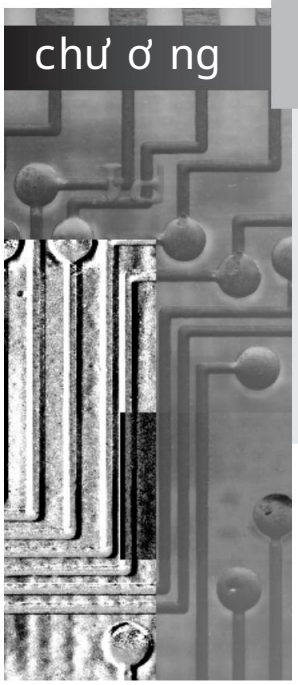
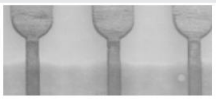
chư ơ ng 4



thuật toán:

Các phư ơ ng pháp cơ bản



Bây giờ chúng ta đã biết cách biểu diễn đầu vào và đầu ra, đã đến lúc xem xét bản thân các thuật toán học. Chư ơ ng này giải thích những ý tư ởng cơ bản đằng sau các kỹ thuật đư ợc sử dụng trong khai thác dữ liệu thực tế. Chúng tôi sẽ không đi sâu vào các vấn đề phức tạp hơ n—các phiên bản nâng cao của thuật toán, khả năng tối ư u hóa, các vấn đề phức tạp phát sinh trong thực tế. Các chủ đề này đư ợc chuyển sang Chư ơ ng 6, nơ i chúng ta nắm bắt đư ợc các triển khai thực tế của các phư ơ ng pháp học máy, chẳng hạn như các phư ơ ng pháp có trong bộ công cụ khai thác dữ liệu và đư ợc sử dụng cho các ứng dụng trong thế giới thực. Điều quan trọng là phải hiểu những vấn đề nâng cao hơ n này để bạn biết điều gì đang thực sự xảy ra khi bạn phân tích một tập dữ liệu cụ thể.

Trong chư ơ ng này chúng ta xem xét các ý tư ởng cơ bản. Một trong những bài học mang tính hư ớng dẫn nhất là những ý tư ởng đơ n giản thư ờng hoạt động rất tốt và chúng tôi thực sự khuyên bạn nên áp dụng phư ơ ng pháp "ư u tiên sự đơ n giản" khi phân tích các bộ dữ liệu thực tế. Có nhiều loại cấu trúc đơ n giản khác nhau mà bộ dữ liệu có thể thể hiện. Trong một tập dữ liệu, có thể có một thuộc tính thực hiện tất cả công việc và những thuộc tính khác có thể không liên quan hoặc dư thừa. Trong tập dữ liệu khác, các thuộc tính có thể

đóng góp một cách độc lập và bình đẳng vào kết quả cuối cùng. Một phần ba có thể có cấu trúc logic đơn giản, chỉ liên quan đến một vài thuộc tính có thể được cây quyết định nắm bắt. Trong trường hợp thứ tư , có thể có một số quy tắc độc lập chi phối việc gán các thể hiện cho các lớp khác nhau. Phần thứ năm có thể thể hiện sự phụ thuộc giữa các tập hợp con khác nhau của các thuộc tính. Thứ sáu có thể liên quan đến sự phụ thuộc tuyến tính giữa các thuộc tính số, trong đó điều quan trọng là tổng trọng số của các giá trị thuộc tính với các trọng số được chọn phù hợp. Trong trường hợp thứ bảy, các phân loại phù hợp với các vùng cụ thể của không gian thể hiện có thể bị chi phối bởi khoảng cách giữa chính các thể hiện. Và trong phần tám, có thể không có giá trị lớp nào được cung cấp: việc học không được giám sát.

Trong vô số các bộ dữ liệu có thể có, có nhiều loại cấu trúc khác nhau có thể xuất hiện và một công cụ khai thác dữ liệu—bất kể khả năng như thế nào—đang tìm kiếm một loại cấu trúc có thể hoàn toàn bỏ sót các quy tắc của một loại khác, bất kể cách thức thô sơ đó có thể được. Kết quả là một cấu trúc phân loại baroque và mờ đục của một loại thay vì một cấu trúc đơn giản, tao nhã, dễ hiểu ngay lập tức của một loại khác.

Mỗi ví dụ trong số tám ví dụ về các loại tập dữ liệu khác nhau được phác thảo trước đây đều dẫn đến một phương pháp học máy khác rất phù hợp để khám phá ra nó. Các phần của chương này lần lượt xem xét từng cấu trúc này.

4.1 Suy ra các quy tắc sơ cấp

Đây là một cách dễ dàng để tìm các quy tắc phân loại rất đơn giản từ một tập hợp các trường hợp. Được gọi là 1R cho 1 quy tắc, nó tạo ra cây quyết định một cấp được biểu thị dưới dạng một bộ quy tắc kiểm tra tất cả một thuộc tính cụ thể. 1R là một phương pháp đơn giản, rẻ tiền, thường đưa ra các quy tắc khá tốt để mô tả cấu trúc trong dữ liệu. Hóa ra những quy tắc đơn giản thường đạt được độ chính xác cao đến kinh ngạc. Có lẽ điều này là do cấu trúc bên dưới nhiều bộ dữ liệu trong thế giới thực khá thô sơ và chỉ cần một thuộc tính là đủ để xác định lớp của một thể hiện khá chính xác. Trong mọi trường hợp, luôn luôn là một kế hoạch tốt để thử những điều đơn giản nhất trước tiên.

Ý tưởng là thế này: chúng tôi tạo ra các quy tắc kiểm tra một thuộc tính đơn lẻ và phân nhánh phù hợp với nhau. Mỗi nhánh tương ứng với một giá trị khác nhau của thuộc tính. Rõ ràng đâu là cách phân loại tốt nhất để cung cấp cho mỗi nhánh: sử dụng lớp xuất hiện thường xuyên nhất trong dữ liệu huấn luyện. Sau đó, tỷ lệ lỗi của các quy tắc có thể dễ dàng được xác định. Chỉ cần đếm các lỗi xảy ra trên dữ liệu huấn luyện, nghĩa là số trường hợp không có lớp đa số.

Mỗi thuộc tính tạo ra một bộ quy tắc khác nhau, một quy tắc cho mọi giá trị của thuộc tính. Đánh giá tỷ lệ lỗi đối với bộ quy tắc của từng thuộc tính và chọn bộ quy tắc tốt nhất. Nó đơn giản mà! Hình 4.1 thể hiện thuật toán dưới dạng mã giả.

```
Đối với mỗi thuộc tính,
   Đối với mỗi giá trị của thuộc tính đó, hãy tạo một quy tắc như sau:
      đếm tần suất mỗi lớp xuất hiện
      tìm lớp thường xuyên nhất
      làm cho quy tắc gán lớp đó cho thuộc tính-giá trị này.
   Tính tỷ lệ lỗi của các quy tắc.
Chọn các quy tắc có tỷ lệ lỗi nhỏ nhất.
```

Hình 4.1 Mã giả cho 1R.

Bảng 4.1 Đánh giá các thuộc tính trong dữ liệu thời tiết.

	Thuộc tính	Quy tắc	lỗi	Lỗi hoàn toàn
1	quan điểm	nắng → không	2/5	14/4
		u ám → có mưa →	0/4	
		có nóng →	2/5	
2	nhiệt độ	không* nhẹ	2/4	14/5
		→ có mát →	2/6	
		có cao →	1/4	
3	độ ẩm	không bình	3/7	14/4
		thường → có sai	1/7	
4	co gio	→ có đúng →	2/8	14/5
		không*	3/6	

* Một sự lựa chọn ngẫu nhiên đã được thực hiện giữa hai kết quả có khả năng như nhau.

Để thấy được hiệu quả của phương pháp 1R, hãy xem xét dữ liệu thời tiết trong Bảng 1.2 (chúng tôi sẽ gặp lại nó nhiều lần khi nhìn vào cách thức hoạt động của các thuật toán học tập). Để phân loại ở cột cuối cùng, hãy chơi, 1R xem xét bốn bộ quy tắc, mỗi bộ quy tắc một thuộc tính. Các quy tắc này được thể hiện trong Bảng 4.1. Dấu hoa thị chỉ ra rằng một ngẫu nhiên lựa chọn đã được thực hiện giữa hai kết quả có khả năng như nhau. Số lượng lỗi được đưa ra cho mỗi quy tắc, cùng với tổng số lỗi cho quy tắc thiết lập như một tổng thể. 1R chọn thuộc tính tạo ra các luật có giá trị nhỏ nhất số lỗi—nghĩa là, bộ quy tắc thứ nhất và thứ ba. Tự ý phá vỡ ràng buộc giữa hai bộ quy tắc này mang lại:

```
triển vọng: nắng → không
            u ám → vâng
            mưa → vâng
```

Chúng tôi đã lưu ý ngay từ đầu rằng trò chơi dành cho dữ liệu thời tiết không được chỉ định. Thật kỳ lạ, nó dường như được chơi khi trời u ám hoặc mưa chứ không phải khi trời nắng. Có lẽ đó là một cuộc theo đuổi trong nhà.

Thiếu các giá trị và thuộc tính số Mặc dù là một phương pháp học rất thô sơ, nhưng 1R không hỗ trợ cả các giá trị và thuộc tính số bị thiếu. Nó giải quyết những vấn đề này theo những cách đơn giản nhưng hiệu quả. Thiếu chỉ được coi là một giá trị thuộc tính khác, vì vậy, ví dụ: nếu dữ liệu thời tiết chứa các giá trị bị thiếu cho thuộc tính triển vọng, một bộ quy tắc được hình thành trên triển vọng sẽ chỉ định bốn giá trị lớp có thể, mỗi giá trị cho nắng, u ám và mưa và một phần tư cho mất tích.

Chúng ta có thể chuyển đổi các thuộc tính số thành các thuộc tính danh nghĩa bằng cách sử dụng phương pháp xác thực đĩa đơn giản. Đầu tiên, sắp xếp các ví dụ đào tạo theo các giá trị của thuộc tính số. Điều này tạo ra một chuỗi các giá trị lớp. Ví dụ: sắp xếp phiên bản số của dữ liệu thời tiết (Bảng 1.3) theo các giá trị của nhiệt độ sẽ tạo ra chuỗi

64 65 68 69 70 71 72 72 75 75 80 81 83 85

có không có có có không không có có có không có có không

Discretization liên quan đến việc phân vùng trình tự này bằng cách đặt các điểm dừng trong đó. Một khả năng là đặt các điểm ngắt ở bất cứ nơi nào lớp thay đổi, tạo ra tám loại:

có | không | vâng vâng vâng | không không | vâng vâng vâng | không | vâng vâng | KHÔNG

Việc chọn các điểm dừng ở giữa các ví dụ ở hai bên sẽ đặt chúng ở các mức 64,5, 66,5, 70,5, 72, 77,5, 80,5 và 84. Tuy nhiên, hai trường hợp có giá trị 72 gây ra sự cố vì chúng có cùng giá trị nhiệt độ nhưng rơi vào các nhiệt độ khác nhau. các lớp học. Cách khắc phục đơn giản nhất là di chuyển điểm ngắt ở 72 lên một ví dụ, thành 73,5, tạo ra một phân vùng hỗn hợp trong đó không là lớp chiếm đa số.

Một vấn đề nghiêm trọng hơn là thủ tục này có xu hướng hình thành một số lượng lớn các danh mục. Phương pháp 1R đương nhiên sẽ thiên về việc chọn một thuộc tính chia thành nhiều danh mục, bởi vì điều này sẽ phân vùng tập dữ liệu thành nhiều lớp, làm cho nhiều khả năng các cá thể sẽ có cùng lớp với đa số trong phân vùng của chúng. Trên thực tế, trường hợp giới hạn là một thuộc tính có giá trị khác nhau cho mỗi phiên bản—nghĩa là thuộc tính mã nhận dạng xác định các phiên bản duy nhất—và điều này sẽ mang lại tỷ lệ lỗi bằng 0 trên tập huấn luyện vì mỗi phân vùng chỉ chứa một phiên bản. Tất nhiên, các thuộc tính phân nhánh cao thường không hoạt động tốt trên các ví dụ thử nghiệm; thật vậy, thuộc tính mã nhận dạng sẽ không bao giờ dự đoán chính xác bất kỳ ví dụ nào bên ngoài tập huấn luyện. Hiện tượng này được gọi là trang bị quá mức; chúng tôi đã sẵn sàng

đã mô tả xu hướng tránh trang bị quá mức trong Chương 1 (trang 35) và chúng ta sẽ gặp lại vấn đề này nhiều lần trong các chương tiếp theo.

Đối với 1R, việc trang bị quá mức có thể xảy ra bất cứ khi nào một thuộc tính có một số lượng lớn các giá trị có thể. Do đó, khi rời rạc hóa một thuộc tính số, một quy tắc được thông qua quy định số lượng ví dụ tối thiểu của lớp đa số trong mỗi phân vùng. Giả sử rằng mức tối thiểu được đặt ở mức ba. Điều này loại bỏ tất cả trừ hai trong số các phân vùng trước đó. Thay vào đó, quá trình phân vùng bắt đầu

```
có không có có | Đúng...
```

đảm bảo rằng có ba lần xuất hiện có, lớp đa số, trong phân vùng đầu tiên. Tuy nhiên, vì ví dụ tiếp theo cũng có, nên chúng tôi cũng không mất gì bằng cách đưa ví dụ đó vào phân vùng đầu tiên. Điều này dẫn đến một bộ phận mới:

```
có không có có có | không không có có có | không có có không
```

trong đó mỗi phân vùng chứa ít nhất ba phiên bản của lớp đa số, ngoại trừ phiên bản cuối cùng, thường sẽ có ít hơn. Ranh giới phân vùng luôn nằm giữa các ví dụ của các lớp khác nhau.

Bất cứ khi nào các phân vùng liền kề có cùng một lớp đa số, cũng như hai phân vùng đầu tiên ở trên, chúng có thể được hợp nhất với nhau mà không ảnh hưởng đến ý nghĩa của các bộ quy tắc. Do đó, sự rời rạc hóa cuối cùng là

```
có không có có có không không có có | không có có không
```

dẫn đến bộ quy tắc

```
nhiệt độ: £ 77,5 Æ có
                  > 77,5 Æ không
```

Quy tắc thứ hai liên quan đến sự lựa chọn tùy ý; như nó xảy ra, không được chọn. Thay vào đó, nếu chúng tôi chọn có , thì sẽ không cần bất kỳ điểm dừng nào cả— và như ví dụ này minh họa, có thể tốt hơn nếu sử dụng các danh mục liền kề để giúp phá vỡ các mối quan hệ. Trên thực tế, quy tắc này tạo ra năm lỗi trên tập huấn luyện và do đó kém hiệu quả hơn quy tắc trước đối với triển vọng. Tuy nhiên, quy trình tương tự dẫn đến quy tắc này đối với độ ẩm:

```
độ ẩm: £ 82,5 Æ có
              > 82,5 và £ 95,5 Æ không
              > 95,5 Æ có
```

Điều này chỉ tạo ra ba lỗi trên tập huấn luyện và là "quy tắc 1" tốt nhất cho dữ liệu trong Bảng 1.3.

Cuối cùng, nếu một thuộc tính số có các giá trị bị thiếu, thì một danh mục bổ sung sẽ được tạo cho chúng và quy trình rời rạc trước đó chỉ được áp dụng cho các trường hợp mà giá trị của thuộc tính được xác định.

Cuộc thảo luận

Trong một bài báo chuyên đề có tiêu đề "Các quy tắc phân loại rất đơn giản hoạt động tốt trên hầu hết bộ dữ liệu thường được sử dụng" (Holte 1993), một nghiên cứu toàn diện về hiệu suất của thủ tục 1R đã được báo cáo trên 16 bộ dữ liệu thường được sử dụng bởi các nhà nghiên cứu học máy để đánh giá các thuật toán của họ. Xuyên suốt nghiên cứu đã sử dụng xác thực chéo, một kỹ thuật đánh giá mà chúng tôi sẽ giải thích trong Chương 5, để đảm bảo rằng kết quả là đại diện cho những gì độc lập bộ kiểm tra sẽ mang lại. Sau một số thử nghiệm, số lượng tối thiểu các ví dụ trong mỗi phân vùng của một thuộc tính số được đặt ở mức sáu, không phải ba như được sử dụng cho hình minh họa trước đó.

Ngạc nhiên thay, bất chấp sự đơn giản của nó, 1R đã làm một cách đáng kinh ngạc—thậm chí xấu hổ—tốt so với các phương pháp học tập hiện đại và các quy tắc nó được tạo ra hóa ra chỉ kém chính xác hơn một vài điểm phần trăm, trên hầu hết tất cả các bộ dữ liệu, hơn là các cây quyết định được tạo ra bởi sơ đồ cảm ứng cây quyết định hiện đại nhất. Những cây này, nói chung, lớn hơn đáng kể hơn các quy tắc của 1R. Các quy tắc kiểm tra một thuộc tính thường là một giải pháp thay thế khả thi cho các cấu trúc phức tạp hơn, và điều này khuyến khích mạnh mẽ phương pháp luận ưu tiên sự đơn giản, trong đó hiệu suất cơ bản được thiết lập bằng cách sử dụng các kỹ thuật đơn giản, thô sơ trước khi chuyển sang các phương pháp học phức tạp hơn, chắc chắn sẽ tạo ra đầu ra mà mọi người khó giải thích hơn.

Thủ tục 1R học cây quyết định một cấp có các lá đại diện cho nhiều lớp khác nhau. Một kỹ thuật diễn đạt hơn một chút là sử dụng một quy tắc ent khác nhau cho mỗi lớp. Mỗi quy tắc là sự kết hợp của các bài kiểm tra, một bài kiểm tra cho mỗi thuộc tính. Đối với các thuộc tính số, phép thử kiểm tra xem giá trị có nằm trong một khoảng giá trị nhất định hay không; đối với những cái danh nghĩa, nó kiểm tra xem nó có nằm trong một tập hợp con nhất định của thuộc tính đó không các giá trị. Hai loại thử nghiệm này—khoảng và tập hợp con—được học từ dữ liệu huấn luyện liên quan đến mỗi lớp. Đối với một thuộc tính số, các điểm cuối của khoảng thời gian là các giá trị tối thiểu và tối đa xảy ra trong quá trình đào tạo dữ liệu của lớp đó. Đối với một danh nghĩa, tập hợp con chỉ chứa những giá trị mà xảy ra đối với thuộc tính đó trong dữ liệu huấn luyện của lớp. Các quy tắc đại diện cho các lớp khác nhau thường chồng chéo lên nhau và tại thời điểm dự đoán, quy tắc có nhiều kiểm tra phù hợp được dự đoán. Kỹ thuật đơn giản này thường mang lại hiệu quả đầu tiên ấn tượng của một tập dữ liệu. Nó cực kỳ nhanh và có thể được áp dụng cho rất lớn số lượng dữ liệu.

4.2 Mô hình thống kê

Phương pháp 1R sử dụng một thuộc tính duy nhất làm cơ sở cho các quyết định của nó và chọn một trong những hoạt động tốt nhất. Một kỹ thuật đơn giản khác là sử dụng tất cả các thuộc tính và cho phép họ đóng góp vào quyết định quan trọng như nhau và độc lập với nhau, được đưa ra lớp. Điều này là không thực tế, tất nhiên: những gì

Bảng 4.2 Dữ liệu thời tiết với số lượng và xác suất.

Quan điểm			Nhiệt độ			độ ẩm			Co gio			Chơi	
	có	không		có	không		có	không		có	không	có	không
nhiều	2	3	nóng	2	2	cao		4	SAI			9	5
nắng	4	0	nhẹ	4	2	bình thường	3 6	1	ĐÚNG VẬY	6 3	2 3		
nhiều mưa	3	2	mát mẻ	3	1								
nắng u ám 4/9 mưa 3/9	2/9		3/5 nóng 2/9 0/5 nhẹ 4/9 2/5 mát 3/9			2/5 cao 3/9 2/5 bình thường 6/9 1/5			4/5 sai 1/5 đúng			9/6 3/9	5/2 14/9 5/3 14/5

Bảng 4.3 Một ngày mới.

Quan điểm	Nhiệt độ	độ ẩm	Co gio	Chơi
nhiều nắng	mát mẻ	cao	ĐÚNG VẬY	?

làm cho bộ dữ liệu thực tế trở nên thú vị là các thuộc tính chắc chắn không bằng nhau quan trọng hoặc độc lập. Như ng nó dẫn đến một sơ đồ đơ n giản lại hoạt động tốt một cách đáng ngạc nhiên trong thực tế.

Bảng 4.2 trình bày tóm tắt dữ liệu thời tiết thu đư ợc bằng cách đếm cách nhiều lần mỗi cặp thuộc tính-giá trị xảy ra với mỗi giá trị (có và không) cho chơi. Ví dụ, bạn có thể thấy từ Bảng 1.2 rằng triển vọng tốt cho năm bài kiểm tra, hai trong số đó có hiệu quả = có và ba trong số đó có hiệu quả = không. Các tế bào trong hàng đầu tiên của bảng mới, chỉ cần đếm các lần xuất hiện này cho tất cả các lần có thể các giá trị của từng thuộc tính và con số phát trong cột cuối cùng sẽ tính tổng số lần xuất hiện của có và không. Ở phần dư ới của bảng, chúng tôi đã viết lại cùng một thông tin dư ới dạng phân số, hoặc xác suất quan sát đư ợc. Vì ví dụ, trong chín ngày chơi là có, triển vọng là nắng trong hai ngày, mang lại một phân số của 2/9. Để chơi, các phân số là khác nhau: chúng là tỷ lệ của ngày chơi tư ơ ng ứng là có và không .

Bây giờ, giả sử chúng ta gặp một ví dụ mới với các giá trị đư ợc hiển thị trong Bảng 4.3. Chúng tôi coi năm đặc điểm trong Bảng 4.2—triển vọng, nhiệt độ, độ ẩm, gió và khả năng chung là trận đấu là có hoặc không—là những bằng chứng độc lập, quan trọng không kém và nhân các phân số tư ơ ng ứng.
Nhìn vào kết quả có cho:

khả năng có =¥¥¥¥ = 2/9 3 9/3 9 3/9 9 14/ 0 005/3 . .

Các phân số đư ợc lấy từ các mục có trong bảng theo các giá trị của các thuộc tính cho ngày mới và ngày 14/9 cuối cùng là phần tổng thể

đại diện cho tỷ lệ số ngày chơi là có. Một tính toán tương tự
cho kết quả không dẫn đến

khả năng không = ¥¥0/¥0206/3.5 1 5 4 5/3 5 5/14 .

Điều này cho thấy rằng đối với một ngày mới, khả năng là không nhiều hơn là có—gấp bốn lần rất có thể. Các con số có thể được biến thành xác suất bằng cách bình thường hóa chúng để mà họ cộng lại thành 1:

$$\text{Xác suất có} = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%,$$

$$\text{Xác suất của không} = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%.$$

Phương pháp đơn giản và trực quan này dựa trên quy tắc Bayes về khả năng xác định có điều kiện. Quy tắc Bayes nói rằng nếu bạn có giả thuyết H và bằng chứng E phù hợp trên giả thuyết đó, sau đó

$$\text{trước}_{\text{ông}}[H|E] = \frac{\text{trước}[E|H] \times \text{trước}[H]}{\text{trước}[E]}.$$

Chúng tôi sử dụng ký hiệu Pr[A] biểu thị xác suất của một sự kiện A và rằng Pr[A|B] biểu thị xác suất của A có điều kiện đối với một sự kiện B khác . giả thuyết H là trò chơi sẽ diễn ra, giả sử, có, và Pr[H|E] sẽ diễn ra 20,5%, giống như đã xác định trước đó. Bằng chứng E là tổ hợp quốc gia cụ thể của các giá trị thuộc tính cho ngày mới, triển vọng = nắng, nhiệt độ = mát mẻ, độ ẩm = cao, và gió = đúng. Hãy gọi bốn bằng chứng này là E1, E2, E3 và E4 tương ứng. Giả sử rằng những mẩu bằng chứng này là độc lập (với lớp), xác suất kết hợp của chúng có được bằng cách nhân xác suất:

$$\Pr[\text{có}|E] = \frac{\text{trước}[\text{Nắng}|\text{có}] \times \text{trước}[\text{Mát}|\text{có}] \times \text{trước}[\text{Cao}|\text{có}] \times \text{trước}[\text{Đúng}|\text{có}]}{\text{trước}[E]}.$$

Đừng lo lắng về mẫu số: chúng tôi sẽ bỏ qua nó và loại bỏ nó trong bước chuẩn hóa cuối cùng khi chúng ta biến xác suất của tổng có và không thành 1, giống như chúng tôi đã làm trước đây. Pr [yes] ở cuối là xác suất của câu trả lời có kết quả mà không biết bất kỳ bằng chứng E nào, nghĩa là không biết bất kỳ điều gì về ngày cụ thể được tham chiếu—nó được gọi là xác suất trước của giả thuyết H. Trong trường hợp này, chỉ là 14/9, bởi vì 9 trong số 14 ví dụ huấn luyện có một giá trị có để chơi. Thay thế các phân số trong Bảng 4.2 cho các xác suất bằng chứng thích hợp dẫn đến

$$\text{trước}[\text{có}|E] = \frac{2/9 \times 3/9 \times 3/9 \times 9/14}{\text{trước}[E]},$$

giống như chúng tôi đã tính toán trước đó. Một lần nữa, Pr[E] trong mẫu số sẽ không xuất hiện khi chúng ta chuẩn hóa.

Phương pháp này có tên là Naïve Bayes, bởi vì nó dựa trên Bayes's quy tắc và "ngây thơ" giả định tính độc lập—nó chỉ có giá trị để nhân xác suất khi các sự kiện là độc lập. Giả định rằng các thuộc tính là độc lập (được đưa ra trong lớp) trong cuộc sống thực chắc chắn là một điều đơn giản. Nhưng dù cái tên bị chê bai, Naïve Bayes hoạt động rất tốt khi thử nghiệm trên thực tế bộ dữ liệu, đặc biệt khi được kết hợp với một số quy trình lựa chọn thuộc tính được giới thiệu trong Chương 7 để loại bỏ các thuộc tính dư thừa và do đó không phụ thuộc.

Một điều có thể sai với Naïve Bayes là nếu một thuộc tính cụ thể giá trị không xảy ra trong tập huấn luyện kết hợp với mọi giá trị lớp, mọi thứ trở nên tồi tệ. Giả sử trong ví dụ rằng dữ liệu đào tạo là khác nhau và giá trị thuộc tính triển vọng = nắng luôn được liên kết với kết quả không. Sau đó, xác suất triển vọng = nắng nếu có , nghĩa là, Pr[triển vọng = nắng | có], sẽ bằng 0 và bởi vì các xác suất khác là nhân với điều này, xác suất cuối cùng của câu trả lời có sẽ bằng 0 cho dù lớn đến đâu họ đã từng. Xác suất bằng 0 giữ quyền phủ quyết đối với những xác suất khác. Đây không phải là một ý kiến hay. Nhưng lỗi này dễ dàng được khắc phục bằng những điều chỉnh nhỏ đối với phương pháp tính toán xác suất từ tần số.

Ví dụ, phần trên của Bảng 4.2 cho thấy đối với play = yes, triển vọng là nắng cho hai ví dụ, u ám cho bốn ví dụ và mưa cho ba và phần dưới cho các sự kiện này xác suất lần lượt là 2/9, 4/9 và 3/9. thay vào đó, chúng tôi có thể thêm 1 vào mỗi tử số và bù lại bằng cách thêm 3 vào mẫu số, cho các xác suất tương ứng là 3/12, 5/12 và 4/12. Điều này sẽ đảm bảo rằng một giá trị thuộc tính không xảy ra lần nào sẽ nhận được một xác suất khác không, dù nhỏ. Chiến lược thêm 1 vào mỗi lần đếm là một kỹ thuật tiêu chuẩn được gọi là công cụ ước tính Laplace sau nhà toán học vĩ đại người Pháp thế kỷ thứ mười tám Pierre Laplace. Mặc dù nó hoạt động tốt trong thực tế, không có lý do cụ thể để thêm 1 vào số đếm: thay vào đó, chúng ta có thể chọn một hằng số m nhỏ và sử dụng

$$\frac{2 + m/3}{9 + m}, \frac{4 + m/3}{9 + m} \text{ Và } \frac{3 + m/3}{9 + m}$$

Giá trị của m, được đặt thành 3, cung cấp hiệu quả một trọng số xác định mức độ ảnh hưởng của các giá trị tiên nghiệm 1/3, 1/3 và 1/3 đối với từng giá trị trong số ba các giá trị thuộc tính có thể. Một người lớn nói rằng những tiên đoán này rất quan trọng so với bằng chứng mới đến từ tập huấn luyện, trong khi một người nhỏ một cho họ ít ảnh hưởng hơn. Cuối cùng, không có lý do cụ thể nào để chia m thành ba phần bằng nhau ở các tử số: chúng ta có thể sử dụng

$$\frac{2 + m_1}{9 + m}, \frac{4 + m_2}{9 + m}, \text{Và } \frac{3 + m_3}{9 + m}$$

trang 12

thay vào đó, trong đó p1, p2 và p3 có tổng bằng 1. Thực tế, ba số này là tiên nghiệm xác suất của các giá trị của thuộc tính triển vọng là nắng, u ám và mưa, tương ứng.

Đây hiện là một công thức Bayes hoàn chỉnh trong đó các xác suất trước đó đã được gán cho mọi thứ trong tầm nhìn. Nó có ưu điểm là hoàn toàn nghiêm ngặt, nhưng nhược điểm là thường không rõ ràng về cách thức chỉ định những khả năng thăm dò trước đó. Trong thực tế, các xác suất trước đó làm cho ít sự khác biệt với điều kiện là có một số lượng hợp lý các trường hợp đào tạo, và mọi người thường chỉ ước tính tần số bằng cách sử dụng công cụ ước tính Laplace theo khởi tạo tất cả các số đếm thành một thay vì bằng không.

Thiếu giá trị và thuộc tính số

Một trong những điều thực sự hay về công thức Bayesian là thiếu các giá trị không có vấn đề gì cả. Ví dụ: nếu giá trị của triển vọng bị thiếu trong ví dụ của Bảng 4.3, phép tính sẽ đơn giản bỏ qua thuộc tính này, năng suất

khả năng xảy ra 3 = 9 ¥ 3 ¥ 3 ¥ 9 = 3 9 9 14 0 0238 .
Dùng

kế sinh nhai của KHÔNG = 5 ¥ 4 ¥ 8 ¥ 5 = 3 5 5 14 0 0343 . .

Hai con số này cao hơn rất nhiều so với trước đây, bởi vì một trong các phân số bị thiếu. Nhưng đó không phải là vấn đề vì một phân số là bị thiếu trong cả hai trường hợp và những khả năng này phải tuân theo quy trình bình thường hóa hơn nữa. Điều này mang lại xác suất cho có và không là 41% và 59%, tương ứng.

Nếu một giá trị bị thiếu trong một phiên bản đào tạo, thì đơn giản là nó không được đưa vào số lượng tần suất và tỷ lệ xác suất dựa trên số lượng giá trị điều đó thực sự xảy ra chứ không phải trên tổng số trường hợp.

Các giá trị số thường được xử lý bằng cách giả định rằng chúng có giá trị "bình thường" hoặc phân phối xác suất “Gaussian”. Bảng 4.4 tóm tắt về thời tiết dữ liệu với các tính năng số từ Bảng 1.3. Đối với các thuộc tính danh nghĩa, chúng tôi tính toán số lượng như trước đây và đối với các thuộc tính số, chúng tôi chỉ liệt kê các giá trị mà xảy ra. Sau đó, trong khi chúng tôi chuẩn hóa số đếm cho các thuộc tính danh nghĩa thành xác suất, chúng tôi đã tính giá trị trung bình và độ lệch chuẩn cho từng loại và từng thuộc tính số. Do đó, giá trị trung bình của nhiệt độ trên có trường hợp là 73 và độ lệch chuẩn của nó là 6,2. Giá trị trung bình chỉ đơn giản là trung bình của các giá trị trước đó, nghĩa là tổng chia cho số lượng giá trị. Các độ lệch chuẩn là căn bậc hai của phương sai mẫu, mà chúng ta có thể tính toán như sau: trừ giá trị trung bình từ mỗi giá trị, bình phương kết quả, tính tổng chúng với nhau, rồi chia cho một số nhỏ hơn số giá trị. Sau khi chúng tôi có tìm thấy phương sai mẫu này, hãy tìm căn bậc hai của nó để xác định độ lệch chuẩn. Đây là cách tiêu chuẩn để tính giá trị trung bình và độ lệch chuẩn của một

Bảng 4.4 Dữ liệu thời tiết dạng số với số liệu thống kê tóm tắt.

Quan điểm		Nhiệt độ	độ ẩm		Co gio		Chơi	
có	không	có không	có	không	có	không	có	không
2 nắng	3	83 85 70	86	85	sai 6 3		9	5
u ám 4 mưa 3	0	80 68 65	96	90	ĐÚNG VẬY	2 3		
	2	64 72 69	80	70				
		71 75 75	65	95				
		72 81	70	91				
			80					
			70					
			90					
			75					
trời	9/2 3/5 có nghĩa là	73 74,6 có nghĩa là	79.1 86.2 sai 6/9 2/5 9/14 5/14					
nhiều mây 4/9 0/5 tiêu chuẩn. nhà phát		7,9 std. nhà phát triển 10.2	9,7 đúng 3/9 3/5					
triển 6.2 mưa 3/9 2/5								

tập hợp các số ("ít hơn một" liên quan đến số bậc của tự do trong mẫu, một khái niệm thống kê mà chúng tôi không muốn đề cập ở đây).

Hàm mật độ xác suất cho phân phối chuẩn với giá trị trung bình m và độ lệch chuẩn s được đưa ra bởi biểu thức khá ghê gớm:

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

Nhưng đừng sợ! Tất cả điều này có nghĩa là nếu chúng ta đang xem xét một kết quả có khi nhiệt độ có giá trị, giả sử, là 66, chúng ta chỉ cần thay x = 66, m = 73 và s = 6.2 vào công thức. Vậy giá trị của hàm mật độ xác suất là

$$f(\text{nhiệt độ} = 66 \mid \text{Đúng}) = \frac{1}{\sqrt{2\pi}\,6.2} e^{\frac{(66-73)^2}{2\cdot 6.2^2}} = 0.0340.$$

Tương tự như vậy, mật độ xác suất của kết quả có khi độ ẩm có giả sử, giá trị của 90 được tính theo cùng một cách:

$$f(\text{độ ẩm} = 90 \mid \text{có}) = 0.0221.$$

Hàm mật độ xác suất cho một sự kiện có liên quan rất chặt chẽ với khả năng xác suất của nó. Tuy nhiên, nó không hoàn toàn giống nhau. Nếu nhiệt độ liên tục thang đo, xác suất của nhiệt độ chính xác là 66—hoặc chính xác bất kỳ giá trị nào khác giá trị, chẳng hạn như 63.14159262—bằng không. Ý nghĩa thực sự của hàm mật độ f(x) là xác suất để đại lượng nằm trong một miền nhỏ xung quanh x, giả sử, giữa x - e/2 và x + e/2, là e f(x). Những gì chúng tôi đã viết ở trên là chính xác

nếu nhiệt độ được đo đến độ gần nhất và độ ẩm được đo đến điểm phần trăm gần nhất. Bạn có thể nghĩ rằng chúng ta nên tính đến con số e chính xác khi sử dụng các xác suất này, nhưng điều đó là không cần thiết. Giống nhau e sẽ xuất hiện trong cả khả năng có và không xảy ra theo sau và hủy bỏ khi các xác suất được tính toán.

Sử dụng các xác suất này cho ngày mới trong Bảng 4.5 mang lại kết quả

$$\text{khả năng}_{\text{Đúng}} = 2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036,$$

$$\text{kế sinh nhai của}_{\text{KHÔNG}} = 3/5 \times 0.0221 \times 0.0381 \times 3/5 \times 5/14 = 0.000108;$$

dẫn đến xác suất

$$\text{Xác suất có} = \frac{0.000036}{0.000036 + 0.000108} = 25\%,$$

$$\text{Xác suất không} = \frac{0.000108}{0.000036 + 0.000108} = 75\%.$$

Những con số này rất gần với xác suất được tính toán trước đó cho cái mới ngày trong bảng 4.3, vì các giá trị nhiệt độ và độ ẩm của 66 và 90 xác suất tương tự với các giá trị mát mẻ và cao được sử dụng trước đó.

Giả định phân phối chuẩn giúp dễ dàng mở rộng Naïve Bayes bộ phân loại để xử lý các thuộc tính số. Nếu giá trị của bất kỳ thuộc tính số nào bị thiếu, các phép tính trung bình và độ lệch chuẩn chỉ dựa trên những người có mặt.

Mô hình Bayesian để phân loại tài liệu

Một lĩnh vực quan trọng cho học máy là phân loại tài liệu, trong mà mỗi thể hiện đại diện cho một tài liệu và lớp của thể hiện là chủ đề của tài liệu. Tài liệu có thể là các mục tin tức và các lớp có thể là tin tức trong nước, tin tức nước ngoài, tin tức tài chính và thể thao. Tài liệu được đặc trưng bằng các từ xuất hiện trong đó và một cách để áp dụng học máy vào phân loại tài liệu là coi sự hiện diện hay vắng mặt của mỗi từ là một thuộc tính Boolean. Naïve Bayes là một kỹ thuật phổ biến cho ứng dụng này vì nó rất nhanh và khá chính xác.

Tuy nhiên, điều này không tính đến số lần xuất hiện của mỗi từ, có khả năng là thông tin hữu ích khi xác định danh mục

Bảng 4.5 Lại một ngày mới.

Quan điểm	Nhiệt độ	độ ẩm	Co gio	Chơi
nhiều nắng	66	90	ĐÚNG VẬY	?

của một tài liệu. Thay vào đó, một tài liệu có thể được xem như một túi từ—một tập hợp chứa tất cả các từ trong tài liệu, với nhiều lần xuất hiện của một từ xuất hiện nhiều lần (về mặt kỹ thuật, một tập hợp bao gồm mỗi phần tử của nó chỉ một lần, trong khi một túi có thể có các phần tử lặp lại). Tần số từ có thể là hỗ trợ bằng cách áp dụng một dạng Naïve Bayes đã được sửa đổi đôi khi được mô tả là Naïve Bayes đa danh .

Giả sử n1, n2,..., nk là số lần từ i xuất hiện trong tài liệu, và P1, P2,..., Pk là xác suất lấy được từ i khi lấy mẫu từ tất cả các tài liệu trong danh mục H. Giả sử rằng xác suất không phụ thuộc vào ngữ cảnh và vị trí của từ trong tài liệu. Những giả định này dẫn đến một phân phối đa thức cho xác suất tài liệu. Đối với phân phối này, xác suất của một tài liệu E với lớp H của nó—nói cách khác, công thức cho tính xác suất Pr[E|H] trong quy tắc Bayes—là

$$\Pr[E|H] = N! \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!}$$

trong đó N = n1 + n2 + ··· + nk là số từ trong tài liệu. Nguyên nhân đối với giai thừa là để tính đến thực tế là thứ tự của các lần xuất hiện của mỗi từ là không quan trọng theo mô hình túi từ. Pi được ước tính bằng cách tính toán tần số tương đối của từ i trong văn bản của tất cả các tài liệu đào tạo liên quan đến loại H. Trong thực tế, nên có một thuật ngữ nữa mà đưa ra xác suất mà mô hình cho loại H tạo ra một tài liệu có chiều dài bằng với chiều dài của E (đó là lý do tại sao chúng ta sử dụng ký hiệu ª thay thế của =), như ng ngư ời ta thư ờng cho rằng điều này giống nhau đối với tất cả các lớp và do đó có thể được thả.

Ví dụ, giả sử chỉ có hai từ, màu vàng và màu xanh, trong từ vựng và một tài liệu cụ thể lớp H có Pr[yellow|H] = 75% và Pr[blue|H] = 25% (bạn có thể gọi H là loại tài liệu màu lục vàng). Giả sử E là tài liệu màu xanh vàng xanh có độ dài N = 3 từ. Ở đó là bốn túi có thể có ba từ. Một là {vàng vàng vàng}, và khả năng xác suất của nó theo công thức trư ớc đó là

$$\Pr[\{\text{vàng vàng vàng}\} \mid H] = 3! \cdot \frac{0.75^3}{3!} \cdot \frac{0.25^0}{0!} = \frac{27}{64}$$

Ba cái còn lại, với xác suất của chúng, là

$$\Pr[\{\text{xanh xanh xanh}\} \mid H] = \frac{1}{64}$$

$$\Pr[\{\text{vàng vàng xanh}\} \mid H] = \frac{27}{64}$$

$$\Pr[\{\text{vàng xanh vàng}\} \mid H] = \frac{9}{64}$$

Ở đây, E tương ứng với trường hợp cuối cùng (hãy nhớ lại rằng trong một túi từ, thứ tự là phi vật chất); do đó, xác suất nó được tạo ra bởi mô hình tài liệu màu lục vàng là 9/64, hay 14%. Giả sử một lớp khác, tài liệu có màu xanh hơi xanh (gọi là HC), có Pr[vàng | HC] = 10%, Pr[lam | HC] = 90%. Xác suất
mà E được tạo ra bởi mô hình này là 24%.

Nếu đây là hai lớp duy nhất, điều đó có nghĩa là E có màu rất xanh ?
lớp tài liệu xanh ? Không cần thiết. Quy tắc Bayes, được đưa ra trước đó, nói rằng bạn phải tính đến xác suất trước của mỗi giả thuyết. Nếu bạn biết rằng trên thực tế, các tài liệu có màu xanh lục rất hiếm gấp đôi so với các tài liệu có màu xanh lục hơi vàng , điều này sẽ vừa đủ để vượt qua mức chênh lệch 14% đến 24% trước đó và nghiêng cán cân nghiêng về lớp màu xanh lục vàng .

Các giai thừa trong công thức xác suất trước đó không thực sự cần phải được tính toán bởi vì—là giống nhau đối với mọi lớp—dù sao thì chúng cũng bị bỏ qua trong quá trình chuẩn hóa bình thường. Tuy nhiên, công thức vẫn liên quan đến việc nhân nhiều xác suất nhỏ cùng nhau, sẽ sớm mang lại những con số cực kỳ nhỏ gây ra tràn trên các tài liệu lớn. Vấn đề có thể tránh được bằng cách sử dụng logarit của xác suất thay vì bản thân xác suất.

Trong công thức Naïve Bayes đa thức, lớp của tài liệu được xác định không chỉ bởi những từ xuất hiện trong đó mà còn bởi số lần chúng xuất hiện. Nói chung, nó hoạt động tốt hơn so với mô hình Naïve Bayes thông thường để phân loại tài liệu, đặc biệt đối với kích thước từ điển lớn.

Cuộc thảo luận

Naïve Bayes đưa ra một cách tiếp cận đơn giản, với ngữ nghĩa rõ ràng, để biểu diễn, sử dụng và học kiến thức xác suất. Kết quả ấn tượng có thể đạt được sử dụng nó. Người ta thường chứng minh rằng các đối thủ của Naïve Bayes, và thực sự là các dạng bên ngoài, các bộ phân loại phức tạp hơn trên nhiều bộ dữ liệu. Đạo đức là, luôn luôn cố gắng những điều đơn giản đầu tiên. Nhiều lần trong học máy, cuối cùng mọi người đã, sau một cuộc đấu tranh kéo dài, thu được kết quả tốt bằng cách sử dụng học tập tinh vi nhiều năm sau mới khám phá ra rằng các phương pháp đơn giản như 1R và Naïve Bayes cũng làm tốt — hoặc thậm chí tốt hơn.

Tuy nhiên, có nhiều bộ dữ liệu mà Naïve Bayes không làm tốt lắm, và thật dễ hiểu tại sao. Bởi vì các thuộc tính được xử lý như thể chúng hoàn toàn độc lập, nên việc bổ sung các thuộc tính dư thừa sẽ làm sai lệch quá trình học. Như một ví dụ điển hình, nếu bạn bao gồm một thuộc tính mới có cùng các giá trị như nhiệt độ đối với dữ liệu thời tiết, ảnh hưởng của thuộc tính nhiệt độ sẽ được nhân lên: tất cả các xác suất của nó sẽ được bình phương, mang lại cho nó một nhiều ảnh hưởng hơn trong quyết định. Nếu bạn thêm 10 thuộc tính như vậy, sau đó các quyết định sẽ được đưa ra một cách hiệu quả chỉ dựa trên nhiệt độ . Sự phụ thuộc giữa các thuộc tính chắc chắn làm giảm sức mạnh của Naïve Bayes để phân biệt những gì đang xảy ra. Tuy nhiên, chúng có thể được cải thiện bằng cách sử dụng một tập hợp con của

các thuộc tính trong thủ tục quyết định, lựa chọn cẩn thận những thuộc tính nào sẽ sử dụng. Chư ơ ng 7 cho thấy làm thế nào.

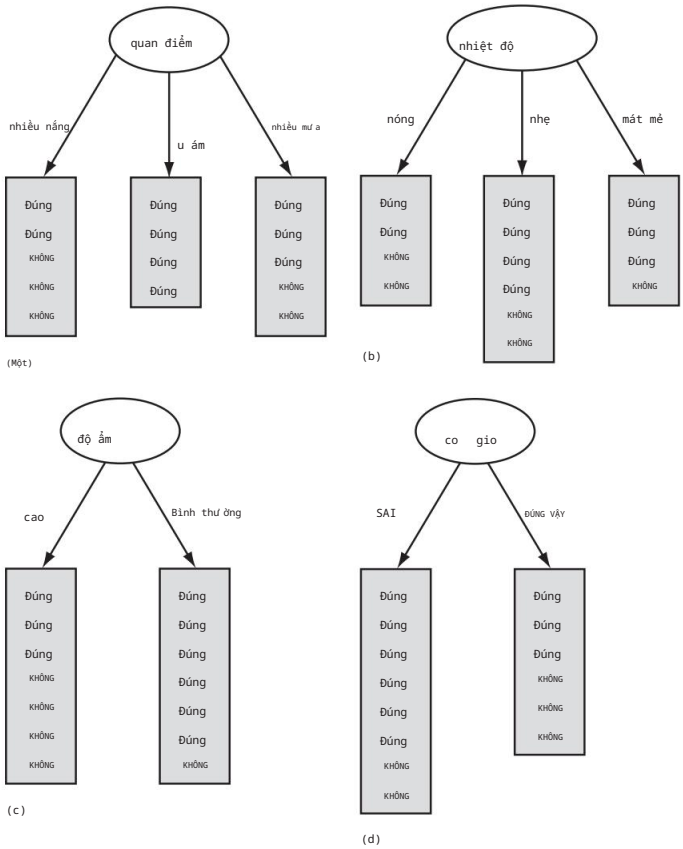
Giả định phân phối chuẩn cho các thuộc tính số là một hạn chế khác đối với Naïve Bayes khi chúng tôi đã xây dựng nó ở đây. Nhiều tính năng đơ n giản là không đư ợc phân phối bình thư ờng. Tuy nhiên, không có gì ngăn cản chúng ta sử dụng các phân phối khác cho các thuộc tính số: không có gì kỳ diệu về phân phối bình thư ờng. Nếu bạn biết rằng một thuộc tính cụ thể có khả năng tuân theo một số phân phối khác, thì có thể sử dụng các quy trình ư ớc lư ợng tiêu chuẩn cho phân phối đó. Nếu bạn nghi ngờ điều đó không bình thư ờng như ng không biết phân phối thực tế, thì có các quy trình "ư ớc tính mật độ hạt nhân" không giả định bất kỳ phân phối cụ thể nào cho các giá trị thuộc tính. Một khả năng khác đơ n giản là rời rạc hóa dữ liệu trư ớc.

4.3 Chia để trị: Xây dựng cây quyết định

Bài toán xây dựng cây quyết định có thể biểu diễn đệ quy. Đầu tiên, chọn một thuộc tính để đặt tại nút gốc và tạo một nhánh cho mỗi giá trị có thể. Điều này chia tập hợp ví dụ thành các tập hợp con, một tập hợp cho mọi giá trị của thuộc tính. Bây giờ quá trình có thể đư ợc lặp lại đệ quy cho mỗi nhánh, chỉ sử dụng những trư ờng hợp thực sự đến nhánh. Nếu tại bất kỳ thời điểm nào, tất cả các phiên bản tại một nút có cùng phân loại, hãy ngừng phát triển phần đó của cây.

Điều duy nhất còn lại để quyết định là làm thế nào để xác định thuộc tính nào sẽ phân chia, dựa trên một tập hợp các ví dụ với các lớp khác nhau. Xem xét (một lần nữa!) dữ liệu thời tiết. Có bốn khả năng cho mỗi lần phân chia, và ở mức cao nhất, chúng tạo ra các cây như trong Hình 4.2. Lựa chọn nào là tốt nhất? Số lớp có và không đư ợc hiển thị trên lá. Bất kỳ lá nào chỉ có một lớp—có hoặc không—sẽ không phải chia thêm nữa và quá trình đệ quy xuống nhánh đó sẽ kết thúc. Bởi vì chúng tôi tìm kiếm những cây nhỏ, chúng tôi muốn điều này xảy ra càng sớm càng tốt. Nếu chúng ta có thư ớc đo độ tinh khiết của mỗi nút, chúng ta có thể chọn thuộc tính tạo ra các nút con tinh khiết nhất. Hãy dành một chút thời gian để xem Hình 4.2 và suy nghĩ xem bạn nghĩ thuộc tính nào là lựa chọn tốt nhất.

Thư ớc đo độ tinh khiết mà chúng ta sẽ sử dụng đư ợc gọi là thông tin và đư ợc đo bằng đơ n vị gọi là bit. Đư ợc liên kết với một nút của cây, nó biểu thị lư ợng thông tin dự kiến cần thiết để xác định liệu một thể hiện mới có nên đư ợc phân loại có hay không, với điều kiện là ví dụ đã đến đư ợc nút đó.
Không giống như các bit trong bộ nhớ máy tính, lư ợng thông tin dự kiến thư ờng bao gồm các phân số của một bit—và thư ờng ít hơ n một! Chúng tôi tính toán nó dựa trên số lư ợng lớp có và không có tại nút; chúng ta sẽ xem xét các chi tiết của phép tính ngay sau đây. Như ng trư ớc tiên hãy xem nó đư ợc sử dụng như thế nào. Khi đánh giá cây đầu tiên trong Hình 4.2, số lớp có và không có tại các nút lá lần lư ợt là [2,3], [4,0] và [3,2] và giá trị thông tin của các nút này là :



Hình 4.2 Gốc cây cho dữ liệu thời tiết.

thông tin ([ ]) = 0.971 chút ít

thông tin ([ ]) = 0.00 chút ít

thông tin ([ ]) = 0.971 chút ít

Chúng ta có thể tính toán giá trị thông tin trung bình của những thứ này, có tính đến số trường hợp đi xuống mỗi nhánh—năm xuống nhánh thứ nhất và thứ ba và bốn xuống thứ hai:

thông tin ([ ]) = (+) × ( ) bit = 0.971 bit

Giá trị trung bình này đại diện cho lượng thông tin mà chúng ta mong đợi là cần thiết để xác định lớp của một thể hiện mới, dựa trên cấu trúc cây trong Hình 4.2(a).

Trư ớc khi chúng ta tạo bất kỳ cấu trúc cây non trẻ nào trong Hình 4.2, các ví dụ huấn luyện ở gốc bao gồm chín nút có và năm nút không , tư ơ ng ứng đến một giá trị thông tin của

$$\text{thông tin}([9,5]) = 0.940 \text{ bit.}$$

Do đó, cây trong Hình 4.2(a) chịu trách nhiệm thu đư ợc thông tin về

$$\text{thu được thông tin (triển vọng)} = 9(\text{thông tin}([3,4])) = 0.940 \cdot \frac{7}{14} = 0.693 \text{ bit,}$$

có thể đư ợc hiểu là giá trị thông tin của việc tạo một nhánh trên thuộc tính triển vọng .

Con đư ờng phía trư ớc là rõ ràng. Chúng tôi tính toán mức thu đư ợc thông tin cho mỗi thuộc tính và chọn một thuộc tính thu đư ợc nhiều thông tin nhất để phân chia. Trong tình huống của Hình 4.2,

$$\begin{aligned}
\text{tăng (triển vọng)} &= 0.247 \text{ bit} \\
\text{tăng (nhiệt độ)} &= 0.029 \text{ bit} \\
\text{tăng (độ ẩm)} &= 0.152 \text{ bit} \\
\text{tăng (gió)} &= 0.048 \text{ bit,}
\end{aligned}$$

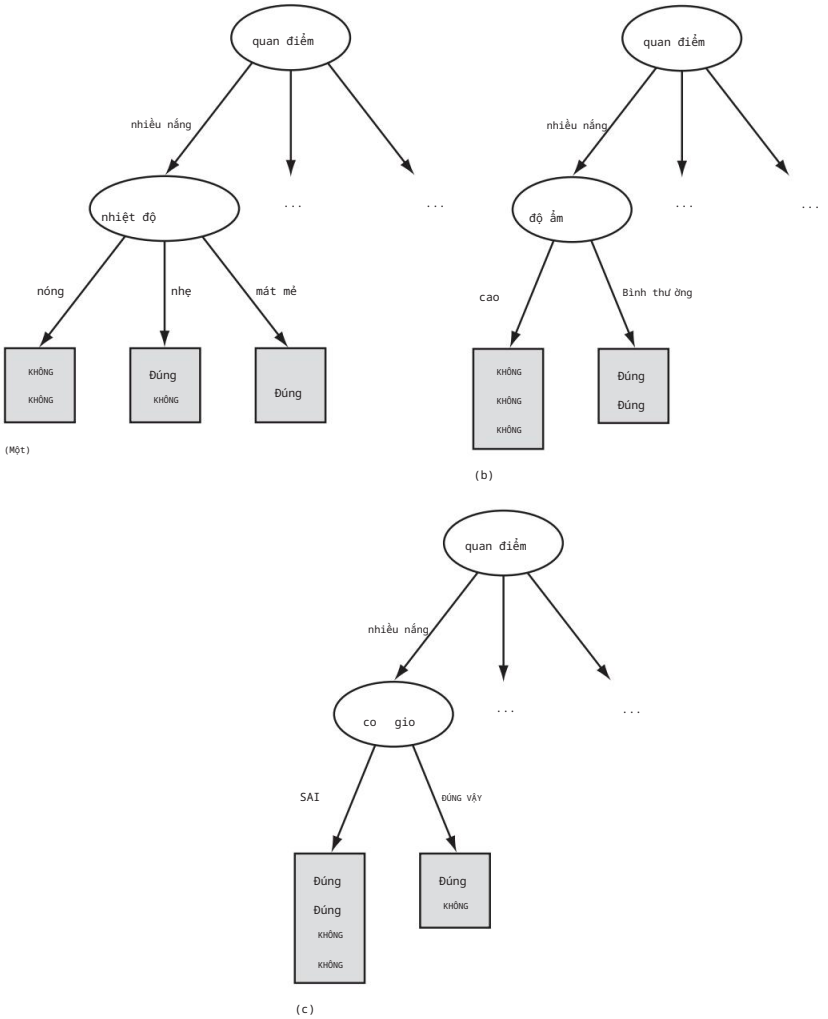
vì vậy chúng tôi chọn triển vọng làm thuộc tính phân tách ở gốc của cây. hy vọng điều này phù hợp với trực giác của bạn là thứ tốt nhất để chọn. Đó là sự lựa chọn duy nhất mà một nút con hoàn toàn trong sạch, và điều này mang lại cho nó một giá trị đáng kể lợi thế so với các thuộc tính khác. Độ ẩm là sự lựa chọn tốt nhất tiếp theo bởi vì nó tạo ra một nút con lớn hơ n gần như hoàn toàn tinh khiết.

Sau đó, chúng tôi tiếp tục, đệ quy. Hình 4.3 cho thấy các khả năng cho một nhánh tại nút đạt đư ợc khi trời nắng . Rõ ràng, một sự phân chia hơ n nữa về triển vọng sẽ không tạo ra gì mới, vì vậy chúng tôi chỉ xem xét ba thuộc tính còn lại. Thông tin đạt đư ợc cho mỗi hóa ra là

$$\begin{aligned}
\text{đạt được (nhiệt độ)} &= 0.571 \text{ bit} \\
\text{tăng (độ ẩm)} &= 0.971 \text{ bit} \\
\text{đạt được (gió)} &= 0.020 \text{ bit,}
\end{aligned}$$

vì vậy chúng tôi chọn độ ẩm làm thuộc tính phân tách tại thời điểm này. Không cần đến tách các nút này ra nữa, vậy là xong nhánh này.

Tiếp tục áp dụng cùng một ý tư ởng dẫn đến cây quyết định của Hình 4.4 cho dữ liệu thời tiết. Lý tư ởng nhất là quá trình kết thúc khi tất cả các nút lá đư ợc thuần túy, nghĩa là khi chúng chứa các thể hiện có cùng phân loại. Tuy nhiên, có thể không đạt đư ợc tình huống hạnh phúc này vì có không có gì ngăn cản tập huấn luyện chứa hai ví dụ với các tập giống hệt nhau thuộc tính như ng khác lớp. Do đó, chúng tôi dừng lại khi dữ liệu không thể đư ợc chia xa nữa.

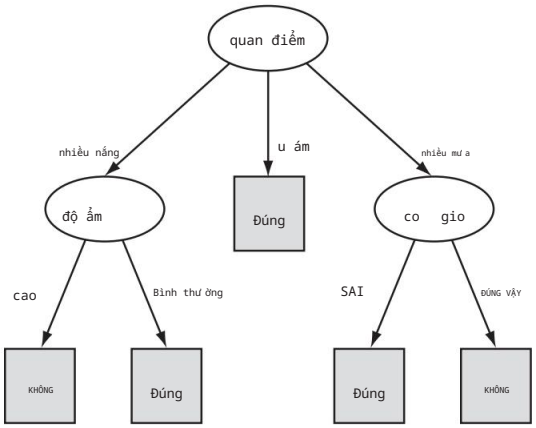


Hình 4.3 Mở rộng gốc cây cho dữ liệu thời tiết.

Thông tin tính toán

Bây giờ là lúc giải thích cách tính thước đo thông tin được sử dụng
làm cơ sở để đánh giá các cách chia khác nhau. Chúng tôi mô tả ý tưởng cơ bản trong phần này,
sau đó trong phần tiếp theo, chúng tôi kiểm tra một hiệu chỉnh thường được thực hiện để chống lại sự thiên vị
hướng tới việc chọn các phần tách trên các thuộc tính có số lượng lớn các giá trị có thể.

Trước khi kiểm tra công thức chi tiết để tính lượng thông tin cần thiết để xác định loại
của một ví dụ cho rằng nó vượn tới một cái cây
nút có một số câu trả lời có và không nhất định, trước tiên hãy xem xét loại ràng buộc phù
hợp mà chúng tôi mong đợi số lượng này sẽ có:



Hình 4.4 Cây quyết định cho dữ liệu thời tiết.

1. Khi số câu trả lời có hoặc không bằng 0, thông tin là
   số không.
2. Khi số lư ợng câu trả lời có và không bằng nhau, thông tin đạt đến mức
   tối đa.

Hơ n nữa, biện pháp nên đư ợc áp dụng cho các tình huống đa lớp, không chỉ cho
loại hai lớp.

Thư ớc đo thông tin liên quan đến lư ợng thông tin thu đư ợc bởi
đư a ra quyết định và một thuộc tính tinh vi hơ n của thông tin có thể đư ợc suy ra
bằng cách xem xét bản chất của các quyết định. Các quyết định có thể đư ợc đư a ra trong một giai đoạn duy nhất,
hoặc chúng có thể đư ợc thực hiện trong một số giai đoạn và lư ợng thông tin liên quan
là như nhau trong cả hai trư ờng hợp. Ví dụ, quyết định liên quan đến

thông tin 2,3,4 ([ ])

có thể đư ợc thực hiện trong hai giai đoạn. Trư ớc tiên hãy quyết định xem đó là trư ờng hợp đầu tiên hay một trong những
hai trư ờng hợp khác:

thông tin 2,7 ([ ])

và sau đó quyết định đó là trư ờng hợp nào trong hai trư ờng hợp còn lại:

thông tin 3,4 ([ ])

Trong một số trư ờng hợp, quyết định thứ hai sẽ không cần phải đư ợc đư a ra, cụ thể là khi quyết
định trở thành quyết định đầu tiên. Tính đến điều này dẫn đến
phư ơ ng trình

([thông tin 2,3,4]) = ([ ]) + (79) ¥ ([ ]) .

Tất nhiên, không có gì đặc biệt về những con số cụ thể này, và tương tự mối quan hệ phải giữ bất kể các giá trị thực tế. Vì vậy, chúng ta có thể thêm một tiêu chí cho danh sách trước:

3. Thông tin phải tuân theo thuộc tính nhiều tầng được minh họa trước đó.

Đáng chú ý, hóa ra chỉ có một chức năng thỏa mãn tất cả những điều này thuộc tính, và nó được gọi là giá trị thông tin hoặc entropy:

$$\text{entropy}(p_1, p_2, \ldots, p_n) = -\log(p_1) - \log(p_2) - \cdots - \log(p_n)$$

Sở dĩ có dấu trừ là logarit của các phân số p1, p2,..., pn là âm, vì vậy entropy thực sự là dương. Thông thường các logarit là biểu thị trong cơ số 2, thì entropy tính bằng đơn vị gọi là bit—chỉ là loại thông thường bit được sử dụng với máy tính.

Các đối số p1, p2, . . . của công thức entropy được biểu thị dưới dạng phân số cộng lại thành một, vì vậy, ví dụ,

$$\text{thông tin}(2,3,4) = \text{entropy}(2/9, 3/9, 4/9)$$

Do đó, thuộc tính quyết định nhiều tầng có thể được viết một cách tổng quát là

$$\text{entropy}(p,q,r) = \text{entropy}(p,q+r) + (q+r)\,\text{entropy}\left(\frac{q}{q+r}, \frac{r}{q+r}\right)$$

trong đó p + q + r = 1.

Do cách thức hoạt động của chức năng nhật ký, bạn có thể tính toán thông tin đo lường mà không cần phải tìm ra các phân số riêng lẻ:

$$\text{thông tin}(2,3,4) = -2/9 \times \log 2/9 - 3/9 \times \log 3/9 - 4/9 \times \log 4/9$$
$$= [\,2 \log 2 + 3 \log 3 + 4 \log 4 - 9 \log 9\,]/9$$

Đây là cách mà thước đo thông tin thường được tính toán trong thực tế. Vậy giá trị thông tin cho nút lá đầu tiên của cây đầu tiên trong Hình 4.2 là

$$\text{thông tin}(2,3,4) = [\,2 \log 2 + 3 \log 3 + 4 \log 4 - 9 \log 9\,]/9 = 3.5 - 0.971 \text{ bit},$$

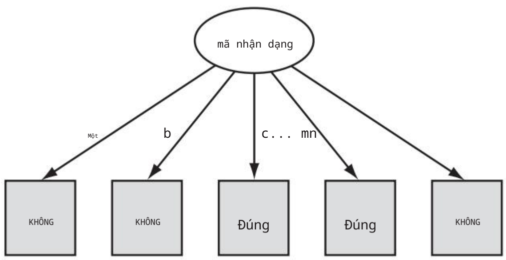
như đã nêu ở trang 98.

Thuộc tính phân nhánh cao

Khi một số thuộc tính có một số lượng lớn các giá trị có thể, dẫn đến một nhánh nhiều nhánh với nhiều nút con, một vấn đề phát sinh với thông tin tính toán được. Vấn đề có thể được đánh giá tốt nhất trong trường hợp cực đoan khi một thuộc tính có một giá trị khác nhau cho mỗi phiên bản trong tập dữ liệu—chẳng hạn như đối với ví dụ, một thuộc tính mã nhận dạng có thể.

Bảng 4.6 Dữ liệu thời tiết với mã nhận dạng.

mã nhận dạng	Quan điểm	Nhiệt độ	độ ẩm	Co gio	Chơ i
	nắng		cao	SAI	KHÔNG
ab	nhiều		cao	đúng	KHÔNG
	mây		cao	sai	Đúng
đĩa CD	mư a	nóng	cao	sai	Đúng
thành	mư a	nóng	bình	sai	Đúng
công	mư a	nóng	thư ờng	ĐÚNG VẬY	KHÔNG
	u ám	nhẹ	bình	đúng	Đúng
	nắng	mát		sai	KHÔNG
	nắng	mát	thư ờng	sai	Đúng
	mư a	mát	bình	sai	Đúng
	nắng u	nhẹ	thư ờng	ĐÚNG VẬY	Đúng
ghijkl	ám	mát	cao	đúng	Đúng
tôi	u ám	nhẹ	bình	sai	Đúng
N	nhiều mư a	nhẹ nhẹ nóng nhẹ	thư ờng bình thư ờng cao bình thư ờng cao	ĐÚNG VẬY	KHÔNG



Hình 4.5 Gốc cây cho **thuộc tính** mã ID .

Bảng 4.6 cung cấp dữ liệu thời tiết với thuộc tính bổ sung này. Phân nhánh trên ID mã tạo gốc cây trong Hình 4.5. Các thông tin cần thiết để xác định lớp đư ợc cung cấp giá trị của thuộc tính này là

([thông tin 0,1([ ])thông tin 0,1]) +thông tin 1,0 .([tin +1,0 (thông +tin[0,1]) ,

bằng không vì mỗi số trong 14 số hạng bằng không. Điều này không có gì đáng ngạc nhiên: ID **thuộc tính** mã xác định thể hiện, xác định lớp mà không có bất kỳ mơ hồ—như Bảng 4.6 cho thấy. Do đó, việc thu đư ợc thông tin này thuộc tính chỉ là thông tin ở gốc, thông tin ([9,5]) = 0,940 bit. Đây là lớn hơ n mức tăng thông tin của bất kỳ thuộc tính nào khác và do đó mã ID sẽ chắc chắn đư ợc chọn làm thuộc tính phân tách. Như ng phân nhánh trên mã nhận dạng không tốt cho việc dự đoán lớp của các thể hiện chư a biết và cho biết không có gì về cấu trúc của quyết định, mà xét cho cùng là hai mục tiêu của học máy.

Hiệu quả tổng thể là thước đo thu được thông tin có xu hướng ưu tiên các thuộc tính với số lượng lớn các giá trị có thể. Để bù đắp cho điều này, một phép đo sửa đổi được gọi là tỷ số khuếch đại được sử dụng rộng rãi. Hệ số tăng là bắt nguồn bằng cách tính đến số lượng và kích thước của các nút con vào mà một thuộc tính phân chia tập dữ liệu, bỏ qua bất kỳ thông tin nào về lớp học. Trong tình huống được minh họa trong Hình 4.5, tất cả các số đếm đều có giá trị là 1, vì vậy giá trị thông tin của sự phân tách là

$$([\text{thông tin } 1,1,\ldots,1])=14\times \frac{1}{14}\log\frac{1}{14}\times,$$

bởi vì cùng một phân số, 1/14, xuất hiện 14 lần. Số tiền này là log 14, hoặc 3,807 bit, đây là một giá trị rất cao. Điều này là do giá trị thông tin của một sự phân chia là số bit cần thiết để xác định nhánh nào trong mỗi phiên bản được gán và càng có nhiều nhánh thì giá trị này càng lớn. Thành quả tỷ lệ được tính bằng cách chia mức tăng thông tin ban đầu, 0,940 trong trường hợp này, theo giá trị thông tin của thuộc tính, 3.807—mang lại giá trị tỷ lệ khuếch đại là 0,247 cho thuộc tính mã ID .

Quay trở lại gốc cây để xem dữ liệu thời tiết trong Hình 4.2, triển vọng chia tách tập dữ liệu thành ba tập hợp con có kích thước 5, 4 và 5 và do đó có giá trị thông tin nội tại là

$$\text{thông tin } 5,4,5([\,])=1\ 577\ .$$

mà không chú ý đến các lớp tham gia vào các tập con. Như những gì chúng ta có đã thấy, giá trị thông tin nội tại này cao hơn đối với hệ thống phân nhánh cao hơn thuộc tính chẳng hạn như mã ID giả định. Một lần nữa, chúng ta có thể sửa độ lợi thông tin bằng cách chia cho giá trị thông tin nội tại để có tỷ lệ độ lợi.

Kết quả tính toán cho các gốc cây trong Hình 4.2 được tổng hợp trong Bảng 4.7. Triển vọng vẫn đứng đầu, nhưng độ ẩm bây giờ là nhiều đối thủ cạnh tranh hơn vì nó chia dữ liệu thành hai tập hợp con thay vì ba. TRONG ví dụ cụ thể này, thuộc tính mã ID giả định , với tỷ lệ khuếch đại là 0,247, vẫn sẽ được ưu tiên hơn bất kỳ trong số bốn giá trị này. Tuy nhiên ưu điểm của nó là

Bảng 4.7 Tính toán tỷ lệ khuếch đại cho các gốc cây trong Hình 4.2.

Quan điểm		Nhiệt độ		độ ẩm		Co gio	
thông	0,693	thông	0,911	thông	0,788	thông	0,892
tin: tăng: 0,940- 0,693	0,247	tin: mức tăng: 0,940-	0,029	tin: tăng: 0,940- 0,788	0,152	tin: tăng: 0,940- 0,892	0,048
chia thông tin: thông tin([5,4,5])	1.577	0,911 thông tin chia: thông tin ([4,6,4])	1.557	chia thông tin: thông tin ([7,7]) tỷ lệ tăng: 0,152/1	1.000	chia thông tin: thông tin([8,6]) tỷ lệ tăng: 0,048/0,985	0,985
tỷ lệ tăng: 0,247/1,577	0,157	tỷ lệ tăng: 0,029/1,557	0,019		0,152		0,049

giảm đáng kể. Trong triển khai thực tế, chúng ta có thể sử dụng một bài kiểm tra đặc biệt để bảo vệ chống lại việc chia tách một thuộc tính vô dụng như vậy.

Thật không may, trong một số trường hợp, việc sửa đổi tỷ lệ khuếch đại bù đắp quá mức và có thể dẫn đến việc ưu tiên một thuộc tính hơn chỉ vì thông tin nội tại của nó thấp hơn nhiều so với thông tin của các thuộc tính khác. Một sửa chữa tiêu chuẩn là chọn

thuộc tính tối đa hóa tỷ lệ khuếch đại, với điều kiện là thông tin thu được vì thuộc tính đó ít nhất cũng lớn bằng mức tăng thông tin trung bình cho tất cả các thuộc tính được kiểm tra.

Cuộc thảo luận

Cách tiếp cận chia để trị đối với quy nạp cây quyết định, đôi khi được gọi là cảm ứng từ trên xuống của cây quyết định, đã được phát triển và hoàn thiện trong nhiều năm của J. Ross Quinlan thuộc Đại học Sydney, Australia. Mặc dù những người khác có làm việc trên các phương pháp tương tự, nghiên cứu của Quinlan luôn đi đầu trong quy nạp cây quyết định. Phương pháp đã được mô tả bằng cách sử dụng tiêu chí thu được thông tin về cơ bản giống như tiêu chí được gọi là ID3. việc sử dụng của tỷ lệ khuếch đại là một trong nhiều cải tiến đã được thực hiện cho ID3 so với vài năm; Quinlan mô tả nó là mạnh mẽ trong nhiều hoàn cảnh khác nhau. Mặc dù là một giải pháp mạnh mẽ và thiết thực, nhưng nó hy sinh một số động lực lý thuyết rõ ràng và thanh lịch của tiêu chí thu được thông tin.

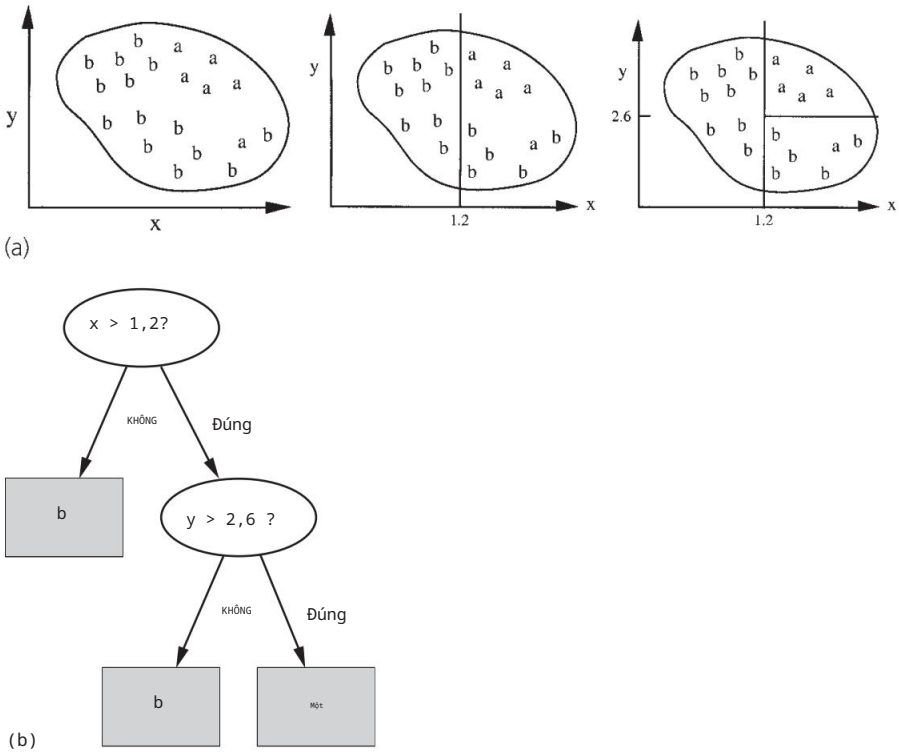
Một loạt các cải tiến đối với ID3 đã đạt đến đỉnh điểm trong thực tế và có ảnh hưởng hệ thống cảm ứng cây quyết định được gọi là C4.5. Những cải tiến này bao gồm các phương pháp xử lý các thuộc tính số, giá trị bị thiếu, dữ liệu nhiễu và tạo luật từ cây, và chúng được mô tả trong Phần 6.1.

4.4 Thuật toán phủ: Xây dựng luật

Như chúng ta đã thấy, thuật toán cây quyết định dựa trên nguyên tắc chia để trị cách tiếp cận bài toán phân lớp. Họ làm việc từ trên xuống, tìm kiếm ở mỗi giai đoạn, một thuộc tính để phân chia trên đó phân tách tốt nhất các lớp; sau đó tái diễn xử lý tích cực các bài toán con phát sinh từ việc tách. Chiến lược này tạo ra một cây quyết định, nếu cần thiết có thể chuyển đổi thành một tập hợp các quy tắc phân loại—mặc dù nếu nó tạo ra các quy tắc hiệu quả thì việc chuyển đổi không phải là không đáng kể.

Một cách tiếp cận khác là chọn lần lượt từng lớp và tìm cách bao quát tất cả các trường hợp trong đó, đồng thời loại trừ các trường hợp không có trong lớp. Đây được gọi là **cách tiếp cận** bao trùm bởi vì ở mỗi giai đoạn, bạn xác định một quy tắc "bao gồm" một số trường hợp. Về bản chất, cách tiếp cận bao trùm này dẫn đến vào một tập hợp các quy tắc chứ không phải là một cây quyết định.

Phương pháp bao phủ có thể dễ dàng được hình dung trong không gian hai chiều của các trường hợp như trong Hình 4.6(a). Trước tiên, chúng tôi thực hiện một quy tắc bao gồm a. Vì



Hình 4.6 Thuật toán phủ: (a) phủ các thể hiện và (b) cây quyết định cho cùng một vấn đề.

bài kiểm tra đầu tiên trong quy tắc, chia không gian theo chiều dọc như trong hình ở giữa. Điều này đưa ra sự khởi đầu của một quy tắc:

Nếu x > 1,2 thì hạng = a

Tuy nhiên, quy tắc bao gồm nhiều b cũng như a, do đó, một thử nghiệm mới được thêm vào quy tắc bằng cách chia thêm không gian theo chiều ngang như thể hiện trong sơ đồ thứ ba:

Nếu x > 1,2 và y > 2,6 thì hạng = a

Điều này đưa ra một quy tắc bao gồm tất cả trừ một trong số a. Có thể để nguyên như vậy là hợp lý, nhưng nếu cảm thấy cần thiết phải bao hàm chữ a cuối cùng, thì cần có một quy tắc khác—có lẽ

Nếu x > 1,4 và y < 2,4 thì hạng = a

Thủ tục tương tự dẫn đến hai quy tắc bao gồm b's:

Nếu x £ 1,2 thì lớp = b

Nếu x > 1,2 và y £ 2,6 thì hạng = b

Một lần nữa, một a bị các quy tắc này che phủ một cách sai lầm. Nếu cần phải loại trừ nó, thì sẽ phải thêm nhiều bài kiểm tra hơ n vào quy tắc thứ hai và các quy tắc bổ sung sẽ cần đư ợc đư a ra để bao gồm các điểm b mà các bài kiểm tra mới này loại trừ.

Quy tắc so với cây

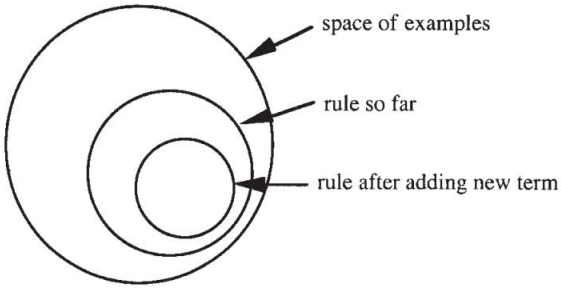
Thuật toán chia để trị từ trên xuống hoạt động trên cùng một dữ liệu theo cách, ít nhất là về mặt bề ngoài, khá giống với thuật toán bao trùm. Trư ớc tiên, nó có thể phân tách tập dữ liệu bằng cách sử dụng thuộc tính x và cuối cùng có thể sẽ phân tách nó ở cùng một vị trí, x = 1,2. Tuy nhiên, trong khi thuật toán bao phủ chỉ liên quan đến việc bao phủ một lớp duy nhất, phép chia sẽ tính đến cả hai lớp, bởi vì thuật toán chia để trị tạo ra một mô tả khái niệm duy nhất áp dụng cho tất cả các lớp. Lần phân tách thứ hai cũng có thể ở cùng một vị trí, y = 2,6, dẫn đến cây quyết định trong Hình 4.6(b). Cây này tư ơ ng ứng chính xác với tập quy tắc và trong trư ờng hợp này không có sự khác biệt về hiệu quả giữa thuật toán bao trùm và thuật toán chia để trị.

Như ng trong nhiều tình huống, có sự khác biệt giữa luật và cây xét về tính rõ ràng của biểu diễn. Ví dụ, khi chúng tôi mô tả vấn đề cây con sao chép trong Phần 3.3, chúng tôi lư u ý rằng các quy tắc có thể đối xứng trong khi các cây phải chọn một thuộc tính để phân tách trư ớc và điều này có thể dẫn đến các cây lớn hơ n nhiều so với một bộ quy tắc tư ơ ng đư ơ ng. Một điểm khác biệt nữa là, trong trư ờng hợp nhiều lớp, việc phân chia cây quyết định sẽ tính đến tất cả các lớp, cố gắng tối đa hóa độ tinh khiết của sự phân chia, trong khi phư ơ ng pháp tạo quy tắc tập trung vào một lớp tại một thời điểm, bất kể điều gì xảy ra với các lớp khác .

Một thuật toán phủ đơ n giản Các

thuật toán phủ hoạt động bằng cách thêm các thử nghiệm vào quy tắc đang đư ợc xây dựng, luôn cố gắng tạo ra một quy tắc với độ chính xác tối đa. Ngư ợc lại, các thuật toán phân chia và chinh phục hoạt động bằng cách thêm các thử nghiệm vào cây đang đư ợc xây dựng, luôn cố gắng tối đa hóa sự phân tách giữa các lớp.
Mỗi trong số này liên quan đến việc tìm kiếm một thuộc tính để phân chia. Như ng tiêu chí cho thuộc tính tốt nhất là khác nhau trong từng trư ờng hợp. Trong khi các thuật toán phân chia và chinh phục chẳng hạn như ID3 chọn một thuộc tính để tối đa hóa mức thu đư ợc thông tin, thì thuật toán che phủ mà chúng tôi sẽ mô tả chọn một cặp thuộc tính-giá trị để tối đa hóa xác suất phân loại mong muốn.

Hình 4.7 đư a ra một bức tranh về tình huống, hiển thị không gian chứa tất cả các thể hiện, một quy tắc đư ợc xây dựng một phần và cùng một quy tắc sau khi một thuật ngữ mới đư ợc thêm vào. Thuật ngữ mới hạn chế phạm vi của quy tắc: ý tư ởng là bao gồm càng nhiều thể hiện của lớp mong muốn càng tốt và loại trừ càng nhiều thể hiện của các lớp khác càng tốt. Giả sử quy tắc mới sẽ bao gồm tổng số t trư ờng hợp, trong đó p là ví dụ tích cực của lớp và t - p thuộc lớp khác.



Hình 4.7 Không gian thể hiện trong quá trình vận hành thuật toán bao phủ.

các lớp—tức là, chúng là lỗi do quy tắc tạo ra. Sau đó chọn thuật ngữ mới để tối đa hóa tỷ lệ p/t.

Một ví dụ sẽ giúp ích. Để thay đổi, chúng tôi sử dụng bài toán kính áp tròng của Bảng 1.1. Chúng tôi sẽ hình thành các quy tắc bao gồm cả ba loại, cứng, mềm và không, lần lượt. Để bắt đầu, chúng tôi tìm kiếm một quy tắc:

Nếu như ? sau đó khuyến nghị = khó

Đối với số hạng chưa biết ?, ta có chín cách chọn:

tuổi = trẻ	2/8
tuổi = tuổi tiền viễn thị =	1/8
đơn thuốc đeo kính	1/8
viễn thị = đơn thuốc đeo kính cận = loạn	3/12
thị siêu thị = không loạn thị = có tỷ lệ sản xuất	12/1
nước mắt = tỷ lệ sản	0/12
xuất nước mắt giảm =	4/12
bình thường	0/12
	4/12

Các con số ở bên phải hiển thị tỷ lệ các phiên bản "chính xác" trong tập hợp nổi bật bởi sự lựa chọn đó. Trong trường hợp này, đúng có nghĩa là khuyến nghị là cứng. Chẳng hạn, age = young chọn tám trường hợp, hai trong số đó khuyến nghị kính áp tròng cứng nên phân số thứ nhất là 2/8. (Để làm theo điều này, bạn sẽ cần phải xem lại dữ liệu kính áp tròng trong Bảng 1.1 trên trang 6 và đếm các mục trong bảng.) Chúng tôi chọn phân số lớn nhất, 4/12, tùy ý chọn giữa lựa chọn thứ bảy và là lựa chọn cuối cùng trong danh sách trước và tạo quy tắc:

Nếu loạn thị = có thì khuyến nghị = khó

Quy tắc này là một quy tắc không chính xác, chỉ có 4 trường hợp đúng trong số 12 mà nó bao gồm, thể hiện trong Bảng 4.8. Vì vậy, chúng tôi tinh chỉnh nó thêm:

Nếu loạn thị = có và ? sau đó khuyến nghị = khó

Bảng 4.8 Một phần của dữ liệu kính áp tròng mà loạn thị = có.

Tuổi	toa kính	loạn thị	Xé mắt Ống kính khuyên dùng tỷ lệ	
cận thị trẻ cận thị trẻ		vâng	giảm bình	không có
hypermetrope trẻ hypermetrope		vâng	thư ờng	cứng
viễn	trẻ cận thị tiền	vâng	giảm bình	không có
thị thị trẻ cận thị tiền viễn thị cận		vâng	thư ờng	cứng
tiền viễn thị Hypermetrope tiền		vâng	giảm bình	không có
viễn thị viễn thị cận thị trẻ		vâng	thư ờng	cứng
lão thị cận thị viễn thị hypermetrope		vâng	giảm bình	không có
viễn thị		vâng	thư ờng	không có
		vâng	giảm bình	không có
		vâng	thư ờng	cứng
			giảm bình	không có
			thư ờng	không có

Xét các khả năng của số hạng chư a biết? mang lại bảy lựa chọn:

tuổi = trẻ	2/4
tuổi = tuổi tiền viễn thị =	1/4
đơ n thuốc đeo kính	1/4
viễn thị = đơ n thuốc đeo kính cận thị =	3/6
tỷ lệ sản xuất nư ớc mắt siêu thị = giảm tỷ lệ sản	1/6
xuất nư ớc mắt = bình thư ờng	0/6
	4/6

(Một lần nữa, đếm các mục trong Bảng 4.8.) Ngư ời cuối cùng là ngư ời chiến thắng rõ ràng, nhận đư ợc bốn các trư ờng hợp đúng trong số sáu trư ờng hợp mà nó đề cập và tư ơ ng ứng với quy tắc

Nếu loạn thị = có và tỷ lệ sản xuất nư ớc mắt = bình thư ờng
sau đó khuyến nghị = khó

Chúng ta có nên dừng lại ở đây không? Có lẽ. Như ng giả sử chúng ta đang tìm kiếm các quy tắc chính xác, không bất kể chúng trở nên phức tạp như thế nào. Bảng 4.9 cho thấy các trư ờng hợp đư ợc bao phủ bởi quy tắc cho đến nay. Các khả năng cho nhiệm kỳ tiếp theo bây giờ là

tuổi = trẻ	2/2
tuổi = tuổi tiền lão thị =	1/2
toa kính viễn thị =	1/2
toa kính cận thị = hypermetrope	3/3
	1/3

Chúng ta cần chọn giữa thứ nhất và thứ tư . Cho đến giờ chúng ta đã xử lý các phân số bằng số, như ng mặc dù hai phân số này bằng nhau (cả hai đều ư ớc lư ợng bằng 1), như ng chúng có phạm vi bảo hiểm khác nhau: một chọn chỉ hai trư ờng hợp chính xác và trư ờng hợp còn lại

Bảng 4.9 Một phần của dữ liệu kính áp tròng mà loạn thị = có và tỷ lệ sản xuất rách = bình thường.

Tuổi	toa kính	loạn thị	Sản xuất nước mắt tỷ lệ	thấu kính
cận thị trẻ tuổi lão thị		Có	bình thường	cứng
tuổi	cận thị trẻ	có	bình thường	cứng
lão thị hypermetrope trẻ		có	bình thường	cứng
lão thị cận thị lão thị hypermetrope		có	bình thường	không có
		có	bình thường	cứng
		có	bình thường	không có

chọn ba. Trong trường hợp hòa, chúng tôi chọn quy tắc có phạm vi bảo hiểm lớn hơn, đưa ra quy tắc cuối cùng:

Nếu loạn thị = có và tỷ lệ sản xuất nước mắt = bình thường
và toa kính = myope sau đó khuyến nghị = khó

Đây thực sự là một trong những quy tắc được đưa ra cho vấn đề về kính áp tròng. Nhưng nó chỉ bao gồm ba trong số bốn khuyến nghị cứng . Vì vậy, chúng tôi xóa ba cái này khỏi tập hợp các thể hiện và bắt đầu lại, tìm kiếm một quy tắc khác có dạng:

Nếu như ? sau đó khuyến nghị = khó

Theo quy trình tương tự, cuối cùng chúng ta sẽ thấy rằng tuổi = trẻ là tốt nhất lựa chọn cho nhiệm kỳ đầu tiên. Phạm vi bảo hiểm của nó là bảy; lý do cho số bảy là 3 các phiên bản đã bị xóa khỏi bộ ban đầu, để lại 21 phiên bản gộp lại. Sự lựa chọn tốt nhất cho thuật ngữ thứ hai là loạn thị = đúng, chọn 1/3 (thực ra là huề); tốc độ sản xuất nước mắt = bình thường là tốt nhất cho thứ ba, chọn 1/1.

Nếu tuổi = trẻ và loạn thị = có và
tỷ lệ sản xuất nước mắt = bình thường sau đó khuyến nghị = khó

Quy tắc này thực sự bao gồm ba trong số các trường hợp ban đầu, hai trong số đó là được bảo vệ bởi quy tắc trước đó — nhưng điều đó không sao cả vì khuyến nghị là giống nhau cho mỗi quy tắc.

Bây giờ tất cả các hộp đựng ống kính cứng đã được bảo hiểm, bước tiếp theo là tiến hành những cái ống kính mềm theo cách tương tự. Cuối cùng, các quy tắc được tạo cho không có trường hợp—trừ khi chúng tôi đang tìm kiếm một bộ quy tắc có quy tắc mặc định, trong trường hợp đó rõ ràng quy tắc cho kết quả cuối cùng là không cần thiết.

Những gì chúng tôi vừa mô tả là phương pháp PRISM để xây dựng các quy tắc. Nó chỉ tạo ra các quy tắc đúng hoặc "hoàn hảo". Nó đo lường sự thành công của một quy tắc bởi công thức độ chính xác p/t. Bất kỳ quy tắc nào có độ chính xác dưới 100% là "không chính xác" trong

rằng nó chỉ định các trư ờng hợp cho lớp đư ợc đề cập mà thực tế không có lớp đó.
PRISM tiếp tục thêm các mệnh đề vào mỗi quy tắc cho đến khi nó hoàn hảo: độ chính xác của nó là
100%. Hình 4.8 tóm tắt thuật toán. Vòng lặp bên ngoài lặp đi lặp lại
lần lư ợt tạo ra các quy tắc cho từng lớp. Lư u ý rằng chúng tôi khởi tạo lại để
tập hợp đầy đủ các ví dụ mỗi lần. Sau đó, chúng tôi tạo quy tắc cho lớp đó và
loại bỏ các ví dụ khỏi tập hợp cho đến khi không còn mẫu nào trong lớp đó. Bất cứ khi nào chúng tôi
tạo quy tắc, hãy bắt đầu với quy tắc trống (bao gồm tất cả các ví dụ),
và sau đó hạn chế nó bằng cách thêm các bài kiểm tra cho đến khi nó chỉ bao gồm các ví dụ về mong muốn
lớp học. Ở mỗi giai đoạn, hãy chọn phép thử hứa hẹn nhất, nghĩa là phép thử tối đa hóa độ chính xác
của quy tắc. Cuối cùng, phá vỡ ràng buộc bằng cách chọn bài kiểm tra có độ bao phủ ư ớc tính lớn nhất.

Quy tắc so với danh sách quyết định

Xem xét các quy tắc đư ợc tạo ra cho một lớp cụ thể, đó là thuật toán trong Hình
4.8 đã loại bỏ vòng lặp bên ngoài. Có vẻ rõ ràng từ cách mà các quy tắc này
đư ợc tạo ra để chúng đư ợc diễn giải theo thứ tự, nghĩa là, như một danh sách quyết định, kiểm tra
lần lư ợt các quy tắc cho đến khi một quy tắc đư ợc áp dụng và sau đó sử dụng quy tắc đó. Đây là
bởi vì các phiên bản đư ợc bao phủ bởi một quy tắc mới bị xóa khỏi tập hợp phiên bản
ngay sau khi quy tắc đư ợc hoàn thành (trong dòng thứ ba từ cuối mã trong
Hình 4.8): do đó, các quy tắc tiếp theo đư ợc thiết kế cho các trư ờng hợp không đư ợc đề cập
theo quy định. Tuy nhiên, mặc dù có vẻ như chúng ta phải kiểm tra các quy tắc
đổi lại, chúng ta không phải làm như vậy. Hãy xem xét rằng mọi quy tắc tiếp theo đư ợc tạo
đối với lớp này sẽ có tác dụng giống nhau—tất cả chúng đều dự đoán cùng một lớp. Cái này
có nghĩa là chúng đư ợc thực hiện theo thứ tự nào không quan trọng: một trong hai quy tắc sẽ

```
Đối với mỗi lớp C
    Khởi tạo E cho bộ thể hiện
    Trong khi E chứa các thể hiện trong lớp C
        Tạo quy tắc R với phía bên trái trống dự đoán lớp C
        Cho đến khi R hoàn hảo (hoặc không còn thuộc tính nào để sử dụng), hãy làm
            Đối với mỗi thuộc tính A không đư ợc đề cập trong R và mỗi giá trị v,
                Cân nhắc thêm điều kiện A=v vào LHS của R
                Chọn A và v để tối đa độ chính xác p/t
                    (phá vỡ ràng buộc bằng cách chọn điều kiện có p lớn nhất)
            Thêm A=v vào R
        Xóa các trư ờng hợp đư ợc bao phủ bởi R khỏi E
```

Hình 4.8 Mã giả cho ngư ời học quy tắc cơ bản.

được tìm thấy bao gồm trường hợp này, trong trường hợp đó, lớp được đề cập được dự đoán hoặc không tìm thấy quy tắc nào như vậy, trong trường hợp đó, lớp không được dự đoán.

Bây giờ trở lại thuật toán tổng thể. Mỗi lớp được xem xét lần lượt, và các quy tắc được tạo để phân biệt các thể hiện trong lớp đó với các thể hiện khác. KHÔNG thứ tự được ngụ ý giữa các quy tắc cho một lớp và các quy tắc cho lớp khác. Do đó, các quy tắc được tạo ra có thể được thực hiện độc lập với thứ tự.

Như được mô tả trong Phần 3.3, các quy tắc độc lập với thứ tự dường như cung cấp nhiều hơn tính mô-đun bằng cách mỗi người đóng vai trò là những "kiến thức" độc lập, nhưng chúng phải chịu bất lợi là không rõ ràng phải làm gì khi xung đột quy tắc áp dụng. Với các quy tắc được tạo theo cách này, một ví dụ thử nghiệm có thể nhận được nhiều phân loại, nghĩa là các quy tắc áp dụng cho các lớp khác nhau có thể chấp nhận nó. Khác các ví dụ thử nghiệm có thể không nhận được sự phân loại nào cả. Một chiến lược đơn giản để buộc một quyết định trong những trường hợp mơ hồ này là lựa chọn, từ các phân loại được được dự đoán, loại có nhiều ví dụ huấn luyện nhất hoặc, nếu không có phân loại nào được dự đoán, để chọn loại có nhiều ví dụ huấn luyện nhất về tổng thể. Những cái này khó khăn không xảy ra với danh sách quyết định bởi vì chúng được diễn giải theo thứ tự và việc thực thi dừng ngay khi một quy tắc được áp dụng: việc thêm một quy tắc mặc định ở cuối đảm bảo rằng mọi phiên bản thử nghiệm đều nhận được phân loại. Có thể tạo danh sách quyết định tốt cho trường hợp nhiều lớp bằng cách sử dụng một chút phương pháp khác, như chúng ta sẽ thấy trong Phần 6.2.

Các phương pháp như PRISM có thể được mô tả như **các thuật toán** tách biệt và chinh phục : bạn xác định một quy tắc bao gồm nhiều trường hợp trong lớp (và loại trừ những trường hợp không có trong lớp), hãy tách riêng các trường hợp được bảo hiểm vì chúng đã được được xử lý theo quy tắc và tiếp tục quá trình trên những phần còn lại. Cái này tương phản độc đáo với cách tiếp cận chia để trị của cây quyết định. Các bước riêng biệt làm tăng đáng kể hiệu quả của phương pháp vì thể hiện thiết lập liên tục co lại khi hoạt động tiến hành.

4.5 Quy tắc hiệp hội khai thác

Quy tắc kết hợp giống như quy tắc phân loại. Bạn có thể tìm thấy chúng trong cùng một cách, bằng cách thực hiện thủ tục quy nạp quy tắc chia để trị cho từng biểu thức khả dĩ có thể xảy ra ở vế phải của quy tắc. Nhưng không chỉ bất kỳ thuộc tính nào có thể xảy ra ở phía bên tay phải với bất kỳ giá trị có thể nào; Một quy tắc kết hợp duy nhất thường dự đoán giá trị của nhiều thuộc tính. ĐẾN tìm thấy các quy tắc như vậy, bạn sẽ phải thực hiện thủ tục quy nạp quy tắc một lần cho mọi sự kết hợp **có thể có** của các thuộc tính, với mọi sự kết hợp có thể có của các giá trị, ở phía bên tay phải. Điều đó sẽ dẫn đến một số lượng lớn các luật kết hợp, mà sau đó sẽ phải được cắt bớt trên cơ sở phạm vi bảo hiểm của họ (số trường hợp mà họ dự đoán chính xác) và

độ chính xác (cùng một số đư ợc biểu thị bằng tỷ lệ của số trư ờng hợp áp dụng quy tắc). Cách tiếp cận này là khá không khả thi. (Lư u ý rằng, như chúng tôi đã đề cập trong Phần 3.4, cái mà chúng tôi gọi là phạm vi bảo hiểm thư ờng đư ợc gọi là hỗ trợ và cái mà chúng tôi gọi là độ chính xác thư ờng đư ợc gọi là độ tin cậy.)

Thay vào đó, chúng tôi tận dụng thực tế là chúng tôi chỉ quan tâm đến các luật kết hợp có độ bao phủ cao. Hiện tại, chúng tôi bỏ qua sự khác biệt giữa bên trái và bên phải của một quy tắc và tìm kiếm sự kết hợp của các cặp thuộc tính-giá trị có phạm vi bao phủ tối thiểu đư ợc chỉ định trư ớc. Chúng đư ợc gọi là các bộ vật phẩm: một cặp thuộc tính-giá trị là một vật phẩm. Thuật ngữ này bắt nguồn từ phân tích giỏ thị trư ờng, trong đó các mặt hàng là các mặt hàng trong giỏ hàng của bạn và ngư ời quản lý siêu thị đang tìm kiếm mối liên hệ giữa các giao dịch mua này.

Bộ sản phẩm

Cột đầu tiên của Bảng 4.10 hiển thị các mục riêng lẻ cho dữ liệu thời tiết của Bảng 1.2, với số lần mỗi mục xuất hiện trong tập dữ liệu đư ợc đư a ra ở bên phải. Đây là những bộ một mục. Bư ớc tiếp theo là tạo hai tập hợp mục bằng cách tạo các cặp tập hợp một mục. Tất nhiên, không ích gì khi tạo một tập hợp chứa hai giá trị khác nhau của cùng một thuộc tính (chẳng hạn như triển vọng = nắng và triển vọng = u ám), bởi vì điều đó không thể xảy ra trong bất kỳ thực tế nào.
ví dụ.

Giả sử rằng chúng ta tìm kiếm các luật kết hợp có độ bao phủ tối thiểu là 2: do đó chúng ta loại bỏ bất kỳ tập mục nào bao gồm ít hơ n hai trư ờng hợp. Điều này để lại 47 hai bộ vật phẩm, một số trong số đó đư ợc hiển thị trong cột thứ hai cùng với số lần chúng xuất hiện. Bư ớc tiếp theo là tạo bộ ba mục, trong đó có 39 mục có phạm vi bao phủ từ 2 trở lên. Có 6 tập hợp bốn mục và không có tập hợp năm mục—đối với dữ liệu này, tập hợp năm mục có phạm vi bao phủ 2 hoặc cao hơ n chỉ có thể tư ơ ng ứng với một phiên bản lặp lại. Ví dụ, hàng đầu tiên của bảng cho thấy rằng có năm ngày khi triển vọng = nắng, hai trong số đó có nhiệt độ = ôn hòa, và trên thực tế, cả hai ngày đó độ ẩm = cao và vui chơ i = không có gì tốt .

quy tắc hiệp hội

Chúng tôi sẽ giải thích ngay cách tạo các bộ vật phẩm này một cách hiệu quả. Như ng trư ớc tiên chúng ta hãy kết thúc câu chuyện. Khi tất cả các bộ mục có phạm vi bao phủ bắt buộc đã đư ợc tạo, bư ớc tiếp theo là biến từng bộ thành quy tắc hoặc bộ quy tắc với ít nhất độ chính xác tối thiểu đư ợc chỉ định. Một số bộ vật phẩm sẽ tạo ra nhiều quy tắc; những ngư ời khác sẽ không sản xuất. Ví dụ: có một bộ ba mục có độ bao phủ là 4 (hàng 38 của Bảng 4.10):

độ ẩm = bình thư ờng, gió = sai, chơ i = có

Bộ này dẫn đến bảy quy tắc tiềm năng:

Bảng 4.10 Bộ mục cho dữ liệu thời tiết có phạm vi phủ sóng 2 hoặc cao hơn.

	Bộ một món	Bộ hai món	Bộ ba món	Bộ bốn món
1	triển vọng = nắng (5)	triển vọng = triển vọng nắng = nhiệt độ nắng = nhẹ (2) nhiệt độ = độ ẩm nóng = cao (2)		triển vọng = nắng nhiệt độ = nóng độ ẩm = cao chơi = không (2)
2	triển vọng = u ám (4)	triển vọng = nắng nhiệt độ = nóng (2)	triển vọng = nhiệt độ nắng = chơi nóng = không (2)	triển vọng = nắng độ ẩm = cao gió = sai chơi = không (2)
3	triển vọng = mưa (5)	triển vọng = độ ẩm nắng = bình thường (2)	triển vọng = nắng ẩm = chơi bình thường = có (2)	triển vọng = u ám nhiệt độ = nóng gió = sai chơi = có (2)
4	nhiệt độ = mát mẻ (4)	triển vọng = nắng độ ẩm = cao (3)	triển vọng = độ ẩm nắng = gió cao = sai (2)	triển vọng = mưa nhiệt độ = nhẹ gió = sai chơi = có (2)
5	nhiệt độ = ôn hòa (6)	triển vọng = nắng gió = đúng (2)	triển vọng = nắng ẩm = chơi cao = không (3)	triển vọng = mưa độ ẩm = bình thường gió = sai chơi = có (2)
6	nhiệt độ = nóng (4)	triển vọng = nắng gió = sai (3)	triển vọng = nắng gió = giả vờ = không (2)	nhiệt độ = mát mẻ độ ẩm = bình thường gió = sai chơi = có (2)
7	độ ẩm = bình thường (7)	triển vọng = chơi nắng = có (2)	triển vọng = u ám nhiệt độ = nóng gió = sai (2)	
8	độ ẩm = cao (7)	triển vọng = chơi nắng = không (3)	triển vọng = u ám nhiệt độ = nóng chơi = có (2)	
9	gió = đúng (6)	triển vọng = nhiệt độ u ám = nóng (2)	triển vọng = u ám độ ẩm = bình thường chơi = có (2)	
10	gió = sai (8)	triển vọng = độ ẩm u ám = bình thường (2)	triển vọng = u ám độ ẩm = cao chơi = có (2)	
11	chơi = có (9)	triển vọng = độ ẩm u ám = cao (2)	triển vọng = u ám gió = đúng chơi = có (2)	
12	chơi = không (5)	triển vọng = u ám có gió = đúng (2)	triển vọng = u ám gió = sai chơi = có (2)	
13		triển vọng = u ám có gió = sai (2)	triển vọng = mưa nhiệt độ = mát mẻ độ ẩm = bình thường (2)	

Bảng 4.10 (còn tiếp)

	Bộ một món	Bộ hai món	Bộ ba món	Bộ bốn món
...		...	...	
38		độ ẩm = gió bình thư ờng = sai (4)	độ ẩm = bình thư ờng gió = sai chơ i = có (4)	
39		độ ẩm = chơ i bình thư ờng = có (6)	độ ẩm = cao gió = sai chơ i = không (2)	
40		độ ẩm = cao gió = đúng (3)		
...		...		
47		gió = sai chơ i = không (2)		

Nếu độ ẩm = bình thư ờng và gió = sai thì chơ i = có 4/4
Nếu độ ẩm = bình thư ờng và chơ i = có thì có gió = sai 4/6
Nếu có gió = sai và phát = có thì độ ẩm = bình thư ờng 4/6
Nếu độ ẩm = bình thư ờng thì gió = sai và phát = có 4/7
Nếu có gió = sai thì độ ẩm = bình thư ờng và chơ i = có 4/8
Nếu phát = có thì độ ẩm = bình thư ờng và có gió = sai 9/4
Nếu – thì độ ẩm = bình thư ờng và có gió = sai và phát = đúng 4/12

Các số liệu ở bên phải hiển thị số lư ợng phiên bản mà cả ba điều kiện đều đúng—nghĩa là phạm vi bảo hiểm—chia cho số lư ợng phiên bản cho mà các điều kiện trong tiền đề là đúng. Giải thích như là một phân số, họ đại diện cho tỷ lệ các trư ờng hợp mà quy tắc là chính xác, nghĩa là sự chính xác. Giả sử rằng độ chính xác đư ợc chỉ định tối thiểu là 100%, chỉ lần đầu tiên trong số các quy tắc này sẽ biến nó thành bộ quy tắc cuối cùng. Mẫu số của các phân số có thể dễ dàng thu đư ợc bằng cách tra cứu biểu thức tiền đề trong Bảng 4.10 (mặc dù một số không đư ợc hiển thị trong Bảng). Quy tắc cuối cùng ở trên không có điều kiện trong tiền đề và mẫu số của nó là tổng số trư ờng hợp trong bộ dữ liệu.

Bảng 4.11 cho thấy quy tắc cuối cùng đư ợc thiết lập cho dữ liệu thời tiết, với độ phủ tối thiểu 2 và độ chính xác tối thiểu 100%, đư ợc sắp xếp theo độ phủ. Có 58 quy tắc, 3 với vùng phủ sóng 4, 5 với vùng phủ sóng 3 và 50 với vùng phủ sóng 2. Chỉ 7 có hai điều kiện trong hệ quả, và không có điều kiện nào có nhiều hơ n hai. Quy tắc đầu tiên đến từ bộ mục đư ợc mô tả trư ớc đó. Đôi khi một số quy tắc phát sinh từ cùng một bộ mục. Ví dụ: quy tắc 9, 10 và 11 đều phát sinh từ bộ bốn mục ở dòng 6 của Bảng 4.10:

nhiệt độ = mát mẻ, độ ẩm = bình thư ờng, gió = sai, chơ i = có

Bảng 4.11 Quy tắc kết hợp cho dữ liệu thời tiết.

luật kết hợp			Độ chính xác của vùng phủ sóng
1 độ ẩm = gió bình thường = sai 2	chơi fi = có	4	100%
nhiệt độ = mát mẻ 3	fi ẩm = chơi fi bình	4	100%
triển vọng = u ám 4	thường = có	4	100%
nhiệt độ = mát mẻ = có 5 triển	fi ẩm = chơi fi bình	3	100%
vọng = mưa gió = sai 6 triển	thường = có	3	100%
vọng = phát mưa = có 7 triển	fi có gió = chơi	3	100%
vọng = độ ẩm nắng = cao 8 triển	fi sai =	3	100%
vọng = nắng phát = không 9	không có độ ẩm fi	3	100%
nhiệt độ = gió mát = sai	= độ ẩm fi cao = chơi	2	100%
	bình thường = có		
10 nhiệt độ = độ ẩm mát = chơi fi có gió bình thường = có = sai		2	100%
11 nhiệt độ = gió mát = phát sai = có độ ẩm fi = bình thường 12 triển			100%
vọng = độ ẩm mưa = gió bình thường phát fi = có = sai		2 2	100%
13 triển vọng = độ ẩm mưa = chơi bình thường = có fi có gió = sai			100%
14 triển vọng = nhiệt độ mưa = có gió nhẹ fi chơi = có = sai		2 2	100%
15 triển vọng = nhiệt độ mưa = phát nhẹ = có gió mạnh = sai 16 nhiệt		2	100%
độ = gió nhẹ = phát sai = có triển vọng sai = mưa 17 triển vọng = nhiệt		2	100%
độ u ám = nóng có gió = phát sai = có		2	100%
18 triển vọng = u ám có gió = sai	nhiệt độ fi = chơi	2	100%
	nóng = có		
19 nhiệt độ = chơi nóng = có	fi triển vọng = u ám	2	100%
	có gió = sai		
20 triển vọng = nhiệt độ u ám = gió nóng = sai	chơi fi = có	2	100%
21 triển vọng = nhiệt độ u ám = chơi nóng = có	fi gió = sai	2	100%
22 triển vọng = u ám gió = chơi sai = có 23	nhiệt độ fi = triển	2	100%
nhiệt độ = gió nóng = chơi sai = có 24 gió =	vọng fi nóng = triển	2	100%
chơi sai = không	vọng fi u ám = độ	2	100%
	ẩm nắng = cao		
25 triển vọng = độ ẩm nắng = gió cao = phát sai fi = không triển		2	100%
vọng 26 = nắng gió = phát sai = độ ẩm không fi = cao 27 độ ẩm = gió lớn		2	100%
= phát sai = triển vọng không fi = nắng 28 triển vọng = nhiệt độ nắng =		2	100%
nóng fi độ ẩm = chơi cao = không		2	100%
29 nhiệt độ = phát nóng = không	fi triển vọng = độ	2	100%
	ẩm nắng = cao		
30 triển vọng = nhiệt độ nắng = độ ẩm nóng = cao	chơi fi = không	2	100%
31 triển vọng = nhiệt độ nắng = chơi nóng = không có độ ẩm = cao		2	100%
... ...		...	...
58 triển vọng = nhiệt độ nắng = nóng	độ ẩm cao = cao	2	100%

có phạm vi bảo hiểm 2. Ba tập hợp con của tập mục này cũng có phạm vi bảo hiểm 2:

nhiệt độ = mát, gió = sai nhiệt độ = mát, độ
ẩm = bình thư ờng, gió = sai nhiệt độ = mát, gió = sai, chơ i = đúng

và những quy tắc này dẫn đến quy tắc 9, 10 và 11, tất cả đều chính xác 100% (trên dữ liệu huấn luyện).

Tạo luật kết hợp một cách hiệu

quả Bây giờ chúng ta xem xét chi tiết hơ n một thuật toán để tạo luật kết hợp với độ bao phủ và độ chính xác tối thiểu đư ợc chỉ định. Có hai giai đoạn: tạo ra các tập mục với phạm vi bao phủ tối thiểu đã chỉ định và từ mỗi tập mục xác định các quy tắc có độ chính xác tối thiểu đã chỉ định.

Giai đoạn đầu tiên tiến hành bằng cách tạo tất cả các tập hợp một mục với phạm vi tối thiểu nhất định (cột đầu tiên của Bảng 4.10) và sau đó sử dụng điều này để tạo tập hợp hai mục (cột thứ hai), tập hợp ba mục (cột thứ ba), và như thế. Mỗi thao tác liên quan đến việc chuyển qua tập dữ liệu để đếm các mục trong mỗi bộ và sau khi chuyển, các tập mục còn lại đư ợc lư u trữ trong bảng băm— một cấu trúc dữ liệu tiêu chuẩn cho phép tìm thấy các phần tử đư ợc lư u trữ trong đó rất nhanh. Từ các tập hợp một mục, các tập hợp hai mục ứng cử viên đư ợc tạo và sau đó chuyển qua tập dữ liệu, đếm phạm vi bao phủ của từng tập hợp hai mục; cuối cùng, các bộ ứng cử viên có phạm vi bao phủ nhỏ hơ n mức tối thiểu sẽ bị xóa khỏi bảng. Các tập hợp hai mục ứng viên chỉ đơ n giản là tất cả các tập hợp một mục đư ợc lấy theo cặp, bởi vì một tập hợp hai mục không thể có mức bao phủ tối thiểu trừ khi cả hai tập hợp một mục cấu thành của nó cũng có mức bao phủ tối thiểu. Điều này áp dụng chung: một bộ ba mục chỉ có thể có phạm vi bao phủ tối thiểu nếu cả ba trong số các tập hợp con hai mục của nó cũng có phạm vi bao phủ tối thiểu và tư ơ ng tự đối với các tập bốn mục.

Một ví dụ sẽ giúp giải thích cách các bộ mục ứng viên đư ợc tạo ra.
Giả sử có năm bộ ba mục—(ABC), (ABD), (ACD), (ACE) và (BCD)—ví dụ: trong đó A là một đối tư ợng như triển vọng = nắng . Hợp của hai phần tử đầu tiên, (ABCD), là một tập bốn phần tử ứng viên vì ba tập con phần tử khác của nó (ACD) và (BCD) có phạm vi bao phủ lớn hơ n mức tối thiểu. Nếu các bộ ba phần tử đư ợc sắp xếp theo thứ tự từ vựng, như chúng có trong danh sách này, thì chúng ta chỉ cần xem xét các cặp có hai phần tử đầu tiên giống nhau. Ví dụ: chúng tôi không xem xét (ACD) và (BCD) vì (ABCD) cũng có thể đư ợc tạo ra từ (ABC) và (ABD), và nếu hai cái này không phải là bộ ba phần tử ứng cử viên thì (ABCD) không thể là ứng cử viên bộ bốn món. Điều này để lại các cặp (ABC) và (ABD), mà chúng ta đã giải thích, và (ACD) và (ACE).

Cặp thứ hai này dẫn đến tập hợp (ACDE) có các tập hợp con ba phần tử không có phạm vi bao phủ tối thiểu, vì vậy nó bị loại bỏ. Bảng băm hỗ trợ việc kiểm tra này: chúng ta chỉ cần loại bỏ lần lư ợt từng mục khỏi tập hợp và kiểm tra xem

bộ ba mục còn lại thực sự có mặt trong bảng băm. Vì vậy, trong ví dụ này chỉ có một bộ bốn mục ứng viên, (ABCD). Nó có thực sự có phạm vi bảo hiểm tối thiểu hay không chỉ có thể được xác định bằng cách kiểm tra các phiên bản trong tập dữ liệu.

Giai đoạn thứ hai của quy trình lấy từng tập hợp mục và tạo quy tắc từ đó, kiểm tra xem chúng có độ chính xác tối thiểu được chỉ định hay không. Nếu chỉ tìm kiếm các quy tắc với một thử nghiệm duy nhất ở phía bên tay phải, thì vấn đề đơn giản là xem xét từng điều kiện lần lượt là hệ quả của quy tắc, xóa nó khỏi tập mục và chia phạm vi bao phủ của toàn bộ mục được thiết lập bởi độ bao phủ của tập hợp con kết quả—thu được từ bảng băm—để mang lại độ chính xác của quy tắc tương ứng. Do chúng ta cũng quan tâm đến các luật kết hợp với nhiều phép thử trong hệ quả, nên có vẻ như chúng ta phải đánh giá tác động của việc đặt từng tập con của tập mục ở vế phải, để phần còn lại của tập làm tiền đề.

Phương pháp brute-force này sẽ tốn nhiều công sức tính toán trừ khi các tập mục nhỏ, bởi vì số lượng các tập con có thể tăng theo cấp số nhân với kích thước của tập mục. Tuy nhiên, có một cách tốt hơn. Chúng ta đã quan sát thấy khi mô tả các luật kết hợp trong Phần 3.4 rằng nếu luật hệ quả kép

Nếu gió = sai và chơi = không thì triển vọng = nắng
và độ ẩm = cao

giữ với phạm vi và độ chính xác tối thiểu nhất định, thì cả hai quy tắc hệ quả duy nhất được hình thành từ cùng một bộ mục cũng phải giữ:

Nếu độ ẩm = cao và gió = sai và chơi = không thì triển vọng = nắng

Nếu triển vọng = nắng và gió = giả và chơi = không thì độ ẩm = cao

Ngược lại, nếu một trong các quy tắc hệ quả đơn này hoặc quy tắc kia không đúng, thì không có ích gì khi xem xét quy tắc hệ quả kép. Điều này đưa ra một cách xây dựng từ các quy tắc hệ quả đơn thành các quy tắc hệ quả kép ứng viên, từ các quy tắc hệ quả kép thành các quy tắc hệ quả ba ứng cử viên, v.v. Tất nhiên, mỗi quy tắc ứng cử viên phải được kiểm tra đối với bảng băm để xem liệu nó có thực sự vượt quá độ chính xác tối thiểu được chỉ định hay không. Nhưng điều này thường liên quan đến việc kiểm tra ít quy tắc hơn nhiều so với phương pháp brute force. Điều thú vị là cách xây dựng các quy tắc ứng viên (n + 1)-hệ quả từ n quy tắc hệ quả thực tế này thực sự giống như cách xây dựng các tập ứng viên (n + 1)-phần tử từ các tập n phần tử thực tế, đã mô tả trước đó.

Cuộc thảo luận

Các luật kết hợp thường được tìm kiếm cho các tập dữ liệu rất lớn và các thuật toán thuật toán hiệu quả được đánh giá cao. Phương pháp được mô tả trước đây thực hiện một lượt

thông qua tập dữ liệu cho từng kích thư ớc khác nhau của tập hợp mặt hàng. Đôi khi tập dữ liệu là quá lớn để đọc vào bộ nhớ chính và phải đư ợc lư u trên đĩa; sau đó nó có thể là đáng để giảm số lần vư ợt qua bằng cách kiểm tra các bộ mục của hai liên tiếp kích thư ớc trong một lần. Ví dụ: một khi các tập hợp có hai mục đã đư ợc tạo, tất cả bộ ba mục có thể đư ợc tạo từ chúng trư ớc khi đi qua tập hợp để đếm số lư ợng mục thực tế trong tập hợp. Thêm ba mục bộ hơ n mức cần thiết sẽ đư ợc xem xét, như ng số lần đi qua toàn bộ tập dữ liệu sẽ bị giảm.

Trong thực tế, số lư ợng tính toán cần thiết để tạo ra luật kết hợp phụ thuộc rất nhiều vào phạm vi bảo hiểm tối thiểu đư ợc chỉ định. Độ chính xác kém hơ n ảnh hư ởng bởi vì nó không ảnh hư ởng đến số lư ợng đư ờng chuyền mà chúng ta phải thực hiện thông qua tập dữ liệu. Trong nhiều tình huống, chúng ta sẽ muốn có đư ợc một số quy tắc nhất định—ví dụ 50—với phạm vi bao phủ lớn nhất có thể tại một quy tắc đư ợc chỉ định trư ớc. mức độ chính xác tối thiểu. Một cách để làm điều này là bắt đầu bằng cách chỉ định mức độ che phủ khá cao và sau đó giảm dần nó, thực hiện lại toàn bộ thuật toán tìm quy tắc cho từng giá trị bao phủ và lặp lại điều này cho đến khi số quy tắc mong muốn đã đư ợc tạo.

Định dạng đầu vào dạng bảng mà chúng tôi sử dụng xuyên suốt cuốn sách này, và đặc biệt là tệp ARFF chuẩn dựa trên nó, rất kém hiệu quả đối với nhiều quy tắc kết hợp. các vấn đề. Các luật kết hợp thư ờng đư ợc sử dụng khi các thuộc tính là nhị phân—hoặc hiện diện hoặc vắng mặt—và hầu hết các giá trị thuộc tính đư ợc liên kết với một ví dụ vắng mặt. Đây là trư ờng hợp biểu diễn dữ liệu thư a thớt đư ợc mô tả trong Mục 2.4; áp dụng thuật toán tìm luật kết hợp tư ơ ng tự.

4.6 Mô hình tuyến tính

Các phư ơ ng pháp chúng tôi đã xem xét cho cây quyết định và quy tắc hoạt động hiệu quả nhất tự nhiên với các thuộc tính danh nghĩa. Chúng có thể đư ợc mở rộng cho các thuộc tính số bằng cách kết hợp các bài kiểm tra giá trị số trực tiếp vào cây quyết định hoặc quy tắc sơ đồ quy nạp, hoặc bằng cách phân biệt trư ớc các thuộc tính số thành các thuộc tính danh nghĩa. Chúng ta sẽ thấy cách làm tư ơ ng ứng trong Chư ơ ng 6 và 7. Tuy nhiên, có những phư ơ ng pháp hoạt động tự nhiên nhất với các thuộc tính số. Chúng tôi nhìn vào những cái đơ n giản ở đây, những cái tạo thành các thành phần của các phư ơ ng pháp học tập phức tạp hơ n, mà chúng ta sẽ kiểm tra sau.

Dự đoán số: Hồi quy tuyến tính

Khi kết quả hoặc lớp là số và tất cả các thuộc tính là số, tuyến tính hồi quy là một kỹ thuật tự nhiên để xem xét. Đây là một phư ơ ng pháp chủ yếu trong thống kê. Ý tư ởng là thể hiện lớp như một sự kết hợp tuyến tính của các thuộc tính, với các trọng số đư ợc xác định trư ớc:

11=2+2+ ++ xw wa wa wa kk 0 .

trong đó x là lớp; a1, a2, . . ., ak là các giá trị thuộc tính; và w0, w1, . . ., tuần là tạ.

Các trọng số được tính toán từ dữ liệu huấn luyện. Ở đây ký hiệu nhận được một hơi nặng nề, bởi vì chúng ta cần một cách thể hiện các giá trị thuộc tính cho mỗi dụ huấn luyện. Phiên bản đầu tiên sẽ có một lớp, giả sử x(1) và các giá trị thuộc tính a1(1), a2(1), . . ., ak (1), trong đó ký tự trên biểu thị rằng đó là ví dụ đầu tiên. Hơn nữa, thuận tiện về mặt ký hiệu khi giả sử một thuộc tính bổ sung a0 có giá trị luôn luôn là 1.

Giá trị dự đoán cho lớp của trường hợp đầu tiên có thể được viết là

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

Đây là giá trị được dự đoán, không phải thực tế, cho lớp của cá thể đầu tiên. Điều đáng quan tâm là sự khác biệt giữa giá trị dự đoán và giá trị thực tế. phương pháp của hồi quy tuyến tính là chọn các hệ số wj—có k + 1 trong số chúng—để giảm thiểu tổng bình phương của những khác biệt này trong tất cả các khóa đào tạo trường hợp. Giả sử có n trường hợp đào tạo; biểu thị cái thứ i bằng một chỉ số trên (i). Sau đó, tổng bình phương của sự khác biệt là

$$\sum_{i=1}^{n} \left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)} \right)^2$$

trong đó biểu thức bên trong dấu ngoặc đơn là sự khác biệt giữa thứ i lớp thực tế của cá thể và lớp dự đoán của nó. Tổng bình phương này là những gì chúng ta phải tối thiểu hóa bằng cách chọn các hệ số thích hợp.

Đây là tất cả bắt đầu trông khá ghê gớm. Tuy nhiên, việc giảm thiểu kỹ thuật này rất đơn giản nếu bạn có nền tảng toán học phù hợp. Chỉ cần nói rằng đã có đủ ví dụ—nói đại khái, nhiều ví dụ hơn hơn là các thuộc tính—việc chọn các trọng số để giảm thiểu tổng các bình phương chênh lệch thực sự không khó. Nó liên quan đến một hoạt động đảo ngược ma trận, nhưng điều này sẵn có dưới dạng phần mềm đóng gói sẵn.

Khi phép toán đã được hoàn thành, kết quả là một tập hợp các trọng số, dựa trên dữ liệu đào tạo mà chúng ta có thể sử dụng để dự đoán lớp mới trường hợp. Chúng tôi đã thấy một ví dụ về điều này khi xem xét hiệu suất của CPU dữ liệu, và trọng số số thực tế được đưa ra trong Hình 3.7(a). công thức này có thể được sử dụng để dự đoán hiệu suất CPU của các phiên bản thử nghiệm mới.

Hồi quy tuyến tính là một phương pháp tuyệt vời, đơn giản để dự đoán số và nó đã được sử dụng rộng rãi trong các ứng dụng thống kê trong nhiều thập kỷ. Tất nhiên, tuyến tính các mô hình chịu bất lợi của tuyến tính. Nếu dữ liệu thể hiện sự phụ thuộc phi tuyến tính, thì đường thẳng phù hợp nhất sẽ được tìm thấy, trong đó “tốt nhất” là được hiểu là sự khác biệt bình phương trung bình nhỏ nhất. Dòng này có thể không phù hợp lắm.

Tuy nhiên, các mô hình tuyến tính phục vụ tốt như các khối xây dựng cho các phư ơng pháp học phức tạp hơ n.

Phân loại tuyến tính: Hồi quy logistic

Hồi quy tuyến tính có thể dễ dàng đư ợc sử dụng để phân loại trong các miền có số thuộc tính. Thật vậy, chúng ta có thể sử dụng bất kỳ kỹ thuật hồi quy nào, dù là tuyến tính hay phi tuyến tính, để phân loại. Bí quyết là thực hiện hồi quy cho mỗi lớp, đặt đầu ra bằng một cho các phiên bản đào tạo thuộc về lớp và số không cho những ngư ời không. Kết quả là một biểu thức tuyến tính cho lớp. Sau đó, đư a ra một ví dụ kiểm tra của lớp chư a biết, hãy tính giá trị của từng biểu thức tuyến tính và chọn biểu thức lớn nhất. Phư ơng pháp này đôi khi đư ợc gọi là hồi quy tuyến tính đa phản hồi.

Một cách để xem xét hồi quy tuyến tính đa phản hồi là tư ởng tư ợng rằng nó xấp xỉ một hàm thành viên số cho mỗi lớp. Thành viên chức năng là 1 cho các trư ờng hợp thuộc về lớp đó và 0 cho các trư ờng hợp khác. Đư a ra một thể hiện mới, chúng tôi tính toán tư cách thành viên của nó cho mỗi lớp và chọn To nhất.

Hồi quy tuyến tính đa phản hồi thư ờng mang lại kết quả tốt trong thực tế. Tuy nhiên, nó có hai như ợc điểm. Đầu tiên, các giá trị thành viên mà nó tạo ra không xác suất phù hợp vì chúng có thể nằm ngoài phạm vi từ 0 đến 1. Thứ hai, hồi quy bình phư ơng nhỏ nhất giả định rằng các sai số không chỉ độc lập về mặt thống kê mà còn đư ợc phân phối chuẩn với cùng độ lệch chuẩn, một giả định bị vi phạm trắng trợn khi phư ơng pháp này đư ợc áp dụng cho các vấn đề phân loại bởi vì các quan sát chỉ nhận các giá trị 0 và 1.

Một kỹ thuật thống kê liên quan đư ợc gọi là hồi quy logistic không bị những vấn đề này. Thay vì xấp xỉ trực tiếp các giá trị 0 và 1, do đó mạo hiểm các giá trị xác suất không hợp lệ khi mục tiêu bị bắn quá mức, hồi quy logistic xây dựng một mô hình tuyến tính dựa trên một biến mục tiêu đã chuyển đổi.

Đầu tiên giả sử rằng chỉ có hai lớp. Hồi quy logistic thay thế biến mục tiêu ban đầu

$$\Pr[1 \mid a_1, a_2, \ldots, a_k],$$

không thể xấp xỉ chính xác bằng cách sử dụng hàm tuyến tính, với

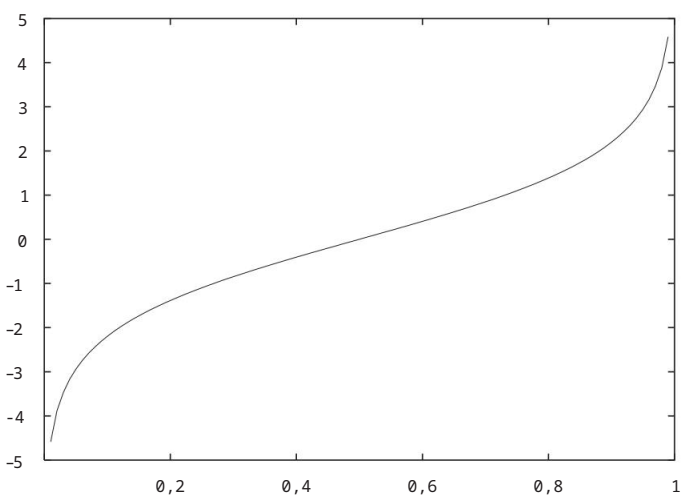
$$\log(\Pr[1 \mid a_1, a_2, \ldots, a_k] / (1 - \Pr[1 \mid a_1, a_2, \ldots, a_k])).$$

Các giá trị kết quả không còn bị ràng buộc trong khoảng từ 0 đến 1 như ng có thể nằm ở bất kỳ đâu giữa vô cực âm và vô cực dư ơng. Hình 4.9(a) vẽ đồ thị hàm biến đổi, thư ờng đư ợc gọi là phép biến đổi logit.

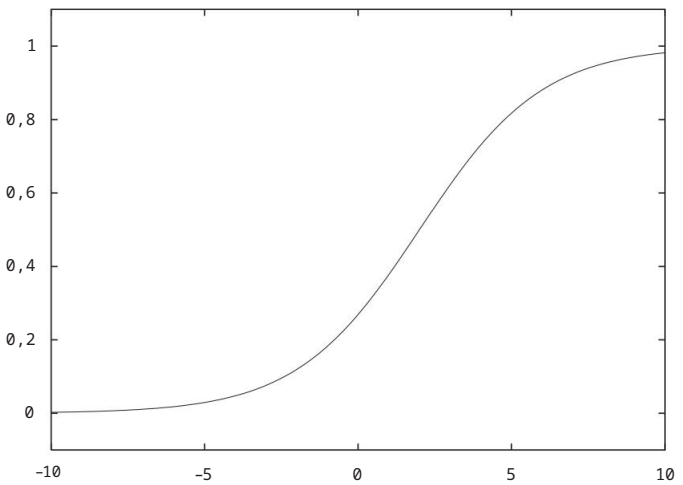
Biến đư ợc chuyển đổi đư ợc xấp xỉ bằng cách sử dụng hàm tuyến tính giống như những cái đư ợc tạo ra bởi hồi quy tuyến tính. Mô hình kết quả là

$$\text{đầu ra} = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k,$$

$$\Pr[1 \mid a_1, a_2, \ldots, a_k] = 1 / (1 + \exp(-w_0 - w_1 a_1 - \ldots - w_k a_k)),$$



0 (a)



(b)

Hình 4.9 Hồi quy logistic: (a) biến đổi logit và (b) ví dụ về hàm hồi quy logistic.

với trọng số w. Hình 4.9(b) cho thấy một ví dụ về chức năng này trong một chiều, với hai trọng số w0 = 0,5 và w1 = 1.

Cũng giống như trong hồi quy tuyến tính, trọng số phải được tìm thấy phù hợp với dữ liệu đào tạo Tốt. Hồi quy tuyến tính đo mức độ phù hợp bằng cách sử dụng lỗi bình phương. TRONG thay vào đó , hồi quy logistic , khả năng đăng nhập của mô hình được sử dụng. Điều này được đưa ra qua

N
Â x (11 -) ( ) aa a k [nhật ký trước | tôi ( ) , 12 ( ) i ( ) 1 , ... , +)] tôi ( ) x ([nhật ký trước 1 | aa a k , 12 ( ) tôi ( ) tôi ( ) , ... ,])
= 1

trong đó x(i) bằng không hoặc bằng một.

Các trọng số cần được chọn để tối đa hóa khả năng đăng nhập. Có một số phương pháp để giải quyết vấn đề tối đa hóa này. Một cách đơn giản là giải lặp đi lặp lại một chuỗi các bài toán hồi quy bình phương nhỏ nhất có trọng số cho đến khi khả năng đăng nhập hội tụ đến mức tối đa, điều này thường xảy ra trong một vài lặp đi lặp lại.

Để khái quát hóa hồi quy logistic cho một số lớp, một khả năng là tiến hành theo cách được mô tả trước đây cho hồi quy tuyến tính đa phản hồi bằng cách hình thành hồi quy logistic một cách độc lập cho từng lớp. thật không may kết quả ước tính xác suất sẽ không tổng bằng một. Để có được các xác suất phù hợp, cần phải kết hợp các mô hình riêng lẻ cho mỗi lớp. Điều này mang lại một vấn đề tối ưu hóa chung và có các phương pháp giải pháp hiệu quả cho vấn đề này.

Một cách đơn giản hơn về mặt khái niệm và rất chung chung để giải quyết các vấn đề đa lớp được gọi là phân loại theo cặp. Ở đây, một bộ phân loại được xây dựng cho mỗi cặp các lớp, chỉ sử dụng các thể hiện từ hai lớp này. Đầu ra trên một ví dụ kiểm tra không xác định dựa trên lớp nào nhận được nhiều phiếu bầu nhất. Cái này phương pháp thường mang lại kết quả chính xác về lỗi phân loại. Nó có thể cũng được sử dụng để tạo ra các ước tính xác suất bằng cách áp dụng một phương pháp được gọi là khớp nối thông minh theo cặp, phương pháp này hiệu chỉnh các ước tính xác suất riêng lẻ từ các bộ phân loại khác nhau.

Nếu có k lớp, phân loại theo cặp sẽ xây dựng tổng cộng k(k - 1)/2 bộ phân lớp. Mặc dù điều này nghe có vẻ tốn nhiều công sức tính toán một cách không cần thiết, nhưng không phải vậy. TRONG thực tế, nếu các lớp được phân bổ đồng đều thì việc phân loại theo cặp ít nhất cũng nhanh như bất kỳ phương pháp đa lớp nào khác. Lý do là mỗi vấn đề học theo cặp chỉ liên quan đến các trường hợp liên quan đến hai lớp đang được xem xét. Nếu n thể hiện được chia đều cho k lớp, thì số tiền này là 2n/k trường hợp cho mỗi vấn đề. Giả sử thuật toán học cho bài toán hai lớp với n trường hợp mất thời gian tỷ lệ thuận với n giây để thực thi. Sau đó chạy thời gian để phân loại theo cặp tỷ lệ với k(k - 1)/2 ¥ 2n/k giây, đó là (k - 1)n. Nói cách khác, phương pháp quy mô tuyến tính với số lượng của các lớp học. Nếu thuật toán học tốn nhiều thời gian hơn—chẳng hạn tỉ lệ với n2—lợi thế của cách tiếp cận theo cặp càng trở nên rõ rệt hơn.

Việc sử dụng các hàm tuyến tính để phân loại có thể dễ dàng được hình dung trong không gian thể hiện. Ranh giới quyết định cho hồi quy logistic hai lớp nằm trong đó xác suất dự đoán là 0,5, nghĩa là:

ưu tiên [1 | à 1,2 , . . . , một k] -- (1 1 wa 0 11 kk w wa = + (kinh nghiệm . . .)) = 0 5 .

Điều này xảy ra khi

$$w_0 + w_1 a_1 + \ldots + w_k a_k > 0.$$

Bởi vì đây là một đẳng thức tuyến tính trong các giá trị thuộc tính, ranh giới là một tuyến tính mặt phẳng, hoặc siêu phẳng, trong không gian thể hiện. Thật dễ dàng để hình dung các tập hợp các điểm mà không thể được phân tách bằng một siêu phẳng duy nhất và chúng không thể được phân biệt đối xử chính xác bằng hồi quy logistic.

Hồi quy tuyến tính đa phản hồi cũng gặp phải vấn đề tương tự. Mỗi lớp nhận được một vectơ trọng số được tính toán từ dữ liệu đào tạo. Tập trung cho thời điểm này trên một cặp lớp cụ thể. Giả sử vectơ trọng số của lớp 1 là

$$w_0^{(1)} + w_1^{(1)} a_1 + w_2^{(1)} a_2 + \ldots + w_k^{(1)} a_k$$

và tương tự cho lớp 2 với các chỉ số trên phù hợp. Sau đó, một trường hợp sẽ được phân vào lớp 1 thay vì lớp 2 nếu

$$w_0^{(1)} + w_1^{(1)} a_1 + \ldots + w_k^{(1)} a_k > w_0^{(2)} + w_1^{(2)} a_1 + \ldots + w_k^{(2)} a_k$$

Nói cách khác, nó sẽ được gán cho lớp 1 nếu

$$(w_0^{(1)} - w_0^{(2)}) + (w_1^{(1)} - w_1^{(2)}) a_1 + \ldots + (w_k^{(1)} - w_k^{(2)}) a_k > 0.$$

Đây là bất đẳng thức tuyến tính trong các giá trị thuộc tính, vì vậy ranh giới giữa mỗi cặp lớp là một siêu phẳng. Điều tương tự cũng đúng khi thực hiện theo cặp phân loại. Điểm khác biệt duy nhất là ranh giới giữa hai lớp là được điều chỉnh bởi các trường hợp đào tạo trong các lớp đó và không bị ảnh hưởng bởi Các lớp khác.

Phân loại tuyến tính bằng perceptron

Hồi quy logistic cố gắng tạo ra các ước tính xác suất chính xác bằng cách tối đa hóa xác suất của dữ liệu huấn luyện. Tất nhiên, ước tính xác suất chính xác dẫn đến phân loại chính xác. Tuy nhiên, không nhất thiết phải thực hiện ước tính xác suất nếu mục đích duy nhất của mô hình là dự đoán các nhãn lớp. Một cách tiếp cận khác là tìm hiểu một siêu phẳng phân tách các thể hiện tương ứng với các lớp khác nhau—hãy giả sử rằng chỉ có hai trong số chúng. Nếu như dữ liệu có thể được tách hoàn toàn thành hai nhóm bằng siêu phẳng, người ta nói có thể tách tuyến tính. Nó chỉ ra rằng nếu dữ liệu có thể phân tách tuyến tính, thì có là một thuật toán rất đơn giản để tìm một siêu phẳng phân cách.

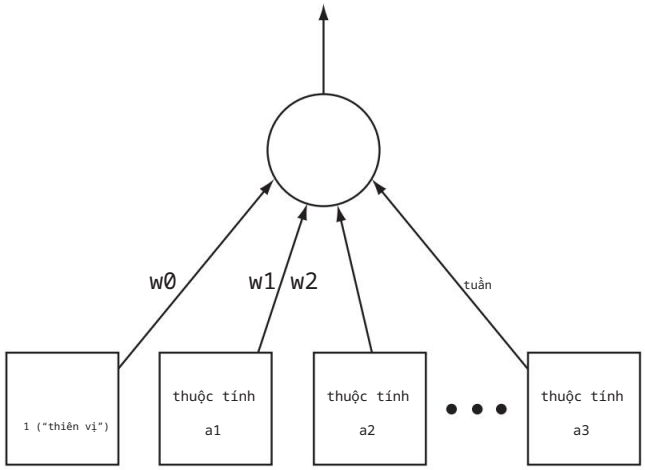
Thuật toán được gọi là quy tắc học perceptron. Trước khi nhìn vào nó trong chi tiết, hãy kiểm tra lại phương trình của một siêu phẳng:

$$w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k = 0.$$

Ở đây, a1, a2, . . ., ak là các giá trị thuộc tính và w0, w1, . . ., wk là các trọng số định nghĩa siêu phẳng. Chúng tôi sẽ giả sử rằng mỗi trường hợp đào tạo a1, a2, ·. . được mở rộng bởi một thuộc tính bổ sung a0 luôn có giá trị 1 (như chúng ta đã làm trong trường hợp hồi quy tuyến tính). Phần mở rộng này, được gọi là thiên vị, chỉ

```
Đặt tất cả các trọng số thành 0
Cho đến khi tất cả các trư ờng hợp trong dữ liệu đào tạo đư ợc phân loại chính xác
    Đối với mỗi trư ờng hợp tôi trong dữ liệu huấn luyện
        Nếu tôi đư ợc phân loại không chính xác bởi perceptron
            Nếu tôi thuộc về lớp đầu tiên, hãy thêm nó vào vectơ trọng số
            khác trừ nó khỏi vectơ trọng lư ợng
```

(Một)



(b)

Hình 4.10 Perceptron: (a) quy tắc học và (b) biểu diễn dư ới dạng mạng thần kinh.

có nghĩa là chúng ta không cần phải thêm một phần tử không đổi bổ sung vào tổng.

Nếu tổng lớn hơ n 0, chúng tôi sẽ dự đoán lớp đầu tiên; nếu không, chúng tôi sẽ dự đoán lớp thứ hai. Chúng tôi muốn tìm các giá trị cho các trọng số sao cho dữ liệu huấn luyện đư ợc phân loại chính xác bởi siêu phẳng.

Hình 4.10(a) đư a ra quy tắc học perceptron để tìm một siêu phẳng phân cách. Thuật toán lặp lại cho đến khi tìm thấy một giải pháp hoàn hảo, như ng nó sẽ chỉ hoạt động bình thư ờng nếu tồn tại một siêu phẳng phân tách, tức là nếu dữ liệu có thể phân tách tuyến tính. Mỗi lần lặp đi qua tất cả các trư ờng hợp đào tạo. Nếu gặp phải một thực thể bị phân loại sai, các tham số của siêu phẳng sẽ đư ợc thay đổi để thực thể bị phân loại sai di chuyển đến gần siêu phẳng hơ n hoặc thậm chí có thể băng qua siêu phẳng về phía bên phải. Nếu đối tư ợng thuộc về lớp đầu tiên, điều này đư ợc thực hiện bằng cách thêm các giá trị thuộc tính của nó vào vectơ trọng số; nếu không, họ bị trừ khỏi nó.

Để xem lý do tại sao điều này hoạt động, hãy xem xét tình huống sau một trường hợp liên quan đến lớp đầu tiên đã được thêm vào:

$$(w_0 + w_1 a_1^{(2)} + w_2 a_2^{(2)}) + (\quad) + \cdots + (w_k a_k^{(2)})$$

Điều này có nghĩa là đầu ra cho a đã tăng thêm

$$a_0^{(2)} + a_1^{(2)} + a_2^{(2)} + \cdots + a_k^{(2)}$$

Con số này luôn dương. Như vậy siêu phẳng đã di chuyển đúng hướng để phân loại trường hợp a là tích cực. Ngược lại, nếu một cá thể thuộc loại thứ hai bị phân loại sai, thì kết quả của cá thể đó sẽ giảm sau khi sửa đổi, một lần nữa di chuyển siêu phẳng đến đúng hướng.

Những chỉnh sửa này là gia tăng và có thể ảnh hưởng đến các bản cập nhật trước đó. Tuy nhiên, có thể chỉ ra rằng thuật toán hội tụ trong một số hữu hạn lặp lại nếu dữ liệu có thể phân tách tuyến tính. Tất nhiên, nếu dữ liệu không tuyến tính có thể tách rời, thuật toán sẽ không kết thúc, vì vậy giới hạn trên cần phải được áp đặt về số lần lặp khi áp dụng phương pháp này vào thực tế.

Siêu phẳng kết quả được gọi là một perceptron, và nó là ông nội của mạng nơ-ron (chúng ta quay lại mạng nơ-ron trong Phần 6.3). Hình 4.10(b) biểu diễn perceptron dưới dạng một biểu đồ với các nút và các cạnh có trọng số, được gọi một cách trực quan là một "mạng" gồm các "nơ-ron". Có hai lớp nút: đầu vào và đầu ra. Lớp đầu vào có một nút cho mọi thuộc tính, cộng với một nút bổ sung luôn được đặt thành một. Lớp đầu ra chỉ bao gồm một nút. Mỗi nút trong lớp đầu vào được kết nối với lớp đầu ra. Các kết nối có trọng số, và các trọng số là những con số được tìm thấy bởi quy tắc học perceptron.

Khi một thể hiện được trình bày cho perceptron, các giá trị thuộc tính của nó phục vụ cho "kích hoạt" lớp đầu vào. Chúng được nhân với trọng số và tính tổng tại nút đầu ra. Nếu tổng trọng số lớn hơn 0 thì tín hiệu đầu ra là 1, đại diện cho lớp học đầu tiên; mặt khác, nó là -1, đại diện cho thứ hai.

Phân loại tuyến tính bằng Winnow

Thuật toán perceptron không phải là phương pháp duy nhất được đảm bảo để tìm ra tách siêu phẳng cho bài toán tách tuyến tính. Đối với bộ dữ liệu có nhị phân có một giải pháp thay thế được gọi là Winnow, được minh họa trong Hình 4.11(a). Cấu trúc của hai thuật toán rất giống nhau. Giống như perceptron, Winnow chỉ cập nhật vectơ trọng số khi gặp trường hợp phân loại sai— đó là sai lầm điều khiển.

Hai phương pháp khác nhau về cách cập nhật trọng số. Quy tắc perceptron sử dụng một cơ chế cộng làm thay đổi vectơ trọng số bằng cách thêm (hoặc kéo phụ) vectơ thuộc tính của cá thể. Winnow sử dụng các bản cập nhật nhân và thay đổi trọng số riêng lẻ bằng cách nhân chúng với người dùng chỉ định tham số a (hoặc nghịch đảo của nó). Các giá trị thuộc tính ai là 0 hoặc 1 vì chúng ta

```
Trong khi một số trường hợp được phân loại sai
          đối với mọi trường hợp, hãy
             phân loại a bằng cách sử dụng trọng số hiện tại nếu lớp
             dự đoán không chính xác nếu thuộc về lớp đầu tiên

                   với mỗi ai là 1, nhân wi với a
                   (nếu ai là 0, giữ nguyên wi )
               nếu không thì
                   với mỗi ai là 1, chia wi cho a
                   (nếu ai là 0, giữ nguyên wi )
```

(Một)

```
Trong khi một số trường hợp được phân loại sai
    cho mọi trường hợp a
       phân loại a bằng trọng số hiện tại nếu lớp dự đoán không
       chính xác nếu a thuộc về lớp đầu tiên

            với mỗi ai là 1,
               nhân với          + bởi một
               chia wi - bởi a
            (nếu ai bằng 0 bỏ wi            + và wi - không thay đổi)
          mặt khác cho
            với mỗi ai là 1,
               nhân wi - với a
               chia wi       + bởi một
            (nếu ai bằng 0 bỏ wi            + và wi - không thay đổi)
```

(b)

Hình 4.11 Thuật toán Winnow: (a) phiên bản không cân bằng và (b) phiên bản cân bằng.

đang làm việc với dữ liệu nhị phân. Trọng số không thay đổi nếu giá trị thuộc tính là 0, vì khi đó họ không tham gia vào quyết định. Nếu không, hệ số nhân là a nếu thuộc tính đó giúp đưa ra quyết định đúng và 1/a nếu không.

Một điểm khác biệt nữa là ngưỡng trong hàm tuyến tính cũng là một tham số do người dùng chỉ định. Chúng tôi gọi ngưỡng này là q và phân loại một thực thể thuộc loại 1 khi và chỉ khi

$$w_1 a_1 + w_2 a_2 + \cdots + w_k a_k \geq q$$

Hệ số a cần phải lớn hơn một. Wi được đặt thành hằng số tại bắt đầu.

Thuật toán mà chúng tôi đã mô tả không cho phép trọng số âm, mà—tùy thuộc vào miền—có thể là một nhược điểm. Tuy nhiên, có một phiên bản, được gọi là Winnow cân bằng, cho phép họ. Phiên bản này duy trì hai vectơ trọng số, một cho mỗi lớp. Một thể hiện được phân loại là thuộc về lớp 1 nếu:

$$(w_1^+ a_1 + w_2^+ a_2 + \cdots + w_k^+ a_k) - (w_1^- a_1 + w_2^- a_2 + \cdots + w_k^- a_k) \geq q$$

Hình 4.11(b) thể hiện thuật toán cân bằng.

Winnow rất hiệu quả trong việc tìm kiếm các tính năng có liên quan trong tập dữ liệu—do đó nó được gọi là **bộ học hiệu quả thuộc tính** . Điều đó có nghĩa là nó có thể là một thuật toán ứng cử viên tốt nếu tập dữ liệu có nhiều tính năng (nhị phân) và hầu hết chúng không liên quan. Cả winnow và thuật toán perceptron đều có thể được sử dụng trong một cài đặt trực tuyến trong đó các phiên bản mới đến liên tục, bởi vì chúng có thể cập nhật dần dần các giả thuyết của họ khi các trường hợp mới xuất hiện.

4.7 Học tập dựa trên phiên bản

Trong học tập dựa trên cá thể, các ví dụ đào tạo được lưu trữ nguyên văn và hàm khoảng cách được sử dụng để xác định thành viên nào của tập huấn luyện gần nhất đến một trường hợp thử nghiệm không xác định. Khi phiên bản đào tạo gần nhất đã được định vị, lớp của nó được dự đoán cho phiên bản thử nghiệm. Vấn đề duy nhất còn lại đang xác định hàm khoảng cách và điều đó không khó thực hiện lắm, đặc biệt nếu các thuộc tính là số.

hàm khoảng cách

Mặc dù có những lựa chọn khả thi khác, nhưng hầu hết những người học dựa trên cá thể đều sử dụng khoảng cách Euclide. Khoảng cách giữa một thể hiện với các giá trị thuộc tính $a_1^{(1)}, a_2^{(1)}, \ldots, a_k^{(1)}$ (với k là số thuộc tính) và một thuộc tính có giá trị $a_1^{(2)}, a_2^{(2)}, \ldots, a_k^{(2)}$ được định nghĩa là

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \cdots + (a_k^{(1)} - a_k^{(2)})^2}.$$

Khi so sánh khoảng cách, không cần thiết phải thực hiện phép toán căn bậc hai; tổng bình phương có thể được so sánh trực tiếp. Một thay thế cho
Khoảng cách Euclide là số liệu Manhattan hoặc khối thành phố, trong đó sự khác biệt giữa các giá trị thuộc tính không bình phương mà chỉ được cộng lại (sau khi lấy giá trị tuyệt đối). Những người khác có được bằng cách lấy quyền hạn cao hơn bình phương. Quyền hạn cao hơn làm tăng ảnh hưởng của những khác biệt lớn với chi phí nhỏ sự khác biệt. Nói chung, khoảng cách Euclide đại diện cho một sự thỏa hiệp tốt. Các số liệu khoảng cách khác có thể phù hợp hơn trong những trường hợp đặc biệt. Các điều quan trọng là nghĩ về các trường hợp thực tế và ý nghĩa của việc tách chúng ra bằng một khoảng cách nhất định—chẳng hạn, hai lần khoảng cách đó có nghĩa là gì?

Các thuộc tính khác nhau được đo trên các thang đo khác nhau, và vậy nếu công thức khoảng cách Euclide được sử dụng trực tiếp, tác động của một số thuộc tính có thể bị lấn át hoàn toàn bởi các thuộc tính khác có thang đo lớn hơn. Do đó, thông thường sẽ chuẩn hóa tất cả các giá trị thuộc tính nằm trong khoảng từ 0 đến 1, bằng cách tính toán

$$\text{Một} = \frac{\text{tối} - \text{tối thiểu } v}{\text{lớn nhất } v - \text{nhỏ nhất}}$$

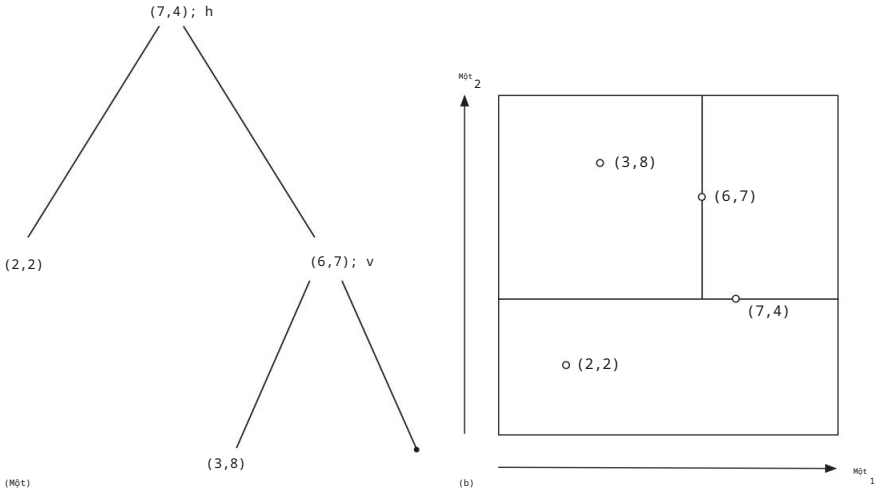
trong đó vi là giá trị thực của thuộc tính i, giá trị lớn nhất và nhỏ nhất là chiếm tất cả các trường hợp trong tập huấn luyện.

Các công thức này mặc nhiên giả định các thuộc tính số. Ở đây, sự khác biệt giữa hai giá trị chỉ là sự khác biệt về số giữa chúng và đây là sự khác biệt được bình phương và thêm vào để tạo ra hàm khoảng cách. Đối với danh nghĩa các thuộc tính nhận các giá trị tượng trưng hơn là số, sự khác biệt giữa hai giá trị không giống nhau thường được coi là một, trong khi nếu các giá trị giống nhau thì sự khác biệt bằng không. Không cần mở rộng quy mô trong này trường hợp vì chỉ các giá trị 0 và 1 được sử dụng.

Một chính sách chung để xử lý các giá trị bị thiếu như sau. Đối với danh nghĩa các thuộc tính, giả sử rằng một tính năng bị thiếu khác biệt tối đa với bất kỳ tính năng nào khác giá trị tính năng. Vì vậy, nếu một trong hai hoặc cả hai giá trị bị thiếu, hoặc nếu các giá trị khác nhau, thì sự khác biệt giữa chúng được coi là một; sự khác biệt chỉ bằng không nếu chúng không bị thiếu và cả hai đều giống nhau. Đối với các thuộc tính số, sự khác biệt giữa hai giá trị bị thiếu cũng được coi là một. Tuy nhiên, nếu chỉ một giá trị bị thiếu, sự khác biệt thường được coi là kích thước (chuẩn hóa) của giá trị khác hoặc một trừ đi kích thước đó, tùy theo giá trị nào lớn hơn. Điều này có nghĩa là nếu các giá trị bị thiếu, sự khác biệt lớn nhất có thể.

Tìm hàng xóm gần nhất một cách hiệu quả

Mặc dù học tập dựa trên phiên bản đơn giản và hiệu quả, nhưng nó thường chậm. Các cách rõ ràng để tìm thành viên nào của tập huấn luyện gần nhất với một ẩn số trường hợp kiểm tra là tính khoảng cách từ mọi thành viên của tập huấn luyện

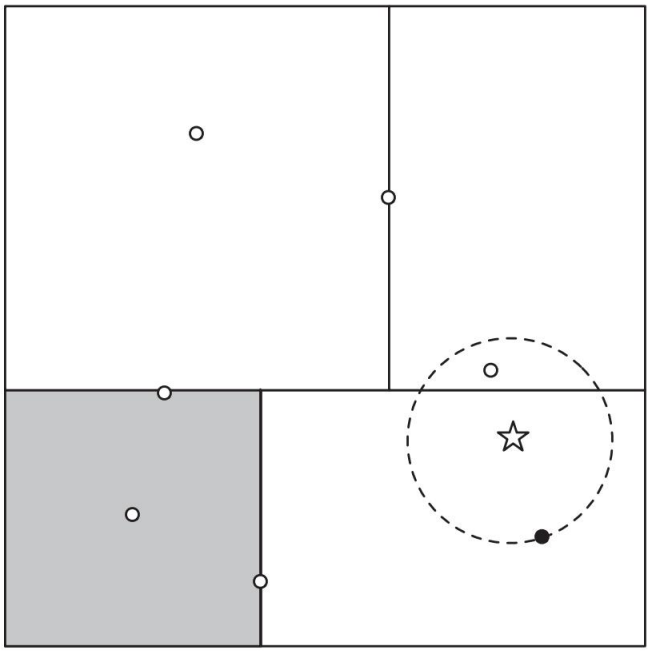


Hình 4.12 Một cây kD cho bốn trường hợp huấn luyện: (a) cây và (b) trường hợp và chia tách.

và chọn cái nhỏ nhất. Thủ tục này là tuyến tính trong số lượng đào tạo trường hợp: nói cách khác, thời gian cần thiết để đưa ra một dự đoán duy nhất tỷ lệ thuận với số lượng trường hợp đào tạo. Xử lý toàn bộ bộ kiểm tra mất thời gian tỷ lệ thuận với tích của số lượng trường hợp trong quá trình đào tạo và bộ kiểm tra.

Hàng xóm gần nhất có thể được tìm thấy hiệu quả hơn bằng cách đại diện cho đào tạo được thiết lập như một cái cây, mặc dù nó không hoàn toàn rõ ràng bằng cách nào. Một cấu trúc phù hợp là một kD-cây. Đây là một cây nhị phân chia không gian đầu vào bằng một siêu phẳng và sau đó chia lại từng phân vùng theo cách đệ quy. Tất cả các phân chia được thực hiện song song với một trong các trục, theo chiều dọc hoặc chiều ngang, trong trường hợp hai chiều. Cấu trúc dữ liệu được gọi là cây kD vì nó lưu trữ một tập hợp các điểm trong không gian k chiều, k là số thuộc tính.

Hình 4.12(a) đưa ra một ví dụ nhỏ với k = 2, và Hình 4.12(b) cho thấy bốn trường hợp đào tạo mà nó đại diện, cùng với các siêu phẳng cấu thành cái cây. Lưu ý rằng các siêu phẳng này không phải là các ranh giới quyết định: các quyết định là được thực hiện trên cơ sở lân cận gần nhất như được giải thích sau. Đường xẻ đầu tiên theo chiều ngang (h), qua điểm (7,4)—đây là gốc của cây. Nhánh bên trái không chia nhỏ hơn nữa: nó chứa một điểm duy nhất (2,2), là một chiếc lá của cây. Các nhánh phải chẻ dọc (v) tại điểm (6,7). Phần tử con bên trái của nó trống rỗng, và con phải của nó chứa điểm (3,8). Như ví dụ này minh họa, mỗi khu vực chỉ chứa một điểm—hoặc, có lẽ, không có điểm nào. Nhánh anh em của cây—ví dụ, hai nhánh con của gốc trong Hình 4.12(a)—không nhất thiết phải được phát triển ở cùng độ sâu. Mỗi điểm trong tập huấn luyện tương ứng đến một nút duy nhất và tới một nửa là các nút lá.



Hình 4.13 Sử dụng cây kD để tìm láng giềng gần nhất của ngôi sao.

Làm cách nào để bạn xây dựng cây kD từ tập dữ liệu? Nó có thể được cập nhật một cách hiệu quả khi các ví dụ đào tạo mới được thêm vào không? Và làm thế nào để nó tăng tốc độ tính toán hàng xóm gần nhất? Chúng tôi giải quyết câu hỏi cuối cùng đầu tiên.

Để xác định vị trí hàng xóm gần nhất của một điểm mục tiêu nhất định, hãy đi theo cây từ gốc của nó để xác định vùng chứa mục tiêu. Hình 4.13 cho thấy một không gian giống như Hình 4.12(b) nhưng có thêm một số trường hợp và một giới hạn bổ sung. Mục tiêu, không phải là một trong các trường hợp trong cây, được đánh dấu bằng một ngôi sao.

Nút lá của vùng chứa mục tiêu có màu đen. Đây không nhất thiết là hàng xóm gần nhất của mục tiêu, như ví dụ này minh họa, nhưng nó là một xấp xỉ đầu tiên tốt. Đặc biệt, bất kỳ hàng xóm nào gần hơn phải nằm gần hơn—trong vòng tròn nét đứt trong Hình 4.13. Để xác định xem một nút có tồn tại hay không, trước tiên hãy kiểm tra xem liệu một nút lân cận gần hơn có thể nằm trong nút anh chị em của nút hay không.

Anh chị em của nút đen được tô bóng trong Hình 4.13, và vòng tròn không giao cắt với nó, vì vậy anh chị em không thể chứa nút hàng xóm gần hơn. Sau đó, sao lưu vào nút cha và kiểm tra anh chị em của nó —ở đây bao gồm mọi thứ bên trên đường ngang. Trong trường hợp này, nó phải được khám phá, bởi vì khu vực mà nó bao phủ giao với đường tròn tốt nhất cho đến nay. Để khám phá nó, hãy tìm các con gái của nó (hai dì của điểm ban đầu), kiểm tra xem chúng có giao nhau với đường tròn không (hình bên trái thì không, nhưng hình bên phải thì có) và đi xuống để xem liệu nó có chứa điểm gần hơn không (nó có).

Trong trường hợp điển hình, thuật toán này nhanh hơn nhiều so với việc kiểm tra tất cả các điểm để tìm hàng xóm gần nhất. Công việc liên quan đến việc tìm hàng xóm gần nhất gần đúng ban đầu—điểm đen trong Hình 4.13—phụ thuộc vào độ sâu của cây, được cho bởi logarit của số nút, log2n . Khối lượng công việc liên quan đến quay lui để kiểm tra xem đây có thực sự là hàng xóm gần nhất hay không phụ thuộc một chút vào cây và mức độ xấp xỉ ban đầu tốt như thế nào. Nhưng đối với một cây được xây dựng tốt có các nút gần như hình vuông, chứ không phải là các hình chữ nhật mảnh dài, thì nó cũng có thể được biểu thị bằng logarit trong số lượng các nút.

Làm thế nào để bạn xây dựng một cây tốt cho một tập hợp các ví dụ đào tạo? Vấn đề tập trung vào việc chọn phiên bản đào tạo đầu tiên để phân tách tại và hướng phân tách. Khi bạn có thể làm điều đó, hãy áp dụng đệ quy cùng một phương pháp cho từng phần tử con của phần tách ban đầu để xây dựng toàn bộ cây.

Để tìm hướng tốt cho quá trình phân tách, hãy tính toán phương sai của các điểm dữ liệu dọc theo từng trục riêng lẻ, chọn trục có phương sai lớn nhất và tạo một siêu phẳng phân tách vuông góc với nó. Để tìm một vị trí tốt cho siêu phẳng, xác định giá trị trung bình dọc theo trục đó và chọn điểm tương ứng. Điều này làm cho đường phân chia vuông góc với hướng phân tán lớn nhất, với một nửa số điểm nằm ở hai bên. Điều này tạo ra một cây cân đối. Để tránh các vùng mỏng dài, tốt nhất là các đường phân tách liên tiếp dọc theo các trục khác nhau, điều này có thể là do thứ nguyên của phương sai lớn nhất được chọn ở mỗi giai đoạn.
Tuy nhiên, nếu sự phân bố các điểm bị sai lệch nghiêm trọng, việc chọn giá trị trung bình có thể tạo ra một số phần tách liên tiếp theo cùng một hướng, tạo ra các siêu hình chữ nhật dài và mảnh. Một chiến lược tốt hơn là tính giá trị trung bình thay vì trung bình và sử dụng điểm gần nhất với điểm đó. Cây sẽ không hoàn toàn cân đối, nhưng các vùng của nó sẽ có xu hướng vuông vức vì có nhiều khả năng các hướng khác nhau sẽ được chọn cho các lần phân tách liên tiếp.

Một lợi thế của học tập dựa trên cá thể so với hầu hết các phương pháp học máy khác là các ví dụ mới có thể được thêm vào tập huấn luyện bất kỳ lúc nào. Để duy trì lợi thế này khi sử dụng cây kD, chúng ta cần có khả năng cập nhật nó dần dần với các điểm dữ liệu mới. Để làm điều này, hãy xác định nút lá nào chứa điểm mới và tìm siêu hình chữ nhật của nó. Nếu nó trống, chỉ cần đặt điểm mới ở đó. Nếu không, hãy chia siêu hình chữ nhật, chia nó dọc theo chiều dài nhất của nó để bảo toàn tính vuông góc. Heuristic đơn giản này không đảm bảo rằng việc thêm một loạt các điểm sẽ duy trì sự cân bằng của cây, cũng không đảm bảo rằng các đám rối hyperrec sẽ được định hình tốt để tìm kiếm hàng xóm gần nhất. Thỉnh thoảng, bạn nên xây dựng lại cây từ đầu—ví dụ: khi độ sâu của nó tăng lên gấp đôi độ sâu tốt nhất có thể.

Như chúng ta đã thấy, kD-trees là cấu trúc dữ liệu tốt để tìm kiếm các bor lân cận gần nhất một cách hiệu quả. Tuy nhiên, chúng không hoàn hảo. Các bộ dữ liệu sai lệch thể hiện xung đột cơ bản giữa mong muốn cây được cân bằng hoàn hảo và mong muốn các vùng vuông vắn. Quan trọng hơn, hình chữ nhật - thậm chí là hình vuông - dù sao cũng không phải là hình dạng tốt nhất để sử dụng vì các góc của chúng. Nếu đường tròn nét đứt trong

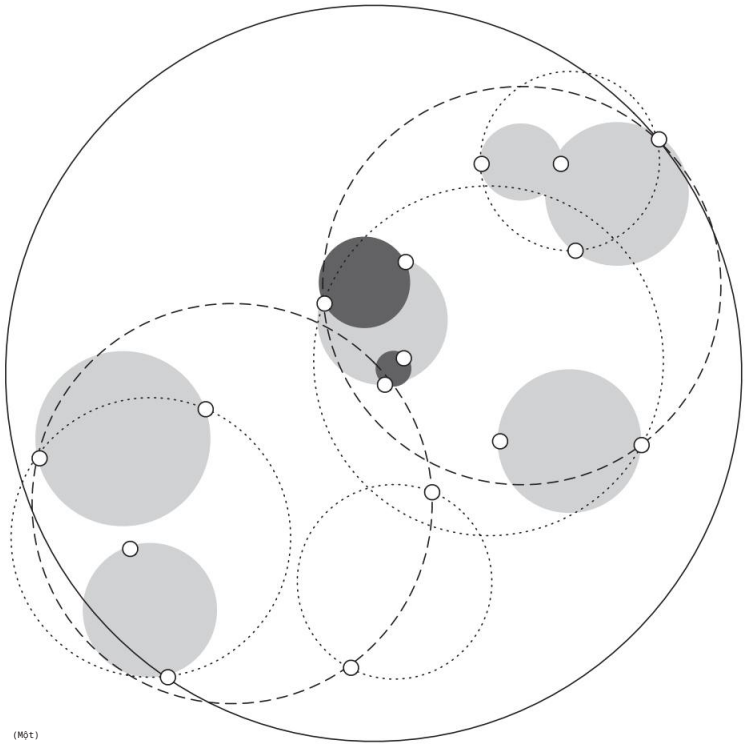
Hình 4.13 lớn hơn bất kỳ, điều đó sẽ xảy ra nếu đối tượng màu đen ở xa mục tiêu hơn một chút, nó sẽ cắt góc dưới bên phải của hình chữ nhật ở trên cùng bên trái và sau đó hình chữ nhật đó cũng sẽ phải được điều tra— mặc dù thực tế là các trường hợp đào tạo xác định nó còn lâu mới đến góc được đề cập. Các góc của khu vực hình chữ nhật là khó xử.

Giải pháp? Sử dụng siêu hình cầu, không phải siêu hình chữ nhật. Các hình cầu lân cận có thể chồng lên nhau trong khi hình chữ nhật có thể tiếp giáp, nhưng đây không phải là vấn đề vì thuật toán lân cận gần nhất cho cây kD được mô tả trước đây không phụ thuộc vào các vùng rời rạc. Một cấu trúc dữ liệu được gọi là cây bóng xác định các siêu cầu k chiều ("quả bóng") bao phủ các điểm dữ liệu và sắp xếp chúng thành một cây.

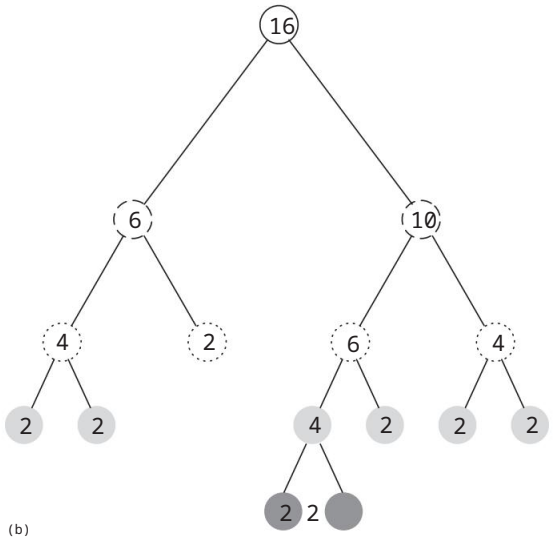
Hình 4.14(a) hiển thị 16 trường hợp huấn luyện trong không gian hai chiều, được đặt trên một mẫu các vòng tròn chồng lên nhau và Hình 4.14(b) hiển thị một cây được hình thành từ các vòng tròn này. Các vòng tròn ở các cấp độ khác nhau của cây được biểu thị bằng các kiểu gạch ngang khác nhau và các vòng tròn nhỏ hơn được vẽ bằng các sắc thái xám. Mỗi nút của cây đại diện cho một quả bóng và nút đó được gạch ngang hoặc tô bóng theo cùng một quy ước để bạn có thể xác định các quả bóng đang ở cấp độ nào. Để giúp bạn hiểu về cây, các số được đặt trên các nút để hiển thị có bao nhiêu điểm dữ liệu được coi là bên trong quả bóng đó. Nhưng hãy cẩn thận: điều này không nhất thiết giống với số điểm rơi vào vùng không gian mà quả bóng đại diện. Các vùng ở mỗi cấp độ đôi khi trùng nhau, nhưng các điểm rơi vào vùng chồng lấp chỉ được gán cho một trong các quả bóng chồng lên nhau (sơ đồ không hiển thị cái nào). Thay vì số lượng chiếm chỗ trong Hình 4.14(b), các nút của cây bóng thực tế lưu trữ tâm và bán kính của quả bóng của chúng; các nút lá cũng ghi lại các điểm mà chúng chứa.

Để sử dụng cây bóng để tìm hàng xóm gần nhất với một mục tiêu nhất định, hãy bắt đầu bằng cách duyệt cây từ trên xuống để xác định vị trí chiếc lá chứa mục tiêu và tìm điểm gần mục tiêu nhất trong quả bóng đó. Điều này đưa ra một giới hạn trên cho khoảng cách của mục tiêu từ hàng xóm gần nhất của nó. Sau đó, giống như đối với cây kD, hãy kiểm tra nút anh chị em. Nếu khoảng cách từ mục tiêu đến trung tâm của anh chị em vượt quá bán kính của nó cộng với giới hạn trên hiện tại, thì nó không thể chứa điểm gần hơn; nếu không thì anh chị em phải được kiểm tra bằng cách đi xuống cây hơn nữa. Trong Hình 4.15, mục tiêu được đánh dấu bằng một ngôi sao và chấm đen là hàng xóm gần nhất hiện được biết đến của nó. Có thể loại trừ toàn bộ nội dung của quả bóng xám: nó không thể chứa một điểm gần hơn vì tâm của nó quá xa.
Tiến hành đệ quy sao lưu cây về gốc của nó, kiểm tra bất kỳ quả bóng nào có thể chứa một điểm gần hơn giới hạn trên hiện tại.

Cây bóng được xây dựng từ trên xuống và như với cây kD, vấn đề cơ bản là tìm ra cách tốt để tách một quả bóng chứa một tập hợp các điểm dữ liệu thành hai.
Trong thực tế, bạn không cần phải tiếp tục cho đến khi các quả bóng lá chỉ chứa hai điểm: bạn có thể dừng lại sớm hơn, sau khi đạt đến số lượng tối thiểu được xác định trước—và điều tương tự cũng xảy ra với các cây kD. Đây là một phương pháp phân tách có thể.

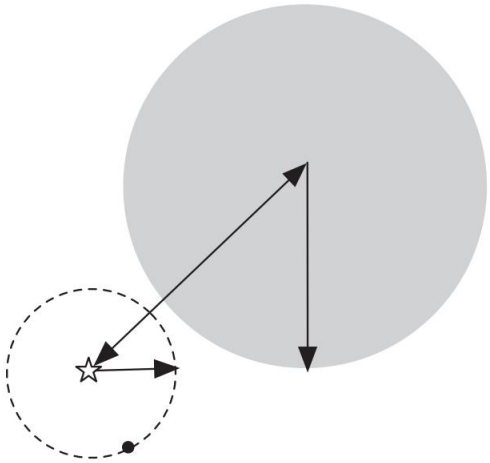


(Một)



(b)

Hình 4.14 Cây bóng cho 16 trường hợp huấn luyện: (a) trường hợp và quả bóng và (b) cây.



Hình 4.15 Loại trừ toàn bộ quả bóng (màu xám) dựa trên điểm mục tiêu (ngôi sao) và dòng điện của nó láng giềng gần nhất.

Chọn điểm trong quả bóng cách xa tâm của nó nhất, sau đó chọn điểm thứ hai điểm xa điểm đầu tiên nhất. Gán tất cả các điểm dữ liệu trong quả bóng cho gần nhất một trong hai trung tâm cụm này, sau đó tính toán trọng tâm của mỗi cụm và bán kính tối thiểu cần thiết để nó bao quanh tất cả các điểm dữ liệu mà nó đa i diê n. Phư ơ ng pháp này có ư u điểm là chi phí tách một quả bóng chứa n điểm chỉ tuyến tính theo n. Có nhiều thuật toán phức tạp hơ n tạo ra những quả bóng chặt hơ n, như ng chúng đòi hỏi nhiều tính toán hơ n. Chúng tôi sẽ không mô tả các thuật toán phức tạp để xây dựng các cây bóng hoặc cập nhật chúng dần dần khi gặp phải các trư ờng hợp đào tạo mới.

Cuộc thảo luận

Học tập dựa trên phiên bản lân cận gần nhất rất đơ n giản và thư ờng hoạt động rất tốt. Trong phư ơ ng pháp đư ợc mô tả trư ớc đây, mỗi thuộc tính có cùng một ảnh hư ởng đến quyết định, giống như trong phư ơ ng pháp Naïve Bayes. Khác vấn đề là cơ sở dữ liệu có thể dễ dàng bị hỏng bởi các ví dụ ồn ào. Một giải pháp là áp dụng chiến lư ợc k-láng giềng gần nhất, trong đó một số cố định, nhỏ, số k hàng xóm gần nhất—chẳng hạn như năm—đư ợc định vị và sử dụng cùng nhau để xác định loại của phiên bản thử nghiệm thông qua biểu quyết đa số đơ n giản. (Ghi chú mà chúng tôi đã sử dụng k để biểu thị số lư ợng thuộc tính trư ớc đó; đây là một cách sử dụng khác, độc lập.) Một cách khác để kiểm tra cơ sở dữ liệu chống nhiễu là chọn những mẫu mực đư ợc thêm vào nó một cách có chọn lọc và thận trọng; quy trình đư ợc cải thiện, đư ợc mô tả trong Chư ơ ng 6, giải quyết những thiếu sót này.

Phương pháp láng giềng gần nhất bắt nguồn từ nhiều thập kỷ trước, và các nhà thống kê đã phân tích lược đồ k-láng giềng gần nhất vào đầu những năm 1950. Nếu số lượng phiên bản đào tạo lớn, sẽ có ý nghĩa trực quan khi sử dụng nhiều hơn một phiên bản gần nhất hàng xóm, nhưng rõ ràng điều này là nguy hiểm nếu có ít trường hợp. Nó có thể được hiển thị rằng khi cả k và số n trường hợp đều trở thành vô hạn theo cách như vậy rằng k/n Æ 0, xác suất lỗi đạt đến mức tối thiểu theo lý thuyết đối với bộ dữ liệu. Phương pháp láng giềng gần nhất đã được thông qua như là một phân loại vào đầu những năm 1960 và đã được sử dụng rộng rãi trong lĩnh vực nhận dạng mẫu trong hơn ba thập kỷ.

Phân loại hàng xóm gần nhất nổi tiếng là chậm cho đến khi cây kD bắt đầu được áp dụng vào đầu những năm 1990, mặc dù bản thân cấu trúc dữ liệu đã được phát triển sớm hơn nhiều. Trong thực tế, những cây này trở nên kém hiệu quả khi kích thước của không gian tăng lên và chỉ có giá trị khi số lượng thuộc tính là nhỏ—lên đến 10. Cây bóng đã được phát triển gần đây hơn nhiều và là một ví dụ về một cấu trúc tổng quát hơn đôi khi được gọi là cây số liệu. Các thuật toán tinh vi có thể tạo ra các cây số liệu xử lý thành công hàng nghìn của kích thước.

Thay vì lưu trữ tất cả các phiên bản đào tạo, bạn có thể nén chúng thành các vùng. Một kỹ thuật rất đơn giản, được đề cập ở cuối Phần 4.1, là chỉ cần ghi lại phạm vi giá trị được quan sát trong dữ liệu huấn luyện cho từng thuộc tính và danh mục. Đưa ra một phiên bản thử nghiệm, bạn tìm ra phạm vi giá trị thuộc tính rơi vào và chọn danh mục có số lượng phạm vi chính xác lớn nhất cho điều đó ví dụ. Một kỹ thuật phức tạp hơn một chút là xây dựng các khoảng cho mỗi thuộc tính và sử dụng tập huấn luyện để đếm số lần mỗi lớp xảy ra cho mỗi khoảng thời gian trên mỗi thuộc tính. Các thuộc tính số có thể được rời rạc hóa thành khoảng thời gian và "khoảng thời gian" bao gồm một điểm duy nhất có thể được sử dụng cho danh nghĩa cái. Sau đó, với một phiên bản thử nghiệm, bạn có thể xác định khoảng thời gian mà nó cư trú và phân loại nó bằng cách bỏ phiếu, một phương pháp được gọi là khoảng thời gian tính năng bỏ phiếu. Những cái này các phương pháp rất gần đúng, nhưng rất nhanh và có thể hữu ích cho phân tích ban đầu của các tập dữ liệu lớn.

4.8 Phân cụm

Các kỹ thuật phân cụm được áp dụng khi không có lớp nào được dự đoán mà thay vào đó khi các thể hiện được chia thành các nhóm tự nhiên. Các cụm này có thể phản ánh một số cơ chế đang hoạt động trong miền mà từ đó các trường hợp được rút ra, một cơ chế làm cho một số trường hợp có sự tương đồng mạnh mẽ hơn với nhau so với các trường hợp còn lại. Tập hợp cụm tự nhiên đòi hỏi các kỹ thuật khác nhau để phân loại và học tập kết hợp phương pháp chúng tôi đã xem xét cho đến nay.

Như chúng ta đã thấy trong Phần 3.9, có nhiều cách khác nhau để biểu thị kết quả của việc phân cụm. Các nhóm được xác định có thể là loại trừ để bất kỳ trường hợp nào chỉ thuộc về một nhóm. Hoặc chúng có thể chồng lên nhau để một ví dụ có thể rơi vào một số nhóm. Hoặc chúng có thể mang tính xác suất, theo đó một thể hiện thuộc về mỗi nhóm với một xác suất nhất định. Hoặc chúng có thể có cấu trúc phân cấp hơn, sao cho có sự phân chia thô sơ các thể hiện thành các nhóm ở trên cùng cấp độ, và mỗi nhóm trong số này được tinh chỉnh thêm—có lẽ tất cả các cách xuống đến trường hợp cá nhân. Thực sự, sự lựa chọn giữa những khả năng này nên được quyết định bởi bản chất của các cơ chế được cho là nền tảng của hiện tượng phân cụm cụ thể. Tuy nhiên, do các cơ chế này hiếm khi được đã biết—sự tồn tại của các cụm, xét cho cùng, là thứ mà chúng tôi đang cố gắng để khám phá—và vì những lý do thực dụng nữa, sự lựa chọn thường được quyết định bởi các công cụ phân cụm có sẵn.

Chúng ta sẽ xem xét một thuật toán hình thành các cụm trong miền số, phân vùng các thể hiện thành các cụm rời rạc. Giống như phương pháp láng giềng gần nhất cơ bản của học tập dựa trên cá thể, đây là một kỹ thuật đơn giản và dễ hiểu đã được sử dụng trong nhiều thập kỷ. Trong Chương 6, chúng ta xem xét phân cụm mới hơn các phương pháp thực hiện phân cụm gia tăng và xác suất.

Phân cụm dựa trên khoảng cách lặp lại

Kỹ thuật phân cụm cổ điển được gọi là k-means. Đầu tiên, bạn xác định trước có bao nhiêu cụm đang được tìm kiếm: đây là tham số k. Khi đó k điểm là được chọn ngẫu nhiên làm trung tâm cụm. Tất cả các trường hợp được gán cho gần nhất của họ trung tâm cụm theo số liệu khoảng cách Euclide thông thường. Tiếp theo, cen troid, hoặc trung bình, của các cá thể trong mỗi cụm được tính—đây là "trung bình" phần. Các trọng tâm này được coi là các giá trị trung tâm mới cho các cụm tương ứng của chúng. Cuối cùng, toàn bộ quá trình được lặp lại với các trung tâm cụm mới. Quá trình lặp lại tiếp tục cho đến khi các điểm giống nhau được gán cho từng cụm liên tiếp các vòng, ở giai đoạn đó, các trung tâm cụm đã ổn định và sẽ vẫn là giống nhau mãi mãi.

Phương pháp phân cụm này đơn giản và hiệu quả. Dễ dàng chứng minh rằng việc chọn trung tâm cụm làm trọng tâm sẽ giảm thiểu tổng bình phương khoảng cách từ mỗi điểm của cụm đến trung tâm của nó. Khi vòng lặp đã ổn định, mỗi điểm được gán cho trung tâm cụm gần nhất của nó, và vì vậy hiệu quả tổng thể là tối thiểu hóa tổng bình phương khoảng cách từ tất cả các điểm đến trung tâm cụm của chúng. Nhưng tối thiểu là một địa phương; không có gì đảm bảo rằng đó là mức tối thiểu toàn cầu. Các cụm cuối cùng khá nhạy cảm với các trung tâm cụm ban đầu. Những sắp xếp hoàn toàn khác biệt có thể phát sinh từ những thay đổi nhỏ trong lựa chọn ngẫu nhiên ban đầu. Trên thực tế, điều này đúng với tất cả các kỹ thuật phân cụm thực tế: hầu như không thể tìm được các cụm tối ưu toàn cục. Để tăng cơ hội tìm kiếm một toàn cầu

những người tối thiểu thường chạy thuật toán nhiều lần với các lựa chọn ban đầu khác nhau và chọn kết quả cuối cùng tốt nhất—kết quả có tổng khoảng cách bình phương nhỏ nhất.

Thật dễ dàng để tưởng tượng các tình huống trong đó phương tiện k không tìm được cụm tốt. Xét bốn trường hợp được sắp xếp tại các đỉnh của một hình chữ nhật trong không gian hai chiều. Có hai cụm tự nhiên, được hình thành bằng cách nhóm hai đỉnh ở hai đầu của một cạnh ngắn lại với nhau. Nhưng giả sử rằng hai trung tâm cụm ban đầu rơi vào trung điểm của các cạnh dài . Điều này tạo thành một cấu hình ổn định. Hai cụm, mỗi cụm chứa hai phiên bản ở hai đầu của cạnh dài—bất kể sự khác biệt giữa cạnh dài và cạnh ngắn lớn đến mức nào.

Tính toán khoảng cách nhanh hơn

Thuật toán phân cụm k-means thường yêu cầu một số lần lặp lại, mỗi lần lặp lại liên quan đến việc tìm khoảng cách của k trung tâm cụm từ mọi phiên bản để xác định cụm của nó. Có những phép tính gần đúng đơn giản giúp tăng tốc độ này một cách đáng kể. Ví dụ: bạn có thể chiếu tập dữ liệu và thực hiện các vết cắt dọc theo các trục đã chọn, thay vì sử dụng các phân chia siêu phẳng tùy ý được ngụ ý bằng cách chọn trung tâm cụm gần nhất. Nhưng điều này chắc chắn sẽ ảnh hưởng đến chất lượng của các cụm kết quả.

Đây là một cách tốt hơn để tăng tốc mọi thứ. Tìm trung tâm cụm gần nhất không khác nhiều so với tìm hàng xóm gần nhất trong học tập dựa trên cá thể.
Có thể sử dụng các giải pháp hiệu quả tương tự—cây kD và cây bóng—được không? Đúng! Thật vậy, chúng có thể được áp dụng theo cách thậm chí còn hiệu quả hơn, bởi vì trong mỗi lần lặp lại k-mean, tất cả các điểm dữ liệu đều được xử lý cùng nhau, trong khi ở các phiên bản thử nghiệm học tập dựa trên phiên bản được xử lý riêng lẻ.

Đầu tiên, xây dựng cây kD hoặc cây bóng cho tất cả các điểm dữ liệu, điểm này sẽ duy trì trạng thái tĩnh trong suốt quy trình phân cụm. Mỗi lần lặp lại của phương tiện k tạo ra một tập hợp các trung tâm cụm và tất cả các điểm dữ liệu phải được kiểm tra và gán cho trung tâm gần nhất. Một cách để xử lý các điểm là đi xuống cây từ gốc cho đến khi chạm tới một chiếc lá và kiểm tra từng điểm riêng lẻ trong chiếc lá để tìm trung tâm cụm gần nhất của nó. Nhưng có thể khu vực được đại diện bởi một nút bên trong cao hơn hoàn toàn nằm trong miền của một trung tâm cụm duy nhất. Trong trường hợp đó, tất cả các điểm dữ liệu bên dưới nút đó có thể được xử lý trong một lần!

Rốt cuộc, mục đích của bài tập là tìm vị trí mới cho các trung tâm cụm bằng cách tính trọng tâm của các điểm mà chúng chứa. Trọng tâm có thể được tính toán bằng cách giữ tổng vectơ đang chạy của các điểm trong cụm và đếm xem có bao nhiêu điểm cho đến nay. Cuối cùng, chỉ cần chia từng cái một để tìm trọng tâm. Giả sử rằng với mỗi nút của cây, chúng ta lưu trữ tổng vectơ của các điểm trong nút đó và số lượng điểm. Nếu toàn bộ nút nằm trong phạm vi của một cụm duy nhất, tổng số hoạt động cho cụm đó

có thể được cập nhật ngay lập tức. Nếu không, hãy tìm bên trong nút bằng cách tiến hành lặp lại một cách chủ động xuống cây.

Hình 4.16 cho thấy các trường hợp và cây bóng giống như Hình 4.14, nhưng với hai trung tâm cụm được đánh dấu là các ngôi sao màu đen. Bởi vì tất cả các phiên bản được gán cho trung tâm gần nhất, không gian được chia làm hai bởi đường kẻ đậm như trong Hình 4.16(a). Bắt đầu từ gốc của cây trong Hình 4.16(b), với các giá trị ban đầu cho tổng vectơ và số lượng cho mỗi cụm; tất cả các giá trị ban đầu bằng không. Tiến hành lặp đi lặp lại một cách chủ động xuống cây. Khi đạt đến nút A, tất cả các điểm bên trong nó nằm trong cụm 1, do đó, tổng và số của cụm 1 có thể được cập nhật với tổng và số của nút A và chúng ta không cần phải đi xuống nữa. Quay trở lại nút B, quả bóng của nó nằm trên ranh giới giữa các cụm, vì vậy các điểm của nó phải được kiểm tra riêng lẻ.

Khi đạt đến nút C, nó hoàn toàn nằm trong cụm 2; một lần nữa, chúng tôi có thể cập nhật cụm 2 ngay lập tức và không cần phải đi xuống nữa. Cây chỉ được kiểm tra xuống đến đường biên được đánh dấu bằng đường đứt nét trong Hình 4.16(b), và ưu điểm là các nút bên dưới không cần phải mở ra—ít nhất là không phải trên bước lặp đặc biệt này của phương tiện k . Lần tới, các trung tâm cụm sẽ thay đổi và mọi thứ có thể khác.

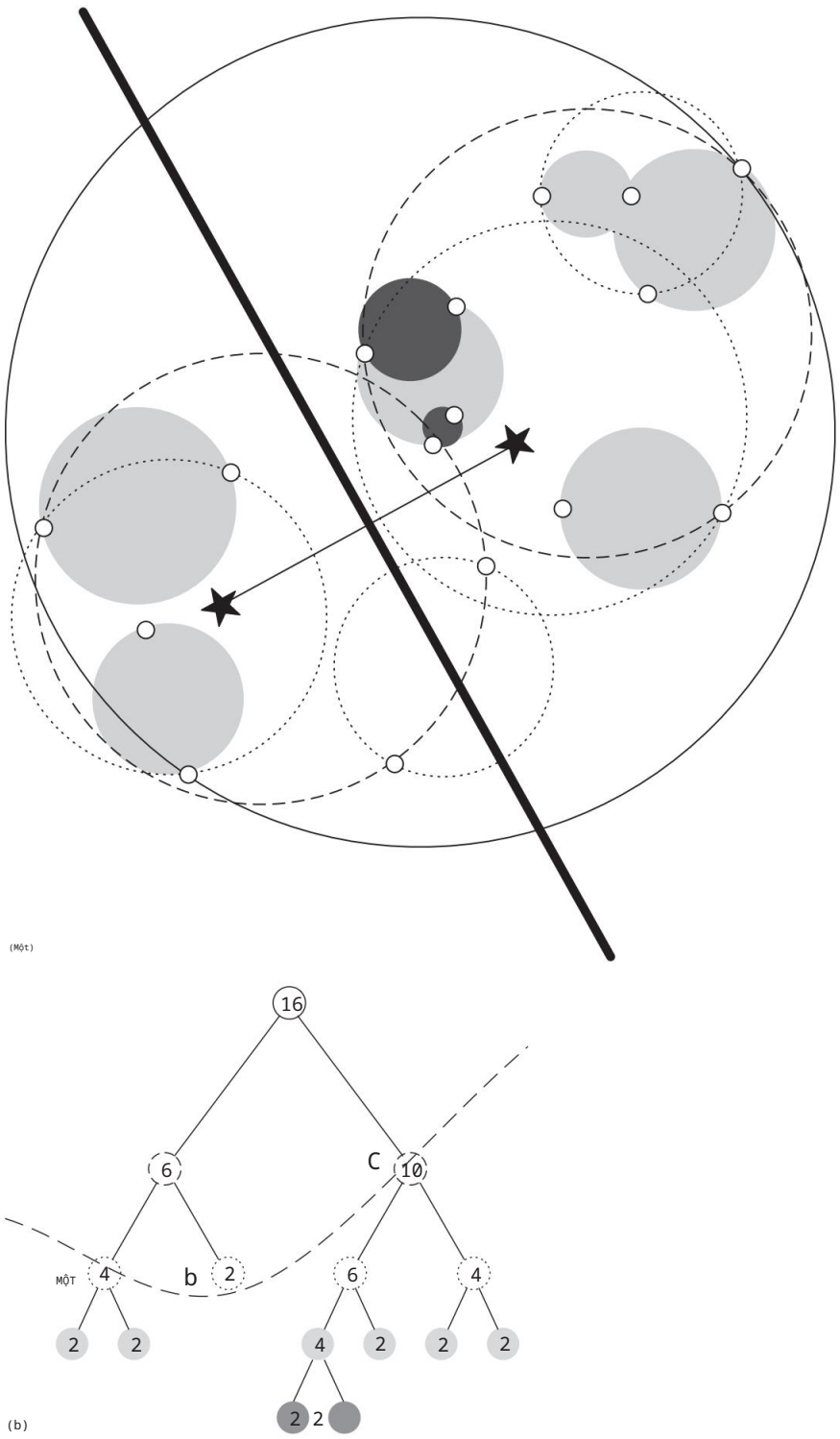
Cuộc thảo luận

Nhiều biến thể của quy trình k-means cơ bản đã được phát triển. Một số tạo ra một phân cụm theo thứ bậc bằng cách áp dụng thuật toán với k = 2 cho tập dữ liệu tổng thể và sau đó lặp lại, theo cách đệ quy, trong mỗi cụm.

Làm thế nào để bạn chọn k? Thông thường không có gì được biết về số lượng cụm có thể xảy ra và toàn bộ điểm của cụm là để tìm ra. Một cách là thử các giá trị khác nhau và chọn cái tốt nhất. Để làm được điều này, bạn cần học cách đánh giá sự thành công của máy học, đó là nội dung của Chương 5. Chúng tôi trở lại phân cụm trong Phần 6.6.

4.9 Đọc thêm

Kế hoạch 1R đã được đề xuất và nghiên cứu kỹ lưỡng bởi Holte (1993). Nó chưa bao giờ thực sự được dự định là một "phương pháp" học máy: mục đích là để chứng minh rằng các cấu trúc rất đơn giản làm cơ sở cho hầu hết các bộ dữ liệu thực tế được sử dụng để đánh giá các phương pháp học máy vào thời điểm đó và việc đưa các phương pháp suy luận quy nạp mạnh mẽ vào hoạt động trên các bộ dữ liệu đơn giản giống như dùng búa tạ để bẻ một chiếc đai ốc. Tại sao phải vật lộn với một cây quyết định phức tạp khi một quy tắc đơn giản sẽ làm được? Phương pháp tạo ra một quy tắc đơn giản cho mỗi lớp là kết quả công việc của Lucio de Souza Coelho ở Brazil và Len Trigg ở New Zealand, và nó được đặt tên là hyperpipes. Một thuật toán rất đơn giản, nó có ưu điểm là cực kỳ nhanh và khá khả thi ngay cả với số lượng thuộc tính khổng lồ.



Hình 4.16 Một cây bóng: (a) hai trung tâm cụm và đường phân chia của chúng và (b) cây phản hồi cor.

Bayes là một triết gia người Anh ở thế kỷ mười tám, người đã đặt ra lý thuyết xác suất của mình trong "Một bài luận hướng tới việc giải quyết một vấn đề trong học thuyết về cơ hội," được xuất bản trong Giao dịch triết học của Hiệp hội Hoàng gia Luân Đôn (Bayes 1763); quy tắc mang tên ông đã trở thành nền tảng của lý thuyết xác suất kể từ đó. Khó khăn với việc áp dụng quy tắc Bayes trong thực tế là việc gán xác suất trước. Một số nhà thống kê, được mệnh danh là Bayesian, coi quy tắc này là phúc âm và nhấn mạnh rằng mọi người phải nỗ lực nghiêm túc để ước tính chính xác các xác suất trước đó—mặc dù những ước tính như vậy thường mang tính chủ quan. Những người khác, không phải người theo trường phái Bayes, thích loại phân tích tự do trước thường tạo ra các khoảng tin cậy thống kê, mà chúng ta sẽ gặp trong chương tiếp theo. Với một tập dữ liệu cụ thể, các xác suất trước đó thường dễ ước tính một cách hợp lý, điều này khuyến khích cách tiếp cận Bayesian để học. Tuy nhiên, giả định về tính độc lập do phương pháp Naïve Bayes đưa ra là một trở ngại lớn, và một số nỗ lực đang được thực hiện để áp dụng phân tích Bayes mà không giả định về tính độc lập. Các mô hình kết quả được gọi là công trình mạng Bayesian (Heckerman et al. 1995), và chúng tôi mô tả chúng trong Phần 6.7.

Các kỹ thuật Bayes đã được sử dụng trong lĩnh vực nhận dạng mẫu (Duda và Hart 1973) trong 20 năm trước khi chúng được các nhà nghiên cứu máy học chấp nhận (ví dụ: xem Langley và cộng sự 1992) và được thực hiện để hoạt động trên các bộ dữ liệu có thuộc tính dư thừa (Langley và Sage 1994) và các thuộc tính số (John và Langley 1995). Cái tên Naïve Bayes thật đáng tiếc vì khó có thể sử dụng phương pháp này mà không cảm thấy đầu óc đơn giản. Tuy nhiên, không có gì ngây thơ về việc sử dụng nó trong những trường hợp thích hợp. Mô hình Naïve Bayes đa thức, đặc biệt thích hợp cho việc phân loại văn bản, đã được nghiên cứu bởi McCallum và Nigam (1998).

Bài viết kinh điển về cảm ứng cây quyết định là của Quinlan (1986), người đã mô tả thủ tục ID3 cơ bản được phát triển trong chương này. Mô tả toàn diện về phương pháp, bao gồm cả những cải tiến được thể hiện trong C4.5, xuất hiện trong một cuốn sách cổ điển của Quinlan (1993), đưa ra danh sách toàn bộ hệ thống C4.5, được viết bằng ngôn ngữ lập trình C. PRISM được phát triển bởi Cendrowska (1987), người cũng đã giới thiệu bộ dữ liệu kính áp tròng.

Luật kết hợp được giới thiệu và mô tả trong tài liệu về cơ sở dữ liệu hơn là trong tài liệu học máy. Ở đây, người ta nhấn mạnh rất nhiều vào việc xử lý lượng dữ liệu khổng lồ hơn là các cách nhạy cảm để kiểm tra và đánh giá các thuật toán trên các bộ dữ liệu hạn chế. Thuật toán được giới thiệu trong chương này là phương pháp Apriori được phát triển bởi Agrawal và các cộng sự của ông (Agrawal và cộng sự 1993a, 1993b; Agrawal và Srikant 1994). Một cuộc khảo sát về khai phá luật kết hợp xuất hiện trong một bài viết của Chen et al. (1996).

Hồi quy tuyến tính được mô tả trong hầu hết các văn bản thống kê tiêu chuẩn, và cách xử lý đặc biệt toàn diện có thể tìm thấy trong cuốn sách của Lawson và Hanson (1995). Việc sử dụng các mô hình tuyến tính để phân loại rất phổ biến trong những năm 1960; Nilsson (1965) cung cấp một tài liệu tham khảo tuyệt vời. Ông định nghĩa

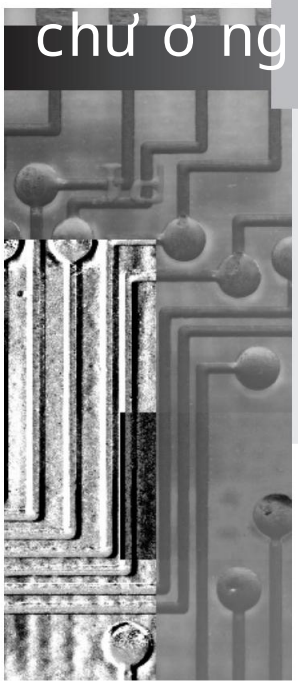
một đơn vị ngưỡng tuyến tính dưới dạng phép thử nhị phân xem một hàm tuyến tính lớn hơn hay nhỏ hơn 0 và một máy tuyến tính là một tập hợp các hàm tuyến tính, một hàm cho mỗi lớp, có giá trị cho một ví dụ chưa biết được so sánh và giá trị lớn nhất được chọn làm dự đoán của nó lớp học. Trong quá khứ xa xôi, các perceptron không còn được ưa chuộng khi xuất bản một cuốn sách có ảnh hưởng cho thấy chúng có những hạn chế cơ bản (Minsky và Papert 1969); tuy nhiên, các hệ thống hàm tuyến tính phức tạp hơn đã được hồi sinh trong những năm gần đây dưới dạng mạng thần kinh, được mô tả trong Phần 6.3. Thuật toán Winnow được giới thiệu bởi Nick Littlestone trong luận án tiến sĩ của ông vào năm 1989 (Littlestone 1988, 1989). Các bộ phân loại tuyến tính đa phản hồi gần đây đã tìm thấy một ứng dụng mới cho một hoạt động được gọi là xếp chồng kết hợp đầu ra của các thuật toán học tập khác, được mô tả trong Chương 7 (xem Wolpert 1992). Friedman (1996) mô tả kỹ thuật phân loại theo cặp, Fürnkranz (2002) phân tích kỹ hơn về kỹ thuật này, và Hastie và Tibshirani (1998) mở rộng nó để ước tính xác suất bằng cách sử dụng ghép cặp theo cặp.

Fix và Hodges (1951) thực hiện phép phân tích đầu tiên của phương pháp láng giềng gần nhất, và Johns (1961) đi tiên phong trong việc sử dụng phương pháp này trong các bài toán phân loại. Cover và Hart (1967) đã thu được kết quả lý thuyết cổ điển rằng, đối với các bộ dữ liệu đủ lớn, xác suất lỗi của nó không bao giờ vượt quá hai lần mức tối thiểu theo lý thuyết; Devroye và cộng sự. (1996) chỉ ra rằng k-láng giềng gần nhất là tối ưu tiệm cận đối với k lớn và n với k/n Æ 0. Phương pháp lân cận gần nhất đã trở nên phổ biến trong học máy nhờ công trình của Aha (1992), người đã chỉ ra rằng học tập dựa trên ví dụ có thể kết hợp với việc cắt tỉa mẫu mực ồn ào và trọng số thuộc tính và các phương pháp thu được hoạt động tốt so với các phương pháp học tập khác. Chúng ta sẽ nhắc lại vấn đề này trong Chương 6.

Cấu trúc dữ liệu kD-tree được phát triển bởi Friedman et al. (1977). Mô tả của chúng tôi bám sát lời giải thích do Andrew Moore đưa ra trong luận án tiến sĩ của mình (Moore 1991), người cùng với Omohundro (1987) đã đi tiên phong trong việc sử dụng nó trong học máy. Moore (2000) mô tả những cách phức tạp để xây dựng cây bóng hoạt động tốt ngay cả với hàng nghìn thuộc tính. Chúng tôi lấy ví dụ về cây bóng từ bài giảng của Alexander Gray của Đại học Carnegie-Mellon. Phương pháp khoảng thời gian của tính năng biểu quyết được đề cập trong phần phụ Thảo luận ở cuối Phần 4.7 được mô tả bởi Demiroz và Guvenir (1997).

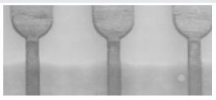
Thuật toán k-means là một kỹ thuật cổ điển, có nhiều mô tả và biến thể (ví dụ, xem Hartigan 1975). Việc sử dụng cây kD một cách thông minh để tăng tốc độ phân cụm k-mean, mà chúng tôi đã chọn để minh họa bằng cách sử dụng cây bóng thay thế, đã được Moore và Pelleg (2000) đi tiên phong trong thuật toán phân cụm X-mean của họ. Thuật toán đó cũng chứa một số đổi mới khác, được mô tả trong Phần 6.6.

chư ơ ng 5



Sự uy tín:

Đánh giá những gì đã đư ợc học



Đánh giá là chìa khóa để đạt đư ợc tiến bộ thực sự trong khai thác dữ liệu. Có rất nhiều cách suy luận cấu trúc từ dữ liệu: chúng ta đã gặp nhiều cách rồi và sẽ thấy các cải tiến tiếp theo cũng như các phư ơ ng pháp mới trong chư ơ ng tiếp theo. Như ng để xác định nên sử dụng phư ơ ng pháp nào cho một vấn đề cụ thể, chúng ta cần các cách có hệ thống để đánh giá cách thức hoạt động của các phư ơ ng pháp khác nhau và so sánh phư ơ ng pháp này với phư ơ ng pháp khác. Việc đánh giá không đơ n giản như thoạt nhìn.

Vấn đề là gì? Chúng tôi có bộ đào tạo; chắc chắn chúng ta chỉ có thể xem các phư ơ ng pháp khác nhau hoạt động tốt như thế nào đối với điều đó. Chà, không: như chúng ta sẽ sớm thấy, hiệu suất trên tập huấn luyện chắc chắn không phải là một chỉ báo tốt về hiệu suất trên tập kiểm tra độc lập. Chúng tôi cần các cách dự đoán giới hạn hiệu suất trong thực tế, dựa trên các thử nghiệm với bất kỳ dữ liệu nào có thể thu đư ợc.

Khi có sẵn một lư ợng lớn dữ liệu, điều này không thành vấn đề: chỉ cần tạo một mô hình dựa trên một tập huấn luyện lớn và thử nó trên một tập kiểm tra lớn khác. Như ng mặc dù việc khai thác dữ liệu đôi khi liên quan đến “dữ liệu lớn”—đặc biệt là trong các ứng dụng tiếp thị, bán hàng và hỗ trợ khách hàng—dữ liệu, dữ liệu chất lư ợng, thư ờng khan hiếm. Các vết dầu loang đư ợc đề cập trong Chư ơ ng 1 (trang 23–24) phải đư ợc phát hiện

và được đánh dấu thủ công—một quy trình đòi hỏi nhiều kỹ năng và lao động—trước khi được sử dụng làm dữ liệu huấn luyện. Ngay cả trong ứng dụng thẻ tín dụng (trang 22–23), hóa ra chỉ có 1000 ví dụ đào tạo thuộc loại phù hợp. Dữ liệu cung cấp điện cho thành phố (trang 24–25) đã quay ngược lại 15 năm, 5000 ngày—nhưng chỉ có 15 Ngày Giáng sinh và Lễ Tạ ơn, và chỉ 4 ngày 29 tháng 2 và các cuộc bầu cử tổng thống. Ứng dụng chẩn đoán cơ điện (trang 25–26) đã có thể tận dụng 20 năm kinh nghiệm được ghi lại, nhưng điều này chỉ mang lại 300 ví dụ lỗi có thể sử dụng được. Các ứng dụng tiếp thị và bán hàng (trang 26–28) chắc chắn liên quan đến dữ liệu lớn, nhưng nhiều ứng dụng khác thì không: dữ liệu đào tạo thường dựa vào kiến thức chuyên môn của con người—và điều đó luôn bị thiếu hụt.

Câu hỏi về dự đoán hiệu suất dựa trên dữ liệu hạn chế là một câu hỏi thú vị và vẫn còn gây tranh cãi. Chúng ta sẽ gặp nhiều kỹ thuật khác nhau, trong đó một kỹ thuật—xác thực chéo lặp lại—đang đạt được ưu thế và có lẽ là phương pháp đánh giá được lựa chọn trong hầu hết các tình huống dữ liệu hạn chế thực tế.

So sánh hiệu suất của các phương pháp học máy khác nhau đối với một vấn đề nhất định là một vấn đề khác không dễ dàng như người ta tưởng: để chắc chắn rằng sự khác biệt rõ ràng không phải do hiệu ứng ngẫu nhiên gây ra, cần có các bài kiểm tra thống kê. Cho đến nay, chúng tôi đã ngầm giả định rằng những gì đang được dự đoán là khả năng phân loại các trường hợp thử nghiệm một cách chính xác; tuy nhiên, một số tình huống liên quan đến việc dự đoán xác suất của lớp thay vì bản thân các lớp và những trường hợp khác liên quan đến việc dự đoán các giá trị số thay vì giá trị danh nghĩa. Các phương pháp khác nhau là cần thiết trong từng trường hợp.

Sau đó, chúng tôi xem xét câu hỏi về chi phí. Trong hầu hết các tình huống khai thác dữ liệu thực tế, chi phí của một lỗi phân loại sai phụ thuộc vào loại lỗi đó - ví dụ, liệu một ví dụ tích cực đã bị phân loại sai thành tiêu cực hay ngược lại. Khi thực hiện khai thác dữ liệu và đánh giá hiệu suất của nó, điều cần thiết là phải tính đến các chi phí này. May mắn thay, có những kỹ thuật đơn giản để làm cho hầu hết các sơ đồ học tập trở nên nhạy cảm với chi phí mà không phải vật lộn với nội bộ của thuật toán. Cuối cùng, toàn bộ khái niệm đánh giá có những mối liên hệ triết học hấp dẫn. Trong 2000 năm, các nhà triết học đã tranh luận về câu hỏi làm thế nào để đánh giá các lý thuyết khoa học và các vấn đề này được khai thác dữ liệu làm trọng tâm bởi vì những gì được trích xuất về cơ bản là một "lý thuyết" của dữ liệu.

5.1 Huấn luyện và kiểm tra

Đối với các bài toán phân loại, việc đo lường hiệu suất của bộ phân loại theo tỷ lệ lỗi là điều đương nhiên. Trình phân loại dự đoán lớp của từng trường hợp: nếu đúng, điều đó được tính là thành công; nếu không, đó là một lỗi. Tỷ lệ lỗi chỉ là tỷ lệ lỗi được thực hiện trên toàn bộ các phiên bản và nó đo lường hiệu suất tổng thể của trình phân loại.

Tất nhiên, điều chúng tôi quan tâm là hiệu suất có thể xảy ra trong tương lai trên dữ liệu mới, chứ không phải hiệu suất trước đây trên dữ liệu cũ. Chúng ta đã biết phân loại

của từng trư ờng hợp trong tập huấn luyện, đó là lý do tại sao chúng ta có thể sử dụng nó để huấn luyện. Chúng tôi thư ờng không quan tâm đến việc tìm hiểu về các phân loại đó— mặc dù chúng tôi có thể quan tâm nếu mục đích của chúng tôi là làm sạch dữ liệu hơ n là dự đoán.
Vì vậy, câu hỏi đặt ra là tỷ lệ lỗi trên dữ liệu cũ có thể là một chỉ báo tốt về tỷ lệ lỗi trên dữ liệu mới không? Câu trả lời chắc chắn là không—không phải nếu dữ liệu cũ đư ợc sử dụng trong quá trình học để huấn luyện bộ phân loại.

Đây là một thực tế đáng ngạc nhiên, và là một thực tế rất quan trọng. Tỷ lệ lỗi trên tập huấn luyện không có khả năng là một chỉ báo tốt về hiệu suất trong tư ơ ng lai. Tại sao? Bởi vì bộ phân loại đã đư ợc học từ cùng một dữ liệu đào tạo, mọi ư ớc tính về hiệu suất dựa trên dữ liệu đó sẽ lạc quan và có thể lạc quan một cách vô vọng.

Chúng ta đã thấy một ví dụ về điều này trong bộ dữ liệu quan hệ lao động. Hình 1.3(b) đư ợc tạo trực tiếp từ dữ liệu huấn luyện và Hình 1.3(a) đư ợc lấy từ nó bằng một quá trình cắt tỉa. Cái trư ớc có thể có tốc độ chính xác hơ n đối với dữ liệu đư ợc sử dụng để huấn luyện bộ phân loại như ng có thể sẽ hoạt động kém hơ n đối với dữ liệu thử nghiệm độc lập vì nó đư ợc khớp quá mức với dữ liệu huấn luyện.
Cây đầu tiên sẽ trông đẹp theo tỷ lệ lỗi trên dữ liệu huấn luyện, tốt hơ n cây thứ hai. Như ng điều này không phản ánh cách họ sẽ thực hiện trên dữ liệu thử nghiệm độc lập.

Tỷ lệ lỗi trên dữ liệu huấn luyện đư ợc gọi là lỗi thay thế lại, bởi vì nó đư ợc tính bằng cách thay thế lại các cá thể huấn luyện vào một bộ phân loại đư ợc xây dựng từ chúng. Mặc dù nó không phải là một yếu tố dự đoán đáng tin cậy về tỷ lệ lỗi thực sự trên dữ liệu mới, tuy nhiên, nó thư ờng hữu ích khi biết.

Để dự đoán hiệu suất của bộ phân loại trên dữ liệu mới, chúng ta cần đánh giá tỷ lệ lỗi của nó trên tập dữ liệu không đóng vai trò gì trong việc hình thành bộ phân loại. Tập dữ liệu độc lập này đư ợc gọi là tập kiểm tra. Chúng tôi giả định rằng cả dữ liệu huấn luyện và dữ liệu thử nghiệm đều là các mẫu đại diện cho vấn đề tiềm ẩn.

Trong một số trư ờng hợp, dữ liệu thử nghiệm có thể khác biệt về bản chất với dữ liệu huấn luyện. Ví dụ, xem xét vấn đề rủi ro tín dụng ở Phần 1.3. Giả sử ngân hàng có dữ liệu đào tạo từ các chi nhánh ở Thành phố New York và Florida và muốn biết một bộ phân loại đư ợc đào tạo về một trong những bộ dữ liệu này sẽ hoạt động tốt như thế nào ở một chi nhánh mới ở Nebraska. Có lẽ nên sử dụng dữ liệu của Florida làm dữ liệu thử nghiệm để đánh giá bộ phân loại do New York đào tạo và dữ liệu của New York để đánh giá bộ phân loại do Florida đào tạo. Nếu các bộ dữ liệu đư ợc hợp nhất trư ớc khi đào tạo, thì hiệu suất trên dữ liệu thử nghiệm có thể không phải là chỉ báo tốt về hiệu suất trên dữ liệu trong tư ơ ng lai ở trạng thái hoàn toàn khác.

Điều quan trọng là dữ liệu thử nghiệm không đư ợc sử dụng theo bất kỳ cách nào để tạo ra bộ xác định lớp. Ví dụ: một số phư ơ ng pháp học bao gồm hai giai đoạn, một là tìm ra cấu trúc cơ bản và giai đoạn thứ hai là tối ư u hóa các tham số liên quan đến cấu trúc đó và có thể cần các bộ dữ liệu riêng biệt trong hai giai đoạn. Hoặc bạn có thể thử một số lư ợc đồ học tập trên dữ liệu huấn luyện và sau đó đánh giá chúng—dĩ nhiên là trên một tập dữ liệu mới—để xem cái nào hoạt động tốt nhất. Như ng không ai trong số

dữ liệu này có thể được sử dụng để xác định ước tính về tỷ lệ lỗi trong tương lai. Trong những tình huống như vậy, người ta thường nói về ba bộ dữ liệu: **dữ liệu huấn luyện** , **dữ liệu xác nhận** và **dữ liệu thử nghiệm** . Dữ liệu đào tạo được sử dụng bởi một hoặc nhiều phương pháp học tập để đưa ra các bộ phân loại. Dữ liệu xác thực được sử dụng để tối ưu hóa các tham số của các bộ phân loại đó hoặc để chọn một bộ phân loại cụ thể. Sau đó, dữ liệu thử nghiệm được sử dụng để tính toán tỷ lệ lỗi của phương pháp cuối cùng, được tối ưu hóa. Mỗi bộ trong số ba bộ phải được chọn độc lập: bộ xác nhận phải khác với bộ huấn luyện để đạt được hiệu suất tốt trong giai đoạn tối ưu hóa hoặc lựa chọn và bộ kiểm tra phải khác với cả hai để có được ước tính đáng tin cậy về giá trị thực. tỷ lệ lỗi.

Có thể là khi tỷ lệ lỗi đã được xác định, dữ liệu thử nghiệm sẽ được gộp lại vào dữ liệu huấn luyện để tạo ra một bộ phân loại mới để sử dụng thực tế.
Không có gì sai với điều này: nó chỉ là một cách để tối đa hóa lượng dữ liệu được sử dụng để tạo bộ phân loại sẽ thực sự được sử dụng trong thực tế.
Điều quan trọng là tỷ lệ lỗi không được trích dẫn dựa trên bất kỳ dữ liệu nào trong số này. Ngoài ra, khi dữ liệu xác thực đã được sử dụng—có thể để xác định loại sơ đồ học tập tốt nhất sẽ sử dụng—thì dữ liệu đó có thể được gộp lại vào dữ liệu huấn luyện để đào tạo lại sơ đồ học tập đó, tối đa hóa việc sử dụng dữ liệu.

Nếu có sẵn nhiều dữ liệu thì không có vấn đề gì: chúng tôi lấy một mẫu lớn và sử dụng nó để đào tạo; sau đó, một mẫu lớn độc lập khác của dữ liệu khác nhau và sử dụng nó để thử nghiệm. Với điều kiện là cả hai mẫu đều mang tính đại diện, tỷ lệ lỗi trên bộ thử nghiệm sẽ đưa ra dấu hiệu thực sự về hiệu suất trong tương lai. Nói chung, mẫu đào tạo càng lớn thì bộ phân loại càng tốt, mặc dù lợi nhuận bắt đầu giảm khi vượt quá một khối lượng dữ liệu đào tạo nhất định. Và mẫu thử nghiệm càng lớn thì ước tính sai số càng chính xác. Độ chính xác của ước tính lỗi có thể được định lượng theo thống kê, như chúng ta sẽ thấy trong phần tiếp theo.

Vấn đề thực sự xảy ra khi không có nguồn cung cấp dữ liệu lớn. Trong nhiều tình huống, dữ liệu huấn luyện phải được phân loại thủ công—dĩ nhiên, dữ liệu kiểm tra cũng phải như vậy để có được các ước tính lỗi. Điều này giới hạn lượng dữ liệu có thể được sử dụng để đào tạo, xác thực và kiểm tra và vấn đề trở thành cách tận dụng tối đa tập dữ liệu hạn chế. Từ bộ dữ liệu này, một lượng nhất định được giữ lại để thử nghiệm—đây được gọi là **quy trình giữ lại** — và phần còn lại được sử dụng để đào tạo (và, nếu cần, một phần trong số đó được dành để xác thực).
Có một vấn đề nan giải ở đây: để tìm một bộ phân loại tốt, chúng tôi muốn sử dụng càng nhiều dữ liệu càng tốt để đào tạo; để có được ước tính lỗi tốt, chúng tôi muốn sử dụng càng nhiều càng tốt để thử nghiệm. Phần 5.3 và 5.4 xem xét các phương pháp được sử dụng rộng rãi để giải quyết tình trạng tiến thoái lưỡng nan này.

5.2 Dự đoán hiệu suất

Giả sử chúng ta đo lỗi của một bộ phân loại trên một tập kiểm tra và thu được một tỷ lệ lỗi số nhất định—chẳng hạn là 25%. Trên thực tế, trong phần này, chúng tôi đề cập đến tỷ lệ thành công

thay vì tỷ lệ lỗi, vì vậy tỷ lệ này tương ứng với tỷ lệ thành công là 75%. Bây giờ, đây chỉ là một ước tính. Bạn có thể nói gì về tỷ lệ thành công thực sự trên dân số mục tiêu? Chắc chắn, nó được mong đợi là gần 75%. Nhưng gần đến mức nào — trong vòng 5%?
Trong vòng 10%? Nó phải phụ thuộc vào kích thước của tập kiểm tra. Đương nhiên, chúng tôi sẽ tự tin hơn về con số 75% nếu nó dựa trên tập hợp thử nghiệm gồm 10.000 phiên bản thay vì dựa trên tập hợp thử nghiệm gồm 100 phiên bản. Nhưng chúng ta sẽ tự tin hơn bao nhiêu?

Để trả lời những câu hỏi này, chúng ta cần một số suy luận thống kê. Trong thống kê, một chuỗi các sự kiện độc lập thành công hoặc thất bại được gọi là quá trình Bernoulli. Ví dụ kinh điển là tung đồng xu. Mỗi lần tung là một sự kiện độc lập.
Giả sử chúng ta luôn dự đoán mặt ngửa; nhưng thay vì "ngửa" hay "sấp", mỗi lần tung được coi là "thành công" hay "thất bại". Giả sử đồng xu bị thiên vị, nhưng chúng ta không biết xác suất mặt ngửa là bao nhiêu. Sau đó, nếu chúng ta thực sự tung đồng xu 100 lần và 75 trong số đó là mặt ngửa, thì chúng ta có một tình huống giống như tình huống được mô tả trước đây cho một bộ phân loại với tỷ lệ thành công được quan sát là 75% trên một bộ thử nghiệm. Chúng ta có thể nói gì về xác suất thành công thực sự? Nói cách khác, hãy tưởng tượng rằng có một quy trình Bernoulli—một đồng xu bị thiên lệch—có tỷ lệ thành công thực sự (nhưng chưa biết) là p. Giả sử rằng trong số N thử nghiệm, S là thành công: do đó tỷ lệ thành công quan sát được là f = S/N. Câu hỏi đặt ra là, điều này cho bạn biết điều gì về tỷ lệ thành công thực sự p?

Câu trả lời cho câu hỏi này thường được thể hiện dưới dạng khoảng tin cậy; nghĩa là, p nằm trong một khoảng xác định nhất định với độ tin cậy xác định nhất định.
Ví dụ: nếu quan sát thấy S = 750 thành công trong số N = 1000 thử nghiệm, thì điều này cho thấy tỷ lệ thành công thực sự phải vào khoảng 75%. Nhưng làm thế nào gần với 75%? Hóa ra với độ tin cậy 80%, tỷ lệ thành công thực p nằm trong khoảng từ 73,2% đến 76,7%. Nếu quan sát thấy S = 75 thành công trong số N = 100 thử nghiệm, thì điều này cũng cho thấy rằng tỷ lệ thành công thực sự phải vào khoảng 75%. Nhưng thử nghiệm nhỏ hơn và khoảng tin cậy 80% cho p rộng hơn, trải dài từ 69,1% đến 80,1%.

Những số liệu này dễ liên quan đến định tính, nhưng làm thế nào chúng có được từ định lượng? Chúng tôi lập luận như sau: giá trị trung bình và phương sai của một thử nghiệm Bernoulli với tỷ lệ thành công p lần lượt là p và p(1 - p). Nếu N thử nghiệm được lấy từ quy trình Bernoulli, thì tỷ lệ thành công dự kiến f = S/N là một biến ngẫu nhiên có cùng giá trị trung bình p; phương sai được giảm theo hệ số N thành p(1 - p)/N. Đối với N lớn , phân phối của biến ngẫu nhiên này tiệm cận với phân phối chuẩn. Đây là tất cả các sự kiện của thống kê: chúng ta sẽ không đi sâu vào cách chúng được bắt nguồn.

Xác suất để một biến ngẫu nhiên X, với giá trị trung bình bằng 0, nằm trong một phạm vi tin cậy nhất định của chiều rộng 2z là

Pr[ ] - £ £ zXz c = ·

Đối với phân phối chuẩn, các giá trị của c và các giá trị tương ứng của z được đưa ra trong các bảng được in ở mặt sau của hầu hết các văn bản thống kê. Tuy nhiên, các bảng quy ước có dạng hơi khác: chúng mang lại sự tin tưởng rằng X sẽ

nằm ngoài phạm vi và họ chỉ cung cấp cho phần trên của phạm vi:

trước [] Xz ≥·

Đây được gọi là xác suất một đầu vì nó chỉ đề cập đến "đuôi" phía trên của phân phối. Phân phối bình thường là đối xứng, vì vậy xác suất cho đuôi dưới

trước [] X z £ -

đều giống nhau.

Bảng 5.1 đưa ra một ví dụ. Giống như các bảng phân phối chuẩn khác, bảng này giả định rằng biến ngẫu nhiên X có trung bình bằng 0 và phương sai bằng 1. Ngoài ra, bạn có thể nói rằng các số liệu z được đo theo độ lệch chuẩn so với giá trị trung bình. Do đó, con số cho Pr[X ≥ z] = 5% ngụ ý rằng có một 5% khả năng X nằm trên 1,65 độ lệch chuẩn so với giá trị trung bình. Bởi vì phân phối là đối xứng, cơ hội mà X nằm trên 1,65 độ lệch chuẩn so với giá trị trung bình (trên hoặc dưới) là 10% hoặc

Pr[ ] ££ 1 65 1 65 90 %.

Tất cả những gì chúng ta cần làm bây giờ là rút gọn biến ngẫu nhiên f để có trung bình và đơn vị bằng không. phương sai. Chúng tôi làm điều này bằng cách trừ giá trị trung bình p và chia cho tiêu chuẩn độ lệch $\sqrt{p(1-p)/N}$. Điều này dẫn đến

$$\Pr\left[-z < \frac{f-p}{\sqrt{p(1-p)/N}} < z\right] = c.$$

Bây giờ đây là quy trình tìm giới hạn tin cậy. Với một con số tin cậy cụ thể c, hãy tham khảo Bảng 5.1 để biết giá trị z tương ứng . Để sử dụng bảng trước tiên bạn sẽ phải trừ c từ 1 và sau đó chia đôi kết quả, sao cho c = 90% bạn sử dụng mục nhập bảng cho 5%. Nội suy tuyến tính có thể được sử dụng cho liên

Bảng 5.1 Giới hạn tin cậy cho phân phối bình thường.

Pr[X ≥ z]	z
0,1%	3.09
0,5%	2,58
1%	2,33
5%	1,65
10%	1,28
20%	0,84
40%	0,25

trung gian mức độ tin cậy. Sau đó viết bất đẳng thức trong biểu thức trư ớc dư ới dạng đẳng thức và đảo ngư ợc nó để tìm biểu thức cho p.

Bư ớc cuối cùng liên quan đến việc giải một phư ơ ng trình bậc hai. Mặc dù không khó để thực hiện, như ng nó dẫn đến một biểu hiện ghê gớm khó chịu đối với giới hạn độ tin cậy:

$$p = \frac{f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

± trong biểu thức này đư a ra hai giá trị cho p đại diện cho ranh giới độ tin cậy trên và dư ới. Mặc dù công thức có vẻ phức tạp như ng không khó để tìm ra trong các trư ờng hợp cụ thể.

Kết quả này có thể đư ợc sử dụng để lấy các giá trị trong ví dụ số trư ớc đó. Đặt f = 75%, N = 1000 và c = 80% (sao cho z = 1,28) dẫn đến khoảng [0,732,0,767] cho p và N = 100 dẫn đến [0,691,0,801] cho cùng một mức sự tự tin. Lư u ý rằng giả định phân phối chuẩn chỉ có giá trị đối với N lớn (giả sử N > 100). Do đó, f = 75% và N = 10 dẫn đến các giới hạn tin cậy [0,549,0,881]—như ng những giới hạn này nên đư ợc coi là muối bỏ bể.

5.3 Xác thực chéo

Bây giờ hãy cân nhắc xem phải làm gì khi lư ợng dữ liệu dành cho huấn luyện và kiểm tra bị hạn chế. Phư ơ ng pháp giữ lại dự trữ một số tiền nhất định để thử nghiệm và sử dụng phần còn lại để đào tạo (và dành một phần trong số đó để xác thực, nếu cần). Về mặt thực tế, thông thư ờng sẽ giữ lại một phần ba dữ liệu để thử nghiệm và sử dụng hai phần ba còn lại để đào tạo.

Tất nhiên, bạn có thể không may mắn: mẫu đư ợc sử dụng để đào tạo (hoặc thử nghiệm) có thể không mang tính đại diện. Nói chung, bạn không thể biết liệu một mẫu có phản hồi hay không. Như ng có một kiểm tra đơ n giản có thể đáng giá: mỗi lớp trong tập dữ liệu đầy đủ phải đư ợc biểu diễn theo tỷ lệ phù hợp trong tập huấn luyện và tập kiểm tra. Nếu không may, tất cả các ví dụ với một lớp nhất định bị thiếu trong tập huấn luyện, bạn khó có thể mong đợi một bộ phân loại học đư ợc từ dữ liệu đó để thực hiện tốt các ví dụ của lớp đó—và tình hình sẽ trở nên trầm trọng hơ n bởi thực tế là lớp nhất thiết phải đư ợc gửi quá mức trong tập kiểm tra vì không có phiên bản nào của nó đư ợc đư a vào tập huấn luyện! Thay vào đó, bạn nên đảm bảo rằng việc lấy mẫu ngẫu nhiên đư ợc thực hiện theo cách đảm bảo rằng mỗi lớp đư ợc thể hiện đúng trong cả tập huấn luyện và tập kiểm tra. Thủ tục này đư ợc gọi là phân tầng, và chúng ta có thể nói về sự nắm giữ phân tầng. Mặc dù nói chung là rất đáng làm, như ng phân tầng chỉ cung cấp một biện pháp bảo vệ sơ khai chống lại sự thể hiện không đồng đều trong các tập huấn luyện và kiểm tra.

Một cách tổng quát hơ n để giảm thiểu bất kỳ sai lệch nào do mẫu cụ thể đư ợc chọn để loại trừ gây ra là lặp lại toàn bộ quy trình, đào tạo và thử nghiệm, nhiều lần với các mẫu ngẫu nhiên khác nhau. Trong mỗi lần lặp, một tỷ lệ nhất định—

giả sử hai phần ba—dữ liệu được chọn ngẫu nhiên để đào tạo, có thể có phân tầng và phần còn lại được sử dụng để thử nghiệm. Tỷ lệ lỗi trên các lần lặp lại khác nhau được tính trung bình để mang lại tỷ lệ lỗi tổng thể. Đây là phương pháp ước tính tỷ lệ lỗi lặp đi lặp lại .

Trong một quy trình loại bỏ duy nhất, bạn có thể cân nhắc hoán đổi vai trò của dữ liệu thử nghiệm và dữ liệu huấn luyện—nghĩa là huấn luyện hệ thống trên dữ liệu thử nghiệm và kiểm tra hệ thống trên dữ liệu huấn luyện—và tính trung bình hai kết quả, do đó giảm tác động của việc biểu diễn không đồng đều trong tập huấn luyện và kiểm tra. Thật không may, điều này chỉ thực sự hợp lý với tỷ lệ phân chia 50: 50 giữa dữ liệu huấn luyện và kiểm tra, điều này thường không lý tưởng—tốt hơn là sử dụng hơn một nửa dữ liệu để huấn luyện ngay cả khi phải trả giá bằng dữ liệu kiểm tra. Tuy nhiên, một biến thể đơn giản tạo thành cơ sở của một kỹ thuật thống kê quan trọng được gọi là xác thực chéo. Trong xác thực chéo, bạn quyết định số lượng nếp gấp hoặc phân vùng cố định của dữ liệu. Giả sử chúng ta sử dụng ba. Sau đó, dữ liệu được chia thành ba phần xấp xỉ bằng nhau và mỗi phần lần lượt được sử dụng để kiểm tra và phần còn lại được sử dụng để huấn luyện. Nghĩa là, sử dụng hai phần ba để huấn luyện và một phần ba để kiểm tra và lặp lại quy trình ba lần để cuối cùng, mọi phiên bản đều được sử dụng chính xác một lần để kiểm tra. Điều này được gọi là xác thực chéo gấp ba lần và nếu sự phân tầng cũng được chấp nhận—điều thường xảy ra—thì nó được gọi là xác thực chéo ba lần phân tầng.

Cách tiêu chuẩn để dự đoán tỷ lệ lỗi của một kỹ thuật học tập được cung cấp một mẫu dữ liệu cố định, duy nhất là sử dụng xác thực chéo 10 lần được phân tầng. Dữ liệu được chia ngẫu nhiên thành 10 phần trong đó lớp được biểu diễn theo tỷ lệ gần giống như trong tập dữ liệu đầy đủ. Mỗi phần được tổ chức lần lượt và kế hoạch học tập được đào tạo trên chín phần mười còn lại; sau đó tỷ lệ lỗi của nó được tính trên tập hợp giữ lại. Do đó, thủ tục học tập được thực hiện tổng cộng 10 lần trên các tập huấn luyện khác nhau (mỗi tập có nhiều điểm chung). Cuối cùng, 10 ước tính lỗi được tính trung bình để mang lại ước tính lỗi tổng thể.

Tại sao lại là 10? Các thử nghiệm mở rộng trên nhiều bộ dữ liệu, với các kỹ thuật học tập khác nhau, đã chỉ ra rằng 10 là số lần gấp phù hợp để có được ước tính sai số tốt nhất và cũng có một số bằng chứng lý thuyết chứng minh điều này.
Mặc dù những lập luận này không có nghĩa là kết luận và cuộc tranh luận vẫn tiếp tục nổ ra trong giới khai thác dữ liệu và máy học về kế hoạch tốt nhất để đánh giá là gì, xác thực chéo 10 lần đã trở thành phương pháp tiêu chuẩn trong điều kiện thực tế. Các thử nghiệm cũng chỉ ra rằng việc sử dụng phân tầng cải thiện kết quả một chút. Do đó, kỹ thuật đánh giá tiêu chuẩn trong các tình huống chỉ có sẵn dữ liệu hạn chế là xác thực chéo 10 lần phân tầng. Lưu ý rằng cả việc phân tầng và chia thành 10 nếp gấp đều không cần phải chính xác: chỉ cần chia dữ liệu thành 10 tập hợp xấp xỉ bằng nhau trong đó các giá trị lớp khác nhau được biểu diễn theo tỷ lệ xấp xỉ phù hợp. Đánh giá thống kê không phải là một khoa học chính xác. Hơn nữa, không có gì kỳ diệu về con số chính xác 10: xác thực chéo 5 lần hoặc 20 lần có khả năng gần như tốt.

Xác thực chéo 10 lần duy nhất có thể không đủ để nhận được lỗi đáng tin cậy ước lượng. Các thử nghiệm xác thực chéo 10 lần khác nhau với cùng một cách học phương pháp và tập dữ liệu thường tạo ra các kết quả khác nhau, do ảnh hưởng của sự thay đổi ngẫu nhiên trong việc chọn chính các nếp gấp. Sự phân tầng làm giảm biến thể, nhưng nó chắc chắn không loại bỏ nó hoàn toàn. Khi tìm kiếm ước tính lỗi tỷ lệ chính xác, quy trình tiêu chuẩn là lặp lại xác thực chéo xử lý 10 lần—nghĩa là xác thực chéo 10 lần 10 lần—và tính trung bình kết quả. Điều này liên quan đến việc gọi thuật toán học 100 lần trên các bộ dữ liệu đều bằng chín phần mười kích thước của bản gốc. Có được một thước đo tốt về hình thức là một công việc đòi hỏi nhiều tính toán.

5.4 Các ước tính khác

Xác thực chéo gấp mười lần là cách tiêu chuẩn để đo tỷ lệ lỗi của một sơ đồ học tập trên một tập dữ liệu cụ thể; cho kết quả đáng tin cậy, 10 lần 10 lần xác thực chéo. Nhưng nhiều phương pháp khác được sử dụng để thay thế. Hai đặc biệt phổ biến là xác thực chéo bỏ qua một lần và bootstrap.

Bỏ đi một lần

Xác thực chéo loại bỏ một lần chỉ đơn giản là xác thực chéo n lần, trong đó n là số lượng trường hợp trong tập dữ liệu. Lần lượt từng trường hợp bị loại bỏ, và phương pháp học tập được đào tạo trên tất cả các trường hợp còn lại. Nó được đánh giá bởi tính chính xác của nó trong trường hợp còn lại—một hoặc không đối với thành công hay thất bại, tương ứng. Kết quả của tất cả n phán đoán, một cho mỗi thành viên của tập dữ liệu, là được tính trung bình và mức trung bình đó đại diện cho ước tính lỗi cuối cùng.

Thủ tục này là một trong những hấp dẫn vì hai lý do. Đầu tiên, lượng dữ liệu lớn nhất có thể được sử dụng để huấn luyện trong từng trường hợp, điều này có lẽ sẽ tăng lên cơ hội mà bộ phân loại là một bộ phân loại chính xác. Thứ hai, thủ tục được xác định tối thiểu: không có lấy mẫu ngẫu nhiên. Không có điểm nào trong việc lặp lại nó 10 hoặc lặp đi lặp lại tất cả: mỗi lần sẽ thu được cùng một kết quả. Thiết lập chống lại đây là chi phí tính toán cao, bởi vì toàn bộ thủ tục học tập phải được thực hiện n lần và điều này thường không khả thi đối với các tập dữ liệu lớn. Tuy nhiên, loại bỏ một lần dường như mang lại cơ hội vắt kiệt tối đa một tập dữ liệu nhỏ và thu được ước tính chính xác nhất có thể.

Nhưng có một nhược điểm đối với xác thực chéo một lần, ngoài chi phí tính toán. Theo bản chất của nó, nó không thể được phân tầng - tệ hơn rằng, nó đảm bảo một mẫu không phân tầng. Sự phân tầng liên quan đến việc lấy tỷ lệ chính xác các ví dụ trong mỗi lớp vào tập kiểm tra và điều này là không thể khi tập kiểm tra chỉ chứa một ví dụ duy nhất. Một kịch tính, mặc dù rất giả tạo, minh họa cho những vấn đề mà điều này có thể gây ra là tưởng tượng một tập dữ liệu hoàn toàn ngẫu nhiên có chứa cùng một số trong hai

các lớp học. Điều tốt nhất mà một bộ quy nạp có thể làm với dữ liệu ngẫu nhiên là dự đoán lớp đa số, cho tỷ lệ lỗi thực sự là 50%. Nhưng trong mỗi lần loại bỏ một, lớp đối lập với thực thể kiểm tra chiếm đa số—và do đó dự đoán sẽ luôn không chính xác, dẫn đến tỷ lệ lỗi ước tính là 100%!

khởi động

Phương pháp ước tính thứ hai mà chúng tôi mô tả, bootstrap, dựa trên quy trình thống kê lấy mẫu có thay thế. Trước đây, bất cứ khi nào một mẫu được lấy từ tập dữ liệu để tạo thành tập huấn luyện hoặc kiểm tra, nó được rút ra mà không cần thay thế. Đó là, cùng một trường hợp, một khi đã chọn, không thể chọn lại. Nó giống như chọn đội bóng đá: bạn không thể chọn cùng một người hai lần. Nhưng các trường hợp tập dữ liệu không giống như mọi người. Hầu hết các phương pháp học tập đều có thể sử dụng cùng một trường hợp hai lần và nó tạo ra sự khác biệt trong kết quả học tập nếu nó có mặt trong tập huấn luyện hai lần. (Những người thích toán học sẽ nhận thấy rằng chúng ta hoàn toàn không nên nói về "bộ" nếu cùng một đối tượng có thể xuất hiện nhiều hơn hơn một lần.)

Ý tưởng của bootstrap là lấy mẫu tập dữ liệu bằng cách thay thế để tạo thành một tập huấn luyện. Chúng tôi sẽ mô tả một biến thể cụ thể, một cách bí ẩn (nhưng đối với một lý do sẽ sớm trở nên rõ ràng) được gọi là bootstrap 0,632. Đối với điều này, một tập dữ liệu của n phiên bản được lấy mẫu n lần, với sự thay thế, để đưa ra một phiên bản khác tập dữ liệu của n trường hợp. Bởi vì một số phần tử trong tập dữ liệu thứ hai này sẽ (gần như chắc chắn) được lặp lại, phải có một số trường hợp trong tập dữ liệu gốc chưa được chọn: chúng tôi sẽ sử dụng chúng làm phiên bản thử nghiệm.

Cơ hội mà một trường hợp cụ thể sẽ không được chọn cho tập huấn luyện là gì? Nó có xác suất được chọn là 1/n mỗi lần và do đó xác suất không được chọn là 1 - 1/n. Nhân các xác suất này với nhau theo số lượng cơ hội chọn, đó là n, và kết quả là một con số của

$$\left(1 - \frac{1}{n}\right)^n = 0.368$$

(trong đó e là cơ số của logarit tự nhiên, 2,7183, không phải tỷ lệ lỗi!). điều này mang lại cơ hội của một trường hợp cụ thể hoàn toàn không được chọn. Do đó, đối với tập dữ liệu khá lớn, tập kiểm tra sẽ chứa khoảng 36,8% số phiên bản và tập huấn luyện sẽ chứa khoảng 63,2% trong số chúng (bây giờ bạn có thể hiểu tại sao nó được gọi là trình khởi động 0,632). Một số trường hợp sẽ được lặp lại trong tập huấn luyện, đưa nó lên tổng kích thước là n, giống như trong tập dữ liệu gốc.

Con số thu được bằng cách đào tạo một hệ thống học tập trên tập huấn luyện và tính toán lỗi của nó trên tập kiểm tra sẽ là ước tính bi quan của lỗi thực tỷ lệ, bởi vì tập huấn luyện, mặc dù kích thước của nó là n, tuy nhiên chỉ chứa 63% các trường hợp, ví dụ, đây không phải là nhiều so với

90% được sử dụng trong xác thực chéo 10 lần. Để bù đắp cho điều này, chúng tôi kết hợp các tỷ lệ lỗi tập kiểm tra với lỗi thay thế lại trên các phiên bản trong quá trình đào tạo bộ. Con số thay thế, như chúng tôi đã cảnh báo trước đó, đưa ra ước tính rất lạc quan về sai số thực và chắc chắn không nên được sử dụng làm con số sai số trên sở hữu. Nhưng thủ tục bootstrap kết hợp nó với tỷ lệ lỗi kiểm tra để đưa ra một ước tính cuối cùng e như sau:

=¥ +¥ 0 .632 ee e phiên bản 0 .368 tạo
. . . trường hợp thử nghiệm

Sau đó, toàn bộ quy trình bootstrap được lặp lại nhiều lần, với các các mẫu thay thế cho tập huấn luyện và kết quả tính trung bình.

Thủ tục bootstrap có thể là cách tốt nhất để ước tính lỗi cho rất tập dữ liệu nhỏ. Tuy nhiên, giống như xác thực chéo bỏ qua một lần, nó có nhược điểm điều đó có thể được minh họa bằng cách xem xét một tình huống nhân tạo, đặc biệt. Trên thực tế tập dữ liệu mà chúng tôi đã xem xét trước đây sẽ làm được: một tập dữ liệu hoàn toàn ngẫu nhiên với Hai lớp. Tỷ lệ lỗi thực sự là 50% cho bất kỳ quy tắc dự đoán nào. Nhưng một kế hoạch mà ghi nhớ tập huấn luyện sẽ cho điểm thay thế hoàn hảo là 100% sao cho các phiên bản đào tạo = 0 và bootstrap 0,632 sẽ trộn cái này với một trọng số là 0,368 để đưa ra tỷ lệ lỗi chung chỉ là 31,6% (0,632 ¥ 50% + 0,368 ¥ 0%), đó là lạc quan một cách sai lầm.

5.5 So sánh các phương pháp khai thác dữ liệu

Chúng ta thường cần so sánh hai phương pháp học khác nhau về cùng một vấn đề để xem cái nào tốt hơn để sử dụng. Nó có vẻ đơn giản: ước tính sai số bằng cách sử dụng xác thực chéo (hoặc bất kỳ thủ tục ước tính phù hợp nào khác), có thể được lặp lại nhiều lần và chọn lược đồ có ước lượng nhỏ hơn. Điều này khá đủ trong nhiều ứng dụng thực tế: nếu một phương pháp có ước tính thấp hơn lỗi khác trên một tập dữ liệu cụ thể, điều tốt nhất chúng ta có thể làm là sử dụng cái cũ mô hình của phương pháp Tuy nhiên, có thể sự khác biệt chỉ đơn giản là do lỗi ước tính và trong một số trường hợp, điều quan trọng là phải xác định xem liệu một sơ đồ thực sự tốt hơn sơ đồ khác về một vấn đề cụ thể. Đây là một thách thức khó khăn đối với các nhà nghiên cứu máy học. Nếu một thuật toán học tập mới là được đề xuất, những người đề xuất nó phải chứng minh rằng nó cải thiện tình trạng của nghệ thuật cho vấn đề hiện tại và chứng minh rằng sự cải thiện quan sát được không phải là chỉ là một hiệu ứng ngẫu nhiên trong quá trình ước tính.

Đây là công việc cho một bài kiểm tra thống kê đưa ra giới hạn tin cậy, loại mà chúng tôi đã gặp trước đây khi cố gắng dự đoán hiệu suất thực từ một bộ kiểm tra nhất định tỷ lệ lỗi. Nếu có dữ liệu không giới hạn, chúng ta có thể sử dụng một lượng lớn dữ liệu để huấn luyện và đánh giá hiệu suất trên một tập kiểm tra độc lập lớn, đạt được giới hạn tin cậy như trước đây. Tuy nhiên, nếu sự khác biệt hóa ra là đáng kể chúng tôi phải đảm bảo rằng điều này không chỉ vì tập dữ liệu cụ thể mà chúng tôi

đã xảy ra để dựa vào thí nghiệm trên. Những gì chúng tôi muốn xác định là trung bình một lược đồ tốt hơn hay tệ hơn một sơ đồ khác, trên tất cả các tập dữ liệu thử nghiệm và huấn luyện có thể được rút ra từ miền. Vì số lượng dữ liệu đào tạo ảnh hưởng đến hiệu suất một cách tự nhiên, nên tất cả các bộ dữ liệu phải có cùng kích thước: thực tế, thử nghiệm có thể được lặp lại với các kích thước khác nhau để có được đường cong học tập.

Hiện tại, giả sử rằng nguồn cung cấp dữ liệu là không giới hạn. Để chắc chắn, giả sử rằng xác thực chéo đang được sử dụng để thu được các ước tính lỗi (các công cụ ước tính khác, chẳng hạn như xác thực chéo lặp lại, đều khả thi như nhau). Đối với mỗi phương pháp học, chúng ta có thể vẽ một số tập dữ liệu có cùng kích thước, có được ước tính chính xác cho từng tập dữ liệu bằng cách sử dụng xác thực chéo và tính giá trị trung bình của các ước tính. Mỗi thử nghiệm xác thực chéo mang lại một ước tính lỗi độc lập, khác nhau. Điều chúng tôi quan tâm là độ chính xác trung bình trên tất cả các bộ dữ liệu có thể có cùng kích thước và liệu giá trị trung bình này có lớn hơn đối với lược đồ này hay lược đồ kia hay không.

Từ quan điểm này, chúng tôi đang cố gắng xác định xem giá trị trung bình của một tập hợp mẫu—ước tính xác thực chéo cho các bộ dữ liệu khác nhau mà chúng tôi đã lấy mẫu từ miền—lớn hơn hoặc nhỏ hơn đáng kể so với giá trị trung bình của một miền khác. Đây là công việc dành cho một thiết bị thống kê được gọi là bài kiểm tra t, hoặc bài kiểm tra t của Sinh viên. Bởi vì cùng một thử nghiệm xác thực chéo có thể được sử dụng cho cả hai phương pháp học tập để thu được một cặp kết quả phù hợp cho mỗi tập dữ liệu, nên có thể sử dụng **một phiên bản** thử nghiệm t nhạy cảm hơn được gọi là thử nghiệm t được ghép nối .

Chúng tôi cần một số ký hiệu. Có một tập hợp các mẫu x1, x2,..., xk thu được bằng cách xác thực chéo 10 lần liên tiếp bằng cách sử dụng một lược đồ học tập và một tập hợp mẫu thứ hai y1, y2,..., yk thu được bằng cách xác thực chéo 10 lần liên tiếp xác nhận chéo bằng cách sử dụng khác. Mỗi ước tính xác thực chéo được tạo bằng cách sử dụng một tập dữ liệu khác (nhưng tất cả các tập dữ liệu đều có cùng kích thước và từ cùng một miền). Chúng tôi sẽ nhận được kết quả tốt nhất nếu chính xác các phân vùng xác thực chéo giống nhau được sử dụng cho cả hai lược đồ sao cho x1 và y1 thu được bằng cách sử dụng cùng một phân chia xác thực chéo, cũng như x2 và y2, v.v. Biểu thị giá trị trung bình của tập hợp mẫu đầu tiên theo x – và giá trị trung bình của tập hợp thứ hai theo y –. Chúng tôi đang cố gắng xác định xem x có khác biệt đáng kể so với y hay không –.

Nếu có đủ mẫu, giá trị trung bình (x –) của một tập hợp các mẫu độc lập (x1, x2,..., xk) có phân phối chuẩn (tức là Gaussian), bất kể phân phối bên dưới chính các mẫu đó. Chúng ta sẽ gọi giá trị thực của giá trị trung bình là m. Nếu chúng ta biết phương sai của phân phối chuẩn đó, để nó có thể giảm xuống có trung bình bằng 0 và phương sai đơn vị, thì chúng ta có thể thu được giới hạn tin cậy trên m với giá trị trung bình của các mẫu (x -) . Tuy nhiên, phương sai là không xác định và cách duy nhất chúng ta có thể có được nó là ước tính nó từ tập hợp các mẫu.

Điều đó không khó để làm. Phương sai của x có thể được ước tính bằng cách chia phương sai tính được từ các mẫu x1, x2,..., xk—gọi nó là s2 x—cho k. Nhưng

thực tế là chúng ta phải ước tính phương sai thay đổi mọi thứ phần nào. Chúng ta có thể giảm phân phối của x để có giá trị trung bình bằng 0 và phương sai đơn vị bằng cách sử dụng

$$\frac{\bar{x} - m}{\sqrt{s^2_{\text{giây}} / k}}.$$

Bởi vì phương sai chỉ là một ước tính, nên nó không có phân phối chuẩn (mặc dù nó trở nên chuẩn đối với các giá trị lớn của k). Thay vào đó, nó có những gì được gọi là phân phối Student với k - 1 bậc tự do. Điều này có nghĩa là gì trong thực tế là chúng ta phải sử dụng bảng khoảng tin cậy cho Student's phân phối thay vì bảng độ tin cậy cho phân phối bình thường đã cho sớm hơn. Đối với 9 bậc tự do (là con số chính xác nếu chúng ta đang sử dụng trung bình của 10 xác nhận chéo) các giới hạn độ tin cậy phù hợp được hiển thị trong Bảng 5.2. Nếu bạn so sánh chúng với Bảng 5.1, bạn sẽ thấy rằng học sinh của các số liệu thận trọng hơn một chút—đối với một mức độ tin cậy nhất định, khoảng rộng hơn một chút—và điều này phản ánh sự không chắc chắn bổ sung do phải ước tính phương sai. Các bảng khác nhau là cần thiết cho các mục đích khác nhau số bậc tự do, và nếu có hơn 100 bậc tự do tự do thì giới hạn tin cậy rất gần với giới hạn cho phân phối chuẩn. Giống như Bảng 5.1, các số liệu trong Bảng 5.2 là cho độ tin cậy "một phía" khoảng.

Để quyết định xem phương tiện x – và y –, mỗi giá trị trung bình của cùng một số k mẫu giống nhau hay không, ta xem xét sự khác biệt di giữa các lần quan sát tương ứng, di = xi - yi. Điều này là hợp pháp bởi vì các quan sát được ghép nối. Ý nghĩa của sự khác biệt này chỉ là sự khác biệt giữa hai có nghĩa là, $d = x - y$, và, giống như bản thân phương tiện, nó có phân phối của Học sinh với k - 1 bậc tự do. Nếu các phương tiện là như nhau, sự khác biệt là bằng không (điều này được gọi là giả thuyết vô hiệu); nếu chúng khác nhau đáng kể, sự khác biệt sẽ khác đáng kể so với không. Vì vậy, đối với một mức độ tin cậy nhất định, chúng tôi sẽ kiểm tra xem sự khác biệt thực tế có vượt quá giới hạn tin cậy hay không.

Bảng 5.2 Giới hạn tin cậy của phân phối Student với 9 bậc tự do.

Pr[X ≥ z]	z
0,1%	4h30
0,5%	3,25
1%	2,82
5%	1,83
10%	1,38
20%	0,88

Đầu tiên, giảm sự khác biệt thành một biến trung bình bằng 0, phương sai đơn vị được gọi là thống kê t:

$$t = \frac{\bar{d}}{\sqrt{s_d^2 / k}}$$

s2 ở đâu $\bar{d}$ là phương sai của các mẫu khác biệt. Sau đó, quyết định một sự tự tin mức—nói chung, 5% hoặc 1% được sử dụng trong thực tế. Từ đó giới hạn tin cậy z được xác định bằng cách sử dụng Bảng 5.2 nếu k là 10; nếu không, một bảng tin cậy của Phân phối của sinh viên cho giá trị k trong câu hỏi được sử dụng. Một bài kiểm tra hai đuôi là phù hợp bởi vì chúng tôi không biết trước liệu giá trị trung bình của x là có khả năng lớn hơn của y hoặc ngược lại: do đó, đối với thử nghiệm 1%, chúng tôi sử dụng giá trị tương ứng với 0,5% trong Bảng 5.2. Nếu giá trị của t theo công thức trước lớn hơn z hoặc nhỏ hơn -z, chúng tôi bác bỏ giả thuyết không cho rằng phương tiện là như nhau và kết luận rằng thực sự có sự khác biệt đáng kể giữa hai phương pháp học trên miền đó đối với kích thước tập dữ liệu đó.

Có hai nhận xét đáng được thực hiện về quy trình này. Đầu tiên là kỹ thuật: nếu các quan sát không được ghép nối thì sao? Đó là, điều gì sẽ xảy ra nếu chúng ta không thể, vì một số lý do, để đánh giá lỗi của từng lược đồ học tập trên cùng một bộ dữ liệu? Điều gì sẽ xảy ra nếu số lượng bộ dữ liệu cho mỗi lược đồ thậm chí không giống nhau? Những cái này điều kiện có thể phát sinh nếu người khác đã đánh giá một trong các phương pháp và đã công bố một số ước tính khác nhau cho một miền cụ thể và kích thước tập dữ liệu—hoặc có lẽ chỉ là giá trị trung bình và phương sai của chúng—và chúng tôi muốn so sánh điều này với một phương pháp học tập khác. Sau đó, nó là cần thiết để sử dụng một **bài kiểm tra t thường xuyên, không ghép đôi** . Nếu các phương tiện được phân phối bình thường, như chúng ta đang giả định, sự khác biệt giữa các phương tiện cũng được phân phối bình thường. Thay vì lấy ý nghĩa của sự khác biệt, $\bar{d}$, chúng tôi sử dụng sự khác biệt của phương tiện, $\bar{x} - \bar{y}$. Tất nhiên, đó là điều tương tự: ý nghĩa của sự khác biệt là sự khác biệt của phương tiện. Như ng phương sai của sự khác biệt d là không giống nhau. Nếu phương sai của các mẫu x1, x2, ..., xk là s_x^2 và phương sai của các mẫu y1, y2,..., y1 là s_y^2, ước tính tốt nhất về phương sai của sự khác biệt của phương tiện là

$$\frac{s_x^2}{k} + \frac{s_y^2}{1}.$$

Chính phương sai này (hay đúng hơn là căn bậc hai của nó) nên được sử dụng làm mẫu số của thống kê t đã cho trước đó. Bậc tự do, cần thiết cho tư vấn bảng tin cậy của học sinh, nên được coi là một cách thận trọng bậc tự do nhỏ nhất của hai mẫu. Về cơ bản, biết rằng các quan sát được ghép nối cho phép sử dụng một ước tính tốt hơn cho phương sai, điều này sẽ tạo ra các giới hạn tin cậy chặt chẽ hơn.

Quan sát thứ hai liên quan đến giả định rằng về cơ bản có dữ liệu không giới hạn để có thể sử dụng một số bộ dữ liệu độc lập có kích thước phù hợp.

Trong thực tế, thường chỉ có một tập dữ liệu có kích thước giới hạn. Chuyện gì có thể xong? Chúng tôi có thể chia dữ liệu thành (có thể là 10) tập hợp con và thực hiện xác thực chéo trên từng tập hợp con. Tuy nhiên, kết quả tổng thể sẽ chỉ cho chúng ta biết liệu một kế hoạch học tập có phù hợp hơn với quy mô cụ thể đó hay không—có lẽ là một phần mười của tập dữ liệu gốc. Ngoài ra, bộ dữ liệu gốc có thể được sử dụng lại—cho ví dụ, với sự ngẫu nhiên hóa khác nhau của tập dữ liệu cho mỗi lần xác thực chéo.[2] Tuy nhiên, các ước tính xác thực chéo kết quả sẽ không độc lập bởi vì chúng không dựa trên các bộ dữ liệu độc lập. Trong thực tế, điều này có nghĩa là một sự khác biệt có thể được đánh giá là đáng kể trong khi thực tế không phải vậy. Trên thực tế, chỉ tăng số lượng mẫu k, nghĩa là số lần chạy xác thực chéo, cuối cùng sẽ mang lại một sự khác biệt rõ ràng đáng kể bởi vì giá trị của thống kê t tăng không giới hạn.

Nhiều sửa đổi khác nhau của kiểm định t tiêu chuẩn đã được đề xuất để giải quyết vấn đề này, tất cả chúng đều mang tính kinh nghiệm và thiếu cơ sở lý thuyết hợp lý. Một trong số đó dường như hoạt động tốt trong thực tế là thử nghiệm t được lấy mẫu lại đã sửa. Giả sử tại thời điểm này, phương thức loại trừ lặp đi lặp lại được sử dụng thay vì xác thực chéo, lặp lại k lần trên các phần tách ngẫu nhiên khác nhau của cùng một tập dữ liệu để có được ước tính chính xác cho hai phương pháp học tập. Mỗi lần, n1 trường hợp được sử dụng để đào tạo và n2 để kiểm tra và sự khác biệt di được tính từ hiệu suất trên dữ liệu thử nghiệm. Thử nghiệm t được lấy mẫu lại đã sửa sử dụng sửa đổi thống kê

$$t = \frac{đ}{\sqrt{\left(\frac{1}{k} + \frac{N_2}{N_1}\right) \hat{S}^2_đ}}$$

theo cách chính xác giống như thống kê t tiêu chuẩn. Nhìn kỹ hơn vào công thức chỉ ra rằng giá trị của nó không thể tăng lên đơn giản bằng cách tăng k. Thống kê đã sửa đổi tương tự có thể được sử dụng với xác thực chéo lặp lại, đây chỉ là một đặc biệt trường hợp loại bỏ nhiều lần trong đó các bộ thử nghiệm riêng lẻ cho một lần xác thực chéo không trùng nhau. Đối với xác thực chéo 10 lần được lặp lại 10 lần, k = 100, n2/n1 = 0,1/0,9 và $s^2_đ$ được dựa trên 100 sự khác biệt.

5.6 Dự đoán xác suất

Trong suốt phần này, chúng ta đã ngầm giả định rằng mục tiêu là tối đa hóa tỷ lệ thành công của các dự đoán. Kết quả cho mỗi trường hợp thử nghiệm là đúng, nếu dự đoán phù hợp với giá trị thực tế cho trường hợp đó, hoặc không chính xác, nếu không. Không có màu xám: mọi thứ đều đen hoặc trắng, đúng hoặc sai

[2] Phương pháp này đã được ủng hộ trong ấn bản đầu tiên của cuốn sách này.

không đúng. Trong nhiều tình huống, đây là quan điểm thích hợp nhất. Nếu kế hoạch học tập, khi nó thực sự được áp dụng, dẫn đến kết quả đúng hoặc sai. dự đoán không chính xác, thành công là biện pháp đúng đắn để sử dụng. Điều này đôi khi được gọi là hàm **mất mát 0 - 1**: "tổn thất" bằng 0 nếu dự đoán đúng hoặc một nếu không phải vậy. Việc sử dụng tổn thất là thông thường, mặc dù thay vào đó, một thuật ngữ lạc quan hơn có thể diễn đạt kết quả dưới dạng lợi nhuận.

Các tình huống khác nhẹ nhàng hơn. Hầu hết các phương pháp học tập có thể liên kết một xác suất với mỗi dự đoán (như phương pháp Naïve Bayes thực hiện). Có thể là tự nhiên hơn khi tính đến xác suất này khi đánh giá tính đúng đắn. Vì ví dụ, một kết quả chính xác được dự đoán với xác suất 99% có lẽ nên nặng hơn một dự đoán với xác suất là 51%, và trong tình huống hai lớp, có lẽ trường hợp sau không tốt hơn nhiều so với một tình huống không chính xác kết quả dự đoán với xác suất 51%. Việc tính đến các xác suất dự đoán có phù hợp hay không tùy thuộc vào ứng dụng. Nếu cuối cùng ứng dụng thực sự chỉ là một dự đoán về kết quả và không có giải thưởng nào được trao để đánh giá thực tế về khả năng dự đoán, có vẻ như không thích hợp để sử dụng xác suất. Tuy nhiên, nếu dự đoán có thể được xử lý thêm—có lẽ liên quan đến đánh giá của một người, hoặc phân tích chi phí, hoặc thậm chí có thể đóng vai trò là đầu vào cho quá trình học tập ở cấp độ thứ hai—sau đó nó có thể cũng phù hợp để tính đến xác suất dự đoán.

Hàm mất bậc hai

Giả sử rằng đối với một trường hợp duy nhất có k kết quả hoặc lớp có thể xảy ra và trong một trường hợp nhất định, sơ đồ học tập đưa ra một vectơ xác suất p1, p2,..., pk cho các lớp (trong đó các xác suất này có tổng bằng 1). Thực tế kết quả cho trường hợp đó sẽ là một trong các lớp có thể. Tuy nhiên, thuận tiện là biểu diễn nó dưới dạng véc tơ a1, a2,..., ak có thành phần thứ i, trong đó i là lớp thực tế, là 1 và tất cả các thành phần khác là 0. Chúng ta có thể biểu thị hình phạt liên kết với tình huống này như một hàm mất mát phụ thuộc vào cả vectơ p và véc tơ a .

Một tiêu chí thường được sử dụng để đánh giá dự đoán xác suất là hàm mất bậc hai:

$$\sum_j (p_j - a_j)^2.$$

Lưu ý rằng đây là trường hợp duy nhất: tổng kết vượt quá các đầu ra có thể không qua các trường hợp khác nhau. Chỉ một trong số a sẽ là 1 và phần còn lại sẽ là 0, vì vậy tổng chứa đóng góp của p_j^2 cho các dự đoán không chính xác và $(1 - p_i)^2$ cho đúng. Do đó, nó có thể được viết

$$1 - 2p_j + \sum_j p_j^2$$

nơ i tôi là đúng lớp. Khi bộ kiểm tra chứa một số trư ờng hợp, tổn thất chức năng đư ợc tổng hợp trên tất cả chúng.

Một thực tế lý thuyết thú vị là nếu bạn tìm cách giảm thiểu giá trị của hàm mất bậc hai trong tình huống trong đó lớp thực đư ợc tạo về mặt xác suất, chiến lư ợc tốt nhất là chọn cho vectơ p khả năng xác suất thực tế của các kết quả khác nhau, nghĩa là pi = Pr[class = i]. Nếu xác suất thực đư ợc biết, chúng sẽ là giá trị tốt nhất cho p. Nếu không, một hệ thống cố gắng giảm thiểu hàm mất bậc hai sẽ đư ợc khuyến khích sử dụng ư ớc tính tốt nhất của Pr[class = i] là giá trị của pi.

Điều này khá dễ nhận thấy. Biểu thị các xác suất thực bởi p* 1, p* 2,..., p* k sao cho p*i = Pr[class = i]. Giá trị mong đợi của hàm mất bậc hai cho một bài kiểm tra ví dụ có thể đư ợc viết lại như sau:

$$\begin{aligned} E\left[\left(p_1 - a_1\right)^2 + \left(p_2 - a_2\right)^2 + \cdots\right] &= E\left[p_1^2 - 2p_1a_1 + a_1^2\right] + E\left[p_2^2 - 2p_2a_2 + a_2^2\right] + \cdots \\ &= p_1^2 - 2p_1p_1^* + p_1^* + p_2^2 - 2p_2p_2^* + p_2^* + \cdots \\ &= \left(p_1 - p_1^*\right)^2 + \left(p_2 - p_2^*\right)^2 + \cdots + 1 - p_1^{*2} - p_2^{*2} - \cdots \end{aligned}$$

Giai đoạn đầu tiên chỉ bao gồm việc đư a kỳ vọng vào bên trong tổng và khai triển bình phư ơ ng. Đối với trư ờng hợp thứ hai, pj chỉ là một hằng số và giá trị kỳ vọng của aj chỉ đơ n giản là p*j ; hơ n nữa, vì aj hoặc là 0 hoặc 1, nên $a_j^2 = a_j$ và giá trị kỳ vọng của nó là p * j quá. Giai đoạn thứ ba là đại số đơ n giản. Để giảm thiểu kết quả Tóm lại, rõ ràng là tốt nhất nên chọn pj = p*j sao cho số hạng bình phư ơ ng biến mất và tất cả những gì còn lại là số hạng chỉ là phư ơ ng sai của phân phối thực chi phối lớp thực tế.

Giảm thiểu lỗi bình phư ơ ng có một lịch sử lâu dài trong các vấn đề dự đoán. TRONG bối cảnh hiện tại, hàm mất bậc hai buộc ngư ời dự đoán phải trung thực về việc lựa chọn ư ớc tính tốt nhất của nó về xác suất – hay nói đúng hơ n là nó ư u tiên hơ n cho những ngư ời dự đoán có khả năng đư a ra dự đoán tốt nhất về xác suất thực. Hơ n nữa, hàm mất bậc hai có một số tính chất lý thuyết hữu ích mà chúng tôi sẽ không đi vào đây. Vì tất cả những lý do này, nó thư ờng đư ợc sử dụng như là tiêu chí thành công trong các tình huống dự đoán xác suất.

Chức năng mất thông tin

Một tiêu chí phổ biến khác để đánh giá dự đoán xác suất là chức năng mất thông tin:

$$-\log_2 p_i$$

trong đó dự đoán thứ i là dự đoán đúng. Trên thực tế, điều này giống với giá trị âm của hàm log-likelihood đư ợc tối ư u hóa bằng hồi quy logistic, mô tả trong Mục 4.6. Nó đại diện cho thông tin (theo bit) cần thiết để thể hiện lớp thực tế i đối với phân bố xác suất p1, p2,...,

pk. Nói cách khác, nếu bạn được cung cấp phân phối xác suất và ai đó
phải liên lạc với bạn lớp nào là lớp thực sự xảy ra, điều này
là số bit mà người đó sẽ cần để mã hóa thông tin nếu họ
đã làm nó một cách hiệu quả nhất có thể. (Tất nhiên, luôn có thể sử dụng nhiều bit hơn.)
Vì xác suất luôn nhỏ hơn 1 nên logarit của chúng âm và
dấu trừ làm cho kết quả tích cực. Ví dụ, trong một tình huống hai
loại—ngửa hoặc sấp—với xác suất của mỗi loại như nhau, sự xuất hiện của
đầu sẽ mất 1 bit để truyền, vì -log21/2 là 1.

Giá trị kỳ vọng của hàm mất thông tin, nếu các mối quan hệ xác suất thực sự là
p* 1, p* 2,..., p* k , là

$$-\left(p^*_1 \log p_1 + p^*_2 \log p_2 + \cdots + p^*_k \log p_k\right).$$

Giống như hàm mất bậc hai, biểu thức này được giảm thiểu bằng cách chọn pj =
p*j , trong trường hợp đó, biểu thức trở thành entropy của phân phối thực:

$$-\left(p^*_1 \log p^*_1 + p^*_2 \log p^*_2 + \cdots + p^*_k \log p^*_k\right).$$

Do đó, chức năng mất thông tin cũng thưởng cho sự trung thực trong các dự đoán
biết xác suất thực sự và khuyến khích những người dự đoán không đặt
chuyển tiếp dự đoán tốt nhất của họ.

Hàm mất thông tin cũng có cách giải thích cờ bạc trong đó
bạn tưởng tượng đánh cược vào kết quả, đặt tỷ lệ cược vào mỗi lớp có thể và
chiến thắng theo lớp mà đi lên. Các trường hợp liên tiếp giống như cá
cược liên tiếp: bạn mang thắng (hoặc thua) từ ván này sang ván khác. logarit
trong tổng số tiền bạn giành được trong toàn bộ bộ thử nghiệm là giá trị của
chức năng mất thông tin. Trong cờ bạc, nó trả tiền để có thể dự đoán tỷ lệ cược
Càng chính xác càng tốt; theo nghĩa đó, sự trung thực cũng được đền đáp.

Một vấn đề với chức năng mất thông tin là nếu bạn chỉ định một
xác suất bằng không đối với một sự kiện thực sự xảy ra, giá trị của hàm là âm
vô cực. Điều này tương ứng với việc mất áo khi đánh bạc. người đánh cược thận trọng
không bao giờ đặt cược mọi thứ vào một sự kiện cụ thể, bất kể nó có vẻ chắc chắn đến đâu.
Tương tự như vậy, các yếu tố dự đoán thận trọng hoạt động theo chức năng mất thông tin
không gán xác suất bằng không cho bất kỳ kết quả nào. Điều này dẫn đến một vấn đề khi
không có thông tin nào về kết quả đó để dựa vào dự đoán:
đây được gọi là vấn đề tần số bằng không và nhiều giải pháp hợp lý khác nhau đã
đã được đề xuất, chẳng hạn như công cụ ước tính Laplace được thảo luận cho Naïve Bayes trên trang
91.

Cuộc thảo luận

Nếu bạn đang kinh doanh đánh giá các dự đoán về xác suất,
hai chức năng mất bạn nên sử dụng? Đó là một câu hỏi hay và không có câu trả lời
được thống nhất chung — đó thực sự là vấn đề về sở thích. Cả hai đều làm cơ bản-

công việc tinh thần mong đợi của một hàm mất mát: họ trao phần thư ởng tối đa cho những ngư ời dự đoán có khả năng dự đoán xác suất thực một cách chính xác. Tuy nhiên, có một số khác biệt khách quan giữa hai điều này có thể giúp bạn hình thành ý kiến.

Hàm mất mát bậc hai không chỉ tính đến xác suất đư ợc gán cho sự kiện đã thực sự xảy ra mà còn cả các xác suất khác. Ví dụ: trong tình huống có bốn lớp, giả sử bạn đã chỉ định 40% cho lớp thực sự nghĩ ra và phân phối phần còn lại cho ba lớp còn lại. Tổn thất bậc hai sẽ phụ thuộc vào cách bạn phân phối nó do tổng của p2 xảy ra trong biểu thức đã cho trư ớc đó đối với hàm mất mát bậc hai.

j

Tổn thất sẽ là nhỏ nhất nếu 60% đư ợc phân phối đồng đều giữa ba loại: phân phối không đồng đều sẽ làm tăng tổng bình phư ơ ng. Mặt khác, hàm mất mát thông tin chỉ phụ thuộc vào xác suất đư ợc gán cho lớp thực sự xảy ra. Nếu bạn đang đánh bạc vào một sự kiện cụ thể sắp diễn ra, và đúng như vậy, ai quan tâm bạn phân bổ số tiền còn lại của mình như thế nào cho các sự kiện khác?

Nếu bạn chỉ định một xác suất rất nhỏ cho lớp thực sự xảy ra, hàm mất thông tin sẽ phạt bạn rất nhiều. Hình phạt tối đa, đối với xác suất bằng không, là vô hạn. Thế giới cờ bạc cũng phạt nặng những lỗi như thế này! Mặt khác, hàm mất bậc hai nhẹ hơ n, bị giới hạn bởi

$$1^2 + \hat{A}_j pj ,$$

mà không bao giờ có thể vư ợt quá 2.

Cuối cùng, những ngư ời ủng hộ hàm mất thông tin chỉ ra một lý thuyết chung về đánh giá hiệu suất trong học tập đư ợc gọi là nguyên tắc độ dài mô tả tối thiểu (MDL). Họ lập luận rằng kích thư ớc của các cấu trúc mà một sơ đồ học đư ợc có thể đư ợc đo bằng các bit thông tin và nếu các đơ n vị giống nhau đư ợc sử dụng để đo tổn thất, cả hai có thể đư ợc kết hợp theo những cách hữu ích và mạnh mẽ. Chúng tôi trở lại vấn đề này trong Phần 5.9.

5.7 Tính chi phí

Các đánh giá đã đư ợc thảo luận cho đến nay không tính đến chi phí đư a ra quyết định sai, phân loại sai. Tối ư u hóa tỷ lệ phân loại mà không xem xét chi phí của các lỗi thư ờng dẫn đến kết quả kỳ lạ. Trong một trư ờng hợp, máy học đã đư ợc sử dụng để xác định chính xác ngày mà mỗi con bò trong đàn bò sữa bắt đầu động dục hoặc "động dục". Bò đư ợc nhận dạng bằng thẻ tai điện tử và nhiều thuộc tính khác nhau đư ợc sử dụng như khối lư ợng sữa và thành phần hóa học (đư ợc máy vắt sữa công nghệ cao ghi lại tự động) và thứ tự vắt sữa—vì bò là động vật thông thư ờng và thư ờng đến nơ i vắt sữa.

lột xác theo cùng một thứ tự, ngoại trừ những trường hợp bất thường như động dục. trong một hoạt động chăn nuôi bò sữa hiện đại, điều quan trọng là phải biết khi nào một con bò đã sẵn sàng: động vật được thụ tinh bằng phương pháp thụ tinh nhân tạo và thiếu một chu kỳ sẽ làm chậm quá trình đẻ không cần thiết, gây ra các biến chứng xuống dòng. Trong những thí nghiệm ban đầu, phương pháp học máy đã dự đoán một cách bướng bỉnh rằng mỗi con bò không bao giờ động dục. Cũng như con người, bò có chu kỳ kinh nguyệt xấp xỉ 30 ngày nên quy tắc "không" này đúng khoảng 97% thời gian—một mức độ chính xác ấn tượng trong bất kỳ lĩnh vực nông nghiệp nào! Tất nhiên, những gì được mong muốn là những quy tắc mà dự đoán tình trạng "đang động dục" chính xác hơn tình trạng "không động dục": chi phí của hai loại lỗi là khác nhau. Đánh giá theo phân loại độ chính xác mặc nhiên giả định chi phí lỗi bằng nhau.

Các ví dụ khác trong đó sai sót gây ra những chi phí khác nhau bao gồm các quyết định cho vay: chi phí cho người vỡ nợ vay lớn hơn nhiều so với chi phí kinh doanh thua lỗ từ chối một khoản vay cho một người không mặc định. Và phát hiện vết dầu loang: chi phí thất bại để phát hiện một vết dầu loang thực sự đe dọa đến môi trường lớn hơn nhiều so với chi phí của một báo động giả. Và dự báo phụ tải: chi phí điều chỉnh máy phát điện đối với một cơn bão không ập đến ít hơn nhiều so với chi phí bị bắt hoàn toàn không chuẩn bị. Và chẩn đoán: chi phí xác định sai sự cố với máy hóa ra là không có lỗi ít hơn chi phí bỏ qua các vấn đề với một thứ sắp hỏng. Và gửi thư quảng cáo: chi phí gửi thư rác gửi thư cho một hộ gia đình không trả lời ít hơn nhiều so với chi phí kinh doanh bị mất về việc không gửi nó đến một hộ gia đình sẽ phản hồi. Tại sao—đây là tất cả các ví dụ của Chương 1! Trên thực tế, bạn khó có thể tìm thấy một ứng dụng nào trong đó chi phí của các loại lỗi khác nhau là như nhau.

Trong trường hợp hai lớp với các lớp có và không, cho mượn hoặc không cho mượn, đánh dấu một chỗ đáng ngờ là vết dầu loang hay không, v.v., một dự đoán duy nhất có bốn kết quả khả dĩ khác nhau được trình bày trong Bảng 5.3. Các tích cực thực sự (TP) và đúng phủ định (TN) là những phân loại đúng. Dương tính giả (FP) xảy ra khi kết quả được dự đoán không chính xác là có (hoặc tích cực) khi nó thực sự là không (tiêu cực). Âm tính giả (FN) xảy ra khi kết quả được dự đoán không chính xác là tiêu cực khi nó thực sự tích cực. Tỷ lệ tích cực thực sự là TP chia

Bảng 5.3 kết quả khác nhau của một dự đoán hai lớp.

		lớp dự đoán	
		Đúng	KHÔNG
lớp học thực tế	Đúng	tích cực thực sự	âm tính giả
	KHÔNG	dương tính giả	âm tính thật

bằng tổng số dương, đó là TP + FN; tỷ lệ dương tính giả là FP chia cho tổng số âm tính, FP + TN. Tỷ lệ thành công chung là số phân loại đúng chia cho tổng số
phân loại:

$$\frac{TP + TN}{TP + TN + FP + FN}.$$

Cuối cùng, tỷ lệ lỗi là một trừ đi điều này.

Trong dự đoán nhiều lớp, kết quả trên tập kiểm tra thường được hiển thị dưới dạng ma trận nhầm lẫn hai chiều với một hàng và cột cho mỗi lớp. Mỗi ma trận phần tử hiển thị số lượng ví dụ kiểm tra mà lớp thực tế là hàng và lớp dự đoán là cột. Kết quả tốt tương ứng với số lượng lớn xuống đường chéo chính và các phần tử nhỏ, lý tưởng là bằng 0, nằm ngoài đường chéo. Bàn 5.4(a) hiển thị một ví dụ số với ba lớp. Trong trường hợp này, tập kiểm tra có 200 trường hợp (tổng của chín số trong ma trận) và 88 + 40 + 12 = 140 trong số đó được dự đoán đúng nên tỷ lệ thành công là 70%.

Như ng liệu đây có phải là thước đo công bằng cho thành công tổng thể? Có bao nhiêu thỏa thuận sẽ bạn mong đợi một cách tình cờ? Công cụ dự đoán này dự đoán tổng cộng 120 a's, 60 b's và 20 cs; điều gì sẽ xảy ra nếu bạn có một công cụ dự đoán ngẫu nhiên dự đoán tổng số giống nhau của ba lớp? Câu trả lời được thể hiện trong Bảng 5.4(b). Hàng đầu tiên của nó chia 100 a trong bài kiểm tra được đặt thành các tỷ lệ tổng thể này, và thứ hai và hàng thứ ba làm điều tương tự cho hai lớp còn lại. Tất nhiên, hàng và tổng số cột cho ma trận này vẫn giống như trước – số lượng trường hợp đã không thay đổi và chúng tôi đã đảm bảo rằng công cụ dự đoán ngẫu nhiên dự đoán cùng số a, b và c như công cụ dự đoán thực tế.

Dự đoán ngẫu nhiên này có 60 + 18 + 4 = 82 trường hợp đúng. Một biện pháp được gọi là thống kê Kappa tính đến con số kỳ vọng này bằng cách trừ nó khỏi các lần thành công của người dự đoán và biểu thị kết quả dưới dạng tỷ lệ của tổng số đối với một người dự đoán hoàn hảo, để mang lại 140 - 82 = 58 lần thành công bổ sung

Bảng 5.4 Các kết quả khác nhau của dự đoán ba lớp: (a) thực tế và (b) dự kiến.

		lớp dự đoán			
		Một	b	c	Tổng cộng
lớp học		88	10		100
thực tế	ab	14	40		60
	c	18	10	2 6 12	40
	Tổng cộng	120	60	20	

(Một)

		lớp dự đoán			
		Một	b	c	Tổng cộng
lớp học		60	30	10	100
thực tế	ab	36	18		60
	c	24	12	6 4	40
	Tổng cộng	120	60	20	

(b)

trong tổng số có thể là 200 - 82 = 118, hay 49,2%. Giá trị tối đa của Kappa là 100% và giá trị dự kiến cho một công cụ dự đoán ngẫu nhiên có cùng cột tổng bằng không. Tóm lại, thống kê Kappa được sử dụng để đo lường mức độ thỏa thuận giữa các phân loại dự đoán và quan sát của tập dữ liệu, trong khi sửa đổi đối với thỏa thuận xảy ra một cách tình cờ. Tuy nhiên, giống như tỷ lệ thành công đơn giản, nó không không tính đến chi phí.

Phân loại nhạy cảm với chi phí

Nếu các chi phí được biết, chúng có thể được đưa vào phân tích tài chính của quy trình ra quyết định. Trong trường hợp hai lớp, trong đó ma trận nhầm lẫn giống như trong Bảng 5.3, hai loại lỗi—dương tính giả và âm tính giả—sẽ có các chi phí khác nhau; tương tự như vậy, hai loại phân loại chính xác có thể có những lợi ích khác nhau. Trong trường hợp hai lớp, chi phí có thể được tóm tắt trong dạng ma trận 2 ¥ 2 trong đó các phần tử đường chéo đại diện cho hai các loại phân loại chính xác và các phần tử ngoài đường chéo đại diện cho hai các loại lỗi. Trong trường hợp đa lớp, điều này tổng quát thành một ma trận vuông có kích thước là số lượng lớp và một lần nữa các phần tử đường chéo biểu thị chi phí của phân loại chính xác. Bảng 5.5(a) và (b) trình bày ma trận chi phí mặc định cho trường hợp hai và ba loại có giá trị đơn giản là số lượng lỗi: chi phí phân loại sai đều bằng 1.

Tính đến ma trận chi phí sẽ thay thế tỷ lệ thành công bằng tỷ lệ trung bình chi phí (hoặc, suy nghĩ tích cực hơn, lợi nhuận) cho mỗi quyết định. Mặc dù chúng tôi sẽ không làm vì vậy ở đây, một phân tích tài chính hoàn chỉnh về quá trình ra quyết định cũng có thể tính đến chi phí sử dụng công cụ máy học—bao gồm cả chi phí chi phí thu thập dữ liệu huấn luyện—và chi phí sử dụng mô hình hoặc cấu trúc quyết định mà nó tạo ra—tức là chi phí xác định các thuộc tính cho các trường hợp thử nghiệm. Nếu tất cả các chi phí được biết đến, và số lượng dự kiến của

Bảng 5.5 Ma trận chi phí mặc định: (a) trường hợp hai lớp và (b) trường hợp ba lớp.

		lớp dự đoán	
		Đúng	KHÔNG
lớp học thực tế	có	0	
	không	1	1 0

(Một)

			lớp dự đoán		
			Một	b	c
lớp học thực tế		bụng 1	0		1
			1	1 0	

c110

(b)

các kết quả khác nhau trong ma trận chi phí có thể được ước tính—chẳng hạn như sử dụng xác thực chéo—việc thực hiện loại phân tích tài chính này rất đơn giản.

Đưa ra một ma trận chi phí, bạn có thể tính toán chi phí của một mô hình đã học cụ thể trên một tập kiểm tra nhất định chỉ bằng cách tính tổng các yếu tố có liên quan của ma trận chi phí cho dự đoán của mô hình cho từng trường hợp thử nghiệm. Ở đây, các chi phí được bỏ qua khi đưa ra dự đoán, nhưng được tính đến khi đánh giá chúng.

Nếu mô hình đưa ra xác suất liên quan đến từng dự đoán, nó có thể được điều chỉnh để giảm thiểu chi phí dự kiến của các dự đoán. Đưa ra một tập hợp các xác suất dự đoán cho từng kết quả trong một trường hợp thử nghiệm nhất định, một thông thường chọn kết quả có khả năng nhất. Thay vào đó, mô hình có thể dự đoán lớp với chi phí phân loại sai dự kiến nhỏ nhất. Ví dụ: giả sử trong tình huống ba lớp, mô hình gán các lớp a, b và c cho một thể hiện thử nghiệm với xác suất pa, pb và pc, và ma trận chi phí là trong Bảng 5.5(b) . Nếu nó dự đoán a, chi phí mong đợi của dự đoán thu được bằng cách nhân giá trị đầu tiên cột của ma trận, [0,1,1], theo vectơ xác suất, [pa, pb, pc], mang lại pb + pc hoặc 1 - pa vì ba xác suất tổng bằng 1. Tương tự, chi phí cho dự đoán 2 lớp còn lại là 1 - pb và 1 - pc. Đối với ma trận chi phí này, việc chọn dự đoán có chi phí dự kiến thấp nhất cũng giống như chọn dự đoán với xác suất lớn nhất. Đối với một ma trận chi phí khác, nó có thể khác.

Chúng tôi đã giả định rằng phương pháp học tập đưa ra xác suất, như Naïve Bayes thì có. Ngay cả khi chúng không đưa ra xác suất thông thường, hầu hết các bộ phân loại có thể dễ dàng được điều chỉnh để tính toán chúng. Ví dụ, trong cây quyết định, phân phối khả năng xác suất cho một trường hợp thử nghiệm chỉ là phân phối của các lớp tại lá tương ứng.

Học tập nhạy cảm với chi phí

Chúng ta đã thấy cách một bộ phân loại, được xây dựng mà không tính đến chi phí, có thể được sử dụng để đưa ra dự đoán nhạy cảm với ma trận chi phí. Trong trường hợp này, chi phí được bỏ qua tại thời điểm đào tạo nhưng được sử dụng tại thời điểm dự đoán. Một thay thế là làm điều ngược lại: tính đến ma trận chi phí trong quá trình đào tạo xử lý và bỏ qua chi phí tại thời điểm dự đoán. Về nguyên tắc, hiệu suất tốt hơn có thể thu được nếu trình phân loại được thuật toán học điều chỉnh phù hợp với ma trận chi phí.

Trong tình huống hai lớp, có một cách đơn giản và tổng quát để thực hiện bất kỳ phương pháp học nhạy cảm với chi phí. Ý tưởng là tạo dữ liệu đào tạo với tỷ lệ khác nhau giữa các trường hợp có và không . Giả sử rằng bạn tăng giả tạo số lượng không có trường hợp nào theo hệ số 10 và sử dụng tập dữ liệu kết quả cho đào tạo. Nếu sơ đồ học tập đang cố gắng giảm thiểu số lượng lỗi, thì nó sẽ đưa ra một cấu trúc quyết định thiên về việc tránh sai sót trên không có trường hợp nào, bởi vì những lỗi như vậy bị phạt gấp 10 lần. Nếu dữ liệu

với tỷ lệ ban đầu là không có trường hợp nào được sử dụng để thử nghiệm, các trường hợp này sẽ mắc ít lỗi hơn so với **trường hợp** có —tức là sẽ có ít trường hợp dương tính giả hơn so với trường hợp âm tính giả—vì dương tính giả được đánh giá cao hơn gấp 10 lần so với sai sót tiêu cực. Thay đổi tỷ lệ các phiên bản trong tập huấn luyện là một kỹ thuật chung để xây dựng các bộ phân loại nhạy cảm với chi phí.

Một cách để thay đổi tỷ lệ các phiên bản đào tạo là sao chép các phiên bản trong tập dữ liệu. Tuy nhiên, nhiều chương trình học tập cho phép các trường hợp được tính trọng số.
(Như chúng tôi đã đề cập trong Phần 3.2, đây là một kỹ thuật phổ biến để xử lý các giá trị bị thiếu.)
Trọng số của phiên bản thường được khởi tạo thành một. Để xây dựng các cây nhạy cảm với chi phí, các trọng số có thể được khởi tạo thành chi phí tương đối của hai loại lỗi, dương tính giả và âm tính giả.

Biểu đồ thang

máy Trong thực tế, chi phí hiếm khi được biết với bất kỳ mức độ chính xác nào và mọi người sẽ muốn cân nhắc các tình huống khác nhau. Hãy tưởng tượng bạn đang kinh doanh dịch vụ gửi thư trực tiếp và đang dự tính gửi hàng loạt khuyến mại cho 1.000.000 hộ gia đình—tất nhiên là hầu hết trong số họ sẽ không phản hồi. Giả sử rằng, dựa trên kinh nghiệm trước đây, tỷ lệ người thường trả lời được biết là 0,1% (1000 người trả lời). Giả sử có sẵn một công cụ khai thác dữ liệu, dựa trên thông tin đã biết về các hộ gia đình, xác định một tập hợp con gồm 100.000 hộ gia đình với tỷ lệ phản hồi là 0,4% (400 người trả lời). Việc hạn chế gửi thư cho 100.000 hộ gia đình này có thể mang lại kết quả tốt—điều đó phụ thuộc vào chi phí gửi thư so với lợi nhuận thu được từ mỗi phản hồi đối với thư mời. Trong thuật ngữ tiếp thị, mức tăng tỷ lệ phản hồi, hệ số bốn trong trường hợp này, được gọi là **hệ số** nâng do công cụ học tập mang lại. Nếu bạn biết chi phí, bạn có thể xác định kết quả thu được từ một hệ số nâng cụ thể.

Nhưng bạn cũng có thể muốn đánh giá các khả năng khác. Cùng một lược đồ khai thác dữ liệu, với các cài đặt tham số khác nhau, có thể xác định được 400.000 hộ gia đình với tỷ lệ phản hồi sẽ là 0,2% (800 người trả lời), tương ứng với hệ số nâng là hai. Một lần nữa, liệu đây có phải là một mục tiêu có lợi hơn cho việc gửi thư hay không có thể được tính toán từ các chi phí liên quan. Có thể cần phải tính đến chi phí tạo và sử dụng mô hình—bao gồm cả việc thu thập thông tin cần thiết để đưa ra các giá trị thuộc tính.

Xét cho cùng, nếu việc phát triển mô hình là rất tốn kém, thì việc gửi thư hàng loạt có thể hiệu quả hơn về mặt chi phí so với việc gửi thư mục tiêu.

Đưa ra một phương pháp học đưa ra xác suất cho lớp được dự đoán của từng thành viên trong tập hợp các phiên bản thử nghiệm (như Naïve Bayes đã làm), công việc của bạn là tìm các tập hợp con của các phiên bản thử nghiệm có tỷ lệ phiên bản dương tính cao, cao hơn trong thử nghiệm thiết lập như một tổng thể. Để làm điều này, các phiên bản phải được sắp xếp theo thứ tự giảm dần của xác suất dự đoán là có. Sau đó, để tìm một mẫu có kích thước nhất định với tỷ lệ các trường hợp tích cực nhất có thể, chỉ cần đọc

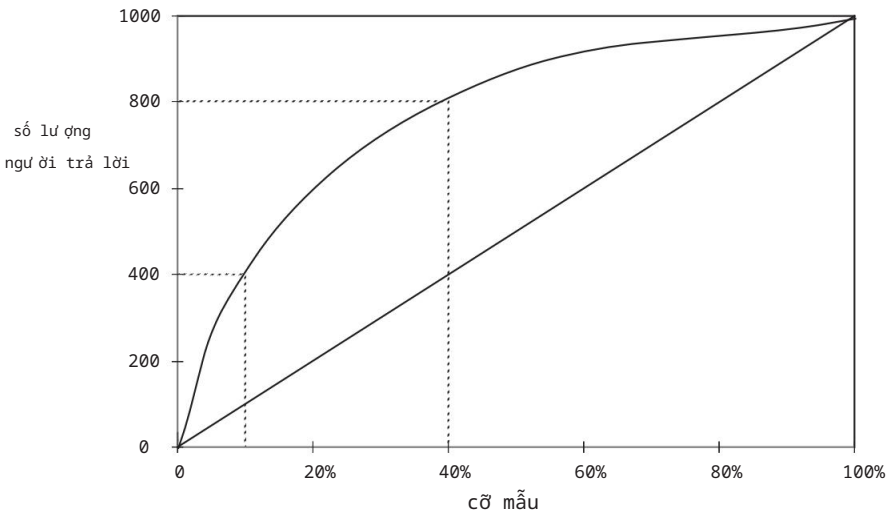
Bảng 5.6 Dữ liệu cho biểu đồ thang máy.

Thứ hạng	xác suất dự đoán	lớp học thực tế	Thứ hạng	xác suất dự đoán	lớp học thực tế
	0,95	vâng	11	0,77	KHÔNG
1	0,93	vâng	12	0,76	Đúng
2	0,93	không	13	0,73	Đúng
3	0,88	vâng	14	0,65	KHÔNG
4	0,86	vâng	15	0,63	Đúng
5	0,85	vâng	16	0,58	KHÔNG
6	0,82	vâng	17	0,56	Đúng
7	0,80	vâng	18	0,49	KHÔNG
8	0,80	không	19	0,48	Đúng
9 10	0,79	Đúng	...	...	...

số lư ợng phiên bản cần thiết ngoài danh sách, bắt đầu từ trên cùng. Nếu mỗi bài kiểm tra lớp của cá thể đã biết, bạn có thể tính toán hệ số nâng bằng cách chỉ cần đếm số trư ờng hợp tích cực mà mẫu bao gồm, chia cho mẫu size để có đư ợc tỷ lệ thành công và chia cho tỷ lệ thành công cho bộ thử nghiệm hoàn chỉnh để xác định hệ số nâng.

Bảng 5.6 cho thấy một ví dụ về tập dữ liệu nhỏ với 150 phiên bản, trong đó 50 câu trả lời có —tỷ lệ thành công tổng thể là 33%. Các trư ờng hợp có đư ợc sắp xếp theo thứ tự xác suất giảm dần theo xác suất dự đoán của câu trả lời có . Trư ờng hợp đầu tiên là trư ờng hợp mà chư ơ ng trình học tập nghĩ là có khả năng tích cực nhất, thứ hai là có khả năng nhất tiếp theo, v.v. Các giá trị số của xác suất là không quan trọng: thứ hạng là thứ duy nhất đó là vấn đề. Với mỗi thứ hạng đư ợc đư a ra lớp thực tế của thể hiện. Như vậy các phư ơ ng pháp học tập đã đúng về mục 1 và 2—chúng thực sự là tích cực—như ng sai về mục 3, hóa ra là tiêu cực. Bây giờ, nếu bạn đang tìm kiếm mẫu có kích thư ớc 10 hứa hẹn nhất như ng chỉ biết xác suất dự đoán và không phải là các lớp thực tế, đặt cư ợc tốt nhất của bạn sẽ là mư ời trư ờng hợp xếp hạng hàng đầu. Tám trong số này là tích cực, vì vậy tỷ lệ thành công cho mẫu này là 80%, tư ơ ng ứng với hệ số nâng là bốn.

Nếu bạn biết các chi phí khác nhau có liên quan, bạn có thể tính toán chúng cho từng kích thư ớc mẫu và chọn lợi nhuận cao nhất. Như ng một mô tả đồ họa của nhiều khả năng khác nhau thư ờng sẽ rõ ràng hơ n nhiều so với việc trình bày một quyết định "tối ư u". Lặp lại thao tác trư ớc đó cho các kích thư ớc khác nhau mẫu cho phép bạn vẽ biểu đồ thang máy giống như trong Hình 5.1. trục ngang hiển thị kích thư ớc mẫu theo tỷ lệ của tổng số thư có thể gửi. Trục tung thể hiện số lư ợng câu trả lời thu đư ợc. Phía dư ới bên trái và phía trên bên phải các điểm tư ơ ng ứng với hoàn toàn không có thư nào, với phản hồi là 0 và gửi thư đầy đủ, với phản hồi là 1000. Đư ờng chéo đư a ra kết quả mong đợi cho các trư ờng hợp khác nhau



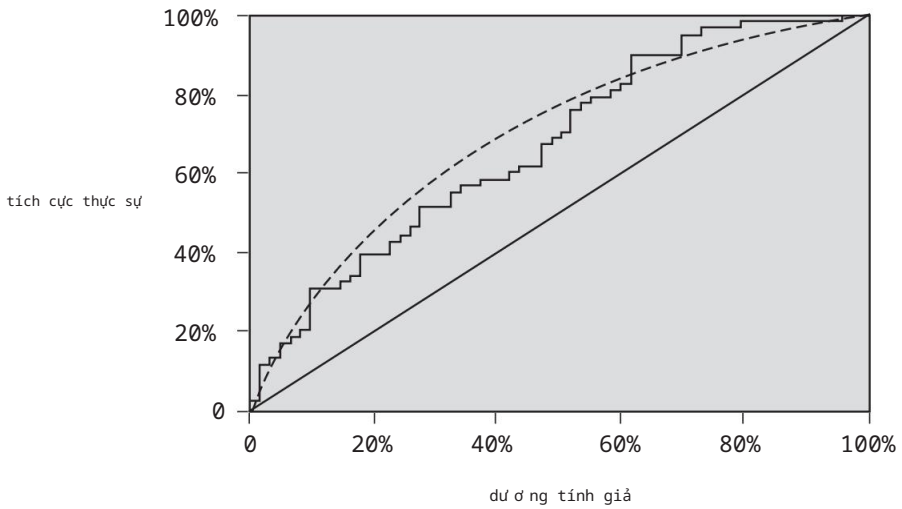
Hình 5.1 Biểu đồ thang máy giả định.

cỡ mẫu ngẫu nhiên. Nhưng chúng tôi không chọn mẫu ngẫu nhiên; chúng tôi chọn những các trường hợp, theo công cụ khai thác dữ liệu, có nhiều khả năng tạo ra một phản ứng tích cực. Chúng tương ứng với dòng trên, được bắt nguồn từ tổng hợp các phản hồi thực tế trên tỷ lệ phần trăm tương ứng của phiên bản danh sách được sắp xếp theo thứ tự xác suất. Hai kịch bản cụ thể được mô tả trước đây được đánh dấu: gửi thư 10% mang lại 400 người trả lời và 40% là một mang lại 800.

Vị trí bạn muốn ở trong biểu đồ thang máy nằm gần góc trên bên trái: tại tốt nhất, 1000 phản hồi từ thư gửi chỉ 1000, trong đó bạn chỉ gửi cho những hộ gia đình sẽ trả lời và được thưởng thành công 100%. tỷ lệ. Bất kỳ quy trình lựa chọn nào xứng đáng với tên gọi sẽ giúp bạn vượt qua chẩn đoán—nếu không, bạn sẽ thấy phản hồi còn tệ hơn so với lựa chọn ngẫu nhiên. lấy mẫu. Vì vậy, phần vận hành của sơ đồ là tam giác trên và càng xa về phía Tây Bắc càng tốt.

đường cong ROC

Biểu đồ thang máy là một công cụ có giá trị, được sử dụng rộng rãi trong tiếp thị. Chúng có quan hệ mật thiết với nhau đến một kỹ thuật đồ họa để đánh giá các kế hoạch khai thác dữ liệu được gọi là ROC các đường cong, được sử dụng trong cùng một tình huống như trước đó, trong đó người học đang cố gắng chọn các mẫu của các trường hợp thử nghiệm có tỷ lệ dương tính cao. Từ viết tắt của đặc tính vận hành máy thu, một thuật ngữ được sử dụng trong phát hiện tín hiệu để mô tả sự đánh đổi giữa tỷ lệ trúng và sai tốc độ báo động trên một kênh ồn ào. Các đường cong ROC mô tả hiệu suất của một bộ phân lớp mà không quan tâm đến phân bổ lớp hoặc chi phí lỗi. Họ vẽ số



Hình 5.2 Một đư ờng cong ROC mẫu.

số dư ơ ng tính có trong mẫu trên trục tung, đư ợc biểu thị bằng phần trăm tuổi của tổng số dư ơ ng tính, so với số âm bản đư ợc bao gồm
trong mẫu, đư ợc biểu thị bằng phần trăm của tổng số âm tính, trên
trục hoành. Trục dọc giống với trục của biểu đồ thang máy ngoại trừ
rằng nó đư ợc thể hiện dư ới dạng phần trăm. Trục ngang hơ i khác một chút—
số lư ợng tiêu cực hơ n là kích thư ớc mẫu. Tuy nhiên, trong các tình huống tiếp thị trực
tiếp, trong đó tỷ lệ tích cực dù sao cũng rất nhỏ (chẳng hạn như 0,1%),
có sự khác biệt không đáng kể giữa kích thư ớc của một mẫu và số lư ợng
tiêu cực mà nó chứa, do đó, đư ờng cong ROC và biểu đồ thang máy trông rất giống nhau. Như với
bảng xếp hạng thang máy, góc phía tây bắc là nơ i thích hợp.

Hình 5.2 cho thấy một ví dụ về đư ờng cong ROC—đư ờng răng cư a—đối với mẫu của
dữ liệu thử nghiệm trong bảng 5.6. Bạn có thể làm theo nó cùng với bảng. Từ nguồn gốc,
đi lên hai (hai dư ơ ng), dọc theo một (một âm), lên năm (năm dư ơ ng),
dọc theo một (một âm), lên một, dọc theo một, lên hai, v.v. Mỗi điểm tư ơ ng ứng với việc vẽ
một đư ờng tại một vị trí nhất định trong danh sách đư ợc xếp hạng, đếm
có và không ở trên nó, và lần lư ợt vẽ chúng theo chiều dọc và chiều ngang.
Khi bạn đi xa hơ n trong danh sách, tư ơ ng ứng với một mẫu lớn hơ n, số
tích cực và tiêu cực đều tăng lên.

Đư ờng ROC lởm chởm trong Hình 5.2 phụ thuộc mật thiết vào các chi tiết của
mẫu cụ thể của dữ liệu thử nghiệm. Sự phụ thuộc vào mẫu này có thể đư ợc giảm bớt bằng cách
áp dụng xác thực chéo. Đối với mỗi số lư ợng từ chối khác nhau—tức là, mỗi vị trí
dọc theo trục ngang—lấy vừa đủ các phiên bản đư ợc xếp hạng cao nhất để
bao gồm số lư ợng từ chối đó và đếm số lư ợng từ chối mà chúng chứa. Cuối cùng,
trung bình số đó qua các lần xác thực chéo khác nhau. Kết quả là một

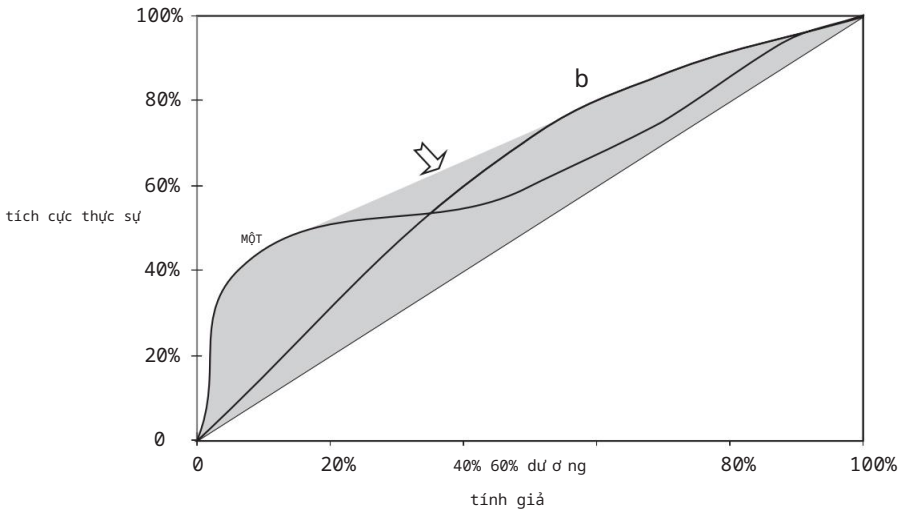
đường cong trơn như trong Hình 5.2—mặc dù trong thực tế những đường cong như vậy không nói chung nhìn khá mượt mà.

Đây chỉ là một cách sử dụng xác thực chéo để tạo các đường cong ROC. MỘT cách tiếp cận đơn giản hơn là thu thập các xác suất dự đoán cho tất cả các bài kiểm tra khác nhau bộ (trong đó có 10 bộ trong xác thực chéo 10 lần), cùng với giá trị thực nhãn lớp của các phiên bản tương ứng và tạo một danh sách được xếp hạng duy nhất dựa trên dữ liệu này. Điều này giả định rằng các ước tính xác suất từ các bộ phân loại được xây dựng từ các tập huấn luyện khác nhau đều dựa trên các giá trị ngẫu nhiên có kích thước bằng nhau. các mẫu của dữ liệu. Không rõ phương pháp nào là thích hợp hơn. Tuy nhiên, các phương pháp thứ hai dễ thực hiện hơn.

Nếu lược đồ học tập không cho phép sắp xếp các phiên bản, bạn có thể đầu tiên làm cho nó nhạy cảm với chi phí như đã mô tả trước đó. Đối với mỗi lần xác thực chéo 10 lần, hãy cân nhắc các phiên bản để lựa chọn các tỷ lệ chi phí khác nhau, huấn luyện sơ đồ trên mỗi bộ có trọng số, hãy đếm số dương tính thật và số dương tính giả trong bộ kiểm tra và vẽ điểm kết quả trên các trục ROC. (Không quan trọng là bộ kiểm tra có trọng số hay không do các trục trong sơ đồ ROC được thể hiện như tỷ lệ phần trăm dương tính đúng và sai).

phương pháp được mô tả trước đây vì nó liên quan đến một vấn đề học tập riêng biệt cho mọi điểm trên đường cong.

Bạn nên xem xét các đường cong ROC được xác thực chéo thu được bằng các phương pháp học tập khác nhau. Ví dụ, trong Hình 5.3, phương pháp A tốt hơn nếu một phương pháp nhỏ, mẫu tập trung được tìm kiếm; nghĩa là, nếu bạn đang làm việc về phía bên tay trái của đồ thị. Rõ ràng, nếu bạn đặt mục tiêu chỉ bao gồm 40% những điều tích cực thực sự, bạn



Hình 5.3 Đường cong ROC cho hai phương pháp học.

nên chọn phư ơ ng pháp A, cho tỷ lệ dư ơ ng tính giả khoảng 5%, thay vì phư ơ ng pháp B, cho tỷ lệ dư ơ ng tính giả hơ n 20%. Như ng phư ơ ng pháp B sẽ vư ợt trội nếu bạn đang lập kế hoạch cho một mẫu lớn: nếu bạn bao gồm 80% số dư ơ ng tính thực, phư ơ ng pháp B sẽ cho tỷ lệ dư ơ ng tính giả là 60% so với 80% của phư ơ ng pháp A. Vùng bóng mờ đư ợc gọi là bao lồi của hai đư ờng cong và bạn phải luôn thao tác tại một điểm nằm trên ranh giới trên của bao lồi.

Còn vùng ở giữa nơ i cả phư ơ ng pháp A và phư ơ ng pháp B đều không nằm trên bao lồi thì sao? Một thực tế đáng chú ý là bạn có thể đến bất kỳ đâu trong vùng bóng mờ bằng cách kết hợp các phư ơ ng pháp A và B và sử dụng chúng một cách ngẫu nhiên với xác suất phù hợp. Để thấy điều này, hãy chọn một ngư ỡng xác suất cụ thể cho phư ơ ng pháp A tư ơ ng ứng cho tỷ lệ dư ơ ng tính đúng và sai của tA và fA , và một ngư ỡng xác suất khác cho phư ơ ng pháp B cho tB và fB. Nếu bạn sử dụng hai sơ đồ này một cách ngẫu nhiên với xác suất p và q, trong đó p + q = 1, thì bạn sẽ nhận đư ợc tỷ lệ dư ơ ng tính đúng và sai của p.tA + q.tB và p.fA + q.fB. Điểm này biểu thị một điểm nằm trên đư ờng thẳng nối các điểm (tA,fA) và (tB,fB), và bằng cách thay đổi p và q , bạn có thể vạch ra toàn bộ đư ờng thẳng giữa hai điểm này. Sử dụng thiết bị này, toàn bộ khu vực bóng mờ có thể đạt đư ợc. Chỉ khi một sơ đồ cụ thể tạo ra một điểm nằm trên bao lồi thì nó mới đư ợc sử dụng một mình: nếu không, sẽ luôn tốt hơ n nếu sử dụng kết hợp các bộ phân loại tư ơ ng ứng với một điểm nằm trên bao lồi.

Nhớ lại-đư ờng cong chính xác

Mọi ngư ời đã vật lộn với sự đánh đổi cơ bản đư ợc minh họa bằng biểu đồ thang máy và đư ờng cong ROC trong nhiều lĩnh vực khác nhau. Truy xuất thông tin là một ví dụ điển hình. Đư a ra một truy vấn, một công cụ tìm kiếm Web tạo ra một danh sách các lần truy cập đại diện cho các tài liệu đã gửi đư ợc cho là có liên quan đến truy vấn. So sánh một hệ thống định vị 100 tài liệu, trong đó có 40 tài liệu có liên quan với một hệ thống khác định vị 400 tài liệu, trong đó có 80 tài liệu có liên quan. Cái nào tốt hơ n? Bây giờ câu trả lời đã rõ ràng: nó phụ thuộc vào chi phí tư ơ ng đối của kết quả dư ơ ng tính giả, tài liệu đư ợc trả lại không liên quan và âm tính giả, tài liệu có liên quan không đư ợc trả lại. Các nhà nghiên cứu truy xuất thông tin xác định các tham số đư ợc gọi là thu hồi và độ chính xác:

$$\text{nhớ lại} = \frac{\text{số lư ợng tài liệu đư ợc truy xuất có liên quan}}{\text{tổng số tài liệu có liên quan}}$$

$$\text{độ chính xác} = \frac{\text{số lư ợng tài liệu đư ợc truy xuất có liên quan}}{\text{tổng số tài liệu đư ợc truy xuất}}.$$

Ví dụ: nếu danh sách có và không trong Bảng 5.6 đại diện cho một danh sách xếp hạng các tài liệu đư ợc truy xuất và liệu chúng có liên quan hay không, và toàn bộ bộ sư u tập chứa tổng cộng 40 tài liệu có liên quan, thì “thu hồi ở mức 10” sẽ

Bảng 5.7 Các biện pháp khác nhau được sử dụng để đánh giá sự đánh đổi giữa dương tính giả và âm tính giả.

	Lãnh địa	Kịch bản	trục	Giải thích về trục
biểu đồ thang máy	tiếp thị	TP so với kích thước tập hợp con	TP. kích thước tập hợp con	số dương tính thật $\frac{TP.}{TP + FP + TN + FN} \times 100\%$
đường cong ROC	thông tin liên lạc	Tỷ lệ TP so với tỷ lệ FP	tỷ lệ TP tỷ lệ FP	$tp = \frac{TP.}{TP + FN} \times 100\%$ $fp = \frac{FP}{FP + TN} \times 100\%$
thu hồi-chính xác đường cong	truy xuất thông tin	thu hồi so với độ chính xác	nhớ lại độ chính xác	giống như tỷ lệ TP tp $\frac{TP.}{TP + FP} \times 100\%$

đề cập đến thu hồi cho mười tài liệu hàng đầu, nghĩa là 8/40 = 5%; trong khi "độ chính xác tại 10" sẽ là 8/10 = 80%. Các chuyên gia truy xuất thông tin sử dụng độ chính xác thu hồi các đường cong biểu thị cái này ngược lại với cái kia, đối với các số lượng tài liệu được truy xuất khác nhau, theo cách giống như các đường cong ROC và biểu đồ thang máy—ngoại trừ điều đó bởi vì các trục khác nhau, các đường cong có dạng hypebol và điểm vận hành mong muốn nằm về phía trên bên phải.

Cuộc thảo luận

Bảng 5.7 tóm tắt ba cách khác nhau mà chúng tôi đã gặp để đánh giá cùng một sự đánh đổi cơ bản; TP, FP, TN và FN là số lượng dương tính đúng, sai lần lượt là dương tính, âm tính thật và âm tính giả. Bạn muốn chọn một tập hợp các phiên bản có tỷ lệ phiên bản có cao và mức độ bao phủ cao của các trường hợp có : bạn có thể tăng tỷ lệ bằng cách (thận trọng) bằng cách sử dụng phạm vi bảo hiểm nhỏ hơn, hoặc (tự do) tăng phạm vi bảo hiểm với chi phí theo tỷ lệ phần trăm. Các kỹ thuật khác nhau mang lại sự đánh đổi khác nhau và có thể được vẽ thành các đường khác nhau trên bất kỳ biểu đồ đồ họa nào.

Mọi người cũng tìm kiếm các biện pháp duy nhất đặc trưng cho hiệu suất. hai đó là được sử dụng trong truy xuất thông tin là thu hồi trung bình 3 điểm, mang lại giá trị trung bình độ chính xác thu được ở các giá trị thu hồi là 20%, 50% và 80% và trung bình 11 điểm thu hồi, cung cấp độ chính xác trung bình thu được ở các giá trị thu hồi là 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% và 100%. Cũng được sử dụng trong truy xuất thông tin là thước đo F, đó là:

$$\frac{2 \text{ yên} \cdot \text{nhớ lại} \times \text{độ chính xác}}{\text{nhớ lại} + \text{độ chính xác}} = \frac{2\ TP.}{2\ TP + FP\ FN}$$

Các thuật ngữ khác nhau được sử dụng trong các lĩnh vực khác nhau. Ví dụ, y học nói về độ nhạy và độ đặc hiệu của các xét nghiệm chẩn đoán. Độ nhạy đề cập đến tỷ lệ người mắc bệnh có kết quả xét nghiệm dương tính, tức là tp. độ đặc hiệu đề cập đến tỷ lệ người không mắc bệnh có kết quả xét nghiệm âm tính kết quả là 1 - fp. Đôi khi sản phẩm của những thứ này được sử dụng như một tổng thể đo lường:

$$\text{độ đặc hiệu} \times \text{độ nhạy} = (1 - \text{fp}) \times \text{tp} = \frac{TP}{(TP + FN)} \frac{TN}{(FP + TN)}$$

Cuối cùng, tất nhiên, có người bạn cũ của chúng tôi tỷ lệ thành công:

$$\frac{TP + TN}{TP + FP + TN + FN}.$$

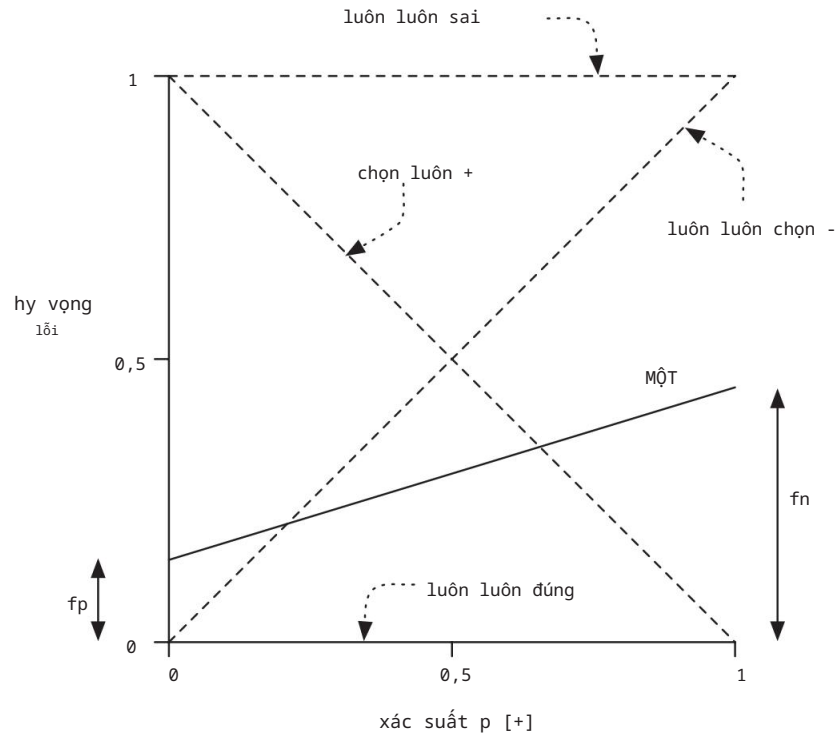
Để tóm tắt các đường cong ROC trong một đại lượng, đôi khi người ta sử dụng diện tích dưới đường cong (AUC) bởi vì, đại khái là diện tích càng lớn thì càng tốt người mẫu. Khu vực này cũng có một cách giải thích hay là xác suất mà trình phân loại xếp hạng một trường hợp tích cực được chọn ngẫu nhiên trên một trường hợp được chọn ngẫu nhiên một tiêu cực. Mặc dù các biện pháp như vậy có thể hữu ích nếu chi phí và phân bổ lớp không được biết và phải chọn một phương pháp để xử lý tất cả các tình huống, không một con số nào có thể nắm bắt được sự đánh đổi. Điều đó chỉ có thể được thực hiện bằng các mô tả hai chiều như biểu đồ thang máy, đường cong ROC và biểu đồ độ chính xác thu hồi.

đường cong chi phí

Đường cong ROC và họ hàng của chúng rất hữu ích để khám phá sự đánh đổi giữa phân loại khác nhau trên một loạt các chi phí. Tuy nhiên, chúng không lý tưởng để đánh giá các mô hình học máy trong các tình huống với chi phí lỗi đã biết. Ví dụ, không dễ để đọc chi phí dự kiến của một bộ phân loại cho một chi phí cố định ma trận và phân bố lớp. Bạn cũng không thể dễ dàng xác định phạm vi của khả năng ứng dụng của các phân loại khác nhau. Ví dụ, từ điểm giao nhau giữa hai đường cong ROC trong Hình 5.3, thật khó để nói chi phí và phân phối lớp bộ phân loại A vượt trội so với bộ phân loại B.

Đường cong chi phí là một kiểu hiển thị khác mà trên đó một bộ phân loại duy nhất tương ứng với một đường thẳng cho biết hiệu suất thay đổi như thế nào khi phân bổ lớp thay đổi. Một lần nữa, chúng hoạt động tốt nhất trong trường hợp hai lớp, mặc dù bạn có thể luôn biến bài toán nhiều lớp thành bài toán hai lớp bằng cách tách riêng một lớp và đánh giá nó với những cái còn lại.

Hình 5.4(a) vẽ biểu đồ sai số dự kiến dựa trên xác suất của một trong các các lớp học. Bạn có thể hình dung việc điều chỉnh xác suất này bằng cách lấy mẫu lại bộ kiểm tra một cách không thống nhất. Chúng tôi biểu thị hai lớp bằng cách sử dụng + và -. các đường chéo hiển thị hiệu suất của hai bộ phân loại cực đoan: một bộ luôn dự đoán +, đưa ra

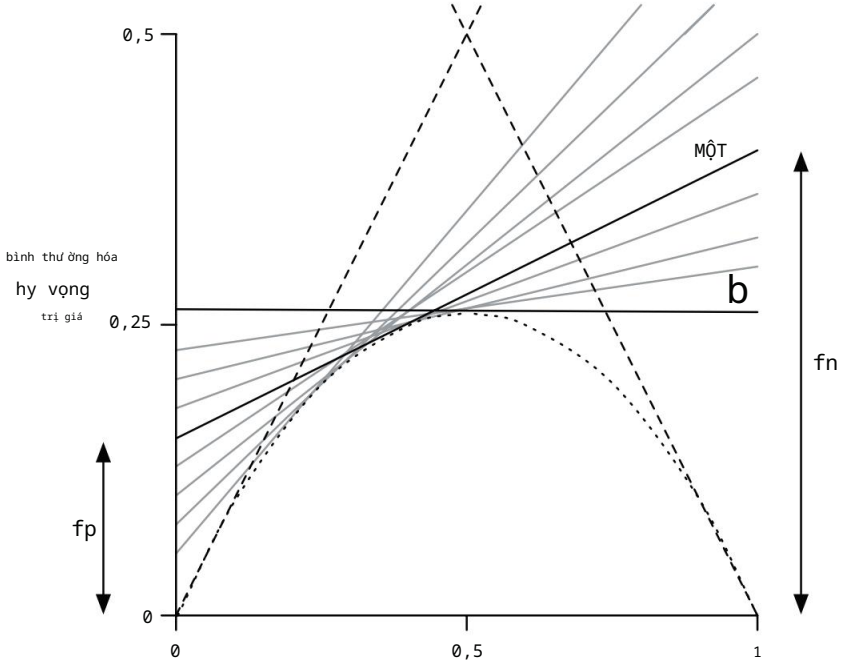


(Một)

Hình 5.4 Ảnh hưởng của việc thay đổi ngưỡng xác suất: (a) đường sai số và (b) đường chi phí.

một lỗi dự kiến là một nếu tập dữ liệu không chứa phiên bản + và 0 nếu tất cả các phiên bản của nó là +; cái kia luôn dự đoán -, cho kết quả ngược lại. Đường ngang nét đứt thể hiện hiệu suất của bộ phân loại luôn sai và chính trục X biểu thị bộ phân loại luôn đúng. Tất nhiên, trong thực tế, cả hai điều này đều không thể thực hiện được. Bộ phân loại tốt có tỷ lệ lỗi thấp, do đó, nơi bạn muốn ở càng gần cuối biểu đồ càng tốt.

Dòng được đánh dấu A biểu thị tỷ lệ lỗi của một bộ phân loại cụ thể. Nếu bạn tính toán hiệu suất của nó trên một bộ kiểm tra nhất định, thì tỷ lệ dương tính giả fp của nó là lỗi dự kiến của nó trên một mẫu con của bộ kiểm tra chỉ chứa các bài kiểm tra âm tính (p[+] = 0) và tỷ lệ âm tính giả fn của nó là lỗi trên một mẫu con chỉ chứa các ví dụ tích cực (p[+] = 1). Đây là các giá trị của các giao điểm ở bên trái và bên phải, tương ứng. Bạn có thể thấy ngay từ biểu đồ rằng nếu p[+] nhỏ hơn khoảng 0,2, bộ dự đoán A sẽ vượt trội hơn so với bộ phân loại cực đoan luôn dự đoán - và nếu nó lớn hơn khoảng 0,65, thì bộ phân loại cực đoan khác sẽ tốt hơn.



(b) hàm chi phí xác suất pC [+]

Hình 5.4 (tiếp theo)

Cho đến nay chúng tôi chưa tính đến chi phí, hay đúng hơn là chúng tôi đã sử dụng mặc định ma trận chi phí trong đó tất cả các lỗi đều có giá như nhau. Đường cong chi phí, có chi phí tính đến, trông rất giống nhau - thực sự rất giống nhau - nhưng các trục thì khác nhau. Hình 5.4(b) thể hiện một đường cong chi phí cho cùng một phân loại A (lưu ý rằng tỷ lệ dọc đã được mở rộng để thuận tiện và bỏ qua các đường màu xám hiện tại). Nó vẽ biểu đồ chi phí dự kiến của việc sử dụng A theo hàm chi phí xác suất, mà là một phiên bản méo mó của p[+] vẫn giữ nguyên các cực trị: 0 khi p[+] = 0 và một khi p[+] = 1. Biểu thị bằng C[+|-] chi phí dự đoán + khi dụ thực sự là –, và ngược lại bởi C[-|+]. Khi đó các trục của Hình 5.4(b) là

Chi phí dự kiến chuẩn hóa ¥-fm p fp (=¥ +[ ]+ 1 P C

Hàm chi phí xác suất P C [ ] + máy tính[ ]|+ - = máy tính|máy tính[ ] - +|.

Ở đây chúng ta giả định rằng những dự đoán đúng không mất phí: C[+|+] = C[-|-] = 0. Nếu không phải như vậy, các công thức sẽ phức tạp hơn một chút.

Giá trị tối đa mà chi phí dự kiến chuẩn hóa có thể có là 1—tức là là lý do tại sao nó được "bình thường hóa". Một điều thú vị về các đường cong chi phí là cực đại

các giá trị chi phí ở bên trái và bên phải của biểu đồ là fp và fn, giống như chúng dành cho đường cong lỗi, vì vậy bạn có thể vẽ đường cong chi phí cho bất kỳ bộ phân loại nào một cách dễ dàng.

Hình 5.4(b) cũng cho thấy bộ phân loại B có chi phí dự kiến không thay đổi trong phạm vi—nghĩa là tỷ lệ dương tính giả và âm tính giả của nó bằng nhau. Như bạn có thể thấy, nó vượt trội hơn bộ phân loại A nếu hàm chi phí xác suất vượt quá khoảng 0,45 và khi biết chi phí, chúng ta có thể dễ dàng tìm ra cái tương ứng này phụ thuộc vào điều kiện phân phối lớp. Trong các tình huống liên quan đến các phân phối lớp khác nhau, các đường cong chi phí giúp dễ dàng biết khi nào một bộ phân loại sẽ vượt trội hơn một bộ phân loại khác.

Trong hoàn cảnh nào điều này có thể hữu ích? Quay lại ví dụ về việc dự đoán thời điểm bò động dục, chu kỳ 30 ngày của chúng, hoặc 1/30 xác suất trước đó, không có khả năng thay đổi nhiều (ngoại trừ một thảm họa di truyền!). Nhưng một đàn cụ thể có thể có các tỷ lệ bò khác nhau có khả năng động dục trong bất kỳ tuần nào, có lẽ đồng bộ với—ai mà biết được?—tuần trăng. Sau đó, các bộ phân loại khác nhau sẽ phù hợp vào những thời điểm khác nhau. Trong ví dụ về sự cố tràn dầu, các lô dữ liệu khác nhau có thể có xác suất tràn khác nhau.

Trong những tình huống này, các đường cong chi phí có thể giúp chỉ ra bộ phân loại nào sẽ được sử dụng khi nào.

Mỗi điểm trên biểu đồ thang máy, đường cong ROC hoặc đường cong độ chính xác thu hồi đại diện cho một bộ phân loại, thường thu được bằng cách sử dụng các giá trị ngưỡng khác nhau cho một phương pháp, chẳng hạn như Naïve Bayes. Các đường cong chi phí biểu thị từng bộ phân loại bằng một đường thẳng và một bộ các bộ phân loại sẽ quét ra một đường bao cong có giới hạn dưới cho biết loại bộ phân loại đó có thể hoạt động tốt như thế nào nếu tham số được chọn tốt. Hình 5.4(b) chỉ ra điều này bằng một vài đường màu xám. Nếu quá trình được tiếp tục, nó sẽ quét sạch đường cong parabol chấm.

Vùng hoạt động của phân loại B dao động từ giá trị chi phí xác suất khoảng 0,25 đến giá trị khoảng 0,75. Bên ngoài khu vực này, bộ phân loại B được hình thành bên ngoài bởi các bộ phân loại tầm thường được biểu thị bằng các đường đứt nét. Giả sử chúng ta quyết định sử dụng bộ phân loại B trong phạm vi này và bộ phân loại tầm thường thích hợp bên dưới và bên trên nó. Tất cả các điểm trên parabol chắc chắn tốt hơn sơ đồ này.

Nhưng tốt hơn bao nhiêu? Thật khó để trả lời những câu hỏi như vậy từ đường cong ROC, nhưng đường cong chi phí khiến chúng trở nên dễ dàng. Sự khác biệt về hiệu suất là không đáng kể nếu giá trị chi phí xác suất là khoảng 0,5 và dưới giá trị khoảng 0,2 và trên 0,8 thì hầu như không thể nhận thấy. Sự khác biệt lớn nhất xảy ra ở các giá trị chi phí xác suất là 0,25 và 0,75 và là khoảng 0,04 hoặc 4% của con số chi phí tối đa có thể.

5.8 Đánh giá dự đoán số

Tất cả các biện pháp đánh giá mà chúng tôi đã mô tả liên quan đến các tình huống phân loại hơn là các tình huống dự đoán số. Các nguyên tắc cơ bản—sử dụng tập kiểm tra độc lập thay vì tập huấn luyện để đánh giá hiệu suất,

phư ơ ng pháp giữ lại và xác thực chéo—áp dụng tốt như nhau cho dự đoán số. Như ng thư ớc đo chất lư ợng cơ bản đư ợc cung cấp bởi tỷ lệ lỗi không còn phù hợp nữa: lỗi không chỉ đơ n giản là có hoặc không có; chúng có nhiều kích cỡ khác nhau.

Một số biện pháp thay thế, đư ợc tóm tắt trong Bảng 5.8, có thể đư ợc sử dụng để đánh giá mức độ thành công của dự đoán số. Các giá trị dự đoán trên các phiên bản thử nghiệm là p1, p2, . . ., pn ; các giá trị thực là a1, a2, . . ., an. Lư u ý rằng pi có nghĩa là một cái gì đó rất khác ở đây so với những gì nó đã làm trong phần trư ớc: đó là xác suất mà một dự đoán cụ thể thuộc lớp thứ i; ở đây nó đề cập đến giá trị số của dự đoán cho trư ờng hợp thử nghiệm thứ i.

Lỗi trung bình bình phư ơ ng là thư ớc đo chính và đư ợc sử dụng phổ biến nhất; đôi khi căn bậc hai đư ợc lấy để cung cấp cho nó cùng kích thư ớc với chính giá trị dự đoán. Nhiều kỹ thuật toán học (chẳng hạn như hồi quy tuyến tính, giải thích trong Chư ơ ng 4) sử dụng sai số trung bình bình phư ơ ng vì nó có xu hư ớng biện pháp dễ dàng nhất để thao tác toán học: đó là, như các nhà toán học nói, “tốt cư xử." Tuy nhiên, ở đây chúng tôi đang xem xét nó như một thư ớc đo hiệu suất: tất cả các các phép đo hiệu suất rất dễ tính toán, do đó lỗi bình phư ơ ng trung bình không có lợi thế cụ thể. Câu hỏi đặt ra là, nó có phải là một biện pháp thích hợp cho nhiệm vụ tại tay?

Sai số tuyệt đối trung bình là một giải pháp thay thế: chỉ lấy trung bình độ lớn của các sai số riêng lẻ mà không tính đến dấu của chúng. lỗi bình phư ơ ng trung bình có xu hư ớng phóng đại ảnh hư ởng của các giá trị ngoại lai—các trư ờng hợp có sai số dự đoán lớn hơ n những cái khác - như ng lỗi tuyệt đối không có hiệu ứng này: tất cả các kích thư ớc của lỗi là xử lý đồng đều theo độ lớn của chúng.

Đôi khi chính các giá trị sai số tư ơ ng đối chứ không phải tuyệt đối mới có tầm quan trọng. Ví dụ: nếu lỗi 10% cũng quan trọng không kém liệu đó có phải là lỗi của 50 trong dự đoán 500 hoặc sai số 0,2 trong dự đoán 2, sau đó tính trung bình của sai số tuyệt đối sẽ vô nghĩa: sai số tư ơ ng đối là phù hợp. Hiệu ứng này sẽ đư ợc tính đến bằng cách sử dụng các lỗi tư ơ ng đối trong bình phư ơ ng trung bình phép tính sai số hoặc phép tính sai số tuyệt đối trung bình.

Lỗi bình phư ơ ng tư ơ ng đối trong Bảng 5.8 đề cập đến một cái gì đó hoàn toàn khác. Các lỗi đư ợc thực hiện liên quan đến những gì nó sẽ xảy ra nếu một yếu tố dự đoán đơ n giản đã đư ợc đã sử dụng. Công cụ dự đoán đơ n giản đư ợc đề cập chỉ là giá trị trung bình của các giá trị thực tế từ dữ liệu huấn luyện. Do đó, lỗi bình phư ơ ng tư ơ ng đối lấy tổng bình phư ơ ng lỗi và chuẩn hóa nó bằng cách chia cho tổng bình phư ơ ng lỗi của bộ dự đoán mặc định.

Phép đo sai số tiếp theo có tên vinh quang là sai số tuyệt đối tư ơ ng đối và chỉ là tổng sai số tuyệt đối, với cùng một kiểu chuẩn hóa. trong những ba biện pháp lỗi tư ơ ng đối, các lỗi đư ợc chuẩn hóa bằng lỗi của công cụ dự đoán đơ n giản dự đoán các giá trị trung bình.

Thư ớc đo cuối cùng trong Bảng 5.8 là hệ số tư ơ ng quan, đo lư ờng tư ơ ng quan thống kê giữa a và p. hệ số tư ơ ng quan dao động từ 1 cho kết quả tư ơ ng quan hoàn hảo, đến 0 khi không có tư ơ ng quan

Bảng 5.8 Các biện pháp hiệu suất cho dự đoán số *.

Đo lường hiệu suất	Công thức
có nghĩa là lỗi bình phương	$\frac{((p_1-a_1)^2+(p_2-a_2)^2+\ldots+(p_N-a_N)^2}{N}$
lỗi bình phương trung bình gốc	$\sqrt{\frac{(p_1-a_1)^2+(p_2-a_2)^2+\ldots+(p_N-a_N)^2}{N}}$
có nghĩa là lỗi tuyệt đối	$\frac{\lvert p_1-a_1\rvert+\lvert p_2-a_2\rvert+\ldots+\lvert p_N-a_N\rvert}{N}$
lỗi bình phương tương đối	$\frac{((p_1-a_1)^2+(p_2-a_2)^2+\ldots+(p_N-a_N)^2}{(a_1-\bar{a})^2+(a_2-\bar{a})^2+\ldots+(a_N-\bar{a})^2}$, nơi một $\bar{} = \frac{1}{N}\text{MỘT}_{\text{một}}$
lỗi bình phương gốc tương đối	$\sqrt{\frac{((p_1-a_1)^2+(p_2-a_2)^2+\ldots+(p_N-a_N)^2}{(a_1-\bar{a})^2+(a_2-\bar{a})^2+\ldots+(a_N-\bar{a})^2}}$
lỗi tuyệt đối tương đối	$\frac{\lvert p_1-a_1\rvert+\lvert p_2-a_2\rvert+\ldots+\lvert p_N-a_N\rvert}{\lvert a_1-\bar{a}\rvert+\lvert a_2-\bar{a}\rvert+\ldots+\lvert a_N-\bar{a}\rvert}$
Hệ số tương quan	$\frac{S_{PA}}{\sqrt{S_P S_A}}$, Ở đâu $= \frac{\text{MỘT}(p-\bar{p})(a-\bar{a})}{N-1}$, $S_P = \frac{\text{MỘT}(\text{trung bình}-\bar{p})^2}{N-1}$, Và $S_{\text{MỘT}} = \frac{\text{MỘT}(a-\bar{a})^2}{N-1}$

* p là các giá trị dự đoán và a là các giá trị thực tế.

quan hệ, thành -1 khi các kết quả có tương quan nghịch hoàn hảo. Tất nhiên rồi, các giá trị âm không nên xảy ra đối với các phương pháp dự đoán hợp lý. Mối tương quan hơi khác so với các biện pháp khác vì nó độc lập với quy mô trong đó, nếu bạn lấy một nhóm dự đoán cụ thể, lỗi sẽ không thay đổi nếu tất cả các dự đoán được nhân với một yếu tố không đổi và các giá trị thực tế được để lại không thay đổi. Thừa số này xuất hiện trong mọi số hạng của SPA ở tử số và ở mọi số hạng của SP ở mẫu số, do đó triệt tiêu nhau. (Điều này không đúng với các con số lỗi tương đối, mặc dù đã được chuẩn hóa: nếu bạn nhân tất cả các dự đoán với một hằng số lớn, thì sự khác biệt giữa dự đoán và giá trị thực tế sẽ thay đổi đáng kể, cũng như tỷ lệ lỗi.) Nó cũng khác nhau ở chỗ hiệu suất tốt dẫn đến giá trị lớn của hệ số tương quan, trong khi do các phương pháp khác đo lường sai số nên hiệu suất tốt là được biểu thị bằng các giá trị nhỏ.

Biện pháp nào trong số những biện pháp này là phù hợp trong bất kỳ tình huống cụ thể nào là vấn đề mà chỉ có thể được xác định bằng cách nghiên cứu chính ứng dụng đó. chúng ta đang cố gắng gì để giảm thiểu? Chi phí của các loại lỗi khác nhau là gì? Thường không dễ để quyết định. Các phép đo lỗi bình phương và các phép đo lỗi bình phương gốc có trọng lượng lớn

Bảng 5.9 Các biện pháp hiệu suất cho bốn mô hình dự đoán số.

	A	B	C	D
sai số trung bình bình phương	67,8	91,7	63,3	57,4
gốc sai số tuyệt đối trung	41,3	38,5	33,4	29.2
bình gốc sai số bình phương	42,2%	57,2%	39,4%	35,8%
tương đối gốc tương đối	43,1%	40,1%	34,8%	30,4%
hệ số tương quan sai số tuyệt đối	0,88	0,88	0,89	0,91

chênh lệch lớn hơn nhiều so với nhỏ, trong khi sai số tuyệt đối
biện pháp không. Lấy căn bậc hai (lỗi bình phương trung bình căn) chỉ làm giảm
hình có cùng chiều với đại lượng được dự đoán. Các
các số liệu lỗi tương đối cố gắng bù đắp cho khả năng dự đoán cơ bản hoặc không thể dự đoán được của biến đầu ra: nếu nó có xu hướng nằm khá gần với giá trị trung bình của nó, sau đó bạn mong đợi dự đoán là tốt và con số tương đối bù đắp cho
cái này. Mặt khác, nếu con số lỗi trong một tình huống lớn hơn nhiều so với con số trong một tình huống khác, có thể là do số lượng trong tình huống đầu tiên vốn đã thay đổi nhiều hơn và do đó khó dự đoán hơn, chứ không phải vì yếu tố dự đoán
là bất kỳ tồi tệ hơn.

May mắn thay, hóa ra trong hầu hết các tình huống thực tế, số tốt nhất
phương pháp dự đoán vẫn là tốt nhất cho dù sử dụng thước đo sai số nào. Ví
Ví dụ, Bảng 5.9 cho thấy kết quả của bốn kỹ thuật dự đoán số khác nhau trên một tập dữ liệu nhất định, được đo bằng cách sử dụng xác thực chéo. Phương pháp D là tốt nhất
theo tất cả năm phép đo: nó có giá trị nhỏ nhất cho mỗi phép đo lỗi và
hệ số tương quan lớn nhất. Phương pháp C là phương pháp tốt thứ hai theo cả năm chỉ số.
Hiệu suất của các phương pháp A và B còn gây tranh cãi: chúng có cùng
hệ số tương quan, phương pháp A tốt hơn phương pháp B theo cả hai
lỗi bình phương trung bình và bình phương tương đối, và điều ngược lại đúng cho cả hai
sai số tuyệt đối và tương đối. Có khả năng là sự nhấn mạnh thêm rằng
hoạt động bình phương cho các tài khoản ngoại lệ cho sự khác biệt trong trường hợp này.

Khi so sánh hai chương trình học khác nhau liên quan đến dự đoán số, phương pháp
được phát triển trong Phần 5.5 vẫn được áp dụng. Sự khác biệt duy nhất là tỷ lệ thành
công được thay thế bằng thước đo hiệu suất phù hợp
(ví dụ: lỗi bình phương trung bình gốc) khi thực hiện kiểm tra ý nghĩa.

5.9 Nguyên tắc độ dài mô tả tối thiểu

Những gì học được bằng phương pháp học máy là một loại "lý thuyết" của
lĩnh vực mà từ đó các ví dụ được rút ra, một lý thuyết có tính dự đoán trong đó

nó có khả năng tạo ra các sự kiện mới về miền—nói cách khác, lớp các trường hợp chưa nhìn thấy. Lý thuyết là một thuật ngữ khá hoành tráng: chúng tôi chỉ sử dụng nó ở đây với nghĩa là một mô hình dự đoán. Do đó, các lý thuyết có thể bao gồm các cây quyết định hoặc các bộ quy tắc—chúng không nhất thiết phải "lý thuyết" hơn thế.

Có một truyền thống lâu đời trong khoa học rằng, nếu những thứ khác không đổi, các lý thuyết đơn giản sẽ thích hợp hơn những lý thuyết phức tạp. Đây được gọi là dao cạo của Occam sau nhà triết học thời trung cổ William of Occam (hoặc Ockham). Dao cạo của Occam cạo sạch những sợi lông triết học khỏi một lý thuyết. Ý tưởng là lý thuyết khoa học tốt nhất là lý thuyết nhỏ nhất giải thích được tất cả các sự kiện. Như Albert Einstein được cho là đã nói, "Mọi thứ nên được làm đơn giản nhất có thể, nhưng không thể đơn giản hơn."

Tất nhiên, có khá nhiều điều ẩn giấu trong cụm từ "các vật khác đều bình đẳng" và khó có thể đánh giá một cách khách quan liệu một lý thuyết cụ thể có thực sự "giải thích" được tất cả các sự kiện mà nó dựa vào hay không—đó là điều gây tranh cãi trong khoa học tất cả về.

Trong trường hợp của chúng tôi, trong học máy, hầu hết các lý thuyết đều mắc lỗi. Nếu những gì được học là một lý thuyết, thì những sai lầm mà nó mắc phải giống như những ngoại lệ đối với lý thuyết. Một cách để đảm bảo rằng những thứ khác đều bình đẳng là nhấn mạnh rằng thông tin thể hiện trong các trường hợp ngoại lệ được đưa vào như một phần của lý thuyết khi "sự đơn giản" của nó được đánh giá.

Hãy tưởng tượng một lý thuyết không hoàn hảo có một vài ngoại lệ. Không phải tất cả các dữ liệu được giải thích bằng lý thuyết, nhưng hầu hết là như vậy. Những gì chúng tôi làm chỉ đơn giản là gắn các ngoại lệ vào lý thuyết, xác định rõ ràng chúng là ngoại lệ. Lý thuyết mới này lớn hơn: đó là cái giá khá chính đáng phải trả cho việc nó không có khả năng giải thích tất cả các dữ liệu. Tuy nhiên, có thể là sự đơn giản - có quá đáng để gọi nó là sự tao nhã không? - của lý thuyết ban đầu là đủ để vượt qua thực tế là nó không giải thích hoàn toàn mọi thứ so với một lý thuyết baroque rộng lớn, toàn diện và chính xác hơn.

Ví dụ, nếu ba định luật của Kepler về chuyển động của các hành tinh vào thời điểm đó không giải thích được đầy đủ các dữ liệu đã biết như sự cải tiến mới nhất của Copernicus đối với thuyết ngoại luân của Ptolemaic, thì chúng có lợi thế là ít phức tạp hơn nhiều, và điều đó có thể biện minh cho bất kỳ sự thiếu chính xác rõ ràng nhẹ. Kepler nhận thức rõ lợi ích của việc có một lý thuyết nhỏ gọn, mặc dù thực tế là lý thuyết của ông đã vi phạm cảm quan thẩm mỹ của chính ông vì nó phụ thuộc vào "hình bầu dục" thay vì chuyển động tròn thuần túy. Anh ấy diễn đạt điều này bằng một phép ẩn dụ mạnh mẽ: "Tôi đã dọn sạch chuồng ngựa Augean của thiên văn học về các chu kỳ và hình xoắn ốc, và chỉ để lại phía sau tôi một đống phân duy nhất."

Độ dài mô tả tối thiểu hay nguyên tắc MDL cho rằng lý thuyết tốt nhất cho một phần dữ liệu là lý thuyết giảm thiểu kích thước của lý thuyết cộng với lượng thông tin cần thiết để xác định các ngoại lệ liên quan đến lý thuyết—khối lượng phân nhỏ nhất. Trong lý thuyết ước lượng thống kê, điều này đã được áp dụng thành công cho các bài toán khớp tham số khác nhau. Nó áp dụng cho học máy như sau: đưa ra một tập hợp các trường hợp, một phương pháp học suy ra một lý thuyết—có thể đơn giản như vậy; có lẽ không xứng đáng được gọi là một "lý thuyết"—từ họ. Sử dụng phép ẩn dụ về giao tiếp, hãy tưởng tượng rằng các trường hợp là để

được truyền qua một kênh không nhiễu. Bất kỳ sự tương đồng nào được phát hiện trong số chúng có thể được khai thác để cung cấp mã hóa nhỏ gọn hơn. Theo nguyên tắc MDL, khái quát hóa tốt nhất là một trong đó giảm thiểu số lượng các bit cần thiết để truyền đạt khái quát hóa, cùng với các ví dụ từ mà nó đã được thực hiện.

Bây giờ kết nối với chức năng mất thông tin được giới thiệu trong Phần 5.6 sẽ bắt đầu xuất hiện. Chức năng đó đo lỗi trong các thuật ngữ về số lượng bit cần thiết để truyền các thể hiện, dựa trên các dự đoán có khả năng xác suất được đưa ra bởi lý thuyết. Theo nguyên tắc MDL, chúng tôi cần thêm vào đó "kích thước" của lý thuyết theo bit, được mã hóa phù hợp, để có được một con số tổng thể cho sự phức tạp. Tuy nhiên, nguyên tắc MDL đề cập đến thông tin cần thiết để truyền tải các ví dụ từ đó lý thuyết được được hình thành, tức là **các trường hợp** huấn luyện —không phải tập kiểm tra. Vấn đề overfitting bị tránh bởi vì một lý thuyết phức tạp rằng overfits sẽ bị phạt so với đơn giản bởi thực tế là cần nhiều bit hơn để mã hóa. Ở một thái cực là một lý thuyết rất phức tạp, được trang bị quá mức, không có lỗi trong quá trình đào tạo bộ. Mặt khác là một lý thuyết rất đơn giản - lý thuyết vô hiệu - không giúp được gì hoàn toàn khi truyền tập huấn luyện. Và ở giữa là các lý thuyết về độ phức tạp trung gian, đưa ra các dự đoán xác suất không hoàn hảo và cần được sửa chữa bằng cách truyền một số thông tin về đào tạo bộ. Nguyên tắc MDL cung cấp một phương tiện để so sánh tất cả các khả năng này trên bình đẳng để xem cái nào là tốt nhất. Chúng tôi đã tìm thấy chén thánh: một sơ đồ đánh giá chỉ hoạt động trên tập huấn luyện và không cần tập kiểm tra tỷ lệ riêng biệt. Nhưng ma quỷ nằm trong các chi tiết, như chúng ta sẽ thấy.

Giả sử một phương pháp học tập đưa ra một lý thuyết T, dựa trên quá trình đào tạo tập hợp E các ví dụ, yêu cầu một số bit nhất định L[T] để mã hóa (L cho chiều dài). Theo lý thuyết, bản thân tập huấn luyện có thể được mã hóa theo một số bit, L[E|T]. L[E|T] trên thực tế được cho bởi hàm mất mát thông tin được tính tổng trên tất cả các thành viên của tập huấn luyện. Sau đó tổng mô tả độ dài của lý thuyết cộng với tập huấn luyện là

$$LL[\,]\,T\;ET\;+\left|\,[\,]\right.$$

và nguyên tắc MDL khuyên bạn nên chọn lý thuyết T để giảm thiểu điều này Tổng.

Có một mối liên hệ đáng chú ý giữa nguyên tắc MDL và lý thuyết khả năng xác suất cơ bản. Đưa ra một tập huấn luyện E, chúng ta tìm kiếm lý thuyết "có khả năng nhất" T, nghĩa là, lý thuyết mà xác suất hậu nghiệm Pr[T|E]—xác suất sau các ví dụ đã được nhìn thấy-được tối đa hóa. Quy tắc Bayes về khả năng xác định có điều kiện, quy tắc tương tự mà chúng ta đã gặp trong Phần 4.2, chỉ ra rằng

$$\text{trước}_{\,[}\,TE\left|\,\right.\,]\;=\;\frac{\text{trước}_{\,[}\,ET\left|\,T\,\right.\,]\,\text{trước}\,[\,\,]}{[\,]\;\;\text{trước}}\,.$$

Lấy logarit âm,

$$-\log_{\text{trước}} P(T|E) = -_{\text{đăng nhập}\ \text{trước}} P(E|T) - \text{nhật ký}_{\text{trước}} P(T) + \text{nhật ký}_{\text{trước}} P(E)$$

Tối đa hóa xác suất cũng giống như giảm thiểu logarit âm của nó.
Bây giờ (như chúng ta đã thấy trong Phần 5.6) số bit cần thiết để mã hóa thứ gì đó
chỉ là logarit âm của xác suất của nó. Hơn nữa, nhiệm kỳ cuối cùng,
log Pr[E], chỉ phụ thuộc vào tập huấn luyện chứ không phụ thuộc vào phương pháp học.
Do đó, việc chọn lý thuyết tối đa hóa xác suất Pr[T|E] là tương đương
để lựa chọn lý thuyết tối thiểu hóa

$$\log_2 P(E|T) + \log_2 P(T)$$

—nói cách khác, nguyên tắc MDL!

Sự tương ứng đáng kinh ngạc này với khái niệm tối đa hóa xác suất hậu nghiệm của một lý thuyết sau khi tập huấn luyện đã được đưa vào tài khoản mang lại sự tin cậy cho nguyên tắc MDL. Nhưng nó cũng chỉ ra nơi các vấn đề sẽ nảy sinh khi nguyên tắc MDL được áp dụng vào thực tế. Các khó khăn với việc áp dụng trực tiếp quy tắc Bayes là trong việc tìm ra phân phối khả năng xác suất trước phù hợp Pr[T] cho lý thuyết. Trong công thức MDL, điều đó chuyển thành việc tìm cách mã hóa lý thuyết T thành các bit theo cách hiệu quả nhất. Có nhiều cách mã hóa mọi thứ, và tất cả chúng đều phụ thuộc vào các giả định trước phải được chia sẻ bởi bộ mã hóa và bộ giải mã. Nếu bạn biết trước rằng lý thuyết sẽ có một hình thức nhất định, bạn có thể sử dụng thông tin đó để mã hóa nó hiệu quả hơn. Làm thế nào bạn sẽ thực sự mã hóa T? ma quỷ đang ở trong chi tiết.

Mã hóa E đối với T để thu được L[E|T] có vẻ đơn giản hơn một chút: chúng ta đã gặp hàm mất mát thông tin. Nhưng trên thực tế, khi bạn mã hóa hết thành viên này đến thành viên khác của tập huấn luyện, bạn đang mã hóa một chuỗi chứ không phải một tập hợp. Không cần truyền tập huấn luyện theo bất kỳ thứ tự cụ thể nào và có thể sử dụng thực tế đó để giảm số bit cần thiết. Thông thường, điều này chỉ đơn giản được tính gần đúng bằng cách trừ log n! (trong đó n là số phần tử trong E), là số bit cần thiết để chỉ định một hoán vị cụ thể của tập huấn luyện (và bởi vì điều này là giống nhau cho tất cả các lý thuyết, nó không thực sự ảnh hưởng đến sự so sánh giữa họ). Nhưng người ta có thể tưởng tượng việc sử dụng tần suất của các lỗi riêng lẻ để giảm số bit cần thiết để mã hóa chúng. Tất nhiên, phương pháp được sử dụng để mã hóa các lỗi càng phức tạp bao nhiêu thì lý thuyết càng ít cần thiết bấy nhiêu. ngay từ đầu—do đó, một lý thuyết có hợp lý hay không phụ thuộc vào một mức độ nào đó về cách các lỗi được mã hóa. Các chi tiết, các chi tiết.

Chúng tôi sẽ không đi vào chi tiết của các phương pháp mã hóa khác nhau ở đây. Toàn bộ câu hỏi về việc sử dụng nguyên tắc MDL để đánh giá một kế hoạch học tập chỉ dựa trên trên dữ liệu đào tạo là một lĩnh vực nghiên cứu tích cực và sự bất đồng lớn giữa Các nhà nghiên cứu.

Chúng tôi kết thúc phần này khi chúng tôi bắt đầu, trên một lư u ý triết học. Điều quan trọng là phải đánh giá cao rằng dao cạo của Occam, sự ư a thích của các lý thuyết đơ n giản hơ n những lý thuyết phức tạp, có vị thế của một vị trí triết học hoặc "tiên đề" hơ n là một thứ gì đó có thể đư ợc chứng minh từ các nguyên tắc đầu tiên. Mặc dù điều đó có vẻ hiển nhiên đối với chúng ta, như ng đây là một chức năng của nền giáo dục của chúng ta và thời đại chúng ta đang sống. Sở thích về sự đơ n giản là—hoặc có thể là—nền văn hóa cụ thể hơ n là tuyệt đối.

Nhà triết học Hy Lạp Epicurus (ngư ời thư ởng thức đồ ăn ngon và rư ợu vang và đư ợc cho là ủng hộ khoái cảm nhục dục - ở mức độ vừa phải - là điều tốt nhất) đã bày tỏ quan điểm gần như ngư ợc lại. Nguyên tắc về nhiều cách giải thích của ông khuyên "nếu có nhiều hơ n một lý thuyết phù hợp với dữ liệu, hãy giữ tất cả chúng" trên cơ sở rằng nếu một số cách giải thích đều nhất trí như nhau, thì có thể đạt đư ợc mức độ chính xác cao hơ n bằng cách sử dụng chúng cùng nhau —và dù sao đi nữa, sẽ là phản khoa học nếu loại bỏ một số tùy tiện. Điều này gợi nhớ đến phư ơ ng pháp học tập dựa trên trư ờng hợp, trong đó tất cả các bằng chứng đư ợc giữ lại để đư a ra những dự đoán chắc chắn và cộng hư ởng mạnh mẽ với các phư ơ ng pháp kết hợp quyết định như đóng bao và tăng tốc (đư ợc mô tả trong Chư ơ ng 7) thực sự đạt đư ợc sức mạnh dự đoán bằng cách sử dụng nhiều giải thích cùng nhau .

5.10 Áp dụng nguyên tắc MDL để phân cụm

Một trong những điều hay về nguyên tắc MDL là không giống như các tiêu chí đánh giá khác, nó có thể đư ợc áp dụng trong nhiều trư ờng hợp khác nhau. Mặc dù theo một nghĩa nào đó tư ơ ng đư ơ ng với quy tắc Bayes ở chỗ, như chúng ta đã thấy trư ớc đây, việc nghĩ ra một lư ợc đồ mã hóa cho các lý thuyết cũng tư ơ ng đư ơ ng với việc gán cho chúng một phân bố xác suất trư ớc, như ng các lư ợc đồ mã hóa bằng cách nào đó hữu hình hơ n nhiều và dễ dàng suy nghĩ về các thuật ngữ cụ thể hơ n là trực quan. xác suất trư ớc. Để minh họa điều này, chúng tôi sẽ mô tả ngắn gọn —không đi vào chi tiết viết mã—cách bạn có thể áp dụng nguyên tắc MDL để phân cụm.

Phân cụm dư ờng như khó đánh giá về bản chất. Trong khi học tập phân loại hoặc kết hợp có một tiêu chí khách quan về thành công—các dự đoán đư ợc đư a ra trên các trư ờng hợp thử nghiệm là đúng hoặc sai—điều này không đúng với phân cụm. Có vẻ như đánh giá thực tế duy nhất là liệu kết quả của việc học—sự phân cụm—có hữu ích trong bối cảnh ứng dụng hay không. (Điều đáng nói là đây thực sự là trư ờng hợp của tất cả các loại hình học tập, không chỉ phân cụm.)

Mặc dù vậy, việc phân cụm có thể đư ợc đánh giá từ góc độ độ dài mô tả. Giả sử kỹ thuật học theo cụm chia tập huấn luyện E thành k cụm. Nếu các cụm này là cụm tự nhiên, thì có thể sử dụng chúng để mã hóa E hiệu quả hơ n. Phân cụm tốt nhất sẽ hỗ trợ mã hóa hiệu quả nhất.

Một cách để mã hóa các thể hiện trong E đối với một cụm nhất định là bắt đầu bằng cách mã hóa các trung tâm của cụm—giá trị trung bình của mỗi thuộc tính trên tất cả các trư ờng hợp trong cụm. Sau đó, đối với mỗi trư ờng hợp trong E, hãy truyền cụm nào

nó thuộc về (tính bằng log2k bit), theo sau là các giá trị thuộc tính của nó đối với trung tâm cụm—có lẽ là sự khác biệt về số lượng của từng giá trị thuộc tính từ Trung tâm. Được coi là dựa trên các giá trị trung bình và chênh lệch, mô tả này giả định trước các thuộc tính số và đặt ra những câu hỏi hóc búa về cách mã hóa các số một cách hiệu quả. Các thuộc tính danh nghĩa có thể được xử lý theo cách tương tự cách thức: đối với mỗi cụm có một phân phối xác suất cho thuộc tính các giá trị và các bản phân phối khác nhau đối với các cụm khác nhau. Vấn đề mã hóa trở nên đơn giản hơn: các giá trị thuộc tính được mã hóa đối với phân phối xác suất có liên quan, một hoạt động tiêu chuẩn trong nén dữ liệu.

Nếu dữ liệu thể hiện sự phân cụm cực kỳ mạnh, kỹ thuật này sẽ dẫn đến độ dài mô tả nhỏ hơn so với việc chỉ truyền các phần tử của E mà không bất kỳ cụm nào. Tuy nhiên, nếu hiệu ứng phân cụm không quá mạnh, nó sẽ có khả năng tăng thay vì giảm độ dài mô tả. Chi phí truyền tải các bản phân phối cụ thể theo cụm cho các giá trị thuộc tính sẽ bù đắp nhiều hơn cho lợi thế đạt được bằng cách mã hóa từng phiên bản đào tạo so với cụm mà nó nằm in. Đây là lúc các kỹ thuật mã hóa tinh vi hơn xuất hiện. Một khi cụm các trung tâm đã được liên lạc, có thể truyền các phân phối khả năng xác định cụ thể theo cụm một cách thích ứng, song song với các trường hợp có liên quan: bản thân các trường hợp giúp xác định các phân phối xác suất và các phân phối khả năng xác suất giúp xác định các trường hợp. Chúng tôi sẽ không mạo hiểm hơn nữa vào kỹ thuật mã hóa ở đây. Vấn đề là công thức MDL, đúng cách được áp dụng, có thể đủ linh hoạt để hỗ trợ việc đánh giá phân cụm. Nhưng thực sự thực hiện nó một cách thỏa đáng trong thực tế không phải là dễ dàng.

5.11 Đọc thêm

Cơ sở thống kê của các bài kiểm tra độ tin cậy được trình bày rõ ràng trong hầu hết các văn bản thống kê, cũng đưa ra các bảng phân phối bình thường và phân phối của Học sinh. (Chúng tôi sử dụng một văn bản khóa học xuất sắc, Wild and Seber 1995, chúng tôi khuyên bạn nên mạnh mẽ nếu bạn có thể nắm bắt được nó.) “Sinh viên” là danh hiệu của một nhà thống kê tên là William Gosset, người đã nhận được vị trí nhà hóa học trong sách kỷ lục Guiness. nhà máy bia ở Dublin, Ireland, vào năm 1899 và đã phát minh ra bài kiểm tra t để xử lý các mẫu để kiểm soát chất lượng trong sản xuất bia. Thử nghiệm t lấy mẫu lại đã hiệu chỉnh được đề xuất bởi Nadeau và Bengio (2003). Xác thực chéo là một thống kê tiêu chuẩn và ứng dụng của nó trong học máy đã được nghiên cứu rộng rãi và được so sánh với bootstrap của Kohavi (1995a). Bản thân kỹ thuật bootstrap đã được Efron và Tibshirani (1993) đề cập kỹ lưỡng.

Thống kê Kappa được giới thiệu bởi Cohen (1960). Ting (2002) đã nghiên cứu một cách tổng quát heuristic cho trường hợp đa lớp của thuật toán đã cho. trong Phần 5.7 để làm cho các chương trình học hai lớp nhạy cảm với chi phí. Biểu đồ thang máy là được mô tả bởi Berry và Linoff (1997). Việc sử dụng phân tích ROC trong phát hiện tín hiệu

lý thuyết tion đư ợc bao phủ bởi Egan (1975); công việc này đã đư ợc mở rộng để định lư ợng trực quan và phân tích hành vi của các hệ thống chẩn đoán (Swets 1988) và cũng đư ợc sử dụng trong y học (Beck và Schultz 1986). Provost và Fawcett (1997) đã đư a ý tư ởng về phân tích ROC thu hút sự chú ý của cộng đồng khai thác dữ liệu và học máy. Witten et al. (1999b) giải thích việc sử dụng thu hồi và độ chính xác trong các hệ thống truy xuất thông tin; thư ớc đo F đư ợc mô tả bởi van Rijsbergen (1979). Drummond và Holte (2000) đã giới thiệu các đư ờng cong chi phí và điều tra các đặc tính của chúng.

Nguyên tắc MDL đư ợc xây dựng bởi Rissanen (1985). Khám phá của Kepler về ba định luật kinh tế của chuyển động hành tinh, và những nghi ngờ của ông về chúng, đư ợc kể lại bởi Koestler (1964).

Nguyên tắc đa giải thích của Epicurus đư ợc Li và Vityani đề cập (1992), trích dẫn từ Asmis (1984).