Mục lục

Phần I

Kiến thức cơ bản về Toán ứng dụng và Học máy

Chương 4

Phương pháp tính

Các thuật toán học máy thường đòi hỏi một lượng lớn tính toán số. Điều này thường ám chỉ các thuật toán giải quyết các bài toán toán học bằng các phương pháp cập nhật các ước lượng của nghiệm thông qua một quá trình lặp, thay vì suy ra một công thức giải tích cho lời giải đúng. Các thao tác phổ biến bao gồm tối ưu hóa (tìm giá trị của biến số làm cực tiểu hoặc cực đại một hàm số) và giải các hệ phương trình tuyến tính. Ngay cả việc đánh giá một hàm số trên máy tính kỹ thuật số cũng có thể khó khăn khi hàm số đó bao gồm các số thực, vốn không thể được biểu diễn chính xác bằng một lượng bộ nhớ hữu hạn.

4.1 Tràn số và hạ số

Khó khăn cơ bản trong việc thực hiện đại lượng liên tục trong toán học trên máy tính số là ta cần biểu diễn vô số số thực bằng một số lượng mẫu bit hữu hạn. Điều này có nghĩa là đối với hầu hết các số thực, ta sẽ gặp phải một số sai số xấp xỉ khi biểu diễn số đó trên máy tính. Trong nhiều trường hợp, đây chỉ là lỗi làm tròn. Lỗi làm tròn có thể gây ra vấn đề, đặc biệt khi nó tích lũy qua nhiều phép toán và có thể khiến các thuật toán hoạt động tốt trong lý thuyết nhưng thất bại trong thực tế nếu chúng không được thiết kế để giảm thiểu sự tích lũy của lỗi làm tròn.

Một dạng lỗi làm tròn đặc biệt nghiêm trọng là **hạ số** (underflow). Hạ số xảy ra khi các số gần bằng 0 bị làm tròn về 0. Nhiều hàm số có hành vi khác nhau đáng kể khi đối số của chúng bằng 0 so với khi là một số dương nhỏ. Ví dụ, ta thường muốn tránh chia cho 0 (một số môi trường phần mềm sẽ đưa ra ngoại lệ khi điều này xảy ra, trong khi những môi trường khác sẽ trả về một kết quả với giá trị giữ chỗ "không phải là số" – not-a-number) hoặc lấy logarit của 0 (điều này thường

được coi là $-\infty$, và nếu sử dụng cho nhiều phép toán số học tiếp theo, nó sẽ trở thành "không phải là số").

Một dạng lỗi số học gây hại cao khác là **tràn số** (overflow). Tràn số xảy ra khi các số có giá trị lớn bị xấp xỉ thành ∞ hoặc $-\infty$. Các phép toán số học tiếp theo thường sẽ biến những giá trị vô hạn này thành các giá trị "không phải là số".

Một ví dụ về một hàm cần được ổn định để chống lại tràn số và hạ lưu số là hàm softmax. Hàm softmax thường được sử dụng để dự đoán các xác suất liên quan đến phân phối Bernoulli bội. Hàm softmax được định nghĩa là:

$$\text{softmax}(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}. \tag{4.1}$$

Xét trường hợp khi tất cả các giá trị x_i đều bằng một hằng số c. Về mặt lý thuyết, ta có thể thấy rằng tất cả các đầu ra của hàm softmax sẽ bằng $\frac{1}{n}$. Tuy nhiên, về mặt tính toán, điều này có thể không xảy ra khi c có giá trị lớn. Nếu c âm và rất nhỏ, thì e^c sẽ bị hạ số, dẫn đến mẫu số của hàm softmax trở thành 0, làm cho kết quả cuối cùng không xác định. Khi c dương và rất lớn, e^c sẽ gặp tràn số, lại khiến cho biểu thức không xác định. Cả hai vấn đề này có thể được giải quyết bằng cách tính softmax $(\boldsymbol{z})$ với $\boldsymbol{z} = \boldsymbol{x} - \max_i x_i$. Một biến đổi đại số đơn giản cho thấy rằng giá trị của hàm softmax không thay đổi khi cộng hoặc trừ một hằng số từ véctơ đầu vào. Việc trừ $\max_i x_i$ đảm bảo rằng đối số lớn nhất của hàm mũ là 0, loại trừ khả năng tràn số. Tương tự, ít nhất một thành phần trong mẫu số có giá trị là 1, loại trừ khả năng hạ số trong mẫu số dẫn đến phép chia cho 0.

Vẫn còn một vấn đề nhỏ. Hạ số trong tử số vẫn có thể khiến cho toàn bộ biểu thức đánh giá về 0. Điều này có nghĩa là nếu ta triển khai hàm log softmax (x) bằng cách chạy chương trình con softmax trước, sau đó truyền kết quả vào hàm log, ta có thể thu được giá trị sai lầm $-\infty$. Thay vào đó, ta phải triển khai một hàm riêng biệt để tính log softmax theo cách ổn định về mặt số học. Hàm log softmax có thể được ổn định bằng cách sử dụng cùng thủ thuật mà ta đã sử dụng để ổn định hàm softmax.

Phần lớn cuốn sách không trình bày chi tiết tất cả các vấn đề số học liên quan đến việc triển khai các thuật toán được mô tả trong cuốn sách này. Những nhà phát triển các thư viện cấp thấp nên lưu ý đến các vấn đề số học khi triển khai các thuật toán học sâu. Hầu hết độc giả của cuốn sách này có thể chỉ cần dựa vào các thư viện cấp thấp cung cấp các triển khai ổn định. Trong một số trường hợp, có thể triển khai một thuật toán mới và để triển khai đó tự động được ổn định. Theano

(*Theano: A CPU and GPU Math Compiler in Python*, Bergstra và các tác giả, 2010, [1]; *Theano: new features and speed improvements*, Bastien và các tác giả, 2012, [2]) là một ví dụ về phần mềm tự động phát hiện và ổn định nhiều biểu thức số học không ổn định thường gặp trong bối cảnh học sâu.

4.2 Điều kiện xấu

Thuật ngữ điều kiện xấu hay tốt đề cập đến mức độ nhanh chóng mà một hàm thay đổi khi các đầu vào của nó bị xáo trộn một chút. Các hàm thay đổi nhanh chóng khi đầu vào bị biến đổi nhẹ có thể gây ra vấn đề trong tính toán khoa học, vì lỗi làm tròn trong đầu vào có thể dẫn đến thay đổi lớn trong đầu ra.

Xét hàm $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}^{-1}\boldsymbol{x}$, trong đó $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ có phân tích giá trị riêng. **Số điều kiện** của nó là

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right| . \tag{4.2}$$

Đó là tỷ số giữa độ lớn của giá trị riêng lớn nhất và nhỏ nhất của ma trận A. Khi số điều kiện này lớn, phép nghịch đảo ma trận trở nên đặc biệt nhạy cảm với lỗi trong đầu vào.

Độ nhạy này là thuộc tính nội tại của ma trận, không phải là kết quả của lỗi làm tròn trong quá trình nghịch đảo ma trận. Các ma trận có điều kiện xấu sẽ khuếch đại các lỗi đã tồn tại sẵn khi ta nhân với ma trận nghịch đảo thực sự. Trong thực tế, lỗi sẽ được khuếch đại thêm bởi các sai số số học trong chính quá trình nghịch đảo.

4.3 Tối ưu dựa trên gradient

Hầu hết các thuật toán học sâu đều liên quan đến việc tối ưu hóa ở một mức độ nào đó. Tối ưu hóa đề cập đến nhiệm vụ tìm cách cực tiểu hoặc cực đại hóa một hàm số $f(\boldsymbol{x})$ bằng cách thay đổi giá trị của $\boldsymbol{x}$. Thông thường, các bài toán tối ưu hóa được diễn đạt dưới dạng cực tiểu hóa $f(\boldsymbol{x})$. Nếu cần cực đại hóa, ta có thể thực hiện bằng cách sử dụng thuật toán cực tiểu hóa cho hàm $-f(\boldsymbol{x})$.

Hàm mà chúng ta muốn cực tiểu hóa hoặc cực đại hóa được gọi là **hàm mục tiêu** hoặc **hàm tiêu chí**. Khi thực hiện cực tiểu hóa, hàm này còn được gọi là

- **hàm chi phí**: tổng chi phí mà mô hình tạo ra khi đưa ra dự đoán; hoặc
- **hàm mất mát**: đo lường sai số cho một dự đoán cụ thể; hoặc

- **hàm lỗi**: đôi khi được dùng như một cách gọi khác của hàm mất mát, đặc biệt là trong ngữ cảnh đánh giá sai số tổng thể của một mô hình.

Các thuật ngữ này đều mô tả quá trình tìm cách làm giảm sự sai lệch giữa dự đoán của mô hình và giá trị thực tế trong quá trình học máy. Trong sách này, các thuật ngữ này được sử dụng thay thế lẫn nhau, mặc dù một số tài liệu trong lĩnh vực học máy có thể gán những ý nghĩa đặc biệt cho một số thuật ngữ này.

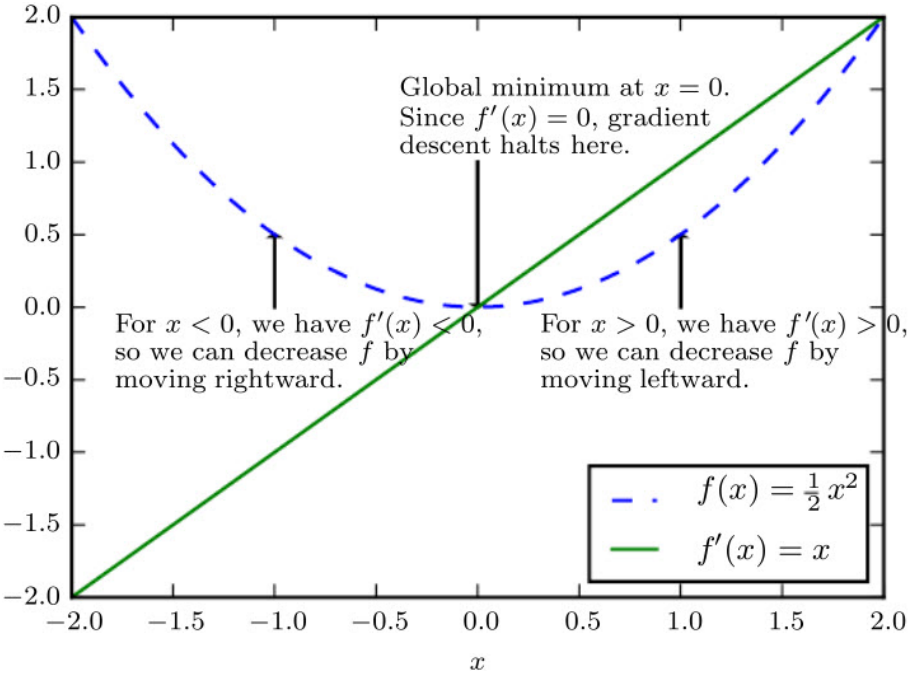
Ta thường ký hiệu giá trị cực tiểu hoặc cực đại của một hàm bằng một dấu chỉ số trên $*$. Ví dụ, ta có thể viết $\boldsymbol{x}^* = \arg\min\ f(\boldsymbol{x})$, để chỉ rằng $\boldsymbol{x}^*$ là giá trị của $\boldsymbol{x}$ tại đó hàm $f(x)$ đạt giá trị nhỏ nhất. Mặc dù giả định rằng người đọc đã quen thuộc với phép tính vi phân, dưới đây là một phần tóm tắt ngắn về cách các khái niệm trong phép tính vi phân liên quan đến bài toán tối ưu hóa.

Giả sử ta có một hàm số $y = f(x)$, trong đó cả x và y đều là các số thực. **Đạo hàm** của hàm này được ký hiệu là $f'(x)$ hoặc $\frac{dy}{dx}$. Đạo hàm $f'(x)$ cho biết độ dốc của hàm $f(x)$ tại điểm x. Nói cách khác, đạo hàm xác định cách thay đổi của đầu ra tương ứng với một thay đổi nhỏ trong đầu vào: $f(x + \varepsilon) \approx f(x) + \varepsilon \cdot f'(x)$.

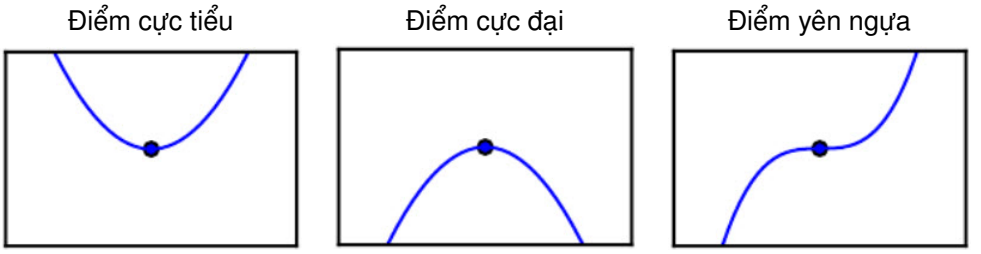
Đạo hàm rất hữu ích trong việc cực tiểu hóa một hàm vì nó cho ta biết cách thay đổi x để làm cải thiện nhỏ giá trị của y. Ví dụ, ta biết rằng $f\left(x - \varepsilon \text{sign}\left(f'(x)\right)\right)$ sẽ nhỏ hơn $f(x)$ với một giá trị ε đủ nhỏ. Do đó, ta có thể giảm giá trị của $f(x)$ bằng cách di chuyển x theo các bước nhỏ theo chiều ngược lại với dấu của đạo hàm. Kỹ thuật này được gọi là **hướng giảm** (*General Method for Solving Systems of Simultaneous Equations.* Cauchy, 1847, [3]). Xem Hình 4.1 để có ví dụ minh họa về kỹ thuật này.

Khi $f'(x) = 0$, đạo hàm không cung cấp thông tin về hướng di chuyển. Các điểm mà $f'(x) = 0$ được gọi là **điểm tới hạn** hoặc **điểm dừng**. **Cực tiểu địa phương** là một điểm tại đó $f(x)$ nhỏ hơn tại tất cả các điểm lân cận, vì vậy không thể giảm $f(x)$ bằng cách thực hiện các bước nhỏ. **Cực đại địa phương** là một điểm tại đó $f(x)$ lớn hơn tại tất cả các điểm lân cận, nên không thể tăng $f(x)$ bằng các bước nhỏ. Một số điểm tới hạn không phải là cực đại hay cực tiểu, được gọi là **điểm yên ngựa**. Xem Hình 4.2 để có ví dụ về từng loại điểm tới hạn.

Một điểm mà $f(x)$ đạt giá trị thấp nhất tuyệt đối được gọi là điểm **cực tiểu toàn cục**. Có thể tồn tại một hoặc nhiều điểm cực tiểu toàn cục của hàm số. Cũng có thể có các điểm cực tiểu địa phương không phải là tối ưu toàn cục. Trong bối cảnh học sâu, ta tối ưu hóa các hàm số có thể có nhiều điểm cực tiểu địa phương không phải nghiệm tối ưu và nhiều điểm yên ngựa được bao quanh bởi các vùng rất phẳng. Tất cả những điều này khiến việc tối ưu hóa trở nên rất khó khăn, đặc biệt trong trường



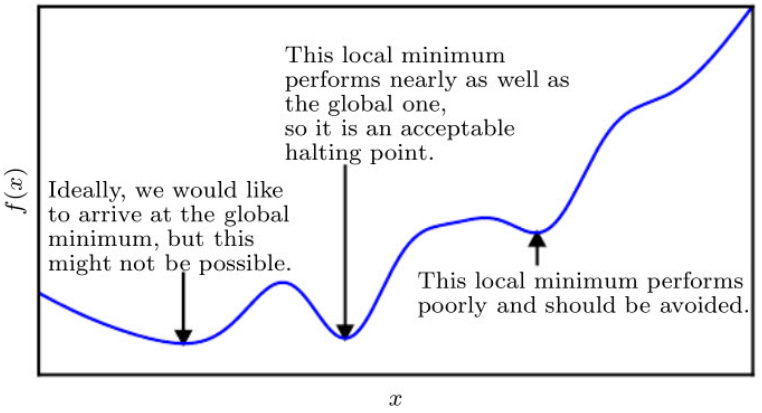
Hình 4.1: Một minh họa cho thấy cách thuật toán hướng giảm sử dụng đạo hàm của hàm số để di chuyển theo hướng giảm dần của hàm số nhằm đạt đến điểm cực tiểu.



Hình 4.2: Ví dụ về ba loại điểm tới hạn trong không gian 1 chiều. Một điểm tới hạn là điểm có độ dốc bằng không. Điểm này có thể là cực tiểu địa phương, nơi mà giá trị thấp hơn các điểm lân cận; cực đại địa phương, nơi mà giá trị cao hơn các điểm lân cận; hoặc điểm yên ngựa, nơi mà các điểm lân cận có giá trị vừa cao hơn vừa thấp hơn điểm đó.

hợp hàm số nhiều biến. Do đó, ta thường chấp nhận việc tìm một giá trị của f đủ thấp, nhưng không nhất thiết phải là giá trị tối thiểu theo định nghĩa. Xem Hình 4.3 để có ví dụ minh họa.

Ta thường cực tiểu hóa các hàm nhiều biến: $f : \mathbb{R}^n \to \mathbb{R}$. Để khái niệm "cực



Hình 4.3: Các thuật toán tối ưu có thể không tìm được điểm cực tiểu toàn cục khi có nhiều điểm cực tiểu địa phương hoặc các bề mặt phẳng. Trong bối cảnh học sâu, ta thường chấp nhận những nghiệm này mặc dù chúng không thực sự là điểm cực tiểu, miễn là chúng tương ứng với các giá trị đủ thấp của hàm chi phí.

tiểu hóa" có ý nghĩa, hàm này phải nhận giá trị chỉ có một thành phần (dạng vô hướng).

Đối với các hàm số nhiều biến, ta phải sử dụng khái niệm **đạo hàm riêng**. Đạo hàm riêng $\frac{\partial}{\partial x_i} f(\boldsymbol{x})$ đo sự thay đổi của f khi chỉ có biến x_i tăng tại điểm x. **Gradient** tổng quát hóa khái niệm đạo hàm thành trường hợp mà đạo hàm được xem như một véctơ: gradient của f là véctơ chứa tất cả các đạo hàm riêng, được ký hiệu là $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. Phần tử thứ i của gradient là đạo hàm riêng của f theo x_i. Trong không gian nhiều chiều, các điểm tới hạn là những điểm mà mọi phần tử của gradient đều bằng không.

Đạo hàm theo hướng $\boldsymbol{u}$ (một véctơ đơn vị) là độ dốc của hàm f theo hướng $\boldsymbol{u}$. Nói cách khác, đạo hàm theo hướng là đạo hàm của hàm $f(\boldsymbol{x} + \alpha\boldsymbol{u})$ theo α, được tính tại $\alpha = 0$. Sử dụng quy tắc đạo hàm của hàm hợp, ta có thể thấy rằng $\frac{\partial}{\partial \alpha} f(\boldsymbol{x} + \alpha\boldsymbol{u})$ khi $\alpha = 0$ được tính bằng $\boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$.

Để cực tiểu hóa f, ta muốn tìm hướng mà f giảm nhanh nhất. Ta có thể làm điều này bằng cách sử dụng đạo hàm theo hướng:

$$\min_{\boldsymbol{u}:\, \boldsymbol{u}^T\boldsymbol{u}=1} \boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{4.3}$$

$$= \min_{\boldsymbol{u}:\, \boldsymbol{u}^T\boldsymbol{u}=1} \|\boldsymbol{u}\|_2 \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2 \cos\theta \tag{4.4}$$

trong đó θ là góc giữa $\boldsymbol{u}$ và gradient. Thay $\|\boldsymbol{u}\|_2 = 1$ và bỏ qua các yếu tố không phụ thuộc vào $\boldsymbol{u}$, điều này rút gọn thành $\min_{\boldsymbol{u}} \cos\theta$. Giá trị này nhỏ nhất khi $\boldsymbol{u}$

hướng ngược lại với gradient. Nói cách khác, gradient chỉ trực tiếp lên phía trên, và gradient âm chỉ xuống phía dưới. Ta có thể giảm f bằng cách di chuyển theo hướng của gradient âm. Phương pháp này được gọi là **phương pháp dốc nhất** hoặc **hướng giảm**.

Phương pháp dốc nhất đề xuất một điểm mới

$$\boldsymbol{x}' = \boldsymbol{x} - \varepsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{4.5}$$

trong đó ε là **tốc độ học**, một hệ số dương xác định kích thước bước đi. Ta có thể chọn ε theo nhiều cách khác nhau. Một cách phổ biến là đặt ε nhận một hằng số nhỏ. Đôi khi, ta có thể giải cỡ bước này vì nó làm cho đạo hàm theo hướng bằng 0. Một cách tiếp cận khác là đánh giá $f(\boldsymbol{x} - \varepsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}))$ với một số giá trị của ε và chọn giá trị cho kết quả nhỏ nhất của hàm mục tiêu. Chiến lược này được gọi là **tìm kiếm theo đường thẳng**.

Phương pháp dốc nhất hội tụ khi mọi phần tử của gradient bằng 0 (hoặc, trong thực tế, rất gần 0). Trong một số trường hợp, ta có thể tránh việc chạy thuật toán lặp này và tìm trực tiếp điểm tới hạn bằng cách giải phương trình $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{0}$ theo $\boldsymbol{x}$.

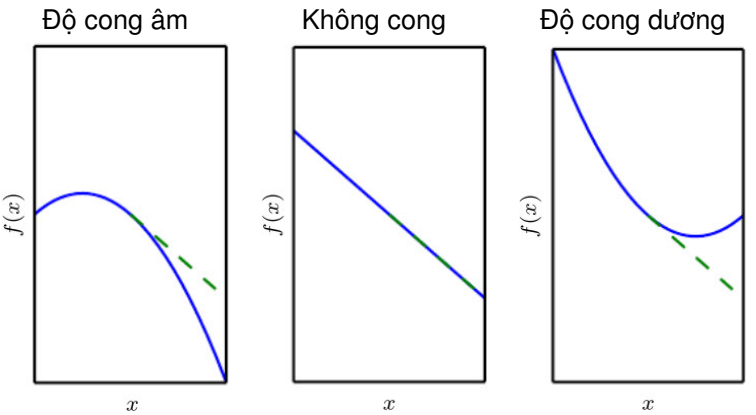
Mặc dù phương pháp hướng giảm chỉ giới hạn khi tối ưu hóa trong không gian liên tục, khái niệm tổng quát về việc lặp lại các bước di chuyển nhỏ (xấp xỉ bước tốt nhất) hướng đến các cấu hình tốt hơn có thể được mở rộng cho không gian rời rạc. Việc leo lên một hàm mục tiêu với các tham số rời rạc được gọi là **thuật toán leo đồi** (*Artificial Intelligence: A Modern Approach*, Russel và Norvig, 2021, [16]).

4.3.1 Mở rộng khái niệm gradient: ma trận Jacobi và ma trận Hess

Đôi khi ta cần tìm tất cả các đạo hàm riêng của một hàm mà đầu vào và đầu ra đều là các véctơ, hay hàm véctơ nhiều biến. Ma trận chứa tất cả các đạo hàm riêng đó được gọi là **ma trận Jacobi**. Cụ thể, nếu ta có một hàm $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, thì ma trận Jacobi $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ của $\boldsymbol{f}$ được định nghĩa bởi $J_{i,j} = \dfrac{\partial}{\partial x_j} f(\boldsymbol{x})_i$.

Đôi khi, ta cũng quan tâm đến **đạo hàm cấp hai**, tức là đạo hàm của một đạo hàm. Ví dụ, đối với hàm $f : \mathbb{R}^n \to \mathbb{R}$, đạo hàm theo biến x_i của đạo hàm của f theo biến x_j được ký hiệu là $\dfrac{\partial^2}{\partial x_i \partial x_j} f$. Trong trường hợp hàm số một biến, ta có thể ký hiệu $\dfrac{d^2}{dx^2} f$ bằng $f''(x)$. Đạo hàm cấp hai cho biết đạo hàm cấp một sẽ thay đổi

như thế nào khi ta thay đổi biến đầu vào. Điều này quan trọng vì nó cho ta biết liệu một bước gradient có mang lại cải thiện nhiều như mong đợi dựa trên gradient hay không. Ta có thể coi đạo hàm bậc hai là đo **độ cong**. Giả sử ta có một hàm bậc hai (nhiều hàm xuất hiện trong thực tế không phải là bậc hai nhưng có thể được xấp xỉ bởi hàm bậc hai trong một phạm vi nhỏ). Nếu hàm này có đạo hàm cấp hai bằng 0, thì không có độ cong nào cả. Nó là một đường hoàn toàn phẳng, và giá trị của nó có thể dự đoán chỉ dựa trên gradient. Nếu gradient là 1, ta có thể thực hiện một bước có kích thước ε theo hướng ngược với gradient, và hàm mục tiêu sẽ giảm đi ε. Nếu đạo hàm cấp hai âm, hàm số cong xuống dưới, do đó hàm mục tiêu sẽ thực tế giảm nhiều hơn ε. Cuối cùng, nếu đạo hàm bậc hai dương, hàm số cong lên trên, do đó hàm mục tiêu có thể giảm ít hơn ε. Hãy xem Hình 4.4 để thấy cách các dạng độ cong khác nhau ảnh hưởng đến mối quan hệ giữa giá trị của hàm mục tiêu được dự đoán bởi gradient và giá trị thực tế.



Hình 4.4: Đạo hàm bậc hai xác định độ cong của một hàm. Ở đây, ta minh họa các hàm bậc hai với các độ cong khác nhau. Đường nét đứt biểu thị giá trị của hàm mục tiêu mà ta mong đợi dựa trên thông tin từ gradient khi thực hiện một bước giảm theo hướng gradient. Trong trường hợp độ cong âm, hàm mục tiêu thực tế giảm nhanh hơn dự đoán của gradient. Trong trường hợp không có độ cong, gradient dự đoán chính xác mức giảm. Trong trường hợp độ cong dương, hàm số giảm chậm hơn dự kiến và cuối cùng bắt đầu tăng lên, do đó các bước quá lớn có thể thực sự làm tăng hàm số một cách không mong muốn

Khi hàm số có nhiều biến đầu vào, sẽ có nhiều đạo hàm cấp hai. Các đạo hàm này có thể được tập hợp lại thành một ma trận gọi là **ma trận Hess**. Ma trận

Hessian $\boldsymbol{H}(f)(\boldsymbol{x})$ được định nghĩa bởi

$$H(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}). \tag{4.6}$$

Nói cách khác, ma trận Hess là ma trận Jacobi của gradient.

Bất kỳ điểm nào mà các đạo hàm riêng cấp hai liên tục, các toán tử vi phân này là giao hoán, nghĩa là thứ tự của biến lấy đạo hàm có thể hoán đổi:

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}) = \frac{\partial^2}{\partial x_j \partial x_i} f(\boldsymbol{x}). \tag{4.7}$$

Điều này ngụ ý $H_{i,j} = H_{j,i}$, do đó ma trận Hess đối xứng tại những điểm đó. Hầu hết các hàm mà ta gặp trong bối cảnh học sâu đều có ma trận Hess đối xứng hầu khắp nơi. Vì ma trận Hess là ma trận thực và đối xứng, ta có thể phân tích nó thành một tập các trị riêng thực và một cơ sở trực giao của các véctơ riêng. Đạo hàm cấp hai theo một hướng cụ thể được biểu diễn bởi một véctơ đơn vị $\boldsymbol{d}$ được cho bởi $\boldsymbol{d}^T\boldsymbol{H}\boldsymbol{d}$. Khi $\boldsymbol{d}$ là một véctơ riêng của $\boldsymbol{H}$, đạo hàm cấp hai theo hướng đó được cho bởi trị riêng tương ứng. Với các hướng khác của $\boldsymbol{d}$, đạo hàm cấp hai theo hướng đó là trung bình trọng số của tất cả các trị riêng, với các trọng số từ 0 đến 1, và các véctơ riêng có góc với $\boldsymbol{d}$ nhỏ hơn sẽ nhận được trọng số lớn hơn. Giá trị riêng lớn nhất xác định đạo hàm cấp hai lớn nhất và giá trị riêng nhỏ nhất xác định đạo hàm cấp hai nhỏ nhất.

Đạo hàm cấp hai (theo hướng) cho ta biết mức độ hiệu quả của một bước trong phương pháp hướng giảm có thể đạt được. Ta có thể thực hiện xấp xỉ Taylor bậc hai cho hàm $f(x)$ quanh điểm hiện tại $\boldsymbol{x}^{(0)}$:

$$f(\boldsymbol{x}) \approx f\left(\boldsymbol{x}^{(0)}\right) + \left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)^T \boldsymbol{g} + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)^T \boldsymbol{H}\left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right) \tag{4.8}$$

trong đó $\boldsymbol{g}$ là gradient và $\boldsymbol{H}$ là ma trận Hess tại $\boldsymbol{x}^{(0)}$. Nếu ta sử dụng một tốc độ học ε, thì điểm mới $\boldsymbol{x}$ sẽ được cho bởi $\boldsymbol{x}^{(0)} - \varepsilon\boldsymbol{g}$. Thay vào xấp xỉ trên, ta được

$$f\left(\boldsymbol{x}^{(0)} - \varepsilon\boldsymbol{g}\right) \approx f\left(\boldsymbol{x}^{(0)}\right) + \varepsilon\boldsymbol{g}^T\boldsymbol{g} + \frac{1}{2}\varepsilon^2\boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g}. \tag{4.9}$$

Có ba thành phần ở đây: giá trị ban đầu của hàm, cải thiện kỳ vọng do độ dốc của hàm, và sự điều chỉnh mà ta phải áp dụng để tính đến độ cong của hàm. Khi thành phần cuối cùng này quá lớn, bước của hướng giảm thực tế có thể di chuyển ngược lên phía trên. Khi $\boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g}$ bằng 0 hoặc âm, xấp xỉ Taylor dự đoán rằng việc tăng ε mãi mãi sẽ làm giảm f mãi mãi. Trong thực tế, chuỗi Taylor khó có khả năng vẫn chính xác với ε lớn, vì vậy trong trường hợp này ta cần dùng các lựa chọn ε

mang tính kinh nghiệm. Khi $\boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g}$ dương, việc giải kích thước bước tối ưu làm giảm tối đa xấp xỉ Taylor của hàm một cách tối đa sẽ cho

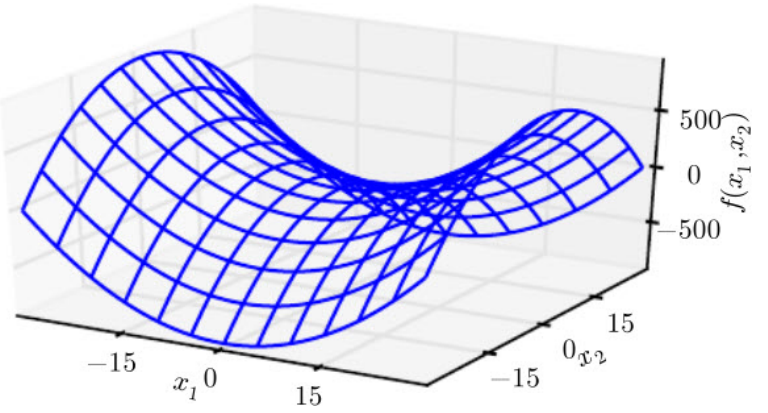
$$\varepsilon^* = \frac{\boldsymbol{g}^T\boldsymbol{g}}{\boldsymbol{g}^T\boldsymbol{H}\boldsymbol{g}}. \tag{4.10}$$

Trong trường hợp xấu nhất, khi $\boldsymbol{g}$ thẳng hàng với véctơ riêng của $\boldsymbol{H}$ tương ứng với giá trị riêng lớn nhất $\lambda_{\max}$, thì kích thước bước tối ưu sẽ là $\frac{1}{\lambda_{\max}}$. Trong chừng mực mà hàm cần cực tiểu hóa có thể được xấp xỉ tốt bằng một hàm bậc hai, các giá trị riêng của ma trận Hess sẽ xác định mức độ của tốc độ học.

Đạo hàm cấp hai có thể được sử dụng để xác định liệu một điểm tới hạn là cực đại địa phương, cực tiểu địa phương, hay là điểm yên ngựa. Nhớ rằng tại một điểm tới hạn, $f'(x) = 0$. Khi đạo hàm cấp hai $f''(x) > 0$, đạo hàm cấp một $f'(x)$ tăng khi ta di chuyển sang phải và giảm khi ta di chuyển sang trái. Điều này có nghĩa là $f'(x-\varepsilon) < 0$ và $f'(x+\varepsilon) > 0$ với ε đủ nhỏ. Nói cách khác, khi di chuyển sang phải, độ dốc bắt đầu hướng lên phải, và khi di chuyển sang trái, độ dốc bắt đầu hướng lên trái. Do đó, khi $f'(x) = 0$ và $f''(x) > 0$, ta có thể kết luận x là một cực tiểu địa phương. Tương tự, khi $f'(x) = 0$ và $f''(x) < 0$, ta có thể kết luận x là một cực đại địa phương. Đây được gọi là **kiểm tra đạo hàm cấp hai**. Tuy nhiên, khi $f''(x) = 0$, phép kiểm tra không đưa ra kết luận. Trong trường hợp này, x có thể là một điểm yên ngựa hoặc là một phần của một vùng phẳng.

Trong trường hợp hàm số nhiều biến, ta cần xem xét tất cả các đạo hàm cấp hai của hàm. Bằng cách phân tích giá trị riêng cho ma trận Hess, ta có thể tổng quát hóa phép kiểm tra đạo hàm cấp hai cho hàm số nhiều chiều. Tại một điểm tới hạn, nơi $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{0}$, ta có thể xem xét các trị riêng của ma trận Hess để xác định liệu điểm tới hạn đó là cực đại địa phương, cực tiểu địa phương, hay điểm yên ngựa. Khi ma trận Hess xác định dương (tất cả các trị riêng của nó đều dương), điểm đó là cực tiểu địa phương. Điều này có thể thấy rõ khi đạo hàm cấp hai theo mọi hướng đều dương, tương tự như phép kiểm tra đạo hàm cấp hai trong trường hợp hàm số một biến. Tương tự, khi ma trận Hess là xác định âm (tất cả các trị riêng của nó đều âm), điểm đó là cực đại địa phương. Trong trường hợp hàm số nhiều biến, ta cũng có thể tìm thấy bằng chứng xác thực về điểm yên ngựa trong một số trường hợp. Khi ít nhất một trị riêng là dương và ít nhất một trị riêng là âm, ta biết rằng $\boldsymbol{x}$ là cực đại địa phương trên một mặt cắt của f nhưng là cực tiểu địa phương trên mặt cắt khác. Xem Hình 4.5 để minh họa. Cuối cùng, phép kiểm tra đạo hàm cấp hai trong trường hợp hàm số nhiều biến có thể không đưa ra được kết luận, giống như trong trường hợp hàm số một biến, khi tất cả các trị riêng khác

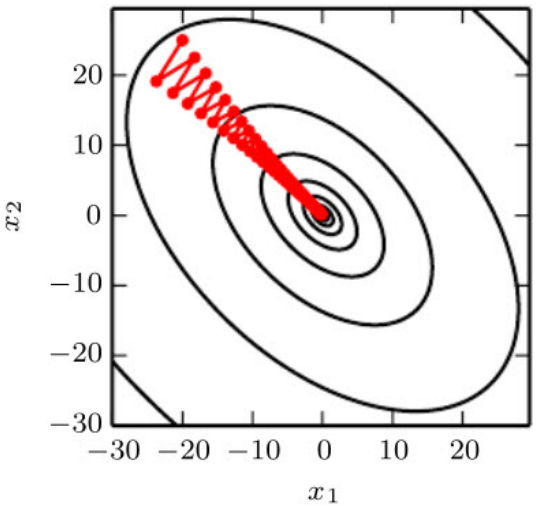
không có cùng dấu, nhưng ít nhất một trị riêng bằng không. Điều này xảy ra vì phép kiểm tra đạo hàm cấp hai trong trường hợp hàm số một biến cũng không đưa ra kết luận trên mặt cắt tương ứng với giá trị riêng bằng không.



Hình 4.5: Một điểm yên ngựa có cả độ cong dương và độ cong âm. Hàm số trong ví dụ này là $f(\boldsymbol{x}) = x_1^2 - x_2^2$. Dọc theo trục tương ứng với x_1, hàm số có độ cong hướng lên. Trục này là một véctơ riêng của ma trận Hess và có trị riêng dương. Dọc theo trục tương ứng với x_2, hàm số có độ cong hướng xuống. Hướng này là một véctơ riêng của ma trận Hess với trị riêng âm. Tên gọi "điểm yên ngựa" xuất phát từ hình dạng giống yên ngựa của hàm số này. Đây là ví dụ tiêu biểu của một hàm có điểm yên ngựa. Trong trường hợp hàm số nhiều biến, không cần phải có giá trị riêng bằng 0 để xuất hiện điểm yên ngựa: chỉ cần có cả trị riêng dương và âm. Ta có thể hình dung điểm yên ngựa với cả hai dấu của các giá trị riêng như một cực đại địa phương trong một mặt cắt và cực tiểu địa phương trong một mặt cắt khác.

Trong trường hợp hàm số nhiều biến, tại một điểm, có các đạo hàm cấp hai khác nhau theo từng hướng. Số điều kiện của ma trận Hess tại điểm này đo lường mức độ khác nhau của các đạo hàm cấp hai này. Khi ma trận Hess có số điều kiện xấu, phương pháp hướng giảm hoạt động kém hiệu quả. Nguyên nhân là do ở một hướng, đạo hàm tăng nhanh, trong khi ở hướng khác, đạo hàm tăng chậm. Phương pháp hướng giảm không nhận biết được sự thay đổi này, vì vậy không biết rằng cần phải ưu tiên khám phá hướng mà đạo hàm vẫn âm trong thời gian lâu hơn. Điều này cũng gây khó khăn trong việc chọn kích thước bước hợp lý. Kích thước bước phải đủ nhỏ để tránh vượt quá cực tiểu và đi lên ở các hướng có độ cong dương lớn. Điều này thường dẫn đến kích thước bước quá nhỏ để đạt tiến bộ đáng kể ở các hướng có độ cong thấp hơn. Xem Hình 4.6 để minh họa.

Vấn đề này có thể được giải quyết bằng cách sử dụng thông tin từ ma trận



Hình 4.6: Phương pháp hướng giảm không tận dụng được thông tin về độ cong có trong ma trận Hess. Ở đây, ta dùng phương pháp hướng giảm để cực tiểu hóa một hàm bậc hai $f(\boldsymbol{x})$ có ma trận Hess với số điều kiện là 5. Điều này có nghĩa là hướng có độ cong lớn nhất có độ cong gấp năm lần so với hướng có độ cong nhỏ nhất. Trong trường hợp này, độ cong lớn nhất là theo hướng $(1, 1)^T$, và độ cong nhỏ nhất là theo hướng $(1, -1)^T$. Các đường màu đỏ cho thấy đường đi mà phương pháp hướng giảm đã theo. Hàm bậc hai rất kéo dài này giống như một hẻm núi dài. Phương pháp hướng giảm mất thời gian liên tục đi xuống các vách hẻm núi, vì đó là đặc điểm dốc nhất. Do kích thước bước đi hơi lớn, phương pháp hướng giảm có xu hướng vượt quá đáy hàm và do đó cần phải đi xuống vách núi đối diện trong lần lặp tiếp theo. Giá trị riêng dương lớn của ma trận Hess tương ứng với véctơ riêng chỉ theo hướng này cho thấy rằng đạo hàm theo hướng này đang tăng nhanh, vì vậy một thuật toán tối ưu dựa trên ma trận Hess có thể dự đoán rằng hướng dốc nhất không phải là một hướng tìm kiếm triển vọng trong ngữ cảnh này.

Hess để hướng dẫn quá trình tìm kiếm. Phương pháp đơn giản nhất để làm điều này được gọi là **phương pháp Newton**. Phương pháp Newton dựa trên việc sử dụng khai triển Taylor bậc hai để xấp xỉ $f(\boldsymbol{x})$ gần một điểm $\boldsymbol{x}^{(0)}$:

$$f(\boldsymbol{x}) \approx f\left(\boldsymbol{x}^{(0)}\right) + \left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)^T \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^{(0)}\right) + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)^T \left[\boldsymbol{H}(f)\left(\boldsymbol{x}^{(0)}\right)\right]\left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right). \tag{4.11}$$

Nếu sau đó ta giải tìm điểm dừng của hàm này, ta thu được:

$$\boldsymbol{x}^* = \boldsymbol{x}^{(0)} - \left[\boldsymbol{H}(f)\left(\boldsymbol{x}^{(0)}\right)\right]^{-1} \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^{(0)}\right). \tag{4.12}$$

Phần II

Mạng nơron sâu: các thực hành hiện đại

Phần III

Nghiên cứu học sâu

Tài liệu tham khảo

1. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., và Bengio, Y. *Theano: A CPU and GPU Math Compiler in Python* **in** *Proceedings of the 9th Python in Science Conference* (**editors** van der Walt, S., và Millman, J.) (2010), trang 18–24.
2. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., và Bengio, Y. *Theano: new features and speed improvements* 2012. arXiv: 1211.5590 [cs.SC]. https://arxiv.org/abs/1211.5590.
3. A., C. Méthode générale pour la résolution de systèmes d'équations simultanées. *Compte rendu des séances de l'académie des sciences* Tập 25, trang 536–538. https://cir.nii.ac.jp/crid/1573668925829538688 (1847).
4. Bonaccorso, G. *Machine Learning Algorithms* In lần thứ 2. 514 trang (Packt, 2018).
5. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* In lần thứ 3. 864 trang (O'Reilly, 2022).
6. Goodfellow, I. *Deep Learning* 801 trang (2016).
7. Kaare Brandt Petersen, M. S. P. *The Matrix Cookbook* 72 trang (2012).
8. Shilov, G. E. *Linear Algebra* 864 trang (Dover Publications, 1977).
9. Jaynes, E. T. *Probability Theory: The Logic of Science* In lần thứ 22. 759 trang (Cambridge University Press, 2003).
10. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* 573 trang (Morgan Kaufmann, 1988).
11. Ramsey, F. P. *The Foundations of Mathematics and Other Logical Essays* 310 trang (Routledge, 1926).

12. Murphy, K. P. *Machine Learning: A Probabilistic Perspective* 1098 trang (MIT Press, 2012).

13. Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., và Garcia, R. *Incorporating Second-Order Functional Knowledge for Better Option Pricing* **in** *Advances in Neural Information Processing Systems* (**editors** Leen, T., Dietterich, T., và Tresp, V.) Tập 13 (MIT Press, 2000).

14. Cover, T. M., và Thomas, J. A. *Elements of Information Theory* In lần thứ 2. 774 trang (Wiley-Interscience, 2006).

15. D.J.C., M. *Information Theory, Inference and Learning Algorithms* 642 trang (Cambridge University Press, 2003).

16. Stuart J. Russell, P. N. *Artificial Intelligence: A Modern Approach, Global Edition* In lần thứ 4. 1167 trang (Pearson, 2021).