CHƯƠNG 1. KHÁI NIỆM VỀ MẠNG NƠ RON NHÂN TẠO

Lợi ích của nội dung này đối với người đọc bao gồm:

- ✓ *Hiểu được khái niệm về trí tuệ nhân tạo, học máy và mạng nơ ron nhân tạo*
- ✓ *Sử dụng MATLAB để phát triển các mạng nơ ron nhân tạo cho bài toán hồi quy và phân loại*

1.1. Trí tuệ nhân tạo

Trong khoa học máy tính (computer science), trí tuệ nhân tạo hay AI (artificial intelligence) là khái niệm về trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên của con người. Khi đó, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc (bao gồm cả máy tính) có khả năng bắt chước các chức năng "nhận thức" của con người.

Những tiến bộ gần đây của AI đã ảnh hưởng đáng kể đến hầu hết mọi ngành nghề và lĩnh vực nghiên cứu khoa học. Sau đây là một số dạng AI thường được dùng nhiều nhất:

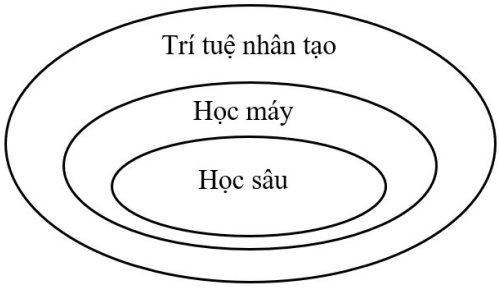
- ***Các hệ chuyên gia (Expert systems):*** được ứng dụng để giải quyết các vấn đề bằng một công cụ suy luận rút ra từ một cơ sở tri thức được trang bị thông tin về một lĩnh vực, chủ yếu ở dạng các quy tắc "nếu-thì". Được sử dụng từ những năm 1970, các hệ thống này kém linh hoạt hơn các kỹ thuật AI mới hơn nhưng dễ thực hiện và dễ lập trình hơn.
- ***Các hệ thống điều khiển logic mờ (Fuzzy logic control systems):*** cho phép tạo ra các quy tắc theo cách máy móc phản ứng với các đầu vào để giải thích cho một chuỗi các điều kiện có thể xảy ra, thay cho lý thuyết của logic kinh điển và hệ nhị phân đơn giản.
- ***Học máy (Machine learning):*** bao gồm một loạt các thuật toán và mô hình thống kê giúp các hệ thống có thể tìm thấy các mẫu, rút ra suy luận và học cách thực hiện các tác vụ mà không cần hướng dẫn cụ thể.
- ***Mạng nơ ron (Neural networks):*** là các hệ thống học máy với bao gồm rơ ron nhân tạo xử lý thông tin dựa trên cấu trúc và chức năng của bộ não sinh học.
- ***Học sâu (Deep learning)****:* là một dạng của học máy dựa trên mạng nơ ron nhân tạo. Kiến trúc học sâu có thể kết hợp với quá trình trích xuất đặc trưng tối ưu làm cho chúng đặc biệt hữu ích cho các mục đích như nhận dạng giọng nói và hình ảnh hay xử lý ngôn ngữ tự nhiên.

1.2. Học máy

Học máy là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các thuật toán cho phép các hệ thống có khả năng "học" từ dữ liệu. Ngày nay, một nhánh của học máy là học sâu đang phát triển rất mạnh mẽ dựa trên các mạng nơ ron nhân tạo.

Mối liên hệ giữ trí tuệ nhân tạo, học máy và học sâu được trình bày như hình 1.1 hay một cách khác ta có thể nói:

- Học sâu là một tập con của học máy.
- Học máy là một tập con của trí tuệ nhân tạo.



Hình 1.1. Học sâu là một tập con của học máy và học máy là một tập con của trí tuệ nhân tạo.

1.3. Phân loại các thuật toán học máy theo dữ liệu

Các thuật toán học máy có thể được phân loại theo kiểu dữ liệu học bao gồm:

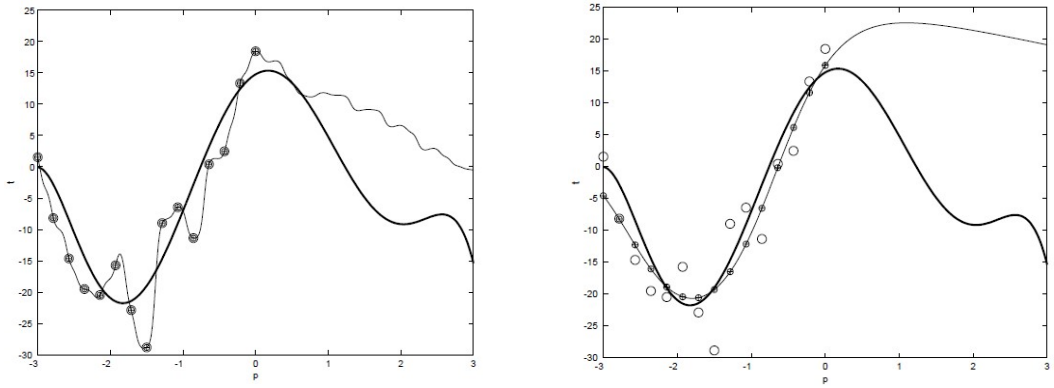
- ***Học có giám sát (Supervised learning):*** là các kỹ thuật học máy xây dựng một hàm từ dữ liệu huấn luyện. Dữ liệu huấn luyện bao gồm các đối tượng đầu vào (thường là ở dạng véc tơ) và đầu ra mong muốn. Nói một cách khác, dữ liệu huấn luyện được dán nhãn. Đầu ra của hàm có thể là một giá trị liên tục (được gọi hồi quy) hay có thể là dự đoán một nhãn phân loại cho một đối tượng đầu vào (được gọi phân loại). Các thuật toán học có giám sát điển hình bao gồm: hồi quy tuyến tính (Linear regression), cây quyết định (Decision tree), máy véc tơ hỗ trợ (Support vector machine), Gaussian Naive Bayes, K hàng xóm gần nhất (K-nearest neighbors), rừng ngẫu nhiên (Random forest), mạng nơ ron nhân tạo (Artificial neural networks).
- ***Học không có giám sát (Unsupervised learning):*** là các kỹ thuật học máy tìm ra một mô hình phù hợp với các quan sát. Tương phản với học có giám sát, học không có giám sát được thực hiện với dữ liệu không được dán nhãn.
- ***Học bán giám sát (Semi-supervised learning*)*:*** là các kỹ thuật học máy sử dụng cả dữ liệu đã dán nhãn và chưa dán nhãn để huấn luyện: điển hình là một lượng nhỏ dữ liệu có dán nhãn cùng với lượng lớn dữ liệu chưa dán nhãn.

1.4. Tính tổng quát của học máy

Trong học máy, tính tổng quát hóa (generalisation) là một khái niệm thể hiện mức độ hiệu quả của một mô hình được đào tạo để phân loại hoặc dự báo dữ liệu chưa nhìn thấy (chưa từng sử dụng để huấn luyện mô hình). Do đó, tính đa dạng của dữ liệu là yếu tố rất quan trọng để đưa ra một phân loại hay dự báo tốt.

Tính đa dạng của dữ liệu không phải là yếu tố quan tâm duy nhất để mô hình có tính tổng quát cao. Thực vậy, giá trị của các siêu thông số (hyperparameters) cũng ảnh hưởng đến tính tổng quát hóa của mô hình. Để xác định giá trị phù hợp của các siêu thông số, một số phương pháp chuẩn hóa được áp dụng trong quá trình huấn luyện mô hình.

Ví dụ một mạng nơ ron có kích thước lớn được huấn luyện để cực tiểu hóa một hàm sai số bình phương của bài toán hồi quy với 15 điểm của dữ liệu huấn luyện như hình 1.2. Theo hình 1.2a, đáp ứng của mạng nơ ron khớp chính xác với 15 điểm của dữ liệu huấn luyện. Tuy nhiên, đáp ứng của mạng không thể hiện được đầu ra của hàm thật do mạng sau khi huấn luyện đã khớp với tất cả các dữ liệu bao gồm cả dữ liệu nhiễu hay mang tính không chắc chắn. Lúc này, ta có thể nói đáp ứng của mạng là quá phức tạp.



(a) (b)

Hình 1.2. Ví dụ về quá khớp a) và vừa khớp b)

1.5. Các tập dữ liệu học máy

Một tập dữ liệu học máy là một tập hợp dữ liệu được sử dụng để huấn luyện các mô hình học. Sau khi được học từ dữ liệu, một thuật toán học máy có thể đưa ra dự đoán với các dữ liệu mới chưa từng được sử dụng để huấn luyện mô hình. Các loại dữ liệu phổ biến học máy bao gồm:

- Dữ liệu văn bản
- Dữ liệu hình ảnh
- Dữ liệu âm thanh
- Dữ liệu video
- Dữ liệu số

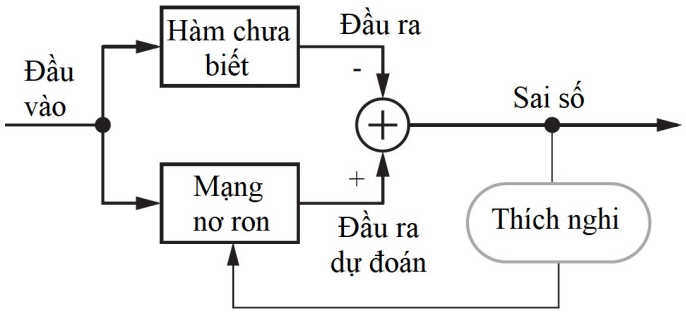
Đối với một tập dữ liệu sẵn có ban đầu, chúng ta có thể chia nhỏ tập dữ liệu này thành ba tập dữ liệu như sau:

- ***Tập dữ liệu huấn luyện (Training dataset):*** Đây được xem như là tập dữ liệu quan trọng nhất chiếm 60% của toàn bộ tập dữ liệu ban đầu được sử dụng để huấn luyện mô hình. Nói một cách khác, tập dữ liệu này được sử dụng để huấn luyện thuật toán những gì cần tìm kiếm trong dữ liệu.
- ***Tập dữ liệu hợp thức hóa (Validation dataset):*** Tập dữ liệu này chiếm khoảng 20% của toàn bộ tập dữ liệu ban đầu và được sử dụng để đánh giá các tham số của mô hình sau khi quá trình huấn luyện mô hình được hoàn tất. Tập dữ liệu này được sử dụng để kiểm tra mô hình sau khi huấn luyện để khảo sát hiện tượng chưa khớp (underfitting) hay quá khớp (overfitting) với dữ liệu hợp thức hóa.
- ***Tập dữ liệu thử nghiệm (Test dataset):*** Tập dữ liệu này chiếm 20% tổng dữ liệu ban đầu và được sử dụng để kiểm tra độ chính xác của mô hình.

1.6. Mạng nơ ron

Mạng nơ ron được sử dụng rộng rãi trong các mô hình học máy và có một lịch sử phát triển dài với nhiều thành tựu thu được qua hàng loạt các công trình nghiên cứu và ứng dụng. Thêm đó, các tài liệu về mạng nơ ron cũng rất phong phú và đa dạng. Cùng với việc phát triển gần đây của khái niệm học sâu, lý thuyết và ứng dụng của mạng nơ ron đã trở nên vô cùng quan trọng. Đối với những người mới bắt đầu làm quen với mạng nơ ron, phần này sẽ bổ sung các khái niệm cơ bản.

Có nhiều loại mạng nơ ron khác nhau. Tuy nhiên, các mạng nơ ron được phát triển cho hai dạng bài toán cơ bản sau: các bài toán hồi quy (regression problems) và các bài toán phân loại (classification problems). Đối với bài toán hồi quy, mạng nơ ron có thể được xem là một xấp xỉ hàm (function approximator) cho một hàm chưa biết với nguyên lý có dạng như hình 1.3.



Hình 1.3. Nguyên lý của mạng nơ ron được sử dụng như một xấp xỉ hàm.

Các thông tin đầu vào và đầu ra của hàm chưa biết được sử dụng để cập nhật thông số của mạng nơ ron. Sai số giữa đầu ra của hàm chưa biết và đầu ra của mạng nơ ron được dùng để hiệu chỉnh thông số của mạng qua một khâu thích nghi. Quá trình cập nhật thông số của mạng nơ ron hay còn gọi là quá trình huấn luyện mạng nơ ron thực chất là một quá trình lặp được thực hiện đến khi sai số giữa đầu ra của mạng và đầu ra mong muốn (mục tiêu) nhỏ hơn một giá trị cho trước hoặc số bước lặp đạt đến giá trị cho trước.

1.6.1. Các nút của mạng nơ ron

Trong một mạng nơ ron nhân tạo, phần tử cơ bản là các nơ ron nhân tạo hay còn được gọi là các nút (nodes). Nguyên tắc xử lý thông tin của một mạng nơ ron nhân tạo dựa trên phương thức xử lý thông tin của nơ ron sinh học có kết nối giữa các nút với nhau. Bảng 1.1 là sự tương đồng của não bộ và mạng nơ ron.

Bảng 1.1. Sự tương đồng của não bộ và mạng nơ ron.

Não bộ	Mạng nơ ron
Nơ ron sinh học	Nút (nơ ron nhân tạo)
Kết nối của các nơ ron sinh học	Trọng số kết nối các nút

Hình 1.4 là ví dụ về mô hình toán học của một nút với ba tín hiệu đầu vào là x_1, x_2 và x_3. w_1, w_2 và w_3 là các trọng số (weights) ứng với các tín hiệu đầu vào. b được gọi là độ lệch (bias). y là đầu ra của nút.

Kích hoạt (activation) của nút được tính như sau như sau:

$$a = w_1x_1 + w_2x_2 + w_3x_3 + b = wx + b$$

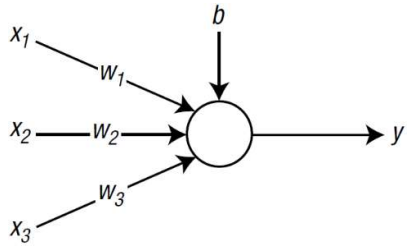
Trong đó:

$$w = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Tiếp đó kích hoạt của nút là đầu vào của một hàm số $f(.)$. Hàm này được gọi là hàm kích hoạt (activation function) của nút.

$$y = f(a)$$



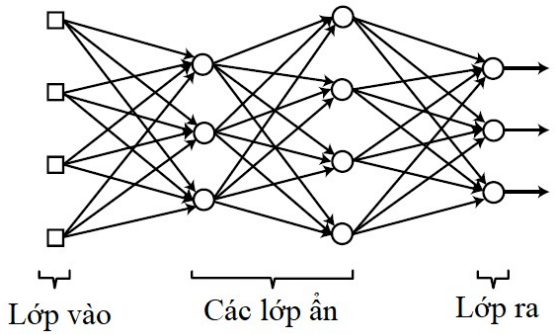
Hình 1.4. Một nút nhận ba đầu vào.

Tùy theo từng loại mạng nơ ron nhân tạo, hàm kích hoạt có dạng khác nhau.

1.6.2. Các lớp của mạng nơ ron

Một mạng nơ ron nhân tạo là một mạng của các nút. Cấu trúc phổ biến của mạng nơ ron nhân tạo bao gồm các lớp của các nút có dạng như hình 1.5. Lớp ngoài cùng bên trái được gọi là lớp đầu vào (input layer) và lớp ngoài cùng bên phải được gọi là lớp đầu ra (output layer). Giữa lớp đầu vào và lớp đầu ra là các lớp ẩn (hidden layers). Độ phức tạp của mạng nơ ron phụ thuộc vào số lượng lớp ẩn và số lượng nút của các lớp ẩn. Điều này có nghĩa là mạng nơ ron có càng có nhiều lớp ẩn và nhiều nút thì mạng càng phức tạp. Hình 1.6 là các dạng mạng nơ ron theo cấu trúc lớp.

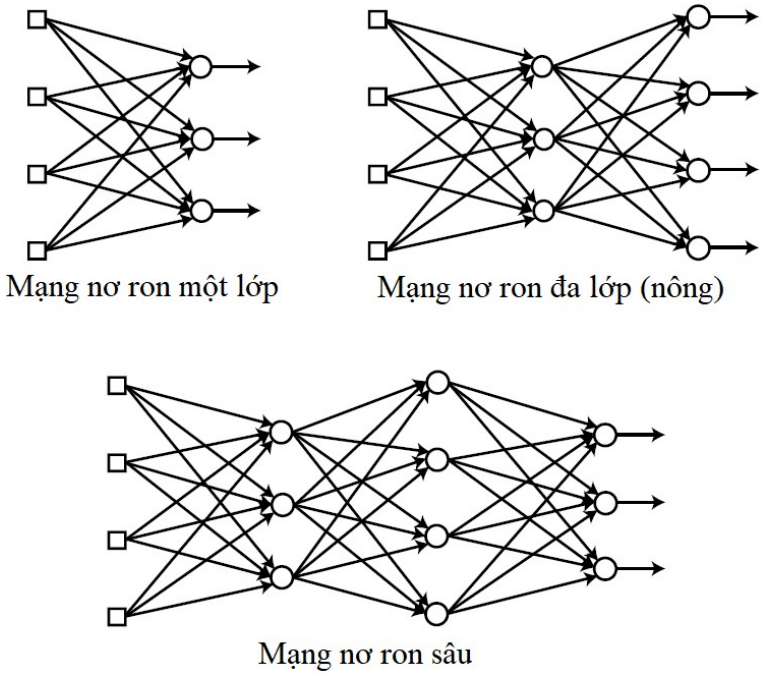
- Dạng cấu trúc đơn giản nhất của mạng nơ ron nhân tạo chỉ có một lớp đầu vào và một lớp đầu ra. Các mạng nơ ron này được gọi là các mạng nơ ron một lớp (single-layer neural networks).
- Các mạng nơ ron đa lớp chỉ có một lớp ẩn được gọi là các mạng nơ ron nông (shallow neural networks).
- Các mạng nơ ron đa lớp có từ hai lớp ẩn trở lên được gọi là các mạng nơ ron sâu (deep neural networks).



Hình 1.5. Cấu trúc lớp của các nút.

Bảng 1.1. Tổng hợp các loại mạng nơ ron theo cấu trúc lớp.

Mạng nơ ron một lớp		Lớp vào - Lớp ra
Mạng nơ ron đa lớp	Mạng nơ ron đa lớp (nông)	Lớp vào – Một lớp ẩn – Lớp ra
	Mạng nơ ron sâu	Lớp vào – Nhiều lớp ẩn – Lớp ra



Hình 1.6. Các dạng mạng nơ ron theo cấu trúc lớp.

1.6.3. Mạng nơ ron nông

Mạng nơ ron nông có duy nhất một lớp ẩn. Đối với bài toán hồi quy, mạng nơ nông có thể xấp xỉ bất kỳ một hàm nào. Số nơ ron của lớp ẩn được xác định theo phương pháp đơn giản bằng cách tăng hay giảm số nút ẩn đến khi mạng đạt giá trị sai số nhỏ nhất. Mạng sử dụng véc tơ của đầu vào thực, x_i , và để tính kích hoạt các nút ẩn như sau:

$$a_j^{(1)} = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)} \tag{1.1}$$

Trong đó $w_{ji}^{(1)}$ là trọng số liên kết nút vào thứ i đến nút ẩn thứ j. b_j là độ lệch của nút ẩn thứ j. $a_j^{(1)}$ là kích hoạt của nút ẩn thứ j. d là số nút vào.

Đầu ra của nút ẩn thứ j được tính như sau:

$$y_j = f^{(1)}\left(a_j^{(1)}\right) \tag{1.2}$$

Trong đó, $f^{(1)}(.)$ là hàm kích hoạt của các nút ẩn và y_j là đầu ra nút ẩn thứ j. Hàm kích hoạt của lớp ẩn là hàm tan hyperbolic (tanh) có dạng như sau:

$$y_j = f^{(1)}\left(a_j^{(1)}\right) = \frac{\exp\left(a_j^{(1)}\right) - \exp\left(-a_j^{(1)}\right)}{\exp\left(a_j^{(1)}\right) + \exp\left(-a_j^{(1)}\right)} \tag{1.3}$$

Hàm (1.3) có tính chất như sau:

$$\frac{\partial f^{(1)}\left(a_j^{(1)}\right)}{\partial a_j^{(1)}} = 1 - \left[f^{(1)}\left(a_j^{(1)}\right)\right]^2 = 1 - y_j^2 \tag{1.4}$$

Kích hoạt của nút ra thứ k được tính như sau:

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} y_j + b_k^{(2)} \tag{1.5}$$

Trong đó $w_{kj}^{(2)}$ là trọng số liên kết từ nút ẩn thứ j đến nút ra thứ k. $b_k^{(2)}$ là độ lệch của nút ra thứ k. M là số nút ẩn.

Đầu ra thứ k của mạng được xác định bằng một hàm kích hoạt $f^{(2)}(.)$ có dạng như sau:

$$z_k = f^{(2)}\left(a_k^{(2)}\right) \tag{1.6}$$

Đối với mạng nơ ron cho bài toán hồi quy, hàm kích hoạt đầu ra là một hàm tuyến tính có dạng như sau:

$$f^{(2)}\left(a_k^{(2)}\right) = a_k^{(2)} \tag{1.7}$$

Đối với mạng nơ ron cho bài toán phân loại chỉ có hai nhóm, hàm kích hoạt đầu ra là hàm 'sigmoid' có dạng như sau:

$$f^{(2)}\left(a_k^{(2)}\right) = \frac{1}{1+\exp\left(-a_k^{(2)}\right)} \tag{1.8}$$

Đối với mạng nơ ron cho bài toán phân loại nhiều hơn hai nhóm, hàm kích hoạt đầu ra là hàm 'softmax' có dạng như sau:

$$f^{(2)}\left(a_k^{(2)}\right) = \frac{\exp\left(a_k^{(2)}\right)}{\sum_{k'} \exp\left(a_{k'}^{(2)}\right)} \tag{1.9}$$

1.7. Huấn luyện mạng nơ ron một lớp

1.7.1. Quy tắc Delta

Mục đích của quá trình huấn luyện mạng nơ ron là để thu được giá trị của các trọng số và độ lệch thích hợp dựa trên dữ liệu huấn luyện mạng. Hiệu năng của mạng nơ ron sau khi huấn luyện sẽ dựa trên các dữ liệu mới (dữ liệu để kiểm nghiệm hiệu năng của mạng sau khi được huấn luyện). Trong phần này, chúng ta xem xét đến quy tắc Delta được ứng dụng để huấn luyện mạng nơ ron một lớp. Theo quy tắc này, trọng số được cập nhật tại các bước lặp như sau:

$$e_k(m) = z_k(m) - d_k(m) \tag{1.10}$$

$$w_{ki}(m+1) = w_{ki}(m) - \eta e_k(m) x_i \tag{1.11}$$

$$b_k(m+1) = b_k(m) - \eta e_k(m) \tag{1.12}$$

Trong đó:

$e_k(m)$ là sai số đầu ra của mạng $z_k(m)$ và đầu ra thực $d_k(m)$ tại bước lặp thứ m và x_i là đầu vào thứ i.

$w_{ki}(m)$ là trọng số liên kết giữa nút vào thứ i đến nút ra thứ k tại bước lặp thứ m. Tương tự, $w_{ki}(m+1)$ là trọng số liên kết giữa nút vào thứ i đến nút ra thứ k tại bước lặp thứ $m+1$.

$b_k(m)$ là độ lệch của nút ra thứ k tại bước lặp thứ m. Tương tự, $b_k(m+1)$ là độ lệch của nút ra thứ k tại bước lặp thứ $m+1$.

Hệ số η được gọi là tốc độ học ($0 < \eta \le 1$).

1.7.2. Quy tắc Delta tổng quát

Quy tắc delta tổng quát có dạng như sau:

$$\delta_k(m) = f'(a_k(m))e_k(m) \tag{1.13}$$

$$w_{ki}(m+1) = w_{ki}(m) - \eta\delta_k(m)x_i \tag{1.14}$$

$$b_k(m+1) = b_k(m) - \eta\delta_k(m) \tag{1.15}$$

Trong đó, $f'(a_k(m))$ là đạo hàm của hàm kích hoạt đầu ra tại bước lặp thứ m.

Ví dụ 1.1:

Thiết kế mạng nơ ron hồi quy để xấp xỉ hàm tuyến tính có dạng như sau:

$$y = 10x + 7$$

Khi có nhiễu, phương trình trên có dạng như sau:

$$t = y + \sigma$$

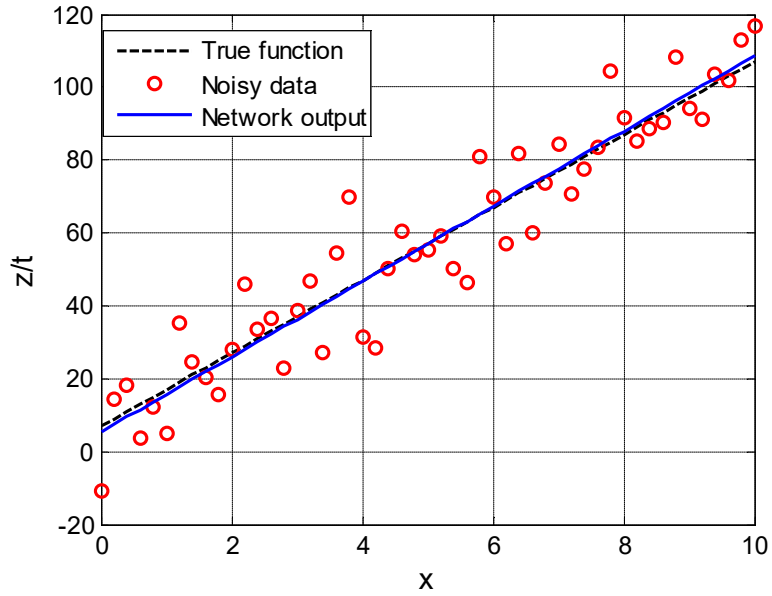
Trong đó, σ là nhiễu và t là giá trị của hàm với nhiễu. Do đó, dữ liệu dùng để huấn luyện mạng nơ ron là các giá trị của x và t tương ứng. Bảng 1.2 là code MATLAB để vẽ đồ thị hàm số và tạo dữ liệu với nhiều. Hình 1.5 là đồ thị của hàm tuyến tính (đường thẳng màu đen nét đứt), các giá trị của hàm với nhiễu (các vòng tròn màu đỏ) và đầu ra của mạng (đường thẳng màu xanh nét liền).

Bảng 1.2. Code MATLAB thiết kế mạng nơ ron một lớp để xấp xỉ hàm tuyến tính trong ví dụ 1.1.

Cú pháp MATLAB	Chức năng

x = 0:0.2:10;
N = length(x);
y_true = 10*x + 7;
sigma = 10*randn(1,N);
t = y_true + sigma;
plot(x,y_true,'k--','linewidth',1.5);
hold on;
plot(x,t,'ro','linewidth',1.5);
grid on;
xlabel('x','FontName','Arial','FontSize',12);
ylabel('z/t','FontName','Arial','FontSize',12);
hold on;
max_x = max(x);
min_x = min(x);
max_t = max(t);
min_t = min(t);
x1 = (x - min_x)/(max_x - min_x);
t1 = (t - min_t)/(max_t - min_t);
	Khoảng giá trị khảo sát của biến từ 0 đến 10 với độ phân giải là 0,2.
	Xác định số lượng các giá của biến.
	Phương trình của nhiễu. Các giá trị nhiễu có phân bố Normal.
	Tính các giá trị của hàm với nhiễu.
	Vẽ đồ thị của hàm và các điểm của hàm với nhiễu.
	
	Chuẩn hóa dữ liện đầu vào và đầu ra của hàm bằng cách ánh xạ dải giá trị đầu vào và đầu ra thành các giá trị nằm trong khoảng [0,1].

nin = size(x1,1);
nout = 1;
w = randn(nout,nin)/sqrt(nin + 1);
b = randn(nout,1)/sqrt(nin + 1);
M = 5000;
nu = 0.05;
for i = 1:M
  a = w*x1 + b*ones(1,N);
  z = a;
  e = z - t1;
  delta = e;
  dw = nu*delta*x1';
  db = nu*delta*ones(N,1);
  w = w - dw;
  b = b - db;
end
a = w*x1 + b*ones(1,N);
z = a;
z1 = z*(max_t - min_t) + min_t;
plot(x,z1,'b-','linewidth',1.5)
hold on;
legend('True function','Noisy data', 'Network output');
	Xác định số nút đầu vào theo dữ liệu.
	Số nút đầu ra là 1.
	Khởi tạo trọng số và độ lệch.
	Đặt số bước lặp $M = 5000$.
	Đặt tốc độ học $\eta = 0,05$.
	Thực hiện quá trình lặp để cập nhật trọng số và độ lệch.
	
	Kiểm tra hiệu năng của mạng sau khi huấn luyện qua quan sát đồ thị của hàm, giá trị của hàm với nhiễu và đầu ra của mạng nơ ron với dữ liệu đầu vào.



Hình 1.5. Đồ thị hàm tuyến tính, dữ liệu nhiễu và đầu ra của mạng nơ ron một lớp trong ví dụ 1.1.

Ví dụ 1.2:

Thiết kế mạng nơ ron một lớp dùng để phân loại hai đầu vào của một cổng logic HOẶC có bảng chân lý như bảng 1.3.

Bảng 1.3. Bảng chân lý của cổng logic hoặc hai đầu vào trong ví dụ 1.1.

Đầu vào 1 (x_1)	Đầu vào 2 (x_2)	Đầu ra cổng logic (t)

0	0	0
0	1	1
1	0	1
1	1	1

Bảng 1.4 là code MATLAB thiết kế mạng nơ ron một lớp. Bảng 1.5 là so sánh giữa đầu ra của mạng và đầu ra mong muốn sử dụng thông tin của hai đầu vào.

Bảng 1.4. Code MATLAB xây dựng mạng nơ ron một lớp trong ví dụ 1.2.

Cú pháp MATLAB	Chức năng

x = [0 0; 0 1; 1 0; 1 1];
t = [0 1 1 1]';
N = size(x,1);
nin = size(x,2);
nout = 1;
w = randn(nout,nin)/sqrt(nin + 1);
b = randn(nout,1)/sqrt(nin + 1);
M = 2000;
nu = 0.5;
for i = 1:M
  a = w*x' + b*ones(1,N);
  z = 1./(1 + exp(-a));
  e = z - t';
  delta = z.*(1 - z).*e;
  dw = delta*x;
  db = delta*ones(N,1);
  w = w - nu*dw;
  b = b - nu*db;
end
a = w*x' + b*ones(1,N);
z = 1./(1 + exp(-a));
result = [z' t]
	Nhập dữ liệu đầu vào
	Nhập dữ liệu đầu ra mong muốn
	Nhập số mẫu của dữ liệu huấn luyện mạng
	Xác định số đầu vào của mạng
	Nhập số đầu ra của mạng
	Khởi tạo trọng số ngẫu nhiên
	Thực hiện quá trình lặp để cập nhật trọng số và độ lệch:
	• Số bước lặp là $M = 2000$
	• Tốc độ học $\eta = 0,5$
	• Tính kích hoạt đầu ra
	• Tính sai số đầu ra mạng và đầu ra mong muốn
	• Tính $\delta = z(1-z)e$
	• Cập nhật trọng số $w(m+1) = w(m) - \eta\delta(m)x$
	• Cập nhật độ lệch $b(m+1) = b(m) - \eta\delta(m)$
	Kiểm tra hiệu năng của mạng sau khi huấn luyện: so sánh đầu ra mạng và đầu ra mong muốn.

Bảng 1.5. So sánh đầu ra mạng và đầu ra mong muốn với 4 mẫu thông tin đầu vào.

Đầu vào 1 (x_1)	Đầu vào 2 (x_2)	Đầu ra của mạng (z)	Đầu ra thực (t)
0	0	0,0550	0
0	1	0,9656	0
1	0	0,9656	1
1	1	0,9999	1

Trọng số liên kết nút vào thứ nhất với nút ra: $w_1 = 6,1793$.

Trọng số liên kết nút vào thứ hai với nút ra: $w_2 = 6,1794$.

Độ lệch của nút ra: $b = -2,8435$.

Hạn chế của mạng nơ ron một lớp là loại mạng nơ ron này chỉ phù hợp với bài toán hồi quy tuyến tính (hàm cần xấp xỉ là hàm bậc nhất) và bài toán phân loại tuyến tính (biên giới của các nhóm là tuyến tính). Do đó, đối với bài toán hồi quy và phân loại phi tuyến, chúng ta phải thiết kế các mạng nơ ron có ít nhất một lớp ẩn còn được gọi là mạng nơ ron đa lớp.

1.8. Huấn luyện mạng nơ ron đa lớp

Để khắc phục hạn chế của mạng nơ ron một lớp, người ta sử dụng mạng nơ ron đa lớp cho các bài toán hồi quy và phân loại phi tuyến. Khi đó, quá trình cập nhật các trọng số và độ lệch của mạng yêu cầu phải tính đạo riêng của hàm sai số với các trọng số và độ lệch. Quá trình này được thực hiện dựa trên quy tắc chuỗi của các đạo hàm riêng và dẫn tới một thuật toán trong đó các tín hiệu sai số được lan truyền ngược qua mạng bắt đầu từ đầu ra quay trở lại đầu vào.

1.8.1. Lan truyền ngược (Back-Propagation)

Tùy thuộc tích chất của bài toán, hàm sai số dữ liệu được định nghĩa khác nhau:

Đối với bài toán hồi quy, hàm sai số dữ liệu có dạng như sau:

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{z_k^n - t_k^n\right\}^2 \tag{1.16}$$

Ở đây E_D được gọi là hàm sai số dữ liệu tổng các bình phương và N là số mẫu của dữ liệu huấn luyện mạng. z_k^n là đầu ra thứ k của mạng ứng với mẫu dữ liệu huấn luyện thứ n. t_k^n là đầu ra mong muốn (mục tiêu) ứng với đầu ra thứ k và mẫu dữ liệu huấn luyện thứ n. Quá trình huấn luyện mạng rơ ron với dữ liệu đầu vào và đầu ra mong muốn được gọi là quá trình học có giám sát (supervised learning).

Đối với bài toán phân loại chỉ có hai nhóm, hàm sai số dữ liệu là hàm "cross-entropy" có dạng như sau:

$$E_D(w) = -\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{t_k^n \ln\left(z_k^n\right) + \left(1-t_k^n\right)\ln\left(1-z_k^n\right)\right\} \tag{1.17}$$

Đối với bài toán phân loại có nhiều hơn hai nhóm, hàm sai số dữ liệu là hàm "entropy" có dạng như sau:

$$E_D(w) = \sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n \ln\left(z_k^n\right) \tag{1.18}$$

Vấn đề huấn luyện mạng nơ ron có thể xem như là một quá trình xác định các trọng số và độ lệch để cực tiểu hóa hàm sai số dữ liệu E_D là một hàm phi tuyến với nhiều cực tiểu địa phương thỏa mãn điều kiện sau:

$$\nabla E_D(w) = 0 \tag{1.19}$$

Giá trị cực tiểu nhỏ nhất của hàm chi phí được gọi là cực tiểu toàn cục. Trong khi đó, các giá trị cực tiểu khác được gọi là các cực tiểu địa phương. Trong thực tế, việc xác định lời giải chính xác để cực tiểu hàm sai số là không thể. Thay vào đó, chúng ta xem xét các thuật toán tìm kiếm qua không gian của các trọng số và độ lệch với hàng loạt các bước lặp để cập nhật véc tơ các trọng số và các độ lệch như sau:

$$w(m+1) = w(m) + \alpha(m)\,d(m) \tag{1.20}$$

Trong phương trình (1.20), m chỉ thị cho bước lặp. $w(m)$ và $w(m+1)$ lần lượt là các véc tơ trọng số và độ lệch tại bước lặp thứ m và $m+1$. $d(m)$ và $\alpha(m)$ được gọi là hướng tìm kiếm (search direction) và độ lớn của bước (step size) tại bước lặp thứ m.

Việc xác định $d(m)$ và $\alpha(m)$ thể được thực hiện đơn giản với thuật toán suy giảm độ dốc (Gradient Descent). Khi đó, $\alpha(m) = \eta$ được gọi là tốc độ học có giá trị không đổi và $d(m)$

được chọn ngược chiều với chiều biến thiên của độ dốc của hàm sai số. Khi đó, các trọng số và độ lệch được cập nhật như sau:

$$w(m+1) = w(m) - \eta \frac{\partial E}{\partial w(m)} \tag{1.21}$$

Như vậy, quá trình huấn luyện mạng nơ ron yêu cầu phải tính các đạo hàm của hàm sai số theo trọng số và độ lệch tại các bước lặp sử dụng kỹ thuật lan truyền ngược sai số. Đối với cả ba dạng hàm sai số dữ liệu, đạo hàm riêng của sai số dữ liệu đối với kích hoạt đầu ra ứng với mẫu dữ liệu thứ n tại một bước lặp được tính như sau:

$$\frac{\partial E_D^n}{\partial a_k^{(2)n}} = z_k^n - t_k^n \tag{1.22}$$

Sai số ứng với mẫu dữ liệu thứ n được tính như sau:

$$\delta_k^{(2)n} = z_k^n - t_k^n \tag{1.23}$$

Đạo hàm riêng của E_D đối với trọng số liên kết từ lớp ẩn đến lớp ra được tính như sau:

$$\frac{\partial E_D}{\partial w_{kj}^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} y_j^n \tag{1.24}$$

Đạo hàm riêng của E_D đối với độ lệch của lớp ra được tính như sau:

$$\frac{\partial E_D}{\partial b_k^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} \tag{1.25}$$

Để tính đạo hàm riêng của E_D đối với trọng số liên kết lớp đầu vào và lớp ẩn, sai số $\delta_k^{(2)n}$ cần được lan truyền ngược qua lớp ra để thu được sai số cho các nút ẩn. Phương trình lan truyền ngược có dạng như sau:

$$\delta_j^{(1)n} = \frac{\partial f_j^{(1)}}{\partial a_j^{(1)n}} \sum_{k=1}^{c} w_{kj}^{(2)} \delta_k^{(2)n} \tag{1.26}$$

Nếu hàm kích hoạt của lớp ẩn là hàm tanh, phương trình (1.26) có dạng như sau:

$$\delta_j^{(1)n} = \left[1 - \left(f_j^{(1)}\right)^2\right] \sum_{k=1}^{c} w_{kj}^{(2)} \delta_k^{(2)n} \tag{1.27}$$

Đạo hàm riêng của hàm sai số E_D đối với trọng số liên kết nút vào và nút ẩn được tính như sau:

$$\frac{\partial E_D}{\partial w_{ji}^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n} x_i^n \tag{1.28}$$

Đạo hàm riêng của hàm sai số E_D đối với độ lệch của nút ẩn được tính như sau:

$$\frac{\partial E_D}{\partial b_j^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n} \tag{1.29}$$

Cuối cùng, các trọng số được cập nhật tại các bước lặp như sau:

$$w_{kj}^{(2)}(m+1) = w_{kj}^{(2)}(m) - \eta \frac{\partial E}{\partial w_{kj}^{(2)}(m)} \tag{1.30}$$

$$b_{k}^{(2)}(m+1) = b_{k}^{(2)}(m) - \eta \frac{\partial E}{\partial b_{k}^{(2)}(m)} \tag{1.31}$$

$$w_{ji}^{(1)}(m+1) = w_{ji}^{(1)}(m) - \eta \frac{\partial E}{w_{ji}^{(1)}(m)} \tag{1.32}$$

$$b_{j}^{(1)}(m+1) = b_{j}^{(1)}(m) - \eta \frac{\partial E}{\partial b_{j}^{(1)}(m)} \tag{1.33}$$

Ví dụ 1.3:

Thiết kế mạng nơ ron có một lớp ẩn để xấp xỉ hàm phi tuyến có dạng như sau:

$$y = \frac{1}{4}\left(x^3 + 3x - 6x - 8\right)$$

$$-6 \le x \le 4$$

Khi có nhiễu, phương trình trên có dạng như sau:

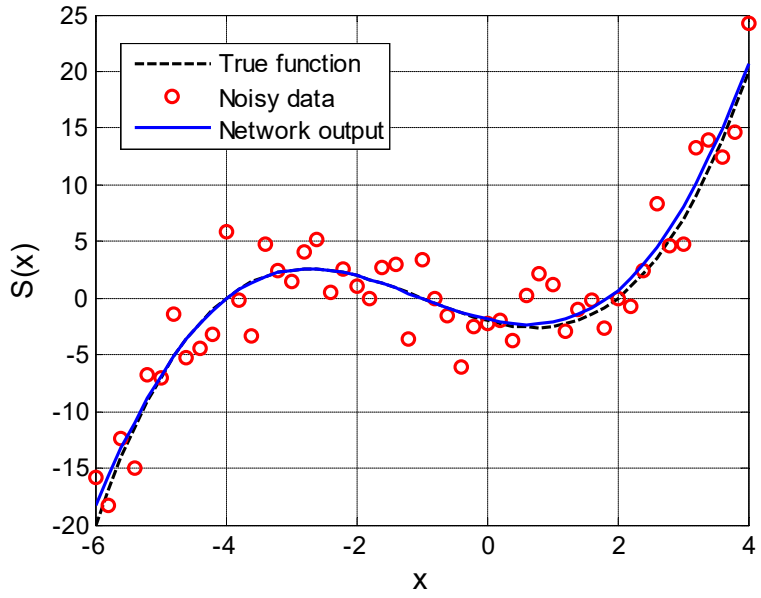
$$t = y + \delta$$

Bảng 1.6 là code MATLAB thiết kế mạng nơ ron đa lớp để xấp xỉ hàm phi tuyến. Hình 1.6 là đồ thị hàm phi tuyến, dữ liệu nhiễu và đầu ra của mạng.

Bảng 1.6. Code MATLAB thiết kế mạng nơ ron đa lớp trong ví dụ 1.3.

Cú pháp MATLAB	Chức năng
x = -6:0.2:4;	Khoảng giá trị khảo sát của biến từ -6 đến 4 với độ phân giải là 0,2.
N = length(x);	Xác định số lượng các giá của biến.
y_true = 1/4*(x.^3 + 3*x.^2 - 6*x - 8);	Tính các giá trị của hàm.
sigma = 3*randn(1,N);	Các giá trị nhiễu có phân bố Normal.
t = y_true + sigma;	Tính các giá trị của hàm với nhiễu.
plot(x,y_true,'k--','linewidth',1.5);	Vẽ đồ thị của hàm và các điểm của hàm với nhiễu.
grid on;	
hold on;	
plot(x,t,'ro','linewidth',1.5);	
grid on;	
xlabel('x','FontName','Arial','FontSize',12);	
ylabel('S(x)','FontName','Arial','FontSize',12);	
hold on;	
max_x = max(x);	
min_x = min(x);	Chuẩn hóa dữ liện đầu vào và đầu ra của hàm bằng cách ánh xạ dải giá trị đầu vào và đầu ra thành các giá trị nằm trong khoảng [0,1].
max_t = max(t);	
min_t = min(t);	
x1 = (x - min_x)/(max_x - min_x);	
t1 = (t - min_t)/(max_t - min_t);	
nin = size(x,1);	Xác định số nút theo dữ liệu.
nhidden = 5;	Số nút ẩn là 5.
nout = 1;	Số nút ra là 1.

w1 = randn(nhidden,nin)/sqrt(nin + 1);
b1 = randn(nhidden,1)/sqrt(nin + 1);
w2 = randn(nout,nhidden)/sqrt(nhidden + 1);
b2 = randn(nout,1)/sqrt(nhidden + 1);
M = 100000;
nu = 0.005;
for i = 1:M
  a1 = w1*x1 + b1*ones(1,N);
  y = tanh(a1);
  a2 = w2*y + b2*ones(1,N);
  z1 = a2;
  delta_2 = z1 - t1;
  dEdw2 = delta_2*y';
  dEdb2 = delta_2*ones(N,1);
  g1 = 1 - y.^2;
  delta_1 = g1.*(w2'*delta_2);
  dEdw1 = delta_1*x1';
  dEdb1 = delta_1*ones(N,1);
  w1 = w1 - nu*dEdw1;
  b1 = b1 - nu*dEdb1;
  w2 = w2 - nu*dEdw2;
  b2 = b2 - nu*dEdb2;
end
a1 = w1*x1 + b1*ones(1,N);
y = tanh(a1);
a2 = w2*y + b2*ones(1,N);
z = a2;
z1 = z*(max_t - min_t) + min_t;
plot(x,z1,'b-','linewidth',1.5)
hold on;
legend('True function','Noisy data', 'Network output');
	Khởi tạo trọng số và độ lệch.
	Đặt số bước lặp $M = 100000$.
	Đặt tốc độ học $\eta = 0,005$.
	Thực hiện quá trình lặp để cập nhật trọng số và độ lệch.
	
	Kiểm tra hiệu năng của mạng sau khi huấn luyện qua quan sát đồ thị của hàm, giá trị của hàm với nhiễu và đầu ra của mạng nơ ron với dữ liệu đầu vào.



Hình 1.6. Đồ thị hàm phi tuyến, dữ liệu nhiễu và đầu ra của mạng trong ví dụ 1.3.

Ví dụ 1.4:

Thiết kế mạng nơ ron đa lớp dùng để phân loại hai đầu vào của một cổng logic hoặc loại trừ (EXOR) có bảng chân lý như bảng 1.6. Bảng 1.7 là code MATLAB thiết kế mạng nơ ron một lớp ẩn.

Bảng 1.6. Bảng chân lý của cổng logic hai đầu vào trong ví dụ 1.4.

Đầu vào 1 (x_1)	Đầu vào 2 (x_2)	Đầu ra (t)
0	0	0
0	1	1
1	0	1
1	1	0

Bảng 1.7. Code MATLAB xây dựng mạng nơ ron đa lớp trong ví dụ 1.4.

Cú pháp MATLAB	**Chức năng**

x = [0 0; 0 1; 1 0; 1 1];
t = [0 1 1 0]';
N = size(x,1);
nin = size(x,2);
nhidden = 3;
nout = 1;
w1 = randn(nhidden,nin)/sqrt(nin + 1);
b1 = randn(nhidden,1)/sqrt(nin + 1);
w2 = randn(nout,nhidden)/sqrt(nhidden + 1);
b2 = randn(nout,1)/sqrt(nhidden + 1);
M = 2000;
nu = 0.1;
for i = 1:M
  a1 = w1*x' + b1*ones(1,N);
	Nhập dữ liệu để huấn luyện mạng bao gồm đầu vào và đầu ra mong muốn.
	Xác định số mẫu của dữ liệu huấn luyện mạng.
	Số nút vào 2, số nút ẩn là 3 và số nút ra là 1
	Khởi tạo các trọng số liên kết các nút vào và các nút ẩn, khởi tạo độ lệch của các nút ẩn, khởi tạo các trọng số liên kết các nút ẩn và nút ra. Khởi tạo độ lệch của nút ra.
	Thực hiện quá trình lặp để cập nhật các trọng số và độ lệch với số bước lặp tối đa $M = 2000$ và tốc độ học $\eta = 0{,}1$.

y = tanh(a1);
a2 = w2*y + b2*ones(1,N);
z = 1./(1 + exp(-a2));
delta_2 = z - t';
dEdw2 = delta_2*y';
dEdb2 = delta_2*ones(N,1);
g1 = 1 - y.^2;
delta_1 = g1.*(w2'*delta_2);
dEdw1 = delta_1*x;
dEdb1 = delta_1*ones(N,1);
w1 = w1 - nu*dEdw1;
b1 = b1 - nu*dEdb1;
w2 = w2 - nu*dEdw2;
b2 = b2 - nu*dEdb2;
end
a1 = w1*x' + b1*ones(1,N);
y = tanh(a1);
a2 = w2*y + b2*ones(1,N);
z = 1./(1 + exp(-a2));
result = [z' t]
	So sánh đầu ra của mạng và đầu ra mong muốn với dữ liệu huấn luyện mạng.

Bảng 1.8. So sánh đầu ra mạng và đầu ra mong muốn trong ví dụ 1.4.

Đầu vào 1 (x_1)	Đầu vào 2 (x_2)	Đầu ra của mạng (z)	Đầu ra mong muốn (t)
0	0	0,0011	0
0	1	0,9995	1
1	0	0,9985	1
1	1	0,0010	0

1.9. Huấn luyện mạng nơ ron sâu

Để thuận tiện, chúng ta xét đến một mạng nơ ron sâu với 2 lớp ẩn. Như vậy, trọng số và độ lệch theo các lớp của mạng bao gồm:

- Trọng số và độ lệch liên kết giữa lớp vào và lớp ẩn thứ nhất: $w_{ji}^{(1)}$ và $b_j^{(1)}$.
- Trọng số và độ lệch liên kết giữa lớp ẩn thứ nhất và và lớp ẩn thứ hai: $w_{kj}^{(2)}$ và $b_k^{(2)}$.
- Trọng số và độ lệch liên kết giữa lớp ẩn thứ hai và và lớp ra: $w_{ml}^{(3)}$ và $b_m^{(3)}$.

Hàm kích hoạt của các nút ẩn là hàm tanh. Do đó quá trình lan truyền hướng thẳng có dạng như sau. Đầu ra của các nút của lớp ẩn thứ nhất có dạng như sau:

$$y_j^{(1)n} = f_1\left(w_{ji}^{(1)n}x + b_j^{(1)n}\right) \tag{1.34}$$

Đầu ra của các nút của lớp ẩn thứ nhất đến lớp ẩn thứ hai có dạng như sau:

$$y_k^{(2)n} = f_2\left(w_{kj}^{(2)n}y_j^{(2)n} + b_k^{(2)n}\right) \tag{1.35}$$

Đầu ra của các nút của lớp ẩn thứ hai đến lớp ra có dạng như sau:

$$z_l^n = f_3\left(w_{lk}^{(3)n}y_k^{(2)n} + b_l^{(2)n}\right) \tag{1.36}$$

Các hàm kích hoạt của các nút ẩn $f_1(.)$ và $f_2(.)$ là hàm tan hyperbolic (tanh). Hàm kích hoạt nút ra $f_3(.)$ có dạng tùy thuộc bài toán hồi quy hay phân loại.

Tương tự như mạng nơ ron đa lớp, quá trình huấn luyện mạng nơ ron sâu cũng dựa trên kỹ thuật lan truyền ngược sai số. Đạo hàm riêng của sai số dữ liệu E_D đối với kích hoạt đầu ra ứng với mẫu dữ liệu thứ n tại một bước lặp được tính như sau:

$$\frac{\partial E_D^n}{\partial a_l^{(3)n}} = z_l^n - t_l^n \tag{1.37}$$

Sai số đầu ra của mạng ứng với mẫu dữ liệu thứ n được tính như sau:

$$\delta_l^{(3)n} = z_l^n - t_l^n \tag{1.38}$$

Đạo hàm riêng của E_D đối với trọng số liên kết từ lớp ẩn thứ hai đến lớp đầu ra được tính như sau:

$$\frac{\partial E_D}{\partial w_{lk}^{(3)}} = \sum_{n=1}^{N} \delta_l^{(3)n} y_k^{(2)n} \tag{1.39}$$

Đạo hàm riêng của E_D đối với độ lệch của lớp ra được tính như sau:

$$\frac{\partial E_D}{\partial b_l^{(3)}} = \sum_{n=1}^{N} \delta_l^{(3)n} \tag{1.40}$$

Để tính đạo hàm riêng của E_D đối với trọng số liên kết lớp ẩn thứ nhất và lớp ẩn thứ hai, sai số $\delta_l^{(3)n}$ cần được lan truyền ngược qua lớp ra để thu được sai số cho các nút của lớp ẩn thứ hai (có số nút ẩn là M_2). Khi đó phương trình lan truyền ngược có dạng như sau:

$$\delta_k^{(2)n} = \left(1 - \left(y_k^{(2)n}\right)^2\right) \sum_{l=1}^{M_2} w_{lk}^{(3)} \delta_l^{(3)n} \tag{1.41}$$

Đạo hàm riêng của hàm sai số đối với trọng số liên kết nút vào và nút ẩn được tính như sau:

$$\frac{\partial E_D}{\partial w_{kj}^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} y_k^{(1)n} \tag{1.42}$$

Đạo hàm riêng của hàm sai số E_D đối với độ lệch của nút ẩn được tính như sau:

$$\frac{\partial E_D}{\partial w_k^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} \tag{1.43}$$

$\delta_k^{(2)n}$ được lan truyền ngược qua lớp ẩn thứ hai để thu được sai số cho lớp ẩn thứ nhất có dạng như sau:

$$\delta_j^{(1)n} = \left(1 - \left(y_j^{(1)n}\right)^2\right) \sum_{n=1}^{N} w_{kj}^{(2)} \delta_k^{(2)n} \tag{1.44}$$

Đạo hàm riêng của hàm sai số E_D đối với trọng số liên kết nút vào và nút ẩn thứ nhất được tính như sau:

$$\frac{\partial E_D}{\partial w_{ji}^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n} x_i^n \tag{1.45}$$

Cuối cùng, các trọng số và độ lệch được cập nhật tại các bước lặp $m+1$ theo bước lặp m như sau:

$$w_{lk}^{(3)}(m+1) = w_{lk}^{(3)}(m) - \eta \frac{\partial E_D}{\partial w_{lk}^{(3)}(m)} \tag{1.46}$$

$$b_l^{(3)}(m+1) = b_l^{(3)}(m) - \eta \frac{\partial E_D}{\partial b_l^{(3)}(m)} \tag{1.47}$$

$$w_{kj}^{(2)}(m+1) = w_{kj}^{(2)}(m) - \eta \frac{\partial E_D}{\partial w_{kj}^{(2)}(m)} \tag{1.48}$$

$$b_k^{(2)}(m+1) = b_k^{(2)}(m) - \eta \frac{\partial E_D}{\partial b_k^{(2)}(m)} \tag{1.49}$$

$$w_{ji}^{(1)}(m+1) = w_{ji}^{(1)}(m) - \eta \frac{\partial E_D}{w_{ji}^{(1)}(m)} \tag{1.50}$$

$$b_j^{(1)}(m+1) = b_j^{(1)}(m) - \eta \frac{\partial E_D}{\partial b_j^{(1)}(m)} \tag{1.51}$$

Ví dụ 1.5:

Thiết kế mạng nơ ron sâu có hai lớp ẩn để xấp xỉ hàm phi tuyến có dạng như sau:

$$y = \frac{1}{4}\left(x^3 + 3x - 6x - 8\right)$$

$$-6 \le x \le 4$$

Khi có nhiễu, phương trình trên có dạng như sau:

$$t = y + \delta$$

Bảng 1.9 là code MATLAB thiết kế mạng nơ ron sâu hai lớp ẩn.

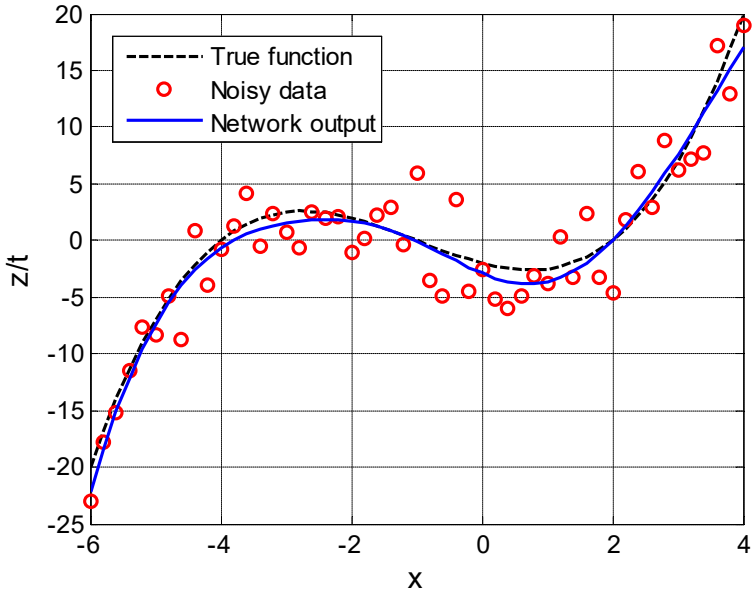
Bảng 1.9. Code MATLAB thiết kế mạng nơ ron sâu hai lớp ẩn trong ví dụ 1.5.

Cú pháp MATLAB	Chức năng
clc; clear; x = -6:0.2:4;	Khoảng giá trị khảo sát của biến từ -6 đến 4 với độ phân giải là 0,2.
N = length(x);	Xác định số lượng các giá của biến.
y_true = 1/4*(x.^3 + 3*x.^2 - 6*x - 8);	Tính các giá trị của hàm.
sigma = 3*randn(1,N);	Các giá trị nhiễu có phân bố Normal.
t = y_true + sigma;	Tính các giá trị của hàm với nhiễu.

plot(x,y_true,'k--','linewidth',1.5);
hold on;
plot(x,t,'ro','linewidth',1.5);
grid on;
xlabel('x','FontName','Arial','FontSize',12);
	Vẽ đồ thị của hàm và các điểm của hàm với nhiễu.

ylabel('z/t','FontName','Arial','FontSize',12);
hold on;
max_x = max(x);
min_x = min(x);
max_t = max(t);
min_t = min(t);
x1 = (x - min_x)/(max_x - min_x);
t1 = (t - min_t)/(max_t - min_t);
N = size(x,2);
nin = size(x,1);
nhidden_1 = 5;
nhidden_2 = 5;
nout = 1;
w1 = randn(nhidden_1,nin)/sqrt(nin + 1);
b1 = randn(nhidden_1,1)/sqrt(nin + 1);
w2 = randn(nhidden_2,nhidden_1)/sqrt(nhidden_1 + 1);
b2 = randn(nhidden_2,1)/sqrt(nhidden_1 + 1);
w3 = randn(nout,nhidden_2)/sqrt(nhidden_2 + 1);
b3 = randn(nout,1)/sqrt(nhidden_2 + 1);
M = 100000;
nu = 0.001;
for i = 1:M
  a1 = w1*x1 + b1*ones(1,N);
  y1 = tanh(a1);
  a2 = w2*y1 + b2*ones(1,N);
  y2 = tanh(a2);
  a3 = w3*y2 + b3*ones(1,N);
  z = a3;
  delta_3 = z - t1;
  dEdw3 = delta_3*y2';
  dEdb3 = delta_3*ones(N,1);
  g2 = 1 - y2.^2;
  delta_2 = g2.*(w3'*delta_3);
  dEdw2 = delta_2*y1';
  dEdb2 = delta_2*ones(N,1);
  g1 = 1 - y1.^2;
  delta_1 = g1.*(w2'*delta_2);
  dEdw1 = delta_1*x';
  dEdb1 = delta_1*ones(N,1);
  w1 = w1 - nu*dEdw1;
  b1 = b1 - nu*dEdb1;
  w2 = w2 - nu*dEdw2;
  b2 = b2 - nu*dEdb2;
  w3 = w3 - nu*dEdw3;
  b3 = b3 - nu*dEdb3;
end
a1 = w1*x1 + b1*ones(1,N);
y1 = tanh(a1);
a2 = w2*y1 + b2*ones(1,N);
y2 = tanh(a2);
	Chuẩn hóa dữ liện đầu vào và đầu ra của hàm bằng cách ánh xạ dải giá trị đầu vào và đầu ra thành các giá trị nằm trong khoảng [0,1].
	
	Xác định số nút theo dữ liệu.
	Số nút lớp ẩn thứ nhất là 5.
	Số nút lớp ẩn thứ hai là 5.
	Số nút ra là 1.
	Khởi tạo trọng số và độ lệch.
	Đặt số bước lặp $M = 100000$.
	Đặt tốc độ học $\eta = 0,001$.
	Thực hiện quá trình lặp để cập nhật trọng số và độ lệch theo kỹ thuật lan truyền ngược sai số.
	
	Kiểm tra hiệu năng của mạng sau khi huấn luyện qua quan sát đồ thị của hàm, giá trị của hàm

a3 = w3*y2 + b3*ones(1,N);
z = a3;
z1 = z*(max_t - min_t) + min_t;
plot(x,z1,'b-','linewidth',1.5);
hold on;
legend('True function','Noisy data', 'Network output');
	với nhiễu và đầu ra của mạng nơ ron với dữ liệu đầu vào.



Hình 1.7. Đồ thị hàm phi tuyến, dữ liệu nhiễu và đầu ra của mạng trong ví dụ 1.5.

Ví dụ 1.6:

Thiết kế mạng nơ ron sâu với hai lớp ẩn dùng để phân loại hai đầu vào của một cổng logic hoặc loại trừ (EXOR) có bảng chân lý như bảng 1.10. Bảng 1.11 là code MATLAB thiết kế mạng nơ ron một lớp ẩn.

Bảng 1.10. Bảng chân lý của cổng logic hai đầu vào trong ví dụ 1.6.

Đầu vào 1 (x_1)	Đầu vào 2 (x_2)	Đầu ra (t)
0	0	0
0	1	1
1	0	1
1	1	0

Bảng 1.11. Code MATLAB thiết kế mạng nơ ron sâu hai lớp ẩn trong ví dụ 1.6.

Cú pháp MATLAB	**Chức năng**

clc;
clear;
x = [0 0; 0 1; 1 0; 1 1];
t = [0 1 1 0]';
N = size(x,1);
nin = size(x,2);
nhidden_1 = 5;
nhidden_2 = 5;
nout = 1;
	Nhập dữ liệu để huấn luyện mạng bao gồm đầu vào và đầu ra mong muốn.
	Xác định số mẫu của dữ liệu huấn luyện mạng.
	Số nút lớp ẩn thứ nhất là 5.
	Số nút lớp ẩn thứ hai là 5.
	Số nút ra là 1.

w1 = randn(nhidden_1,nin)/sqrt(nin + 1);
b1 = randn(nhidden_1,1)/sqrt(nin + 1);
w2 = randn(nhidden_2,nhidden_1)/sqrt(nhidden_1 + 1);
b2 = randn(nhidden_2,1)/sqrt(nhidden_1 + 1);
w3 = randn(nout,nhidden_2)/sqrt(nhidden_2 + 1);
b3 = randn(nout,1)/sqrt(nhidden_2 + 1);
	Khởi tạo các trọng số liên kết các nút vào và các nút ẩn, khởi tạo độ lệch của các nút ẩn, khởi tạo các trọng số liên kết các nút ẩn và nút ra. Khởi tạo độ lệch của nút ra.

M = 100000;
nu = 0.01;
for i = 1:M
a1 = w1*x' + b1*ones(1,N);
y1 = tanh(a1);
a2 = w2*y1 + b2*ones(1,N);
y2 = tanh(a2);
a3 = w3*y2 + b3*ones(1,N);
z = 1./(1 + exp(-a3));
delta_3 = z - t';
dEdw3 = delta_3*y2';
dEdb3 = delta_3*ones(N,1);
g2 = 1 - y2.^2;
delta_2 = g2.*(w3'*delta_3);
dEdw2 = delta_2*y1';
dEdb2 = delta_2*ones(N,1);
g1 = 1 - y1.^2;
delta_1 = g1.*(w2'*delta_2);
dEdw1 = delta_1*x;
dEdb1 = delta_1*ones(N,1);
w1 = w1 - nu*dEdw1;
b1 = b1 - nu*dEdb1;
w2 = w2 - nu*dEdw2;
b2 = b2 - nu*dEdb2;
w3 = w3 - nu*dEdw3;
b3 = b3 - nu*dEdb3;
end
	Thực hiện quá trình lặp để cập nhật các trọng số và độ lệch với số bước lặp tối đa $M = 100000$ và tốc độ học $\eta = 0,01$.

a1 = w1*x' + b1*ones(1,N);
y1 = tanh(a1);
a2 = w2*y1 + b2*ones(1,N);
y2 = tanh(a2);
a3 = w3*y2 + b3*ones(1,N);
z = 1./(1 + exp(-a3));
result = [z' t]
	Kiểm tra hiệu năng của mạng sau khi ứng dụng bằng cách so sánh đầu ra của mạng và đầu ra mong muốn với dữ liệu đầu vào.

Bảng 1.12. So sánh đầu ra mạng và đầu ra mong muốn trong ví dụ 1.6.

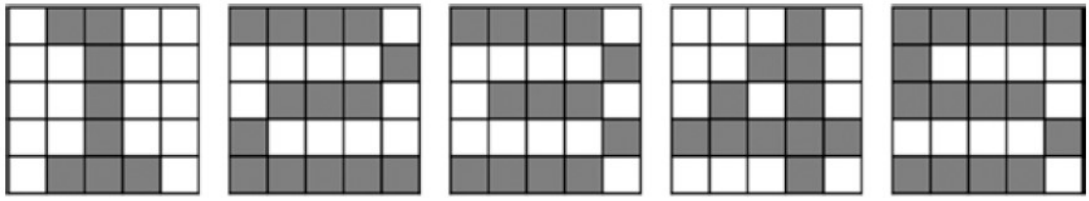
Đầu vào 1 (x_1)	Đầu vào 2 (x_2)	Đầu ra của mạng (z)	Đầu ra mong muốn (t)
0	0	0,0001	0
0	1	0,9998	1
1	0	0,9998	1
1	1	0,0001	0

1.10. Huấn luyện mạng nơ ron cho bài toán phân loại nhiều hơn hai nhóm

Trong phần này ta xem xét đến ứng dụng của mạng nơ ron cho bài toán phân loại nhiều hơn hai nhóm sử dụng mạng nơ ron đa lớp và mạng nơ ron sâu.

Ví dụ 1.7:

Thiết kế mạng nơ ron một lớp ẩn dùng để phân loại số thứ tự từ số từ 1 đến 5. Hình 1.8 là các hình vuông 5 x 5 phần tử hiển thị năm số từ 1 đến 5.



Hình 1.8. Các hình vuông 5 x 5 ô vuông hiển thị năm số từ 1 đến 5.

Theo hình 1.8, các số được biểu diễn theo thông tin 2 chiều của một hình vuông 5 x 5 ô vuông. Thông tin này cần thiết chuyển thành thông tin 1 chiều sau đó đưa đến đầu vào của mạng nơ ron nhân tạo. Do đó, thông tin một chiều của các số có dạng bao gồm 25 phần tử được mã hóa theo nguyên tắc cuối của một cột sẽ nối tiếp bởi đầu của các cột sau của mỗi hình vuông. Các ô màu đen được mã hóa là 1, trong khi các ô màu trắng được mã hóa là 0. Do bài toán có 5 nhóm nên đầu ra của mạng sẽ là 5 nút. Các đầu ra ứng với các từng dữ liệu đầu của mỗi số sẽ là một véc tơ như sau:

$$t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Bảng 1.13. Code MATLAB thiết kế mạng nơ ron một lớp ẩn trong ví dụ 1.7.

Cú pháp MATLAB	Chức năng
X1 = [0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0]; X2 = [1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1 1 1 1 1]; X3 = [1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0]; X4 = [0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1	Các hình vuông hiển thị 5 chữ số từ 1 đến 5.

0 0 0 1 0]; X5 = [1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0];	

x1 = reshape(X1(:, :, 1), 1, 25);
x2 = reshape(X2(:, :, 1), 1, 25);
x3 = reshape(X3(:, :, 1), 1, 25);
x4 = reshape(X4(:, :, 1), 1, 25);
x5 = reshape(X5(:, :, 1), 1, 25);
x = [x1;x2;x3;x4;x5];
	Chuyển dữ liệu 2D của 5 số thành dữ liệu 1 D.

t = [1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1];	Đầu ra mong muốn (mục tiêu) ứng với 5 số thứ tự từ 1 đến 5.
N = size(x,1);	Số nút vào là 25.
nin = size(x,2);	Số nút ẩn là 3.
nhidden = 3;	Số nút đầu ra là 5.
nout = 5;	
w1 = randn(nhidden,nin)/sqrt(nin + 1); b1 = randn(nhidden,1)/sqrt(nin + 1);	Khởi tạo giá trị các trọng số và độ lệch từ lớp vào đến lớp ẩn.
w2 = randn(nout,nhidden)/sqrt(nhidden + 1); b2 = randn(nout,1)/sqrt(nhidden + 1);	Khởi tạo giá trị các trọng số và độ lệch từ lớp ẩn đến lớp ra.
M = 1000;	Số bước lặp $M = 5000$.
nu = 0.1;	Tốc độ học $\eta = 0{,}1$.

for i = 1:M
a1 = w1*x' + b1*ones(1,N);
y = tanh(a1);
a2 = w2*y + b2*ones(1,N);
z1 = exp(a2(:,1))/sum(exp(a2(:,1)));
z2 = exp(a2(:,2))/sum(exp(a2(:,2)));
z3 = exp(a2(:,3))/sum(exp(a2(:,3)));
z4 = exp(a2(:,4))/sum(exp(a2(:,4)));
z5 = exp(a2(:,5))/sum(exp(a2(:,5)));
z = [z1 z2 z3 z4 z5];
delta_2 = z - t;
dEdw2 = delta_2*y';
dEdb2 = delta_2*ones(N,1);
g1 = 1 - y.^2;
delta_1 = g1.*(w2'*delta_2);
dEdw1 = delta_1*x;
dEdb1 = delta_1*ones(N,1);
w1 = w1 - nu*dEdw1;
b1 = b1 - nu*dEdb1;
w2 = w2 - nu*dEdw2;
b2 = b2 - nu*dEdb2;
end
	Cập nhật các trọng số và độ lệch theo kĩ thuật lan truyền ngược sai số.

a1 = w1*x' + b1*ones(1,N);	

y = tanh(a1);
a2 = w2*y + b2*ones(1,N);
z1 = exp(a2(:,1))/sum(exp(a2(:,1)));
z2 = exp(a2(:,2))/sum(exp(a2(:,2)));
z3 = exp(a2(:,3))/sum(exp(a2(:,3)));
z4 = exp(a2(:,4))/sum(exp(a2(:,4)));
z5 = exp(a2(:,5))/sum(exp(a2(:,5)));
z = [z1 z2 z3 z4 z5]
	Kiểm tra hiệu năng của mạng sau khi huấn luyện với dữ liệu đầu vào.

Bảng 1.14. Đầu ra của mạng nơ ron trong ví dụ 1.7.

z_1	z_2	z_3	z_4	z_5
0,9995	0,0000	0,0002	0,0002	0,0000
0,0000	**0,9996**	0,0002	0,0001	0,0001
0,0002	0,0002	**0,9995**	0,0000	0,0002
0,0002	0,0001	0,0000	**0,9996**	0,0000
0,0001	0,0000	0,0001	0,0001	**0,9997**

Ví dụ 1.8:

Thiết kế mạng nơ ron hai lớp ẩn dùng để phân loại số thứ tự từ số từ 1 đến 5 với thông số như ví dụ 1.7.

Bảng 1.15. Code MATLAB thiết kế mạng nơ ron hai lớp ẩn trong ví dụ 1.8.

Cú pháp MATLAB	Chức năng
clc; clear; X1 = [0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0]; X2 = [1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1 1 1 1 1]; X3 = [1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0]; X4 = [0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0]; X5 = [1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0];	Các hình vuông hiển thị 5 chữ số từ 1 đến 5.

x1 = reshape(X1(:, :, 1), 1, 25);
x2 = reshape(X2(:, :, 1), 1, 25);
x3 = reshape(X3(:, :, 1), 1, 25);
x4 = reshape(X4(:, :, 1), 1, 25);
x5 = reshape(X5(:, :, 1), 1, 25);
x = [x1;x2;x3;x4;x5];
	Chuyển dữ liệu 2D của 5 số thành dữ liệu 1 D.

t = [1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1];	Đầu ra mong muốn (mục tiêu) ứng với 5 số thứ tự từ 1 đến 5.

N = size(x,1);
nin = size(x,2);
nhidden_1 = 3;
nhidden_2 = 3;
nout = 5;
	Số nút vào là 25.
	Số nút ẩn là 3.
	Số nút đầu ra là 5.

w1 = randn(nhidden_1,nin)/sqrt(nin + 1);
b1 = randn(nhidden_1,1)/sqrt(nin + 1);
w2 = randn(nhidden_2,nhidden_1)/sqrt(nhidden_1 + 1);
b2 = randn(nhidden_2,1)/sqrt(nhidden_1 + 1);
w3 = randn(nout,nhidden_2)/sqrt(nhidden_2 + 1);
b3 = randn(nout,1)/sqrt(nhidden_2 + 1);
	Khởi tạo giá trị các trọng số và độ lệch từ lớp vào đến lớp ẩn.
	Khởi tạo giá trị các trọng số và độ lệch từ lớp ẩn đến lớp ra.

M = 1000; nu = 0.1;	Số bước lặp $M = 5000$. Tốc độ học $\eta = 0{,}1$.

for i = 1:M
a1 = w1*x' + b1*ones(1,N);
y1 = tanh(a1);
a2 = w2*y1 + b2*ones(1,N);
y2 = tanh(a2);
a3 = w3*y2 + b3*ones(1,N);
z1 = exp(a3(:,1))/sum(exp(a3(:,1)));
z2 = exp(a3(:,2))/sum(exp(a3(:,2)));
z3 = exp(a3(:,3))/sum(exp(a3(:,3)));
z4 = exp(a3(:,4))/sum(exp(a3(:,4)));
z5 = exp(a3(:,5))/sum(exp(a3(:,5)));
z = [z1 z2 z3 z4 z5];
delta_3 = z - t;
dEdw3 = delta_3*y2';
dEdb3 = delta_3*ones(N,1);
g2 = 1 - y2.^2;
delta_2 = g2.*(w3'*delta_3);
dEdw2 = delta_2*y1';
dEdb2 = delta_2*ones(N,1);
g1 = 1 - y1.^2;
delta_1 = g1.*(w2'*delta_2);
dEdw1 = delta_1*x;
dEdb1 = delta_1*ones(N,1);
w1 = w1 - nu*dEdw1;
b1 = b1 - nu*dEdb1;
w2 = w2 - nu*dEdw2;
	Cập nhật các trọng số và độ lệch theo kĩ thuật lan truyền ngược sai số.
	Kiểm tra hiệu năng của mạng sau khi huấn luyện với dữ liệu đầu vào.

```
    b2 = b2 - nu*dEdb2;
    w3 = w3 - nu*dEdw3;
    b3 = b3 - nu*dEdb3;
end
a1 = w1*x' + b1*ones(1,N);
y1 = tanh(a1);
a2 = w2*y1 + b2*ones(1,N);
y2 = tanh(a2);
a3 = w3*y2 + b3*ones(1,N);
z1 = exp(a3(:,1))/sum(exp(a3(:,1)));
z2 = exp(a3(:,2))/sum(exp(a3(:,2)));
z3 = exp(a3(:,3))/sum(exp(a3(:,3)));
z4 = exp(a3(:,4))/sum(exp(a3(:,4)));
z5 = exp(a3(:,5))/sum(exp(a3(:,5)));
z = [z1 z2 z3 z4 z5]
```

Bảng 1.16. Đầu ra của mạng nơ ron hai lớp ẩn trong ví dụ 1.8.

z_1	z_2	z_3	z_4	z_5
0,9965	0,0007	0,0000	0,0007	0,0019
0,0003	**0,9951**	0,0021	0,0005	0,0022
0,0000	0,0015	**0,9958**	0,0024	0,0003
0,0008	0,0009	0,0018	**0,9965**	0,0000
0,0024	0,0017	0,0003	0,0000	**0,9956**