Chương 5

Kiến thức cơ bản về học máy

Học sâu là một lĩnh vực cụ thể trong học máy. Để hiểu kỹ hơn về học sâu, chúng ta cần nắm vững các nguyên lý cơ bản của học máy. Chương này cung cấp một khóa học ngắn gọn về các nguyên lý quan trọng nhất sẽ được áp dụng trong phần còn lại của cuốn sách. Các độc giả mới bắt đầu hoặc những người muốn có cái nhìn rộng hơn nên tham khảo các sách giáo khoa về học máy với phạm vi bao quát đầy đủ hơn về các nguyên lý cơ bản, chẳng hạn như *Machine Learning: A Probabilistic Perspective* (Murphy, 2012, [1]) hoặc *Pattern Recognition and Machine Learning* (Bishop, 2006, [2]). Nếu bạn đã quen thuộc với các khái niệm cơ bản của học máy, bạn có thể chuyển đến ngay Mục 5.11. Mục đó bao gồm

một số góc nhìn về các kỹ thuật học máy truyền thống đã ảnh hưởng mạnh mẽ đến sự phát triển của các thuật toán học sâu.

Chúng ta bắt đầu bằng việc định nghĩa thuật toán học là gì và giới thiệu một ví dụ minh họa: thuật toán hồi quy tuyến tính. Tiếp theo, chúng tôi mô tả sự khác biệt giữa việc khớp dữ liệu huấn luyện và việc tìm ra các mẫu có thể tổng quát hóa cho dữ liệu mới. Hầu hết các thuật toán học máy có các siêu tham số cần được xác định bên ngoài bản thân thuật toán học; ta thảo luận cách thiết lập các tham số này bằng cách sử dụng thêm dữ liệu khác. Về cơ bản, học máy là một dạng thống kê ứng dụng với nhấn mạnh vào việc sử dụng máy tính để ước lượng thống kê các hàm phức tạp, và ít chú trọng vào việc chứng minh các khoảng tin cậy quanh những hàm này; do đó, chúng tôi trình bày hai phương pháp tiếp cận trung tâm trong thống kê: ước lượng theo tần suất và suy luận Bayes. Hầu hết các thuật toán học máy có thể được chia thành các loại học có giám sát và học không giám sát; chúng tôi mô tả các loại này và đưa ra một số ví dụ về các thuật toán học đơn giản trong từng loại. Phần lớn các thuật toán học sâu đều dựa trên một thuật toán tối ưu hóa gọi là hướng giảm ngẫu nhiên. Chúng tôi mô tả cách kết hợp các thành phần của thuật toán như thuật toán tối ưu, hàm chi phí, mô hình, và tập dữ liệu để xây dựng một thuật toán học máy. Cuối cùng, trong Mục 5.11, chúng tôi mô tả một số yếu tố đã giới hạn khả năng tổng quát hóa của học máy truyền thống. Các thách thức này đã thúc đẩy sự phát triển của các thuật toán học sâu nhằm vượt qua những trở ngại này.

5.1 Các thuật toán học

Thuật toán học máy là một thuật toán có khả năng học từ dữ liệu. Nhưng "học" ở đây có nghĩa là gì? *Machine Learning* (Mitchell, 1997) đưa ra định nghĩa: "Một chương trình máy tính được cho là học từ trải nghiệm E đối với một lớp tác vụ T và thước đo hiệu suất P, nếu hiệu suất của nó trong các tác vụ thuộc T, được đo bằng P, cải thiện khi có thêm trải nghiệm E.". Ta có thể tưởng tượng ra rất nhiều loại trải nghiệm E, tác vụ T, và thước đo hiệu suất P, và trong cuốn sách này, chúng tôi không cố gắng đưa ra một định nghĩa chính thức cho những yếu tố này. Thay vào đó, các phần sau sẽ cung cấp các mô tả trực quan và ví dụ về các loại tác vụ, thước đo hiệu suất và trải nghiệm khác nhau có thể được sử dụng để xây dựng các thuật toán học máy.

5.1.1 Tác vụ T

Học máy cho phép chúng ta giải quyết các tác vụ quá phức tạp để có thể xử lý bằng các chương trình cố định do con người viết và thiết kế. Từ góc độ khoa học và triết học, học máy trở nên thú vị vì việc phát triển sự hiểu biết của chúng ta về học máy đồng nghĩa với việc phát triển sự hiểu biết về các nguyên tắc cơ bản của trí tuệ.

Trong định nghĩa khá chính thức này về từ "tác vụ", quá trình học tập không phải là tác vụ mà chỉ là phương tiện để đạt được khả năng thực hiện tác vụ. Ví dụ, nếu ta muốn một robot có khả năng đi bộ, thì đi bộ là tác vụ. Ta có thể lập trình để robot học cách đi, hoặc có thể thử viết trực tiếp một chương trình hướng dẫn robot cách đi theo cách thủ công.

Các tác vụ học máy thường được mô tả dựa trên cách hệ thống học máy nên xử lý một **ví dụ**. Một ví dụ là một tập hợp **các đặc trưng** đã được đo lường định lượng từ một đối tượng hoặc sự kiện nào đó mà ta muốn hệ thống học máy xử lý. Ta thường biểu diễn một ví dụ dưới dạng một véctơ $\boldsymbol{x} \in \mathbb{R}^n$, trong đó các phần tử x_i của véctơ là một đặc trưng khác nhau. Ví dụ, các đặc trưng của một hình ảnh thường là các giá trị của các điểm ảnh trong hình ảnh đó.

Nhiều loại tác vụ khác nhau có thể được giải quyết bằng học máy. Một số tác vụ học máy phổ biến nhất bao gồm:

- **Phân loại**: Trong loại tác vụ này, chương trình máy tính được yêu cầu xác định xem một đầu vào thuộc về một trong k loại. Để giải quyết tác vụ này, thuật toán học thường được yêu cầu tạo ra một hàm $f : \mathbb{R}^n \to \{1, \ldots, k\}$. Khi $y = f(\boldsymbol{x})$, mô hình sẽ gán đầu vào được mô tả bằng véctơ $\boldsymbol{x}$ với một loại được nhận diện bởi mã số y. Có những biến thể khác của tác vụ phân loại, chẳng hạn như khi f trả về một phân phối xác suất theo các lớp. Một ví dụ về tác vụ phân loại là nhận dạng đối tượng, nơi đầu vào là một hình ảnh (thường được mô tả dưới dạng tập hợp các giá trị độ sáng của điểm ảnh) và đầu ra là mã số nhận diện đối tượng trong hình. Ví dụ, robot PR2 của Willow Garage có thể hoạt động như một người phục vụ, nhận dạng các loại đồ uống khác nhau và mang đến cho người theo yêu cầu (*Help Me Help You: Interfaces for Personal Robots*, Goodfellow và cộng sự, 2010, [3]). Công nghệ nhận dạng đối tượng hiện đại được thực hiện tốt nhất với học sâu (*ImageNet Classification with Deep Convolutional Neural Networks*, Krizhevsky và cộng sự, 2012, [4]; *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*,

Ioffe và Szegedy, 2015, [5]). Nhận dạng đối tượng là công nghệ cơ bản cho phép máy tính nhận diện khuôn mặt (*DeepFace: Closing the Gap to Human-Level Performance in Face Verification*, Taigman và cộng sự, 2014, [6]), giúp tự động gắn thẻ người trong các bộ sưu tập ảnh và cho phép máy tính tương tác tự nhiên hơn với người dùng.

- **Phân loại với đầu vào bị thiếu**: Phân loại trở nên thách thức hơn khi chương trình máy tính không đảm bảo rằng mọi phép đo cho véctơ đầu vào sẽ luôn được cung cấp đầy đủ. Để giải quyết tác vụ phân loại, thuật toán học máy chỉ cần định nghĩa một hàm đơn trị ánh xạ mỗi véctơ đầu vào với một đầu ra phân loại. Tuy nhiên, khi một số đặc trưng của các đầu vào có thể bị thiếu, thuật toán học phải học một tập hợp các hàm. Mỗi hàm tương ứng với việc phân loại $\boldsymbol{x}$ khi một tập con đặc trưng khác nhau của các đầu vào bị thiếu. Tình huống này thường xảy ra trong chẩn đoán y khoa, vì nhiều loại xét nghiệm y tế có chi phí cao hoặc xâm lấn. Một cách để định nghĩa một cách hiệu quả tập hợp lớn các hàm này là học một phân phối xác suất trên tất cả các biến liên quan, sau đó giải quyết tác vụ phân loại bằng cách lấy tổng các biến bị thiếu. Với n biến đầu vào, ta có thể tạo ra 2^n hàm phân loại khác nhau cần thiết cho từng tập hợp đầu vào bị thiếu, nhưng chỉ cần học một hàm đơn trị mô tả phân phối xác suất đồng thời. Xem *Multi-Prediction Deep Boltzmann Machines* (Goodfellow và cộng sự, 2013, [7]) để biết ví dụ về một mô hình xác suất sâu áp dụng cho tác vụ này. Nhiều tác vụ khác được mô tả trong phần này cũng có thể được tổng quát hóa để hoạt động với đầu vào bị thiếu; phân loại với đầu vào bị thiếu chỉ là một ví dụ về khả năng của học máy.

- **Hồi quy**: Trong loại tác vụ này, chương trình máy tính được yêu cầu dự đoán một giá trị số cho một đầu vào nào đó. Để giải quyết tác vụ này, thuật toán học máy được yêu cầu tạo ra một hàm $f : \mathbb{R}^n \to \mathbb{R}$. Loại tác vụ này tương tự như phân loại, ngoại trừ việc định dạng đầu ra khác nhau. Một ví dụ về tác vụ hồi quy là dự đoán số tiền yêu cầu bồi thường dự kiến mà một người được bảo hiểm sẽ thực hiện (được sử dụng để xác định phí bảo hiểm) hoặc dự đoán giá tương lai của chứng khoán. Những loại dự đoán này cũng được sử dụng trong giao dịch thuật toán.

- **Phiên âm**: Trong loại tác vụ này, hệ thống học máy được yêu cầu quan sát một dạng dữ liệu không có cấu trúc rõ ràng và chuyển nó thành dạng văn

bản rời rạc. Ví dụ, trong nhận dạng ký tự quang học, chương trình máy tính được cung cấp một bức ảnh chứa hình ảnh văn bản và được yêu cầu trả về văn bản này dưới dạng một chuỗi ký tự (ví dụ, ở định dạng ASCII hoặc Unicode). Google Street View sử dụng học sâu để xử lý số địa chỉ theo cách này (*Multi-digit number recognition from Street View imagery using deep convolutional neural networks*, Goodfellow và cộng sự, 2014, [8]). Một ví dụ khác là nhận dạng giọng nói, trong đó chương trình máy tính được cung cấp một dạng sóng âm thanh và tạo ra một chuỗi ký tự hoặc mã từ để mô tả các từ đã được nói trong bản ghi âm. Học sâu là thành phần quan trọng trong các hệ thống nhận dạng giọng nói hiện đại được sử dụng tại các công ty lớn như Microsoft, IBM và Google (*Deep Neural Networks for Acoustic Modeling in Speech Recognition*, Hinton và cộng sự, 2012, [9]).

- **Dịch máy**: Trong tác vụ dịch máy, đầu vào là một chuỗi ký hiệu trong một ngôn ngữ nhất định và chương trình máy tính phải chuyển đổi nó thành một chuỗi ký hiệu trong ngôn ngữ khác. Tác vụ này thường được áp dụng cho các ngôn ngữ tự nhiên, chẳng hạn như dịch từ tiếng Anh sang tiếng Việt. Gần đây, học sâu đã bắt đầu có tác động quan trọng đối với loại tác vụ này (*Sequence to Sequence Learning with Neural Networks*, Sutskever và cộng sự, 2014, [10]; *Neural Machine Translation by Jointly Learning to Align and Translate*, Bahdanau và cộng sự, 2015, [11]).

- **Đầu ra có cấu trúc**: Các tác vụ đầu ra có cấu trúc liên quan đến bất kỳ tác vụ nào mà đầu ra là một véctơ (hoặc cấu trúc dữ liệu khác chứa nhiều giá trị) với các mối quan hệ quan trọng giữa các phần tử khác nhau. Đây là một phạm trù rộng, bao gồm cả các tác vụ phiên âm và dịch thuật đã được mô tả ở trên, nhưng cũng bao gồm nhiều tác vụ khác. Một ví dụ là phân tích cú pháp — chuyển đổi một câu trong ngôn ngữ tự nhiên thành một cây mô tả cấu trúc ngữ pháp của nó và gắn thẻ các nút của cây như động từ, danh từ, hoặc trạng từ, v.v. Tham khảo *Deep Learning for Efficient Discriminative Parsing*, (Collobert, 2011, [12]) để xem ví dụ về học sâu được áp dụng vào tác vụ phân tích cú pháp. Một ví dụ khác là phân đoạn hình ảnh theo từng điểm ảnh, nơi chương trình máy tính gán từng điểm ảnh trong một bức ảnh với một danh mục cụ thể. Ví dụ, học sâu có thể được sử dụng để chú thích vị trí của các con đường trong các bức ảnh chụp từ trên cao (*Learning to Detect Roads in High-Resolution Aerial Images*, Mnih và Hinton, 2010, [13]). Hình thức đầu ra không nhất thiết phải phản ánh

cấu trúc đầu vào một cách chặt chẽ như trong các tác vụ kiểu chú thích này. Chẳng hạn, trong chú thích hình ảnh, chương trình máy tính quan sát một hình ảnh và tạo ra một câu bằng ngôn ngữ tự nhiên để mô tả hình ảnh đó (*Multimodal Neural Language Models* và *Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models*, Kiros và cộng sự, 2014, [14, 15]; *Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN)*, Mao và cộng sự, 2015, [16]; *Show and Tell: A Neural Image Caption Generator*, Vinyals và cộng sự, 2015, [17]; *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*, Donahue và cộng sự, 2014, [18]; *Deep Visual-Semantic Alignments for Generating Image Descriptions*, Karpathy và Li, 2015, [19]; *From Captions to Visual Concepts and Back*, Fang và cộng sự, 2015, [20]; *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*, Xu và cộng sự, 2015, [21]). Những tác vụ này được gọi là tác vụ đầu ra có cấu trúc vì chương trình phải xuất ra nhiều giá trị liên quan mật thiết với nhau. Ví dụ, các từ do chương trình chú thích hình ảnh tạo ra phải tạo thành một câu hoàn chỉnh.

- **Phát hiện bất thường**: Trong loại tác vụ này, chương trình máy tính sẽ sàng lọc một tập hợp các sự kiện hoặc đối tượng và đánh dấu một số trong số đó là bất thường hoặc không điển hình. Một ví dụ về tác vụ phát hiện bất thường là phát hiện gian lận thẻ tín dụng. Bằng cách mô hình hóa thói quen mua sắm của bạn, công ty thẻ tín dụng có thể phát hiện việc sử dụng thẻ sai mục đích của bạn. Nếu một tên trộm đánh cắp thẻ tín dụng hoặc thông tin thẻ tín dụng của bạn, các giao dịch mua sắm của tên trộm thường sẽ đến từ một phân phối xác suất khác so với các loại giao dịch mua sắm của bạn. Công ty thẻ tín dụng có thể ngăn chặn gian lận bằng cách tạm ngưng tài khoản ngay khi thẻ đó được sử dụng cho một giao dịch không điển hình. Để biết thêm thông tin, xem *Anomaly Detection: A Survey* (Chandola và cộng sự, 2009, [22]) về khảo sát các phương pháp phát hiện bất thường.

- **Tổng hợp và lấy mẫu**: Trong loại tác vụ này, thuật toán học máy được yêu cầu tạo ra các ví dụ mới tương tự như các ví dụ trong dữ liệu huấn luyện. Việc tổng hợp và lấy mẫu thông qua học máy có thể hữu ích cho các ứng dụng truyền thông, nơi mà việc tạo ra số lượng lớn nội dung bằng tay có thể tốn kém hoặc nhàm chán đối với nghệ sĩ. Ví dụ, các trò chơi điện tử có thể tự động tạo ra các kết cấu cho các đối tượng hoặc cảnh quan lớn, thay vì yêu cầu nghệ sĩ phải gán nhãn từng điểm ảnh thủ công (*Texture Modeling with*

Convolutional Spike-and-Slab RBMs and Deep Extensions, Luo và cộng sự, 2013, [23]). Trong một số trường hợp, ta muốn quy trình lấy mẫu hoặc tổng hợp tạo ra một loại đầu ra cụ thể dựa trên đầu vào. Ví dụ, trong tác vụ tổng hợp giọng nói, chúng ta cung cấp một câu viết và yêu cầu chương trình tạo ra dạng sóng âm thanh chứa phiên bản nói của câu đó. Đây là một loại tác vụ đầu ra có cấu trúc, nhưng có thêm đặc điểm là không có một đầu ra đúng duy nhất cho mỗi đầu vào, và ta mong muốn đầu ra có sự đa dạng lớn để đầu ra trông tự nhiên và chân thực hơn.

- **Điền giá trị bị thiếu**: Trong loại tác vụ này, thuật toán học máy được cung cấp một ví dụ mới $\boldsymbol{x} \in \mathbb{R}^n$, nhưng với một số thành phần x_i của $\boldsymbol{x}$ bị thiếu. Thuật toán phải đưa ra dự đoán về các giá trị của các thành phần bị thiếu đó.

- **Khử nhiễu**: Trong loại tác vụ này, thuật toán học máy được cung cấp một *ví dụ đầu vào bị nhiễu* $\widetilde{\boldsymbol{x}} \in \mathbb{R}^n$, thu được từ một quá trình làm nhiễu không xác định từ *ví dụ gốc sạch* $\boldsymbol{x} \in \mathbb{R}^n$. Thuật toán phải dự đoán ví dụ sạch $\boldsymbol{x}$ từ phiên bản bị nhiễu $\widetilde{\boldsymbol{x}}$, hoặc nói chung là dự đoán phân phối xác suất có điều kiện $p\left(x \mid \widetilde{\boldsymbol{x}}\right)$.

- **Ước lượng mật độ** hoặc **ước lượng hàm trọng số xác suất**: Trong bài toán ước lượng mật độ, thuật toán học máy được yêu cầu học một hàm $p_{\text{model}} : \mathbb{R}^n \rightarrow \mathbb{R}$, trong đó $p_{\text{model}}(\boldsymbol{x})$ có thể được hiểu là hàm mật độ xác suất (nếu $\boldsymbol{x}$ là liên tục) hoặc hàm trọng số xác suất (nếu $\boldsymbol{x}$ là rời rạc) trên không gian mà các ví dụ được lấy ra. Để thực hiện tốt tác vụ này (chúng ta sẽ giải thích cụ thể điều đó nghĩa là gì khi thảo luận về các chỉ số đo lường hiệu suất P), thuật toán cần phải học được cấu trúc của dữ liệu mà nó đã thấy. Nó phải biết các điểm dữ liệu tập trung chặt chẽ ở đâu và đâu là nơi các điểm dữ liệu có khả năng không xuất hiện. Hầu hết các tác vụ được mô tả ở trên yêu cầu thuật toán học ít nhất phải ngầm nắm bắt cấu trúc của phân phối xác suất. Ước lượng mật độ cho phép ta nắm bắt phân phối đó một cách tường minh. Về nguyên tắc, ta có thể thực hiện các phép tính trên phân phối đó để giải quyết các tác vụ khác. Ví dụ, nếu ta đã thực hiện ước lượng mật độ để thu được phân phối xác suất $p(\boldsymbol{x})$, ta có thể sử dụng phân phối đó để giải quyết tác vụ bổ sung giá trị bị thiếu. Nếu một giá trị x_i bị thiếu và tất cả các giá trị khác, được ký hiệu là $\boldsymbol{x}_{-i}$, đã biết, thì ta biết rằng phân phối của nó được cho bởi $p\left(x_i \mid \boldsymbol{x}_{-i}\right)$. Trong thực tế, ước lượng mật độ không

phải lúc nào cũng cho phép ta giải quyết tất cả các tác vụ liên quan này, bởi vì trong nhiều trường hợp, các phép tính cần thiết trên $p(\boldsymbol{x})$ là không thể thực hiện được về mặt tính toán.

Tất nhiên, có thể còn rất nhiều tác vụ và loại tác vụ khác. Các loại tác vụ mà chúng tôi liệt kê ở đây chỉ nhằm cung cấp các ví dụ về những gì học máy có thể thực hiện, chứ không phải để xác định một hệ thống phân loại cứng nhắc về các tác vụ.

5.1.2 Thước đo hiệu suất P

Để đánh giá khả năng của một thuật toán học máy, ta cần thiết kế một thước đo định lượng cho hiệu suất của nó. Thông thường, thước đo hiệu suất P này được thiết kế đặc biệt cho tác vụ T mà hệ thống đang thực hiện.

Đối với các tác vụ như phân loại, phân loại với dữ liệu đầu vào bị thiếu, và phiên âm, ta thường đo lường **độ chính xác** của mô hình. Độ chính xác đơn giản là tỷ lệ các ví dụ mà mô hình dự đoán đầu ra đúng. Ta cũng có thể thu được thông tin tương đương bằng cách đo **tỷ lệ lỗi**, tức là tỷ lệ các ví dụ mà mô hình dự đoán đầu ra sai. Ta thường gọi tỷ lệ lỗi này là kỳ vọng tổn thất $0-1$. Tổn thất $0-1$ trên một ví dụ cụ thể là 0 nếu được phân loại đúng và là 1 nếu không đúng. Đối với các tác vụ như ước lượng mật độ, việc đo lường độ chính xác, tỷ lệ lỗi hay bất kỳ loại tổn thất $0-1$ nào khác là không hợp lý. Thay vào đó, ta phải sử dụng một thước đo hiệu suất khác cho mô hình một điểm số liên tục cho mỗi ví dụ. Cách tiếp cận phổ biến nhất là đưa ra trung bình các logit của xác suất mà mô hình gán cho mỗi ví dụ.

Thông thường, ta quan tâm đến việc thuật toán học máy hoạt động tốt như thế nào trên dữ liệu mà nó chưa từng thấy trước đây, vì điều này quyết định hiệu quả của nó khi triển khai trong thế giới thực. Do đó, ta đánh giá các thước đo hiệu suất này bằng cách sử dụng một **tập dữ liệu kiểm tra** riêng biệt với dữ liệu được dùng để huấn luyện hệ thống học máy.

Việc lựa chọn thước đo hiệu suất có thể có vẻ đơn giản và khách quan, nhưng thường rất khó để chọn một thước đo hiệu suất phù hợp với hành vi mong muốn của hệ thống.

Trong một số trường hợp, điều này xảy ra vì khó quyết định nên đo lường điều gì. Ví dụ, khi thực hiện một tác vụ phiên âm, ta nên đo lường độ chính xác của hệ thống khi phiên âm toàn bộ chuỗi, hay nên sử dụng một thước đo hiệu suất chi tiết hơn, cho điểm từng phần khi một số phần tử của chuỗi được phiên âm đúng? Khi thực hiện một tác vụ hồi quy, ta nên phạt hệ thống nhiều hơn nếu nó thường

xuyên mắc các lỗi có kích thước trung bình hay nếu nó hiếm khi mắc các lỗi rất lớn? Những lựa chọn thiết kế loại này phụ thuộc vào ứng dụng cụ thể.

Trong các trường hợp khác, ta biết số liệu lý tưởng mà mình muốn đo lường, nhưng việc đo lường nó là không khả thi. Ví dụ, điều này thường xảy ra trong bối cảnh ước lượng mật độ. Nhiều mô hình xác suất tốt nhất chỉ ngầm biểu diễn các phân phối xác suất. Việc tính giá trị xác suất thực tế được gán cho một điểm cụ thể trong không gian trong nhiều mô hình như vậy là không khả thi. Trong các trường hợp này, cần phải thiết kế một tiêu chí thay thế mà vẫn tương ứng với các mục tiêu thiết kế, hoặc thiết kế một cách xấp xỉ tốt cho tiêu chí mong muốn.

5.1.3 Trải nghiệm *E*

Các thuật toán học máy có thể được phân loại rộng rãi thành **không giám sát** hoặc **có giám sát** dựa trên loại trải nghiệm mà chúng được phép có trong quá trình học.

Hầu hết các thuật toán học trong cuốn sách này có thể được hiểu là được phép trải nghiệm toàn bộ một **tập dữ liệu**. Một tập dữ liệu là một tập hợp gồm nhiều ví dụ, như đã được định nghĩa trong Mục 5.1.1. Đôi khi ta cũng gọi các ví dụ là các **điểm dữ liệu**.

Một trong những tập dữ liệu lâu đời nhất được các nhà thống kê và nghiên cứu học máy nghiên cứu là tập dữ liệu Iris (*The Use of Multiple Measurements in Taxonomic Problems*, Fisher, 1936, [24]). Đây là một tập hợp các phép đo về các phần khác nhau của 150 cây hoa iris. Mỗi cây tương ứng với một ví dụ. Các đặc trưng trong mỗi ví dụ là các phép đo của từng phần của cây: chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa và chiều rộng cánh hoa. Tập dữ liệu cũng ghi lại loài của mỗi cây. Ba loài khác nhau được đại diện trong tập dữ liệu này.

Các **thuật toán học không giám sát** tiếp cận một tập dữ liệu chứa nhiều đặc trưng, sau đó học các thuộc tính hữu ích về cấu trúc của tập dữ liệu này. Trong bối cảnh học sâu, ta thường muốn học toàn bộ phân phối xác suất đã sinh ra tập dữ liệu, dù là một cách tường minh như trong ước lượng mật độ hay một cách ngầm định cho các tác vụ như tổng hợp hoặc loại bỏ nhiễu. Một số thuật toán học không giám sát khác thực hiện các vai trò khác, chẳng hạn như phân cụm, tức là chia tập dữ liệu thành các cụm gồm các ví dụ tương tự nhau.

Các **thuật toán học có giám sát** tiếp cận một tập dữ liệu chứa các đặc trưng, nhưng mỗi ví dụ còn đi kèm với một **nhãn** hoặc **mục tiêu**. Chẳng hạn, tập dữ liệu Iris được chú thích với loài của từng cây hoa iris. Một thuật toán học có giám sát có

thể nghiên cứu tập dữ liệu Iris và học cách phân loại các cây hoa iris thành ba loài khác nhau dựa trên các phép đo của chúng.

Nói một cách tổng quát, học không giám sát bao gồm việc quan sát nhiều ví dụ của một véctơ ngẫu nhiên $\boldsymbol{X}$ và cố gắng học một cách ngầm định hoặc tường minh phân phối xác suất $p(\boldsymbol{X})$, hoặc một số thuộc tính thú vị của phân phối đó. Trong khi đó, học có giám sát bao gồm việc quan sát nhiều ví dụ của một véctơ ngẫu nhiên $\boldsymbol{X}$ và một giá trị hoặc véctơ liên quan $\boldsymbol{y}$, và học cách dự đoán $\boldsymbol{y}$ từ $\boldsymbol{X}$, thường bằng cách ước lượng $p\left(\boldsymbol{y} \mid \boldsymbol{X}\right)$. Thuật ngữ **học có giám sát** bắt nguồn từ quan điểm rằng mục tiêu $\boldsymbol{y}$ được cung cấp bởi một người hướng dẫn hoặc giáo viên, người chỉ cho hệ thống học máy phải làm gì. Trong học không giám sát, không có người hướng dẫn hoặc giáo viên, và thuật toán phải tự học cách hiểu dữ liệu mà không có sự chỉ dẫn này.

Học không giám sát và học có giám sát không phải là các thuật ngữ được định nghĩa một cách chặt chẽ. Ranh giới giữa chúng thường không rõ ràng, và nhiều công nghệ học máy có thể được sử dụng cho cả hai loại tác vụ. Chẳng hạn, quy tắc nhân xác suất chỉ ra rằng đối với một véctơ ngẫu nhiên $\boldsymbol{X} \in \mathbb{R}^n$, phân phối đồng thời có thể được phân tích thành

$$p(\boldsymbol{X}) = \prod_{i=1}^{n} p\left(X_i \mid X_1, \ldots, X_{i-1}\right). \tag{5.1}$$

Phép phân tích này cho phép ta giải quyết bài toán học không giám sát, tức là mô hình hóa $p(\boldsymbol{X})$, bằng cách chia nó thành n bài toán học có giám sát. Ngược lại, ta có thể giải quyết bài toán học có giám sát, tức là học $p\left(y \mid \boldsymbol{X}\right)$, bằng cách sử dụng các công nghệ học không giám sát truyền thống để học phân phối đồng thời $p(\boldsymbol{X}, y)$ và suy luận

$$p\left(y \mid \boldsymbol{X}\right) = \frac{p(\boldsymbol{X}, y)}{\sum_{y'} p(\boldsymbol{X}, y')}. \tag{5.2}$$

Mặc dù học không giám sát và học có giám sát không phải là các khái niệm hoàn toàn chính thức hay tách biệt, nhưng chúng giúp phân loại một cách sơ lược một số công việc mà ta thực hiện với các thuật toán học máy. Theo truyền thống, người ta gọi các bài toán hồi quy, phân loại và đầu ra có cấu trúc là học có giám sát. Ước lượng mật độ để hỗ trợ các tác vụ khác thường được xem là học không giám sát.

Các biến thể khác của mô hình học cũng có thể tồn tại. Ví dụ, trong học bán giám sát, một số ví dụ có mục tiêu giám sát nhưng những ví dụ khác thì không. Trong học đa thể hiện, toàn bộ một tập hợp các ví dụ được gắn nhãn là có hoặc

không có một ví dụ của một lớp, nhưng các phần tử riêng lẻ trong tập hợp không được gắn nhãn. Để xem ví dụ gần đây về học đa thể hiện với các mô hình sâu, xem *From Group to Individual Labels Using Deep Features* (Kotzias và cộng sự, 2015, [25]).

Một số thuật toán học máy không chỉ trải nghiệm một tập dữ liệu cố định. Chẳng hạn, các thuật toán **học tăng cường** tương tác với một môi trường, tạo nên một vòng phản hồi giữa hệ thống học và các trải nghiệm của nó. Những thuật toán như vậy nằm ngoài phạm vi của cuốn sách này. Có thể tham khảo *Reinforcement Learning: An Introduction*, (Sutton và Barto, 1998, [26]) hoặc *Neuro-Dynamic Programming*, (Bertsekas và Tsitsiklis, 1996, [27]) để biết thêm thông tin về học tăng cường, và *Playing Atari with Deep Reinforcement Learning*, (Mnih và cộng sự, 2013, [28]) cho cách tiếp cận học sâu trong học tăng cường.

Một cách phổ biến để mô tả một tập dữ liệu là sử dụng **ma trận thiết kế**. Ma trận thiết kế là một ma trận chứa một ví dụ khác nhau trên mỗi hàng, và các cột của ma trận tương ứng với các đặc trưng khác nhau. Chẳng hạn, tập dữ liệu Iris có 150 ví dụ với bốn đặc trưng cho mỗi ví dụ. Điều này có nghĩa là ta có thể biểu diễn tập dữ liệu bằng một ma trận thiết kế $\boldsymbol{X} \in \mathbb{R}^{150\times 4}$, trong đó $X_{i,1}$ là chiều dài đài hoa của cây thứ i, $X_{i,2}$ là chiều rộng đài hoa của cây thứ i, v.v. Chúng tôi sẽ mô tả hầu hết các thuật toán học trong cuốn sách này dưới dạng cách chúng hoạt động trên các tập dữ liệu ma trận thiết kế.

Tất nhiên, để mô tả một tập dữ liệu như một ma trận thiết kế, cần phải có khả năng mô tả mỗi ví dụ dưới dạng một véctơ, và các véctơ này phải có cùng kích thước. Điều này không phải lúc nào cũng khả thi. Ví dụ, nếu bạn có một tập hợp các bức ảnh với các chiều rộng và chiều cao khác nhau, thì mỗi bức ảnh sẽ có số lượng điểm ảnh khác nhau, vì vậy không phải tất cả các bức ảnh đều có thể được mô tả bằng một véctơ có cùng độ dài. Mục 12.1 và Chương 13 sẽ mô tả cách xử lý các loại dữ liệu không đồng nhất như vậy. Trong các trường hợp như thế này, thay vì mô tả tập dữ liệu dưới dạng một ma trận với m hàng, ta sẽ mô tả nó như một tập hợp chứa m phần tử: $\left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)}\right\}$. Ký hiệu này không ngụ ý rằng bất kỳ hai véctơ ví dụ $\boldsymbol{x}^{(i)}$ và $\boldsymbol{x}^{(j)}$ nào cũng có cùng kích thước.

Trong trường hợp học có giám sát, ví dụ chứa một nhãn hoặc mục tiêu cùng với tập hợp các đặc trưng. Ví dụ, nếu ta muốn sử dụng một thuật toán học để nhận diện đối tượng từ các bức ảnh, ta cần chỉ định đối tượng nào xuất hiện trong mỗi bức ảnh. Chúng ta có thể làm điều này bằng một mã số, với 0 biểu thị người, 1 biểu thị ô tô, 2 biểu thị mèo, v.v. Thông thường, khi làm việc với một tập dữ liệu chứa ma trận thiết kế $\boldsymbol{X}$ của các quan sát đặc trưng, ta cũng cung cấp một véctơ nhãn

$\boldsymbol{Y}$, bao gồm các y_i cung cấp nhãn cho ví dụ thứ i.

Tất nhiên, đôi khi nhãn có thể nhiều hơn chỉ là một con số đơn lẻ. Ví dụ, nếu ta muốn huấn luyện một hệ thống nhận dạng giọng nói để phiên âm toàn bộ câu, thì nhãn cho mỗi câu ví dụ là một chuỗi các từ.

Cũng như không có định nghĩa chính thức cho học có giám sát và không giám sát, không có phân loại cứng nhắc nào cho các tập dữ liệu hoặc trải nghiệm. Các cấu trúc được mô tả ở đây bao quát hầu hết các trường hợp, nhưng luôn có khả năng thiết kế những cấu trúc mới cho các ứng dụng mới.

5.1.4 Ví dụ: hồi quy tuyến tính

Định nghĩa của chúng ta về một thuật toán học máy như một thuật toán có khả năng cải thiện hiệu suất của một chương trình máy tính trong một tác vụ nào đó thông qua trải nghiệm có phần trừu tượng. Để làm rõ hơn điều này, ta xét một ví dụ về một thuật toán học máy đơn giản: **hồi quy tuyến tính**. Ta sẽ quay lại ví dụ này nhiều lần khi giới thiệu thêm các khái niệm học máy giúp hiểu rõ hơn về hành vi của nó.

Như tên gọi của nó, hồi quy tuyến tính giải quyết một bài toán hồi quy. Nói cách khác, mục tiêu là xây dựng một hệ thống có thể nhận một véctơ $x \in \mathbb{R}^n$ làm đầu vào và dự đoán giá trị của một số vô hướng $y \in \mathbb{R}$ làm đầu ra. Trong trường hợp hồi quy tuyến tính, đầu ra là một hàm tuyến tính của đầu vào. Gọi $\widehat{y}$ là giá trị mà mô hình của ta dự đoán cho y. Ta định nghĩa đầu ra là

$$\widehat{y} = \boldsymbol{w}^T \boldsymbol{x} \tag{5.3}$$

trong đó $\boldsymbol{w} \in \mathbb{R}^n$ là một véctơ các tham số.

Tham số là các giá trị điều khiển hành vi của hệ thống. Trong trường hợp này, w_i là hệ số mà ta nhân với đặc trưng x_i trước khi cộng đóng góp từ tất cả các đặc trưng. Ta có thể xem $\boldsymbol{w}$ như một tập hợp các trọng số xác định cách mỗi đặc trưng ảnh hưởng đến dự đoán. Nếu một đặc trưng x_i nhận trọng số dương w_i, thì việc tăng giá trị của đặc trưng đó làm tăng giá trị của dự đoán $\widehat{y}$. Nếu một đặc trưng nhận trọng số âm, thì việc tăng giá trị của đặc trưng đó làm giảm giá trị của dự đoán. Nếu trọng số của một đặc trưng có độ lớn lớn, thì nó có ảnh hưởng lớn đến dự đoán. Nếu trọng số của một đặc trưng bằng không, thì nó không ảnh hưởng đến dự đoán.

Như vậy, ta có một định nghĩa cho tác vụ T: dự đoán y từ $\boldsymbol{x}$ bằng cách xuất ra $\widehat{y} = \boldsymbol{w}^T \boldsymbol{x}$. Tiếp theo, ta cần một định nghĩa cho thước đo hiệu suất, P.

Giả sử ta có một ma trận thiết kế gồm m ví dụ đầu vào mà ta sẽ không sử dụng để huấn luyện, chỉ để đánh giá mức độ hiệu quả của mô hình. Chúng ta cũng có một véctơ các mục tiêu hồi quy cung cấp giá trị đúng của y cho mỗi ví dụ này. Vì tập dữ liệu này chỉ được dùng để đánh giá, ta gọi nó là **tập kiểm tra**. Ta gọi ma trận thiết kế của các đầu vào là $\boldsymbol{X}^{(\text{test})}$ và véctơ các mục tiêu hồi quy là $\boldsymbol{Y}^{(\text{test})}$.

Một cách để đo lường hiệu suất của mô hình là tính **sai số bình phương trung bình** của mô hình trên tập kiểm tra. Nếu $\widehat{\boldsymbol{Y}}^{(\text{test})}$ gồm các dự đoán $\widehat{y}_i^{(\text{test})}$ của mô hình trên tập kiểm tra, thì sai số bình phương trung bình được cho bởi

$$\text{MSE}_{\text{test}} = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{y}_i^{(\text{test})} - y_i^{((\text{test}))}\right)^2. \tag{5.4}$$

Về trực quan, có thể thấy rằng thước đo sai số này giảm xuống 0 khi $\widehat{\boldsymbol{Y}}^{(\text{test})} = \boldsymbol{Y}^{(\text{test})}$. Ta cũng có thể thấy rằng

$$\text{MSE}_{\text{test}} = \frac{1}{m}\left\|\widehat{\boldsymbol{Y}}^{(\text{test})} - \boldsymbol{Y}^{(\text{test})}\right\|_2^2, \tag{5.5}$$

nên sai số tăng lên khi khoảng cách Euclid giữa các dự đoán và các mục tiêu tăng lên.

Để tạo ra một thuật toán học máy, ta cần thiết kế một thuật toán có khả năng cải thiện các trọng số $\boldsymbol{w}$ theo cách làm giảm MSE_{test} khi thuật toán được phép tích lũy kinh nghiệm bằng cách quan sát một tập huấn luyện $\left(\boldsymbol{X}^{(\text{train})}, \boldsymbol{Y}^{(\text{train})}\right)$. Một cách trực quan để làm điều này (sẽ được giải thích trong Mục 5.5.1) là chỉ cần cực tiểu hóa sai số bình phương trung bình trên tập huấn luyện, $\text{MSE}_{\text{train}}$.

Để cực tiểu hóa $\text{MSE}_{\text{train}}$, ta chỉ cần giải phương trình tại đó gradient của nó bằng $\boldsymbol{0}$:

$$\nabla_{\boldsymbol{w}}\text{MSE}_{\text{train}} = \boldsymbol{0} \tag{5.6}$$

$$\Rightarrow \nabla_{\boldsymbol{w}}\frac{1}{m}\left\|\widehat{\boldsymbol{Y}}^{(\text{train})} - \boldsymbol{Y}^{(\text{train})}\right\|_2^2 = \boldsymbol{0} \tag{5.7}$$

$$\Rightarrow \frac{1}{m}\nabla_{\boldsymbol{w}}\left\|\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{Y}^{(\text{train})}\right\|_2^2 = \boldsymbol{0} \tag{5.8}$$

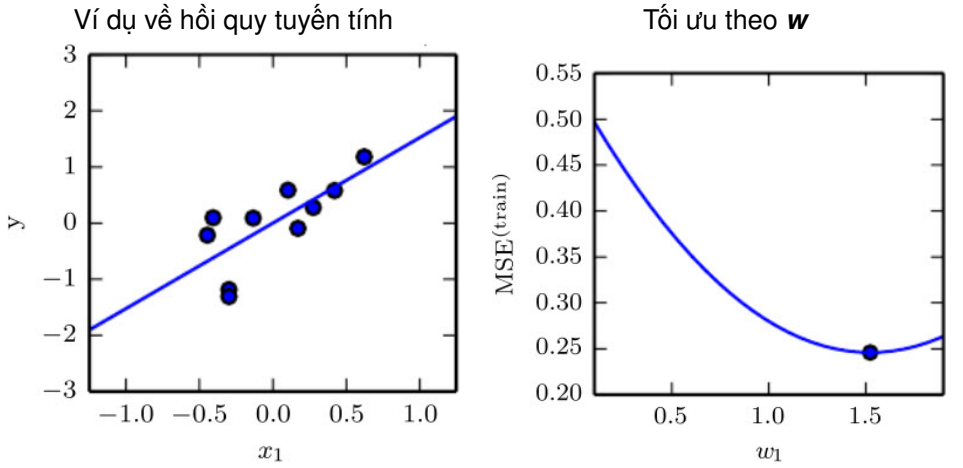
$$\Rightarrow \nabla_{\boldsymbol{w}}\left(\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{Y}^{(\text{train})}\right)^T\left(\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{Y}^{(\text{train})}\right) = \boldsymbol{0} \tag{5.9}$$

$$\Rightarrow \nabla_{\boldsymbol{w}}\left(\boldsymbol{w}^T\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{Y}^{(\text{train})} + \boldsymbol{Y}^{(\text{train})\,T}\boldsymbol{Y}^{(\text{train})}\right) = \boldsymbol{0} \tag{5.10}$$

$$\Rightarrow 2\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{Y}^{(\text{train})} = \boldsymbol{0} \tag{5.11}$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{X}^{(\text{train})}\right)^{-1}\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{Y}^{(\text{train})} \tag{5.12}$$

Hệ phương trình có nghiệm được cho bởi phương trình (5.12) được gọi là **phương trình chuẩn tắc**. Việc tính toán phương trình (5.12) tạo nên một thuật toán học đơn giản. Để thấy ví dụ về cách thuật toán học hồi quy tuyến tính hoạt động, hãy xem Hình 5.1.



Hình 5.1: Bài toán hồi quy tuyến tính với tập huấn luyện gồm 10 điểm dữ liệu, mỗi điểm chứa một đặc trưng. Vì chỉ có một đặc trưng, véctơ trọng số $\boldsymbol{w}$ chỉ chứa một tham số cần học, w_1. *Hình trái*: Quan sát thấy rằng hồi quy tuyến tính học cách đặt w_1 sao cho đường $y = w_1 x$ gần như đi qua tất cả các điểm trong tập huấn luyện. *Hình phải*: Điểm được vẽ cho thấy giá trị của w_1 tìm được bằng cách giải phương trình chuẩn tắc, mà ta có thể thấy là cực tiểu hóa sai số bình phương trung bình trên tập huấn luyện.

Đáng chú ý là thuật ngữ "hồi quy tuyến tính" thường được sử dụng để chỉ một mô hình phức tạp hơn một chút với một tham số bổ sung — một hằng số chặn b. Trong mô hình này

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b \tag{5.13}$$

do đó ánh xạ từ tham số tới dự đoán vẫn là một hàm tuyến tính, nhưng ánh xạ từ các đặc trưng tới dự đoán giờ đây là một hàm affine. Sự mở rộng này sang hàm affine có nghĩa là đồ thị của các dự đoán của mô hình vẫn trông giống một đường thẳng, nhưng không nhất thiết phải đi qua gốc tọa độ. Thay vì thêm tham số chặn b, ta có thể tiếp tục sử dụng mô hình chỉ với các trọng số nhưng mở rộng $\boldsymbol{x}$ với một thành phần bổ sung luôn được đặt bằng 1. Trọng số tương ứng với thành phần bổ sung này đóng vai trò của tham số chặn. Trong suốt cuốn sách này, ta sẽ thường xuyên sử dụng thuật ngữ "tuyến tính" khi đề cập đến các hàm affine.

Hằng số chặn b thường được gọi là tham số **chệch** của phép biến đổi affine. Thuật ngữ này bắt nguồn từ quan điểm cho rằng đầu ra của phép biến đổi có xu hướng hướng về b khi không có đầu vào nào. Thuật ngữ này khác với khái niệm độ chệch trong thống kê, trong đó ước lượng kỳ vọng của một thuật toán ước lượng thống kê cho một đại lượng không bằng với giá trị thực của đại lượng đó.

Hồi quy tuyến tính tất nhiên là một thuật toán học cực kỳ đơn giản và hạn chế, nhưng nó cung cấp một ví dụ về cách một thuật toán học có thể hoạt động. Trong các phần tiếp theo, chúng tôi sẽ mô tả một số nguyên tắc cơ bản làm nền tảng cho thiết kế thuật toán học và trình bày cách những nguyên tắc này có thể được sử dụng để xây dựng các thuật toán học phức tạp hơn.

5.2 Năng lực mô hình, quá khớp và chưa khớp

Thách thức trung tâm trong học máy là chúng ta phải hoạt động tốt trên các đầu vào *mới*, *chưa từng thấy trước đây* — không chỉ trên những đầu vào mà mô hình đã được huấn luyện. Khả năng hoạt động tốt trên các đầu vào chưa được quan sát trước đó được gọi là **tổng quát hóa**.

Thông thường, khi huấn luyện một mô hình học máy, ta có sẵn một tập huấn luyện, có thể tính toán một thước đo sai số trên tập huấn luyện gọi là **sai số huấn luyện**, và ta giảm thiểu sai số huấn luyện này. Cho đến nay, những gì ta đã mô tả đơn giản chỉ là một bài toán tối ưu. Điều phân biệt học máy với tối ưu là ta muốn **sai số tổng quát hóa**, còn gọi là **sai số kiểm tra**, cũng thấp. Sai số tổng quát hóa được định nghĩa là giá trị kỳ vọng của sai số trên một đầu vào mới. Ở đây, kỳ vọng được tính trên các đầu vào có thể có, được rút ra từ phân phối của các đầu vào mà ta kỳ vọng hệ thống sẽ gặp trong thực tế.

Ta thường ước lượng sai số tổng quát hóa của một mô hình học máy bằng cách đo lường hiệu suất của nó trên một **tập kiểm tra** gồm các ví dụ được thu thập tách biệt khỏi tập huấn luyện.

Trong ví dụ hồi quy tuyến tính, ta huấn luyện mô hình bằng cách cực tiểu hóa sai số huấn luyện,

$$\frac{1}{m^{(\text{train})}} \left\| \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{Y}^{(\text{train})} \right\|_2^2, \tag{5.14}$$

nhưng điều ta thực sự quan tâm là sai số kiểm tra, $\frac{1}{m^{(\text{test})}} \left\| \boldsymbol{X}^{(\text{test})} \boldsymbol{w} - \boldsymbol{Y}^{(\text{test})} \right\|_2^2$.

Làm thế nào chúng ta có thể ảnh hưởng đến hiệu suất trên tập kiểm tra khi chỉ có thể quan sát tập huấn luyện? **Lý thuyết học thống kê** cung cấp một số câu trả lời. Nếu tập huấn luyện và tập kiểm tra được thu thập một cách tùy ý, thực sự có

rất ít điều ta có thể làm. Nếu ta được phép đưa ra một số giả định về cách thu thập tập huấn luyện và tập kiểm tra, thì ta có thể đạt được một số tiến bộ.

Dữ liệu huấn luyện và dữ liệu kiểm tra được tạo ra bởi một phân phối xác suất trên các tập dữ liệu gọi là **quá trình sinh dữ liệu**. Ta thường đưa ra một tập hợp các giả định được gọi chung là các **giả định độc lập cùng phân phối**. Các giả định này nghĩa là các ví dụ trong mỗi tập dữ liệu **độc lập** với nhau và tập huấn luyện và tập kiểm tra có **phân phối giống hệt nhau**, được rút ra từ cùng một phân phối xác suất. Giả định này cho phép ta mô tả quá trình sinh dữ liệu bằng một phân phối xác suất trên một ví dụ đơn lẻ. Sau đó, cùng một phân phối được sử dụng để tạo ra mọi ví dụ trong tập huấn luyện và mọi ví dụ trong tập kiểm tra. Ta gọi phân phối chung cơ bản này là **phân phối sinh dữ liệu**, ký hiệu là p_{data}. Khung xác suất này và các giả định độc lập cùng phân phối cho phép ta nghiên cứu mối quan hệ giữa sai số huấn luyện và sai số kiểm tra một cách toán học.

Một mối liên hệ trực tiếp mà ta có thể quan sát được giữa sai số huấn luyện và sai số kiểm tra là kỳ vọng sai số huấn luyện của một mô hình được chọn ngẫu nhiên bằng với kỳ vọng sai số kiểm tra của mô hình đó. Giả sử ta có một phân phối xác suất $p(\boldsymbol{x}, y)$ và ta lấy mẫu từ đó nhiều lần để tạo ra tập huấn luyện và tập kiểm tra. Với một giá trị cố định $\boldsymbol{w}$, kỳ vọng sai số trên tập huấn luyện chính xác bằng với kỳ vọng sai số trên tập kiểm tra, vì cả hai kỳ vọng đều được hình thành bằng cách sử dụng cùng một quá trình lấy mẫu tập dữ liệu. Sự khác biệt duy nhất giữa hai điều kiện này là tên mà ta gán cho tập dữ liệu đã được lấy mẫu.

Tuy nhiên, khi sử dụng một thuật toán học máy, ta không cố định các tham số trước, sau đó mới lấy mẫu cả hai tập dữ liệu. Thay vào đó, ta lấy mẫu tập huấn luyện, sau đó sử dụng nó để chọn các tham số nhằm giảm sai số trên tập huấn luyện, rồi mới lấy mẫu tập kiểm tra. Theo quá trình này, kỳ vọng sai số kiểm tra sẽ lớn hơn hoặc bằng với kỳ vọng sai số huấn luyện. Các yếu tố quyết định mức độ hoạt động của một thuật toán học máy bao gồm khả năng của nó trong việc:

1. Làm cho sai số trên tập huấn luyện nhỏ.

2. Làm cho khoảng cách giữa sai số trên tập huấn luyện và tập kiểm tra nhỏ.

Hai yếu tố này tương ứng với hai thách thức chính trong học máy: **chưa khớp** và **quá khớp**. Chưa khớp xảy ra khi mô hình không thể đạt được một giá trị sai số đủ nhỏ trên tập huấn luyện. Quá khớp xảy ra khi khoảng cách giữa sai số huấn luyện và sai số kiểm tra quá lớn.

Ta có thể kiểm soát khả năng mô hình có bị quá khớp hay chưa khớp bằng cách thay đổi **năng lực** của nó. Một cách xúc tích, năng lực của một mô hình là

khả năng của nó trong việc khớp với nhiều loại hàm khác nhau. Các mô hình có năng lực thấp có thể gặp khó khăn trong việc khớp với tập huấn luyện. Các mô hình có năng lực cao có thể quá khớp do ghi nhớ các thuộc tính của tập huấn luyện không có ích cho tập kiểm tra.

Một cách để kiểm soát năng lực của một thuật toán học là lựa chọn **không gian giả thuyết** của nó, tức là tập hợp các hàm mà thuật toán học được phép chọn làm giải pháp. Ví dụ, thuật toán hồi quy tuyến tính có tập hợp tất cả các hàm tuyến tính của đầu vào là không gian giả thuyết của nó. Chúng ta có thể tổng quát hóa hồi quy tuyến tính để bao gồm cả các hàm đa thức, chứ không chỉ là các hàm tuyến tính, trong không gian giả thuyết của nó. Làm như vậy sẽ tăng năng lực của mô hình.

Một đa thức bậc nhất cho ta mô hình hồi quy tuyến tính mà ta đã quen thuộc, với dự đoán

$$\widehat{y} = b + wx. \tag{5.15}$$

Bằng cách đưa x^2 làm một đặc trưng khác vào mô hình hồi quy tuyến tính này, ta có thể học một mô hình là hàm bậc hai của x:

$$\widehat{y} = b + w_1 x + w_2 x^2. \tag{5.16}$$

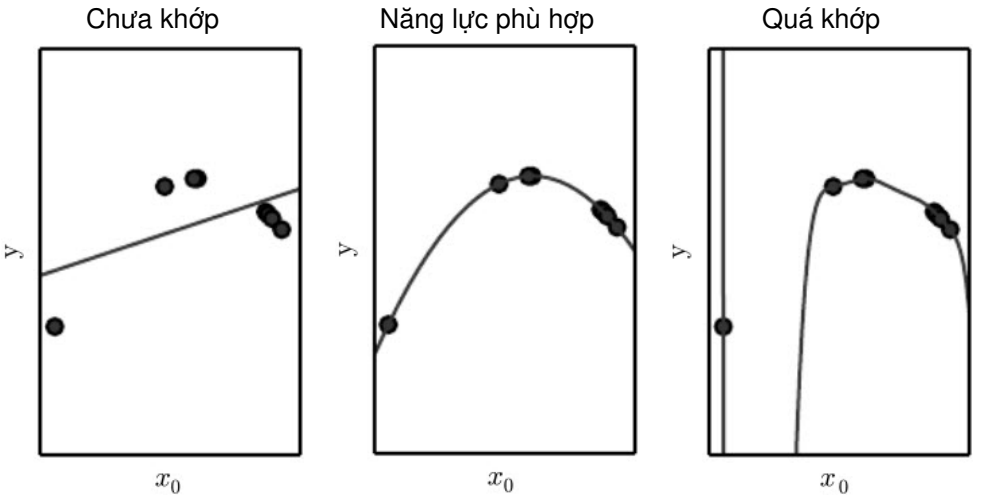
Mặc dù mô hình này triển khai một hàm bậc hai của *đầu vào*, đầu ra vẫn là hàm tuyến tính của *các tham số*, vì vậy ta vẫn có thể sử dụng phương trình chuẩn tắc để huấn luyện mô hình trong dạng đóng. Ta có thể tiếp tục thêm các lũy thừa cao hơn của x làm các đặc trưng bổ sung, chẳng hạn để có một đa thức bậc 9:

$$\widehat{y} = b + \sum_{i=1}^{9} w_i x^i. \tag{5.17}$$

Các thuật toán học máy sẽ hoạt động tốt nhất khi năng lực của chúng phù hợp với độ phức tạp thực sự của tác vụ cần thực hiện và lượng dữ liệu huấn luyện mà chúng được cung cấp. Các mô hình có năng lực không đủ sẽ không thể giải quyết các tác vụ phức tạp. Các mô hình có năng lực cao có thể giải quyết các tác vụ phức tạp, nhưng khi năng lực của chúng vượt quá mức cần thiết để giải quyết tác vụ hiện tại, chúng có thể bị quá khớp.

Hình 5.2 minh họa nguyên tắc này trong thực tế. Ta so sánh một mô hình dự đoán tuyến tính, bậc hai và bậc 9 trong việc cố gắng khớp một bài toán mà hàm cơ bản thực sự là bậc hai. Hàm tuyến tính không thể nắm bắt được độ cong trong bài toán thực sự, do đó nó chưa khớp. Mô hình bậc 9 có khả năng biểu diễn hàm

đúng, nhưng nó cũng có thể biểu diễn vô số hàm khác đi qua chính xác các điểm huấn luyện, vì ta có nhiều tham số hơn số lượng ví dụ huấn luyện. Khi có quá nhiều lời giải khác nhau như vậy, khả năng chọn một lời giải tổng quát tốt là rất ít. Trong ví dụ này, mô hình bậc hai phù hợp hoàn hảo với cấu trúc thực sự của tác vụ, do đó nó tổng quát hóa tốt với dữ liệu mới.



Hình 5.2: Chúng ta khớp ba mô hình với tập huấn luyện ví dụ này. Dữ liệu huấn luyện được tạo ra tổng hợp bằng cách lấy mẫu ngẫu nhiên các giá trị x và chọn y theo cách xác định bằng cách đánh giá một hàm bậc hai. *Hình trái*: Một hàm tuyến tính khớp với dữ liệu gặp tình trạng chưa khớp — nó không thể nắm bắt được độ cong có trong dữ liệu. *Hình giữa*: Một hàm bậc hai khớp với dữ liệu tổng quát hóa tốt với các điểm chưa thấy trước. Nó không gặp phải tình trạng đáng kể của sự quá khớp hoặc chưa khớp. *Hình phải*: Một đa thức bậc 9 khớp với dữ liệu gặp tình trạng quá khớp. Ở đây, ta đã sử dụng ma trận giả nghịch đảo Moore – Penrose để giải phương trình chuẩn tắc không xác định. Nghiệm đi qua chính xác tất cả các điểm huấn luyện, nhưng không may mắn khi nó không trích xuất đúng cấu trúc thực sự. Nó xuất hiện một vùng trũng sâu giữa hai điểm huấn luyện, điều này không xuất hiện trong hàm thực sự. Nó cũng tăng mạnh ở phía bên trái của dữ liệu, trong khi hàm thực sự giảm ở khu vực này.

Cho đến nay, ta mới chỉ mô tả một cách duy nhất để thay đổi năng lực của một mô hình: bằng cách thay đổi số lượng đặc trưng đầu vào mà mô hình có, đồng thời thêm các tham số mới liên quan đến các đặc trưng đó. Trên thực tế, có nhiều cách để thay đổi năng lực của một mô hình. Năng lực không chỉ được xác định bởi lựa chọn mô hình. Mô hình chỉ định họ hàm mà thuật toán học có thể chọn khi thay

đổi các tham số nhằm giảm một mục tiêu huấn luyện. Điều này được gọi là **năng lực biểu diễn** của mô hình. Trong nhiều trường hợp, việc tìm hàm tốt nhất trong họ hàm này là một bài toán tối ưu rất khó. Trong thực tế, thuật toán học không thực sự tìm thấy hàm tốt nhất, mà chỉ tìm được một hàm giảm đáng kể sai số huấn luyện. Các hạn chế bổ sung này, chẳng hạn như sự không hoàn hảo của thuật toán tối ưu, khiến **năng lực thực sự** của thuật toán học có thể nhỏ hơn năng lực biểu diễn của họ mô hình.

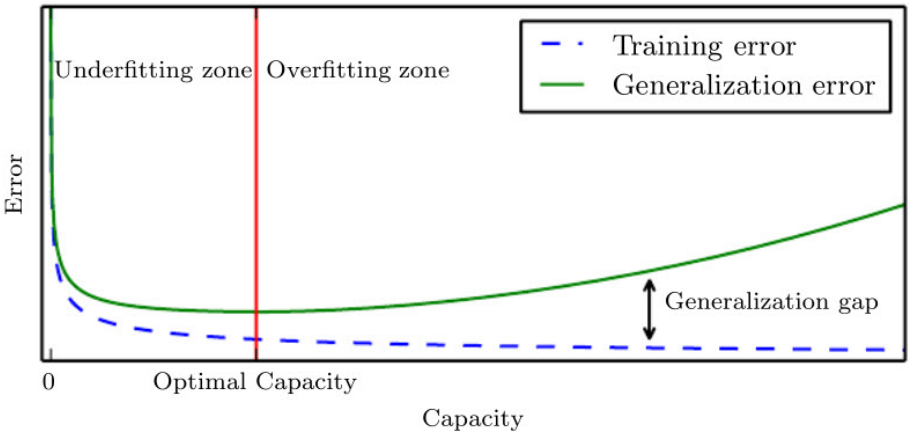
Các ý tưởng hiện đại của chúng ta về việc cải thiện khả năng tổng quát hóa của các mô hình học máy là sự tinh chỉnh của những suy nghĩ có từ thời các triết gia, ít nhất là từ Ptolemy. Nhiều học giả từ thời kỳ đầu đã nhắc đến một nguyên tắc tiết kiệm mà ngày nay được biết đến rộng rãi nhất dưới tên gọi **dao cạo Occam** (khoảng 1287 – 1347). Nguyên tắc này cho rằng, giữa các giả thuyết cạnh tranh có khả năng giải thích các quan sát đã biết một cách ngang nhau, ta nên chọn giả thuyết "đơn giản" nhất. Ý tưởng này đã được các nhà sáng lập lý thuyết học thống kê chính thức hóa và làm cho cụ thể hơn vào thế kỷ 20 (*On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, Vapnik và Chervonenkis, 1971, [29]; *Estimation of Dependences Based on Empirical Data*, Vapnik, 2010, [30]; Blumer*Learnability and the Vapnik-Chervonenkis dimension* và cộng sự, 1989, [31]; *The Nature of Statistical Learning Theory*, Vapnik, 2010, [32]).

Lý thuyết học thống kê cung cấp nhiều phương pháp để định lượng năng lực của mô hình. Trong số đó, nổi tiếng nhất là **chiều Vapnik – Chervonenkis**, hay chiều VC. Chiều VC đo lường năng lực của một bộ phân loại nhị phân. Chiều VC được định nghĩa là giá trị lớn nhất có thể của m mà tại đó tồn tại một tập huấn luyện gồm m điểm $\boldsymbol{x}$ khác nhau mà bộ phân loại có thể gán nhãn một cách tùy ý.

Việc định lượng năng lực của mô hình cho phép lý thuyết học thống kê đưa ra các dự đoán định lượng. Các kết quả quan trọng nhất trong lý thuyết học thống kê cho thấy rằng chênh lệch giữa sai số huấn luyện và sai số tổng quát hóa bị chặn trên bởi một đại lượng, mà đại lượng này tăng khi năng lực mô hình tăng nhưng giảm khi số lượng mẫu huấn luyện tăng lên (*On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, Vapnik và Chervonenkis, 1971, [29]; *Estimation of Dependences Based on Empirical Data*, Vapnik, 2010, [30]; Blumer*Learnability and the Vapnik-Chervonenkis dimension* và cộng sự, 1989, [31]; *The Nature of Statistical Learning Theory*, Vapnik, 2010, [32]). Những cận trên này cung cấp cơ sở lý thuyết để chứng minh rằng các thuật toán học máy có thể hoạt động, nhưng chúng hiếm khi được sử dụng trong thực tế khi làm việc với các thuật toán học sâu. Điều này một phần là do các cận trên này thường khá lỏng lẻo

và một phần vì việc xác định năng lực của các thuật toán học sâu có thể rất khó khăn. Vấn đề xác định năng lực của một mô hình học sâu đặc biệt khó khăn vì năng lực thực sự bị giới hạn bởi khả năng của thuật toán tối ưu, và chúng ta có ít hiểu biết lý thuyết về các bài toán tối ưu không lồi rất tổng quát liên quan đến học sâu.

Cần nhớ rằng mặc dù các hàm đơn giản hơn có xu hướng tổng quát hóa tốt hơn (có khoảng cách nhỏ giữa sai số huấn luyện và sai số kiểm tra), ta vẫn cần chọn một giả thuyết đủ phức tạp để đạt được sai số huấn luyện thấp. Thông thường, sai số huấn luyện giảm dần cho đến khi tiệm cận giá trị sai số nhỏ nhất có thể khi năng lực mô hình tăng lên (giả sử thước đo sai số có giá trị nhỏ nhất). Thông thường, sai số tổng quát hóa có dạng đường cong hình chữ U như hàm của năng lực mô hình. Điều này được minh họa trong Hình 5.3.



Hình 5.3: Mối quan hệ điển hình giữa năng lực mô hình và sai số. Sai số huấn luyện và sai số kiểm tra có hành vi khác nhau. Ở phía bên trái của đồ thị, cả sai số huấn luyện và sai số tổng quát hóa đều cao. Đây là **giai đoạn chưa khớp**. Khi tăng năng lực mô hình, sai số huấn luyện giảm, nhưng khoảng cách giữa sai số huấn luyện và sai số tổng quát hóa tăng lên. Cuối cùng, độ lớn của khoảng cách này lớn hơn mức giảm của sai số huấn luyện, và ta bước vào **giai đoạn quá khớp**, nơi năng lực mô hình quá lớn, vượt quá **năng lực tối ưu**.

Để đạt đến trường hợp cực đoan nhất của năng lực mô hình cao tùy ý, chúng tôi giới thiệu khái niệm mô hình **phi tham số**. Cho đến nay, ta chỉ thấy các mô hình có tham số, chẳng hạn như hồi quy tuyến tính. Mô hình có tham số học một hàm được mô tả bởi một véctơ tham số có kích thước hữu hạn và được cố định trước khi quan sát bất kỳ dữ liệu nào. Ngược lại, mô hình phi tham số không có hạn chế như vậy.

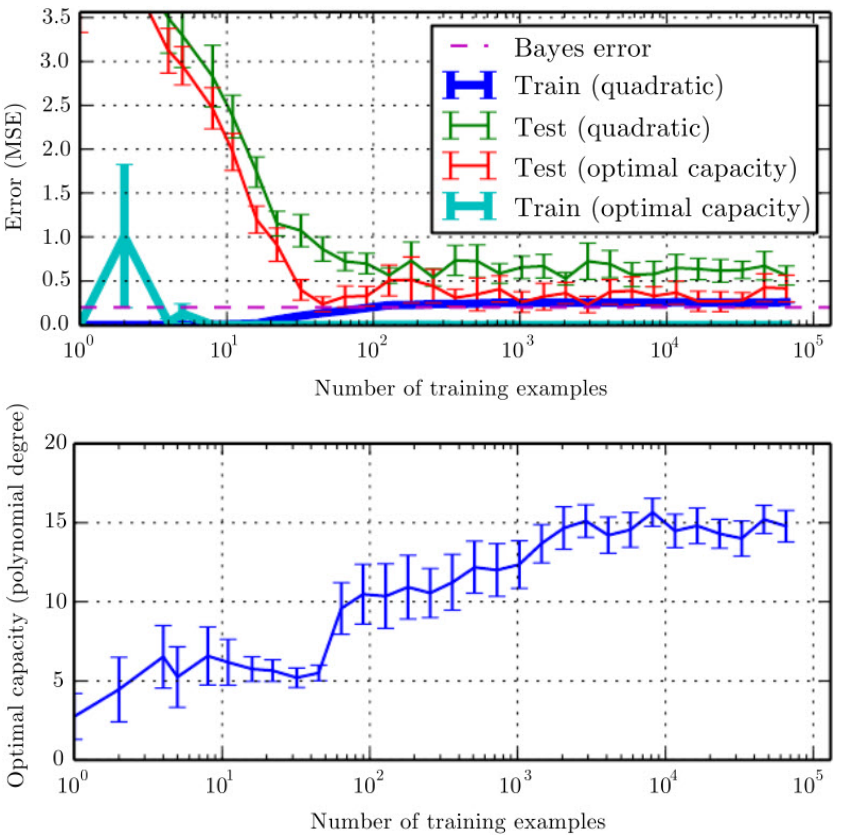
Đôi khi, các mô hình phi tham số chỉ là các trừu tượng lý thuyết (chẳng hạn như một thuật toán tìm kiếm qua tất cả các phân phối xác suất có thể có) và không thể triển khai trong thực tế. Tuy nhiên, ta cũng có thể thiết kế các mô hình phi tham số thực tiễn bằng cách làm cho độ phức tạp của chúng là một hàm phụ thuộc vào kích thước của tập huấn luyện. Một ví dụ về thuật toán như vậy là **hồi quy láng giềng gần nhất**. Không giống như hồi quy tuyến tính, vốn có véctơ trọng số có độ dài cố định, mô hình hồi quy láng giềng gần nhất chỉ cần lưu trữ các giá trị $\boldsymbol{X}$ và $\boldsymbol{Y}$ từ tập huấn luyện. Khi được yêu cầu dự đoán điểm kiểm tra $\boldsymbol{x}$, mô hình sẽ tìm điểm gần nhất trong tập huấn luyện và trả về mục tiêu hồi quy tương ứng. Nói cách khác, $\widehat{y} = y_i$ với $i = \arg\min \left\| \boldsymbol{X}_{i,:} - \boldsymbol{x} \right\|_2^2$. Thuật toán này cũng có thể được tổng quát hóa sang các loại khoảng cách khác ngoài chuẩn L^2, chẳng hạn như các khoảng cách được học (*Neighbourhood Components Analysis*, Goldberger và cộng sự, 2004, [33]). Nếu thuật toán được phép phá vỡ thế bế tắc bằng cách lấy trung bình các giá trị y_i cho tất cả các $\boldsymbol{X}_{i,:}$ có cùng khoảng cách gần nhất, thì thuật toán này có thể đạt được sai số huấn luyện tối thiểu có thể (có thể lớn hơn không, nếu hai đầu vào giống hệt nhau có các đầu ra khác nhau) trên bất kỳ tập dữ liệu hồi quy nào.

Cuối cùng, chúng ta cũng có thể tạo ra một thuật toán học phi tham số bằng cách lồng một thuật toán học có tham số vào bên trong một thuật toán khác để tăng số lượng tham số khi cần thiết. Chẳng hạn, ta có thể tưởng tượng một vòng lặp bên ngoài của quá trình học làm thay đổi bậc của đa thức được học bởi hồi quy tuyến tính áp dụng trên một phép mở rộng đa thức của đầu vào.

Mô hình lý tưởng là một "nhà tiên tri" có thể biết chính xác phân phối xác suất thực sự sinh ra dữ liệu. Tuy nhiên, ngay cả một mô hình như vậy vẫn có thể gặp sai số trong nhiều bài toán, vì có thể tồn tại nhiễu trong phân phối. Trong trường hợp học có giám sát, ánh xạ tương ứng $\boldsymbol{x}$ với y có thể vốn là ngẫu nhiên, hoặc y có thể là một hàm xác định nhưng phụ thuộc vào các biến khác ngoài những biến đã bao gồm trong $\boldsymbol{x}$. Sai số gặp phải khi một mô hình lý tưởng đưa ra các dự đoán từ phân phối thực $p(\boldsymbol{x}, y)$ được gọi là **sai số Bayes**.

Sai số huấn luyện và sai số tổng quát hóa thay đổi khi kích thước của tập huấn luyện thay đổi. Sai số tổng quát hóa kỳ vọng không bao giờ tăng khi số lượng ví dụ huấn luyện tăng. Đối với các mô hình phi tham số, thêm dữ liệu sẽ giúp tổng quát hóa tốt hơn cho đến khi đạt được sai số tốt nhất có thể. Bất kỳ mô hình tham số cố định nào có năng lực thấp hơn năng lực tối ưu sẽ hội tụ đến một giá trị sai số vượt quá sai số Bayes. Xem Hình 5.4 để minh họa. Lưu ý rằng mô hình có thể có năng lực tối ưu nhưng vẫn có khoảng cách lớn giữa sai số huấn luyện và sai số tổng quát hóa. Trong tình huống này, ta có thể giảm khoảng cách này bằng cách

thu thập thêm các ví dụ huấn luyện.



Hình 5.4: Ảnh hưởng của kích thước tập dữ liệu huấn luyện lên sai số huấn luyện và sai số kiểm tra, cũng như năng lực tối ưu của mô hình. Chúng tôi xây dựng một bài toán hồi quy tổng hợp bằng cách thêm một lượng nhiễu vừa phải vào đa thức bậc 5, tạo ra một tập kiểm tra cố định, sau đó tạo ra các tập huấn luyện với nhiều kích thước khác nhau. Đối với mỗi kích thước, chúng tôi tạo ra 40 tập huấn luyện khác nhau để vẽ các thanh sai số thể hiện khoảng tin cậy 95%. *Hình trên*: Sai số bình phương trung bình trên tập huấn luyện và tập kiểm tra của hai mô hình khác nhau: một mô hình bậc hai và một mô hình có bậc được chọn để cực tiểu hóa sai số kiểm tra. Cả hai đều được khớp ở dạng đóng. Đối với mô hình bậc hai, sai số huấn luyện tăng khi kích thước tập huấn luyện tăng do tập dữ liệu lớn hơn khó khớp hơn. Đồng thời, sai số kiểm tra giảm vì ít giả thuyết sai hơn nhất quán với dữ liệu huấn luyện. Mô hình bậc hai không có đủ năng lực để giải quyết tác vụ, vì vậy sai số kiểm tra của nó hội tụ về một giá trị cao. Sai số kiểm tra tại năng lực tối ưu hội tụ về sai số Bayes. Sai số huấn luyện có thể thấp hơn sai số Bayes do khả năng của thuật toán huấn luyện ghi nhớ các trường hợp cụ thể của tập huấn luyện. Khi kích thước tập huấn luyện tăng đến vô hạn, sai số huấn luyện của bất kỳ mô hình có năng lực cố định nào (ở đây là mô hình bậc hai) phải tăng lên ít nhất bằng sai số Bayes. *Hình dưới*: Khi kích thước tập huấn luyện tăng lên, năng lực tối ưu (ở đây biểu thị bằng bậc của bộ hồi quy đa thức tối ưu) tăng lên. Năng lực tối ưu dần đạt đến trạng thái bão hòa sau khi đạt đủ độ phức tạp để giải quyết tác vụ.

5.2.1 Định lý không có bữa ăn miễn phí

Lý thuyết học máy cho rằng một thuật toán học máy có thể tổng quát hóa tốt từ một tập hữu hạn các ví dụ huấn luyện. Điều này dường như mâu thuẫn với một số nguyên lý cơ bản của logic. Lập luận quy nạp, hay suy luận ra các quy tắc tổng quát từ một tập hợp giới hạn các ví dụ, không hợp lý theo logic. Để suy luận một cách logic ra một quy tắc mô tả mọi phần tử của một tập hợp, ta cần có thông tin về từng phần tử trong tập hợp đó.

Một phần, học máy tránh được vấn đề này bằng cách chỉ cung cấp các quy tắc có tính chất xác suất, thay vì các quy tắc chắc chắn hoàn toàn như trong lập luận logic thuần túy. Học máy hứa hẹn sẽ tìm ra các quy tắc *có khả năng* đúng với *phần lớn* các phần tử trong tập hợp mà nó quan tâm.

Tuy nhiên, đáng tiếc là điều này vẫn chưa giải quyết được toàn bộ vấn đề. **Định lý không có bữa ăn miễn phí** cho học máy (*The Lack of A Priori Distinctions Between Learning Algorithms*, Wolpert, 1996, [34]) chỉ ra rằng, khi tính trung bình trên tất cả các phân phối sinh dữ liệu có thể có, mọi thuật toán phân loại đều có cùng tỷ lệ lỗi khi phân loại các điểm chưa được quan sát trước đó. Nói cách khác, theo một nghĩa nào đó, không có thuật toán học máy nào vượt trội hơn tất cả các thuật toán khác một cách phổ quát. Thuật toán tinh vi nhất mà ta có thể nghĩ ra có hiệu suất trung bình (trên mọi tác vụ có thể có) cũng chỉ ngang với việc đơn giản dự đoán rằng mọi điểm đều thuộc cùng một lớp.

May mắn thay, những kết quả này chỉ đúng khi ta tính trung bình trên tất cả các phân phối sinh dữ liệu có thể có. Nếu ta đưa ra các giả định về các loại phân phối xác suất mà ta gặp trong các ứng dụng thực tế, thì có thể thiết kế các thuật toán học máy hoạt động tốt trên những phân phối này.

Điều này có nghĩa là mục tiêu của nghiên cứu học máy không phải là tìm một thuật toán học phổ quát hay thuật toán tốt nhất tuyệt đối. Thay vào đó, mục tiêu của ta là hiểu những loại phân phối nào có liên quan đến "thế giới thực" mà một tác nhân AI trải nghiệm, và những loại thuật toán học máy nào hoạt động tốt với dữ liệu được rút ra từ những phân phối dữ liệu mà ta quan tâm.

5.2.2 Điều chuẩn

Định lý không có bữa ăn miễn phí hàm ý rằng ta phải thiết kế các thuật toán học máy để hoạt động tốt trên một tác vụ cụ thể. Chúng ta làm điều này bằng cách xây dựng một tập hợp các ưu tiên vào thuật toán học. Khi các ưu tiên này phù hợp với các bài toán mà ta yêu cầu thuật toán giải quyết, nó sẽ hoạt động tốt hơn.

Cho đến nay, phương pháp duy nhất mà chúng ta đã thảo luận một cách cụ thể để thay đổi thuật toán học là tăng hoặc giảm năng lực biểu diễn của mô hình bằng cách thêm hoặc bớt các hàm trong không gian giả thuyết của nghiệm mà thuật toán có thể chọn. Ta đã đưa ra ví dụ cụ thể về việc tăng hoặc giảm bậc của một đa thức cho một bài toán hồi quy. Tuy nhiên, cách nhìn mà chúng ta đã mô tả cho đến nay là quá đơn giản.

Hành vi của thuật toán của chúng ta không chỉ bị ảnh hưởng bởi kích thước của tập hợp các hàm được phép trong không gian giả thuyết mà còn bởi đặc tính cụ thể của những hàm đó. Thuật toán học mà chúng ta đã nghiên cứu cho đến nay, hồi quy tuyến tính, có một không gian giả thuyết bao gồm tập hợp các hàm tuyến tính của đầu vào. Các hàm tuyến tính này có thể rất hữu ích cho các bài toán mà mối quan hệ giữa đầu vào và đầu ra gần như là tuyến tính. Tuy nhiên, chúng kém hữu ích hơn đối với các bài toán có hành vi phi tuyến tính rõ rệt. Ví dụ, hồi quy tuyến tính sẽ không hoạt động tốt nếu ta cố gắng sử dụng nó để dự đoán hàm sin x từ x. Do đó, ta có thể điều chỉnh hiệu suất của các thuật toán bằng cách chọn loại hàm nào cho phép chúng lấy các nghiệm từ đó, cũng như kiểm soát số lượng các hàm này.

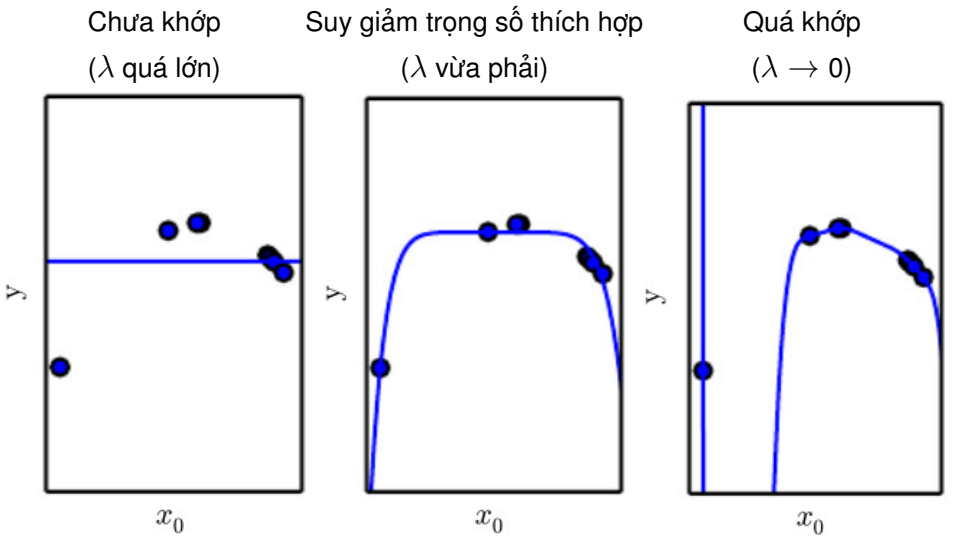
Chúng ta cũng có thể cung cấp cho một thuật toán học một ưu tiên cho một nghiệm trong không gian giả thuyết so với nghiệm khác. Điều này có nghĩa là cả hai hàm đều đủ điều kiện, nhưng một hàm được ưa thích hơn. Nghiệm không được ưa thích sẽ chỉ được chọn nếu nó phù hợp với dữ liệu huấn luyện tốt hơn đáng kể so với nghiệm được ưa thích.

Chẳng hạn, ta có thể điều chỉnh tiêu chí huấn luyện của hồi quy tuyến tính để bao gồm yếu tố **suy giảm trọng số**. Để thực hiện hồi quy tuyến tính với suy giảm trọng số, ta cần cực tiểu hóa tổng gồm cả sai số bình phương trung bình trên tập huấn luyện và tiêu chí $J(\boldsymbol{w})$ thể hiện sự ưu tiên cho trọng số có chuẩn L^2 bình phương nhỏ hơn. Cụ thể:

$$J(\boldsymbol{w}) = \text{MSE}_{\text{train}} + \lambda \boldsymbol{w}^T \boldsymbol{w}, \tag{5.18}$$

trong đó λ là giá trị được chọn trước để kiểm soát mức độ ưu tiên của ta đối với các trọng số nhỏ hơn. Khi $\lambda = 0$, ta không áp dụng ưu tiên nào, và giá trị λ lớn hơn buộc các trọng số trở nên nhỏ hơn. Việc cực tiểu hóa $J(\boldsymbol{w})$ dẫn đến lựa chọn trọng số sao cho có sự cân bằng giữa việc khớp với dữ liệu huấn luyện và duy trì giá trị nhỏ. Điều này mang lại các nghiệm có độ dốc nhỏ hơn hoặc đặt trọng số lên ít đặc trưng hơn. Như một ví dụ về cách ta có thể kiểm soát xu hướng quá khớp hoặc chưa khớp của mô hình thông qua suy giảm trọng số, ta có thể huấn luyện

một mô hình hồi quy đa thức bậc cao với các giá trị khác nhau của λ. Hãy xem kết quả trong Hình 5.5.



Hình 5.5: Chúng tôi áp dụng mô hình hồi quy đa thức bậc cao vào tập ví dụ huấn luyện từ Hình 5.2. Hàm thực sự là hàm bậc hai, nhưng ở đây chúng tôi chỉ sử dụng các mô hình với bậc 9. Chúng tôi thay đổi lượng suy giảm trọng số để ngăn chặn hiện tượng quá khớp của các mô hình bậc cao này. *Hình trái*: Với giá trị λ rất lớn, ta có thể buộc mô hình học một hàm không có độ dốc. Điều này dẫn đến hiện tượng chưa khớp vì mô hình chỉ có thể biểu diễn một hàm hằng số. *Hình giữa*: Với giá trị λ vừa phải, thuật toán học khôi phục được đường cong đúng với hình dạng tổng quát. Mặc dù mô hình có năng lực biểu diễn các hàm với hình dạng phức tạp hơn, suy giảm trọng số đã khuyến khích nó sử dụng một hàm đơn giản hơn được mô tả bởi các hệ số nhỏ hơn. *Hình phải*: Khi suy giảm trọng số tiến gần đến 0 (tức là sử dụng ma trận giả nghịch đảo Moore – Penrose để giải bài toán không xác định với điều chuẩn nhỏ nhất), đa thức bậc 9 bị quá khớp nghiêm trọng, như ta đã thấy ở Hình 5.2.

Tổng quát hơn, ta có thể điều chuẩn một mô hình học hàm $f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ bằng cách thêm một hạng tử phạt, gọi là **bộ điều chuẩn**, vào hàm chi phí. Trong trường hợp suy giảm trọng số, bộ điều chuẩn là $\Omega\left(\boldsymbol{w}\right) = \boldsymbol{w}^T\boldsymbol{w}$. Trong Chương 10, ta sẽ thấy nhiều bộ điều chuẩn khác có thể áp dụng.

Việc biểu đạt ưu tiên cho một hàm này so với một hàm khác là cách kiểm soát năng lực của mô hình một cách tổng quát hơn so với việc bao gồm hoặc loại bỏ các phần tử trong không gian giả thuyết. Chúng ta có thể nghĩ rằng việc loại trừ

một hàm khỏi không gian giả thuyết tương đương với việc thể hiện sự ưu tiên vô hạn đối với việc chống lại hàm đó.

Trong ví dụ về suy giảm trọng số, ta đã thể hiện sự ưu tiên đối với các hàm tuyến tính được định nghĩa bằng các trọng số nhỏ hơn một cách rõ ràng thông qua một hạng mục bổ sung trong tiêu chuẩn mà ta cực tiểu hóa. Có rất nhiều cách khác để thể hiện ưu tiên cho các nghiệm khác nhau, cả ngầm hiểu và tường minh. Tập hợp các phương pháp này được gọi chung là **điều chuẩn**. *Điều chuẩn là bất kỳ sự điều chỉnh nào mà ta thực hiện đối với một thuật toán học nhằm mục đích giảm lỗi tổng quát hóa nhưng không giảm lỗi huấn luyện*. Điều chuẩn là một trong những mối quan tâm chính trong lĩnh vực học máy, có tầm quan trọng ngang với tối ưu hóa.

Định lý không có bữa ăn miễn phí đã chỉ ra rằng không có thuật toán học máy nào là tốt nhất, và đặc biệt là không có dạng điều chuẩn nào là tốt nhất. Thay vào đó, ta phải chọn một dạng điều chuẩn phù hợp với tác vụ cụ thể mà ta muốn giải quyết. Triết lý của học sâu nói chung và của cuốn sách này nói riêng là một phạm vi rất rộng các tác vụ (chẳng hạn như tất cả các tác vụ trí tuệ mà con người có thể thực hiện) đều có thể được giải quyết hiệu quả bằng các dạng điều chuẩn có tính tổng quát cao.

5.3 Siêu tham số và tập kiểm định

Hầu hết các thuật toán học máy đều có nhiều thiết lập mà ta có thể sử dụng để kiểm soát hành vi của thuật toán học. Những thiết lập này được gọi là **siêu tham số**. Giá trị của các siêu tham số không được thuật toán học tự điều chỉnh (mặc dù ta có thể thiết kế một quy trình học lồng nhau, trong đó một thuật toán học sẽ tìm các siêu tham số tốt nhất cho một thuật toán học khác).

Trong ví dụ hồi quy đa thức mà ta thấy ở Hình 5.2, có một siêu tham số duy nhất: bậc của đa thức, đóng vai trò như một siêu tham số điều khiển năng lực của mô hình. Giá trị λ được sử dụng để kiểm soát cường độ của việc suy giảm trọng số là một ví dụ khác về siêu tham số.

Đôi khi một thiết lập được chọn làm siêu tham số mà thuật toán học không học được do khó tối ưu hóa. Thường thì thiết lập này phải là siêu tham số vì không thích hợp để học siêu tham số đó trên tập huấn luyện. Điều này áp dụng cho tất cả các siêu tham số kiểm soát năng lực của mô hình. Nếu học được trên tập huấn luyện, các siêu tham số như vậy sẽ luôn chọn năng lực mô hình tối đa có thể, dẫn đến

hiện tượng quá khớp (tham khảo Hình 5.3). Ví dụ, ta luôn có thể khớp với tập huấn luyện bằng một đa thức bậc cao và thiết lập suy giảm trọng số với $\lambda = 0$ tốt hơn so với khi sử dụng một đa thức bậc thấp và thiết lập suy giảm trọng số dương.

Để giải quyết vấn đề này, ta cần một **tập kiểm định** bao gồm các ví dụ mà thuật toán huấn luyện không quan sát thấy.

Trước đó, chúng ta đã thảo luận về cách một tập kiểm tra độc lập được giữ lại, bao gồm các ví dụ có cùng phân phối với tập huấn luyện, có thể được sử dụng để ước lượng sai số tổng quát hóa của một mô hình học sau khi quá trình học đã hoàn thành. Điều quan trọng là các ví dụ trong tập kiểm tra không được sử dụng theo bất kỳ cách nào để đưa ra lựa chọn về mô hình, bao gồm cả các siêu tham số của nó. Vì lý do này, không có ví dụ nào từ tập kiểm tra được sử dụng trong tập kiểm định. Do đó, ta luôn xây dựng tập kiểm định từ dữ liệu *huấn luyện*. Cụ thể, ta chia dữ liệu huấn luyện thành hai tập hợp rời nhau. Một trong các tập hợp này được sử dụng để học các tham số, tập còn lại là tập kiểm định của chúng ta, dùng để ước lượng lỗi tổng quát hóa trong hoặc sau khi huấn luyện, cho phép các siêu tham số được cập nhật tương ứng. Tập hợp dữ liệu được sử dụng để học các tham số vẫn thường được gọi là tập huấn luyện, mặc dù điều này có thể gây nhầm lẫn với tập dữ liệu lớn hơn được sử dụng cho toàn bộ quá trình huấn luyện. Tập dữ liệu được sử dụng để hướng dẫn lựa chọn các siêu tham số được gọi là tập kiểm định. Thông thường, người ta sử dụng khoảng 80% dữ liệu huấn luyện cho huấn luyện và 20% cho kiểm định. Vì tập kiểm định được sử dụng để "huấn luyện" các siêu tham số, sai số trên tập kiểm định sẽ đánh giá thấp sai số tổng quát hóa, dù thường nhỏ hơn so với sai số trên tập huấn luyện. Sau khi hoàn tất tối ưu hóa siêu tham số, lỗi tổng quát hóa có thể được ước lượng bằng tập kiểm tra.

Trong thực tế, khi cùng một tập kiểm tra được sử dụng lặp đi lặp lại để đánh giá hiệu suất của các thuật toán khác nhau trong nhiều năm, và đặc biệt là nếu chúng ta xem xét tất cả các nỗ lực từ cộng đồng khoa học trong việc vượt qua hiệu suất tiên tiến đã được công bố trên tập kiểm tra đó, chúng ta sẽ có những đánh giá lạc quan với tập kiểm tra này. Do đó, các bộ đánh giá có thể trở nên lỗi thời và không phản ánh đúng hiệu suất thực tế của một hệ thống đã qua huấn luyện. May mắn thay, cộng đồng thường có xu hướng chuyển sang các bộ dữ liệu đánh giá mới (thường là tham vọng hơn và lớn hơn).

5.3.1 Xác thực chéo

Việc chia tập dữ liệu thành một tập huấn luyện cố định và một tập kiểm tra cố định có thể gây vấn đề nếu tập kiểm tra quá nhỏ. Một tập kiểm tra nhỏ dẫn đến sự bất định thống kê trong ước lượng sai số trung bình trên tập kiểm tra, khiến việc khẳng định thuật toán *A* hoạt động tốt hơn thuật toán *B* trên tác vụ cụ thể trở nên khó khăn.

Khi tập dữ liệu có hàng trăm nghìn ví dụ trở lên, vấn đề này không quá nghiêm trọng. Tuy nhiên, khi tập dữ liệu quá nhỏ, có những phương pháp thay thế cho phép sử dụng tất cả các ví dụ để ước lượng sai số trung bình, nhưng phải đánh đổi bằng chi phí tính toán cao hơn. Các phương pháp này dựa trên ý tưởng lặp lại quá trình huấn luyện và kiểm tra trên các tập con hoặc chia nhỏ ngẫu nhiên khác nhau của tập dữ liệu ban đầu. Phương pháp phổ biến nhất là xác thực chéo *k* lần, như được trình bày trong Thuật toán 2. Trong phương pháp này, tập dữ liệu được chia thành *k* tập con rời nhau. Sai số kiểm tra có thể được ước lượng bằng cách lấy trung bình lỗi kiểm tra qua *k* lần thử. Trong lần thử thứ *i*, tập con thứ *i* được sử dụng làm tập kiểm tra và các tập con còn lại làm tập huấn luyện. Một vấn đề là không tồn tại ước lượng không chệch cho phương sai của các ước lượng sai số trung bình như vậy (*No Unbiased Estimator of the Variance of K-Fold Cross-Validation*, Bengio và Grandvalet, 2003, [35]), nhưng các xấp xỉ vẫn thường được sử dụng.

5.4 Ước lượng, độ lệch và phương sai

Lĩnh vực thống kê cung cấp cho chúng ta nhiều công cụ hữu ích để đạt được mục tiêu của học máy trong giải quyết một tác vụ không chỉ trên tập huấn luyện mà còn có khả năng tổng quát hóa. Các khái niệm cơ bản như ước lượng tham số, độ chệch và phương sai rất hữu ích để đặc trưng hóa một cách chính thức các khái niệm về khả năng tổng quát hóa, hiện tượng chưa khớp và quá khớp.

5.4.1 Ước lượng điểm

Ước lượng điểm là nỗ lực cung cấp dự đoán đơn trị "tốt nhất" cho một đại lượng quan tâm nào đó. Nhìn chung, đại lượng này có thể là một tham số duy nhất hoặc một véctơ tham số trong một mô hình tham số nào đó, chẳng hạn như các trọng số trong ví dụ hồi quy tuyến tính của chúng ta ở Mục 5.1.4, nhưng cũng có thể là một hàm hoàn chỉnh.

Thuật toán 2: Thuật toán xác thực chéo k lần. Thuật toán này có thể được sử dụng để ước lượng sai số tổng quát hóa của một thuật toán học A khi tập dữ liệu D quá nhỏ để việc chia tập thành tập huấn luyện/kiểm tra hoặc huấn luyện/kiểm định đơn giản cho ra kết quả chính xác của ước lượng sai số tổng quát hóa, vì trung bình của một hàm mất mát L trên một tập kiểm tra nhỏ có thể có phương sai quá cao. Tập dữ liệu D chứa các ví dụ trừu tượng $\boldsymbol{z}^{(i)}$ (cho ví dụ thứ i), có thể là cặp (đầu vào, mục tiêu) $\boldsymbol{z}^{(i)} = \left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$ trong học có giám sát, hoặc chỉ là đầu vào $\boldsymbol{z}^{(i)} = \boldsymbol{x}^{(i)}$ trong học không giám sát. Thuật toán trả về một véctơ sai số $\boldsymbol{e}$ cho mỗi ví dụ trong D, trong đó giá trị trung bình của các sai số là ước lượng của sai số tổng quát hóa. Các sai số trên từng ví dụ có thể được dùng để tính khoảng tin cậy xung quanh giá trị trung bình (phương trình (5.47)). Mặc dù các khoảng tin cậy này không được đảm bảo chắc chắn khi sử dụng xác thực chéo, việc sử dụng chúng vẫn là thực hành phổ biến, nhằm khẳng định thuật toán A tốt hơn thuật toán B nếu khoảng tin cậy của sai số của thuật toán A nằm dưới và không giao với khoảng tin cậy của thuật toán B.

Đầu vào:

- D: Tập dữ liệu cho trước, với các phần tử là $\boldsymbol{z}^{(i)}$ (ví dụ thứ i).
- A: Thuật toán học, được coi là một hàm nhận tập dữ liệu làm đầu vào và xuất ra một hàm đã học.
- L: Hàm mất mát, được coi là một hàm, tương ứng hàm đã học f và một ví dụ $\boldsymbol{z}^{(i)} \in D$ với một giá trị vô hướng trong $\mathbb{R}$.
- k: Số tập con chia D ra.

Đầu ra: Véctơ $\boldsymbol{e}$ gồm các sai số cho mỗi ví dụ trong D.

Chia D thành k tập con rời nhau D_i sao cho hợp của chúng là D.;
for $i \leftarrow 1$ **to** k **do**
    $f_i = A\left(D \backslash D_i\right)$;
    for $\boldsymbol{z}^{(j)} \in D_i$ **do**
        $e_j = L\left(f_i, \boldsymbol{z}^{(j)}\right)$;
    end
end

Để phân biệt ước lượng của các tham số với giá trị thật của chúng, ta quy ước ký hiệu lượng điểm của một tham số $\boldsymbol{\theta}$ bằng $\widehat{\boldsymbol{\theta}}$.

Giả sử $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\right\}$ là tập dữ liệu gồm m điểm dữ liệu độc lập cùng phân phối. Một **ước lượng điểm** hay **thống kê** của $\boldsymbol{\theta}$ là hàm bất kỳ của dữ liệu:

$$\widehat{\boldsymbol{\theta}}_m = g\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\right). \tag{5.19}$$

Định nghĩa này không yêu cầu g trả về giá trị gần với giá trị thực của $\boldsymbol{\theta}$ hoặc thậm chí không cần miền giá trị của g phải giống với tập hợp các giá trị cho phép của $\boldsymbol{\theta}$. Định nghĩa của một ước lượng điểm rất tổng quát và cho phép nhiều linh hoạt trong thiết kế ước lượng. Dù hầu như bất kỳ hàm nào cũng có thể là một ước lượng, một ước lượng tốt là hàm có đầu ra gần với θ thực sự, thứ góp phần tạo ra dữ liệu huấn luyện.

Hiện tại, chúng ta áp dụng quan điểm tần suất thống kê. Nghĩa là, chúng ta giả định giá trị thực của tham số $\boldsymbol{\theta}$ là cố định nhưng chưa biết, trong khi ước lượng điểm $\widehat{\boldsymbol{\theta}}$ là một hàm của dữ liệu. Do dữ liệu được lấy từ một quá trình ngẫu nhiên, bất kỳ hàm nào của dữ liệu cũng là ngẫu nhiên. Do đó, $\widehat{\boldsymbol{\theta}}$ là một biến ngẫu nhiên.

Ước lượng điểm cũng có thể đề cập đến việc ước lượng mối quan hệ giữa biến đầu vào và biến mục tiêu. Chúng ta gọi các loại ước lượng điểm này là ước lượng hàm.

Ước lượng hàm: Như đã đề cập ở trên, đôi khi ta quan tâm đến việc ước lượng hàm (hoặc xấp xỉ hàm). Trong trường hợp này, ta muốn dự đoán một biến $\boldsymbol{y}$ dựa trên một véctơ đầu vào $\boldsymbol{x}$. Ta giả định rằng có một hàm $\boldsymbol{f}(\boldsymbol{x})$ mô tả mối quan hệ xấp xỉ giữa $\boldsymbol{y}$ và $\boldsymbol{x}$. Ví dụ, ta có thể giả định rằng $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \varepsilon$, trong đó ε đại diện cho phần của $\boldsymbol{y}$ không thể dự đoán được từ $\boldsymbol{x}$. Trong ước lượng hàm, ta quan tâm đến việc xấp xỉ $\boldsymbol{f}$ bằng một mô hình hoặc ước lượng $\widehat{\boldsymbol{f}}$. Ước lượng hàm thực chất giống như việc ước lượng một tham số $\boldsymbol{\theta}$; ước lượng hàm $\widehat{\boldsymbol{f}}$ đơn giản là một ước lượng điểm trong không gian hàm. Ví dụ hồi quy tuyến tính (đã thảo luận ở Mục 5.1.4) và ví dụ hồi quy đa thức (đã thảo luận ở Mục 5.2) đều là các trường hợp có thể được hiểu là ước lượng tham số $\boldsymbol{w}$ hay ước lượng một hàm $\widehat{f}$ cho tương ứng $\boldsymbol{x}$ với y.

Bây giờ, chúng ta sẽ xem xét các thuộc tính thường được nghiên cứu của các ước lượng điểm và thảo luận về ý nghĩa của chúng.

5.4.2 Độ lệch

Độ lệch (bias) của một ước lượng được định nghĩa là:

$$\text{bias}\left(\widehat{\boldsymbol{\theta}}_m\right) = E\left(\widehat{\boldsymbol{\theta}}_m\right) - \boldsymbol{\theta} \tag{5.20}$$

trong đó, kỳ vọng E được lấy theo dữ liệu (được coi như các mẫu từ một biến ngẫu nhiên) và $\boldsymbol{\theta}$ là giá trị thật sự của $\boldsymbol{\theta}$, dùng để xác định phân phối sinh dữ liệu. Một ước lượng $\widehat{\boldsymbol{\theta}}_m$ được gọi là **không chệch** nếu bias $\left(\widehat{\boldsymbol{\theta}}_m\right) = \mathbf{0}$, điều này ngụ ý rằng $E\left(\widehat{\boldsymbol{\theta}}_m\right) = \boldsymbol{\theta}$. Nghĩa là, giá trị trung bình của ước lượng khớp với giá trị thật. Một ước lượng $\widehat{\boldsymbol{\theta}}_m$ được gọi là **tiệm cận không chệch** nếu $\lim_{m\to\infty}$ bias $\left(\widehat{\boldsymbol{\theta}}_m\right) = \mathbf{0}$, điều này ngụ ý rằng $\lim_{m\to\infty} E\left(\widehat{\boldsymbol{\theta}}_m\right) = \boldsymbol{\theta}$. Điều này có nghĩa là khi số lượng mẫu m tăng lên vô hạn, độ lệch của ước lượng sẽ tiến dần về không, và giá trị kỳ vọng của nó sẽ tiến dần đến giá trị thật của $\boldsymbol{\theta}$.

Ví dụ: Phân phối Bernoulli Xét tập mẫu $\{x^{(1)}, \ldots, x^{(m)}\}$ độc lập cùng phân phối theo phân phối Bernoulli với trung bình θ:

$$P\left(x^{(i)}; \theta\right) = \theta^{x^{(i)}}\left(1-\theta\right)^{\left(1-x^{(i)}\right)}. \quad \text{(xem (3.21))} \tag{5.21}$$

Một ước lượng phổ biến cho tham số θ của phân phối này là trung bình của các mẫu huấn luyện:

$$\widehat{\theta}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}. \tag{5.22}$$

Để xác định ước lượng này có bị lệch hay không, ta có thể thay phương trình (5.22) vào phương trình (5.20):

$$\begin{aligned}
\text{bias}\left(\widehat{\theta}\right) &= E\left(\widehat{\theta}_m\right) - \theta && (5.23)\\
&= E\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right) - \theta && (5.24)\\
&= \frac{1}{m}\sum_{i=1}^{m} E\left(x^{(i)}\right) - \theta && (5.25)\\
&= \frac{1}{m}\sum_{i=1}^{m} \left(\theta\right) - \theta \quad \text{(xem (3.22))} && (5.26)\\
&= \theta - \theta = 0 && (5.27)
\end{aligned}$$

Vì bias$(\widehat{\theta}) = 0$, nên ước lượng $\widehat{\theta}$ của θ là không chệch.

Ví dụ: Ước lượng trung bình của phân phối xác suất Xét tập mẫu $\{x^{(1)}, \ldots, x^{(m)}\}$ độc lập cùng phân phối theo phân phối xác suất có kỳ vọng μ. Một ước lượng phổ biến cho tham số trung bình μ của phân phối xác suất được gọi là **trung bình mẫu**:

$$\widehat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}. \tag{5.28}$$

Để xác định độ lệch của trung bình mẫu, ta lại tính kỳ vọng của nó:

$$\begin{aligned}\text{bias}\left(\widehat{\mu}_m\right) &= E\left(\widehat{\mu}_m\right) - \mu \\ &= E\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right) - \mu \\ &= \left(\frac{1}{m}\sum_{i=1}^{m} E\left(x^{(i)}\right)\right) - \mu \\ &= \left(\frac{1}{m}\sum_{i=1}^{m} \mu\right) - \mu \\ &= \mu - \mu = 0.\end{aligned}$$

Do đó, ta thấy rằng trung bình mẫu là một ước lượng không chệch của tham số trung bình của phân phối xác suất.

Ví dụ: Các ước lượng phương sai của phân phối xác suất Trong ví dụ này, ta so sánh hai ước lượng khác nhau cho tham số phương sai σ^2 của một phân phối xác suất để xác định xem liệu ước lượng nào có bị lệch hay không.

Ước lượng đầu tiên cho σ^2 là **phương sai mẫu**:

$$\widehat{\sigma}_m^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \widehat{\mu}_m\right)^2, \tag{5.29}$$

trong đó $\widehat{\mu}_m$ là trung bình mẫu, đã được định nghĩa ở trên. Ta muốn tính toán độ lệch của ước lượng này:

$$\text{bias}\left(\widehat{\sigma}_m^2\right) = E\left(\widehat{\sigma}_m^2\right) - \sigma^2. \tag{5.30}$$

Bắt đầu với việc đánh giá kỳ vọng của phương sai mẫu:

$$E\left(\widehat{\sigma}_m^2\right) = E\left[\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \widehat{\mu}_m\right)^2\right] \tag{5.31}$$

$$= \frac{m-1}{m}\sigma^2. \tag{5.32}$$

Quay lại phương trình (5.30), ta kết luận rằng độ lệch của $\widehat{\sigma}_m^2$ là $-\frac{\sigma^2}{m}$. Do đó, phương sai mẫu là một ước lượng bị lệch của tham số phương sai của phân phối xác suất.

Ước lượng **phương sai mẫu không chệch**

$$\widetilde{\sigma}_m^2 = \frac{1}{m-1}\sum_{i=1}^{m}\left(x^{(i)} - \widehat{\mu}_m\right)^2 \tag{5.33}$$

Như tên gọi của nó, ước lượng này là không chệch. Tức là, ta có $E\left(\widetilde{\sigma}_m^2\right) = \sigma^2$:

$$E\left(\widetilde{\sigma}_m^2\right) = E\left[\frac{1}{m-1}\sum_{i=1}^{m}\left(x^{(i)} - \widehat{\mu}_m^2\right)^2\right] \tag{5.34}$$

$$= \frac{m}{m-1}E\left(\widehat{\sigma}_m^2\right) \tag{5.35}$$

$$= \frac{m}{m-1}\left(\frac{m-1}{m}\sigma^2\right) \tag{5.36}$$

$$= \sigma^2. \tag{5.37}$$

Với hai ước lượng đã xem xét: một ước lượng bị lệch và một không chệch. Mặc dù các ước lượng không lệch thường được ưu tiên, nhưng chúng không phải luôn là "tốt nhất". Thực tế, ta thường sử dụng các ước lượng bị lệch do chúng có các đặc tính quan trọng khác.

5.4.3 Phương sai và sai số chuẩn

Một đặc tính khác của ước lượng mà ta có thể xem xét là mức độ dao động của nó phụ thuộc vào mẫu dữ liệu. Tương tự như cách tính kỳ vọng của ước lượng để xác định độ lệch, ta có thể tính phương sai của nó. **Phương sai** của một ước lượng là

$$\text{Var}\left(\widehat{\theta}\right) \tag{5.38}$$

trong đó biến ngẫu nhiên là tập huấn luyện. Ngoài ra, căn bậc hai của phương sai gọi là **sai số chuẩn**, ký hiệu là $\text{SE}\left(\widehat{\theta}\right)$:

$$\text{SE}\left(\widehat{\theta}\right) = \sqrt{\text{Var}\left(\widehat{\theta}_m\right)}. \tag{5.39}$$

Phương sai hoặc sai số chuẩn của một ước lượng cung cấp thước đo về mức độ biến thiên dự kiến của ước lượng khi ta lấy mẫu lại tập dữ liệu một cách độc lập từ quá trình sinh dữ liệu. Giống như việc mong muốn một ước lượng có độ lệch thấp, ta cũng muốn nó có phương sai tương đối thấp.

Khi tính bất kỳ thống kê nào từ một số lượng mẫu hữu hạn, ước lượng của tham số thật ẩn sẽ không chắc chắn, vì có thể thu được các mẫu khác từ cùng một phân phối với các thống kê khác nhau. Độ dao động kỳ vọng này trong bất kỳ ước lượng nào là một nguồn sai số mà ta muốn định lượng.

Ví dụ, vẫn xét tập mẫu $\{x^{(1)}, \ldots, x^{(m)}\}$ độc lập cùng phân phối theo phân phối xác suất có phương sai σ^2. Phương sai của kỳ vọng mẫu bằng

$$\text{Var}\left(\widehat{\theta}_m\right) = \text{Var}\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right) \tag{5.40}$$

$$= \left(\frac{1}{m}\right)^2 \text{Var}\left(\sum_{i=1}^{m} x^{(i)}\right) \tag{5.41}$$

$$= \frac{1}{m^2}\sum_{i=1}^{m} \text{Var}\left(x^{(i)}\right) \tag{5.42}$$

$$= \frac{1}{m^2}\sum_{i=1}^{m} \sigma^2 \tag{5.43}$$

$$= \frac{1}{m^2} m\sigma^2 \tag{5.44}$$

$$= \frac{\sigma^2}{m}, \tag{5.45}$$

và sai số chuẩn của trung bình mẫu được cho bởi:

$$\text{SE}\left(\widehat{\mu}_m\right) = \sqrt{\text{Var}\left(\widehat{\theta}_m\right)} = \frac{\sigma}{\sqrt{m}}. \tag{5.46}$$

Sai số chuẩn thường được ước lượng bằng cách sử dụng một ước lượng của σ. Tuy nhiên, căn bậc hai của phương sai mẫu hoặc căn bậc hai của ước lượng phương sai không lệch đều không cung cấp một ước lượng không lệch cho độ lệch chuẩn. Cả hai cách tiếp cận này đều có xu hướng đánh giá thấp độ lệch chuẩn thật, nhưng vẫn được sử dụng trong thực tế. Căn bậc hai của ước lượng phương sai không lệch sẽ đánh giá thấp ít hơn. Đối với giá trị *m* lớn, cách xấp xỉ này khá hợp lý.

Sai số chuẩn của trung bình rất hữu ích trong các thí nghiệm học máy. Ta thường ước lượng sai số tổng quát hóa bằng cách tính trung bình mẫu của sai số trên tập kiểm tra. Số lượng mẫu trong tập kiểm tra quyết định độ chính xác của ước lượng này. Nhờ vào định lý giới hạn trung tâm, vốn cho biết rằng trung bình sẽ có phân phối xấp xỉ với phân phối chuẩn, ta có thể sử dụng sai số chuẩn để tính xác suất rằng kỳ vọng thật nằm trong khoảng bất kỳ cho trước. Ví dụ, khoảng tin cậy 95% với trung tâm tại trung bình $\widehat{\mu}_m$ là:

$$\left(\widehat{\mu}_m - 1.96\text{SE}\left(\widehat{\mu}_m\right),\ \widehat{\mu}_m + 1.96\text{SE}\left(\widehat{\mu}_m\right)\right), \tag{5.47}$$

dưới giả thiết phân phối chuẩn có trung bình $\widehat{\mu}_m$ và phương sai $\text{SE}\left(\widehat{\mu}_m\right)^2$. Trong các thí nghiệm học máy, người ta thường nói rằng thuật toán *A* tốt hơn thuật toán

B nếu cận trên của khoảng tin cậy 95% cho sai số của thuật toán A nhỏ hơn cận dưới của khoảng tin cậy 95% cho sai số của thuật toán B.

Ví dụ: Phân phối Bernoulli Ta lại xét một tập mẫu $\{x^{(1)}, \ldots, x^{(m)}\}$ được chọn một cách độc lập và phân phối đồng nhất từ một phân phối Bernoulli (nhớ lại rằng $P\left(x^{(i)}; \theta\right) = \theta^{x^{(i)}} \left(1-\theta\right)^{\left(1-x^{(i)}\right)}$). Lần này, ta quan tâm đến việc tính phương sai của ước lượng $\widehat{\theta}_m = \dfrac{1}{m}\sum_{i=1}^{m} x^{(i)}$:

$$\text{Var}\left(\widehat{\theta}_m\right) = \frac{1}{m}\theta\left(1-\theta\right). \quad \text{(xem (3.23))} \tag{5.48}$$

Phương sai của ước lượng giảm theo hàm số m, số lượng mẫu trong tập dữ liệu. Đây là một tính chất hay gặp của các ước lượng phổ biến mà ta sẽ quay lại khi thảo luận về tính ước lượng vững (xem Mục 5.4.5).

5.4.4 Đánh đổi giữa độ lệch và phương sai để cực tiểu sai số bình phương trung bình

Độ lệch và phương sai đo lường hai nguồn sai số khác nhau trong một ước lượng. Độ lệch đo sự sai lệch kỳ vọng so với giá trị thực của hàm số hoặc tham số. Ngược lại, phương sai cung cấp một thước đo về mức độ sai lệch so với giá trị ước lượng kỳ vọng mà một mẫu cụ thể của dữ liệu có thể gây ra.

Điều gì xảy ra khi ta được chọn giữa hai ước lượng, một có độ lệch lớn hơn và một có phương sai lớn hơn? Làm thế nào để ta lựa chọn giữa chúng? Ví dụ, giả sử ta quan tâm đến việc xấp xỉ hàm số được biểu diễn trong Hình 5.2 và chỉ được chọn giữa một mô hình có độ lệch lớn và một mô hình chịu ảnh hưởng của phương sai lớn. Làm thế nào để chọn giữa chúng?

Cách phổ biến nhất để cân bằng sự đánh đổi này là sử dụng phương pháp kiểm tra chéo. Thực nghiệm cho thấy kiểm tra chéo thành công cao trong nhiều tác vụ thực tế. Ngoài ra, ta cũng có thể so sánh **sai số bình phương trung bình** (MSE) của các ước lượng:

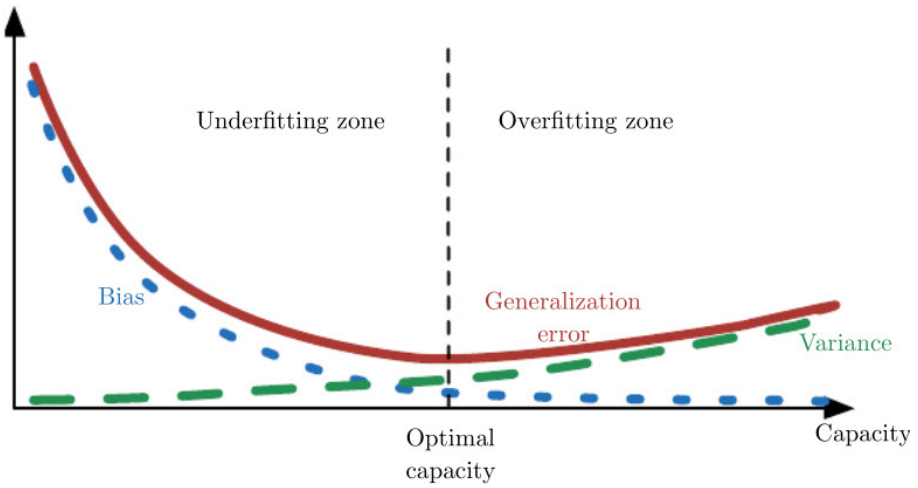
$$\text{MSE}\left(\widehat{\theta}_m\right) = E\left[\left(\widehat{\theta}_m - \theta\right)^2\right] \tag{5.49}$$

$$= \text{bias}\left(\widehat{\theta}_m\right)^2 + \text{Var}\left(\widehat{\theta}_m\right) \tag{5.50}$$

MSE đo lường sai lệch kỳ vọng tổng thể (theo nghĩa sai số bình phương) giữa ước lượng và giá trị thực của tham số θ. Như rõ ràng từ phương trình (5.50), đánh

giá MSE kết hợp cả độ lệch và phương sai. Các ước lượng mong muốn là những ước lượng có MSE nhỏ, tức là chúng giữ độ lệch và phương sai trong một mức độ kiểm soát nhất định.

Mối quan hệ giữa độ lệch và phương sai gắn chặt với các khái niệm trong học máy như năng lực mô hình, hiện tượng chưa khớp và quá khớp. Trong trường hợp sai số tổng quát hóa được đo bằng MSE (khi độ lệch và phương sai là các thành phần có ý nghĩa của sai số tổng quát hóa), việc tăng năng lực mô hình thường dẫn đến việc tăng phương sai và giảm độ lệch. Điều này được minh họa trong Hình 5.6, nơi chúng ta thấy lại đường cong hình chữ U của sai số tổng quát hóa như là một hàm số của năng lực mô hình.



Hình 5.6: Khi năng lực mô hình tăng lên (trục x), độ lệch (đường chấm) có xu hướng giảm và phương sai (đường gạch) có xu hướng tăng, dẫn đến một đường cong hình chữ U khác cho sai số tổng quát hóa (đường in đậm). Nếu chúng ta thay đổi năng lực theo một trục, sẽ có một mức năng lực tối ưu, với hiện tượng chưa khớp xảy ra khi năng lực thấp hơn mức tối ưu này và hiện tượng quá khớp xảy ra khi năng lực vượt quá mức này. Mối quan hệ này tương tự với mối quan hệ giữa năng lực mô hình, chưa khớp và quá khớp được thảo luận trong Mục 5.2 và Hình 5.3.

5.4.5 Ước lượng vững

Cho đến nay, chúng ta đã thảo luận về các thuộc tính của các ước lượng khác nhau với một tập huấn luyện có kích thước cố định. Thông thường, ta cũng quan tâm đến hành vi của một ước lượng khi lượng dữ liệu huấn luyện tăng lên. Đặc biệt, ta thường mong muốn rằng, khi số điểm dữ liệu m trong tập dữ liệu tăng lên, các

ước lượng điểm sẽ hội tụ đến giá trị thực của các tham số tương ứng. Cụ thể hơn, ta mong muốn

$$\operatorname*{plim}_{m\to\infty} \widehat{\theta}_m = \theta. \tag{5.51}$$

Ký hiệu plim biểu thị sự hội tụ theo xác suất, có nghĩa là với $\varepsilon > 0$ bất kỳ, $P\left(\left|\widehat{\theta}_m - \theta\right| > \varepsilon\right) \to 0$ khi $m \to \infty$. Điều kiện được mô tả bởi phương trình (5.51) được gọi là **tính vững**. Đôi khi, điều này còn được gọi là tính vững yếu, trong khi tính vững mạnh biểu thị sự hội tụ **hầu chắc chắn** của $\widehat{\theta}$ tới θ. **Hội tụ hầu chắc chắn** của một dãy biến ngẫu nhiên $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots$ tới biến ngẫu nhiên $\boldsymbol{X}$ xảy ra khi $P\left(\lim_{m\to\infty} \boldsymbol{X}^{(m)} = \boldsymbol{X}\right) = 1$.

Tính vững đảm bảo rằng độ lệch gây ra bởi ước lượng sẽ giảm dần khi số lượng ví dụ dữ liệu tăng lên. Tuy nhiên, điều ngược lại không đúng — tính không chệch tiệm cận không kéo theo tính vững. Ví dụ, xét việc ước lượng tham số trung bình μ của một phân phối chuẩn $N\left(x; \mu, \sigma^2\right)$ với một tập dữ liệu gồm m mẫu: $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$. Ta có thể sử dụng mẫu đầu tiên $x^{(1)}$ của tập dữ liệu làm ước lượng không chệch: $\widehat{\theta}_m = x^{(1)}$. Trong trường hợp này, $E\left(\widehat{\theta}_m\right) = \theta$, vì vậy ước lượng là không chệch bất kể có bao nhiêu điểm dữ liệu. Tất nhiên điều này ngụ ý rằng ước lượng là không chệch tiệm cận. Tuy nhiên, đây không phải là một ước lượng vững vì $\widehat{\theta}_m$ không hội tụ về θ khi $m \to \infty$.

5.5 Ước lượng hợp lý cực đại

Trước đây, chúng ta đã thấy một số định nghĩa về các ước lượng thường gặp và phân tích các tính chất của chúng. Nhưng các ước lượng này từ đâu mà có? Thay vì đoán rằng một hàm nào đó có thể là ước lượng tốt rồi sau đó phân tích độ lệch và phương sai của nó, ta mong muốn có một nguyên lý từ đó có thể suy ra các hàm cụ thể là ước lượng tốt cho các mô hình khác nhau.

Nguyên lý phổ biến nhất là nguyên lý cực đại hợp lý.

Xét một tập hợp gồm m ví dụ $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\right\}$ được rút ra một cách độc lập từ phân phối sinh dữ liệu thực nhưng không biết $p_{\text{data}}(\boldsymbol{x})$.

Giả sử $p_{\text{model}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ là một họ các phân phối xác suất có tham số trên cùng không gian được chỉ số hóa bởi $\boldsymbol{\theta}$. Nói cách khác, $p_{\text{model}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ ánh xạ cấu hình $\boldsymbol{x}$ bất kỳ với một số thực ước lượng xác suất thực $p_{\text{data}}(\boldsymbol{x})$.

Khi đó, ước lượng hợp lý cực đại $\boldsymbol{\theta}_{\text{ML}}$ được định nghĩa là:

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\text{model}}\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}; \boldsymbol{\theta}\right) \tag{5.52}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \tag{5.53}$$

Tích của nhiều xác suất có thể bất tiện vì nhiều lý do, chẳng hạn như dễ gây ra hiện tượng hạ dưới. Để có một bài toán tối ưu tiện lợi hơn nhưng tương đương, ta nhận thấy rằng việc lấy logarit của hàm hợp lý không làm thay đổi arg max của nó, nhưng lại thuận tiện biến tích thành tổng:

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right). \tag{5.54}$$

Do arg max không thay đổi khi ta co giãn hàm mục tiêu, ta có thể chia cho m để có một phiên bản tiêu chuẩn hóa của tiêu chí, được biểu diễn như kỳ vọng của logarit hàm hợp lý theo biến ngẫu ngẫu nhiên X có phân phối thực nghiệm $\widehat{p}_{\text{data}}$ được định nghĩa bởi dữ liệu huấn luyện:

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} E\left[\log p_{\text{model}}\left(X; \boldsymbol{\theta}\right)\right]. \tag{5.55}$$

Một cách để diễn giải ước lượng hợp lý cực đại là xem nó như việc tối thiểu hóa sự khác biệt giữa phân phối kinh nghiệm $\widehat{p}_{\text{data}}$ xác định bởi tập huấn luyện và phân phối của mô hình, trong đó mức độ khác biệt giữa hai phân phối được đo bằng độ phân kỳ KL. Độ phân kỳ KL được cho bởi

$$D_{\text{KL}}\left(\widehat{p}_{\text{data}} \parallel p_{\text{model}}\right) = E\left[\log \widehat{p}_{\text{data}}(X) - \log p_{\text{model}}(X)\right]. \tag{5.56}$$

Hạng tử bên trái chỉ là một hàm của quá trình sinh dữ liệu, không phải của mô hình. Điều này có nghĩa là khi ta huấn luyện mô hình để cực tiểu hóa độ phân kỳ KL, chúng ta chỉ cần cực tiểu hóa

$$-E\left[\log p_{\text{model}}(X)\right] \tag{5.57}$$

mà hiển nhiên cũng chính là bài toán cực đại hóa trong phương trình (5.55).

Việc tối thiểu hóa độ phân kỳ KL này tương ứng chính xác với việc tối thiểu hóa hàm cross–entropy giữa các phân phối. Nhiều tác giả sử dụng thuật ngữ "cross–entropy" để chỉ cụ thể đối của logarit hàm hợp lý của phân phối Bernoulli hoặc softmax, nhưng đó là một cách gọi chưa chính xác. Bất kỳ hàm mất mát nào bao gồm đối của logarit hàm hợp lý âm đều là một dạng cross–entropy giữa phân phối thực nghiệm xác định bởi tập huấn luyện và phân phối xác suất xác định bởi mô hình. Ví dụ, sai số bình phương trung bình là cross–entropy chéo giữa phân phối thực nghiệm và một mô hình Gauss.

Do đó, ta có thể xem phương pháp hợp lý cực đại như một nỗ lực để làm cho phân phối của mô hình khớp với phân phối kinh nghiệm $\hat{p}_{\text{data}}$. Lý tưởng nhất, ta muốn khớp với phân phối sinh dữ liệu thực sự p_{data}, nhưng ta vốn không thể truy cập trực tiếp vào phân phối này.

Trong khi giá trị tối ưu của $\boldsymbol{\theta}$ là như nhau dù chúng ta cực đại hóa hàm hợp lý hay cực tiểu hóa độ phân kỳ KL, các giá trị của hàm mục tiêu lại khác nhau. Trong phần mềm, chúng ta thường diễn đạt cả hai cách tiếp cận này dưới dạng cực tiểu hóa một hàm chi phí. Do đó, phương pháp hợp lý cực đại trở thành cực tiểu hóa đối của logarit hàm hợp lý, hoặc tương đương là cực tiểu hóa cross – entropy. Quan điểm coi hợp lý cực đại là độ phân kỳ KL cực tiểu hữu ích trong trường hợp này vì độ phân kỳ KL có giá trị tối thiểu được biết là bằng không. Hàm đối của logarit hàm hợp lý hợp lý có thể có giá trị âm khi $\boldsymbol{x}$ nhận giá trị thực.

5.5.1 Logarithm của hàm hợp lý có điều kiện và sai số bình phương trung bình

Bộ ước lượng hợp lý cực đại có thể được tổng quát hóa dễ dàng trong trường hợp mục tiêu là ước lượng xác suất có điều kiện $P\left(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}\right)$ nhằm dự đoán $\boldsymbol{y}$ khi biết $\boldsymbol{x}$. Đây thực sự là tình huống phổ biến nhất vì nó tạo nền tảng cho hầu hết các bài toán học có giám sát. Nếu X đại diện cho tất cả các đầu vào và Y đại diện cho tất cả các mục tiêu quan sát được, thì bộ ước lượng hợp lý cực đại có điều kiện là

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} P\left(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta}\right). \tag{5.58}$$

Nếu các mẫu được giả định là độc lập cùng phân phối, thì biểu thức này có thể được phân tích thành

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P\left(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right). \tag{5.59}$$

Ví dụ: Hồi quy tuyến tính như là ước lượng hợp lý cực đại Hồi quy tuyến tính, được giới thiệu trước đó trong Mục 5.1.4, có thể được chứng minh như một quy trinhf hợp lý cực đại. Trước đó, ta đã xem xét hồi quy tuyến tính như một thuật toán học cách lấy một đầu vào $\boldsymbol{x}$ và tạo ra một giá trị đầu ra $\hat{y}$. Ánh xạ tương ứng x với $\hat{y}$ được chọn để cực tiểu hóa sai số bình phương trung bình, một tiêu chí mà chúng ta đưa ra khá tùy ý. Giờ đây, chúng ta xem xét lại hồi quy tuyến tính từ quan điểm của ước lượng hợp lý cực đại. Thay vì chỉ đưa ra một dự đoán duy nhất $\hat{y}$, giờ đây ta

xem mô hình là đưa ra một phân phối có điều kiện $p\left(y \mid \boldsymbol{x}\right)$. Ta có thể hình dung rằng với một tập huấn luyện vô cùng lớn, ta có thể thấy một số mẫu huấn luyện với cùng giá trị đầu vào $\boldsymbol{x}$ nhưng giá trị $\boldsymbol{y}$ khác nhau. Mục tiêu của thuật toán học bây giờ là làm khớp phân phối $p\left(y \mid \boldsymbol{x}\right)$ với tất cả các giá trị y khác nhau tương thích với x đó. Để suy ra cùng thuật toán hồi quy tuyến tính mà ta đã phát triển trước đó, ta định nghĩa $p\left(y \mid x\right) = N\left(y; \widehat{y}\left(\boldsymbol{x}; \boldsymbol{w}\right), \sigma^2\right)$. Hàm $\widehat{y}\left(\boldsymbol{x}; \boldsymbol{w}\right)$ cung cấp dự đoán của trung bình của phân phối Gauss. Trong ví dụ này, ta giả định phương sai là một hằng số cố định σ^2 do người dùng chọn. Ta sẽ thấy rằng lựa chọn dạng của hàm $p\left(y \mid \boldsymbol{x}\right)$ này sẽ khiến thủ tục ước lượng hợp lý cực đại tạo ra cùng thuật toán học mà ta đã phát triển trước đó. Vì các mẫu được giả định là độc lập cùng phân phối, logarit của hàm hợp lý có điều kiện (phương trình (5.59)) được tính là

$$\sum_{i=1}^{m} \log p\left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) = -m \log \sigma - \frac{m}{2} \log\left(2\pi\right) - \sum_{i=1}^{m} \frac{1}{2\sigma^2} \left|\widehat{y}^{(i)} - y^{(i)}\right|^2, \quad (5.60)$$

trong đó $\widehat{y}^{(i)}$ là đầu ra của hồi quy tuyến tính trên đầu vào $\boldsymbol{x}^{(i)}$, và m là số lượng mẫu huấn luyện. So sánh logarit của hàm hợp lý với sai số bình phương trung bình,

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{m} \sum_{i=1}^{m} \left|\widehat{y}^{(i)} - y^{(i)}\right|^2, \quad (5.61)$$

ta ngay lập tức thấy rằng việc cực đại hóa logarit của hàm hợp lý theo $\boldsymbol{w}$ cho ra cùng một ước lượng các tham số $\boldsymbol{w}$ như việc cực tiểu hóa sai số bình phương trung bình. Hai tiêu chí có giá trị khác nhau nhưng cùng vị trí tối ưu. Điều này chứng minh cho việc sử dụng sai số bình phương trung bình như là một thủ tục ước lượng hợp lý cực đại. Như ta sẽ thấy, bộ ước lượng hợp lý cực đại có một số tính chất mong muốn.

5.5.2 Tính chất của ước lượng hợp lý cực đại

Sức hấp dẫn chính của ước lượng hợp lý cực đại là nó được chứng minh là ước lượng tốt nhất một cách tiệm cận, khi số lượng ví dụ $m \to \infty$, về tốc độ hội tụ khi m tăng.

Dưới các điều kiện thích hợp, ước lượng hợp lý cực đại có tính chất vững (xem Mục 5.4.5 ở trên), nghĩa là khi số lượng ví dụ huấn luyện tiến tới vô cùng, ước lượng hợp lý cực đại của một tham số sẽ hội tụ đến giá trị thực của tham số đó. Các điều kiện này là:

- Phân phối thực sự p_{data} phải nằm trong họ mô hình $p_{\mathrm{model}}(\cdot\,; \boldsymbol{\theta})$. Nếu không, không ước lượng nào có thể khôi phục lại p_{data}.

- Phân phối thực sự p_{data} phải tương ứng với duy nhất một giá trị của $\boldsymbol{\theta}$. Nếu không, ước lượng hợp lý cực đại có thể khôi phục chính xác p_{data}, nhưng không thể xác định giá trị $\boldsymbol{\theta}$ nào đã được sử dụng trong quá trình sinh dữ liệu.

Có nhiều nguyên lý quy nạp khác ngoài ước lượng hợp lý cực đại, trong đó nhiều nguyên lý có tính chất vững. Tuy nhiên, các ước lượng vững có thể khác nhau về **hiệu quả thống kê**, nghĩa là một ước lượng vững có thể có sai số tổng quát hóa thấp hơn với một số lượng mẫu cố định m, hoặc tương đương, có thể yêu cầu ít ví dụ hơn để đạt được mức sai số tổng quát hóa cố định.

Hiệu quả thống kê thường được nghiên cứu trong **trường hợp có tham số** (như trong hồi quy tuyến tính) khi mục tiêu là ước lượng giá trị của một tham số (và giả sử rằng có thể xác định tham số thực sự), không phải giá trị của một hàm. Một cách đo mức độ gần của ước lượng với tham số thực sự là thông qua kỳ vọng của sai số bình phương trung bình, tính bình phương độ lệch giữa giá trị tham số ước lượng và tham số thực sự, trong đó kỳ vọng được lấy qua m mẫu huấn luyện từ phân phối sinh dữ liệu. Sai số bình phương trung bình có tham số đó giảm khi m tăng, và đối với m lớn, cận dưới Cramér – Rao (*Information and the Accuracy Attainable in the Estimation of Statistical Parameters*, Rao, 1992, [36]; *Mathematical Methods of Statistics*, Cramér, 2016, [37]) chỉ ra rằng không có ước lượng vững nào có sai số bình phương trung bình thấp hơn ước lượng hợp lý cực đại.

Vì những lý do này (tính vững và hiệu quả), ước lượng hợp lý cực đại thường được coi là ước lượng ưu tiên sử dụng cho học máy. Khi số lượng ví dụ nhỏ đến mức có thể gây ra hiện tượng quá khớp, các chiến lược điều chuẩn như suy giảm trọng số có thể được sử dụng để thu được một phiên bản bị lệch của ước lượng hợp lý cực đại có phương sai nhỏ hơn khi dữ liệu huấn luyện bị hạn chế.

5.6 Thống kê Bayes

Cho đến nay, chúng ta đã thảo luận về **thống kê tần suất** và các phương pháp dựa trên việc ước lượng một giá trị duy nhất của $\boldsymbol{\theta}$, sau đó sử dụng tất cả các dự đoán dựa trên ước lượng đó. Một phương pháp khác là xem xét tất cả các giá trị có thể có của $\boldsymbol{\theta}$ khi đưa ra một dự đoán. Phương pháp này thuộc lĩnh vực **thống kê Bayes**.

Như đã thảo luận trong Mục 5.4.1, quan điểm của tần suất học là giá trị tham số thực sự $\boldsymbol{\theta}$ là cố định nhưng chưa biết, trong khi ước lượng điểm $\widehat{\boldsymbol{\theta}}$ là một biến

ngẫu nhiên do nó là hàm của bộ dữ liệu (được coi là ngẫu nhiên).

Quan điểm của Bayes về thống kê khá khác biệt. Bayes sử dụng xác suất để phản ánh mức độ chắc chắn của các trạng thái kiến thức. Bộ dữ liệu được quan sát trực tiếp và do đó không phải là ngẫu nhiên. Ngược lại, tham số thực sự $\boldsymbol{\theta}$ là chưa biết hoặc không chắc chắn và do đó được biểu diễn như một biến ngẫu nhiên.

Trước khi quan sát dữ liệu, chúng ta biểu diễn kiến thức của mình về $\boldsymbol{\theta}$ bằng cách sử dụng **phân phối xác suất tiên nghiệm**, $p\left(\boldsymbol{\theta}\right)$ (đôi khi được gọi đơn giản là "tiên nghiệm"). Nói chung, những nhà thực hành học máy chọn phân phối tiên nghiệm khá rộng (tức là có entropy cao) để phản ánh mức độ không chắc chắn cao về giá trị của $\boldsymbol{\theta}$ trước khi quan sát bất kỳ dữ liệu nào. Ví dụ, người ta có thể giả định một "*tiên nghiệm*" rằng $\boldsymbol{\theta}$ nằm trong một khoảng hoặc một khối hữu hạn, với một phân phối đều. Nhiều tiên nghiệm khác phản ánh sự ưu tiên cho các nghiệm "đơn giản" hơn (chẳng hạn như hệ số có độ lớn nhỏ hơn hoặc một hàm gần với hàm hằng hơn).

Giờ ta hãy xét một tập mẫu dữ liệu $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$. Ta có thể khôi phục tác động của dữ liệu đối với sự tin cậy của mình về $\boldsymbol{\theta}$ bằng cách kết hợp xác suất hợp lý của dữ liệu $p\left(x^{(1)}, \ldots, x^{(m)} \mid \theta\right)$ với phân phối tiên nghiệm thông qua quy tắc Bayes:

$$p\left(\boldsymbol{\theta} \mid x^{(1)}, \ldots, x^{(m)}\right) = \frac{p\left(x^{(1)}, \ldots, x^{(m)} \mid \boldsymbol{\theta}\right)\ p\left(\boldsymbol{\theta}\right)}{p\left(x^{(1)}, \ldots, x^{(m)}\right)} \tag{5.62}$$

Trong các trường hợp mà ước lượng Bayes thường được sử dụng, phân phối tiên nghiệm bắt đầu bởi một phân phối tương đối gần với phân phối đều hoặc phân phối Gauss với entropy cao, và sự quan sát dữ liệu thường khiến phân phối hậu nghiệm giảm entropy và tập trung quanh một vài giá trị của các tham số có xác suất cao.

So sánh với phương pháp ước lượng hợp lý cực đại, ước lượng Bayes có hai khác biệt quan trọng. Đầu tiên, không giống như phương pháp hợp lý cực đại sử dụng một điểm ước lượng của $\boldsymbol{\theta}$ để đưa ra dự đoán, phương pháp Bayes sử dụng phân phối đầy đủ của θ. Chẳng hạn, sau khi quan sát m mẫu, phân phối dự đoán cho mẫu dữ liệu tiếp theo, $x^{(m+1)}$, được cho bởi:

$$p\left(x^{(m+1)} \mid x^{(1)}, \ldots, x^{(m)}\right) = \int p\left(x^{(m+1)} \mid \boldsymbol{\theta}\right)\ p\left(\boldsymbol{\theta} \mid x^{(1)}, \ldots, x^{(m)}\right)\ d\boldsymbol{\theta}. \tag{5.63}$$

Ở đây, mỗi giá trị của $\boldsymbol{\theta}$ có mật độ xác suất dương đều đóng góp vào dự đoán cho mẫu tiếp theo, với trọng số đóng góp bằng mật độ hậu nghiệm của chính nó. Sau khi đã quan sát $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$, nếu ta vẫn còn khá không chắc chắn về giá trị

của $\boldsymbol{\theta}$, thì tính không chắc chắn này sẽ được tích hợp trực tiếp vào bất kỳ dự đoán nào mà ta có thể thực hiện.

Trong Mục 5.4, chúng ta đã thảo luận về cách phương pháp tần suất xử lý tính không chắc chắn của một ước lượng điểm của $\boldsymbol{\theta}$ bằng cách đánh giá phương sai của nó. Phương sai của ước lượng là một đánh giá về việc ước lượng có thể thay đổi thế nào với các lần lấy mẫu khác nhau từ dữ liệu quan sát. Cách tiếp cận Bayes đối với câu hỏi làm thế nào để xử lý tính không chắc chắn trong ước lượng chỉ đơn giản là tích phân theo nó, điều này giúp chống lại hiện tượng quá khớp tốt hơn. Tích phân này tất nhiên chỉ là một ứng dụng của các quy luật xác suất, làm cho phương pháp Bayes trở nên dễ dàng xác minh, trong khi cơ chế của tần suất để xây dựng một ước lượng dựa trên quyết định khá tùy tiện khi tóm tắt toàn bộ thông tin trong tập dữ liệu chỉ bằng một ước lượng điểm.

Sự khác biệt quan trọng thứ hai giữa phương pháp Bayes và phương pháp hợp lý cực đại là do sự đóng góp của phân phối tiên nghiệm Bayes. Phân phối tiên nghiệm có ảnh hưởng bằng cách dịch chuyển mật độ xác suất về phía các miền của không gian tham số được ưu tiên "tiên nghiệm". Trong thực tế, phân phối tiên nghiệm thường thể hiện sự ưu tiên cho các mô hình đơn giản hoặc trơn hơn. Những nhà phê bình cách tiếp cận Bayes coi phân phối tiên nghiệm là một nguồn ảnh hưởng mang tính chủ quan của con người đến dự đoán.

Các phương pháp Bayes thường tổng quát hóa tốt hơn khi có dữ liệu huấn luyện hạn chế, nhưng thường chịu chi phí tính toán cao khi số lượng mẫu huấn luyện lớn.

Ví dụ: Hồi quy Tuyến tính Bayesian Ở đây, chúng ta xem xét phương pháp ước lượng Bayes để học các tham số hồi quy tuyến tính. Trong hồi quy tuyến tính, chúng ta học một phép ánh xạ tuyến tính tương ứng một véctơ đầu vào $\boldsymbol{x} \in \mathbb{R}^n$ để dự đoán giá trị của một vô hướng $y \in \mathbb{R}$. Dự đoán này được tham số hóa bởi véctơ $\boldsymbol{w} \in \mathbb{R}^n$:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x}. \tag{5.64}$$

Cho một tập hợp gồm m mẫu huấn luyện $\left(\boldsymbol{X}^{(\text{train})}, \boldsymbol{Y}^{(\text{train})}\right)$, ta có thể biểu diễn dự đoán của y trên toàn bộ tập huấn luyện như sau:

$$\hat{\boldsymbol{Y}}^{(\text{train})} = \boldsymbol{X}^{(\text{train})} \boldsymbol{w}. \tag{5.65}$$

Biểu diễn dưới dạng phân phối có điều kiện Gauss trên $\boldsymbol{Y}^{(\text{train})}$, ta có:

$$p\left(\boldsymbol{Y}^{(\text{train})} \mid \boldsymbol{X}^{(\text{train})}, w\right) = N\left(\boldsymbol{Y}^{(\text{train})}; \boldsymbol{X}^{(\text{train})} \boldsymbol{w}, \boldsymbol{I}\right) \tag{5.66}$$

$$\propto \exp\left(-\frac{1}{2}\left(\boldsymbol{Y}^{(\text{train})} - \boldsymbol{X}^{(\text{train})}\boldsymbol{w}\right)^T \left(\boldsymbol{Y}^{(\text{train})} - \boldsymbol{X}^{(\text{train})}\boldsymbol{w}\right)\right), \tag{5.67}$$

trong đó, ta tuân theo định nghĩa chuẩn của MSE bằng cách giả định rằng phương sai phân phối Gauss trên y bằng 1. Để giảm bớt gánh nặng ký hiệu, ta sẽ gọi $\left(\boldsymbol{X}^{(\text{train})}, \boldsymbol{Y}^{(\text{train})}\right)$ đơn giản là $(\boldsymbol{X}, \boldsymbol{Y})$.

Để xác định phân phối hậu nghiệm trên véctơ tham số mô hình $\boldsymbol{w}$, trước hết, ta cần chỉ định một phân phối tiên nghiệm. Phân phối tiên nghiệm này sẽ phản ánh niềm tin ban đầu của chúng ta về giá trị của các tham số này. Mặc dù đôi khi khó hoặc không tự nhiên để diễn tả các niềm tin tiên nghiệm bằng các tham số của mô hình, trong thực tế, ta thường giả định một phân phối khá rộng để thể hiện mức độ không chắc chắn cao về $\boldsymbol{\theta}$. Đối với các tham số có giá trị thực, phân phối Gauss thường được sử dụng làm phân phối tiên nghiệm:

$$p(\boldsymbol{w}) = N\left(\boldsymbol{w}; \boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0\right) \propto \exp\left(-\frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{\mu}_0\right)^T \boldsymbol{\Lambda}_0^{-1} \left(\boldsymbol{w} - \boldsymbol{\mu}_0\right)\right), \tag{5.68}$$

trong đó $\boldsymbol{\mu}_0$ và $\boldsymbol{\Lambda}_0$ lần lượt là véctơ trung bình và ma trận hiệp phương sai của phân phối tiên nghiệm.*

Với phân phối tiên nghiệm đã được xác định, chúng ta có thể tiến hành xác định phân phối **hậu nghiệm** trên các tham số mô hình như sau:

$$p\left(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}\right) \tag{5.69}$$

$$\propto p\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}\right)\ p(\boldsymbol{w}) \tag{5.70}$$

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w})\right)\ \exp\left(-\frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{\mu}_0\right) \boldsymbol{\Lambda}_0^{-1} \left(\boldsymbol{w} - \boldsymbol{\mu}_0\right)\right) \tag{5.71}$$

$$\propto \exp\left(-\frac{1}{2}\left(-2\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{\Lambda}_0^{-1}\boldsymbol{w} - 2\boldsymbol{\mu}_0^T\boldsymbol{\Lambda}_0^{-1}\boldsymbol{w}\right)\right). \tag{5.72}$$

Ta định nghĩa $\boldsymbol{\Lambda}_m = \left(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Lambda}_0^{-1}\right)^{-1}$ và $\boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m\left(\boldsymbol{X}^T\boldsymbol{Y} + \boldsymbol{\Lambda}_0^{-1}\boldsymbol{\mu}_0\right)$. Sử dụng các biến mới này, ta có thể viết lại phân phối hậu nghiệm dưới dạng phân phối Gauss:

$$p\left(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}\right) \propto \exp\left(-\frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{\mu}_m\right)^T \boldsymbol{\Lambda}_m^{-1} \left(\boldsymbol{w} - \boldsymbol{\mu}_m\right) + \frac{1}{2}\boldsymbol{\mu}_m^T\boldsymbol{\Lambda}_m^{-1}\boldsymbol{\mu}_m\right) \tag{5.73}$$

*Trừ khi có lý do để giả định một cấu trúc hiệp phương sai cụ thể, thông thường ta giả định ma trận hiệp phương sai là ma trận đường chéo: $\boldsymbol{\Lambda}_0 = \text{diag}\left(\boldsymbol{\lambda}_0\right)$.

$$\propto \exp\left(-\frac{1}{2}\left(\boldsymbol{w}-\boldsymbol{\mu}_m\right)^T \boldsymbol{\Lambda}_m^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_m\right)\right). \tag{5.74}$$

Tất cả các hạng tử không chứa véctơ tham số $\boldsymbol{w}$ đã bị lược bỏ; chúng được ngầm định bởi thực tế là phân phối phải được chuẩn hóa sao cho tích phân của nó bằng 1. Phương trình (3.26) chỉ ra cách chuẩn hóa một phân phối Gauss nhiều chiều.

Nhận xét về phân phối hậu nghiệm giúp ta có cái nhìn trực quan về ảnh hưởng của suy luận Bayes. Trong hầu hết các trường hợp, ta đặt $\boldsymbol{\mu}_0 = \boldsymbol{0}$. Nếu ta đặt $\boldsymbol{\Lambda}_0 = \frac{1}{\alpha}\boldsymbol{I}$, thì $\boldsymbol{\mu}_m$ cung cấp cùng một ước lượng của $\boldsymbol{w}$ như hồi quy tuyến tính theo quan điểm tần suất với một hạng tử phạt suy giảm trọng số $\alpha \boldsymbol{w}^T \boldsymbol{w}$. Một điểm khác biệt là ước lượng Bayes sẽ không xác định nếu ta đặt $\alpha = 0$—ta không thể bắt đầu quá trình học Bayes với một tiên nghiệm quá rộng cho $\boldsymbol{w}$. Điểm khác biệt quan trọng hơn là ước lượng Bayes cung cấp một ma trận hiệp phương sai, cho thấy khả năng các giá trị khác nhau của $\boldsymbol{w}$, thay vì chỉ cung cấp ước lượng $\boldsymbol{\mu}_m$.

5.6.1 Ước lượng hậu nghiệm cực đại

Mặc dù cách tiếp cận nguyên tắc nhất là dự đoán bằng cách sử dụng phân phối hậu nghiệm đầy đủ của tham số $\boldsymbol{\theta}$, nhưng trong thực tế, ta vẫn thường muốn có một ước lượng điểm duy nhất. Một lý do phổ biến để mong muốn có ước lượng điểm là vì hầu hết các thao tác với phân phối hậu nghiệm Bayes trong các mô hình phức tạp đều không thể tính toán được, và một ước lượng điểm sẽ cung cấp một xấp xỉ có thể tính toán. Thay vì đơn giản quay lại với ước lượng hợp lý cực đại, ta vẫn có thể tận dụng lợi ích của phương pháp Bayes bằng cách cho phép tiên nghiệm ảnh hưởng đến việc lựa chọn ước lượng điểm. Một cách hợp lý để làm điều này là chọn ước lượng **hậu nghiệm cực đại** (MAP). MAP chọn điểm có xác suất hậu nghiệm cực đại (hoặc mật độ xác suất cực đại trong trường hợp phổ biến hơn với $\boldsymbol{\theta}$ liên tục):

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p\left(\boldsymbol{\theta} \mid \boldsymbol{x}\right) = \arg\max_{\boldsymbol{\theta}} \left(\log \max p\left(\boldsymbol{x} \mid \boldsymbol{\theta}\right) + \log p\left(\boldsymbol{\theta}\right)\right). \tag{5.75}$$

Ở vế phải của phương trình trên, ta thấy $\log p\left(\boldsymbol{x} \mid \boldsymbol{\theta}\right)$, tức là logarit của hàm hợp lý chuẩn, và $\log p\left(\boldsymbol{\theta}\right)$, tương ứng với phân phối tiên nghiệm.

Ví dụ, xét một mô hình hồi quy tuyến tính với phân phối tiên nghiệm Gauss trên các trọng số $\boldsymbol{w}$. Nếu tiên nghiệm này được cho bởi $N\left(\boldsymbol{w}; \boldsymbol{0}, \frac{1}{\lambda}\boldsymbol{I}^2\right)$, thì hạng tử logarit của phân phối tiên nghiệm trong phương trình trên sẽ tỷ lệ thuận với hạng

tử phạt suy giảm trọng số $\lambda \boldsymbol{w}^T \boldsymbol{w}$, cộng với một hạng tử không phụ thuộc vào $\boldsymbol{w}$ và không ảnh hưởng đến quá trình học. Như vậy, suy luận Bayes MAP với phân phối tiên nghiệm Gauss trên trọng số tương đương với việc áp dụng suy giảm trọng số.

Giống như suy luận Bayes đầy đủ, suy luận Bayes MAP có lợi thế trong việc tận dụng thông tin từ tiên nghiệm mà không thể tìm thấy trong dữ liệu huấn luyện. Thông tin bổ sung này giúp giảm phương sai trong ước lượng điểm MAP (so với ước lượng hợp lý cực đại). Tuy nhiên, điều này phải trả giá bằng việc tăng độ lệch.

Nhiều chiến lược ước lượng có điều chỉnh, chẳng hạn như học hợp lý cực đại được điều chỉnh bằng suy giảm trọng số, có thể được hiểu là xấp xỉ MAP cho suy luận Bayes. Quan điểm này áp dụng khi điều chỉnh bao gồm việc thêm một hạng tử vào hàm mục tiêu tương ứng với $\log p\left(\boldsymbol{\theta}\right)$. Tuy nhiên, không phải mọi hạng tử phạt điều chỉnh đều tương ứng với suy luận Bayes MAP. Ví dụ, một số hạng tử điều chỉnh có thể không phải là logarit của một phân phối xác suất. Các hạng tử điều chỉnh khác có thể phụ thuộc vào dữ liệu, điều mà một phân phối xác suất tiên nghiệm không được phép.

Suy luận Bayes MAP cung cấp một cách trực quan để thiết kế các điều khoản điều chỉnh phức tạp nhưng dễ hiểu. Ví dụ, một hạng tử phạt phức tạp hơn có thể được suy ra bằng cách sử dụng một hỗn hợp các phân phối Gauss làm tiên nghiệm, thay vì một phân phối Gauss đơn lẻ (*Simplifying Neural Networks by Soft Weight-Sharing*, Nowlan và Hinton, 1992, [38]).

5.7 Thuật toán học có giám sát

Nhắc lại Mục 5.1.3 đã định nghĩa các thuật toán học có giám sát, nói một cách đơn giản, là các thuật toán học để liên kết một số đầu vào với một số đầu ra, dựa trên một tập huấn luyện gồm các ví dụ đầu vào $\boldsymbol{x}$ và đầu ra $\boldsymbol{y}$. Trong nhiều trường hợp, các đầu ra $\boldsymbol{y}$ có thể khó thu thập tự động và cần được cung cấp bởi một "người giám sát" là con người. Tuy nhiên, thuật ngữ này vẫn được sử dụng ngay cả khi các nhãn mục tiêu trong tập huấn luyện được thu thập tự động.

5.7.1 Học có giám sát xác suất

Hầu hết các thuật toán học có giám sát trong sách này đều dựa trên việc ước lượng phân phối xác suất $p\left(y \mid \boldsymbol{x}\right)$. Ta có thể thực hiện điều này đơn giản bằng cách sử dụng phương pháp ước lượng hợp lý cực đại để tìm ra véctơ tham số $\boldsymbol{\theta}$ tốt nhất cho một họ phân phối có tham số $p\left(y \mid \boldsymbol{x}; \boldsymbol{\theta}\right)$.

Ta đã thấy rằng hồi quy tuyến tính tương ứng với họ phân phối:

$$p\left(y \mid \boldsymbol{x}; \boldsymbol{\theta}\right) = N\left(y; \boldsymbol{\theta}^T \boldsymbol{x}, \boldsymbol{I}\right). \tag{5.76}$$

Ta có thể tổng quát hóa hồi quy tuyến tính sang trường hợp phân loại bằng cách định nghĩa một họ phân phối xác suất khác. Nếu ta có hai lớp, lớp 0 và lớp 1, thì chỉ cần xác định xác suất của một trong hai lớp này. Xác suất của lớp 1 xác định xác suất của lớp 0, vì hai giá trị này phải có tổng bằng 1.

Phân phối chuẩn trên các số thực mà ta sử dụng trong hồi quy tuyến tính được tham số hóa dưới dạng giá trị trung bình. Bất kỳ giá trị nào ta cung cấp cho giá trị trung bình này đều hợp lệ. Một phân phối trên một biến nhị phân phức tạp hơn một chút, vì giá trị trung bình của nó phải luôn nằm trong khoảng từ 0 đến 1. Một cách để giải quyết vấn đề này là sử dụng hàm sigmoid logistic để nén đầu ra của hàm tuyến tính vào khoảng $(0, 1)$ và diễn giải giá trị này như một xác suất:

$$p\left(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}\right) = \sigma\left(\boldsymbol{\theta}^T \boldsymbol{x}\right). \tag{5.77}$$

Phương pháp này được gọi là **hồi quy logistic** (một tên gọi hơi kỳ lạ vì ta dùng mô hình này cho phân loại chứ không phải hồi quy).

Trong trường hợp hồi quy tuyến tính, chúng ta có thể tìm các trọng số tối ưu bằng cách giải phương trình chuẩn tắc. Hồi quy logistic phức tạp hơn một chút. Không có lời giải dưới dạng công thức đóng cho các trọng số tối ưu của nó. Thay vào đó, chúng ta phải tìm các trọng số này bằng cách cực đại hóa logarit của hàm hợp lý. Ta có thể thực hiện điều này bằng cách cực tiểu hóa giá trị đối của logarit hàm hợp lý bằng cách sử dụng phương pháp hướng giảm.

Chiến lược này có thể được áp dụng cho hầu hết mọi vấn đề học có giám sát, bằng cách viết ra một họ phân phối xác suất có điều kiện có tham số phù hợp với loại biến đầu vào và đầu ra mong muốn.

5.7.2 Phương pháp véctơ hỗ trợ

Một trong những phương pháp có ảnh hưởng lớn đến học có giám sát là phương pháp véctơ hỗ trợ (support vector machine, SVM) (*A Training Algorithm for Optimal Margin Classifiers*, Boser và cộng sự, 1992, [39]; *Support-Vector Networks*, Cortes và Vapnik, 1995, [40]). Mô hình này tương tự hồi quy logistic ở chỗ nó được dẫn dắt bởi một hàm tuyến tính $\boldsymbol{w}^T \boldsymbol{x} + b$. Tuy nhiên, không giống như hồi quy logistic, SVM không cung cấp các xác suất mà chỉ đưa ra danh tính của lớp. SVM dự đoán

rằng lớp dương tồn tại khi $\boldsymbol{w}^T\boldsymbol{x} + b$ là dương. Tương tự, nó dự đoán rằng lớp âm tồn tại khi $\boldsymbol{w}^T\boldsymbol{x} + b$ là âm.

Một đổi mới quan trọng liên quan đến SVM là **thủ thuật kernel**. Thủ thuật kernel bao gồm việc nhận thấy rằng nhiều thuật toán học máy có thể được viết hoàn toàn dưới dạng các tích vô hướng giữa các ví dụ. Chẳng hạn, có thể chứng minh rằng hàm tuyến tính được SVM sử dụng có thể được viết lại thành:

$$\boldsymbol{w}^T\boldsymbol{x} + b = b + \sum_{i=1}^{m} \alpha_i \boldsymbol{x}^T \boldsymbol{x}^{(i)} \tag{5.78}$$

trong đó $\boldsymbol{x}^{(i)}$ là một ví dụ trong tập huấn luyện và $\boldsymbol{\alpha}$ là một véctơ các hệ số. Việc viết lại thuật toán học theo cách này cho phép ta thay $\boldsymbol{x}$ bằng đầu ra của một hàm đặc trưng nhất định $\phi(\boldsymbol{x})$ và tích vô hướng bằng một hàm $k\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right) = \phi(\boldsymbol{x}) \cdot \phi\left(\boldsymbol{x}^{(i)}\right)$, được gọi là một kernel. Toán tử $\cdot$ biểu diễn một tích trong tương tự như $\phi(\boldsymbol{x})^T \phi\left(\boldsymbol{x}^{(i)}\right)$. Đối với một số không gian đặc trưng, ta không nhất thiết phải sử dụng tích trong của véctơ một cách trực tiếp. Trong một số không gian vô hạn chiều, ta cần sử dụng các loại tích trong khác, ví dụ, tích trong dựa trên tích phân thay vì tổng. Một phát triển đầy đủ về các loại tích trong này nằm ngoài phạm vi của cuốn sách này.

Sau khi thay thế các tích vô hướng bằng các phép tính kernel, ta có thể đưa ra dự đoán sử dụng hàm

$$f(\boldsymbol{x}) = b + \sum_i \alpha_i \, k\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right). \tag{5.79}$$

Hàm này là phi tuyến theo $\boldsymbol{x}$, nhưng mối quan hệ giữa $\phi(\boldsymbol{x})$ và $f(\boldsymbol{x})$ lại là tuyến tính. Đồng thời, mối quan hệ giữa $\boldsymbol{\alpha}$ và $f(\boldsymbol{x})$ cũng là tuyến tính. Hàm dựa trên kernel tương đương chính xác với việc tiền xử lý dữ liệu bằng cách áp dụng $\phi(\boldsymbol{x})$ lên tất cả các đầu vào, sau đó học một mô hình tuyến tính trong không gian đã được biến đổi.

Thủ thuật kernel mạnh mẽ vì hai lý do. Thứ nhất, nó cho phép chúng ta học các mô hình phi tuyến theo hàm của $\boldsymbol{x}$ bằng các kỹ thuật tối ưu lồi, vốn đảm bảo hội tụ một cách hiệu quả. Điều này khả thi bởi vì ta coi ϕ là cố định và chỉ tối ưu hóa $\boldsymbol{\alpha}$, tức là thuật toán tối ưu có thể xem hàm quyết định là tuyến tính trong một không gian khác. Thứ hai, hàm kernel k thường có thể được triển khai một cách hiệu quả về mặt tính toán hơn nhiều so với việc tạo trực tiếp hai véctơ $\phi(\boldsymbol{x})$ và lấy tích vô hướng của chúng.

Trong một số trường hợp, $\phi(\boldsymbol{x})$ có thể thậm chí là vô hạn chiều, điều này sẽ dẫn đến chi phí tính toán vô hạn nếu sử dụng cách tiếp cận trực tiếp và rõ ràng.

Trong nhiều trường hợp, $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ là một hàm phi tuyến và có thể tính toán được theo $\boldsymbol{x}$ ngay cả khi không xử lý được $\phi(\boldsymbol{x})$. Một ví dụ về không gian đặc trưng vô hạn chiều với một kernel khả thi là chúng ta xây dựng một ánh xạ đặc trưng $\phi(x)$ trên các số nguyên không âm x. Giả sử ánh xạ này trả về một véctơ chứa x phần tử bằng một, tiếp theo là vô số số không. Ta có thể viết một hàm kernel $k\left(x, x^{(i)}\right) = \min\left(x, x^{(i)}\right)$, tương đương chính xác với tích vô hướng trong không gian vô hạn chiều tương ứng.

Kernel phổ biến nhất được sử dụng là **kernel Gauss**:

$$k(\boldsymbol{u}, \boldsymbol{v}) = N\left(\boldsymbol{u} - \boldsymbol{v}, \mathbf{0}, \sigma^2 \boldsymbol{I}\right) \tag{5.80}$$

với $N\left(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ là mật độ chuẩn. Kernel này còn được gọi là kernel **hàm cơ sở xuyên tâm** (RBF), vì giá trị của nó giảm dần dọc theo các đường tỏa ra trong không gian v từ u. Kernel Gauss tương ứng với tích vô hướng trong không gian vô hạn chiều, nhưng phép suy luận không đơn giản như ví dụ kernel min trên các số nguyên.

Chúng ta có thể nghĩ về kernel Gauss như là thực hiện một loại **khớp mẫu**. Một mẫu huấn luyện $\boldsymbol{x}$ đi kèm với nhãn huấn luyện y trở thành một mẫu cho lớp y. Khi một điểm thử nghiệm $\boldsymbol{x}'$ gần $\boldsymbol{x}$ theo khoảng cách Euclid, kernel Gauss có phản hồi cao, cho thấy rằng $\boldsymbol{x}'$ rất giống với mẫu $\boldsymbol{x}$. Mô hình sau đó gán trọng số lớn cho nhãn huấn luyện tương ứng y. Tổng quan, dự đoán sẽ kết hợp nhiều nhãn huấn luyện như vậy, được đánh trọng số theo độ tương đồng của các mẫu huấn luyện tương ứng.

SVM không phải là thuật toán duy nhất có thể được cải thiện bằng thủ thuật kernel. Nhiều mô hình tuyến tính khác cũng có thể được cải thiện theo cách này. Nhóm các thuật toán sử dụng thủ thuật kernel được gọi là các **máy kernel** hoặc **phương pháp kernel** (*Gaussian Processes for Regression*, Williams và Rasmussen, 1995, [41]; *Advances in Kernel Methods: Support Vector Learning*, Schölkopf và cộng sự, 1998, [42]).

Một nhược điểm lớn của các phương pháp kernel là chi phí tính toán hàm quyết định tăng tuyến tính theo số lượng mẫu huấn luyện, vì mẫu thứ i đóng góp một số hạng $\alpha_i k\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}\right)$ vào hàm quyết định. SVM có thể giảm bớt điều này bằng cách học một véctơ $\boldsymbol{\alpha}$ chủ yếu là số không. Khi đó, việc phân loại một mẫu mới chỉ cần đánh giá hàm kernel cho các mẫu huấn luyện có α_i khác không. Những mẫu huấn luyện này được gọi là các véctơ hỗ trợ.

Phương pháp kernel cũng gặp phải chi phí tính toán cao khi huấn luyện trên tập dữ liệu lớn. Ta sẽ xem xét lại ý tưởng này trong Mục 5.9. Phương pháp kernel

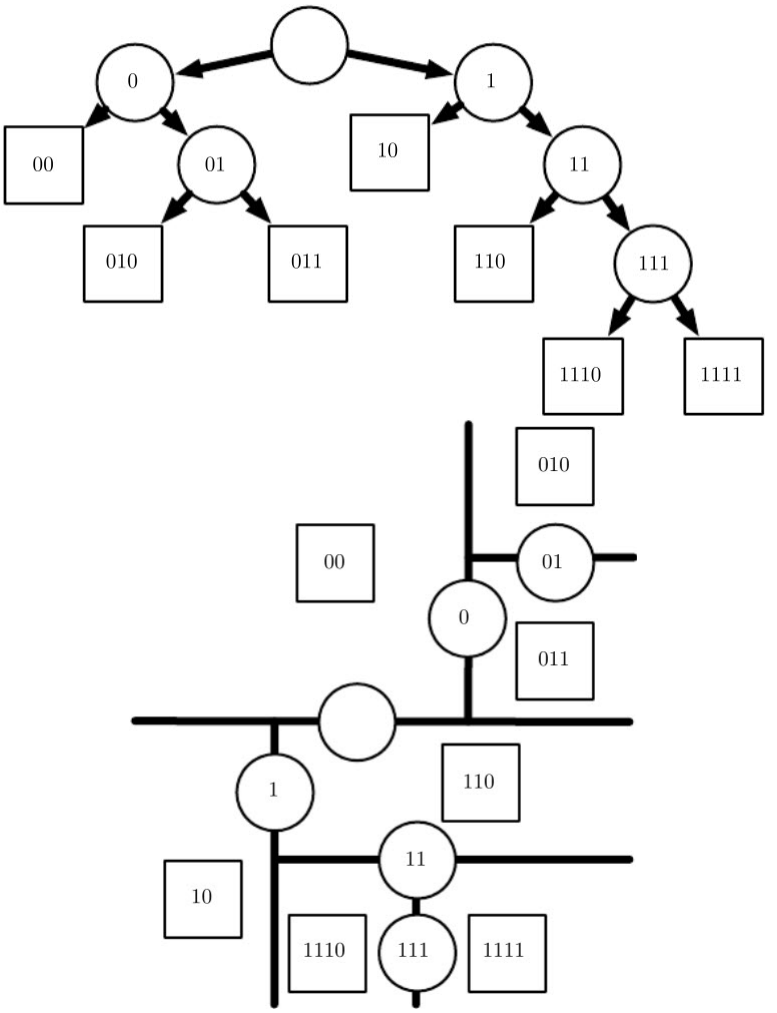
với các kernel chung gặp khó khăn trong việc tổng quát hóa tốt. Ta sẽ giải thích lý do tại Mục 5.11. Sự xuất hiện hiện đại của học sâu được thiết kế để vượt qua những hạn chế của phương pháp kernel. Kỷ nguyên học sâu hiện nay bắt đầu khi *A Fast Learning Algorithm for Deep Belief Nets*, (Hinton và cộng sự, 2006, [43]) chứng minh rằng mạng nơron có thể vượt trội hơn SVM với kernel RBF khi kiểm chuẩn trên dữ liệu MNIST.

5.7.3 Một số thuật toán học có giám sát đơn giản

Chúng ta đã gặp thoáng qua một thuật toán học có giám sát phi xác suất khác, hồi quy láng giềng gần nhất. Tổng quát hơn, $k-$láng giềng gần nhất ($k-$nearest neighbors, $k-$NN) là một họ các kỹ thuật có thể được sử dụng cho cả phân loại và hồi quy. Là một thuật toán học phi tham số, $k-$láng giềng gần nhất không bị giới hạn bởi một số lượng tham số cố định. Chúng ta thường nghĩ rằng thuật toán này không có tham số nào cả, mà chỉ thực hiện một hàm đơn giản của dữ liệu huấn luyện. Thực tế, thậm chí không có một giai đoạn huấn luyện hoặc quá trình học tập nào. Thay vào đó, ở thời điểm kiểm tra, khi muốn tạo ra đầu ra y cho một đầu vào kiểm tra mới $\boldsymbol{x}$, ta tìm k láng giềng gần nhất với $\boldsymbol{x}$ trong dữ liệu huấn luyện $\boldsymbol{X}$. Sau đó, ta trả về giá trị trung bình của các giá trị y tương ứng trong tập huấn luyện. Điều này hoạt động cho hầu hết các loại bài toán học có giám sát, nơi chúng ta có thể xác định trung bình trên các giá trị y. Trong trường hợp phân loại, ta có thể lấy trung bình các véctơ mã hóa one$-$hot $\boldsymbol{c}$ với $c_y = 1$ và $c_i = 0$ cho tất cả các giá trị khác của i. Sau đó, ta có thể diễn giải trung bình này dưới dạng một phân phối xác suất trên các lớp. Là một thuật toán học phi tham số, mô hình $k-$láng giềng gần nhất có thể đạt được năng lực rất cao. Ví dụ, giả sử ta có một bài toán phân loại đa lớp và đánh giá hiệu suất bằng hàm mất mát $0-1$. Trong trường hợp này, $1-$láng giềng gần nhất hội tụ tới hai lần sai số Bayes khi số lượng ví dụ huấn luyện tiến gần đến vô hạn. Phần sai số vượt quá sai số Bayes là kết quả của việc chọn một láng giềng duy nhất bằng cách ngẫu nhiên phá vỡ sự cân bằng giữa các láng giềng có khoảng cách bằng nhau. Khi có dữ liệu huấn luyện vô hạn, mọi điểm kiểm tra $\boldsymbol{x}$ sẽ có vô số láng giềng trong tập huấn luyện với khoảng cách bằng không. Nếu ta cho phép thuật toán sử dụng tất cả các láng giềng này để bỏ phiếu thay vì chọn ngẫu nhiên một trong số đó, thủ tục này sẽ hội tụ về hệ số sai số Bayes. Năng lực mô hình cao của $k-$láng giềng gần nhất cho phép nó đạt được độ chính xác cao với một tập huấn luyện lớn. Tuy nhiên, nhược điểm của nó là chi phí tính toán cao, và nó có thể giảm mạnh tính tổng quát khi tập huấn luyện nhỏ và hữu hạn. Một

điểm yếu của $k-$láng giềng gần nhất là nó không thể học được rằng một đặc trưng có tính phân biệt hơn đặc trưng khác. Ví dụ, giả sử ta có một bài toán hồi quy với $\boldsymbol{x} \in \mathbb{R}^{100}$ được lấy từ một phân phối Gauss đẳng hướng, nhưng chỉ một biến duy nhất x_1 là có liên quan đến đầu ra. Giả sử thêm rằng đặc trưng này đơn giản là mã hóa đầu ra trực tiếp, tức là $y = x_1$ trong mọi trường hợp. Hồi quy láng giềng gần nhất sẽ không thể phát hiện ra mô hình đơn giản này. Láng giềng gần nhất của hầu hết các điểm $\boldsymbol{x}$ sẽ được xác định bởi số lượng lớn các đặc trưng x_2 đến x_{100}, chứ không phải bởi đặc trưng đơn lẻ x_1. Do đó, đầu ra trên các tập huấn luyện nhỏ sẽ gần như là ngẫu nhiên.

Một loại thuật toán học khác cũng chia không gian đầu vào thành các miền và có các tham số riêng cho từng miền là **cây quyết định** (*Classification and Regression Trees*, Breiman và cộng sự, 1984, [44]) và nhiều biến thể của nó. Như được thể hiện trong Hình 5.7, mỗi nút của cây quyết định được gắn với một miền trong không gian đầu vào, và các nút bên trong chia miền đó thành một miền con cho mỗi con của nút (thường bằng cách cắt thẳng theo trục tọa độ). Không gian do đó được chia nhỏ thành các miền rời nhau, với một mối quan hệ một – một giữa các nút lá và các miền đầu vào. Mỗi nút lá thường ánh xạ mọi điểm trong miền đầu vào của nó đến cùng một đầu ra. Cây quyết định thường được huấn luyện với các thuật toán chuyên biệt vượt ra ngoài phạm vi của cuốn sách này. Thuật toán học có thể được coi là phi tham số nếu nó được phép học một cây có kích thước tùy ý, mặc dù trong thực tế cây quyết định thường được điều chỉnh với các ràng buộc về kích thước, biến chúng thành các mô hình có tham số. Các cây quyết định, khi được sử dụng với các phép chia thẳng theo trục tọa độ và đầu ra không đổi trong mỗi nút, gặp khó khăn khi giải quyết một số vấn đề mà ngay cả hồi quy logistic cũng có thể dễ dàng giải quyết. Ví dụ, nếu chúng ta có một bài toán hai lớp và lớp dương xuất hiện khi $x_2 > x_1$, biên quyết định không thẳng theo trục. Do đó, cây quyết định sẽ cần nhiều nút để xấp xỉ biên quyết định, thực hiện một hàm bước liên tục di chuyển qua lại dọc theo hàm quyết định thực với các bước thẳng theo trục tọa độ.



Hình 5.7: Các sơ đồ mô tả cách thức hoạt động của cây quyết định. *Hình trên*: Mỗi nút của cây quyết định sẽ chọn gửi ví dụ đầu vào tới nút con bên trái (0) hoặc nút con bên phải (1). Các nút bên trong được biểu diễn bằng hình tròn và các nút lá được biểu diễn bằng hình vuông. Mỗi nút được hiển thị bằng một xâu nhị phân làm mã định danh, tương ứng với vị trí của nó trong cây, nhận được bằng cách thêm một bit vào mã định danh của nút cha (0 = chọn bên trái hoặc trên, 1 = chọn bên phải hoặc dưới). *Hình dưới*: Cây quyết định chia không gian thành các miền. Mặt phẳng 2D cho thấy cách một cây quyết định có thể chia $\mathbb{R}^2$. Các nút của cây được vẽ trong mặt phẳng này, với mỗi nút bên trong được vẽ dọc theo đường phân chia mà nó sử dụng để phân loại các ví dụ, và các nút lá được vẽ ở trung tâm của miền chứa các ví dụ mà chúng nhận được. Kết quả là một hàm hằng từng nhánh, với mỗi nhánh tương ứng với một nút lá. Mỗi nút lá cần ít nhất một ví dụ huấn luyện để xác định, vì vậy cây quyết định không thể học một hàm có nhiều cực đại địa phương hơn số lượng ví dụ huấn luyện.

Như chúng ta đã thấy, các bộ dự đoán láng giềng gần nhất và cây quyết định có nhiều hạn chế. Tuy nhiên, chúng vẫn là các thuật toán học hữu ích khi tài nguyên tính toán bị hạn chế. Ta cũng có thể xây dựng trực giác cho các thuật toán học phức tạp hơn bằng cách suy nghĩ về những điểm tương đồng và khác biệt giữa các thuật toán tinh vi và các mô hình cơ sở như k – láng giềng gần nhất hoặc cây quyết định.

Xem *Machine Learning: A Probabilistic Perspective* (Murphy, 2012, [1]), *Pattern Recognition and Machine Learning* (Bishop, 2011, [2]), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Hastie và cộng sự, 2013, [45]) hoặc các sách giáo khoa học máy khác để có thêm tài liệu về các thuật toán học có giám sát truyền thống.

5.8 Thuật toán học không giám sát

Nhớ lại từ mục Mục 5.1.3 rằng các thuật toán không giám sát là những thuật toán chỉ xử lý các "đặc trưng" mà không có tín hiệu giám sát. Sự khác biệt giữa thuật toán có giám sát và không giám sát không được xác định một cách chính thức và cứng nhắc, vì không có một tiêu chí khách quan nào để phân biệt một giá trị là đặc trưng hay mục tiêu được cung cấp bởi một giám sát viên. Một cách không chính thức, học không giám sát thường ám chỉ các nỗ lực trích xuất thông tin từ một phân phối mà không đòi hỏi công sức chú thích từ con người. Thuật ngữ này thường liên quan đến ước lượng mật độ, học cách lấy mẫu từ một phân phối, học cách làm sạch dữ liệu từ một phân phối nào đó, tìm một đa tạp mà dữ liệu nằm gần, hoặc phân cụm dữ liệu thành các nhóm các ví dụ có liên quan.

Một tác vụ học không giám sát cổ điển là tìm "biểu diễn tốt nhất" cho dữ liệu. "Tốt nhất" có thể mang nhiều ý nghĩa khác nhau, nhưng nói chung là chúng ta tìm kiếm một biểu diễn giữ được nhiều thông tin nhất về $\boldsymbol{x}$ trong khi vẫn tuân theo một số hình thức phạt hoặc ràng buộc nhằm giữ cho biểu diễn *đơn giản hơn* hoặc dễ truy cập hơn so với $\boldsymbol{x}$ ban đầu.

Có nhiều cách khác nhau để định nghĩa một biểu diễn *đơn giản hơn*. Ba phương pháp phổ biến nhất bao gồm các biểu diễn có số chiều thấp, biểu diễn thưa, và biểu diễn độc lập. Biểu diễn có số chiều thấp cố gắng nén càng nhiều thông tin về x càng tốt vào trong một biểu diễn nhỏ hơn. Biểu diễn thưa (Barlow, 1989; Olshausen và Field, 1996; Hinton và Ghahramani, 1997) nhúng tập dữ liệu vào một biểu diễn mà trong đó hầu hết các phần tử bằng không với hầu hết các đầu vào. Sử dụng

biểu diễn thưa thường đòi hỏi tăng số chiều của biểu diễn để biểu diễn thưa không loại bỏ quá nhiều thông tin. Điều này tạo ra một cấu trúc tổng thể của biểu diễn có xu hướng phân bổ dữ liệu dọc theo các trục của không gian biểu diễn. Biểu diễn độc lập cố gắng *tách rời* các nguồn biến thiên ẩn sau phân phối dữ liệu sao cho các chiều của biểu diễn là độc lập về mặt thống kê.

Tất nhiên, ba tiêu chí này không loại trừ lẫn nhau. Các biểu diễn có số chiều thấp thường tạo ra các phần tử có các phụ thuộc ít hơn yếu hơn so với dữ liệu gốc có số chiều cao. Điều này là do một cách để giảm kích thước của biểu diễn là tìm và loại bỏ các dư thừa. Xác định và loại bỏ nhiều dư thừa hơn cho phép thuật toán giảm số chiều đạt được nhiều nén hơn mà không mất quá nhiều thông tin.

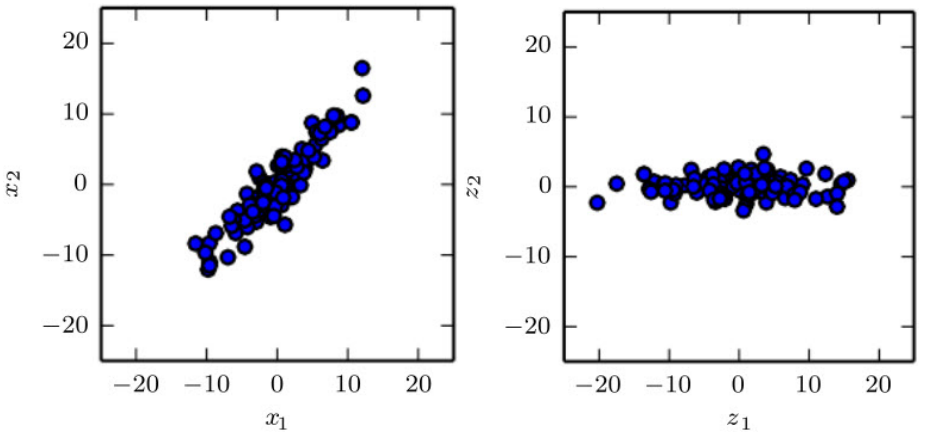
Khái niệm về biểu diễn là một trong những chủ đề trung tâm của học sâu và do đó là một trong những chủ đề trung tâm trong cuốn sách này. Trong phần này, chúng ta phát triển một số ví dụ đơn giản về các thuật toán học biểu diễn. Các thuật toán ví dụ này cùng nhau cho thấy cách hiện thực hóa cả ba tiêu chí trên. Hầu hết các chương còn lại giới thiệu thêm các thuật toán học biểu diễn phát triển các tiêu chí này theo nhiều cách khác nhau hoặc giới thiệu các tiêu chí khác.

5.8.1 Phân tích thành phần chính

Trong Mục 2.12, chúng ta đã thấy rằng thuật toán phân tích thành phần chính (PCA) cung cấp một cách để nén dữ liệu. Chúng ta cũng có thể xem PCA là một thuật toán học không giám sát giúp học biểu diễn của dữ liệu. Biểu diễn này dựa trên hai trong số các tiêu chí cho một biểu diễn đơn giản được mô tả ở trên. PCA học một biểu diễn có số chiều thấp hơn so với đầu vào ban đầu. Nó cũng học một biểu diễn mà các phần tử không có tương quan tuyến tính với nhau. Đây là một bước đầu tiên hướng đến tiêu chí học các biểu diễn mà các phần tử độc lập về mặt thống kê. Để đạt được sự độc lập hoàn toàn, một thuật toán học biểu diễn cũng phải loại bỏ các mối quan hệ phi tuyến giữa các biến.

PCA học một phép biến đổi tuyến tính trực giao của dữ liệu, chiếu một đầu vào $\boldsymbol{x}$ vào một biểu diễn $\boldsymbol{z}$ như trong Hình 5.8. Trong Mục 2.12, chúng ta đã thấy rằng ta có thể học một biểu diễn một chiều tốt nhất để tái tạo lại dữ liệu gốc (theo nghĩa sai số bình phương trung bình) và biểu diễn này thực ra tương ứng với thành phần chính đầu tiên của dữ liệu. Do đó, ta có thể sử dụng PCA như một phương pháp giảm số chiều đơn giản và hiệu quả, bảo toàn càng nhiều thông tin trong dữ liệu càng tốt (một lần nữa, được đo bằng sai số tái tạo bình phương nhỏ nhất). Trong phần tiếp theo, chúng ta sẽ nghiên cứu cách biểu diễn PCA làm mất tương quan

biểu diễn dữ liệu gốc $\boldsymbol{X}$.



Hình 5.8: PCA học một phép chiếu tuyến tính nhằm căn chỉnh hướng có phương sai lớn nhất với các trục của không gian mới. *Hình trái*: Dữ liệu ban đầu gồm các mẫu của $\boldsymbol{x}$. Trong không gian này, phương sai có thể xuất hiện theo các hướng không căn chỉnh với các trục. *Hình phải*: Dữ liệu biến đổi $\boldsymbol{z} = \boldsymbol{x}^T \boldsymbol{W}$ bây giờ biến thiên nhiều nhất dọc theo trục z_1. Hướng có phương sai lớn thứ hai hiện nằm dọc theo z_2.

Hãy xem xét ma trận thiết kế $m \times n$ chiều $\boldsymbol{X}$. Ta sẽ giả định rằng dữ liệu có trung bình bằng không, $E(\boldsymbol{x}) = \boldsymbol{0}$. Nếu không phải như vậy, dữ liệu có thể dễ dàng được làm trung bình bằng cách trừ đi trung bình của tất cả các ví dụ trong một bước tiền xử lý.

Ma trận hiệp phương sai mẫu không chệch liên kết với $\boldsymbol{X}$ được cho bởi:

$$\text{Var}(\boldsymbol{x}) = \frac{1}{m-1} \boldsymbol{X}^T \boldsymbol{X}. \tag{5.81}$$

PCA tìm một biểu diễn (thông qua biến đổi tuyến tính) $\boldsymbol{z} = \boldsymbol{x}^T \boldsymbol{W}$ sao cho $\text{Var}(\boldsymbol{z})$ là ma trận đường chéo.

Trong Mục 2.12, chúng ta đã thấy rằng các thành phần chính của ma trận thiết kế $\boldsymbol{X}$ được cho bởi các véctơ riêng của $\boldsymbol{X}^T \boldsymbol{X}$. Từ góc nhìn này,

$$\boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{W} \boldsymbol{\Lambda} \boldsymbol{W}^T. \tag{5.82}$$

Trong phần này, chúng ta khai thác một cách dẫn xuất thay thế của các thành phần chính. Các thành phần chính cũng có thể được thu được thông qua phân tích giá trị kỳ dị (SVD). Cụ thể, chúng là các véctơ kỳ dị phải của $\boldsymbol{X}$. Để thấy điều này,

đặt $\boldsymbol{W}$ là các véctơ kỳ dị phải trong phân tích $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T$. Ta sẽ khôi phục phương trình véctơ riêng ban đầu với $\boldsymbol{W}$ làm cơ sở véctơ riêng:

$$\boldsymbol{X}^T\boldsymbol{X} = \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T\right)^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T = \boldsymbol{W}\boldsymbol{\Sigma}^2\boldsymbol{W}^T. \tag{5.83}$$

SVD giúp chỉ ra rằng PCA tạo ra một Var $(\boldsymbol{z})$ có dạng đường chéo. Sử dụng SVD của $\boldsymbol{X}$, ta có thể biểu diễn phương sai của $\boldsymbol{x}$ bởi:

$$\begin{aligned}
\text{Var}(\boldsymbol{x}) &= \frac{1}{m-1}\boldsymbol{X}^T\boldsymbol{X} && (5.84)\\
&= \frac{1}{m-1}\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T\right)^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T && (5.85)\\
&= \frac{1}{m-1}\boldsymbol{W}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^T && (5.86)\\
&= \frac{1}{m-1}\boldsymbol{W}\boldsymbol{\Sigma}^2\boldsymbol{W}^T, && (5.87)
\end{aligned}$$

trong đó chúng ta sử dụng thực tế rằng $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ vì ma trận $\boldsymbol{U}$ của phân tích giá trị kỳ dị được định nghĩa là trực giao. Điều này cho thấy rằng nếu ta lấy $\boldsymbol{z} = \boldsymbol{x}^T\boldsymbol{W}$, ta có thể đảm bảo rằng hiệp phương sai của $\boldsymbol{z}$ là ma trận đường chéo như yêu cầu:

$$\begin{aligned}
\text{Var}(\boldsymbol{z}) &= \frac{1}{m-1}\boldsymbol{Z}^T\boldsymbol{Z} && (5.88)\\
&= \frac{1}{m-1}\boldsymbol{W}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} && (5.89)\\
&= \frac{1}{m-1}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{\Sigma}^2\boldsymbol{W}^T\boldsymbol{W} && (5.90)\\
&= \frac{1}{m-1}\boldsymbol{\Sigma}^2, && (5.91)
\end{aligned}$$

trong đó lần này ta lại sử dụng thực tế rằng $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$, từ định nghĩa của SVD.

Phân tích trên cho thấy rằng khi ta chiếu dữ liệu $\boldsymbol{x}$ sang $\boldsymbol{z}$ thông qua phép biến đổi tuyến tính $\boldsymbol{W}$, biểu diễn thu được sẽ có ma trận hiệp phương sai là ma trận đường chéo (và được cho bởi $\boldsymbol{\Sigma}^2$), điều này ngụ ý ngay rằng các phần tử riêng lẻ của $\boldsymbol{z}$ là không tương quan lẫn nhau.

Khả năng của PCA trong việc biến đổi dữ liệu thành một biểu diễn mà các phần tử là không tương quan lẫn nhau là một thuộc tính rất quan trọng của PCA. Đây là một ví dụ đơn giản về một biểu diễn cố gắng *tách rời các yếu tố biến thiên không biết trước* thông qua dữ liệu. Trong trường hợp của PCA, việc tách rời này mang hình thức tìm một phép xoay không gian đầu vào (được mô tả bởi $\boldsymbol{W}$) để căn chỉnh các trục chính của phương sai với cơ sở của không gian biểu diễn mới liên kết với $\boldsymbol{z}$.

Dù tương quan là một dạng phụ thuộc quan trọng giữa các phần tử của dữ liệu, chúng ta cũng quan tâm đến việc học các biểu diễn có khả năng tách rời các dạng phụ thuộc phức tạp hơn giữa các đặc trưng. Để làm được điều này, ta cần nhiều hơn những gì có thể làm với một phép biến đổi tuyến tính đơn giản.

5.8.2 Phân cụm $k-$trung bình

Một ví dụ khác về thuật toán học biểu diễn đơn giản là phân cụm $k-$trung bình. Thuật toán phân cụm $k-$trung bình chia tập huấn luyện thành k cụm khác nhau gồm các ví dụ gần nhau. Chúng ta có thể xem thuật toán này như việc cung cấp một véctơ mã hóa one−hot có k chiều, $\boldsymbol{h}$, để biểu diễn đầu vào $\boldsymbol{x}$. Nếu $\boldsymbol{x}$ thuộc cụm i, thì $h_i = 1$ và tất cả các phần tử khác của biểu diễn $\boldsymbol{h}$ đều là không.

Mã hóa one−hot do phân cụm $k-$trung bình cung cấp là một ví dụ về biểu diễn thưa, bởi vì phần lớn các phần tử của nó đều là không với mỗi đầu vào. Sau này, chúng ta sẽ phát triển các thuật toán khác để học các biểu diễn thưa linh hoạt hơn, trong đó có thể có nhiều hơn một phần tử khác không cho mỗi đầu vào $\boldsymbol{x}$. Mã hóa one−hot là một ví dụ cực đoan của biểu diễn thưa mất nhiều lợi ích của biểu diễn phân tán. Tuy nhiên, mã hóa one−hot vẫn mang lại một số lợi thế về mặt thống kê (nó truyền tải ý tưởng một cách tự nhiên rằng tất cả các ví dụ trong cùng một cụm là tương tự nhau) và lợi thế về mặt tính toán khi toàn bộ biểu diễn có thể được biểu thị bằng một số nguyên duy nhất.

Thuật toán $k-$trung bình hoạt động bằng cách khởi tạo k trung tâm cụm khác nhau $\{\boldsymbol{\mu}^{(1)}, ..., \boldsymbol{\mu}^{(k)}\}$ với các giá trị khác nhau, sau đó xen kẽ giữa hai bước khác nhau cho đến khi hội tụ. Ở một bước, mỗi ví dụ huấn luyện được gán cho cụm i, trong đó i là chỉ số của trung tâm cụm gần nhất $\boldsymbol{\mu}^{(i)}$. Ở bước kia, mỗi trung tâm cụm $\boldsymbol{\mu}^{(i)}$ được cập nhật thành giá trị trung bình của tất cả các ví dụ huấn luyện $\boldsymbol{x}^{(j)}$ được gán cho cụm i.

Một khó khăn liên quan đến phân cụm là bài toán phân cụm vốn dĩ là một bài toán chưa xác định rõ ràng, bởi vì không có một tiêu chí duy nhất nào đo lường được mức độ tương ứng giữa phân cụm dữ liệu và thực tế. Chúng ta có thể đo lường các thuộc tính của phân cụm, chẳng hạn như khoảng cách Euclid trung bình từ một tâm cụm đến các phần tử của cụm. Điều này cho phép chúng ta đánh giá khả năng tái tạo dữ liệu huấn luyện từ các gán cụm. Tuy nhiên, chúng ta không biết các gán cụm có tương ứng tốt với các thuộc tính của thế giới thực hay không. Hơn nữa, có thể có nhiều cách phân cụm khác nhau đều phù hợp với một số thuộc tính trong thực tế. Chúng ta có thể muốn tìm một phân cụm liên quan đến một đặc điểm

nhất định nhưng lại nhận được một phân cụm khác, cũng hợp lý nhưng không liên quan đến nhiệm vụ của chúng ta. Ví dụ, giả sử chúng ta chạy hai thuật toán phân cụm trên một tập dữ liệu gồm các ảnh về xe tải đỏ, xe hơi đỏ, xe tải xám và xe hơi xám. Nếu chúng ta yêu cầu mỗi thuật toán tìm hai cụm, thì một thuật toán có thể tìm ra một cụm gồm các xe hơi và một cụm gồm các xe tải, trong khi thuật toán khác có thể tìm ra một cụm gồm các xe màu đỏ và một cụm gồm các xe màu xám. Giả sử chúng ta cũng chạy một thuật toán phân cụm thứ ba, cho phép xác định số lượng cụm. Thuật toán này có thể gán các ví dụ thành bốn cụm: xe hơi đỏ, xe tải đỏ, xe hơi xám và xe tải xám. Phân cụm mới này ít nhất đã nắm bắt được thông tin về cả hai thuộc tính, nhưng nó đã mất đi thông tin về sự tương đồng. Xe hơi đỏ nằm trong một cụm khác với xe hơi xám, giống như chúng nằm trong một cụm khác với xe tải xám. Kết quả đầu ra của thuật toán phân cụm không cho chúng ta biết rằng xe hơi đỏ giống với xe hơi xám hơn là với xe tải xám. Chúng khác với cả hai loại kia, và đó là tất cả những gì ta biết.

Những vấn đề này minh họa một số lý do khiến chúng ta có thể ưu tiên biểu diễn phân tán hơn là biểu diễn one – hot. Một biểu diễn phân tán có thể bao gồm hai thuộc tính cho mỗi phương tiện — một thuộc tính đại diện cho màu sắc và một thuộc tính xác định liệu đó là xe hơi hay xe tải. Vẫn chưa hoàn toàn rõ ràng biểu diễn phân tán tối ưu là gì (làm sao thuật toán học có thể biết hai thuộc tính mà chúng ta quan tâm là màu sắc và phân loại xe hơi hay xe tải thay vì là nhà sản xuất và tuổi thọ?), nhưng việc có nhiều thuộc tính giúp giảm gánh nặng cho thuật toán trong việc đoán thuộc tính nào chúng ta cần, và cho phép chúng ta đo lường sự tương đồng giữa các đối tượng theo cách chi tiết hơn bằng cách so sánh nhiều thuộc tính thay vì chỉ kiểm tra xem một thuộc tính có khớp hay không.

5.9 Phương pháp hướng giảm ngẫu nhiên

Hầu hết các thuật toán học sâu đều được thúc đẩy bởi một thuật toán rất quan trọng: **hướng giảm ngẫu nhiên** (stochastic gradient descent, SGD). Thuật toán hướng giảm ngẫu nhiên là một phần mở rộng của thuật toán hướng giảm đã được giới thiệu trong Mục 4.3.

Một vấn đề lặp lại trong học máy là các bộ dữ liệu huấn luyện lớn cần thiết để đạt được khả năng tổng quát hóa tốt, nhưng các bộ dữ liệu lớn cũng tốn nhiều chi phí tính toán hơn.

Hàm chi phí được sử dụng bởi một thuật toán học máy thường phân tích thành

tổng của các hàm mất mát theo từng ví dụ huấn luyện. Ví dụ, đối của logarit hàm hợp lý có điều kiện của dữ liệu huấn luyện có thể được viết dưới dạng

$$J\left(\boldsymbol{\theta}\right) = E_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}}\left[L\left(\boldsymbol{x},y,\boldsymbol{\theta}\right)\right] = \frac{1}{m}\sum_{i=1}^{m} L\left(\boldsymbol{x}^{(i)},y^{(i)},\boldsymbol{\theta}\right) \tag{5.92}$$

trong đó L là hàm mất mát cho từng ví dụ $L\left(\boldsymbol{x},y,\boldsymbol{\theta}\right) = -\log p\left(y \mid \boldsymbol{x};\boldsymbol{\theta}\right)$.

Đối với các hàm chi phí cộng tính này, thuật toán hướng giảm yêu cầu tính toán

$$\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}\right) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{x}^{(i)},y^{(i)},\boldsymbol{\theta}\right). \tag{5.93}$$

Chi phí tính toán của thao tác này là $O\left(m\right)$. Khi kích thước của tập huấn luyện tăng lên đến hàng tỷ ví dụ, thời gian để thực hiện một bước gradient trở nên quá dài để thực hiện.

Ý tưởng của phương pháp hướng giảm ngẫu nhiên là gradient là một kỳ vọng. Kỳ vọng này có thể được ước lượng xấp xỉ bằng cách sử dụng một tập mẫu nhỏ. Cụ thể, ở mỗi bước của thuật toán, ta có thể lấy một *nhóm nhỏ* các ví dụ $B = \left\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m')}\right\}$ từ tập huấn luyện một cách ngẫu nhiên. Kích thước của nhóm nhỏ m' thường được chọn là một số lượng nhỏ các ví dụ, dao động từ 1 đến vài trăm. Điều quan trọng là m' thường được giữ cố định khi kích thước của tập huấn luyện m tăng lên. Chúng ta có thể huấn luyện một tập dữ liệu với hàng tỷ ví dụ bằng các cập nhật chỉ được tính toán trên một trăm ví dụ.

Ước lượng gradient được cấu thành bởi

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{m'} L\left(\boldsymbol{x}^{(i)},y^{(i)},\boldsymbol{\theta}\right) \tag{5.94}$$

bằng cách sử dụng các ví dụ từ nhóm nhỏ B. Thuật toán hướng giảm ngẫu nhiên sau đó đi theo hướng giảm đã được ước lượng:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \varepsilon\boldsymbol{g}, \tag{5.95}$$

trong đó ε là tốc độ học.

Thuật toán hướng giảm nói chung thường được coi là chậm hoặc không đáng tin cậy. Trước đây, việc áp dụng phương pháp hướng giảm cho các bài toán tối ưu không lồi bị xem là liều lĩnh hoặc thiếu nguyên tắc. Ngày nay, chúng ta biết rằng các mô hình học máy được mô tả trong Phần II hoạt động rất hiệu quả khi được huấn luyện bằng phương pháp hướng giảm. Thuật toán tối ưu không được đảm

bảo sẽ đạt tới một cực tiểu địa phương trong một khoảng thời gian hợp lý, nhưng nó thường tìm được một giá trị rất thấp của hàm chi phí đủ nhanh để ta thấy nó hữu ích.

Phương pháp hướng giảm ngẫu nhiên có nhiều ứng dụng quan trọng ngoài ngữ cảnh của học sâu. Đây là phương pháp chính để huấn luyện các mô hình tuyến tính lớn trên các tập dữ liệu cực lớn. Đối với một kích thước mô hình cố định, chi phí cho mỗi lần cập nhật của SGD không phụ thuộc vào kích thước tập huấn luyện m. Trong thực tế, ta thường sử dụng một mô hình lớn hơn khi kích thước tập huấn luyện tăng, nhưng điều này không bắt buộc. Số lần cập nhật cần thiết để đạt tới hội tụ thường tăng cùng với kích thước tập huấn luyện. Tuy nhiên, khi m tiến tới vô cùng, mô hình cuối cùng sẽ hội tụ tới sai số kiểm thử tốt nhất của nó trước khi SGD lấy mẫu hết tất cả các ví dụ trong tập huấn luyện. Việc tăng m thêm nữa sẽ không kéo dài thời gian huấn luyện cần thiết để đạt tới sai số kiểm thử tốt nhất của mô hình. Từ quan điểm này, có thể lập luận rằng chi phí tiệm cận của việc huấn luyện một mô hình bằng SGD là $O(1)$ như một hàm của m.

Trước khi học sâu ra đời, phương pháp chính để học các mô hình phi tuyến là sử dụng thủ thuật kernel kết hợp với mô hình tuyến tính. Nhiều thuật toán học kernel yêu cầu xây dựng một ma trận $m \times m$ là $G_{i,j} = k\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$. Việc xây dựng ma trận này có chi phí tính toán $O\left(m^2\right)$, điều này rõ ràng không mong muốn cho các tập dữ liệu với hàng tỷ ví dụ. Trong giới học thuật, bắt đầu từ năm 2006, học sâu ban đầu thu hút sự quan tâm vì nó có khả năng tổng quát hóa với các ví dụ mới tốt hơn các thuật toán cạnh tranh khi được huấn luyện trên các tập dữ liệu cỡ trung bình với hàng chục nghìn ví dụ. Ngay sau đó, học sâu thu hút thêm sự quan tâm từ ngành công nghiệp vì nó cung cấp một phương pháp có thể mở rộng để huấn luyện các mô hình phi tuyến trên các tập dữ liệu lớn.

Thuật toán hướng giảm ngẫu nhiên và nhiều cải tiến của nó được mô tả chi tiết hơn ở Chương 11.

5.10 Xây dựng thuật toán học máy

Hầu hết các thuật toán học sâu đều có thể được mô tả như những trường hợp cụ thể của một công thức khá đơn giản: kết hợp một đặc tả của tập dữ liệu, một hàm chi phí, một quy trình tối ưu hóa và một mô hình.

Ví dụ, thuật toán hồi quy tuyến tính kết hợp một tập dữ liệu gồm $\boldsymbol{X}$ và $\boldsymbol{Y}$, hàm

chi phí

$$J(\boldsymbol{w}, b) = -E_{\boldsymbol{x}, y \sim \hat{p}_{\text{data}}} \log p_{\text{model}}\left(y \mid \boldsymbol{x}\right), \tag{5.96}$$

đặc tả mô hình $p_{\text{model}}\left(y \mid \boldsymbol{x}\right) = N\left(y; \boldsymbol{x}^T \boldsymbol{w} + b, 1\right)$, và, trong hầu hết các trường hợp, thuật toán tối ưu hóa được xác định bằng cách giải cho điểm mà gradient của hàm chi phí bằng không sử dụng các phương trình chuẩn tắc.

Bằng cách nhận ra rằng chúng ta có thể thay thế bất kỳ thành phần nào trong số này một cách tương đối độc lập với các thành phần khác, chúng ta có thể tạo ra một loạt các thuật toán đa dạng.

Hàm chi phí thường bao gồm ít nhất một hạng tử khiến quá trình học thực hiện ước lượng thống kê. Hàm chi phí phổ biến nhất là đối của logarith hàm hợp lý, sao cho việc cực tiểu hóa hàm chi phí sẽ dẫn đến ước lượng hợp lý cực đại.

Hàm chi phí cũng có thể bao gồm các hạng tử bổ sung, chẳng hạn như các hạng tử điều chuẩn. Ví dụ, ta có thể thêm suy giảm trọng số vào hàm chi phí của hồi quy tuyến tính để thu được

$$J(\boldsymbol{w}, b) = -E_{\boldsymbol{x}, y \sim \hat{p}_{\text{data}}} \log p_{\text{model}}\left(y \mid \boldsymbol{x}\right) + \lambda \left\|\boldsymbol{w}\right\|_2^2. \tag{5.97}$$

Điều này vẫn cho phép tối ưu hóa bằng dạng đóng.

Nếu ta thay đổi mô hình thành phi tuyến, thì hầu hết các hàm chi phí không còn có thể tối ưu được bằng dạng đóng. Điều này yêu cầu ta chọn một phương pháp tính tối ưu lặp, chẳng hạn như phương pháp hướng giảm.

Công thức để xây dựng một thuật toán học bằng cách kết hợp các mô hình, chi phí và các thuật toán tối ưu hỗ trợ cả học có giám sát và không có giám sát. Ví dụ hồi quy tuyến tính minh họa cách hỗ trợ học có giám sát. Học không có giám sát có thể được hỗ trợ bằng cách định nghĩa một tập dữ liệu chỉ chứa X và cung cấp một hàm chi phí và mô hình không giám sát phù hợp. Ví dụ, ta có thể thu được véctơ PCA đầu tiên bằng cách xác định rằng hàm mất mát là

$$J(\boldsymbol{w}) = E_{\boldsymbol{x} \sim \hat{p}_{\text{data}}} \left\|\boldsymbol{x} - \boldsymbol{r}(\boldsymbol{x}; \boldsymbol{w})\right\|_2^2 \tag{5.98}$$

trong khi mô hình của ta được định nghĩa với $\boldsymbol{w}$ có chuẩn bằng một và hàm tái tạo là $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} \boldsymbol{w}$.

Trong một số trường hợp, hàm chi phí có thể là một hàm mà chúng ta không thể thực sự đánh giá được do các lý do tính toán. Trong những trường hợp này, ta vẫn có thể xấp xỉ việc cực tiểu hóa hàm này bằng cách sử dụng các phương pháp tính tối ưu lặp nếu có cách xấp xỉ các gradient của nó.

Hầu hết các thuật toán học máy đều sử dụng công thức này, mặc dù có thể không rõ ràng ngay lập tức. Nếu một thuật toán học máy trông đặc biệt độc đáo

hoặc được thiết kế thủ công, thường có thể hiểu nó như là một bộ tối ưu hóa trường hợp đặc biệt. Một số mô hình, chẳng hạn như cây quyết định hoặc $k-$trung bình, yêu cầu các bộ tối ưu hóa đặc biệt vì hàm chi phí của chúng có các vùng bằng phẳng khiến chúng không thích hợp cho việc cực tiểu hóa bằng các bộ tối ưu hóa dựa trên gradient. Nhận ra rằng hầu hết các thuật toán học máy có thể được mô tả bằng công thức này giúp chúng ta xem các thuật toán khác nhau như một phần của phân loại các phương pháp thực hiện các tác vụ liên quan với các lý do hoạt động tương tự, thay vì chỉ là một danh sách dài các thuật toán với những cơ sở lý thuyết riêng biệt.

5.11 Các thách thức thúc đẩy học sâu

Các thuật toán học máy đơn giản được mô tả trong chương này hoạt động rất tốt trên nhiều vấn đề quan trọng khác nhau. Tuy nhiên, chúng chưa thành công trong việc giải quyết các vấn đề trung tâm trong AI, chẳng hạn như nhận dạng giọng nói hoặc nhận dạng đối tượng.

Sự phát triển của học sâu được thúc đẩy một phần bởi sự thất bại của các thuật toán truyền thống trong việc tổng quát hóa tốt cho các nhiệm vụ AI như vậy.

Phần này nói về việc làm thế nào mà thách thức trong việc tổng quát hóa với các ví dụ mới trở nên khó khăn theo cấp độ hàm mũ khi làm việc với dữ liệu có chiều cao, và các cơ chế được sử dụng để đạt được khả năng tổng quát hóa trong học máy truyền thống là không đủ để học các hàm phức tạp trong không gian có chiều cao. Các không gian này cũng thường áp đặt chi phí tính toán cao. Học sâu được thiết kế để vượt qua những trở ngại này cùng nhiều thách thức khác.

5.11.1 Lời nguyền của số chiều

Nhiều vấn đề học máy trở nên cực kỳ khó khăn khi số chiều của dữ liệu cao. Hiện tượng này được gọi là **lời nguyền của số chiều**. Đặc biệt đáng lo ngại là số lượng các cấu hình khác biệt có thể có của một tập hợp các biến tăng theo cấp độ hàm mũ khi số lượng biến tăng lên.

Lời nguyền của số chiều chiều xuất hiện ở nhiều lĩnh vực trong khoa học máy tính, đặc biệt là trong học máy.

Một thách thức do lời nguyền của số chiều đặt ra là một thách thức thống kê. Như minh họa trong Hình 5.9, thách thức thống kê này nảy sinh do số lượng cấu hình khả dĩ của x lớn hơn nhiều so với số lượng ví dụ trong tập huấn luyện. Để hiểu

rõ vấn đề này, ta có thể hình dung không gian đầu vào được tổ chức thành một lưới ô vuông, giống như trong hình. Ta có thể mô tả không gian chiều thấp với số lượng ô lưới nhỏ, phần lớn trong số đó được dữ liệu chiếm giữ. Khi tổng quát hóa cho một điểm dữ liệu mới, ta thường có thể xác định phải làm gì đơn giản bằng cách xem xét các ví dụ huấn luyện nằm trong cùng một ô lưới với đầu vào mới.

Hình 5.9: Khi số lượng chiều có liên quan của dữ liệu tăng lên (từ trái sang phải), số lượng cấu hình quan tâm có thể tăng theo cấp độ hàm mũ. *Hình trái*: Trong ví dụ một chiều này, chúng ta có một biến mà ta chỉ cần phân biệt 10 vùng quan tâm. Với đủ ví dụ nằm trong mỗi vùng này (mỗi vùng tương ứng với một ô trong hình minh họa), các thuật toán học có thể dễ dàng tổng quát hóa đúng. Một cách đơn giản để tổng quát hóa là ước lượng giá trị của hàm mục tiêu trong mỗi vùng (và có thể nội suy giữa các miền lân cận). *Hình giữa*: Với 2 chiều, việc phân biệt 10 giá trị khác nhau của mỗi biến trở nên khó khăn hơn. Ta cần theo dõi đến $10 \times 10 = 100$ miền, và cần ít nhất ngần ấy ví dụ để bao phủ tất cả các miền đó. *Hình phải*: Với 3 chiều, con số này tăng lên thành $10^3 = 1000$ miền và cần ít nhất ngần ấy ví dụ. Với d chiều và v giá trị để phân biệt dọc theo mỗi trục, chúng ta có vẻ cần $O\left(v^d\right)$ miền và ví dụ. Đây là một ví dụ về lời nguyền của tính chiều cao. Hình ảnh được cung cấp bởi Nicolas Chapados.

Ví dụ, nếu ước lượng mật độ xác suất tại một điểm $\boldsymbol{x}$ nào đó, ta có thể trả về số lượng ví dụ huấn luyện nằm trong cùng một ô thể tích đơn vị với $\boldsymbol{x}$, chia cho tổng số ví dụ huấn luyện. Nếu ta muốn phân loại một ví dụ, ta có thể trả về lớp phổ biến nhất của các ví dụ huấn luyện trong cùng ô đó. Nếu đang thực hiện hồi quy, ta có thể tính trung bình các giá trị mục tiêu quan sát được trên các ví dụ trong ô đó. Nhưng đối với các ô mà ta chưa thấy ví dụ nào thì sao? Trong các không gian chiều cao, số lượng cấu hình là khổng lồ, lớn hơn nhiều so với số lượng ví dụ của chúng ta, dẫn đến phần lớn các ô lưới không có ví dụ huấn luyện nào gắn liền với chúng. Làm thế nào chúng ta có thể đưa ra điều gì đó có ý nghĩa về những cấu

hình mới này? Nhiều thuật toán học máy truyền thống chỉ đơn giản giả định rằng đầu ra tại một điểm mới nên gần giống với đầu ra tại điểm huấn luyện gần nhất.

5.11.2 Hằng địa phương và điều chuẩn tính trơn

Để tổng quát hóa tốt, các thuật toán học máy cần được hướng dẫn bởi các giả định tiên nghiệm về loại hàm mà chúng nên học. Trước đây, chúng ta đã thấy các tiên nghiệm này được tích hợp dưới dạng niềm tin rõ ràng dưới dạng phân phối xác suất trên các tham số của mô hình. Không chính thức, chúng ta cũng có thể thảo luận về các niềm tin tiên nghiệm như là tác động trực tiếp lên chính *hàm* và chỉ tác động gián tiếp lên các tham số thông qua ảnh hưởng của chúng đối với hàm. Ngoài ra, chúng ta cũng có thể thảo luận về các niềm tin tiên nghiệm được biểu đạt một cách ngầm định, thông qua việc lựa chọn các thuật toán thiên về lựa chọn một lớp hàm này hơn lớp hàm khác, mặc dù những thiên kiến này có thể không được biểu đạt (hoặc thậm chí không thể biểu đạt) dưới dạng một phân phối xác suất đại diện cho mức độ tin tưởng của chúng ta vào các hàm khác nhau.

Một trong những "tiên nghiệm" ngầm phổ biến nhất là **tiên nghiệm tính trơn** hay **tiên nghiệm hằng địa phương**. Tiên nghiệm này cho rằng hàm mà chúng ta học không nên thay đổi quá nhiều trong một miền nhỏ.

Nhiều thuật toán đơn giản dựa hoàn toàn vào tiên nghiệm này để tổng quát hóa tốt, và do đó chúng không thể mở rộng để đối phó với những thách thức thống kê trong việc giải quyết các nhiệm vụ cấp độ AI. Trong suốt cuốn sách này, chúng ta sẽ mô tả cách học sâu giới thiệu thêm các tiên nghiệm (cả tường minh và ngầm định) nhằm giảm thiểu sai số tổng quát hóa trên các nhiệm vụ phức tạp. Tại đây, chúng ta sẽ giải thích lý do tại sao tiên nghiệm tính trơn là không đủ cho các nhiệm vụ này.

Có rất nhiều cách khác nhau để thể hiện ngầm hoặc tường minh một niềm tin trước rằng hàm học được nên trơn hay hằng số cục bộ. Tất cả các phương pháp này đều được thiết kế để khuyến khích quá trình học tạo ra một hàm f^* thỏa mãn điều kiện

$$f^*(\boldsymbol{x}) \approx f^*(\boldsymbol{x} + \varepsilon) \tag{5.99}$$

với hầu hết các cấu hình $\boldsymbol{x}$ và một thay đổi nhỏ ε. Nói cách khác, nếu ta biết một đáp án tốt cho một đầu vào $\boldsymbol{x}$ (ví dụ, nếu $\boldsymbol{x}$ là một ví dụ được gán nhãn trong tập huấn luyện) thì đáp án đó có lẽ cũng đúng trong lân cận của $\boldsymbol{x}$. Nếu ta có nhiều đáp án tốt trong một vùng lân cận nào đó, ta sẽ kết hợp chúng (bằng một dạng

trung bình hoặc nội suy nào đó) để tạo ra một đáp án phù hợp với nhiều đáp án nhất có thể.

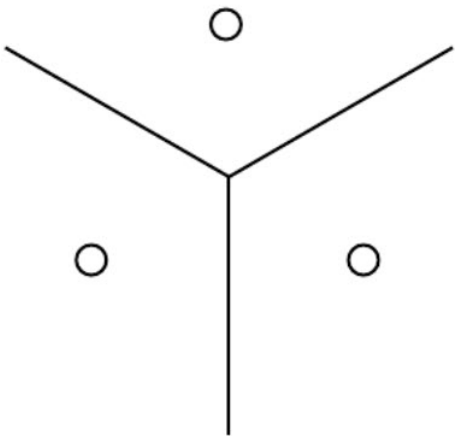
Một ví dụ cực đoan của cách tiếp cận hằng số địa phương là nhóm các thuật toán học $k-$láng giềng gần nhất. Những dự đoán này thực sự không đổi trong từng miền chứa tất cả các điểm $\boldsymbol{x}$ có cùng một tập k láng giềng gần nhất trong tập huấn luyện. Khi $k = 1$, số miền phân biệt không thể lớn hơn số ví dụ huấn luyện.

Trong khi thuật toán $k-$láng giềng gần nhất sao chép đầu ra từ các ví dụ huấn luyện lân cận, hầu hết các phương pháp kernel sẽ nội suy giữa các đầu ra của tập huấn luyện liên kết với các ví dụ huấn luyện lân cận. Một lớp kernel quan trọng là nhóm **kernel địa phương**, trong đó $k(\boldsymbol{u}, \boldsymbol{v})$ lớn khi $\boldsymbol{u} = \boldsymbol{v}$ và giảm dần khi $\boldsymbol{u}$ và $\boldsymbol{v}$ cách xa nhau. Kernel địa phương có thể được coi như một hàm tương đồng thực hiện đối chiếu mẫu bằng cách đo xem một ví dụ kiểm tra $\boldsymbol{x}$ giống với từng ví dụ huấn luyện $\boldsymbol{x}^{(i)}$ đến mức nào. Động lực hiện đại của học sâu phần lớn xuất phát từ việc nghiên cứu các giới hạn của phương pháp đối chiếu mẫu địa phương và cách các mô hình sâu có thể thành công trong những trường hợp mà đối chiếu mẫu địa phương thất bại (*The Curse of Highly Variable Functions for Local Kernel Machines*, Bengio và cộng sự, 2005, [46]).

Cây quyết định cũng chịu ảnh hưởng từ các giới hạn của học tập dựa hoàn toàn vào tính trơn, bởi vì chúng phân chia không gian đầu vào thành nhiều miền như số lá và sử dụng một tham số riêng biệt (hoặc đôi khi nhiều tham số đối với các mở rộng của cây quyết định) trong mỗi vùng. Nếu hàm mục tiêu yêu cầu một cây có ít nhất n lá để được biểu diễn chính xác, thì cần ít nhất n ví dụ huấn luyện để điều chỉnh cây. Thậm chí còn cần một bội số của n để đạt được một mức độ tin cậy thống kê nhất định trong đầu ra dự đoán.

Nói chung, để phân biệt $O(k)$ miền trong không gian đầu vào, tất cả các phương pháp này đều yêu cầu $O(k)$ ví dụ. Thông thường có $O(k)$ tham số, với $O(1)$ tham số liên kết với mỗi miền trong số $O(k)$ miền. Trường hợp của kịch bản láng giềng gần nhất, nơi mỗi ví dụ huấn luyện có thể được sử dụng để xác định tối đa một miền, được minh họa trong Hình 5.10.

Có cách nào để biểu diễn một hàm phức tạp với số lượng miền cần phân biệt lớn hơn nhiều so với số lượng ví dụ huấn luyện không? Rõ ràng, chỉ giả định tính trơn của hàm cơ bản sẽ không cho phép một mô hình học làm điều đó. Chẳng hạn, hãy tưởng tượng hàm mục tiêu là một loại bàn cờ. Bàn cờ có nhiều biến thể nhưng có một cấu trúc đơn giản trong đó. Hãy tưởng tượng điều gì xảy ra khi số lượng ví dụ huấn luyện nhỏ hơn đáng kể so với số ô đen và trắng trên bàn cờ. Dựa trên chỉ sự tổng quát hóa địa phương và giả định tiên nghiệm trơn hay tính bất biến địa



Hình 5.10: Minh họa cách thuật toán láng giềng gần nhất phân chia không gian đầu vào thành các miền. Một ví dụ (được biểu diễn bằng một hình tròn trong hình minh họa) trong mỗi miền xác định ranh giới của miền đó (được biểu diễn bằng các đường thẳng). Giá trị y liên kết với mỗi ví dụ xác định đầu ra cho tất cả các điểm nằm trong miền tương ứng. Các miền được xác định bằng việc khớp láng giềng gần nhất tạo thành một mẫu hình học gọi là sơ đồ Voronoi. Số lượng vùng liền kề này không thể tăng nhanh hơn số lượng ví dụ huấn luyện. Mặc dù hình này minh họa cụ thể hành vi của thuật toán láng giềng gần nhất, các thuật toán học máy khác chỉ dựa vào giả định trơn địa phương để tổng quát hóa cũng thể hiện hành vi tương tự: mỗi ví dụ huấn luyện chỉ cung cấp thông tin cho mô hình học về cách tổng quát hóa trong một vùng lân cận ngay xung quanh ví dụ đó.

phương, ta chỉ có thể đoán đúng màu của một điểm mới nếu nó nằm trong cùng ô bàn cờ với một ví dụ huấn luyện. Không có gì đảm bảo rằng mô hình học có thể mở rộng mẫu bàn cờ một cách chính xác đến các điểm nằm trong các ô không chứa ví dụ huấn luyện. Chỉ mỗi với giả định này, thông tin duy nhất mà một ví dụ cho chúng ta biết là màu của ô đó, và cách duy nhất để có được màu của toàn bộ bàn cờ là phủ từng ô của nó bằng ít nhất một ví dụ.

Giả định về tính trơn và các thuật toán học phi tham số liên quan hoạt động rất tốt miễn là có đủ ví dụ để thuật toán học có thể quan sát các điểm cao trên hầu hết các đỉnh và các điểm thấp trên hầu hết các thung lũng của hàm thực sự cần học. Điều này thường đúng khi hàm cần học đủ trơn và biến thiên trên số chiều đủ ít. Trong không gian nhiều chiều, ngay cả một hàm rất trơn cũng có thể thay đổi một cách trơn nhưng khác nhau theo từng chiều. Nếu hàm còn có hành vi khác nhau

trong các miền khác nhau, thì nó có thể trở nên cực kỳ phức tạp để mô tả bằng một tập hợp các ví dụ huấn luyện. Nếu hàm này phức tạp (ta muốn phân biệt một số lượng lớn các miền so với số lượng ví dụ), liệu có hy vọng nào để tổng quát hóa tốt không?

Câu trả lời cho cả hai câu hỏi này — liệu có thể biểu diễn một hàm phức tạp một cách hiệu quả và liệu hàm ước lượng có thể tổng quát hóa tốt cho các đầu vào mới hay không — là có. Ý tưởng chính là một số lượng miền rất lớn, ví dụ $O\left(2^k\right)$, có thể được xác định chỉ với $O(k)$ ví dụ, miễn là ta đưa vào một số phụ thuộc giữa các miền thông qua các giả định bổ sung về phân phối dữ liệu sinh cơ bản. Bằng cách này, ta có thể thực sự tổng quát hóa không theo hướng địa phương (*Non-Local Manifold Tangent Learning*, Bengio và Monperrus, 2004, [47]; *Non-Local Manifold Parzen Windows*, Bengio và cộng sự, 2005, [48]). Nhiều thuật toán học sâu khác nhau cung cấp các giả định ngầm hoặc tường minh hợp lý cho nhiều tác vụ AI khác nhau để nắm bắt những lợi thế này.

Các phương pháp khác trong học máy thường đưa ra các giả định mạnh hơn, cụ thể cho từng tác vụ. Ví dụ, ta có thể dễ dàng giải quyết bài toán bàn cờ bằng cách đưa vào giả định rằng hàm mục tiêu là tuần hoàn. Thông thường, ta không đưa những giả định mạnh mẽ và cụ thể về tác vụ như vậy vào mạng nơron để chúng có thể tổng quát hóa cho nhiều loại cấu trúc đa dạng hơn. Các tác vụ AI có cấu trúc phức tạp quá mức, không thể giới hạn vào những đặc tính đơn giản và được chỉ định thủ công như tính tuần hoàn, vì vậy ta mong muốn các thuật toán học có thể thể hiện các giả định mục đích tổng quát hơn. Ý tưởng cốt lõi trong học sâu là chúng ta giả định rằng dữ liệu được tạo ra từ *sự kết hợp của các yếu tố* hoặc đặc trưng, có thể ở nhiều cấp độ trong một hệ thống phân cấp. Nhiều giả định tương tự có thể cải thiện thêm các thuật toán học sâu. Những giả định dường như nhẹ nhàng này cho phép đạt được mức tăng theo hàm mũ trong mối quan hệ giữa số lượng ví dụ và số lượng miền có thể phân biệt. Những lợi thế theo hàm mũ này được mô tả chi tiết hơn trong các Mục 9.4.1, 14.4 và 14.5. Lợi ích theo hàm mũ nhờ vào việc sử dụng các biểu diễn sâu và phân tán đã đối trọng với các thách thức theo hàm mũ do lời nguyền về số chiều gây ra.

5.11.3 Học đa tạp

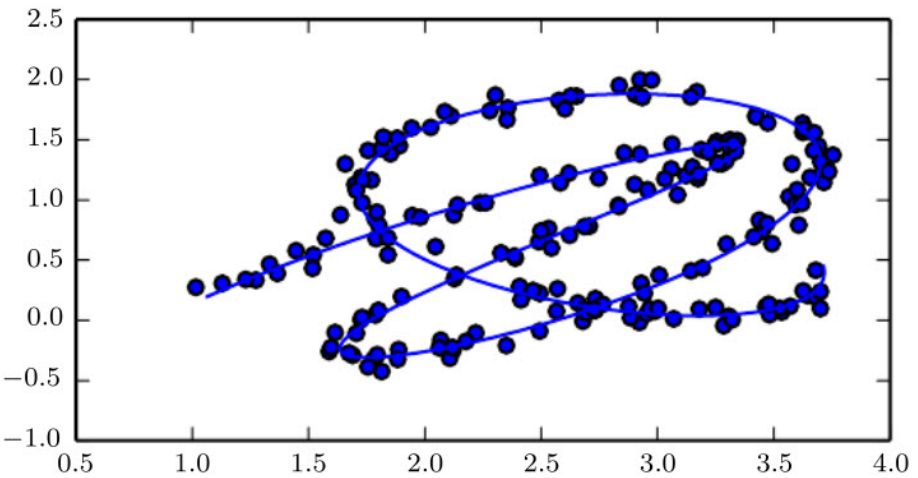
Một khái niệm quan trọng nằm ở nền tảng của nhiều ý tưởng trong học máy là đa tạp.

Đa tạp là một miền liên thông. Về mặt toán học, đó là một tập hợp các điểm,

mỗi điểm liên kết với một miền lân cận xung quanh nó. Từ bất kỳ điểm nào, đa tạp xuất hiện dưới dạng một không gian Euclid địa phương. Trong đời sống hàng ngày, chúng ta cảm nhận bề mặt Trái Đất như một mặt phẳng hai chiều, nhưng thực chất, đó là một đa tạp hình cầu trong không gian ba chiều.

Định nghĩa về một vùng lân cận xung quanh mỗi điểm ngụ ý sự tồn tại của các phép biến đổi có thể được áp dụng để di chuyển trên đa tạp từ một vị trí này đến một vị trí lân cận khác. Trong ví dụ về bề mặt Trái Đất như một đa tạp, ta có thể đi về hướng bắc, nam, đông hoặc tây.

Mặc dù thuật ngữ "đa tạp" có ý nghĩa toán học chính thức, trong học máy, nó thường được dùng với ý nghĩa rộng hơn để chỉ một tập hợp các điểm liên thông có thể được xấp xỉ tốt khi chỉ xét đến một số ít bậc tự do, hoặc chiều, được nhúng trong một không gian có số chiều lớn hơn. Mỗi chiều tương ứng với một hướng biến thiên địa phương. Xem Hình 5.11 để thấy ví dụ về dữ liệu huấn luyện nằm gần một đa tạp một chiều được nhúng trong không gian hai chiều. Trong bối cảnh học máy, chúng ta cho phép số chiều của đa tạp thay đổi từ điểm này sang điểm khác. Điều này thường xảy ra khi một đa tạp tự cắt nhau. Ví dụ, một hình số tám là một đa tạp có một chiều ở hầu hết các vị trí, nhưng có hai chiều tại điểm giao nhau ở trung tâm.



Hình 5.11: Dữ liệu được lấy mẫu từ một phân phối trong không gian hai chiều nhưng thực chất tập trung gần một đa tạp một chiều, giống như một sợi dây xoắn. Đường liền nét biểu thị đa tạp cơ bản mà mô hình học cần suy luận.

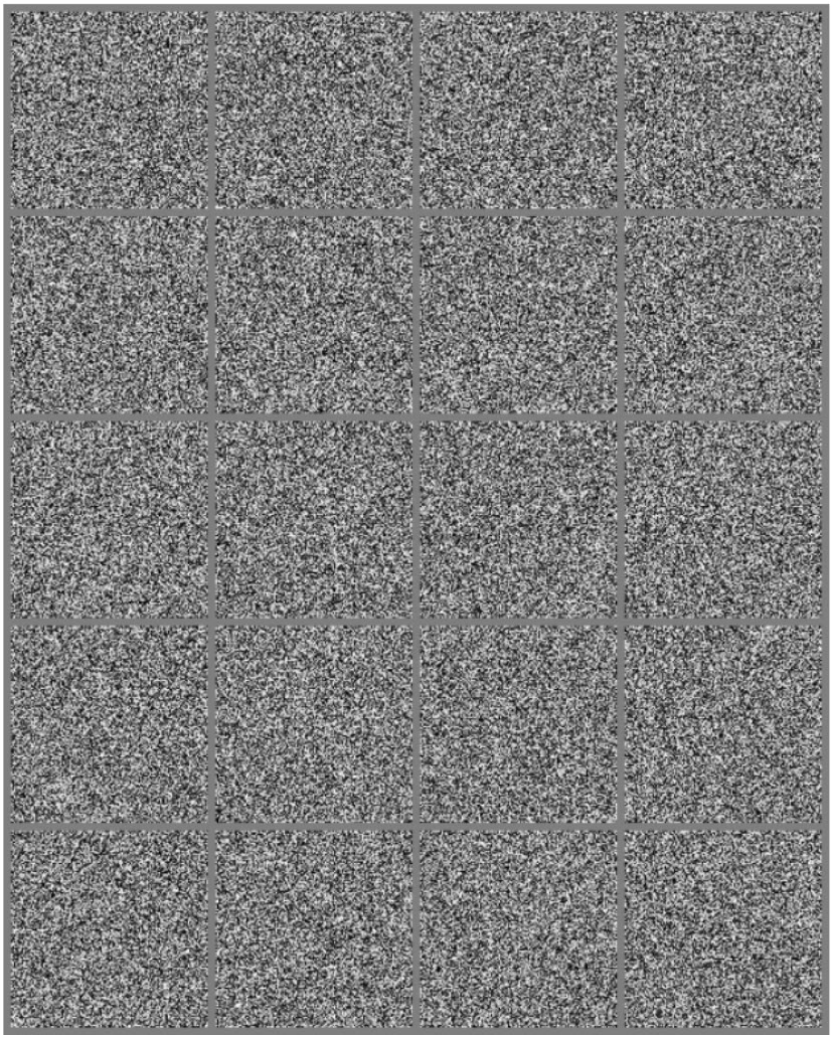
Nhiều bài toán học máy có vẻ vô vọng nếu chúng ta mong đợi thuật toán học máy học các hàm có biến đổi thú vị trên toàn bộ $\mathbb{R}^n$. Các thuật toán **học đa tạp** vượt qua trở ngại này bằng cách giả định rằng phần lớn $\mathbb{R}^n$ bao gồm các đầu vào

không hợp lệ, và các đầu vào thú vị chỉ xảy ra dọc theo một tập hợp các đa tạp chứa một tập hợp nhỏ các điểm, với các biến đổi thú vị trong đầu ra của hàm đã học chỉ xảy ra dọc theo các hướng nằm trên đa tạp, hoặc các biến đổi thú vị chỉ xảy ra khi chúng ta chuyển từ đa tạp này sang đa tạp khác. Học đa tạp được giới thiệu trong trường hợp dữ liệu liên tục và trong bối cảnh học không giám sát, mặc dù ý tưởng tập trung xác suất này có thể được khái quát hóa cho cả dữ liệu rời rạc và bối cảnh học có giám sát: giả định chính vẫn là trọng số xác suất tập trung cao.

Giả định rằng dữ liệu nằm dọc theo một đa tạp có số chiều thấp có thể không phải lúc nào cũng đúng hoặc hữu ích. Chúng tôi lập luận rằng trong bối cảnh các bài toán AI, chẳng hạn như các bài toán liên quan đến xử lý hình ảnh, âm thanh hoặc văn bản, giả định đa tạp ít nhất là xấp xỉ đúng. Bằng chứng ủng hộ cho giả định này bao gồm hai loại quan sát.

Quan sát đầu tiên ủng hộ **giả thuyết đa tạp** là phân phối xác suất trên hình ảnh, chuỗi văn bản và âm thanh xuất hiện trong đời sống thực tế có sự tập trung rất cao. Nhiễu ngẫu nhiên hầu như không bao giờ giống với các đầu vào có cấu trúc từ các lĩnh vực này. Hình 5.12 minh họa rằng các điểm được lấy mẫu một cách đồng đều trông giống như các mẫu nhiễu tĩnh xuất hiện trên các tivi analog khi không có tín hiệu. Tương tự, nếu bạn tạo một tài liệu bằng cách chọn ngẫu nhiên các chữ cái với xác suất đều, xác suất để bạn có được một văn bản có ý nghĩa bằng tiếng Anh là bao nhiêu? Gần như bằng không, bởi vì hầu hết các chuỗi dài của chữ cái không tương ứng với một chuỗi ngôn ngữ tự nhiên: phân phối của các chuỗi ngôn ngữ tự nhiên chiếm một vùng rất nhỏ trong không gian tổng thể của các chuỗi chữ cái.

Tất nhiên, các phân phối xác suất tập trung không đủ để chứng minh rằng dữ liệu nằm trên một số lượng nhỏ các đa tạp. Chúng ta cũng phải chứng minh rằng các ví dụ mà ta gặp phải là liên thông với nhau qua các ví dụ khác, với mỗi ví dụ được bao quanh bởi các ví dụ khác có độ tương đồng cao và có thể đến được bằng cách áp dụng các phép biến đổi để di chuyển dọc theo đa tạp. Lập luận thứ hai ủng hộ giả thuyết đa tạp là chúng ta cũng có thể tưởng tượng ra những khu vực lân cận và các phép biến đổi này, ít nhất là một cách không chính thức. Trong trường hợp của hình ảnh, chúng ta có thể dễ dàng nghĩ đến nhiều phép biến đổi có thể giúp chúng ta vẽ ra một đa tạp trong không gian hình ảnh: chúng ta có thể làm tối hoặc làm sáng dần, di chuyển hoặc xoay dần các đối tượng trong hình ảnh, thay đổi dần màu sắc trên bề mặt của các đối tượng, v.v. Có khả năng là có nhiều đa tạp tham gia vào hầu hết các ứng dụng. Chẳng hạn, đa tạp của các hình ảnh về khuôn mặt con người có thể không liên thông với đa tạp của các hình ảnh về khuôn mặt mèo.



Hình 5.12: Lấy mẫu hình ảnh một cách ngẫu nhiên (bằng cách chọn ngẫu nhiên từng điểm ảnh theo phân phối đều) tạo ra những hình ảnh nhiễu. Mặc dù có một xác suất khác không để tạo ra một hình ảnh về khuôn mặt hoặc bất kỳ đối tượng nào thường gặp trong các ứng dụng AI, trên thực tế chúng ta hầu như không bao giờ quan sát thấy điều này xảy ra. Điều này cho thấy rằng các hình ảnh xuất hiện trong các ứng dụng AI chiếm một tỷ lệ không đáng kể trong toàn bộ không gian hình ảnh.

Những thí nghiệm tư duy ủng hộ giả thuyết đa tạp cung cấp một số lý do trực quan để ủng hộ nó. Các thí nghiệm nghiêm ngặt hơn (*Algorithms for Manifold Learning*, Cayton, 2005, [49]; *Sample Complexity of Testing the Manifold Hypothesis*, Narayanan và Mitter, 2010, [50]; *Nonlinear Component Analysis as a Kernel Eigenvalue*

Problem, Schölkopf và cộng sự, 1998, [51]; *Nonlinear Dimensionality Reduction by Locally Linear Embedding*, Roweis và Saul, 2000, [52]; *A Global Geometric Framework for Nonlinear Dimensionality Reduction*, Tenenbaum và cộng sự, 2000, [53]; *Charting a Manifold*, Brand, 2002, [54]; *Laplacian Eigenmaps for Dimensionality Reduction and Data Representation*, Belkin và Niyogi, 2003, [55]; *Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data*, Donoho và Grimes, 2003, [56]; *Unsupervised Learning of Image Manifolds by Semidefinite Programming*, Weinberger và Saul, 2004, [57]) rõ ràng ủng hộ giả thuyết này cho một lớp dữ liệu lớn trong các bài toán AI.

Khi dữ liệu nằm trên một đa tạp có số chiều thấp, sẽ tự nhiên hơn khi các thuật toán học máy biểu diễn dữ liệu theo tọa độ trên đa tạp, thay vì theo tọa độ trong $\mathbb{R}^n$. Trong cuộc sống hàng ngày, chúng ta có thể coi các con đường là các đa tạp một chiều nhúng trong không gian ba chiều. Chúng ta chỉ đường đến các địa chỉ cụ thể theo số nhà dọc theo các con đường một chiều này, chứ không phải theo tọa độ trong không gian ba chiều. Việc trích xuất các tọa độ trên đa tạp này rất thách thức, nhưng hứa hẹn cải thiện nhiều thuật toán học máy. Nguyên tắc tổng quát này được áp dụng trong nhiều bối cảnh. Hình 5.13 minh họa cấu trúc đa tạp của một tập dữ liệu gồm các khuôn mặt. Đến cuối cuốn sách này, chúng ta sẽ phát triển các phương pháp cần thiết để học được cấu trúc đa tạp như vậy. Trong Hình 17.1, chúng ta sẽ thấy cách một thuật toán học máy có thể thực hiện thành công mục tiêu này.

Hình 5.13: Ví dụ huấn luyện từ bộ dữ liệu khuôn mặt đa góc nhìn của QMUL – Queen Mary University of London (*Dynamic Vision: From Images to Face Recognition*, Gong và cộng sự, 2000, [80]), trong đó các đối tượng được yêu cầu di chuyển sao cho phủ kín đa tạp hai chiều tương ứng với hai góc xoay. Chúng ta mong muốn các thuật toán học có thể khám phá và phân tách được các tọa độ đa tạp như vậy. Hình 17.1 minh họa một thành tựu như vậy.

Phần I kết thúc tại đây, đã cung cấp các khái niệm cơ bản về toán học và học máy sẽ được sử dụng trong các phần còn lại của cuốn sách. Bạn đã sẵn sàng bắt đầu nghiên cứu về học sâu.