

Chương 1

Toàn cảnh Học Máy

Phần lớn mọi người khi nghe tới thuật ngữ “Học Máy” sẽ liên tưởng ngay đến những con rô bốt phục vụ hay người máy chết chóc trong bộ phim Kẻ Hủy Diệt. Tuy nhiên, Học Máy giờ đây không chỉ là giấc mơ viễn tưởng nữa, mà chúng đã có mặt ở đây rồi. Trên thực tế, Học Máy đã được sử dụng từ hàng thập kỷ qua với các ứng dụng chuyên biệt, chẳng hạn như nhận dạng ký tự quang học. Thế nhưng ứng dụng đầu tiên đưa Học Máy đến với đại chúng, thứ đã cải thiện chất lượng cuộc sống của hàng trăm triệu người và trở nên cực kỳ phổ biến vào những năm 1990 là *bộ lọc thư rác*. Ứng dụng này không cao siêu như mạng Skynet trong Kẻ Hủy Diệt, nhưng lại đủ tiêu chuẩn để được xem là một kỹ thuật Học Máy. Và thực tế, bộ lọc hoạt động tốt đến nỗi hiếm khi người dùng phải tự gán nhãn thư rác nữa. Theo sau đó, hàng trăm ứng dụng Học Máy được âm thầm đưa vào hàng trăm sản phẩm và tính năng mà chúng ta đang dùng mỗi ngày, từ các hệ thống đề xuất tới tính năng tìm kiếm bằng giọng nói.

Vậy Học Máy bắt đầu từ đâu và sẽ đi đến đâu? Việc một chiếc máy *học* được một điều gì đó thật sự là thế nào? Nếu ta tải về trang Wikipedia thì máy tính có thật sự học được cái gì không? Nó sẽ tự nhiên thông minh hơn chăng? Trong chương này, chúng ta sẽ bắt đầu giải thích rõ định nghĩa và lý do mà ta có thể muốn sử dụng Học Máy.

Trước khi bắt đầu khám phá thế giới Học Máy, hãy cùng nhìn tổng quan để nắm rõ các “vùng” chính và các “địa điểm” nổi bật trong Học Máy: học có và không giám sát, học trực tuyến và học theo batch, học dựa trên mẫu và học dựa trên mô hình. Sau đó, chúng ta sẽ xem xét các bước tiến hành một dự án Học Máy, thảo luận về các thách thức sẽ gặp phải và giải thích cách đánh giá cũng như tinh chỉnh một hệ thống Học Máy.

Chương này giới thiệu rất nhiều lý thuyết căn bản (và thuật ngữ chuyên ngành) mà hầu hết các nhà khoa học dữ liệu đều phải thuộc lòng. Chương này sẽ mang tính khái quát cao (đây là chương duy nhất không có nhiều đoạn mã) và khá đơn giản. Tuy nhiên, bạn nên nắm rõ ngọn ngành trước khi tiếp tục tìm hiểu phần còn lại của cuốn sách. Giờ thì hãy chuẩn bị một ly cà phê và bắt đầu thôi!

Mẹo

Nếu đã quen thuộc với các khái niệm cơ bản của Học Máy, bạn có thể bắt đầu đọc từ [Chương 2](#). Nếu bạn không chắc chắn lắm về điều này, hãy thử trả lời các câu hỏi ở cuối chương trước khi đọc tiếp.

Học Máy là gì?

Học Máy là một môn khoa học (và cả nghệ thuật) về cách lập trình máy tính để chúng có thể *học từ dữ liệu*.

Dưới đây là định nghĩa tổng quát hơn:

[Học Máy là] lĩnh vực nghiên cứu nhằm giúp máy tính có khả năng học mà không cần lập trình một cách tưởng minh.

— Arthur Samuel, 1959

Và định nghĩa mang tính kỹ thuật hơn:

Một chương trình máy tính được cho là học từ kinh nghiệm E với tác vụ T và phép đo chất lượng P nào đó, nếu chất lượng của tác vụ T , được đo bởi P , cải thiện theo kinh nghiệm E .

— Tom Mitchell, 1997

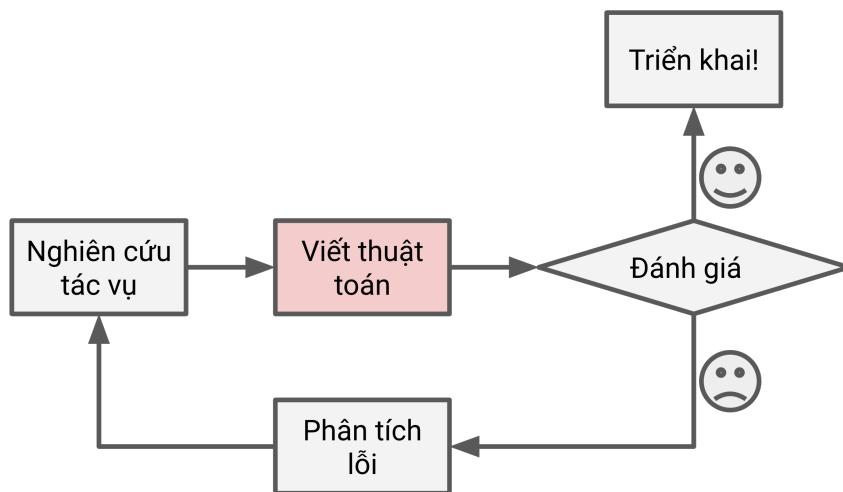
Bộ lọc thư rác chính là một chương trình Học Máy có khả năng học để phân loại đâu là thư rác từ các mẫu cho trước (thư được đánh dấu là rác bởi người dùng) và các mẫu thư thường (không phải thư rác). Tập hợp các mẫu mà hệ thống dùng để học được gọi là *tập huấn luyện*. Mỗi mẫu dữ liệu huấn luyện được gọi là *mẫu huấn luyện* (hay *mẫu*). Trong ví dụ này, tác vụ T là việc gán nhãn thư rác cho thư điện tử mới, kinh nghiệm E là *dữ liệu huấn luyện*, và ta cần định nghĩa thêm phép đo chất lượng P . Một lựa chọn khả thi cho P là tỷ lệ phân loại thư đúng, và phép đo chất lượng cụ thể này được gọi là *độ chính xác*. Phép đo này thường được dùng trong các bài toán phân loại.

Nếu chỉ tải xuống một bản sao của trang Wikipedia, máy tính của bạn sẽ có thêm rất nhiều dữ liệu nhưng nó sẽ không tự nhiên làm việc tốt hơn. Do đó việc tải xuống một bản sao của trang Wikipedia không phải là Học Máy.

Tại sao lại dùng Học Máy?

Hãy xem xét cách viết một bộ lọc thư rác bằng kỹ thuật lập trình truyền thống ([Hình 1.1](#)):

- Đầu tiên ta cần kiểm tra xem thư rác thường trông như thế nào. Ta có thể phát hiện một số từ hoặc cụm từ (như “4U,” “credit card,” “free,” và “amazing”) hay xuất hiện trong tiêu đề thư. Ta cũng có thể thấy một vài khuôn mẫu khác ở tên người gửi, nội dung thư và ở các phần khác của thư.
- Nếu ta viết thuật toán nhận diện cho từng khuôn mẫu trên, chương trình sẽ đánh dấu một thư điện tử là thư rác nếu một vài khuôn mẫu đó được tìm thấy.
- Tiếp đến, ta kiểm thử chương trình và lặp lại hai bước trên cho đến khi đạt mức chất lượng để triển khai.



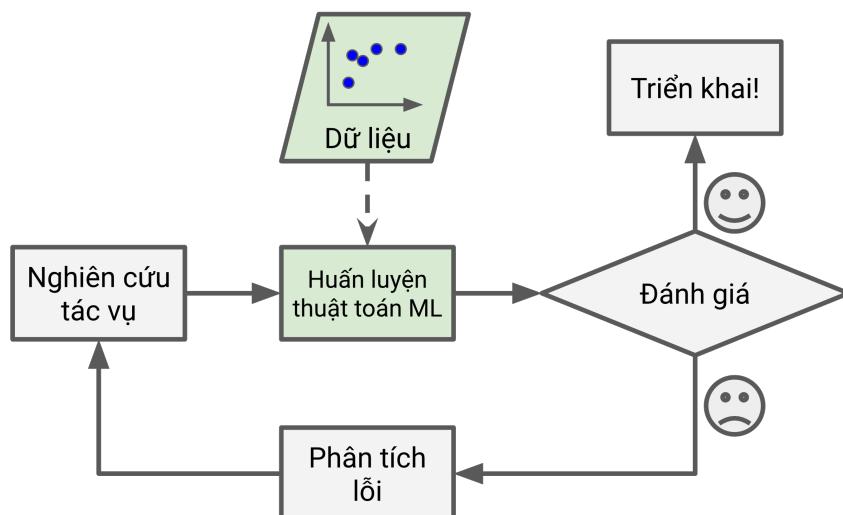
Hình 1.1. Tiếp cận theo hướng truyền thống

Vì đây là một bài toán khó, khả năng cao chương trình này sẽ trở thành một danh sách dài các quy luật phức tạp và khó bảo trì.

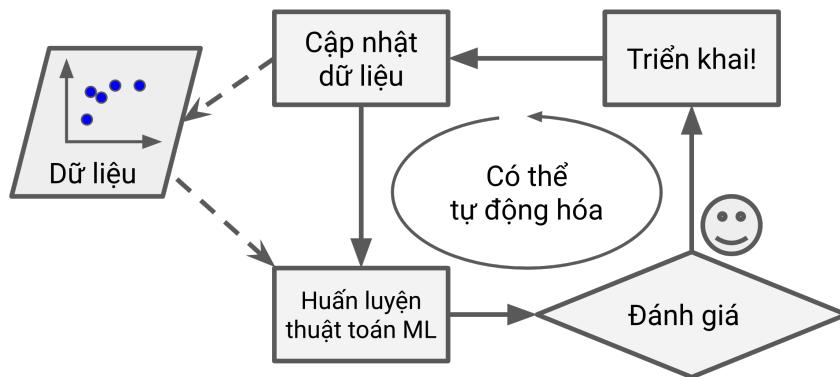
Trái lại, một bộ lọc thư rác dựa trên các kỹ thuật Học Máy sẽ tự động học được những từ và cụm từ nào là dấu hiệu của thư rác bằng cách nhận diện các khuôn mẫu có tần suất cao bất thường trong các mẫu thư rác so với các mẫu thư bình thường (Hình 1.2). Chương trình này ngắn hơn hẳn, dễ bảo trì hơn, và rất có thể sẽ chính xác hơn.

Nếu những kẻ gửi thư rác nhận ra rằng các thư điện tử chứa cụm “4U” bị chặn thì sao? Có thể chúng sẽ bắt đầu thay thế “4U” bằng “For U”. Một bộ lọc thư rác được lập trình theo hướng truyền thống sẽ cần được cập nhật để đánh dấu những thư điện tử chứa cụm “For U”. Nếu những kẻ này vẫn cố gắng lách qua bộ lọc thư rác, ta phải luôn phải cập nhật thêm các quy luật mới.

Trái lại, một bộ lọc thư rác dựa trên các kỹ thuật Học Máy sẽ tự động nhận thấy rằng cụm từ “For U” đã bắt đầu xuất hiện nhiều bất thường trong các bức thư rác được đánh dấu bởi người dùng, và nó sẽ bắt đầu đánh dấu các thư này mà không cần sự can thiệp bên ngoài (Hình 1.3).



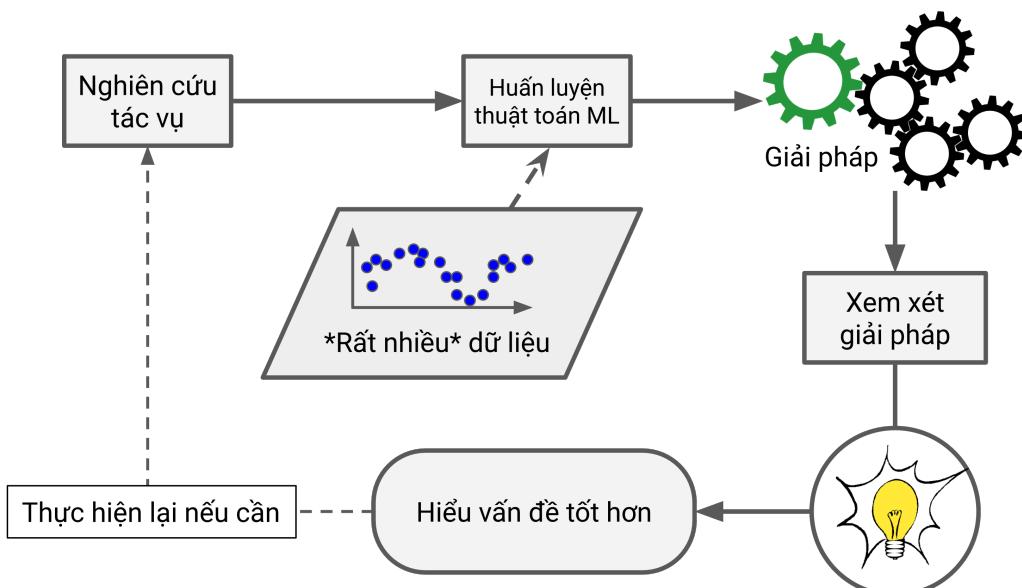
Hình 1.2. Tiếp cận theo hướng Học Máy



Hình 1.3. Tự động thích ứng với thay đổi

Một lĩnh vực khác mà Học Máy tỏa sáng là những bài toán quá phức tạp để có thể giải quyết theo hướng truyền thống hoặc không có sẵn giải thuật. Hãy lấy nhận diện giọng nói làm ví dụ. Giả sử ta muốn viết một chương trình đơn giản có khả năng phân biệt hai từ “one” và “two”. Để thấy rằng từ “two” bắt đầu với một âm cao (“T”), nên ta chỉ cần dùng cao độ âm thanh làm tiêu chí phân loại. Nhưng tất nhiên kỹ thuật này không thể hoạt động tốt với hàng ngàn từ được phát âm bởi hàng triệu người khác nhau trong không gian ồn ào và khi có cả tá ngôn ngữ khác nhau. Giải pháp tốt nhất (ít nhất là tại thời điểm viết quyển sách này) là viết một thuật toán có khả năng tự học khi được cung cấp nhiều bản thu âm mẫu của mỗi từ.

Cuối cùng, Học Máy còn có thể giúp con người học ([Hình 1.4](#)). Ta có thể kiểm tra các thuật toán học máy để biết những gì mà chúng đã học được (mặc dù việc này có thể trở nên khó khăn đối với một số thuật toán nhất định). Ví dụ, sau khi một bộ lọc thư rác đã được huấn luyện với đủ mẫu thư rác, ta có thể dễ dàng xem được danh sách các từ và tổ hợp từ được thuật toán cho là những dấu hiệu tốt nhất để nhận biết thư rác. Đôi khi chúng sẽ tiết lộ sự tương quan mà ta không hề hay biết hoặc các xu hướng mới, từ đó giúp ta hiểu rõ bài toán hơn. Việc áp dụng các kỹ thuật học máy để khai phá lượng dữ liệu lớn có thể giúp tìm ra các khuôn mẫu mà ta không thể thấy trực tiếp. Đây được gọi là *khai phá dữ liệu* (*data mining*).



Hình 1.4. Học Máy có thể giúp con người học

Tóm lại, Học Máy rất tốt cho:

- Những bài toán mà các giải pháp hiện có đòi hỏi quá nhiều quy luật hoặc cần tinh chỉnh nhiều: một thuật toán Học Máy thường có thể đơn giản hóa mã nguồn và hoạt động tốt hơn so với hướng truyền thống.
- Những bài toán phức tạp mà các phương pháp truyền thống không hoạt động tốt: giải pháp có thể là những kỹ thuật Học Máy tốt nhất.
- Môi trường thay đổi: một hệ thống Học Máy có thể thích ứng với dữ liệu mới.
- Việc khám phá tri thức từ các bài toán phức tạp và lượng dữ liệu lớn.

Các Ứng dụng Tiêu biểu

Hãy cùng điểm qua một vài ví dụ cụ thể về các tác vụ Học Máy, cũng như một số kỹ thuật để giải quyết các tác vụ đó:

Phân tích hình ảnh và tự động phân loại sản phẩm trên dây chuyền sản xuất

Đây là bài toán phân loại ảnh và thường được giải quyết bằng mạng nơ-ron tích chập (*Convolutional Neural Network – CNN*; sẽ được đề cập trong Tập 2).

Phát hiện khối u trong ảnh quét não

Đây là bài toán phân vùng theo nhóm (*semantic segmentation*), trong đó mọi điểm ảnh đều được phân loại (vì ta cần xác định vị trí chính xác và hình dạng của khối u). CNN cũng là phương pháp thường được sử dụng trong bài toán này.

Phân loại tin tức tự động

Đây là bài toán xử lý ngôn ngữ tự nhiên (*Natural Language Processing – NLP*), cụ thể hơn là phân loại văn bản, được giải quyết bằng mạng nơ-ron hồi tiếp (*Recurrent Neural Network – RNN*), CNN, hoặc Transformer (sẽ được đề cập trong Tập 2).

Đánh dấu bình luận phản cảm trong diễn đàn một cách tự động

Đây cũng là bài toán phân loại văn bản, sử dụng chung các công cụ NLP đã kể trên.

Tóm tắt tài liệu tự động

Đây là một nhánh của NLP được gọi là tóm tắt văn bản (*text summarization*), và cũng sử dụng các công cụ như trên.

Tạo một chatbot hoặc trợ lý cá nhân

Bài toán này liên quan đến nhiều tác vụ trong NLP, bao gồm hiểu ngôn ngữ tự nhiên (*Natural Language Understanding – NLU*) và hệ thống hỏi-đáp.

Dự báo doanh thu công ty của năm tiếp theo, dựa trên nhiều chỉ số hiệu suất

Đây là tác vụ hồi quy (nghĩa là dự đoán giá trị) và có thể được giải quyết bằng bất kỳ mô hình hồi quy nào, như Hồi quy Tuyến tính (*Linear Regression*) hoặc Hồi quy Đa thức (tham khảo [Chương 4](#)), SVM hồi quy (tham khảo [Chương 5](#)), Rừng Ngẫu nhiên Hồi quy (tham khảo [Chương 7](#)), hoặc mạng nơ-ron nhân tạo (*artificial neural network* – sẽ được đề cập trong Tập 2). Nếu muốn đưa vào mô hình một chuỗi các chỉ số hiệu suất trong quá khứ, ta có thể sử dụng RNN, CNN, hoặc Transformer (sẽ được đề cập trong Tập 2).

Tương tác với ứng dụng thông qua giọng nói

Đây là bài toán nhận diện giọng nói (*speech recognition*) và đòi hỏi ta phải xử lý các đoạn âm thanh. Vì bản thân các đoạn âm thanh là các chuỗi dài và phức tạp, chúng thường được xử lý bằng RNN, CNN hoặc Transformer (sẽ được đề cập trong Tập 2).

Phát hiện gian lận thẻ tín dụng

Đây là bài toán phát hiện bất thường (*anomaly detection* – tham khảo [Chương 9](#)).

Phân nhóm khách hàng dựa trên sản phẩm tiêu thụ để thiết kế chiến lược tiếp thị khác nhau cho mỗi phân khúc

Đây là bài toán phân cụm (*clustering* – tham khảo [Chương 9](#)).

Biểu diễn một tập dữ liệu phức tạp, nhiều chiều trong biểu đồ một cách rõ ràng và hữu ích

Đây là bài toán trực quan hóa dữ liệu, thường liên quan tới các kỹ thuật giảm chiều (tham khảo [Chương 8](#)).

Gợi ý sản phẩm mà khách hàng có thể sẽ quan tâm dựa trên những sản phẩm mà họ đã mua trong quá khứ

Đây là bài toán xây dựng hệ thống đề xuất. Một hướng tiếp cận là đưa các đơn hàng trong quá khứ (và các thông tin khác về khách hàng) vào một mạng nơ-ron nhân tạo (sẽ được đề cập trong Tập 2) để dự đoán sản phẩm có khả năng cao sẽ được mua tiếp theo. Mạng nơ-ron này thường được huấn luyện với chuỗi các sản phẩm đã mua trong quá khứ của mọi khách hàng.

Xây dựng bot thông minh biết chơi trò chơi

Bài toán này thường được giải quyết thông qua Học Tăng Cường (*Reinforcement Learning* hay *RL* – sẽ được đề cập trong Tập 2), một nhánh của Học Máy với mục tiêu huấn luyện tác nhân (*agent* – ở đây là bot) để chọn các hành động sao cho phần thưởng được cực đại hóa theo thời gian (ví dụ, con bot có thể được thưởng mỗi khi người chơi bị mất máu), trong một môi trường cho trước (trường hợp này là trong một trò chơi). Chương trình AlphaGo nổi tiếng từng đánh bại nhà vô địch thế giới trong bộ môn cờ vây đã được xây dựng thông qua RL.

Danh sách này còn rất rất dài, nhưng hy vọng những ví dụ trên đã giúp bạn nắm được phần nào về độ rộng và phức tạp của các tác vụ mà Học Máy có thể giải quyết cũng như các kỹ thuật tương ứng mà ta có thể áp dụng.

Các kiểu Hệ thống Học Máy

Có rất nhiều kiểu hệ thống Học Máy khác nhau, và chúng có thể được phân loại thành các hạng mục rộng theo các tiêu chí sau:

- Chúng có được huấn luyện dưới sự giám sát của con người hay không (học giám sát, học không giám sát, học bán giám sát và học tăng cường)
- Chúng có thể học từ các dòng dữ liệu gia tăng hay không (học trực tuyến so với học theo batch)
- Chúng hoạt động bằng cách chỉ đơn thuần so sánh các điểm dữ liệu mới với các điểm dữ liệu đã biết, hay bằng cách phát hiện các khuôn mẫu trong dữ liệu huấn luyện và xây dựng một mô hình dự đoán, tương tự công việc của các nhà khoa học (học dựa trên mẫu so với học dựa trên mô hình).

Các tiêu chí này không khắc lẫn nhau, và có thể được kết hợp một cách tùy ý. Ví dụ, một hệ thống lọc thư rác tân tiến có thể học liên tục bằng cách huấn luyện một mô hình mạng nơ-ron sâu trên các mẫu thư rác và thư thông thường mới. Do đó, đây là một hệ thống học trực tuyến, dựa trên mô hình và có giám sát.

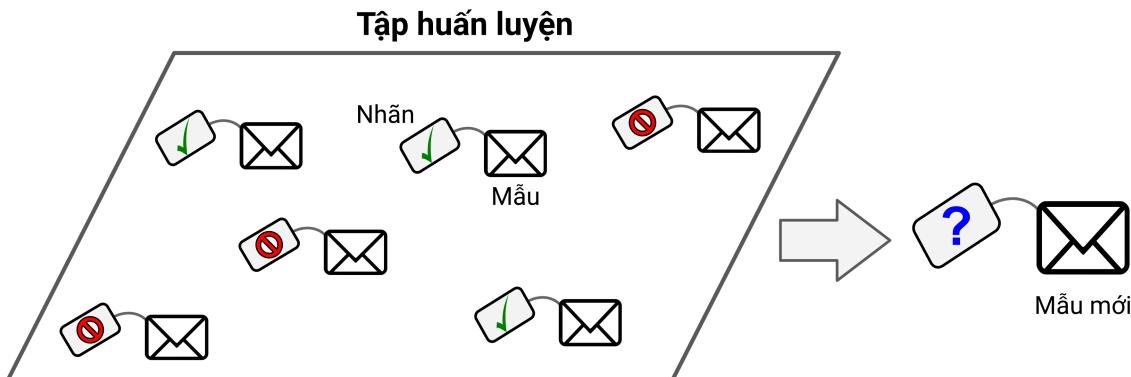
Hãy cùng xem xét từng yếu tố trên kỹ hơn một chút.

Học có Giám sát/không Giám sát

Các hệ thống học máy có thể được phân loại dựa trên mức độ và kiểu giám sát được thực hiện khi chúng đang ở trong giai đoạn huấn luyện. Có bốn hạng mục chính: học có giám sát, học không giám sát, học bán giám sát và học tăng cường.

Học Có Giám Sát

Trong *học có giám sát* (*supervised learning*), tập huấn luyện mà ta đưa vào thuật toán đã bao gồm cả kết quả mong muốn, và kết quả đó được gọi là *nhãn* ([Hình 1.5](#)).



Hình 1.5. Một tập huấn luyện đã được gán nhãn cho tác vụ phân loại thư rác (một ví dụ của học có giám sát)

Một tác vụ học có giám sát điển hình là *phân loại* (*classification*). Bộ lọc thư rác là ví dụ tiêu biểu cho tác vụ này: nó được huấn luyện với nhiều mẫu thư điện tử cùng với *nhãn* tương ứng (thư rác hoặc thư thông thường), từ đó học cách phân loại các thư điện tử mới.

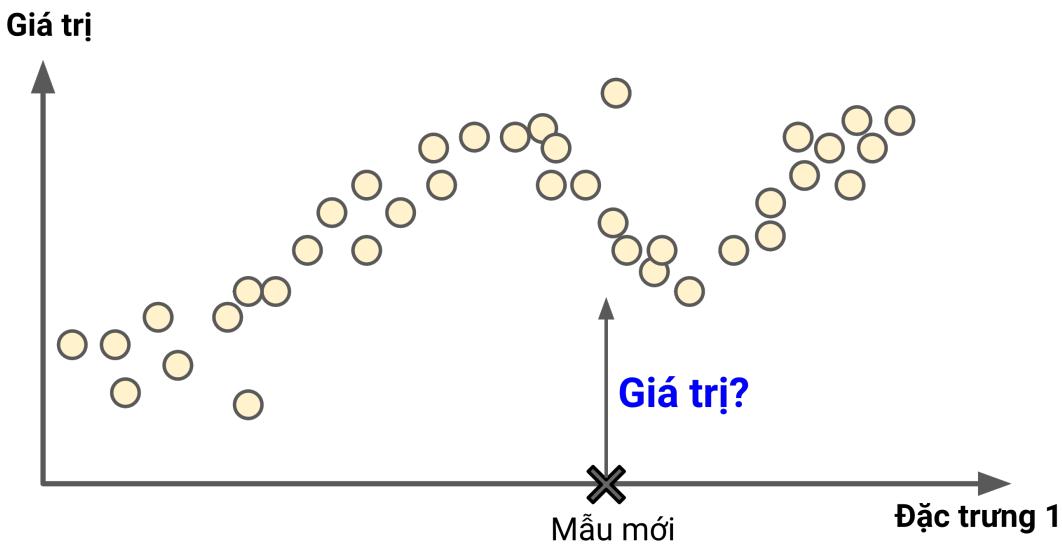
Một tác vụ điển hình khác là dự đoán giá trị số *mục tiêu*, chẳng hạn như giá xe hơi từ một tập hợp các *đặc trưng* cho trước (số dặm, tuổi xe, hãng xe, v.v.) được gọi là *yếu tố dự đoán* (*predictor*). Tác vụ này có tên gọi là *hồi quy* (*regression* – [Hình 1.6](#)).¹ Để huấn luyện hệ thống này, ta cần cung cấp cho nó rất nhiều các mẫu xe hơi cùng các yếu tố dự đoán và nhãn của chúng (giá xe).

Ghi chú

Trong Học Máy, *thuộc tính* (*attribute*) là một kiểu dữ liệu (ví dụ: “số dặm”), trong khi *đặc trưng* (*feature*) có thể có nhiều nghĩa tùy thuộc vào ngữ cảnh. Nhìn chung, một đặc trưng là một thuộc tính đi kèm với giá trị của nó (ví dụ, “số dặm = 15,000”). Tuy nhiên, nhiều người thường sử dụng hai từ *thuộc tính* và *đặc trưng* với ý nghĩa giống nhau.

Lưu ý rằng một số thuật toán hồi quy cũng có thể được sử dụng để phân loại và ngược lại. Ví dụ, *Hồi quy Logistic* (*Logistic Regression*) thường được sử dụng để phân loại, bởi nó có thể trả về một giá trị tương ứng với xác suất mà mẫu dữ liệu thuộc về một lớp nhất định (ví dụ: 20% khả năng là thư rác).

¹ Cái tên nghe có vẻ kỳ lạ này là một thuật ngữ thống kê được giới thiệu bởi Francis Galton khi ông đang nghiên cứu về việc con cái của những người cao thường có xu hướng thấp hơn cha mẹ của chúng. Vì những đứa trẻ có chiều cao thấp hơn, ông gọi hiện tượng này là *hồi quy đến trung bình* (*regression to the mean*). Tên gọi này sau đó được sử dụng cho các phương pháp phân tích mối tương quan giữa các biến của ông.



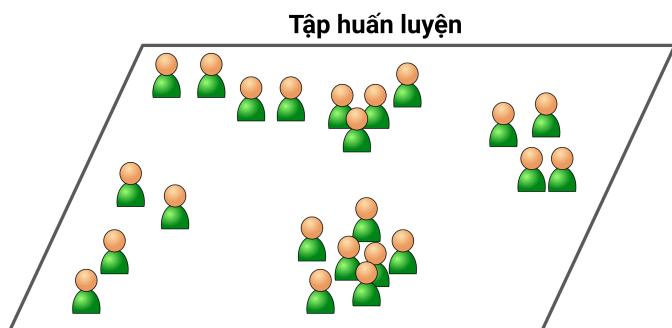
Hình 1.6. Một bài toán hồi quy: dự đoán một giá trị dựa trên đặc trưng đầu vào (thông thường có thể có nhiều đầu vào và đôi khi là nhiều đầu ra)

Sau đây là một số thuật toán học có giám sát quan trọng nhất (được đề cập trong cuốn sách này):

- k-Điểm Gần nhất
- Hồi quy Tuyến tính
- Hồi quy Logistic
- Máy Vector Hỗ trợ
- Cây Quyết định và Rừng Ngẫu nhiên
- Mạng nơ-ron²

Học Không Giám Sát

Trong *học không giám sát* (*unsupervised learning*), như bạn đã có thể đoán được, dữ liệu huấn luyện không được gán nhãn (Hình 1.7). Hệ thống cố gắng tự học mà không cần giáo viên.



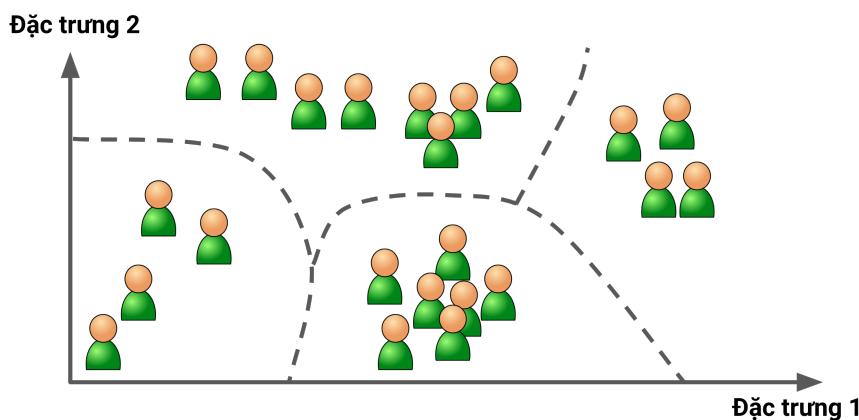
Hình 1.7. Một tập dữ liệu không có nhãn dành cho tác vụ học không giám sát

² Một số kiến trúc mạng nơ-ron có thể học một cách không giám sát, ví dụ như bộ tự mã hóa (*autoencoder*) và máy Boltzmann giới hạn (*restricted Boltzmann machines*). Chúng cũng có thể học theo hướng bán giám sát, ví dụ như mạng niềm tin sâu (*deep belief network*) hoặc thông qua quá trình tiền huấn luyện không giám sát.

Dưới đây là một số thuật toán học không giám sát (đa số sẽ được đề cập trong các [Chương 8](#) và [Chương 9](#)).

- Phân cụm (Clustering)
 - K-Điểm trung bình (K-Means)
 - DBSCAN
 - Phân cụm phân cấp (Hierarchical Cluster Analysis – HCA)
- Phát hiện bất thường và tính mới (Anomaly detection and novelty detection)
 - SVM một lớp (One-class SVM)
 - Rừng cô lập (Isolation Forest)
- Trực quan hóa (visualization) và giảm chiều (dimensionality reduction)
 - Phân tích thành phần chính (Principal Component Analysis – PCA)
 - PCA hạt nhân (Kernel PCA)
 - Embedding tuyến tính cục bộ (Locally Linear Embedding – LLE)
 - Embedding lân cận ngẫu nhiên theo phân phối t (t-Distributed Stochastic Neighbor Embedding – t-SNE)
- Học luật kết hợp (Association rule)
 - Apriori
 - Eclat

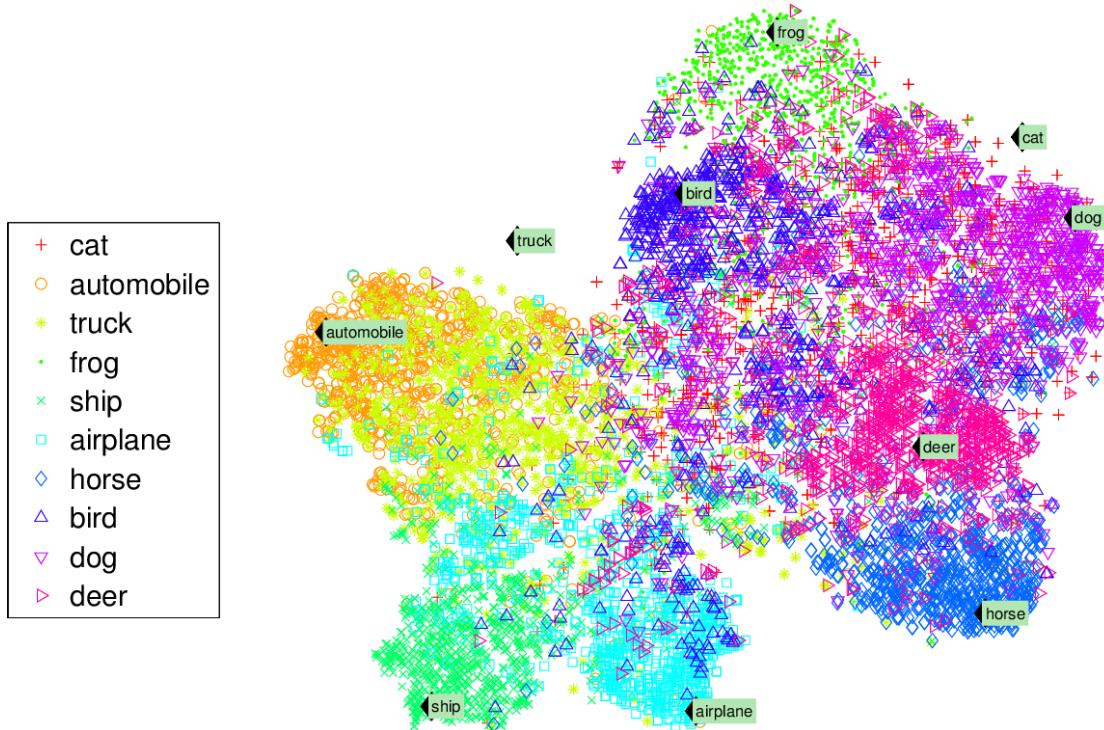
Giả sử bạn có rất nhiều dữ liệu về người đọc blog của mình. Bạn có thể sẽ muốn chạy một thuật toán *phân cụm* để phát hiện các nhóm người đọc giống nhau ([Hình 1.8](#)). Bạn không hề cho thuật toán biết mỗi người thuộc nhóm nào mà nó sẽ phải tự tìm các mối liên kết. Ví dụ, thuật toán có thể thấy rằng 40% người đọc là nam giới, thích đọc truyện tranh và thường đọc blog của bạn vào buổi tối, trong khi 20% còn trẻ, thích khoa học viễn tưởng và thường đọc blog vào cuối tuần. Nếu bạn sử dụng thuật toán *phân cụm phân cấp*, mỗi nhóm có thể được phân chia thành các nhóm nhỏ hơn. Việc phân cụm có thể giúp bạn viết bài nhắm đến từng nhóm.



Hình 1.8. Phân cụm

Các thuật toán *trực quan hóa* cũng là ví dụ tiêu biểu của học không giám sát: ta đưa vào rất nhiều dữ liệu phức tạp, không có nhãn, và thuật toán trả về biểu diễn hai hoặc ba chiều của

dữ liệu, có thể được minh họa dễ dàng bằng đồ thị ([Hình 1.9](#)). Các thuật toán này cố gắng giữ nguyên cấu trúc nhiều nhất có thể (ví dụ như giữ các cụm phân biệt trong không gian đầu vào không chồng lấn lên nhau khi biểu diễn), nên ta có thể hiểu cách dữ liệu được tổ chức và xác định các khuôn mẫu ẩn trong dữ liệu.



Hình 1.9. Ví dụ trực quan hóa bằng t-SNE cho các cụm ngữ nghĩa³

Một tác vụ liên quan là *giảm chiều*, với mục tiêu là đơn giản hóa dữ liệu mà không làm mất quá nhiều thông tin. Một phương pháp khả thi là gộp các đặc trưng tương quan với nhau thành một. Ví dụ, quãng đường đi được của một chiếc xe có thể tương quan mạnh với tuổi thọ của chiếc xe đó, nên thuật toán giảm chiều sẽ gộp chúng lại thành một đặc trưng biểu diễn sự hao mòn của chiếc xe. Đây được gọi là *trích xuất đặc trưng* (*feature extraction*).

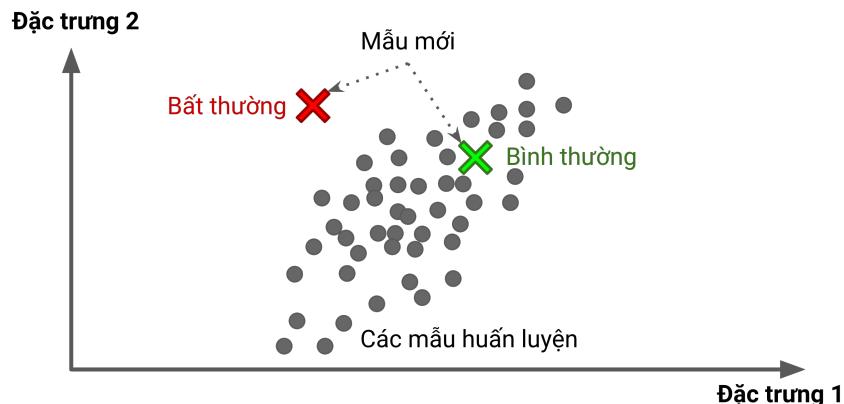
Mẹo

Giảm chiều dữ liệu huấn luyện trước khi đưa chúng vào một thuật toán Học Máy (ví dụ như một thuật toán học có giám sát) thường là một ý tưởng tốt. Thuật toán sẽ chạy nhanh hơn, dữ liệu sẽ chiếm ít dung lượng ổ cứng và bộ nhớ hơn, và trong một vài trường hợp có thể đem lại chất lượng tốt hơn.

Một tác vụ không giám sát quan trọng khác là *phát hiện bất thường*, ví dụ như phát hiện các giao dịch thẻ tín dụng bất thường để ngăn chặn sai phạm, bắt lỗi trong dây chuyền sản xuất, hoặc tự động loại bỏ các điểm ngoại lai trong tập dữ liệu trước khi đưa chúng vào các thuật toán học. Hệ thống được huấn luyện trên hầu hết các mẫu bình thường nên có thể nhận ra các mẫu này. Sau đó, khi thấy một mẫu mới, hệ thống có thể chỉ ra mẫu này là bình thường hay bất

³ Có thể thấy rằng động vật và phương tiện giao thông nằm ở hai phía riêng biệt, và ngựa gần với hươu nhưng xa so với chim. Hình ảnh được tái tạo dưới sự cho phép của Richard Socher và các cộng sự, “Zero-Shot Learning Through Cross-Modal Transfer,” *Proceedings of the 26th International Conference on Neural Information Processing Systems* 1 (2013): 935–943.

thường (tham khảo [Hình 1.10](#)). Một tác vụ tương tự là *phát hiện tính mới*. Tác vụ này nhằm đến việc phát hiện các điểm không giống các điểm khác trong tập huấn luyện. Tác vụ này đòi hỏi một tập huấn luyện rất “sạch”, không hề chứa các điểm mà thuật toán sẽ cần phải phát hiện. Ví dụ, nếu bạn có hàng nghìn ảnh chó, và 1% số ảnh đó là của giống chó Chihuahua, thuật toán phát hiện tính mới sẽ không xét các bức ảnh Chihuahua mới là ảnh có tính mới. Mặt khác, các thuật toán phát hiện bất thường vẫn có thể xem giống chó này là hiếm và khác các giống chó còn lại, nên có lẽ sẽ phân loại chúng là bất thường (ở đây không có ý kỉ thị Chihuahua).

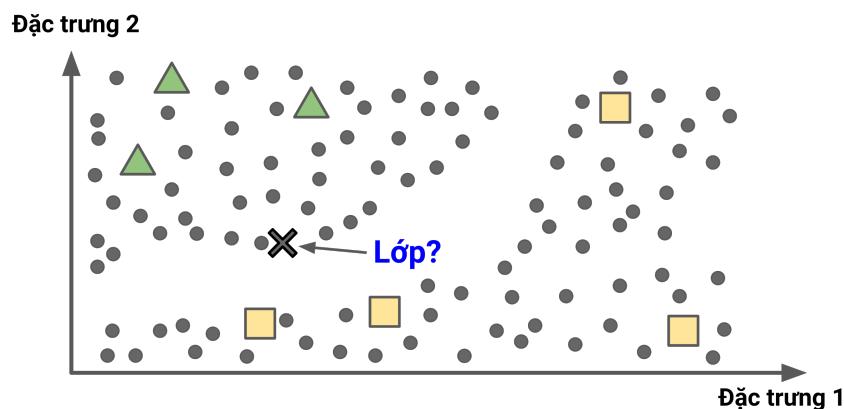


Hình 1.10. Phát hiện bất thường

Cuối cùng, một tác vụ không giám sát phổ biến khác là *học luật kết hợp (association rule learning)*, với mục tiêu là đào sâu vào lượng lớn dữ liệu để khám phá các mối quan hệ thú vị giữa các thuộc tính. Để lấy ví dụ, giả sử bạn sở hữu một siêu thị. Việc áp dụng luật kết hợp cho các hóa đơn bán hàng có thể hé lộ rằng người mua sôt BBQ và bim bim có xu hướng mua thêm thịt bò. Nhờ đó, bạn có thể đặt các mặt hàng này gần nhau.

Học bán giám sát

Sự tồn kén về thời gian lắn chi phí của quá trình gán nhãn dữ liệu dẫn đến hệ quả là chỉ một phần nhỏ dữ liệu được gán nhãn. Các thuật toán có thể làm việc với tập dữ liệu được gán nhãn một phần được gọi là thuật toán *học bán giám sát (semisupervised learning – [Hình 1.11](#))*.



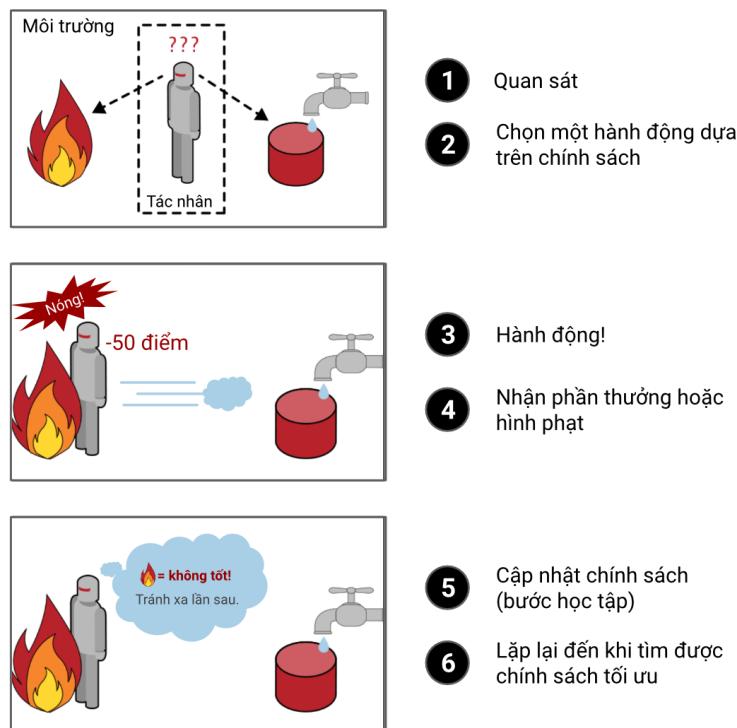
Hình 1.11. Học bán giám sát với hai lớp (hình tam giác và hình vuông): các mẫu chưa gán nhãn (hình tròn) giúp phân loại một đối tượng mới (hình chữ thập) vào lớp hình tam giác thay vì hình vuông, mặc dù nó gần với các hình vuông đã được gán nhãn hơn

Các dịch vụ lưu trữ ảnh là một ví dụ tiêu biểu, điển hình như Google Photos. Sau khi tải lên tất cả ảnh gia đình của bạn lên đó, chúng sẽ tự động nhận diện cùng một người A xuất hiện trong các ảnh 1, 5 và 11, còn người B xuất hiện ở ảnh 2, 5 và 7. Đây là phần không giám sát của thuật toán (phân cụm). Giờ hệ thống chỉ cần bạn chỉ ra những người này là ai. Bằng cách thêm một nhãn cho mỗi người⁴, hệ thống có thể gán tên cho tất cả mọi người trong ảnh, giúp việc tìm kiếm ảnh trở nên thuận tiện hơn.

Đa phần các thuật toán học bán giám sát là sự kết hợp giữa các thuật toán học không giám sát và có giám sát. Ví dụ, *mạng niềm tin sâu* (*deep belief network* – DBN) dựa trên các thành phần không giám sát có tên là *máy Boltzmann giới hạn* (*restricted Boltzmann machine* – RBM) chồng lên nhau. RBM được huấn luyện tuần tự theo cơ chế không giám sát, rồi sau đó toàn bộ hệ thống được tinh chỉnh bằng các kỹ thuật có giám sát.

Học Tăng cường

Học Tăng cường là phương pháp học có cấu trúc rất khác. Hệ thống học trong RL được gọi là *tác nhân* (*agent*). Nó có thể quan sát môi trường xung quanh, chọn và thực hiện các hành động, sau đó nhận về *điểm thưởng* – *reward* (hoặc *lượng phạt* – *penalty* dưới dạng điểm thưởng âm, như trong [Hình 1.12](#)). Hệ thống sau đó cần tự học chiến lược tốt nhất để nhận về nhiều điểm thưởng nhất qua thời gian, và chiến lược này được gọi là *chính sách* (*policy*). Một chính sách định nghĩa hành động mà tác nhân nên chọn khi ở trong một tình huống cụ thể.



Hình 1.12. Học Tăng cường

⁴ Đó là khi hệ thống hoạt động hoàn hảo. Trong thực tế, dịch vụ có thể tạo nhiều cụm cho mỗi người hoặc đôi lúc nhầm lẫn giữa hai người nhìn giống nhau. Nên có thể ta sẽ cần cung cấp một vài nhãn cho mỗi người và xoá bớt một vài cụm một cách thủ công.

Ví dụ, nhiều rô bốt được lập trình với thuật toán học tăng cường để học cách đi lại. AlphaGo của DeepMind cũng là một ví dụ về học tăng cường: nó xuất hiện trên trang nhất của các bản tin trong tháng 5 năm 2017 khi đánh bại nhà vô địch cờ vây thế giới Ke Jie (Kha Khiết). AlphaGo đã học được chính sách dẫn đến chiến thắng bằng cách phân tích hàng triệu ván cờ, rồi tự chơi rất nhiều ván với chính nó. Chú ý rằng AlphaGo không học trong khi thi đấu với Ke Jie mà chỉ đơn thuần áp dụng chính sách nó đã học được từ trước.

Học theo Batch và Học Trực tuyến

Một tiêu chí khác để phân loại các hệ thống Học Máy nằm ở việc chúng có thể liên tục học từ các dòng dữ liệu gia tăng hay không.

Học theo Batch

Các hệ thống *học theo batch* không có khả năng học gia tăng: chúng phải được huấn luyện bằng tất cả dữ liệu khả dụng. Nhìn chung thì việc này sẽ tốn rất nhiều thời gian và tài nguyên tính toán, nên quá trình huấn luyện thường diễn ra một cách ngoại tuyến. Đầu tiên, hệ thống được huấn luyện và hoàn tất việc học trước khi được triển khai thực tế. Trong quá trình triển khai thực tế, hệ thống chỉ áp dụng lại những gì đã được học. Đây được gọi là *học ngoại tuyến (offline learning)*. Nếu ta muốn một hệ thống học theo batch cập nhật thêm dữ liệu mới (ví dụ như một loại thư rác mới), ta cần huấn luyện một phiên bản mới của hệ thống từ đầu với toàn bộ dữ liệu (bao gồm cả dữ liệu mới và cũ), rồi thay hệ thống cũ bằng hệ thống mới.

May mắn là toàn bộ quá trình huấn luyện, đánh giá và triển khai một hệ thống Học Máy có thể được tự động hóa một cách khá dễ dàng (minh họa trong [Hình 1.3](#)), nên ngay cả một hệ thống học theo batch cũng có thể thích ứng với thay đổi. Ta chỉ cần cập nhật dữ liệu và huấn luyện một phiên bản mới lại từ đầu khi cần thiết.

Phương án này đơn giản và thường hoạt động tốt, nhưng huấn luyện trên toàn bộ dữ liệu có thể tốn hàng giờ, nên hệ thống thường được huấn luyện lại chỉ sau 24 giờ hoặc thậm chí sau mỗi tuần. Nếu hệ thống cần thích ứng với dữ liệu thay đổi nhanh (ví dụ như giá cổ phiếu), ta sẽ cần một giải pháp linh hoạt hơn.

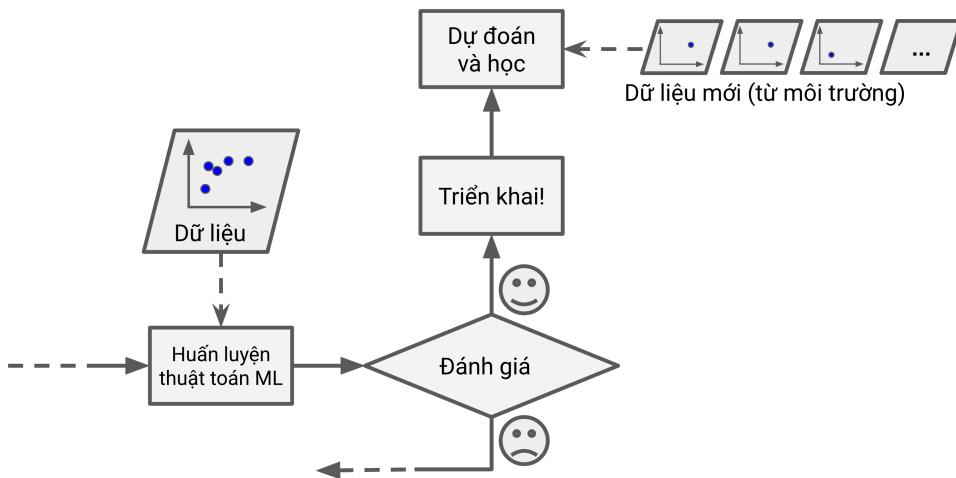
Thêm vào đó, huấn luyện trên toàn bộ dữ liệu tiêu tốn rất nhiều tài nguyên tính toán (CPU, dung lượng bộ nhớ và ổ đĩa, I/O ổ đĩa và mạng, v.v.). Nếu có rất nhiều dữ liệu, việc tự động huấn luyện lại mô hình hàng ngày có thể tiêu tốn rất nhiều tiền. Nếu lượng dữ liệu có kích thước khổng lồ, học theo batch còn có thể trở nên bất khả thi.

Cuối cùng, nếu hệ thống cần có khả năng học độc lập với tài nguyên hạn chế (ví dụ như ứng dụng điện thoại thông minh hoặc rover trên Sao Hoả), việc lưu trữ lượng lớn dữ liệu huấn luyện và sử dụng nhiều tài nguyên để huấn luyện hàng giờ mỗi ngày là không thể.

May mắn thay, chúng ta có một lựa chọn tốt hơn cho tất cả các trường hợp trên, đó là sử dụng các thuật toán có khả năng học gia tăng.

Học Trực tuyến

Trong *học trực tuyến (online learning)*, ta huấn luyện mô hình một cách gia tăng bằng cách tuần tự truyền dữ liệu theo từng điểm dữ liệu hoặc theo lô nhỏ gọi là *mini-batch*. Mỗi bước học rất nhanh và không tốn nhiều tài nguyên, nên hệ thống có thể dễ dàng học với dữ liệu mới khi cần (tham khảo [Hình 1.13](#)).



Hình 1.13. Trong học trực tuyến, mô hình được huấn luyện và triển khai thực tế, rồi tiếp tục học khi có dữ liệu mới

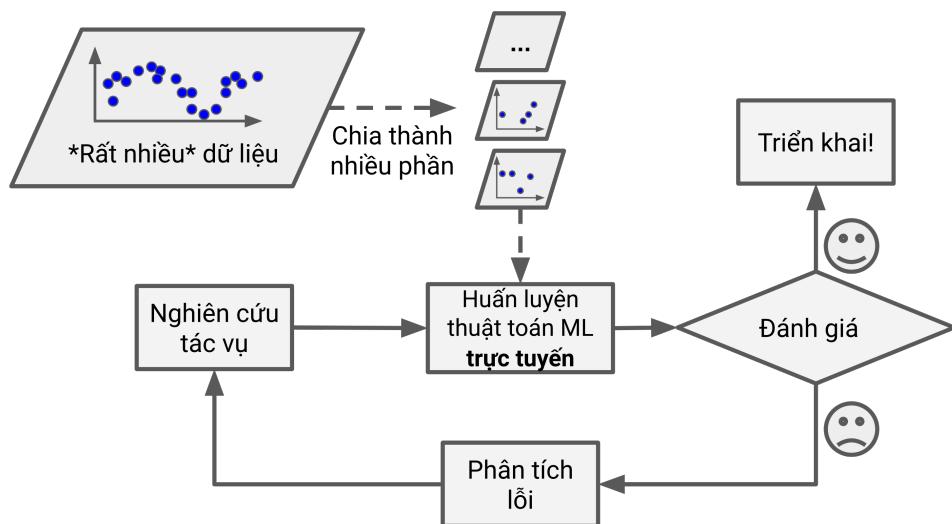
Học trực tuyến là giải pháp tốt cho các hệ thống nhận dữ liệu là các luồng liên tục (như giá cổ phiếu) và cần thích ứng với thay đổi một cách nhanh chóng hoặc độc lập. Phương pháp này cũng là lựa chọn tốt nếu tài nguyên tính toán bị giới hạn: một khi hệ thống đã học xong từ dữ liệu mới, dữ liệu này không còn cần thiết nữa và có thể được loại bỏ (trừ khi ta muốn quay lại trạng thái trước đó và “nạp lại” dữ liệu). Điều này giúp tiết kiệm rất nhiều dung lượng lưu trữ.

Các thuật toán học trực tuyến cũng có thể được sử dụng để huấn luyện hệ thống trên các tập dữ liệu khổng lồ và không nằm vừa trong bộ nhớ chính (đây được gọi là *học ngoài bộ nhớ chính – out-of-core learning*). Thuật toán chỉ nạp một phần dữ liệu, huấn luyện trên phần đó, rồi lặp lại quá trình cho tới khi đã chạy trên toàn bộ dữ liệu (tham khảo [Hình 1.14](#)).

Lưu ý

Học ngoài bộ nhớ chính thường được thực hiện ngoại tuyến (không được thực hiện trên hệ thống đang được triển khai), nên cái tên *học trực tuyến* có thể sẽ gây hiểu lầm. Hãy nghĩ về nó như là *học gia tăng*.

Một tham số quan trọng trong các hệ thống học trực tuyến là tốc độ thích ứng với dữ liệu đang thay đổi: tham số này được gọi là *tốc độ học* (*learning rate*). Nếu tốc độ học cao, hệ thống sẽ nhanh chóng thích ứng với dữ liệu mới, nhưng cũng sẽ có xu hướng quên dữ liệu cũ nhanh hơn (ta không muốn một bộ lọc thư rác chỉ lọc các loại thư rác nó thấy gần đây nhất). Ngược lại, nếu tốc độ học thấp, hệ thống sẽ có sức Ý lớn hơn – tức sẽ học chậm hơn, nhưng cũng sẽ bớt nhạy cảm với nhiễu trong dữ liệu mới hoặc với chuỗi dữ liệu không có tính đại diện (ngoại lai).



Hình 1.14. Sử dụng học trực tuyến cho các tập dữ liệu khổng lồ

Một thách thức lớn với học trực tuyến là nếu dữ liệu kém chất lượng được đưa vào mô hình, chất lượng của mô hình sẽ giảm. Nếu đó là một hệ thống trực tiếp (*live system*), khách hàng sẽ nhận ra. Ví dụ, dữ liệu kém chất lượng có thể đến từ một cảm biến bị hỏng trên rõ bối, hoặc từ một người đang cố gắng đánh lừa công cụ tìm kiếm để kết quả của họ nằm ở vị trí đầu trên trang tìm kiếm. Để giảm thiểu rủi ro này, ta cần giám sát hệ thống chặt chẽ và ngay lập tức dừng huấn luyện (và có thể quay lại trạng thái hoạt động tốt trước đó) khi phát hiện thấy chất lượng của mô hình giảm đi đáng kể. Ta cũng có thể giám sát dữ liệu đầu vào và xử lý dữ liệu bất thường (ví dụ, bằng cách sử dụng một thuật toán phát hiện bất thường).

Học dựa trên Mẫu và dựa trên Mô hình

Một cách khác để phân loại các thuật toán học máy là dựa vào cách chúng *khái quát hóa*. Phần lớn các tác vụ học máy sẽ xoay quanh việc đưa ra dự đoán. Nghĩa là với các mẫu dữ liệu huấn luyện cho trước, mô hình cần có khả năng đưa ra những dự đoán tốt (khái quát hóa tốt) đối với các mẫu dữ liệu mà nó chưa từng thấy. Việc có một phép đo phù hợp để đánh giá quá trình huấn luyện là tốt nhưng vẫn không đủ. Mục tiêu cuối cùng của mô hình là hoạt động tốt trên dữ liệu mới.

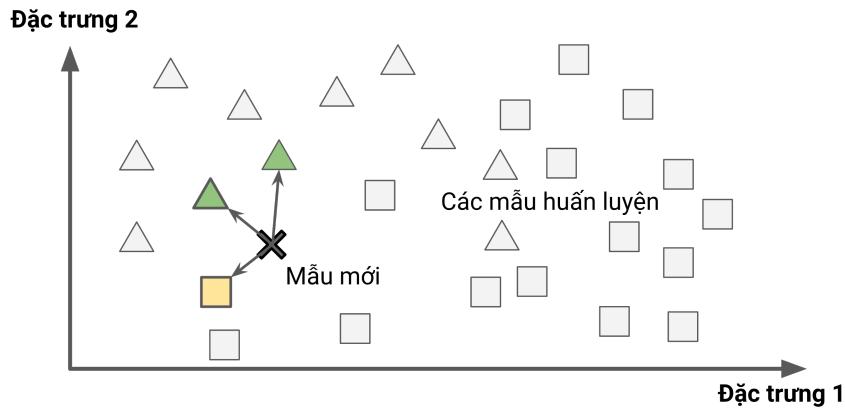
Có hai hướng tiếp cận chính để khái quát hóa: học dựa trên mẫu (*instance-based learning*) và học dựa trên mô hình (*model-based learning*).

Học dựa trên Mẫu

Có lẽ phương thức học đơn giản nhất chính là học thuộc lòng. Nếu ta muốn tạo một bộ lọc thư rác theo cách này, nó chỉ cần lọc ra tất cả các thư giống hệt thư được người dùng đánh dấu trước đó là thư rác. Đây không phải là giải pháp tệ nhất, nhưng chắc chắn cũng không phải là giải pháp tốt nhất.

Thay vì chỉ đánh dấu các thư hoàn toàn giống thư rác, bộ lọc cũng có thể được lập trình để phát hiện các thư gần giống với các thư rác đã biết. Để làm vậy, ta cần có một *phép đo độ tương đồng* (*measure of similarity*) giữa hai lá thư. Một phép đo độ tương đồng (rất cơ bản) giữa hai lá thư là đếm số từ cùng xuất hiện trong cả hai. Hệ thống sẽ đánh dấu một bức thư là thư rác nếu nó có nhiều từ trùng lặp với một thư rác đã biết.

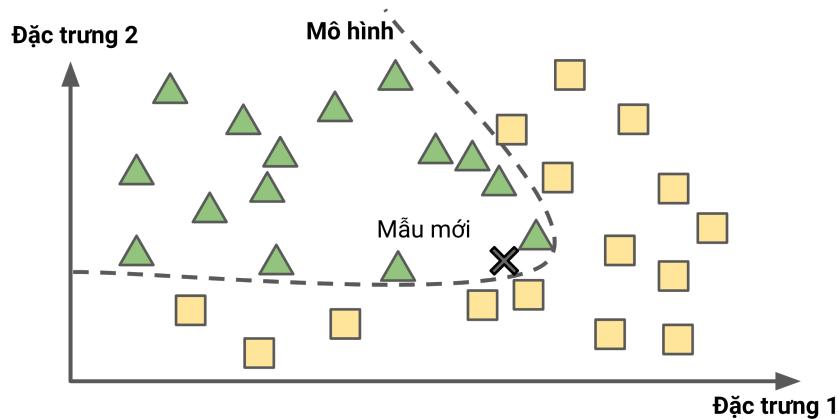
Phương pháp này được gọi là *học dựa trên mẫu* (*instance-based learning*): hệ thống học thuộc các mẫu dữ liệu, rồi khái quát hóa với các mẫu dữ liệu mới bằng việc sử dụng một phép đo độ tương đồng để so sánh với toàn bộ (hoặc một phần) dữ liệu đã học. Ví dụ, trong [Hình 1.15](#), mẫu mới sẽ được gán nhãn là hình tam giác vì đa số các mẫu giống nó nhất đều thuộc lớp hình tam giác.



Hình 1.15. Học dựa trên mẫu

Học dựa trên Mô hình

Một cách khác để khái quát hóa từ dữ liệu cho trước là xây dựng một mô hình từ dữ liệu rồi dùng mô hình đó để đưa ra các *dự đoán*. Phương pháp này được gọi là *học dựa trên mô hình* – *model-based learning* ([Hình 1.16](#)).



Hình 1.16. Học dựa trên mô hình

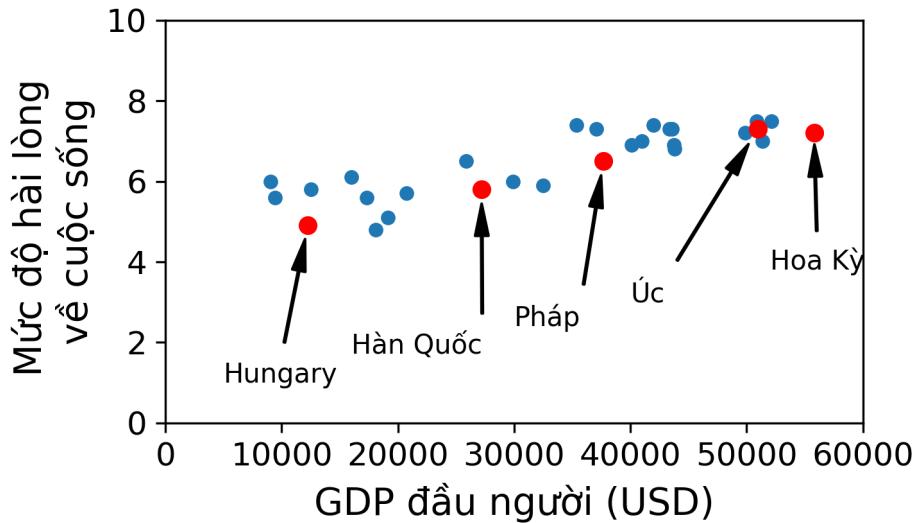
Ví dụ, giả sử ta muốn biết tiền có làm con người hạnh phúc hay không, nên ta tải xuống tập dữ liệu Better Life Index từ [trang web của OECD](#), và thống kê về tổng sản phẩm quốc nội (GDP) bình quân đầu người từ [trang web của IMF](#). Sau đó ta gộp hai bảng lại và sắp xếp theo GDP đầu người. [Bảng 1.1](#) là một phần được trích từ bảng này.

Hãy vẽ đồ thị cho dữ liệu của các nước trên ([Hình 1.17](#)).

CHƯƠNG 1. TOÀN CẢNH HỌC MÁY

Bảng 1.1. Tiền có giúp con người hạnh phúc không?

Quốc gia	GDP đầu người (USD)	Mức độ hài lòng về cuộc sống
Hungary	12,240	4.9
Hàn Quốc	27,195	5.8
Pháp	37,675	6.5
Úc	50,962	7.3
Hoa Kỳ	55,805	7.2



Hình 1.17. Bạn có nhận thấy xu hướng nào không?

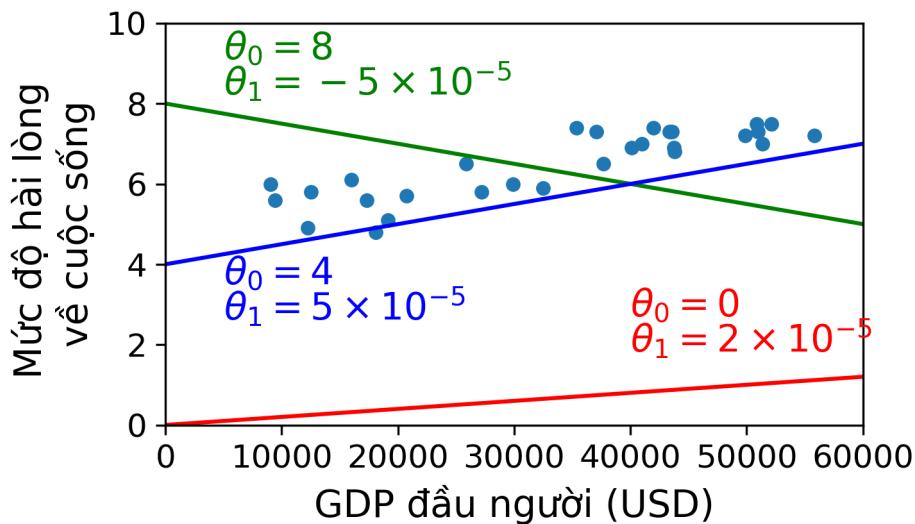
Chúng ta có thể thấy một xu hướng ở đây! Mặc dù dữ liệu có *nhiều* (tức là phần nào đó có tính ngẫu nhiên), ta vẫn nhận ra rằng mức độ hài lòng về cuộc sống thường như tăng lên tuyến tính với GDP đầu người. Do đó, ta quyết định mô hình hóa mức độ hài lòng bằng một hàm tuyến tính theo GDP đầu người. Bước này được gọi là *lựa chọn mô hình*: ta chọn một *mô hình tuyến tính* của mức độ hài lòng với duy nhất một thuộc tính là GDP đầu người ([Phương trình 1.1](#)).

$$\text{life_satisfaction} = \theta_0 + \theta_1 \times \text{GDP_per_capita}$$

[Phương trình 1.1](#). Một mô hình tuyến tính đơn giản

Mô hình này có hai *tham số* (*model parameter*) là θ_0 và θ_1 .⁵ Bằng cách thay đổi tham số này, mô hình có thể biểu diễn bất kỳ hàm tuyến tính nào, như trong [Hình 1.18](#).

⁵ Theo quy ước, chữ cái Hy Lạp θ (theta) thường được sử dụng để biểu diễn các tham số mô hình.



Hình 1.18. Một vài hàm tuyến tính khả thi

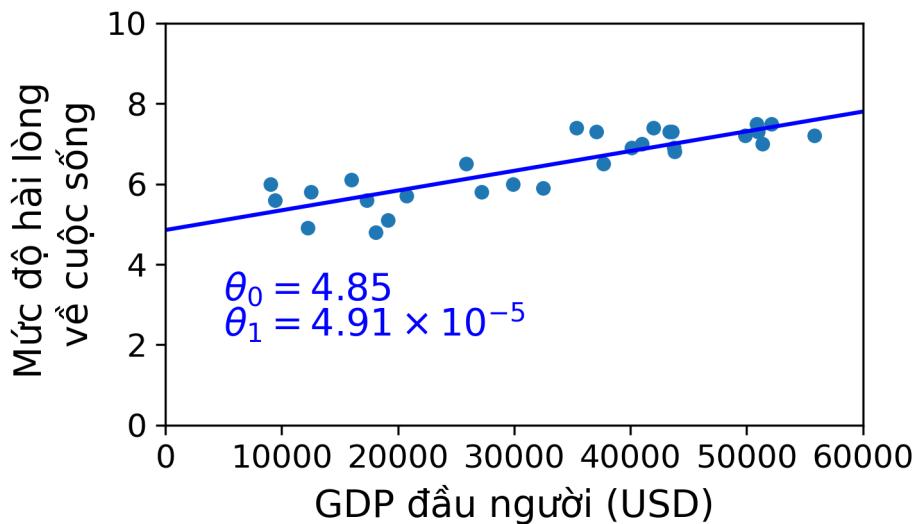
Trước khi có thể sử dụng mô hình, ta cần chọn giá trị cho các tham số θ_0 và θ_1 . Làm sao để biết giá trị nào giúp mô hình hoạt động tốt nhất? Để trả lời câu hỏi này, ta cần chỉ định một phép đo chất lượng. Ta có thể định nghĩa một *hàm lợi ích – utility function* (hoặc *hàm khớp – fitness function*) để đo độ *tốt* của mô hình, hoặc một *hàm chi phí – cost function* để đo độ *tệ* của mô hình đó. Với các bài toán Hồi quy Tuyến tính, ta thường dùng một *hàm chi phí* để đo khoảng cách giữa dự đoán của mô hình và mẫu huấn luyện, với mục tiêu là cực tiểu hóa khoảng cách này.

Đây là lúc thuật toán Hồi quy Tuyến tính phát huy tác dụng: chỉ cần đưa vào các mẫu huấn luyện và thuật toán sẽ tìm các tham số giúp mô hình tuyến tính khớp dữ liệu tốt nhất. Quá trình này được gọi là *huấn luyện* mô hình. Trong trường hợp của ta, thuật toán tìm được các giá trị tham số tối ưu là $\theta_0 = 4.85$ và $\theta_1 = 4.91 \times 10^{-5}$.

Lưu ý

Cùng một từ “mô hình” có thể hiểu là *kiểu mô hình* (như Hồi quy Tuyến tính), *kiến trúc mô hình được định nghĩa hoàn toàn* (*fully-specified model architecture* – như Hồi quy Tuyến tính một đầu vào một đầu ra), hoặc *mô hình đã được huấn luyện cuối cùng* (*final trained model*) và sẵn sàng đưa ra dự đoán (như Hồi quy Tuyến tính có một đầu vào và một đầu ra, với $\theta_0 = 4.85$ và $\theta_1 = 4.91 \times 10^{-5}$). Việc lựa chọn mô hình bao gồm chọn kiểu mô hình và định nghĩa toàn bộ kiến trúc của nó. Huấn luyện mô hình là chạy một thuật toán để tìm tham số mô hình sao cho dữ liệu được khớp tốt nhất (và hy vọng có thể dự đoán tốt trên dữ liệu mới).

Giờ mô hình đã khớp tốt nhất có thể với dữ liệu huấn luyện (với một mô hình tuyến tính), như có thể thấy trong [Hình 1.19](#).



Hình 1.19. Mô hình tuyến tính khớp dữ liệu huấn luyện tốt nhất

Cuối cùng thì mô hình đã sẵn sàng để đưa ra dự đoán. Ví dụ, giả sử ta muốn biết mức độ hạnh phúc của người Cộng hoà Síp, nhưng dữ liệu OECD lại không có câu trả lời. Rất may mắn, mô hình có thể đưa ra một dự đoán tốt: ta tra cứu GDP đầu người của Cộng hoà Síp và biết con số đó là 22,587, rồi áp dụng mô hình và tính được mức độ hài lòng về cuộc sống nằm đâu đó quanh mức $4.85 + 22,587 \times 4.91 \times 10^{-5} = 5.96$.

Để giúp mọi thứ trở nên thú vị hơn, [Ví dụ 1.1](#) chia mã nguồn Python để nạp, chuẩn bị dữ liệu⁶, minh họa trực quan bằng đồ thị phân tán, rồi huấn luyện một mô hình tuyến tính và đưa ra dự đoán.⁷

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model

# Load the data
oecd_bli = pd.read_csv("oecd_bli_2015.csv", thousands=', ')
gdp_per_capita = pd.read_csv("gdp_per_capita.csv", thousands=', ', delimiter='\t',
                             encoding='latin1', na_values="n/a")

# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
plt.figure()
plt.plot(X, y, "o")
plt.show()
```

⁶ Định nghĩa hàm `prepare_country_stats()` không được đưa ra ở đây (hãy xem Jupyter Notebook của chương này để biết chi tiết). Đó chỉ là một vài dòng mã pandas để gộp dữ liệu mức độ hài lòng từ OECD với dữ liệu GDP đầu người từ IMF.

⁷ Nếu bạn chưa hiểu hết toàn bộ đoạn mã thì cũng không sao, chúng tôi sẽ trình bày về Scikit-Learn ở những chương tiếp theo.

```
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select a linear model
model = sklearn.linear_model.LinearRegression()

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[22587]] # Cyprus's GDP per capita
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```

Ví dụ 1.1. Huấn luyện và chạy mô hình tuyến tính bằng Scikit-Learn

Ghi chú

Nếu thay vào đó ta sử dụng thuật toán học dựa trên mẫu, ta sẽ nhận thấy rằng Slovenia có GDP đầu người gần giống Cộng hoà Síp nhất (\$20,732), và vì dữ liệu OECD cho biết người Slovenia có mức độ hài lòng về cuộc sống là 5.7, ta có thể dự đoán rằng mức độ hài lòng của người Cộng hoà Síp là 5.7. Hai nước gần nhất tiếp theo là Bồ Đào Nha và Tây Ban Nha với mức độ hài lòng lần lượt là 5.1 và 6.5. Nếu lấy trung bình các giá trị trên, ta thu được 5.77, khá gần với dự đoán của thuật toán học dựa trên mô hình. Thuật toán đơn giản này được gọi là Hồi quy *k*-Điểm Gần nhất – *k*-Nearest Neighbors regression (trong ví dụ này thì $k = 3$).

Việc thay thế mô hình Hồi quy Tuyến tính bằng mô hình k-Điểm Gần nhất trong đoạn mã trên rất đơn giản, bạn chỉ cần thay hai dòng sau:

```
import sklearn.linear_model
model = sklearn.linear_model.LinearRegression()
```

bằng hai dòng dưới đây:

```
import sklearn.neighbors
model = sklearn.neighbors.KNeighborsRegressor(n_neighbors=3)
```

Nếu không có vấn đề gì, mô hình sẽ đưa ra các dự đoán tốt. Còn nếu không, ta có thể cần thêm nhiều thuộc tính hơn (tỉ lệ có việc làm, sức khoẻ, ô nhiễm không khí, v.v.), thu thập thêm hoặc cải thiện chất lượng dữ liệu huấn luyện, hoặc có thể chọn một mô hình mạnh hơn (như Hồi quy Đa thức).

Tóm lại, ta đã:

- Nghiên cứu dữ liệu.
- Lựa chọn mô hình.
- Huấn luyện mô hình trên tập dữ liệu huấn luyện (tức sử dụng thuật toán học để tìm kiếm các tham số mô hình sao cho hàm chi phí đạt giá trị nhỏ nhất).
- Và cuối cùng, áp dụng mô hình để đưa ra dự đoán trên dữ liệu mới (việc này được gọi là *suy luận – inference*), với hy vọng rằng mô hình này sẽ khái quát tốt.

Có thể nói đây là quy trình của một dự án Học Máy điển hình. Trong [Chương 2](#) bạn sẽ được trải nghiệm thực tế với một dự án từ đầu tới cuối.

Cho tới giờ, chúng ta đã đề cập đến khá nhiều kiến thức nền tảng: giờ đây bạn đã biết Học Máy thật sự là gì, tại sao nó lại hữu ích, những loại hệ thống Học Máy phổ biến nhất, và quy trình làm việc của một dự án điển hình. Tiếp theo hãy xem xét những vấn đề có thể xảy ra trong quá trình học, gây ảnh hưởng đến độ chính xác của các dự đoán.

Những Thách thức Chính của Học Máy

Nói ngắn gọn, vì nhiệm vụ chính của ta là chọn và huấn luyện một thuật toán trên một tập dữ liệu, hai vấn đề có thể xảy ra là “thuật toán tệ” và “dữ liệu xấu”. Hãy bắt đầu với các ví dụ của dữ liệu xấu.

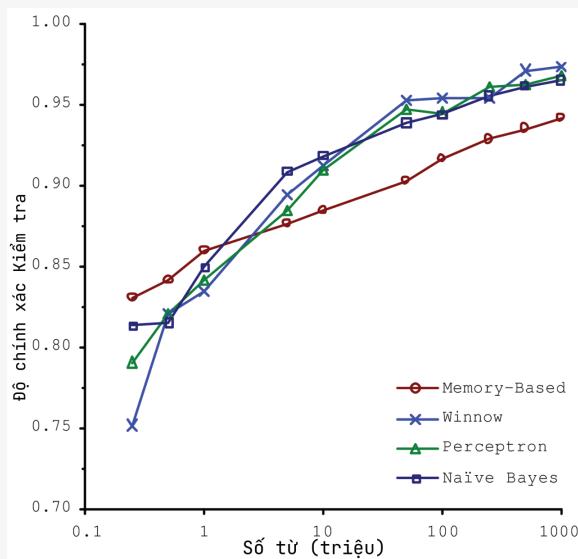
Không đủ Dữ liệu Huấn luyện

Để dạy một đứa trẻ quả táo là gì, ta chỉ cần chỉ vào quả táo và nói “quả táo” (có thể cần lặp lại quy trình này nhiều lần). Nay giờ đứa trẻ có thể nhận ra quả táo với đủ loại màu sắc và hình dạng. Quả thật xuất sắc.

Học Máy thì vẫn chưa đạt đến trình độ đó. Hầu hết các thuật toán Học Máy cần rất nhiều dữ liệu để có thể hoạt động hiệu quả. Ngay cả với những bài toán đơn giản, ta cũng cần đến hàng nghìn mẫu, và với những bài toán phức tạp như nhận diện ảnh hoặc giọng nói thì có thể lên đến hàng triệu mẫu (trừ khi ta có thể tận dụng một mô hình có sẵn).

Sự Hiệu quả Khó lý giải của Dữ liệu

Trong một [bài báo nổi tiếng](#) được xuất bản vào năm 2001, hai nhà nghiên cứu của Microsoft là Michele Banko và Eric Brill đã chỉ ra rằng các thuật toán Học Máy rất khác nhau, bao gồm cả những thuật toán khá đơn giản, hoạt động tốt gần như ngang nhau trong cùng một bài toán phức tạp là khử nhập nhằng ngôn ngữ tự nhiên (*natural language disambiguation*)⁸ một khi chúng được cung cấp đủ dữ liệu (có thể thấy trong [Hình 1.20](#)).



Hình 1.20. Tầm quan trọng của dữ liệu so với thuật toán⁹

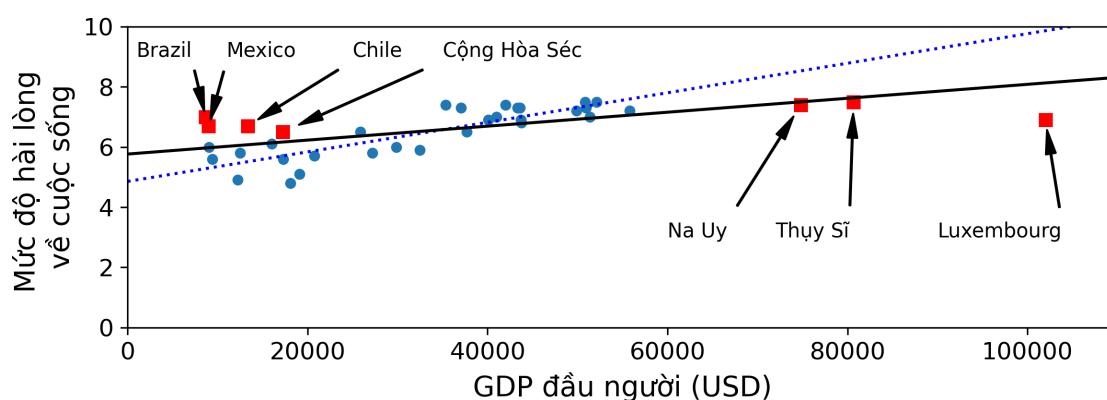
Theo lời của các tác giả, “những kết quả này cho thấy rằng chúng ta có thể cần xem xét lại sự đánh đổi giữa việc dành thời gian và tiền bạc để phát triển thuật toán và tập trung làm giàu kho ngữ liệu.”⁸

Ý tưởng về việc dữ liệu quan trọng hơn thuật toán trong các bài toán phức tạp đã được phổ biến hơn nữa bởi Peter Norvig và cộng sự trong bài báo với tiêu đề “The Unreasonable Effectiveness of Data”, xuất bản vào năm 2009.¹⁰ Tuy nhiên, cần lưu ý rằng các tập dữ liệu vừa và nhỏ vẫn rất phổ biến, và không phải lúc nào cũng có thể kiểm thêm dữ liệu huấn luyện một cách dễ dàng hoặc ít tốn kém. Vậy nên đừng vội bỏ rơi các thuật toán.

Dữ liệu Huấn luyện Không mang tính Đại diện

Để khái quát hóa tốt, điều quan trọng là dữ liệu huấn luyện phải có tính đại diện cho các trường hợp mới mà ta muốn khái quát hóa. Điều này đúng cho cả phương pháp học dựa trên mẫu hay học dựa trên mô hình.

Ví dụ, tập hợp các quốc gia mà lúc trước ta đã sử dụng để huấn luyện mô hình tuyến tính không có tính đại diện hoàn toàn, vì một vài quốc gia vẫn còn thiếu. [Hình 1.21](#) minh họa dữ liệu sau khi các quốc gia còn thiếu đó được bổ sung.



Hình 1.21. Một mẫu huấn luyện có tính đại diện hơn

Nếu huấn luyện một mô hình tuyến tính trên tập dữ liệu mới này, ta sẽ thu được đường liền, trong khi mô hình cũ được biểu diễn bởi đường chấm. Có thể thấy rằng việc thêm một vài quốc gia bị thiếu không chỉ thay đổi đáng kể mô hình, mà còn giúp ta nhận ra rằng một mô hình tuyến tính đơn giản như vậy sẽ không bao giờ có thể hoạt động tốt. Có vẻ như các quốc gia rất giàu có không hạnh phúc hơn các quốc gia có thu nhập thấp (thực chất họ còn có vẻ kém hạnh phúc hơn), và ngược lại, một số quốc gia nghèo lại có vẻ hạnh phúc hơn nhiều quốc gia giàu có.

Bằng cách huấn luyện trên một tập dữ liệu không có tính đại diện, các dự đoán của mô hình khó có thể chính xác, đặc biệt là đối với các nước rất nghèo và rất giàu.

Điều quan trọng ở đây là sử dụng tập huấn luyện có tính đại diện cho các trường hợp mà ta muốn khái quát hóa. Điều này nói dễ hơn là: nếu lượng mẫu quá nhỏ, ta sẽ có *nhiều do lây*

⁸ Ví dụ như việc lựa chọn giữa các từ “to,” “two,” hay “too,” tùy thuộc vào ngữ cảnh.

⁹ Hình ảnh được sao chép với sự cho phép của Michele Banko và Eric Brill, “Scaling to Very Large Corpora for Natural Language Disambiguation,” *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (2001): 26–33.

¹⁰ Peter Norvig và cộng sự, “The Unreasonable Effectiveness of Data,” *IEEE Intelligent Systems* 24, no. 2 (2009): 8–12.

mẫu (*sampling noise* – dữ liệu không mang tính đại diện do sự ngẫu nhiên), nhưng kể cả lượng mẫu lớn cũng có thể không có tính đại diện bởi sai sót trong phương pháp lấy mẫu. Hiện tượng thứ hai được gọi là *thiên kiến lấy mẫu* (*sampling bias*).

Các ví dụ về Thiên kiến lấy Mẫu

Có lẽ ví dụ nổi tiếng nhất về thiên kiến lấy mẫu xảy ra trong cuộc bầu cử tổng thống Mỹ năm 1936 giữa Landon và Roosevelt: tờ *Literary Digest* đã tiến hành một cuộc thăm dò khổng lồ bằng cách gửi thư cho khoảng 10 triệu người. Họ nhận được 2.4 triệu hồi âm và dựa vào đó để dự đoán với độ tin cậy cao rằng Landon sẽ nhận được 57% số phiếu bầu. Tuy nhiên, Roosevelt đã chiến thắng với 62% phiếu bầu. Lỗi hỏng trong phương pháp lấy mẫu của *Literary Digest* là:

- Đầu tiên, để có được địa chỉ cho việc gửi phiếu thăm dò, tờ *Literary Digest* đã sử dụng danh bạ điện thoại, danh sách người đăng ký tạp chí, danh sách thành viên câu lạc bộ và những nguồn tương tự. Tất cả những danh sách này đều thiên về tầng lớp giàu có, những người khả năng cao sẽ bầu cho Đảng Cộng Hòa (mà Landon là người đại diện).
- Thứ hai, chỉ có ít hơn 25% số người được hỏi đã trả lời. Điều này lại gây ra thiên kiến lấy mẫu bằng cách loại trừ những người không quan tâm nhiều đến chính trị, những người không thích tờ *Literary Digest*, cũng như nhiều nhóm quan trọng khác. Đây là một dạng đặc biệt của thiên kiến lấy mẫu, thường được gọi là *thiên kiến không phản hồi* (*nonresponse bias*).

Một ví dụ khác: giả sử ta muốn xây dựng một hệ thống nhận diện các video nhạc funk. Một cách để xây dựng tập huấn luyện là tìm kiếm từ khóa “nhạc funk” trên Youtube và sử dụng các video thu được. Tuy nhiên, điều này giả định rằng công cụ tìm kiếm của Youtube trả về một tập hợp các video đại diện cho tất cả các video nhạc funk tồn tại trên nền tảng Youtube. Trên thực tế, kết quả tìm kiếm có thể thiên về những nghệ sĩ nổi tiếng (nếu bạn sống ở Brazil, đa số kết quả nhận được là các video “funk carioca”, và chúng nghe hoàn toàn khác so với James Brown). Mặt khác, ta còn có thể làm gì hơn để thu được một tập huấn luyện lớn?

Dữ liệu Kém Chất lượng

Tất nhiên, nếu tập huấn luyện của bạn chứa đầy lỗi, điểm ngoại lai và nhiễu (chẳng hạn như do sai số đo lường), hệ thống sẽ gặp nhiều khó khăn trong việc phát hiện các khuôn mẫu ẩn, và ít có khả năng hoạt động tốt. Việc dành thời gian làm sạch dữ liệu huấn luyện thường rất cần thiết. Sự thật là đa số các nhà khoa học dữ liệu dành phần lớn thời gian của họ chỉ để làm việc đó. Những ví dụ sau đây là các trường hợp mà ta cần làm sạch dữ liệu huấn luyện:

- Nếu một số mẫu rõ ràng là ngoại lai, ta có thể đơn thuần loại bỏ chúng hoặc sửa lỗi một cách thủ công.
- Với những mẫu bị thiếu một số đặc trưng (ví dụ: 5% khách hàng của bạn không cung cấp tuổi của họ), ta cần quyết định giữa việc bỏ qua luôn thuộc tính này, bỏ qua những mẫu bị thiếu, điền vào các giá trị còn thiếu (ví dụ, bằng tuổi trung vị), hoặc huấn luyện hai mô hình: một mô hình với đặc trưng đó và một mô hình thì không.

Các Đặc trưng Không liên quan

Có một câu nói là “rác vào, rác ra”. Hệ thống chỉ có thể học được nếu dữ liệu huấn luyện chứa đủ các đặc trưng liên quan và không quá nhiều các đặc trưng không liên quan. Một phần quan trọng dẫn đến một dự án Học Máy thành công là xây dựng được một bộ đặc trưng tốt để huấn luyện. Quy trình này được gọi là *thiết kế đặc trưng* (*feature engineering*) và bao gồm các bước sau:

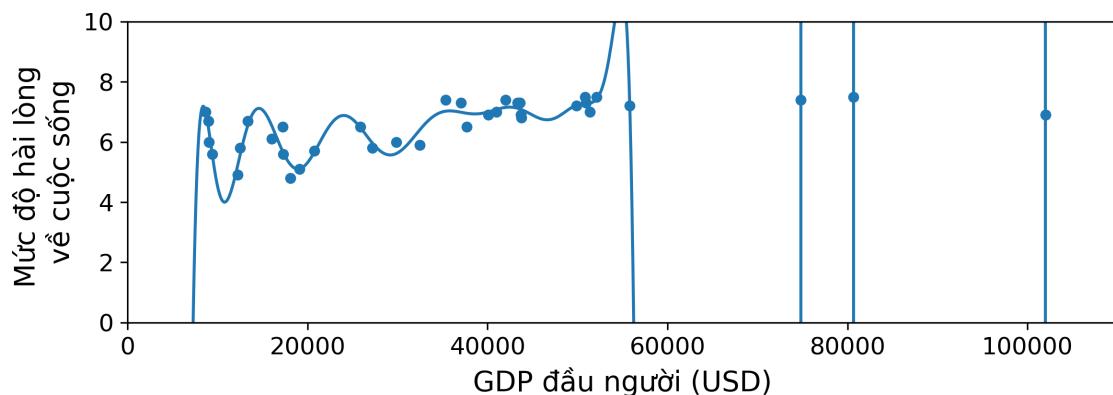
- *Lựa chọn đặc trưng* (lựa chọn những đặc trưng hữu ích nhất từ các đặc trưng sẵn có để huấn luyện)
- *Trích xuất đặc trưng* (kết hợp các đặc trưng sẵn có để tạo ra một đặc trưng hữu ích hơn, và như đã đề cập phía trên, các thuật toán giảm chiều có thể có ích)
- Tạo ra các đặc trưng mới bằng cách thu thập thêm dữ liệu

Ta đã xem xét nhiều ví dụ về dữ liệu xấu, giờ hãy cùng xem xét các ví dụ về thuật toán tệ.

Quá khớp Dữ liệu Huấn luyện

Giả sử bạn đang đi du lịch nước ngoài tham quan và bị tài xế taxi “chặt chém”. Bạn có thể sẽ nghĩ rằng *tất cả* tài xế taxi ở đất nước đó đều là kẻ cắp. Con người thường có thói quen “vơ đũa cả nắm”, và thật không may máy móc cũng có thể rơi vào cái bẫy tương tự nếu chúng ta không cẩn thận. Trong Học Máy, khái niệm này được gọi là *quá khớp* (*overfitting*), có nghĩa là mô hình hoạt động tốt trên dữ liệu huấn luyện nhưng lại không có tính khái quát hóa.

Hình 1.22 minh họa một mô hình đa thức bậc cao dự đoán mức độ hài lòng về cuộc sống đang quá khớp trên dữ liệu huấn luyện. Mặc dù nó hoạt động trên dữ liệu huấn luyện tốt hơn nhiều so với mô hình tuyến tính đơn giản, bạn có thật sự tin vào những dự đoán của nó không?



Hình 1.22. Quá khớp dữ liệu huấn luyện

Những mô hình phức tạp như mạng nơ-ron sâu có thể phát hiện được những quy luật không quá rõ ràng trong dữ liệu, nhưng nếu tập huấn luyện chứa nhiều, hoặc nếu kích thước của tập này quá nhỏ (gây ra nhiều do lấy mẫu), thì rất có thể mô hình sẽ lại phát hiện cả khuôn mẫu trong nhiều. Rõ ràng những khuôn mẫu này sẽ không khái quát hóa cho các mẫu dữ liệu mới. Ví dụ, giả sử bạn huấn luyện mô hình dự đoán mức độ hài lòng về cuộc sống với nhiều đặc trưng hơn, bao gồm cả những đặc trưng không hữu ích như tên quốc gia. Trong trường hợp này, một mô hình phức tạp có thể phát hiện những quy luật như: tất cả các quốc gia trong tập huấn luyện

với tên tiếng Anh chứa chữ cái *w* đều có mức độ hài lòng về cuộc sống lớn hơn 7: New Zealand (7.3), Norway (Na Uy – 7.4), Sweden (Thụy Điển – 7.2), và Switzerland (Thụy Sĩ – 7.5). Bạn có chắc rằng “quy luật hài lòng *w*” này áp dụng cho Rwanda hoặc Zimbabwe không? Rõ ràng khuôn mẫu này chỉ xảy ra trong dữ liệu huấn luyện một cách tình cờ, nhưng mô hình không có cách nào để phân biệt được nó là thật hay chỉ đơn giản là nhiễu trong dữ liệu.

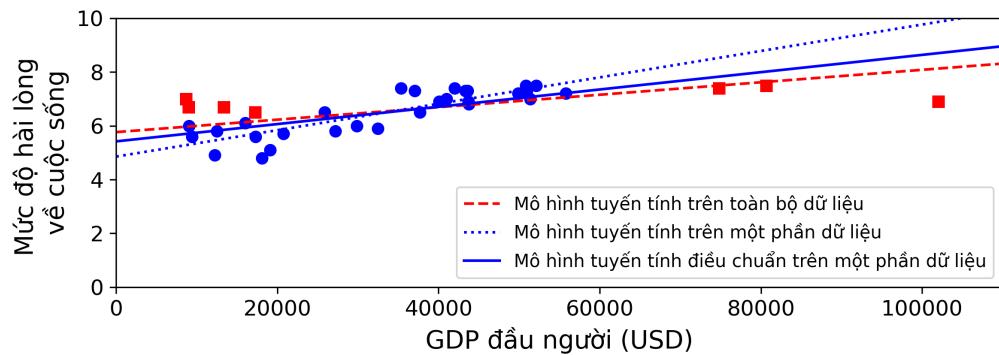
Lưu ý

Quá khớp xảy ra khi mô hình quá phức tạp so với lượng mẫu và nhiễu của dữ liệu huấn luyện. Đây là những giải pháp để tránh vấn đề này:

- Đơn giản hóa mô hình bằng cách chọn mô hình có ít tham số hơn (ví dụ, một mô hình tuyến tính thay vì mô hình đa thức bậc cao), giảm số lượng thuộc tính trong dữ liệu huấn luyện, hoặc ràng buộc mô hình.
 - Thu thập thêm dữ liệu huấn luyện.
 - Giảm nhiễu trong dữ liệu huấn luyện (ví dụ như chỉnh sửa lỗi sai trong dữ liệu và loại bỏ những mẫu ngoại lai).
-

Việc ràng buộc một mô hình để làm cho nó đơn giản hơn và giảm nguy cơ quá khớp được gọi là *điều chuẩn (regularization)*. Ví dụ, mô hình tuyến tính được định nghĩa ở trên có hai tham số là θ_0 và θ_1 . Chúng cho phép thuật toán học với hai *mức độ tự do (degrees of freedom)* để thích ứng mô hình với dữ liệu huấn luyện: nó có thể điều chỉnh cả chiều cao (θ_0) và độ dốc (θ_1) của đường thẳng. Nếu chúng ta ép $\theta_1 = 0$, thuật toán chỉ còn một mức độ tự do và sẽ gặp khó khăn trong việc khớp dữ liệu: nó chỉ có thể di chuyển đường thẳng lên xuống sao cho càng gần với dữ liệu huấn luyện càng tốt, nên việc huấn luyện sẽ kết thúc gần điểm trung bình. Đây là một mô hình quá đơn giản! Nếu chúng ta cho phép thuật toán điều chỉnh θ_1 nhưng buộc nó có giá trị nhỏ thì thuật toán học sẽ có đâu đó từ một đến hai mức độ tự do. Kết quả là một mô hình đơn giản hơn mô hình có hai mức độ tự do, nhưng lại phức tạp hơn mô hình có một mức độ tự do. Chúng ta muốn tìm sự cân bằng giữa việc khớp tập huấn luyện một cách hoàn hảo và giữ mô hình đủ đơn giản để đảm bảo tính khái quát hóa.

[Hình 1.23](#) biểu diễn ba mô hình. Đường chấm là mô hình ban đầu được huấn luyện chỉ trên các nước được đại diện bởi hình tròn (không bao gồm các nước đại diện bằng hình vuông), đường nét đứt là mô hình thứ hai được huấn luyện với tất cả các nước (hình tròn và hình vuông), và đường nét liền là mô hình được huấn luyện trên cùng dữ liệu với mô hình đầu tiên nhưng có điều chuẩn. Có thể thấy rằng việc điều chuẩn buộc mô hình có độ dốc nhỏ hơn: mô hình này không khớp với dữ liệu huấn luyện (hình tròn) tốt như mô hình đầu tiên, nhưng nó khái quát hóa tốt hơn trên các mẫu dữ liệu mới mà mô hình chưa thấy ở quá trình huấn luyện (hình vuông).



Hình 1.23. Điều chỉnh giảm nguy cơ quá khít

Mức độ điều chỉnh áp dụng trong quá trình học có thể được kiểm soát bởi một *siêu tham số* (*hyperparameter*). Một siêu tham số là một tham số của thuật toán chứ không phải của mô hình. Vì vậy, nó không bị ảnh hưởng bởi quá trình học. Giá trị của siêu tham số cần được đặt trước khi huấn luyện và sẽ được giữ nguyên trong suốt quá trình huấn luyện. Nếu ta đặt siêu tham số điều chỉnh quá cao, ta sẽ nhận được một mô hình gần như nằm ngang (độ dốc gần bằng 0). Khi đó, thuật toán gần như chắc chắn sẽ không quá khớp dữ liệu huấn luyện, nhưng sẽ khó để tìm được một mô hình tốt. Điều chỉnh siêu tham số là một phần quan trọng trong quá trình xây dựng một hệ thống Học Máy (ví dụ chi tiết sẽ được trình bày ở chương tiếp theo).

Dưới khớp Dữ liệu Huấn luyện

Như bạn có thể đoán được, *dưới khớp* (*underfitting*) ngược lại với quá khớp: nó xảy ra khi mô hình quá đơn giản để học được cấu trúc của dữ liệu. Ví dụ, một mô hình tuyến tính dự đoán mức độ hài lòng về cuộc sống sẽ dễ bị dưới khớp, đơn giản vì đời thực luôn phức tạp hơn mô hình. Vì vậy những dự đoán của nó chắc chắn sẽ không chính xác, kể cả với các mẫu huấn luyện.

Đây là những giải pháp chính để giải quyết vấn đề này:

- Chọn một mô hình mạnh hơn, với nhiều tham số hơn.
- Cung cấp đặc trưng tốt hơn cho thuật toán học (thiết kế đặc trưng).
- Giảm ràng buộc lên mô hình (ví dụ như giảm siêu tham số điều chỉnh).

Ôn tập

Cho tới giờ thì bạn đã biết khá nhiều về Học Máy. Tuy nhiên, việc học về quá nhiều khái niệm có thể khiến bạn hơi choáng ngợp, vì vậy hãy cùng lùi lại một chút và nhìn vào bức tranh tổng thể:

- Học Máy xoay quanh việc giúp máy móc thực hiện một số tác vụ tốt hơn bằng cách học từ dữ liệu thay vì phải lập trình các quy luật một cách tưởng минh.
- Có nhiều loại hệ thống ML: có hoặc không giám sát, học theo batch hay học trực tuyến, học dựa trên mẫu hay học dựa trên mô hình.

- Trong một dự án ML, ta thu thập dữ liệu huấn luyện và đưa chúng vào một thuật toán. Nếu thuật toán đó học dựa trên mô hình, nó sẽ điều chỉnh một vài tham số để khớp mô hình trên dữ liệu huấn luyện (tức để dự đoán tốt trên tập huấn luyện). Sau đó, ta hy vọng rằng mô hình cũng có thể dự đoán tốt trên dữ liệu mới. Nếu thuật toán đó học dựa trên mẫu, nó chỉ học thuộc lòng các mẫu dữ liệu và khái quát hóa cho các trường hợp mới bằng cách sử dụng một phép đo độ tương đồng để so sánh mẫu dữ liệu mới với mẫu dữ liệu đã học.
- Hệ thống sẽ hoạt động không hiệu quả nếu lượng dữ liệu huấn luyện quá nhỏ, hoặc nếu dữ liệu không mang tính đại diện, có nhiễu hay chứa các đặc trưng không liên quan (“rác vào, rác ra”). Cuối cùng, mô hình không nên quá đơn giản (gây dưới khớp) hoặc quá phức tạp (gây quá khớp).

Còn một chủ đề quan trọng nữa cần được đề cập: một khi mô hình đã được huấn luyện, ta không chỉ ngồi không và “hy vọng” nó sẽ khái quát cho các trường hợp mới. Ta cần phải đánh giá và tinh chỉnh mô hình nếu cần thiết. Hãy cùng xem xét cách để làm điều đó.

Kiểm tra và Đánh giá

Cách duy nhất để biết một mô hình khái quát hóa tốt đến đâu là thử nó với những mẫu dữ liệu mới. Một cách để thực hiện việc này là triển khai mô hình rồi giám sát chất lượng của nó. Cách này ổn, nhưng nếu mô hình hoạt động cực kỳ tệ, người dùng sẽ than phiền. Vì thế nên đây không phải là ý tưởng hay nhất.

Một lựa chọn tốt hơn là tách dữ liệu ra thành hai tập: *tập huấn luyện* (*training set*) và *tập kiểm tra* (*test set*). Như có thể đoán được từ cái tên, mô hình được huấn luyện trên tập huấn luyện và được kiểm tra trên tập kiểm tra. Tỉ lệ lỗi trên dữ liệu mới được gọi là *sai số khái quát – generalization error* (hoặc *sai số ngoài mẫu – out-of-sample error*) và bằng cách đánh giá mô hình trên tập kiểm tra, chúng ta sẽ ước lượng được giá trị sai số này. Giá trị này cho biết mô hình sẽ hoạt động tốt đến đâu trên dữ liệu mới.

Nếu sai số huấn luyện thấp (nghĩa là mô hình có ít lỗi trên tập huấn luyện) nhưng sai số khái quát lại cao, mô hình đã quá khớp dữ liệu huấn luyện.

Mẹo

Thông thường 80% dữ liệu được dùng để huấn luyện và 20% được *giữ lại* để kiểm tra. Tuy nhiên, việc này tuỳ thuộc vào kích cỡ của tập dữ liệu: nếu tập dữ liệu chứa 10 triệu mẫu, việc giữ lại 1% nghĩa là tập kiểm tra sẽ chứa 100,000 mẫu. Số lượng này có thể đã quá đủ để ước lượng tốt sai số khái quát.

Tinh Chỉnh Siêu Tham Số và Lựa Chọn Mô Hình

Việc đánh giá một mô hình khá đơn giản: chỉ cần dựa vào tập kiểm tra. Nhưng giả sử bạn đang lưỡng lự giữa hai loại mô hình (ví dụ, giữa một mô hình tuyến tính và một mô hình đa thức). Làm sao để đưa ra quyết định? Một lựa chọn là huấn luyện cả hai và so sánh khả năng khái quát của chúng trên tập kiểm tra.

Giờ hãy giả sử mô hình tuyến tính khái quát tốt hơn nhưng bạn muốn sử dụng thêm điều chuẩn để tránh quá khớp. Câu hỏi được đặt ra là ta chọn giá trị siêu tham số điều chuẩn như thế nào? Một giải pháp là huấn luyện 100 mô hình sử dụng 100 giá trị khác nhau cho siêu tham số này. Giả sử bạn đã tìm được siêu tham số tốt nhất sao cho mô hình có sai số khái quát thấp nhất,

chỉ ở mức 5%. Bạn triển khai mô hình này nhưng tiếc rằng nó không tốt như mong đợi với sai số ở mức 15%. Chuyện gì đã xảy ra?

Vấn đề là bạn đã tính toán sai số khái quát nhiều lần trên tập kiểm tra và cố gắng tìm các siêu tham số dẫn đến chất lượng tốt nhất *trên tập dữ liệu cố định* đó. Kết quả là mô hình khó có thể đạt được chất lượng tốt như mong đợi trên dữ liệu mới.

Giải pháp phổ biến cho vấn đề này là *kiểm định giữ lại* (*holdout validation*): ta đơn thuần giữ lại một phần của tập huấn luyện để đánh giá nhiều mô hình và chọn cái tốt nhất. Tập dữ liệu được giữ lại đó có tên là *tập kiểm định* (*validation set* – hoặc đôi khi được gọi là *tập phát triển – development set/dev set*). Cụ thể, ta huấn luyện nhiều mô hình với các siêu tham số khác nhau trên tập huấn luyện nhỏ hơn (do đã lấy ra tập kiểm định) và chọn mô hình hoạt động tốt nhất trên tập kiểm định. Sau khi hoàn tất việc kiểm định trên tập giữ lại, ta huấn luyện mô hình tốt nhất trên toàn bộ tập huấn luyện (bao gồm cả tập kiểm định) để thu được mô hình cuối. Cuối cùng, ta đánh giá mô hình này trên tập kiểm tra để ước lượng sai số khái quát.

Giải pháp này thường cho kết quả tốt. Tuy nhiên, nếu tập kiểm định quá nhỏ thì việc đánh giá mô hình sẽ không chính xác, và ta có thể vô tình chọn phải mô hình không tối ưu. Ngược lại, nếu tập kiểm định quá lớn, phần dữ liệu còn lại để huấn luyện sẽ nhỏ hơn rất nhiều so với tập huấn luyện đầy đủ. Vì sao điều này lại không tốt? Bởi vì mô hình cuối cùng sẽ được huấn luyện trên toàn bộ tập huấn luyện, việc so sánh giữa các mô hình được huấn luyện trên tập huấn luyện nhỏ hơn hẳn là không hợp lý. Để giải quyết vấn đề này, ta có thể dùng *kiểm định chéo* (*cross-validation*) với nhiều tập kiểm định nhỏ. Mỗi mô hình sẽ được đánh giá một lần trên mỗi tập kiểm định sau khi nó được huấn luyện trên phần dữ liệu còn lại. Bằng cách lấy trung bình các lần đánh giá, ta sẽ có một thước đo chính xác hơn nhiều cho chất lượng của mô hình. Tuy nhiên, phương pháp này có một hạn chế: thời gian huấn luyện sẽ tăng theo số tập kiểm định.

Dữ liệu không tương đồng

Trong một số trường hợp, ta có thể dễ dàng thu được một lượng lớn dữ liệu để huấn luyện, nhưng có lẽ chúng sẽ không đại diện hoàn toàn cho dữ liệu mà ta sẽ gặp phải trong thực tế. Giả sử bạn muốn tạo một ứng dụng điện thoại để chụp ảnh hoa và tự động xác định loài hoa. Bạn có thể dễ dàng tải xuống hàng triệu bức ảnh về hoa trên mạng, nhưng chúng sẽ không phải là đại diện hoàn hảo cho những bức ảnh sẽ được chụp bởi ứng dụng trên thiết bị di động. Có thể chỉ có 10,000 ảnh được chụp bằng ứng dụng đó. Trong trường hợp này, quy tắc quan trọng cần nhớ là tập kiểm định và tập kiểm tra phải mang tính đại diện cho dữ liệu trong thực tế càng nhiều càng tốt. Vì thế, hai tập này chỉ nên chứa các ảnh được chụp bằng ứng dụng: bạn có thể xáo trộn chúng và chia một nửa cho tập kiểm định, một nửa cho tập kiểm tra (đảm bảo rằng giữa hai tập không có mẫu nào bị trùng hoặc giống nhau). Nhưng sau khi huấn luyện mô hình bằng ảnh trên mạng, nếu bạn thấy chất lượng của mô hình trên tập kiểm định không tốt, bạn sẽ không biết được nguyên nhân là do mô hình đã quá khớp dữ liệu huấn luyện, hay chỉ là do sự không tương đồng giữa ảnh trên mạng và ảnh được chụp bằng ứng dụng di động. Một giải pháp là giữ lại một vài ảnh huấn luyện (ảnh trên mạng) cho một tập khác mà Andrew Ng gọi là *tập huấn luyện - phát triển* (*train-dev set*). Sau khi mô hình được huấn luyện (trên tập huấn luyện, *không phải* trên tập huấn luyện-phát triển), ta có thể đánh giá nó trên tập huấn luyện - phát triển. Nếu chất lượng tốt, ta biết được rằng mô hình không quá khớp tập huấn luyện. Nếu sau đó mô hình đạt chất lượng kém trên tập kiểm định, vấn đề chắc chắn đến từ việc dữ liệu không tương đồng. Bạn có thể thử giải quyết vấn đề này bằng cách tách xử lý ảnh tải trên mạng để làm cho chúng giống với ảnh được chụp bởi ứng dụng và sau đó huấn luyện lại mô hình. Ngược lại, nếu mô hình hoạt động kém trên tập huấn luyện - phát triển, mô hình chắc hẳn đã quá khớp dữ liệu huấn luyện, vì vậy bạn nên đơn giản hóa hoặc điều chỉnh mô hình, thu thập thêm và làm sạch dữ liệu huấn luyện.

Định Lý Không Có Bữa Trưa Miễn Phí

Mô hình là một phiên bản được đơn giản hóa của các mẫu. Đơn giản hóa ở đây đồng nghĩa với việc loại bỏ các chi tiết thừa không có khả năng khái quát hóa cho các trường hợp mới. Để quyết định phần dữ liệu nào cần loại bỏ và phần dữ liệu nào cần giữ lại, bạn cần phải đặt ra các *giả định*. Ví dụ, một mô hình tuyến tính đưa ra giả định rằng dữ liệu có bản chất tuyến tính và khoảng cách giữa các mẫu và đường thẳng chỉ là nhiễu, và ta có thể bỏ qua khoảng cách đó mà không ảnh hưởng gì.

Trong một bài báo nổi tiếng năm 1996,¹¹ David Wolpert đã chứng minh rằng nếu bạn không đặt bất kỳ giả định nào về dữ liệu thì không có lý do gì để nói rằng mô hình này tốt hơn mô hình kia. Điều này được gọi là định lý *Không có Bữa trưa Miễn phí* (*No Free Lunch – NFL*). Với một số tập dữ liệu, mô hình tốt nhất là mô hình tuyến tính, trong khi với các tập dữ liệu khác, mô hình tốt nhất là một mạng nơ-ron. Không có mô hình nào được *tiên nghiệm* là sẽ hoạt động tốt hơn (do đó mà định lý có tên như trên). Cách duy nhất để biết chắc rằng mô hình nào tốt nhất là đánh giá tất cả các mô hình. Vì điều này là bất khả thi, trong thực tế, ta cần đưa ra một số giả định hợp lý về dữ liệu và chỉ đánh giá một số mô hình phù hợp. Ví dụ: đối với các tác vụ đơn giản, ta có thể đánh giá các mô hình tuyến tính với nhiều mức điều chỉnh khác nhau. Ngược lại, đối với một bài toán phức tạp, ta có thể đánh giá các mạng nơ-ron khác nhau.

Bài tập

Trong chương này, chúng tôi đã trình bày một số khái niệm quan trọng nhất trong Học Máy. Trong các chương tiếp theo, chúng ta sẽ đi sâu hơn và viết mã nhiều hơn, nhưng trước khi đi tiếp, hãy đảm bảo bạn biết câu trả lời cho các câu hỏi sau:

1. Định nghĩa của Học Máy là gì?
2. Bạn có thể liệt kê bốn loại bài toán mà Học Máy giải quyết tốt không?
3. Tập huấn luyện đã gán nhãn là gì?
4. Hai tác vụ học có giám sát phổ biến nhất là gì?
5. Bạn có thể liệt kê bốn tác vụ học không giám sát phổ biến không?
6. Bạn sẽ sử dụng loại thuật toán Học Máy nào để cho phép rô bốt di lại trong các địa hình chưa biết?
7. Bạn sẽ sử dụng loại thuật toán nào để phân nhóm khách hàng thành nhiều nhóm?
8. Bạn sẽ đặt bài toán phát hiện thư rác là bài toán học có giám sát hay học không giám sát?
9. Hệ thống học trực tuyến là gì?
10. Thế nào là học ngoài bộ nhớ chính?
11. Loại thuật toán nào dựa vào phép đo độ tương đồng để đưa ra dự đoán?
12. Sự khác biệt giữa tham số mô hình và siêu tham số của thuật toán là gì?

¹¹ David Wolpert, “The Lack of A Priori Distinctions Between Learning Algorithms,” *Neural Computation* 8, no. 7 (1996): 1341–1390.

CHƯƠNG 1. TOÀN CẢNH HỌC MÁY

13. Thuật toán học dựa trên mô hình đang tìm kiếm thứ gì? Chiến lược phổ biến nhất mà chúng sử dụng để thành công là gì? Chúng đưa ra dự đoán như thế nào?
14. Bạn có thể liệt kê bốn thách thức chính trong Học Máy không?
15. Nếu mô hình của bạn hoạt động tốt trên dữ liệu huấn luyện nhưng lại khai quát kém đối với dữ liệu mới, điều gì đang xảy ra? Bạn có thể liệt kê ba giải pháp khả thi cho vấn đề này không?
16. Tập kiểm tra là gì và tại sao bạn lại muốn sử dụng nó?
17. Mục đích của tập kiểm định là gì?
18. Tập huấn luyện - phát triển là gì, khi nào bạn cần sử dụng và làm thế nào để sử dụng nó?
19. Vấn đề gì có thể xảy ra nếu bạn tinh chỉnh siêu tham số bằng tập kiểm tra?

Lời giải cho những bài tập trên nằm ở [Phụ lục A](#).