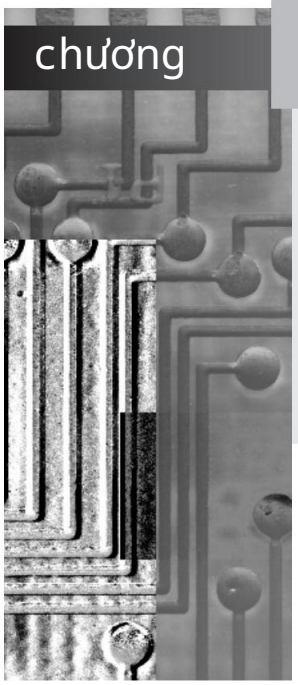
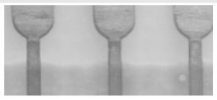
chương 7



# Biến đổi:

## Kỹ thuật đầu vào và đầu ra



Trong chương trước, chúng ta đã xem xét một loạt các phương pháp học máy: cây quyết định, quy tắc quyết định, mô hình tuyến tính, lược đồ dựa trên cá thể, kỹ thuật dự đoán số, thuật toán phân cụm và mạng Bayes. Tất cả đều là những kỹ thuật chắc chắn, mạnh mẽ có thể áp dụng rõ ràng cho các vấn đề khai thác dữ liệu thực tế.

Nhưng việc khai thác dữ liệu thành công liên quan đến nhiều thứ hơn là chọn một thuật toán học tập và chạy nó trên dữ liệu của bạn. Đối với một điều, nhiều phương pháp học tập có các tham số khác nhau và các giá trị phù hợp phải được chọn cho các tham số này. Trong hầu hết các trường hợp, kết quả có thể được cải thiện rõ rệt bằng cách lựa chọn giá trị tham số phù hợp và lựa chọn phù hợp phụ thuộc vào dữ liệu có sẵn. Ví dụ, cây quyết định có thể được cắt tỉa hoặc không cắt tỉa, và trong trường hợp trước, một tham số cắt tỉa có thể phải được chọn. Trong phương pháp học tập dựa trên cá thể k-láng giềng gần nhất, một giá trị cho k sẽ phải được chọn. Tổng quát hơn, bản thân sơ đồ học tập sẽ phải được chọn từ một loạt các sơ đồ có sẵn. Trong mọi trường hợp, các lựa chọn đúng phụ thuộc vào chính dữ liệu.

Bạn nên thử một số lược đồ học tập và một số giá trị tham số trên dữ liệu của mình và xem cái nào hoạt động tốt nhất. Nhưng hãy cẩn thận! sự lựa chọn tốt nhất

không nhất thiết là dữ liệu hoạt động tốt nhất trên dữ liệu huấn luyện. Chúng ta có đã nhiều lần cảnh báo về vấn đề trang bị quá mức, trong đó một mô hình đã học được liên kết quá chặt chẽ với dữ liệu đào tạo cụ thể mà từ đó nó được xây dựng. Nó là không đúng khi cho rằng hiệu suất trên dữ liệu huấn luyện thể hiện trung thực mức hiệu suất có thể được mong đợi trên dữ liệu mới mà mô hình học được sẽ được áp dụng trong thực tế.

May mắn thay, chúng tôi đã gặp phải giải pháp cho vấn đề này trong Chương 5. Có hai phương pháp tốt để ước tính hiệu suất đúng như mong đợi của một lược đồ học tập: sử dụng một tập dữ liệu lớn khá riêng biệt từ dữ liệu đào tạo, trong trường hợp dữ liệu phong phú và xác thực chéo (Mục 5.3), nếu dữ liệu khan hiếm. Trong trường hợp thứ hai, việc xác thực chéo 10 lần duy nhất thường được sử dụng trong thực tế, mặc dù để có được ước tính đáng tin cậy hơn, toàn bộ quy trình phải được lặp lại 10 lần. Khi các tham số phù hợp đã được chọn cho lược đồ học tập, hãy sử dụng toàn bộ tập huấn luyện—tất cả các trường hợp đào tạo có sẵn—để tạo ra mô hình đã học cuối cùng nhằm được áp dụng cho dữ liệu mới.

Lưu ý rằng hiệu suất thu được với giá trị tham số đã chọn trong quá trình quá trình điều chỉnh không phải là ước tính đáng tin cậy về hiệu suất của mô hình cuối cùng, bởi vì mô hình cuối cùng có khả năng phù hợp với dữ liệu đã được sử dụng để điều chỉnh. Để xác định xem nó sẽ hoạt động tốt như thế nào, bạn cần một bộ dữ liệu lớn khác đó là hoàn toàn tách biệt với bất kỳ dữ liệu nào được sử dụng trong quá trình học và điều chỉnh. Điều này cũng đúng để xác thực chéo: bạn cần xác thực chéo “bên trong” để điều chỉnh tham số và xác thực chéo "bên ngoài" để ước tính lỗi. Với xác thực chéo 10 lần, điều này liên quan đến việc chạy sơ đồ học tập 100 lần. Để tóm tắt: khi đánh giá hiệu suất của sơ đồ học tập, bất kỳ điều chỉnh tham số nào điều đó tiếp tục nên được coi như thể nó là một phần không thể thiếu của quá trình đào tạo.

Có những quy trình quan trọng khác có thể cải thiện đáng kể thành công khi áp dụng các kỹ thuật học máy cho các vấn đề khai thác dữ liệu thực tế, và đây là chủ đề của chương này. Chúng tạo thành một loại kỹ thuật dữ liệu: kỹ thuật hóa dữ liệu đầu vào thành một dạng phù hợp với sơ đồ học tập. chọn và thiết kế mô hình đầu ra để làm cho nó hiệu quả hơn. Bạn có thể xem chúng như một túi thủ thuật mà bạn có thể áp dụng cho các vấn đề khai thác dữ liệu thực tế để nâng cao cơ hội thành công. Đôi khi chúng hoạt động; lần khác họ không—và với trình độ kỹ thuật hiện nay, thật khó để nói trước liệu họ sẽ hay không. Trong một lĩnh vực như thế này, nơi thử và sai là hướng dẫn đáng tin cậy nhất, điều đặc biệt quan trọng là phải tháo vát và hiểu những gì các thủ thuật là.

Chúng ta bắt đầu bằng cách kiểm tra bốn cách khác nhau trong đó đầu vào có thể được masage để làm cho nó phù hợp hơn với các phương pháp học tập: lựa chọn thuộc tính, rời rạc hóa thuộc tính, chuyển đổi dữ liệu và làm sạch dữ liệu. xem xét đầu tiên, lựa chọn thuộc tính. Trong nhiều tình huống thực tế có quá nhiều

các thuộc tính cho các kế hoạch học tập để xử lý, và một số trong số chúng—có lẽ chiếm đa số áp đảo—rõ ràng là không liên quan hoặc dư thừa. Do đó, dữ liệu phải được xử lý trước để chọn một tập hợp con các thuộc tính để sử dụng trong học tập. Của nhiên, bản thân các phương pháp học cố gắng lựa chọn các thuộc tính một cách phù hợp và bỏ qua những cái không liên quan hoặc dư thừa, nhưng trong thực tế, hiệu suất của chúng thường có thể được cải thiện bằng cách chọn trước. Ví dụ, thí nghiệm cho thấy rằng việc thêm các thuộc tính vô ích gây ra hiệu suất của các lược đồ học tập, chẳng hạn như cây quyết định và quy tắc, hồi quy tuyến tính, người học dựa trên cá thể và phương pháp phân cụm xuống cấp.

Việc rời rạc hóa các thuộc tính số là hoàn toàn cần thiết nếu nhiệm vụ liên quan đến thuộc tính số nhưng phương pháp học đã chọn chỉ có thể xử lý thuộc tính phân loại cái. Ngay cả các phương pháp có thể xử lý các thuộc tính số thường tạo ra tốt hơn kết quả hoặc hoạt động nhanh hơn nếu các thuộc tính được xác định trước. Tình huống ngược lại, trong đó các thuộc tính phân loại phải được biểu diễn bằng số, cũng xảy ra (mặc dù ít thường xuyên hơn); và chúng tôi cũng mô tả các kỹ thuật cho trường hợp này.

Chuyển đổi dữ liệu bao gồm nhiều kỹ thuật khác nhau. Một chuyển đổi, mà chúng ta đã gặp trước đây khi xem dữ liệu quan hệ trong Chương 2 và máy vectơ hỗ trợ trong Chương 6, là thêm các thuộc tính tổng hợp mới mục đích là trình bày thông tin hiện có dưới dạng phù hợp với kế hoạch học máy để tiếp tục. Các kỹ thuật tổng quát hơn mà làm không phụ thuộc quá mật thiết vào ngữ nghĩa của vấn đề khai thác dữ liệu cụ thể hiện có bao gồm phân tích các thành phần chính và các phép chiếu ngẫu nhiên.

Dữ liệu không rõ ràng gây hại cho việc khai thác dữ liệu. Chúng tôi đã nhấn mạnh trong Chương 2 về sự cần thiết của việc tìm hiểu dữ liệu của bạn: hiểu ý nghĩa của tất cả các thuộc tính khác nhau, các quy ước được sử dụng để mã hóa chúng, tầm quan trọng của việc thiếu sót. các giá trị và dữ liệu trùng lặp, nhiễu đo lường, lỗi đánh máy và sự hiện diện của các lỗi hệ thống, ngay cả những lỗi cố ý. Nhiều hình ảnh đơn giản khác nhau thường giúp ích cho nhiệm vụ này. Ngoài ra còn có các phương pháp làm sạch tự động dữ liệu, phát hiện các ngoại lệ và phát hiện các điểm bất thường mà chúng tôi mô tả.

Sau khi nghiên cứu cách xoa bóp đầu vào, chúng ta chuyển sang câu hỏi kỹ thuật hóa đầu ra từ các sơ đồ học máy. Đặc biệt, chúng tôi kiểm tra các kỹ thuật để kết hợp các mô hình khác nhau đã học được từ dữ liệu. Có một số bất ngờ trong cửa hàng. Ví dụ, thường thuận lợi khi lấy dữ liệu huấn luyện và rút ra một số tập huấn luyện khác nhau từ dữ liệu đó, học một mô hình từ từng cái và kết hợp các mô hình kết quả! Thật vậy, các kỹ thuật để làm điều này có thể trở nên rất mạnh mẽ. Ví dụ, có thể chuyển đổi một phương pháp học tương đối yếu thành một phương pháp cực kỳ mạnh (theo nghĩa chính xác là chúng ta sẽ giải thích). Ngoài ra, nếu có sẵn một số lược đồ học tập, có thể tốt hơn là không nên chọn lược đồ hoạt động tốt nhất cho tập dữ liệu của bạn (sử dụng xác thực chéo) mà nên sử dụng tất cả chúng và kết hợp các kết quả. Cuối cùng, tiêu chuẩn cách rõ ràng để mô hình hóa một tình huống học tập nhiều lớp như một lớp học hai lớp có thể được cải thiện bằng cách sử dụng một kỹ thuật đơn giản nhưng tinh tế.

Nhiều kết quả trong số này là phản trực giác, ít nhất là thoạt nhìn hơi đỏ mặt. Làm thế nào nó có thể là một ý tưởng tốt để sử dụng nhiều mô hình khác nhau với nhau? Làm thế nào bạn có thể làm tốt hơn là chọn mô hình hoạt động tốt nhất? Chắc chắn tất cả điều này chạy phản đến dao cạo râu của Occam, ủng hộ sự đơn giản. Làm thế nào bạn có thể có được hiệu suất hạng nhất bằng cách kết hợp các mô hình khác nhau, như một trong những kỹ thuật này dường như để làm gì? Nhưng hãy xem xét các ủy ban của con người, thường đưa ra quyết định khôn ngoan hơn so với các chuyên gia cá nhân. Nhớ lại quan điểm của Epicurus rằng, đối mặt với giải thích thay thế, người ta nên giữ lại tất cả. Hãy tưởng tượng một nhóm các chuyên gia, mỗi người đều xuất sắc trong một lĩnh vực hạn chế mặc dù không ai giỏi cả. mọi mặt. Khi đấu tranh để hiểu cách thức hoạt động của các phương pháp này, các nhà nghiên cứu đã phơi bày tất cả các loại kết nối và liên kết dẫn đến thậm chí những cải tiến lớn hơn.

Một thực tế phi thường khác là hiệu suất phân loại thường có thể được được cải thiện bằng cách bổ sung một lượng dữ liệu đáng kể không được gắn nhãn, trong nói cách khác, các giá trị của lớp là không xác định. Một lần nữa, điều này dường như bay thẳng vào khuôn mặt của lẽ thường, giống như một dòng sông chảy ngược dòng hoặc vĩnh viễn máy chuyển động. Nhưng nếu nó là sự thật—và đúng như vậy, như chúng tôi sẽ cho bạn thấy trong Phần 7.6—nó sẽ có tầm quan trọng thực tiễn rất lớn vì có nhiều tình huống trong đó dữ liệu được gắn nhãn là khan hiếm nhưng dữ liệu không được gắn nhãn lại dồi dào. Đọc tiếp—và chuẩn bị để được ngạc nhiên.

## 7.1 Lựa chọn thuộc tính

Hầu hết các thuật toán học máy được thiết kế để học cái nào hiệu quả nhất các thuộc tính thích hợp để sử dụng cho việc đưa ra quyết định của họ. Ví dụ, các phương pháp cây quyết định chọn thuộc tính hứa hẹn nhất để phân chia tại mỗi điểm và - theo lý thuyết - không bao giờ chọn các thuộc tính không liên quan hoặc không hữu ích. Việc có nhiều tính năng hơn chắc chắn—về lý thuyết—sẽ dẫn đến khả năng phân biệt đối xử cao hơn chứ không bao giờ kém đi. “Sự khác biệt giữa lý thuyết và thực hành là gì?” một câu hỏi cũ hỏi. "Không có sự khác biệt," câu trả lời đi,
“—về lý thuyết. Nhưng, trong thực tế, có." Ở đây cũng vậy: trong thực tế, thêm các thuộc tính không liên quan hoặc gây mất tập trung cho một tập dữ liệu thường “gây nhầm lẫn” cho các hệ thống máy học.

Các thử nghiệm với bộ học cây quyết định (C4.5) đã chỉ ra rằng việc thêm vào bộ dữ liệu tiêu chuẩn một thuộc tính nhị phân ngẫu nhiên được tạo bằng cách ném một xu ảnh hưởng đến hiệu suất phân loại, khiến nó giảm giá trị (thường là 5% đến 10% trong các tình huống được thử nghiệm). Điều này xảy ra bởi vì tại một thời điểm nào đó trên cây được học, thuộc tính không liên quan luôn được chọn để phân nhánh, gây ra lỗi ngẫu nhiên khi dữ liệu thử nghiệm được xử lý. Làm sao có thể, khi những người học cây quyết định được thiết kế khéo léo để chọn thuộc tính tốt nhất cho việc chia tách tại mỗi nút? Lý do là tinh tế. Khi bạn tiến sâu hơn xuống cây, ít hơn

và có ít dữ liệu hơn để giúp đưa ra quyết định lựa chọn. Tại một số điểm, với ít dữ liệu, thuộc tính ngẫu nhiên sẽ trông đẹp một cách tình cờ. Bởi vì số lượng nút ở mỗi cấp độ tăng theo cấp số nhân với độ sâu, cơ hội thuộc tính lừa đảo có vẻ tốt ở đâu đó dọc theo biên giới nhân lên như cây sâu thêm. Vấn đề thực sự là bạn chắc chắn đạt đến độ sâu mà tại đó chỉ một lượng nhỏ dữ liệu có sẵn để lựa chọn thuộc tính. Nếu tập dữ liệu lớn hơn thì điều đó không nhất thiết hữu ích—có lẽ bạn chỉ cần tìm hiểu sâu hơn.

Người học cây chia để trị và người học quy tắc tách rời cả hai đều chịu ảnh hưởng này vì chúng làm giảm lượng dữ liệu dựa vào đó họ phán xét. Người học dựa trên cá thể rất dễ bị các thuộc tính không liên quan vì chúng luôn hoạt động trong các vùng lân cận địa phương, lấy chỉ một vài trường hợp đào tạo được tính đến cho mỗi quyết định. Thật vậy, nó đã được đã chỉ ra rằng số lượng phiên bản đào tạo cần thiết để tạo ra mức hiệu suất khai thác được xác định trước cho quá trình học tập dựa trên phiên bản tăng theo cấp số nhân với số lượng các thuộc tính không liên quan hiện có. Ngược lại, Naïve Bayes không không phân mảnh không gian thể hiện và mạnh mẽ bỏ qua các thuộc tính không liên quan. Nó giả định theo thiết kế rằng tất cả các thuộc tính đều độc lập với nhau, một giả định phù hợp với các thuộc tính ngẫu nhiên "phân tâm". Nhưng thông qua điều này rất cùng một giả định, Naïve Bayes phải trả giá đắt theo những cách khác vì hoạt động của nó bị tổn hại do thêm các thuộc tính dư thừa.

Thoạt tiên, thực tế là những yếu tố gây phân tâm không liên quan làm giảm hiệu suất của cây quyết định và quy tắc tiên tiến nhất là điều đáng ngạc nhiên. Ngạc nhiên hơn nữa là các thuộc tính có liên quan cũng có thể gây hại. Ví dụ, giả sử rằng trong một tập dữ liệu hai lớp, một thuộc tính mới đã được thêm vào có cùng giá trị với lớp được dự đoán hầu hết thời gian (65%) và giá trị ngược lại phần còn lại của thời gian, được phân phối ngẫu nhiên giữa các trường hợp. Thí nghiệm với tiêu chuẩn bộ dữ liệu đã chỉ ra rằng điều này có thể làm giảm độ chính xác của phân loại (bởi 1% đến 5% trong các tình huống được thử nghiệm). Vấn đề là thuộc tính mới (tự nhiên) được chọn để phân tách cao trong cây. Điều này có tác dụng phân mảnh tập hợp các phiên bản có sẵn tại các nút bên dưới để các lựa chọn khác được dựa trên dữ liệu thưa thớt hơn.

Do tác động tiêu cực của các thuộc tính không liên quan đến hầu hết các sơ đồ học máy, thông thường trước khi học có một giai đoạn lựa chọn thuộc tính. cố gắng loại bỏ tất cả trừ các thuộc tính có liên quan nhất. Cách tốt nhất để chọn các thuộc tính có liên quan theo cách thủ công, dựa trên sự hiểu biết sâu sắc về vấn đề học tập và ý nghĩa thực sự của các thuộc tính. Tuy nhiên, tự động phương pháp cũng có thể hữu ích. Việc giảm kích thước của dữ liệu bằng cách xóa các thuộc tính không phù hợp sẽ cải thiện hiệu suất của các thuật toán học tập. Nó cũng tăng tốc chúng, mặc dù điều này có thể bị ảnh hưởng bởi tính toán tham gia vào việc lựa chọn thuộc tính. Quan trọng hơn, giảm kích thước mang lại một đại diện nhỏ gọn hơn, dễ hiểu hơn về mục tiêu khái niệm, tập trung sự chú ý của người dùng vào các biến có liên quan nhất.

### Lựa chọn độc lập với chương trình

Khi chọn một tập hợp con thuộc tính tốt, có hai khác biệt cơ bản
cách tiếp cận. Một là thực hiện đánh giá độc lập dựa trên các đặc điểm chung của dữ liệu;
cách khác là đánh giá tập hợp con bằng máy
thuật toán học mà cuối cùng sẽ được sử dụng cho việc học. Đầu tiên là
được gọi là phương pháp lọc , bởi vì tập thuộc tính được lọc để tạo ra tập hợp con hứa hẹn
nhất trước khi quá trình học bắt đầu. Thứ hai là trình bao bọc
phương pháp, bởi vì thuật toán học tập được gói gọn trong quá trình lựa chọn. Thực hiện đánh
giá độc lập về một tập hợp con thuộc tính sẽ dễ dàng
nếu có một cách tốt để xác định khi một thuộc tính có liên quan đến
chọn lớp. Tuy nhiên, không có biện pháp đo lường "mối quan hệ" được chấp nhận rộng rãi, mặc
dù một số biện pháp khác nhau đã được đề xuất.

Một phương pháp lựa chọn thuộc tính độc lập với lược đồ đơn giản là chỉ sử dụng
đủ thuộc tính để phân chia không gian phiên bản theo cách phân tách tất cả
trường hợp đào tạo. Ví dụ: nếu chỉ sử dụng một hoặc hai thuộc tính, sẽ có
thường là một số trường hợp có cùng sự kết hợp thuộc tính
các giá trị. Ở một thái cực khác, tập hợp đầy đủ các thuộc tính sẽ có khả năng phân biệt
thể hiện duy nhất sao cho không có hai thể hiện nào có cùng giá trị cho tất cả các thuộc
tính. (Tuy nhiên, điều này không nhất thiết phải đúng; bộ dữ liệu đôi khi chứa
trường hợp có cùng giá trị thuộc tính nhưng khác lớp.) Nó làm cho trực quan
ý nghĩa để chọn tập hợp con thuộc tính nhỏ nhất phân biệt tất cả các trường hợp
độc nhất. Điều này có thể dễ dàng được tìm thấy bằng cách sử dụng tìm kiếm toàn diện, mặc dù
với chi phí tính toán đáng kể. Thật không may, sự thiên vị mạnh mẽ này đối với xu hướng nhất
quán của thuộc tính được đặt trên dữ liệu huấn luyện là không có cơ sở về mặt thống kê và
có thể dẫn đến trang bị thừa—thuật toán có thể mất nhiều thời gian không cần thiết để sửa chữa
một sự không nhất quán mà trên thực tế chỉ đơn thuần là do tiếng ồn gây ra.

Các thuật toán học máy có thể được sử dụng để lựa chọn thuộc tính. Ví dụ,
trước tiên bạn có thể áp dụng thuật toán cây quyết định cho tập dữ liệu đầy đủ, sau đó chọn
chỉ những thuộc tính thực sự được sử dụng trong cây. Mặc dù sự lựa chọn này
sẽ không có tác dụng gì nếu giai đoạn thứ hai chỉ xây dựng một cây khác, nó sẽ
có ảnh hưởng đến một thuật toán học tập khác. Ví dụ, thuật toán hàng xóm gần nhất nổi tiếng
là dễ bị ảnh hưởng bởi các thuộc tính không liên quan và
hiệu suất có thể được cải thiện bằng cách sử dụng trình tạo cây quyết định làm bộ lọc cho
lựa chọn thuộc tính đầu tiên. Phương pháp láng giềng gần nhất kết quả cũng có thể
hoạt động tốt hơn thuật toán cây quyết định được sử dụng để lọc. như một người khác
dụ, sơ đồ 1R đơn giản được mô tả trong Chương 4 đã được sử dụng để chọn
các thuộc tính cho một người học cây quyết định bằng cách đánh giá hiệu quả của việc phân nhánh
trên các thuộc tính khác nhau (mặc dù phương pháp dựa trên lỗi như 1R có thể không
lựa chọn tối ưu cho các thuộc tính xếp hạng, như chúng ta sẽ thấy sau này khi đề cập đến
vấn đề liên quan đến rời rạc hóa có giám sát). Thông thường cây quyết định thực hiện
cũng như khi chỉ có hai hoặc ba thuộc tính hàng đầu được sử dụng để xây dựng nó

và nó dễ hiểu hơn nhiều. Trong cách tiếp cận này, người dùng xác định có bao nhiêu thuộc tính được sử dụng để xây dựng cây quyết định.

Một khả năng khác là sử dụng thuật toán xây dựng mô hình tuyến tính—cho ví dụ, một máy vectơ hỗ trợ tuyến tính—và xếp hạng các thuộc tính dựa trên kích thước của các hệ số. Một biến thể phức tạp hơn áp dụng việc học thuật toán lặp đi lặp lại. Nó xây dựng một mô hình, xếp hạng các thuộc tính dựa trên các hệ số, loại bỏ thuộc tính được xếp hạng cao nhất và lặp lại quy trình cho đến khi tất cả các thuộc tính được loại bỏ. Phương pháp loại bỏ tính năng đệ quy này đã được phát hiện là mang lại kết quả tốt hơn trên một số bộ dữ liệu nhất định (ví dụ: khi xác định các gen quan trọng để phân loại ung thư) hơn là chỉ xếp hạng các thuộc tính dựa trên một mô hình duy nhất. Với cả hai phương pháp, điều quan trọng là phải đảm bảo rằng các thuộc tính được đo lường trên cùng một thang đo; mặt khác, các hệ số không thể so sánh được. Lưu ý rằng những kỹ thuật này chỉ tạo ra một bảng xếp hạng; một phương pháp khác phải được sử dụng để xác định số lượng thuộc tính thích hợp để sử dụng.

Các thuộc tính cũng có thể được chọn bằng các phương pháp học tập dựa trên cá thể. Bạn có thể lấy mẫu các trường hợp ngẫu nhiên từ tập huấn luyện và kiểm tra các trường hợp lân cận các bản ghi của các loại giống nhau và khác nhau—"suýt trúng" và "suýt trượt." Nếu một cú đánh gần có một giá trị khác đối với một thuộc tính nhất định, thuộc tính đó dường như là không liên quan và trọng lượng của nó nên được giảm xuống. Mặt khác, nếu suýt chút nữa có giá trị khác, thì thuộc tính có vẻ phù hợp và trọng số của nó sẽ được tăng lên. Tất nhiên, đây là loại thủ tục tiêu chuẩn được sử dụng để tính trọng số thuộc tính cho học tập dựa trên cá thể, được mô tả trong Phần 6.4. Sau khi lặp lại thao tác này nhiều lần, quá trình lựa chọn diễn ra: chỉ các thuộc tính có giá trị dương trọng số được chọn. Như trong công thức gia tăng tiêu chuẩn của học tập dựa trên cá thể, các kết quả khác nhau sẽ thu được mỗi khi quá trình này được thực hiện. lặp đi lặp lại, do thứ tự khác nhau của các ví dụ. Điều này có thể tránh được bằng cách sử dụng tất cả các trường hợp huấn luyện và tính đến tất cả các lần suýt trúng và suýt trúng của mỗi.

Một nhược điểm nghiêm trọng hơn là phương pháp sẽ không phát hiện thuộc tính điều đó là dư thừa vì nó có tương quan với một thuộc tính khác. trong cùng cực trường hợp, hai thuộc tính giống hệt nhau sẽ được xử lý theo cùng một cách, hoặc cả hai được chọn hoặc cả hai đều bị từ chối. Một sửa đổi đã được đề xuất dường như đi một số cách để giải quyết vấn đề này bằng cách lấy trọng số thuộc tính hiện tại tính đến khi tính toán các lần truy cập và bỏ lỡ gần nhất.

Một cách khác để loại bỏ các thuộc tính dư thừa cũng như không liên quan là để chọn một tập hợp con các thuộc tính tương quan tốt với lớp nhưng có ít tương quan với nhau. Mối tương quan giữa hai thuộc tính danh nghĩa A và B có thể được đo bằng độ không đảm bảo đối xứng:

$$\text{bạn}(A,B) = 2\,\frac{H(A)+H(B)-H(A,B)}{H(A)+H(B)},$$

trong đó H là hàm entropy được mô tả trong Phần 4.3. Các entropi là
dựa trên xác suất liên quan đến từng giá trị thuộc tính; H(A,B), khớp
entropy của A và B, được tính toán từ các xác suất chung của tất cả các tổ hợp giá trị của
A và B. Độ không đảm bảo đối xứng luôn nằm trong khoảng từ 0 đến 1. Việc lựa chọn đặc trưng
dựa trên tương quan xác định mức độ tốt của một tập hợp các
thuộc tính sử dụng

$$, \hat{A}UACj \hat{A}\hat{A} ( ,\Big/\sqrt{\quad UA\ A\ -\ \ j} \quad -\ \ j ,)$$

trong đó C là thuộc tính lớp và các chỉ số i và j nằm trên tất cả các thuộc tính trong
bộ. Nếu tất cả m thuộc tính trong tập con tương quan hoàn hảo với lớp và với
nhau, tử số trở thành m và mẫu số trở thành m. Do đó, số đo là 1, hóa ra là giá trị $\sqrt{m2}$ ,
lớn nhất

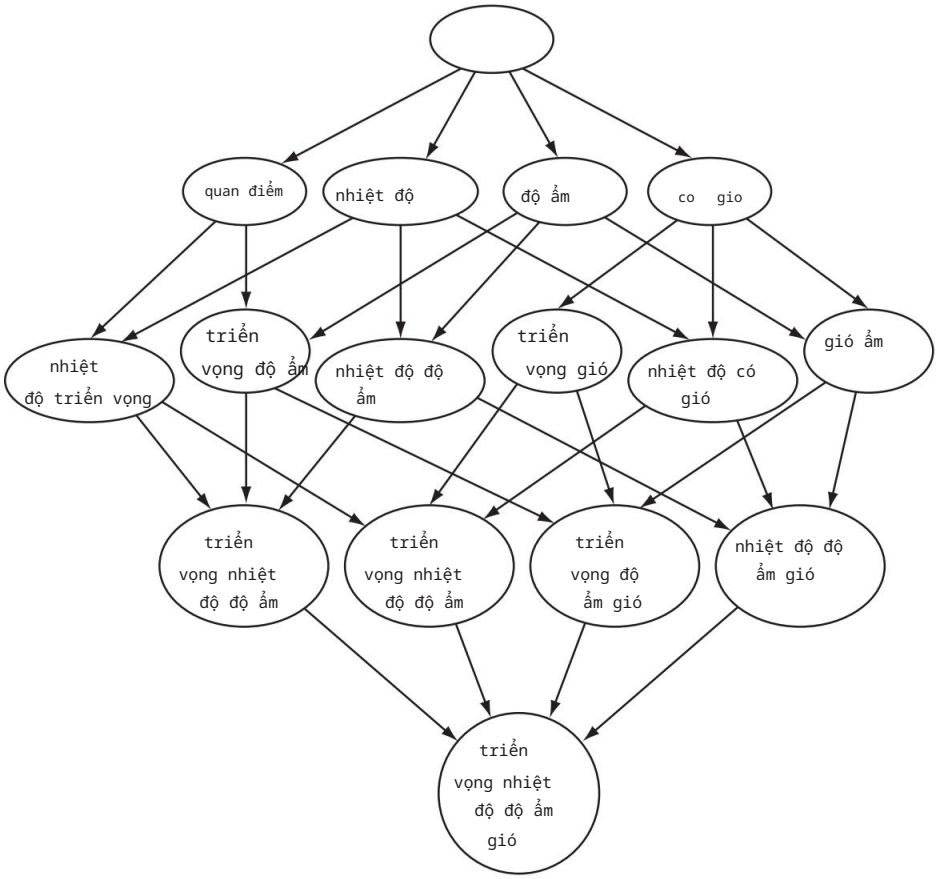
giá trị nó có thể đạt được (tối thiểu là 0). Rõ ràng đây không phải là lý tưởng, bởi vì chúng tôi muốn
để tránh các thuộc tính dư thừa. Tuy nhiên, bất kỳ tập hợp con nào của tập hợp này cũng sẽ có giá trị
1. Khi sử dụng tiêu chí này để tìm kiếm một tập con tốt của các thuộc tính, nó tạo ra
ý nghĩa để phá vỡ các ràng buộc có lợi cho tập hợp con nhỏ nhất.

## Tìm kiếm không gian thuộc tính

Hầu hết các phương pháp lựa chọn thuộc tính liên quan đến việc tìm kiếm không gian của các thuộc tính
cho tập hợp con có nhiều khả năng dự đoán lớp tốt nhất. Hình 7.1 minh họa
không gian thuộc tính cho bộ dữ liệu thời tiết—đến nay đã quá quen thuộc—. Các
số tập hợp con thuộc tính có thể tăng theo cấp số nhân với số lượng
của các thuộc tính, làm cho việc tìm kiếm toàn diện trở nên không thực tế đối với tất cả các vấn đề trừ những vấn
đề đơn giản nhất.

Thông thường, không gian được tìm kiếm một cách tham lam theo một trong hai hướng, từ trên xuống dưới.
dưới hoặc từ dưới lên trên trong hình. Ở mỗi giai đoạn, một thay đổi cục bộ được thực hiện để
tập hợp con thuộc tính hiện tại bằng cách thêm hoặc xóa một thuộc tính. Các
hướng đi xuống, nơi bạn bắt đầu không có thuộc tính nào và thêm từng thuộc tính
thời gian, được gọi là lựa chọn chuyển tiếp. Hướng lên, nơi bạn bắt đầu với đầy đủ
đặt và xóa từng thuộc tính một, là loại bỏ ngược.

Trong lựa chọn chuyển tiếp, mỗi thuộc tính chưa có trong tập hợp con hiện tại
được tạm thời thêm vào nó và tập hợp các thuộc tính kết quả được đánh giá bằng cách sử dụng,
ví dụ: xác thực chéo như được mô tả trong phần sau. Việc đánh giá này tạo ra thước đo bằng
số về hiệu suất dự kiến của tập hợp con.
Hiệu quả của việc thêm từng thuộc tính lần lượt được định lượng bằng biện pháp này, tốt nhất
một được chọn, và thủ tục tiếp tục. Tuy nhiên, nếu không có thuộc tính nào sinh ra
một cải tiến khi được thêm vào tập hợp con hiện tại, tìm kiếm kết thúc. Đây là một
thủ tục tìm kiếm tham lam tiêu chuẩn và đảm bảo tìm thấy một tập hợp các thuộc tính tối ưu
cục bộ—chứ không nhất thiết là toàn cục. Loại bỏ ngược hoạt động trong
một thời trang hoàn toàn tương tự. Trong cả hai trường hợp, một sự thiên vị nhẹ thường được đưa ra




Hình 7.1 Không gian thuộc tính cho bộ dữ liệu thời tiết.

hướng tới các tập thuộc tính nhỏ hơn. Điều này có thể được thực hiện đối với lựa chọn chuyển tiếp bằng cách nhấn mạnh rằng nếu tiếp tục tìm kiếm, thước đo đánh giá không chỉ phải tăng mà còn phải tăng ít nhất một lượng nhỏ đã xác định trước. Một sửa đổi tương tự hoạt động để loại bỏ lạc hậu.

Có nhiều phương pháp tìm kiếm phức tạp hơn. Lựa chọn chuyển tiếp và loại bỏ ngược có thể được kết hợp thành một tìm kiếm hai chiều; một lần nữa, người ta có thể bắt đầu với tất cả các thuộc tính hoặc không có thuộc tính nào. Tìm kiếm ưu tiên tốt nhất là phương pháp không chỉ chấm dứt khi hiệu suất bắt đầu giảm mà còn giữ danh sách tất cả các tập hợp con thuộc tính được đánh giá cho đến nay, được sắp xếp theo thứ tự thước đo hiệu suất, để thay vào đó, nó có thể truy cập lại cấu hình trước đó. Đủ thời gian nhất định, nó sẽ khám phá toàn bộ không gian, trừ khi điều này bị ngăn cản bởi một số loại tiêu chí dừng. Tìm kiếm chùm tương tự nhưng cắt bớt danh sách các tập hợp con thuộc tính của nó ở mỗi giai đoạn để nó chỉ chứa một số cố định—độ rộng chùm—

trong số những ứng cử viên triển vọng nhất. Các thủ tục tìm kiếm thuật toán di truyền dựa trên nguyên tắc chọn lọc tự nhiên một cách lỏng lẻo: chúng "tiến hóa" các tập hợp con tính năng tốt bằng cách sử dụng các nhiễu loạn ngẫu nhiên của danh sách các tập hợp con ứng cử viên hiện tại.

Lựa chọn theo lược đồ cụ thể Hiệu

suất của một tập hợp con thuộc tính với lựa chọn theo lược đồ cụ thể được đo lường theo hiệu suất phân loại của lược đồ học chỉ sử dụng các thuộc tính đó. Với một tập hợp con các thuộc tính, độ chính xác được ước tính bằng cách sử dụng quy trình xác thực chéo thông thường được mô tả trong Phần 5.3. Tất nhiên, các phương pháp đánh giá khác như hiệu suất trên một tập hợp nắm giữ (Phần 5.3) hoặc công cụ ước tính bootstrap (Phần 5.4) cũng có thể được sử dụng tương tự.

Toàn bộ quá trình lựa chọn thuộc tính là tính toán chuyên sâu. Nếu mỗi đánh giá liên quan đến xác thực chéo 10 lần, quy trình học tập phải được thực hiện 10 lần. Với k thuộc tính, lựa chọn chuyển tiếp theo kinh nghiệm hoặc loại bỏ ngược sẽ nhân thời gian đánh giá lên tới k2 —và đối với các tìm kiếm phức tạp hơn, hình phạt sẽ lớn hơn nhiều, lên tới 2k đối với thuật toán toàn diện kiểm tra từng tập con trong số 2k tập hợp con có thể .

Kết quả tốt đã được chứng minh trên nhiều bộ dữ liệu. Nói chung, loại bỏ ngược tạo ra các tập thuộc tính lớn hơn và độ chính xác phân loại tốt hơn so với lựa chọn chuyển tiếp. Lý do là thước đo hiệu suất chỉ là ước tính và một ước tính lạc quan duy nhất sẽ khiến cả hai quy trình tìm kiếm này dừng sớm—loại bỏ ngược với quá nhiều thuộc tính và lựa chọn chuyển tiếp với không đủ. Nhưng lựa chọn chuyển tiếp sẽ hữu ích nếu trọng tâm là tìm hiểu các cấu trúc quyết định có liên quan, bởi vì nó thường làm giảm số lượng thuộc tính chỉ ảnh hưởng rất nhỏ đến độ chính xác của phân loại. Kinh nghiệm dường như cho thấy rằng các kỹ thuật tìm kiếm tinh vi hơn nói chung là không hợp lý—mặc dù chúng có thể tạo ra kết quả tốt hơn nhiều trong một số trường hợp nhất định.

Một cách để tăng tốc quá trình tìm kiếm là ngừng đánh giá một tập hợp con các thuộc tính ngay khi thấy rõ rằng nó không có khả năng dẫn đến độ chính xác cao hơn một tập hợp con ứng cử viên khác. Đây là công việc đối với kiểm tra ý nghĩa thống kê theo cặp, được thực hiện giữa bộ phân loại dựa trên tập hợp con này và tất cả các bộ phân loại ứng cử viên khác dựa trên các tập hợp con khác. Sự khác biệt về hiệu suất giữa hai bộ phân loại trong một phiên bản thử nghiệm cụ thể có thể được coi là -1, 0 hoặc 1 tùy thuộc vào việc bộ phân loại đầu tiên kém hơn, bằng hoặc tốt hơn bộ phân loại thứ hai trên phiên bản đó. Có thể áp dụng thử nghiệm t theo cặp (được mô tả trong Phần 5.5) cho các số liệu này trên toàn bộ tập hợp thử nghiệm, xử lý hiệu quả các kết quả cho từng trường hợp như một ước tính độc lập về sự khác biệt trong hiệu suất . Sau đó, việc xác thực chéo cho một bộ phân loại có thể bị chấm dứt sớm ngay khi nó trở nên tồi tệ hơn đáng kể so với một bộ phân loại khác—điều này, tất nhiên, có thể không bao giờ xảy ra. Chúng tôi có thể muốn loại bỏ các bộ phân loại mạnh mẽ hơn bằng cách sửa đổi

kiểm tra t để tính xác suất mà một bộ phân loại này tốt hơn một bộ phân loại khác ít nhất bằng một ngưỡng nhỏ do người dùng chỉ định. Nếu xác suất này trở nên rất nhỏ, chúng ta có thể loại bỏ bộ phân loại trước đó trên cơ sở rằng nó không có khả năng hoạt động tốt hơn đáng kể so với bộ phân loại sau.

Phương pháp này được gọi là tìm kiếm theo chủng tộc và có thể được thực hiện với các chiến lược tìm kiếm cơ bản khác nhau. Khi được sử dụng với lựa chọn chuyển tiếp, chúng tôi chạy đua đồng thời với tất cả các bổ sung thuộc tính đơn lẻ có thể có và loại bỏ những bổ sung không hoạt động đủ tốt. Trong loại bỏ ngược, chúng tôi chạy đua với tất cả các lần xóa một thuộc tính. Tìm kiếm lược đồ là một phương pháp phức tạp hơn được thiết kế dành riêng cho đua xe; nó chạy một loạt các cuộc đua lặp đi lặp lại mà mỗi cuộc đua xác định xem có nên đưa vào một thuộc tính cụ thể hay không. Các thuộc tính khác cho chủng tộc này được bao gồm hoặc loại trừ ngẫu nhiên tại mỗi thời điểm trong quá trình đánh giá. Ngay sau khi một cuộc đua có người chiến thắng rõ ràng, vòng lặp tiếp theo của cuộc đua sẽ bắt đầu, lấy người chiến thắng làm điểm xuất phát. Một chiến lược tìm kiếm khác là xếp hạng các thuộc tính trước, ví dụ như sử dụng thông tin thu được của chúng (giả sử chúng rời rạc), sau đó chạy đua xếp hạng. Trong trường hợp này, cuộc đua không bao gồm các thuộc tính, thuộc tính được xếp hạng cao nhất, hai thuộc tính hàng đầu, ba thuộc tính hàng đầu, v.v.

Dù bạn làm theo cách nào, việc lựa chọn thuộc tính dành riêng cho sơ đồ không có nghĩa là mang lại sự cải thiện đồng nhất về hiệu suất. Do tính phức tạp của quy trình, vốn tăng lên rất nhiều do tác động phản hồi của việc đưa thuật toán máy học mục tiêu vào vòng lựa chọn phân bổ, nên khá khó để dự đoán các điều kiện mà theo đó nó sẽ trở nên đáng giá. Như trong nhiều tình huống học máy, thử và sai khi sử dụng nguồn dữ liệu cụ thể của riêng bạn là người phân xử cuối cùng.

Có một loại bộ phân loại mà việc lựa chọn thuộc tính dành riêng cho lược đồ là một phần thiết yếu của quá trình học tập: bảng quyết định. Như đã đề cập trong Phần 3.1, toàn bộ vấn đề của việc học các bảng quyết định bao gồm việc lựa chọn các thuộc tính phù hợp để đưa vào. Thông thường, điều này được thực hiện bằng cách đo hiệu suất xác thực chéo của bảng đối với các tập con thuộc tính khác nhau và chọn tập con hoạt động tốt nhất. May mắn thay, xác thực chéo loại bỏ một lần là rất rẻ đối với loại phân loại này. Lấy lỗi xác thực chéo từ bảng quyết định bắt nguồn từ dữ liệu huấn luyện chỉ là vấn đề thao túng số lượng lớp được liên kết với từng mục của bảng, bởi vì cấu trúc của bảng không thay đổi khi các phiên bản được thêm hoặc xóa. Không gian thuộc tính thường được tìm kiếm theo phương pháp tìm kiếm tốt nhất đầu tiên vì chiến lược này ít có khả năng bị kẹt ở giá trị cực đại cục bộ hơn các chiến lược khác, chẳng hạn như lựa chọn chuyển tiếp.

Hãy kết thúc cuộc thảo luận của chúng ta với một câu chuyện thành công. Một phương pháp học mà cách tiếp cận lựa chọn thuộc tính cụ thể theo sơ đồ đơn giản đã cho thấy kết quả tốt là Naïve Bayes. Mặc dù phương pháp này xử lý tốt các thuộc tính ngẫu nhiên, nhưng nó có khả năng bị đánh lừa khi có sự phụ thuộc giữa các thuộc tính và đặc biệt khi các thuộc tính dư thừa được thêm vào. Tuy nhiên, kết quả tốt đã được báo cáo bằng cách sử dụng thuật toán lựa chọn chuyển tiếp—thuật toán này có khả năng phát hiện tốt hơn

khi một thuộc tính dư thừa sắp được thêm vào so với cách tiếp cận loại bỏ ngược—kết hợp với một phép đo rất đơn giản, gần như "ngây thơ"
xác định chất lượng của một tập hợp con thuộc tính chỉ đơn giản là hiệu suất của thuật toán đã học trên tập huấn luyện . Như đã được nhấn mạnh trong Chương 5, hiệu suất của tập huấn luyện chắc chắn không phải là một chỉ báo đáng tin cậy về hiệu suất của tập kiểm tra. Tuy nhiên, các thí nghiệm cho thấy rằng sửa đổi đơn giản này đối với Naïve Bayes cải thiện rõ rệt hiệu suất của nó trên các bộ dữ liệu tiêu chuẩn mà nó thực hiện không hoạt động tốt như các bộ phân loại dựa trên cây hoặc quy tắc và không có bất kỳ tiêu cực nào ảnh hưởng đến kết quả trên các bộ dữ liệu mà Naïve Bayes đã làm tốt. chọn lọc Naïve Bayes, tên gọi của phương pháp học tập này, là một kỹ thuật máy học khả thi, hoạt động tốt và đáng tin cậy trong thực tế.

## 7.2 Phân biệt các thuộc tính số

Một số thuật toán phân loại và phân cụm chỉ xử lý các thuộc tính danh nghĩa và không thể xử lý những cái được đo trên thang số. Để sử dụng chúng trên chung bộ dữ liệu, các thuộc tính số trước tiên phải được "rời rạc hóa" thành một số lượng nhỏ các dãy riêng biệt. Ngay cả các thuật toán học xử lý các thuộc tính số đôi khi xử lý chúng theo những cách không hoàn toàn thỏa đáng. thống kê các phương pháp phân cụm thường giả định rằng các thuộc tính số có phân phối chuẩn—thường không phải là một giả định hợp lý lắm trong thực tế—và tiêu chuẩn phần mở rộng của bộ phân loại Naïve Bayes để xử lý các thuộc tính số thông qua cùng một giả thiết. Mặc dù hầu hết những người học cây quyết định và luật quyết định đều có thể xử lý các thuộc tính số, một số triển khai hoạt động chậm hơn nhiều khi các thuộc tính số xuất hiện vì chúng liên tục sắp xếp thuộc tính các giá trị. Vì tất cả những lý do này, câu hỏi được đặt ra: đâu là cách tốt để rời rạc hóa các thuộc tính số thành các phạm vi trước khi bất kỳ quá trình học nào diễn ra?

Chúng ta đã gặp một số phương pháp để phân biệt các thuộc tính số. Sơ đồ học tập 1R được mô tả trong Chương 4 sử dụng một cách đơn giản nhưng hiệu quả.
kỹ thuật: sắp xếp các thể hiện theo giá trị của thuộc tính và gán giá trị vào phạm vi tại các điểm mà giá trị lớp thay đổi—ngoại trừ mức tối thiểu nhất định số lượng trường hợp trong lớp đa số (sáu) phải nằm trong mỗi phạm vi, có nghĩa là bất kỳ phạm vi nhất định nào cũng có thể bao gồm hỗn hợp các giá trị lớp. Cái này là một phương pháp rời rạc "toàn cầu" được áp dụng cho tất cả các thuộc tính liên tục trước khi bắt đầu học.

Mặt khác, những người học cây quyết định xử lý các thuộc tính số trên một cơ sở cục bộ, kiểm tra các thuộc tính tại mỗi nút của cây khi nó đang được xây dựng để xem liệu chúng có đáng để phân nhánh hay không—và chỉ tại thời điểm đó quyết định nơi tốt nhất để phân chia các thuộc tính liên tục. Mặc dù phương pháp xây dựng cây mà chúng ta đã xem xét trong Chương 6 chỉ xem xét các phân chia nhị phân của các thuộc tính liên tục, nhưng người ta có thể tưởng tượng một sự rời rạc hóa hoàn toàn diễn ra tại đó.

điểm, mang lại sự phân chia nhiều chiều trên một thuộc tính số. Những ưu và nhược điểm của cách tiếp cận địa phương so với toàn cầu là rõ ràng. Sự rời rạc hóa cục bộ được điều chỉnh cho phù hợp với bối cảnh thực tế được cung cấp bởi mỗi nút cây và sẽ tạo ra các sự phân chia khác nhau của cùng một thuộc tính tại các vị trí khác nhau trong cây nếu điều đó có vẻ phù hợp. Tuy nhiên, các quyết định của nó dựa trên ít dữ liệu hơn khi độ sâu của cây tăng lên, điều này ảnh hưởng đến độ tin cậy của chúng. Nếu cây được phát triển hoàn toàn thành lá đơn lẻ trước khi được cắt tỉa trở lại, như với kỹ thuật cắt tỉa ngược thông thường, thì rõ ràng là nhiều quyết định rời rạc sẽ dựa trên dữ liệu hoàn toàn không đầy đủ.

Khi sử dụng rời rạc hóa toàn cục trước khi áp dụng một phương pháp học, có hai cách khả thi để trình bày dữ liệu rời rạc cho người học. Rõ ràng nhất là coi các thuộc tính rời rạc giống như các thuộc tính danh nghĩa: mỗi khoảng rời rạc được biểu thị bằng một giá trị của thuộc tính danh nghĩa. Tuy nhiên, vì một thuộc tính rời rạc được lấy từ một số, nên các giá trị của nó được sắp xếp theo thứ tự và việc coi nó là danh nghĩa sẽ loại bỏ thông tin sắp xếp có giá trị tiềm tàng này.

Tất nhiên, nếu một lược đồ học tập có thể xử lý trực tiếp các thuộc tính có thứ tự, thì giải pháp là hiển nhiên: mỗi thuộc tính rời rạc được khai báo là thuộc loại “có thứ tự”.

Nếu phương pháp học không thể xử lý các thuộc tính có thứ tự, thì vẫn có một cách đơn giản để cho phép nó khai thác thông tin có thứ tự: chuyển đổi từng thuộc tính được xác định thành một tập hợp các thuộc tính nhị phân trước khi áp dụng lược đồ học. Giả sử thuộc tính rời rạc có k giá trị, nó được chuyển đổi thành k - 1 **thuộc tính nhị phân**, i - 1 đầu tiên trong số đó được đặt thành false bất cứ khi nào giá trị thứ i của thuộc tính rời rạc xuất hiện trong dữ liệu và thành true nếu không.

Các thuộc tính còn lại được đặt thành false. Nói cách khác, thuộc **tính nhị phân thứ** (i - 1) thể hiện liệu thuộc tính rời rạc có nhỏ hơn i hay không. Nếu một bộ học cây quyết định phân tách trên thuộc tính này, thì nó hoàn toàn sử dụng thông tin đặt hàng mà nó mã hóa. Lưu ý rằng phép biến đổi này không phụ thuộc vào phương pháp xác định đĩa cụ thể đang được áp dụng: nó chỉ đơn giản là một cách mã hóa một thuộc tính có thứ tự bằng cách sử dụng một tập hợp các thuộc tính nhị phân.

Rời rạc hóa không giám sát Có hai

cách tiếp cận cơ bản đối với vấn đề rời rạc hóa. Một là định lượng từng thuộc tính trong trường hợp không có bất kỳ kiến thức nào về các lớp của các thể hiện trong tập huấn luyện—cái gọi là **sự rời rạc hóa** không giám sát . Cách khác là tính đến các lớp khi rời rạc hóa— sự rời rạc hóa có giám sát. Khả năng đầu tiên là khả năng duy nhất khi xử lý các vấn đề phân cụm trong đó các lớp chưa biết hoặc không tồn tại.

Cách rõ ràng để phân biệt một thuộc tính số là chia phạm vi của nó thành một số khoảng bằng nhau được xác định trước: một thước đo cố định, độc lập với dữ liệu.

Điều này thường được thực hiện tại thời điểm dữ liệu được thu thập. Tuy nhiên, giống như bất kỳ phương pháp rời rạc hóa không được kiểm duyệt nào, nó có nguy cơ phá hủy sự khác biệt vốn có.

hóa ra lại hữu ích trong quá trình học tập bằng cách sử dụng các chuyển màu quá thô hoặc bằng các lựa chọn ranh giới không may mà gộp nhiều trường hợp của các lớp khác nhau lại với nhau một cách không cần thiết.

Việc tạo thùng theo khoảng thời gian bằng nhau thường phân phối các phiên bản rất không đồng đều: một số thùng chứa nhiều phiên bản và những thùng khác không chứa phiên bản nào. Điều này có thể làm suy giảm nghiêm trọng khả năng của thuộc tính giúp xây dựng các cấu trúc quyết định tốt. Thường thì tốt hơn là cho phép các khoảng có kích thước khác nhau, chọn chúng sao cho cùng một số ví dụ đào tạo rơi vào mỗi khoảng. Phương pháp này, chia tần số bằng nhau, chia phạm vi của thuộc tính thành một số lượng ngăn được xác định trước dựa trên sự phân bố các ví dụ dọc theo trục đó—đôi khi được gọi là cân bằng biểu đồ, bởi vì nếu bạn lấy một biểu đồ về nội dung của các ngăn kết quả thì nó sẽ hoàn toàn phẳng. Nếu bạn xem số lượng thùng như một tài nguyên, thì phương pháp này sẽ tận dụng nó một cách tốt nhất.

Tuy nhiên, việc tạo thùng theo tần số bằng nhau vẫn không được biết đến đối với các lớp của cá thể và điều này có thể gây ra các ranh giới xấu. Ví dụ: nếu tất cả các phiên bản trong ngăn có một lớp và tất cả các phiên bản trong ngăn cao hơn tiếp theo có một phiên bản khác ngoại trừ phiên bản đầu tiên có lớp ban đầu, chắc chắn sẽ hợp lý khi tôn trọng sự phân chia lớp và đưa phiên bản đầu tiên đó vào thùng trước đó, hy sinh thuộc tính tần số bằng nhau vì mục đích đồng nhất. Sự rời rạc có giám sát - tính đến các lớp trong quá trình - chắc chắn có lợi thế.
Tuy nhiên, người ta đã phát hiện ra rằng việc tạo thùng tần số bằng nhau có thể mang lại kết quả tuyệt vời, ít nhất là kết hợp với sơ đồ học Naïve Bayes, khi số lượng thùng được chọn theo kiểu phụ thuộc vào dữ liệu bằng cách đặt nó thành căn bậc hai của số của các trường hợp. Phương pháp này được gọi là rời rạc hóa khoảng k tỷ lệ.

### Sự rời rạc hóa dựa trên entropy

Bởi vì tiêu chí được sử dụng để phân tách một thuộc tính số trong quá trình hình thành cây quyết định hoạt động tốt trong thực tế, nên có vẻ như một ý tưởng hay là mở rộng nó thành sự rời rạc tổng quát hơn bằng cách phân tách đệ quy các khoảng cho đến khi đến lúc dừng lại. Trong Chương 6, chúng ta đã thấy cách sắp xếp các thể hiện theo giá trị của thuộc tính và xem xét, đối với mỗi điểm phân tách có thể, thông tin thu được từ kết quả phân tách. Để rời rạc hóa thuộc tính, khi lần phân tách đầu tiên được xác định, quá trình phân tách có thể được lặp lại ở phần trên và phần dưới của phạm vi, v.v., theo cách đệ quy.

Để thấy điều này hoạt động trong thực tế, chúng ta xem lại ví dụ ở trang 189 để biết cretizing thuộc tính nhiệt độ của dữ liệu thời tiết, có giá trị là

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

có không có có có không không có

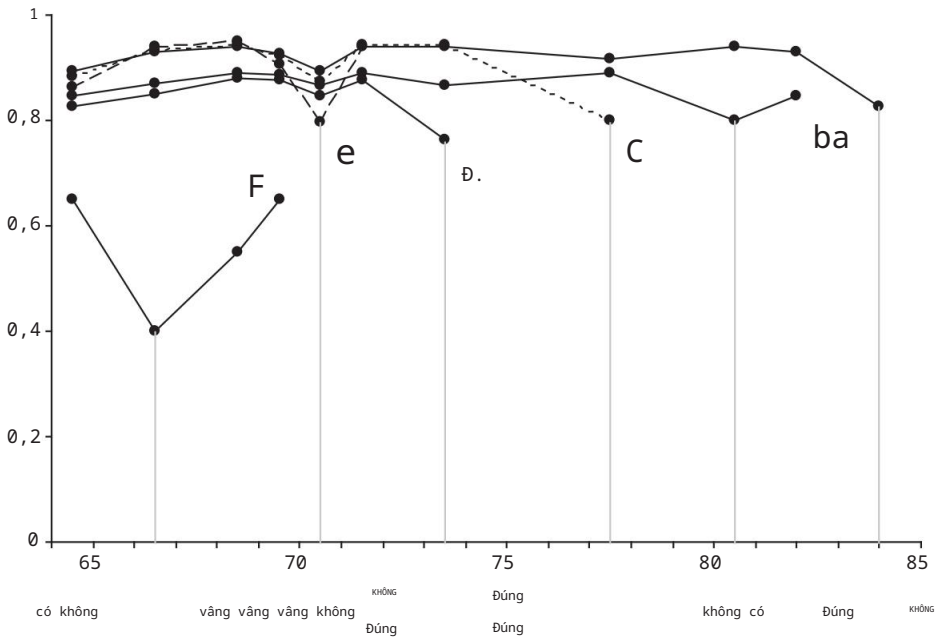
Vâng vâng không có có không

(Các giá trị lặp lại đã được thu gọn cùng nhau.) Thông tin thu được cho mỗi
trong số 11 vị trí có thể cho điểm ngắt được tính theo cách thông thường.
Ví dụ: giá trị thông tin của nhiệt độ thử nghiệm < 71,5, phân tách
phạm vi thành bốn có và hai không so với năm có và ba không, là

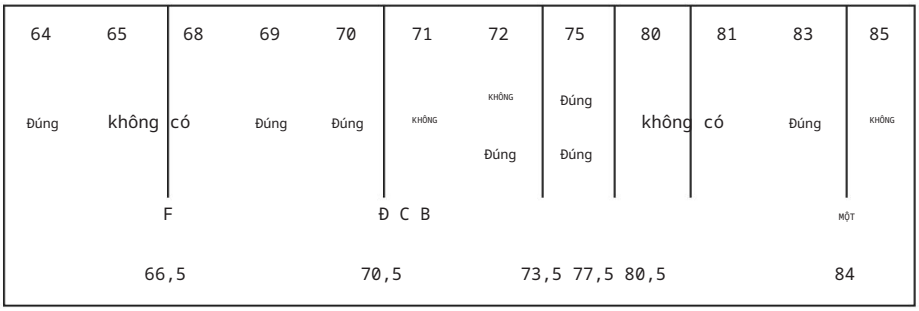
( thông 4,tin2 [ ][ ] , 5 3 6 14 8 14 0,939 ) + ( [ thông 2,tin 3 ] ) ¥ ( [ thông/tin 5, 3 ) ] = . bit

Điều này thể hiện lượng thông tin cần thiết để xác định cá nhân
giá trị của có và không được phân chia. Chúng tôi tìm kiếm một sự rời rạc làm cho
khoảng con càng tinh khiết càng tốt; do đó, chúng tôi chọn tách tại điểm mà
giá trị thông tin là nhỏ nhất. (Điều này cũng giống như việc chia nhỏ khi thông tin thu
được, được định nghĩa là sự khác biệt giữa giá trị thông tin mà không có giá trị
split và cái cùng với split là lớn nhất.) Như trước đây, chúng tôi đặt các ngưỡng số
nằm giữa các giá trị phân định ranh giới của một khái niệm.

Biểu đồ có nhãn A trong Hình 7.2 hiển thị các giá trị thông tin tại mỗi điểm cắt có
thể có ở giai đoạn đầu tiên này. Bộ phận sạch nhất — thông tin nhỏ nhất
giá trị—ở nhiệt độ 84 (0,827 bit), chỉ tách biệt phần rất
giá trị cuối cùng, không có trường hợp nào, từ danh sách trước đó. Các lớp thể hiện được viết
bên dưới trục hoành để dễ hiểu hơn. Gọi thuật toán
một lần nữa trên phạm vi nhiệt độ thấp hơn, từ 64 đến 83, mang lại biểu đồ có nhãn
B. Giá trị này có giá trị tối thiểu là 80,5 (0,800 bit), phân tách hai giá trị tiếp theo,




Hình 7.2 Phân biệt thuộc tính nhiệt độ bằng phương pháp entropy.




Hình 7.3 Kết quả của việc rời rạc hóa thuộc tính nhiệt độ .

cả hai trường hợp có . Một lần nữa gọi thuật toán trên phạm vi thấp hơn, bây giờ từ 64 đến 80, tạo ra biểu đồ có nhãn C (được hiển thị chấm để giúp phân biệt nó với những người khác). Mức tối thiểu là 77,5 (0,801 bit), tách ra một số khác ví dụ. Đồ thị D có giá trị tối thiểu là 73,5 (0,764 bit), tách hai có trường hợp. Biểu đồ E (một lần nữa được gạch ngang, hoàn toàn để dễ nhìn thấy hơn), cho phạm vi nhiệt độ từ 64 đến 72, có mức tối thiểu là 70,5 (0,796 bit), phân chia tất hai không và một có. Cuối cùng, đồ thị F, cho khoảng từ 64 đến 70, có cực tiểu ở 66,5 (0,4 bit).

Sự rời rạc hóa cuối cùng của thuộc tính nhiệt độ được thể hiện trong Hình 7.3. Thực tế là đệ quy chỉ xảy ra trong khoảng thời gian đầu tiên của mỗi lần phân tách là một tạo tác của ví dụ này: nói chung, cả khoảng trên và khoảng dưới sẽ phải chia nhỏ hơn nữa. Bên dưới mỗi phần là nhãn của biểu đồ trong Hình 7.2 chịu trách nhiệm về nó và bên dưới là giá trị thực của sự phân chia điểm.

Về mặt lý thuyết, có thể chỉ ra rằng điểm giới hạn làm giảm thiểu giá trị thông tin sẽ không bao giờ xảy ra giữa hai trường hợp của cùng một loại. Điều này dẫn để tối ưu hóa hữu ích: chỉ cần xem xét các bộ phận tiềm năng mà các trường hợp riêng biệt của các lớp khác nhau. Lưu ý rằng nếu nhãn lớp được gán cho các khoảng dựa trên lớp đa số trong khoảng, sẽ không có gì đảm bảo rằng các khoảng liền kề sẽ nhận được các nhãn khác nhau. Bạn có thể là muốn xem xét hợp nhất các khoảng thời gian với cùng một loại đa số (ví dụ: hai khoảng đầu tiên của Hình 7.3), nhưng như chúng ta sẽ thấy ở phần sau (trang 302–304), đây là không phải là một điều tốt để làm nói chung.

Vấn đề duy nhất còn lại để xem xét là tiêu chí dừng. Trong ví dụ về nhiệt độ, hầu hết các khoảng thời gian đã được xác định là “tinh khiết” trong đó tất cả các thể hiện của chúng có cùng một lớp và rõ ràng không có ích gì khi cố gắng phân tách một khoảng như vậy. (Ngoại lệ là khoảng thời gian cuối cùng, mà chúng tôi đã ngầm quyết định không chia nhỏ và khoảng từ 70,5 đến 73,5.) Tuy nhiên, nói chung, mọi thứ đều không đơn giản như vậy.

Một cách hay để dừng thủ tục phân tách dựa trên entropy lần lượt ra là nguyên tắc MDL mà chúng ta đã gặp trong Chương 5. Theo Với nguyên tắc đó, chúng tôi muốn giảm thiểu kích thước của "lý thuyết" cộng với kích thước của thông tin cần thiết để xác định tất cả các dữ liệu đưa ra lý thuyết đó. trong này trường hợp, nếu chúng tôi thực hiện phân tách, thì "lý thuyết" là điểm phân tách và chúng tôi đang so sánh tình huống mà chúng tôi chia tay với tình huống mà chúng tôi không chia tay. Trong cả hai trường hợp, chúng tôi giả sử rằng các thể hiện được biết nhưng nhãn lớp của chúng thì không. Nếu chúng ta không chia tay, các lớp có thể được truyền đi bằng cách mã hóa nhãn của từng cá thể. Nếu chúng tôi làm, trước tiên chúng tôi mã hóa điểm phân tách (theo log2[N - 1] bit, trong đó N là số lượng phiên bản), sau đó là các lớp của các thực thể bên dưới điểm đó, và sau đó là các lớp của các thực thể đó phía trên nó. Bạn có thể tưởng tượng rằng nếu sự phân chia là tốt—chẳng hạn, tất cả các lớp bên dưới nó là có và tất cả những điều trên là không—khi đó, có rất nhiều thứ có thể đạt được bằng cách chia nhỏ. Nếu có số lượng phiên bản có và không bằng nhau , mỗi phiên bản có giá 1 bit mà không tách nhưng hầu như không nhiều hơn 0 bit khi tách—nó không hoàn toàn 0 vì các giá trị lớp được liên kết với bản thân phần tách phải được mã hóa, nhưng hình phạt này được khấu hao trên tất cả các trường hợp. Trong trường hợp này, nếu có nhiều ví dụ, hình phạt của việc phải mã hóa điểm phân tách sẽ vượt xa bởi thông tin được lưu bằng cách chia tách.

Chúng tôi đã nhấn mạnh trong Phần 5.9 rằng khi áp dụng nguyên tắc MDL, ma quỷ là trong các chi tiết. Trong trường hợp rời rạc hóa tương đối đơn giản, tình huống có thể xử lý được mặc dù không đơn giản. Lượng thông tin có thể được thu được chính xác theo các giả định hợp lý nhất định. Chúng tôi sẽ không đi sâu vào chi tiết, nhưng kết quả cuối cùng là sự phân chia được quyết định bởi một điểm cắt cụ thể có giá trị nếu thông tin đạt được cho sự phân chia đó vượt quá một giá trị nhất định phụ thuộc vào trên số trường hợp N, số lớp k, entropy của trường hợp E, entropy của các trường hợp trong mỗi khoảng con E1 và E2, và số lớp biểu diễn trong mỗi khoảng con k1 và k2:

$$\text{nhận được} > \frac{\log_2(N-1)}{N} + \frac{\log_2(3^k-2) - kE + k_1E_1 + k_2E_2}{N}.$$

Thành phần đầu tiên là thông tin cần thiết để xác định điểm chia tách; thứ hai là hiệu chỉnh do cần truyền tải lớp nào tương ứng đến các khoảng con trên và dưới.

Khi áp dụng cho ví dụ về nhiệt độ, tiêu chí này hoàn toàn ngăn ngừa bất kỳ sự phân tách nào. Phần tách đầu tiên chỉ loại bỏ ví dụ cuối cùng và như bạn có thể tưởng tượng rất ít thông tin thực tế thu được bằng cách này khi truyền các lớp— trên thực tế, tiêu chí MDL sẽ không bao giờ tạo khoảng chỉ chứa một ví dụ. Việc không thể phân biệt nhiệt độ một cách hiệu quả sẽ ngăn không cho nó phát bất kỳ vai trò trong cấu trúc quyết định cuối cùng bởi vì cùng một giá trị rời rạc sẽ được cấp cho tất cả các trường hợp. Trong tình huống này, điều này là hoàn toàn phù hợp: sự nóng nảy

thuộc tính ature không xảy ra trong cây quyết định tốt hoặc quy tắc cho thời tiết
dữ liệu. Trên thực tế, việc không thể rời rạc hóa tương đương với việc lựa chọn thuộc tính.

### Các phương pháp rời rạc hóa khác

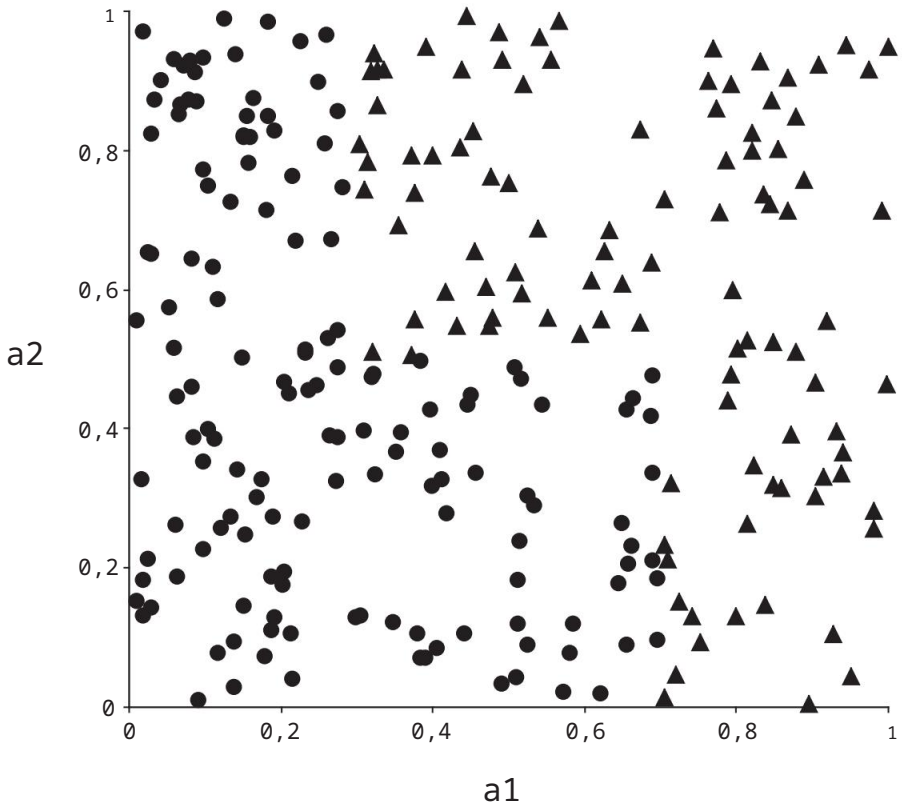
Phương pháp dựa trên entropy với tiêu chí dừng MDL là một trong những phương pháp tốt nhất
các kỹ thuật chung cho sự rời rạc hóa có giám sát. Tuy nhiên, nhiều phương pháp khác
Đã được điều tra. Ví dụ: thay vì tiến hành từ trên xuống bằng cách lặp lại một cách tự động
các khoảng thời gian cho đến khi một số tiêu chí dừng được thỏa mãn, bạn có thể
làm việc từ dưới lên, đầu tiên đặt từng trường hợp vào khoảng thời gian riêng của nó và sau
đó xem xét liệu có hợp nhất các khoảng thời gian liền kề hay không. Bạn có thể áp dụng một
tiêu chí thống kê để xem đâu sẽ là hai khoảng thời gian tốt nhất để hợp nhất và hợp nhất chúng nếu
thống kê vượt quá một mức độ tin cậy đặt trước nhất định, lặp lại thao tác
cho đến khi không có sự hợp nhất tiềm năng nào vượt qua bài kiểm tra. Bài thi c2 là một bài thi phù hợp và có
đã được sử dụng cho mục đích này. Thay vì chỉ định ngưỡng ý nghĩa đặt trước,
các kỹ thuật phức tạp hơn có sẵn để tự động xác định mức thích hợp.

Một cách tiếp cận khá khác là đếm số lỗi mà một đĩa cretization tạo ra khi dự đoán lớp
của từng cá thể huấn luyện, giả sử rằng
mỗi khoảng nhận lớp đa số. Ví dụ, phương pháp 1R được mô tả
trước đó dựa trên lỗi—nó tập trung vào lỗi hơn là entropy. Tuy nhiên,
sự rời rạc tốt nhất có thể về số lượng lỗi có được bằng cách sử dụng
số lượng khoảng thời gian lớn nhất có thể và nên tránh trường hợp suy biến này
bằng cách hạn chế số lượng khoảng thời gian trước. Ví dụ, bạn có thể hỏi,
cách tốt nhất để phân biệt một thuộc tính thành các khoảng k theo cách tối thiểu hóa số lượng
lỗi là gì?

Phương pháp brute-force để tìm ra cách tốt nhất để phân vùng một thuộc tính
thành k khoảng thời gian theo cách giảm thiểu số lỗi là theo cấp số nhân theo k và
do đó không khả thi. Tuy nhiên, có nhiều chương trình hiệu quả hơn dựa trên
về ý tưởng lập trình động. Lập trình động không chỉ áp dụng
đối với phép đo đếm sai số mà còn đối với bất kỳ hàm tạp chất phụ gia đã cho nào, và
nó có thể tìm thấy sự phân chia của N trường hợp thành k khoảng thời gian theo cách giảm
thiểu tạp chất theo thời gian tỷ lệ với $kN^2$ sự rời rạc. Điều này cung cấp một cách để tìm
dựa trên entropy tốt nhất, mang lại sự cải thiện tiềm năng trong
chất lượng của sự rời rạc hóa (nhưng trong thực tế là không đáng kể) so với đệ quy
phương pháp dựa trên entropy đã mô tả trước đây. Tin tức cho việc tạo đĩa dựa trên lỗi thậm
chí còn tốt hơn, bởi vì có một phương pháp giảm thiểu lỗi
tính theo thời gian tuyến tính tính bằng N.

### Phân biệt dựa trên entropy so với dựa trên lỗi

Tại sao không sử dụng rời rạc dựa trên lỗi, vì có thể rời rạc hóa tối ưu
tìm thấy rất nhanh? Câu trả lời là có một nhược điểm nghiêm trọng đối với dựa trên lỗi




Hình 7.4 Phân phối lớp cho bài toán hai lớp, hai thuộc tính.

rời rạc hóa: nó không thể tạo ra các khoảng thời gian liền kề có cùng nhãn (chẳng hạn như hai hình đầu tiên của Hình 7.3). Lý do là việc hợp nhất hai khoảng thời gian như vậy sẽ không ảnh hưởng đến số lượng lỗi nhưng nó sẽ giải phóng một khoảng thời gian có thể được sử dụng ở nơi khác để giảm số lượng lỗi.

Tại sao mọi người muốn tạo các khoảng thời gian liền kề có cùng nhãn? Lý do được minh họa rõ nhất bằng một ví dụ. Hình 7.4 cho thấy trường hợp không gian cho một vấn đề hai lớp đơn giản với hai thuộc tính số khác nhau, từ 0 đến 1. Các trường hợp thuộc về một lớp (dấu chấm) nếu thuộc tính đầu tiên của chúng (a1) nhỏ hơn hơn 0,3 hoặc nếu nó nhỏ hơn 0,7 và thuộc tính thứ hai của chúng (a2) nhỏ hơn 0,5. Mặt khác, chúng thuộc về lớp khác (hình tam giác). Dữ liệu trong Hình 7.4 có được tạo ra một cách giả tạo theo quy tắc này.

Bây giờ, giả sử chúng ta đang cố gắng rời rạc hóa cả hai thuộc tính với mục đích học các lớp từ các thuộc tính rời rạc. Sự phân chia rời rạc tốt nhất a1 thành ba khoảng (0 đến 0,3, 0,3 đến 0,7 và 0,7 đến 1,0) và a2 thành hai khoảng (0 đến 0,5 và 0,5 đến 1,0). Với những danh nghĩa này

thuộc tính, sẽ dễ dàng học cách phân biệt các lớp bằng cây quyết định đơn giản hoặc thuật toán quy tắc. Rời rạc hóa a2 là không có vấn đề. Tuy nhiên, đối với a1, khoảng đầu tiên và cuối cùng sẽ có các nhãn đối diện (tương ứng là dấu chấm và hình tam giác ). Thứ hai sẽ có bất kỳ nhãn nào xảy ra nhiều nhất trong khu vực từ 0,3 đến 0,7 (thực tế là chấm đối với dữ liệu trong Hình 7.4). Dù bằng cách nào, nhãn này chắc chắn phải giống với một trong các nhãn liền kề—tất nhiên điều này đúng bất kể xác suất của lớp xảy ra ở khu vực giữa. Do đó, việc xác định rõ ràng này sẽ không đạt được bằng bất kỳ phương pháp nào giúp giảm thiểu số lượng lỗi, bởi vì một phương pháp như vậy không thể tạo ra các khoảng thời gian liền kề có cùng nhãn.

Vấn đề là những gì thay đổi khi giá trị của a1 vượt qua ranh giới ở mức 0,3 không phải là giai cấp đa số mà là sự phân bố giai cấp. Tầng lớp đa số còn lại dấu chấm. Tuy nhiên, phân phối thay đổi rõ rệt, từ 100% trước giới hạn đến chỉ hơn 50% sau giới hạn. Và phân phối lại thay đổi khi vượt qua giới hạn ary tại 0,7, từ 50% thành 0%. Các phương pháp rời rạc hóa dựa trên Entropy nhạy cảm với những thay đổi trong phân phối mặc dù lớp đa số không không thay đổi. Các phương pháp dựa trên lỗi thì không.

## Chuyển đổi rời rạc thành thuộc tính số

Có một vấn đề ngược lại với sự rời rạc. Một số thuật toán học—đáng chú ý là phương pháp dựa trên cá thể lân cận gần nhất và dự đoán số các kỹ thuật liên quan đến hồi quy—tự nhiên chỉ xử lý các thuộc tính được số. Làm thế nào chúng có thể được mở rộng cho các thuộc tính danh nghĩa?

Trong học tập dựa trên cá thể, như được mô tả trong Phần 4.7, các thuộc tính rời rạc có thể được coi là số bằng cách xác định "khoảng cách" giữa hai giá trị danh nghĩa giống 0 và giữa hai giá trị khác 1—ít quan tâm đến các giá trị thực liên quan. Thay vì sửa đổi chức năng khoảng cách, điều này có thể đạt được bằng cách sử dụng phép biến đổi thuộc tính: thay thế giá trị k thuộc tính danh nghĩa với k thuộc tính nhị phân tổng hợp, một thuộc tính cho mỗi giá trị cho biết thuộc tính có giá trị đó hay không. Nếu các thuộc tính có bằng nhau trọng lượng, điều này đạt được hiệu quả tương tự đối với chức năng khoảng cách. khoảng cách là không nhạy cảm với các giá trị thuộc tính bởi vì chỉ có thông tin “giống nhau” hoặc “khác nhau” được mã hóa, không phải các sắc thái khác biệt có thể liên quan đến các giá trị có thể khác nhau của thuộc tính. Sự khác biệt tinh tế hơn có thể được thực hiện nếu các thuộc tính có trọng số phản ánh tầm quan trọng tương đối của chúng.

Nếu các giá trị của thuộc tính có thể được sắp xếp theo thứ tự, thì sẽ có nhiều khả năng hơn. Cho một vấn đề dự đoán số, giá trị lớp trung bình tương ứng với mỗi giá trị của một thuộc tính danh nghĩa có thể được tính toán từ các trường hợp đào tạo và được sử dụng để xác định thứ tự—kỹ thuật này đã được giới thiệu cho cây mô hình trong Phần 6.5. (Thật khó để đưa ra một cách đặt hàng tương tự các giá trị thuộc tính cho một vấn đề phân loại.) Một thuộc tính danh nghĩa được sắp xếp có thể được thay thế bằng một số nguyên theo cách hiển nhiên—nhưng điều này ngụ ý không chỉ

một thứ tự mà còn là một số liệu về các giá trị của thuộc tính. Hàm ý của
có thể tránh được một số liệu bằng cách tạo k - 1 thuộc tính nhị phân tổng hợp cho một thuộc
tính danh nghĩa có giá trị k, theo cách được mô tả ở trang 297. Mã hóa này vẫn bao hàm một thứ
tự giữa các giá trị khác nhau của thuộc tính—liền kề
các giá trị chỉ khác nhau ở một trong các thuộc tính tổng hợp, trong khi các giá trị ở xa khác
nhau ở một vài thuộc tính — nhưng nó không ngụ ý khoảng cách bằng nhau giữa các thuộc tính
các giá trị.

## 7.3 Một số phép biến đổi hữu ích

Những người khai thác dữ liệu tháo vát có một hộp công cụ chứa đầy các kỹ thuật, chẳng hạn như
phân biệt, để chuyển đổi dữ liệu. Như chúng tôi đã nhấn mạnh trong Phần 2.4, khai phá dữ liệu là
hầu như không bao giờ chỉ đơn giản là lấy một tập dữ liệu và áp dụng một thuật toán học tập cho
nó. Mọi vấn đề đều khác nhau. Bạn cần suy nghĩ về dữ liệu và
ý nghĩa của nó, và xem xét nó từ nhiều quan điểm khác nhau—một cách sáng tạo!—để
đạt được một quan điểm phù hợp. Chuyển đổi nó theo những cách khác nhau có thể giúp bạn
bắt đầu.

Bạn không cần phải tạo hộp công cụ của riêng mình bằng cách thực hiện các kỹ thuật
bản thân bạn. Môi trường toàn diện để khai thác dữ liệu, chẳng hạn như môi trường
được mô tả trong Phần II của cuốn sách này, bao gồm nhiều công cụ phù hợp dành cho bạn
để sử dụng. Bạn không nhất thiết phải hiểu chi tiết về cách chúng
thực hiện. Những gì bạn cần là hiểu những gì các công cụ làm và làm thế nào
chúng có thể được áp dụng. Trong Phần II, chúng tôi liệt kê và mô tả ngắn gọn tất cả các chuyển
đổi trong bàn làm việc khai thác dữ liệu Weka.

Dữ liệu thường yêu cầu các phép biến đổi toán học chung của một tập hợp các thuộc tính. Có
thể hữu ích khi xác định các thuộc tính mới bằng cách áp dụng các hàm toán học cụ thể cho các
thuộc tính hiện có. Hai thuộc tính ngày có thể được trừ vào
đưa ra một thuộc tính thứ ba đại diện cho tuổi—một ví dụ về sự biến đổi ngữ nghĩa được thúc đẩy
bởi ý nghĩa của các thuộc tính ban đầu. Các phép biến đổi khác
có thể được đề xuất bởi các thuộc tính đã biết của thuật toán học tập. Nếu tuyến tính
mối quan hệ liên quan đến hai thuộc tính, A và B, bị nghi ngờ và thuật toán
chỉ có khả năng phân tách song song trục (vì hầu hết người học cây quyết định và quy tắc
là), tỷ lệ A/B có thể được định nghĩa là một thuộc tính mới. Các phép biến hình là
không nhất thiết là toán học nhưng có thể liên quan đến kiến thức thế giới như
các ngày trong tuần, ngày lễ của công dân, hoặc số nguyên tử hóa học. Họ có thể là
được thể hiện dưới dạng các thao tác trong bảng tính hoặc dưới dạng các chức năng được triển khai
bằng các chương trình máy tính tùy ý. Hoặc bạn có thể giảm một số thuộc tính danh nghĩa
thành một bằng cách nối các giá trị của chúng, tạo ra một thuộc tính có giá trị k1 ¥ k2 từ các
thuộc tính có giá trị k1 và k2 tương ứng. Sự rời rạc hóa chuyển đổi một
thuộc tính số thành danh nghĩa và trước đó chúng ta đã thấy cách chuyển đổi sang thuộc tính khác
hướng quá.

Là một loại chuyển đổi khác, bạn có thể áp dụng quy trình phân cụm vào tập dữ liệu và sau đó xác định một thuộc tính mới có giá trị cho bất kỳ trường hợp cụ thể nào là cụm chứa nó bằng cách sử dụng nhãn tùy ý cho cụm. Ngoài ra, với phân cụm xác suất, bạn có thể tăng cường từng trường hợp bằng xác suất thành viên cho mỗi cụm, bao gồm càng nhiều thuộc tính mới càng tốt có cụm.

Đôi khi sẽ hữu ích khi thêm nhiễu vào dữ liệu, có lẽ để kiểm tra độ chắc chắn của một thuật toán học tập. Để lấy một thuộc tính danh nghĩa và thay đổi một phần trăm tuổi nhất định của các giá trị của nó. Để xáo trộn dữ liệu bằng cách đổi tên quan hệ, tên thuộc tính, và các giá trị thuộc tính chuỗi và danh nghĩa—vì thường cần phải ẩn danh các bộ dữ liệu nhạy cảm. Để ngẫu nhiên hóa thứ tự của các trường hợp hoặc tạo ra một mẫu ngẫu nhiên của tập dữ liệu bằng cách lấy mẫu lại. Để giảm tập dữ liệu bằng cách loại bỏ một tỷ lệ phần trăm nhất định các phiên bản hoặc tất cả các phiên bản có giá trị nhất định cho thuộc tính danh nghĩa hoặc giá trị số trên hoặc dưới một ngưỡng nhất định. Hoặc để loại bỏ các ngoại lệ bằng cách áp dụng phương pháp phân loại cho tập dữ liệu và xóa trường hợp phân loại sai.

Các loại đầu vào khác nhau gọi cho các phép biến đổi của riêng chúng. Nếu bạn có thể nhập tệp dữ liệu thưa thớt (xem Phần 2.4), bạn có thể cần chuyển đổi bộ dữ liệu sang dạng không thưa thớt và ngược lại. Lệnh gọi đầu vào chuỗi thời gian và đầu vào văn bản cho các chuyển đổi chuyên biệt của riêng chúng, được mô tả trong các tiểu mục tiếp theo. Nhưng đầu tiên chúng ta xem xét hai kỹ thuật chung để chuyển đổi dữ liệu bằng số các thuộc tính thành dạng chiều thấp hơn có thể hữu ích hơn cho dữ liệu khai thác mỏ.

## Phân tích thành phần chính

Trong tập dữ liệu có k thuộc tính số, bạn có thể trực quan hóa dữ liệu dưới dạng một đám mây điểm trong không gian k chiều—những vì sao trên bầu trời, một bầy ruồi đông cứng trong thời gian, một biểu đồ phân tán hai chiều trên giấy. Các thuộc tính đại diện cho tọa độ của không gian. Nhưng các trục bạn sử dụng, bản thân hệ tọa độ, là arbi trary. Bạn có thể đặt các trục ngang và dọc trên giấy và biểu thị các điểm của biểu đồ phân tán bằng cách sử dụng các tọa độ đó hoặc bạn có thể vẽ một đường thẳng arbi để biểu thị trục X và một đường vuông góc với nó để biểu thị trục Y. Để ghi lại vị trí của ruồi, bạn có thể sử dụng một biểu đồ thông thường hệ tọa độ với trục bắc-nam, trục đông-tây và trục lên-xuống trục. Nhưng các hệ tọa độ khác sẽ hoạt động tốt như nhau. Các sinh vật như ruồi không biết về bắc, nam, đông và tây—mặc dù, phải tuân theo trọng lực, họ có thể cảm nhận lên xuống như một thứ gì đó đặc biệt. Đối với các ngôi sao trên bầu trời, ai sẽ nói hệ tọa độ "đúng" là gì? Qua nhiều thế kỷ tổ tiên của chúng ta đã chuyển từ quan điểm địa tâm sang quan điểm nhật tâm sang quan điểm hoàn toàn tương đối tính, mỗi sự thay đổi quan điểm được kèm theo turbu

cho thấy những biến động tôn giáo-khoa học và sự xem xét lại đau đớn về vai trò của loài người trong vũ trụ của Chúa.

Quay lại tập dữ liệu. Giống như trong các ví dụ này, không có gì ngăn cản bạn chuyển đổi tất cả các điểm dữ liệu sang một hệ tọa độ khác. Nhưng không giống như những ví dụ này, trong khai thác dữ liệu thường có một hệ tọa độ ưu tiên, được xác định không phải bởi một số quy ước bên ngoài mà bởi chính dữ liệu đó. Dù bạn sử dụng tọa độ nào, đám mây điểm có một phương sai nhất định theo từng hướng, biểu thị mức độ lan truyền xung quanh giá trị trung bình theo hướng đó.
Có một thực tế kỳ lạ là nếu bạn cộng các phương sai dọc theo mỗi trục và sau đó biến đổi các điểm thành một hệ tọa độ khác và thực hiện tương tự ở đó, bạn sẽ nhận được tổng phương sai như nhau trong cả hai trường hợp. Điều này luôn đúng với điều kiện là các hệ tọa độ là trực giao, nghĩa là mỗi trục vuông góc với các trục khác.

Ý tưởng của phân tích thành phần chính là sử dụng một hệ tọa độ đặc biệt phụ thuộc vào đám mây điểm như sau: đặt trục đầu tiên theo hướng có phương sai lớn nhất của các điểm để cực đại hóa phương sai dọc theo trục đó. Trục thứ hai vuông góc với nó. Trong hai chiều không có sự lựa chọn—hướng của nó được xác định bởi trục thứ nhất—nhưng trong ba chiều, nó có thể nằm ở bất kỳ đâu trong mặt phẳng vuông góc với trục thứ nhất và trong các chiều cao hơn thậm chí còn có nhiều lựa chọn hơn, mặc dù nó luôn bị hạn chế vuông góc với trục thứ nhất. Tùy thuộc vào ràng buộc này, hãy chọn trục thứ hai theo cách tối đa hóa phương sai dọc theo nó. Tiếp tục, chọn từng trục để tối đa hóa phần phương sai còn lại của nó.

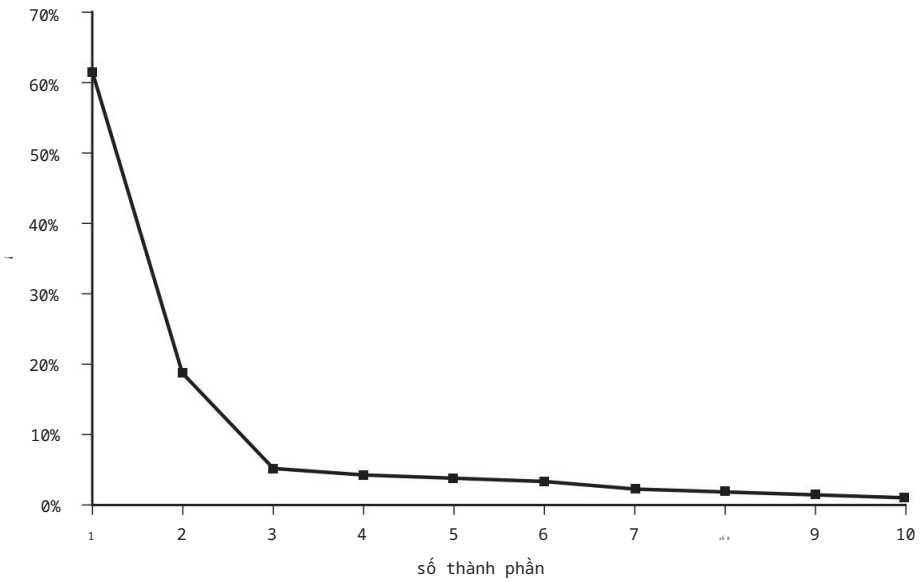
Làm thế nào để bạn làm điều này? Nó không khó nếu được cung cấp một chương trình máy tính phù hợp và không khó hiểu nếu được cung cấp các công cụ toán học phù hợp. Về mặt kỹ thuật—đối với những người hiểu các thuật ngữ in nghiêng—bạn tính toán ma trận hiệp phương sai của tọa độ ban đầu của các điểm và chéo hóa nó để tìm các vectơ riêng. Đây là các trục của không gian được biến đổi, được sắp xếp theo thứ tự giá trị riêng—bởi vì mỗi giá trị riêng cho phương sai dọc theo trục của nó.

Hình 7.5 cho thấy kết quả của việc chuyển đổi một tập dữ liệu cụ thể với 10 thuộc tính số, tương ứng với các điểm trong không gian 10 chiều. Hãy tưởng tượng tập dữ liệu ban đầu là một đám mây các điểm trong 10 chiều—chúng ta không thể vẽ nó!
Chọn trục đầu tiên dọc theo hướng có phương sai lớn nhất, trục thứ hai vuông góc với nó dọc theo hướng có phương sai lớn nhất tiếp theo, v.v. Bảng đưa ra phương sai dọc theo mỗi trục tọa độ mới theo thứ tự các trục được chọn. Vì tổng của các phương sai là không đổi bất kể hệ tọa độ, nên chúng được biểu thị bằng phần trăm của tổng đó. Chúng tôi gọi các thành phần **trục** và nói rằng mỗi thành phần "chiếm" phần phương sai của nó.

Hình 7.5(b) vẽ sơ đồ phương sai mà mỗi thành phần tính theo số của thành phần. Bạn có thể sử dụng tất cả các thành phần làm thuộc tính mới để khai thác dữ liệu hoặc bạn có thể chỉ muốn chọn một số thành phần đầu tiên, các thành phần chính,

| trục | phương sai | tích lũy |
|---|---|---|
| 1 | 61,2% | 61,2% |
| 2 | 18,0% | 79,2% |
| 3 | 4,7% | 83,9% |
| 4 | 4,0% | 87,9% |
| 5 | 3,2% | 91,1% |
| 6 | 2,9% | 94,0% |
| 7 | 2,0% | 96,0% |
| ... | 1,7% | 97,7% |
| 9 | 1,4% | 99,1% |
| 10 | 0,9% | 100% |

(Một)




(b)

Hình 7.5 Biến đổi các thành phần chính của một tập dữ liệu: (a) phương sai của từng thành phần và (b) biểu đồ phương sai.

và loại bỏ phần còn lại. Trong trường hợp này, ba thành phần chính chiếm 84% của phương sai trong tập dữ liệu; bảy chiếm hơn 95%.

Trên các bộ dữ liệu số, người ta thường sử dụng phân tích thành phần chính trước khi khai thác dữ liệu như một hình thức dọn dẹp dữ liệu và tạo thuộc tính. Ví dụ, bạn có thể muốn thay thế các thuộc tính số bằng thuộc tính chính các trục thành phần hoặc với một tập hợp con của chúng chiếm một tỷ lệ nhất định—chẳng hạn, 95%—của phương sai. Lưu ý rằng quy mô của các thuộc tính ảnh hưởng đến

kết quả của phân tích các thành phần chính, và thông lệ trước tiên là chuẩn hóa tất cả các thuộc tính về giá trị trung bình bằng 0 và phương sai đơn vị.

Một khả năng khác là áp dụng đệ quy phép phân tích thành phần chính trong bộ học cây quyết định. Ở mỗi giai đoạn, một người học cây quyết định thông thường chọn tách theo hướng song song với một trong các trục. Tuy nhiên, giả sử một phép biến đổi thành phần chính được thực hiện trước và người học chọn một trục trong không gian được biến đổi. Điều này tương đương với sự phân chia dọc theo một đường xiên trong không gian ban đầu. Nếu phép biến đổi được thực hiện lại trước mỗi lần phân chia, kết quả sẽ là một cây quyết định đa biến có các đường phân chia theo các hướng không song song với các trục hoặc với nhau.

### Phép chiếu ngẫu nhiên

Phân tích thành phần chính biến đổi dữ liệu tuyến tính thành không gian có chiều thấp hơn. Nhưng nó đắt. Thời gian cần thiết để tìm ra sự hình thành trans (là một ma trận bao gồm các vectơ riêng của ma trận hiệp phương sai) là lập phương trong số thứ nguyên. Điều này làm cho nó không khả thi đối với các bộ dữ liệu có số lượng lớn các thuộc tính. Một giải pháp thay thế đơn giản hơn nhiều là sử dụng phép chiếu ngẫu nhiên của dữ liệu vào một không gian con với số lượng kích thước được xác định trước. Rất dễ dàng để tìm một ma trận chiếu ngẫu nhiên. Nhưng nó sẽ được bất kỳ tốt?

Trên thực tế, lý thuyết cho thấy rằng các phép chiếu ngẫu nhiên trung bình duy trì các mối quan hệ khoảng cách khá tốt. Điều này có nghĩa là chúng có thể được sử dụng kết hợp với cây kD hoặc cây bóng để thực hiện tìm kiếm lân cận gần nhất trong không gian có số lượng kích thước khổng lồ. Đầu tiên chuyển đổi dữ liệu để giảm số lượng thuộc tính; sau đó xây dựng một cái cây cho không gian được chuyển đổi. Trong trường hợp phân loại hàng xóm gần nhất, bạn có thể làm cho kết quả ổn định hơn và ít phụ thuộc hơn vào việc lựa chọn phép chiếu ngẫu nhiên, bằng cách xây dựng một bộ phân loại tập hợp sử dụng nhiều ma trận ngẫu nhiên.

Không có gì đáng ngạc nhiên, các phép chiếu ngẫu nhiên hoạt động kém hơn các phép chiếu được lựa chọn cẩn thận bởi phân tích thành phần chính khi được sử dụng để xử lý trước dữ liệu cho một loạt các bộ phân loại tiêu chuẩn. Tuy nhiên, kết quả thử nghiệm đã chỉ ra rằng sự khác biệt không quá lớn—và nó có xu hướng giảm khi số lượng thứ nguyên tăng lên. Và tất nhiên, các phép chiếu ngẫu nhiên rẻ hơn nhiều về mặt tính toán.

### Chuyển văn bản thành vectơ thuộc tính

Trong Phần 2.4, chúng tôi đã giới thiệu các thuộc tính chuỗi chứa các đoạn văn bản và nhận xét rằng giá trị của thuộc tính chuỗi thường là toàn bộ tài liệu. Các thuộc tính chuỗi về cơ bản là danh nghĩa, với số lượng giá trị không xác định. Nếu chúng được coi đơn giản là các thuộc tính danh nghĩa, các mô hình có thể được xây dựng phụ thuộc vào việc giá trị của hai thuộc tính chuỗi có bằng nhau hay không. Nhưng điều đó không

nắm bắt bất kỳ cấu trúc bên trong nào của chuỗi hoặc đưa ra bất kỳ khía cạnh thú vị nào của văn bản mà nó đại diện.

Bạn có thể tưởng tượng việc phân tách văn bản trong thuộc tính chuỗi thành các đoạn văn, câu, hoặc cụm từ. Tuy nhiên, nói chung, từ là đơn vị hữu ích nhất. Các văn bản trong một thuộc tính chuỗi thường là một chuỗi các từ và thường được diễn đạt tốt nhất theo các từ mà nó chứa. Ví dụ, bạn có thể biến đổi thuộc tính chuỗi thành một tập hợp các thuộc tính số, một thuộc tính cho mỗi từ, biểu thị tần suất xuất hiện của từ đó. Tập hợp các từ—nghĩa là tập hợp các thuộc tính mới—được xác định từ tập dữ liệu và thường khá lớn. Nếu có một số thuộc tính chuỗi mà các thuộc tính của chúng nên được xử lý riêng biệt, thuộc tính mới tên thuộc tính phải được phân biệt, có thể bằng tiền tố do người dùng xác định.

Chuyển đổi thành từ—tokenization— không phải là một hoạt động đơn giản vì nó âm thanh. Mã thông báo có thể được hình thành từ các chuỗi chữ cái liền kề với các ký tự không phải chữ cái bị loại bỏ. Nếu có số, các dãy số có thể cũng được giữ lại. Các số có thể bao gồm dấu + hoặc -, có thể chứa dấu thập phân, và có thể có ký hiệu hàm mũ—nói cách khác, chúng phải được phân tích cú pháp theo một cú pháp số xác định. Một dãy chữ số có thể là được coi là một mã thông báo duy nhất. Có lẽ ký tự khoảng trắng là dấu phân cách mã thông báo; có lẽ khoảng trắng (bao gồm tab và ký tự dòng mới) là dấu phân cách và có lẽ dấu chấm câu cũng vậy. Thời kỳ có thể khó khăn: đôi khi họ nên được coi là một phần của từ (ví dụ: với tên viết tắt, tiêu đề, chữ viết tắt, và số), nhưng đôi khi chúng không nên (ví dụ: nếu chúng là dấu phân cách câu). Dấu gạch ngang và dấu nháy đơn cũng có vấn đề tương tự.

Tất cả các từ có thể được chuyển đổi thành chữ thường trước khi được thêm vào từ điển. Các từ trong một danh sách các từ chức năng hoặc từ dừng cố định, được xác định trước — chẳng hạn như , và, và nhưng—có thể bỏ qua. Lưu ý rằng danh sách từ dừng là ngôn ngữ sự phụ thuộc. Trên thực tế, quy ước viết hoa cũng vậy (tiếng Đức viết hoa tất cả danh từ), cú pháp số (người châu Âu sử dụng dấu phẩy cho dấu thập phân), quy ước về dấu câu (tiếng Tây Ban Nha có dấu chấm hỏi ở đầu), và, tất nhiên, bộ ký tự. Văn bản là phức tạp!

Các từ có tần suất xuất hiện thấp như hapax legomena[3] cũng thường bị loại bỏ. Đôi khi, việc giữ lại k từ thường gặp nhất sau khi các từ dừng đã bị xóa — hoặc có lẽ là k từ hàng đầu cho mỗi loại được cho là có ích.

Cùng với tất cả các tùy chọn mã thông báo này, cũng có câu hỏi về giá trị của mỗi thuộc tính từ nên là gì. Giá trị có thể là từ đếm—số lần từ xuất hiện trong chuỗi—hoặc có thể đơn giản là chỉ ra sự hiện diện hoặc vắng mặt của từ. Tần số từ có thể được chuẩn hóa thành cung cấp cho mỗi vectơ thuộc tính của tài liệu cùng độ dài Euclide. Ngoài ra,

[3] Hapax legomena là một từ chỉ xuất hiện một lần trong một văn bản nhất định.

các tần số fij cho từ i trong tài liệu j có thể được chuyển đổi theo nhiều cách khác nhau. Một thước đo tần số thuật ngữ logarit tiêu chuẩn là log (1 + fij). MỘT biện pháp được sử dụng rộng rãi trong truy xuất thông tin là TF ¥ IDF, hay "tần suất thuật ngữ nhân với tần suất tài liệu nghịch đảo". Ở đây, thuật ngữ tần suất được điều chỉnh bởi một yếu tố phụ thuộc vào mức độ phổ biến của từ này trong các ngôn ngữ khác. các tài liệu. Số liệu TF ¥ IDF thường được định nghĩa là

$$f_{ij} \frac{\text{số tài liệu}}{\text{nhật ký số lượng tài liệu bao gồm từ}}.$$

Ý tưởng là một tài liệu về cơ bản được đặc trưng bởi các từ xuất hiện thường xuyên trong đó, chiếm yếu tố đầu tiên, ngoại trừ các từ được sử dụng trong mọi tài liệu hoặc hầu hết mọi tài liệu đều vô dụng vì là công cụ phân biệt đối xử, điều này chiếm thứ hai. TF ¥ IDF không chỉ được sử dụng để chỉ cụ thể này công thức mà còn cho một lớp chung các biện pháp cùng loại. Ví dụ, hệ số tần suất fij có thể được thay thế bằng một số hạng logarit chẳng hạn như log (1 + fij).

Chuỗi thời gian

Trong dữ liệu chuỗi thời gian, mỗi trường hợp đại diện cho một bước thời gian khác nhau và các thuộc tính cung cấp các giá trị liên quan đến thời gian đó—chẳng hạn như trong dự báo thời tiết hoặc dự đoán thị trường chứng khoán. Đôi khi bạn cần để có thể thay thế một giá trị của thuộc tính trong trường hợp hiện tại với giá trị tương ứng trong một số trường hợp khác trong quá khứ hoặc tương lai. Nó thậm chí còn phổ biến hơn để thay thế giá trị của một thuộc tính với sự khác biệt giữa giá trị hiện tại và giá trị trong một số trường hợp trước đó. Ví dụ, sự khác biệt - thường được gọi là Delta—giữa giá trị hiện tại và giá trị trước đó thường mang nhiều thông tin hơn bản thân giá trị đó. Trường hợp đầu tiên, trong đó thời gian thay đổi giá trị không xác định, có thể bị xóa hoặc thay thế bằng một giá trị bị thiếu. đồng bằng giá trị về cơ bản là đạo hàm đầu tiên được chia tỷ lệ theo một số hằng số phụ thuộc vào kích thước của bước thời gian. Các phép biến đổi Delta liên tiếp có đạo hàm cao hơn.

Trong một số chuỗi thời gian, các trường hợp không đại diện cho các mẫu thông thường, nhưng thời gian của mỗi trường hợp được cung cấp bởi một **thuộc tính** dấu thời gian . Sự khác biệt giữa các dấu thời gian là kích thước bước cho trường hợp đó và nếu sự khác biệt liên tiếp được thực hiện đối với các thuộc tính khác, chúng nên được chia cho kích thước bước để chuẩn hóa phát sinh. Trong các trường hợp khác, mỗi thuộc tính có thể đại diện cho một thời gian khác, thay vì hơn mỗi trường hợp, sao cho chuỗi thời gian là từ thuộc tính này sang thuộc tính tiếp theo thay vì hơn từ trường hợp này sang trường hợp khác. Sau đó, nếu sự khác biệt là cần thiết, họ phải được lấy giữa giá trị của một thuộc tính và giá trị của thuộc tính tiếp theo cho mỗi ví dụ.

## 7.4 Tự động làm sạch dữ liệu

Một vấn đề gây khó khăn cho việc khai thác dữ liệu thực tế là chất lượng dữ liệu kém. Lỗi trong cơ sở dữ liệu lớn là cực kỳ phổ biến. Giá trị thuộc tính và giá trị lớp thường không đáng tin cậy và bị hỏng. Mặc dù một cách để giải quyết vấn đề này là kiểm tra cẩn thận dữ liệu, nhưng bản thân các kỹ thuật khai thác dữ liệu đôi khi có thể giúp giải quyết vấn đề.

Cải thiện cây quyết định Có

một thực tế đáng ngạc nhiên là cây quyết định được tạo ra từ dữ liệu huấn luyện thường có thể được đơn giản hóa mà không làm giảm độ chính xác bằng cách loại bỏ các trường hợp bị phân loại sai khỏi tập huấn luyện, học lại và sau đó lặp lại cho đến khi không còn trường hợp nào bị phân loại sai. Các thử nghiệm trên bộ dữ liệu tiêu chuẩn đã chỉ ra rằng điều này hầu như không ảnh hưởng đến độ chính xác phân loại của C4.5, một sơ đồ cảm ứng cây quyết định tiêu chuẩn. Trong một số trường hợp, nó cải thiện đôi chút; ở những người khác, nó xấu đi một chút. Sự khác biệt hiếm khi có ý nghĩa thống kê—và ngay cả khi có, lợi thế có thể đi theo cả hai hướng. Những gì kỹ thuật ảnh hưởng là kích thước cây quyết định. Các cây kết quả luôn nhỏ hơn nhiều so với các cây ban đầu, mặc dù chúng hoạt động gần như giống nhau.

Lý do cho điều này là gì? Khi một phương pháp quy nạp cây quyết định loại bỏ một cây con, nó sẽ áp dụng một bài kiểm tra thống kê để quyết định xem cây con đó có được "chứng minh" bởi dữ liệu hay không. Quyết định cắt bớt chấp nhận một sự hy sinh nhỏ về độ chính xác phân loại trên tập huấn luyện với niềm tin rằng điều này sẽ cải thiện hiệu suất của tập kiểm tra. Một số trường hợp đào tạo đã được phân loại chính xác bởi cây chưa cắt tỉa giờ sẽ bị phân loại sai bởi cây đã cắt tỉa. Trên thực tế, quyết định đã được đưa ra để bỏ qua các trường hợp đào tạo này.

Nhưng quyết định đó chỉ được áp dụng cục bộ, trong cây con được cắt tỉa. Hiệu ứng của nó không được phép thấm sâu hơn vào cây, có lẽ dẫn đến các lựa chọn khác nhau về các thuộc tính để phân nhánh. Loại bỏ các trường hợp được phân loại sai khỏi tập huấn luyện và học lại cây quyết định chỉ là đưa các quyết định cắt tỉa đến kết luận hợp lý của chúng. Nếu chiến lược cắt tỉa là một chiến lược tốt, điều này sẽ không gây hại cho hiệu suất. Nó thậm chí có thể cải thiện nó bằng cách cho phép thực hiện các lựa chọn thuộc tính tốt hơn.

Chắc chắn sẽ tốt hơn nếu tham khảo ý kiến của một chuyên gia về con người. Các trường hợp đào tạo bị phân loại sai có thể được đưa ra để xác minh và những trường hợp bị phát hiện là sai có thể bị xóa — hoặc tốt hơn là sửa chữa.

Lưu ý rằng chúng tôi đang giả định rằng các trường hợp không bị phân loại sai theo bất kỳ cách nào có hệ thống. Nếu các trường hợp bị hỏng một cách có hệ thống trong cả tập huấn luyện và tập kiểm tra—ví dụ: một giá trị lớp có thể được thay thế bằng một giá trị khác—thì chỉ có thể hy vọng rằng việc huấn luyện trên tập huấn luyện bị lỗi sẽ mang lại hiệu suất tốt hơn trên tập kiểm tra (cũng bị lỗi).

Thật thú vị, người ta đã chỉ ra rằng khi tiếng ồn nhân tạo được thêm vào các thuộc tính (chứ không phải cho các lớp), hiệu suất của tập kiểm tra được cải thiện nếu tiếng ồn tương tự được thêm vào tập huấn luyện theo cùng một cách. Nói cách khác, khi tiếng ồn thuộc tính là vấn đề, thì không nên huấn luyện trên một bộ "sạch" nếu hiệu suất được đánh giá trên một bộ "bẩn". Một phương pháp học có thể học cách bù nhiễu thuộc tính, trong một số biện pháp, nếu có cơ hội. Về bản chất, nó có thể tìm hiểu những thuộc tính nào không đáng tin cậy và nếu tất cả chúng đều không đáng tin cậy, thì cách tốt nhất để sử dụng chúng cùng nhau để mang lại kết quả đáng tin cậy hơn. Để loại bỏ nhiễu khỏi các thuộc tính cho tập huấn luyện sẽ từ chối cơ hội học cách tốt nhất để chống lại nhiễu đó. Nhưng với tiếng ồn của lớp (chứ không phải tiếng ồn thuộc tính), tốt nhất là đào tạo trên các trường hợp không có tiếng ồn nếu có thể.

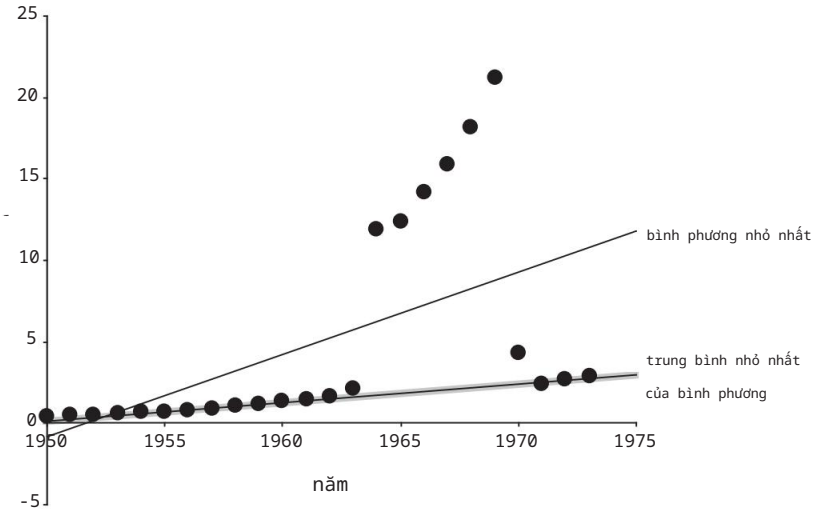
Hồi quy mạnh mẽ Các vấn

đề gây ra bởi dữ liệu nhiễu đã được biết đến trong hồi quy tuyến tính trong nhiều năm. Các nhà thống kê thường kiểm tra dữ liệu để tìm các ngoại lệ và loại bỏ chúng theo cách thủ công. Trong trường hợp hồi quy tuyến tính, các ngoại lệ có thể được xác định một cách trực quan—mặc dù không bao giờ hoàn toàn rõ ràng liệu một ngoại lệ là một lỗi hay chỉ là một giá trị đáng ngạc nhiên nhưng chính xác. Các ngoại lệ ảnh hưởng đáng kể đến hồi quy bình phương nhỏ nhất thông thường vì phép đo khoảng cách bình phương làm nổi bật ảnh hưởng của các điểm ở xa đường hồi quy.

Các phương pháp thống kê giải quyết vấn đề ngoại lệ được gọi là mạnh mẽ. Một cách để làm cho hồi quy mạnh mẽ hơn là sử dụng thước đo khoảng cách giá trị tuyệt đối thay vì bình phương thông thường. Điều này làm suy yếu ảnh hưởng của các ngoại lệ.
Một khả năng khác là cố gắng tự động xác định các ngoại lệ và loại bỏ chúng khỏi sự cân nhắc. Ví dụ: người ta có thể tạo một đường hồi quy và sau đó loại bỏ 10% số điểm nằm xa nhất khỏi đường đó.
Khả năng thứ ba là giảm thiểu trung bình (chứ không phải giá trị trung bình) của bình phương của các phân kỳ từ đường hồi quy. Hóa ra là công cụ ước lượng này rất mạnh mẽ và thực sự đối phó với các ngoại lệ theo hướng X cũng như các ngoại lệ theo hướng Y - đó là hướng bình thường mà người ta nghĩ về các ngoại lệ.

Một bộ dữ liệu thường được sử dụng để minh họa hồi quy mạnh là biểu đồ các cuộc gọi điện thoại quốc tế được thực hiện từ Bỉ từ năm 1950 đến năm 1973, được thể hiện trong Hình 7.6. Dữ liệu này được lấy từ Khảo sát thống kê của Bỉ do Min istry of Economy công bố. Cốt truyện dường như cho thấy một xu hướng đi lên trong những năm qua, nhưng có một nhóm điểm bất thường từ năm 1964 đến năm 1969. Hóa ra trong khoảng thời gian này, kết quả đã bị ghi nhầm vào tổng số phút của các cuộc gọi . Những năm 1963 và 1970 cũng bị ảnh hưởng một phần. Lỗi này gây ra một phần lớn các ngoại lệ theo hướng Y.

Không có gì ngạc nhiên khi đường hồi quy bình phương nhỏ nhất thông thường bị ảnh hưởng nghiêm trọng bởi dữ liệu bất thường này. Tuy nhiên, trung bình nhỏ nhất của đường bình phương vẫn còn




Hình 7.6 Số cuộc gọi điện thoại quốc tế từ Bỉ, 1950–1973.

đáng kể không bị xáo trộn. Dòng này có một giải thích đơn giản và tự nhiên. Về mặt địa lý, nó tương ứng với việc tìm dải hẹp nhất bao phủ một nửa các quan sát, trong đó độ dày của dải được đo theo hướng thẳng đứng—dải này được đánh dấu màu xám trong Hình 7.6; bạn cần nhìn kỹ mới thấy.

Đường trung tuyến nhỏ nhất nằm ở tâm chính xác của dải này. Lưu ý rằng khái niệm này thường dễ giải thích và hình dung hơn so với bình phương nhỏ nhất bình thường định nghĩa hồi quy. Thật không may, có một bất lợi nghiêm trọng đối với kỹ thuật hồi quy dựa trên trung bình: chúng phải chịu chi phí tính toán cao, điều này thường làm cho chúng không khả thi đối với các vấn đề thực tế.

## Phát hiện bất thường

Một vấn đề nghiêm trọng với bất kỳ hình thức tự động phát hiện dữ liệu có vẻ không chính xác nào là em bé có thể bị ném ra ngoài cùng với nước tắm. Nếu không tham khảo ý kiến của một chuyên gia về con người, thực sự không có cách nào để biết liệu một người cụ thể có ví dụ thực sự là một lỗi hoặc liệu nó không phù hợp với loại mô hình đang được áp dụng. Trong hồi quy thống kê, trực quan hóa giúp ích. Nó thường sẽ là rõ ràng bằng mắt thường, ngay cả đối với người không chuyên, nếu loại đường cong sai đang được được trang bị—ví dụ: một đường thẳng đang được trang bị cho dữ liệu nằm trên một hình parabol. Các ngoại lệ trong Hình 7.6 chắc chắn rất nổi bật. Nhưng hầu hết các vấn đề không thể hình dung dễ dàng như vậy: khái niệm "kiểu mẫu" tinh tế hơn một đường Hồi quy. Và mặc dù người ta biết rằng hầu hết các kết quả tốt đều đạt được bộ dữ liệu tiêu chuẩn bằng cách loại bỏ các trường hợp không phù hợp với mô hình cây quyết định, điều này không hẳn là rất thoải mái khi xử lý một vấn đề mới cụ thể

tập dữ liệu. Vẫn còn nghi ngờ rằng có lẽ bộ dữ liệu mới đơn giản là không phù hợp với mô hình cây quyết định.

Một giải pháp đã được thử là sử dụng một số lược đồ học tập khác nhau—chẳng hạn như cây quyết định, học viên lân cận gần nhất và hàm phân biệt tuyến tính—để lọc dữ liệu. Một cách tiếp cận thận trọng là yêu cầu cả ba lược đồ đều không phân loại chính xác một thực thể trước khi nó bị coi là có lỗi và bị xóa khỏi dữ liệu. Trong một số trường hợp, lọc dữ liệu theo cách này và sử dụng dữ liệu đã lọc làm đầu vào cho lược đồ học tập cuối cùng sẽ mang lại hiệu suất tốt hơn so với việc chỉ sử dụng ba lược đồ học tập và để chúng bình chọn về kết quả. Huấn luyện cả ba lược đồ trên dữ liệu đã lọc và để chúng bỏ phiếu có thể mang lại kết quả tốt hơn nữa. Tuy nhiên, có một mối nguy hiểm đối với các kỹ thuật bỏ phiếu: một số thuật toán học tập phù hợp với một số loại dữ liệu nhất định hơn các loại dữ liệu khác và phương pháp phù hợp nhất có thể đơn giản bị bỏ phiếu! Chúng ta sẽ xem xét một phương pháp tinh tế hơn để kết hợp đầu ra từ các bộ phân loại khác nhau, được gọi là xếp chồng, trong phần tiếp theo. Như thường lệ, bài học là tìm hiểu dữ liệu của bạn và xem xét dữ liệu đó theo nhiều cách khác nhau.

Một mối nguy hiểm có thể xảy ra với các phương pháp lọc là chúng có thể hình dung được rằng chúng chỉ đang hy sinh các thể hiện của một lớp cụ thể (hoặc nhóm các lớp) để cải thiện độ chính xác cho các lớp còn lại. Mặc dù không có cách chung để bảo vệ chống lại điều này, nhưng nó không được coi là một vấn đề phổ biến trong thực tế.

Cuối cùng, một lần nữa, điều đáng chú ý là lọc tự động là một giải pháp thay thế kém để lấy dữ liệu ngay từ đầu. Nếu điều này quá tốn thời gian và tốn kém để trở thành hiện thực, thì việc kiểm tra của con người có thể bị giới hạn ở những trường hợp được bộ lọc xác định là đáng ngờ.

## 7.5 Kết hợp nhiều mô hình

Khi những người khôn ngoan đưa ra những quyết định quan trọng, họ thường cân nhắc ý kiến của một số chuyên gia hơn là dựa vào sự đánh giá của chính họ hoặc của một cố vấn đáng tin cậy duy nhất. Ví dụ, trước khi chọn một hướng chính sách mới quan trọng, một nhà độc tài nhân từ sẽ tham khảo ý kiến rộng rãi: anh ta hoặc cô ta sẽ bị khuyên là mù quáng làm theo ý kiến của một chuyên gia. Trong một môi trường dân chủ, thảo luận về các quan điểm khác nhau có thể tạo ra sự đồng thuận; nếu không, một cuộc bỏ phiếu có thể được yêu cầu. Trong cả hai trường hợp, các ý kiến chuyên gia khác nhau đang được kết hợp.

Trong khai thác dữ liệu, một mô hình được tạo bởi máy học có thể được coi là một chuyên gia. Chuyên gia có lẽ là một từ quá mạnh!—tùy thuộc vào số lượng và chất lượng của dữ liệu đào tạo, và liệu thuật toán học có phù hợp với vấn đề hiện tại hay không, chuyên gia thực sự có thể là người thiếu hiểu biết—nhưng chúng tôi vẫn sử dụng thuật ngữ này. Một cách tiếp cận rõ ràng để đưa ra quyết định đáng tin cậy hơn là kết hợp đầu ra của các mô hình khác nhau. Một số máy

các kỹ thuật học tập thực hiện điều này bằng cách học một tập hợp các mô hình và sử dụng chúng kết hợp: nổi bật trong số này là các kế hoạch được gọi là đóng bao, tăng tốc và xếp chồng. Tất cả chúng đều có thể, thường xuyên hơn không, tăng hiệu suất dự đoán trên một mô hình duy nhất. Và chúng là những kỹ thuật chung có thể được áp dụng cho các bài toán dự đoán số và các nhiệm vụ phân loại.

Tính năng đóng gói, tăng tốc và xếp chồng chỉ mới được phát triển trong thập kỷ qua và hiệu suất của chúng thường tốt một cách đáng kinh ngạc. Các nhà nghiên cứu máy học đã phải vật lộn để hiểu tại sao. Và trong cuộc đấu tranh đó, các phương pháp mới đôi khi còn tốt hơn đã xuất hiện. Ví dụ: trong khi các ủy ban con người hiếm khi được hưởng lợi từ sự phiền nhiễu ồn ào, thì việc thay đổi cách đóng gói bằng cách thêm các biến thể ngẫu nhiên của bộ phân loại có thể cải thiện hiệu suất. Phân tích kỹ hơn cho thấy rằng việc tăng cường — có lẽ là phương pháp mạnh nhất trong ba phương pháp — có liên quan chặt chẽ với kỹ thuật thống kê đã được thiết lập của các mô hình phụ gia và nhận thức này đã dẫn đến các quy trình được cải tiến.

Các mô hình kết hợp này có chung nhược điểm là khó phân tích: chúng có thể bao gồm hàng chục hoặc thậm chí hàng trăm mô hình riêng lẻ và mặc dù chúng hoạt động tốt nhưng không dễ hiểu bằng trực giác những yếu tố nào đang góp phần cải thiện các quyết định. Trong vài năm gần đây, các phương pháp đã được phát triển để kết hợp lợi ích hoạt động của các ủy ban với các mô hình dễ hiểu. Một số sản xuất các mô hình cây quyết định tiêu chuẩn; những người khác giới thiệu các biến thể mới của cây cung cấp các đường dẫn tùy chọn.

Chúng tôi kết thúc bằng cách giới thiệu thêm một kỹ thuật kết hợp các mô hình sử dụng mã đầu ra sửa lỗi. Điều này chuyên biệt hơn ba kỹ thuật còn lại: nó chỉ áp dụng cho các bài toán phân loại, và thậm chí sau đó chỉ áp dụng cho những bài toán có nhiều hơn ba lớp.

### Đóng gói

Kết hợp các quyết định của các mô hình khác nhau có nghĩa là hợp nhất các kết quả đầu ra khác nhau thành một dự đoán duy nhất. Cách đơn giản nhất để làm điều này trong trường hợp phân loại là bỏ phiếu (có thể là phiếu có trọng số); trong trường hợp dự đoán số, đó là tính trung bình (có lẽ là trung bình có trọng số). Bagging và boost đều áp dụng cách tiếp cận này, nhưng chúng lấy được các mô hình riêng lẻ theo những cách khác nhau. Trong đóng gói, các mô hình nhận được trọng số bằng nhau, trong khi trong tăng cường, trọng số được sử dụng để tạo thêm ảnh hưởng cho những người thành công hơn—giống như một giám đốc điều hành có thể đặt các giá trị khác nhau theo lời khuyên của các chuyên gia khác nhau tùy thuộc vào mức độ kinh nghiệm của họ.

Để giới thiệu tính năng đóng gói, giả sử rằng một số tập dữ liệu huấn luyện có cùng kích thước được chọn ngẫu nhiên từ miền vấn đề. Hãy tưởng tượng sử dụng một kỹ thuật học máy cụ thể để xây dựng cây quyết định cho từng tập dữ liệu. Bạn có thể mong đợi những cây này giống hệt nhau trên thực tế và đưa ra cùng một dự đoán cho mỗi trường hợp thử nghiệm mới. Đáng ngạc nhiên, giả định này thường khá sai,

đặc biệt nếu tập dữ liệu huấn luyện khá nhỏ. Đây là một điều khá đáng lo ngại thực tế và dường như phủ bóng đen lên toàn bộ doanh nghiệp! lý do cho nó là cảm ứng cây quyết định đó (ít nhất, phương pháp từ trên xuống tiêu chuẩn được mô tả trong Chương 4) là một quá trình không ổn định: những thay đổi nhỏ đối với dữ liệu huấn luyện có thể dễ dàng dẫn đến một thuộc tính khác được chọn tại một nút cụ thể, với sự phân nhánh đáng kể cho cấu trúc của cây con bên dưới nút đó. Cái này

tự động ngụ ý rằng có những trường hợp thử nghiệm mà một số cây quyết định đưa ra dự đoán đúng còn những trường hợp khác thì không.

Quay trở lại phép loại suy về các chuyên gia trước đó, hãy coi các chuyên gia là cây quyết định cá nhân. Chúng ta có thể kết hợp các cây bằng cách để chúng biểu quyết trên mỗi ví dụ thử nghiệm. Nếu một lớp nhận được nhiều phiếu bầu hơn bất kỳ lớp nào khác, nó sẽ được coi là đúng một. Nói chung, càng đông càng vui: dự đoán bằng cách bỏ phiếu trở nên đáng tin cậy hơn khi có nhiều phiếu bầu hơn được tính đến. Quyết định hiếm khi xấu đi nếu tập huấn luyện mới được phát hiện, cây cối được xây dựng cho chúng và dự đoán cũng tham gia bỏ phiếu. Đặc biệt, bộ phân loại kết hợp sẽ hiếm khi kém chính xác hơn một cây quyết định được xây dựng từ chỉ một trong các bộ dữ liệu. (Tuy nhiên, sự cải thiện không được đảm bảo. Về mặt lý thuyết, có thể chỉ ra rằng các tình huống bệnh lý tồn tại trong đó các quyết định kết hợp là tệ hơn.)

Hiệu quả của việc kết hợp nhiều giả thuyết có thể được xem xét thông qua một thiết bị lý thuyết được gọi là phép phân rã sai lệch-phương sai. Giả sử rằng chúng ta có thể có vô số tập huấn luyện độc lập có cùng kích thước và mục đích sử dụng chúng để tạo ra vô số bộ phân loại. Một trường hợp thử nghiệm được xử lý bởi tất cả các bộ phân loại và một câu trả lời duy nhất được xác định theo đa số phiếu bầu. Trong tình huống lý tưởng hóa này, lỗi vẫn sẽ xảy ra vì không có kế hoạch học tập nào là hoàn hảo: tỷ lệ lỗi sẽ phụ thuộc vào mức độ phù hợp của phương pháp máy học với vấn đề trong tầm tay và cũng có ảnh hưởng của nhiễu trong dữ liệu, điều này không thể có thể được học. Giả sử tỷ lệ lỗi dự kiến được đánh giá bằng cách lấy trung bình lỗi của bộ phân loại kết hợp trên vô số mẫu thử nghiệm được chọn độc lập. Tỷ lệ lỗi cho một thuật toán học tập cụ thể là

được gọi là sự thiên vị của nó đối với vấn đề học tập và đo lường mức độ học tập tốt như thế nào phương pháp phù hợp với vấn đề. Định nghĩa kỹ thuật này là một cách định lượng khái niệm mơ hồ hơn về sự thiên vị đã được giới thiệu trong Phần 1.5: nó đo lường lỗi "liên tục" của thuật toán học không thể loại bỏ ngay cả khi thực hiện một số lượng vô hạn các tập huấn luyện được tính đến. Tất nhiên, nó không thể được tính toán chính xác trong các tình huống thực tế; nó chỉ có thể là gần đúng.

Nguồn lỗi thứ hai trong một mô hình đã học, trong một tình huống thực tế, bắt nguồn từ từ tập huấn luyện cụ thể được sử dụng, chắc chắn là hữu hạn và do đó không đại diện đầy đủ cho dân số thực tế của các trường hợp. Giá trị kỳ vọng của thành phần lỗi này, trên tất cả các tập huấn luyện có thể có với kích thước và tất cả các tập kiểm tra có thể, được gọi là phương sai của phương pháp học cho tập đó vấn đề. Tổng lỗi dự kiến của một bộ phân loại được tạo thành từ tổng sai lệch

và phương sai: đây là phép phân tích phương sai-độ lệch.4 Kết hợp nhiều bộ phân loại làm giảm lỗi dự kiến bằng cách giảm thành phần phương sai. Các càng nhiều phân loại được đưa vào thì phương sai càng giảm.

Tất nhiên, một khó khăn phát sinh khi áp dụng phương thức bỏ phiếu này: thường chỉ có một tập huấn luyện và việc thu thập thêm dữ liệu là không thể hoặc tốn kém.

Đóng gói cố gắng vô hiệu hóa sự không ổn định của các phương pháp học tập bằng cách mô phỏng quá trình được mô tả trước đó bằng cách sử dụng một tập huấn luyện nhất định. Thay vì lấy mẫu một tập dữ liệu đào tạo mới, độc lập mỗi lần, dữ liệu đào tạo ban đầu được thay đổi bằng cách xóa một số trường hợp và sao chép các trường hợp khác. Các phiên bản được chạy lấy mẫu ngẫu nhiên, có thay thế, từ tập dữ liệu gốc để tạo một tập dữ liệu mới có cùng kích thước. Quy trình lấy mẫu này chắc chắn lặp lại một số các trường hợp và xóa những trường hợp khác. Nếu ý tưởng này phù hợp, đó là bởi vì chúng tôi đã mô tả nó trong Chương 5 khi giải thích phương pháp bootstrap để ước tính lỗi tổng quát hóa của một phương pháp học tập (Phần 5.4): thật vậy, thuật ngữ đóng bao là viết tắt của bootstrap tổng hợp. Đóng gói áp dụng chương trình học tập—ví dụ, một công cụ cảm ứng cây quyết định—đối với mỗi một trong số những thứ được tạo ra một cách giả tạo này bộ dữ liệu và các trình phân loại được tạo từ chúng bỏ phiếu cho lớp được dự đoán. Thuật toán được tóm tắt trong Hình 7.7.

Sự khác biệt giữa đóng gói và quy trình lý tưởng hóa được mô tả trước đây là cách mà các tập dữ liệu đào tạo được lấy. Thay vì lấy các bộ dữ liệu độc lập từ miền, việc đóng gói chỉ lấy mẫu lại bản gốc dữ liệu huấn luyện. Các bộ dữ liệu được tạo bằng cách lấy mẫu lại khác với một khác nhưng chắc chắn không độc lập vì tất cả đều dựa trên một tập dữ liệu. Tuy nhiên, hóa ra việc đóng bao lại tạo ra một mô hình kết hợp thường hoạt động tốt hơn đáng kể so với mô hình đơn lẻ được xây dựng từ dữ liệu huấn luyện ban đầu và không bao giờ tệ hơn về cơ bản.

Đóng gói cũng có thể được áp dụng cho các phương pháp học để dự đoán số—ví dụ cây mẫu. Sự khác biệt duy nhất là, thay vì bỏ phiếu trên kết quả, các dự đoán riêng lẻ, là số thực, được tính trung bình. Các phân tích sai lệch phương sai cũng có thể được áp dụng cho dự đoán số bằng cách phân tách giá trị mong đợi của lỗi bình phương trung bình của các dự đoán trên dữ liệu mới. Xu hướng được định nghĩa là lỗi bình phương trung bình dự kiến khi tính trung bình trên các mô hình được xây dựng từ tất cả các tập dữ liệu huấn luyện có thể có cùng kích thước và phương sai là thành phần của lỗi dự kiến của một mô hình duy nhất do đến dữ liệu đào tạo cụ thể mà nó được xây dựng từ đó. Nó có thể được hiển thị về mặt lý thuyết luôn tính trung bình trên nhiều mô hình được xây dựng từ các tập huấn luyện độc lập

4 Đây là một phiên bản đơn giản hóa của toàn bộ câu chuyện. Một số phương pháp khác nhau để thực hiện các có thể tìm thấy phép phân tích sai lệch-phương sai trong tài liệu; không có cách làm thống nhất cái này.

```
thế hệ người mẫu
Gọi n là số trường hợp trong dữ liệu huấn luyện.
Đối với mỗi t lần lặp:
    n trường hợp mẫu với sự thay thế từ dữ liệu huấn luyện.
    Áp dụng thuật toán học cho mẫu.
    Lưu trữ mô hình kết quả.
phân loại
Đối với mỗi mô hình t:
    Dự đoán lớp thể hiện bằng mô hình.
Trả lại lớp đã được dự đoán thường xuyên nhất.
```

Hình 7.7 Thuật toán đóng bao.

làm giảm giá trị mong đợi của lỗi bình phương trung bình. (Như chúng tôi đã đề cập trước đó, kết quả tương tự không đúng với phân loại.)

Đóng gói với chi phí

Việc đóng gói giúp ích nhiều nhất nếu phương pháp học cơ bản không ổn định ở chỗ những thay đổi nhỏ trong dữ liệu đầu vào có thể dẫn đến các bộ phân loại khá khác nhau. Thật vậy, nó có thể giúp tăng tính đa dạng trong tập hợp các bộ phân loại bằng cách làm cho phương pháp học càng không ổn định càng tốt. Ví dụ: khi đóng gói các cây quyết định vốn đã không ổn định, hiệu suất tốt hơn thường đạt được bằng cách tắt tính năng cắt tỉa, điều này khiến chúng thậm chí còn không ổn định hơn. Một cải tiến khác có thể đạt được bằng cách thay đổi cách kết hợp các dự đoán để phân loại. Theo công thức ban đầu, đóng gói sử dụng bỏ phiếu. Nhưng khi các mô hình có thể đưa ra các ước tính xác suất và không chỉ phân loại đơn giản, thì việc tính trung bình các xác suất này thay vào đó sẽ có ý nghĩa trực quan. Điều này không chỉ thường cải thiện một chút khả năng phân loại, mà bộ phân loại đóng gói còn tạo ra các ước tính xác suất—những ước tính thường chính xác hơn so với ước tính do các mô hình riêng lẻ tạo ra. Việc triển khai đóng gói thường sử dụng phương pháp kết hợp các dự đoán này.

Trong Phần 5.7, chúng tôi đã chỉ ra cách làm cho một bộ phân loại nhạy cảm về chi phí bằng cách thu nhỏ chi phí dự đoán dự kiến. Ước tính xác suất chính xác là cần thiết vì chúng được sử dụng để thu được chi phí dự kiến của mỗi dự đoán.
Đóng gói là một ứng cử viên chính cho phân loại nhạy cảm với chi phí vì nó tạo ra các ước tính xác suất rất chính xác từ cây quyết định và các bộ phân loại mạnh nhưng không ổn định khác. Tuy nhiên, một nhược điểm là các bộ phân loại đóng gói rất khó phân tích.

Một phương pháp được gọi là MetaCost kết hợp các lợi ích dự đoán của việc đóng gói với một mô hình dễ hiểu để dự đoán nhạy cảm về chi phí. Nó xây dựng một bộ phân loại tập hợp bằng cách sử dụng đóng gói và sử dụng nó để dán nhãn lại dữ liệu đào tạo bằng cách đưa ra mọi

ví dụ đào tạo dự đoán giảm thiểu chi phí dự kiến, dựa trên ước tính xác suất thu được từ việc đóng bao. MetaCost sau đó loại bỏ các nhãn lớp ban đầu và tìm hiểu một trình phân loại mới duy nhất—ví dụ: một cây quyết định đã được cắt bớt—từ dữ liệu được gắn nhãn lại. Mô hình mới này tự động tính đến chi phí vì chúng đã được tích hợp vào nhãn lớp! Kết quả là một bộ phân loại nhạy cảm với chi phí duy nhất có thể được phân tích để xem các dự đoán được thực hiện như thế nào.

Ngoài kỹ thuật phân loại nhạy cảm với chi phí vừa được đề cập, Phần 5.7 cũng mô tả một phương pháp học nhạy cảm với chi phí để học một bộ phân loại nhạy cảm với chi phí bằng cách thay đổi tỷ lệ của từng lớp trong dữ liệu huấn luyện để phản ánh ma trận chi phí. MetaCost dường như tạo ra kết quả chính xác hơn phương pháp này, nhưng nó đòi hỏi nhiều tính toán hơn. Nếu không cần một mô hình dễ hiểu, thì bước xử lý hậu kỳ của MetaCost là không cần thiết: tốt hơn là sử dụng trực tiếp bộ phân loại đóng gói kết hợp với phương pháp chi phí dự kiến tối thiểu.

## ngẫu nhiên hóa

Tính năng đóng gói tạo ra một tập hợp các bộ phân loại đa dạng bằng cách đưa tính ngẫu nhiên vào đầu vào của thuật toán học tập, thường mang lại kết quả xuất sắc. Nhưng có nhiều cách khác để tạo ra sự đa dạng bằng cách giới thiệu ngẫu nhiên. Một số thuật toán học đã có sẵn thành phần ngẫu nhiên. Ví dụ: khi học các tri giác đa lớp sử dụng thuật toán lan truyền ngược (như được mô tả trong Phần 6.3), trọng số mạng được đặt thành các giá trị nhỏ được chọn ngẫu nhiên. Trình phân loại đã học phụ thuộc vào các số ngẫu nhiên vì thuật toán thuật toán có thể tìm thấy một cực tiểu cục bộ khác của hàm lỗi. Một cách để làm cho kết quả phân loại ổn định hơn là chạy máy học nhiều lần với các hạt số ngẫu nhiên khác nhau và kết hợp các dự đoán của bộ phân loại bằng cách bỏ phiếu hoặc tính trung bình.

Hầu hết mọi phương pháp học tập đều tuân theo một số kiểu ngẫu nhiên.
Hãy xem xét một thuật toán tham lam chọn tùy chọn tốt nhất ở mọi bước—chẳng hạn như một trình học cây quyết định chọn thuộc tính tốt nhất để phân tách tại mỗi nút.
Nó có thể được chọn ngẫu nhiên bằng cách chọn ngẫu nhiên một trong N tùy chọn tốt nhất thay vì một người chiến thắng duy nhất hoặc bằng cách chọn một tập hợp con ngẫu nhiên các tùy chọn và chọn tùy chọn tốt nhất từ đó. Tất nhiên, có một sự đánh đổi: tính ngẫu nhiên cao hơn sẽ tạo ra nhiều sự đa dạng hơn ở người học nhưng sử dụng ít dữ liệu hơn, có thể làm giảm độ chính xác của từng mô hình riêng lẻ. Liều lượng tốt nhất của sự ngẫu nhiên chỉ có thể được quy định bằng thực nghiệm.

Mặc dù đóng gói và sắp xếp ngẫu nhiên mang lại kết quả tương tự nhau, nhưng đôi khi sẽ đáng để kết hợp chúng vì chúng tạo ra tính ngẫu nhiên theo những cách khác nhau, có lẽ bổ sung cho nhau. Một thuật toán phổ biến để học các khu rừng ngẫu nhiên xây dựng một cây quyết định ngẫu nhiên trong mỗi lần lặp lại của thuật toán đóng bao và thường tạo ra các yếu tố dự đoán xuất sắc.

Ngẫu nhiên hóa đòi hỏi nhiều công việc hơn so với đóng gói vì thuật toán học tập phải được sửa đổi, nhưng nó có thể được áp dụng một cách có lợi cho nhiều loại hơn.
người học. Chúng tôi đã lưu ý trước đó rằng việc đóng gói không thành công với các thuật toán học tập ổn định có đầu ra không nhạy cảm với những thay đổi nhỏ trong đầu vào. Ví dụ, nó là vô nghĩa để đóng gói các bộ phân loại lân cận gần nhất vì đầu ra của chúng thay đổi rất rất ít nếu dữ liệu đào tạo bị nhiễu do lấy mẫu. Nhưng ngẫu nhiên hóa có thể được được áp dụng ngay cả với những người học ổn định: thủ thuật là ngẫu nhiên hóa theo cách khiến các bộ phân loại đa dạng mà không phải hy sinh quá nhiều hiệu suất. Dự đoán của bộ phân loại hàng xóm gần nhất phụ thuộc vào khoảng cách giữa các trường hợp,
do đó phụ thuộc rất nhiều vào thuộc tính nào được sử dụng để tính toán chúng,
vì vậy các bộ phân loại lân cận gần nhất có thể được chọn ngẫu nhiên bằng cách sử dụng các bộ phân loại ngẫu nhiên khác nhau tập con được chọn của các thuộc tính.

## tăng cường

Chúng tôi đã giải thích rằng việc đóng gói khai thác tính không ổn định vốn có trong các thuật toán học tập. Theo trực giác, việc kết hợp nhiều mô hình chỉ hữu ích khi các mô hình này khác biệt đáng kể với nhau và khi mỗi người xử lý một tỷ lệ phần trăm hợp lý của dữ liệu một cách chính xác. Lý tưởng nhất là các mô hình bổ sung cho một
khác, mỗi người là một chuyên gia trong một phần của lĩnh vực mà các mô hình khác không thực hiện tốt lắm—cũng giống như các giám đốc điều hành con người tìm kiếm những cố vấn có kỹ năng và kinh nghiệm bổ sung, thay vì trùng lặp, lẫn nhau.

Phương pháp thúc đẩy để kết hợp nhiều mô hình khai thác thông tin chi tiết này bằng cách tìm kiếm rõ ràng các mô hình bổ sung cho nhau. Đầu tiên, những điểm tương đồng:
như đóng gói, tăng cường sử dụng biểu quyết (để phân loại) hoặc tính trung bình (đối với số dự đoán) để kết hợp đầu ra của các mô hình riêng lẻ. Một lần nữa như đóng bao, nó kết hợp các mô hình cùng loại—ví dụ, cây quyết định. Tuy nhiên, boost ing là lặp đi lặp lại. Trong khi đóng gói các mô hình riêng lẻ được xây dựng riêng biệt, trong
thúc đẩy mỗi mô hình mới bị ảnh hưởng bởi hiệu suất của những mô hình được xây dựng trước đó. Tăng cường khuyến khích các mô hình mới trở thành chuyên gia cho các trường hợp được xử lý không chính xác bởi những người trước đó. Điểm khác biệt cuối cùng là việc tăng trọng lượng của người mẫu đóng góp bằng hiệu suất của nó thay vì đưa ra trọng số như nhau cho tất cả các mô hình.

Có nhiều biến thể về ý tưởng tăng cường. Chúng tôi mô tả một sử dụng rộng rãi phương pháp được gọi là AdaBoost.M1 được thiết kế đặc biệt để phân loại. Giống đóng gói, nó có thể được áp dụng cho bất kỳ thuật toán học phân loại nào. Để đơn giản hóa các vấn đề chúng tôi cho rằng thuật toán học có thể xử lý các trường hợp có trọng số, trong đó trọng số của một thể hiện là một số dương. (Chúng ta sẽ xem xét lại giả định này sau.) Sự hiện diện của trọng số đối tượng thay đổi cách tính toán sai số của bộ phân loại: nó là tổng trọng số của các đối tượng bị phân loại sai
chia cho tổng trọng số của tất cả các phiên bản, thay vì tỷ lệ của các phiên bản được phân loại sai. Bằng các trường hợp trọng số, thuật toán học tập có thể được buộc phải tập trung vào một tập hợp các trường hợp cụ thể, cụ thể là những trường hợp có tỷ lệ cao

```
thế hệ người mẫu

Chỉ định trọng lượng bằng nhau cho mỗi trường hợp đào tạo.
Đối với mỗi t lần lặp:
    Áp dụng thuật toán học tập cho tập dữ liệu có trọng số và lưu trữ
        mô hình kết quả.
    Tính toán lỗi e của mô hình trên tập dữ liệu có trọng số và lưu trữ lỗi.
    Nếu e bằng 0, hoặc e lớn hơn hoặc bằng 0,5:
        Chấm dứt tạo mô hình.
    Đối với mỗi trường hợp trong tập dữ liệu:
        Nếu phiên bản được phân loại chính xác theo mô hình:
            Nhân trọng số của thể hiện với e / (1 – e).
    Bình thường hóa trọng lượng của tất cả các trường hợp.

phân loại

Gán trọng số bằng 0 cho tất cả các lớp.
Đối với mỗi mô hình t (hoặc ít hơn):
    Thêm –log(e / (1 – e)) vào trọng số của lớp được mô hình dự đoán.
Trả lại lớp có trọng số cao nhất.
```

Hình 7.8 Thuật toán tăng tốc.

cân nặng. Những trường hợp như vậy trở nên đặc biệt quan trọng vì có động lực lớn hơn để phân loại chúng một cách chính xác. Thuật toán C4.5, được mô tả trong Phần 6.1, là một ví dụ về phương pháp học có thể điều chỉnh các trường hợp có trọng số mà không cần sửa đổi vì thuật toán này đã sử dụng khái niệm về các trường hợp phân số để xử lý các giá trị bị thiếu.

Thuật toán tăng tốc, được tóm tắt trong Hình 7.8, bắt đầu bằng cách gán trọng số bằng nhau cho tất cả các trường hợp trong dữ liệu huấn luyện. Sau đó, nó gọi thuật toán học tập để tạo thành một bộ phân loại cho dữ liệu này và tính lại trọng số của từng trường hợp theo đầu ra của bộ phân loại. Trọng lượng của các trường hợp được phân loại chính xác giảm xuống và trọng lượng của các trường hợp được phân loại sai tăng lên. Điều này tạo ra một tập hợp các phiên bản "dễ dàng" có trọng số thấp và một tập hợp các phiên bản "khó" có trọng số cao. Trong lần lặp lại tiếp theo—và tất cả các lần lặp tiếp theo—một trình phân loại được xây dựng cho dữ liệu được đánh trọng số lại, do đó tập trung vào việc phân loại chính xác các trường hợp cứng. Sau đó, trọng số của các cá thể được tăng hoặc giảm theo đầu ra của bộ phân loại mới này. Kết quả là, một số trường hợp khó có thể trở nên khó hơn và những trường hợp dễ hơn có thể trở nên dễ hơn; mặt khác, những trường hợp khó khăn khác có thể trở nên dễ dàng hơn, và những trường hợp dễ dàng hơn có thể trở nên khó khăn hơn—mọi khả năng đều có thể xảy ra trong thực tế. Sau mỗi lần lặp lại, các trọng số phản ánh tần suất các phiên bản đã bị phân loại sai bởi các bộ phân loại được tạo ra cho đến nay. Bằng cách duy trì thước đo "độ cứng" với mỗi phiên bản, quy trình này cung cấp một cách tinh tế để tạo ra một loạt các chuyên gia bổ sung cho nhau.

Trọng số nên được thay đổi bao nhiêu sau mỗi lần lặp? Câu trả lời phụ thuộc vào lỗi chung của bộ phân loại hiện tại. Cụ thể hơn, nếu e biểu thị lỗi của trình phân loại trên dữ liệu có trọng số (một phân số giữa 0 và 1), sau đó trọng số được cập nhật bởi

e 1 e ) ¨ ¥- ( trọng lượng /

đối với các trường hợp được phân loại chính xác và trọng số không thay đổi đối với các trường hợp được phân loại sai. Tất nhiên, điều này không làm tăng trọng lượng của việc phân loại sai. các trường hợp như đã tuyên bố trước đó. Tuy nhiên, sau khi tất cả các trọng số đã được cập nhật chúng được chuẩn hóa lại để tổng của chúng vẫn giữ nguyên như trước đây. Mỗi trọng số của cá thể được chia cho tổng các trọng số mới và nhân với tổng của cái cũ. Điều này sẽ tự động tăng trọng lượng của từng trường hợp được phân loại sai và giảm trọng số của từng trường hợp được phân loại đúng.

Bất cứ khi nào lỗi trên dữ liệu huấn luyện có trọng số vượt quá hoặc bằng 0,5, thủ tục tăng cường xóa bộ phân loại hiện tại và không thực hiện nữa lặp đi lặp lại. Điều tương tự cũng xảy ra khi lỗi bằng 0, bởi vì sau đó tất cả các trường hợp trọng số trở thành 0.

Chúng tôi đã giải thích cách phương pháp tăng cường tạo ra một loạt các bộ phân loại. Để hình thành dự đoán, đầu ra của chúng được kết hợp bằng cách sử dụng phiếu bầu có trọng số. Để xác định các trọng số, lưu ý rằng một bộ phân loại hoạt động tốt trên dữ liệu huấn luyện có trọng số mà nó được xây dựng từ đó (e gần bằng 0) sẽ nhận được trọng số cao và một bộ phân loại hoạt động kém (e gần bằng 0,5) sẽ nhận được mức thấp. Hơn đặc biệt,

$$\text{cân nặng} = \text{nhật ký } \frac{e}{1 - e},$$

là một số dương trong khoảng từ 0 đến vô cùng. Ngẫu nhiên, công thức này giải thích tại sao các bộ phân loại hoạt động hoàn hảo trên dữ liệu huấn luyện phải bị xóa, vì khi e bằng 0 thì trọng số không được xác định. Để đưa ra một dự đoán, các trọng số của tất cả các bộ phân loại bỏ phiếu cho một lớp cụ thể được tính tổng và lớp với tổng số lớn nhất được chọn.

Chúng tôi bắt đầu bằng cách giả định rằng thuật toán học có thể đối phó với trọng số trường hợp. Chúng tôi đã giải thích cách điều chỉnh các thuật toán học tập để đối phó với trọng số các trường hợp ở cuối Phần 6.5 trong Hồi quy tuyến tính có trọng số cục bộ. Thay vì thay đổi thuật toán học tập, có thể tạo ra một tập dữ liệu không có trọng số từ dữ liệu có trọng số bằng cách lấy mẫu lại—kỹ thuật tương tự mà túi sử dụng. Trong khi đối với việc đóng bao, mỗi phiên bản được chọn với khả năng xác suất như nhau, đối với các phiên bản tăng cường được chọn với xác suất tỷ lệ thuận với cân nặng. Kết quả là, các phiên bản có trọng số cao được sao chép thường xuyên và những phiên bản với trọng lượng thấp có thể không bao giờ được chọn. Khi tập dữ liệu mới trở nên lớn như bản gốc, nó được đưa vào phương pháp học thay vì trọng số dữ liệu. Nó đơn giản như vậy.

Một nhược điểm của quy trình này là một số trường hợp có trọng lượng thấp không biến nó thành tập dữ liệu được lấy mẫu lại, vì vậy thông tin sẽ bị mất trước khi học phương pháp được áp dụng. Tuy nhiên, điều này có thể được biến thành một lợi thế. Nếu phương pháp học tạo ra một bộ phân loại có lỗi vượt quá 0,5, thì việc tăng cường phải chấm dứt nếu dữ liệu trọng số được sử dụng trực tiếp, trong khi với việc lấy mẫu lại thì có thể là có thể tạo bộ phân loại có lỗi dưới 0,5 bằng cách loại bỏ tập dữ liệu đã lấy mẫu lại và tạo một bộ mới từ một hạt giống ngẫu nhiên khác. Thỉnh thoảng nhiều lần lặp lại tăng cường có thể được thực hiện bằng cách lấy mẫu lại so với khi sử dụng phiên bản có trọng số ban đầu của thuật toán.

Ý tưởng tăng tốc bắt nguồn từ một nhánh nghiên cứu máy học được gọi là lý thuyết học tính toán. Các nhà lý thuyết quan tâm đến việc tăng cường bởi vì có thể đạt được các đảm bảo về hiệu suất. Ví dụ, nó có thể được chỉ ra rằng lỗi của bộ phân loại kết hợp trên dữ liệu huấn luyện tiến tới 0 rất nhanh khi thực hiện nhiều lần lặp hơn (theo cấp số nhân nhanh về số lần lặp). Thật không may, như đã giải thích trong Phần 5.1, đảm bảo cho lỗi đào tạo không thú vị lắm vì chúng không nhất thiết chỉ ra hiệu suất tốt trên dữ liệu mới. Tuy nhiên, nó có thể được hiển thị về mặt lý thuyết, việc tăng cường chỉ thất bại trên dữ liệu mới nếu các bộ phân loại riêng lẻ quá "phức tạp" đối với lượng dữ liệu huấn luyện hiện có hoặc nếu lỗi huấn luyện của chúng trở nên quá lớn quá nhanh (theo nghĩa chính xác được giải thích bởi Schapire et al. 1997). Như thường lệ, vấn đề nằm ở việc tìm ra sự cân bằng phù hợp giữa độ phức tạp của các mô hình riêng lẻ và sự phù hợp của chúng với dữ liệu.

Nếu tăng cường thành công trong việc giảm lỗi trên dữ liệu thử nghiệm mới, nó thường làm như vậy một cách ngoạn mục. Một phát hiện rất đáng ngạc nhiên là việc thực hiện nhiều lần lặp tăng cường hơn có thể giảm lỗi trên dữ liệu mới rất lâu sau lỗi của bộ phân loại kết hợp trên dữ liệu đào tạo đã giảm xuống không. Các nhà nghiên cứu bối rối với kết quả này vì nó có vẻ mâu thuẫn với dao cạo của Occam, vốn tuyên bố rằng trong hai giả thuyết giải thích bằng chứng thực nghiệm tốt như nhau thì giả thuyết đơn giản hơn sẽ được ưu tiên hơn. Thực hiện nhiều lần lặp tăng cường hơn không giảm lỗi đào tạo không giải thích dữ liệu đào tạo tốt hơn, và nó chắc chắn làm tăng thêm độ phức tạp cho bộ phân loại kết hợp. May mắn thay, mâu thuẫn có thể được giải quyết bằng cách xem xét độ tin cậy của bộ phân loại đối với các dự đoán của nó. Độ tin cậy được đo lường bằng sự khác biệt giữa ước tính xác suất của lớp thực và của lớp dự đoán có khả năng nhất khác với lớp thực sự – một đại lượng được gọi là lề. Biên độ càng lớn thì càng nhiều tự tin rằng bộ phân loại đang dự đoán lớp thực. Hóa ra là thúc đẩy có thể tăng tỷ suất lợi nhuận trong thời gian dài sau khi lỗi đào tạo giảm xuống bằng không. Các hiệu ứng có thể được hình dung bằng cách vẽ sơ đồ phân phối tích lũy của lề giá trị của tất cả các phiên bản đào tạo cho số lần lặp tăng cường khác nhau, đưa ra một biểu đồ được gọi là đường cong lề. Do đó, nếu lời giải thích về bằng chứng thực nghiệm có tính đến lề, thì dao cạo của Occam vẫn sắc bén như bao giờ.

Điều thú vị về việc tăng cường là một bộ phân loại kết hợp mạnh mẽ có thể được xây dựng từ những cái rất đơn giản miễn là chúng đạt được ít hơn 50% lỗi trên dữ liệu đã được điều chỉnh lại. Thông thường, điều này dễ dàng—chắc chắn là đối với các vấn đề học tập với Hai lớp! Các phương pháp học tập đơn giản được gọi là người học yếu , và tăng cường chuyển đổi người học yếu thành người giỏi. Ví dụ, kết quả tốt cho hai lớp các vấn đề có thể đạt được bằng cách tăng các cây quyết định cực kỳ đơn giản có chỉ có một mức—được gọi là gốc quyết định. Một khả năng khác là áp dụng thúc đẩy đến một thuật toán học một quy tắc liên kết duy nhất—chẳng hạn như một đường dẫn duy nhất trong một cây quyết định—và phân loại các thể hiện dựa trên quy tắc có bao hàm hay không họ. Tất nhiên, các bộ dữ liệu nhiều lớp khiến việc đạt được tỷ lệ lỗi trở nên khó khăn hơn dưới 0,5. Cây quyết định vẫn có thể được tăng cường, nhưng chúng thường cần nhiều hơn phức tạp hơn các gốc quyết định. Các thuật toán phức tạp hơn đã được phát triển cho phép các mô hình rất đơn giản được tăng cường thành công trong các tình huống đa lớp.

Việc tăng cường thường tạo ra các bộ phân loại chính xác hơn đáng kể trên dữ liệu mới so với dữ liệu được tạo bằng cách đóng bao. Tuy nhiên, không giống như đóng gói, thúc đẩy đôi khi thất bại trong các tình huống thực tế: nó có thể tạo ra một bộ phân loại kém chính xác hơn đáng kể so với một bộ phân loại duy nhất được xây dựng từ cùng một dữ liệu. Điều này cho thấy rằng bộ phân loại kết hợp phù hợp với dữ liệu.

## Hồi quy cộng

Khi lần đầu tiên tăng cường được điều tra, nó đã gây ra sự quan tâm mạnh mẽ giữa các nhà nghiên cứu vì nó có thể thu hút thành tích hạng nhất từ những người học thờ ơ. Các nhà thống kê sớm phát hiện ra rằng nó có thể được viết lại như một thuật toán tham lam để phù hợp với một mô hình phụ gia. Các mô hình phụ gia có một lịch sử lâu dài trong thống kê. Nhìn chung, thuật ngữ này đề cập đến bất kỳ cách tạo dự đoán nào bằng cách tổng hợp đóng góp thu được từ các mô hình khác. Hầu hết các thuật toán học tập cho các mô hình bổ sung không xây dựng các mô hình cơ sở một cách độc lập nhưng đảm bảo rằng chúng bổ sung cho nhau và cố gắng tạo thành một tập hợp các mô hình cơ sở tối ưu hóa hiệu suất dự đoán theo một số tiêu chí cụ thể.

Tăng cường thực hiện mô hình phụ gia theo giai đoạn về phía trước. Lớp các thuật toán thuật toán này bắt đầu với một tập hợp trống và lần lượt kết hợp các thành viên mới. Ở mỗi giai đoạn, mô hình tối đa hóa hiệu suất dự đoán của toàn bộ quần thể được thêm vào mà không làm thay đổi những thứ đã có trong quần thể. Tối ưu hóa hiệu suất của bộ đồng phục ngụ ý rằng mô hình tiếp theo sẽ tập trung vào những trường hợp đào tạo mà nhóm hoạt động kém. Cái này chính xác là những gì việc tăng cường thực hiện bằng cách cho các trường hợp đó trọng số lớn hơn.

Đây là một phương pháp mô hình hóa phụ gia theo từng giai đoạn chuyển tiếp nổi tiếng cho dự đoán số Đầu tiên hãy xây dựng một mô hình hồi quy chuẩn, ví dụ, một cây hồi quy. Các lỗi mà nó thể hiện trên dữ liệu huấn luyện—sự khác biệt giữa các giá trị dự đoán và quan sát—được gọi là phần dư. Sau đó sửa cho

những lỗi này bằng cách học một mô hình thứ hai—có lẽ là một cây hồi quy khác—mô hình này cố gắng dự đoán các phần dư quan sát được. Để làm điều này, chỉ cần thay thế các giá trị lớp ban đầu bằng phần dư của chúng trước khi tìm hiểu mô hình thứ hai. Việc thêm các dự đoán của mô hình thứ hai vào các dự đoán của mô hình thứ nhất sẽ tự động mang lại lỗi thấp hơn trên dữ liệu huấn luyện. Thông thường, một số phần dư vẫn còn, vì mô hình thứ hai không phải là mô hình hoàn hảo, vì vậy chúng tôi tiếp tục với mô hình thứ ba học cách dự đoán phần dư của phần dư, v.v. Quy trình này không phù hợp với việc sử dụng các quy tắc có ngoại lệ để phân loại mà chúng ta đã gặp trong Phần 3.5.

Nếu các mô hình riêng lẻ giảm thiểu sai số bình phương của các dự đoán, giống như các mô hình hồi quy tuyến tính, thì thuật toán này sẽ giảm thiểu sai số bình phương của toàn bộ tập hợp. Trong thực tế, nó cũng hoạt động tốt khi bộ học cơ sở sử dụng xấp xỉ heuristic thay thế, chẳng hạn như bộ học hồi quy và cây mô hình được mô tả trong Phần 6.5. Trên thực tế, không có ích gì khi sử dụng hồi quy tuyến tính tiêu chuẩn làm trình học cơ sở cho hồi quy cộng, bởi vì tổng của các mô hình hồi quy tuyến tính lại là một mô hình hồi quy tuyến tính và bản thân thuật toán hồi quy sẽ giảm thiểu sai số bình phương. Tuy nhiên, đó là một câu chuyện khác nếu người học cơ sở là một mô hình hồi quy dựa trên một thuộc tính duy nhất, một thuộc tính giảm thiểu lỗi bình phương. Các nhà thống kê gọi đây là **hồi quy tuyến tính** đơn giản , trái ngược với phương pháp đa thuộc tính tiêu chuẩn, được gọi đúng là **hồi quy tuyến tính** bội . Trên thực tế, sử dụng hồi quy cộng kết hợp với hồi quy tuyến tính đơn giản và lặp lại cho đến khi sai số bình phương của tập hợp giảm không còn tạo ra một mô hình cộng giống hệt với hàm hồi quy tuyến tính bội bình phương nhỏ nhất.

Hồi quy cộng theo từng giai đoạn chuyển tiếp dễ bị quá khớp vì mỗi mô hình được thêm vào phù hợp hơn với dữ liệu đào tạo. Để quyết định thời điểm dừng, hãy sử dụng xác thực chéo. Ví dụ: thực hiện xác thực chéo cho mọi số lần lặp lại cho đến mức tối đa do người dùng chỉ định và chọn một bước giảm thiểu ước tính sai số bình phương được xác thực chéo. Đây là một tiêu chí dừng tốt vì xác thực chéo mang lại ước tính khá đáng tin cậy về lỗi trên dữ liệu trong tương lai. Ngẫu nhiên, sử dụng phương pháp này kết hợp với hồi quy tuyến tính đơn giản vì trình học cơ sở kết hợp hiệu quả nhiều hồi quy tuyến tính với lựa chọn thuộc tính tích hợp, bởi vì đóng góp của thuộc tính quan trọng nhất tiếp theo chỉ được đưa vào nếu nó làm giảm lỗi xác thực chéo.

Để thuận tiện cho việc triển khai, hồi quy cộng theo từng giai đoạn chuyển tiếp thường bắt đầu bằng mô hình cấp 0 chỉ dự đoán giá trị trung bình của lớp trên dữ liệu huấn luyện để mọi mô hình tiếp theo phù hợp với phần dư. Điều này gợi ý một khả năng khác để ngăn chặn trang bị thừa: thay vì trừ toàn bộ dự đoán của mô hình để tạo giá trị đích cho mô hình tiếp theo, hãy thu nhỏ các dự đoán bằng cách nhân chúng với hệ số hằng số do người dùng chỉ định trong khoảng từ 0 đến 1 trước khi trừ. Điều này làm giảm sự phù hợp của mô hình với phần dư và do đó làm giảm khả năng trang bị quá mức. Tất nhiên, nó có thể tăng số lượng

lặp đi lặp lại cần thiết để đi đến một mô hình phụ gia tốt. Giảm số nhân
làm giảm hiệu quả quá trình học tập, tăng cơ hội dừng lại
vào đúng thời điểm—mà còn tăng thời gian chạy.

## Hồi quy logistic phụ gia

Hồi quy bổ sung cũng có thể được áp dụng để phân loại giống như hồi quy tuyến tính
Có thể. Nhưng chúng ta đã biết từ Phần 4.6 rằng hồi quy logistic tốt hơn tuyến tính
hồi quy để phân loại. Nó chỉ ra rằng một sự thích ứng tương tự có thể được thực hiện
sang các mô hình phụ gia bằng cách sửa đổi phương pháp mô hình hóa từng giai đoạn chuyển tiếp thành
thực hiện **hồi quy logistic bổ sung** . Sử dụng biến đổi logit để dịch
vấn đề ước tính xác suất thành một vấn đề hồi quy, như chúng ta đã làm trong Phần
4.6 và giải quyết nhiệm vụ hồi quy bằng cách sử dụng một tập hợp các mô hình—ví dụ:
cây hồi quy—giống như đối với hồi quy cộng. Ở mỗi giai đoạn, thêm mô hình
điều đó tối đa hóa xác suất của dữ liệu được cung cấp cho bộ phân loại tập hợp.

Giả sử fj là mô hình hồi quy thứ j trong tập hợp và fj(a) là dự đoán của nó đối với ví dụ a.
Giả sử bài toán hai lớp, sử dụng mô hình cộng Sfj(a)
để có được ước tính xác suất cho lớp đầu tiên:

$$P(1 \mid a) = \frac{1}{1 + e^{-\sum f_j(a)}}$$

Điều này gần giống với biểu thức được sử dụng trong Phần 4.6 (trang 121), ngoại trừ việc
ở đây nó được viết tắt bằng cách sử dụng ký hiệu véc-tơ cho trường hợp a và tổng trọng số ban đầu
của các giá trị thuộc tính được thay thế bằng tổng của các giá trị phức tạp tùy ý.
các mô hình hồi quy f.

Hình 7.9 cho thấy phiên bản hai lớp của **thuật toán** LogitBoost , thuật toán này tạo thành hồi
quy logistic bổ sung và tạo ra các mô hình riêng lẻ fj. Đây,
yi là 1 cho một thể hiện trong lớp thứ nhất và 0 cho một thể hiện trong lớp thứ hai. TRONG
mỗi lần lặp lại thuật toán này phù hợp với mô hình hồi quy fj với phiên bản có trọng số của

```
thế hệ người mẫu
Đối với các lần lặp j = 1 đến t:
      Đối với mỗi trường hợp a[i]:
            Đặt giá trị mục tiêu cho hồi quy thành
                  z[i] = (y[i] – p(1 | a[i])) / [p(1 | a[i])                    (1 – p(1 | a[i])]
            Đặt trọng số của thể hiện a[i] thành p(1 | a[i])                    (1 – p(1 | a[i])
      Khớp mô hình hồi quy f[j] với dữ liệu có giá trị lớp z[i] và trọng số w[i].
phân loại
Dự đoán hạng nhất nếu p(1 | a) > 0,5, ngược lại dự đoán hạng hai.
```

Hình 7.9 Thuật toán hồi quy logistic cộng.

tập dữ liệu gốc dựa trên các giá trị lớp giả zi và trọng số wi. Chúng tôi giả định rằng p(1 | a) được tính toán bằng cách sử dụng fj đã được xây dựng trong các lần lặp lại trước đó.

Nguồn gốc của thuật toán này nằm ngoài phạm vi của cuốn sách này, nhưng có thể chỉ ra rằng thuật toán tối đa hóa xác suất của dữ liệu đối với tập hợp nếu mỗi mô hình fj được xác định bằng cách giảm thiểu sai số bình phương trên bài toán hồi quy tương ứng. Trên thực tế, nếu hồi quy tuyến tính bội được sử dụng để tạo thành fj, thì thuật toán sẽ hội tụ về mô hình hồi quy logistic tuyến tính có khả năng lớn nhất: đó là hiện thân của phương pháp bình phương nhỏ nhất có trọng số lặp lại được đề cập trong Phần 4.6.

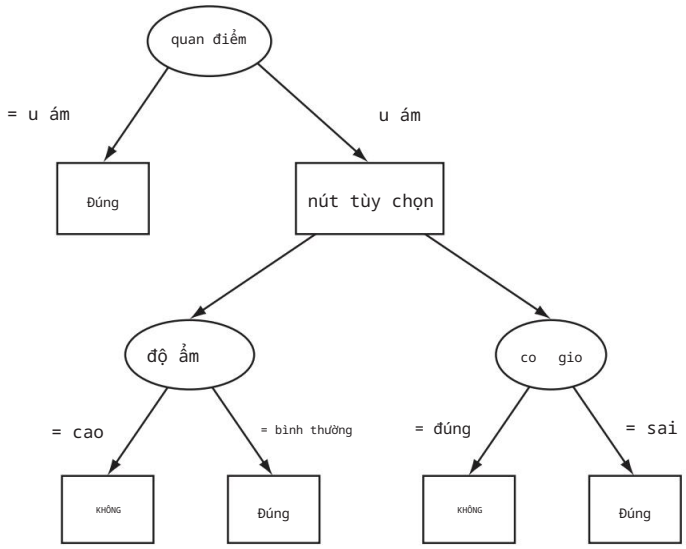
Nhìn bề ngoài, LogitBoost trông khá khác so với AdaBoost, nhưng các yếu tố dự đoán mà chúng tạo ra khác nhau chủ yếu ở chỗ cái trước tối ưu hóa khả năng xảy ra trực tiếp trong khi cái sau tối ưu hóa hàm mất mát theo cấp số nhân có thể được coi là gần đúng với nó. Từ góc độ thực tế, sự khác biệt là LogitBoost sử dụng phương pháp hồi quy làm trình học cơ sở trong khi AdaBoost hoạt động với các thuật toán phân loại.

Chúng tôi chỉ trình bày phiên bản hai lớp của LogitBoost, nhưng thuật toán có thể được khái quát hóa cho các bài toán nhiều lớp. Cũng giống như hồi quy cộng, nguy cơ khớp quá mức có thể giảm bằng cách thu nhỏ các dự đoán của fj riêng lẻ bằng một hệ số nhân xác định trước và sử dụng xác thực chéo để xác định số lần lặp thích hợp.

### Cây tùy chọn

Đóng gói, tăng cường và ngẫu nhiên hóa tất cả đều tạo ra một tập hợp các bộ phân loại. Điều này gây khó khăn cho việc phân tích loại thông tin nào đã được trích xuất từ dữ liệu. Sẽ thật tuyệt nếu có một mô hình duy nhất có cùng hiệu suất dự đoán. Một khả năng là tạo tập dữ liệu nhân tạo, bằng cách lấy mẫu ngẫu nhiên các điểm từ không gian cá thể và gán cho chúng các nhãn lớp được dự đoán bởi bộ phân loại tập hợp, sau đó tìm hiểu cây quyết định hoặc bộ quy tắc từ tập dữ liệu mới này. Để có được hiệu suất dự đoán tương tự từ cây cũng như từ tập hợp, có thể cần có một bộ dữ liệu khổng lồ, nhưng trong giới hạn, chiến lược này sẽ có thể sao chép hiệu suất của trình phân loại tập hợp – và nó chắc chắn sẽ như vậy nếu bản thân tập hợp bao gồm các cây quyết định.

Một cách tiếp cận khác là rút ra một cấu trúc duy nhất có thể đại diện cho một tập hợp các bộ phân loại một cách cô đọng. Điều này có thể được thực hiện nếu quần thể bao gồm các cây quyết định; kết quả được gọi là cây tùy chọn. Cây tùy chọn khác với cây quyết định ở chỗ chúng chứa hai loại nút: nút quyết định và nút tùy chọn.
Hình 7.10 cho thấy một ví dụ đơn giản về dữ liệu thời tiết, chỉ với một nút tùy chọn. Để phân loại một phiên bản, hãy lọc nó xuống thông qua cây. Tại một nút quyết định, chỉ lấy một trong các nhánh, như thường lệ, nhưng tại một nút tùy chọn, lấy tất cả các nhánh. Điều này có nghĩa là thể hiện kết thúc trong nhiều hơn một lá và các phân loại thu được từ những lá đó bằng cách nào đó phải được kết hợp thành một

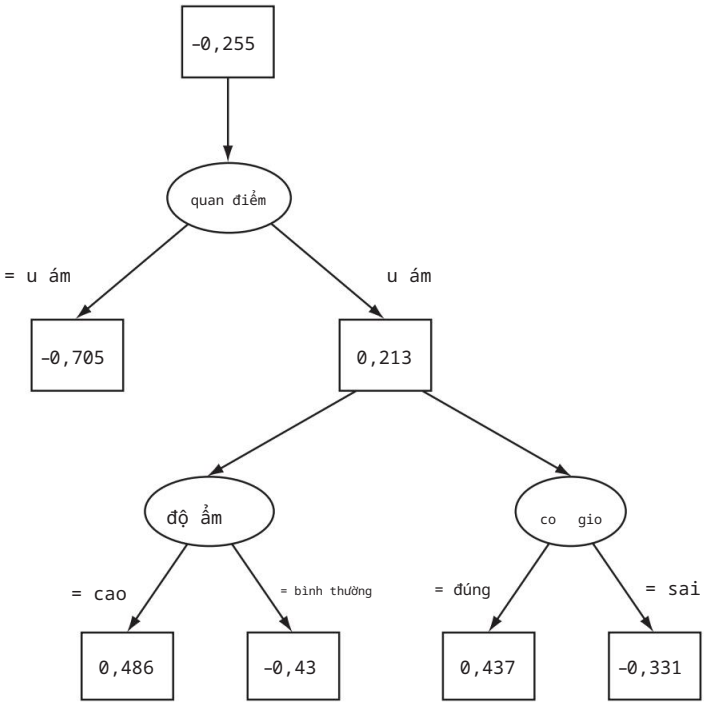



Hình 7.10 Cây tùy chọn đơn giản cho dữ liệu thời tiết.

phân loại tổng thể. Điều này có thể được thực hiện đơn giản bằng cách biểu quyết, chiếm đa số bỏ phiếu tại một nút tùy chọn là dự đoán của nút. Trong trường hợp đó nó làm cho không có ý nghĩa gì khi có các nút tùy chọn chỉ có hai tùy chọn (như trong Hình 7.10) bởi vì sẽ chỉ có đa số nếu cả hai nhánh đồng ý. Một khả năng khác là lấy trung bình các ước tính xác suất thu được từ các đường dẫn khác nhau, sử dụng trung bình không trọng số hoặc Bayesian phức tạp hơn.
tiếp cận.

Cây tùy chọn có thể được tạo bằng cách sửa đổi trình học cây quyết định hiện có để tạo một nút tùy chọn nếu có một số phần tách có vẻ hữu ích tương tự theo mức thu được thông tin của họ. Tất cả các lựa chọn trong một phạm vi nhất định do người dùng chỉ định khả năng chịu đựng của cái tốt nhất có thể được thực hiện thành các tùy chọn. Trong quá trình cắt tỉa, lỗi của một nút tùy chọn là lỗi trung bình của các tùy chọn của nó.

Một khả năng khác là phát triển một cây tùy chọn bằng cách tăng dần các nút với nó. Điều này thường được thực hiện bằng cách sử dụng thuật toán tăng cường và các cây kết quả thường được gọi là cây quyết định xen kẽ thay vì cây tùy chọn. Trong ngữ cảnh này các nút quyết định được gọi là các nút chia tách và các nút tùy chọn được gọi là các nút tiền định hướng. Các nút dự đoán là các lá nếu không có nút bộ chia nào được thêm vào cho họ chưa. Cây quyết định xen kẽ tiêu chuẩn áp dụng cho các bài toán hai lớp và với mỗi nút dự đoán được liên kết với một số dương hoặc âm.
giá trị. Để có được một dự đoán cho một trường hợp, hãy lọc nó xuống tất cả các trường hợp áp dụng các nhánh và tổng hợp các giá trị từ bất kỳ nút dự đoán nào gặp phải; dự đoán lớp này hay lớp khác tùy thuộc vào tổng có dương hay không
hoặc tiêu cực.




Hình 7.11 Cây quyết định thay thế cho dữ liệu thời tiết.

Một cây ví dụ đơn giản cho dữ liệu thời tiết được thể hiện trong Hình 7.11, trong đó một giá trị dương tương ứng với lớp chơi = không và giá trị âm tương ứng với lớp chơi = có. Để phân loại một phiên bản với triển vọng = nắng, nhiệt độ = nóng, độ ẩm = bình thường và có gió = sai, lọc nó xuống các lá tương ứng, thu được các giá trị -0,255, 0,213, -0,430 và -0,331. Tổng của các giá trị này là âm; do đó dự đoán chơi = có. Các cây quyết định luân phiên luôn có một nút dự đoán ở gốc, như trong ví dụ này.

Cây xen kẽ được trồng bằng cách sử dụng thuật toán tăng cường—ví dụ: thuật toán tăng cường sử dụng một trình học cơ sở để dự đoán số, chẳng hạn như phương pháp LogitBoost đã mô tả trước đây. Giả sử rằng người học cơ sở tạo ra một quy tắc liên hợp duy nhất trong mỗi lần lặp tăng cường. Sau đó luân phiên cây quyết định có thể được tạo bằng cách thêm từng quy tắc vào cây. Các điểm số liên quan đến các nút dự đoán được lấy từ các quy tắc. Tuy nhiên, cây kết quả sẽ phát triển lớn rất nhanh vì các quy tắc từ các lần lặp tăng cường khác nhau có khả năng khác nhau. Do đó, thuật toán học đối với các cây quyết định xen kẽ, chỉ xem xét các luật mở rộng một trong các các đường dẫn hiện có trong cây bằng cách thêm một nút bộ chia và hai nút định hướng trước tương ứng (giả sử phân tách nhị phân). Trong phiên bản tiêu chuẩn của thuật toán,

mọi vị trí có thể có trong cây đều được xem xét để bổ sung và một nút được thêm vào theo thước đo hiệu suất phụ thuộc vào thuật toán tăng tốc cụ thể được sử dụng. Tuy nhiên, heuristic có thể được sử dụng thay vì tìm kiếm toàn diện để tăng tốc quá trình học tập.

Cây mô hình logistic Cây

tùy chọn và cây xen kẽ mang lại hiệu suất phân loại rất tốt dựa trên một cấu trúc duy nhất, nhưng chúng vẫn có thể khó diễn giải khi có nhiều nút tùy chọn vì khó có thể biết cách bắt nguồn từ một dự đoán cụ thể. Tuy nhiên, hóa ra việc tăng cường cũng có thể được sử dụng để xây dựng các cây quyết định rất hiệu quả mà không bao gồm bất kỳ tùy chọn nào. Ví dụ: thuật toán LogitBoost đã được sử dụng để tạo ra các cây có mô hình hồi quy logistic tuyến tính ở các lá. Chúng được gọi là cây mô hình logistic và được diễn giải giống như cây mô hình cho hồi quy được mô tả trong Phần 6.5.

LogitBoost thực hiện hồi quy logistic phụ gia. Giả sử rằng mỗi lần lặp lại của thuật toán tăng tốc phù hợp với một hàm hồi quy đơn giản bằng cách duyệt qua tất cả các thuộc tính, tìm hàm hồi quy đơn giản có lỗi nhỏ nhất và thêm nó vào mô hình cộng. Nếu chạy thuật toán LogitBoost cho đến khi hội tụ, thì kết quả là mô hình hồi quy đa logistic có khả năng xảy ra tối đa. Tuy nhiên, để có hiệu suất tối ưu đối với dữ liệu trong tương lai, thường không cần thiết phải chờ đợi sự hội tụ—và làm như vậy thường gây bất lợi. Có thể xác định số lần lặp tăng cường thích hợp bằng cách ước tính hiệu suất dự kiến cho một số lần lặp nhất định bằng cách sử dụng xác thực chéo và dừng quá trình khi hiệu suất ngừng tăng.

Một phần mở rộng đơn giản của thuật toán này dẫn đến cây mô hình hậu cần. Quá trình tăng cường kết thúc khi không còn cấu trúc nào trong dữ liệu có thể được mô hình hóa bằng hàm hồi quy logistic tuyến tính. Tuy nhiên, vẫn có thể có một cấu trúc mà các mô hình tuyến tính có thể phù hợp nếu sự chú ý bị hạn chế đối với các tập hợp con của dữ liệu, chẳng hạn như thu được bằng một tiêu chí cây quyết định tiêu chuẩn, chẳng hạn như thu được thông tin. Sau đó, khi không thể cải thiện thêm nữa bằng cách thêm các mô hình tuyến tính đơn giản hơn, dữ liệu sẽ được phân tách và quá trình tăng cường được tiếp tục một cách riêng biệt trong mỗi tập hợp con. Quá trình này lấy mô hình logistic được tạo cho đến nay và tinh chỉnh nó một cách riêng biệt cho dữ liệu trong mỗi tập hợp con. Một lần nữa, xác thực chéo được chạy trong mỗi tập hợp con để xác định số lần lặp thích hợp để thực hiện trong tập hợp con đó.

Quá trình này được áp dụng một cách đệ quy cho đến khi các tập con trở nên quá nhỏ. Cây kết quả chắc chắn sẽ phù hợp với dữ liệu đào tạo và một trong những phương pháp học cây quyết định tiêu chuẩn có thể được sử dụng để cắt bớt nó. Các thí nghiệm chỉ ra rằng hoạt động cắt tỉa là rất quan trọng. Sử dụng chiến lược chọn kích thước cây phù hợp bằng xác thực chéo, thuật toán tạo ra những cây nhỏ nhưng rất chính xác với các mô hình logistic tuyến tính ở các lá.

## Xếp chồng

Khái quát hóa xếp chồng, hay gọi tắt là xếp chồng, là một cách khác để kết hợp các mô hình nhiều mô hình. Mặc dù đã được phát triển cách đây vài năm nhưng nó ít được sử dụng rộng rãi hơn so với đóng gói và tăng cường, một phần vì khó phân tích về mặt lý thuyết và một phần vì không có cách làm tốt nhất được chấp nhận rộng rãi—ý tưởng cơ bản có thể được áp dụng trong nhiều biến thể khác nhau.

Không giống như đóng gói và tăng tốc, xếp chồng thường không được sử dụng để kết hợp các mô hình cùng loại—ví dụ, một tập hợp các cây quyết định. thay vào đó là áp dụng cho các mô hình được xây dựng bởi các thuật toán học tập khác nhau. Giả sử bạn có một trình tạo cây quyết định, trình học Naïve Bayes và học tập dựa trên cá thể và bạn muốn tạo một trình phân loại cho một tập dữ liệu nhất định. Quy trình thông thường sẽ là ước tính lỗi dự kiến của từng thuật toán bằng cách xác thực chéo và chọn thuật toán tốt nhất để tạo thành mô hình dự đoán trong tương lai.
dữ liệu. Nhưng không phải là có một cách tốt hơn? Với ba thuật toán học có sẵn, không thể chúng tôi sử dụng cả ba để dự đoán và kết hợp các kết quả đầu ra với nhau?

Một cách để kết hợp các đầu ra là bỏ phiếu—cơ chế tương tự được sử dụng trong đóng bao. Tuy nhiên, bỏ phiếu (không có trọng số) chỉ có ý nghĩa nếu việc học các chương trình thực hiện tương đối tốt. Nếu hai trong số ba bộ phân loại đưa ra những dự đoán hoàn toàn không chính xác, chúng ta sẽ gặp rắc rối! Thay vào đó, giới thiệu xếp chồng lên nhau tạo ra khái niệm về một người kiếm tiền, thay thế cho thủ tục bỏ phiếu. Các vấn đề với việc bỏ phiếu là không rõ nên tin tưởng bộ phân loại nào. cố gắng xếp chồng để tìm hiểu bộ phân loại nào là bộ phân loại đáng tin cậy, sử dụng một thuật toán học tập khác—the metalearner—để khám phá cách tốt nhất để kết hợp đầu ra của người học cơ sở.

Đầu vào của siêu mô hình - còn được gọi là mô hình cấp 1 - là những dự đoán của các mô hình cơ sở hoặc mô hình cấp 0. Một thể hiện cấp 1 có nhiều thuộc tính bằng số học viên cấp 0 và các giá trị thuộc tính đưa ra dự đoán về những người học này trên phiên bản cấp 0 tương ứng. Khi người học xếp chồng lên nhau được sử dụng để phân loại, một phiên bản đầu tiên được đưa vào các mô hình cấp 0 và mỗi người ta đoán một giá trị lớp. Những phỏng đoán này được đưa vào mô hình cấp 1, mô hình này kết hợp chúng thành dự đoán cuối cùng.

Vẫn còn vấn đề đào tạo người học cấp 1. Để làm được điều này, chúng ta cần để tìm cách biến đổi dữ liệu huấn luyện cấp 0 (được sử dụng để huấn luyện người học cấp 0) vào dữ liệu huấn luyện cấp 1 (dùng để huấn luyện người học cấp 1). Điều này có vẻ đơn giản: hãy để mỗi mô hình cấp 0 phân loại một phiên bản đào tạo, và đính kèm với dự đoán của họ giá trị lớp thực tế của cá thể để tạo ra một cá thể đào tạo cấp 1. Thật không may, điều này không hoạt động tốt. Nó sẽ cho phép các quy tắc được học chẳng hạn như luôn tin vào đầu ra của bộ phân loại A và bỏ qua B và C. Quy tắc này có thể phù hợp với các phân loại cơ sở cụ thể A, B và C; nếu vậy, nó có thể sẽ được học. Nhưng chỉ vì nó có vẻ thích hợp trên dữ liệu huấn luyện không nhất thiết có nghĩa là nó sẽ hoạt động tốt trên dữ liệu thử nghiệm—

bởi vì nó chắc chắn sẽ học cách ưu tiên các bộ phân loại phù hợp hơn với dữ liệu đào tạo hơn những người đưa ra quyết định thực tế hơn.

Do đó, xếp chồng không chỉ đơn giản là chuyển đổi dữ liệu đào tạo cấp 0 vào dữ liệu cấp 1 theo cách này. Nhớ lại từ Chương 5 rằng có tốt hơn phương pháp ước tính hiệu suất của bộ phân loại hơn là sử dụng lỗi trên tập huấn luyện. Một là giữ lại một số trường hợp và sử dụng chúng để đánh giá độc lập. Áp dụng điều này để xếp chồng, chúng tôi dành một số trường hợp để tạo thành dữ liệu đào tạo cho người học cấp 1 và xây dựng bộ phân loại cấp 0 từ dữ liệu còn lại. Khi các bộ phân loại cấp 0 đã được xây dựng, chúng được sử dụng để phân loại các phiên bản trong tập hợp giữ lại, tạo thành dữ liệu đào tạo cấp 1 như được mô tả trước đó. Bởi vì bộ phân loại cấp 0 chưa được đào tạo về những trường hợp này, dự đoán của họ là không thiên vị; do đó, dữ liệu đào tạo cấp 1 chính xác phản ánh hiệu suất thực sự của các thuật toán học cấp 0. Khi dữ liệu cấp 1 đã được tạo bởi thủ tục giữ lại này, người học cấp 0 có thể được được áp dụng lại để tạo các bộ phân loại từ tập huấn luyện đầy đủ, làm cho tốt hơn một chút sử dụng dữ liệu và dẫn đến dự đoán tốt hơn.

Phương thức holdout chắc chắn làm mất đi một số đặc điểm của mô hình cấp 1. dữ liệu huấn luyện. Trong Chương 5, xác thực chéo đã được giới thiệu như một phương tiện để giải quyết vấn đề này đối với việc ước lượng lỗi. Điều này có thể được áp dụng cùng với xếp chồng bằng cách thực hiện xác thực chéo cho mọi người học cấp 0. Mỗi trường hợp trong dữ liệu đào tạo xảy ra ở chính xác một trong các nếp gấp thử nghiệm của xác thực chéo và các dự đoán của trình tạo cảm ứng cấp 0 được tạo từ nếp gấp đào tạo phản hồi tương ứng được sử dụng để xây dựng phiên bản đào tạo cấp 1 từ nếp gấp đó. Điều này tạo ra một phiên bản đào tạo cấp 1 cho mỗi phiên bản đào tạo cấp 0. Của Tất nhiên, nó chậm vì bộ phân loại cấp 0 phải được đào tạo cho mỗi lần gấp của xác thực chéo, nhưng nó cho phép trình phân loại cấp 1 tận dụng tối đa dữ liệu huấn luyện.

Đưa ra một ví dụ thử nghiệm, hầu hết các phương pháp học đều có thể đưa ra xác suất cho mọi nhãn lớp thay vì đưa ra một dự đoán phân loại duy nhất. cái này có thể được khai thác để cải thiện hiệu suất xếp chồng bằng cách sử dụng xác suất để tạo thành dữ liệu cấp 1. Sự khác biệt duy nhất đối với quy trình tiêu chuẩn là mỗi thuộc tính cấp 1 danh nghĩa—đại diện cho lớp được dự đoán bởi thuộc tính cấp 0 learner—được thay thế bằng một số thuộc tính số, mỗi thuộc tính đại diện cho một lớp đầu ra xác suất của người học cấp 0. Nói cách khác, số thuộc tính trong dữ liệu mức 1 được nhân với số lớp. thủ tục này có lợi thế là người học cấp 1 có được sự tự tin rằng mỗi người học ở cấp độ 0 liên kết với các dự đoán của nó, do đó khuếch đại giao tiếp giữa hai cấp độ học tập.

Một câu hỏi nổi bật vẫn là: thuật toán nào phù hợp với người học cấp 1? Về nguyên tắc, bất kỳ kế hoạch học tập nào cũng có thể được áp dụng. Tuy nhiên, vì hầu hết công việc đã được thực hiện bởi những người học ở cấp độ 0, trình phân loại cấp độ 1 là về cơ bản chỉ là một trọng tài và thật hợp lý khi chọn một thuật toán khá đơn giản

vì mục đích này. Theo lời của David Wolpert, người phát minh ra xếp chồng, điều hợp lý là các công cụ khái quát hóa cấp 1 "tương đối toàn cầu, mượt mà" sẽ hoạt động tốt. Các mô hình tuyến tính đơn giản hoặc cây có mô hình tuyến tính ở lá thường hoạt động tốt.

Xếp chồng cũng có thể được áp dụng cho dự đoán số. Trong trường hợp đó, các mô hình cấp 0 và mô hình cấp 1 đều dự đoán các giá trị số. Cơ chế cơ bản vẫn giữ nguyên; sự khác biệt duy nhất nằm ở bản chất của dữ liệu cấp 1. Trong trường hợp số, mỗi thuộc tính cấp 1 đại diện cho dự đoán số được thực hiện bởi một trong các mô hình cấp 0 và thay vì giá trị lớp, giá trị đích số được đính kèm với các phiên bản đào tạo cấp 1.

Mã đầu ra sửa lỗi Mã đầu ra sửa

lỗi là một kỹ thuật để cải thiện hiệu suất của thuật toán phân loại trong các bài toán học nhiều lớp. Nhớ lại từ Chương 6 rằng một số thuật toán học tập—ví dụ, máy vectơ hỗ trợ tiêu chuẩn—chỉ hoạt động với các bài toán hai lớp. Để áp dụng các thuật toán như vậy cho các bộ dữ liệu nhiều lớp, bộ dữ liệu được phân tách thành một số bài toán hai lớp độc lập, thuật toán được chạy trên từng lớp và kết quả đầu ra của các bộ phân loại kết quả được kết hợp. Các mã đầu ra sửa lỗi là một phương pháp để tận dụng tối đa quá trình chuyển đổi này. Trên thực tế, phương pháp này hoạt động tốt đến mức thường có lợi khi áp dụng nó ngay cả khi thuật toán học có thể xử lý trực tiếp các tập dữ liệu đa lớp.

Trong Phần 4.6 (trang 123), chúng ta đã học cách chuyển đổi tập dữ liệu nhiều lớp thành nhiều tập dữ liệu hai lớp. Đối với mỗi lớp, một tập dữ liệu được tạo có chứa một bản sao của từng phiên bản trong dữ liệu gốc, nhưng với giá trị lớp đã sửa đổi. Nếu cá thể có lớp được liên kết với tập dữ liệu tương ứng thì nó được gắn thẻ có; nếu không thì không. Sau đó, các bộ phân loại được xây dựng cho từng bộ dữ liệu nhị phân này, các bộ phân loại đưa ra con số đáng tin cậy với các dự đoán của chúng—ví dụ: xác suất ước tính rằng lớp đó là có. Trong quá trình phân loại, một phiên bản thử nghiệm được đưa vào từng bộ phân loại nhị phân và lớp cuối cùng là lớp được liên kết với bộ phân loại dự đoán có một cách tự tin nhất. Tất nhiên, phương pháp này nhạy cảm với độ chính xác của các số liệu độ tin cậy do các nhà phân loại tạo ra: nếu một số nhà phân loại có quan điểm phóng đại về dự đoán của chính họ, thì kết quả chung sẽ bị ảnh hưởng.

Xét một bài toán nhiều lớp với bốn lớp a, b, c và d. Sự hình thành chuyển đổi có thể được hình dung như trong Bảng 7.1(a), trong đó có và không tương ứng được ánh xạ tới 1 và 0. Mỗi giá trị lớp ban đầu được chuyển đổi thành một từ mã 4 bit, 1 bit cho mỗi lớp và bốn bộ phân loại dự đoán các bit một cách độc lập. Giải thích quá trình phân loại theo các từ mã này, lỗi xảy ra khi bit nhị phân sai nhận được độ tin cậy cao nhất.

Bảng 7.1 Chuyển đổi một bài toán nhiều lớp thành một bài toán hai lớp: (a) phương pháp tiêu chuẩn và (b) mã sửa lỗi.

| Lớp học | véc tơ lớp | Lớp học | véc tơ lớp |
|---|---|---|---|
| | 1 0 0 0 | | 1 1 1 1 1 1 1 |
| ab | 0 1 0 0 | ab | 0 0 0 0 1 1 1 |
| | 0 0 1 0 | | 0 0 1 1 0 0 1 |
| | 0 0 0 1 | | 0 1 0 1 0 1 0 |
| cd (a) | | cd (b) | |

Tuy nhiên, chúng tôi không phải sử dụng các từ mã cụ thể được hiển thị. Thực vậy, không có lý do tại sao mỗi lớp phải được biểu diễn bằng 4 bit. thay vào đó nhìn vào mã của Bảng 7.1(b), trong đó các lớp được biểu diễn bằng 7 bit. Khi đươ c AP du ng đối với một tập dữ liệu, bảy bộ phân loại phải được xây dựng thay vì bốn. Để xem những gì có thể mua, xem xét phân loại của một trường hợp cụ thể. Giả sử nó thuộc về lớp a và dự đoán của các bộ phân loại riêng lẻ là 1 0 1 1 1 1 1 (tương ứng). Rõ ràng, so sánh từ mã này với từ trong Bảng 7.1(b), bộ phân loại thứ hai đã mắc lỗi: nó dự đoán 0 thay vì 1, thay vào đó là không của vâng. Tuy nhiên, so sánh các bit dự đoán với từ mã được liên kết với mỗi lớp, thể hiện rõ ràng gần với a hơn bất kỳ lớp nào khác. Cái này có thể được định lượng bằng số lượng bit phải được thay đổi để chuyển đổi từ mã dự đoán vào bảng 7.1(b): khoảng cách Hamming, hoặc sự khác biệt giữa các chuỗi bit, là 1, 3, 3 và 5 cho các lớp a, b, c và d, tương ứng. Chúng ta có thể kết luận một cách an toàn rằng bộ phân loại thứ hai đã phạm sai lầm và xác định chính xác a là lớp thực của cá thể.

Loại sửa lỗi tương tự không thể thực hiện được với các từ mã của Bảng 7.1(a), bởi vì bất kỳ chuỗi 4 bit dự đoán nào ngoài bốn từ 4 bit này có cùng khoảng cách với ít nhất hai trong số chúng. Các mã đầu ra không phải là "lỗi đính chính."

Điều gì quyết định liệu mã có được sửa lỗi hay không? xem xét Khoảng cách Hamming giữa các từ mã đại diện cho các lớp khác nhau. Các số lỗi có thể sửa được phụ thuộc vào khoảng cách tối thiểu giữa bất kỳ cặp từ mã nào, chẳng hạn như d. Mã có thể đảm bảo chính xác tối đa (d - 1)/2 lỗi 1 bit, bởi vì nếu số bit này của từ mã chính xác bị đảo lộn, nó vẫn sẽ là từ mã gần nhất và do đó sẽ được xác định một cách chính xác. Trong Bảng 7.1(a) khoảng cách Hamming cho mỗi cặp từ mã là 2. Do đó, khoảng cách nhỏ nhất d cũng là 2, và chúng ta có thể sửa không quá 0 lỗi! Tuy nhiên, trong mã của Bảng 7.1(b) khoảng cách tối thiểu là 4 (trong thực tế, khoảng cách là 4 cho tất cả các cặp). Điều đó có nghĩa là nó được đảm bảo đúng 1 bit lỗi.

Chúng tôi đã xác định được một thuộc tính của mã sửa lỗi tốt: mã các từ phải được phân tách rõ ràng về khoảng cách Hamming của chúng. Vì họ bao gồm các hàng của bảng mã, thuộc tính này được gọi là phân tách hàng. Ở đó là yêu cầu thứ hai mà một mã sửa lỗi tốt phải đáp ứng: cột tách biệt. Khoảng cách Hamming giữa mỗi cặp cột phải là lớn, cũng như khoảng cách giữa mỗi cột và phần bù của mỗi cột khác. Trong Bảng 7.1(b), bảy cột được tách ra từ một khác (và phần bổ sung của chúng) ít nhất 1 bit.

Việc tách cột là cần thiết vì nếu hai cột giống hệt nhau (hoặc nếu cột này là phần bù của cột kia), bộ phân loại tương ứng sẽ mắc lỗi giống nhau. Khả năng sửa lỗi bị suy yếu nếu các lỗi tương quan với nhau—nói cách khác, nếu nhiều vị trí bit đồng thời không chính xác.
Khoảng cách giữa các cột càng lớn thì khả năng sửa lỗi càng nhiều.

Với ít hơn bốn lớp thì không thể xây dựng mã sửa lỗi hiệu quả vì không thể đạt được đồng thời việc tách hàng tốt và tách cột tốt. Ví dụ, với ba lớp thì chỉ có tám cột có thể có (23 ), bốn trong số đó là phần bù của bốn cột còn lại. Hơn nữa, các cột có tất cả các số 0 hoặc tất cả các số không cung cấp sự phân biệt đối xử. Điều này chỉ để lại ba cột có thể và kết quả là

mã không sửa lỗi gì cả. (Trên thực tế, đó là tiêu chuẩn “mỗi người một lớp” mã hóa.)

Nếu có ít lớp, mã sửa lỗi đầy đủ chẳng hạn như mã trong Bảng 7.1(b) có thể được xây dựng. Trong một mã đầy đủ cho k lớp, các cột bao gồm mọi chuỗi k-bit có thể, ngoại trừ các phần bù và các chuỗi tất cả không hoặc tất cả một. Mỗi từ mã chứa 2k-1 - 1 bit. Mã được cấu trúc như sau: từ mã cho lớp đầu tiên bao gồm tất cả các mã; cái đó cho lớp thứ hai có 2k-2 số không theo sau là 2k-2 - 1; thứ ba có 2k-3 số không tiếp theo là 2k-3 , tiếp theo là 2k-3 số 0, tiếp theo là 2k-3 - 1; và vì thế TRÊN. Từ mã thứ i bao gồm các lần chạy xen kẽ của 2k-i số 0 và số 1, chạy cuối cùng là một ngắn.

Với nhiều lớp hơn, mã đầy đủ là không khả thi vì số lượng cột tăng theo cấp số nhân và quá nhiều bộ phân loại phải được xây dựng. TRONG trường hợp đó các phương pháp tinh vi hơn được sử dụng, có thể tạo mã với thuộc tính sửa lỗi tốt từ số lượng cột ít hơn.

Mã đầu ra sửa lỗi không hoạt động đối với các thuật toán học tập cục bộ như với tư cách là những người học dựa trên cá thể, dự đoán lớp của một cá thể bằng cách xem trường hợp đào tạo lân cận. Trong trường hợp phân loại hàng xóm gần nhất, tất cả đầu ra các bit sẽ được dự đoán bằng cách sử dụng cùng một phiên bản đào tạo. Vấn đề có thể là phá vỡ bằng cách sử dụng các tập hợp con thuộc tính khác nhau để dự đoán từng bit đầu ra, giải mã các dự đoán.

## 7.6 Sử dụng dữ liệu chưa gán nhãn

Khi giới thiệu quy trình học máy trong Chương 2, chúng tôi đã phân biệt rõ ràng giữa học có giám sát và học không giám sát—phân loại và phân cụm. Gần đây, các nhà nghiên cứu đã bắt đầu khám phá lãnh thổ giữa hai loại, đôi khi được gọi là học bán giám sát, trong đó mục tiêu là phân loại nhưng đầu vào chứa cả dữ liệu không được gắn nhãn và được gắn nhãn. Tất nhiên, bạn không thể phân loại mà không có dữ liệu được gắn nhãn, vì chỉ có nhãn mới cho biết các lớp đó là gì.

Nhưng đôi khi việc bổ sung một lượng nhỏ dữ liệu được gắn nhãn với một lượng lớn dữ liệu không được gắn nhãn là điều hấp dẫn. Hóa ra dữ liệu chưa được gắn nhãn có thể giúp bạn học các lớp. Làm sao có thể?

Đầu tiên, tại sao bạn muốn nó? Nhiều tình huống đưa ra khối lượng dữ liệu thô khổng lồ, nhưng việc gán các lớp rất tốn kém vì nó đòi hỏi sự hiểu biết sâu sắc của con người. Khai thác văn bản cung cấp một số ví dụ cổ điển. Giả sử bạn muốn phân loại các trang Web thành các nhóm được xác định trước. Trong môi trường học thuật, bạn có thể quan tâm đến các trang của khoa, trang sinh viên tốt nghiệp, trang thông tin khóa học, trang nhóm nghiên cứu và trang khoa. Bạn có thể dễ dàng tải xuống hàng nghìn hoặc hàng triệu trang liên quan từ các trang web của trường đại học. Nhưng việc ghi nhãn dữ liệu đào tạo là một quy trình thủ công tốn nhiều công sức. Hoặc giả sử công việc của bạn là sử dụng công nghệ máy học để phát hiện tên trong văn bản, phân biệt giữa tên cá nhân, tên công ty và tên địa danh. Bạn có thể dễ dàng tải xuống hàng megabyte hoặc gigabyte văn bản, nhưng việc biến điều này thành dữ liệu đào tạo bằng cách chọn tên và phân loại chúng chỉ có thể được thực hiện thủ công. Lập danh mục các bài báo, sắp xếp thư điện tử, tìm hiểu sở thích đọc của người dùng—các ứng dụng có vô số. Để văn bản sang một bên, giả sử bạn muốn học cách nhận ra một số người nổi tiếng trong bản tin truyền hình. Bạn có thể dễ dàng ghi lại hàng trăm hoặc hàng nghìn giờ tin tức, nhưng việc ghi nhãn lại là thủ công. Trong bất kỳ tình huống nào trong số này, sẽ vô cùng hấp dẫn nếu có thể tận dụng một lượng lớn dữ liệu chưa được gắn nhãn để đạt được hiệu suất xuất sắc chỉ từ một vài ví dụ được gắn nhãn—đặc biệt nếu bạn là sinh viên mới tốt nghiệp phải thực hiện việc dán nhãn!

Phân cụm để phân loại Làm thế nào dữ

liệu chưa được gắn nhãn có thể được sử dụng để cải thiện việc phân loại? Đây là một ý tưởng đơn giản. Sử dụng Naïve Bayes để học các lớp từ tập dữ liệu nhỏ được gắn nhãn, sau đó mở rộng nó sang tập dữ liệu lớn không được gắn nhãn bằng cách sử dụng thuật toán phân cụm lặp EM (kỳ vọng-tối đa hóa) của Phần 6.6. Thủ tục là thế này. Đầu tiên, đào tạo một sifier lớp sử dụng dữ liệu được dán nhãn. Thứ hai, áp dụng nó cho dữ liệu chưa được gắn nhãn để gắn nhãn với xác suất của lớp (bước “kỳ vọng”). Thứ ba, đào tạo một trình phân loại mới bằng cách sử dụng nhãn cho tất cả dữ liệu (bước “tối đa hóa”). Thứ tư, lặp lại cho đến khi hội tụ. Bạn có thể coi đây là phân cụm lặp đi lặp lại, trong đó các điểm bắt đầu

và nhãn cụm được thu thập từ dữ liệu được dán nhãn. Quy trình EM đảm bảo tìm ra các tham số mô hình có khả năng bằng hoặc lớn hơn ở mỗi lần lặp. Câu hỏi quan trọng, chỉ có thể được trả lời theo kinh nghiệm, là liệu các ước tính tham số có khả năng cao hơn này có cải thiện độ chính xác của phân loại hay không.

Theo trực giác, điều này có thể hoạt động tốt. Xem xét phân loại tài liệu. Một số cụm từ là biểu thị của các lớp học. Một số xuất hiện trong các tài liệu được dán nhãn, trong khi một số khác chỉ xuất hiện trong các tài liệu không được dán nhãn. Nhưng có thể có một số tài liệu chứa cả hai và quy trình EM sử dụng những điều này để khái quát hóa mô hình đã học để sử dụng các cụm từ không xuất hiện trong tập dữ liệu được gắn nhãn. Ví dụ: cả người hướng dẫn và chủ đề tiến sĩ có thể chỉ ra trang chủ của sinh viên tốt nghiệp.
Giả sử rằng chỉ có cụm từ cũ xuất hiện trong các tài liệu được dán nhãn. EM iter chủ động khái quát hóa mô hình để phân loại chính xác các tài liệu chỉ chứa tài liệu sau.

Điều này có thể hoạt động với bất kỳ trình phân loại nào và bất kỳ thuật toán phân cụm lặp nào. Nhưng về cơ bản, đó là một quy trình khởi động và bạn phải cẩn thận để đảm bảo rằng vòng phản hồi là một vòng tích cực. Sử dụng xác suất thay vì các quyết định khó khăn có vẻ có lợi vì nó cho phép quy trình hội tụ từ từ thay vì vội vàng đưa ra kết luận có thể sai. Naïve Bayes và thủ tục EM có khả năng xác suất được mô tả trong Phần 6.6 là những lựa chọn đặc biệt thích hợp vì chúng có chung một giả định cơ bản: tính độc lập giữa các thuộc tính—hay chính xác hơn là tính độc lập có điều kiện giữa các thuộc tính đã cho của lớp.

Tất nhiên, giả định độc lập bị vi phạm phổ biến. Ngay cả ví dụ nhỏ của chúng tôi cũng sử dụng chủ đề tiến sĩ cụm từ hai từ , trong khi các triển khai thực tế có thể sẽ sử dụng các từ riêng lẻ làm thuộc tính—và ví dụ này sẽ kém hấp dẫn hơn nhiều nếu chúng tôi thay thế một trong hai thuật ngữ đơn lẻ tiến sĩ hoặc chủ đề. Cụm từ sinh viên tiến sĩ có lẽ chỉ về khoa hơn là trang chủ của sinh viên tốt nghiệp; chủ đề nghiên cứu cụm từ có lẽ ít phân biệt đối xử hơn. Chính thực tế là tiến sĩ và chủ đề không phải là độc lập có điều kiện đối với lớp làm cho ví dụ hoạt động: chính sự kết hợp của chúng là đặc điểm của các trang nghiên cứu sinh.

Tuy nhiên, việc kết hợp Naïve Bayes và EM theo cách này hoạt động tốt trong lĩnh vực phân loại tài liệu. Trong một nhiệm vụ phân loại cụ thể, nó đạt được hiệu suất của một người học truyền thống khi sử dụng ít hơn một phần ba số trường hợp đào tạo được gắn nhãn, cũng như gấp năm lần số trường hợp không được gắn nhãn. Đây là một sự đánh đổi tốt khi các phiên bản được gắn nhãn đắt tiền nhưng các phiên bản không được gắn nhãn hầu như miễn phí. Với một số ít tài liệu được dán nhãn, độ chính xác của việc phân loại có thể được cải thiện đáng kể bằng cách kết hợp nhiều tài liệu không được dán nhãn.

Hai cải tiến đối với quy trình đã được chứng minh là cải thiện hiệu suất.
Đầu tiên được thúc đẩy bởi bằng chứng thực nghiệm rằng khi có nhiều tài liệu được dán nhãn, việc kết hợp dữ liệu không được dán nhãn có thể làm giảm thay vì tăng độ chính xác. Dữ liệu được dán nhãn thủ công vốn đã (hoặc nên) ít nhiễu hơn so với

dữ liệu được gán nhãn tự động. Giải pháp là giới thiệu một tham số trọng số làm giảm sự đóng góp của dữ liệu chưa được gắn nhãn. Điều này có thể được tích hợp vào bước tối đa hóa EM bằng cách tối đa hóa khả năng có trọng số của các phiên bản được gắn nhãn và không được gắn nhãn. Khi tham số gần bằng 0, các tài liệu không được dán nhãn có ít ảnh hưởng đến hình dạng bề mặt leo đồi của EM; khi gần bằng một, thuật toán trở lại phiên bản gốc trong đó bề mặt bị ảnh hưởng như nhau bởi cả hai loại tài liệu.

Cải tiến thứ hai là cho phép mỗi lớp có nhiều cụm. Như đã giải thích trong Phần 6.6, thuật toán phân cụm EM giả định rằng dữ liệu được tạo ngẫu nhiên từ hỗn hợp các phân bố xác suất khác nhau, mỗi phân bố một cụm. Cho đến nay, sự tương ứng một đối một giữa các thành phần hỗn hợp và các lớp đã được giả định. Trong nhiều trường hợp, điều này là không thực tế—bao gồm cả việc phân loại tài liệu, vì hầu hết các tài liệu đề cập đến nhiều chủ đề. Với một số cụm trên mỗi lớp, mỗi tài liệu được dán nhãn ban đầu được gán ngẫu nhiên cho từng thành phần của nó theo kiểu xác suất. Bước tối đa hóa của thuật toán EM vẫn như trước, nhưng bước kỳ vọng được sửa đổi để không chỉ gắn nhãn xác suất cho từng ví dụ với các lớp, mà còn gán xác suất cho các thành phần trong lớp. Số lượng cụm trên mỗi lớp là một tham số phụ thuộc vào miền và có thể được đặt bằng cách xác thực chéo.

### Đồng đào tạo

Một tình huống khác trong đó dữ liệu không được gắn nhãn có thể cải thiện hiệu suất phân loại là khi có hai quan điểm khác nhau và độc lập về nhiệm vụ phân loại. Ví dụ cổ điển lại liên quan đến các tài liệu, lần này là các tài liệu Web, trong đó có hai quan điểm là nội dung của một trang Web và các liên kết đến nó từ các trang khác. Hai quan điểm này đều được biết là hữu ích và khác nhau: các công cụ tìm kiếm Web thành công tận dụng cả hai, sử dụng các công thức bí mật. Văn bản gắn nhãn một liên kết đến một trang Web khác cung cấp manh mối tiết lộ về nội dung của trang đó—có lẽ còn tiết lộ nhiều hơn nội dung của chính trang đó, đặc biệt nếu liên kết đó là một liên kết độc lập. Theo trực giác, một liên kết được gắn nhãn cố vấn của tôi là bằng chứng mạnh mẽ rằng trang đích là trang chủ của một giảng viên.

Ý tưởng, được gọi là đồng đào tạo, là thế này. Đưa ra một vài ví dụ được gắn nhãn, trước tiên hãy tìm hiểu một mô hình khác cho từng phối cảnh—trong trường hợp này là mô hình dựa trên nội dung và siêu liên kết. Sau đó, sử dụng riêng từng cái để gắn nhãn cho các ví dụ chưa được gắn nhãn. Đối với mỗi mô hình, hãy chọn ví dụ mà nó tự tin nhất dán nhãn là tích cực và mẫu mà nó tự tin nhất dán nhãn là phủ định, rồi thêm những ví dụ này vào nhóm các ví dụ đã dán nhãn. Tốt hơn hết, hãy duy trì tỷ lệ các bài kiểm tra tích cực và tiêu cực trong nhóm được dán nhãn bằng cách chọn loại này nhiều hơn loại kia. Trong cả hai trường hợp, hãy lặp lại toàn bộ quy trình, đào tạo cả hai mô hình trên nhóm ví dụ được gắn nhãn tăng cường, cho đến khi cạn kiệt nhóm không được gắn nhãn.

Có một số bằng chứng thực nghiệm, sử dụng Naïve Bayes xuyên suốt với tư cách là người học, rằng quy trình khởi động này hoạt động tốt hơn quy trình sử dụng tất cả các tính năng từ cả hai quan điểm để tìm hiểu một mô hình duy nhất từ dữ liệu được dán nhãn. Nó dựa vào việc có hai chế độ xem khác nhau của một cá thể dư thừa nhưng không tương quan hoàn toàn. Nhiều lĩnh vực khác nhau đã được đề xuất, từ phát hiện những người nổi tiếng trong các bản tin truyền hình sử dụng video và âm thanh riêng biệt cho đến rô-bốt di động có cảm biến tầm nhìn, sonar và phạm vi. Tính độc lập của các quan điểm làm giảm khả năng cả hai giả thuyết đồng ý về một nhãn sai.

### EM và đồng đào tạo

Trên các bộ dữ liệu có hai bộ tính năng thực sự độc lập, các thử nghiệm đã chỉ ra rằng đồng đào tạo mang lại kết quả tốt hơn so với sử dụng EM như đã mô tả trước đây. Tuy nhiên, thậm chí có thể đạt được hiệu suất tốt hơn bằng cách kết hợp cả hai thành một phiên bản đồng đào tạo đã sửa đổi được gọi là co-EM. Đồng đào tạo đào tạo hai bộ phân loại đại diện cho các quan điểm khác nhau, A và B, và sử dụng cả hai để thêm các ví dụ mới vào nhóm đào tạo bằng cách chọn bất kỳ ví dụ nào không được gắn nhãn mà chúng phân loại tích cực hoặc tiêu cực nhất. Các ví dụ mới có số lượng ít và được dán nhãn xác định. Mặt khác, Co-EM đào tạo phối cảnh A trên dữ liệu được gắn nhãn và sử dụng nó để gắn nhãn theo xác suất cho tất cả dữ liệu chưa được gắn nhãn. Tiếp theo, nó huấn luyện bộ phân loại B trên cả dữ liệu được gắn nhãn và dữ liệu không được gắn nhãn với các nhãn dự kiến của bộ phân loại A, sau đó nó dán nhãn lại tất cả dữ liệu theo xác suất để bộ phân loại A sử dụng. Quá trình lặp lại cho đến khi các bộ phân loại hội tụ. Quy trình này dường như hoạt động ổn định tốt hơn so với đồng đào tạo vì nó không cam kết với các nhãn lớp được tạo bởi bộ phân loại A và B mà thay vào đó ước tính lại xác suất của chúng ở mỗi lần lặp lại.

Phạm vi áp dụng của đồng EM, như đồng đào tạo, vẫn bị giới hạn bởi yêu cầu về nhiều quan điểm độc lập. Nhưng có một số bằng chứng thực nghiệm cho thấy rằng ngay cả khi không có sự phân chia tự nhiên các đối tượng địa lý thành các quan điểm độc lập, vẫn có thể đạt được lợi ích bằng cách tạo ra sự phân chia như vậy và sử dụng đồng đào tạo—hoặc tốt hơn nữa là đồng EM—trên sự phân chia dữ liệu. Điều này dường như hoạt động ngay cả khi việc phân chia được thực hiện ngẫu nhiên; hiệu suất chắc chắn có thể được cải thiện bằng kỹ thuật phân tách sao cho các bộ tính năng độc lập tối đa. Tại sao điều này làm việc? Các nhà nghiên cứu đã đưa ra giả thuyết rằng các thuật toán này thành công một phần vì sự phân chia khiến chúng trở nên mạnh mẽ hơn đối với các giả định mà các bộ phân loại cơ bản của chúng đưa ra.

Không có lý do cụ thể nào để hạn chế bộ phân loại cơ sở đối với Naïve Bayes. Máy vectơ hỗ trợ có lẽ là đại diện cho công nghệ phân loại văn bản thành công nhất hiện nay. Tuy nhiên, để phép lặp EM hoạt động, bộ phân loại cần gắn nhãn dữ liệu theo xác suất; nó cũng phải có khả năng sử dụng các ví dụ có trọng số xác suất để đào tạo. Máy vectơ hỗ trợ có thể dễ dàng được điều chỉnh để thực hiện cả hai. Chúng tôi đã giải thích cách điều chỉnh các thuật toán học tập để

xử lý các trường hợp có trọng số trong Phần 6.5 dưới Hồi quy tuyến tính có trọng số cục bộ (trang 252). Một cách để có được ước tính xác suất từ vectơ hỗ trợ máy là để khớp mô hình logistic một chiều với đầu ra, một cách hiệu quả thực hiện hồi quy logistic như được mô tả trong Phần 4.6 trên đầu ra. Kết quả cho vay Excel đã được báo cáo để phân loại văn bản bằng cách sử dụng co-EM với bộ phân loại máy vectơ hỗ trợ (SVM). Nó vượt trội so với các biến thể khác của SVM và có vẻ khá mạnh mẽ đối với các tỷ lệ khác nhau của dữ liệu được gắn nhãn và không được gắn nhãn.

Các ý tưởng về đồng đào tạo và EM—và đặc biệt là sự kết hợp của chúng trong thuật toán đồng EM—rất thú vị, kích thích tư duy và có tiềm năng nổi bật. Nhưng những gì làm cho chúng hoạt động vẫn còn gây tranh cãi và chưa được hiểu rõ. Những kỹ thuật này là chủ đề của nghiên cứu hiện tại: chúng chưa được đưa vào xu hướng chủ đạo của học máy và được khai thác cho dữ liệu thực tế khai thác mỏ.

## 7.7 Đọc thêm

Lựa chọn thuộc tính, theo thuật ngữ lựa chọn tính năng, đã được nghiên cứu trong lĩnh vực nhận dạng mẫu trong nhiều thập kỷ. Loại bỏ ngược, ví dụ, là được giới thiệu vào đầu những năm 1960 (Marill và Green 1963). Khảo sát của Kittler (1978) các thuật toán lựa chọn tính năng đã được phát triển để nhận dạng mẫu. Tìm kiếm tốt nhất đầu tiên và thuật toán di truyền là trí tuệ nhân tạo tiêu chuẩn kỹ thuật (Winston 1992, Goldberg 1989).

John (1997) đã báo cáo các thí nghiệm cho thấy hiệu suất của những người học cây quyết định giảm đi khi các thuộc tính mới được thêm vào. một lời giải thích hay về lựa chọn thuộc tính. Ý tưởng tìm tập thuộc tính nhỏ nhất có thể tạo ra các cá thể một cách duy nhất là của Almuallin và Dietterich. (1991, 1992) và được phát triển thêm bởi Liu và Setiono (1996). Kibler và Aha (1987) và Cardie (1993) đều nghiên cứu việc sử dụng các thuật toán cây quyết định để xác định các đặc trưng cho việc học láng giềng gần nhất; Holmes và Nevill Manning (1995) đã sử dụng 1R để sắp xếp các tính năng để lựa chọn. Kira và Rendell (1992) đã sử dụng các phương pháp dựa trên phiên bản để chọn các tính năng, dẫn đến một lược đồ có tên là TIN CẬY cho việc loại bỏ đệ quy các tính năng. Gilad-Bachrach et al. (2004) cho thấy làm thế nào lược đồ này có thể được sửa đổi để hoạt động tốt hơn với các thuộc tính dư thừa. Các Phương pháp lựa chọn tính năng dựa trên tương quan được phát triển bởi Hall (2000).

Việc sử dụng các phương pháp bao bọc để lựa chọn tính năng là do John et al. (1994) và Kohavi và John (1997), và các thuật toán di truyền đã được áp dụng trong khung bao bọc của Vafaie và DeJong (1992) và Cherkauer và Shavlik (1996). Phương pháp học Naïve Bayes có chọn lọc là do Langley và Sage (1994). Guyon et al. (2002) trình bày và đánh giá sơ đồ loại bỏ tính năng đệ quy kết hợp với máy vectơ hỗ trợ. Phương pháp đua tìm kiếm được phát triển bởi Moore và Lee (1994).

Dougherty et al. (1995) đưa ra một tài khoản ngắn gọn về giám sát và không giám sát rời rạc hóa, cùng với kết quả thử nghiệm so sánh dựa trên entropy với phương pháp tạo thùng có chiều rộng bằng nhau và phương pháp 1R. Frank và Witten (1999) mô tả tác dụng của việc sử dụng thông tin đặt hàng trong các thuộc tính rời rạc. Sự phân biệt khoảng k theo tỷ lệ cho Naïve Bayes được đề xuất bởi Yang và Webb (2001). Phương pháp dựa trên entropy để phân biệt, bao gồm cả việc sử dụng của tiêu chí dừng MDL, được phát triển bởi Fayyad và Irani (1993). Các Phương pháp thống kê từ dưới lên sử dụng kiểm định c2 là của Kerber (1992), và phần mở rộng đến mức ý nghĩa được xác định tự động được mô tả bởi Liu và Setiono (1997). Fulton và cộng sự. (1995) điều tra việc sử dụng lập trình động để rời rạc hóa và rút ra giới hạn thời gian bậc hai cho một tổng quát hàm tạp chất (ví dụ, entropy) và hàm tuyến tính cho sự phân biệt dựa trên lỗi. Ví dụ được sử dụng để chỉ ra điểm yếu của sự rời rạc hóa dựa trên lỗi được chuyển thể từ Kohavi và Sahami (1996), những người đầu tiên xác định rõ ràng hiện tượng này.

Phân tích thành phần chính là một kỹ thuật tiêu chuẩn có thể được tìm thấy trong thống kê nhất sách giáo khoa. Fradkin và Madigan (2003) phân tích hiệu suất của các phép chiếu ngẫu nhiên. Số liệu TF ¥ IDF được mô tả bởi Witten et al. (1999b).

Các thử nghiệm sử dụng C4.5 để lọc dữ liệu đào tạo của chính nó đã được báo cáo của John (1995). Cách tiếp cận thận trọng hơn của bộ lọc đồng thuận liên quan đến một số thuật toán học tập đã được nghiên cứu bởi Brodley và Friedl (1996). Rousseeuw và Leroy (1987) mô tả việc phát hiện các ngoại lệ trong hồi quy thống kê, bao gồm cả phương pháp bình phương nhỏ nhất; họ cũng trình bày dữ liệu điện thoại của hình 7.6. Chính Quinlan (1986) đã chú ý việc loại bỏ nhiễu khỏi các thuộc tính của phiên bản đào tạo có thể làm giảm hiệu suất của trình phân loại trên các phiên bản kiểm tra nhiễu tương tự, đặc biệt là ở cấp độ cao hơn mức độ tiếng ồn.

Kết hợp nhiều mô hình là một chủ đề nghiên cứu phổ biến trong học máy nghiên cứu, với nhiều ấn phẩm liên quan. Thuật ngữ đóng bao (đối với “bootstrap tổng hợp”) được đặt ra bởi Breiman (1996b), người đã điều tra các đặc tính của đóng bao về mặt lý thuyết và thực nghiệm cho cả phân loại và dự đoán số. Domingos (1999) giới thiệu thuật toán MetaCost. ngẫu nhiên hóa được đánh giá bởi Dietterich (2000) và được so sánh với đóng gói và tăng tốc. Bay (1999) gợi ý sử dụng phương pháp ngẫu nhiên hóa để học tập đồng bộ với các bộ phân loại hàng xóm gần nhất. Rừng ngẫu nhiên được giới thiệu bởi Breiman (2001).

Freund và Schapire (1996) đã phát triển thuật toán tăng cường AdaBoost.M1 và dẫn xuất giới hạn lý thuyết cho hiệu suất của nó. Sau đó, họ đã cải thiện những giới hạn sử dụng khái niệm biên (Freund và Schapire 1999). người uống rượu (1997) đã điều chỉnh AdaBoost.M1 để dự đoán số. Thuật toán LogitBoost được phát triển bởi Friedman et al. (2000). Friedman (2001) mô tả cách làm cho việc tăng tốc trở nên linh hoạt hơn khi có dữ liệu nhiễu.

Domingos (1997) mô tả cách rút ra một mô hình có thể giải thích được từ một nhóm sử dụng các ví dụ đào tạo nhân tạo. Cây quyền chọn Bayes được giới thiệu bởi Buntine (1992), và biểu quyết theo đa số đã được đưa vào quyền chọn cây của Kohavi và Kunz (1997). Freund và Mason (1999) giới thiệu cây quyết định thay thế; thí nghiệm với cây quyết định xen kẽ nhiều lớp đã được báo cáo bởi Holmes et al. (2002). Landwehr et al. (2003) đã phát triển cây mô hình logis tic sử dụng thuật toán LogitBoost.

Khái quát hóa xếp chồng bắt nguồn từ Wolpert (1992), người đã trình bày ý tưởng trong tài liệu về mạng thần kinh và được áp dụng cho dự đoán số bởi Breiman (1996a). Ting và Witten (1997a) so sánh các mô hình cấp 1 khác nhau theo kinh nghiệm và thấy rằng một mô hình tuyến tính đơn giản hoạt động tốt nhất; họ cũng đã chứng minh lợi thế của việc sử dụng xác suất làm dữ liệu cấp 1. Sự kết hợp giữa xếp chồng và đóng bao cũng đã được nghiên cứu (Ting và Witten 1997b).

Ý tưởng sử dụng mã đầu ra sửa lỗi để phân loại đã được phổ biến rộng rãi chấp nhận sau bài báo của Dietterich và Bakiri (1995); Ricci và Aha (1998) đã chỉ ra cách áp dụng các mã như vậy cho các bộ phân loại lân cận gần nhất.

Blum và Mitchell (1998) đi tiên phong trong việc sử dụng đồng đào tạo và phát triển một mô hình lý thuyết cho việc sử dụng dữ liệu được gắn nhãn và không được gắn nhãn từ các quan điểm độc lập khác nhau. Nigam và Ghani (2000) đã phân tích hiệu quả và khả năng áp dụng đồng đào tạo, liên quan đến việc sử dụng EM tiêu chuẩn truyền thống để điền các giá trị còn thiếu. Họ cũng giới thiệu thuật toán co-EM. Nigam et al. (2000) đã khám phá kỹ lưỡng cách thuật toán phân cụm EM có thể sử dụng không được gắn nhãn dữ liệu để cải thiện bộ phân loại ban đầu do Naïve Bayes xây dựng, như đã báo cáo trong phần Phân cụm để phân loại . Cho đến thời điểm này, đồng đào tạo và đồng EM chủ yếu được áp dụng cho các bài toán hai lớp nhỏ; Ghani (2002) đã sử dụng mã đầu ra sửa lỗi để giải quyết các tình huống đa lớp với nhiều lớp.

Brefeld và Scheffer (2004) đã mở rộng co-EM để sử dụng máy vectơ hỗ trợ hơn là Naïve Bayes. Seeger (2001) đưa ra một số nghi ngờ về việc liệu những các thuật toán thực sự có bất cứ điều gì để cung cấp so với các thuật toán truyền thống, được sử dụng đúng cách.