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 5.5. 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 5.5, 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, chúng 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 chúng 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 $p_{\text{model}} : \mathbb{R}^n \to \mathbb{R}$, 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 chúng ta nắm bắt phân phối đó một cách tường minh. Về nguyên tắc, chúng ta có thể thực hiện các phép tính trên phân phối đó để giải quyết các tác vụ khác. Ví dụ, nếu chúng ta đã thực hiện ước lượng mật độ để thu được phân phối xác suất $p(\boldsymbol{x})$, chúng 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 chúng 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 chúng 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, chúng 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, chúng 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

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

Phép phân tích này cho phép ta giải quyết bài toán học không giám sát, tức là mô hình hóa $p\left(\boldsymbol{X}\right)$, 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 $p\left(\boldsymbol{X}, y\right)$ và suy luận

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

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 chúng 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 10.1 và Chương 11 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: $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)}\}$. 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 chúng ta muốn sử dụng một thuật toán học để nhận diện đối tượng từ các bức ảnh, chúng 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, chúng 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 5.4.1) 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 $\boldsymbol{0}$:

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

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

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

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

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

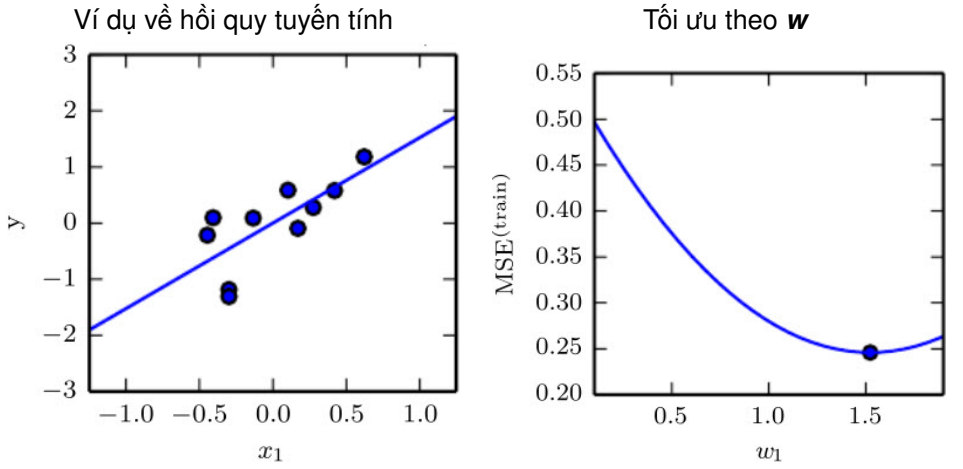
$$\Rightarrow 2\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{X}^{(\text{train})\,T}\boldsymbol{Y}^{(\text{train})} = \boldsymbol{0} \tag{5.11}$$

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

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.

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

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



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.

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, chúng 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, chúng 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 chúng ta đã mô tả đơn giản chỉ là một bài toán tối ưu. Điều phân biệt học máy với tối ưu là chúng 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 chúng 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, chúng 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 chúng 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 chúng ta có thể làm. Nếu chúng ta được phép đưa ra một số giả định về cách thu thập tập huấn luyện và tập kiểm tra, thì chúng 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 chúng 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 chúng 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 chúng 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 chúng 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 chúng ta mô hình hồi quy tuyến tính mà chúng 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:

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

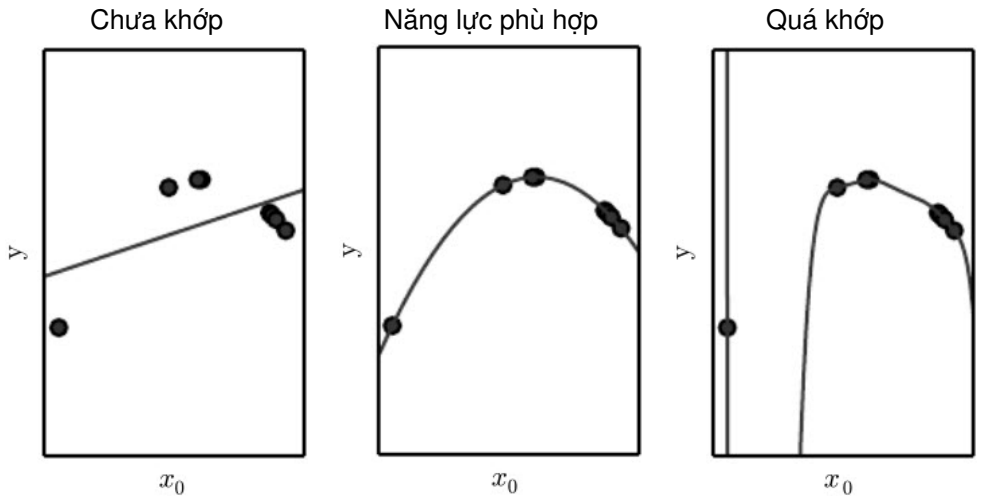
Mặc dù mô hình này triển khai một hàm bậc hai của *đầu vào*, đầu ra vẫn là hàm tuyến tính của *các tham số*, vì vậy ta vẫn có thể sử dụng phương trình chuẩn tắc để huấn luyện mô hình trong dạng đóng. Ta có thể tiếp tục thêm các lũy thừa cao hơn của x làm các đặc trưng bổ sung, chẳng hạn để có một đa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

Cho đến nay, chúng 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,



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.

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.

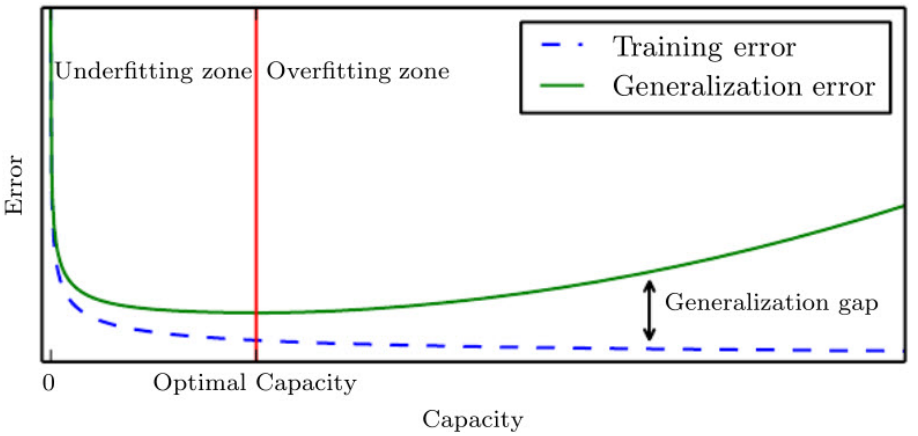
Các ý tưởng hiện đại của chúng ta về việc cải thiện khả năng tổng quát hóa của các mô hình học máy là sự tinh chỉnh của những suy nghĩ có từ thời các triết gia, ít nhất là từ Ptolemy. Nhiều học giả từ thời kỳ đầu đã nhắc đến một nguyên tắc tiết kiệm mà ngày nay được biết đến rộng rãi nhất dưới tên gọi **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), chúng 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 chúng 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, chúng 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, chúng 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.

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, [56]) 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 chúng 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 chúng ta quan tâm.

5.2.2 Điều chuẩn

Định lý không có bữa ăn miễn phí hàm ý rằng chúng 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 chúng 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 chúng ta cố gắng sử dụng nó để dự đoán hàm $\sin x$ từ x. Do đó, chúng 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 chúng ta đối với các trọng số nhỏ hơn. Khi $\lambda = 0$, chúng 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.

Tổng quát hơn, chúng ta có thể điều chuẩn một mô hình học hàm $f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)$ bằng

cách thêm một tham số phạt, gọi là **bộ điều chuẩn**, vào hàm chi phí. Trong trường hợp suy giảm trọng số, bộ điều chuẩn là $\Omega(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{w}$. Trong chương 7, chúng 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, chúng 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 chúng ta cực tiểu hóa. Có rất nhiều cách khác để thể hiện ưu tiên cho các nghiệm khác nhau, cả ngầm hiểu và rõ ràng. Tập hợp các phương pháp này được gọi chung là **điều chuẩn**. *Điều chuẩn là bất kỳ sự điều chỉnh nào mà chúng 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.

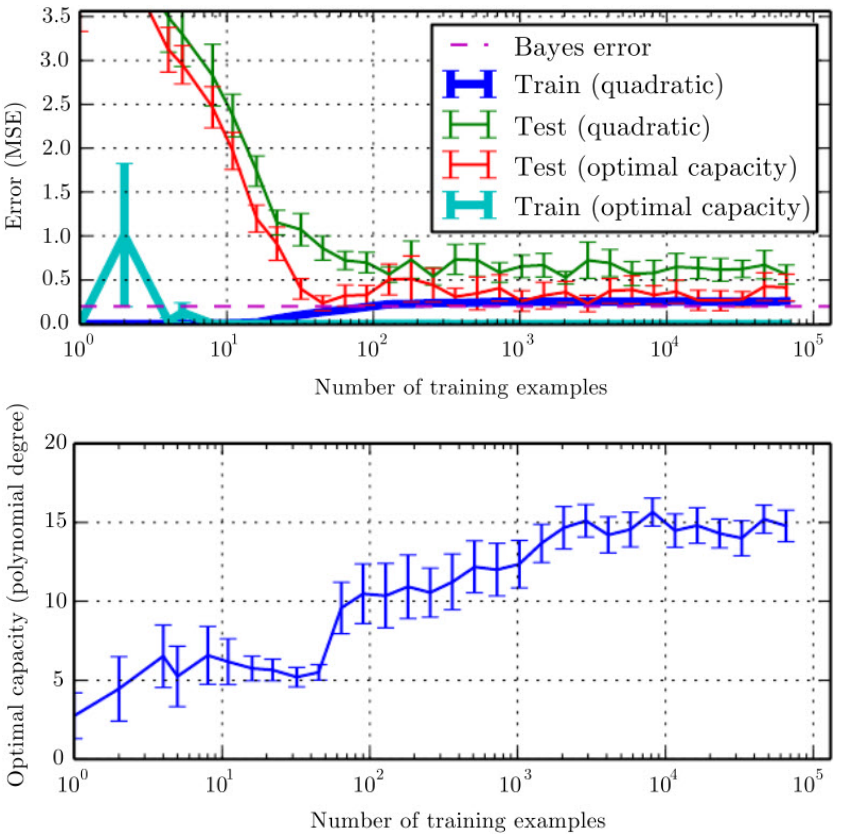
Định lý không có bữa ăn miễn phí đã chỉ ra rằng không có thuật toán học máy nào là tốt nhất, và đặc biệt là không có dạng điều chuẩn nào là tốt nhất. Thay vào đó, chúng ta phải chọn một dạng điều chuẩn phù hợp với tác vụ cụ thể mà chúng ta muốn giải quyết. Triết lý của học sâu nói chung và của cuốn sách 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 hợp kiểm định

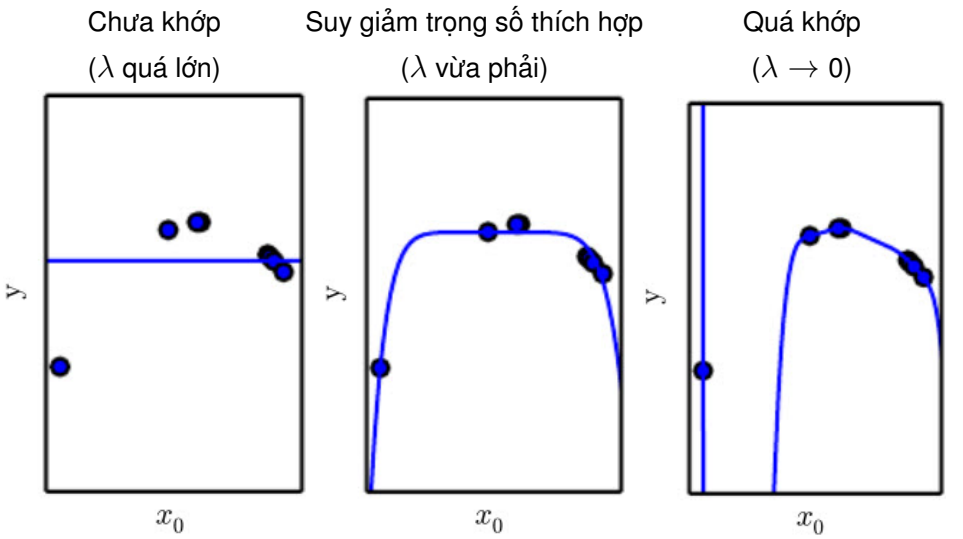
5.4 Ước lượng hợp lý cực đại

5.4.1 Logit của xác suất hợp lý có điều kiện và sai số bình phương trung bình

5.5 Các thách thức thúc đẩy học sâu



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%. *Ảnh 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. *Ảnh 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.



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, chúng 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 chúng ta đã thấy ở Hình 5.2.