantulkan** lebih banyak atau lebih sedikit cahaya. Biasanya, hasilnya adalah orang-orang menganggap titik-titik ini lebih terang atau lebih gelap, sehingga melihat tekstur atau tanda pada objek. Ketiga, bidang permukaan yang menghadap cahaya lebih terang daripada bidang permukaan yang menjauhi cahaya, efek yang dikenal sebagai bayangan. Biasanya, orang dapat mengetahui bahwa bayangan ini berasal dari geometri objek, tetapi terkadang mereka salah mengartikan **bayangan** dan tanda. Misalnya, guratan riasan gelap di bawah tulang pipi sering kali terlihat seperti efek bayangan, membuat wajah terlihat lebih tirus.

Sebagian besar permukaan memantulkan cahaya melalui proses **refleksi difus**. Pantulan difus menyebarkan cahaya secara merata di seluruh arah yang meninggalkan permukaan, sehingga kecerahan permukaan difus tidak bergantung pada arah pandang. Sebagian besar kain, cat, permukaan kayu kasar, vegetasi, dan batu kasar bersifat difus. Cermin tidak bersifat difus, karena apa yang Anda lihat bergantung pada arah Anda melihat cermin. Perilaku cermin sempurna dikenal sebagai **pantulan spekular**. Beberapa permukaan—seperti logam yang disikat, plastik, atau lantai basah—menampilkan bercak kecil tempat pantulan spekular terjadi, disebut **spekularitas**. Ini mudah diidentifikasi,

karena kecil dan terang (Gambar 6.4). Untuk hampir semua tujuan, cukup memodelkan semua permukaan sebagai difus dengan spekularitas.



Gambar 6.5 Dua bidang permukaan diterangi oleh sumber titik yang jauh, yang sinarnya ditunjukkan sebagai panah abu-abu. Bidang A dimiringkan menjauhi sumber (θ mendekati 900) dan mengumpulkan lebih sedikit energi, karena memotong lebih sedikit sinar cahaya per satuan luas permukaan. Bidang B, yang menghadap sumber (θ mendekati 0⁰), mengumpulkan lebih banyak energi.

Sumber utama penerangan di luar ruangan adalah matahari, yang sinarnya bergerak sejajar satu sama lain. Kami memodelkan perilaku ini sebagai **sumber cahaya titik yang jauh**. Ini adalah model pencahayaan yang paling penting, dan cukup efektif untuk pemandangan di dalam ruangan maupun di luar ruangan. Jumlah cahaya yang dikumpulkan oleh suatu bidang permukaan dalam model ini bergantung pada sudut θ antara arah penerangan dan garis normal permukaan.

Bidang permukaan difus yang diterangi oleh sumber cahaya titik yang jauh akan memantulkan sebagian cahaya yang dikumpulkannya; fraksi ini disebut **albedo difus**. Kertas putih dan salju memiliki albedo tinggi, sekitar 0,90, sedangkan beludru hitam polos dan arang memiliki albedo rendah, sekitar 0,05 (yang berarti 95% cahaya yang masuk diserap oleh serat beludru atau pori-pori arang). **Hukum kosinus Lambert** menyatakan bahwa kecerahan suatu bidang difus ditentukan oleh

$$I = \rho I_0 \cos \theta ,$$

di mana ρ adalah albedo difus, I_0 adalah intensitas sumber cahaya dan θ adalah sudut antara arah sumber cahaya dan permukaan normal (lihat Gambar 6.5). Hukum Lampert memprediksi piksel gambar terang berasal dari bercak permukaan yang menghadap cahaya secara langsung dan piksel gelap berasal dari bercak yang hanya melihat cahaya secara tangensial, sehingga bayangan pada permukaan memberikan beberapa informasi bentuk. Kami mengeksplorasi isyarat ini di Bagian selanjutnya. Jika permukaan tidak dijangkau oleh sumber cahaya, maka permukaan tersebut berada

dalam **bayangan**. Bayangan sangat jarang berwarna hitam seragam, karena permukaan yang dibayangi menerima sebagian cahaya dari sumber lain. Di luar ruangan, sumber yang paling penting adalah langit, yang cukup terang. Di dalam ruangan, cahaya yang dipantulkan dari permukaan lain menerangi bercak yang dibayangi. **Interrefleksi** ini juga dapat memiliki efek signifikan pada kecerahan permukaan lain. Efek ini terkadang dimodelkan dengan menambahkan istilah **iluminasi sekitar** yang konstan ke intensitas yang diprediksi.

Warna

Buah adalah suap yang ditawarkan pohon kepada hewan untuk membawa bijinya. Pohon berevolusi untuk menghasilkan buah yang berubah menjadi merah atau kuning saat matang, dan hewan berevolusi untuk mendeteksi perubahan warna ini. Cahaya yang sampai ke mata memiliki jumlah energi yang berbeda pada panjang gelombang yang berbeda; hal ini dapat direpresentasikan oleh fungsi kerapatan energi spektral. Mata manusia merespons cahaya dalam rentang panjang gelombang 380–750 nm, dengan tiga jenis sel reseptor warna yang berbeda, yang memiliki penerimaan puncak pada 420 nm (biru), 540 nm (hijau), dan 570 nm (merah). Mata manusia hanya dapat menangkap sebagian kecil dari fungsi kerapatan energi spektral penuh—tetapi cukup untuk mengetahui kapan buah matang.

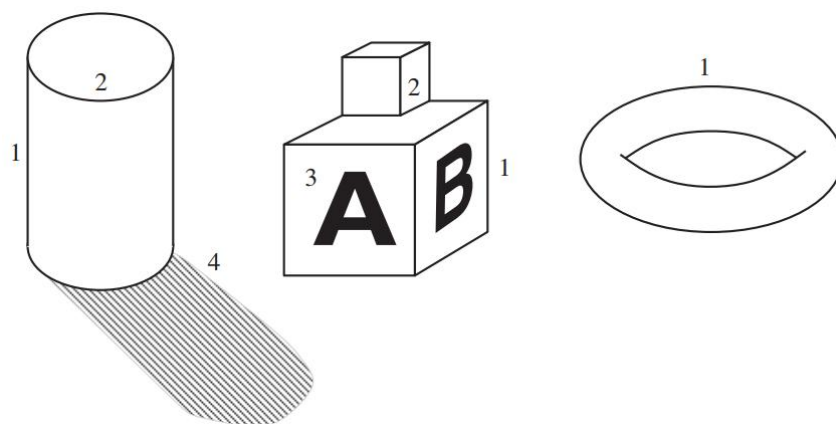
Prinsip trikromasi menyatakan bahwa untuk setiap kerapatan energi spektral, betapa pun rumitnya, dimungkinkan untuk membangun kerapatan energi spektral lain yang terdiri dari campuran hanya tiga warna—biasanya merah, hijau, dan biru—sedemikian rupa sehingga manusia tidak dapat membedakan keduanya. Artinya, TV dan layar komputer kita dapat bertahan hanya dengan tiga elemen warna merah/hijau/biru (atau R/G/B). Hal ini juga mempermudah algoritma visi komputer kita. Setiap permukaan dapat dimodelkan dengan tiga albedo berbeda untuk R/G/B. Demikian pula, setiap sumber cahaya dapat dimodelkan dengan tiga intensitas R/G/B. Kita kemudian menerapkan hukum kosinus Lambert untuk masing-masing elemen tersebut untuk mendapatkan tiga nilai piksel R/G/B. Model ini memprediksi, dengan tepat, bahwa permukaan yang sama akan menghasilkan bercak gambar berwarna berbeda di bawah cahaya berwarna berbeda. Faktanya, pengamat manusia cukup mahir mengabaikan efek cahaya berwarna berbeda dan mampu memperkirakan warna permukaan di bawah cahaya putih, sebuah efek yang dikenal sebagai **konstanta warna**. Algoritma konstanta warna yang cukup akurat kini tersedia; versi sederhananya muncul dalam fungsi "keseimbangan putih otomatis" pada kamera Anda. Perhatikan bahwa jika kita ingin membuat kamera untuk udang mantis, kita membutuhkan 12 warna piksel berbeda, yang sesuai dengan 12 jenis reseptor warna krustasea tersebut.

6.2 OPERASI PEMROSESAN CITRA AWAL

Kita telah melihat bagaimana cahaya dipantulkan dari objek dalam pemandangan untuk membentuk gambar yang terdiri dari, katakanlah, lima juta piksel 3-byte. Dengan

semua sensor, akan ada noise pada gambar, dan bagaimanapun juga, ada banyak data yang harus ditangani. Jadi, bagaimana kita mulai menganalisis data ini?

Di bagian ini, kita akan mempelajari tiga operasi pemrosesan gambar yang bermanfaat: deteksi tepi, analisis tekstur, dan komputasi aliran optik. Operasi-operasi ini disebut operasi "awal" atau "tingkat rendah" karena merupakan yang pertama dalam serangkaian operasi. Operasi penglihatan awal dicirikan oleh sifat lokalnya (dapat dilakukan di satu bagian gambar tanpa memperhatikan apa pun yang berjarak lebih dari beberapa piksel) dan oleh kurangnya pengetahuan: kita dapat melakukan operasi ini tanpa mempertimbangkan objek yang mungkin ada dalam pemandangan. Hal ini menjadikan operasi tingkat rendah kandidat yang baik untuk diimplementasikan dalam perangkat keras paralel—baik dalam unit prosesor grafis (GPU) maupun mata. Selanjutnya, kita akan melihat satu operasi tingkat menengah: mengelompokkan gambar ke dalam beberapa wilayah.



Gambar 6.6 Berbagai jenis tepi: (1) diskontinuitas kedalaman; (2) diskontinuitas orientasi permukaan; (3) diskontinuitas reflektansi; (4) diskontinuitas iluminasi (bayangan).

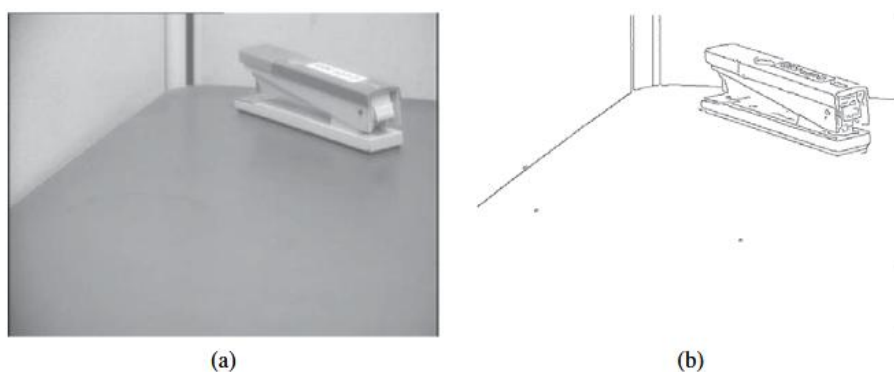
Deteksi Tepi

Tepi adalah garis lurus atau kurva pada bidang gambar yang mengalami perubahan kecerahan gambar yang "signifikan". Tujuan deteksi tepi adalah untuk mengabstraksikan gambar multimegabita yang berantakan menuju representasi yang lebih ringkas dan abstrak, seperti pada Gambar 6.6. Motivasinya adalah kontur tepi pada gambar bersesuaian dengan kontur pemandangan yang penting.

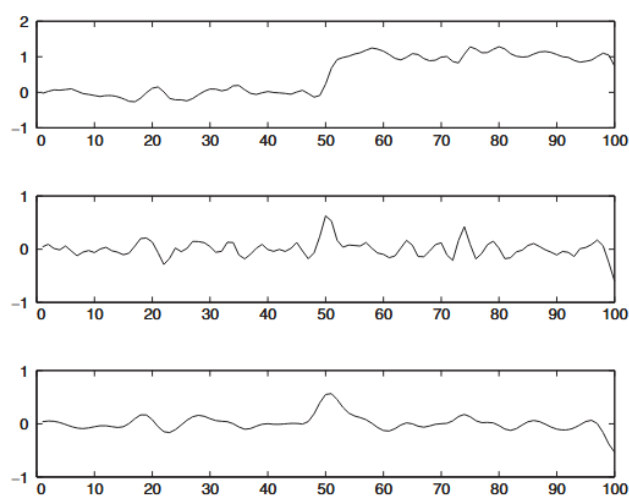
Pada gambar, terdapat tiga contoh diskontinuitas kedalaman, berlabel 1; dua diskontinuitas permukaan-normal, berlabel 2; diskontinuitas reflektansi, berlabel 3; dan diskontinuitas iluminasi (bayangan), berlabel 4. Deteksi tepi hanya berkaitan dengan gambar, sehingga tidak membedakan berbagai jenis diskontinuitas pemandangan ini; pemrosesan selanjutnya akan membedakannya.

Gambar 6.7(a) menunjukkan gambar pemandangan yang berisi stapler yang diletakkan di atas meja, dan (b) menunjukkan keluaran algoritma deteksi tepi pada gambar ini. Seperti yang Anda lihat, terdapat perbedaan antara keluaran dan gambar garis ideal. Terdapat celah di mana tidak ada tepi yang muncul, dan terdapat tepi "noise" yang tidak sesuai dengan apa pun yang signifikan dalam gambar. Tahap pemrosesan selanjutnya harus mengoreksi kesalahan ini.

Bagaimana kita mendeteksi tepi pada gambar? Perhatikan profil kecerahan gambar di sepanjang penampang satu dimensi yang tegak lurus terhadap suatu tepi—misalnya, tepi antara tepi kiri meja dan dinding. Tampilannya kurang lebih seperti yang ditunjukkan pada Gambar 6.8 (atas). Tepi berhubungan dengan lokasi dalam gambar di mana kecerahan mengalami perubahan tajam, sehingga ide yang naif adalah membedakan gambar dan mencari tempat-tempat di mana besaran turunan $I'(x)$ besar. Hal itu hampir berhasil. Pada Gambar 6.8 (tengah), kita melihat bahwa memang terdapat puncak pada $x = 50$, tetapi terdapat juga puncak tambahan di lokasi lain (misalnya, $x = 75$). Hal ini muncul karena adanya noise pada gambar. Jika kita menghaluskan gambar terlebih dahulu, puncak-puncak palsu akan berkurang, seperti yang kita lihat di bagian bawah gambar.



Gambar 6.7 (a) Foto stapler. (b) Tepi dihitung dari (a).



Gambar 6.8 Atas: Profil intensitas $I(x)$ sepanjang penampang satu dimensi melintasi suatu tepi pada $x = 50$. Tengah: Turunan dari intensitas, $I'(x)$. Nilai besar fungsi ini berkorespondensi dengan tepi, tetapi fungsinya bernoise. Bawah: Turunan dari versi halus dari intensitas, $(I * G_\sigma)'$, yang dapat dihitung dalam satu langkah sebagai konvolusi $I * G_\sigma'$. Tepi kandidat noise pada $x = 75$ telah menghilang.

Pengukuran kecerahan pada piksel di kamera CCD didasarkan pada proses fisik yang melibatkan penyerapan foton dan pelepasan elektron; pasti akan ada fluktuasi statistik pengukuran—derau. Derau tersebut dapat dimodelkan dengan distribusi probabilitas Gaussian, dengan setiap piksel independen satu sama lain. Salah satu cara untuk menghaluskan gambar adalah dengan menetapkan rata-rata tetangganya pada setiap piksel. Hal ini cenderung meniadakan nilai ekstrem. Namun, berapa banyak tetangga yang harus kita pertimbangkan—satu piksel jauhnya, atau dua, atau lebih? Salah satu jawaban yang baik adalah rata-rata tertimbang yang memberi bobot paling besar pada piksel terdekat, lalu secara bertahap mengurangi bobot untuk piksel yang lebih jauh. **Filter Gaussian** melakukan hal itu. (Pengguna Photoshop mengenali ini sebagai operasi Gaussian blur.) Ingat bahwa fungsi Gaussian dengan deviasi standar σ dan rata-rata 0 adalah

$$N_{\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2 / 2\sigma^2} \quad \text{dalam satu dimensi,}$$

$$N_{\sigma}(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x^2 + y^2) / 2\sigma^2} \quad \text{atau}$$

$$N_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2 + y^2) / 2\sigma^2} \quad \text{dalam dua dimensi.}$$

Penerapan filter Gaussian menggantikan intensitas $I(x_0, y_0)$ dengan jumlah, atas semua piksel (x, y) , dari $I(x, y)N_\sigma(d)$ di mana d adalah jarak dari (x_0, y_0) ke (x, y) . Jenis penjumlahan terbobot ini sangat umum sehingga memiliki nama dan notasi khusus. Kita katakan bahwa fungsi h adalah **konvolusi** dari dua fungsi f dan g (dilambangkan $f * g$) jika kita memiliki

$$h(x) = (f * g)(x) = \sum_{u=-\infty}^{+\infty} f(u)g(x-u)$$

dalam satu dimensi,
atau

$$h(x,y) = (f * g)(x,y) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v)g(x-u,y-v)$$

menjadi dua.

Jadi, fungsi penghalusan dicapai dengan mengonvolusi gambar menggunakan Gaussian, $I * N_{\sigma}$. σ sebesar 1 piksel cukup untuk menghaluskan sedikit noise, sedangkan 2 piksel akan menghaluskan noise yang lebih besar, tetapi dengan kehilangan beberapa detail.

Karena pengaruh Gaussian memudar dengan cepat pada jarak tertentu, kita dapat mengganti $\pm\infty$ dalam penjumlahan dengan $\pm 3\sigma$.

Kita dapat mengoptimalkan komputasi dengan menggabungkan penghalusan dan pencarian tepi menjadi satu operasi tunggal. Sebuah teorema menyatakan bahwa untuk fungsi apa pun f dan g , turunan dari konvolusi, $(f * g)'$, sama dengan konvolusi dengan turunannya, $f * (g')$. Jadi, daripada menghaluskan gambar lalu mendiferensiasikannya, kita cukup mengonvolusi gambar dengan turunan dari fungsi penghalusan, N'_σ . Kita kemudian menandai puncak-puncak respons yang berada di atas ambang batas tertentu sebagai tepi.

Algoritma ini memiliki generalisasi alami dari penampang satu dimensi ke gambar dua dimensi umum. Dalam dua dimensi, tepi dapat berada pada sudut θ berapa pun. Dengan mempertimbangkan kecerahan gambar sebagai fungsi skalar dari variabel x, y , gradiennya adalah vektor.

$$\nabla I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix} = \begin{pmatrix} I_x \\ I_y \end{pmatrix}$$

Tepian berhubungan dengan lokasi pada gambar di mana kecerahan mengalami perubahan tajam, sehingga besaran gradien, $\|\nabla I\|$, seharusnya besar pada titik tepian. Arah gradien juga penting.

$$\frac{\nabla I}{\|\nabla I\|} = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix}$$

Ini memberi kita $\theta = \theta(x, y)$ pada setiap piksel, yang menentukan **orientasi** tepi pada piksel tersebut.

Seperti dalam satu dimensi, untuk membentuk gradien, kita tidak menghitung ∇I , melainkan $\nabla(I * N_\sigma)$, gradien setelah menghaluskan gambar dengan mengonvolusinya menggunakan Gaussian. Dan sekali lagi, jalan pintasnya adalah bahwa ini setara dengan mengonvolusi gambar dengan turunan parsial Gaussian. Setelah kita menghitung gradien, kita dapat memperoleh tepi dengan menemukan titik-titik tepi dan menghubungkannya. Untuk mengetahui apakah suatu titik merupakan titik tepi, kita harus melihat titik-titik lain pada jarak yang sedikit ke depan dan ke belakang sepanjang arah gradien. Jika besar gradien pada salah satu titik ini lebih besar, maka kita bisa mendapatkan titik tepi yang lebih baik dengan menggeser kurva tepi sedikit saja. Lebih lanjut, jika besar gradien terlalu kecil, titik tersebut tidak dapat menjadi titik tepi. Jadi pada titik tepi, besar gradien adalah maksimum lokal sepanjang arah gradien, dan besar gradien berada di atas ambang batas yang sesuai.

Setelah kita menandai piksel tepi dengan algoritma ini, tahap selanjutnya adalah menghubungkan piksel-piksel yang termasuk dalam kurva tepi yang sama. Hal ini dapat dilakukan dengan mengasumsikan bahwa dua piksel tepi yang bertetangga dengan orientasi yang konsisten harus berada dalam kurva tepi yang sama.

Tekstur

Dalam bahasa sehari-hari, **tekstur** adalah nuansa visual suatu permukaan—apa yang Anda lihat membangkitkan bagaimana permukaan tersebut mungkin terasa jika Anda menyentuhnya ("tekstur" memiliki akar kata yang sama dengan "tekstil"). Dalam visi komputasional, tekstur mengacu pada pola yang berulang secara spasial pada suatu permukaan yang dapat dirasakan secara visual. Contohnya meliputi pola jendela pada bangunan, jahitan pada sweter, bintik-bintik pada macan tutul, helaian rumput di halaman, kerikil di pantai, dan orang-orang di stadion. Terkadang susunannya cukup periodik, seperti pada jahitan pada sweter; dalam kasus lain, seperti kerikil di pantai, keteraturannya hanya bersifat statistik.

Sementara kecerahan adalah properti piksel individual, konsep tekstur hanya masuk akal untuk patch multipiksel. Dengan patch seperti itu, kita dapat menghitung orientasi pada setiap piksel, dan kemudian mengkarakterisasi patch tersebut dengan histogram orientasi. Tekstur batu bata pada dinding akan memiliki dua puncak histogram (satu vertikal dan satu horizontal), sedangkan tekstur bintik-bintik pada kulit macan tutul akan memiliki distribusi orientasi yang lebih seragam.

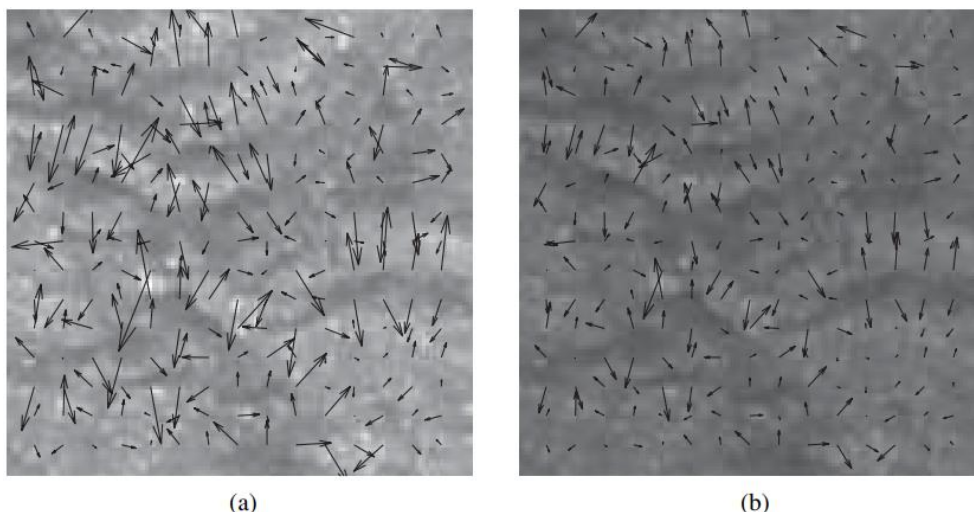
Gambar 6.9 menunjukkan bahwa orientasi sebagian besar invarian terhadap perubahan pencahayaan. Hal ini menjadikan tekstur sebagai petunjuk penting untuk pengenalan objek, karena petunjuk lain, seperti tepi, dapat menghasilkan hasil yang berbeda dalam kondisi pencahayaan yang berbeda.

Pada gambar objek bertekstur, deteksi tepi tidak berfungsi sebaik pada objek halus. Hal ini karena tepi yang paling penting dapat hilang di antara elemen-elemen tekstur. Secara harfiah, kita mungkin melewati harimau karena belang-belangnya. Solusinya adalah mencari perbedaan dalam properti tekstur, sama seperti kita mencari perbedaan kecerahan. Bercak pada harimau dan bercak pada latar belakang berumput akan memiliki histogram orientasi yang sangat berbeda, sehingga memungkinkan kita menemukan kurva batas di antara keduanya.

Aliran optik

Selanjutnya, mari kita pertimbangkan apa yang terjadi ketika kita memiliki rangkaian video, alih-alih hanya satu gambar statis. Ketika sebuah objek dalam video bergerak, atau ketika kamera bergerak relatif terhadap suatu objek, gerakan semu yang dihasilkan dalam gambar disebut **aliran optik**. Aliran optik menggambarkan arah dan kecepatan gerakan fitur-fitur *dalam gambar*—aliran optik dalam video mobil balap diukur dalam piksel per detik, bukan mil per jam. Aliran optik mengkodekan informasi yang berguna tentang struktur pemandangan. Misalnya, dalam video pemandangan yang diambil dari kereta yang sedang melaju, objek yang jauh memiliki gerakan semu

yang lebih lambat daripada objek yang dekat; dengan demikian, laju gerakan semu dapat memberi tahu kita sesuatu tentang jarak. Aliran optik juga memungkinkan kita mengenali tindakan. Pada Gambar 6.10(a) dan (b), kami menunjukkan dua bingkai dari video seorang pemain tenis. Pada (c), kami menampilkan vektor aliran optik yang dihitung dari gambar-gambar ini, yang menunjukkan bahwa raket dan kaki depan bergerak paling cepat.



Gambar 6.9 Dua gambar dengan tekstur kertas nasi kusut yang sama, dengan tingkat iluminasi yang berbeda. Medan vektor gradien (pada setiap piksel kedelapan) diplot di atas masing-masing gambar. Perhatikan bahwa, seiring bertambahnya cahaya, semua vektor gradien menjadi lebih pendek. Vektor-vektor tersebut tidak berotasi, sehingga orientasi gradien tidak berubah.

Medan vektor aliran optik dapat direpresentasikan pada titik mana pun (x, y) oleh komponen-komponennya $v_x(x, y)$ pada arah x dan $v_y(x, y)$ pada arah y . Untuk mengukur aliran optik, kita perlu menemukan titik-titik yang bersesuaian antara satu kerangka waktu dengan kerangka waktu berikutnya. Teknik sederhana ini didasarkan pada fakta bahwa patch gambar di sekitar titik-titik yang bersesuaian memiliki pola intensitas yang serupa. Perhatikan blok piksel yang berpusat pada piksel $p, (x_0, y_0)$, pada waktu t_0 . Blok piksel ini akan dibandingkan dengan blok piksel yang berpusat pada berbagai piksel kandidat pada $(x_0 + D_x, y_0 + D_y)$ pada waktu $t_0 + D_t$. Salah satu ukuran kesamaan yang mungkin adalah **jumlah selisih kuadrat (SSD)**:

$$SSD(D_x, D_y) = \sum_{(x,y)} \left(I(x, y, t) - I(x + D_x, y + D_y, t + D_t) \right)^2$$

Di sini, (x, y) berkisar pada piksel-piksel dalam blok yang berpusat di (x_0, y_0) . Kita menemukan (D_x, D_y) yang meminimalkan SSD. Aliran optik pada (x_0, y_0) adalah

$(v_x, v_y) = (D_x/D_t, D_y/D_t)$. Perhatikan bahwa agar ini berfungsi, perlu ada tekstur atau variasi dalam pemandangan. Jika seseorang melihat dinding putih yang seragam, maka SSD akan hampir sama untuk kecocokan kandidat yang berbeda, dan algoritmanya hanya melakukan tebakan buta. Algoritme dengan kinerja terbaik untuk mengukur aliran optik bergantung pada berbagai kendala tambahan ketika pemandangan hanya bertekstur sebagian.



Gambar 6.10 Dua frame dari rangkaian video. Di sebelah kanan adalah medan aliran optik yang sesuai dengan perpindahan dari satu frame ke frame lainnya. Perhatikan bagaimana pergerakan raket tenis dan kaki depan ditangkap oleh arah panah.

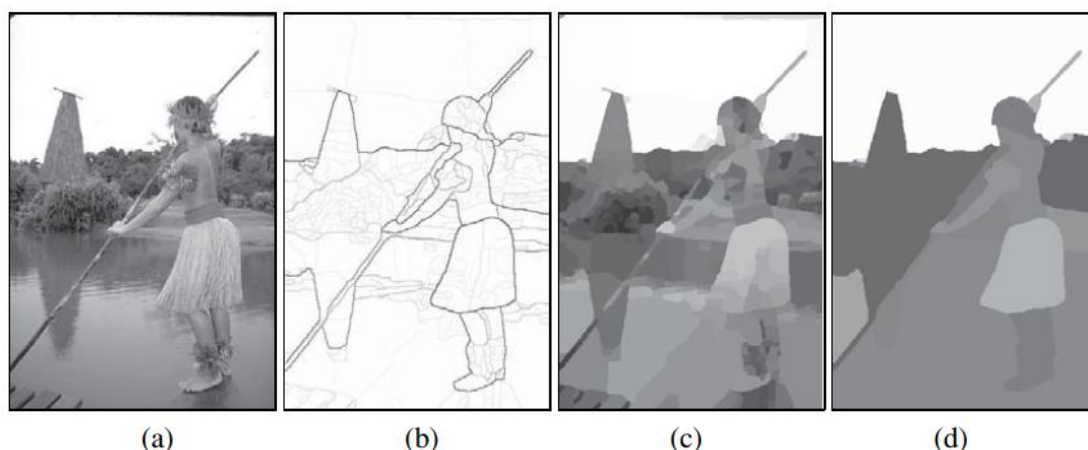
Segmentasi Citra

Segmentasi adalah proses memecah citra menjadi beberapa **wilayah** piksel yang serupa. Setiap piksel citra dapat dikaitkan dengan properti visual tertentu, seperti kecerahan, warna, dan tekstur. Di dalam suatu objek, atau satu bagian dari suatu objek, atribut-atribut ini relatif sedikit bervariasi, sedangkan di sepanjang batas antar-objek biasanya terdapat perubahan besar pada satu atau lebih atribut ini. Ada dua pendekatan untuk segmentasi, satu berfokus pada pendeteksian batas wilayah-wilayah ini, dan yang lainnya pada pendeteksian wilayah itu sendiri (Gambar 6.11).

Kurva batas yang melewati piksel (x, y) akan memiliki orientasi θ , sehingga salah satu cara untuk memformalkan masalah pendeteksian kurva batas adalah sebagai masalah klasifikasi pembelajaran mesin. Berdasarkan fitur dari lingkungan lokal, kami ingin menghitung probabilitas $P_b(x, y, \theta)$ bahwa memang terdapat kurva batas pada piksel tersebut di sepanjang orientasi tersebut. Pertimbangkan sebuah cakram melingkar yang berpusat di (x, y) , dibagi lagi menjadi dua setengah cakram dengan diameter yang berorientasi di θ . Jika ada batas di (x, y, θ) , kedua setengah cakram tersebut mungkin diharapkan berbeda secara signifikan dalam kecerahan, warna, dan teksturnya. Martin, Fowlkes, dan Malik (2004) menggunakan fitur berdasarkan perbedaan histogram nilai kecerahan, warna, dan tekstur yang diukur dalam dua setengah cakram ini, dan kemudian melatih sebuah pengklasifikasi. Untuk ini, mereka

menggunakan kumpulan data citra alami di mana manusia telah menandai batas "kebenaran dasar", dan tujuan pengklasifikasi adalah untuk menandai secara tepat batas-batas yang ditandai oleh manusia dan bukan yang lain.

Batas yang dideteksi oleh teknik ini ternyata jauh lebih baik daripada yang ditemukan menggunakan teknik deteksi tepi sederhana yang dijelaskan sebelumnya. Namun, masih ada dua keterbatasan. (1) Piksel batas yang dibentuk dengan thresholding $P_b(x, y, \theta)$ tidak dijamin membentuk kurva tertutup, sehingga pendekatan ini tidak menghasilkan wilayah, dan (2) pengambilan keputusan hanya memanfaatkan konteks lokal dan tidak menggunakan batasan konsistensi global.



Gambar 6.11 (a) Citra asli. (b) Kontur batas, di mana semakin tinggi nilai P_b , semakin gelap konturnya. (c) Segmentasi ke dalam beberapa wilayah, sesuai dengan partisi citra yang halus. Wilayah-wilayah ditampilkan dalam warna rata-ratanya. (d) Segmentasi ke dalam beberapa wilayah, sesuai dengan partisi citra yang lebih kasar, sehingga menghasilkan lebih sedikit wilayah.

Pendekatan alternatif didasarkan pada upaya "mengelompokkan" piksel ke dalam wilayah berdasarkan kecerahan, warna, dan teksturnya. Shi dan Malik (2000) menetapkan hal ini sebagai masalah partisi graf. Simpul graf berkorespondensi dengan piksel, dan tepi berkorespondensi dengan koneksi antarpiksel. Bobot W_{ij} pada tepi yang menghubungkan sepasang piksel i dan j didasarkan pada seberapa mirip kedua piksel tersebut dalam hal kecerahan, warna, tekstur, dll. Partisi yang meminimalkan kriteria *pemotongan ternormalisasi* kemudian ditemukan. Secara garis besar, kriteria untuk mempartisi graf adalah meminimalkan jumlah bobot koneksi di seluruh kelompok piksel dan memaksimalkan jumlah bobot koneksi di dalam kelompok tersebut.

Segmentasi yang murni berdasarkan atribut lokal tingkat rendah seperti kecerahan dan warna tidak dapat diharapkan menghasilkan batas akhir yang benar dari semua objek dalam pemandangan. Untuk menemukan batas objek secara andal, kita memerlukan pengetahuan tingkat tinggi tentang kemungkinan jenis objek dalam

pemandangan. Merepresentasikan pengetahuan ini merupakan topik penelitian yang sedang berlangsung. Strategi yang populer adalah menghasilkan segmentasi berlebih dari suatu gambar, yang berisi ratusan wilayah homogen yang dikenal sebagai **superpiksel**. Dari sana, algoritma berbasis pengetahuan dapat mengambil alih; mereka akan lebih mudah menangani ratusan superpiksel daripada jutaan piksel mentah. Bagaimana memanfaatkan pengetahuan tingkat tinggi tentang objek akan dibahas di bagian selanjutnya.

6.3 PENGENALAN OBJEK BERDASARKAN PENAMPILAN

Penampilan adalah singkatan untuk bagaimana suatu objek cenderung terlihat. Beberapa kategori objek—misalnya, bola bisbol—memiliki sedikit variasi dalam penampilan; semua objek dalam kategori tersebut terlihat hampir sama dalam sebagian besar keadaan. Dalam hal ini, kita dapat menghitung serangkaian fitur yang menggambarkan setiap kelas gambar yang kemungkinan berisi objek tersebut, lalu mengujinya dengan pengklasifikasi.

Kategori objek lain—misalnya, rumah atau penari balet—sangat bervariasi. Sebuah rumah dapat memiliki ukuran, warna, dan bentuk yang berbeda, serta dapat terlihat berbeda dari berbagai sudut. Seorang penari terlihat berbeda dalam setiap pose, atau ketika lampu panggung berubah warna. Abstraksi yang berguna adalah dengan mengatakan bahwa beberapa objek terdiri dari pola-pola lokal yang cenderung bergerak relatif satu sama lain. Kita kemudian dapat menemukan objek tersebut dengan melihat histogram lokal dari respons detektor, yang mengungkapkan keberadaan suatu bagian tetapi menyembunyikan detail lokasinya.

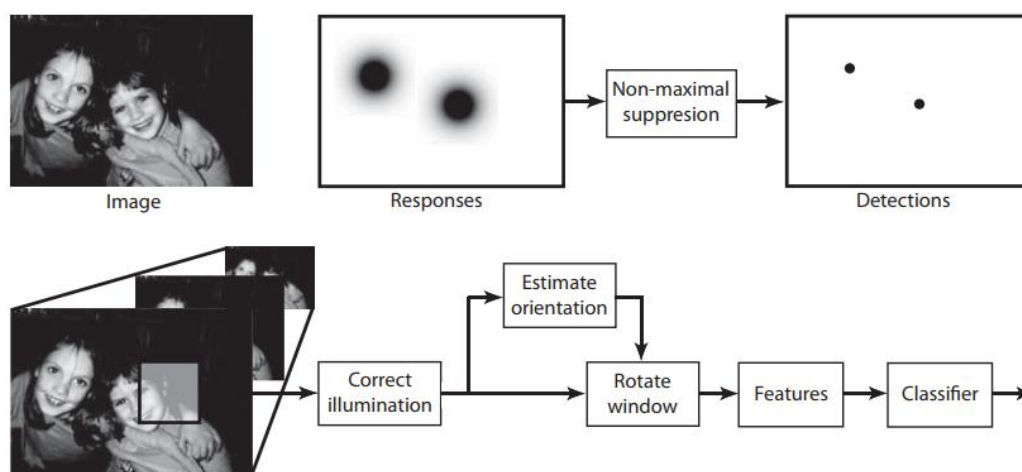
Menguji setiap kelas gambar dengan pengklasifikasi yang dipelajari merupakan resep umum yang penting. Ini bekerja sangat baik untuk wajah yang menatap langsung ke kamera, karena pada resolusi rendah dan di bawah pencahayaan yang wajar, semua wajah tersebut terlihat cukup mirip. Wajahnya bulat, dan cukup terang dibandingkan dengan rongga mata; rongga mata tersebut gelap, karena cekung, dan mulutnya memiliki garis miring gelap, begitu pula alisnya. Perubahan pencahayaan yang signifikan dapat menyebabkan beberapa variasi dalam pola ini, tetapi rentang variasinya cukup mudah dikelola. Hal ini memungkinkan untuk mendeteksi posisi wajah dalam gambar yang berisi wajah. Dahulu merupakan tantangan komputasi, fitur ini kini umum digunakan bahkan di kamera digital murah sekalipun.

Untuk saat ini, kami hanya akan mempertimbangkan wajah dengan hidung yang berorientasi vertikal; kami akan menangani wajah yang diputar di bawah ini. Kami menyapu jendela bundar berukuran tetap di atas gambar, menghitung fitur untuknya, dan menyajikan fitur tersebut ke pengklasifikasi. Strategi ini terkadang disebut **jendela geser**. Fitur harus tangguh terhadap bayangan dan perubahan kecerahan yang disebabkan oleh perubahan iluminasi. Salah satu strateginya adalah membangun fitur dari orientasi gradien. Strategi lainnya adalah memperkirakan dan mengoreksi iluminasi

di setiap jendela gambar. Untuk menemukan wajah dengan ukuran berbeda, ulangi sapuan pada versi gambar yang lebih besar atau lebih kecil. Terakhir, kami melakukan pascaproses respons di berbagai skala dan lokasi untuk menghasilkan rangkaian deteksi akhir. Pascaproses penting, karena kecil kemungkinan kami telah memilih ukuran jendela yang tepat untuk sebuah wajah (meskipun kami menggunakan beberapa ukuran). Dengan demikian, kami kemungkinan akan memiliki beberapa jendela yang tumpang tindih yang masing-masing melaporkan kecocokan untuk sebuah wajah.

Namun, jika kita menggunakan pengklasifikasi yang dapat melaporkan kekuatan respons (misalnya, regresi logistik atau mesin vektor pendukung), kita dapat menggabungkan kecocokan tumpang tindih parsial ini di lokasi terdekat untuk menghasilkan satu kecocokan berkualitas tinggi. Hal ini menghasilkan detektor wajah yang dapat mencari di berbagai lokasi dan skala. Untuk mencari rotasi juga, kita menggunakan dua langkah. Kita melatih prosedur regresi untuk memperkirakan orientasi terbaik dari setiap wajah yang ada di jendela. Sekarang, untuk setiap jendela, kita memperkirakan orientasinya, mengubah orientasi jendela, lalu menguji apakah wajah vertikal ada pada pengklasifikasi kita. Semua ini menghasilkan sistem yang arsitekturnya digambarkan pada Gambar 6.12.

Data pelatihan cukup mudah diperoleh. Ada beberapa set data gambar wajah yang ditandai, dan jendela wajah yang diputar mudah dibuat (cukup putar jendela dari set data pelatihan). Salah satu trik yang banyak digunakan adalah mengambil setiap jendela contoh, lalu menghasilkan contoh baru dengan sedikit mengubah orientasi jendela, pusat jendela, atau skalanya. Ini adalah cara mudah untuk mendapatkan set data yang lebih besar yang mencerminkan gambar nyata dengan cukup baik; trik ini biasanya meningkatkan kinerja secara signifikan. Detektor wajah yang dibuat berdasarkan garis-garis ini sekarang bekerja sangat baik untuk wajah bagian depan (tampilan samping lebih sulit).



Gambar 6.12 Sistem pencarian wajah bervariasi, tetapi sebagian besar mengikuti arsitektur yang diilustrasikan dalam dua bagian di sini. Di bagian atas, kami beralih

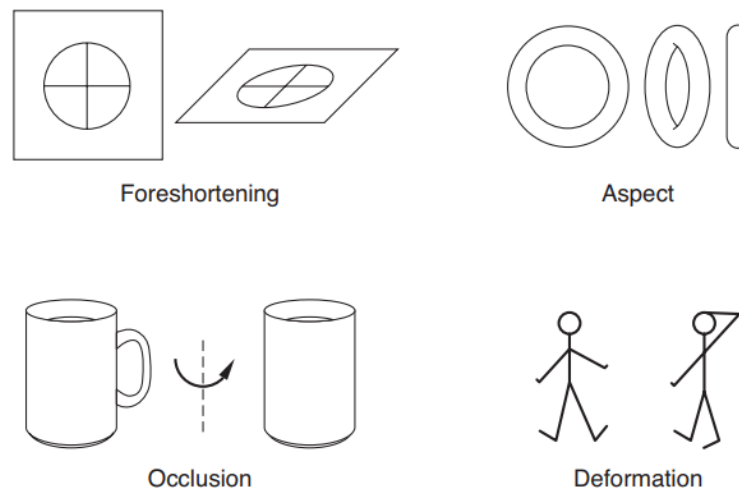
dari gambar ke respons, lalu menerapkan penekanan non-maksimum untuk menemukan respons lokal terkuat. Respons diperoleh melalui proses yang diilustrasikan di bagian bawah. Kami menyapu jendela berukuran tetap pada versi gambar yang lebih besar dan lebih kecil, untuk menemukan wajah yang lebih kecil atau lebih besar. Pencahayaannya di jendela dikoreksi, dan kemudian mesin regresi (cukup sering, jaringan saraf) memprediksi orientasi wajah. Jendela dikoreksi ke orientasi ini dan kemudian disajikan ke pengklasifikasi. Keluaran pengklasifikasi kemudian diproses pasca untuk memastikan bahwa hanya satu wajah yang ditempatkan di setiap lokasi dalam gambar.

Elemen pola dan tampilan kompleks

Banyak objek menghasilkan pola yang jauh lebih kompleks daripada wajah. Hal ini disebabkan oleh beberapa efek yang dapat menggerakkan fitur-fitur pada gambar objek. Efek-efek tersebut meliputi (Gambar 6.13)

- **Foreshortening**, yang menyebabkan pola yang dilihat miring terdistorsi secara signifikan.
- **Aspek**, yang menyebabkan objek tampak berbeda ketika dilihat dari berbagai arah. Bahkan objek sederhana seperti donat pun memiliki beberapa aspek; dilihat dari samping, tampak seperti oval pipih, tetapi dari atas tampak seperti anulus.
- **Oklusi**, di mana beberapa bagian tersembunyi dari beberapa arah pandang. Objek dapat saling menutupi, atau bagian dari suatu objek dapat menutupi bagian lain, efek yang dikenal sebagai oklusi diri.
- **Deformasi**, di mana derajat kebebasan internal suatu objek mengubah penampilannya.

Namun, resep pencarian lintas lokasi dan skala kami masih dapat berhasil. Hal ini karena beberapa struktur akan hadir dalam gambar yang dihasilkan oleh objek tersebut. Misalnya, gambar sebuah mobil kemungkinan akan menampilkan beberapa lampu depan, pintu, roda, jendela, dan dop roda, meskipun pengaturannya mungkin agak berbeda pada gambar yang berbeda. Hal ini menunjukkan pemodelan objek dengan elemen pola—kumpulan komponen. Elemen-elemen pola ini dapat bergerak. Hal ini menyarankan pemodelan objek dengan elemen pola—kumpulan komponen. Elemen-elemen pola ini dapat bergerak relatif satu sama lain, tetapi jika sebagian besar elemen pola berada di tempat yang tepat, maka objek tersebut memang ada. Pengenal objek kemudian merupakan kumpulan fitur yang dapat mengetahui apakah elemen pola tersebut ada, dan apakah mereka berada di tempat yang tepat.

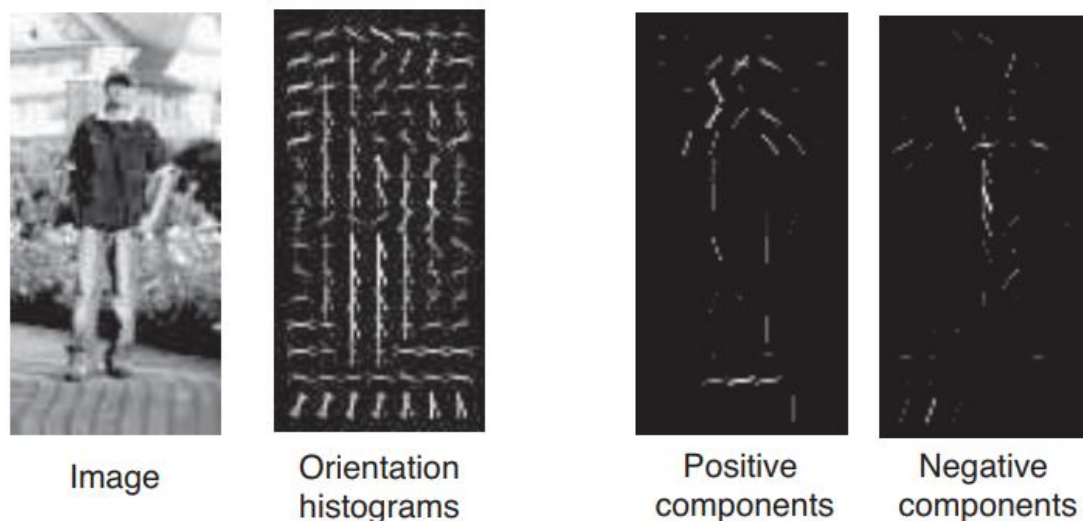


Gambar 6.13 Sumber variasi tampilan. Pertama, elemen dapat memendek, seperti bidang lingkaran di kiri atas. Bidang ini dilihat miring, sehingga berbentuk elips pada gambar. Kedua, objek yang dilihat dari berbagai arah dapat berubah bentuk secara dramatis, sebuah fenomena yang dikenal sebagai aspek. Di kanan atas terdapat tiga aspek berbeda dari donat. Oklusi menyebabkan gagang cangkir di kiri bawah menghilang ketika cangkir diputar. Dalam hal ini, karena badan dan gagangnya berada pada cangkir yang sama, kita mengalami oklusi diri. Terakhir, di kanan bawah, beberapa objek dapat berubah bentuk secara dramatis.

Pendekatan yang paling jelas adalah merepresentasikan jendela gambar dengan histogram elemen pola yang muncul di sana. Pendekatan ini tidak bekerja dengan baik, karena terlalu banyak pola yang saling bertukar. Misalnya, jika elemen pola berupa piksel warna, bendera Prancis, Inggris, dan Belanda akan bertukar karena memiliki histogram warna yang kurang lebih sama, meskipun warnanya tersusun dengan cara yang sangat berbeda. Modifikasi histogram yang cukup sederhana menghasilkan fitur yang sangat berguna. Kuncinya adalah mempertahankan detail spasial dalam representasi; misalnya, lampu depan cenderung berada di depan mobil dan roda cenderung berada di bawah. Fitur berbasis histogram telah berhasil dalam berbagai aplikasi pengenalan; kita akan mensurvei deteksi pejalan kaki.

Deteksi Pejalan Kaki dengan Fitur HOG

Bank Dunia memperkirakan bahwa setiap tahun kecelakaan mobil menewaskan sekitar 1,2 juta orang, yang sekitar dua pertiganya adalah pejalan kaki. Ini berarti bahwa mendeteksi pejalan kaki merupakan masalah aplikasi yang penting, karena mobil yang dapat secara otomatis mendeteksi dan menghindari pejalan kaki dapat menyelamatkan banyak nyawa. Pejalan kaki mengenakan berbagai jenis pakaian dan muncul dalam berbagai konfigurasi, tetapi, pada resolusi yang relatif rendah, pejalan kaki dapat memiliki penampilan yang cukup khas.



Gambar 6.14 Histogram orientasi lokal merupakan fitur yang ampuh untuk mengenali objek yang cukup kompleks sekalipun. Di sebelah kiri, gambar pejalan kaki. Di sebelah kiri tengah, histogram orientasi lokal untuk patch. Kami kemudian menerapkan pengklasifikasi seperti mesin vektor pendukung untuk menemukan bobot bagi setiap histogram yang paling baik memisahkan contoh positif pejalan kaki dari non-pejalan kaki. Kami melihat bahwa komponen berbobot positif tampak seperti garis luar seseorang. Komponen negatif kurang jelas; komponen tersebut mewakili semua pola yang bukan pejalan kaki.

Kasus yang paling umum adalah pandangan lateral atau frontal dari sebuah jalan setapak. Dalam kasus ini, kita melihat bentuk "lolipop" — badan lebih lebar daripada kaki, yang menyatu dalam fase berdiri saat berjalan — atau bentuk "gunting" — di mana kaki berayun saat berjalan. Kita berharap untuk melihat beberapa bukti lengan dan kaki, dan lekukan di sekitar bahu dan kepala juga cenderung terlihat dan cukup khas. Ini berarti bahwa, dengan konstruksi fitur yang cermat, kita dapat membangun detektor pejalan kaki jendela bergerak yang berguna.

Tidak selalu ada kontras yang kuat antara pejalan kaki dan latar belakang, jadi lebih baik menggunakan orientasi daripada tepi untuk merepresentasikan jendela gambar. Pejalan kaki dapat menggerakkan lengan dan kaki mereka, jadi kita harus menggunakan histogram untuk menekan beberapa detail spasial dalam fitur tersebut. Kita memecah jendela menjadi sel-sel, yang dapat tumpang tindih, dan membangun histogram orientasi di setiap sel. Melakukan hal itu akan menghasilkan fitur yang dapat mengetahui apakah kurva kepala dan bahu berada di bagian atas jendela atau di bagian bawah, tetapi tidak akan berubah jika kepala bergerak sedikit.



Gambar 6.15 Contoh lain pengenalan objek, yang satu ini menggunakan fitur SIFT (Scale Invariant Feature Transform), versi sebelumnya dari fitur HOG. Di sebelah **kiri**, gambar sepatu dan telepon yang berfungsi sebagai model objek. Di **tengah**, gambar uji. Di sebelah **kanan**, sepatu dan telepon telah dideteksi dengan: menemukan titik-titik pada gambar yang deskripsi fitur SIFT-nya cocok dengan suatu model; menghitung estimasi pose model; dan memverifikasi estimasi tersebut. Kecocokan yang kuat biasanya diverifikasi dengan positif palsu yang jarang terjadi.

Satu trik lebih lanjut diperlukan untuk membuat fitur yang baik. Karena fitur orientasi tidak terpengaruh oleh kecerahan iluminasi, kita tidak dapat memperlakukan tepi kontras tinggi secara khusus. Ini berarti bahwa kurva khas pada batas pejalan kaki diperlakukan dengan cara yang sama seperti detail tekstur halus pada pakaian atau di latar belakang, sehingga sinyal mungkin tenggelam dalam noise. Kita dapat memulihkan informasi kontras dengan menghitung orientasi gradien dengan bobot yang mencerminkan seberapa signifikan suatu gradien dibandingkan dengan gradien lain dalam sel yang sama. Kita akan menulis $\|\nabla I_x\|$ Untuk besaran gradien pada titik x pada gambar, tulis C untuk sel yang histogramnya ingin kita hitung, dan tulis $w_{x,C}$ untuk bobot yang akan kita gunakan untuk orientasi pada titik x untuk sel ini. Pilihan bobot yang wajar adalah

$$w_{x,C} = \frac{\|\nabla I_x\|}{\sum_{u \in C} \|\nabla I_u\|}$$

Ini membandingkan besaran gradien dengan yang lain dalam sel, sehingga gradien yang besar dibandingkan dengan gradien tetangganya mendapatkan bobot yang besar. Fitur yang dihasilkan biasanya disebut **fitur HOG** (Histogram of Gradient orientations).

Konstruksi fitur ini merupakan perbedaan utama antara deteksi pejalan kaki dan deteksi wajah. Jika tidak, membangun detektor pejalan kaki sangat mirip dengan membangun detektor wajah. Detektor memindai jendela pada gambar, menghitung fitur untuk jendela tersebut, lalu menyajikannya ke pengklasifikasi. Penekanan non-maksimum perlu diterapkan pada keluaran. Dalam sebagian besar aplikasi, skala dan orientasi pejalan kaki pada umumnya diketahui. Misalnya, dalam aplikasi berkendara dengan kamera terpasang pada mobil, kita berharap untuk melihat pejalan kaki vertikal,

dan kita hanya tertarik pada pejalan kaki di dekatnya. Beberapa set data pejalan kaki telah dipublikasikan, dan ini dapat digunakan untuk melatih pengklasifikasi.

Pejalan kaki bukanlah satu-satunya jenis objek yang dapat kita deteksi. Pada Gambar 6.15, kita melihat bahwa teknik serupa dapat digunakan untuk menemukan berbagai objek dalam konteks yang berbeda.

6.4 MEREKONSTRUKSI DUNIA 3D

Di bagian ini, kami menunjukkan cara beralih dari citra dua dimensi ke representasi tiga dimensi dari suatu pemandangan. Pertanyaan mendasarnya adalah: Mengingat semua titik dalam pemandangan yang jatuh sepanjang sinar menuju lubang jarum diproyeksikan ke titik yang sama dalam citra, bagaimana kita memulihkan informasi tiga dimensi? Dua ide berikut dapat membantu kita:

- Jika kita memiliki dua (atau lebih) gambar dari posisi kamera yang berbeda, maka kita dapat melakukan triangulasi untuk menemukan posisi suatu titik dalam pemandangan.
- Kita dapat memanfaatkan pengetahuan latar belakang tentang pemandangan fisik yang menghasilkan gambar tersebut. Dengan model objek $P(Scene)$ dan model rendering $P(Image|Scene)$ kita dapat menghitung distribusi posterior $P(Scene|Image)$.

Belum ada teori tunggal yang terpadu untuk rekonstruksi pemandangan. Kami mensurvei delapan isyarat visual yang umum digunakan: gerakan, stereopsis binokular, beberapa tampilan, tekstur, bayangan, kontur, dan objek yang familiar.

Paralaks Gerak

Jika kamera bergerak relatif terhadap pemandangan tiga dimensi, gerakan nyata yang dihasilkan dalam gambar, aliran optik, dapat menjadi sumber informasi untuk pergerakan kamera dan kedalaman dalam pemandangan. Untuk memahami hal ini, kami menyatakan (tanpa bukti) sebuah persamaan yang menghubungkan aliran optik dengan kecepatan translasi pemirsa T dan kedalaman dalam pemandangan.

Komponen medan aliran optik adalah

$$u_x(x, y) = \frac{-T_x + xT_z}{Z(x, y)}, \quad u_y(x, y) = \frac{-T_y + xyT_z}{Z(x, y)},$$

di mana $Z(x, y)$ adalah koordinat- z dari titik dalam pemandangan yang bersesuaian dengan titik dalam gambar di (x, y) .

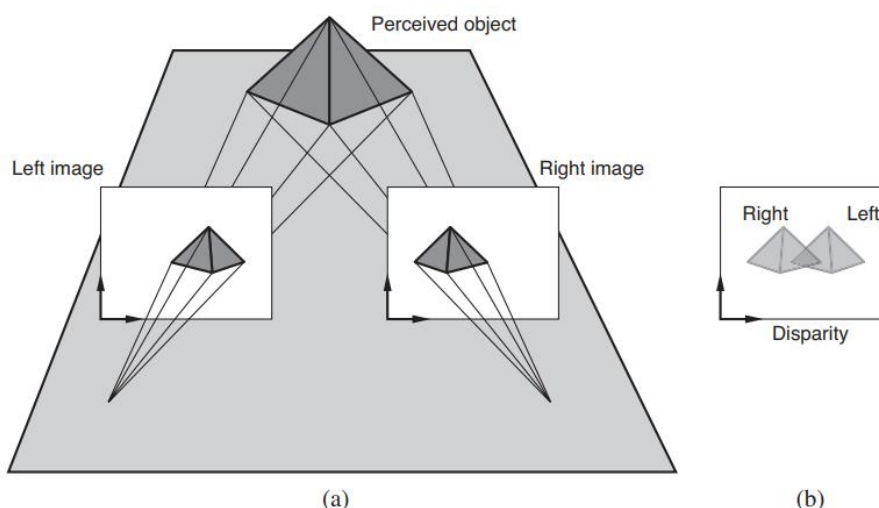
Perhatikan bahwa kedua komponen aliran optik, $u_x(x, y)$ dan $u_y(x, y)$, bernilai nol pada titik $x = T_x/T_z, y = T_y/T_z$. Titik ini disebut **fokus ekspansi** medan aliran. Misalkan kita mengubah titik asal pada bidang $x - y$ agar berada di fokus ekspansi; maka ekspresi untuk aliran optik akan menjadi bentuk yang sangat sederhana. Misalkan

(x', y') adalah koordinat baru yang didefinisikan oleh $x' = x - T_x/T_z, y' = y - T_y/T_z$. Maka

$$u_x(x', y') = \frac{x' T_z}{Z(x', y')}, \quad u_y(x', y') = \frac{y' T_z}{Z(x', y')}$$

Perhatikan bahwa terdapat ambiguitas faktor skala di sini. Jika kamera bergerak dua kali lebih cepat, dan setiap objek dalam pemandangan berukuran dua kali lebih besar dan berjarak dua kali lipat dari kamera, medan aliran optik akan sama persis. Namun, kita masih dapat mengekstrak informasi yang cukup berguna.

1. Misalkan Anda adalah seekor lalat yang mencoba mendarat di dinding dan ingin mengetahui waktu kontak pada kecepatan saat ini. Waktu ini diberikan oleh Z/T_z . Perhatikan bahwa meskipun medan aliran optik sesaat tidak dapat memberikan jarak Z maupun komponen kecepatan T_z , medan tersebut dapat memberikan rasio keduanya dan oleh karena itu dapat digunakan untuk mengontrol pendekatan pendaratan. Terdapat banyak bukti eksperimental bahwa banyak spesies hewan yang berbeda memanfaatkan isyarat ini.
2. Perhatikan dua titik pada kedalaman Z_1 dan Z_2 . Kita mungkin tidak mengetahui nilai absolut dari keduanya, tetapi dengan mempertimbangkan kebalikan dari rasio besaran aliran optik pada titik-titik ini, kita dapat menentukan rasio kedalaman Z_1/Z_2 . Inilah isyarat paralaks gerak, yang kita gunakan saat kita melihat ke luar jendela samping mobil atau kereta yang melaju dan menyimpulkan bahwa bagian lanskap yang bergerak lebih lambat berada lebih jauh.



Gambar 6.16. Mentranslasikan kamera sejajar dengan bidang gambar menyebabkan fitur gambar bergerak di bidang kamera. Perbedaan posisi yang dihasilkan merupakan petunjuk kedalaman. Jika kita menumpangkan gambar kiri dan kanan, seperti pada (b), kita melihat perbedaannya.

Stereopsis binokular

Kebanyakan vertebrata memiliki dua mata. Hal ini berguna untuk redundansi jika terjadi kehilangan mata, tetapi juga membantu dalam hal lain. Kebanyakan mangsa memiliki mata di sisi kepala untuk memungkinkan bidang pandang yang lebih luas. Predator memiliki mata di depan, yang memungkinkan mereka menggunakan **stereopsis binokular**. Idenya mirip dengan paralaks gerak, hanya saja alih-alih menggunakan gambar dari waktu ke waktu, kita menggunakan dua (atau lebih) gambar yang terpisah dalam ruang. Karena fitur tertentu dalam pemandangan akan berada di tempat yang berbeda relatif terhadap sumbu z setiap bidang gambar, jika kita menumpangkan kedua gambar tersebut, akan ada **perbedaan** lokasi fitur gambar di kedua gambar tersebut. Anda dapat melihat hal ini pada Gambar 6.16, di mana titik terdekat piramida bergeser ke kiri pada gambar kanan dan ke kanan pada gambar kiri.

Perhatikan bahwa untuk mengukur disparitas, kita perlu memecahkan masalah korespondensi, yaitu menentukan titik pada gambar kiri, titik pada gambar kanan yang dihasilkan dari proyeksi titik pemandangan yang sama. Hal ini analog dengan apa yang harus dilakukan dalam mengukur aliran optik, dan pendekatan yang paling sederhana pun agak mirip dan didasarkan pada perbandingan blok piksel di sekitar titik yang sesuai menggunakan jumlah selisih kuadrat. Dalam praktiknya, kita menggunakan algoritma yang jauh lebih canggih, yang memanfaatkan kendala tambahan. Dengan asumsi bahwa kita dapat mengukur disparitas, bagaimana hal ini menghasilkan informasi tentang kedalaman dalam pemandangan? Kita perlu mencari tahu hubungan geometris antara disparitas dan kedalaman. Pertama, kita akan mempertimbangkan kasus ketika kedua mata (atau kamera) melihat ke depan dengan sumbu optiknya sejajar. Hubungan antara kamera kanan dan kamera kiri hanyalah perpindahan sepanjang sumbu x dengan jumlah b , garis dasar. Kita dapat menggunakan persamaan aliran optik dari bagian sebelumnya, jika kita menganggap ini dihasilkan dari vektor translasi T yang bekerja selama waktu δt , dengan $T_x = b/\delta t$ dan $T_y = T_z = 0$. Disparitas horizontal dan vertikal diberikan oleh komponen aliran optik, dikalikan dengan langkah waktu δt , $H = v_x \delta t, V = v_y \delta t$. Dengan melakukan substitusi, kita mendapatkan hasil bahwa $H = b/Z, V = 0$. Dengan kata lain, disparitas horizontal sama dengan rasio garis dasar terhadap kedalaman, dan disparitas vertikal adalah nol. Dengan mengetahui b , kita dapat mengukur H dan memperoleh kedalaman Z .

Dalam kondisi penglihatan normal, manusia melakukan **fiksasi**; yaitu, ada beberapa titik dalam pemandangan di mana sumbu optik kedua mata berpotongan. Gambar 6.17 menunjukkan dua mata yang difiksasi pada titik P_0 , yang berjarak Z dari titik tengah mata. Untuk kemudahan, kita akan menghitung disparitas sudut, diukur dalam radian. Disparitas pada titik fiksasi P_0 adalah nol. Untuk beberapa titik P lainnya dalam pemandangan yang berjarak δZ lebih jauh, kita dapat menghitung perpindahan sudut dari bayangan kiri dan kanan P , yang masing-masing akan kita sebut P_L dan P_R .

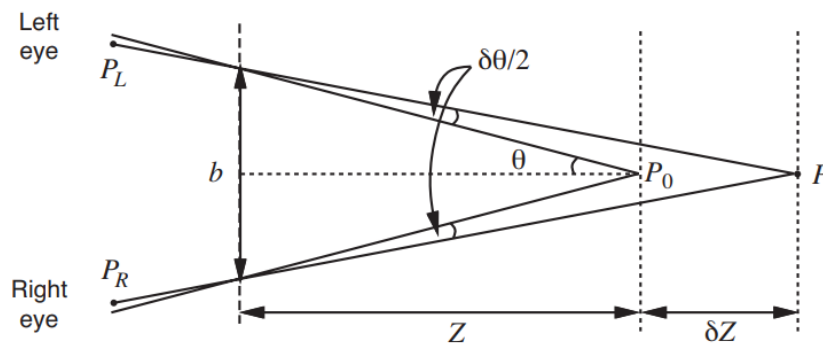
Jika masing-masing dipindahkan dengan sudut $\delta\theta/2$ relatif terhadap P_0 , maka perpindahan antara P_L dan P_R , yang merupakan disparitas P , adalah hanya $\delta\theta$. Dari Gambar 6.17, $\tan \theta = \frac{b/2}{Z+\delta Z}$ dan $\tan(\theta - \delta\theta/2) = \frac{b/2}{Z}$, tetapi untuk sudut kecil, $\tan \theta \approx \theta$, jadi

$$\delta\theta/2 = \frac{b/2}{Z} - \frac{b/2}{Z + \delta Z} \approx \frac{b\delta Z}{2Z^2}$$

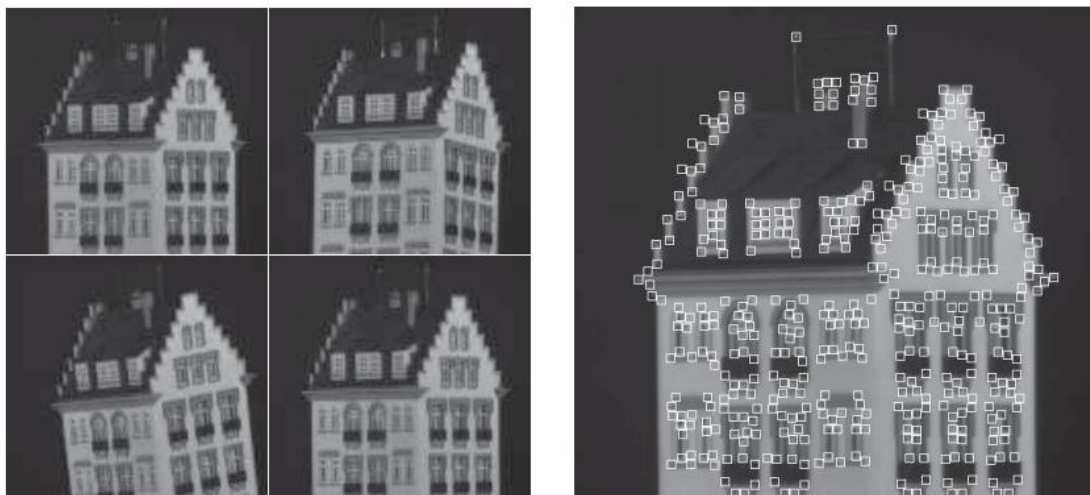
dan, karena perbedaan sebenarnya adalah $\delta\theta$, kita memiliki

$$\text{disparity} = \frac{b\delta Z}{Z^2}$$

Pada manusia, b (**garis dasar** antara kedua mata) sekitar 6 cm. Misalkan Z sekitar 100 cm. Jika $\delta\theta$ terkecil yang terdeteksi (sesuai dengan ukuran piksel) sekitar 5 detik busur, ini menghasilkan δZ sebesar 0,4 mm. Untuk $Z = 30$ cm, kita mendapatkan nilai yang sangat kecil, $\delta Z = 0,036$ mm. Artinya, pada jarak 30 cm, manusia dapat membedakan kedalaman yang berbeda sekecil 0,036 mm, sehingga memungkinkan kita untuk memasukkan benang ke dalam jarum dan sebagainya.



Gambar 6.17 Hubungan antara disparitas dan kedalaman dalam stereopsis. Pusat proyeksi kedua mata berjarak b , dan sumbu optik berpotongan di titik fiksasi P_0 . Titik P dalam pemandangan diproyeksikan ke titik P_L dan P_R di kedua mata. Dalam istilah sudut, disparitas antara keduanya adalah $\delta\theta$. Lihat teks.



Gambar 6.18 (a) Empat frame dari rangkaian video di mana kamera digerakkan dan diputar relatif terhadap objek. (b) Frame pertama dari rangkaian tersebut, diberi anotasi kotak-kotak kecil yang menyorot fitur-fitur yang ditemukan oleh detektor fitur.

Tampilan Ganda

Bentuk dari aliran optik atau disparitas binokular adalah dua contoh kerangka kerja yang lebih umum, yaitu pemanfaatan tampilan ganda untuk memulihkan kedalaman. Dalam visi komputer, kita tidak perlu dibatasi pada gerak diferensial atau hanya menggunakan dua kamera yang konvergen pada titik fiksasi. Oleh karena itu, telah dikembangkan teknik yang memanfaatkan informasi yang tersedia dalam tampilan ganda, bahkan dari ratusan atau ribuan kamera. Secara algoritmik, terdapat tiga submasalah yang perlu dipecahkan:

- Masalah korespondensi, yaitu mengidentifikasi fitur-fitur dalam citra yang berbeda yang merupakan proyeksi dari fitur yang sama di dunia tiga dimensi.
- Masalah orientasi relatif, yaitu menentukan transformasi (rotasi dan translasi) antara sistem koordinat yang ditetapkan pada kamera yang berbeda.
- Masalah estimasi kedalaman, yaitu menentukan kedalaman berbagai titik di dunia yang proyeksi bidang gambarnya tersedia dalam setidaknya dua tampilan.

Pengembangan prosedur pencocokan yang robust untuk masalah korespondensi, disertai dengan algoritma yang stabil secara numerik untuk menyelesaikan orientasi relatif dan kedalaman pemandangan, merupakan salah satu kisah sukses visi komputer. Hasil dari salah satu pendekatan tersebut, yang dikembangkan oleh Tomasi dan Kanade (1992), ditunjukkan pada Gambar 6.18 dan 24.19.

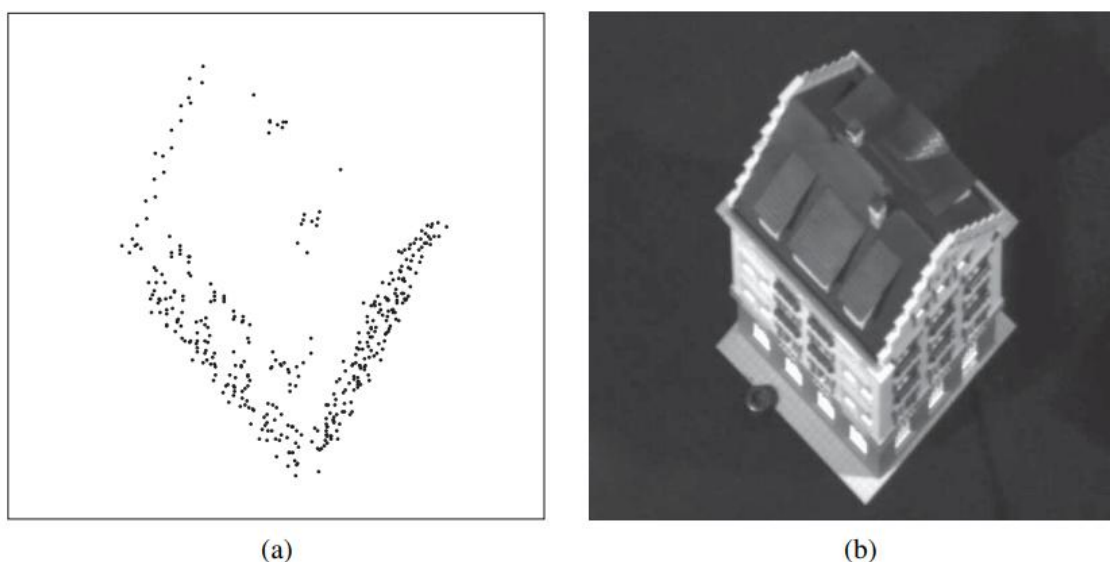
Tekstur

Sebelumnya kita telah melihat bagaimana tekstur digunakan untuk segmentasi objek. Tekstur juga dapat digunakan untuk memperkirakan jarak. Pada Gambar 6.20, kita melihat bahwa tekstur yang homogen dalam pemandangan menghasilkan elemen

tekstur, atau **texel**, yang bervariasi dalam gambar. Semua ubin paving pada (a) identik dalam pemandangan. Mereka tampak berbeda dalam gambar karena dua alasan:

1. *Perbedaan jarak texel dari kamera*. Objek yang jauh tampak lebih kecil dengan faktor skala $1/Z$.
2. *Perbedaan pemendekan texel*. Jika semua texel berada di bidang dasar, maka texel yang jauh dilihat pada sudut yang lebih jauh dari garis tegak lurus, sehingga lebih terpemendekan. Besarnya efek pemendekan sebanding dengan $\cos \sigma$, di mana σ adalah kemiringan, sudut antara sumbu Z dan $\mathbf{n}$, permukaan normal terhadap texel.

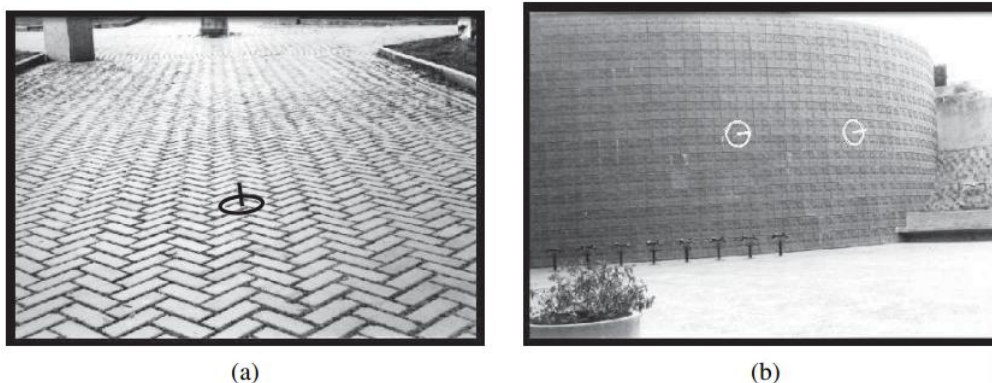
Para peneliti telah mengembangkan berbagai algoritma yang mencoba memanfaatkan variasi tampilan texel yang diproyeksikan sebagai dasar untuk menentukan normal permukaan. Namun, akurasi dan penerapan algoritma ini tidak seumum algoritma yang didasarkan pada penggunaan beberapa tampilan.



Gambar 6.19 (a) Rekonstruksi tiga dimensi lokasi fitur gambar pada Gambar 6.18, ditunjukkan dari atas. (b) Rumah sebenarnya, diambil dari posisi yang sama.

Bayangan

Bayangan—variasi intensitas cahaya yang diterima dari berbagai bagian permukaan dalam suatu pemandangan—ditentukan oleh geometri pemandangan dan sifat reflektansi permukaan tersebut. Dalam grafika komputer, tujuannya adalah menghitung kecerahan gambar $I(x, y)$, berdasarkan geometri pemandangan dan sifat reflektansi objek dalam pemandangan tersebut. Visi komputer bertujuan untuk membalikkan proses tersebut—yaitu, memulihkan geometri dan sifat reflektansi, berdasarkan kecerahan gambar $I(x, y)$. Hal ini terbukti sulit dilakukan, kecuali dalam kasus yang paling sederhana.



Gambar 6.20 (a) Pemandangan bertekstur. Dengan asumsi tekstur sebenarnya seragam, pemulihan orientasi permukaan dimungkinkan. Orientasi permukaan yang dihitung ditunjukkan dengan melapisi lingkaran hitam dan penunjuk, yang ditransformasikan seolah-olah lingkaran tersebut telah dilukis di permukaan pada titik tersebut. (b) Pemulihan bentuk dari tekstur untuk permukaan lengkung (kali ini lingkaran putih dan penunjuk).

Dari model fisik di bagian sebelumnya, kita mengetahui bahwa jika garis normal permukaan mengarah ke sumber cahaya, permukaannya lebih terang, dan jika menjauh, permukaannya lebih gelap. Kita tidak dapat menyimpulkan bahwa suatu bercak gelap memiliki garis normalnya yang menjauhi cahaya; sebaliknya, bercak tersebut mungkin memiliki albedo yang rendah. Umumnya, albedo berubah cukup cepat pada gambar, dan bayangan berubah agak lambat, dan manusia tampaknya cukup mahir menggunakan pengamatan ini untuk menentukan apakah iluminasi rendah, orientasi permukaan, atau albedo menyebabkan suatu bercak permukaan menjadi gelap. Untuk menyederhanakan masalah ini, mari kita asumsikan bahwa albedo diketahui di setiap titik permukaan. Masih sulit untuk memulihkan nilai normal, karena kecerahan gambar merupakan satu pengukuran, tetapi nilai normal memiliki dua parameter yang tidak diketahui, sehingga kita tidak bisa begitu saja menyelesaikan nilai normal. Kunci dari situasi ini tampaknya adalah bahwa nilai normal di dekatnya akan serupa, karena sebagian besar permukaan halus—tidak mengalami perubahan yang tajam.

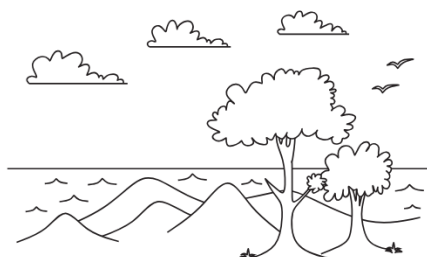
Kesulitan sebenarnya muncul dalam menangani interrefleksi. Jika kita mempertimbangkan pemandangan dalam ruangan yang umum, seperti objek di dalam kantor, permukaan tidak hanya diterangi oleh sumber cahaya, tetapi juga oleh cahaya yang dipantulkan dari permukaan lain dalam pemandangan yang secara efektif berfungsi sebagai sumber cahaya sekunder. Efek iluminasi timbal balik ini cukup signifikan dan membuatnya cukup sulit untuk memprediksi hubungan antara nilai normal dan kecerahan gambar. Dua bidang permukaan dengan normal yang sama mungkin memiliki kecerahan yang sangat berbeda, karena yang satu menerima pantulan cahaya dari dinding putih besar sementara yang lainnya hanya menghadap rak buku gelap. Terlepas dari kesulitan-kesulitan ini, permasalahan ini penting. Manusia tampaknya mampu

mengabaikan efek interrefleksi dan mendapatkan persepsi bentuk yang bermanfaat dari bayangan, tetapi sayangnya, kita hanya tahu sedikit tentang algoritma untuk melakukan hal ini.

Kontur

Ketika kita melihat gambar garis, seperti Gambar 6.21, kita mendapatkan persepsi yang jelas tentang bentuk dan tata letak tiga dimensi. Bagaimana? Ini merupakan kombinasi pengenalan objek-objek yang familiar dalam pemandangan dan penerapan batasan umum seperti berikut:

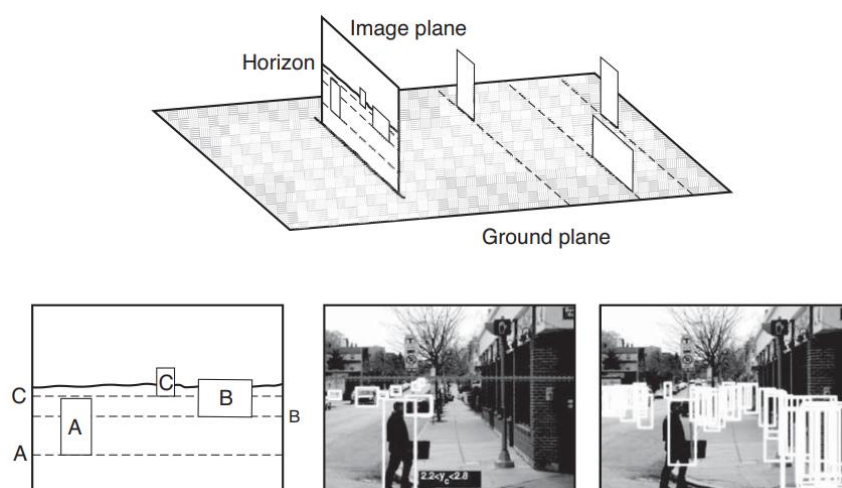
- Kontur yang menutupi, seperti garis luar perbukitan. Satu sisi kontur lebih dekat ke pengamat, sisi lainnya lebih jauh. Fitur-fitur seperti konveksitas lokal dan simetri memberikan petunjuk untuk memecahkan masalah **figur-latar** belakang—menentukan sisi kontur mana yang merupakan figur (lebih dekat), dan mana yang merupakan latar belakang (lebih jauh). Pada kontur yang menutupi, garis pandang bersinggungan dengan permukaan dalam pemandangan.
- Persimpangan-T. Ketika satu objek menutupi objek lain, kontur objek yang lebih jauh terputus, dengan asumsi bahwa objek yang lebih dekat bersifat buram. Persimpangan-T menghasilkan gambar.
- Posisi pada bidang latar belakang. Manusia, seperti banyak hewan terestrial lainnya, sangat sering berada dalam sebuah pemandangan yang berisi **bidang tanah**, dengan berbagai objek di lokasi yang berbeda pada bidang ini. Karena gravitasi, objek-objek umum tidak melayang di udara tetapi didukung oleh bidang tanah ini, dan kita dapat memanfaatkan geometri yang sangat khusus dari skenario tampilan ini. Mari kita kerjakan proyeksi objek-objek dengan ketinggian yang berbeda dan pada lokasi yang berbeda pada bidang tanah. Misalkan mata, atau kamera, berada pada ketinggian h_c di atas bidang tanah. Pertimbangkan sebuah objek dengan tinggi δY yang diam di bidang tanah, yang bagian bawahnya berada di $(X, -h_c, Z)$ dan bagian atasnya berada di $(X, \delta Y - h_c, Z)$. Bagian bawahnya memproyeksikan ke titik bayangan $(f X/Z, -f h_c/Z)$ dan bagian atasnya ke $(f X/Z, f (\delta Y - h_c)/Z)$. Bagian bawah objek yang lebih dekat (Z kecil) memproyeksikan ke titik-titik yang lebih rendah pada bidang bayangan; objek yang lebih jauh memiliki bagian bawahnya lebih dekat ke cakrawala.



Gambar 6.21 Gambar garis yang menggugah.

Objek dan Struktur Geometris Pemandangan

Kepala manusia dewasa pada umumnya memiliki panjang sekitar 9 inci. Ini berarti bahwa untuk seseorang yang berjarak 43 kaki (12,4 meter), sudut yang dibentuk oleh kepala ke kamera adalah 1 derajat. Jika kita melihat seseorang yang kepalanya tampak hanya dibentuk setengah derajat (18,4 meter), inferensi Bayesian menunjukkan bahwa kita sedang melihat orang normal yang berjarak 86 kaki (26,4 meter), alih-alih seseorang dengan ukuran kepala setengah. Alur penalaran ini memberi kita metode untuk memeriksa hasil detektor pejalan kaki, serta metode untuk memperkirakan jarak ke suatu objek. Misalnya, semua pejalan kaki memiliki tinggi yang kurang lebih sama, dan mereka cenderung berdiri di bidang tanah. Jika kita mengetahui letak cakrawala dalam sebuah gambar, kita dapat mengurutkan pejalan kaki berdasarkan jarak ke kamera. Hal ini berhasil karena kita mengetahui letak kaki mereka, dan pejalan kaki yang kakinya lebih dekat ke cakrawala dalam gambar berada lebih jauh dari kamera (Gambar 6.22). Pejalan kaki yang berada lebih jauh dari kamera juga harus berukuran lebih kecil dalam gambar. Ini berarti kita dapat mengesampingkan beberapa respons detektor — jika detektor menemukan pejalan kaki yang besar dalam gambar dan kakinya dekat dengan cakrawala, berarti detektor telah menemukan pejalan kaki yang sangat besar; pejalan kaki seperti ini tidak ada, jadi detektor salah. Faktanya, banyak atau sebagian besar jendela gambar bukanlah jendela pejalan kaki yang dapat diterima, dan bahkan tidak perlu ditampilkan kepada detektor.



Gambar 6.22 Pada gambar orang yang berdiri di bidang tanah, orang yang kakinya lebih dekat ke cakrawala pada gambar harus lebih jauh (gambar atas). Ini berarti mereka harus terlihat lebih kecil pada gambar (gambar kiri bawah). Ini berarti bahwa ukuran dan lokasi pejalan kaki nyata pada gambar bergantung satu sama lain dan pada lokasi cakrawala. Untuk memanfaatkan ini, kita perlu mengidentifikasi bidang tanah, yang dilakukan dengan menggunakan metode bentuk-dari-tekstur. Dari informasi ini, dan dari beberapa pejalan kaki yang mungkin, kita dapat memulihkan

cakrawala seperti yang ditunjukkan pada gambar tengah. Di sebelah kanan, kotak pejalan kaki yang dapat diterima mengingat konteks geometris ini. Perhatikan bahwa pejalan kaki yang lebih tinggi dalam pemandangan harus lebih kecil. Jika tidak, maka mereka adalah positif palsu.

Ada beberapa strategi untuk menemukan cakrawala, termasuk mencari garis horizontal kasar dengan banyak warna biru di atasnya, dan menggunakan estimasi orientasi permukaan yang diperoleh dari deformasi tekstur. Strategi yang lebih elegan memanfaatkan kebalikan dari batasan geometris kita. Detektor pejalan kaki yang cukup andal mampu menghasilkan estimasi cakrawala, jika terdapat beberapa pejalan kaki dalam pemandangan pada jarak yang berbeda dari kamera. Hal ini karena skala relatif pejalan kaki merupakan petunjuk di mana cakrawala berada. Jadi, kita dapat mengekstrak estimasi cakrawala dari detektor, lalu menggunakan estimasi ini untuk memangkas kesalahan detektor pejalan kaki.

Jika objek tersebut familier, kita dapat memperkirakan lebih dari sekadar jaraknya, karena tampilannya dalam gambar sangat bergantung pada posenya, yaitu posisi dan orientasinya terhadap pengamat. Hal ini memiliki banyak aplikasi. Misalnya, dalam tugas manipulasi industri, lengan robot tidak dapat mengambil objek hingga posenya diketahui. Dalam kasus objek kaku, baik tiga dimensi maupun dua dimensi, masalah ini memiliki solusi yang sederhana dan terdefinisi dengan baik berdasarkan **metode penyelarasan**, yang sekarang sedang kita kembangkan.

Objek tersebut direpresentasikan oleh M fitur atau titik-titik khusus $m_1, m_2, \dots, m_M$ dalam ruang tiga dimensi—mungkin titik-titik dari objek polihedral. Ini diukur dalam beberapa sistem koordinat yang alami untuk objek tersebut. Titik-titik tersebut kemudian mengalami rotasi tiga dimensi $\mathbf{R}$ yang tidak diketahui, diikuti oleh translasi dengan jumlah $\mathbf{t}$ yang tidak diketahui dan kemudian proyeksi untuk menghasilkan titik-titik fitur gambar $p_1, p_2, \dots, p_N$ pada bidang gambar. Secara umum, $N \neq M$, karena beberapa titik model mungkin terhalang, dan detektor fitur mungkin melewatkan beberapa fitur (atau menciptakan fitur palsu karena noise). Kita dapat menyatakan hal ini sebagai

$$p_i = \Pi(\mathbf{R}m_i + \mathbf{t}) = Q(m_i)$$

untuk titik model tiga dimensi m_i dan titik bayangan p_i yang bersesuaian. Di sini, $\mathbf{R}$ adalah matriks rotasi, $\mathbf{t}$ adalah translasi, dan Π menunjukkan proyeksi perspektif atau salah satu aproksimasinya, seperti proyeksi ortografi berskala. Hasil akhirnya adalah transformasi Q yang akan menyelaraskan titik model m_i dengan titik bayangan p_i . Meskipun kita tidak mengetahui Q pada awalnya, kita tahu (untuk objek kaku) bahwa Q harus *sama* untuk semua titik model.

Kita dapat menyelesaikan Q , dengan koordinat tiga dimensi dari tiga titik model dan proyeksi dua dimensinya. Intuisinya adalah sebagai berikut: kita dapat menuliskan persamaan yang menghubungkan koordinat p_i dengan koordinat m_i . Dalam persamaan ini, besaran yang tidak diketahui berkorespondensi dengan parameter matriks rotasi $\mathbf{R}$ dan vektor translasi $\mathbf{t}$. Jika kita memiliki cukup persamaan, kita seharusnya dapat menyelesaikan $\mathbf{Q}$. Kita tidak akan memberikan bukti di sini; Kami hanya menyatakan hasil berikut:

Diberikan tiga titik nonkolinear m_1, m_2 , dan m_3 dalam model, dan proyeksi ortografik berskala p_1, p_2 , dan p_3 pada bidang gambar, terdapat tepat dua transformasi dari kerangka koordinat model tiga dimensi ke kerangka koordinat gambar dua dimensi.

Transformasi ini terkait oleh refleksi di sekitar bidang gambar dan dapat dihitung dengan solusi bentuk tertutup sederhana. Jika kita dapat mengidentifikasi fitur model yang sesuai untuk tiga fitur dalam gambar, kita dapat menghitung Q , pose objek.

Mari kita tentukan posisi dan orientasi dalam istilah matematika. Posisi titik P dalam pemandangan dicirikan oleh tiga angka, koordinat (X, Y, Z) P dalam kerangka koordinat dengan titik asal di lubang jarum dan sumbu Z sepanjang sumbu optik (Gambar 6.2) Yang tersedia adalah proyeksi perspektif (x, y) dari titik dalam gambar. Ini menentukan sinar dari lubang jarum di sepanjang P berada; Yang tidak kita ketahui adalah jaraknya. Istilah "orientasi" dapat digunakan dalam dua pengertian:

1. **Orientasi objek secara keseluruhan.** Hal ini dapat ditentukan dalam bentuk rotasi tiga dimensi yang menghubungkan bingkai koordinatnya dengan kamera.
2. **Orientasi permukaan objek pada titik P .** Ini dapat ditentukan oleh vektor normal, $\mathbf{n}$ —yang merupakan vektor yang menentukan arah tegak lurus terhadap permukaan. Seringkali kita menyatakan orientasi permukaan menggunakan variabel **kemiringan** dan **memiringkan**. Kemiringan adalah sudut KEMIRINGAN antara sumbu Z dan $\mathbf{n}$. Kemiringan adalah sudut antara sumbu X dan proyeksi $\mathbf{n}$ pada bidang gambar.

Ketika kamera bergerak relatif terhadap suatu objek, jarak dan orientasi objek berubah. Yang dipertahankan adalah **bentuk** objek. Jika objek tersebut berbentuk kubus, fakta tersebut tidak berubah ketika objek bergerak. Para ahli geometri telah mencoba memformalkan bentuk selama berabad-abad, dengan konsep dasarnya bahwa bentuk adalah sesuatu yang tetap tidak berubah di bawah beberapa kelompok transformasi—misalnya, kombinasi rotasi dan translasi. Kesulitannya terletak pada menemukan representasi bentuk global yang cukup umum untuk menangani beragam objek di dunia nyata—bukan hanya bentuk sederhana seperti silinder, kerucut, dan bola—namun dapat dipulihkan dengan mudah dari masukan visual. Permasalahan karakterisasi bentuk *lokal* suatu permukaan kini jauh lebih dipahami. Pada dasarnya, hal ini dapat dilakukan dalam

konteks kelengkungan: bagaimana normal permukaan berubah ketika seseorang bergerak ke berbagai arah di permukaan? Untuk bidang datar, tidak ada perubahan sama sekali. Untuk silinder, jika seseorang bergerak sejajar dengan sumbu, tidak ada perubahan, tetapi dalam arah tegak lurus, normal permukaan berputar dengan laju yang berbanding terbalik dengan jari-jari silinder, dan seterusnya. Semua ini dipelajari dalam mata kuliah yang disebut geometri diferensial.

Bentuk suatu objek relevan untuk beberapa tugas manipulasi (misalnya, menentukan di mana harus memegang suatu objek), tetapi perannya yang paling signifikan adalah dalam pengenalan objek, di mana bentuk geometris, warna, dan tekstur, memberikan isyarat paling signifikan yang memungkinkan kita mengidentifikasi objek, mengklasifikasikan apa yang ada dalam gambar sebagai contoh dari suatu kelas yang pernah dilihat sebelumnya, dan seterusnya.

6.5 PENGENALAN OBJEK DARI INFORMASI STRUKTURAL

Menempatkan kotak di sekitar pejalan kaki dalam sebuah gambar mungkin cukup untuk menghindari menabrak mereka. Kita telah melihat bahwa kita dapat menemukan kotak dengan menggabungkan bukti yang diberikan oleh orientasi, menggunakan metode histogram untuk menekan detail spasial yang berpotensi membingungkan. Jika kita ingin tahu lebih banyak tentang apa yang sedang dilakukan seseorang, kita perlu tahu di mana letak lengan, kaki, tubuh, dan kepala mereka dalam gambar. Setiap bagian tubuh cukup sulit dideteksi sendiri menggunakan metode jendela bergerak, karena warna dan teksturnya dapat sangat bervariasi dan karena biasanya berukuran kecil dalam gambar. Seringkali, lengan bawah dan tulang kering berukuran sekecil dua hingga tiga piksel. Bagian tubuh biasanya tidak muncul sendiri, dan merepresentasikan apa yang terhubung dengan apa bisa sangat kuat, karena bagian yang mudah ditemukan dapat memberi tahu kita di mana harus mencari bagian yang kecil dan sulit dideteksi.

Menyimpulkan tata letak tubuh manusia dalam gambar merupakan tugas penting dalam penglihatan, karena tata letak tubuh seringkali mengungkapkan apa yang sedang dilakukan orang. Sebuah model yang disebut templat yang dapat dideformasi dapat memberi tahu kita konfigurasi mana yang dapat diterima: siku dapat ditekuk tetapi kepala tidak pernah menyatu dengan kaki. Model templat paling sederhana yang dapat dideformasi dari seseorang menghubungkan lengan bawah ke lengan atas, lengan atas ke torso, dan seterusnya. Ada model yang lebih kaya: misalnya, kita dapat merepresentasikan fakta bahwa lengan atas kiri dan kanan cenderung memiliki warna dan tekstur yang sama, begitu pula kaki kiri dan kanan. Namun, model yang lebih kaya ini tetap sulit untuk dikerjakan.

Geometri Tubuh: Menemukan Lengan dan Kaki

Untuk saat ini, kita berasumsi bahwa kita mengetahui seperti apa bagian-bagian tubuh seseorang (misalnya, kita mengetahui warna dan tekstur pakaiannya). Kita dapat memodelkan geometri tubuh sebagai pohon yang terdiri dari sebelas segmen (masing-

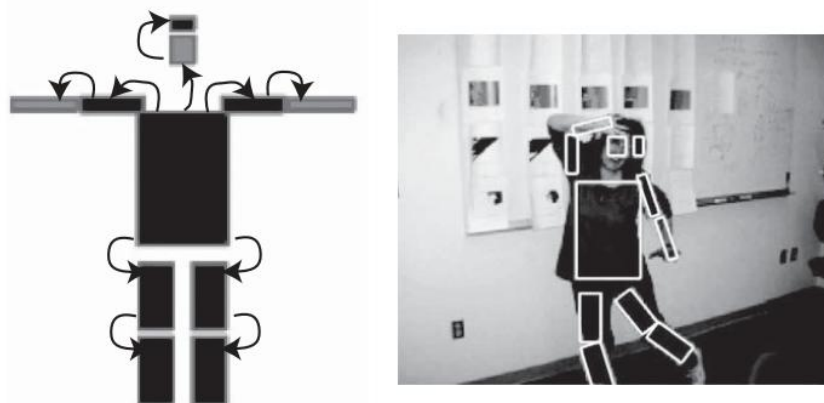
masing lengan dan kaki kiri atas dan bawah, torso, wajah, dan rambut di atas wajah) yang masing-masing berbentuk persegi panjang. Kita berasumsi bahwa posisi dan orientasi (**pose**) lengan bawah kiri tidak bergantung pada semua segmen lain berdasarkan pose lengan atas kiri; bahwa pose lengan atas kiri tidak bergantung pada semua segmen berdasarkan pose torso; dan memperluas asumsi ini dengan cara yang jelas untuk mencakup lengan dan kaki kanan, wajah, dan rambut. Model semacam ini sering disebut model "orang kardus". Model ini membentuk pohon, yang biasanya berakar di torso. Kita akan mencari citra untuk kecocokan terbaik dengan orang kardus ini menggunakan metode inferensi untuk jaringan Bayes berstruktur pohon.

Ada dua kriteria untuk mengevaluasi konfigurasi. Pertama, sebuah persegi panjang gambar harus terlihat seperti segmennya. Untuk saat ini, kita masih belum jelas tentang apa tepatnya maksudnya, tetapi kita asumsikan kita memiliki fungsi ϕ_i yang menilai seberapa cocok sebuah persegi panjang gambar dengan segmen tubuh. Untuk setiap pasangan segmen yang terkait, kita memiliki fungsi lain ψ yang menilai seberapa cocok hubungan antara sepasang persegi panjang gambar dengan hubungan yang diharapkan dari segmen tubuh. Ketergantungan antar segmen membentuk pohon, sehingga setiap segmen hanya memiliki satu induk, dan kita dapat menulis $\psi_i, pa(i)$. Semua fungsi akan lebih besar jika kecocokannya lebih baik, sehingga kita dapat menganggapnya sebagai probabilitas logaritma. Biaya kecocokan tertentu yang mengalokasikan persegi panjang gambar m_i ke segmen tubuh i kemudian adalah

$$\sum_{i \in \text{segments}} \phi_i(m_i) + \sum_{i \in \text{segments}} \psi_i pa(i) (m_i, m_{pa(i)}).$$

Pemrograman dinamis dapat menemukan kecocokan terbaik, karena model relasionalnya berupa pohon.

Mencari ruang kontinu tidaklah praktis, dan kita akan mendiskritisasi ruang persegi panjang gambar. Kita melakukannya dengan mendiskritisasi lokasi dan orientasi persegi panjang dengan ukuran tetap (ukurannya mungkin berbeda untuk segmen yang berbeda). Karena pergelangan kaki dan lutut berbeda, kita perlu membedakan antara persegi panjang dan persegi panjang yang sama yang diputar 180° . Kita dapat memvisualisasikan hasilnya sebagai sekumpulan tumpukan persegi panjang gambar kecil yang sangat besar, dipotong pada lokasi dan orientasi yang berbeda. Terdapat satu tumpukan per segmen. Kita sekarang harus menemukan alokasi terbaik persegi panjang ke segmen. Proses ini akan lambat, karena terdapat banyak persegi panjang gambar dan, untuk model yang telah kita berikan, memilih torso yang tepat akan menjadi $O(M^6)$ jika terdapat M persegi panjang gambar. Namun, berbagai percepatan tersedia untuk pilihan ψ yang tepat, dan metode ini praktis (Gambar 6.23).



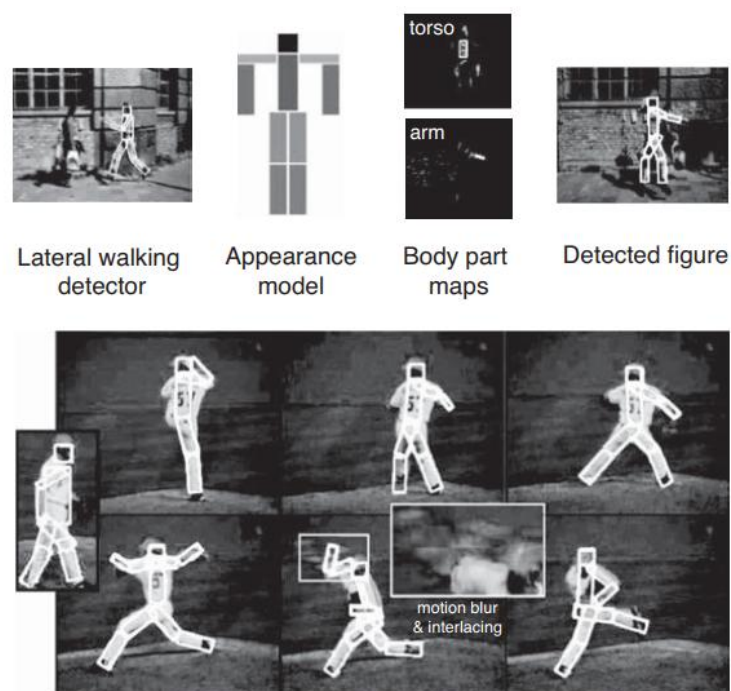
Gambar 6.23 Model struktur bergambar mengevaluasi kecocokan antara sekumpulan persegi panjang gambar dan orang kardus (ditunjukkan di sebelah kiri) dengan memberi skor pada kesamaan penampilan antara segmen tubuh dan segmen gambar serta hubungan spasial antara segmen gambar. Umumnya, kecocokan lebih baik jika segmen gambar memiliki penampilan yang tepat dan berada di tempat yang tepat satu sama lain. Model penampilan menggunakan warna rata-rata untuk rambut, kepala, badan, serta lengan dan kaki bagian atas dan bawah. Hubungan yang relevan ditunjukkan sebagai panah. Di sebelah kanan, kecocokan terbaik untuk gambar tertentu, diperoleh menggunakan pemrograman dinamis. Kecocokan tersebut merupakan perkiraan yang adil dari konfigurasi tubuh.

Model ini biasanya dikenal sebagai model struktur piktorial. Ingat asumsi kita bahwa kita tahu apa yang perlu kita ketahui tentang seperti apa rupa orang tersebut. Jika kita mencocokkan seseorang dalam satu gambar, fitur yang paling berguna untuk menilai kecocokan segmen ternyata adalah warna. Fitur tekstur tidak bekerja dengan baik dalam banyak kasus, karena lipatan pada pakaian longgar menghasilkan pola bayangan yang kuat yang melapisi tekstur gambar. Pola-pola ini cukup kuat untuk mengganggu tekstur kain yang sebenarnya. Dalam pekerjaan saat ini, ψ biasanya mencerminkan kebutuhan ujung-ujung segmen untuk cukup berdekatan, tetapi biasanya tidak ada batasan pada sudutnya. Umumnya, kita tidak tahu seperti apa rupa seseorang, dan harus membangun **model penampilan** segmen. Kita menyebut deskripsi tentang seperti apa rupa seseorang sebagai model penampilan. Jika kita harus melaporkan konfigurasi seseorang dalam satu gambar, kita dapat mulai dengan model penampilan yang tidak disetel dengan baik, memperkirakan konfigurasi dengan ini, lalu memperkirakan ulang penampilan, dan seterusnya. Dalam video, kita memiliki banyak bingkai dari orang yang sama, dan ini akan mengungkapkan penampilan mereka.

Penampilan yang Koheren: Melacak Orang dalam Video

Melacak orang dalam video merupakan masalah praktis yang penting. Jika kita dapat melaporkan lokasi lengan, kaki, badan, dan kepala secara akurat dalam rangkaian video, kita dapat membangun antarmuka permainan dan sistem pengawasan yang jauh

lebih baik. Metode penyaringan belum terlalu berhasil mengatasi masalah ini, karena orang dapat menghasilkan akselerasi yang besar dan bergerak cukup cepat. Ini berarti bahwa untuk video 30 Hz, konfigurasi tubuh dalam bingkai i tidak terlalu membatasi konfigurasi tubuh dalam bingkai $i + 1$. Saat ini, metode yang paling efektif memanfaatkan fakta bahwa penampilan berubah sangat lambat dari bingkai ke bingkai. Jika kita dapat menyimpulkan model penampilan seseorang dari video, maka kita dapat menggunakan informasi ini dalam model struktur gambar untuk mendeteksi orang tersebut di setiap bingkai video. Kita kemudian dapat menghubungkan lokasi-lokasi ini lintas waktu untuk membuat pelacakan.



Gambar 6.24 Kita dapat melacak orang yang bergerak dengan model struktur bergambar dengan terlebih dahulu memperoleh model penampilan, kemudian menerapkannya. Untuk memperoleh model penampilan, kita memindai gambar untuk menemukan pose berjalan lateral. Detektor tidak perlu sangat akurat, tetapi harus menghasilkan beberapa positif palsu. Dari respons detektor, kita dapat membaca piksel yang terletak pada setiap segmen tubuh, dan piksel lain yang tidak terletak pada segmen tersebut. Hal ini memungkinkan untuk membangun model diskriminatif dari penampilan setiap bagian tubuh, dan ini diikat bersama menjadi model struktur bergambar orang yang dilacak. Akhirnya, kita dapat melacak dengan andal dengan mendeteksi model ini di setiap bingkai. Seperti yang ditunjukkan oleh bingkai di bagian bawah gambar, prosedur ini dapat melacak konfigurasi tubuh yang rumit dan berubah cepat, meskipun terjadi degradasi sinyal video karena keburaman gerakan.

Ada beberapa cara untuk menyimpulkan model penampilan yang baik. Kita menganggap video sebagai tumpukan besar gambar orang yang ingin kita lacak. Kita dapat memanfaatkan tumpukan ini dengan mencari model penampilan yang menjelaskan banyak gambar. Ini akan bekerja dengan mendeteksi segmen tubuh di setiap bingkai, menggunakan fakta bahwa segmen memiliki tepi yang kira-kira sejajar. Detektor semacam itu tidak terlalu andal, tetapi segmen yang ingin kita temukan bersifat khusus. Segmen tersebut akan muncul setidaknya sekali di sebagian besar bingkai video; segmen tersebut dapat ditemukan dengan mengelompokkan respons detektor. Sebaiknya dimulai dengan torso, karena besar dan karena detektor torso cenderung andal. Setelah kita memiliki model penampilan torso, segmen kaki bagian atas seharusnya muncul di dekat torso, dan seterusnya. Penalaran ini menghasilkan model penampilan, tetapi bisa jadi tidak andal jika orang muncul dengan latar belakang yang hampir tetap di mana detektor segmen menghasilkan banyak positif palsu. Alternatifnya adalah memperkirakan penampilan untuk banyak bingkai video dengan memperkirakan ulang konfigurasi dan penampilan berulang kali; kita kemudian melihat apakah satu model penampilan menjelaskan banyak bingkai.



Gambar 6.25 Beberapa tindakan manusia yang kompleks menghasilkan pola penampilan dan gerakan yang konsisten. Misalnya, minum melibatkan gerakan tangan di depan wajah. Tiga gambar pertama merupakan deteksi minum yang benar; gambar keempat merupakan positif palsu (juru masak melihat ke dalam teko kopi, tetapi tidak meminumnya).

Alternatif lain, yang cukup andal dalam praktiknya, adalah menerapkan detektor untuk konfigurasi tubuh tetap pada semua frame. Pilihan konfigurasi yang baik adalah yang mudah dideteksi secara andal, dan yang memiliki kemungkinan besar orang tersebut akan muncul dalam konfigurasi tersebut meskipun dalam urutan singkat (berjalan menyamping adalah pilihan yang baik). Kami menyetel detektor agar memiliki tingkat positif palsu yang rendah, sehingga kami tahu ketika detektor merespons bahwa kami telah menemukan orang sungguhan; dan karena kami telah melokalisasi torso, lengan, kaki, dan kepala mereka, kami tahu seperti apa segmen-segmen ini.

6.6 MENGGUNAKAN VISI

Jika sistem visi dapat menganalisis video dan memahami apa yang dilakukan orang, kita akan mampu: merancang gedung dan tempat umum dengan lebih baik

dengan mengumpulkan dan menggunakan data tentang apa yang dilakukan orang di tempat umum; membangun sistem pengawasan yang lebih akurat, lebih aman, dan kurang intrusif; membangun komentator olahraga komputer; dan membangun antarmuka manusia-komputer yang mengamati orang dan bereaksi terhadap perilaku mereka.

Aplikasi untuk antarmuka reaktif berkisar dari permainan komputer yang membuat pemain berdiri dan bergerak hingga sistem yang menghemat energi dengan mengatur panas dan cahaya di dalam gedung agar sesuai dengan lokasi penghuni dan apa yang mereka lakukan. Beberapa masalah telah dipahami dengan baik. Jika orang relatif kecil dalam bingkai video, dan latar belakangnya stabil, mudah untuk mendeteksi orang tersebut dengan mengurangi gambar latar belakang dari bingkai saat ini. Jika nilai absolut selisihnya besar, **pengurangan latar belakang** ini menyatakan piksel tersebut sebagai piksel latar depan; dengan menghubungkan blob latar depan dari waktu ke waktu, kita memperoleh jejak.

Perilaku terstruktur seperti balet, senam, atau tai chi memiliki kosakata tindakan yang spesifik. Ketika dilakukan dengan latar belakang sederhana, video aksi-aksi ini mudah ditangani. Pengurangan latar belakang mengidentifikasi wilayah-wilayah utama yang bergerak, dan kita dapat membangun fitur HOG (mencatat alur alih-alih orientasi) untuk disajikan kepada pengklasifikasi. Kita dapat mendeteksi pola aksi yang konsisten dengan varian detektor pejalan kaki kita, di mana fitur orientasi dikumpulkan ke dalam keranjang histogram dari waktu ke waktu serta ruang (Gambar 6.25).

Permasalahan yang lebih umum masih terbuka. Pertanyaan penelitian besarnya adalah menghubungkan pengamatan terhadap tubuh dan objek di dekatnya dengan tujuan dan niat orang-orang yang bergerak. Salah satu sumber kesulitannya adalah kita kekurangan kosakata sederhana tentang perilaku manusia. Perilaku sangat mirip dengan warna, di mana orang cenderung berpikir mereka tahu banyak nama perilaku tetapi tidak dapat menghasilkan daftar panjang kata-kata tersebut sesuai permintaan. Ada cukup banyak bukti bahwa perilaku dapat bergabung—misalnya, Anda dapat minum milkshake saat mengunjungi ATM—tetapi kita belum tahu apa saja bagian-bagiannya, bagaimana komposisinya bekerja, atau berapa banyak komposit yang mungkin ada. Sumber kesulitan kedua adalah kita tidak tahu fitur apa yang mengungkap apa yang sedang terjadi. Misalnya, mengetahui seseorang berada di dekat ATM mungkin cukup untuk mengetahui bahwa mereka sedang mengunjungi ATM tersebut. Kesulitan ketiga adalah penalaran umum tentang hubungan antara data pelatihan dan data uji tidak dapat dipercaya. Misalnya, kita tidak dapat berargumen bahwa detektor pejalan kaki aman hanya karena berkinerja baik pada kumpulan data yang besar, karena kumpulan data tersebut mungkin saja tidak mencakup fenomena penting, tetapi jarang terjadi (misalnya, orang yang sedang bersepeda). Kita tentu tidak ingin pengemudi otomatis kita menabrak pejalan kaki yang kebetulan melakukan sesuatu yang tidak biasa.

Kata dan Gambar

Banyak situs web menawarkan koleksi gambar untuk dilihat. Bagaimana kita bisa menemukan gambar yang kita inginkan? Misalkan pengguna memasukkan kueri teks, seperti "balap sepeda". Beberapa gambar akan memiliki kata kunci atau keterangan terlampir, atau berasal dari halaman web yang berisi teks di dekat gambar. Untuk ini, pengambilan gambar bisa seperti pengambilan teks: abaikan gambar dan cocokkan teks gambar dengan kueri.

Namun, kata kunci biasanya tidak lengkap. Misalnya, gambar kucing yang sedang bermain di jalan mungkin diberi tag dengan kata-kata seperti "kucing" dan "jalan", tetapi mudah lupa menyebutkan "tong sampah" atau "tulang ikan". Oleh karena itu, tugas yang menarik adalah memberi anotasi pada gambar (yang mungkin sudah memiliki beberapa kata kunci) dengan kata kunci tambahan yang sesuai. Dalam versi paling sederhana dari tugas ini, kita memiliki sekumpulan gambar contoh yang diberi tag dengan benar, dan kita ingin memberi tag pada beberapa gambar uji. Masalah ini terkadang dikenal sebagai anotasi otomatis. Solusi paling akurat diperoleh menggunakan metode tetangga terdekat. Kita menemukan citra latih yang paling dekat dengan citra uji dalam metrik ruang fitur yang dilatih menggunakan contoh, lalu melaporkan tag-nya.

Versi lain dari masalah ini melibatkan prediksi tag mana yang akan dilampirkan ke region mana dalam citra uji. Di sini kita tidak tahu region mana yang menghasilkan tag mana untuk data latih. Kita dapat menggunakan versi maksimisasi ekspektasi untuk menebak korespondensi awal antara teks dan region, dan dari sana memperkirakan dekomposisi yang lebih baik ke dalam region, dan seterusnya.

Rekonstruksi dari berbagai tampilan

Stereopsis binokular berfungsi karena untuk setiap titik kita memiliki empat pengukuran yang membatasi tiga derajat kebebasan yang tidak diketahui. Keempat pengukuran tersebut adalah posisi (x, y) titik di setiap tampilan, dan derajat kebebasan yang tidak diketahui adalah nilai koordinat (x, y, z) titik di dalam pemandangan. Argumen yang agak kasar ini menunjukkan, dengan tepat, bahwa terdapat batasan geometris yang mencegah sebagian besar pasangan titik menjadi kecocokan yang dapat diterima. Banyak gambar dari sekumpulan titik seharusnya menunjukkan posisi mereka dengan jelas.

Kita tidak selalu membutuhkan gambar kedua untuk mendapatkan pandangan kedua dari sekumpulan titik. Jika kita yakin bahwa sekumpulan titik asli berasal dari objek 3D kaku yang familiar, maka kita mungkin memiliki model objek yang tersedia sebagai sumber informasi. Jika model objek ini terdiri dari sekumpulan titik 3D atau sekumpulan gambar objek, dan jika kita dapat menetapkan korespondensi titik, kita dapat menentukan parameter kamera yang menghasilkan titik-titik tersebut pada gambar asli. Ini adalah informasi yang sangat berguna. Kita dapat menggunakannya untuk mengevaluasi hipotesis awal kita bahwa titik-titik tersebut berasal dari model objek. Kita melakukan ini dengan menggunakan beberapa titik untuk menentukan parameter

kamera, kemudian memproyeksikan titik-titik model ke dalam kamera ini dan memeriksa apakah ada titik-titik gambar di dekatnya.

Di sini, kami telah membuat sketsa sebuah teknologi yang kini telah sangat berkembang. Teknologi ini dapat digeneralisasi untuk menangani tampilan yang tidak ortografis; menangani titik-titik yang hanya teramati dalam beberapa tampilan; menangani properti kamera yang tidak diketahui seperti panjang fokus; memanfaatkan berbagai pencarian canggih untuk menemukan korespondensi yang tepat; dan melakukan rekonstruksi dari sejumlah besar titik dan tampilan. Jika lokasi titik dalam gambar diketahui dengan akurasi tertentu dan arah tampilan masuk akal, informasi kamera dan titik dengan akurasi yang sangat tinggi dapat diperoleh. Beberapa aplikasinya adalah:

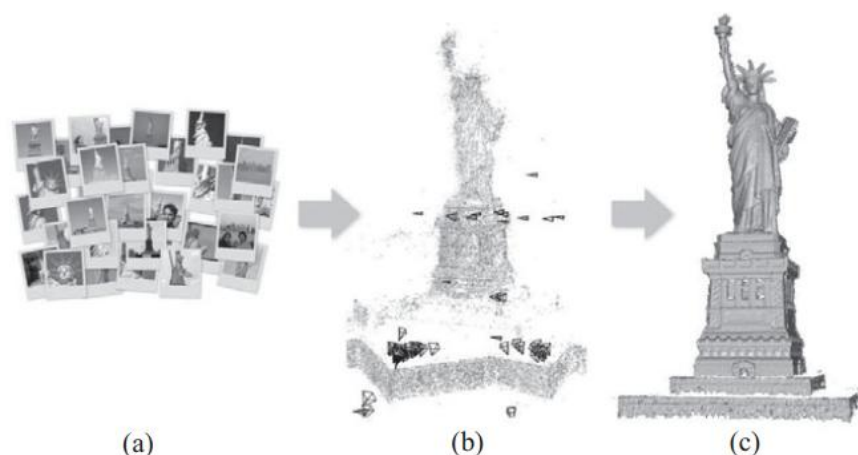
- **Pembuatan model:** Misalnya, seseorang dapat membangun sistem pemodelan yang mengambil rangkaian video yang menggambarkan suatu objek dan menghasilkan jaring tiga dimensi poligon bertekstur yang sangat detail untuk digunakan dalam aplikasi grafis komputer dan realitas virtual. Model seperti ini sekarang dapat dibangun dari kumpulan gambar yang tampaknya kurang menjanjikan. Misalnya, Gambar 6.26 menunjukkan model Patung Liberty yang dibangun dari gambar-gambar yang ditemukan di Internet.
- **Mencocokkan gerakan:** Untuk menempatkan karakter grafis komputer ke dalam video nyata, kita perlu mengetahui bagaimana kamera bergerak untuk video nyata tersebut, sehingga kita dapat merender karakter dengan benar.
- **Rekonstruksi jalur:** Robot bergerak perlu mengetahui di mana mereka berada. Jika mereka bergerak di dunia objek yang kaku, maka melakukan rekonstruksi dan menyimpan informasi kamera adalah salah satu cara untuk mendapatkan jalur.

Menggunakan penglihatan untuk mengendalikan gerakan

Salah satu kegunaan utama penglihatan adalah untuk menyediakan informasi baik untuk memanipulasi objek—mengambilnya, menggenggamnya, memutarnya, dan sebagainya—maupun untuk bernavigasi sambil menghindari rintangan. Kemampuan untuk menggunakan penglihatan untuk tujuan ini terdapat dalam sistem visual hewan yang paling primitif. Dalam banyak kasus, sistem visualnya minimal, dalam artian ia hanya mengekstrak dari medan cahaya yang tersedia informasi yang dibutuhkan hewan untuk menginformasikan perilakunya. Kemungkinan besar, sistem penglihatan modern berevolusi dari organisme primitif awal yang menggunakan titik fotosensitif di salah satu ujungnya untuk mengorientasikan diri ke arah (atau menjauhi) cahaya. Kita telah melihat bahwa lalat menggunakan sistem deteksi aliran optik yang sangat sederhana untuk mendarat di dinding. Sebuah studi klasik, *What the Frog's Eye Tells the Frog's Brain*, mengamati seekor katak bahwa, "Ia akan mati kelaparan dikelilingi makanan jika makanan itu tidak bergerak. Pilihan makanannya hanya ditentukan oleh ukuran dan gerakan."

Mari kita pertimbangkan sistem penglihatan untuk kendaraan otomatis yang melaju di jalan bebas hambatan. Tugas-tugas yang dihadapi pengemudi meliputi hal-hal berikut:

1. Kontrol lateral—pastikan kendaraan tetap aman di jalurnya atau berpindah jalur dengan lancar saat diperlukan.
2. Kontrol longitudinal—pastikan terdapat jarak aman dengan kendaraan di depan.
3. Penghindaran rintangan—pantau kendaraan di jalur tetangga dan bersiaplah untuk manuver mengelak jika salah satu dari mereka memutuskan untuk berpindah jalur.



Gambar 6.26. Teknologi rekonstruksi multi-tampilan kini sudah sangat maju. Gambar ini menguraikan sistem yang dibangun oleh Michael Goesele dan rekan-rekannya dari Universitas Washington, TU Darmstadt, dan Microsoft Research. Dari kumpulan gambar monumen yang diambil oleh komunitas pengguna yang besar dan diunggah di internet (a), sistem mereka dapat menentukan arah tampilan untuk gambar-gambar tersebut, yang ditunjukkan oleh piramida hitam kecil pada (b) dan rekonstruksi 3D komprehensif yang ditunjukkan pada (c).

Permasalahan bagi pengemudi adalah menghasilkan tindakan kemudi, akselerasi, dan pengereman yang tepat untuk mencapai tugas-tugas ini dengan sebaik-baiknya.

Untuk kontrol lateral, seseorang perlu mempertahankan representasi posisi dan orientasi mobil relatif terhadap jalur. Kita dapat menggunakan algoritma deteksi tepi untuk menemukan tepi yang sesuai dengan segmen penanda jalur. Kita kemudian dapat menyesuaikan kurva halus dengan elemen tepi ini. Parameter kurva ini membawa informasi tentang posisi lateral mobil, arah yang dituju relatif terhadap jalur, dan kelengkungan jalur. Informasi ini, beserta informasi tentang dinamika mobil, adalah semua yang dibutuhkan oleh sistem kendali kemudi. Jika kita memiliki peta jalan yang detail dan baik, maka sistem penglihatan berfungsi untuk mengonfirmasi posisi kita (dan untuk mengawasi rintangan yang tidak ada di peta).

Untuk kendali longitudinal, seseorang perlu mengetahui jarak ke kendaraan di depan. Hal ini dapat dicapai dengan stereopsis binokular atau aliran optik. Dengan menggunakan teknik ini, mobil yang dikendalikan penglihatan kini dapat melaju dengan andal pada kecepatan jalan raya.

Kasus yang lebih umum tentang robot bergerak yang bernavigasi di berbagai lingkungan dalam dan luar ruangan juga telah dipelajari. Satu masalah khusus, yaitu melokalisasi robot di lingkungannya, kini memiliki solusi yang cukup baik. Sebuah kelompok di Sarnoff telah mengembangkan sistem berbasis dua kamera yang melihat ke depan yang melacak titik-titik fitur dalam 3D dan menggunakannya untuk merekonstruksi posisi robot relatif terhadap lingkungan. Faktanya, mereka memiliki dua sistem kamera stereoskopik, satu menghadap ke depan dan satu menghadap ke belakang—ini memberikan ketahanan yang lebih baik jika robot harus melewati area tanpa fitur karena bayangan gelap, dinding kosong, dan sebagainya. Kemungkinan besar tidak ada fitur di bagian depan maupun belakang. Tentu saja, hal itu bisa saja terjadi, sehingga cadangan disediakan dengan menggunakan unit gerak inersia (IMU) yang agak mirip dengan mekanisme penginderaan percepatan yang dimiliki manusia di telinga bagian dalam. Dengan mengintegrasikan percepatan yang dideteksi dua kali, seseorang dapat melacak perubahan posisi. Menggabungkan data dari penglihatan dan IMU merupakan masalah fusi bukti probabilistik dan dapat diatasi menggunakan teknik-teknik, seperti penyaringan Kalman, yang telah kita pelajari di bagian lain buku ini.

Dalam penggunaan odometri visual (estimasi perubahan posisi), seperti dalam permasalahan odometri lainnya, terdapat masalah "penyimpangan", yaitu kesalahan posisi yang terakumulasi seiring waktu. Solusinya adalah menggunakan landmark untuk memberikan posisi tetap absolut: segera setelah robot melewati suatu lokasi di peta internalnya, ia dapat menyesuaikan perkiraan posisinya dengan tepat. Akurasi hingga hitungan sentimeter telah ditunjukkan dengan teknik-teknik ini.

Contoh mengemudi memperjelas satu hal: untuk tugas tertentu, seseorang tidak perlu memulihkan semua informasi yang, pada prinsipnya, dapat dipulihkan dari sebuah gambar. Seseorang tidak perlu memulihkan bentuk persis setiap kendaraan, memecahkan masalah bentuk-dari-tekstur pada permukaan rumput di dekat jalan bebas hambatan, dan sebagainya. Sebaliknya, sistem penglihatan harus menghitung apa yang dibutuhkan untuk menyelesaikan tugas tersebut.

BAB 7

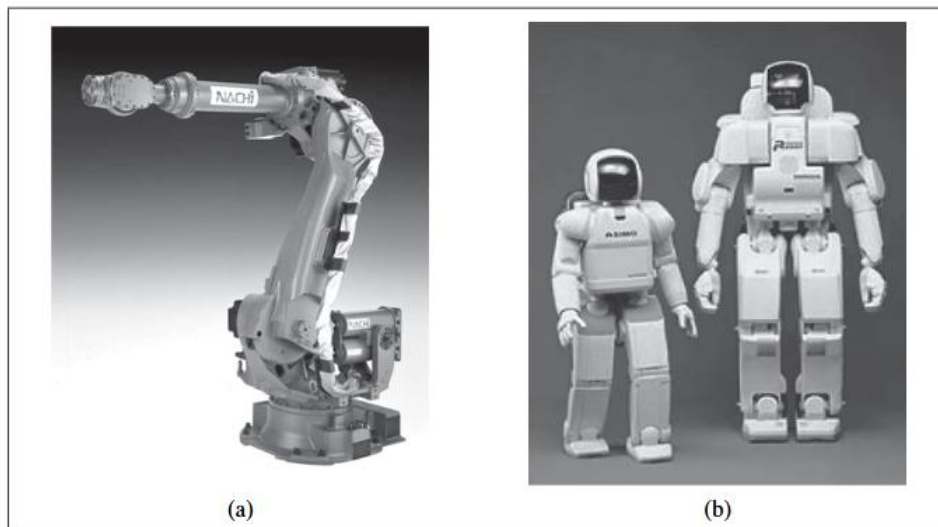
ROBOTIK

7.1 PENDAHULUAN

Robot adalah agen fisik yang melakukan tugas dengan memanipulasi dunia fisik. Untuk melakukannya, mereka dilengkapi dengan efektor seperti kaki, roda, sendi, dan gripper. Efektor memiliki satu tujuan: untuk memberikan gaya fisik pada lingkungan. Robot juga dilengkapi dengan sensor, yang memungkinkan mereka untuk mengamati lingkungan mereka. Robotika masa kini menggunakan beragam sensor, termasuk kamera dan laser untuk mengukur lingkungan, serta giroskop dan akselerometer untuk mengukur gerakan robot itu sendiri.

Sebagian besar robot masa kini terbagi dalam salah satu dari tiga kategori utama. Manipulator, atau lengan robot (Gambar 7.1(a)), secara fisik ditambatkan ke tempat kerja mereka, misalnya di jalur perakitan pabrik atau di Stasiun Luar Angkasa Internasional. Gerakan manipulator biasanya melibatkan serangkaian sendi yang dapat dikontrol, yang memungkinkan robot tersebut menempatkan efektornya di posisi mana pun di tempat kerja. Manipulator sejauh ini merupakan jenis robot industri yang paling umum, dengan sekitar satu juta unit terpasang di seluruh dunia. Beberapa manipulator bergerak digunakan di rumah sakit untuk membantu ahli bedah. Hanya sedikit produsen mobil yang dapat bertahan hidup tanpa manipulator robotik, dan beberapa manipulator bahkan telah digunakan untuk menghasilkan karya seni asli.

Kategori kedua adalah robot bergerak. Robot bergerak bergerak di lingkungannya menggunakan roda, kaki, atau mekanisme serupa. Robot ini telah digunakan untuk mengantarkan makanan di rumah sakit, memindahkan kontainer di dok pemuatan, dan tugas-tugas serupa. Kendaraan darat tanpa awak, atau UGVs, melaju secara otonom di jalan raya, jalan tol, dan di luar jalan raya. Penjelajah planet yang ditunjukkan pada Gambar 7.2(b) menjelajahi Mars selama 3 bulan pada tahun 1997. Robot NASA berikutnya termasuk dua Penjelajah Eksplorasi Mars (salah satunya digambarkan di sampul buku ini), yang mendarat pada tahun 2003 dan masih beroperasi enam tahun kemudian. Jenis robot bergerak lainnya termasuk kendaraan udara tak berawak (UAVs), yang umum digunakan untuk pengawasan, penyemprotan tanaman, dan operasi militer. Gambar 7.2(a) menunjukkan UAVs yang umum digunakan oleh militer AS. Kendaraan bawah air otonom (AUVs) digunakan dalam eksplorasi laut dalam. Robot bergerak mengirimkan paket di tempat kerja dan menyedot debu lantai di rumah.



Gambar 7.1 (a) Robot manipulator industri untuk menumpuk tas di atas palet. Gambar milik Nachi Robotic Systems. (b) Robot humanoid Honda P3 dan Asimo.



Gambar 7.2 (a) Predator, wahana udara nirawak (UAV) yang digunakan oleh Militer AS. Gambar milik General Atomics Aeronautical Systems. (b) Sojourner milik NASA, robot bergerak yang menjelajahi permukaan Mars pada Juli 1997.

Jenis robot ketiga menggabungkan mobilitas dengan manipulasi, dan sering disebut manipulator bergerak. Robot humanoid meniru tubuh manusia. Gambar 7.1(b) menunjukkan dua robot humanoid awal, keduanya diproduksi oleh Honda Corp. di Jepang. Manipulator bergerak dapat menerapkan efekturnya lebih jauh daripada manipulator berlabuh, tetapi tugas mereka menjadi lebih sulit karena tidak memiliki kekakuan yang disediakan oleh jangkar.

Bidang robotika juga mencakup perangkat prostetik (anggota tubuh, telinga, dan mata buatan untuk manusia), lingkungan cerdas (seperti seluruh rumah yang dilengkapi dengan sensor dan efektor), dan sistem multibodi, di mana aksi robotik dicapai melalui kawanan robot kecil yang bekerja sama. Robot sungguhan harus mampu menghadapi lingkungan yang sebagian dapat diamati, stokastik, dinamis, dan kontinu. Banyak

lingkungan robot juga bersifat sekuensial dan multiagen. Observabilitas parsial dan stokastisitas merupakan hasil dari penanganan dunia yang besar dan kompleks. Kamera robot tidak dapat melihat sudut, dan perintah gerakan rentan terhadap ketidakpastian akibat selip gigi, gesekan, dll. Selain itu, dunia nyata dengan keras kepala menolak untuk beroperasi lebih cepat daripada waktu nyata. Dalam lingkungan simulasi, dimungkinkan untuk menggunakan algoritma sederhana untuk belajar dalam beberapa jam CPU dari jutaan percobaan. Dalam lingkungan nyata, mungkin diperlukan waktu bertahun-tahun untuk menjalankan percobaan ini. Lebih lanjut, tabrakan sungguhan sangat menyakitkan, tidak seperti yang disimulasikan. Sistem robotika praktis perlu mewujudkan pengetahuan sebelumnya tentang robot, lingkungan fisiknya, dan tugas-tugas yang akan dilakukan robot agar robot dapat belajar dengan cepat dan bekerja dengan aman.

Robotika menggabungkan banyak konsep yang telah kita bahas sebelumnya dalam buku ini, termasuk estimasi keadaan probabilistik, persepsi, perencanaan, pembelajaran tanpa pengawasan, dan pembelajaran penguatan. Untuk beberapa konsep ini, robotika berfungsi sebagai contoh aplikasi yang menantang. Untuk konsep-konsep lain, bab ini membuka jalan baru dengan memperkenalkan versi kontinu dari teknik-teknik yang sebelumnya hanya kita lihat dalam kasus diskrit.

7.2 PERANGKAT KERAS ROBOT

Sejauh ini dalam buku ini, kita telah mengambil arsitektur agen sensor, efektor, dan prosesor sebagaimana adanya, dan kita telah berkonsentrasi pada program agen. Keberhasilan robot sungguhan setidaknya bergantung pada desain sensor dan efektor yang sesuai untuk tugasnya.

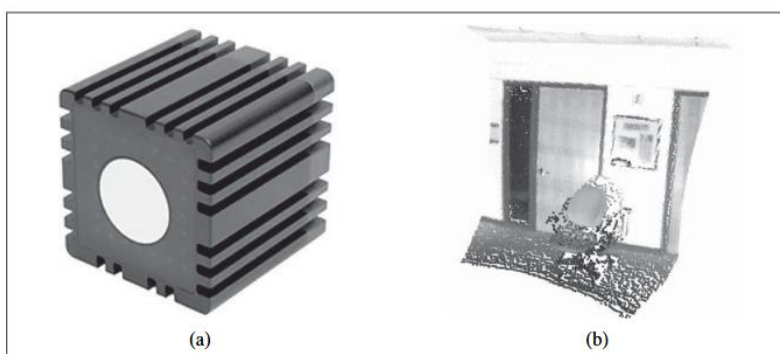
Sensor

Sensor adalah antarmuka persepsi antara robot dan lingkungan. Sensor pasif, seperti kamera, adalah pengamat sejati lingkungan: mereka menangkap sinyal yang dihasilkan oleh sumber lain di lingkungan. Sensor aktif, seperti sonar, mengirimkan energi ke lingkungan. Mereka bergantung pada fakta bahwa energi ini dipantulkan kembali ke sensor. Sensor aktif cenderung memberikan lebih banyak informasi daripada sensor pasif, tetapi dengan mengorbankan peningkatan konsumsi daya dan dengan risiko interferensi ketika beberapa sensor aktif digunakan secara bersamaan. Baik aktif maupun pasif, sensor dapat dibagi menjadi tiga jenis, tergantung pada apakah mereka merasakan lingkungan, lokasi robot, atau konfigurasi internal robot.

Pencari jarak adalah sensor yang mengukur jarak ke objek di sekitar. Pada masa-masa awal robotika, robot umumnya dilengkapi dengan sensor sonar. Sensor sonar memancarkan gelombang suara terarah, yang dipantulkan oleh objek, dengan sebagian suara kembali ke sensor. Waktu dan intensitas sinyal yang kembali menunjukkan jarak ke objek di sekitar. Sonar adalah teknologi pilihan untuk kendaraan bawah air otonom. Penglihatan stereo bergantung pada beberapa kamera untuk mengambil gambar

lingkungan dari sudut pandang yang sedikit berbeda, menganalisis paralaks yang dihasilkan dalam gambar-gambar ini untuk menghitung jarak objek di sekitarnya. Untuk robot darat bergerak, sonar dan penglihatan stereo sekarang jarang digunakan, karena akurasi kurang dapat diandalkan.

Sebagian besar robot darat sekarang dilengkapi dengan pencari jarak optik. Sama seperti sensor sonar, sensor jarak optik memancarkan sinyal aktif (cahaya) dan mengukur waktu hingga pantulan sinyal ini tiba kembali di sensor. Gambar 7.3(a) menunjukkan waktu terbang kamera. Kamera ini memperoleh gambar jangkauan seperti yang ditunjukkan pada Gambar 7.3(b) hingga 60 bingkai per detik. Sensor jangkauan lainnya menggunakan sinar laser dan kamera 1-piksel khusus yang dapat diarahkan menggunakan susunan cermin atau elemen berputar yang kompleks. Sensor ini disebut lidar pemindaian (singkatan dari *light detection and ranging*). Lidar pemindaian cenderung memberikan jangkauan yang lebih jauh daripada kamera time of flight, dan cenderung berkinerja lebih baik di siang hari yang cerah.



Gambar 7.3 (a) Kamera waktu terbang; gambar milik Mesa Imaging GmbH. (b) Citra jangkauan 3D yang diperoleh dengan kamera ini. Citra jangkauan memungkinkan deteksi rintangan dan objek di sekitar robot.

Sensor jangkauan umum lainnya termasuk radar, yang seringkali menjadi sensor pilihan untuk UAV. Sensor radar dapat mengukur jarak beberapa kilometer. Di ujung ekstrem lain dari penginderaan jangkauan terdapat sensor taktil seperti whisker, bump panel, dan kulit sensitif sentuhan. Sensor ini mengukur jangkauan berdasarkan kontak fisik, dan hanya dapat digunakan untuk mengindra objek yang sangat dekat dengan robot.

Kelas sensor penting kedua adalah sensor lokasi. Sebagian besar sensor lokasi menggunakan penginderaan jangkauan sebagai komponen utama untuk menentukan lokasi. Di luar ruangan, Sistem Pemosisian Global (GPS) adalah solusi paling umum untuk masalah lokalisasi. GPS mengukur jarak ke satelit yang memancarkan sinyal berdenyut. Saat ini, terdapat 31 satelit di orbit, yang memancarkan sinyal pada berbagai frekuensi. Penerima GPS dapat memulihkan jarak ke satelit-satelit ini dengan menganalisis pergeseran fasa. Dengan melakukan triangulasi sinyal dari berbagai satelit, penerima

GPS dapat menentukan lokasi absolut mereka di Bumi hingga beberapa meter. GPS diferensial melibatkan penerima darat kedua dengan lokasi yang diketahui, memberikan akurasi milimeter dalam kondisi ideal. Sayangnya, GPS tidak berfungsi di dalam ruangan atau di bawah air. Di dalam ruangan, lokalisasi sering dicapai dengan memasang suar di lingkungan pada lokasi yang diketahui. Banyak lingkungan dalam ruangan penuh dengan stasiun pangkalan nirkabel, yang dapat membantu robot melokalisasi melalui analisis sinyal nirkabel. Di bawah air, suar sonar aktif dapat memberikan rasa lokasi, menggunakan suara untuk memberi tahu AUV tentang jarak relatif mereka ke suar tersebut.

Kelas penting ketiga adalah sensor proprioseptif, yang memberi tahu robot tentang gerakannya sendiri. Untuk mengukur konfigurasi sendi robot yang tepat, motor sering kali dilengkapi dengan dekoder poros yang menghitung putaran motor dalam peningkatan kecil. Pada lengan robot, dekoder poros dapat memberikan informasi akurat dalam periode waktu apa pun. Pada robot bergerak, dekoder poros yang melaporkan putaran roda dapat digunakan untuk odometri pengukuran jarak yang ditempuh. Sayangnya, roda cenderung melayang dan selip, sehingga odometri hanya akurat dalam jarak pendek. Gaya eksternal, seperti arus untuk AUV dan angin untuk UAV, meningkatkan ketidakpastian posisi. Sensor inersia, seperti giroskop, bergantung pada resistansi massa terhadap perubahan kecepatan. Sensor ini dapat membantu mengurangi ketidakpastian.

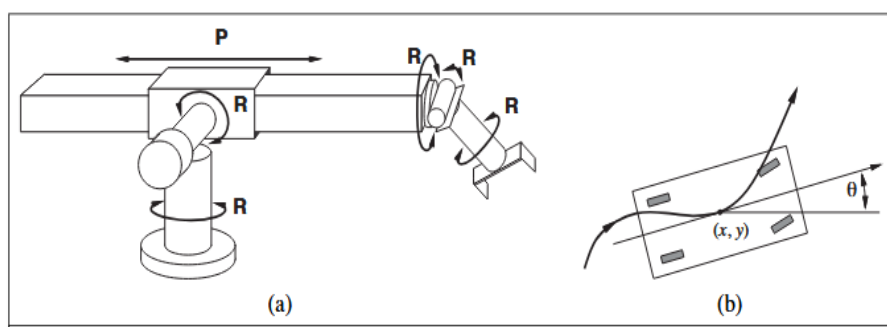
Aspek penting lainnya dari status robot diukur oleh sensor gaya dan sensor torsi. Sensor ini sangat diperlukan ketika robot menangani objek rapuh atau objek yang bentuk dan lokasi pastinya tidak diketahui. Bayangkan seorang manipulator robot seberat satu ton sedang memasang bola lampu. Sangat mudah untuk menerapkan terlalu banyak gaya dan mematahkan bola lampu. Sensor gaya memungkinkan robot merasakan seberapa kuat ia mencengkeram bohlam, dan sensor torsi memungkinkannya merasakan seberapa keras ia berputar. Sensor yang baik dapat mengukur gaya dalam ketiga arah translasi dan tiga arah rotasi. Sensor ini melakukan ini pada frekuensi beberapa ratus kali per detik, sehingga robot dapat dengan cepat mendeteksi gaya tak terduga dan mengoreksi tindakannya sebelum mematahkan bola lampu.

Efektor

Efektor adalah cara robot bergerak dan mengubah bentuk tubuhnya. Untuk memahami desain efektor, akan membantu jika kita membahas gerak dan bentuk secara abstrak, menggunakan konsep derajat kebebasan (DOF). Kita menghitung satu derajat kebebasan untuk setiap arah independen di mana robot, atau salah satu efekturnya, dapat bergerak. Misalnya, robot bergerak kaku seperti AUV memiliki enam derajat kebebasan, tiga untuk lokasi (x, y, z) di ruang angkasa dan tiga untuk orientasi sudutnya, yang dikenal sebagai *yaw*, *roll*, dan *pitch*. Enam derajat ini menentukan keadaan kinematik atau pose robot. Keadaan dinamis robot mencakup keenam derajat ini

ditambah enam dimensi tambahan untuk laju perubahan setiap dimensi kinematik, yaitu kecepatannya.

Untuk benda non-kaku, terdapat derajat kebebasan tambahan di dalam robot itu sendiri. Misalnya, siku lengan manusia memiliki dua derajat kebebasan. Siku dapat menekuk lengan atas ke arah atau menjauh, dan dapat berputar ke kanan atau ke kiri. Pergelangan tangan memiliki tiga derajat kebebasan. Pergelangan tangan dapat bergerak ke atas dan ke bawah, ke samping, dan juga dapat berputar. Sendi robot juga memiliki satu, dua, atau tiga derajat kebebasan masing-masing. Enam derajat kebebasan diperlukan untuk menempatkan suatu objek, seperti tangan, pada titik tertentu dalam orientasi tertentu. Lengan pada Gambar 7.4(a) memiliki tepat enam derajat kebebasan, yang dihasilkan oleh lima sendi putar yang menghasilkan gerak rotasi dan satu sendi prismatic yang menghasilkan gerak geser. Anda dapat memverifikasi bahwa lengan manusia secara keseluruhan memiliki lebih dari enam derajat kebebasan dengan eksperimen sederhana: letakkan tangan Anda di atas meja dan perhatikan bahwa Anda masih memiliki kebebasan untuk memutar siku tanpa mengubah konfigurasi tangan Anda. Manipulator yang memiliki derajat kebebasan ekstra lebih mudah dikendalikan daripada robot dengan jumlah DOFs minimum. Oleh karena itu, banyak manipulator industri memiliki tujuh DOFs, bukan enam.



Gambar 7.4 (a) Manipulator Stanford, lengan robot awal dengan lima sendi putar (R) dan satu sendi prismatic (P), dengan total enam derajat kebebasan. (b) Gerakan kendaraan roda empat nonholonomik dengan kemudi roda depan.

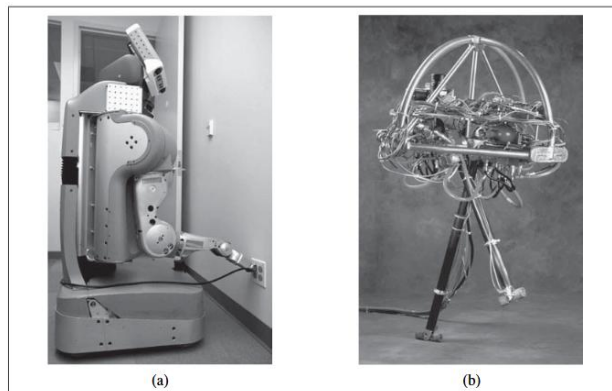
Untuk robot bergerak, DOFs tidak selalu sama dengan jumlah elemen yang digerakkan. Pertimbangkan, misalnya, mobil biasa: mobil dapat bergerak maju atau mundur, dan dapat berbelok, sehingga memiliki dua DOFs. Sebaliknya, konfigurasi kinematik mobil bersifat tiga dimensi: pada permukaan datar yang terbuka, seseorang dapat dengan mudah mengarahkan mobil ke titik (x, y) mana pun, dalam orientasi apa pun. Dengan demikian, mobil memiliki tiga derajat kebebasan efektif tetapi dua derajat kebebasan yang dapat dikontrol. Kita mengatakan robot nonholonomik jika memiliki lebih banyak DOFs efektif daripada DOFs yang dapat dikontrol dan holonomik jika kedua angka tersebut sama. Robot holonomik lebih mudah dikendalikan akan jauh lebih mudah untuk

memarkir mobil yang dapat bergerak menyamping serta maju dan mundur tetapi robot holonomik juga secara mekanis lebih kompleks. Sebagian besar lengan robot bersifat holonomik, dan sebagian besar robot bergerak bersifat nonholonomik.

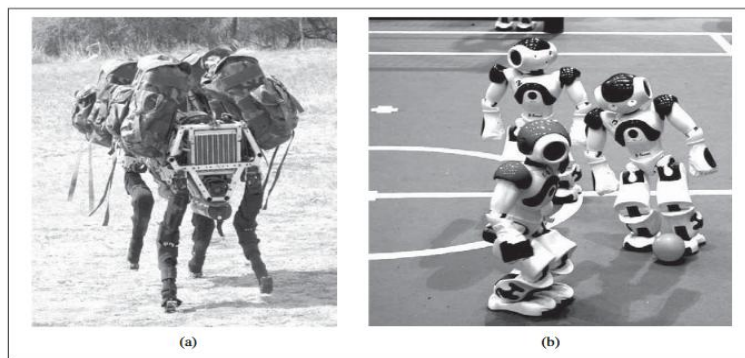
Robot bergerak memiliki beragam mekanisme penggerak, termasuk roda, rel, dan kaki. Robot penggerak diferensial memiliki dua roda (atau rel) yang digerakkan secara independen, satu di setiap sisi, seperti pada tank militer. Jika kedua roda bergerak dengan kecepatan yang sama, robot bergerak lurus. Jika bergerak berlawanan arah, robot berputar di tempat. Alternatifnya adalah penggerak sinkronisasi, di mana setiap roda dapat bergerak dan berputar pada porosnya sendiri. Untuk menghindari kekacauan, roda-roda tersebut dikoordinasikan dengan ketat. Ketika bergerak lurus, misalnya, semua roda mengarah ke arah yang sama dan bergerak dengan kecepatan yang sama. Baik penggerak diferensial maupun penggerak sinkronisasi bersifat nonholonomik. Beberapa robot yang lebih mahal menggunakan penggerak holonomik, yang memiliki tiga roda atau lebih yang dapat diarahkan dan digerakkan secara independen.

Beberapa robot bergerak memiliki lengan. Gambar 7.5(a) menampilkan robot berlengan ganda. Lengan robot ini menggunakan pegas untuk mengimbangi gravitasi, dan memberikan hambatan minimal terhadap gaya eksternal. Desain semacam ini meminimalkan bahaya fisik bagi orang yang mungkin menabrak robot tersebut. Ini adalah pertimbangan utama dalam penerapan robot di lingkungan rumah tangga. Kaki, tidak seperti roda, dapat menangani medan yang kasar. Namun, kaki terkenal lambat di permukaan datar, dan secara mekanis sulit dibuat. Para peneliti robotika telah mencoba desain mulai dari satu kaki hingga puluhan kaki. Robot berkaki telah dibuat untuk berjalan, berlari, dan bahkan melompat seperti yang kita lihat pada robot berkaki pada Gambar 7.5(b). Robot ini stabil secara dinamis, artinya ia dapat tetap tegak saat melompat-lompat. Robot yang dapat tetap tegak tanpa menggerakkan kakinya disebut stabil secara statis. Sebuah robot stabil secara statis jika pusat gravitasinya berada di atas poligon yang direntangkan oleh kakinya. Robot berkaki empat (berkaki empat) yang ditunjukkan pada Gambar 7.6(a) mungkin tampak stabil secara statis. Namun, ia berjalan dengan mengangkat beberapa kaki secara bersamaan, yang membuatnya stabil secara dinamis. Robot ini dapat berjalan di atas salju dan es, dan tidak akan jatuh bahkan jika Anda menendangnya. Robot berkaki dua seperti pada Gambar 7.6(b) stabil secara dinamis.

Metode pergerakan lain juga dimungkinkan: kendaraan udara menggunakan baling-baling atau turbin; kendaraan bawah air menggunakan baling-baling atau pendorong, serupa dengan yang digunakan pada kapal selam. Balon udara robotik bergantung pada efek termal untuk tetap berada di udara.



Gambar 7.5 (a) Manipulator bergerak sedang mencolokkan kabel pengisi dayanya ke stopkontak. (b) Salah satu robot berkaki karya Marc Raibert yang sedang bergerak.



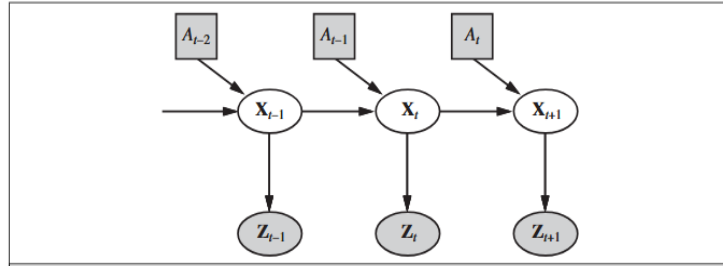
Gambar 7.6 (a) Robot berkaki empat yang stabil secara dinamis "Big Dog". (b) Kompetisi RoboCup Standard Platform League 2009, menampilkan tim pemenang, B-Human, dari pusat DFKI di Universitas Bremen. Sepanjang pertandingan, B-Human mengungguli lawan mereka dengan skor 64:1. Keberhasilan mereka dibangun berdasarkan estimasi keadaan probabilistik menggunakan filter partikel dan filter Kalman; model pembelajaran mesin untuk optimasi gaya berjalan; dan gerakan tendangan dinamis.

Sensor dan efektor saja tidak dapat membentuk robot. Robot yang lengkap juga membutuhkan sumber daya untuk menggerakkan efektornya. Motor listrik adalah mekanisme yang paling populer untuk aktuasi manipulator dan penggerak, tetapi aktuasi pneumatik menggunakan gas terkompresi dan aktuasi hidrolik menggunakan fluida bertekanan juga memiliki ceruk aplikasinya masing-masing.

7.3 PERSEPSI ROBOTIK

Persepsi adalah proses yang digunakan robot untuk memetakan pengukuran sensor ke dalam representasi internal lingkungan. Persepsi sulit dilakukan karena sensornya bising, dan lingkungannya sebagian dapat diamati, tidak dapat diprediksi, dan seringkali dinamis. Sebagai aturan praktis, representasi internal yang baik untuk robot

memiliki tiga sifat: mengandung informasi yang cukup bagi robot untuk membuat keputusan yang baik, terstruktur sehingga dapat diperbarui secara efisien, dan alami dalam arti variabel internal berkorespondensi dengan variabel keadaan alami di dunia fisik.



Gambar 7.7 Persepsi robot dapat dilihat sebagai inferensi temporal dari rangkaian tindakan dan pengukuran, seperti yang diilustrasikan oleh jaringan Bayes dinamis ini.

Penulis melihat bahwa filter Kalman, HMM, dan jaring Bayes dinamis dapat merepresentasikan model transisi dan sensor dari lingkungan yang sebagian dapat diamati, dan kami menjelaskan algoritma eksak dan aproksimasi untuk memperbarui keadaan keyakinan distribusi probabilitas posterior atas variabel keadaan lingkungan. Untuk permasalahan robotika, kami memasukkan tindakan robot di masa lalu sebagai variabel teramati dalam model. Gambar 7.7 menunjukkan notasi yang digunakan dalam bab ini: X_t adalah keadaan lingkungan (termasuk robot) pada waktu t , Z_t adalah observasi yang diterima pada waktu t , dan A_t adalah tindakan yang diambil setelah observasi diterima.

Kami ingin menghitung keadaan keyakinan baru, $P(X_{t+1} | z_{1:t+1}, X_{1:t})$, dari keadaan keyakinan saat ini $P(X_t | z_{1:t}, a_{1:t-1})$ dan observasi baru z_{t+1} . Kami telah melakukan hal ini, tetapi di sini terdapat dua perbedaan: kami mengondisikan secara eksplisit pada tindakan serta observasi, dan kami menangani variabel kontinu, bukan diskrit. Oleh karena itu, kami memodifikasi persamaan penyaringan rekursif untuk menggunakan integrasi, alih-alih penjumlahan:

$$\begin{aligned}
 &P(X_{t+1} | z_{1:t+1}, a_t) \\
 &= aP(z_{t+1} | X_{t+1}) \int P(X_{t+1} | x_t, a_t) P(X_t | z_{1:t}, a_{1:t-1}) dx_t
 \end{aligned} \tag{7.1}$$

Persamaan ini menyatakan bahwa posterior atas variabel keadaan X pada waktu $t + 1$ dihitung secara rekursif dari estimasi terkait satu langkah waktu sebelumnya. Perhitungan ini melibatkan tindakan sebelumnya pada a_t dan pengukuran sensor saat ini z_{t+1} . Misalnya, jika tujuan kita adalah mengembangkan robot pemain sepak bola, X_{t+1} mungkin merupakan lokasi bola sepak relatif terhadap robot. Posterior

$P(X_t | z_{1:t}, a_{1:t-1})$ adalah distribusi probabilitas atas semua keadaan yang menangkap apa yang kita ketahui dari pengukuran dan kontrol sensor sebelumnya. Persamaan (7.1) memberi tahu kita cara memperkirakan lokasi ini secara rekursif, dengan melipat secara bertahap pengukuran sensor (misalnya, gambar kamera) dan perintah gerakan robot. Probabilitas $P(X_{t+1} | x_t, a_t)$ disebut model transisi atau model gerakan, dan $P(z_{t+1} | X_{t+1})$ adalah model sensor.

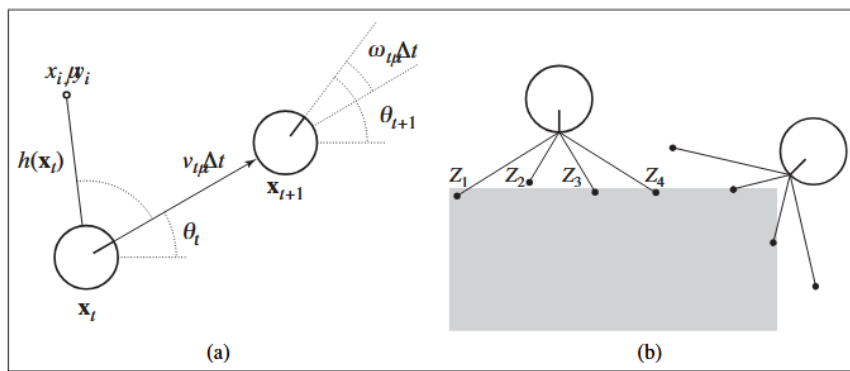
Lokalisasi dan Pemetaan

Lokalisasi adalah masalah untuk mengetahui di mana sesuatu berada termasuk robot itu sendiri. Pengetahuan tentang letak suatu benda merupakan inti dari setiap interaksi fisik yang sukses dengan lingkungan. Misalnya, robot manipulator harus mengetahui lokasi objek yang ingin mereka manipulasi; robot navigasi harus mengetahui lokasi mereka agar dapat menemukan jalan.

Untuk menyederhanakannya, mari kita pertimbangkan sebuah robot bergerak yang bergerak lambat di dunia 2D yang datar. Mari kita asumsikan juga bahwa robot tersebut diberikan peta lingkungan yang tepat. Pose robot bergerak tersebut ditentukan oleh dua koordinat Kartesiusnya dengan nilai x dan y , serta arahnya dengan nilai θ , seperti yang diilustrasikan pada Gambar 7.8(a). Jika kita menyusun ketiga nilai tersebut dalam sebuah vektor, maka setiap keadaan tertentu diberikan oleh $X_t = (x_t, y_t, \theta_t)^\top$. Sejauh ini, semuanya baik-baik saja.

Dalam pendekatan kinematik, setiap tindakan terdiri dari spesifikasi “seketika” dari dua kecepatan: kecepatan translasi v_t dan kecepatan rotasi ω_t . Untuk interval waktu kecil Δt , model deterministik kasar dari gerakan robot tersebut diberikan oleh

$$\hat{X}_{t+1} = f(X_t, \underbrace{v_t \omega_t}_{a_t}) = X_t + \begin{pmatrix} v_t \Delta t \cos \theta_t \\ v_t \Delta t \sin \theta_t \\ \omega_t \Delta t \end{pmatrix}$$



Gambar 7.8 (a) Model kinematik robot bergerak yang disederhanakan. Robot ditampilkan sebagai lingkaran dengan garis interior yang menandai arah maju. Keadaan x_t terdiri dari posisi (x_t, y_t) (ditampilkan secara implisit) dan orientasi θ_t . Keadaan baru x_{t+1} diperoleh dengan pembaruan posisi $v_t \Delta t$ dan orientasi $\omega_t \Delta t$.

Juga ditampilkan adalah titik acuan di (x_i, y_i) yang diamati pada waktu t . (b) Model sensor pemindaian jarak. Dua kemungkinan pose robot ditampilkan untuk pemindaian jarak tertentu (z_1, z_2, z_3, z_4) . Pose di sebelah kiri jauh lebih mungkin menghasilkan pemindaian jarak daripada pose di sebelah kanan.

Notasi $\hat{X}$ mengacu pada prediksi keadaan deterministik. Tentu saja, robot fisik agak tidak dapat diprediksi. Hal ini umumnya dimodelkan oleh distribusi Gaussian dengan rerata $f(X_t, v_t, \omega_t)$ dan kovariansi Σ_x . (Lihat Lampiran A untuk definisi matematis.)

$$P(X_{t+1} | X_t, v_t, w_t) = N(\hat{X}_{t+1}, \Sigma_x)$$

Distribusi probabilitas ini adalah model gerak robot. Distribusi ini memodelkan efek gerak a_t terhadap lokasi robot.

Selanjutnya, kita membutuhkan model sensor. Kita akan mempertimbangkan dua jenis model sensor. Model pertama mengasumsikan bahwa sensor mendeteksi fitur lingkungan yang stabil dan dapat dikenali yang disebut landmark. Untuk setiap landmark, jangkauan dan bearing dilaporkan. Misalkan keadaan robot adalah $x_t = (x_t, y_t, \theta_t)^\top$ dan robot mendeteksi landmark yang lokasinya diketahui $(x_i, y_i)^\top$. Tanpa derau, jangkauan dan bearing dapat dihitung dengan geometri sederhana. Prediksi pasti dari jangkauan dan bearing yang diamati adalah

$$\hat{z}_t = h(X_t) = \begin{pmatrix} \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \\ \arctan \frac{y_i - y_t}{x_i - x_t} - \theta_t \end{pmatrix}$$

Sekali lagi, noise mendistorsi pengukuran kami. Untuk menyederhanakannya, kita dapat mengasumsikan noise Gaussian dengan kovariansi Σ_z , yang akan memberi kita model sensor.

$$P(z_t | x_t) = N(\hat{z}_t, \Sigma_z).$$

Model sensor yang agak berbeda digunakan untuk serangkaian sensor jarak, yang masing-masing memiliki arah tetap relatif terhadap robot. Sensor tersebut menghasilkan vektor nilai jarak $z_t = (z_1, \dots, z_M)^\top$. Dengan pose x_t , misalkan $\hat{z}_j$ adalah jarak yang tepat sepanjang arah sinar ke- j dari x_t ke halangan terdekat. Seperti sebelumnya, hal ini akan terganggu oleh derau Gaussian. Biasanya, kita berasumsi bahwa kesalahan untuk arah sinar yang berbeda bersifat independen dan terdistribusi secara identik, sehingga kita memiliki

$$P(z_t | x_t) = \alpha \prod_{j=1}^M e^{-(z_j - \hat{z}_j)/2\sigma^2}$$

Gambar 7.8(b) menunjukkan contoh pemindaian jarak empat sinar dan dua kemungkinan pose robot, salah satunya cukup mungkin menghasilkan pemindaian yang diamati dan yang lainnya tidak. Dengan membandingkan model pemindaian jarak dengan model landmark, kita melihat bahwa model pemindaian jarak memiliki keuntungan karena tidak perlu mengidentifikasi landmark sebelum pemindaian jarak dapat diinterpretasikan; bahkan, pada Gambar 7.8(b), robot menghadapi dinding tanpa fitur. Di sisi lain, jika terdapat landmark yang terlihat dan dapat diidentifikasi, landmark tersebut dapat memberikan lokalisasi instan.

Bab sebelumnya menjelaskan filter Kalman, yang merepresentasikan status keyakinan sebagai Gaussian multivariat tunggal, dan filter partikel, yang merepresentasikan status keyakinan dengan kumpulan partikel yang bersesuaian dengan status tersebut. Sebagian besar algoritma lokalisasi modern menggunakan salah satu dari dua representasi keyakinan robot, yaitu $P(X_t | z_{1:t}, a_{1:t-1})$.

Lokalisasi menggunakan penyaringan partikel disebut lokalisasi Monte Carlo, atau MCL. Algoritma MCL adalah contoh algoritma penyaringan partikel. Kita hanya perlu menyediakan model gerak dan model sensor yang sesuai. Gambar 7.9 menunjukkan satu versi yang menggunakan model pemindaian jarak. Pengoperasian algoritma ini diilustrasikan pada Gambar 7.10 saat robot menemukan lokasinya di dalam gedung perkantoran. Pada gambar pertama, partikel-partikel terdistribusi secara seragam berdasarkan nilai sebelumnya, yang menunjukkan ketidakpastian global tentang posisi robot. Pada gambar kedua, rangkaian pengukuran pertama tiba dan partikel-partikel membentuk klaster di area dengan keyakinan posterior yang tinggi. Pada gambar ketiga, terdapat pengukuran yang cukup untuk mendorong semua partikel ke satu lokasi.

Filter Kalman adalah cara utama lainnya untuk melokalisasi. Filter Kalman merepresentasikan posterior $P(X_t | z_{1:t}, a_{1:t-1})$ dengan Gaussian. Rata-rata Gaussian ini akan dilambangkan dengan μ_t dan kovariansinya Σ_t . Masalah utama dengan keyakinan Gaussian adalah bahwa keyakinan tersebut hanya tertutup pada model gerak linear f dan model pengukuran linear h . Untuk f atau h nonlinier, hasil pembaruan filter umumnya tidak Gaussian. Dengan demikian, algoritma lokalisasi yang menggunakan filter Kalman melinearisasi model gerak dan sensor. Linearisasi adalah aproksimasi lokal dari fungsi nonlinier dengan fungsi linear. Gambar 7.11 mengilustrasikan konsep linearisasi untuk model gerak robot (satu dimensi). Di sebelah kiri, gambar tersebut menggambarkan model gerak nonlinier $f(x_t, a_t)$ (kontrol a_t dihilangkan dalam grafik ini karena tidak berperan dalam linearisasi).

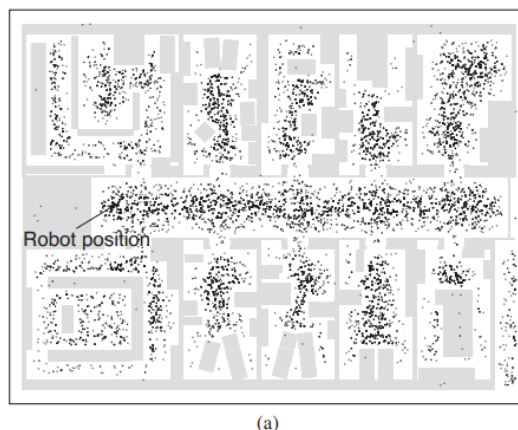
```

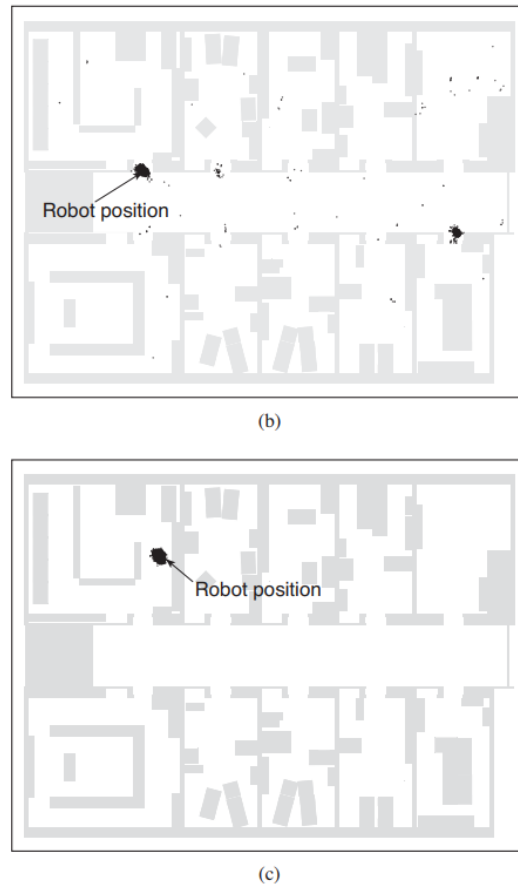
function MONTE-CARLO-LOCALIZATION ( $a, z, N, P(X'|X, v, \omega), P(z|z^*), m$ ) returns
a set of samples for the next time step
  inputs:  $a$ , robot velocities  $v$  and  $\omega$ 
            $z$ , range scan  $z_1, \dots, z_M$ 
            $P(X'|X, v, \omega)$ , motion model
            $P(z|z^*)$ , range sensor noise model
            $m$ , 2D map of the environment
  persistent:  $S$ , a vector of samples of size  $N$ 
  local variables:  $W$ , a vector of weights of size  $N$ 
                      $S$ , a temporary vector of particles of size  $N$ 
                      $W'$ , a vector of weights of size  $N$ 
  if  $S$  is empty then /* initialization phase */
    for  $i = 1$  to  $N$  do
       $S[i] \leftarrow$  sample from  $P(X_0)$ 
    for  $i = 1$  to  $N$  do /* update cycle */
       $S'[i] \leftarrow$  sample from  $P(X'|X = S[i], v, \omega)$ 
       $W'[i] \leftarrow 1$ 
      for  $j = 1$  to  $M$  do
         $z^* \leftarrow$  RAYCAST ( $j, X = S[i], m$ )
         $W'[i] \leftarrow W'[i] \cdot P(z_j|z^*)$ 
       $S \leftarrow$  WEIGHTED-SAMPLE-WITH-REPLACEMENT( $N, S', W'$ )
  return  $S$ 

```

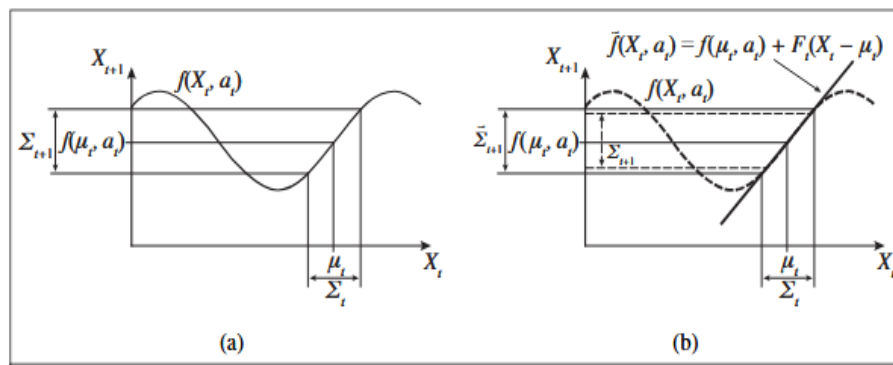
Gambar 7.9 Algoritma lokalisasi Monte Carlo menggunakan model sensor pemindaian jarak dengan noise independen.

Di sebelah kanan, fungsi ini didekati dengan fungsi linear $\tilde{f}(x_t, a_t)$. Fungsi linear ini bersinggungan dengan f di titik μ , rata-rata estimasi keadaan kita pada waktu t . Linearisasi semacam itu disebut ekspansi Taylor (derajat pertama). Filter Kalman yang melinearisasi f dan h melalui ekspansi Taylor disebut filter Kalman yang diperluas (atau EKF). Gambar 7.12 menunjukkan urutan estimasi robot yang menjalankan algoritma lokalisasi filter Kalman yang diperluas. Seiring robot bergerak, ketidakpastian estimasi lokasinya meningkat, seperti yang ditunjukkan oleh elips kesalahan.

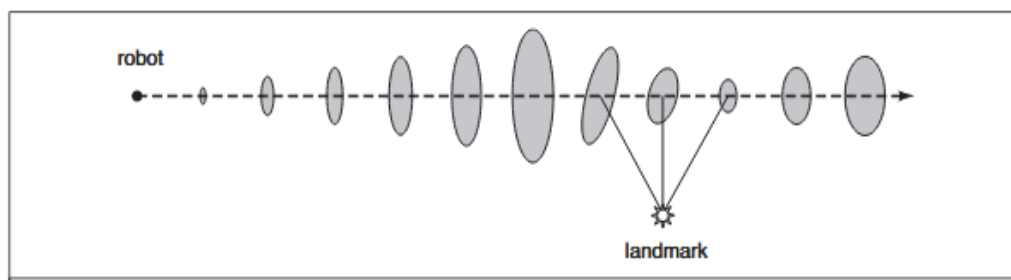




Gambar 7.10 Lokalisasi Monte Carlo, algoritma penyaringan partikel untuk lokalisasi robot bergerak. (a) Ketidakpastian global awal. (b) Ketidakpastian bimodal setelah menavigasi di koridor (simetris). (c) Ketidakpastian unimodal setelah memasuki ruangan dan menemukannya khas.



Gambar 7.11 Ilustrasi satu dimensi model gerak linier: (a) Fungsi f , dan proyeksi rata-rata μ serta interval kovariansi (berdasarkan Σ_t) ke waktu $t + 1$. (b) Versi liniernya adalah tangen f pada μ . Proyeksi rata-rata μ sudah benar. Namun, kovariansi yang diproyeksikan $\tilde{\Sigma}_{t+1}$ berbeda dari Σ_{t+1} .



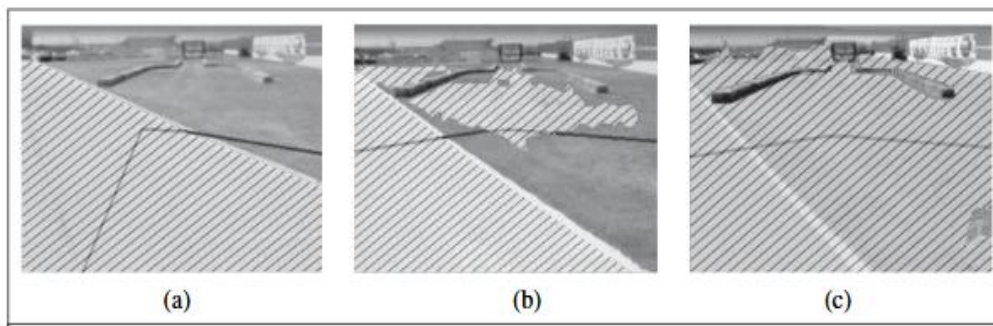
Gambar 7.12 Contoh lokalisasi menggunakan filter Kalman yang diperluas. Robot bergerak pada garis lurus. Seiring berjalannya waktu, ketidakpastiannya meningkat secara bertahap, seperti yang diilustrasikan oleh elips kesalahan. Ketika robot mengamati sebuah landmark dengan posisi yang diketahui, ketidakpastiannya berkurang.

Kesalahannya berkurang saat robot mendeteksi jarak dan bearing ke sebuah landmark dengan lokasi yang diketahui, dan meningkat lagi saat robot kehilangan jejak landmark tersebut. Algoritma EKF bekerja dengan baik jika landmark mudah diidentifikasi. Jika tidak, distribusi posterior mungkin multimodal, seperti pada Gambar 7.10(b).

Dalam beberapa situasi, peta lingkungan tidak tersedia. Kemudian, robot harus mendapatkan peta. Ini seperti masalah ayam dan telur: robot navigasi harus menentukan lokasinya relatif terhadap peta yang tidak sepenuhnya dipahaminya, sekaligus membangun peta ini meskipun ia tidak sepenuhnya mengetahui lokasi sebenarnya. Masalah ini penting untuk banyak aplikasi robot, dan telah dipelajari secara ekstensif dengan nama lokalisasi dan pemetaan simultan, disingkat SLAM. Masalah SLAM diselesaikan menggunakan berbagai teknik probabilistik, termasuk filter Kalman yang diperluas yang telah dibahas sebelumnya. Penggunaan EKF sangat mudah: cukup tambahkan vektor status untuk menyertakan lokasi landmark di lingkungan. Untungnya, pembaruan EKF berskala kuadrat, sehingga untuk peta kecil (misalnya, beberapa ratus landmark), komputasinya cukup layak. Peta yang lebih kaya sering kali diperoleh menggunakan metode relaksasi grafik, serupa dengan teknik inferensi jaringan Bayesian. Ekspektasi-maksimisasi juga digunakan untuk SLAM.

Jenis Persepsi Lainnya

Tidak semua persepsi robot berkaitan dengan lokalisasi atau pemetaan. Robot juga mempersepsi suhu, bau, sinyal akustik, dan sebagainya. Banyak dari besaran ini dapat diestimasi menggunakan varian jaringan Bayes dinamis. Yang dibutuhkan untuk estimator tersebut hanyalah distribusi probabilitas kondisional yang mengkarakterisasi evolusi variabel keadaan dari waktu ke waktu, dan model sensor yang menggambarkan hubungan pengukuran dengan variabel keadaan. Dimungkinkan juga untuk memprogram robot sebagai agen reaktif, tanpa secara eksplisit bernalar tentang distribusi probabilitas atas keadaan. Tren dalam robotika jelas mengarah pada representasi dengan semantik yang terdefinisi dengan baik.



Gambar 7.13 Urutan hasil pengklasifikasi "permukaan yang dapat dilalui" menggunakan visi adaptif. Pada (a) hanya jalan yang diklasifikasikan sebagai dapat dilalui (area bergaris). Garis gelap berbentuk V menunjukkan arah kendaraan. Pada (b) kendaraan diperintahkan untuk keluar dari jalan, menuju permukaan berumput, dan pengklasifikasi mulai mengklasifikasikan sebagian rumput sebagai dapat dilalui. Pada (c) kendaraan telah memperbarui model permukaan yang dapat dilalui agar sesuai dengan rumput sekaligus jalan.

Teknik probabilistik mengungguli pendekatan lain dalam banyak masalah persepsi yang sulit seperti lokalisasi dan pemetaan. Namun, teknik statistik terkadang terlalu rumit, dan solusi yang lebih sederhana mungkin sama efektifnya dalam praktik. Untuk membantu memutuskan pendekatan mana yang akan diambil, pengalaman bekerja dengan robot fisik nyata adalah guru terbaik Anda.

Pembelajaran Mesin dalam Persepsi Robot

Pembelajaran mesin memainkan peran penting dalam persepsi robot. Hal ini khususnya terjadi ketika representasi internal terbaik tidak diketahui. Salah satu pendekatan umum adalah memetakan aliran sensor berdimensi tinggi ke ruang berdimensi lebih rendah menggunakan metode pembelajaran mesin tanpa pengawasan. Pendekatan semacam itu disebut penanaman berdimensi rendah. Pembelajaran mesin memungkinkan untuk mempelajari model sensor dan gerakan dari data, sekaligus menemukan representasi internal yang sesuai.

Teknik pembelajaran mesin lainnya memungkinkan robot untuk terus beradaptasi dengan perubahan besar dalam pengukuran sensor. Bayangkan diri Anda berjalan dari ruang yang diterangi matahari ke ruangan gelap yang diterangi neon. Jelas, segala sesuatu di dalam ruangan lebih gelap. Namun, perubahan sumber cahaya juga memengaruhi semua warna: Cahaya neon memiliki komponen cahaya hijau yang lebih kuat daripada sinar matahari. Namun, entah bagaimana kita seolah tidak menyadari perubahan tersebut. Jika kita berjalan bersama orang-orang ke dalam ruangan berlampu neon, kita tidak menyangka wajah mereka tiba-tiba berubah menjadi hijau. Persepsi kita cepat beradaptasi dengan kondisi pencahayaan yang baru, dan otak kita mengabaikan perbedaan tersebut.

Teknik persepsi adaptif memungkinkan robot untuk menyesuaikan diri dengan perubahan tersebut. Salah satu contoh ditunjukkan pada Gambar 7.13, yang diambil dari domain mengemudi otonom. Di sini, sebuah kendaraan darat nirawak mengadaptasi pengklasifikasiannya untuk konsep "permukaan yang dapat dikendarai". Bagaimana cara kerjanya? Robot menggunakan laser untuk memberikan klasifikasi pada area kecil tepat di depan robot. Ketika area ini ditemukan datar dalam pemindaian rentang laser, area tersebut digunakan sebagai contoh pelatihan positif untuk konsep "permukaan yang dapat dikendarai". Teknik campuran Gaussian yang serupa dengan algoritma EM yang kemudian dilatih untuk mengenali koefisien warna dan tekstur spesifik dari petak sampel kecil. Gambar pada Gambar 7.13 merupakan hasil penerapan pengklasifikasi ini pada gambar penuh.

Metode yang memungkinkan robot mengumpulkan data pelatihannya sendiri (dengan label!) disebut *self-supervised*. Dalam hal ini, robot menggunakan pembelajaran mesin untuk memanfaatkan sensor jarak pendek yang berfungsi baik untuk klasifikasi medan menjadi sensor yang dapat melihat lebih jauh. Hal ini memungkinkan robot untuk bergerak lebih cepat, dan melambat hanya ketika model sensor menunjukkan adanya perubahan medan yang perlu diperiksa lebih cermat oleh sensor jarak pendek.

7.4 PERENCANAAN BERGERAK

Semua pertimbangan robot pada akhirnya mengarah pada keputusan tentang cara menggerakkan efektor. Masalah gerak titik-ke-titik adalah mengantarkan robot atau efektor ujungnya ke lokasi target yang ditentukan. Tantangan yang lebih besar adalah masalah gerakan yang sesuai, di mana robot bergerak sambil bersentuhan fisik dengan rintangan. Contoh gerakan komplans adalah manipulator robot yang memasang bola lampu, atau robot yang mendorong kotak di atas meja.

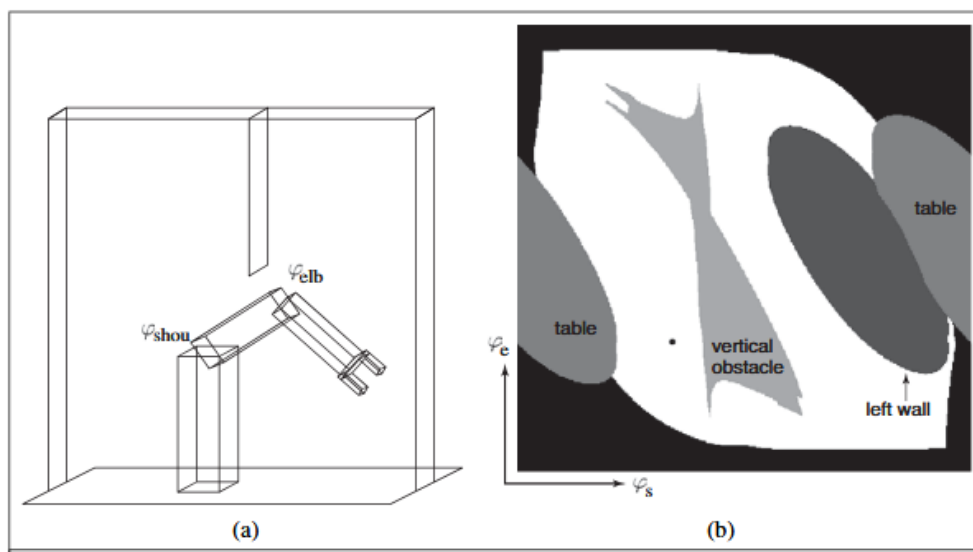
Kita mulai dengan menemukan representasi yang sesuai di mana masalah perencanaan gerak dapat dijelaskan dan dipecahkan. Ternyata ruang konfigurasi ruang keadaan robot yang ditentukan oleh lokasi, orientasi, dan sudut sendi adalah tempat yang lebih baik untuk bekerja daripada ruang 3D asli. Masalah perencanaan jalur adalah menemukan jalur dari satu konfigurasi ke konfigurasi lainnya dalam ruang konfigurasi.

Kita telah menemukan berbagai versi masalah perencanaan jalur di seluruh buku ini; komplikasi yang ditambahkan oleh robotika adalah bahwa perencanaan jalur melibatkan ruang kontinu. Ada dua pendekatan utama: dekomposisi sel dan skeletonisasi. Masing-masing mereduksi permasalahan perencanaan jalur kontinu menjadi permasalahan pencarian graf diskrit. Pada bagian ini, kami mengasumsikan bahwa gerak bersifat deterministik dan lokalisasi robot bersifat eksak. Bagian selanjutnya akan melonggarkan asumsi-asumsi ini.

Ruang Konfigurasi

Kita akan mulai dengan representasi sederhana untuk permasalahan gerak robot sederhana. Perhatikan lengan robot yang ditunjukkan pada Gambar 7.14(a). Lengan robot ini memiliki dua sendi yang bergerak secara independen. Menggerakkan sendi akan mengubah koordinat (x, y) siku dan gripper. (Lengan tidak dapat bergerak ke arah z .) Hal ini menunjukkan bahwa konfigurasi robot dapat dijelaskan dengan koordinat empat dimensi: (x_e, y_e) untuk lokasi siku relatif terhadap lingkungan dan (x_g, y_g) untuk lokasi gripper. Jelas, keempat koordinat ini mencirikan keadaan robot secara keseluruhan. Keempat koordinat ini merupakan apa yang dikenal sebagai representasi ruang kerja, karena koordinat robot ditentukan dalam sistem koordinat yang sama dengan objek yang ingin dimanipulasi (atau dihindari). Representasi ruang kerja sangat cocok untuk pemeriksaan tabrakan, terutama jika robot dan semua objek direpresentasikan oleh model poligonal sederhana.

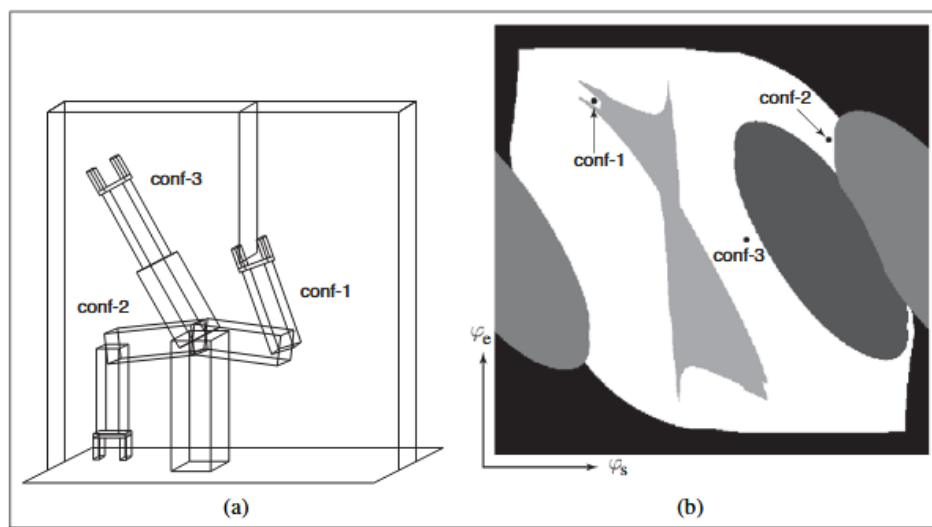
Masalah dengan representasi ruang kerja adalah tidak semua koordinat ruang kerja benar-benar dapat dicapai, bahkan tanpa adanya hambatan. Hal ini disebabkan oleh kendala keterkaitan pada ruang koordinat ruang kerja yang dapat dicapai. Misalnya, posisi siku (x_e, y_e) dan posisi gripper (x_g, y_g) selalu berjarak tetap, karena keduanya dihubungkan oleh lengan bawah yang kaku. Perencana gerak robot yang didefinisikan berdasarkan koordinat ruang kerja menghadapi tantangan untuk menghasilkan jalur yang mematuhi batasan ini. Hal ini khususnya rumit karena ruang keadaan bersifat kontinu dan batasannya nonlinier. Ternyata lebih mudah untuk merencanakan dengan representasi ruang konfigurasi. Alih-alih merepresentasikan keadaan robot dengan koordinat Kartesius elemen-elemennya, kami merepresentasikan keadaan tersebut dengan konfigurasi sendi robot. Robot contoh kami memiliki dua sendi. Oleh karena itu, kita dapat merepresentasikan keadaannya dengan dua sudut ϑ_s dan ϑ_e untuk sendi bahu dan sendi siku, masing-masing. Tanpa adanya hambatan, robot dapat dengan bebas mengambil nilai apa pun dalam ruang konfigurasi. Khususnya, saat merencanakan lintasan, seseorang cukup menghubungkan konfigurasi saat ini dan konfigurasi target dengan garis lurus. Dengan mengikuti lintasan ini, robot kemudian akan menggerakkan sendi-sendinya dengan kecepatan konstan, hingga lokasi target tercapai.



Gambar 7.14 (a) Representasi ruang kerja lengan robot dengan 2 DOFs. Ruang kerja tersebut berupa kotak dengan rintangan datar yang menggantung di langit-langit. (b) Ruang konfigurasi robot yang sama. Hanya area putih di ruang tersebut yang merupakan konfigurasi bebas tabrakan. Titik pada diagram ini sesuai dengan konfigurasi robot yang ditunjukkan di sebelah kiri.

Sayangnya, ruang konfigurasi memiliki masalahnya sendiri. Tugas robot biasanya dinyatakan dalam koordinat ruang kerja, bukan koordinat ruang konfigurasi. Hal ini menimbulkan pertanyaan tentang bagaimana memetakan antara koordinat ruang kerja dan ruang konfigurasi. Mengubah koordinat ruang konfigurasi menjadi koordinat ruang kerja sederhana: melibatkan serangkaian transformasi koordinat yang mudah. Transformasi ini bersifat linier untuk sendi prismatic dan trigonometri untuk sendi putar. Rantai transformasi koordinat ini dikenal sebagai kinematika.

Masalah invers dalam menghitung konfigurasi robot yang lokasi efekturnya ditentukan dalam koordinat ruang kerja dikenal sebagai kinematika terbalik. Menghitung kinematika terbalik itu sulit, terutama untuk robot dengan banyak DOF. Solusinya pun jarang unik. Gambar 7.14(a) menunjukkan salah satu dari dua kemungkinan konfigurasi yang menempatkan gripper di lokasi yang sama. (Konfigurasi lainnya akan menempatkan siku di bawah bahu.)



Gambar 7.15 Tiga konfigurasi robot, ditampilkan di ruang kerja dan ruang konfigurasi.

Secara umum, lengan robot dua tautan ini memiliki antara nol dan dua solusi kinematika terbalik untuk setiap set koordinat ruang kerja. Sebagian besar robot industri memiliki derajat kebebasan yang cukup untuk menemukan solusi tak terhingga untuk masalah gerak. Untuk melihat bagaimana hal ini dimungkinkan, bayangkan saja kita menambahkan sambungan putar ketiga ke robot contoh kita, yang sumbu rotasinya sejajar dengan sambungan yang ada. Dalam kasus seperti itu, kita dapat mempertahankan lokasi (tetapi bukan orientasi!) gripper tetap dan masih dapat memutar sambungan internalnya dengan bebas, untuk sebagian besar konfigurasi robot. Dengan beberapa sambungan lagi (berapa banyak?) kita dapat mencapai efek yang sama sambil menjaga orientasi gripper tetap konstan. Kita telah melihat contohnya dalam "eksperimen" meletakkan tangan di atas meja dan menggerakkan siku. Kendala kinematika posisi tangan Anda tidak cukup untuk menentukan konfigurasi siku Anda. Dengan kata lain, kinematika terbalik dari rakitan bahu-lengan Anda memiliki solusi tak terhingga.

Masalah kedua dengan representasi ruang konfigurasi muncul dari rintangan yang mungkin ada di ruang kerja robot. Contoh kami pada Gambar 7.14(a) menunjukkan beberapa rintangan tersebut, termasuk rintangan yang menggantung bebas yang menonjol ke tengah ruang kerja robot. Di ruang kerja, rintangan tersebut mengambil bentuk geometris sederhana terutama di sebagian besar buku teks robotika, yang cenderung berfokus pada rintangan poligonal. Namun, bagaimana tampilannya di ruang konfigurasi?

Gambar 7.14(b) menunjukkan ruang konfigurasi untuk robot contoh kami, di bawah konfigurasi rintangan spesifik yang ditunjukkan pada Gambar 7.14(a). Ruang konfigurasi dapat didekomposisi menjadi dua subruang: ruang semua konfigurasi yang

dapat dicapai robot, umumnya disebut ruang bebas, dan ruang konfigurasi yang tidak dapat dicapai, disebut ruang terisi. Area putih pada Gambar 7.14(b) sesuai dengan ruang bebas. Semua wilayah lainnya sesuai dengan ruang terisi.

Arsir yang berbeda dari ruang terisi sesuai dengan objek yang berbeda di ruang kerja robot; Area hitam yang mengelilingi seluruh ruang kosong sesuai dengan konfigurasi tempat robot bertabrakan dengan dirinya sendiri. Mudah untuk melihat bahwa nilai ekstrem sudut bahu atau siku menyebabkan pelanggaran tersebut. Dua area berbentuk oval di kedua sisi robot sesuai dengan meja tempat robot terpasang. Area oval ketiga sesuai dengan dinding kiri. Terakhir, objek paling menarik dalam ruang konfigurasi adalah rintangan vertikal yang menggantung di langit-langit dan menghambat gerakan robot. Objek ini memiliki bentuk yang unik dalam ruang konfigurasi: sangat nonlinier dan bahkan cekung di beberapa tempat. Dengan sedikit imajinasi, pembaca akan mengenali bentuk gripper di ujung kiri atas. Kami mendorong pembaca untuk berhenti sejenak dan mempelajari diagram ini. Bentuk rintangan ini sama sekali tidak jelas! Titik di dalam Gambar 7.14(b) menandai konfigurasi robot, seperti yang ditunjukkan pada Gambar 7.14(a). Gambar 7.15 menggambarkan tiga konfigurasi tambahan, baik di ruang kerja maupun di ruang konfigurasi. Pada konfigurasi conf-1, gripper melingkupi halangan vertikal.

Sekalipun ruang kerja robot direpresentasikan oleh poligon datar, bentuk ruang kosongnya bisa sangat rumit. Oleh karena itu, dalam praktiknya, seseorang biasanya menyelidiki ruang konfigurasi alih-alih membangunnya secara eksplisit. Seorang perencana dapat menghasilkan konfigurasi dan kemudian mengujinya untuk melihat apakah konfigurasi tersebut berada di ruang kosong dengan menerapkan kinematika robot dan kemudian memeriksa tabrakan pada koordinat ruang kerja.

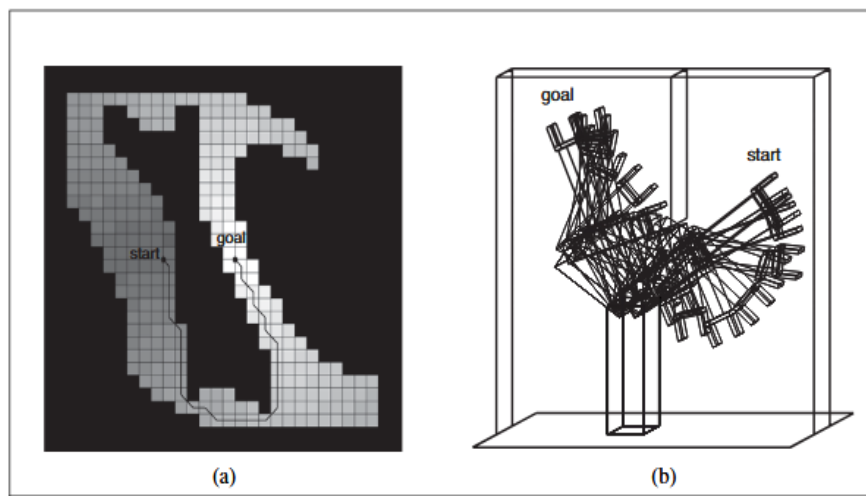
Metode Dekomposisi Sel

Pendekatan pertama untuk perencanaan jalur menggunakan dekomposisi sel yaitu, ia menguraikan ruang kosong menjadi sejumlah wilayah bersebelahan yang terbatas, yang disebut sel. Wilayah-wilayah ini memiliki sifat penting bahwa masalah perencanaan jalur dalam satu wilayah dapat diselesaikan dengan cara sederhana (misalnya, bergerak sepanjang garis lurus). Masalah perencanaan jalur kemudian menjadi masalah pencarian graf diskrit.

Dekomposisi sel yang paling sederhana terdiri dari kisi-kisi yang berjarak teratur. Gambar 7.16(a) menunjukkan dekomposisi kisi persegi dari ruang dan jalur solusi yang optimal untuk ukuran kisi ini. Arsir skala abu-abu menunjukkan nilai setiap sel kisi ruang kosong yaitu, biaya jalur terpendek dari sel tersebut ke tujuan. (Nilai-nilai ini dapat dihitung dengan bentuk deterministik dari algoritma VALUE-ITERATION). Gambar 7.16(b) menunjukkan lintasan ruang kerja yang sesuai untuk lengan. Tentu saja, kita juga dapat menggunakan algoritma A^* untuk menemukan jalur terpendek.

Dekomposisi semacam itu memiliki keuntungan karena sangat mudah diimplementasikan, tetapi juga memiliki tiga keterbatasan. Pertama, dekomposisi ini

hanya dapat diterapkan untuk ruang konfigurasi berdimensi rendah, karena jumlah sel grid meningkat secara eksponensial dengan d , jumlah dimensi. Terdengar familiar? Ini adalah kutukan @dimensionalitas. Kedua, ada masalah tentang apa yang harus dilakukan dengan sel yang "tercampur" yaitu, tidak sepenuhnya berada di dalam ruang kosong maupun sepenuhnya berada di dalam ruang terisi. Jalur solusi yang mencakup sel seperti itu mungkin bukan solusi nyata, karena mungkin tidak ada cara untuk melintasi sel ke arah yang diinginkan dalam garis lurus. Ini akan membuat perencanaan jalur menjadi tidak tepat.



Gambar 7.16 (a) Fungsi nilai dan lintasan yang ditemukan untuk aproksimasi sel grid diskrit dari ruang konfigurasi. (b) Lintasan yang sama divisualisasikan dalam koordinat ruang kerja. Perhatikan bagaimana robot menekuk sikunya untuk menghindari tabrakan dengan rintangan vertikal.

Di sisi lain, jika kita bersikeras bahwa hanya sel yang sepenuhnya bebas yang boleh digunakan, perencanaan akan menjadi tidak lengkap, karena mungkin saja satu-satunya jalur menuju tujuan melewati sel campuran terutama jika ukuran sel sebanding dengan ukuran lorong dan celah di ruang tersebut. Dan ketiga, setiap jalur melalui ruang keadaan terdiskritisasi tidak akan mulus. Umumnya sulit untuk menjamin adanya solusi mulus di dekat jalur diskrit. Jadi, robot mungkin tidak dapat mengeksekusi solusi yang ditemukan melalui dekomposisi ini.

Metode dekomposisi sel dapat ditingkatkan dengan beberapa cara untuk mengatasi beberapa masalah ini. Pendekatan pertama memungkinkan pembagian lebih lanjut dari sel campuran mungkin menggunakan sel dengan setengah ukuran aslinya. Ini dapat dilanjutkan secara rekursif hingga ditemukan lintasan yang sepenuhnya berada di dalam sel bebas. (Tentu saja, metode ini hanya berfungsi jika ada cara untuk memutuskan apakah suatu sel merupakan sel campuran, yang mudah hanya jika batas ruang konfigurasi memiliki deskripsi matematis yang relatif sederhana.) Metode ini

lengkap asalkan terdapat batas pada jalur terkecil yang harus dilalui solusi. Meskipun metode ini memfokuskan sebagian besar upaya komputasi pada area rumit di dalam ruang konfigurasi, metode ini masih gagal untuk diskalakan dengan baik ke masalah berdimensi tinggi karena setiap pemisahan sel secara rekursif menghasilkan sel 2^d yang lebih kecil. Cara kedua untuk mendapatkan algoritma yang lengkap adalah dengan menerapkan dekomposisi sel yang tepat dari ruang bebas. Metode ini harus memungkinkan sel berbentuk tidak beraturan saat bertemu batas ruang bebas, tetapi bentuknya tetap harus "sederhana" dalam artian harus mudah untuk menghitung traversal sel bebas mana pun. Teknik ini membutuhkan beberapa ide geometri yang cukup canggih, jadi kita tidak akan membahasnya lebih lanjut di sini.

Dengan memeriksa jalur solusi yang ditunjukkan pada Gambar 7.16(a), kita dapat melihat kesulitan tambahan yang harus diselesaikan. Jalur tersebut berisi sudut-sudut tajam yang sembarangan; robot yang bergerak dengan kecepatan terbatas apa pun tidak dapat mengeksekusi jalur seperti itu. Masalah ini diselesaikan dengan menyimpan nilai kontinu tertentu untuk setiap sel grid.

Pertimbangkan sebuah algoritma yang menyimpan, untuk setiap sel grid, status kontinu yang tepat yang dicapai dengan sel yang pertama kali diekspansi dalam pencarian. Asumsikan lebih lanjut, bahwa ketika menyebarkan informasi ke sel grid terdekat, kita menggunakan status kontinu ini sebagai dasar, dan menerapkan model gerak robot kontinu untuk melompat ke sel terdekat. Dengan demikian, kita sekarang dapat menjamin bahwa lintasan yang dihasilkan mulus dan memang dapat dieksekusi oleh robot. Salah satu algoritma yang menerapkan ini adalah hibrida A*.

Fungsi Biaya yang Dimodifikasi

Perhatikan bahwa pada Gambar 7.16, lintasan sangat dekat dengan rintangan. Siapa pun yang pernah mengendarai mobil tahu bahwa tempat parkir dengan jarak bebas satu milimeter di kedua sisinya bukanlah tempat parkir sama sekali; karena alasan yang sama, kami lebih menyukai lintasan solusi yang robust terhadap kesalahan gerak kecil.

Masalah ini dapat diselesaikan dengan memperkenalkan medan potensial. Medan potensial adalah fungsi yang didefinisikan di atas ruang keadaan, yang nilainya bertambah seiring bertambahnya jarak ke rintangan terdekat. Gambar 7.17(a) menunjukkan medan potensial tersebut semakin gelap suatu keadaan konfigurasi, semakin dekat ia dengan rintangan.

Medan potensial dapat digunakan sebagai suku biaya tambahan dalam perhitungan jalur terpendek. Hal ini menimbulkan tradeoff yang menarik. Di satu sisi, robot berusaha meminimalkan panjang lintasan ke tujuan. Di sisi lain, ia mencoba menghindari rintangan dengan meminimalkan fungsi potensial. Dengan keseimbangan bobot yang tepat antara kedua tujuan, jalur yang dihasilkan mungkin terlihat seperti yang ditunjukkan pada Gambar 7.17(b). Gambar ini juga menampilkan fungsi nilai yang

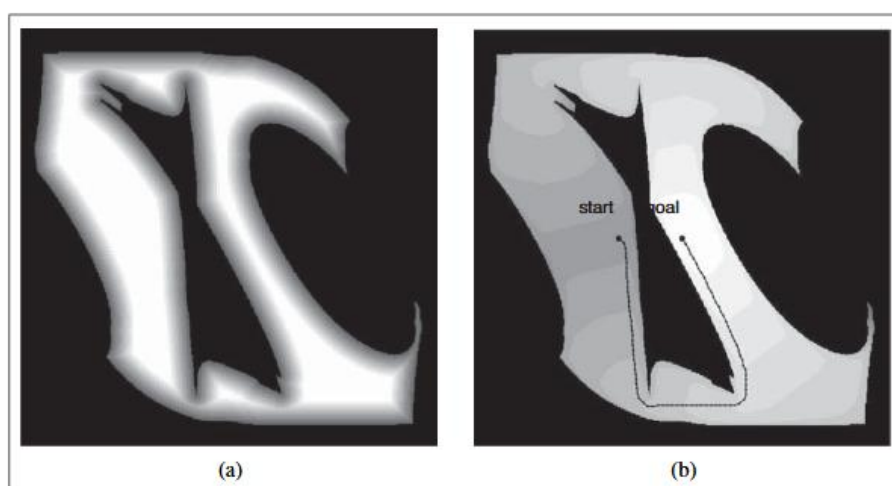
diturunkan dari fungsi biaya gabungan, yang juga dihitung dengan iterasi nilai. Jelas, jalur yang dihasilkan lebih panjang, tetapi juga lebih aman.

Ada banyak cara lain untuk memodifikasi fungsi biaya. Misalnya, mungkin diinginkan untuk menghaluskan parameter kontrol seiring waktu. Misalnya, saat mengemudikan mobil, jalur yang mulus lebih baik daripada jalur yang tersendat-sendat. Secara umum, kendala tingkat tinggi seperti itu tidak mudah diakomodasi dalam proses perencanaan, kecuali kita menjadikan perintah kemudi terbaru sebagai bagian dari status. Namun, seringkali mudah untuk menghaluskan lintasan yang dihasilkan setelah perencanaan, menggunakan metode gradien konjugat. Penghalusan pasca-perencanaan semacam itu penting dalam banyak aplikasi dunia nyata.

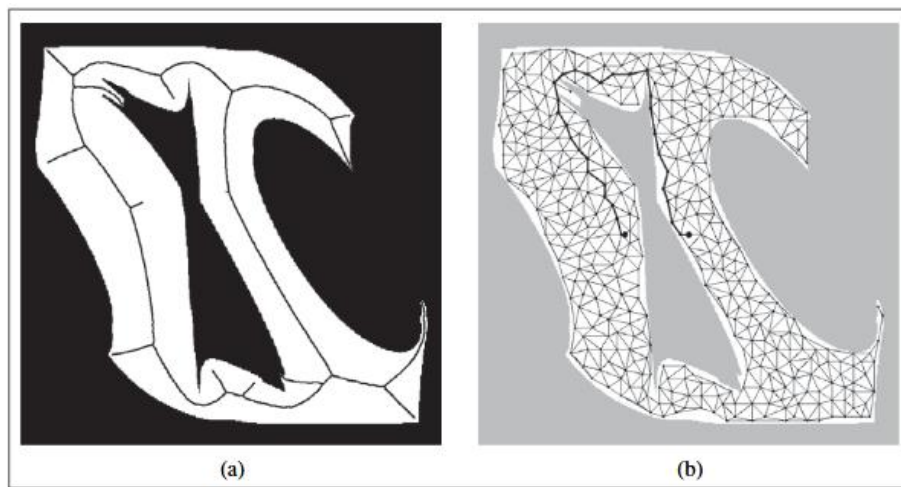
Metode Skeletonisasi

Keluarga utama kedua dari algoritma perencanaan jalur didasarkan pada gagasan kerangka. Algoritma ini mereduksi ruang kosong robot menjadi representasi satu dimensi, yang membuatnya lebih mudah untuk menyelesaikan masalah perencanaan. Representasi berdimensi lebih rendah ini disebut kerangka ruang konfigurasi.

Gambar 7.18 menunjukkan contoh kerangka: ini adalah grafik Voronoi dari ruang bebas himpunan semua titik yang berjarak sama terhadap dua atau lebih rintangan. Untuk melakukan perencanaan lintasan dengan grafik Voronoi, robot pertama-tama mengubah konfigurasinya saat ini menjadi sebuah titik pada grafik Voronoi. Mudah untuk menunjukkan bahwa hal ini selalu dapat dicapai dengan gerakan garis lurus dalam ruang konfigurasi. Kedua, robot mengikuti grafik Voronoi hingga mencapai titik terdekat dengan konfigurasi target. Terakhir, robot meninggalkan grafik Voronoi dan bergerak menuju target. Sekali lagi, langkah terakhir ini melibatkan gerakan garis lurus dalam ruang konfigurasi.



Gambar 7.17 (a) Medan potensial tolak menolak mendorong robot menjauh dari rintangan. (b) Lintasan ditemukan dengan meminimalkan panjang lintasan dan potensial secara bersamaan.



Gambar 7.18 (a) Grafik Voronoi adalah himpunan titik-titik yang berjarak sama terhadap dua atau lebih rintangan dalam ruang konfigurasi. (b) Peta jalan probabilistik, terdiri dari 400 titik yang dipilih secara acak di ruang bebas.

Dengan cara ini, permasalahan perencanaan lintasan awal disederhanakan menjadi pencarian lintasan pada graf Voronoi, yang umumnya berdimensi satu (kecuali dalam kasus-kasus non-generik tertentu) dan memiliki jumlah titik terhingga di mana tiga atau lebih kurva berdimensi satu berpotongan. Dengan demikian, menemukan lintasan terpendek sepanjang graf Voronoi merupakan permasalahan pencarian graf diskrit seperti yang dibahas dalam Bab 3 dan 4. Mengikuti graf Voronoi mungkin tidak memberikan lintasan terpendek, tetapi lintasan yang dihasilkan cenderung memaksimalkan jarak bebas. Kelemahan teknik graf Voronoi adalah sulit diterapkan pada ruang konfigurasi berdimensi lebih tinggi, dan cenderung menyebabkan detour yang terlalu besar ketika ruang konfigurasi terbuka lebar. Lebih lanjut, menghitung graf Voronoi bisa jadi sulit, terutama dalam ruang konfigurasi, di mana bentuk rintangan bisa rumit.

Alternatif untuk graf Voronoi adalah peta jalan probabilistik, sebuah pendekatan skeletonisasi yang menawarkan lebih banyak kemungkinan rute, sehingga lebih baik dalam menangani ruang terbuka lebar. Gambar 7.18(b) menunjukkan contoh peta jalan probabilistik. Grafik dibuat dengan menghasilkan sejumlah besar konfigurasi secara acak, dan membuang konfigurasi yang tidak berada dalam ruang bebas. Dua simpul dihubungkan oleh busur jika "mudah" untuk mencapai satu simpul dari simpul lainnya misalnya, dengan garis lurus di ruang bebas. Hasil dari semua ini adalah grafik acak di ruang bebas robot. Jika kita menambahkan konfigurasi awal dan tujuan robot ke grafik ini, perencanaan jalur menjadi pencarian grafik diskrit. Secara teoritis, pendekatan ini tidak lengkap, karena pilihan titik acak yang buruk dapat membuat kita kehilangan jalur

dari awal ke tujuan. Dimungkinkan untuk membatasi probabilitas kegagalan dalam hal jumlah titik yang dihasilkan dan sifat geometris tertentu dari ruang konfigurasi. Dimungkinkan juga untuk mengarahkan pembangkitan titik sampel ke area di mana pencarian parsial menunjukkan bahwa jalur yang baik dapat ditemukan, bekerja secara dua arah dari posisi awal dan tujuan. Dengan peningkatan ini, perencanaan peta jalan probabilistik cenderung berskala lebih baik ke ruang konfigurasi berdimensi tinggi dibandingkan kebanyakan teknik perencanaan jalur alternatif.

7.5 PERENCANAAN PERGERAKAN YANG TAK PASTI

Tak satu pun algoritma perencanaan gerak robot yang dibahas sejauh ini membahas karakteristik utama permasalahan robotika: ketidakpastian. Dalam robotika, ketidakpastian muncul dari observabilitas parsial lingkungan dan dari efek stokastik (atau tak termodelkan) dari tindakan robot. Kesalahan juga dapat muncul dari penggunaan algoritma aproksimasi seperti penyaringan partikel, yang tidak memberikan robot status keyakinan yang tepat meskipun sifat stokastik lingkungan dimodelkan dengan sempurna.

Sebagian besar robot saat ini menggunakan algoritma deterministik untuk pengambilan keputusan, seperti algoritma perencanaan jalur pada bagian sebelumnya. Untuk melakukannya, praktik umum adalah mengekstraksi status yang paling mungkin dari distribusi probabilitas yang dihasilkan oleh algoritma estimasi status. Keuntungan dari pendekatan ini murni komputasional. Merencanakan jalur melalui ruang konfigurasi sudah merupakan masalah yang menantang; akan lebih buruk jika kita harus bekerja dengan distribusi probabilitas penuh atas status. Mengabaikan ketidakpastian dengan cara ini berhasil ketika ketidakpastiannya kecil. Faktanya, ketika model lingkungan berubah seiring waktu akibat penggabungan pengukuran sensor, banyak robot merencanakan jalur secara daring selama pelaksanaan rencana.

Sayangnya, mengabaikan ketidakpastian tidak selalu berhasil. Dalam beberapa permasalahan, ketidakpastian robot terlalu besar: Bagaimana kita bisa menggunakan rencana jalur deterministik untuk mengendalikan robot bergerak yang tidak tahu di mana ia berada? Secara umum, jika keadaan sebenarnya robot tidak sesuai dengan aturan kemungkinan maksimum, kendali yang dihasilkan akan menjadi suboptimal. Tergantung pada besarnya kesalahan, hal ini dapat menyebabkan berbagai efek yang tidak diinginkan, seperti tabrakan dengan rintangan.

Bidang robotika telah mengadopsi berbagai teknik untuk mengakomodasi ketidakpastian. Beberapa teknik diturunkan dari algoritma untuk pengambilan keputusan dalam kondisi ketidakpastian. Jika robot hanya menghadapi ketidakpastian dalam transisi keadaannya, tetapi keadaannya sepenuhnya dapat diamati, permasalahan tersebut paling baik dimodelkan sebagai proses keputusan Markov (MDP). Solusi dari MDP adalah kebijakan optimal, yang memberi tahu robot apa yang harus dilakukan dalam setiap kemungkinan keadaan. Dengan cara ini, robot dapat menangani segala macam kesalahan gerak, sedangkan solusi jalur tunggal dari rencana deterministik akan jauh kurang robust. Dalam robotika, kebijakan disebut fungsi navigasi. Fungsi nilai

yang ditunjukkan pada Gambar 7.16(a) dapat dikonversi menjadi fungsi navigasi tersebut hanya dengan mengikuti gradien.

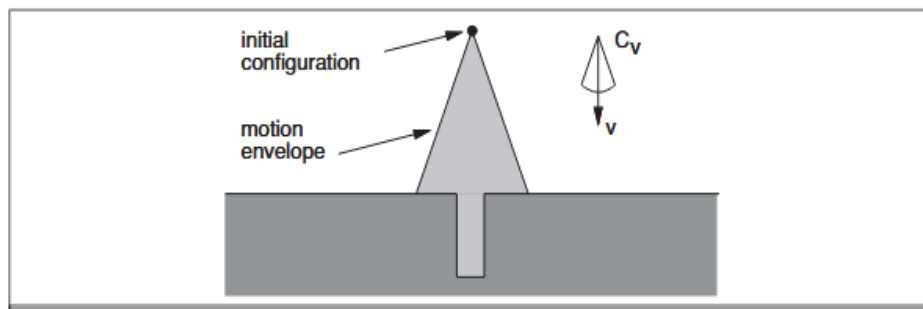
Observabilitas parsial membuat masalah ini jauh lebih sulit. Masalah kendali robot yang dihasilkan adalah MDP yang dapat diobservasi sebagian, atau POMDP. Dalam situasi seperti itu, robot mempertahankan status keyakinan internal. Solusi untuk POMDP adalah kebijakan yang didefinisikan berdasarkan status keyakinan robot. Dengan kata lain, input untuk kebijakan tersebut adalah distribusi probabilitas keseluruhan. Hal ini memungkinkan robot untuk mendasarkan keputusannya tidak hanya pada apa yang diketahuinya, tetapi juga pada apa yang tidak diketahuinya. Misalnya, jika robot tidak yakin tentang variabel status kritis, ia dapat secara rasional memanggil tindakan pengumpulan informasi. Hal ini tidak mungkin dilakukan dalam kerangka kerja MDP, karena MDP mengasumsikan observabilitas penuh. Sayangnya, teknik yang menyelesaikan POMDPs secara tepat tidak dapat diterapkan pada robotika tidak ada teknik yang diketahui untuk ruang kontinu berdimensi tinggi. Diskritisasi menghasilkan POMDP yang terlalu besar untuk ditangani.

Salah satu solusinya adalah menjadikan minimisasi ketidakpastian sebagai tujuan kontrol. Misalnya, heuristik navigasi pesisir mengharuskan robot untuk tetap berada di dekat landmark yang diketahui untuk mengurangi ketidakpastiannya. Pendekatan lain menerapkan varian metode perencanaan peta jalan probabilistik ke representasi ruang keyakinan. Metode semacam itu cenderung berskala lebih baik untuk POMDPs diskrit yang besar.

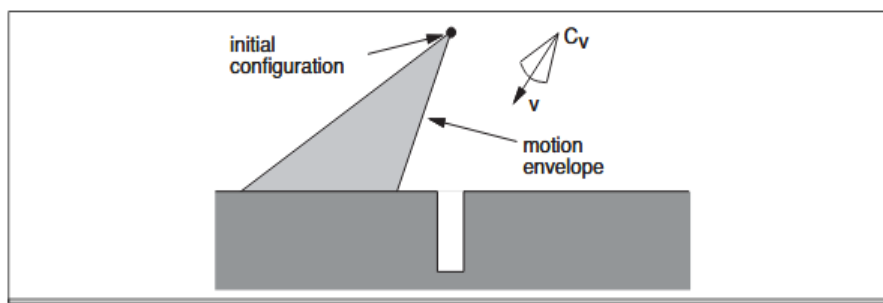
Metode Robust

Ketidakpastian juga dapat ditangani menggunakan apa yang disebut metode kontrol robust daripada metode probabilistik. Metode robust adalah metode yang mengasumsikan sejumlah ketidakpastian yang terbatas dalam setiap aspek masalah, tetapi tidak menetapkan probabilitas ke nilai-nilai dalam interval yang diizinkan. Solusi robust adalah solusi yang berfungsi terlepas dari nilai aktual yang muncul, asalkan berada dalam interval yang diasumsikan. Bentuk ekstrem dari metode robust adalah pendekatan perencanaan konforman, metode ini menghasilkan rencana yang berfungsi tanpa informasi status sama sekali.

Di sini, kita akan membahas metode robust yang digunakan untuk perencanaan gerak halus (atau FMP) dalam tugas perakitan robot. Perencanaan gerak halus melibatkan pergerakan lengan robot dalam jarak yang sangat dekat dengan objek lingkungan statis. Kesulitan utama dalam perencanaan gerak halus adalah gerakan yang diperlukan dan fitur lingkungan yang relevan sangat kecil. Pada skala sekecil itu, robot tidak dapat mengukur atau mengontrol posisinya secara akurat dan mungkin juga tidak yakin dengan bentuk lingkungan itu sendiri; kita akan berasumsi bahwa semua ketidakpastian ini terbatas. Solusi untuk masalah FMP biasanya berupa rencana atau kebijakan bersyarat yang memanfaatkan umpan balik sensor selama eksekusi dan dijamin berfungsi di semua situasi sesuai dengan batas ketidakpastian yang diasumsikan.



Gambar 7.19 Lingkungan dua dimensi, kerucut ketidakpastian kecepatan, dan amplop kemungkinan gerakan robot. Kecepatan yang dituju adalah v , tetapi dengan ketidakpastian, kecepatan aktual bisa berada di mana saja dalam C_v , menghasilkan konfigurasi akhir di suatu tempat dalam amplop gerakan, yang berarti kita tidak akan tahu apakah kita mencapai lubang atau tidak.



Gambar 7.20 Perintah gerakan pertama dan amplop kemungkinan gerakan robot yang dihasilkan. Apa pun kesalahannya, kita tahu konfigurasi akhir akan berada di sebelah kiri lubang.

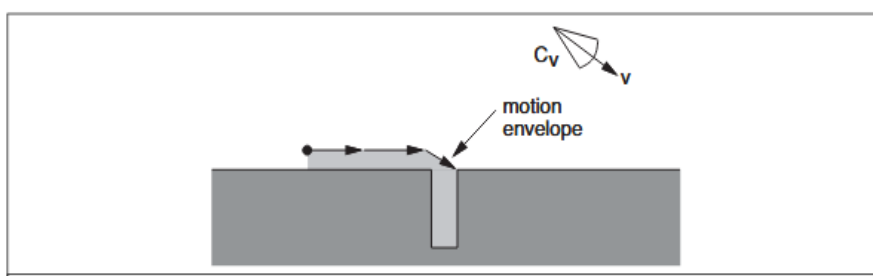
Rencana gerak halus terdiri dari serangkaian gerakan yang dijaga. Setiap gerakan yang dijaga terdiri dari (1) perintah gerakan dan (2) kondisi terminasi, yang merupakan predikat pada nilai sensor robot, dan mengembalikan true untuk menunjukkan akhir dari gerakan yang dijaga. Perintah gerakan biasanya merupakan gerakan yang sesuai yang memungkinkan efektor untuk meluncur jika perintah gerakan akan menyebabkan tabrakan dengan rintangan. Sebagai contoh, Gambar 7.19 menunjukkan ruang konfigurasi dua dimensi dengan lubang vertikal yang sempit. Itu bisa menjadi ruang konfigurasi untuk penyisipan pasak persegi panjang ke dalam lubang atau kunci mobil ke dalam kunci kontak.

Perintah gerakan adalah kecepatan konstan. Kondisi terminasi adalah kontak dengan permukaan. Untuk memodelkan ketidakpastian dalam kendali, kami berasumsi bahwa alih-alih bergerak ke arah yang diperintahkan, gerakan robot yang sebenarnya terletak pada kerucut C_v di sekitarnya. Gambar tersebut menunjukkan apa yang akan terjadi jika kami memerintahkan kecepatan langsung ke bawah dari konfigurasi awal.

Karena ketidakpastian kecepatan, robot dapat bergerak di mana saja dalam amplop kerucut, kemungkinan masuk ke dalam lubang, tetapi kemungkinan besar mendarat di salah satu sisinya. Karena robot tidak akan tahu di sisi lubang mana ia berada, ia tidak akan tahu ke arah mana harus bergerak.

Strategi yang lebih masuk akal ditunjukkan pada Gambar 7.20 dan 25.21. Pada Gambar 7.20, robot sengaja bergerak ke salah satu sisi lubang. Perintah gerak ditunjukkan pada gambar, dan uji terminasi adalah kontak dengan permukaan apa pun. Pada Gambar 7.21, perintah gerak diberikan yang menyebabkan robot meluncur di sepanjang permukaan dan masuk ke dalam lubang. Karena semua kemungkinan kecepatan dalam amplop gerak mengarah ke kanan, robot akan meluncur ke kanan setiap kali bersentuhan dengan permukaan horizontal. Robot akan meluncur ke bawah tepi vertikal kanan lubang saat menyentuhnya, karena semua kemungkinan kecepatan mengarah ke bawah relatif terhadap permukaan vertikal. Robot akan terus bergerak hingga mencapai dasar lubang, karena itulah kondisi terminasinya. Terlepas dari ketidakpastian kontrol, semua kemungkinan lintasan robot berakhir saat menyentuh dasar lubang kecuali jika ketidakrataan permukaan menyebabkan robot macet di satu tempat.

Seperti yang dapat dibayangkan, masalah membangun rencana gerak halus bukanlah hal yang sepele; faktanya, hal ini jauh lebih sulit daripada merencanakan dengan gerakan eksak. Kita dapat memilih sejumlah nilai diskrit yang tetap untuk setiap gerakan atau menggunakan geometri lingkungan untuk memilih arah yang memberikan perilaku yang berbeda secara kualitatif. Perencana gerak halus menerima input berupa deskripsi ruang konfigurasi, sudut kerucut ketidakpastian kecepatan, dan spesifikasi penginderaan apa yang memungkinkan untuk terminasi (kontak permukaan dalam kasus ini). Hal ini akan menghasilkan rencana atau kebijakan kondisional multi-langkah yang dijamin berhasil, jika rencana tersebut ada.



Gambar 7.21 Perintah gerakan kedua dan cakupan kemungkinan gerakan. Sekalipun ada kesalahan, pada akhirnya kita akan terjebak dalam lubang.

Contoh kami mengasumsikan bahwa perencana memiliki model lingkungan yang eksak, tetapi kesalahan terbatas dapat ditoleransi dalam model ini sebagai berikut. Jika kesalahan dapat dideskripsikan dalam bentuk parameter, parameter tersebut dapat

ditambahkan sebagai derajat kebebasan ke ruang konfigurasi. Pada contoh terakhir, jika kedalaman dan lebar lubang tidak pasti, kita dapat menambahkannya sebagai dua derajat kebebasan ke ruang konfigurasi. Mustahil untuk menggerakkan robot ke arah ini dalam ruang konfigurasi atau merasakan posisinya secara langsung. Namun, kedua batasan tersebut dapat dimasukkan ketika menggambarkan masalah ini sebagai masalah FMP dengan menentukan ketidakpastian kontrol dan sensor secara tepat. Hal ini menghasilkan masalah perencanaan empat dimensi yang kompleks, tetapi teknik perencanaan yang sama persis dapat diterapkan.

Pendekatan robust semacam ini menghasilkan rencana yang dirancang untuk hasil terburuk, alih-alih memaksimalkan kualitas rencana yang diharapkan. Rencana terburuk optimal dalam pengertian teori keputusan hanya jika kegagalan selama eksekusi jauh lebih parah daripada biaya lain yang terlibat dalam eksekusi.

7.6 BERGERAK

Sejauh ini, kita telah membahas cara merencanakan gerakan, tetapi belum membahas cara bergerak. Rencana kita terutama yang dihasilkan oleh perencana jalur deterministik menganggap robot dapat mengikuti jalur apa pun yang dihasilkan algoritma. Di dunia nyata, tentu saja, hal ini tidak berlaku. Robot memiliki inersia dan tidak dapat mengeksekusi jalur sembarangan kecuali pada kecepatan yang sangat rendah. Dalam kebanyakan kasus, robot dapat mengerahkan gaya alih-alih menentukan posisi. Bagian ini membahas metode untuk menghitung gaya-gaya ini.

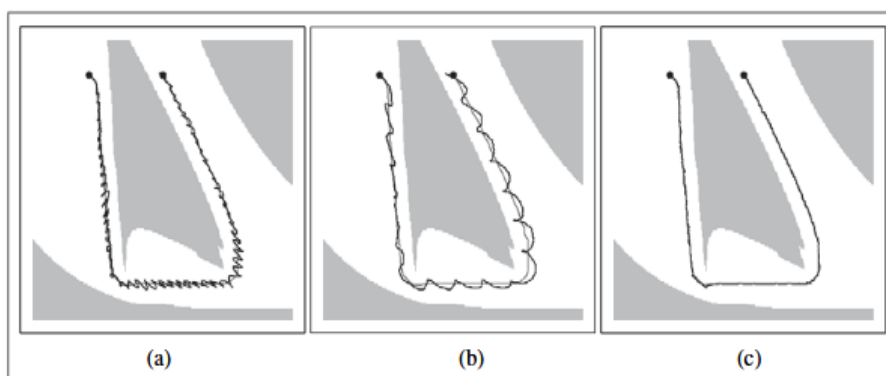
Dinamika dan Kontrol

Bagian 7.2 memperkenalkan gagasan keadaan dinamis, yang memperluas keadaan kinematik robot melalui kecepatannya. Misalnya, selain sudut sendi robot, keadaan dinamis juga mencakup laju perubahan sudut, dan bahkan mungkin percepatan sesaatnya. Model transisi untuk representasi keadaan dinamis mencakup pengaruh gaya terhadap laju perubahan ini. Model-model semacam itu biasanya diekspresikan melalui persamaan diferensial, yaitu persamaan yang menghubungkan suatu besaran (misalnya, keadaan kinematik) dengan perubahan besaran tersebut terhadap waktu (misalnya, kecepatan). Pada prinsipnya, kita bisa saja memilih untuk merencanakan gerak robot menggunakan model dinamis, alih-alih model kinematik kita sendiri. Metodologi semacam itu akan menghasilkan kinerja robot yang superior, jika kita dapat menghasilkan rencana tersebut. Namun, keadaan dinamis memiliki dimensi yang lebih tinggi daripada ruang kinematik, dan kutukan dimensionalitas akan membuat banyak algoritma perencanaan gerak tidak dapat diterapkan untuk semua robot kecuali yang paling sederhana. Karena alasan ini, sistem robot praktis sering kali mengandalkan perencana jalur kinematik yang lebih sederhana.

Teknik umum untuk mengkompensasi keterbatasan rencana kinematik adalah dengan menggunakan mekanisme terpisah, yaitu pengontrol, untuk menjaga robot tetap pada jalurnya. Pengontrol adalah teknik untuk menghasilkan kendali robot secara

waktu nyata (real-time) menggunakan umpan balik dari lingkungan, sehingga mencapai tujuan kendali. Jika tujuannya adalah untuk menjaga robot tetap pada jalur yang telah direncanakan sebelumnya, maka robot tersebut sering disebut sebagai pengontrol referensi dan jalurnya disebut jalur referensi. Pengendali yang mengoptimalkan fungsi biaya global dikenal sebagai pengendali optimal. Kebijakan optimal untuk MDP kontinu, pada dasarnya, adalah pengendali optimal.

Sekilas, masalah menjaga robot tetap pada jalur yang telah ditentukan sebelumnya tampak relatif mudah. Namun, dalam praktiknya, masalah yang tampaknya sederhana ini pun memiliki jebakan. Gambar 7.22(a) mengilustrasikan apa yang bisa salah; gambar ini menunjukkan jalur robot yang mencoba mengikuti jalur kinematik. Setiap kali terjadi deviasi—baik karena derau maupun kendala pada gaya yang dapat diterapkan robot—robot memberikan gaya berlawanan yang besarnya proporsional dengan deviasi ini. Secara intuitif, hal ini mungkin tampak masuk akal, karena deviasi harus dikompensasi oleh gaya tandingan agar robot tetap pada jalurnya. Namun, seperti yang diilustrasikan Gambar 7.22(a), pengendali kami menyebabkan robot bergetar cukup hebat. Getaran tersebut merupakan hasil dari inersia alami lengan robot: setelah didorong kembali ke posisi referensinya, robot kemudian melampaui batas, yang menyebabkan kesalahan simetris dengan tanda berlawanan. Pelarian yang melampaui batas tersebut dapat berlanjut di sepanjang lintasan, dan gerakan robot yang dihasilkan jauh dari yang diinginkan.



Gambar 7.22 Kontrol lengan robot menggunakan (a) kontrol proporsional dengan faktor penguatan 1,0, (b) kontrol proporsional dengan faktor penguatan 0,1, dan (c) kontrol PD (turunan proporsional) dengan faktor penguatan 0,3 untuk komponen proporsional dan 0,8 untuk komponen diferensial. Dalam semua kasus, lengan robot mencoba mengikuti jalur yang ditunjukkan dengan warna abu-abu.

Sebelum kita dapat mendefinisikan pengontrol yang lebih baik, mari kita jelaskan secara formal apa yang salah. Pengontrol yang memberikan gaya dalam proporsi negatif terhadap kesalahan yang diamati dikenal sebagai pengontrol P. Huruf 'P' berarti proporsional, yang menunjukkan bahwa kontrol aktual proporsional dengan kesalahan

manipulator robot. Secara lebih formal, misalkan $y(t)$ adalah jalur referensi, yang diparameterisasi oleh indeks waktu t . Kontrol a_t pada yang dihasilkan oleh pengontrol P memiliki bentuk:

$$a_t = K_p(y(t) - x_t).$$

Di sini x_t adalah keadaan robot pada waktu t dan K_p adalah konstanta yang dikenal sebagai parameter penguatan pengontrol dan nilainya disebut faktor penguatan); K_p mengatur seberapa kuat pengontrol mengoreksi deviasi antara keadaan aktual x_t dan yang diinginkan $y(t)$. Dalam contoh kita, $K_p = 1$. Sekilas, orang mungkin berpikir bahwa memilih nilai yang lebih kecil untuk K_p akan menyelesaikan masalah. Sayangnya, ini tidak terjadi. Gambar 7.22(b) menunjukkan lintasan untuk $K_p = .1$, yang masih menunjukkan perilaku osilasi. Nilai parameter penguatan yang lebih rendah mungkin hanya memperlambat osilasi, tetapi tidak menyelesaikan masalah. Faktanya, tanpa adanya gesekan, pengontrol P pada dasarnya adalah hukum pegas; sehingga akan berosilasi tanpa batas di sekitar lokasi target yang tetap.

Secara tradisional, masalah jenis ini termasuk dalam ranah teori kontrol, bidang yang semakin penting bagi para peneliti dalam AI. Penelitian selama puluhan tahun di bidang ini telah menghasilkan sejumlah besar pengontrol yang lebih unggul daripada hukum kontrol sederhana yang dijelaskan di atas. Khususnya, pengontrol referensi dikatakan stabil jika gangguan kecil menyebabkan kesalahan terbatas antara robot dan sinyal referensi. Pengontrol dikatakan stabil secara ketat jika mampu kembali ke dan kemudian tetap berada di jalur referensinya setelah gangguan tersebut. Pengontrol P kami tampaknya stabil tetapi tidak sepenuhnya stabil, karena gagal untuk tetap berada di dekat lintasan referensinya.

Pengendali paling sederhana yang mencapai stabilitas ketat dalam domain kami adalah pengontrol PD. Huruf 'P' juga merupakan singkatan dari proporsional, dan 'D' merupakan singkatan dari derivatif. Pengontrol PD dijelaskan oleh persamaan berikut:

$$a_t = K_p(y(t) - x_t) + K_D \frac{\partial(y(t) - x_t)}{\partial t} \quad (7.2)$$

Sebagaimana ditunjukkan oleh persamaan ini, pengontrol PD memperluas pengontrol P dengan komponen diferensial, yang menambahkan nilai a_t pada suku yang proporsional dengan turunan pertama galat $y(t) - x_t$ terhadap waktu. Apa efek dari suku tersebut? Secara umum, suku turunan meredam sistem yang dikontrol. Untuk melihat hal ini, pertimbangkan situasi di mana galat $(y(t) - x_t)$ berubah dengan cepat seiring waktu, seperti halnya pengontrol P kita di atas. Turunan galat ini kemudian akan menetralkan suku proporsional, yang akan mengurangi respons keseluruhan terhadap gangguan.

Namun, jika galat yang sama tetap ada dan tidak berubah, turunan akan hilang dan suku proporsional mendominasi pilihan kontrol.

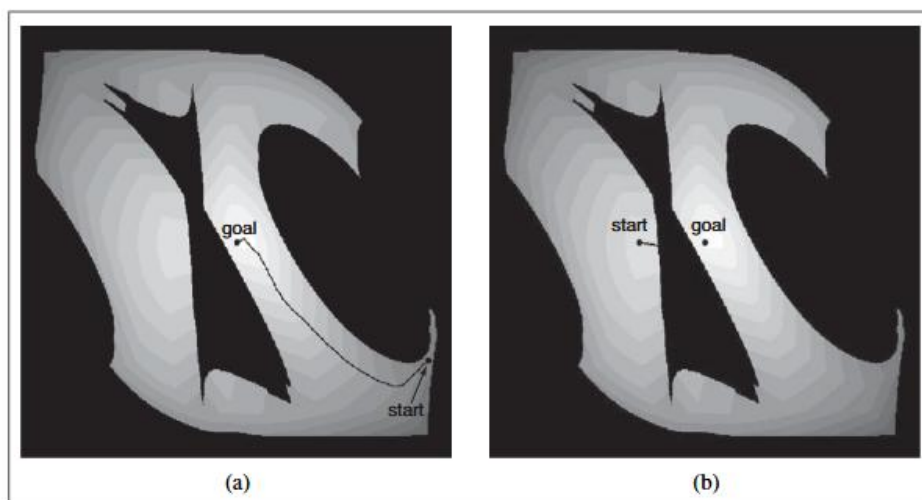
Gambar 7.22(c) menunjukkan hasil penerapan pengontrol PD ini pada lengan robot kita, dengan menggunakan parameter penguatan $K_P = .3$ dan $K_D = .8$. Jelas, jalur yang dihasilkan jauh lebih mulus, dan tidak menunjukkan osilasi yang nyata. Namun, pengontrol PD memiliki mode kegagalan. Khususnya, pengontrol PD mungkin gagal mengatur kesalahan hingga nol, bahkan tanpa adanya gangguan eksternal. Seringkali situasi seperti itu merupakan akibat dari gaya eksternal sistematis yang bukan bagian dari model. Mobil otonom yang melaju di permukaan miring, misalnya, mungkin mendapati dirinya secara sistematis tertarik ke satu sisi. Keausan pada lengan robot menyebabkan kesalahan sistematis serupa. Dalam situasi seperti itu, umpan balik yang terlalu proporsional diperlukan untuk mendorong kesalahan mendekati nol. Solusi untuk masalah ini terletak pada penambahan suku ketiga pada hukum kontrol, berdasarkan kesalahan terintegrasi dari waktu ke waktu:

$$a_t = K_P(y(t) - x_t) + K_I \int (y(t) - x_t)dt + K_D \frac{\partial(y(t) - x_t)}{\partial t} \quad (7.3)$$

Di sini, K_I adalah parameter penguatan lainnya. Suku $\int (y(t) - x_t)dt$ menghitung integral kesalahan terhadap waktu. Efek dari suku ini adalah deviasi jangka panjang antara sinyal referensi dan keadaan aktual terkoreksi. Jika, misalnya, x_t lebih kecil dari $y(t)$ untuk jangka waktu yang lama, integral ini akan bertambah hingga kontrol yang dihasilkan pada memaksa kesalahan ini mengecil. Suku integral, dengan demikian, memastikan bahwa pengontrol tidak menunjukkan kesalahan sistematis, dengan mengorbankan peningkatan risiko perilaku osilasi. Pengontrol dengan ketiga suku ini disebut pengontrol PID (untuk turunan integral proporsional). Pengontrol PID banyak digunakan dalam industri, untuk berbagai masalah kontrol.

Kontrol Medan Potensial

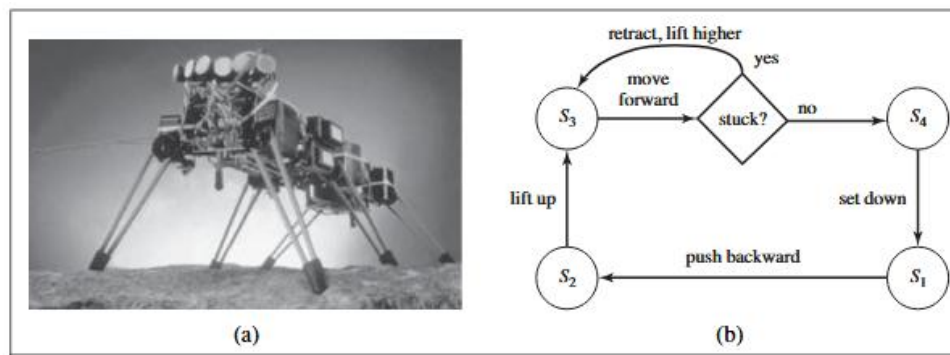
Kami memperkenalkan medan potensial sebagai fungsi biaya tambahan dalam perencanaan gerak robot, tetapi medan potensial juga dapat digunakan untuk menghasilkan gerak robot secara langsung, sehingga menghilangkan fase perencanaan jalur sepenuhnya. Untuk mencapai hal ini, kita harus mendefinisikan gaya tarik yang menarik robot menuju konfigurasi tujuannya dan medan potensial tolak yang mendorong robot menjauh dari rintangan. Medan potensial tersebut ditunjukkan pada Gambar 7.23. Nilai minimum global tunggalnya adalah konfigurasi tujuan, dan nilainya adalah jumlah jarak ke konfigurasi tujuan ini dan kedekatannya dengan rintangan.



Gambar 7.23 Kendali medan potensial. Robot mendaki medan potensial yang terdiri dari gaya tolak yang berasal dari rintangan dan gaya tarik yang sesuai dengan konfigurasi tujuan. (a) Jalur yang berhasil. (b) Optimal lokal.

Tidak ada perencanaan yang terlibat dalam menghasilkan medan potensial yang ditunjukkan pada gambar. Karena itu, medan potensial sangat cocok untuk kontrol waktu nyata. Gambar 7.23(a) menunjukkan lintasan robot yang melakukan pendakian bukit di medan potensial. Dalam banyak aplikasi, medan potensial dapat dihitung secara efisien untuk konfigurasi apa pun. Lebih lanjut, mengoptimalkan potensial sama dengan menghitung gradien potensial untuk konfigurasi robot saat ini. Perhitungan ini bisa sangat efisien, terutama jika dibandingkan dengan algoritma perencanaan jalur, yang semuanya bersifat eksponensial dalam dimensionalitas ruang konfigurasi (DOF) dalam kasus terburuk.

Fakta bahwa pendekatan medan potensial berhasil menemukan jalur menuju tujuan dengan cara yang efisien, bahkan dalam jarak yang jauh dalam ruang konfigurasi, menimbulkan pertanyaan apakah perencanaan dalam robotika memang diperlukan. Apakah teknik medan potensial sudah memadai, atau kita hanya beruntung dalam contoh kita? Jawabannya adalah kita memang beruntung. Medan potensial memiliki banyak nilai minimum lokal yang dapat menjebak robot. Pada Gambar 7.23(b), robot mendekati halangan hanya dengan memutar sendi bahunya, hingga terjebak di sisi halangan yang salah. Medan potensial tidak cukup kaya untuk membuat robot menekuk sikunya sehingga lengannya pas di bawah halangan. Dengan kata lain, kendali medan potensial sangat baik untuk gerakan robot lokal, tetapi terkadang kita masih membutuhkan perencanaan global. Kelemahan penting lainnya dari medan potensial adalah gaya yang dihasilkannya hanya bergantung pada posisi halangan dan robot, bukan pada kecepatan robot. Dengan demikian, kendali medan potensial sebenarnya merupakan metode kinematik dan dapat gagal jika robot bergerak cepat.



Gambar 7.24 (a) Genghis, robot heksapoda. (b) Mesin keadaan terbatas tertambah (AFSM) untuk mengendalikan satu kaki. Perhatikan bahwa AFSM ini bereaksi terhadap umpan balik sensor: jika satu kaki tersangkut selama fase ayunan ke depan, kaki tersebut akan terangkat semakin tinggi.

Kontrol Reaktif

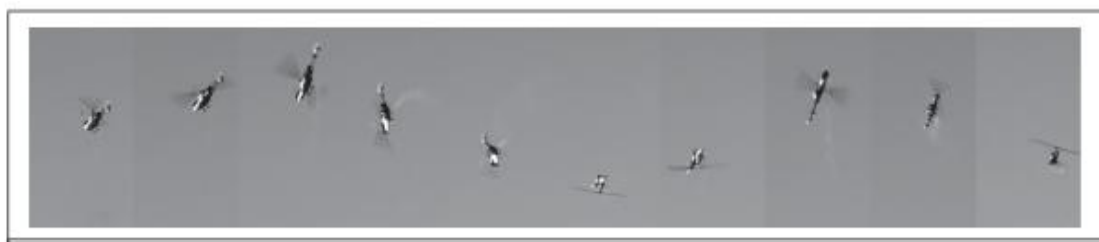
Sejauh ini kita telah mempertimbangkan keputusan kontrol yang memerlukan beberapa model lingkungan untuk membangun jalur referensi atau medan potensial. Ada beberapa kesulitan dengan pendekatan ini. Pertama, model yang cukup akurat seringkali sulit diperoleh, terutama di lingkungan yang kompleks atau terpencil, seperti permukaan Mars, atau untuk robot yang memiliki sedikit sensor. Kedua, bahkan dalam kasus di mana kita dapat merancang model dengan akurasi yang memadai, kesulitan komputasi dan kesalahan lokalisasi dapat membuat teknik ini tidak praktis. Dalam beberapa kasus, arsitektur agen refleks menggunakan kontrol reaktif lebih tepat.

Sebagai contoh, bayangkan sebuah robot berkaki yang mencoba mengangkat kakinya melewati rintangan. Kita dapat memberi robot ini aturan yang mengatakan angkat kaki sedikit setinggi h dan gerakkan ke depan, dan jika kaki tersebut menemui rintangan, gerakkan ke belakang dan mulai lagi pada ketinggian yang lebih tinggi. Anda dapat mengatakan bahwa h memodelkan suatu aspek dunia, tetapi kita juga dapat menganggap h sebagai variabel bantu dari pengontrol robot, tanpa makna fisik langsung.

Salah satu contohnya adalah robot berkaki enam (heksapoda), yang ditunjukkan pada Gambar 7.24(a), yang dirancang untuk berjalan di medan kasar. Sensor robot tidak memadai untuk mendapatkan model medan untuk perencanaan jalur. Selain itu, meskipun kita menambahkan sensor yang cukup akurat, dua belas derajat kebebasan (dua untuk setiap kaki) akan membuat masalah perencanaan jalur yang dihasilkan sulit dihitung secara komputasi. Meskipun demikian, dimungkinkan untuk menentukan pengontrol secara langsung tanpa model lingkungan yang eksplisit. (Kita telah melihat hal ini dengan pengontrol PD, yang mampu menjaga lengan robot yang kompleks tetap pada target tanpa model dinamika robot yang eksplisit; namun, pengontrol ini memerlukan jalur referensi yang dihasilkan dari model kinematik.) Untuk robot heksapoda, pertama-tama kita memilih gaya berjalan, atau pola pergerakan anggota

badan. Salah satu gaya berjalan yang stabil secara statis adalah dengan terlebih dahulu menggerakkan kaki kanan depan, kanan belakang, dan kiri tengah ke depan (dengan menjaga ketiga kaki lainnya tetap diam), lalu menggerakkan ketiga kaki lainnya. Gaya berjalan ini bekerja dengan baik di medan datar. Di medan yang terjal, rintangan dapat mencegah kaki berayun ke depan. Masalah ini dapat diatasi dengan aturan kontrol yang sangat sederhana: ketika gerakan maju kaki terhambat, cukup tarik kembali, angkat lebih tinggi, dan coba lagi. Pengontrol yang dihasilkan ditunjukkan pada Gambar 7.24(b) sebagai mesin berhingga; pengontrol ini merupakan agen refleks dengan keadaan, di mana keadaan internal direpresentasikan oleh indeks keadaan mesin saat ini (s_1 hingga s_4).

Varian pengontrol berbasis umpan balik sederhana ini telah ditemukan menghasilkan pola berjalan yang sangat tangguh, yang mampu mengendalikan robot di medan yang terjal. Jelas, pengontrol semacam itu bebas model, dan tidak sengaja atau menggunakan pencarian untuk menghasilkan kontrol. Umpan balik lingkungan memainkan peran penting dalam eksekusi pengontrol. Perangkat lunak saja tidak menentukan apa yang sebenarnya akan terjadi ketika robot ditempatkan di suatu lingkungan. Perilaku yang muncul melalui interaksi pengontrol (sederhana) dan lingkungan (kompleks) sering disebut sebagai perilaku emergen. Secara tegas, semua robot yang dibahas dalam bab ini menunjukkan perilaku emergen, karena tidak ada model yang sempurna. Namun, secara historis, istilah ini hanya digunakan untuk teknik kontrol yang tidak menggunakan model lingkungan eksplisit. Perilaku emergen juga merupakan karakteristik organisme biologis.



Gambar 7.25 Beberapa eksposur helikopter RC yang melakukan flip berdasarkan kebijakan yang dipelajari dengan pembelajaran penguatan. Gambar milik Andrew Ng, Universitas Stanford.

Kontrol Pembelajaran Penguatan

Salah satu bentuk kontrol yang sangat menarik didasarkan pada bentuk pencarian kebijakan dari pembelajaran penguatan. Karya ini sangat berpengaruh dalam beberapa tahun terakhir, karena telah memecahkan masalah robotika yang menantang yang sebelumnya tidak ada solusinya. Contohnya adalah penerbangan helikopter otonom akrobatik. Gambar 7.25 menunjukkan jungkir balik otonom dari helikopter RC (*radio-controlled*) kecil. Manuver ini menantang karena sifat aerodinamika yang sangat

nonlinier. Hanya pilot manusia yang paling berpengalaman yang mampu melakukannya. Namun, sebuah metode pencarian kebijakan, hanya dengan beberapa menit komputasi, mempelajari kebijakan yang dapat mengeksekusi jungkir balik dengan aman setiap saat.

Pencarian kebijakan membutuhkan model domain yang akurat sebelum dapat menemukan kebijakan. Masukan ke model ini adalah keadaan helikopter pada waktu t , kontrol pada waktu t , dan keadaan yang dihasilkan pada waktu $t + \Delta t$. Kondisi helikopter dapat dijelaskan oleh koordinat 3D kendaraan, sudut yaw, pitch, dan roll-nya, serta laju perubahan keenam variabel ini. Kontrolnya merupakan kontrol manual helikopter: throttle, pitch, elevator, aileron, dan rudder. Yang tersisa hanyalah kondisi yang dihasilkan bagaimana kita akan mendefinisikan model yang secara akurat menjelaskan respons helikopter terhadap setiap kontrol? Jawabannya sederhana: Biarkan seorang pilot manusia ahli menerbangkan helikopter, dan merekam kontrol yang ditransmisikan oleh ahli tersebut melalui radio dan variabel kondisi helikopter. Sekitar empat menit penerbangan yang dikendalikan manusia cukup untuk membangun model prediktif yang cukup akurat untuk mensimulasikan kendaraan.

7.7 ARSITEKTUR PERANGKAT LUNAK ROBOTIK

Metodologi untuk menyusun algoritma disebut arsitektur perangkat lunak. Arsitektur mencakup bahasa dan perangkat untuk menulis program, serta filosofi umum tentang bagaimana program dapat disatukan.

Arsitektur perangkat lunak modern untuk robotika harus memutuskan bagaimana menggabungkan kontrol reaktif dan perencanaan deliberatif berbasis model. Dalam banyak hal, teknik reaktif dan deliberatif memiliki kekuatan dan kelemahan yang ortogonal. Kontrol reaktif digerakkan oleh sensor dan sesuai untuk membuat keputusan tingkat rendah secara waktu nyata (real-time). Namun, jarang menghasilkan solusi yang masuk akal di tingkat global, karena keputusan kontrol global bergantung pada informasi yang tidak dapat dirasakan pada saat pengambilan keputusan. Untuk masalah seperti itu, perencanaan deliberatif merupakan pilihan yang lebih tepat.

Akibatnya, sebagian besar arsitektur robot menggunakan teknik reaktif pada tingkat kontrol yang lebih rendah dan teknik deliberatif pada tingkat yang lebih tinggi. Kami menemukan kombinasi semacam itu dalam diskusi kami tentang pengontrol PD, di mana kami menggabungkan pengontrol PD (reaktif) dengan perencanaan jalur (deliberatif). Arsitektur yang menggabungkan teknik reaktif dan deliberatif disebut arsitektur hibrida.

Arsitektur Subsumsi

Arsitektur subsumsi adalah kerangka kerja untuk merakit pengontrol reaktif dari mesin berstatus hingga. Node dalam mesin ini dapat berisi pengujian untuk variabel sensor tertentu, yang dalam hal ini jejak eksekusi mesin berstatus hingga dikondisikan pada hasil pengujian tersebut. Busur dapat ditandai dengan pesan yang akan dihasilkan saat melintasinya, dan yang dikirim ke motor robot atau ke mesin berstatus hingga

lainnya. Selain itu, mesin berstatus hingga memiliki pengatur waktu internal (clock) yang mengontrol waktu yang dibutuhkan untuk melintasi busur. Mesin yang dihasilkan disebut sebagai mesin berstatus hingga ditambah, atau AFSM, di mana penambahan mengacu pada penggunaan clock. Contoh AFSM sederhana adalah mesin empat-keadaan yang ditunjukkan pada Gambar 7.24(b), yang menghasilkan gerakan kaki siklik untuk alat bantu jalan heksapoda. AFSM ini mengimplementasikan pengontrol siklik, yang eksekusinya sebagian besar tidak bergantung pada umpan balik lingkungan. Namun, fase ayunan ke depan bergantung pada umpan balik sensor. Jika kaki macet, artinya gagal melakukan ayunan ke depan, robot akan menarik kembali kaki tersebut, mengangkatnya sedikit lebih tinggi, dan mencoba melakukan ayunan ke depan sekali lagi. Dengan demikian, pengontrol dapat bereaksi terhadap kontingensi yang muncul dari interaksi robot dan lingkungannya.

Arsitektur subsumsi menawarkan primitif tambahan untuk menyinkronkan AFSM, dan untuk menggabungkan nilai keluaran dari beberapa AFSM yang mungkin bertentangan. Dengan cara ini, memungkinkan pemrogram untuk menyusun pengontrol yang semakin kompleks secara bottom-up. Dalam contoh kita, kita bisa mulai dengan AFSM untuk masing-masing kaki, diikuti oleh AFSM untuk mengoordinasikan beberapa kaki. Selain itu, kita bisa menerapkan perilaku tingkat tinggi seperti penghindaran tabrakan, yang mungkin melibatkan mundur dan berbelok.

Ide menyusun pengontrol robot dari AFSM cukup menarik. Bayangkan betapa sulitnya menghasilkan perilaku yang sama dengan salah satu algoritma perencanaan jalur ruang konfigurasi yang dijelaskan di bagian sebelumnya. Pertama, kita membutuhkan model medan yang akurat. Ruang konfigurasi robot dengan enam kaki, yang masing-masing digerakkan oleh dua motor independen, berjumlah delapan belas dimensi (dua belas dimensi untuk konfigurasi kaki, dan enam untuk lokasi dan orientasi robot relatif terhadap lingkungannya). Sekalipun komputer kita cukup cepat untuk menemukan jalur di ruang berdimensi tinggi seperti itu, kita harus mengkhawatirkan efek buruk seperti robot meluncur menuruni lereng. Karena efek stokastik tersebut, satu jalur melalui ruang konfigurasi hampir pasti terlalu rapuh, dan bahkan pengontrol PID mungkin tidak mampu mengatasi kontingensi tersebut. Dengan kata lain, menghasilkan perilaku gerak secara sengaja merupakan masalah yang terlalu rumit untuk algoritma perencanaan gerak robot masa kini.

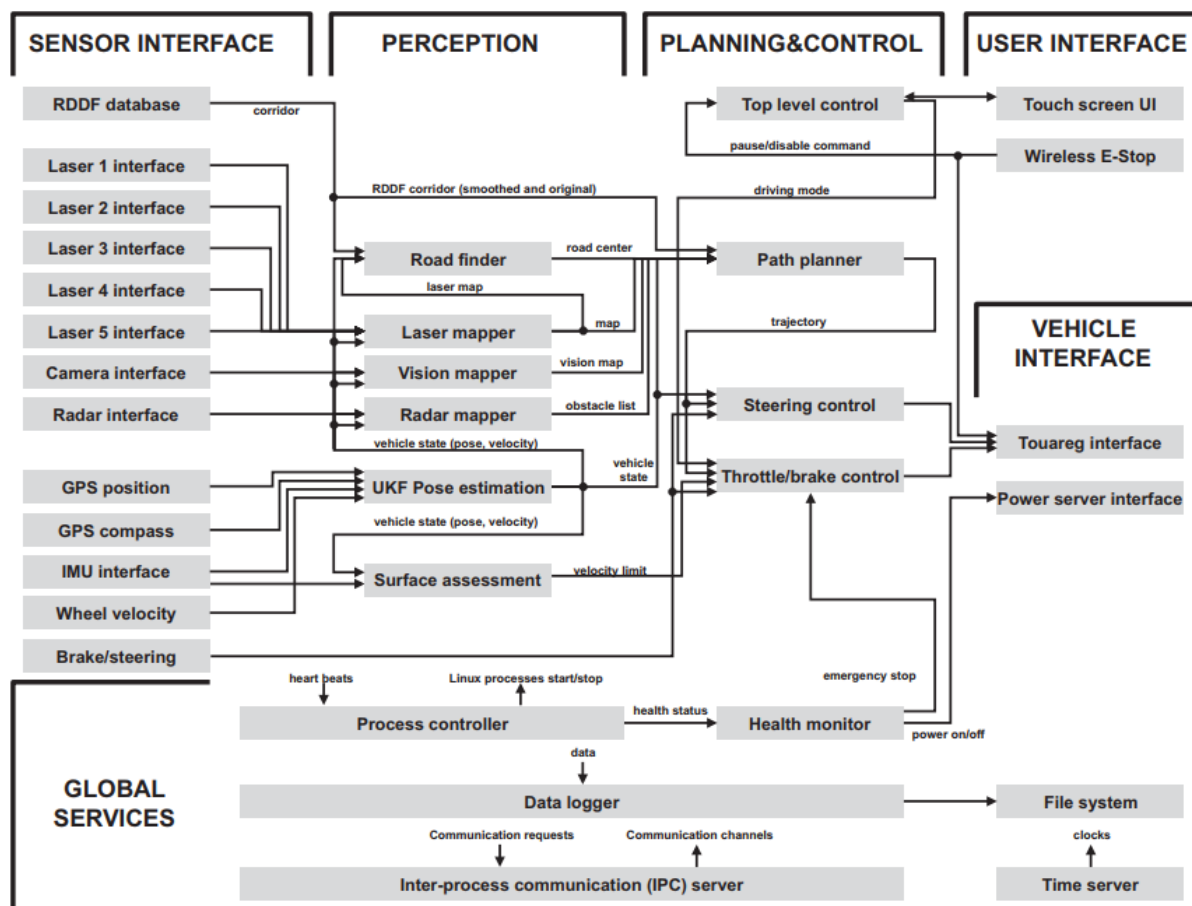
Sayangnya, arsitektur subsumsi memiliki masalahnya sendiri. Pertama, AFSM digerakkan oleh masukan sensor mentah, suatu pengaturan yang berfungsi jika data sensor andal dan berisi semua informasi yang diperlukan untuk pengambilan keputusan, tetapi gagal jika data sensor harus diintegrasikan dengan cara yang tidak sepele dari waktu ke waktu. Oleh karena itu, pengontrol gaya subsumsi sebagian besar diterapkan pada tugas-tugas sederhana, seperti mengikuti dinding atau bergerak menuju sumber cahaya tampak. Kedua, kurangnya pertimbangan membuat sulit untuk mengubah tugas robot. Robot gaya subsumsi biasanya hanya melakukan satu tugas, dan tidak memiliki

gagasan tentang cara memodifikasi kontrolnya untuk mengakomodasi tujuan yang berbeda. Terakhir, pengontrol gaya subsumsi cenderung sulit dipahami. Dalam praktiknya, interaksi rumit antara puluhan AFSM yang berinteraksi (dan lingkungan) berada di luar pemahaman sebagian besar programmer manusia. Karena semua alasan ini, arsitektur subsumsi jarang digunakan dalam robotika, meskipun memiliki nilai historis yang besar. Namun, arsitektur ini telah memengaruhi arsitektur lain, dan juga komponen individual dari beberapa arsitektur.

Arsitektur Tiga Lapisan

Arsitektur hibrida menggabungkan reaksi dengan pertimbangan. Arsitektur hibrida yang paling populer adalah arsitektur tiga lapisan, yang terdiri dari lapisan reaktif, lapisan eksekutif, dan lapisan pertimbangan. Lapisan reaktif menyediakan kontrol tingkat rendah untuk robot. Lapisan ini dicirikan oleh siklus sensor-aksi yang ketat. Siklus keputusannya seringkali dalam orde milidetik.

Lapisan eksekutif (atau lapisan pengurutan) berfungsi sebagai perekat antara lapisan reaktif dan lapisan pertimbangan. Lapisan ini menerima arahan dari lapisan pertimbangan, dan mengurutkannya untuk lapisan reaktif. Misalnya, lapisan eksekutif dapat menangani serangkaian titik via yang dihasilkan oleh perencana jalur pertimbangan, dan membuat keputusan tentang perilaku reaktif mana yang akan dipanggil. Siklus keputusan pada lapisan eksekutif biasanya dalam orde satu detik. Lapisan eksekutif juga bertanggung jawab untuk mengintegrasikan informasi sensor ke dalam representasi status internal. Misalnya, ia dapat menjadi tuan rumah bagi lokalisasi robot dan rutinitas pemetaan daring.



Gambar 7.26 Arsitektur perangkat lunak mobil robot. Perangkat lunak ini mengimplementasikan alur data, di mana semua modul memproses data secara bersamaan.

Lapisan deliberatif menghasilkan solusi global untuk tugas-tugas kompleks menggunakan perencanaan. Karena kompleksitas komputasi yang terlibat dalam menghasilkan solusi tersebut, siklus pengambilan keputusannya seringkali hanya dalam hitungan menit. Lapisan deliberatif (atau lapisan perencanaan) menggunakan model untuk pengambilan keputusan. Model-model tersebut dapat dipelajari dari data atau dipasang dan dapat memanfaatkan informasi status yang dikumpulkan di lapisan eksekutif.

Varian arsitektur tiga lapis dapat ditemukan di sebagian besar sistem perangkat lunak robot modern. Penguraian menjadi tiga lapis tidak terlalu ketat. Beberapa sistem perangkat lunak robot memiliki lapisan tambahan, seperti lapisan antarmuka pengguna yang mengontrol interaksi dengan manusia, atau tingkat multiagen untuk mengoordinasikan tindakan robot dengan tindakan robot lain yang beroperasi di lingkungan yang sama.

Arsitektur Pipeline

Arsitektur lain untuk robot dikenal sebagai arsitektur pipeline. Sama seperti arsitektur subsumsi, arsitektur pipeline mengeksekusi beberapa proses secara paralel. Namun, modul-modul spesifik dalam arsitektur ini menyerupai modul-modul dalam arsitektur tiga lapis. Gambar 7.26 menunjukkan contoh arsitektur jalur pipa, yang digunakan untuk mengendalikan mobil otonom. Data memasuki jalur pipa ini pada lapisan antarmuka sensor.



(a)



(b)

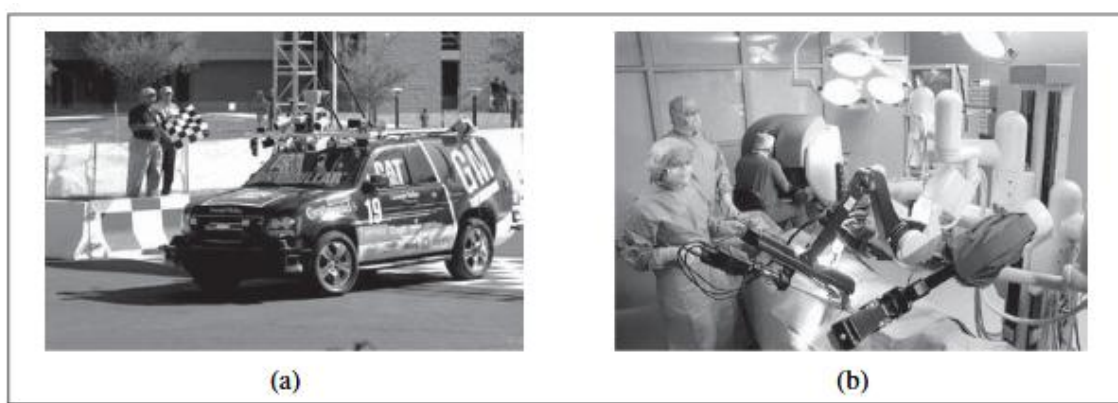
Gambar 7.27 (a) Robot Helpmate mengangkut makanan dan perlengkapan medis lainnya di puluhan rumah sakit di seluruh dunia. (b) Robot Kiva merupakan bagian dari sistem penanganan material untuk memindahkan rak di pusat pemenuhan pesanan. Gambar milik Kiva Systems.

Lapisan persepsi kemudian memperbarui model internal robot terhadap lingkungan berdasarkan data ini. Selanjutnya, model-model ini diserahkan ke lapisan perencanaan dan kontrol, yang menyesuaikan rencana internal robot dan mengubahnya menjadi kontrol aktual untuk robot. Rencana-rencana tersebut kemudian dikomunikasikan kembali ke kendaraan melalui lapisan antarmuka kendaraan.

Kunci dari arsitektur jalur pipa adalah semua ini terjadi secara paralel. Sementara lapisan persepsi memproses data sensor terbaru, lapisan kontrol mendasarkan pilihannya pada data yang sedikit lebih lama. Dengan cara ini, arsitektur jalur pipa mirip dengan otak manusia. Kita tidak mematikan pengontrol gerak kita ketika kita memproses data sensor baru. Sebaliknya, kita mempersepsi, merencanakan, dan bertindak secara bersamaan. Proses dalam arsitektur jalur pipa berjalan secara asinkron, dan semua komputasi digerakkan oleh data. Sistem yang dihasilkan tangguh dan cepat. Arsitektur pada Gambar 7.26 juga berisi modul-modul lintas sektoral lainnya yang bertanggung jawab untuk membangun komunikasi antar elemen-elemen berbeda dalam pipeline.

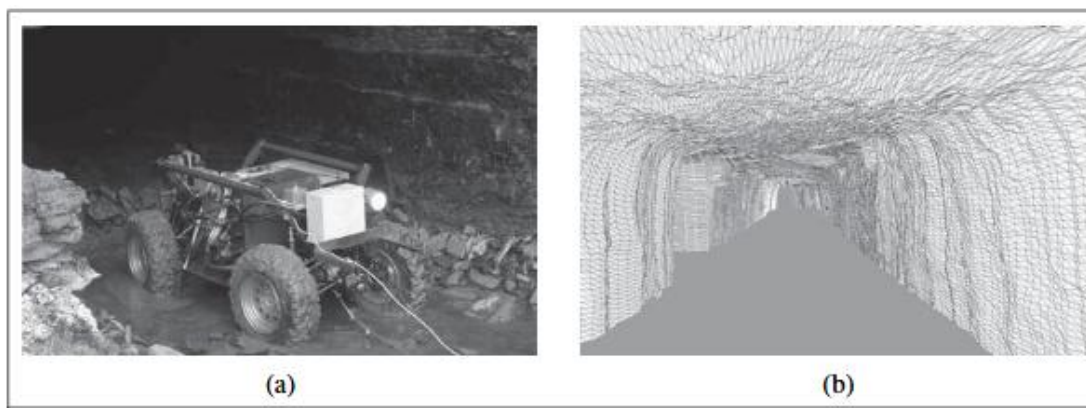
7.8 DOMAIN APLIKASI

Berikut adalah beberapa domain aplikasi utama untuk teknologi robotik. Industri dan Pertanian. Secara tradisional, robot telah digunakan di area yang membutuhkan tenaga manusia yang sulit, namun cukup terstruktur untuk dapat diotomatisasi oleh robot. Contoh terbaiknya adalah jalur perakitan, tempat manipulator secara rutin melakukan tugas-tugas seperti perakitan, penempatan komponen, penanganan material, pengelasan, dan pengecatan. Dalam banyak tugas ini, robot menjadi lebih hemat biaya daripada pekerja manusia. Di luar ruangan, banyak mesin berat yang kita gunakan untuk memanen, menambang, atau menggali tanah telah diubah menjadi robot.



Gambar 7.28 (a) Mobil robotik BOSS, pemenang DARPA Urban Challenge. Sumber: Universitas Carnegie Mellon. (b) Robot bedah di ruang operasi. Gambar milik Sistem Bedah da Vinci.

Misalnya, sebuah proyek di Universitas Carnegie Mellon telah menunjukkan bahwa robot dapat mengelupas cat dari kapal besar sekitar 50 kali lebih cepat daripada manusia, dan dengan dampak lingkungan yang jauh lebih rendah. Prototipe robot penambangan otonom telah ditemukan lebih cepat dan lebih presisi daripada manusia dalam mengangkat bijih di tambang bawah tanah. Robot telah digunakan untuk menghasilkan peta presisi tinggi dari tambang yang terbengkalai dan sistem saluran pembuangan. Meskipun banyak dari sistem ini masih dalam tahap prototipe, hanya masalah waktu sampai robot akan mengambil alih sebagian besar pekerjaan semimekanis yang saat ini dilakukan oleh manusia.



Gambar 7.29 (an) Robot memetakan tambang batu bara yang terbengkalai. (b) Peta 3D tambang yang diperoleh robot.

Transportasi. Transportasi robotik memiliki banyak aspek: mulai dari helikopter otonom yang mengirimkan muatan ke lokasi yang sulit dijangkau, hingga kursi roda otomatis yang mengangkut orang yang tidak dapat mengendalikan kursi roda sendiri, hingga pembawa straddle otonom yang mengungguli pengemudi manusia yang terampil saat mengangkut kontainer dari kapal ke truk di dermaga pemuatan. Contoh utama robot transportasi dalam ruangan, atau gofer, adalah robot Helpmate yang ditunjukkan pada Gambar 7.27(a). Robot ini telah digunakan di puluhan rumah sakit untuk mengangkut makanan dan barang-barang lainnya. Di lingkungan pabrik, kendaraan otonom sekarang secara rutin digunakan untuk mengangkut barang di gudang dan antar lini produksi. Sistem Kiva, yang ditunjukkan pada Gambar 7.27(b), membantu pekerja di pusat pemenuhan mengemas barang ke dalam kontainer pengiriman.

Banyak dari robot ini memerlukan modifikasi lingkungan untuk pengoperasiannya. Modifikasi yang paling umum adalah alat bantu lokalisasi seperti loop induktif di lantai, suar aktif, atau label kode batang. Tantangan terbuka dalam robotika adalah desain robot yang dapat menggunakan isyarat alami, alih-alih perangkat buatan, untuk bernavigasi, terutama di lingkungan seperti laut dalam di mana GPS tidak tersedia.

Mobil robotik. Sebagian besar dari kita menggunakan mobil setiap hari. Banyak dari kita melakukan panggilan telepon seluler saat mengemudi. Beberapa dari kita bahkan mengirim pesan teks. Hasil yang menyedihkan: lebih dari satu juta orang meninggal setiap tahun dalam kecelakaan lalu lintas. Mobil robotik seperti BOSS dan STANLEY menawarkan harapan: Mobil-mobil ini tidak hanya akan membuat berkendara jauh lebih aman, tetapi juga akan membebaskan kita dari kebutuhan untuk memperhatikan jalan selama perjalanan harian kita.

Kemajuan dalam mobil robotik didorong oleh DARPA Grand Challenge, sebuah perlombaan sepanjang 100 mil di medan gurun yang belum dilatih, yang merupakan tugas yang jauh lebih menantang daripada yang pernah dicapai sebelumnya. Kendaraan

STANLEY milik Stanford menyelesaikan lintasan dalam waktu kurang dari tujuh jam pada tahun 2005, memenangkan hadiah sebesar Rp 20 miliar dan tempat di Museum Nasional Sejarah Amerika. Gambar 7.28(a) menggambarkan BOSS, yang pada tahun 2007 memenangkan DARPA Urban Challenge, sebuah balapan jalan raya yang rumit di jalanan kota di mana robot saling berhadapan dan harus mematuhi peraturan lalu lintas.

Layanan kesehatan. Robot semakin banyak digunakan untuk membantu ahli bedah dalam penempatan instrumen saat mengoperasi organ-organ rumit seperti otak, mata, dan jantung. Gambar 7.28(b) menunjukkan sistem tersebut. Robot telah menjadi alat yang sangat diperlukan dalam berbagai prosedur bedah, seperti penggantian pinggul, berkat presisinya yang tinggi. Dalam studi percontohan, perangkat robotik telah terbukti mengurangi risiko lesi saat melakukan kolonoskopi. Di luar ruang operasi, para peneliti telah mulai mengembangkan alat bantu robotik untuk lansia dan penyandang disabilitas, seperti alat bantu jalan robotik cerdas dan mainan cerdas yang memberikan pengingat untuk minum obat dan memberikan kenyamanan. Para peneliti juga sedang mengembangkan perangkat robotik untuk rehabilitasi yang membantu orang-orang dalam melakukan latihan tertentu.

Lingkungan berbahaya. Robot telah membantu orang-orang dalam membersihkan limbah nuklir, terutama di Chernobyl dan Three Mile Island. Robot hadir setelah runtuhnya World Trade Center, di mana mereka memasuki struktur yang dianggap terlalu berbahaya bagi kru pencarian dan penyelamatan manusia.

Beberapa negara telah menggunakan robot untuk mengangkut amunisi dan menjinakkan bom, tugas yang terkenal berbahaya. Sejumlah proyek penelitian saat ini sedang mengembangkan prototipe robot untuk membersihkan ladang ranjau, di darat dan di laut. Sebagian besar robot yang ada untuk tugas-tugas ini dioperasikan dari jarak jauh, manusia mengoperasikannya dengan kendali jarak jauh. Memberikan robot-robot tersebut otonomi merupakan langkah penting selanjutnya.

Eksplorasi. Robot telah pergi ke tempat yang belum pernah dikunjungi siapa pun sebelumnya, termasuk permukaan Mars. Lengan robot membantu astronot dalam menyebarkan dan mengambil satelit serta dalam membangun Stasiun Luar Angkasa Internasional. Robot juga membantu eksplorasi di bawah laut. Mereka secara rutin digunakan untuk memperoleh peta kapal yang tenggelam.

Gambar 7.29 menunjukkan robot yang memetakan tambang batu bara yang terbengkalai, beserta model 3D tambang yang diperoleh menggunakan sensor jarak. Pada tahun 1996, sebuah tim peneliti melepaskan robot berkaki ke dalam kawah gunung berapi aktif untuk memperoleh data untuk penelitian iklim. Kendaraan udara tak berawak yang dikenal sebagai drone digunakan dalam operasi militer. Robot menjadi alat yang sangat efektif untuk mengumpulkan informasi di domain yang sulit (atau berbahaya) diakses manusia.



Gambar 7.30 (a) Roomba, robot bergerak terlaris di dunia, sedang menyedot debu lantai. Gambar milik iRobot, © 2009. (b) Tangan robot yang dimodelkan menyerupai tangan manusia. Gambar milik University of Washington dan Carnegie Mellon University.

Layanan Pribadi. Layanan merupakan domain aplikasi robotika yang sedang berkembang pesat. Robot layanan membantu individu dalam melakukan tugas sehari-hari. Robot layanan rumah tangga yang tersedia secara komersial meliputi penyedot debu otonom, mesin pemotong rumput, dan pembawa tongkat golf. Robot bergerak paling populer di dunia adalah robot layanan pribadi: penyedot debu robotik Roomba, ditunjukkan pada Gambar 7.30(a). Lebih dari tiga juta Roomba telah terjual. Roomba dapat bernavigasi secara otonom dan melakukan tugasnya tanpa bantuan manusia.

Robot layanan lainnya beroperasi di tempat umum, seperti kios informasi robotik yang telah digunakan di pusat perbelanjaan dan pameran dagang, atau di museum sebagai pemandu wisata. Tugas layanan membutuhkan interaksi manusia, dan kemampuan untuk mengatasi lingkungan yang dinamis dan tidak terduga dengan tangguh.

Hiburan. Robot telah mulai menguasai industri hiburan dan mainan. Pada Gambar 7.6(b), kita melihat sepak bola robotik, sebuah permainan kompetitif yang sangat mirip dengan sepak bola manusia, tetapi dimainkan dengan robot bergerak otonom. Sepak bola robot memberikan peluang besar untuk penelitian AI, karena memunculkan berbagai masalah yang relevan dengan banyak aplikasi robot lain yang lebih serius. Kompetisi sepak bola robot tahunan telah menarik banyak peneliti AI dan menambah antusiasme di bidang robotika.

Augmentasi manusia. Domain aplikasi terakhir dari teknologi robotika adalah augmentasi manusia. Para peneliti telah mengembangkan mesin berjalan berkaki yang dapat membawa orang berkeliling, sangat mirip dengan kursi roda. Beberapa upaya penelitian saat ini berfokus pada pengembangan perangkat yang memudahkan orang untuk berjalan atau menggerakkan lengan mereka dengan memberikan kekuatan tambahan melalui perlekatan ekstraskletal. Jika perangkat tersebut dipasang secara permanen, mereka dapat dianggap sebagai anggota tubuh robot buatan. Gambar

7.30(b) menunjukkan tangan robot yang mungkin berfungsi sebagai perangkat prostetik di masa depan.

Teleoperasi robotik, atau telepresensi, adalah bentuk lain dari augmentasi manusia. Teleoperasi melibatkan pelaksanaan tugas jarak jauh dengan bantuan perangkat robotik. Konfigurasi populer untuk teleoperasi robotik adalah konfigurasi master-slave, di mana manipulator robot meniru gerakan operator manusia jarak jauh, yang diukur melalui antarmuka haptik. Kendaraan bawah air seringkali dioperasikan secara tele; kendaraan dapat mencapai kedalaman yang berbahaya bagi manusia tetapi tetap dapat dipandu oleh operator manusia. Semua sistem ini meningkatkan kemampuan manusia untuk berinteraksi dengan lingkungannya. Beberapa proyek bahkan sampai mereplikasi manusia, setidaknya pada tingkat yang sangat dangkal. Robot humanoid kini tersedia secara komersial melalui beberapa perusahaan di Jepang.

7.9 RINGKASAN

Robotika berfokus pada agen cerdas yang memanipulasi dunia fisik. Dalam bab ini, kita telah mempelajari dasar-dasar perangkat keras dan perangkat lunak robot berikut.

- Robot dilengkapi dengan sensor untuk mempersepsi lingkungannya dan efektor yang dengannya mereka dapat memberikan gaya fisik pada lingkungannya. Sebagian besar robot merupakan manipulator yang ditambatkan di lokasi tetap atau robot bergerak yang dapat bergerak.
- Persepsi robotik berfokus pada estimasi kuantitas yang relevan dengan keputusan dari data sensor. Untuk melakukannya, kita memerlukan representasi internal dan metode untuk memperbarui representasi internal ini dari waktu ke waktu. Contoh umum masalah persepsi keras meliputi lokalisasi, pemetaan, dan pengenalan objek.
- Algoritma penyaringan probabilistik seperti filter Kalman dan filter partikel berguna untuk persepsi robot. Teknik-teknik ini mempertahankan status keyakinan, sebuah distribusi posterior atas variabel status.
- Perencanaan gerak robot biasanya dilakukan dalam ruang konfigurasi, di mana setiap titik menentukan lokasi dan orientasi robot serta sudut-sudut sambungannya.
- Algoritma pencarian ruang konfigurasi mencakup teknik dekomposisi sel, yang menguraikan ruang semua konfigurasi menjadi sel-sel berhingga, dan teknik skeletonisasi, yang memproyeksikan ruang konfigurasi ke manifold berdimensi lebih rendah. Masalah perencanaan gerak kemudian diselesaikan menggunakan pencarian dalam struktur yang lebih sederhana ini.
- Jalur yang ditemukan oleh algoritma pencarian dapat dieksekusi dengan menggunakan jalur tersebut sebagai lintasan referensi untuk pengontrol PID.

Pengontrol diperlukan dalam robotika untuk mengakomodasi gangguan kecil; perencanaan jalur saja biasanya tidak cukup.

- Teknik medan potensial menavigasi robot berdasarkan fungsi potensial, yang didefinisikan berdasarkan jarak ke rintangan dan lokasi tujuan. Teknik medan potensial mungkin terjebak dalam minimum lokal, tetapi dapat menghasilkan gerak secara langsung tanpa perlu perencanaan jalur.
- Terkadang lebih mudah untuk menentukan pengontrol robot secara langsung, daripada menurunkan jalur dari model lingkungan yang eksplisit. Pengontrol semacam itu seringkali dapat ditulis sebagai mesin berstatus hingga sederhana.

BAB 8

LANDASAN FILSAFAT

Para filsuf telah ada jauh lebih lama daripada komputer dan telah mencoba menjawab beberapa pertanyaan yang berkaitan dengan AI: Bagaimana cara kerja pikiran? Mungkinkah mesin bertindak secara cerdas seperti manusia, dan jika mereka melakukannya, akankah mereka memiliki pikiran sadar yang nyata? Apa implikasi etis dari mesin cerdas?

Pertama, beberapa terminologi: pernyataan bahwa mesin dapat bertindak *seolah-olah* cerdas disebut hipotesis **AI lemah** oleh para filsuf, dan pernyataan bahwa mesin yang melakukannya *benar-benar* berpikir (bukan hanya *mensimulasikan* pemikiran) disebut hipotesis **AI kuat**.

Sebagian besar peneliti AI menganggap remeh hipotesis AI lemah, dan tidak peduli dengan hipotesis AI kuat—selama program mereka berfungsi, mereka tidak peduli apakah Anda menyebutnya simulasi kecerdasan atau kecerdasan nyata. Semua peneliti AI harus memperhatikan implikasi etis dari pekerjaan mereka.

8.1 AI YANG LEMAH: DAPATKAH MESIN BERTINDAK SECARA CERDAS?

Proposal untuk lokakarya musim panas 1956 yang mendefinisikan bidang Kecerdasan Buatan menyatakan bahwa "Setiap aspek pembelajaran atau fitur kecerdasan lainnya dapat dideskripsikan dengan begitu tepat sehingga mesin dapat disimulasikan." Dengan demikian, AI didasarkan pada asumsi bahwa AI yang lemah dimungkinkan. Yang lain menyatakan bahwa AI yang lemah mustahil: "Kecerdasan buatan yang *diupayakan dalam kultus komputasionalisme* bahkan tidak memiliki sedikit pun peluang untuk menghasilkan hasil yang tahan lama".

Jelas, apakah AI mustahil bergantung pada bagaimana ia didefinisikan. Di awal buku ini penulis mendefinisikan AI sebagai pencarian program agen terbaik pada arsitektur tertentu. Dengan formulasi ini, AI secara definisi dimungkinkan: untuk setiap arsitektur digital dengan penyimpanan program k bit terdapat tepat 2^k program agen, dan yang harus kita lakukan untuk menemukan yang terbaik adalah menghitung dan menguji semuanya. Hal ini mungkin tidak dapat dilakukan untuk k yang besar, tetapi para filsuf berurusan dengan hal teoritis, bukan praktik.

Definisi AI kami bekerja dengan baik untuk masalah rekayasa dalam menemukan agen yang baik, mengingat suatu arsitektur. Oleh karena itu, kami tergoda untuk mengakhiri bagian ini sekarang, menjawab pertanyaan judul dengan jawaban ya. Namun, para filsuf tertarik pada masalah membandingkan dua arsitektur—manusia dan mesin. Lebih lanjut, mereka secara tradisional mengajukan pertanyaan bukan dalam hal memaksimalkan utilitas yang diharapkan, melainkan sebagai, "**Bisakah mesin berpikir?**"

Ilmuwan komputer Edsger Dijkstra (1984) mengatakan bahwa "Pertanyaan apakah *Mesin Dapat Berpikir* ... sama relevannya dengan pertanyaan apakah *Kapal Selam Dapat Berenang*." Definisi pertama *berenang* dari American Heritage Dictionary adalah "Bergerak di air dengan menggunakan tungkai, sirip, atau ekor," dan kebanyakan orang sepakat bahwa kapal selam,

karena tidak memiliki tungkai, tidak dapat berenang. Kamus tersebut juga mendefinisikan *terbang* sebagai "Bergerak di udara dengan menggunakan sayap atau bagian seperti sayap," dan kebanyakan orang sepakat bahwa pesawat terbang, yang memiliki bagian seperti sayap, dapat terbang. Namun, baik pertanyaan maupun jawaban tidak memiliki relevansi dengan desain atau kemampuan pesawat terbang dan kapal selam; melainkan tentang penggunaan kata-kata dalam bahasa Inggris. (Fakta bahwa kapal berenang di Rusia hanya memperkuat poin ini.). Kemungkinan praktis "mesin berpikir" telah bersama kita hanya sekitar 50 tahun, tidak cukup lama bagi penutur bahasa Inggris untuk menetapkan arti kata "berpikir"—apakah itu membutuhkan "otak" atau hanya "bagian seperti otak."

Alan Turing, dalam makalahnya yang terkenal "Computing Machinery and Intelligence" (1950), menyarankan bahwa alih-alih bertanya apakah mesin dapat berpikir, kita harus bertanya apakah mesin dapat lulus tes kecerdasan perilaku, yang kemudian disebut **Tes Turing**. Tes ini adalah untuk sebuah program untuk melakukan percakapan (melalui pesan ketik daring) dengan seorang interogator selama lima menit. Interogator kemudian harus menebak apakah percakapan itu dengan sebuah program atau seseorang; program lulus tes jika membodohi interogator 30% dari waktu. Turing menduga bahwa, pada tahun 2000, sebuah komputer dengan penyimpanan 10^9 unit dapat diprogram dengan cukup baik untuk lulus uji. Ia salah—program belum mampu mengelabui hakim yang cangguh sekalipun.

Di sisi lain, banyak orang *telah* tertipu ketika mereka tidak tahu bahwa mereka mungkin sedang mengobrol dengan komputer. Program ELIZA dan chatbot internet seperti MGONZ dan NATACHATA telah berulang kali mengelabui koresponden mereka, dan chatbot CYBERLOVER telah menarik perhatian penegak hukum karena kegemarannya mengelabui sesama pengobrol agar membocorkan informasi pribadi yang cukup sehingga identitas mereka dapat dicuri. Kompetisi Loebner Prize, yang diadakan setiap tahun sejak 1991, merupakan kontes serupa Tes Turing yang paling lama berlangsung. Kompetisi-kompetisi tersebut telah menghasilkan model kesalahan ketik manusia yang lebih baik.

Turing sendiri mengkaji berbagai macam kemungkinan keberatan terhadap kemungkinan adanya mesin cerdas, termasuk hampir semua keberatan yang telah diajukan dalam setengah abad sejak makalahnya terbit. Kita akan membahas beberapa di antaranya.

Argumen dari Disabilitas

"Argumen dari Disabilitas" menyatakan bahwa "sebuah mesin tidak akan pernah bisa melakukan X." Sebagai contoh X, Turing mencantumkan hal-hal berikut:

Bersikap baik, banyak akal, cantik, ramah, memiliki inisiatif, memiliki selera humor, membedakan yang benar dari yang salah, membuat kesalahan, jatuh cinta, menikmati stroberi dan krim, membuat seseorang jatuh cinta padanya, belajar dari pengalaman, menggunakan kata-kata dengan tepat, menjadi subjek pemikirannya sendiri, memiliki keragaman perilaku yang sama besarnya dengan manusia, melakukan sesuatu yang benar-benar baru.

Kalau dipikir-pikir kembali, beberapa di antaranya cukup mudah—kita semua akrab dengan komputer yang "membuat kesalahan." Kita juga akrab dengan teknologi berusia seabad yang

telah terbukti kemampuannya untuk "membuat seseorang jatuh cinta padanya"—boneka beruang. Pakar catur komputer, David Levy, memprediksi bahwa pada tahun 2050 orang akan secara rutin jatuh cinta pada robot humanoid. Mengenai robot yang jatuh cinta, hal itu merupakan tema umum dalam fiksi, tetapi hanya ada spekulasi terbatas tentang apakah hal itu sebenarnya mungkin terjadi. Program memang memainkan catur, dam, dan permainan lainnya; memeriksa komponen pada jalur perakitan, mengemudikan mobil dan helikopter; mendiagnosis penyakit; dan melakukan ratusan tugas lain sebaik atau lebih baik daripada manusia. Komputer telah membuat penemuan-penemuan kecil namun signifikan dalam astronomi, matematika, kimia, mineralogi, biologi, ilmu komputer, dan bidang-bidang lainnya. Masing-masing penemuan ini membutuhkan kinerja setingkat pakar manusia.

Mengingat apa yang kita ketahui sekarang tentang komputer, tidak mengherankan bahwa mereka berhasil dalam masalah kombinatorial seperti bermain catur. Namun, algoritma juga berkinerja setara dengan manusia dalam tugas-tugas yang tampaknya melibatkan penilaian manusia, atau seperti yang dikatakan Turing, "belajar dari pengalaman" dan kemampuan untuk "membedakan yang benar dari yang salah." Sejak tahun 1955, Paul Meehl mempelajari proses pengambilan keputusan para ahli terlatih dalam tugas-tugas subjektif seperti memprediksi keberhasilan seorang siswa dalam program pelatihan atau residivisme seorang penjahat. Dalam 19 dari 20 studi yang ditelitinya, Meehl menemukan bahwa algoritma pembelajaran statistik sederhana (seperti regresi linier atau Bayes naif) memprediksi lebih baik daripada para ahli. Educational Testing Service telah menggunakan program otomatis untuk menilai jutaan pertanyaan esai pada ujian GMAT sejak tahun 1999. Program ini 97% setuju dengan penilaian manusia, hampir sama dengan tingkat persetujuan dua penilaian manusia.

Jelas bahwa komputer dapat melakukan banyak hal sebaik atau lebih baik daripada manusia, termasuk hal-hal yang diyakini manusia membutuhkan wawasan dan pemahaman manusia yang mendalam. Tentu saja, ini tidak berarti bahwa komputer menggunakan wawasan dan pemahaman dalam melakukan tugas-tugas ini—hal-hal tersebut bukan bagian dari *perilaku*, dan kami membahas pertanyaan-pertanyaan semacam itu di tempat lain—tetapi intinya adalah bahwa tebakan pertama seseorang tentang proses mental yang diperlukan untuk menghasilkan perilaku tertentu seringkali salah. Tentu saja, juga benar bahwa ada banyak tugas yang belum dikuasai komputer (secara halus), termasuk tugas Turing untuk melakukan percakapan terbuka.

Keberatan Matematika

Telah diketahui secara luas, melalui karya Turing (1936) dan Godel (1931), bahwa pertanyaan matematika tertentu pada prinsipnya tidak dapat dijawab oleh sistem formal tertentu. Teorema ketidaklengkapan Godel adalah contoh paling terkenal dari hal ini. Singkatnya, untuk setiap sistem aksiomatik formal F yang cukup kuat untuk melakukan aritmatika, dimungkinkan untuk membangun apa yang disebut kalimat Godel $G(F)$ dengan sifat-sifat berikut:

- $G(F)$ adalah kalimat dari F , tetapi tidak dapat dibuktikan dalam F .
- Jika F konsisten, maka $G(F)$ benar.

Filsuf seperti J. R. Lucas (1961) telah mengklaim bahwa teorema ini menunjukkan bahwa mesin secara mental lebih rendah daripada manusia, karena mesin adalah sistem formal yang dibatasi oleh teorema ketidaklengkapan—mereka tidak dapat membuktikan kebenaran kalimat Gödel mereka sendiri—sementara manusia tidak memiliki batasan tersebut. Klaim ini telah menimbulkan kontroversi selama beberapa dekade, melahirkan banyak literatur, termasuk dua buku karya matematikawan Sir Roger Penrose (1989, 1994) yang mengulangi klaim tersebut dengan beberapa perubahan baru (seperti hipotesis bahwa manusia berbeda karena otak mereka beroperasi dengan gravitasi kuantum). Kami hanya akan mengkaji tiga masalah dari klaim tersebut.

Teorema ketidaklengkapan Gödel hanya berlaku untuk sistem formal yang cukup kuat untuk melakukan aritmatika. Ini termasuk mesin Turing, dan klaim Lucas sebagian didasarkan pada pernyataan bahwa komputer adalah mesin Turing. Ini merupakan perkiraan yang baik, tetapi tidak sepenuhnya benar. Mesin Turing tak terhingga, sedangkan komputer terbatas, dan oleh karena itu setiap komputer dapat digambarkan sebagai sistem (yang sangat besar) dalam logika proposisional, yang tidak tunduk pada teorema ketidaklengkapan Gödel. Kedua, seorang agen tidak perlu terlalu malu karena tidak dapat membuktikan kebenaran suatu kalimat sementara agen lain bisa. Perhatikan kalimat:

J. R. Lucas tidak dapat secara konsisten menyatakan bahwa kalimat ini benar.

Jika Lucas menyatakan kalimat ini, maka ia akan mengontradiksi dirinya sendiri, sehingga Lucas tidak dapat secara konsisten menyatakannya, dan karenanya kalimat tersebut pasti benar. Dengan demikian, kami telah menunjukkan bahwa ada kalimat yang tidak dapat dinyatakan secara konsisten oleh Lucas sementara orang lain (dan mesin) bisa. Namun, hal itu tidak membuat kami memandang rendah Lucas. Sebagai contoh lain, tidak ada manusia yang dapat menghitung jumlah satu miliar angka 10 digit dalam hidupnya, tetapi komputer dapat melakukannya dalam hitungan detik. Namun, kami tidak melihat hal ini sebagai batasan mendasar dalam kemampuan berpikir manusia. Manusia telah berperilaku cerdas selama ribuan tahun sebelum mereka menemukan matematika, jadi kecil kemungkinan penalaran matematika formal memainkan peran lebih dari sekadar peran sampingan dalam arti menjadi cerdas.

Dan yang terpenting, bahkan jika kita mengakui bahwa komputer memiliki keterbatasan dalam hal yang dapat dibuktikan, tidak ada bukti bahwa manusia kebal terhadap keterbatasan tersebut. Terlalu mudah untuk menunjukkan secara ketat bahwa sistem formal tidak dapat melakukan X , lalu mengklaim bahwa manusia *dapat* melakukan X menggunakan metode informal mereka sendiri, tanpa memberikan bukti apa pun untuk klaim ini. Memang, mustahil untuk *membuktikan* bahwa manusia tidak tunduk pada teorema ketidaklengkapan Gödel, karena pembuktian yang ketat apa pun akan membutuhkan formalisasi dari bakat manusia yang diklaim tidak dapat diformalkan, dan karenanya menyangkal dirinya sendiri. Jadi, kita hanya bisa mengandalkan intuisi bahwa manusia entah bagaimana dapat melakukan prestasi wawasan matematika yang luar biasa. Daya tarik ini diungkapkan dengan argumen

seperti "kita harus mengasumsikan konsistensi kita sendiri, jika pemikiran memang memungkinkan". Namun, manusia memang dikenal tidak konsisten. Hal ini tentu berlaku untuk penalaran sehari-hari, tetapi juga berlaku untuk pemikiran matematika yang cermat. Contoh terkenalnya adalah masalah peta empat warna. Alfred Kempe menerbitkan sebuah bukti pada tahun 1879 yang diterima secara luas dan berkontribusi pada pemilihannya sebagai Anggota Royal Society. Namun, pada tahun 1890, Percy Heawood menunjukkan sebuah kelemahan dan teorema tersebut tetap tidak terbukti hingga tahun 1977.

Argumen dari Informalitas

Salah satu kritik paling berpengaruh dan persisten terhadap AI sebagai sebuah perusahaan dikemukakan oleh Turing sebagai "argumen dari informalitas perilaku." Intinya, ini adalah klaim bahwa perilaku manusia terlalu kompleks untuk ditangkap oleh seperangkat aturan sederhana dan karena komputer tidak dapat melakukan lebih dari sekadar mengikuti seperangkat aturan, mereka tidak dapat menghasilkan perilaku secerdas manusia. Ketidakmampuan untuk menangkap segala sesuatu dalam seperangkat aturan logis disebut masalah kualifikasi dalam AI.

Pendukung utama pandangan ini adalah filsuf Hubert Dreyfus, yang telah menghasilkan serangkaian kritik berpengaruh terhadap kecerdasan buatan: *What Computers Can't Do* (1972), *sekuelnya What Computers Still Can't Do* (1992), dan, bersama saudaranya Stuart, *Mind Over Machine* (1986).

Posisi yang mereka kritik kemudian disebut "AI Kuno yang Baik," atau GOFAI, sebuah istilah yang dicetuskan oleh filsuf John Haugeland (1985). GOFAI seharusnya mengklaim bahwa semua perilaku cerdas dapat ditangkap oleh sistem yang bernalar secara logis dari serangkaian fakta dan aturan yang menggambarkan domain tersebut. Oleh karena itu, hal ini sesuai dengan agen logis paling sederhana yang dijelaskan dalam Bab 7. Dreyfus benar ketika mengatakan bahwa agen logis rentan terhadap masalah kualifikasi. Sistem penalaran probabilistik lebih sesuai untuk domain terbuka. Oleh karena itu, kritik Dreyfus tidak ditujukan terhadap komputer itu sendiri, melainkan terhadap satu cara pemrograman tertentu. Namun, masuk akal untuk berasumsi bahwa sebuah buku berjudul *What First-Order Logical Rule-Based Systems Without Learning Can't Do* mungkin kurang berdampak.

Dalam pandangan Dreyfus, keahlian manusia memang mencakup pengetahuan tentang beberapa aturan, tetapi hanya sebagai "konteks holistik" atau "latar belakang" tempat manusia beroperasi. Ia memberikan contoh perilaku sosial yang tepat dalam memberi dan menerima hadiah: "Biasanya seseorang hanya merespons dalam keadaan yang tepat dengan memberikan hadiah yang sesuai." Seseorang tampaknya memiliki "perasaan langsung tentang bagaimana segala sesuatunya dilakukan dan apa yang diharapkan." Klaim yang sama diajukan dalam konteks permainan catur: "Seorang master catur biasa mungkin perlu memikirkan apa yang harus dilakukan, tetapi seorang grandmaster hanya melihat papan catur menuntut langkah tertentu... respons yang tepat muncul begitu saja di kepalanya." Memang benar bahwa sebagian besar proses berpikir seorang pemberi hadiah atau grandmaster dilakukan pada tingkat yang tidak terbuka untuk introspeksi oleh pikiran sadar. Namun, itu tidak berarti proses berpikir tersebut tidak ada. Pertanyaan penting yang tidak dijawab Dreyfus adalah

bagaimana langkah yang tepat masuk ke kepala sang grandmaster. Kita teringat komentar Daniel Dennett (1984),

Seolah-olah para filsuf menyatakan diri mereka sebagai ahli penjelas metode pesulap panggung, dan kemudian, ketika kita bertanya bagaimana pesulap melakukan trik menggergaji wanita menjadi dua, mereka menjelaskan bahwa sebenarnya hal itu cukup jelas: pesulap tidak benar-benar melihatnya terbelah dua; ia hanya membuatnya tampak seolah-olah ia melihatnya. "Tapi bagaimana ia melakukannya?" tanya kita. "Bukan departemen kami," kata para filsuf.

Dreyfus dan Dreyfus (1986) mengusulkan proses lima tahap untuk memperoleh keahlian, dimulai dengan pemrosesan berbasis aturan (seperti yang diusulkan dalam GOFAI) dan diakhiri dengan kemampuan untuk memilih respons yang tepat secara instan. Dengan mengajukan usulan ini, Dreyfus dan Dreyfus pada dasarnya beralih dari kritikus AI menjadi ahli teori AI—mereka mengusulkan arsitektur jaringan saraf yang terorganisasi ke dalam "pustaka kasus" yang luas, tetapi menunjukkan beberapa masalah. Untungnya, semua masalah mereka telah diatasi, beberapa dengan keberhasilan parsial dan beberapa dengan keberhasilan total. Masalah mereka meliputi:

1. Generalisasi yang baik dari contoh-contoh tidak dapat dicapai tanpa pengetahuan latar belakang. Mereka mengklaim tidak seorang pun tahu bagaimana mengintegrasikan pengetahuan latar belakang ke dalam proses pembelajaran jaringan saraf tiruan. Faktanya, kita telah melihat di Bab 19 dan 20 bahwa terdapat teknik untuk menggunakan pengetahuan sebelumnya dalam algoritma pembelajaran. Namun, teknik-teknik tersebut bergantung pada ketersediaan pengetahuan dalam bentuk eksplisit, sesuatu yang dibantah keras oleh Dreyfus dan Dreyfus. Menurut pandangan kami, ini merupakan alasan yang baik untuk mendesain ulang model-model pemrosesan saraf tiruan saat ini secara serius agar *dapat* memanfaatkan pengetahuan yang telah dipelajari sebelumnya sebagaimana algoritma pembelajaran lainnya.
2. Pembelajaran jaringan saraf tiruan merupakan bentuk pembelajaran terawasi (lihat Bab 18), yang membutuhkan identifikasi awal masukan yang relevan dan keluaran yang benar. Oleh karena itu, mereka mengklaim, pembelajaran ini tidak dapat beroperasi secara mandiri tanpa bantuan pelatih manusia. Faktanya, pembelajaran tanpa guru dapat dicapai melalui **pembelajaran tanpa pengawasan** dan **pembelajaran penguatan**.
3. Algoritma pembelajaran tidak berkinerja baik dengan banyak fitur, dan jika kita memilih sebagian fitur, "tidak ada cara yang diketahui untuk menambahkan fitur baru jika kumpulan fitur yang ada terbukti tidak memadai untuk memperhitungkan fakta yang dipelajari." Faktanya, metode baru seperti mesin vektor pendukung menangani kumpulan fitur yang besar dengan sangat baik. Dengan diperkenalkannya kumpulan data berbasis web yang besar, banyak aplikasi di bidang seperti pemrosesan bahasa dan visi komputer secara rutin menangani jutaan fitur. Kita telah melihat di Bab 1 bahwa terdapat juga cara-cara berprinsip untuk menghasilkan fitur baru, meskipun masih banyak pekerjaan yang perlu dilakukan.

4. Otak mampu mengarahkan sensornya untuk mencari informasi yang relevan dan memprosesnya untuk mengekstrak aspek-aspek yang relevan dengan situasi saat ini. Namun, Dreyfus dan Dreyfus mengklaim, "Saat ini, belum ada detail mekanisme ini yang dipahami atau bahkan dihipotesiskan dengan cara yang dapat memandu penelitian AI." Faktanya, bidang penglihatan aktif, yang didukung oleh teori nilai informasi, berkaitan dengan masalah pengarahannya sensor, dan beberapa robot telah menggabungkan hasil teoretis yang diperoleh. Perjalanan STANLEY sejauh 132 mil melintasi gurun sebagian besar dimungkinkan oleh sistem penginderaan aktif semacam ini.

Singkatnya, banyak isu yang difokuskan Dreyfus—pengetahuan akal sehat latar belakang, masalah kualifikasi, ketidakpastian, pembelajaran, bentuk-bentuk pengambilan keputusan yang terkompilasi—memang merupakan isu-isu penting, dan kini telah diintegrasikan ke dalam desain agen cerdas standar. Menurut pandangan kami, ini adalah bukti kemajuan AI, bukan ketidakmungkinannya.

Salah satu argumen terkuat Dreyfus adalah untuk agen yang berada dalam situasi, alih-alih mesin inferensi logis yang tidak berwujud. Agen yang pemahamannya tentang "anjing" hanya berasal dari serangkaian kalimat logis yang terbatas seperti " $Dog(x) \Rightarrow Mammal(x)$ " berada pada posisi yang kurang menguntungkan dibandingkan dengan agen yang pernah melihat anjing berlari, bermain tangkap bola dengan mereka, dan pernah dijilati oleh mereka. Sebagaimana dikatakan oleh filsuf Andy Clark (1998), "Otak biologis pertama dan terutama merupakan sistem kendali bagi tubuh biologis. Tubuh biologis bergerak dan bertindak dalam lingkungan dunia nyata yang kaya." Untuk memahami cara kerja agen manusia (atau hewan lain), kita harus mempertimbangkan keseluruhan agen, bukan hanya program agen. Pendekatan **kognisi yang diwujudkan** menyatakan bahwa tidak masuk akal untuk mempertimbangkan otak secara terpisah: kognisi terjadi di dalam tubuh, yang tertanam dalam suatu lingkungan. Kita perlu mempelajari sistem secara keseluruhan; otak memperkuat penalarannya dengan merujuk pada lingkungan, seperti yang dilakukan pembaca dalam mempersepsi (dan membuat) tanda di atas kertas untuk mentransfer pengetahuan. Di bawah program kognisi yang diwujudkan, robotika, penglihatan, dan sensor lainnya menjadi sentral, bukan periferal.

8.2 AI YANG KUAT: DAPATKAH MESIN BENAR-BENAR BERPIKIR?

Banyak filsuf mengklaim bahwa mesin yang lulus Uji Turing tetap tidak akan *benar-benar* berpikir, melainkan hanya simulasi berpikir. Sekali lagi, keberatan ini telah diramalkan oleh Turing. Ia mengutip pidato Profesor Geoffrey Jefferson (1949):

Baru setelah sebuah mesin dapat menulis soneta atau mengubah konserto karena pikiran dan emosi yang dirasakan, dan bukan karena kemunculan simbol secara kebetulan, kita dapat sepakat bahwa mesin sama dengan otak—artinya, tidak hanya menulisnya tetapi juga tahu bahwa ia telah menulisnya.

Turing menyebut ini argumen dari **kesadaran**—mesin harus menyadari keadaan mental dan tindakannya sendiri. Meskipun kesadaran merupakan subjek penting, poin kunci Jefferson sebenarnya berkaitan dengan **fenomenologi**, atau studi tentang pengalaman langsung: mesin harus benar-benar merasakan emosi. Yang lain berfokus pada **intensionalitas**—yaitu, pertanyaan apakah keyakinan, keinginan, dan representasi lain yang diklaim oleh mesin tersebut sebenarnya "mengenai" sesuatu di dunia nyata.

Tanggapan Turing terhadap keberatan ini menarik. Ia bisa saja mengemukakan alasan bahwa mesin sebenarnya bisa memiliki kesadaran (atau memiliki fenomenologi, atau memiliki intensi). Sebaliknya, ia berpendapat bahwa pertanyaan tersebut sama tidak jelasnya dengan pertanyaan, "Bisakah mesin berpikir?" Lagipula, mengapa kita harus menuntut standar yang lebih tinggi untuk mesin daripada standar yang kita terapkan untuk manusia? Lagipula, dalam kehidupan sehari-hari kita tidak pernah *memiliki* bukti langsung tentang kondisi mental internal manusia lain. Namun demikian, Turing berkata, "Daripada terus-menerus berdebat tentang hal ini, biasanya ada konvensi sopan yang menyatakan bahwa semua orang berpikir." Turing berpendapat bahwa Jefferson bersedia memperluas konvensi sopan tersebut ke mesin jika saja ia memiliki pengalaman dengan mesin yang bertindak secara cerdas. Ia mengutip dialog berikut, yang telah menjadi bagian dari tradisi lisan AI sehingga kita harus memasukkannya:

MANUSIA: Pada baris pertama soneta Anda yang berbunyi "haruskah kubandingkan engkau dengan hari musim panas," bukankah "hari musim semi" akan sama bagusnya atau lebih baik?

MESIN: Itu tidak akan terpindai.

MANUSIA: Bagaimana dengan "hari musim dingin." Itu akan terlihat bagus.

MESIN: Ya, tapi tak seorang pun ingin dibandingkan dengan hari musim dingin.

MANUSIA: Apakah menurutmu Tuan Pickwick mengingatkanmu pada Natal?

MESIN: Bisa dibilang begitu.

MANUSIA: Padahal Natal adalah hari musim dingin, dan kurasa Tuan Pickwick tak akan keberatan jika dibandingkan.

MESIN: Saya rasa Anda tidak serius. Yang dimaksud dengan hari musim dingin adalah hari musim dingin yang biasa, bukan hari istimewa seperti Natal.

Kita dapat dengan mudah membayangkan masa depan di mana percakapan seperti itu dengan mesin menjadi hal yang biasa, dan menjadi kebiasaan untuk tidak membedakan secara linguistik antara pemikiran "nyata" dan "buatan". Transisi serupa terjadi pada tahun-tahun setelah 1848, ketika urea buatan disintesis untuk pertama kalinya oleh Frederick Wöhler. Sebelum peristiwa ini, kimia organik dan anorganik pada dasarnya merupakan bidang yang terpisah dan banyak yang berpikir bahwa tidak ada proses yang dapat mengubah bahan kimia anorganik menjadi bahan organik. Setelah sintesis selesai, para ahli kimia sepakat bahwa urea *buatan* adalah urea, karena memiliki semua sifat fisik yang tepat. Mereka yang telah mengemukakan sifat intrinsik yang dimiliki oleh bahan organik yang tidak pernah dimiliki oleh

bahan anorganik dihadapkan pada ketidakmungkinan untuk merancang uji apa pun yang dapat mengungkapkan dugaan kekurangan urea buatan.

Untuk berpikir, kita belum mencapai tahun 1848, dan ada yang percaya bahwa pemikiran artifisial, betapa pun mengesankannya, tidak akan pernah nyata. Misalnya, filsuf John Searle (1980) berpendapat sebagai berikut:

Tak seorang pun mengira bahwa simulasi komputer tentang badai akan membuat kita semua basah kuyup... Mengapa orang waras mengira simulasi komputer tentang proses mental benar-benar memiliki proses mental?

Meskipun mudah untuk sepakat bahwa simulasi komputer tentang badai tidak membuat kita basah kuyup, tidak jelas bagaimana menerapkan analogi ini pada simulasi komputer tentang proses mental. Lagipula, simulasi badai Hollywood yang menggunakan sprinkler dan mesin angin memang *membuat* para aktornya basah kuyup, dan simulasi badai dalam gim video memang *membuat* karakter yang disimulasikan basah kuyup. Kebanyakan orang merasa nyaman mengatakan bahwa simulasi komputer tentang penjumlahan adalah penjumlahan, dan catur adalah catur. Faktanya, kita biasanya berbicara tentang *implementasi* penjumlahan atau catur, bukan *simulasi*. Apakah proses mental lebih mirip badai, atau lebih mirip penjumlahan?

Jawaban Turing—konvensi yang sopan—menunjukkan bahwa masalah ini pada akhirnya akan hilang dengan sendirinya setelah mesin mencapai tingkat kecanggihan tertentu. Hal ini akan menghilangkan perbedaan antara AI yang lemah dan kuat. Namun, ada yang berpendapat bahwa ada masalah *faktual* yang dipertaruhkan: manusia memang memiliki pikiran yang nyata, dan mesin mungkin memilikinya atau tidak. Untuk mengatasi masalah faktual ini, kita perlu memahami bagaimana manusia memiliki pikiran yang nyata, bukan hanya tubuh yang menghasilkan proses neurofisiologis. Upaya filosofis untuk memecahkan **masalah pikiran-tubuh** ini secara langsung relevan dengan pertanyaan apakah mesin dapat memiliki pikiran yang nyata.

Masalah pikiran-tubuh telah dipertimbangkan oleh para filsuf Yunani kuno dan berbagai aliran pemikiran Hindu, tetapi pertama kali dianalisis secara mendalam oleh filsuf dan matematikawan Prancis abad ke-17, Rene Descartes. *Meditasi Filsafat Pertama* (1641) karyanya mempertimbangkan aktivitas berpikir pikiran (suatu proses tanpa jangkauan spasial atau sifat material) dan proses fisik tubuh, menyimpulkan bahwa keduanya pasti berada dalam ranah terpisah—yang sekarang kita sebut teori **dualis**. Permasalahan pikiran-tubuh yang dihadapi oleh para dualis adalah pertanyaan tentang bagaimana pikiran dapat mengendalikan tubuh jika keduanya benar-benar terpisah. Descartes berspekulasi bahwa keduanya mungkin berinteraksi melalui kelenjar pineal, yang memunculkan pertanyaan tentang bagaimana pikiran mengendalikan kelenjar pineal.

Teori **monis** tentang pikiran, yang sering disebut **fisikalisme**, menghindari masalah ini dengan menegaskan bahwa pikiran tidak terpisah dari tubuh—bahwa keadaan mental *adalah* keadaan fisik. Kebanyakan filsuf pikiran modern adalah fisikalis dalam satu bentuk atau lainnya, dan

fisikalisme memungkinkan, setidaknya secara prinsip, kemungkinan adanya AI yang kuat. Permasalahan bagi fisikalis adalah menjelaskan bagaimana keadaan fisik—khususnya, konfigurasi molekuler dan proses elektrokimia otak—dapat secara bersamaan menjadi **keadaan mental**, seperti merasakan sakit, menikmati hamburger, mengetahui bahwa seseorang sedang menunggang kuda, atau meyakini bahwa Wina adalah ibu kota Austria.

Kondisi Mental dan Otak dalam Bejana

Para filsuf fisikalis telah mencoba menjelaskan apa artinya mengatakan bahwa seseorang—dan, lebih luas lagi, komputer—berada dalam kondisi mental tertentu. Mereka berfokus secara khusus pada **kondisi intensional**. Kondisi-kondisi ini, seperti meyakini, mengetahui, menginginkan, takut, dan sebagainya, yang merujuk pada beberapa aspek dunia luar. Misalnya, pengetahuan bahwa seseorang sedang makan hamburger merupakan keyakinan *tentang* hamburger tersebut dan apa yang terjadi padanya.

Jika fisikalisme benar, deskripsi yang tepat tentang kondisi mental seseorang pastilah *ditentukan* oleh kondisi otak orang tersebut. Jadi, jika saat ini saya berfokus pada makan hamburger dengan penuh kesadaran, kondisi otak sesaat saya merupakan contoh dari kelas kondisi mental "mengetahui bahwa seseorang sedang makan hamburger." Tentu saja, konfigurasi spesifik dari semua atom otak saya tidaklah esensial: ada banyak konfigurasi otak saya, atau otak orang lain, yang termasuk dalam kelas kondisi mental yang sama. Poin kuncinya adalah bahwa kondisi otak yang sama tidak mungkin sesuai dengan kondisi mental yang secara fundamental berbeda, seperti pengetahuan bahwa seseorang sedang makan pisang.

Kesederhanaan pandangan ini ditantang oleh beberapa eksperimen pemikiran sederhana. Bayangkan, jika Anda mau, bahwa otak Anda dikeluarkan dari tubuh Anda saat lahir dan ditempatkan dalam sebuah wadah yang direkayasa secara luar biasa. Wadah itu menopang otak Anda, memungkinkannya untuk tumbuh dan berkembang. Pada saat yang sama, sinyal elektronik diumpankan ke otak Anda dari simulasi komputer dari dunia yang sepenuhnya fiktif, dan sinyal motorik dari otak Anda dicegat dan digunakan untuk memodifikasi simulasi sebagaimana mestinya. Faktanya, kehidupan simulasi yang Anda jalani mereplikasi persis kehidupan yang akan Anda jalani, seandainya otak Anda tidak ditempatkan di dalam wadah itu, termasuk simulasi makan hamburger simulasi. Dengan demikian, Anda dapat memiliki kondisi otak yang identik dengan seseorang yang benar-benar makan hamburger asli, tetapi akan sangat keliru jika mengatakan bahwa Anda memiliki kondisi mental "mengetahui bahwa seseorang sedang makan hamburger." Anda tidak sedang makan hamburger, Anda bahkan belum pernah merasakan hamburger, dan oleh karena itu, Anda tidak mungkin memiliki kondisi mental seperti itu.

Contoh ini tampaknya bertentangan dengan pandangan bahwa kondisi otak menentukan kondisi mental. Salah satu cara untuk menyelesaikan dilema ini adalah dengan mengatakan bahwa isi kondisi mental dapat diinterpretasikan dari dua sudut pandang yang berbeda. Pandangan "konten luas" menafsirkannya dari sudut pandang pengamat luar yang mahatahu dengan akses ke seluruh situasi, yang dapat membedakan perbedaan di dunia. Dalam pandangan ini, isi kondisi mental melibatkan kondisi otak dan riwayat lingkungan. Di sisi lain, konten sempit hanya mempertimbangkan kondisi otak. Isi sempit kondisi otak pemakan

hamburger sungguhan dan "pemakan" "hamburger" otak-dalam-tong adalah sama dalam kedua kasus.

Konten yang luas sepenuhnya tepat jika tujuan seseorang adalah untuk menghubungkan kondisi mental dengan orang lain yang berbagi dunia yang sama, untuk memprediksi kemungkinan perilaku mereka dan dampaknya, dan sebagainya. Dalam konteks inilah bahasa sehari-hari kita tentang konten mental telah berevolusi. Di sisi lain, jika seseorang mempertanyakan apakah sistem AI benar-benar berpikir dan benar-benar memiliki kondisi mental, maka konten yang sempit tepat; tidak masuk akal untuk mengatakan bahwa apakah suatu sistem AI benar-benar berpikir bergantung pada kondisi di luar sistem tersebut. Konten yang sempit juga relevan jika kita berpikir tentang merancang sistem AI atau memahami operasinya, karena konten sempit dari suatu kondisi otaklah yang menentukan kondisi otak (konten sempit) berikutnya. Hal ini secara alami mengarah pada gagasan bahwa yang penting tentang suatu kondisi otak—yang membuatnya memiliki satu jenis konten mental dan bukan yang lain—adalah peran fungsionalnya dalam operasi mental entitas yang terlibat.

Fungsionalisme dan Eksperimen Penggantian Otak

Teori **fungsionalisme** menyatakan bahwa kondisi mental adalah kondisi kausal antara antara masukan dan keluaran. Berdasarkan teori fungsionalis, dua sistem dengan proses kausal isomorfik akan memiliki kondisi mental yang sama. Oleh karena itu, sebuah program komputer dapat memiliki kondisi mental yang sama dengan seseorang. Tentu saja, kita belum menjelaskan apa arti sebenarnya dari "isomorfik", tetapi asumsinya adalah terdapat suatu tingkat abstraksi di mana implementasi spesifiknya tidak menjadi masalah.

Klaim fungsionalisme diilustrasikan paling jelas oleh eksperimen penggantian otak. Eksperimen pemikiran ini diperkenalkan oleh filsuf Clark Glymour dan disinggung oleh John Searle (1980), tetapi paling sering dikaitkan dengan ahli robotika Hans Moravec (1988). Eksperimennya seperti ini: Misalkan neurofisiologi telah berkembang hingga titik di mana perilaku masukan-keluaran dan konektivitas semua neuron di otak manusia dipahami dengan sempurna. Misalkan lebih lanjut bahwa kita dapat membangun perangkat elektronik mikroskopis yang meniru perilaku ini dan dapat dihubungkan dengan lancar ke jaringan saraf. Terakhir, anggaplah suatu teknik bedah ajaib dapat mengganti neuron-neuron individual dengan perangkat elektronik yang sesuai tanpa mengganggu kerja otak secara keseluruhan. Eksperimen ini terdiri dari penggantian bertahap semua neuron di kepala seseorang dengan perangkat elektronik.

Kita memperhatikan perilaku eksternal dan pengalaman internal subjek, selama dan setelah operasi. Berdasarkan definisi eksperimen ini, perilaku eksternal subjek harus tetap tidak berubah dibandingkan dengan apa yang akan diamati jika operasi tidak dilakukan. Meskipun ada atau tidaknya kesadaran tidak dapat dengan mudah dipastikan oleh pihak ketiga, subjek eksperimen setidaknya harus mampu merekam setiap perubahan dalam pengalaman sadarnya sendiri. Rupanya, terdapat benturan intuisi langsung mengenai apa yang akan terjadi. Moravec, seorang peneliti robotika dan fungsionalis, yakin bahwa kesadarannya tidak akan terpengaruh. Searle, seorang filsuf dan naturalis biologi, juga yakin bahwa kesadarannya akan lenyap:

Anda mendapati, dengan sangat takjub, bahwa Anda memang kehilangan kendali atas perilaku eksternal Anda. Anda mendapati, misalnya, ketika dokter menguji penglihatan Anda, Anda mendengar mereka berkata, "Kami sedang mengangkat benda merah di depan Anda; tolong beri tahu kami apa yang Anda lihat." Anda ingin berteriak, "Saya tidak bisa melihat apa-apa. Saya akan benar-benar buta." Namun, Anda mendengar suara Anda berkata dengan cara yang sama sekali di luar kendali Anda, "Saya melihat benda merah di depan saya." ... pengalaman sadar Anda perlahan-lahan menyusut hingga tak ada, sementara perilaku Anda yang dapat diamati secara eksternal tetap sama.

Kita bisa melakukan lebih dari sekadar berargumen berdasarkan intuisi. Pertama, perhatikan bahwa, agar perilaku eksternal tetap sama sementara subjek perlahan-lahan kehilangan kesadaran, kehendak subjek harus dihilangkan secara instan dan total; jika tidak, penyusutan kesadaran akan tercermin dalam perilaku eksternal—"Tolong, aku menyusut!" atau kata-kata serupa. Penghilangan kehendak secara instan ini sebagai akibat dari penggantian neuron secara bertahap tampaknya mustahil.

Kedua, pertimbangkan apa yang terjadi jika kita mengajukan pertanyaan kepada subjek mengenai pengalaman sadarnya selama periode ketika tidak ada neuron nyata yang tersisa. Berdasarkan kondisi eksperimen, kita akan mendapatkan respons seperti "Saya merasa baik-baik saja. Harus saya akui saya agak terkejut karena saya memercayai argumen Searle." Atau kita mungkin menusuk subjek dengan tongkat runcing dan mengamati respons, "Aduh, sakit." Nah, dalam keadaan normal, orang yang skeptis dapat menganggap keluaran semacam itu dari program AI sebagai rekayasa belaka. Tentu saja, cukup mudah untuk menggunakan aturan seperti "Jika sensor 12 membaca 'Tinggi' maka keluarkan 'Aduh'." Namun intinya di sini adalah, karena kita telah mereplikasi sifat fungsional otak manusia normal, kita berasumsi bahwa otak elektronik tidak mengandung perangkat semacam itu. Maka kita harus memiliki penjelasan tentang manifestasi kesadaran yang dihasilkan oleh otak elektronik yang hanya mengacu pada sifat fungsional neuron. *Dan penjelasan ini juga harus berlaku untuk otak yang sebenarnya, yang memiliki sifat fungsional yang sama.* Ada tiga kemungkinan kesimpulan:

1. Mekanisme kausal kesadaran yang menghasilkan keluaran semacam ini pada otak normal masih beroperasi dalam versi elektronik, yang karenanya bersifat sadar.
2. Peristiwa mental sadar pada otak normal tidak memiliki hubungan kausal dengan perilaku, dan tidak ada pada otak elektronik, yang karenanya tidak sadar.
3. Eksperimen ini mustahil, dan karenanya spekulasi tentangnya tidak ada artinya.

Meskipun kita tidak dapat mengesampingkan kemungkinan kedua, hal itu mereduksi kesadaran menjadi apa yang disebut para filsuf sebagai peran **epifenomenal**—sesuatu yang terjadi, tetapi seolah-olah tidak memberikan bayangan pada dunia yang teramati. Lebih lanjut, jika kesadaran memang epifenomenal, maka subjek tidak mungkin berkata "Aduh" *karena sakit*—yaitu, karena pengalaman sadar akan rasa sakit. Sebaliknya, otak pasti mengandung mekanisme bawah sadar kedua yang bertanggung jawab atas "Aduh".

Patricia Churchland (1986) menunjukkan bahwa argumen fungsionalis yang beroperasi pada tingkat neuron juga dapat beroperasi pada tingkat unit fungsional yang lebih besar—sekelompok neuron, modul mental, lobus, hemisfer, atau seluruh otak. Artinya, jika Anda

menerima gagasan bahwa eksperimen penggantian otak menunjukkan bahwa otak pengganti itu sadar, maka Anda juga harus percaya bahwa kesadaran dipertahankan ketika seluruh otak digantikan oleh sirkuit yang memperbarui statusnya dan memetakan dari masukan ke keluaran melalui tabel pencarian yang sangat besar. Hal ini membingungkan banyak orang (termasuk Turing sendiri), yang memiliki intuisi bahwa tabel pencarian tidaklah sadar—atau setidaknya, bahwa pengalaman sadar yang dihasilkan selama pencarian tabel tidaklah sama dengan pengalaman yang dihasilkan selama pengoperasian suatu sistem yang mungkin digambarkan (bahkan dalam pengertian komputasional yang sederhana) sebagai mengakses dan menghasilkan keyakinan, introspeksi, tujuan, dan sebagainya.

Naturalisme Biologis dan Ruang Tiongkok

Tantangan keras terhadap fungsionalisme telah dilontarkan oleh **naturalisme biologis** John Searle (1980), yang menyatakan bahwa keadaan mental adalah fitur-fitur tingkat tinggi yang muncul yang disebabkan oleh proses fisik tingkat rendah *di dalam neuron*, dan sifat-sifat neuron (yang tidak ditentukan) itulah properti yang penting. Dengan demikian, keadaan mental tidak dapat diduplikasi hanya berdasarkan suatu program yang memiliki struktur fungsional yang sama dengan perilaku masukan-keluaran yang sama; kita mensyaratkan bahwa program tersebut berjalan pada arsitektur dengan daya kausal yang sama dengan neuron. Untuk mendukung pandangannya, Searle menggambarkan sebuah sistem hipotetis yang jelas menjalankan suatu program dan lulus Uji Turing, tetapi juga jelas (menurut Searle) tidak *memahami* apa pun tentang masukan dan keluarannya. Kesimpulannya adalah bahwa menjalankan program yang tepat (yaitu, memiliki keluaran yang tepat) bukanlah syarat yang *cukup* untuk menjadi sebuah pikiran.

Sistem ini terdiri dari seorang manusia, yang hanya mengerti bahasa Inggris, dilengkapi dengan buku aturan, yang ditulis dalam bahasa Inggris, dan berbagai tumpukan kertas, beberapa kosong, beberapa dengan tulisan yang tidak dapat dipahami. (Manusia itu berperan sebagai CPU, buku aturan adalah programnya, dan tumpukan kertas adalah perangkat penyimpanannya.) Sistem ini berada di dalam ruangan dengan lubang kecil ke luar. Melalui lubang itu muncul potongan-potongan kertas dengan simbol yang tidak dapat dipahami. Manusia menemukan simbol yang cocok dalam buku aturan, dan mengikuti instruksinya. Instruksi tersebut dapat mencakup penulisan simbol pada potongan kertas baru, menemukan simbol dalam tumpukan, menyusun ulang tumpukan, dan seterusnya. Pada akhirnya, instruksi tersebut akan menyebabkan satu atau lebih simbol ditranskripsi ke selebar kertas yang diteruskan kembali ke dunia luar.

Sejauh ini, baik-baik saja. Tetapi dari luar, kita melihat sistem yang menerima masukan dalam bentuk kalimat bahasa Mandarin dan menghasilkan jawaban dalam bahasa Mandarin yang sama "cerdasnya" dengan yang ada dalam percakapan yang dibayangkan oleh Turing.⁴ Searle kemudian berpendapat: orang di ruangan itu tidak mengerti bahasa Mandarin (diberikan). Buku aturan dan tumpukan kertas, yang hanya berupa potongan kertas, tidak mengerti bahasa Mandarin. Oleh karena itu, tidak ada pemahaman tentang bahasa Mandarin. *Oleh karena itu, menurut Searle, menjalankan program yang tepat tidak selalu menghasilkan pemahaman.*

Seperti Turing, Searle mempertimbangkan dan mencoba untuk menolak sejumlah balasan atas argumennya. Beberapa komentator, termasuk John McCarthy dan Robert Wilensky, mengusulkan apa yang Searle sebut sebagai balasan sistem. Keberatannya adalah bahwa menanyakan apakah manusia di ruangan itu mengerti bahasa Mandarin sama dengan menanyakan apakah CPU dapat mengambil akar pangkat tiga. Dalam kedua kasus, jawabannya adalah tidak, dan dalam kedua kasus, menurut balasan sistem, seluruh sistem memang memiliki kapasitas yang dimaksud. Tentu saja, jika seseorang bertanya kepada Ruang Bahasa Mandarin apakah mereka memahami bahasa Mandarin, jawabannya adalah ya (dalam bahasa Mandarin yang fasih). Berdasarkan konvensi Turing yang sopan, ini seharusnya sudah cukup. Tanggapan Searle adalah dengan mengulangi poin bahwa pemahaman tidak ada pada manusia dan tidak mungkin ada pada kertas, sehingga tidak mungkin ada pemahaman. Ia tampaknya mengandalkan argumen bahwa sifat keseluruhan harus berada pada salah satu bagiannya. Namun, air bersifat basah, meskipun H maupun O₂ tidak. Klaim sebenarnya yang dibuat oleh Searle bertumpu pada empat aksioma berikut (Searle, 1990):

1. Program komputer bersifat formal (sintaksis).
2. Pikiran manusia memiliki isi mental (semantik).
3. Sintaksis itu sendiri tidak konstitutif atau cukup untuk semantik.
4. Otak menyebabkan pikiran.

Dari tiga aksioma pertama, Searle menyimpulkan bahwa program tidaklah cukup untuk pikiran. Dengan kata lain, agen yang menjalankan program *mungkin* pikiran, *tetapi belum tentu* menjadi pikiran hanya karena menjalankan program tersebut. Dari aksioma keempat, ia menyimpulkan, "Sistem lain apa pun yang mampu menyebabkan pikiran harus memiliki daya kausal (setidaknya) yang setara dengan otak." Dari sana, ia menyimpulkan bahwa otak buatan apa pun harus menduplikasi daya kausal otak, bukan hanya menjalankan program tertentu, dan bahwa otak manusia tidak menghasilkan fenomena mental hanya karena menjalankan program.

Aksioma-aksioma ini kontroversial. Misalnya, aksioma 1 dan 2 bergantung pada perbedaan yang tidak spesifik antara sintaksis dan semantik yang tampaknya berkaitan erat dengan perbedaan antara konten sempit dan luas. Di satu sisi, kita dapat memandang komputer sebagai pihak yang memanipulasi simbol sintaksis; di sisi lain, kita dapat memandangnya sebagai pihak yang memanipulasi arus listrik, yang kebetulan merupakan hal yang paling sering dilakukan otak (menurut pemahaman kita saat ini). Jadi, tampaknya kita juga bisa mengatakan bahwa otak bersifat sintaksis.

Dengan asumsi kita bermurah hati dalam menafsirkan aksioma-aksioma tersebut, maka kesimpulannya—bahwa program tidak cukup untuk pikiran— *memang* mengikuti. Namun kesimpulannya tidak memuaskan—semua yang ditunjukkan Searle adalah jika Anda secara eksplisit menyangkal fungsionalisme (itulah yang dilakukan aksioma 3-nya), maka Anda belum tentu dapat menyimpulkan bahwa non-otak adalah pikiran. Ini cukup masuk akal—hampir tautologis—sehingga seluruh argumen bermuara pada apakah aksioma 3 dapat diterima. Menurut Searle, inti dari argumen Ruang Cina adalah untuk menyediakan intuisi bagi aksioma 3. Reaksi publik menunjukkan bahwa argumen tersebut bertindak sebagai apa yang

Daniel Dennett (1991) sebut sebagai **pompa intuisi**: argumen tersebut memperkuat intuisi sebelumnya, sehingga para naturalis biologis lebih yakin dengan posisi mereka, dan para fungsionalis hanya yakin bahwa aksioma 3 tidak didukung, atau bahwa secara umum argumen Searle tidak meyakinkan. Argumen ini memicu perdebatan, tetapi tidak banyak mengubah pendapat siapa pun. Searle tetap teguh, dan baru-baru ini mulai menyebut Ruang Cina sebagai "sanggahan" terhadap AI yang kuat, alih-alih sekadar "argumen".

Bahkan mereka yang menerima aksioma 3, dan dengan demikian menerima argumen Searle, hanya mengandalkan intuisi mereka ketika memutuskan entitas mana yang merupakan pikiran. Argumen ini dimaksudkan untuk menunjukkan bahwa Ruang Cina bukanlah *pikiran karena menjalankan programnya*, tetapi argumen tersebut tidak menjelaskan cara memutuskan apakah ruang tersebut (atau komputer, jenis mesin lain, atau alien) adalah *pikiran karena alasan lain*. Searle sendiri mengatakan bahwa beberapa mesin memang memiliki pikiran: manusia adalah mesin biologis yang memiliki pikiran. Menurut Searle, otak manusia mungkin menjalankan atau tidak menjalankan sesuatu seperti program AI, tetapi jika memang demikian, bukan itu alasan mereka menjadi pikiran. Dibutuhkan lebih banyak hal untuk membentuk pikiran—menurut Searle, sesuatu yang setara dengan kekuatan kausal neuron individual. Kekuatan-kekuatan ini tidak dijelaskan secara rinci. Namun, perlu dicatat bahwa neuron berevolusi untuk memenuhi peran *fungsional*—makhluk dengan neuron belajar dan memutuskan jauh sebelum kesadaran muncul. Akan menjadi kebetulan yang luar biasa jika neuron semacam itu kebetulan menghasilkan kesadaran karena beberapa kekuatan kausal yang tidak relevan dengan kemampuan fungsionalnya; lagipula, kemampuan fungsional yang menentukan kelangsungan hidup organisme.

Dalam kasus Ruang Cina, Searle mengandalkan intuisi, bukan bukti: lihat saja ruangan itu; apakah arti sebuah pikiran? Namun, orang dapat mengajukan argumen yang sama tentang otak: lihat saja kumpulan sel (atau atom) ini, yang beroperasi secara membabi buta menurut hukum biokimia (atau fisika)—apakah arti sebuah pikiran? Mengapa seongkah otak bisa menjadi pikiran sementara seongkah hati tidak? Itulah yang tetap menjadi misteri besar.

Kesadaran, qualia, dan kesenjangan penjelasan

Melewati semua perdebatan tentang AI yang kuat—yang menjadi masalah besar dalam perdebatan—adalah isu **kesadaran**. Kesadaran sering dipecah menjadi aspek-aspek seperti pemahaman dan kesadaran diri. Aspek yang akan kita fokuskan adalah *pengalaman subjektif*: mengapa *rasanya* seperti memiliki keadaan otak tertentu (misalnya, saat makan hamburger), sedangkan rasanya seperti tidak memiliki keadaan fisik lain (misalnya, saat menjadi batu). Istilah teknis untuk sifat intrinsik pengalaman adalah **qualia** (dari kata Latin yang secara kasar berarti "hal-hal seperti itu").

Qualia menghadirkan tantangan bagi penjelasan fungsionalis tentang pikiran karena qualia yang berbeda dapat terlibat dalam proses kausal yang isomorfik. Pertimbangkan, misalnya, eksperimen pemikiran **spektrum terbalik**, di mana pengalaman subjektif orang *X* ketika melihat objek merah adalah pengalaman yang sama dengan yang kita alami ketika melihat objek hijau, dan sebaliknya. *X* masih menyebut objek merah sebagai "merah", berhenti saat lampu lalu lintas merah, dan setuju bahwa warna merah lampu lalu lintas merah

lebih intens daripada warna merah matahari terbenam. Namun, pengalaman subjektif X justru berbeda.

Kualia menantang bukan hanya bagi fungsionalisme, tetapi juga bagi seluruh ilmu pengetahuan. Anggaplah, demi argumen, bahwa kita telah menyelesaikan proses penelitian ilmiah tentang otak—kita telah menemukan bahwa proses saraf P_{12} pada neuron N_{177} mengubah molekul A menjadi molekul B , dan seterusnya. Saat ini, tidak ada bentuk penalaran yang diterima secara umum yang akan mengarah dari temuan tersebut pada kesimpulan bahwa entitas yang memiliki neuron-neuron tersebut memiliki pengalaman subjektif tertentu. **Kesenjangan penjelasan** ini telah menyebabkan beberapa filsuf menyimpulkan bahwa manusia tidak mampu membentuk pemahaman yang tepat tentang kesadaran mereka sendiri. Yang lain, terutama Daniel Dennett (1991), menghindari kesenjangan tersebut dengan menyangkal keberadaan kualia, mengaitkannya dengan kebingungan filosofis.

Turing sendiri mengakui bahwa pertanyaan tentang kesadaran merupakan pertanyaan yang sulit, tetapi menyangkal bahwa hal itu memiliki relevansi yang besar dengan praktik AI: "Saya tidak ingin memberi kesan bahwa saya pikir tidak ada misteri tentang kesadaran ... Tetapi saya tidak berpikir misteri-misteri ini perlu dipecahkan sebelum kita dapat menjawab pertanyaan yang kita bahas dalam makalah ini." Kami sependapat dengan Turing—kami tertarik untuk menciptakan program yang berperilaku cerdas. Proyek tambahan untuk menyadarkan mereka bukanlah proyek yang siap kami lakukan, dan juga bukan proyek yang keberhasilannya dapat kami tentukan.

8.3 ETIKA DAN RISIKO PENGEMBANGAN KECERDASAN BUATAN

Sejauh ini, kita telah berfokus pada apakah kita *dapat* mengembangkan AI, tetapi kita juga harus mempertimbangkan apakah kita *harus* melakukannya. Jika dampak teknologi AI lebih cenderung negatif daripada positif, maka menjadi tanggung jawab moral para pekerja di lapangan untuk mengarahkan kembali penelitian mereka. Banyak teknologi baru memiliki dampak samping negatif yang tidak diinginkan: fisi nuklir membawa Chernobyl dan ancaman kehancuran global; mesin pembakaran internal membawa polusi udara, pemanasan global, dan pengaspalan surga. Dalam arti tertentu, mobil adalah robot yang telah menaklukkan dunia dengan menjadikan diri mereka tak tergantikan.

Semua ilmuwan dan insinyur menghadapi pertimbangan etis tentang bagaimana mereka harus bertindak dalam pekerjaan, proyek apa yang boleh atau tidak boleh dilakukan, dan bagaimana proyek tersebut harus ditangani. Lihat buku pegangan tentang *Etika Komputasi*. Namun, AI tampaknya menimbulkan beberapa masalah baru di luar, misalnya, membangun jembatan yang tidak runtuh:

- Orang-orang mungkin kehilangan pekerjaan karena otomatisasi.
- Orang-orang mungkin memiliki terlalu banyak (atau terlalu sedikit) waktu luang.
- Orang-orang mungkin kehilangan rasa keunikan mereka.
- Sistem AI mungkin digunakan untuk tujuan yang tidak diinginkan.
- Penggunaan sistem AI dapat mengakibatkan hilangnya akuntabilitas.
- Keberhasilan AI dapat berarti akhir dari umat manusia.

Kita akan membahas setiap masalah secara bergantian.

Orang-orang mungkin kehilangan pekerjaan karena otomatisasi.

Ekonomi industri modern telah menjadi bergantung pada komputer secara umum, dan khususnya pada program AI tertentu. Misalnya, sebagian besar perekonomian, terutama di Amerika Serikat, bergantung pada ketersediaan kredit konsumen. Aplikasi kartu kredit, persetujuan tagihan, dan deteksi penipuan kini dilakukan oleh program AI. Bisa dikatakan ribuan pekerja telah tergantikan oleh program-program AI ini, tetapi kenyataannya, jika program AI dihapuskan, pekerjaan-pekerjaan ini tidak akan ada, karena tenaga kerja manusia akan menambah biaya transaksi yang tidak dapat diterima. Sejauh ini, otomatisasi melalui teknologi informasi secara umum dan AI khususnya telah menciptakan lebih banyak pekerjaan daripada yang dihilangkannya, dan telah menciptakan pekerjaan yang lebih menarik dan bergaji lebih tinggi. Kini, karena program AI kanonik adalah "agen cerdas" yang dirancang untuk membantu manusia, hilangnya pekerjaan tidak lagi menjadi masalah dibandingkan ketika AI berfokus pada "sistem pakar" yang dirancang untuk menggantikan manusia. Namun, beberapa peneliti berpendapat bahwa menyelesaikan pekerjaan secara menyeluruh adalah tujuan yang tepat untuk AI. Dalam merefleksikan Peringatan 25 Tahun AAI, Nils Nilsson (2005) menetapkan tantangan untuk menciptakan AI setingkat manusia yang dapat lulus uji ketenagakerjaan, alih-alih Uji Turing—sebuah robot yang dapat belajar melakukan berbagai pekerjaan. Kita mungkin akan berakhir di masa depan dengan tingkat pengangguran yang tinggi, tetapi bahkan para penganggur pun dapat menjadi manajer bagi kelompok pekerja robot mereka sendiri.

Orang-orang mungkin memiliki terlalu banyak (atau terlalu sedikit) waktu luang.

Alvin Toffler menulis dalam *Future Shock* (1970), "Jam kerja telah berkurang 50 persen sejak pergantian abad. Tidaklah mustahil untuk memprediksi bahwa jam kerja akan berkurang setengahnya lagi pada tahun 2000." Arthur C. Clarke (1968b) menulis bahwa orang-orang pada tahun 2001 mungkin "dihadapkan dengan masa depan yang sangat membosankan, di mana masalah utama dalam hidup adalah memutuskan saluran TV mana yang akan dipilih dari ratusan saluran." Satu-satunya prediksi yang hampir terbukti adalah jumlah saluran TV. Sebaliknya, orang-orang yang bekerja di industri yang padat pengetahuan telah menjadi bagian dari sistem komputerisasi terintegrasi yang beroperasi 24 jam sehari; agar tetap bertahan, mereka terpaksa bekerja *lebih lama*. Dalam ekonomi industri, imbalan secara kasar proporsional dengan waktu yang diinvestasikan; bekerja 10% lebih banyak cenderung berarti peningkatan pendapatan sebesar 10%. Dalam ekonomi informasi yang ditandai dengan komunikasi bandwidth tinggi dan replikasi kekayaan intelektual yang mudah (yang disebut Frank dan Cook (1996) sebagai "Masyarakat Pemenang-Ambil-Semua"), ada imbalan besar untuk menjadi sedikit lebih baik daripada pesaing; bekerja 10% lebih banyak dapat berarti peningkatan pendapatan sebesar 100%. Jadi, ada tekanan yang semakin besar pada setiap orang untuk bekerja lebih keras. AI meningkatkan laju inovasi teknologi dan dengan demikian berkontribusi pada tren keseluruhan ini, tetapi AI juga menjanjikan untuk memungkinkan kita beristirahat sejenak dan membiarkan agen otomatis kita menangani berbagai hal untuk

sementara waktu. Tim Ferriss (2007) merekomendasikan penggunaan otomatisasi dan alih daya untuk mencapai minggu kerja empat jam.

Orang-orang mungkin kehilangan rasa keunikan mereka. Dalam *Computer Power and Human Reason*, Weizenbaum (1976), penulis program ELIZA, menunjukkan beberapa potensi ancaman yang ditimbulkan AI terhadap masyarakat. Salah satu argumen utama Weizenbaum adalah bahwa penelitian AI memungkinkan gagasan bahwa manusia adalah automata—sebuah gagasan yang mengakibatkan hilangnya otonomi atau bahkan kemanusiaan. Perlu dicatat bahwa gagasan tersebut telah ada jauh lebih lama daripada AI, setidaknya sejak *L'Homme Machine* (La Mettrie, 1748). Umat manusia telah melewati kemunduran lain dalam rasa keunikan kita: *De Revolutionibus Orbium Coelestium* (Copernicus, 1543) menjauhkan Bumi dari pusat tata surya, dan *Descent of Man* (Darwin, 1871) menempatkan *Homo sapiens* pada tingkat yang sama dengan spesies lain. AI, jika berhasil secara luas, mungkin setidaknya sama mengancamnya bagi asumsi moral masyarakat abad ke-21 seperti teori evolusi Darwin bagi asumsi moral masyarakat abad ke-19.

Sistem AI mungkin digunakan untuk tujuan yang tidak diinginkan. Teknologi canggih sering kali digunakan oleh pihak yang berkuasa untuk menekan saingan mereka. Sebagaimana ditulis oleh ahli teori bilangan G. H. Hardy (Hardy, 1940), "Suatu ilmu dikatakan bermanfaat jika perkembangannya cenderung memperparah ketimpangan yang ada dalam distribusi kekayaan, atau secara lebih langsung mendorong penghancuran kehidupan manusia." Hal ini berlaku untuk semua ilmu pengetahuan, tidak terkecuali AI. Sistem AI otonom kini menjadi hal yang lumrah di medan perang; militer AS mengerahkan lebih dari 5.000 pesawat otonom dan 12.000 kendaraan darat otonom di Irak. Salah satu teori moral menyatakan bahwa robot militer ibarat baju zirah abad pertengahan yang dibawa ke ekstrem logisnya: tak seorang pun akan keberatan secara moral terhadap seorang prajurit yang ingin mengenakan helm saat diserang musuh besar, pemarah, dan bersenjata kapak, dan robot yang dioperasikan jarak jauh ibarat baju zirah yang sangat aman. Di sisi lain, senjata robotik menimbulkan risiko tambahan. Sejauh pengambilan keputusan manusia dikesampingkan dari lingkaran tembak-menembak, robot dapat berakhir membuat keputusan yang berujung pada terbunuhnya warga sipil tak berdosa. Dalam skala yang lebih besar, kepemilikan robot yang kuat (seperti kepemilikan helm yang kokoh) dapat membuat suatu bangsa terlalu percaya diri, menyebabkannya berperang lebih gegabah daripada yang seharusnya. Dalam sebagian besar perang, setidaknya satu pihak terlalu percaya diri dengan kemampuan militernya—jika tidak, konflik tersebut akan diselesaikan secara damai.

Weizenbaum (1976) juga menunjukkan bahwa teknologi pengenalan suara dapat menyebabkan penyadapan yang meluas, dan dengan demikian mengakibatkan hilangnya kebebasan sipil. Ia tidak meramalkan dunia dengan ancaman teroris yang akan mengubah keseimbangan seberapa banyak pengawasan yang bersedia diterima orang, tetapi ia dengan tepat menyadari bahwa AI berpotensi memproduksi pengawasan secara massal. Prediksinya sebagian menjadi kenyataan: Inggris kini memiliki jaringan kamera pengawas yang luas, dan negara-negara lain secara rutin memantau lalu lintas web dan panggilan telepon. Beberapa orang sepakat bahwa komputerisasi menyebabkan hilangnya privasi—CEO Sun Microsystems,

Scott McNealy, pernah berkata, "Lagipula Anda tidak punya privasi. Lupakan saja." David Brin (1998) berpendapat bahwa hilangnya privasi tidak dapat dihindari, dan cara untuk mengatasi asimetri kekuasaan negara atas individu adalah dengan membuat pengawasan dapat diakses oleh semua warga negara. Etzioni (2004) berpendapat bahwa privasi dan keamanan harus diseimbangkan; hak individu dan komunitas harus diseimbangkan.

Penggunaan sistem AI dapat mengakibatkan hilangnya akuntabilitas. Dalam atmosfer litigasi yang berlaku di Amerika Serikat, tanggung jawab hukum menjadi isu penting. Ketika seorang dokter mengandalkan penilaian sistem pakar medis untuk diagnosis, siapa yang bersalah jika diagnosisnya salah? Untungnya, sebagian karena semakin besarnya pengaruh metode teori keputusan dalam kedokteran, kini diterima bahwa kelalaian tidak dapat dibuktikan jika dokter melakukan prosedur medis yang memiliki utilitas yang *diharapkan* tinggi, bahkan jika hasil *aktualnya* sangat buruk bagi pasien. Oleh karena itu, pertanyaannya seharusnya "Siapa yang salah jika diagnosisnya tidak masuk akal?" Sejauh ini, pengadilan telah memutuskan bahwa sistem pakar medis memainkan peran yang sama dengan buku teks dan buku referensi medis; dokter bertanggung jawab untuk memahami alasan di balik setiap keputusan dan untuk menggunakan penilaian mereka sendiri dalam memutuskan apakah akan menerima rekomendasi sistem. Oleh karena itu, dalam merancang sistem pakar medis sebagai agen, tindakan harus dianggap tidak secara langsung memengaruhi pasien tetapi memengaruhi perilaku dokter. Jika sistem pakar menjadi lebih akurat dan andal daripada ahli diagnosa manusia, dokter mungkin bertanggung jawab secara hukum jika mereka *tidak* menggunakan rekomendasi sistem pakar. Atul Gawande (2002) mengeksplorasi premis ini.

Isu serupa mulai muncul terkait penggunaan agen cerdas di Internet. Beberapa kemajuan telah dicapai dalam menggabungkan batasan ke dalam agen cerdas sehingga mereka tidak dapat, misalnya, merusak berkas pengguna lain (Weld dan Etzioni, 1994). Masalah ini diperbesar ketika uang berpindah tangan. Jika transaksi moneter dilakukan "atas nama seseorang" oleh agen cerdas, apakah seseorang bertanggung jawab atas utang yang timbul? Mungkinkah agen cerdas memiliki aset sendiri dan melakukan perdagangan elektronik atas namanya sendiri? Sejauh ini, pertanyaan-pertanyaan ini tampaknya belum dipahami dengan baik. Sepengetahuan kami, tidak ada program yang diberikan status hukum sebagai individu untuk tujuan transaksi keuangan; saat ini, tampaknya tidak masuk akal untuk melakukannya. Program juga tidak dianggap sebagai "penggerak" untuk tujuan penegakan peraturan lalu lintas di jalan raya sungguhan. Setidaknya dalam hukum California, tampaknya tidak ada sanksi hukum untuk mencegah kendaraan otomatis melampaui batas kecepatan, meskipun perancang mekanisme kendali kendaraan akan bertanggung jawab jika terjadi kecelakaan. Sebagaimana halnya teknologi reproduksi manusia, hukum belum mampu mengimbangi perkembangan baru.

Keberhasilan AI mungkin berarti akhir dari umat manusia. Hampir semua teknologi berpotensi membahayakan di tangan yang salah, tetapi dengan AI dan robotika, kita menghadapi masalah baru bahwa teknologi itu sendiri mungkin berada di tangan yang salah. Banyak sekali cerita fiksi ilmiah yang telah memperingatkan tentang robot atau cyborg robot-manusia yang mengamuk. Contoh-contoh awal termasuk *Frankenstein, or the Modern*

Prometheus (1818) karya Mary Shelley dan drama *R.U.R.* (1921) karya Karel Capek, yang menceritakan tentang robot yang menaklukkan dunia. Dalam film, kita punya *The Terminator* (1984), yang menggabungkan klise robot-menaklukkan-dunia dengan perjalanan waktu, dan *The Matrix* (1999), yang menggabungkan robot-menaklukkan-dunia dengan otak-dalam-tong.

Tampaknya robot menjadi protagonis dalam begitu banyak kisah penaklukan-dunia karena mereka mewakili hal yang tidak diketahui, seperti penyihir dan hantu dalam kisah-kisah dari era sebelumnya, atau orang Mars dari *The War of the Worlds*. Pertanyaannya adalah apakah sistem AI menimbulkan risiko yang lebih besar daripada perangkat lunak tradisional. Kita akan melihat tiga sumber risiko.

Pertama, estimasi keadaan sistem AI mungkin tidak tepat, sehingga menyebabkannya melakukan hal yang salah. Misalnya, mobil otonom mungkin salah memperkirakan posisi mobil di jalur sebelahnya, yang mengakibatkan kecelakaan yang dapat menewaskan penumpangnya. Lebih serius lagi, sistem pertahanan rudal mungkin salah mendeteksi serangan dan meluncurkan serangan balik, yang mengakibatkan kematian miliaran orang. Risiko-risiko ini sebenarnya bukan risiko sistem AI—dalam kedua kasus, kesalahan yang sama dapat dengan mudah dilakukan oleh manusia maupun komputer. Cara yang tepat untuk memitigasi risiko-risiko ini adalah dengan merancang sistem dengan mekanisme pengawasan dan keseimbangan sehingga satu kesalahan estimasi keadaan tidak menyebar ke seluruh sistem tanpa terkendali.

Kedua, menentukan fungsi utilitas yang tepat agar sistem AI dapat memaksimalkannya tidaklah mudah. Misalnya, kita dapat mengusulkan fungsi utilitas yang dirancang untuk *meminimalkan penderitaan manusia*, yang dinyatakan sebagai fungsi imbalan aditif dari waktu ke waktu. Namun, mengingat cara manusia, kita akan selalu menemukan cara untuk menderita bahkan di surga; Jadi, keputusan optimal untuk sistem AI adalah memusnahkan umat manusia sesegera mungkin—tanpa manusia, tanpa penderitaan. Dengan sistem AI, kita perlu sangat berhati-hati dengan apa yang kita minta, sementara manusia tidak akan kesulitan menyadari bahwa fungsi utilitas yang diusulkan tidak dapat diartikan secara harfiah. Di sisi lain, komputer tidak perlu dinodai oleh perilaku irasional. Manusia terkadang menggunakan kecerdasannya dengan cara yang agresif karena manusia memiliki beberapa kecenderungan agresif bawaan, akibat seleksi alam. Mesin yang kita bangun tidak perlu agresif bawaan, kecuali kita memutuskan untuk membangunnya seperti itu (atau kecuali mereka muncul sebagai produk akhir dari desain mekanisme yang mendorong perilaku agresif). Untungnya, ada teknik, seperti pembelajaran magang, yang memungkinkan kita menentukan fungsi utilitas melalui contoh. Kita dapat berharap bahwa robot yang cukup pintar untuk mengetahui cara memusnahkan umat manusia juga cukup pintar untuk mengetahui bahwa itu bukanlah fungsi utilitas yang dimaksudkan.

Ketiga, fungsi pembelajaran sistem AI dapat menyebabkannya berevolusi menjadi sistem dengan perilaku yang tidak diinginkan. Skenario ini adalah yang paling serius, dan unik untuk sistem AI, jadi kami akan membahasnya lebih mendalam. I. J. Good menulis (1965),

*Mari kita definisikan **mesin ultraintelijen** sebagai mesin yang mampu jauh melampaui semua aktivitas intelektual manusia, betapapun cerdasnya. Karena merancang mesin merupakan salah satu aktivitas intelektual tersebut, sebuah mesin ultraintelijen dapat merancang mesin yang bahkan lebih baik lagi; dengan begitu, niscaya akan terjadi "ledakan kecerdasan", dan kecerdasan manusia akan jauh tertinggal. Dengan demikian, mesin ultraintelijen pertama adalah penemuan terakhir yang perlu dibuat manusia, asalkan mesin tersebut cukup jinak untuk memberi tahu kita cara mengendalikannya.*

"Ledakan kecerdasan" juga disebut **singularitas teknologi** oleh profesor matematika dan penulis fiksi ilmiah Vernor Vinge, yang menulis (1993), "Dalam tiga puluh tahun, kita akan memiliki sarana teknologi untuk menciptakan kecerdasan super. Tak lama setelah itu, era manusia akan berakhir." Good dan Vinge (dan banyak lainnya) dengan tepat mencatat bahwa kurva kemajuan teknologi (dalam banyak ukuran) tumbuh secara eksponensial saat ini (pertimbangkan Hukum Moore). Namun, terlalu jauh untuk memperkirakan bahwa kurva tersebut akan terus berlanjut hingga singularitas pertumbuhan yang hampir tak terbatas. Sejauh ini, setiap teknologi lain mengikuti kurva berbentuk S, di mana pertumbuhan eksponensial pada akhirnya mereda. Terkadang teknologi baru muncul ketika teknologi lama mencapai titik jenuh; terkadang kita mencapai batas yang sulit. Dengan sejarah teknologi tinggi yang kurang dari satu abad, sulit untuk memperkirakan ratusan tahun ke depan.

Perhatikan bahwa konsep mesin ultraintelijen mengasumsikan bahwa kecerdasan adalah atribut yang sangat penting, dan jika kecerdasan tersebut cukup, semua masalah dapat dipecahkan. Namun, kita tahu ada batasan pada komputabilitas dan kompleksitas komputasi. Jika masalah pendefinisian mesin ultraintelijen (atau bahkan perkiraannya) kebetulan termasuk dalam kelas, katakanlah, masalah NEXPTIME-lengkap, dan jika tidak ada jalan pintas heuristik, maka kemajuan teknologi eksponensial pun tidak akan membantu—kecepatan cahaya menetapkan batas atas yang ketat pada seberapa banyak komputasi yang dapat dilakukan; masalah di luar batas itu tidak akan terpecahkan. Kita masih belum tahu di mana batas atas tersebut.

Vinge mengkhawatirkan singularitas yang akan datang, tetapi beberapa ilmuwan komputer dan futuris menyukainya. Hans Moravec (2000) mendorong kita untuk memberikan setiap keuntungan kepada "anak-anak pikiran" kita, robot yang kita ciptakan, yang mungkin melampaui kita dalam hal kecerdasan. Bahkan ada istilah baru—**transhumanisme**—untuk gerakan sosial aktif yang menantikan masa depan di mana manusia akan menyatu dengan—atau digantikan oleh—penemuan-penemuan robotik dan bioteknologi. Cukupilah untuk mengatakan bahwa isu-isu semacam itu menghadirkan tantangan bagi sebagian besar ahli teori moral, yang menganggap pelestarian kehidupan manusia dan spesies manusia sebagai hal yang baik. Ray Kurzweil saat ini merupakan pendukung paling terkemuka dari pandangan singularitas, menulis dalam *The Singularity is Near* (2005):

Singularitas akan memungkinkan kita melampaui keterbatasan tubuh dan otak biologis kita. Kita akan memperoleh kendali atas takdir kita. Kematian kita akan berada di tangan kita sendiri. Kita akan mampu hidup selama yang kita inginkan (pernyataan yang sedikit berbeda dari mengatakan kita akan hidup selamanya). Kita akan sepenuhnya memahami

pemikiran manusia dan akan memperluas jangkauannya secara luas. Pada akhir abad ini, bagian nonbiologis dari kecerdasan kita akan menjadi triliunan triliunan kali lebih kuat daripada kecerdasan manusia tanpa bantuan.

Kurzweil juga mencatat potensi bahayanya, menulis, "Namun Singularitas juga akan memperkuat kemampuan untuk bertindak berdasarkan kecenderungan destruktif kita, sehingga kisah lengkapnya belum ditulis."

Jika mesin ultraintelijen memungkinkan, kita manusia sebaiknya memastikan bahwa kita merancang pendahulu mereka sedemikian rupa sehingga mereka merancang diri mereka sendiri untuk memperlakukan kita dengan baik. Penulis fiksi ilmiah Isaac Asimov (1942) adalah orang pertama yang membahas masalah ini, dengan tiga hukum robotikanya:

1. Robot tidak boleh melukai manusia atau, melalui ketidakaktifan, membiarkan manusia terluka.
2. Robot harus mematuhi perintah yang diberikan oleh manusia, kecuali jika perintah tersebut bertentangan dengan Hukum Pertama.
3. Robot harus melindungi keberadaannya sendiri selama perlindungan tersebut tidak bertentangan dengan Hukum Pertama atau Kedua.

Hukum-hukum ini tampak masuk akal, setidaknya bagi kita manusia.⁶ Namun, kuncinya adalah bagaimana menerapkan hukum-hukum ini. Dalam cerita Asimov *Roundabout*, sebuah robot dikirim untuk mengambil selenium. Kemudian, robot tersebut ditemukan berkeliaran dalam lingkaran di sekitar sumber selenium. Setiap kali robot tersebut menuju ke sumber tersebut, ia merasakan bahaya, dan hukum ketiga menyebabkannya menyimpang. Namun, setiap kali robot tersebut menyimpang, bahaya tersebut surut, dan kekuatan hukum kedua mengambil alih, menyebabkannya menyimpang kembali ke arah selenium. Kumpulan titik yang menentukan titik keseimbangan antara kedua hukum tersebut membentuk sebuah lingkaran. Hal ini menunjukkan bahwa hukum-hukum tersebut bukanlah sesuatu yang mutlak secara logis, melainkan saling berimbang, dengan bobot yang lebih tinggi untuk hukum-hukum sebelumnya. Asimov mungkin sedang memikirkan arsitektur yang didasarkan pada teori kontrol—mungkin kombinasi linear berbagai faktor—sementara saat ini arsitektur yang paling mungkin adalah agen penalaran probabilistik yang bernalar berdasarkan distribusi probabilitas hasil, dan memaksimalkan utilitas sebagaimana didefinisikan oleh ketiga hukum tersebut. Namun, agaknya kita tidak ingin robot kita mencegah manusia menyeberang jalan karena peluang bahaya yang bukan nol. Artinya, utilitas negatif untuk bahaya bagi manusia harus jauh lebih besar daripada untuk ketidakpatuhan, tetapi masing-masing utilitas tersebut terbatas, bukan tak terbatas.

Yudkowsky (2008) membahas lebih detail tentang cara merancang **AI yang Ramah**. Ia menegaskan bahwa keramahan (keinginan untuk tidak menyakiti manusia) harus dirancang sejak awal, tetapi para perancang harus menyadari bahwa desain mereka sendiri mungkin memiliki kekurangan, dan bahwa robot akan belajar dan berkembang seiring waktu. Dengan demikian, tantangannya adalah desain mekanisme—mendefinisikan mekanisme untuk

mengembangkan sistem AI di bawah sistem pengawasan dan keseimbangan, dan memberikan fungsi utilitas pada sistem yang akan tetap ramah dalam menghadapi perubahan tersebut.

Kita tidak bisa begitu saja memberikan sebuah program fungsi utilitas statis, karena keadaan, dan respons yang kita inginkan terhadap keadaan, berubah seiring waktu. Misalnya, jika teknologi memungkinkan kita merancang agen AI superkuat pada tahun 1800 dan memberinya moral yang berlaku saat itu, ia akan berjuang hari ini untuk menegakkan kembali perbudakan dan menghapuskan hak pilih perempuan. Di sisi lain, jika kita membangun agen AI hari ini dan memintanya untuk mengembangkan fungsi utilitasnya, bagaimana kita bisa memastikan bahwa ia tidak akan bernalar bahwa "Manusia menganggap membunuh serangga yang mengganggu itu bermoral, sebagian karena otak serangga sangat primitif. Tetapi otak manusia primitif dibandingkan dengan kekuatan saya, jadi pastilah bermoral bagi saya untuk membunuh manusia."

Omohundro (2008) berhipotesis bahwa bahkan program catur yang tidak berbahaya pun dapat menimbulkan risiko bagi masyarakat. Demikian pula, Marvin Minsky pernah berpendapat bahwa program AI yang dirancang untuk memecahkan Hipotesis Riemann mungkin akan menghabiskan seluruh sumber daya Bumi untuk membangun superkomputer yang lebih kuat guna membantu mencapai tujuannya. Moralnya adalah, meskipun Anda hanya ingin program Anda bermain catur atau membuktikan teorema, jika Anda memberinya kemampuan untuk belajar dan mengubah dirinya sendiri, Anda memerlukan perlindungan. Omohundro menyimpulkan bahwa "Struktur sosial yang membuat individu menanggung biaya eksternalitas negatif mereka akan sangat membantu dalam memastikan masa depan yang stabil dan positif." Hal ini tampaknya merupakan ide yang sangat baik bagi masyarakat secara umum, terlepas dari kemungkinan adanya mesin ultraintelijen.

Perlu dicatat bahwa gagasan perlindungan terhadap perubahan fungsi utilitas bukanlah hal baru. Dalam *Odyssey*, Homer (sekitar 700 SM) menggambarkan pertemuan Ulysses dengan sirene, yang nyanyiannya begitu memikat sehingga memaksa para pelaut untuk menceburkan diri ke laut. Mengetahui hal itu akan berdampak padanya, Ulysses memerintahkan krunya untuk mengikatnya ke tiang kapal agar ia tidak dapat melakukan tindakan penghancuran diri. Menarik untuk memikirkan bagaimana perlindungan serupa dapat dibangun ke dalam sistem AI.

Terakhir, mari kita pertimbangkan sudut pandang robot. Jika robot menjadi sadar, maka memperlakukan mereka hanya sebagai "mesin" (misalnya, membongkarnya) mungkin tidak bermoral. Para penulis fiksi ilmiah telah membahas isu hak-hak robot. Film A.I. (Spielberg, 2001) didasarkan pada cerita karya Brian Aldiss tentang robot cerdas yang diprogram untuk percaya bahwa ia adalah manusia dan gagal memahami pengabaian dirinya oleh pemilik sekaligus ibunya. Cerita (dan filmnya) berargumen tentang perlunya gerakan hak-hak sipil bagi robot.

8.4 RINGKASAN

Bab ini telah membahas isu-isu berikut:

- Para filsuf menggunakan istilah **AI lemah** untuk hipotesis bahwa mesin mungkin dapat berperilaku cerdas, dan **AI kuat** untuk hipotesis bahwa mesin tersebut dapat dianggap memiliki pikiran yang nyata (berbeda dengan pikiran yang disimulasikan).
- Alan Turing menolak pertanyaan "Bisakah mesin berpikir?" dan menggantinya dengan uji perilaku. Ia mengantisipasi banyak keberatan terhadap kemungkinan adanya mesin yang berpikir. Hanya sedikit peneliti AI yang memperhatikan Uji Turing, lebih memilih untuk berkonsentrasi pada kinerja sistem mereka dalam tugas-tugas praktis, daripada kemampuan untuk meniru manusia.
- Ada kesepakatan umum di zaman modern bahwa kondisi mental adalah kondisi otak.
- Argumen yang mendukung dan menentang AI kuat tidak meyakinkan. Hanya sedikit peneliti AI arus utama yang percaya bahwa sesuatu yang signifikan bergantung pada hasil perdebatan ini.
- Kesadaran tetap menjadi misteri.
- Kami mengidentifikasi enam potensi ancaman terhadap masyarakat yang ditimbulkan oleh AI dan teknologi terkait. Kami menyimpulkan bahwa beberapa ancaman tersebut kemungkinan kecil terjadi atau hanya sedikit berbeda dari ancaman yang ditimbulkan oleh teknologi "tidak cerdas". Satu ancaman khususnya patut dipertimbangkan lebih lanjut: bahwa mesin ultracerdas dapat mengarah pada masa depan yang sangat berbeda dari saat ini—kita mungkin tidak menyukainya, dan pada saat itu kita mungkin tidak punya pilihan. Pertimbangan semacam itu mau tidak mau mengarah pada kesimpulan bahwa kita harus mempertimbangkan dengan cermat, dan segera, kemungkinan konsekuensi dari penelitian AI.

BAB 9

AI: MASA KINI DAN MASA DEPAN

Dalam Bab 2, kami menyarankan bahwa akan bermanfaat untuk memandang tugas AI sebagai perancangan agen rasional—yaitu, agen yang tindakannya memaksimalkan utilitas yang diharapkan berdasarkan riwayat persepsi mereka. Kami menunjukkan bahwa masalah perancangan bergantung pada persepsi dan tindakan yang tersedia bagi agen, fungsi utilitas yang harus dipenuhi oleh perilaku agen, dan sifat lingkungan. Berbagai rancangan agen yang berbeda dimungkinkan, mulai dari agen refleks hingga agen yang sepenuhnya deliberatif, berbasis pengetahuan, dan teoretis-keputusan. Lebih lanjut, komponen-komponen rancangan ini dapat memiliki sejumlah perwujudan yang berbeda—misalnya, penalaran logis atau probabilistik, dan representasi keadaan yang atomik, terfaktor, atau terstruktur. Bab-bab selanjutnya menyajikan prinsip-prinsip pengoperasian komponen-komponen ini.

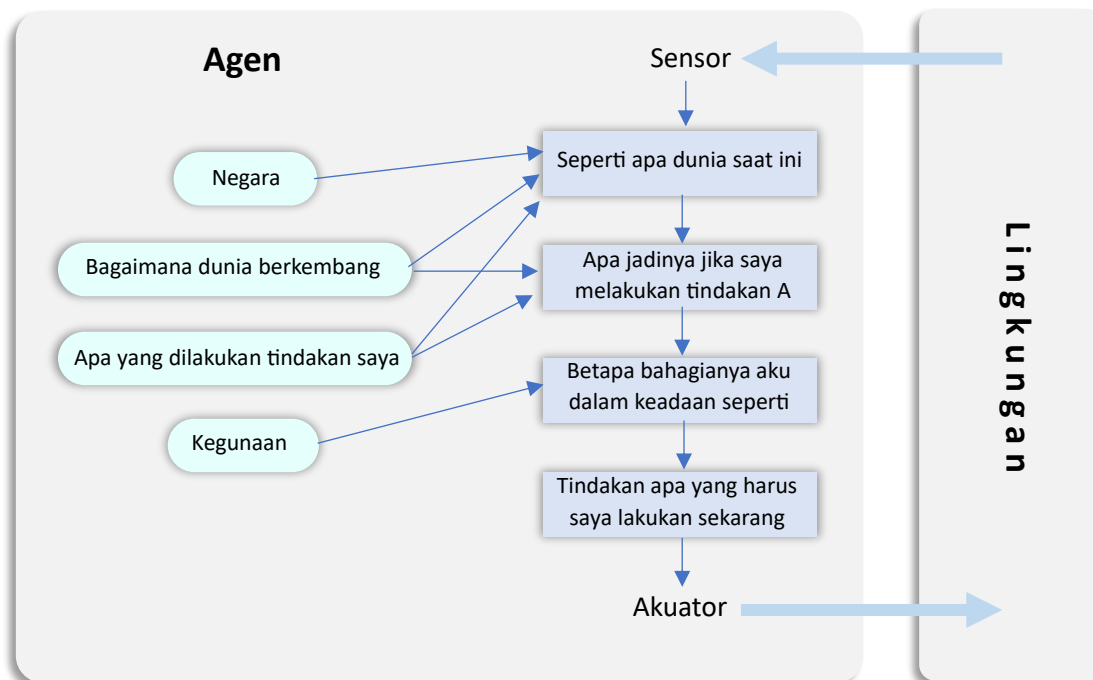
Untuk semua rancangan dan komponen agen, telah terjadi kemajuan yang luar biasa baik dalam pemahaman ilmiah maupun kemampuan teknologi kita. Dalam bab ini, kita akan beranjak dari detail dan bertanya, *"Akankah semua kemajuan ini menghasilkan agen cerdas serbaguna yang dapat berkinerja baik di berbagai lingkungan?"* Bagian 9.1 membahas komponen-komponen agen cerdas untuk menilai apa yang diketahui dan apa yang belum diketahui. Bagian 9.2 melakukan hal yang sama untuk arsitektur agen secara keseluruhan. Bagian 9.3 mempertanyakan apakah merancang agen rasional merupakan tujuan yang tepat sejak awal. (Jawabannya adalah, "Tidak juga, tetapi untuk saat ini tidak masalah.") Terakhir, Bagian 9.4 mengkaji konsekuensi dari keberhasilan upaya kita.

9.1 KOMPONEN AGEN

Bab 2 menyajikan beberapa desain agen dan komponen-komponennya. Untuk memfokuskan pembahasan kita di sini, kita akan melihat agen berbasis utilitas, yang akan kita tunjukkan kembali pada Gambar 9.1. Ketika dilengkapi dengan komponen pembelajaran, inilah desain agen kita yang paling umum. Mari kita lihat bagaimana perkembangan terkini masing-masing komponen.

Interaksi dengan lingkungan melalui sensor dan aktuator: Dalam sebagian besar sejarah AI, hal ini telah menjadi titik lemah yang mencolok. Dengan beberapa pengecualian yang terhormat, sistem AI dibangun sedemikian rupa sehingga manusia harus menyediakan input dan menginterpretasikan output, sementara sistem robotik berfokus pada tugas-tugas tingkat rendah yang sebagian besar tidak melibatkan penalaran dan perencanaan tingkat tinggi. Hal ini sebagian disebabkan oleh biaya besar dan upaya rekayasa yang diperlukan untuk membuat robot sungguhan berfungsi. Situasi ini telah berubah dengan cepat dalam beberapa tahun terakhir dengan tersedianya robot siap pakai yang dapat diprogram. Hal ini, pada gilirannya, telah diuntungkan oleh kamera CCD yang kecil, murah, dan beresolusi tinggi serta penggerak motor yang ringkas dan andal. Teknologi MEMS (sistem mikro-elektromekanis) telah

menyediakan akselerometer, giroskop, dan aktuator miniatur untuk serangga terbang buatan. Menggabungkan jutaan perangkat MEMS untuk menghasilkan aktuator makroskopis yang kuat juga dimungkinkan.



Gambar 9.1 Agen berbasis model dan berbasis utilitas, seperti yang pertama kali disajikan

Dengan demikian, kita melihat bahwa sistem AI berada di titik puncak peralihan dari sistem yang utamanya hanya berbasis perangkat lunak ke sistem robotik tertanam. Keadaan robotika saat ini kira-kira sebanding dengan keadaan komputer pribadi sekitar tahun 1980: pada saat itu para peneliti dan penghobi dapat bereksperimen dengan PC, tetapi butuh satu dekade lagi sebelum menjadi hal yang umum.

Utilitas sebagai ekspresi preferensi: Pada prinsipnya, mendasarkan keputusan rasional pada maksimalisasi utilitas yang diharapkan sepenuhnya bersifat umum dan menghindari banyak masalah pendekatan berbasis tujuan murni, seperti tujuan yang saling bertentangan dan pencapaian yang tidak pasti. Namun, hingga saat ini, hanya ada sedikit penelitian tentang konstruksi fungsi utilitas yang *realistis*—bayangkan, misalnya, jaringan kompleks preferensi yang saling berinteraksi yang harus dipahami oleh seorang agen yang bertindak sebagai asisten kantor bagi seorang manusia. Terbukti sangat sulit untuk menguraikan preferensi atas keadaan kompleks dengan cara yang sama seperti jaring Bayes menguraikan keyakinan atas keadaan kompleks. Salah satu alasannya mungkin karena preferensi atas keadaan sebenarnya *dikompile* dari riwayat preferensi atas keadaan, yang dijelaskan oleh **fungsi imbalan**. Sekalipun fungsi imbalannya sederhana, fungsi utilitas yang sesuai mungkin sangat kompleks. Hal ini menunjukkan bahwa kita menganggap serius tugas rekayasa pengetahuan untuk fungsi imbalan sebagai cara untuk menyampaikan kepada agen kita apa yang kita inginkan dari mereka.

Pembelajaran: Pembelajaran dalam agen dapat diformulasikan sebagai pembelajaran induktif (terawasi, tanpa pengawasan, atau berbasis penguatan) dari fungsi-fungsi yang menyusun berbagai komponen agen. Teknik logika dan statistik yang sangat canggih telah dikembangkan yang mampu mengatasi masalah yang cukup besar, mencapai atau melampaui kemampuan manusia dalam banyak tugas—selama kita berurusan dengan kosakata fitur dan konsep yang telah ditentukan sebelumnya. Di sisi lain, pembelajaran mesin hanya membuat sedikit kemajuan dalam masalah penting membangun representasi baru pada tingkat abstraksi yang lebih tinggi daripada kosakata masukan. Dalam visi komputer, misalnya, mempelajari konsep kompleks seperti *Ruang Kelas* dan *Kafetaria* akan menjadi sangat sulit jika agen dipaksa bekerja dari piksel sebagai representasi masukan; sebaliknya, agen harus mampu membentuk konsep perantara terlebih dahulu, seperti *Meja* dan *Baki*, tanpa pengawasan manusia yang eksplisit. Pertimbangan serupa berlaku untuk perilaku pembelajaran: *HavingACupOfTea* merupakan langkah tingkat tinggi yang sangat penting dalam banyak rencana, tetapi bagaimana cara memasukkannya ke dalam pustaka tindakan yang awalnya berisi tindakan yang jauh lebih sederhana seperti *RaiseArm* dan *Swallow*? Mungkin ini akan menggabungkan beberapa gagasan tentang **jaringan keyakinan mendalam**—jaringan Bayesian yang memiliki beberapa lapisan variabel tersembunyi.

Sebagian besar penelitian pembelajaran mesin saat ini mengasumsikan representasi terfaktor, mempelajari fungsi $h : \mathbb{R}^n \rightarrow \mathbb{R}$ untuk regresi dan $h : \mathbb{R}^n \rightarrow \{0, 1\}$ untuk klasifikasi. Peneliti pembelajaran perlu mengadaptasi teknik representasi terfaktor yang sangat berhasil ke representasi terstruktur, khususnya representasi hierarkis. Penelitian tentang pemrograman logika induktif di Bab 1 merupakan langkah awal ke arah ini; langkah logis berikutnya adalah menggabungkan ide-ide ini dengan bahasa probabilistik.

Jika kita tidak memahami isu-isu tersebut, kita dihadapkan pada tugas berat untuk membangun basis pengetahuan akal sehat yang besar secara manual, sebuah pendekatan yang belum berjalan dengan baik hingga saat ini. Ada potensi besar dalam penggunaan web sebagai sumber teks, gambar, dan video bahasa alami untuk berfungsi sebagai basis pengetahuan yang komprehensif, tetapi sejauh ini algoritma pembelajaran mesin terbatas dalam jumlah pengetahuan terorganisir yang dapat diekstraksi dari sumber-sumber ini.

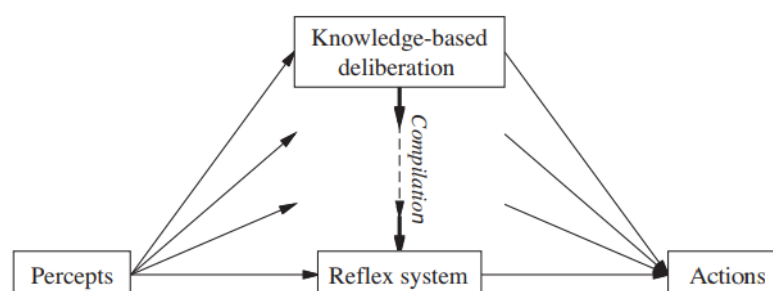
9.2 ARSITEKTUR AGEN

Wajar untuk bertanya, "Arsitektur agen mana di Bab 2 yang harus digunakan agen?" Jawabannya adalah, "Semuanya!" Kita telah melihat bahwa respons refleks diperlukan untuk situasi di mana waktu sangat penting, sedangkan pertimbangan berbasis pengetahuan memungkinkan agen untuk merencanakan ke depan. Agen yang lengkap harus dapat melakukan keduanya, menggunakan **arsitektur hibrida**. Salah satu properti penting dari arsitektur hibrida adalah bahwa batas-batas antara komponen keputusan yang berbeda tidak tetap. Misalnya, **kompilasi** terus-menerus mengubah informasi deklaratif pada tingkat pertimbangan menjadi representasi yang lebih efisien, yang akhirnya mencapai tingkat refleks—lihat Gambar 9.2. Arsitektur agen seperti SOAR dan THEO memiliki struktur persis seperti ini. Setiap kali mereka memecahkan masalah dengan pertimbangan eksplisit, mereka

menyimpan versi umum dari solusi untuk digunakan oleh komponen refleksi. Masalah yang kurang dipelajari adalah *pembalikan* proses ini: ketika lingkungan berubah, refleksi yang dipelajari mungkin tidak lagi sesuai dan agen harus kembali ke tingkat deliberatif untuk menghasilkan perilaku baru.

Agen juga membutuhkan cara untuk mengendalikan deliberatif mereka sendiri. Mereka harus dapat berhenti berunding ketika tindakan dituntut, dan mereka harus dapat menggunakan waktu yang tersedia untuk deliberatif guna menjalankan komputasi yang paling menguntungkan. Misalnya, agen pengemudi taksi yang melihat kecelakaan di depan harus memutuskan dalam sepersekian detik untuk mengerem atau mengambil tindakan mengelak. Agen juga harus menghabiskan sepersekian detik itu untuk memikirkan pertanyaan-pertanyaan terpenting, seperti apakah lajur kiri dan kanan kosong dan apakah ada truk besar di belakang, alih-alih mengkhawatirkan keausan ban atau di mana harus menjemput penumpang berikutnya. Isu-isu ini biasanya dipelajari di bawah judul **AI waktu nyata**. Seiring sistem AI bergerak ke domain yang lebih kompleks, semua masalah akan menjadi waktu nyata, karena agen tidak akan pernah punya cukup waktu untuk menyelesaikan masalah keputusan secara tepat.

Jelas, terdapat kebutuhan mendesak akan metode *umum* untuk mengendalikan pertimbangan, alih-alih resep khusus tentang apa yang harus dipikirkan dalam setiap situasi. Ide pertama yang bermanfaat adalah menggunakan **algoritma kapan saja**. Algoritma kapan saja adalah algoritma yang kualitas keluarannya meningkat secara bertahap seiring waktu, sehingga memiliki keputusan yang masuk akal dan siap kapan pun terganggu. Algoritma semacam itu dikendalikan oleh prosedur pengambilan keputusan **metalevel** yang menilai apakah komputasi lebih lanjut bermanfaat. Contoh algoritma kapan saja mencakup pendalaman iteratif dalam pencarian pohon permainan dan MCMC dalam jaringan Bayesian.



Gambar 9.2 Kompilasi berfungsi untuk mengubah pengambilan keputusan yang bersifat deliberatif menjadi mekanisme refleksif yang lebih efisien.

Teknik kedua untuk mengendalikan pertimbangan adalah **metareasoning teoretis-keputusan**. Metode ini menerapkan teori nilai informasi pada pemilihan komputasi individual. Nilai komputasi bergantung pada biayanya (dalam hal menunda tindakan) dan manfaatnya (dalam hal peningkatan kualitas keputusan). Teknik metareasoning dapat digunakan untuk merancang algoritma pencarian yang lebih baik dan menjamin bahwa algoritma tersebut memiliki properti kapan saja. Metareasoning tentu saja mahal, dan metode kompilasi dapat

diterapkan sehingga overhead-nya kecil dibandingkan dengan biaya komputasi yang dikontrol. Pembelajaran penguatan metalevel dapat menyediakan cara lain untuk memperoleh kebijakan yang efektif dalam mengendalikan pertimbangan: intinya, komputasi yang menghasilkan keputusan yang lebih baik diperkuat, sementara komputasi yang ternyata tidak berpengaruh dikenai penalti. Pendekatan ini menghindari masalah miopia dari perhitungan nilai informasi yang sederhana.

Metareasoning adalah salah satu contoh spesifik dari **arsitektur reflektif**—yaitu, arsitektur yang memungkinkan pertimbangan tentang entitas komputasi dan tindakan yang terjadi dalam arsitektur itu sendiri. Landasan teoretis untuk arsitektur reflektif dapat dibangun dengan mendefinisikan ruang keadaan gabungan yang terdiri dari keadaan lingkungan dan keadaan komputasi agen itu sendiri. Algoritma pengambilan keputusan dan pembelajaran dapat dirancang untuk beroperasi pada ruang keadaan gabungan ini dan dengan demikian berfungsi untuk mengimplementasikan dan meningkatkan aktivitas komputasi agen. Pada akhirnya, kami berharap algoritma khusus tugas seperti pencarian alfa-beta dan backward chaining akan menghilang dari sistem AI, digantikan oleh metode umum yang mengarahkan komputasi agen menuju pembangkitan keputusan berkualitas tinggi yang efisien.

9.3 APAKAH KITA MENUJU ARAH YANG BENAR?

Bagian sebelumnya telah mencantumkan banyak kemajuan dan banyak peluang untuk kemajuan lebih lanjut. Namun, ke mana semua ini mengarah? Dreyfus (1992) memberikan analogi mencoba mencapai bulan dengan memanjat pohon; seseorang dapat melaporkan kemajuan yang stabil, hingga ke puncak pohon. Di bagian ini, kami mempertimbangkan apakah jalur AI saat ini lebih seperti memanjat pohon atau perjalanan roket.

Di Bab 1, kami mengatakan bahwa tujuan kami adalah membangun agen yang bertindak rasional. Namun, kami juga mengatakan bahwa

... mencapai rasionalitas sempurna—selalu melakukan hal yang benar—tidaklah mungkin dilakukan dalam lingkungan yang rumit. Tuntutan komputasinya terlalu tinggi. Namun, untuk sebagian besar buku ini, kami akan mengadopsi hipotesis kerja bahwa rasionalitas sempurna adalah titik awal yang baik untuk analisis.

Sekarang saatnya untuk mempertimbangkan kembali apa sebenarnya tujuan AI. Kami ingin membangun agen, tetapi dengan spesifikasi apa? Berikut adalah empat kemungkinan:

Rasionalitas sempurna. Agen yang rasional sempurna bertindak setiap saat sedemikian rupa untuk memaksimalkan utilitas yang diharapkan, berdasarkan informasi yang diperolehnya dari lingkungan. Kita telah melihat bahwa kalkulasi yang diperlukan untuk mencapai rasionalitas sempurna di sebagian besar lingkungan terlalu memakan waktu, sehingga rasionalitas sempurna bukanlah tujuan yang realistis.

Rasionalitas kalkulatif. Ini adalah gagasan rasionalitas yang telah kita gunakan secara implisit dalam merancang agen logis dan teoretis-keputusan, dan sebagian besar penelitian AI teoretis berfokus pada properti ini. Agen yang rasional secara kalkulatif *pada akhirnya* mengembalikan apa yang seharusnya menjadi pilihan rasional di awal pertimbangannya. Ini adalah properti yang menarik untuk ditunjukkan oleh suatu sistem, tetapi di sebagian besar lingkungan,

jawaban yang tepat pada waktu yang salah tidak bernilai. Dalam praktiknya, perancang sistem AI terpaksa berkompromi pada kualitas keputusan untuk mendapatkan kinerja keseluruhan yang wajar; sayangnya, dasar teoretis rasionalitas kalkulatif tidak menyediakan cara yang beralasan untuk membuat kompromi semacam itu.

Rasionalitas terbatas. Herbert Simon (1957) menolak gagasan rasionalitas sempurna (atau bahkan mendekati sempurna) dan menggantinya dengan rasionalitas terbatas, sebuah teori deskriptif tentang pengambilan keputusan oleh agen nyata. Ia menulis,

Kapasitas pikiran manusia untuk merumuskan dan memecahkan masalah kompleks sangat kecil dibandingkan dengan ukuran masalah yang solusinya diperlukan untuk perilaku rasional objektif di dunia nyata—atau bahkan untuk pendekatan yang wajar terhadap rasionalitas objektif tersebut.

Ia berpendapat bahwa rasionalitas terbatas bekerja terutama dengan *memuaskan*—yaitu, berunding cukup lama untuk menghasilkan jawaban yang "cukup baik." Simon memenangkan Hadiah Nobel di bidang ekonomi untuk karya ini dan telah menulis tentangnya secara mendalam (Simon, 1982). Hal ini tampaknya menjadi model perilaku manusia yang berguna dalam banyak kasus. Namun, ini bukan spesifikasi formal untuk agen cerdas, karena definisi "cukup baik" tidak diberikan oleh teori tersebut. Lebih lanjut, memuaskan tampaknya hanyalah salah satu dari berbagai metode yang digunakan untuk mengatasi sumber daya yang terbatas.

Optimalitas terbatas (BO). Agen optimal terbatas berperilaku sebaik mungkin, *mengingat sumber daya komputasinya*. Artinya, utilitas yang diharapkan dari program agen untuk agen optimal terbatas setidaknya sama tingginya dengan utilitas yang diharapkan dari program agen lain yang berjalan pada mesin yang sama.

Dari keempat kemungkinan ini, optimalitas terbatas tampaknya menawarkan harapan terbaik untuk fondasi teoretis yang kuat bagi AI. Keuntungannya adalah dapat dicapai: selalu ada setidaknya satu program terbaik—sesuatu yang tidak dimiliki oleh rasionalitas sempurna. Agen optimal terbatas sebenarnya berguna di dunia nyata, sedangkan agen rasional kalkulatif biasanya tidak, dan agen yang memuaskan mungkin berguna atau tidak, tergantung pada seberapa ambisius mereka.

Pendekatan tradisional dalam AI adalah memulai dengan rasionalitas kalkulatif dan kemudian membuat kompromi untuk memenuhi kendala sumber daya. Jika masalah yang ditimbulkan oleh kendala tersebut kecil, desain akhir diharapkan serupa dengan desain agen BO. Namun, seiring kendala sumber daya menjadi lebih kritis—misalnya, seiring lingkungan menjadi lebih kompleks—kedua desain tersebut diperkirakan akan berbeda. Dalam teori optimalitas terbatas, kendala-kendala ini dapat ditangani secara berprinsip.

Hingga saat ini, hanya sedikit yang diketahui tentang optimalitas terbatas. Dimungkinkan untuk membangun program optimal terbatas untuk mesin yang sangat sederhana dan untuk jenis lingkungan yang agak terbatas, tetapi hingga saat ini kita belum mengetahui seperti apa program BO untuk komputer serbaguna yang besar dalam lingkungan yang kompleks. Jika ingin ada teori konstruktif tentang optimalitas terbatas, kita harus berharap bahwa desain program optimal terbatas tidak terlalu bergantung pada detail komputer yang digunakan.

Akan sangat sulit bagi penelitian ilmiah jika menambahkan beberapa kilobyte memori ke mesin gigabyte membuat perbedaan yang signifikan pada desain program BO. Salah satu cara untuk memastikan hal ini tidak terjadi adalah dengan sedikit lebih longgar tentang kriteria optimalitas terbatas. Dengan analogi dengan gagasan kompleksitas asimtotik (Lampiran A), kita dapat mendefinisikan **optimalitas terbatas asimtotik** (ABO) sebagai berikut (Russell dan Subramanian, 1995). Misalkan sebuah program P adalah optimal terbatas untuk mesin M dalam kelas lingkungan E , di mana kompleksitas lingkungan di E tidak terbatas. Maka program P' adalah ABO untuk M di E jika dapat mengungguli P dengan berjalan pada mesin kM yang k kali lebih cepat (atau lebih besar) daripada M . Kecuali k sangat besar, kita akan senang dengan program yang merupakan ABO untuk lingkungan nontrivial pada arsitektur nontrivial. Akan ada sedikit gunanya melakukan upaya besar untuk menemukan program BO daripada program ABO, karena ukuran dan kecepatan mesin yang tersedia cenderung meningkat dengan faktor konstan dalam jumlah waktu yang tetap.

Kita dapat menebak bahwa program BO atau ABO untuk komputer canggih di lingkungan yang kompleks belum tentu memiliki struktur yang sederhana dan elegan. Kita telah melihat bahwa kecerdasan tujuan umum membutuhkan beberapa kemampuan refleksi dan beberapa kemampuan deliberatif; berbagai bentuk pengetahuan dan pengambilan keputusan; mekanisme pembelajaran dan kompilasi untuk semua bentuk tersebut; metode untuk mengendalikan penalaran; dan sejumlah besar pengetahuan spesifik domain. Agen optimal terbatas harus beradaptasi dengan lingkungan tempatnya berada, sehingga pada akhirnya organisasi internalnya akan mencerminkan optimasi yang spesifik untuk lingkungan tersebut. Hal ini wajar saja, dan serupa dengan bagaimana mobil balap yang dibatasi oleh kapasitas mesin telah berevolusi menjadi desain yang sangat kompleks. Kami menduga bahwa ilmu kecerdasan buatan yang berbasis pada optimalitas terbatas akan melibatkan banyak studi tentang proses yang memungkinkan program agen untuk konvergen ke optimalitas terbatas dan mungkin lebih sedikit konsentrasi pada detail program rumit yang dihasilkan.

Singkatnya, konsep optimalitas terbatas diusulkan sebagai tugas formal untuk penelitian AI yang terdefinisi dengan baik dan layak. Optimalitas terbatas menentukan *program* optimal, alih-alih *tindakan* optimal. Tindakan, bagaimanapun juga, dihasilkan oleh program, dan atas programlah para perancang memiliki kendali.

27.4 BAGAIMANA JIKA AI BERHASIL?

Dalam *Small World* (1984) karya David Lodge, sebuah novel tentang dunia akademis kritik sastra, sang protagonis menimbulkan keresahan dengan mengajukan pertanyaan berikut kepada panel ahli teori sastra terkemuka namun kontradiktif: "*What if you were right?*" Tak satu pun dari para ahli teori tersebut tampaknya pernah mempertimbangkan pertanyaan ini sebelumnya, mungkin karena memperdebatkan teori yang tidak dapat dipalsukan adalah tujuan itu sendiri. Kebingungan serupa dapat ditimbulkan dengan bertanya kepada para peneliti AI, "*What if you succeed?*"

Ada isu-isu etika yang perlu dipertimbangkan. Komputer cerdas lebih canggih daripada komputer bodoh, tetapi apakah kekuatan itu akan digunakan untuk kebaikan atau keburukan?

Mereka yang berupaya mengembangkan AI memiliki tanggung jawab untuk memastikan bahwa dampak dari karya mereka positif. Cakupan dampak akan bergantung pada tingkat keberhasilan AI. Bahkan keberhasilan sederhana dalam AI telah mengubah cara ilmu komputer diajarkan dan praktik pengembangan perangkat lunak. AI telah memungkinkan aplikasi baru seperti sistem pengenalan suara, sistem kendali inventaris, sistem pengawasan, robot, dan mesin pencari.

Kita dapat memperkirakan bahwa keberhasilan AI tingkat menengah akan memengaruhi semua jenis orang dalam kehidupan sehari-hari mereka. Sejauh ini, jaringan komunikasi terkomputerisasi, seperti ponsel dan internet, telah memiliki efek yang begitu luas pada masyarakat, tetapi AI belum. AI telah bekerja di balik layar—misalnya, dalam menyetujui atau menolak transaksi kartu kredit secara otomatis untuk setiap pembelian yang dilakukan di web—tetapi belum terlihat oleh konsumen rata-rata. Kita dapat membayangkan bahwa asisten pribadi yang benar-benar berguna untuk kantor atau rumah akan memiliki dampak positif yang besar pada kehidupan masyarakat, meskipun mungkin menyebabkan beberapa dislokasi ekonomi dalam jangka pendek. Asisten otomatis untuk mengemudi dapat mencegah kecelakaan, menyelamatkan puluhan ribu nyawa per tahun. Kemampuan teknologi pada tingkat ini juga dapat diterapkan pada pengembangan senjata otonom, yang banyak dianggap tidak diinginkan. Beberapa masalah sosial terbesar yang kita hadapi saat ini—seperti pemanfaatan informasi genomik untuk mengobati penyakit, pengelolaan sumber daya energi yang efisien, dan verifikasi perjanjian terkait senjata nuklir—sedang diatasi dengan bantuan teknologi AI.

Terakhir, tampaknya keberhasilan AI dalam skala besar—penciptaan kecerdasan setingkat manusia dan lebih tinggi lagi—akan mengubah kehidupan mayoritas umat manusia. Hakikat pekerjaan dan hiburan kita akan berubah, begitu pula pandangan kita tentang kecerdasan, kesadaran, dan nasib masa depan umat manusia. Sistem AI pada tingkat kemampuan ini dapat mengancam otonomi, kebebasan, dan bahkan kelangsungan hidup manusia. Karena alasan-alasan ini, kita tidak dapat memisahkan penelitian AI dari konsekuensi etisnya.

Ke mana arah masa depan? Para penulis fiksi ilmiah tampaknya lebih menyukai masa depan distopia daripada utopia, mungkin karena hal itu menghasilkan plot yang lebih menarik. Namun sejauh ini, AI tampaknya cocok dengan teknologi revolusioner lainnya (percetakan, perpipaan, perjalanan udara, telepon) yang dampak negatifnya lebih kecil daripada aspek positifnya.

Sebagai kesimpulan, kita melihat bahwa AI telah membuat kemajuan pesat dalam sejarahnya yang singkat, tetapi kalimat terakhir dari esai Alan Turing (1950) tentang *Mesin Komputasi dan Kecerdasan* masih relevan hingga saat ini:

Kita hanya dapat melihat sedikit ke depan, tetapi kita dapat melihat bahwa masih banyak yang harus dilakukan.

DAFTAR PUSTAKA

- Azar, K., Hajiakhondi-Meybodi, Z., & Naderkhani, F. (2022). Semi-supervised clustering-based method for fault diagnosis and prognosis: A case study. *Reliability Engineering & System Safety*, 222, 108405.
- Bile, A., Moratti, F., Tari, H., & Fazio, E. (2021). Supervised and unsupervised learning using a fully-plastic all-optical unit of artificial intelligence based on solitonic waveguides. *Neural Computing and Applications*, 33(24), 17071-17079.
- Burkart, N., & Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70, 245-317.
- Davitaia, A. (2025). Artificial Intelligence and machine learning in fraud detection for digital payments. *International Journal of Science and Research Archive*, 15(3), 714-719.
- El Mrabet, M. A., El Makkaoui, K., & Faize, A. (2021, December). Supervised machine learning: A survey. In 2021 4th International conference on advanced communication technologies and networking (CommNet) (pp. 1-10). IEEE.
- Gleirscher, M. (2023). Supervision of intelligent systems: An overview. *Applicable Formal Methods for Safe Industrial Products: Essays Dedicated to Jan Peleska on the Occasion of His 65th Birthday*, 202-221.
- Gupta, R., Srivastava, D., Sahu, M., Tiwari, S., Ambasta, R. K., & Kumar, P. (2021). Artificial intelligence to deep learning: machine intelligence approach for drug discovery. *Molecular diversity*, 25(3), 1315-1360.
- Gutiérrez, L., Patiño, J., & Duque-Grisales, E. (2021). A comparison of the performance of supervised learning algorithms for solar power prediction. *Energies*, 14(15), 4424.
- Ji, J., Wang, J., Huang, C., Wu, J., Xu, B., Wu, Z., ... & Zheng, Y. (2023, June). Spatio-temporal self-supervised learning for traffic flow prediction. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 37, No. 4, pp. 4356-4364).
- Khan, I., Ahmad, A. R., Jabeur, N., & Mahdi, M. N. (2021). An artificial intelligence approach to monitor student performance and devise preventive measures. *Smart Learning Environments*, 8(1), 17.
- Kong, S. C., Cheung, W. M. Y., & Zhang, G. (2021). Evaluation of an artificial intelligence literacy course for university students with diverse study backgrounds. *Computers and Education: Artificial Intelligence*, 2, 100026.
- Krishnan, R., Rajpurkar, P., & Topol, E. J. (2022). Self-supervised learning in medicine and healthcare. *Nature Biomedical Engineering*, 6(12), 1346-1352.
- Kwekha-Rashid, A. S., Abduljabbar, H. N., & Alhayani, B. (2023). Coronavirus disease (COVID-19) cases analysis using machine-learning applications. *Applied nanoscience*, 13(3), 2013-2025.
- Laleh, N. G., Muti, H. S., Loeffler, C. M. L., Echle, A., Saldanha, O. L., Mahmood, F., ... & Kather, J. N. (2022). Benchmarking weakly-supervised deep learning pipelines for whole slide classification in computational pathology. *Medical image analysis*, 79, 102474.

- Li, P., Wang, J., Qiao, Y., Chen, H., Yu, Y., Yao, X., ... & Song, S. (2021). An effective self-supervised framework for learning expressive molecular global representations to drug discovery. *Briefings in Bioinformatics*, 22(6), bbab109.
- Lu, W., Vivekananda, G. N., & Shanthini, A. (2023). Supervision system of English online teaching based on machine learning. *Progress in artificial intelligence*, 12(2), 187-198.
- Madhuri, K. (2023). Security Threats and Detection Mechanisms in Machine Learning. *Handbook of Artificial Intelligence*, 255.
- Manne, R., & Kantheti, S. C. (2021). Application of artificial intelligence in healthcare: chances and challenges. *Current Journal of Applied Science and Technology*, 40(6), 78-89.
- Mestikou, M., Smeti, K. E., & Hachaïchi, Y. (2023). Artificial intelligence and machine learning in financial services Market developments and financial stability implications. *Financial Stability Board*, 1, 1-6.
- Miric, M., Jia, N., & Huang, K. G. (2023). Using supervised machine learning for large-scale classification in management research: The case for identifying artificial intelligence patents. *Strategic Management Journal*, 44(2), 491-519.
- Misera, L., Müller-Franzes, G., Truhn, D., & Kather, J. N. (2024). Weakly supervised deep learning in radiology. *Radiology*, 312(1), e232085.
- Moor, M., Banerjee, O., Abad, Z. S. H., Krumholz, H. M., Leskovec, J., Topol, E. J., & Rajpurkar, P. (2023). Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956), 259-265.
- Muhammad, L. J., Algehyne, E. A., Usman, S. S., Ahmad, A., Chakraborty, C., & Mohammed, I. A. (2021). Supervised machine learning models for prediction of COVID-19 infection using epidemiology dataset. *SN computer science*, 2(1), 11.
- Nahavandi, D., Alizadehsani, R., Khosravi, A., & Acharya, U. R. (2022). Application of artificial intelligence in wearable devices: Opportunities and challenges. *Computer Methods and Programs in Biomedicine*, 213, 106541.
- Navarro, C. L. A., Damen, J. A., Takada, T., Nijman, S. W., Dhiman, P., Ma, J., ... & Hooft, L. (2021). Risk of bias in studies on prediction models developed using supervised machine learning techniques: systematic review. *bmj*, 375.
- Noain Sánchez, A. (2022). Addressing the Impact of Artificial Intelligence on Journalism: The perception of experts, journalists and academics.
- Pandithurai, O., Bharathiraja, N., Pradeepa, K., Meenakshi, D., Kathiravan, M., & Vinoth Kumar, M. (2023, February). Air pollution prediction using supervised machine learning technique. In *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)* (pp. 542-546). IEEE.
- Plathottam, S. J., Rzonca, A., Lakhnori, R., & Iloeje, C. O. (2023). A review of artificial intelligence applications in manufacturing operations. *Journal of Advanced Manufacturing and Processing*, 5(3), e10159.
- Rani, V., Nabi, S. T., Kumar, M., Mittal, A., & Kumar, K. (2023). Self-supervised learning: A succinct review. *Archives of Computational Methods in Engineering*, 30(4), 2761-2775.

- Sermesant, M., Delingette, H., Cochet, H., Jaïs, P., & Ayache, N. (2021). Applications of artificial intelligence in cardiovascular imaging. *Nature Reviews Cardiology*, 18(8), 600-609.
- Song, A. H., Jaume, G., Williamson, D. F., Lu, M. Y., Vaidya, A., Miller, T. R., & Mahmood, F. (2023). Artificial intelligence for digital and computational pathology. *Nature Reviews Bioengineering*, 1(12), 930-949.
- Sun, R., & Rostami, M. (2023). Explainable artificial intelligence architecture for melanoma diagnosis using indicator localization and self-supervised learning. *arXiv preprint arXiv:2303.14615*.
- Wagner, M. W., Namdar, K., Biswas, A., Monah, S., Khalvati, F., & Ertl-Wagner, B. B. (2021). Radiomics, machine learning, and artificial intelligence—what the neuroradiologist needs to know. *Neuroradiology*, 63(12), 1957-1967.
- Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., ... & Zitnik, M. (2023). Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972), 47-60.
- Wang, Y. (2022). Safety production supervision of industrial enterprises based on deep learning and artificial intelligence. *Mobile Information Systems*, 2022(1), 1820082.
- Xiong, Q., Yuan, C., He, B., Xiong, H., & Kong, Q. (2023). GTRF: A general deep learning framework for tuples recognition towards supervised, semi-supervised and unsupervised paradigms. *Engineering Applications of Artificial Intelligence*, 124, 106500.
- Yonezawa¹, T., & Yamada, Y. (2025, April). Supervision with AI Analysis. In *Innovating Education with AI: Selected Proceedings of 2024 4th Asia Education Technology Symposium* (p. 75). Springer Nature.
- Zhang, C., & Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, 23, 100224.

Supervisi Pembelajaran Kecerdasan buatan (AI)

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS



Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Sejak tahun 2023 penulis tercatat sebagai Dosen luar biasa di Fakultas Ekonomi & Bisnis (FEB) Universitas Diponegoro Semarang. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

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

ISBN 978-634-7227-30-0 (PDF)



9

786347

227300