ARTIFICIAL NEURAL NETWORKS

Classification and Prediction

Lê Hồng Phương

<*phuonglh@gmail.com*>
Vietnam National University of Hanoi
Hanoi University of Science

April 2015

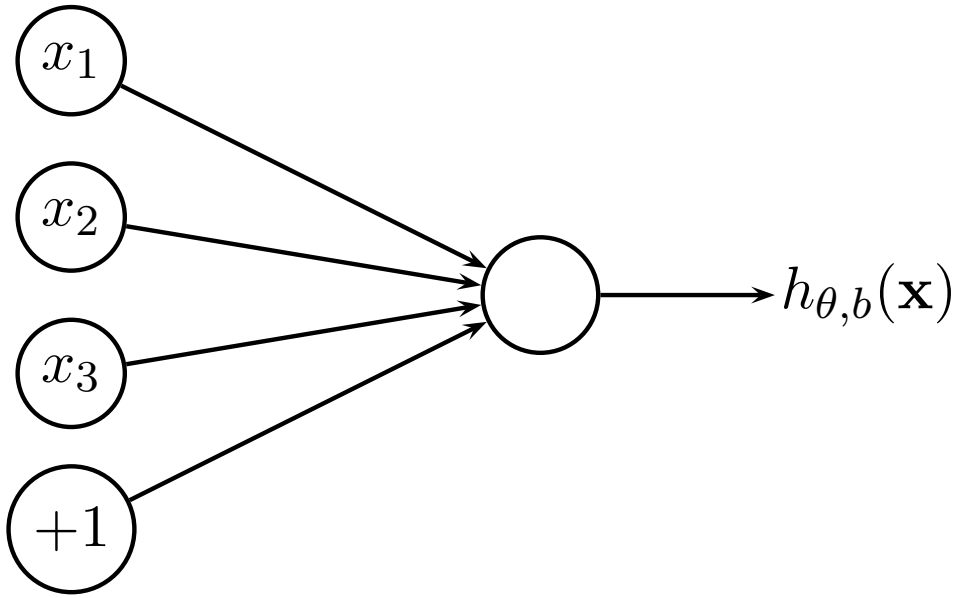
Nội dung

Nội dung

Mạng nơ-ron nhân tạo

- Mô hình toán học mô phỏng mạng nơ-ron sinh học – một nhóm các nơ-ron liên kết với nhau bằng các synapses.
- Ứng dụng thành công cho nhiều bài toán phân loại và dự báo:
  - nhận dạng tiếng nói
  - phân tích ảnh
  - điều khiển tương thích

Mạng nơ-ron đơn giản



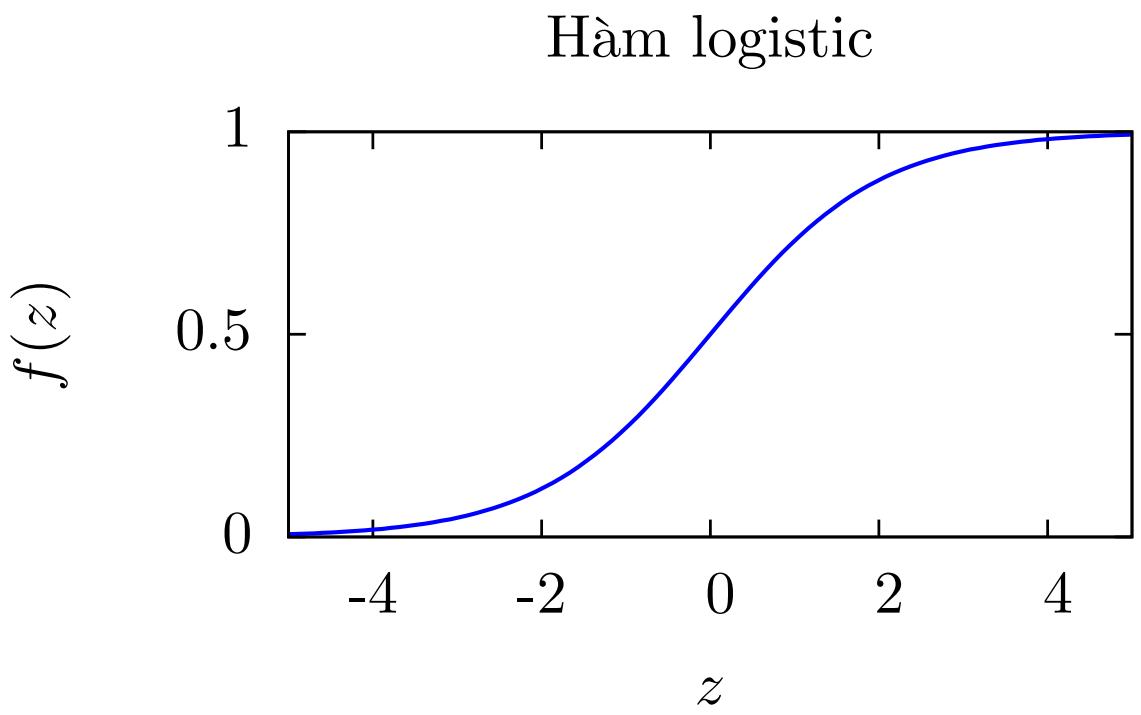
Mạng nơ-ron này chỉ gồm một khối tính toán có ba đơn vị dữ liệu vào là x_1, x_2, x_3 và một số hạng tự do $+1$; đơn vị dữ liệu ra là

$$h_{\theta,b}(\mathbf{x}) = f\left(\sum_{i=1}^{3} \theta_i x_i + b\right),$$

trong đó $f : \mathbb{R} \to \mathbb{R}$ là một *hàm kích hoạt* nào đó.

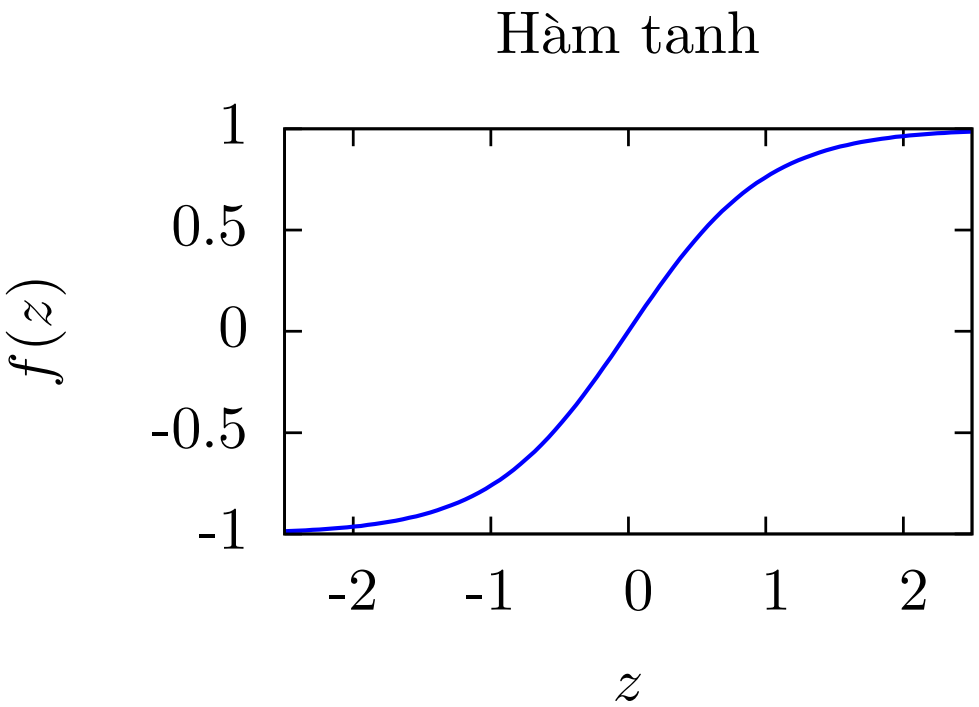
Hàm kích hoạt – Hàm sigmoid (logistic)

$$f(z) = \frac{1}{1 + \exp(-z)}$$

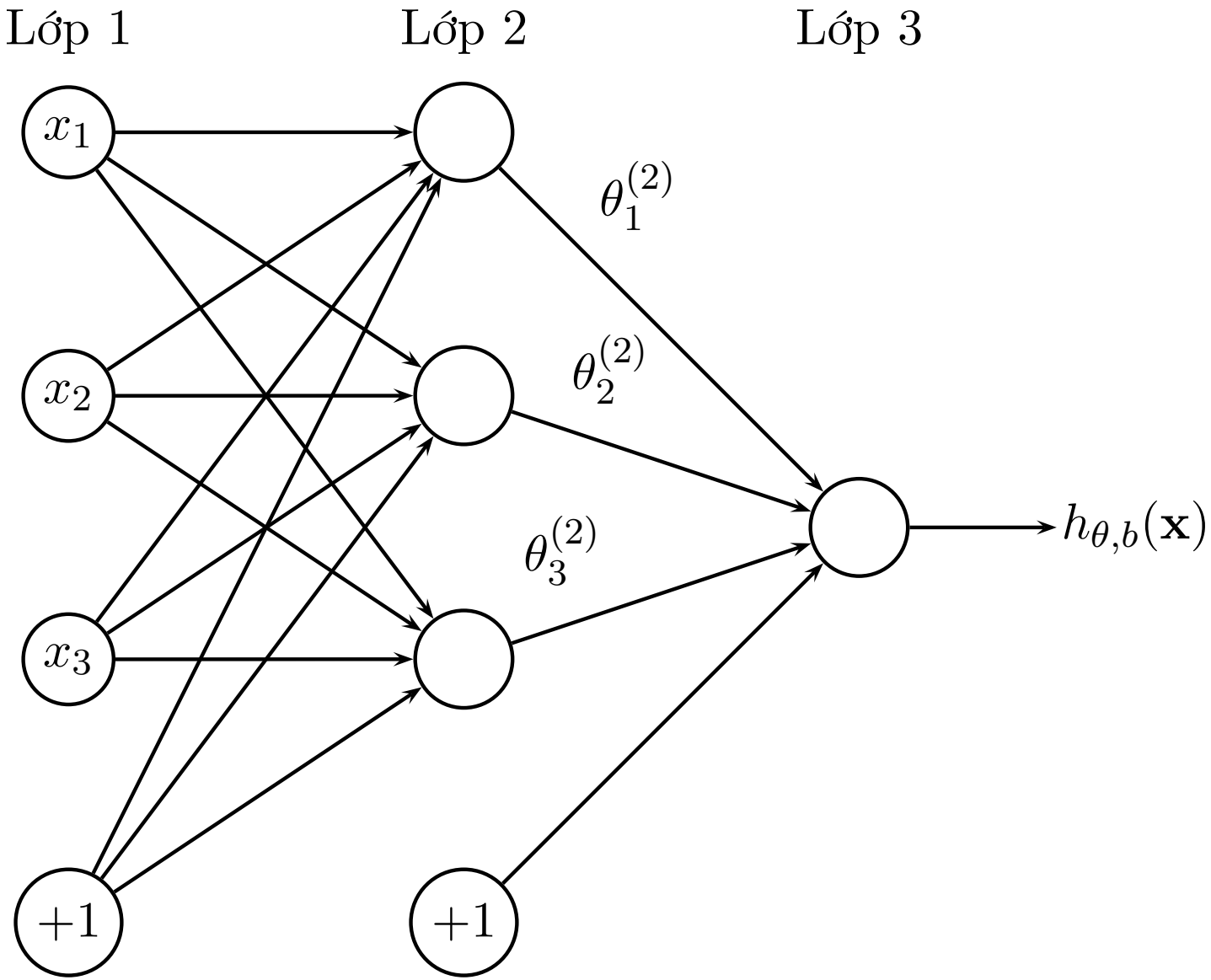


Hàm kích hoạt – Hàm tangent hyperbolic

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$



Mạng nơ-ron ba lớp

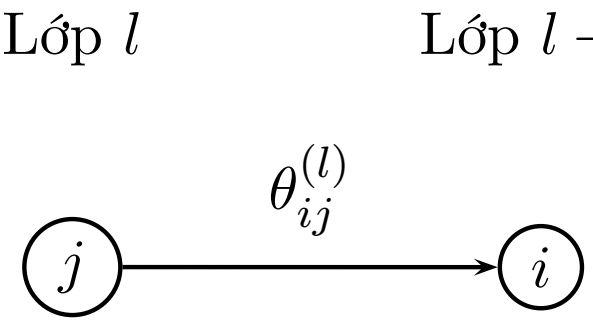


Mạng nơ-ron ba lớp

- Ta biểu diễn mỗi đơn vị bằng một hình tròn. Các dữ liệu vào cũng được biểu diễn bởi hình tròn.
- Các hình tròn có nhãn +1 biểu diễn các *đơn vị chệch*, tương ứng với các số hạng tự do. Lớp bên trái gọi là *lớp vào*, lớp bên phải là *lớp ra*. Trong ví dụ này, lớp ra chỉ gồm 1 đơn vị.
- Lớp giữa được gọi là *lớp ẩn* vì các giá trị của nó không quan sát được.
- Ta nói mạng nơ-ron này có 3 đơn vị vào (không tính đơn vị chệch), 3 đơn vị ẩn và một đơn vị ra.

Mạng nơ-ron n lớp

- Gọi n là số lớp của mạng ($n = 3$ trong ví dụ trên).
- Kí hiệu L_l là lớp thứ l của mạng; L_1 là lớp vào và L_n là lớp ra.
- Mạng nơ-ron ở trên có các tham số là $(\theta, b) = (\theta^{(1)}, b^{(1)}, \theta^{(2)}, b^{(2)})$ trong đó $\theta_{ij}^{(l)}$ biểu diễn tham số gắn với cung nối từ đơn vị j của lớp l tới đơn vị i của lớp $l + 1$:



- $b_i^{(l)}$ là độ chệch của đơn vị i trong lớp l.

Mạng nơ-ron n lớp

Trong ví dụ trên, ta có

$$\theta^{(1)} = \begin{pmatrix} \theta_{11}^{(1)} & \theta_{12}^{(1)} & \theta_{13}^{(1)} \\ \theta_{21}^{(1)} & \theta_{22}^{(1)} & \theta_{23}^{(1)} \\ \theta_{31}^{(1)} & \theta_{32}^{(1)} & \theta_{33}^{(1)} \end{pmatrix} \qquad \theta^{(2)} = \begin{pmatrix} \theta_{11}^{(2)} & \theta_{12}^{(2)} & \theta_{13}^{(2)} \end{pmatrix}$$

$$b^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{pmatrix} \qquad b^{(2)} = \begin{pmatrix} b_1^{(2)} \end{pmatrix}.$$

Mạng nơ-ron n lớp

- Ta gọi $a_i^{(l)}$ là *kích hoạt* (có ý nghĩa là giá trị ra) của đơn vị i trong lớp l.
- Với $l = 1$ thì $a_i^{(1)} = x_i$.
- Mạng nơ-ron tính toán một giá trị ra theo lược đồ sau:

$$
\begin{aligned}
a_i^{(1)} &= x_i, \qquad \forall i = 1, 2, 3; \\
a_1^{(2)} &= f\left(\theta_{11}^{(1)} a_1^{(1)} + \theta_{12}^{(1)} a_2^{(1)} + \theta_{13}^{(1)} a_3^{(1)} + b_1^{(1)}\right) \\
a_2^{(2)} &= f\left(\theta_{21}^{(1)} a_1^{(1)} + \theta_{22}^{(1)} a_2^{(1)} + \theta_{23}^{(1)} a_3^{(1)} + b_2^{(1)}\right) \\
a_3^{(2)} &= f\left(\theta_{31}^{(1)} a_1^{(1)} + \theta_{32}^{(1)} a_2^{(1)} + \theta_{33}^{(1)} a_3^{(1)} + b_3^{(1)}\right) \\
a_1^{(3)} &= f\left(\theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)} + b_1^{(2)}\right).
\end{aligned}
$$

Mạng nơ-ron n lớp

- Kí hiệu $z_i^{(l+1)} = \sum_{j=1}^{3} \theta_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$, khi đó $a_i^{(l)} = f(z_i^{(l)})$.
- Nếu ta mở rộng hàm f cho các tham số véc-tơ dạng

$$f((z_1, z_2, z_3)) = (f(z_1), f(z_2), f(z_3))$$

thì ta có thể biểu diễn các công thức tính kích hoạt dưới dạng ma trận ngắn gọn:

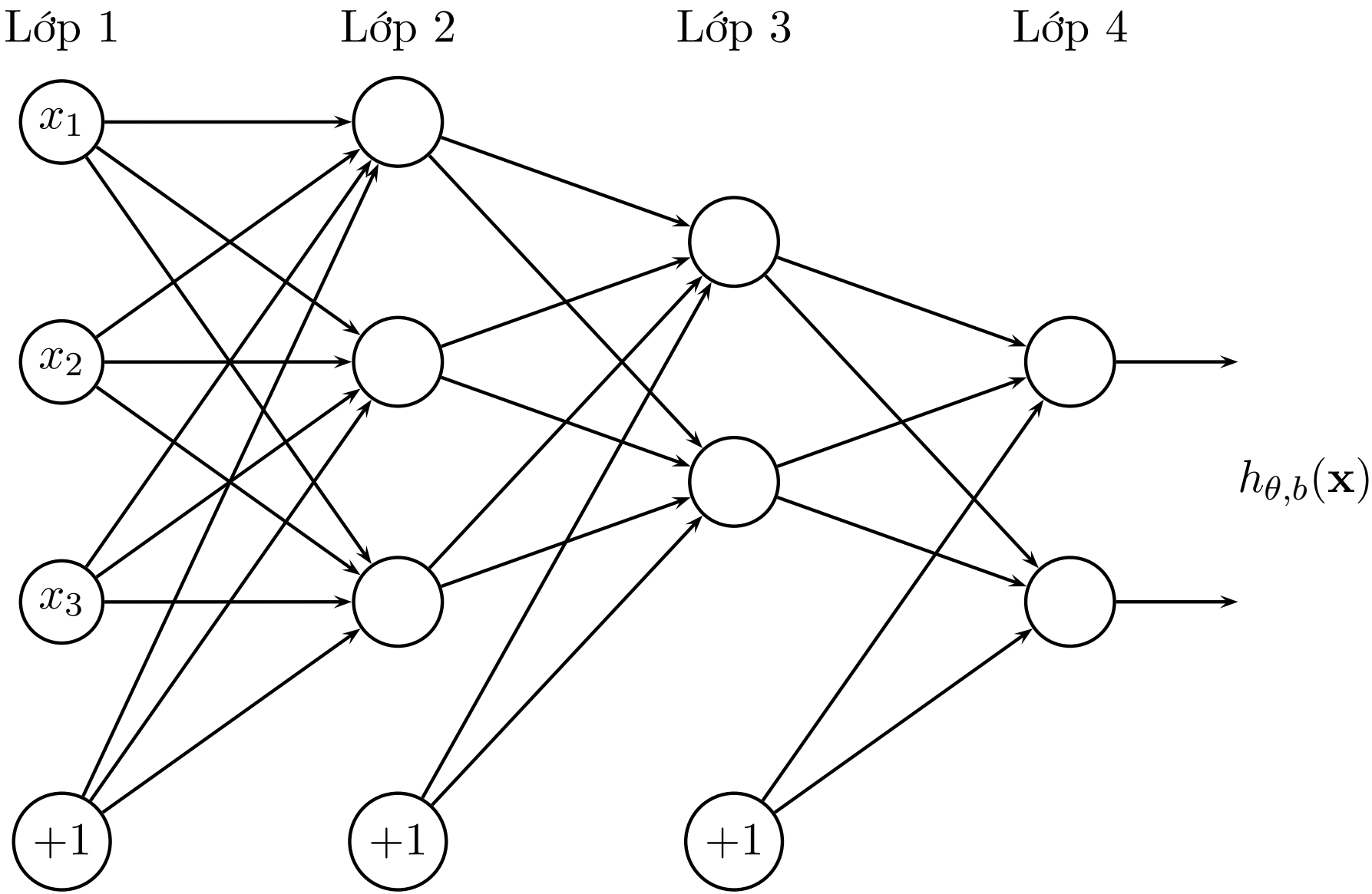
$$\begin{aligned} z^{(2)} &= \theta^{(1)} a^{(1)} + b^{(1)} \\ a^{(2)} &= f\left(z^{(2)}\right) \\ z^{(3)} &= \theta^{(2)} a^{(2)} + b^{(2)} \\ h_{\theta,b}(\mathbf{x}) &= a^{(3)} = f\left(z^{(3)}\right). \end{aligned}$$

Mạng nơ-ron n lớp

- Trong mạng nơ-ron tổng quát với n lớp, ta có thể sử dụng lược đồ tính toán tiến tương tự như trên để tính giá trị ra của mạng.
  - Trước tiên tính các kích hoạt trong lớp thứ 2, sau đó tính các kích hoạt trong lớp thứ 3...cho tới khi tính được $h_{\theta,b}(\mathbf{x}) = f\left(z^{(n)}\right)$.
- Các kích hoạt của lớp thứ $l+1$ được tính từ các kích hoạt của lớp thứ l như sau:

$$z^{(l+1)} = \theta^{(l)} a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l)}).$$

Mạng nơ-ron n lớp



Mạng nơ-ron n lớp

- Chú ý rằng mạng nơ-ron có thể có nhiều đơn vị ra.
- Để huấn luyện mô hình mạng này, ta cần có tập dữ liệu huấn luyện $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ trong đó $\mathbf{y}_i \in \mathbb{R}^2$.
- Mô hình này được sử dụng khi ta cần dự báo giá trị nhiều chiều, ví dụ trong chẩn đoán y tế, $\mathbf{x}_i$ biểu diễn các thuộc tính, đặc điểm của bệnh nhân, $\mathbf{y}_i$ là các giá trị biểu diễn có hay không các bệnh khác nhau nào đó
  - $\mathbf{y}_i = (0, 1)$ biểu diễn bệnh nhân không mắc bệnh thứ nhất nhưng mắc bệnh thứ hai.

Nội dung

Hàm tổn thất

- Giả sử tập dữ liệu huấn luyện gồm N mẫu

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}.$$

- Ta có thể huấn luyện mạng nơ-ron bằng thuật toán giảm gradient loạt.
- Với mỗi mẫu dữ liệu $(\mathbf{x}, y)$ ta gọi hàm tổn thất tương ứng là $J(\mathbf{x}, y; \theta, b)$.

Hàm tổn thất

Hàm tổn thất trên toàn bộ tập dữ liệu là

$$J(\theta, b) = \frac{1}{N}\sum_{i=1}^{N} J(\mathbf{x}_i, y_i; \theta, b) + \underbrace{\frac{\lambda}{2N}\sum_{l=1}^{n-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2}_{\text{số hạng hiệu chỉnh}}$$

trong đó s_l là số đơn vị ở lớp thứ l.

Hàm tổn thất

Hai dạng hàm tổn thất thường dùng:

1. Hàm sai số bình phương:

$$J(\mathbf{x}, y; \theta, b) = \frac{1}{2}\|y - h_{\theta,b}(\mathbf{x})\|^2.$$

2. Hàm cross-entropy:

$$J(\mathbf{x}, y; \theta, b) = -\left[y \log(h_{\theta,b}(\mathbf{x})) + (1-y)\log(1 - h_{\theta,b}(\mathbf{x}))\right],$$

trong đó $y \in \{0, 1\}$.

Thuật toán giảm gradient

- Để huấn luyện mô hình, ta cần tìm các tham số θ, b tối tiểu hóa hàm chi phí:

$$J(\theta, b) \to \min .$$

  Để giải bài toán này ta cần sử dụng các thuật toán tối ưu.
- Thuật toán đơn giản nhất là thuật toán giảm gradient.
- Vì $J(\theta, b)$ không phải là hàm lồi nên giá trị tối ưu cục bộ tìm được có thể không phải là tối ưu toàn cục.
- Tuy nhiên trong thực tế, thuật toán giảm gradient thường tìm được mô hình tốt nếu giá trị khởi đầu của các tham số được chọn một cách thích hợp.

Thuật toán giảm gradient

Trong mỗi bước lặp, thuật toán giảm gradient cập nhật các tham số θ, b như sau:

$$\theta_{ij}^{(l)} = \theta_{ij}^{(l)} - \alpha \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(\theta, b),$$

trong đó α là tốc độ học.

Thuật toán giảm gradient

Ta có

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta, b) = \frac{1}{N} \left[\sum_{i=1}^{N} \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\mathbf{x}_i, y_i; \theta, b) + \lambda \theta_{ij}^{(l)} \right]$$

$$\frac{\partial}{\partial b_i^{(l)}} J(\theta, b) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial b_i^{(l)}} J(\mathbf{x}_i, y_i; \theta, b).$$

Ở đây, ta cần tính các đạo hàm riêng:

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\mathbf{x}_i, y_i; \theta, b), \quad \frac{\partial}{\partial b_i^{(l)}} J(\mathbf{x}_i, y_i; \theta, b)$$

Thuật toán lan truyền ngược là một thuật toán hiệu quả để tính các đạo hàm riêng này.

Nội dung

Thuật toán lan truyền ngược

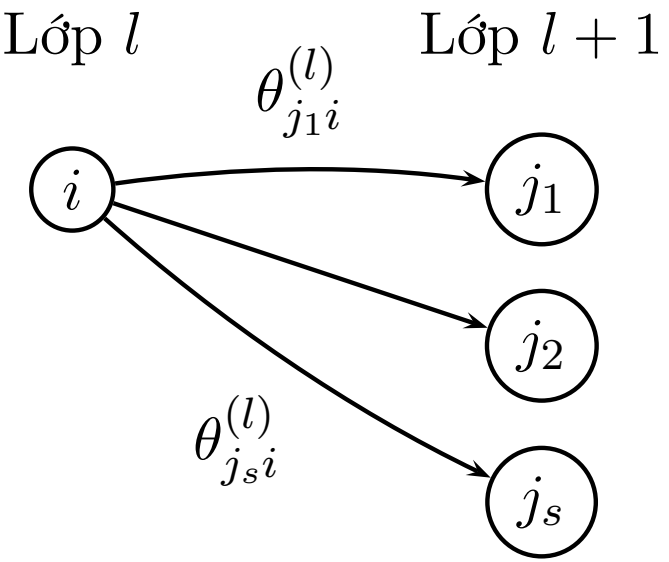
- Trước tiên, với mỗi dữ liệu $(\mathbf{x}, y)$, ta tính toán tiến qua mạng nơ-ron để tìm mọi kích hoạt, gồm cả giá trị ra $h_{\theta,b}(\mathbf{x})$.
- Với mỗi đơn vị i của lớp l, ta tính một giá trị gọi là sai số $\varepsilon_i^{(l)}$, đo phần đóng góp của đơn vị đó vào tổng sai số của đầu ra.
- Với lớp ra $l = n$, ta có thể trực tiếp tính được $\varepsilon_i^{(n)}$ với mọi đơn vị i của lớp ra bằng cách tính độ lệch của kích hoạt tại đơn vị i đó so với giá trị đúng. Cụ thể là, với mọi $i = 1, 2, \ldots, s_n$:

$$\begin{aligned}
\varepsilon_i^{(n)} &= \frac{\partial}{\partial z_i^{(n)}} \frac{1}{2} \|y - f(z_i^{(n)})\|^2 \\
&= -(y_i - f(z_i^{(n)})) f'(z_i^{(n)}) \\
&= -(y_i - a_i^{(n)}) a_i^{(n)} (1 - a_i^{(n)}).
\end{aligned}$$

Thuật toán lan truyền ngược

- Với mỗi đơn vị ẩn, $\varepsilon_i^{(l)}$ được xác định là trung bình có trọng trên các sai số của các đơn vị của lớp tiếp theo có sử dụng đơn vị này để làm đầu vào.

$$\varepsilon_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} \theta_{ji}^{(l)} \varepsilon_j^{(l+1)} \right) f'(z_i^{(l)}).$$



Thuật toán lan truyền ngược

1. Tính toán tiến, tính mọi kích hoạt của các lớp $L_2, L_3 \ldots, L_n$.
2. Với mỗi đơn vị ra i của lớp ra L_n, tính
$$\varepsilon_i^{(n)} = -(y_i - a_i^{(n)}) a_i^{(n)} (1 - a_i^{(n)}).$$
3. Tính các sai số theo thứ tự ngược: với mọi lớp $l = n-1, \ldots, 2$ và với mọi đơn vị i của lớp l, tính
$$\varepsilon_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} \theta_{ji}^{(l)} \varepsilon_j^{(l+1)} \right) f'(z_i^{(l)}).$$
4. Tính các đạo hàm riêng cần tìm như sau:
$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\mathbf{x}, y; \theta, b) = \varepsilon_i^{(l+1)} a_j^{(l)}$$
$$\frac{\partial}{\partial b_i^{(l)}} J(\mathbf{x}, y; \theta, b) = \varepsilon_i^{(l+1)}.$$

Thuật toán lan truyền ngược

- Ta có thể biểu diễn thuật toán trên ngắn gọn hơn thông qua các phép toán trên ma trận.
- Kí hiệu $\bullet$ là toán tử nhân từng phần tử của các véc-tơ, định nghĩa như sau:[1]

$$\mathbf{x} = (x_1, \ldots, x_D), \mathbf{y} = (y_1, \ldots, y_D) \Rightarrow \mathbf{x} \bullet \mathbf{y} = (x_1 y_1, x_2 y_2, \ldots, x_D y_D).$$

- Tương tự, ta mở rộng các hàm $f(\cdot), f'(\cdot)$ cho từng thành phần của véc-tơ. Ví dụ:

$$f(\mathbf{x}) = (f(x_1), f(x_2), \ldots, f(x_D))$$
$$f'(\mathbf{x}) = \left(\frac{\partial}{\partial x_1} f(x_1), \frac{\partial}{\partial x_2} f(x_2), \ldots, \frac{\partial}{\partial x_D} f(x_D) \right).$$

[1] Trong Matlab/Octave thì $\bullet$ là phép toán ".*", còn gọi là tích Hadamard.

Thuật toán lan truyền ngược

1. Thực hiện tính toán tiến, tính mọi kích hoạt của các lớp $L_2, L_3 \ldots$ cho tới lớp ra L_n:

$$z^{(l+1)} = \theta^{(l)} a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l)}).$$

2. Với lớp ra L_n, tính

$$\varepsilon^{(n)} = -(y - a^{(n)}) \bullet f'(z^{(n)}).$$

3. Với mọi lớp $l = n-1, n-2, \ldots, 2$, tính

$$\varepsilon^{(l)} = \left((\theta^{(l)})^T \varepsilon^{(l+1)}\right) \bullet f'(z^{(l)}).$$

4. Tính các đạo hàm riêng cần tìm như sau:

$$\frac{\partial}{\partial \theta^{(l)}} J(\mathbf{x}, y; \theta, b) = \varepsilon^{(l+1)} \left(a^{(l)}\right)^T$$
$$\frac{\partial}{\partial b^{(l)}} J(\mathbf{x}, y; \theta, b) = \varepsilon^{(l+1)}.$$

Thuật toán lan truyền ngược

- Ở trên là thuật toán giảm gradient áp dụng cho một dữ liệu $(\mathbf{x}, y)$.
- Thuật toán giảm gradient trên toàn bộ tập dữ liệu huấn luyện được trình bày như dưới đây.
- Kí hiệu $\nabla\theta^{(l)}$ là ma trận gradient của $\theta^{(l)}$ (cùng số chiều với $\theta^{(l)}$) và $\nabla b^{(l)}$ là véc-tơ gradient của $b^{(l)}$ (cùng số chiều với $b^{(l)}$).

Thuật toán giảm gradient

Algorithm 1: Thuật toán giảm gradient huấn luyện mạng nơ-ron

for $l = 1$ *to* n **do**

  $\nabla\theta^{(l)} \leftarrow 0; \quad \nabla b^{(l)} \leftarrow 0;$

for $i = 1$ *to* N **do**

  Tính $\frac{\partial}{\partial\theta^{(l)}} J(\mathbf{x}_i, y_i; \theta, b)$ và $\frac{\partial}{\partial b^{(l)}} J(\mathbf{x}_i, y_i; \theta, b)$;

  $\nabla\theta^{(l)} \leftarrow \nabla\theta^{(l)} + \frac{\partial}{\partial\theta^{(l)}} J(\mathbf{x}_i, y_i; \theta, b)$;

  $\nabla b^{(l)} \leftarrow \nabla b^{(l)} + \frac{\partial}{\partial b^{(l)}} J(\mathbf{x}_i, y_i; \theta, b)$;

$\theta^{(l)} \leftarrow \theta^{(l)} - \alpha \left(\frac{1}{N}\nabla\theta^{(l)} + \frac{\lambda}{N}\theta^{(l)}\right)$;

$b^{(l)} \leftarrow b^{(l)} - \alpha \left(\frac{1}{N}\nabla b^{(l)}\right)$;

Thuật toán khác

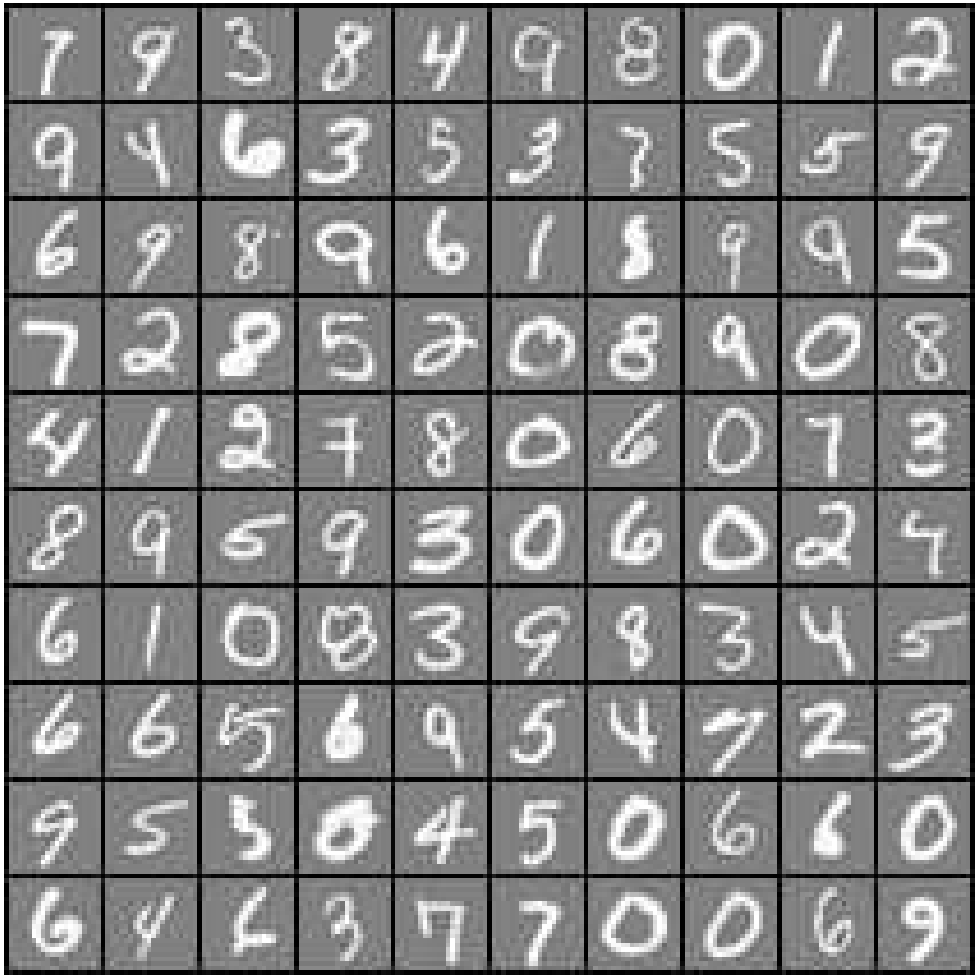
Tương tự như các mô hình khác, ngoài thuật toán giảm gradient ta có thể sử dụng các thuật toán tối ưu khác để ước lượng các tham số $\theta^{(l)}$ và $b^{(l)}$

- Các thuật toán tựa Newton như BFGS, L-BFGS hay thuật toán gradient liên hợp.
- Các thuật toán này đều cần tính các giá trị của $J(\theta, b)$ và các đạo hàm bậc một $\frac{\partial}{\partial \theta^{(l)}} J(\theta, b)$ và $\frac{\partial}{\partial b^{(l)}} J(\theta, b)$.

Nội dung

Nội dung

Nhận dạng chữ viết tay



Nhận dạng chữ viết tay

- Bộ chữ viết tay được giới hạn là các ký số ghi mã bưu điện.
- Tập dữ liệu MNIST[2] gồm 60,000 mẫu huấn luyện và 10,000 mẫu kiểm tra.
- Các tập dữ liệu do chứa các mẫu thu thập từ khoảng 750 người viết khác nhau.
- Tập mẫu kiểm tra chứa các mẫu của tập người viết khác với tập người viết của tập huấn luyện.

[2] http://yann.lecun.com/exdb/mnist/index.html

Nhận dạng chữ viết tay

Mô hình	**Tiền xử lí**	**Lỗi (%)**
Tuyến tính (mạng nơ-ron 1 lớp)	không	12.0
Tuyến tính (mạng nơ-ron 1 lớp)	giảm độ nghiêng	8.4
Mạng nơ-ron 2 lớp, 300 đơn vị ẩn	không	4.7
Mạng nơ-ron 2 lớp, 300 đơn vị ẩn	giảm độ nghiêng	**1.6**
Mạng nơ-ron 2 lớp, 1000 đơn vị ẩn	không	4.5
Mạng nơ-ron 6 lớp, 784-2500-2000-1500-1000-500-10	không	**0.35**

- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, 2010.

Nội dung

Nhận dạng tiếng nói

- Hầu hết các hệ thống nhận dạng tiếng nói hiện nay:
  - sử dụng mô hình Markov ẩn (HMMs) để mô hình quá trình biến đổi theo thời gian của giọng nói;
  - sử dụng các mô hình trộn Gauss (GMMs) để biểu diễn mối quan hệ giữa các trạng thái của HMM với tín hiệu âm thanh vào.
- Những tiến bộ lớn nhất trong lĩnh vực này được thực hiện gần 40 năm trước đây, khi ứng dụng thuật toán cực đại hóa kì vọng để huấn luyện HMM-GMMs.

Nhận dạng tiếng nói

- Gần đây, mạng nơ-ron sâu (DNNs) gồm nhiều lớp ẩn, được huấn luyện bằng một số thuật toán hiện đại đã cho kết quả tốt hơn những kết quả tốt nhất đạt được bởi GMMs.
- Các kết quả đáng chú ý của 4 nhóm nghiên cứu tại University of Toronto, Microsoft Research, Google Research, IBM Research:
  - G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing*, vol. 82, no. 2, 2012.

Nhận dạng tiếng nói

So sánh tỉ lệ lỗi trên các tập dữ liệu lớn giữa hai mô hình GMM-HMM và DNN-HMM:

Task	**Training (h)**	**DNN**	**GMM**
Switchboard (Test set 1)	309	18.5	27.4
Switchboard (Test set 2)	309	16.1	23.6
English Broadcast News	50	17.5	18.8
Bing Voice Search	24	30.4	36.2
Google Voice Input	5,870	12.3	
Youtube	1,400	47.6	52.3

Nội dung

Nội dung

Kiểm tra gradient

- Thuật toán lan truyền ngược là thuật toán khó cài đặt vì nhiều khả năng gây lỗi.
- Một lỗi nhỏ khi cài đặt như lệch chỉ số hay thiếu số hạng tự do trong các công thức đều là các lỗi khó phát hiện, trong khi vẫn huấn luyện được mạng nơ-ron cho kết quả hợp lí (nhưng không tối ưu bằng mạng nơ-ron đúng).
- Chính vì vậy, quá trình cài đặt có bị lỗi hay không là tương đối khó phát hiện.
- Ta trình bày một phương pháp để kiểm tra giá trị của các đạo hàm riêng để đảm bảo rằng kết quả tính toán là đúng.

Kiểm tra gradient

- Giả sử ta cần cực tiểu hóa hàm $J(\theta)$ với tham số θ. Trước hết ta xét hàm một biến, giả sử $\theta \in \mathbb{R}$ và $J : \mathbb{R} \to \mathbb{R}$.
- Mỗi bước trong thuật toán giảm gradient thực hiện cập nhật θ như sau:

$$\theta \leftarrow \theta - \alpha \frac{d}{d\theta} J(\theta).$$

- Giả sử ta đã cài đặt hàm $g(\theta)$ để tính $\frac{d}{d\theta} J(\theta)$ và cập nhật θ

$$\theta \leftarrow \theta - \alpha g(\theta).$$

- Làm thế nào để kiểm tra xem việc cài đặt g là đúng hay sai?

Kiểm tra gradient

- Theo định nghĩa, đạo hàm của J theo θ là

$$\frac{d}{d\theta}J(\theta) = \lim_{\epsilon \to 0} \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}.$$

- Do đó, với mọi giá trị cụ thể của θ, ta có thể xấp xỉ đạo hàm bằng đại lượng

$$\frac{J(\theta + \texttt{EPSILON}) - J(\theta - \texttt{EPSILON})}{2 \times \texttt{EPSILON}}.$$

- Khi cài đặt, ta chọn $\texttt{EPSILON} > 0$ là một hằng số nhỏ, khoảng 10^{-4} là đủ để kiểm tra độ chính xác.

Kiểm tra gradient

- Như vậy, ta có thể kiểm tra hàm $g(\theta)$ bằng cách so sánh xem giá trị của nó có thỏa mãn ràng buộc sau hay không.

$$g(\theta) \approx \frac{J(\theta + \texttt{EPSILON}) - J(\theta - \texttt{EPSILON})}{2 \times \texttt{EPSILON}}.$$

- Nếu đặt $\texttt{EPSILON} = 10^{-4}$ thì thường vế trái và vế phải của xấp xỉ trên sẽ giống nhau ở ít nhất 4 số hạng đầu tiên.

Kiểm tra gradient – Hàm nhiều biến

- Giả sử $\theta \in \mathbb{R}^n$ và $J : \mathbb{R}^n \to \mathbb{R}$. Trong ví dụ về mạng nơ-ron, hàm $J(\theta, b)$ có hai tham số θ và b, khi đó ta có thể gộp cả hai tham số θ và b thành một véc-tơ tham số.
- Giả sử $g_j(\theta)$ là hàm tính $\frac{\partial}{\partial \theta_j} J(\theta)$. Đặt $\theta^{(j+)} \leftarrow \theta + \texttt{EPSILON} \times \vec{e}_j$, trong đó $\vec{e}_j$ là véc-tơ cơ sở thứ j trong không gian n chiều:

$$\vec{e}_j = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

Kiểm tra gradient – Hàm nhiều biến

- Như vậy $\theta^{(j+)}$ chỉ lớn hơn θ một lượng nhỏ `EPSILON` ở thành phần thứ j.
- Tương tự, ta đặt $\theta^{(j-)} \leftarrow \theta - \texttt{EPSILON} \times \vec{e}_j$. Ta kiểm tra tính đúng của $g_j(\theta)$ với mọi $j = 1, 2, \ldots, n$ qua công thức xấp xỉ

$$g_j(\theta) \approx \frac{J(\theta^{(j+)}) - J(\theta^{(j-)})}{2 \times \texttt{EPSILON}}.$$

Kiểm tra gradient – Hàm nhiều biến

- Trong mô hình mạng nơ-ron, ta có

$$\frac{\partial}{\partial \theta^{(l)}} J(\theta, b) = \frac{1}{N} \nabla \theta^{(l)} + \lambda \theta^{(l)}$$
$$\frac{\partial}{\partial b^{(l)}} J(\theta, b) = \frac{1}{N} \nabla b^{(l)}.$$

- Do đó ta kiểm tra các đạo hàm riêng của $J(\theta, b)$ tương ứng theo θ và b và so sánh các giá trị này tương ứng với $\frac{1}{N} \nabla \theta^{(l)} + \lambda \theta^{(l)}$ và $\frac{1}{N} \nabla b^{(l)}$.

Nội dung

Khởi tạo ngẫu nhiên

- Giá trị khởi tạo cho các tham số θ và b có ảnh hưởng quan trọng tới hiệu quả của mô hình mạng nơ-ron.
- Chú ý rằng nếu các tham số có giá trị xấp xỉ 0 thì hàm sigmoid gần như hàm tuyến tính, do đó mạng nơ-ron trở thành mô hình gần tuyến tính.

Khởi tạo ngẫu nhiên

- Thông thường, ta sử dụng các giá trị tham số khởi đầu ngẫu nhiên xung quanh giá trị 0.
- Khi đó, ban đầu mô hình gần như là mô hình tuyến tính, sau đó dần trở thành phi tuyến khi các hệ số tăng dần.
- Nếu sử dụng toàn bộ tham số bằng 0 thì các đạo hàm riêng đều bằng 0 và hoàn toàn đối xứng, do đó thuật toán không chạy.
- Còn nếu lấy các giá trị khởi đầu lớn thì mô hình thường dẫn tới kết quả không tốt.

Nội dung

Số lớp ẩn và đơn vị ẩn

- Nói chung, nếu có nhiều đơn vị ẩn thì kết quả tốt hơn khi có ít đơn vị ẩn.
- Nếu số đơn vị ẩn quá ít thì mô hình không đủ mạnh để mô phỏng được tính phi tuyến của dữ liệu; còn nếu số đơn vị ẩn quá nhiều thì các hệ số thừa có thể bị co về 0 khi sử dụng kĩ thuật hiệu chỉnh.
- Số đơn vị ẩn thông thường nằm trong khoảng từ 5 tới 100, số đơn vị ẩn tỉ lệ thuận với số đầu vào và kích thước dữ liệu huấn luyện.

Số lớp ẩn và đơn vị ẩn

- Số lớp ẩn được chọn dựa trên kinh nghiệm và thí nghiệm.
- Mỗi lớp ẩn sẽ trích chọn các đặc trưng của đầu vào dùng trong phân loại và hồi quy.
- Việc sử dụng nhiều lớp ẩn giúp xây dựng các đặc trưng phân tầng tại các mức phân tích khác nhau.

Nội dung

Bài tập

1. Thử nghiệm mô hình mạng nơ-ron (`MultilayerPerceptron`) được cài đặt trong phần mềm Weka[3] trên một số bộ dữ liệu khác nhau.
2. Cài đặt thuật toán tính hàm tổn thất cross-entropy của mạng nơ-ron 3 lớp cho bài toán nhận dạng chữ viết tay, trong đó lớp ẩn chứa 25 nơ-ron.
3. Xây dựng thuật toán lan truyền ngược khi hàm tổn thất là hàm cross-entropy.
4. Cài đặt thuật toán lan truyền ngược ước lượng tham số của mô hình.
5. Chạy các thuật toán trên dữ liệu ảnh chữ viết tay (3 lớp, lớp ẩn chứa 25 nơ-ron, hàm tổn thất entropy), thông báo kết quả.

[3] `http://www.cs.waikato.ac.nz/ml/weka/`