

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



BÁO CÁO THỰC TẬP TỐT NGHIỆP

Ứng dụng mạng Neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian

GVHD: GS.TS Dương Tuấn Anh

---o0o---

SVTH 1: Đoàn Ngọc Bảo	50800107
SVTH 2: Đinh Kim Ngân	50801336
SVTH 3: Trần Thế Sĩ	50801793
SVTH 4: Ngô Duy Khánh Vy	50802706

TP. HỒ CHÍ MINH, 6/2012

MỤC LỤC

DANH MỤC HÌNH.....	v
Chương 1. GIỚI THIỆU.....	1
1.1. ĐẶT VẤN ĐỀ	1
1.2. MỤC TIÊU CỦA ĐỀ TÀI.....	2
1.3. CẤU TRÚC BÁO CÁO	2
Chương 2. MẠNG NEURON NHÂN TẠO: CẤU TRÚC, NGUYÊN TẮC HOẠT ĐỘNG VÀ CÁC GIẢI THUẬT HUẤN LUYỆN	3
2.1. SƠ LƯỢC VỀ MẠNG NEURON NHÂN TẠO	3
2.2. CẤU TRÚC VỀ MẠNG NEURON NHÂN TẠO	4
2.3. NGUYÊN TẮC HOẠT ĐỘNG VÀ CÁC GIẢI THUẬT HUẤN LUYỆN CỦA MẠNG NEURON NHÂN TẠO	8
2.3.1. Perceptron.....	8
2.3.2. Mạng nhiều lớp và giải thuật lan truyền ngược.....	14
2.3.3. Giải thuật RPROP	19
2.3.4. Hiện tượng quá khớp	23
Chương 3. ỨNG DỤNG MẠNG NEURON NHÂN TẠO VÀO CÔNG TÁC DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN	24
3.1. DỮ LIỆU CHUỖI THỜI GIAN	24
3.1.1. Dữ liệu chuỗi thời gian là gì.....	24
3.1.2. Các thành phần của dữ liệu chuỗi thời gian	25
3.2. Áp dụng mạng Neuron vào dự báo dữ liệu chuỗi thời gian.....	29

3.3.	Các bước xây dựng một mô hình mạng neuron để dự báo dữ liệu chuỗi thời gian	30
3.3.1.	Lựa chọn các biến.....	31
3.3.2.	Thu thập dữ liệu.....	31
3.3.3.	Tiền xử lý dữ liệu	32
3.3.4.	Phân chia tập dữ liệu	32
3.3.5.	Xây dựng cấu trúc mạng.....	34
3.3.6.	Xác định tiêu chuẩn đánh giá	36
3.3.7.	Huấn luyện mạng.....	36
3.3.8.	Dự đoán	36
3.3.9.	Cải tiến.....	37
Chương 4.	CHƯƠNG TRÌNH THỰC NGHIỆM	38
4.1.	CẤU TRÚC CHƯƠNG TRÌNH.....	38
4.1.1.	Cấu trúc lớp NeuronNetwork	39
4.1.2.	Luồng thực thi chương trình.....	41
4.1.3.	Cấu trúc định dạng file XML lưu trữ định dạng mạng	42
4.2.	HƯỚNG DẪN SỬ DỤNG	43
Chương 5.	KẾT QUẢ THỰC NGHIỆM	50
5.1.	THỰC HIỆN VỚI DỮ LIỆU LỚN	50
5.2.	THỰC HIỆN VỚI DỮ LIỆU NHỎ	52
Chương 6.	KẾT LUẬN	56
6.1.	ĐÁNH GIÁ KẾT QUẢ	56
6.1.1.	Những công việc làm được	56
6.1.2.	Những đúc kết về mặt lý luận	56

6.1.3. Mặt hạn chế	57
6.2. HƯỚNG PHÁT TRIỂN.....	57
TÀI LIỆU THAM KHẢO	58

DANH MỤC HÌNH

Hình 1: Perceptron.....	4
Hình 2: Mạng neuron truyền thẳng.....	6
Hình 3: Mạng neuron hồi quy.....	6
Hình 4: Mô hình học có giám sát	7
Hình 5: Perceptron.....	8
Hình 6: Mặt quyết định biểu diễn bởi perceptron hai đầu nhập.	9
Hình 7: Hàm lỗi của một đơn vị tuyến tính.	12
Hình 8: Đơn vị sigmoid	14
Hình 9: Số khách hàng đặt chỗ hàng tháng của hãng Pan Am.....	24
Hình 10: Độ tăng nhiệt độ trung bình hàng năm từ 1856 đến 2005	26
Hình 11: Hàm tự tương quan của chuỗi tăng nhiệt độ trung bình hàng năm (1856 - 2005).....	27
Hình 12: Chuỗi thời gian có tính mùa.	28
Hình 13: Mô hình học với chuỗi thời gian	29
Hình 14: Thủ tục sử dụng phương pháp walk-forward chia tập dữ liệu	34
Hình 15: Lược đồ thực hiện đối với chương trình thực nghiệm	41
Hình 16: Giao diện configure của chương trình	44
Hình 17: Giao diện khởi tạo thành công.....	44
Hình 18: Giao diện configure cho huấn luyện.....	45
Hình 19: Giao diện cài đặt huấn luyện sử dụng giải thuật lan truyền ngược	45
Hình 20: Giao diện cài đặt huấn luyện sử dụng giải thuật RPROP	46
Hình 21: Giao diện kết quả của quá trình huấn luyện	47

Hình 22: Giao diện kiểm tra kết quả huấn luyện.....	47
Hình 23: Giao diện kết quả kiểm tra huấn luyện.....	48
Hình 24: Giao diện cài đặt cho việc dự báo	48
Hình 25: Giao diện kết quả của dự báo	49
Hình 26: Chuỗi thời gian nhu cầu năng lượng ở italia.	51
Hình 27: chuỗi thời gian về tỉ giá giữa đồng euro và đồng đô la.	51
Hình 28: Chỉ số tiêu dùng xăng dầu của người dân thành thị ở Mỹ.....	53
Hình 29: Số người sinh ra theo tháng	53

Chương 1. GIỚI THIỆU

1.1. ĐẶT VẤN ĐỀ

Ngày nay khi mà hầu hết các tổ chức đều hoạt động trong môi trường không chắc chắn, kế hoạch lập ra hôm nay sẽ ảnh hưởng đến sự sống còn của tổ chức trong ngày mai thì việc dự đoán trước một cách chính xác trở nên rất quan trọng đối với các nhà ra quyết định. Các nhà đầu tư cần phải dự đoán được nhu cầu thị trường, sự biến động của nền kinh tế trong tương lai để có thể đầu tư hiệu quả. Các nhà hoạch định chính sách quốc gia cần dự đoán được về môi trường kinh doanh quốc tế, tỷ lệ lạm phát, tỷ lệ thất nghiệp... trong nhiều năm tới để đưa ra các chính sách phù hợp.

Để đưa ra dự báo chính xác và có cơ sở người ta tiến hành thu nhập dữ liệu về các yếu tố liên quan đến vấn đề mình quan tâm. Một kiểu dữ liệu thu nhập thường thấy là kiểu dữ liệu chuỗi thời gian. Dữ liệu chuỗi thời gian, tức là dữ liệu được thu nhập, lưu trữ và quan sát theo sự tăng dần của thời gian. Ví dụ, số lượng thí sinh dự thi đại học vào Trường Đại Học Bách Khoa thành phố Hồ Chí Minh được lưu trữ theo từng năm, hay số lượng hàng hóa đã bán được của một siêu thị được lưu trữ theo từng quý là các dữ liệu chuỗi thời gian.

Việc dự báo dữ liệu chuỗi thời gian ngày càng chiếm vị trí quan trọng trong hoạt động của các đơn vị tổ chức. Có rất nhiều phương pháp được xây dựng để dự báo chuỗi thời gian, nhiều phương pháp (ví dụ: phương pháp hồi quy) đã được xây dựng từ thế kỷ 19 và nhiều phương pháp (ví dụ phương pháp mạng neuron nhân tạo) được phát triển gần đây. Cơ bản có hai kỹ thuật chủ yếu trong việc dự báo chuỗi thời gian là các phương pháp thống kê: hồi quy, làm trơn, ARIMA... và phương pháp dùng mạng neuron nhân tạo.

1.2. MỤC TIÊU CỦA ĐỀ TÀI

Mạng neuron nhân tạo là một mô hình toán học đã được nghiên cứu từ lâu và được ứng dụng nhiều vào các bài toán mô phỏng, nhận dạng, dự đoán. Gần đây mạng neuron nhân tạo được quan tâm và ứng dụng ngày càng nhiều vào các bài toán dự báo dữ liệu chuỗi thời gian.

Mục đích của đề tài này là tìm hiểu về nguyên tắc hoạt động của mạng neuron nhân tạo, các giải thuật huấn luyện mạng neuron: *lan truyền ngược* (backpropagation) và RPROP, cách áp dụng mạng neuron nhân tạo vào việc dự báo dữ liệu chuỗi thời gian, hiện thực một chương trình dự báo dữ liệu chuỗi thời gian sử dụng mạng neuron nhân tạo với hai giải thuật học là lan truyền ngược và RPROP, chạy thử nghiệm chương trình trên một số bộ dữ liệu mẫu để đánh giá độ chính xác dự báo và tính hữu hiệu của các giải thuật.

1.3. CẤU TRÚC BÁO CÁO

Bài báo cáo chia làm 6 chương:

Chương 1: Giới thiệu về bài toán và nhiệm vụ đề tài.

Chương 2: Giới thiệu về cấu trúc, nguyên tắc hoạt động của mạng neuron nhân tạo cùng với hai giải thuật huấn luyện: lan truyền ngược và RPROP.

Chương 3: Giới thiệu về cách ứng dụng mạng neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian.

Chương 4: Trình bày việc hiện thực chương trình dự báo dữ liệu chuỗi thời gian bằng mạng neuron nhân tạo.

Chương 5: Trình bày việc chạy thử nghiệm chương trình trên một số bộ dữ liệu và đánh giá tính hiệu quả của hai giải thuật lan truyền ngược và RPROP.

Chương 6: Kết luận về kết quả đạt được và hướng phát triển tương lai.

Chương 2. MẠNG NEURON NHÂN TẠO: CẤU TRÚC, NGUYÊN TẮC HOẠT ĐỘNG VÀ CÁC GIẢI THUẬT HUẤN LUYỆN

2.1. SƠ LƯỢC VỀ MẠNG NEURON NHÂN TẠO

Mạng neuron nhân tạo (Artificial Neural Network) là một mô hình toán học định nghĩa một hàm số từ một tập đầu vào đến một tập đầu ra. Mạng neuron nhân tạo được mô phỏng theo mạng neuron sinh học trong bộ não người.

Theo các nhà sinh lý học thì bộ não con người chứa khoảng 10^{11} các phần tử liên kết chặt chẽ với nhau gọi là các neuron. Mỗi neuron được cấu tạo bởi các thành phần: tế bào hình cây, tế bào thân và sợi trục thần kinh. Tế bào hình cây có nhiệm vụ mang tín hiệu điện sinh học tới tế bào thân, tế bào thân sẽ thực hiện tính tổng và *phân ngưỡng* (thresholds) các tín hiệu đến. Sợi trục thần kinh có nhiệm vụ đưa tín hiệu từ tế bào thân ra ngoài. Điểm tiếp xúc giữa sợi trục thần kinh này với tế bào hình cây của neuron kia gọi là *khớp thần kinh* (synapse). Sự sắp xếp của các neuron và độ mạnh yếu của các khớp thần kinh quyết định khả năng của mạng neuron. Cấu trúc của mạng neuron sinh học luôn luôn thay đổi và phát triển theo quá trình học tập và lao động của con người, các liên kết mới được tạo ra và các liên kết cũ bị loại bỏ, hay tăng giảm độ mạnh yếu của các liên kết thông qua các khớp thần kinh.

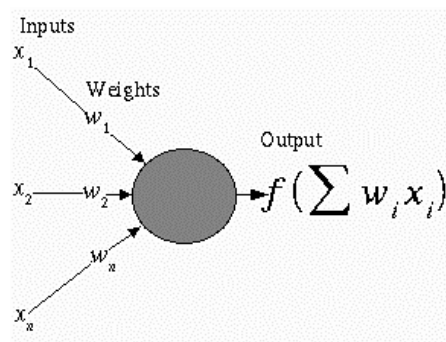
Mạng neuron nhân tạo là một sự mô phỏng đơn giản mạng neuron sinh học. Cấu trúc của mạng bao gồm các đơn vị tính toán đơn giản (tượng trưng cho các neuron) được liên kết với nhau bằng các cạnh có trọng số (tượng trưng cho các khớp thần kinh). Khả năng xấp xỉ hàm số của mạng neuron nhân tạo phụ thuộc vào hình dạng và độ mạnh yếu của các liên kết (giá trị của các trọng số).

Trong quá trình phát triển của mình mạng neuron nhân tạo đã được ứng dụng thành công trong nhiều bài toán thực tế như nhận dạng chữ viết, nhận dạng tiếng nói, điều khiển tự động, dự báo chuỗi thời gian...

2.2. CẤU TRÚC VỀ MẠNG NEURON NHÂN TẠO

Mạng neuron nhân tạo là một mạng gồm một tập các *đơn vị* (unit) được kết nối với nhau bằng các cạnh có trọng số.

Một đơn vị (Hình 1) thực hiện một công việc rất đơn giản: nó nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu ra. Mỗi đơn vị có thể có nhiều tín hiệu đầu vào nhưng chỉ có một tín hiệu đầu ra duy nhất. Đôi khi các đơn vị còn có một giá trị gọi là *độ lệch* (bias) được gộp vào các tính hiệu đầu vào để tính tín hiệu ra. Để đơn giản ký hiệu, độ lệch của một đơn vị được xem như là trọng số nối từ một đơn vị giả có giá trị xuất luôn là 1 đến đơn vị đó.



Hình 1: Perceptron

Trong một mạng neuron có ba kiểu đơn vị:

- Các đơn vị đầu vào, nhận tín hiệu từ bên ngoài.
- Các đơn vị đầu ra, gửi dữ liệu ra bên ngoài.
- Các đơn vị ẩn, tín hiệu vào của nó được truyền từ các đơn vị trước nó và tín hiệu ra được truyền đến các đơn vị sau nó trong mạng.

Khi nhận được các tín hiệu đầu vào, một đơn vị sẽ nhân mỗi tín hiệu với trọng số tương ứng rồi lấy tổng các giá trị vừa nhận được. Kết quả sẽ được đưa vào một hàm số gọi là *hàm kích hoạt* (Activation function) để tính ra tín hiệu đầu ra. Các đơn vị khác nhau có thể có các hàm kích hoạt khác nhau.

Có 4 loại hàm kích hoạt thường dùng:

- Hàm đồng nhất (Identity function):

$$g(x) = x$$

- Hàm ngưỡng:

$$g(x) = \begin{cases} 1, & \text{nếu } (x \geq \theta) \\ 0, & \text{nếu } (x < \theta) \end{cases}$$

- Hàm sigmoid:

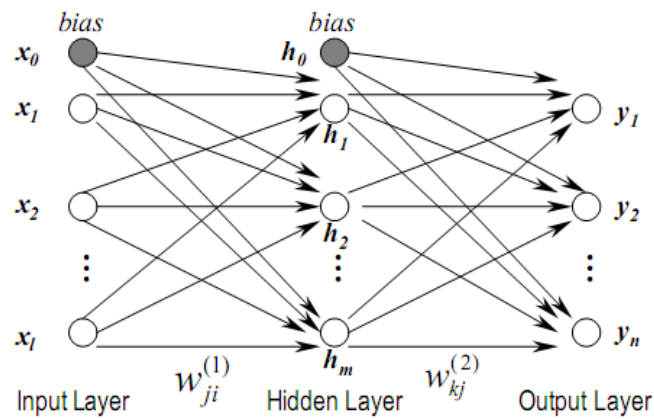
$$g(x) = \frac{1}{1 + e^{-x}}$$

- Hàm sigmoid lưỡng cực

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

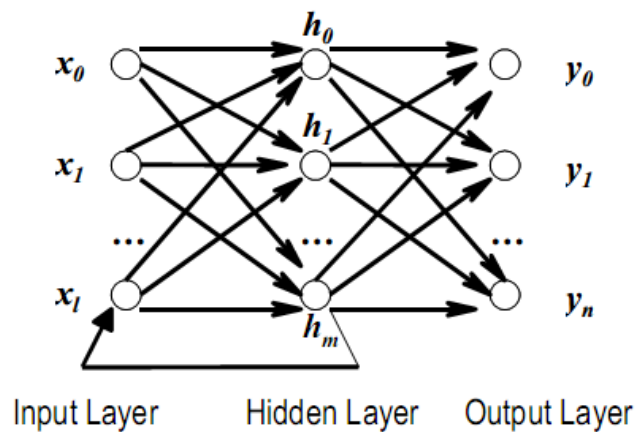
Các đơn vị liên kết với nhau qua các cạnh có trong sơ đồ tạo thành mạng neuron nhân tạo. Tùy theo số lượng các đơn vị và cách thức liên kết của chúng mà tạo thành các mạng neuron khác nhau có khả năng khác nhau. Có hai loại hình dạng mạng neuron nhân tạo cơ bản là mạng truyền thẳng và mạng hồi quy:

- *Mạng truyền thẳng* (Feed-forward neural network): Một đơn vị ở lớp đứng trước sẽ kết nối với tất cả các đơn vị ở lớp đứng sau. Tín hiệu chỉ được truyền theo một hướng từ lớp đầu vào qua các lớp ẩn (nếu có) và đến lớp đầu ra. Nghĩa là tín hiệu ra của một đơn vị không được phép truyền cho các đơn vị trong cùng lớp hay ở lớp trước. Đây là loại mạng rất phổ biến và được dùng nhiều trong việc dự báo dữ liệu chuỗi thời gian. Bài báo cáo này chỉ tập trung vào mô hình mạng này.



Hình 2: Mạng neuron truyền thẳng

- *Mạng hồi quy* (Recurrent neural network): Khác với mạng truyền thẳng, mạng hồi quy có chứa các liên kết ngược từ một đơn vị đến các đơn vị ở lớp trước nó.

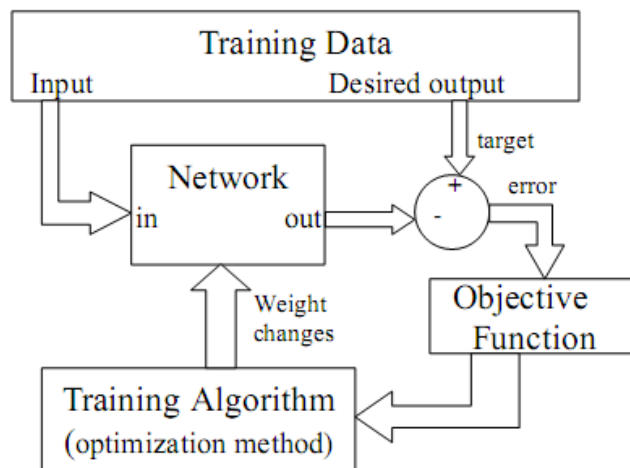


Hình 3: Mạng neuron hồi quy

Chức năng của một mạng neuron được quyết định bởi các nhân tố như: hình dạng mạng (số lớp, số đơn vị trên mỗi lớp, cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Hình dạng của mạng thường là cố định, và các trọng số được quyết định bởi một *thuật toán huấn luyện* (training algorithm). Tiến trình điều chỉnh các trọng số để mạng “nhận biết” được quan hệ giữa đầu vào và

đầu ra mong muốn được gọi là *học* (learning) hay *huấn luyện* (training). Rất nhiều thuật toán huấn luyện đã được phát minh để tìm ra tập trọng số tối ưu làm giải pháp cho các bài toán. Các thuật toán đó có thể chia làm hai nhóm chính: *Học có giám sát* (Supervised learning) và *Học không có giám sát* (Unsupervised Learning) [3].

- **Học có giám sát:** Mạng được huấn luyện bằng cách cung cấp cho nó các cặp mẫu đầu vào và các đầu ra mong muốn (target values). Các cặp này có sẵn trong quá trình thu nhập dữ liệu. Sự khác biệt giữa các đầu ra theo tính toán trên mạng so với các đầu ra mong muốn được thuật toán sử dụng để thích ứng các trọng số trong mạng. Điều này thường được đưa ra như một bài toán xấp xỉ hàm số - cho dữ liệu huấn luyện bao gồm các cặp mẫu đầu vào x , và một đích tương ứng t , mục đích là tìm ra hàm $f(x)$ thỏa mãn tất cả các mẫu học đầu vào [3]. Đây là mô hình học rất phổ biến trong việc áp dụng mạng neuron vào bài toán dự báo dữ liệu chuỗi thời gian. Hai giả thuật được đề cập trong bài báo cáo này, lan truyền ngược và RPROP là hai giải thuật học thuộc mô hình này.



Hình 4: Mô hình học có giám sát

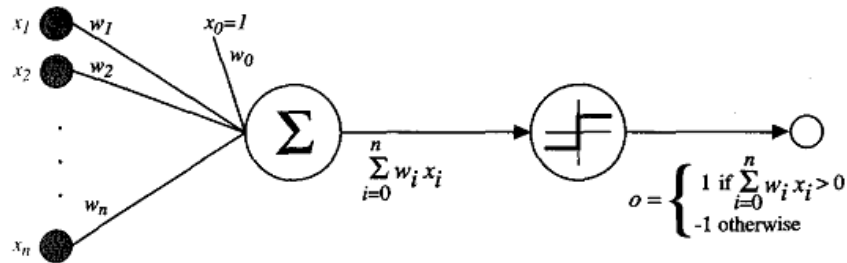
- **Học không có giám sát:** với cách học không có giám sát, không có phản hồi từ môi trường để chỉ ra rằng đầu ra của mạng là đúng. Mạng sẽ phải khám phá các đặc trưng, các điều chỉnh, các mối tương quan, hay các lớp trong dữ liệu vào một cách tự động. Trong thực tế, đối với phần lớn các biến thể của học không có giám sát, các đích trùng

Ứng dụng mạng Neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian với đầu vào. Nói một cách khác, học không có giám sát luôn thực hiện một công việc tương tự như một mạng tự liên hợp, cô đọng thông tin từ dữ liệu vào [3]

2.3. NGUYÊN TẮC HOẠT ĐỘNG VÀ CÁC GIẢI THUẬT HUẤN LUYỆN CỦA MẠNG NEURON NHÂN TẠO

2.3.1. Perceptron

Để hiểu rõ về nguyên tắc hoạt động và cách huấn luyện các mạng neuron nhân tạo trước hết ta khảo sát một mô hình mạng neuron đơn giản được xây dựng trên một đơn vị gọi là perceptron. Một perceptron nhận một vector các giá trị thực, tính tổ hợp tuyến tính của chúng và xuất ra 1 nếu kết quả lớn hơn một ngưỡng nào đó và xuất ra -1 trong các trường hợp còn lại.



Hình 5: Perceptron

Một cách hình thức, khi nhận một vector đầu vào n chiều gồm các giá trị x_1 đến x_n , giá trị xuất sẽ được tính như sau:

$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

Ở đây các số thực w_i là các trọng số biểu diễn mức độ đóng góp của giá trị nhập x_i vào giá trị xuất của perceptron. Đại lượng $(-w_0)$ là ngưỡng mà tổ hợp tuyến các giá trị nhập phải vượt qua để kết quả xuất là 1. Đặt $x_0 = 1$, ta viết lại phương trình trên dưới dạng vector như sau

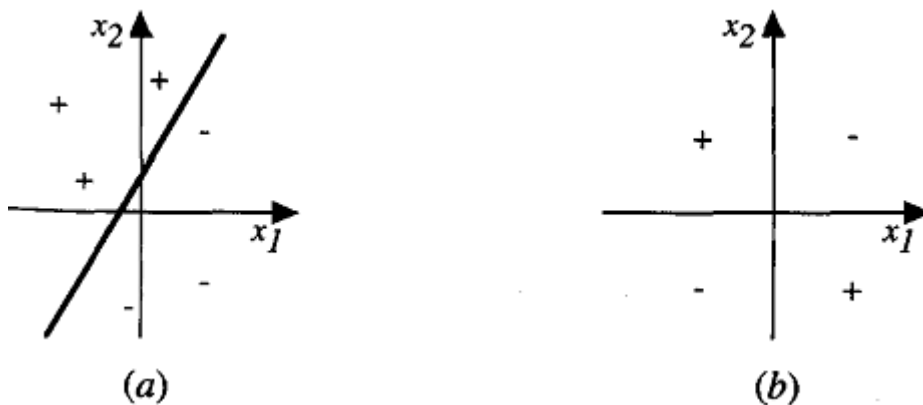
$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

Ở đây \vec{x} và \vec{w} là các vector có $n + 1$ chiều. Hàm $sgn(y)$ được định nghĩa như sau:

$$sgn(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise} \end{cases}$$

Nếu xem các vector nhập (x_0, x_1, \dots, x_n) là các điểm trên không gian $n + 1$ chiều (x_0 luôn là 1) thì perceptron biểu diễn một *mặt quyết định* (decision surface) xem một điểm có nằm trên một *siêu phẳng* (hyperplane) có phương trình là $\vec{w} \cdot \vec{x} = 0$ hay không. Perceptron sẽ xuất ra giá trị 1 cho các điểm nằm trên siêu phẳng này và xuất ra -1 cho các điểm còn lại.

Trong thực tế, ta thường có sẵn một bộ dữ liệu mẫu gồm một tập các điểm được gán nhãn dương và âm. Bài toán huấn luyện perceptron là bài toán xác định vector \vec{w} sau cho siêu phẳng $\vec{w} \cdot \vec{x} = 0$ phân chia các điểm trong tập mẫu một cách chính xác theo các nhãn của nó. Thực tế có một số bộ dữ liệu mà không thể tìm thấy bất kỳ siêu phẳng nào có thể phân chia đúng các điểm của nó, các bộ dữ liệu đó được gọi là tập dữ liệu không *khả phân tuyến tính* (linearly separable). Ngược lại nếu một bộ dữ liệu có thể được phân chia đúng bởi một siêu phẳng nào đó thì gọi là khả phân tuyến tính.



Hình 6: Mặt quyết định biểu diễn bởi perceptron hai đầu nhập.

Hình 6 (a) là một tập mẫu khả phân tuyến tính có thể được phân ra bởi một mặt quyết định của perceptron. Hình 6 (b) là một tập mẫu không khả phân tuyến tính.

Quá trình huấn luyện một perceptron là một quá trình tìm kiếm một vector \vec{w} trên một không gian thực $n + 1$ chiều sau cho nó có khả năng phân xuất ra các giá trị

+1, -1 một cách đúng đắn cho một tập dữ liệu nào đó. Có hai giải thuật huấn luyện cơ bản là *luật huấn luyện perceptron* (perceptron training rule) và *luật delta* (delta rule).

- a) **Luật huấn luyện perceptron:** Để tìm một vector \vec{w} thích hợp, trước hết ta áp dụng một perceptron với trọng số \vec{w} ngẫu nhiên qua từng mẫu của tập dữ liệu huấn luyện và hiệu chỉnh các trọng số này khi có sự phân loại sai tập mẫu. Quá trình này được lặp đi lặp lại cho đến khi perceptron đã phân loại đúng tất cả các mẫu của tập huấn luyện. Các trọng số được cập nhập theo luật

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Ở đây o là giá trị xuất của perceptron, t là giá trị đích của mẫu huấn luyện hiện thời, x_i là giá trị nhập thứ i , η là *hệ số học* (learning rate) có vai trò điều tiết mức độ thay đổi của trọng số trong các bước cập nhập. Nó thông thường được gán một giá trị dương nhỏ (ví dụ 0.1) và được điều chỉnh giảm khi số lần cập nhập trọng số tăng lên [2]. Giải thuật học này được chứng minh hội tụ sau một số hữu hạn lần cập nhập các trọng số đối với các tập dữ liệu mẫu khả phân tuyến tính và một hệ số học đủ nhỏ nhưng đối với các tập dữ liệu không khả phân tuyến tính thì sự hội tụ là không chắc [2].

- b) **Luật delta:** Luật perceptron không đảm bảo tính hội tụ đối với các tập mẫu khả phân tuyến tính do đó người ta thiết kế giải thuật luật delta để vượt qua khó khăn này. Luật delta sẽ hội tụ về một xấp xỉ tốt nhất cho các tập không khả phân tuyến tính. Ý tưởng chính của luật delta là áp dụng phương pháp *giảm độ dốc* (gradient descent) để tìm kiếm vector trọng số đáp ứng tốt nhất tập huấn luyện. Xét một perceptron thực hiện việc lấy tổ hợp tuyến tính các giá trị nhập nhưng không phân ngưỡng kết quả.

Perceptron này gọi là *perceptron không phân ngưỡng* (unthresholded perceptron) hay còn gọi là *đơn vị tuyến tính* (linear unit). Giá trị xuất của perceptron được tính như sau

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

Để áp dụng luật delta ta cần định nghĩa một hàm đánh giá, hay còn gọi là *hàm lỗi* (training error function). Có nhiều hàm lỗi được sử dụng nhưng thường dùng nhất là hàm sau

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Ở đây D là tập dữ liệu huấn luyện, d là một mẫu trong tập D , t_d là giá trị đích của mẫu d , o_d là giá trị xuất của perceptron. Mục đích của luật delta là tìm vector \vec{w} sau cho $E(\vec{w})$ đạt giá trị nhỏ nhất. Hình 7 là một biểu diễn hàm lỗi của một đơn vị tuyến tính. Trục thẳng đứng của đồ thị là giá trị hàm lỗi, hai trục ở mặt phẳng ngang là giá trị của các trọng số.

Phương pháp giảm độ dốc bắt đầu tìm với một vector trọng số ngẫu nhiên và duyệt qua các mẫu của tập huấn luyện, mỗi lần duyệt qua các trọng số sẽ được cập nhập theo hướng làm giảm giá trị hàm lỗi. Quá trình này được lặp đi lặp lại cho đến khi đạt được giá trị cực tiểu của hàm lỗi.

Hướng cập nhập các trọng số để làm giảm giá trị hàm lỗi được xác định theo *vector độ dốc* (gradient) của hàm lỗi E theo \vec{w} , ký hiệu là $\nabla E(\vec{w})$

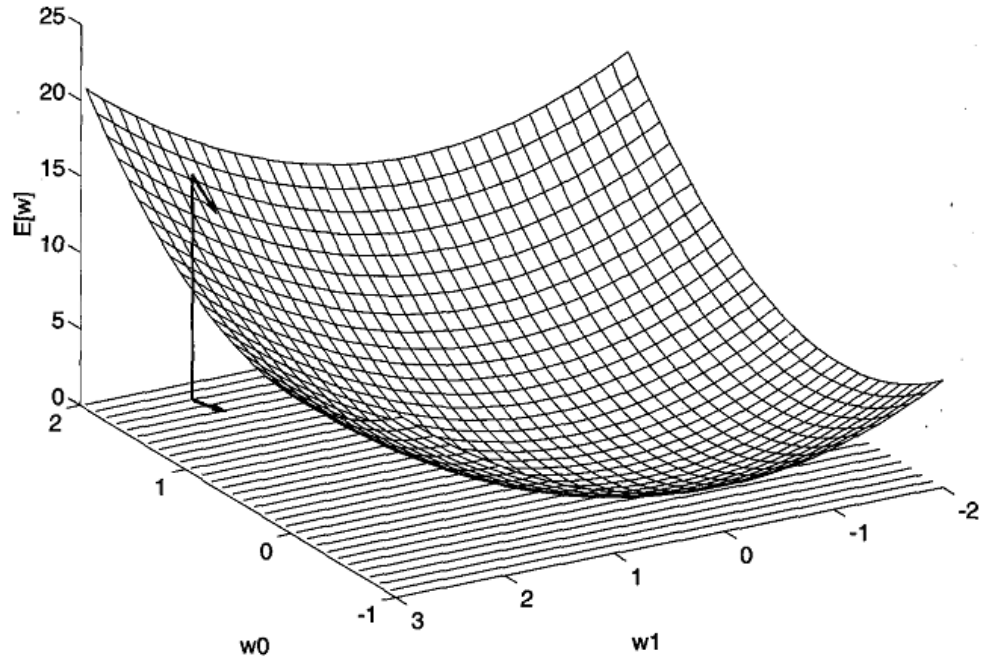
$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

Về mặt toán học vector độ dốc biểu diễn hướng làm tăng giá trị hàm E trong không gian trọng số, do đó $-\nabla E(\vec{w})$ sẽ là hướng làm giảm giá trị hàm E . Trong hình 7 nó được biểu diễn bằng dấu mũi tên. Các trọng số sẽ được cập nhập theo quy luật sau:

$$\begin{aligned} \vec{w} &\leftarrow \vec{w} + \Delta \vec{w} \\ \Delta \vec{w} &= -\eta \nabla E(\vec{w}) \end{aligned}$$

Luật huấn luyện này có thể được viết lại cho từng trọng số như sau:

$$\begin{aligned} w_i &\leftarrow w_i + \Delta w_i \\ \Delta w_i &= -\eta \frac{\partial E}{\partial w_i} \end{aligned} \quad (2.1)$$



Hình 7: Hàm lỗi của một đơn vị tuyến tính.

Để thực hiện cập nhật các trọng số, ta thực hiện tính đạo hàm riêng phần của hàm E theo từng trọng số:

$$\begin{aligned}
 \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
 &= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
 &= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
 &= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\
 \frac{\partial E}{\partial w_i} &= \sum_{d \in D} (t_d - o_d)(-x_{id}) \quad (2.2)
 \end{aligned}$$

Thay (2.2) vào (2.1) ta được giá cập nhật trọng số qua từng bước ta được

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \quad (2.3)$$

Ở đây giá trị x_{id} là giá trị đầu vào thứ i của mẫu d .

Phương pháp giảm độ dốc có hai hạn chế chính là tốc độ hội tụ đôi khi khá chậm và nếu có nhiều *cực tiểu cục bộ* (local minimum) trên bề mặt của hàm lỗi thì giải thuật dễ rơi vào cực tiểu cục bộ mà không đạt được *cực tiểu toàn cục* (global minimum). Để giải quyết các khó khăn này người ta đã phát triển phương pháp giảm độ dốc thành phương pháp *giảm độ dốc tăng cường* (incremental gradient descent). Khác với phương pháp giảm độ dốc ở trên phương pháp giảm độ dốc tăng cường thực hiện việc tính toán lỗi và cập nhật các trọng số ngay khi duyệt qua một mẫu của tập dữ liệu. Giá trị cập nhật cho các trọng số của phương pháp giảm độ dốc tăng cường là

$$\Delta w_i = \eta(t - o) x_i \quad (2.4)$$

Ở đây các giá trị t , o , x_i lần lượt là giá trị đích, giá trị xuất của mạng và giá trị nhập thứ i của mẫu huấn luyện hiện hành. Hàm lỗi của phương pháp giảm độ dốc tăng cường không phải là hàm lỗi toàn cục cho toàn bộ dữ liệu huấn luyện như phương pháp giảm độ dốc thường mà là hàm lỗi cho từng mẫu trong tập dữ liệu

$$E_d(\vec{w}) = \frac{1}{2}(t_d - o_d)^2$$

Ở đây giá trị t_d , o_d lần lượt là giá trị đích và giá trị xuất của mạng cho mẫu d trong tập dữ liệu. Với một hệ số học đủ nhỏ, phương pháp giảm độ dốc tăng cường có thể xấp xỉ tốt tùy ý phương pháp giảm độ dốc thông thường [2]. Theo Tom Mitchell [2] phương pháp giảm độ dốc tăng cường khác với phương pháp giảm độ dốc thông thường ở ba điểm sau. Thứ nhất, giải thuật thực hiện việc tính toán lỗi và cập nhật các trọng số cho mỗi mẫu trong tập huấn luyện chứ không đợi duyệt qua hết các mẫu trong tập huấn luyện. Thứ hai, phương pháp giảm độ dốc thông thường cần nhiều tính toán để cập nhật các trọng số vì nó cần phải tính toán hàm lỗi thực sự cho toàn bộ tập dữ liệu huấn luyện và mỗi lần cập nhật các trọng số được cập nhật một bước lớn hơn phương pháp giảm độ dốc tăng cường. Thứ ba phương pháp giảm độ dốc tăng cường có khả năng không bị rơi vào cực tiểu cục bộ vì nó sử dụng $\nabla E_d(\vec{w})$ thay cho $\nabla E(\vec{w})$ để tìm kiếm.

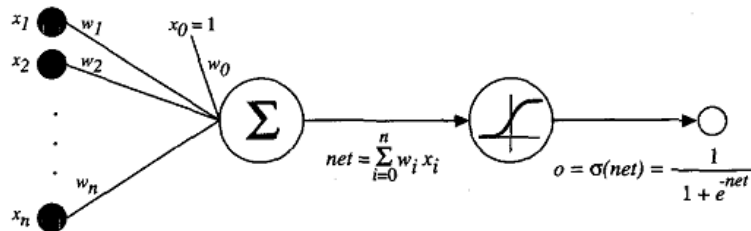
Sự khác biệt giữa hai giải thuật huấn luyện luật delta và luật huấn luyện perceptron khác nhau ở tính chất hội tụ của chúng. Luật huấn luyện perceptron hội tụ

sau một số lần lặp hữu hạn và tìm ra một mặt phẳng phân loại hoàn hảo một tập dữ liệu huấn luyện khả phân tuyến tính trong khi giải thuật luật delta sẽ hội tụ về một điểm cực tiểu của hàm lỗi với một thời gian khá lâu (có thể là vô hạn) nhưng sự hội tụ của nó không bị ảnh hưởng bởi tính khả phân tuyến tính của tập dữ liệu huấn luyện [2].

2.3.2. Mạng nhiều lớp và giải thuật lan truyền ngược

Mạng neuron đơn giản như perceptron chỉ biểu diễn được các hàm tuyến tính, nhưng trong thực tế ta cần biểu diễn các hàm phi tuyến như trong các bài toán nhận dạng giọng nói hay dự báo chuỗi thời gian. Để làm được điều này ta sử dụng các mạng neuron nhiều lớp, tức là mạng gồm một lớp đầu vào, một lớp đầu ra và một hay nhiều lớp ẩn.

Các mạng neuron nhiều lớp ít khi sử dụng các đơn vị tuyến tính hay đơn vị phân ngưỡng mà chúng sử dụng các đơn vị có các hàm kích hoạt là các hàm khả vi. Một trong những đơn vị hay dùng nhất là *đơn vị sigmoid* (sigmoid unit). Một đơn vị sigmoid sẽ tính tổ hợp tuyến tính các giá trị đầu vào và đưa kết quả này vào hàm sigmoid để tính giá trị đầu ra.



Hình 8: Đơn vị sigmoid

Công thức tính giá trị đầu ra của đơn vị sigmoid là

$$o = \sigma(\vec{w} \cdot \vec{x})$$

Với

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

Một thuận lợi khi sử dụng các đơn vị sigmoid là nhờ đạo hàm của hàm sigmoid rất dễ tính ($\sigma'(y) = \sigma(y) * (1 - \sigma(y))$). Điều này làm cho việc áp dụng phương pháp giảm độ dốc được dễ dàng.

Giải thuật lan truyền ngược tìm tập các trọng số thích hợp cho một mạng neuron truyền thẳng nhiều lớp. Nó áp dụng phương pháp giảm độ dốc (gradient descent) để tối thiểu hóa bình phương sai số giữa kết quả xuất của mạng với kết quả xuất mong muốn. Ý tưởng chính của giải thuật là giá trị lỗi sẽ được lan truyền ngược từ tầng xuất về tầng nhập để tính $\nabla E(\vec{w})$

Hàm lỗi của giải thuật lan truyền ngược được định nghĩa tổng quát như sau

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{kd} - o_{kd})^2$$

Ở đây *outputs* là tập các đầu ra của mạng neuron, t_{kd} và o_{kd} lần lượt là giá trị đích và giá trị xuất của đầu ra thứ k của mẫu huấn luyện d

Giải thuật lan truyền ngược áp dụng phương pháp giảm độ dốc để tìm ra điểm tối ưu của hàm lỗi. Với mỗi mẫu trong tập huấn luyện, mạng neuron được áp dụng để tính đầu ra sau đó giá trị độ dốc của hàm lỗi được tính cho từng đơn của mạng. Cuối cùng giải thuật áp dụng phương pháp giảm độ dốc để cập nhập các giá trị trọng số

Để áp dụng phương pháp giảm độ dốc trước hết ta cần thông tin về đạo hàm riêng phần của hàm lỗi cho từng trọng số

Ta tính đạo hàm riêng phần này như sau:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_i} \frac{\partial o_i}{\partial w_{ij}} \quad (2.5)$$

$$\text{với } \frac{\partial o_i}{\partial w_{ij}} = \frac{\partial o_i}{\partial \text{net}_i} \frac{\partial \text{net}_i}{\partial w_{ij}} = f'(\text{net}_i) o_i \quad (2.6)$$

$$\text{net}_i = \sum_{j \in \text{pred}(i)} s_j w_{ij} - \theta_i \quad (2.7)$$

Ở đây:

- w_{ij} là trọng số của cạnh nối đơn vị j đến đơn vị i
- o_i là kết quả xuất của đơn vị i

- $f()$ là hàm kích hoạt của các đơn vị
- $pred(i)$ là các đơn vị đứng trước đơn vị i trong mạng

Giá trị $\frac{\partial E}{\partial o_i}$ được tính theo hai trường hợp tùy theo đơn vị i là đơn vị ở tầng xuất

hay tầng ẩn:

- Nếu đơn vị i là đơn vị ở tầng xuất thì:

$$\frac{\partial E}{\partial o_i} = \frac{\partial}{\partial o_i} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

Đạo hàm $\frac{\partial}{\partial o_j} (t_k - o_k)^2$ bằng 0 đối với mỗi giá trị k khác i nên

$$\begin{aligned} \frac{\partial E}{\partial o_i} &= \frac{\partial}{\partial o_i} \frac{1}{2} (t_i - o_i)^2 \\ &= \frac{1}{2} 2(t_i - o_i) \frac{\partial (t_i - o_i)}{\partial o_i} \\ &= -(t_i - o_i) \end{aligned} \quad (2.8)$$

Thay (2.8) và (2.6) vào (2.5) ta được công thức tính đạo hàm riêng phần của hàm lỗi theo trọng số w_{ij} của đơn vị xuất i

$$\frac{\partial E}{\partial w_{ij}} = -(t_i - o_i) * f'(net_i) o_i \quad (2.9)$$

- Nếu đơn vị i là đơn vị ở tầng ẩn ở tầng ẩn thì việc tính toán phức tạp hơn bởi vì giá trị xuất của i không ảnh hưởng trực tiếp lên giá trị xuất của mạng neuron mà ảnh hưởng gián tiếp thông qua các đơn vị ở sau nó.

$$\begin{aligned} \frac{\partial E}{\partial o_i} &= \sum_{k \in succ(i)} \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial o_i} \\ &= \sum_{k \in succ(i)} \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial o_i} = \sum_{k \in succ(i)} \frac{\partial E}{\partial o_k} f'(net_k) w_{ki} \end{aligned} \quad (2.10)$$

Thay (2.10) và (2.6) vào (2.5) ta được công thức tính đạo hàm riêng phần của hàm lỗi theo trọng số w_{ij} của đơn vị ẩn i

$$\frac{\partial E}{\partial w_{ij}} = \left(\sum_{k \in succ(i)} \frac{\partial E}{\partial o_k} f'(net_k) w_{ki} \right) * f'(net_i) o_i \quad (2.11)$$

Ở đây $succ(i)$ là các đơn vị ở lớp ngay sau đơn vị i . Các công thức này cho phép ta xây dựng một thủ tục tính đạo hàm riêng của hàm lỗi E theo các trọng số w_{ij} như sau: Bắt đầu tính toán từ các đơn vị ở tầng xuất, sau đó sử dụng kết quả vừa tính được vào việc tính toán ở các đơn vị ở tầng trước. Nói cách khác thông tin về độ dốc được lan truyền từ tầng xuất đến tầng nhập. Do đó giả thuật này được gọi là giải thuật lan truyền ngược.

Mỗi khi thông tin về đạo hàm riêng phần đã biết, bước tiếp theo trong giải thuật lan truyền ngược là cập nhập các trọng số w_{ij} .

$$\Delta w(t) = -\eta * \nabla E(t) = -\eta * \frac{\partial E(t)}{\partial w(t)} \quad (2.12)$$

$$w(t+1) = w(t) + \Delta w(t) \quad (2.13)$$

Ở đây η là hệ số học có vai trò điều tiết mức độ thay đổi của trọng số trong các bước cập nhập.

Cơ bản có hai phương pháp cập nhập các trọng số phân loại theo thời điểm cập nhập: *học theo mẫu* (learning by pattern) và *học theo epoch* (learning by epoch). Một epoch là một lần học duyệt qua tất cả các mẫu trong tập dữ liệu mẫu dùng để học.

Trong phương pháp học theo mẫu đôi khi còn được gọi là *học trực tuyến* (online learning) áp dụng phương pháp giảm độ dốc tăng cường, cứ mỗi lần một mẫu trong tập dữ liệu được duyệt qua thì các trọng số sẽ được cập nhập. Phương pháp này cố gắng tối thiểu hàm lỗi tổng thể (overall error) bằng cách tối ưu hàm lỗi cho từng mẫu trong tập dữ liệu học. Phương pháp này làm việc tốt cho các tập dữ liệu mẫu có kích cỡ lớn và chứa đựng nhiều thông tin dư thừa [5].

Phương pháp học theo epoch (learning by epoch) thực hiện lấy tổng tất cả thông tin về độ dốc (gradient) cho toàn bộ tập mẫu (pattern set) sau đó mới cập nhập các trọng số theo phương pháp giảm độ dốc thông thường, nghĩa là nó thực hiện việc cập nhập trọng số sau khi đã duyệt qua hết các mẫu trong tập dữ liệu. Phương pháp này còn có tên gọi khác là học theo bó (batch learning).

Mặc dù giải thuật lan truyền ngược tương đối đơn giản nhưng trong thực tế việc lựa chọn một hệ số học phù hợp là không hề đơn giản. Hệ số học quá nhỏ sẽ dẫn đến thời gian hội tụ của giải thuật quá lâu, ngược lại hệ số học quá lớn sẽ dẫn đến hiện tượng giao động (oscillation), ngăn không cho giá trị hàm mục tiêu hội tụ về một điểm nhất định. Hơn nữa, mặc dù điểm tối ưu cục bộ có thể được chứng minh là luôn có thể đạt được ở một vài trường hợp cụ thể nhưng không có gì đảm bảo giải thuật sẽ tìm được cực toàn cục của hàm lỗi [5]. Một vấn đề khác nữa là kích cỡ của đạo hàm cũng ảnh hưởng đến sự cập nhật các trọng số. Nếu đạo hàm riêng phần quá nhỏ thì Δw nhỏ, nếu đạo hàm riêng phần lớn thì Δw lớn. Độ lớn của đạo hàm riêng phần thay đổi không thể biết trước được theo hình dạng của hàm lỗi E trong mỗi lần lặp. Do đó quá trình học không ổn định.

Để cho quá trình học ổn định người ta thêm vào một *hệ số quán tính* (momentum term)

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}}(t) + \mu * \Delta w_{ij}(t-1) \quad (2.14)$$

Hệ số quán tính μ có tác dụng điều chỉnh mức độ ảnh hưởng của giá trị $\Delta w_{ij}(t-1)$ ở bước lặp trước lên giá trị $\Delta w_{ij}(t)$. Hệ số này có tác dụng giúp cho giải thuật không bị dừng ở tối ưu cục bộ và các vùng phẳng của bề mặt lỗi. Nó cũng giúp tăng giá trị cập nhật ở những vùng mà độ dốc không đổi, do đó tăng tốc độ hội tụ [2].

Sau đây là mã giả cho giải thuật lan truyền ngược theo phương pháp học trực tuyến có áp dụng hệ số quán tính:

1. Khởi tạo tất cả các trọng số bằng các số nhỏ ngẫu nhiên
2. **Loop until** điều kiện dừng thỏa
 - 2.1. **For each** mỗi mẫu trong tập dữ liệu, **do**
 - 2.1.1. Nhập mẫu vào mạng và tính toán giá trị đầu ra.
 - 2.1.2. **For each** mỗi giá trị xuất của đơn vị k

$$\text{Tính } \frac{\partial E}{\partial w_{kj}}$$
 - 2.1.3. **For each** đơn vị ẩn h , từ lớp ẩn cuối cùng đến lớp ẩn đầu tiên
 - 2.1.3.1. $\text{Tính } \frac{\partial E}{\partial w_{hj}}$
 - 2.1.4. **For each** w_{ij} trong mạng
 - 2.1.4.1. $\text{Tính } \Delta w_{ij}(t) = -\varepsilon \frac{\partial E}{\partial w_{ij}}(t) + \mu * \Delta w_{ij}(t-1)$
 - 2.1.4.2. $\text{Tính } w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
 - 2.1.5. **End for**
 - 2.2. **End for**
3. **End loop**

Giải thuật lan truyền ngược cần hai thông số nhập vào đó là hệ số học và hệ số quán tính. Đối với mỗi bài toán khác nhau các thông số này cần có các giá trị khác nhau để đạt được sự hiệu quả trong quá trình học. Việc xác định các thông số này một cách đúng đắn không phải là một việc dễ dàng cần nhiều công sức và kinh nghiệm.

2.3.3. Giải thuật RPROP

Giải thuật lan truyền ngược gặp một vấn đề ở chỗ giá trị cập nhập trọng số ($\Delta w_{ij}(t)$) không những phụ thuộc vào dấu của đạo hàm riêng phần mà còn bị ảnh hưởng bởi độ lớn của nó, điều này làm cho quá trình học không được ổn định. Việc thêm vào hệ số quán tính không giải quyết trọn vẹn vấn đề bởi vì ta không biết giá trị tối ưu cho hệ số này là bao nhiêu.

Nhiều giải thuật đã được phát minh để giải quyết các vấn đề trên, chúng có thể được phân ra làm hai loại: *chiến lược toàn cục* (global strategy) và *chiến lược cục bộ* (local strategy). Chiến lược toàn cục sử dụng kiến thức về trạng thái của toàn bộ mạng

để hiệu chỉnh các thông số toàn cục, trong khi chiến lược cục bộ dùng các thông tin riêng của từng trọng số một để thích nghi các thông số đặc biệt cho từng trọng số [4].

RPROP là viết tắt của từ ‘resilient propagation’, nghĩa là lan truyền đàn hồi là một phương pháp thích nghi cục bộ. RPROP thực hiện cập nhập các trọng số w_{ij} dựa vào thông tin về dấu của các đạo hàm riêng phần điều này giúp nó tránh được sự ảnh hưởng của độ lớn của các đạo hàm riêng phần này. Để thực hiện điều này các trọng số sẽ có một giá trị cập nhập riêng Δ_{ij} chỉ phụ thuộc vào dấu của $\frac{\partial E}{\partial w_{ij}}$. Giá trị này được

cập nhập trong quá trình học theo quy luật sau:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \eta^- * \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{else} \end{cases} \quad (2.15)$$

Ở đây $0 < \eta^- < 1 < \eta^+$ là các hệ số cố định của quá trình học dùng để hiệu chỉnh các giá trị cập nhập cho từng trọng số tùy theo hình dạng của hàm lỗi.

Mỗi lần đạo hàm riêng phần theo trọng số w_{ij} của hàm lỗi E đổi dấu, nghĩa là giá trị cập nhập vừa thực hiện là quá lớn và giải thuật đã nhảy vượt qua điểm tối ưu cục bộ thì giá trị cập nhập Δ_{ij} sẽ giảm đi theo một thừa số η^- . Ngược lại nếu đạo hàm riêng phần vẫn giữ nguyên dấu thì giá trị cập nhập Δ_{ij} sẽ được tăng lên để tăng tốc độ hội tụ. Cứ mỗi lần giá trị cập nhập được biết thì các trọng số được điều chỉnh theo luật sau: nếu đạo hàm riêng phần dương thì trọng số được giảm đi một lượng bằng với giá trị cập nhập (Δ_{ij}), nếu đạo hàm riêng phần âm thì giá trị cập nhập được cộng thêm vào trọng số.

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} , & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(t)} , & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ 0 , & \text{else} \end{cases} \quad (2.16)$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \quad (2.17)$$

Tuy nhiên có một trường hợp đặc biệt đó là khi đạo hàm riêng phần đối dấu, nghĩa là bước cập nhật trước quá lớn làm cho điểm tối ưu bị nhảy vượt qua. Giá trị trọng số phải được trả về giá trị trước khi thay đổi, giá trị cập nhật Δ_{ij} sẽ được giảm xuống và ở bước kế sau ta sẽ không cập nhật giá trị này. Thực tế ta có thể làm việc này bằng cách gán

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)} \text{ và } \frac{\partial E^{(t)}}{\partial w_{ij}} = 0$$

Giải thuật RPROP thực hiện việc thích nghi các giá trị cập nhật của các trọng số tùy theo độ dốc (gradient) của hàm lỗi E, mà thông tin về tổng độ dốc của hàm lỗi cho toàn bộ tập dữ liệu mẫu đáng tin hơn thông tin về độ dốc chỉ cho một mẫu trong tập mẫu nên giải thuật RPROP thực hiện theo mô hình học theo bó (học theo epoch). Các thông tin về đạo hàm riêng phần sẽ được cộng dồn qua từng mẫu trong tập huấn luyện và các trọng số sẽ được cập nhật sau khi đã duyệt qua hết các mẫu.

Giải thuật RPROP ban đầu cũng thực hiện các bước giống như giải thuật lan truyền ngược, các thông tin về đạo hàm riêng phần của hàm lỗi theo các trọng số sẽ được lan truyền ngược từ các lớp sau đến các lớp trước. Khi các thông tin về các đạo hàm riêng phần này có đủ thì giải thuật sẽ thực hiện việc cập nhật các trọng số theo các quy tắc nêu ở trên.

Mã giả cho phần cập nhật trọng số của giải thuật RPROP như sau:

For mọi trọng số và độ lệch {

if ($\frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} > 0$) **then** {

$$\Delta_{ij}^{(t)} = \text{minimum} (\Delta_{ij}^{(t-1)} * \eta^+, \Delta_{\max})$$

$$\Delta w_{ij}^{(t)} = - \text{sign} (\frac{\partial E^{(t)}}{\partial w_{ij}}) * \Delta_{ij}^{(t)}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

} else if ($\frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} < 0$) **then** {

$$\Delta_{ij}^{(t)} = \text{maximum} (\Delta_{ij}^{(t-1)} * \eta^-, \Delta_{\min})$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \Delta w_{ij}^{(t)}$$

$$\frac{\partial E^{(t)}}{\partial w_{ij}} = 0$$

}

else if ($\frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} = 0$) **then** {

$$\Delta w_{ij}^{(t)} = - \text{sign} (\frac{\partial E^{(t)}}{\partial w_{ij}}) * \Delta_{ij}^{(t)}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

}

}

Ở đây hàm số minimum và maximum lần lượt là hai hàm trả về giá trị nhỏ nhất và lớn nhất của hai số. Hàm số sign(x) trả về 1 nếu x dương, trả về -1 nếu x âm và trả về 0 trong các trường hợp còn lại.

Ban đầu các giá trị cập nhập Δ_{ij} sẽ được khởi tạo một giá trị dương ban đầu Δ_0 . Lựa chọn tốt cho Δ_0 là 0.1 nhưng theo các nghiên cứu thì việc lựa chọn tham số này không ảnh hưởng nhiều đến tốc độ hội tụ của giải thuật [4]. Các thông số về Δ_{\min} và Δ_{\max} để tránh vấn đề tràn số của các biến thực. Giá trị Δ_{\min} được thiết lập thường là

$1.0e-6$, còn giá trị Δ_{\max} là 50.0. Thông thường độ hội tụ của giải thuật không bị ảnh hưởng bởi các thông số này nhưng đôi khi thông số Δ_{\max} được chọn là một giá trị nhỏ (ví dụ 1.0) để ngăn giải thuật không rơi quá nhanh vào một cực tiểu cục bộ [4]. Hai thông số η^+ và η^- được cố định ở hai giá trị lần lượt là 1.2 và 0.5, để việc lựa chọn các tham số cho giải thuật được đơn giản. Trong thực tế, hai thông số cần được lựa chọn cho giải thuật RPROP là Δ_0 và Δ_{\max} .

Một trong các thuận tiện của RPROP là trong nhiều bảo toán không cần phải lựa chọn các tham số một cách cẩn thận cũng đạt được tốc độ hội tụ tối ưu hay gần tối ưu [4].

Trong giải thuật lan truyền ngược, giá trị cập nhật trọng số phụ thuộc vào độ lớn của đạo hàm riêng phần của hàm lỗi theo trọng số, mà giá trị này lại giảm theo khoảng cách của các trọng số đối với lớp xuất. Do đó các trọng số ở xa lớp xuất sẽ ít được hiệu chỉnh hơn và việc huấn luyện các trọng số này sẽ chậm hơn các trọng số gần lớp xuất. Tuy nhiên khi dùng RPROP thì giá trị cập nhật trọng số chỉ phụ thuộc vào dấu nên sự huấn luyện sẽ trải đều trên toàn bộ mạng: những trọng số gần lớp nhập cũng có cơ hội được cập nhật và phát triển ngang với các trọng số gần lớp xuất [4].

2.3.4. Hiện tượng quá khớp

Trong việc huấn luyện mạng neuron, đôi khi ta gặp phải hiện tượng mạng xấp xỉ rất tốt tập dữ liệu huấn luyện nhưng cho ra kết quả dự đoán thiếu chính xác, giảm khả năng tổng quát của mạng. Đây được gọi là hiện tượng *quá khớp* (overfitting).

Để đảm bảo tính tổng quát hóa của mạng và tránh hiện tượng quá khớp, ta cần chuẩn bị một tập dữ liệu kiểm tra. Tập dữ liệu này được sử dụng trong giai đoạn huấn luyện, sau khi huấn luyện xong một cấu hình mạng, ta cần tiến hành kiểm tra trên tập dữ liệu này để xem mạng có xấp xỉ tốt tập dữ liệu kiểm tra này hay không. Nếu sai số kiểm tra nhỏ thì mô hình mạng vừa được huấn luyện có khả năng tổng quát hóa tốt và có thể được sử dụng để dự báo. Ngược lại, nếu sai số kiểm tra lớn, ta buộc phải thực hiện việc huấn luyện mạng lại.

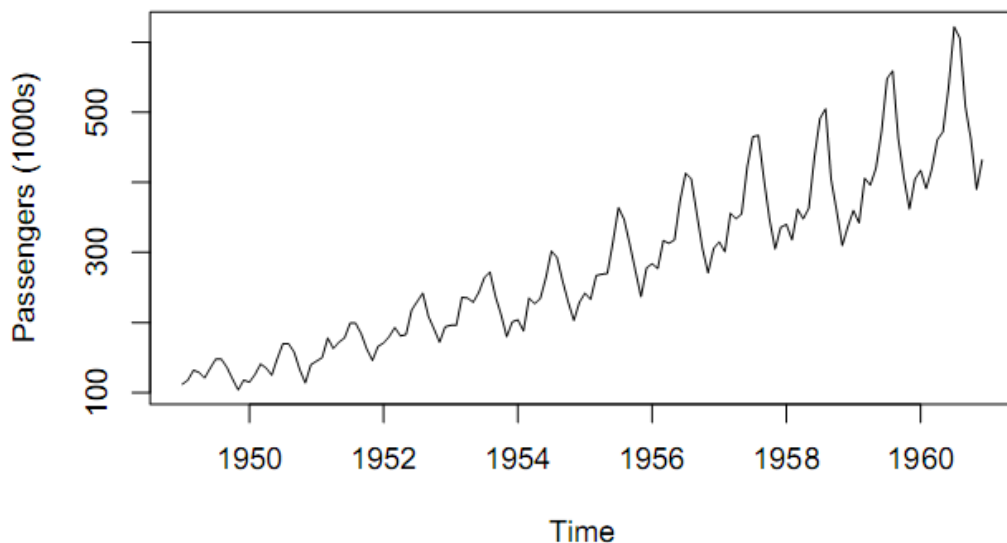
Chương 3. ỨNG DỤNG MẠNG NEURON NHÂN TẠO VÀO CÔNG TÁC DỰ BÁO DỮ LIỆU CHUỖI THỜI GIAN

3.1. DỮ LIỆU CHUỖI THỜI GIAN

3.1.1. Dữ liệu chuỗi thời gian là gì

Trong bài toán dự báo, một kiểu dữ liệu thường gặp là dữ liệu chuỗi thời gian, tức là dữ liệu được thu nhập, lưu trữ và quan sát theo sự tăng dần của thời gian. Ví dụ, số lượng thí sinh dự thi đại học vào Trường Đại Học Bách Khoa thành phố Hồ Chí Minh được lưu trữ theo từng năm, hay số lượng hàng hóa đã bán được của một siêu thị được lưu trữ theo từng quý là các dữ liệu chuỗi thời gian.

Ta ký hiệu chuỗi thời gian là $\{X_t\}$ với t là các số tự nhiên. X_t là các biến ngẫu nhiên rút ra từ một phân bố xác suất nào đó. Các chuỗi thời gian thường được biểu diễn bằng một đồ thị với trục hoành là biến thời gian. Hình 9 là một ví dụ về chuỗi thời gian, số hành khách đặt chỗ hàng tháng của hãng Pan Am



Hình 9: Số khách hàng đặt chỗ hàng tháng của hãng Pan Am

Trong chuỗi thời gian các giá trị ở những thời điểm khác nhau có mối tương quan với nhau. Sự tương quan này được đánh giá bằng hệ số tự tương quan.

Tự tương quan là sự tương quan của một biến với chính nó theo những độ trễ thời gian khác nhau [1]. Hệ số tự tương quan là đại lượng biểu diễn mức độ tự tương quan và được tính theo công thức sau:

$$\rho_k = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sqrt{Var(X_t)Var(X_{t+k})}} \quad (3.1)$$

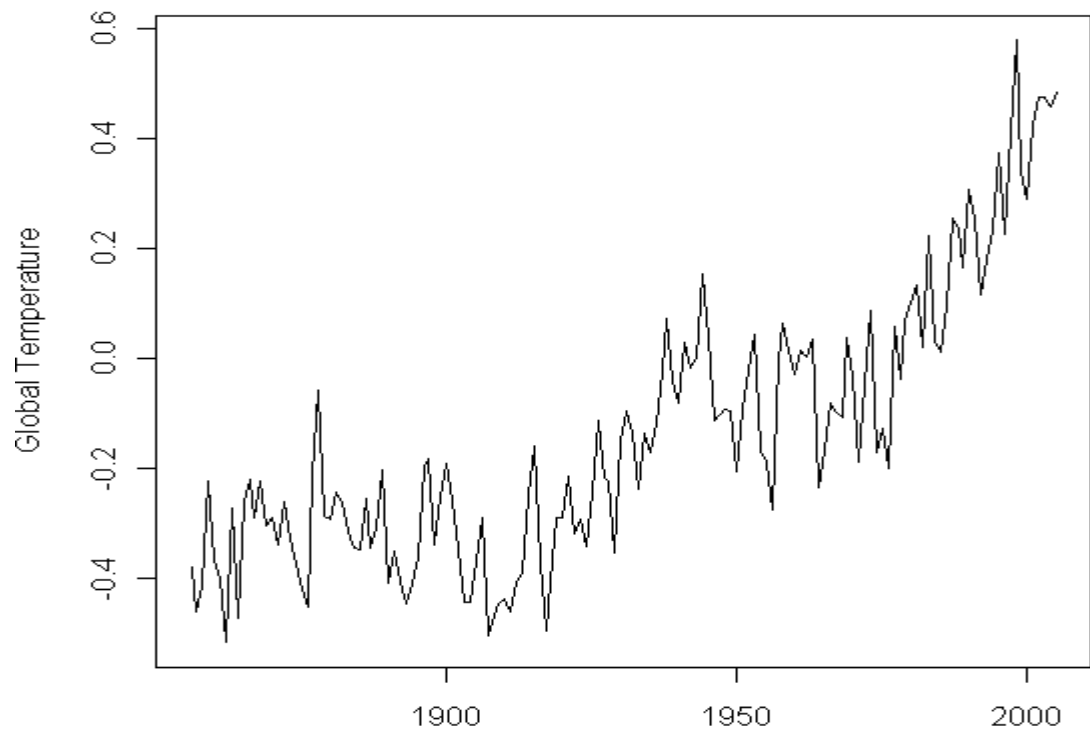
Ở đây ρ_k là hệ số tự tương quan của X ở độ trễ k , μ là giá trị trung bình của X_t . Ký hiệu $E(X)$ là kỳ vọng của biến ngẫu nhiên X , $Var(X)$ là phương sai của biến ngẫu nhiên X . Hệ số tự tương quan là một công cụ quan trọng giúp xác định các thành phần cơ bản của chuỗi thời gian.

3.1.2. Các thành phần của dữ liệu chuỗi thời gian

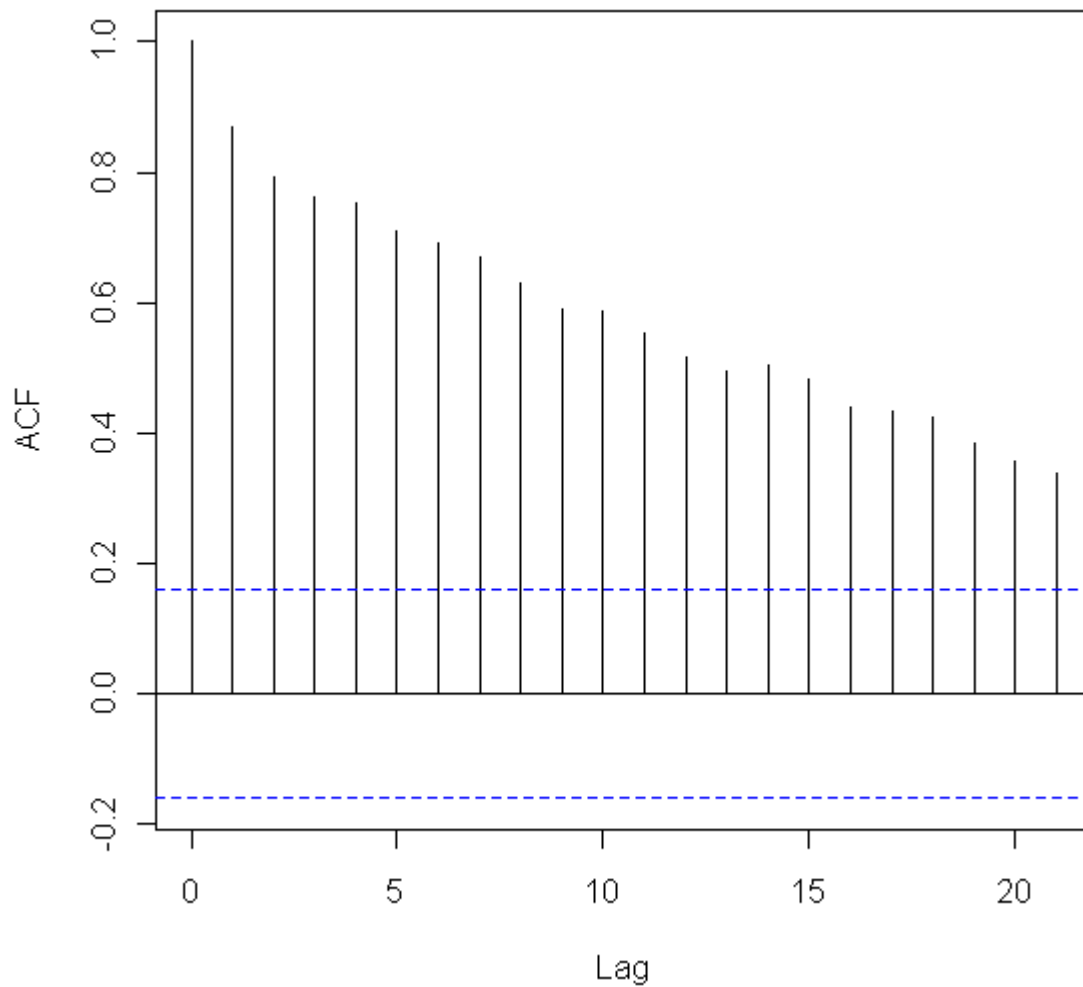
Trong thực tế, khi quan sát chuỗi thời gian ta nhận thấy bốn thành phần ảnh hưởng lên mỗi giá trị của chuỗi thời gian đó là *xu hướng* (trend), *chu kỳ* (cyclical), *mùa* (seasonal), *bất quy tắc* (irregular).

Thành phần xu hướng

Là thành phần thể hiện sự tăng hay giảm giá trị của chuỗi thời gian trong một giai đoạn dài hạn nào đó [1]. Ta có thể xác định một chuỗi thời gian có chứa thành phần xu hướng hay không bằng việc kiểm tra hàm tự tương quan của nó. Nếu một chuỗi thời gian có thành phần xu hướng sẽ có hệ số tự tương quan rất lớn ở những độ trễ đầu tiên và giảm dần về 0 khi độ trễ tăng lên. Hình 10 và 11 là một minh họa về chuỗi thời gian có thành phần xu hướng. Ở đây dù mức tăng nhiệt độ toàn cầu có biến đổi theo từng năm nhưng nhìn chung mức tăng nhiệt độ trung bình có xung hướng tăng theo thời gian. Hệ số tự tương quan rất lớn ở những độ trễ đầu tiên và giảm dần theo sự tăng của độ trễ.



Hình 10: Độ tăng nhiệt độ trung bình hàng năm từ 1856 đến 2005



Hình 11: Hàm tự tương quan của chuỗi tăng nhiệt độ trung bình hàng năm (1856 - 2005)

Thành phần chu kỳ

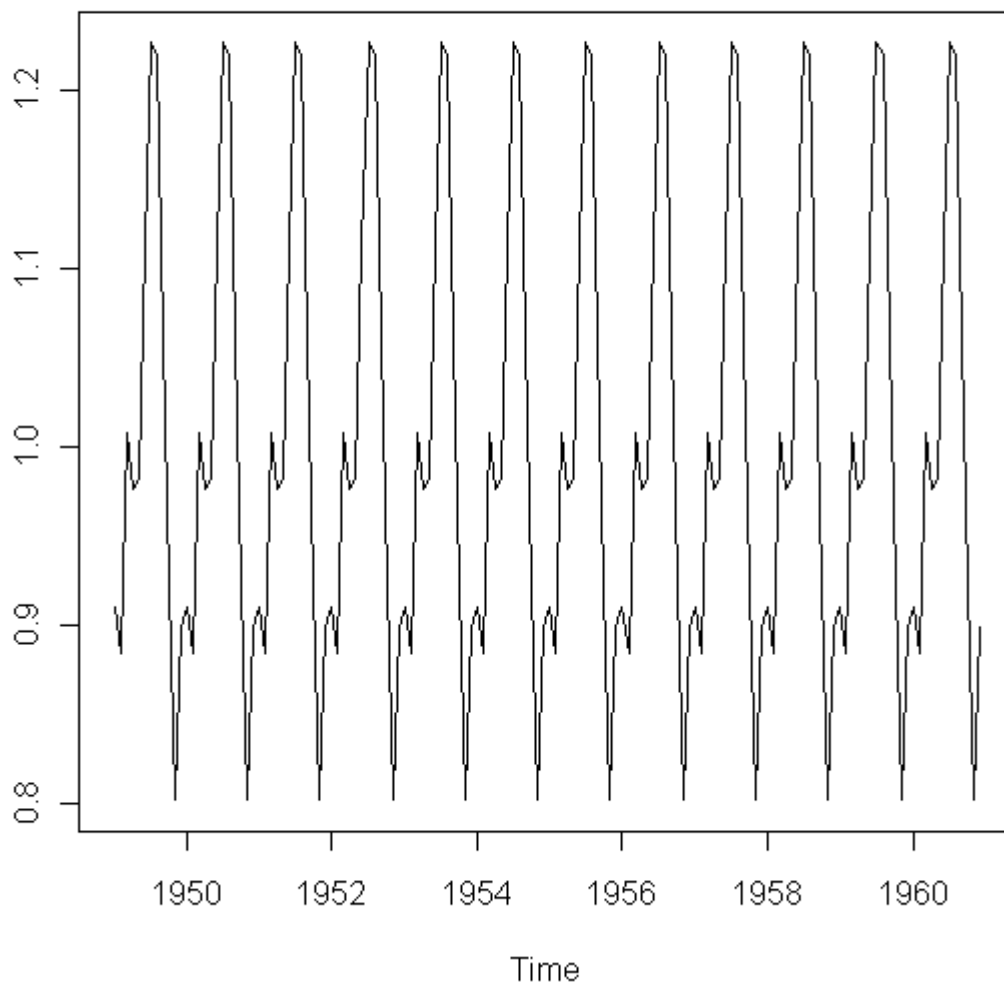
Là chuỗi biến đổi dạng sóng quanh xu hướng [1]. Trong thực tế thành phần này rất khó xác định và người ta thường xem nó như là một phần của thành phần xu hướng.

Thành phần bất quy tắc

Là thành phần thể hiện sự biến đổi ngẫu nhiên không thể đoán được của chuỗi thời gian [1].

Thành phần mùa

Là thành phần thể hiện sự biến đổi lặp đi lặp lại tại từng thời điểm cố định theo từng năm của chuỗi thời gian [1]. Đối với chuỗi thời gian có thành phần mùa thì giá trị tại những thời điểm cố định theo từng năm sẽ có sự tương quan lớn với nhau. Ví dụ một chuỗi thời gian được ghi nhận theo từng quý có tính chất mùa thì hệ số tự tương quan ở độ trễ là 4 sẽ khác không một cách có ý nghĩa. Hình 12 là đồ thị của một chuỗi thời gian có tính mùa

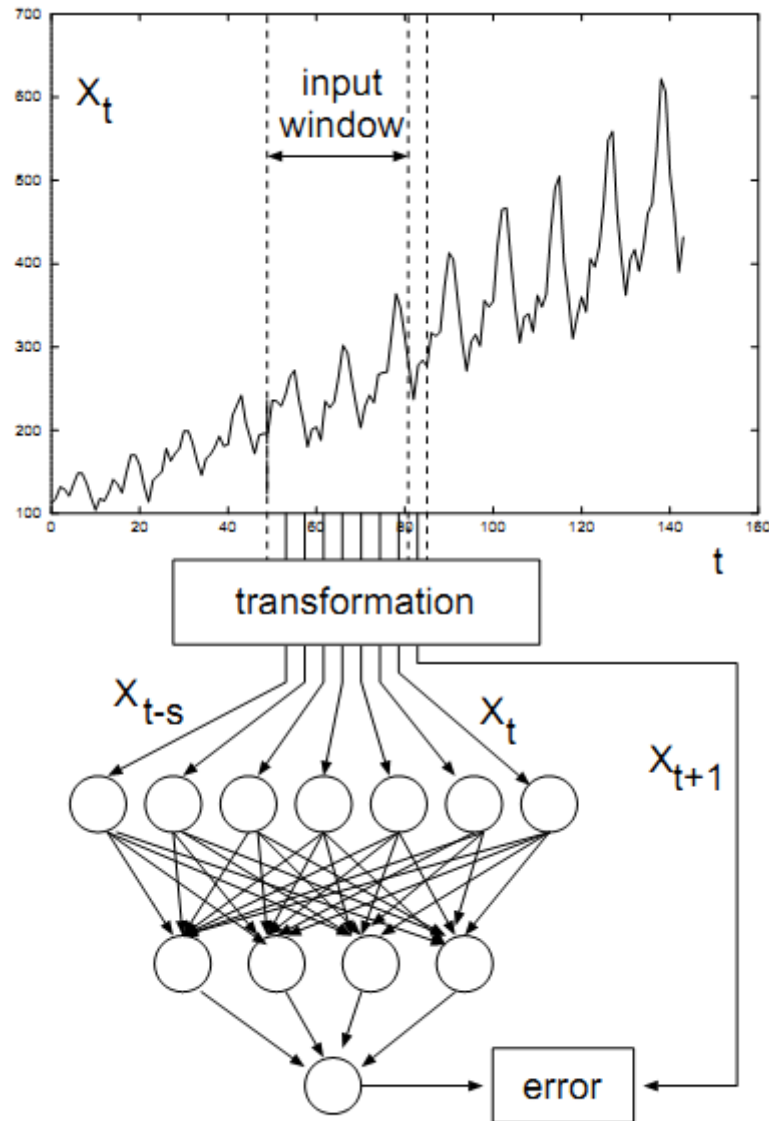


Hình 12: Chuỗi thời gian có tính mùa.

Việc xác định một chuỗi thời gian có thành phần xu hướng hay thành phần mùa không rất quan trọng trong bài toán dự đoán chuỗi thời gian. Nó giúp ta lựa chọn được mô hình dự đoán phù hợp hay giúp cải tiến mô hình đã có chính xác hơn.

3.2. Áp dụng mạng Neuron vào dự báo dữ liệu chuỗi thời gian

Việc sử dụng mạng neuron nhân tạo vào việc dự báo dữ liệu chuỗi thời gian dựa chủ yếu vào dữ liệu mà ta thu nhập. Mạng neuron nhân tạo truyền thẳng với ít nhất một lớp ẩn và đủ số perceptron cho lớp ẩn có thể xấp xỉ bất kỳ hàm khả đánh giá (measurable function) tuyến tính hay phi tuyến nào[6].



Hình 13: Mô hình học với chuỗi thời gian

Như đã đề cập ở trên, dữ liệu chuỗi thời gian là dữ liệu được thu nhập, lưu trữ và quan sát theo sự tăng dần của thời gian X_1, X_2, \dots, X_n .

Mạng neuron học cấu hình mạng từ dữ liệu chuỗi thời gian bằng cách ánh xạ từ một vector dữ liệu đầu vào sang dữ liệu đầu ra. Một số lượng dữ liệu liên tiếp của dữ liệu chuỗi thời gian (cửa sổ đầu vào $X_{t-s}, X_{t-s+1}, \dots, X_t$) được ánh xạ sang khoảng thích hợp (ví dụ $[0,1]$ hoặc $[-1,1]$) và được sử dụng như dữ liệu đầu vào của tầng nhập. Giá trị s của “cửa sổ đầu vào” tương ứng với số perceptron ở tầng nhập. Trong giai đoạn truyền tiến, những giá trị đó được truyền qua tầng ẩn rồi đến perception đầu ra. Khi truyền tới perception đầu ra, giá trị lỗi được tính toán dựa vào sự khác biệt giữa giá trị của perception đầu ra với giá trị của dữ liệu chuỗi thời gian tại thời điểm $t+1$. Sau đó, giá trị lỗi này được truyền ngược lại tới các kết nối giữa tầng ẩn và tầng đầu ra, kết nối giữa tầng đầu vào và tầng ẩn để cập nhập lại trọng số của các kết nối này.

Các cửa sổ đầu vào có thể được chọn một cách ngẫu nhiên hoặc liên tiếp nhau từ dữ liệu chuỗi thời gian. Chọn cửa sổ đầu vào một cách ngẫu nhiên sẽ phức tạp hơn, tuy nhiên sẽ đảm bảo cấu hình mạng tốt hơn và tránh được lỗi tối ưu cục bộ [6].

3.3. Các bước xây dựng một mô hình mạng neuron để dự báo dữ liệu chuỗi thời gian

Quá trình xây dựng một mô hình mạng neuron gồm 9 bước:

- Lựa chọn các biến
- Thu thập dữ liệu
- Tiền xử lý dữ liệu
- Phân chia tập dữ liệu
- Xây dựng cấu trúc mạng
- Xác định tiêu chuẩn đánh giá
- Huấn luyện mạng
- Dự đoán
- Cải tiến

Quá trình này thường không phải là một quá trình liên tiếp các bước, một số bước có thể được lặp lại đặc biệt là: lựa chọn các biến và huấn luyện mạng

3.3.1. Lựa chọn các biến

- Thành công trong việc xây dựng một mạng neuron phụ thuộc vào việc hiểu rõ ràng vấn đề cần giải quyết. Biết được những biến nào cần được xem xét là điểm mấu chốt.
- Trong bài toán dự báo các dữ liệu thương mại thì các học thuyết kinh tế có thể giúp chọn lựa các biến là các chỉ số kinh tế quan trọng. Đối với một bài toán cụ thể cần thực hiện xem xét các vấn đề lý thuyết mà từ đó sẽ xác định được các nhân tố ảnh hưởng đến bài toán. Tại bước này trong quá trình thiết kế, điều cần quan tâm đó là các dữ liệu thô từ đó có thể phát triển thành các chỉ số quan trọng. Các chỉ số này sẽ tạo ra các đầu vào cho mạng.
- Khi lựa chọn các biến, ta có thể chọn biến kỹ thuật hoặc biến cơ bản. Biến kỹ thuật bao gồm các giá trị cũ, trong quá khứ của biến đó hoặc các chỉ số được tính toán từ các giá trị cũ đó. Biến cơ bản bao gồm dữ liệu của các biến khác mà ảnh hưởng đến biến đang xem xét. Mô hình neuron đơn giản nhất sử dụng các dữ liệu của biến kỹ thuật hoặc first differencing của nó như dữ liệu đầu vào của mạng. Một mô khác cũng được áp dụng phổ biến là sử dụng dữ liệu của các biến cơ bản trong quá khứ để dự đoán.
- Tần suất của dữ liệu được ghi nhận phụ thuộc vào mục đích của nhà dự báo. Nếu dùng để dự đoán tình hình giao dịch chứng khoán thì dữ liệu được ghi nhận hàng ngày. Đối với các vấn đề đầu tư dài hạn thì các dữ liệu hàng tuần, hàng tháng được dùng làm đầu vào cho mạng neuron.

3.3.2. Thu thập dữ liệu

- Ta cần phải xem xét chi phí và khả năng có thể thu thập được dữ liệu của các biến đã chọn ra ở bước trước. Các dữ liệu kỹ thuật có thể thu thập được dễ dàng và chi phí ít tốn kém hơn là các dữ liệu cơ bản. Để đảm bảo tính chính xác của mạng neuron, ta phải đảm dữ liệu có chất lượng cao. Sau khi được thu thập, các dữ liệu phải được kiểm tra để đảm tính hợp lệ, tính nhất quán và tránh các dữ liệu bị thiếu sót.

- Các dữ liệu bị thiếu sót thường xuyên xuất hiện và có thể được xử lý bằng nhiều cách khác nhau. Các dữ liệu bị thiếu sót có thể được bỏ qua hoặc chúng có thể xem như không thay đổi so với dữ liệu trước nó, và được tính toán bằng phương pháp nội suy hoặc trung bình các giá trị lân cận.

3.3.3. Tiền xử lý dữ liệu

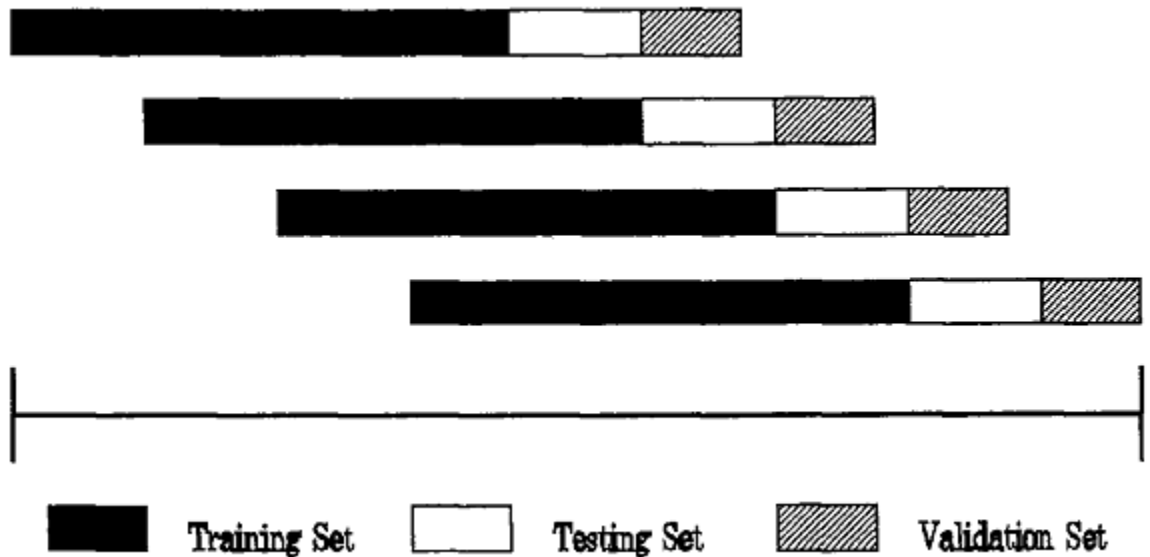
- Tiền xử lý dữ liệu liên quan đến việc phân tích và chuyển đổi giá trị các tham số đầu vào, đầu ra mạng để tối thiểu hóa nhiễu, nhấn mạnh các đặc trưng quan trọng, phát hiện các xu hướng và cân bằng phân bố của dữ liệu. Bởi vì, mạng neuron dùng để học mẫu từ tập dữ liệu, sự biểu diễn dữ liệu có vai trò quyết định trong việc học các mẫu thích hợp. Các dữ liệu dùng cho đầu vào, đầu ra của mạng neuron hiếm khi được đưa trực tiếp vào mạng dưới dạng dữ liệu thô. Chúng thường được chuẩn hóa vào khoảng giữa cận trên và cận dưới của hàm chuyển (thường là giữa đoạn $[0;1]$ hoặc $[-1;1]$).
- Hai phương pháp chuyển đổi dữ liệu thường dùng nhất là first differencing và lấy logaric tự nhiên của biến số. First differencing sử dụng sự thay đổi trong giá trị của biến số, nó có thể được sử dụng để loại bỏ khuynh hướng tuyến tính của dữ liệu. Việc lấy logaric tự nhiên của biến số là hữu ích trong trường hợp biến số lấy các giá trị rất khác nhau, sự thay đổi trong giá trị rất lớn. Việc lấy logaric tự nhiên đồng thời cũng có thể chuyển từ mối liên hệ tỷ lệ sang mối liên hệ cộng giữa các biến dự báo.
- Ngoài phương pháp first differencing và lấy logaric tự nhiên của biến số, ta có thể sử dụng tỉ số của biến đầu vào, trung bình di động. Ta có thể kết hợp các phương pháp để hạn chế dư thừa dữ liệu và cung cấp mạng với tính chính xác cao.

3.3.4. Phân chia tập dữ liệu

- Trong thực tế, khi huấn luyện, người ta thường chia tập dữ liệu thành các tập: huấn luyện, kiểm tra và kiểm định (ngoài các mẫu). Tập huấn luyện thường là tập lớn nhất được sử dụng để huấn luyện cho mạng. Tập kiểm tra thường chứa khoảng 10% đến 30% tập dữ liệu huấn luyện, được sử dụng để kiểm tra mức độ

tổng quát hóa của mạng sau khi huấn luyện. Kích thước của tập kiểm định cần được cân bằng giữa việc cần có đủ số mẫu để có thể kiểm tra mạng đã được huấn luyện và việc cần có đủ các mẫu còn lại cho cả pha huấn luyện và kiểm tra. Tập kiểm định nên bao gồm các giá trị liên tục mới nhất.

- Có hai cách thực hiện xác định tập kiểm tra. Một là lấy ngẫu nhiên các mẫu từ tập huấn luyện ban đầu. Lợi điểm của cách này là có thể tránh được nguy hiểm khi mà đoạn dữ liệu được chọn có thể chỉ điển hình cho một tính chất của dữ liệu (đang tăng hoặc đang giảm). Hai là chỉ lấy các dữ liệu ở phần sau của tập huấn luyện, trong trường hợp các dữ liệu gần với hiện tại là quan trọng hơn các dữ liệu quá khứ.
- Tập dữ liệu kiểm tra ngẫu nhiên không nên lặp lại trong tập huấn luyện, bởi vì điều này có thể làm mất khả năng tổng quát hóa của mạng neuron, đặc biệt trong trường hợp kích thước của tập kiểm tra tương đối lớn so với tập huấn luyện (khoảng 30 %). Phương pháp tắt định, như sử dụng mỗi dữ liệu thứ n làm dữ liệu kiểm tra, cũng không nên được sử dụng bởi vì nó chịu ảnh hưởng bởi tính chu kỳ của dữ liệu.
- Một phương pháp chặt chẽ dùng để đánh giá mạng neuron là walk-forward. Phương pháp walk-forward chia tập dữ liệu thành một chuỗi các tập dữ liệu nhỏ hơn huấn luyện-kiểm tra-kiểm định gộp chồng lên nhau.



Hình 14: Thủ tục sử dụng phương pháp walk-forward chia tập dữ liệu

3.3.5. Xây dựng cấu trúc mạng

- Phương pháp thực hiện xây dựng cấu trúc mạng neuron bao gồm việc xác định sự liên kết giữa các neuron, đồng thời xác định cấu trúc của mạng bao gồm số lớp ẩn, số neuron trong từng lớp. Ta có thể thực hiện lựa chọn số neuron trong các lớp ẩn bằng cách bắt đầu bằng một số nào đó dựa trên các luật. Sau khi thực hiện huấn luyện, kiểm tra lỗi tổng quát hóa của từng cấu trúc, có thể tăng hoặc giảm số các neuron.
- Việc thiết kế cấu hình một mạng neuron có ý nghĩa quyết định quan trọng trong việc dự đoán dữ liệu chuỗi thời gian. Nếu xây dựng mạng có quá nhiều tầng ẩn, hoặc số lượng perceptron ở mỗi tầng quá nhiều sẽ dẫn đến vấn đề quá khớp (overfitting). Tức là khi đó, cấu hình mạng neuron giải thích tập dữ liệu huấn luyện rất tốt, nhưng lại không có khả năng tổng quát hóa, vì thế không thể dùng cấu hình này để dự đoán. Tuy nhiên số tầng hoặc số perceptron trên mỗi tầng quá ít thì mạng neuron không có khả năng giải thích và dự đoán.
- Thực tế đã chứng minh: một mạng neuron với một tầng đầu vào, một tầng ẩn, một tầng đầu ra cùng với sự thay đổi số perceptron tại mỗi tầng là đủ để xấp xỉ bất kỳ một hàm liên tục nào [7]. Thông thường các mạng neuron được khởi tạo

với một hoặc nhiều nhất là hai lớp ẩn. Nếu kết quả huấn luyện từ mạng trên mà vẫn không thỏa mãn sau khi đã thử với nhiều giá trị khởi tạo ngẫu nhiên của trọng số thì ta nên xem xét hiệu chỉnh lại số perceptron trên các lớp ẩn hay kiểm tra dữ liệu đầu vào (ví dụ dữ liệu dùng để huấn luyện mạng có phải đã lỗi thời không?) chứ không nên tăng thêm số tầng ẩn. Cả lý thuyết và các kết quả thực nghiệm gần đây đều kết luận rằng các mạng với hơn hai tầng ẩn sẽ không cải thiện được kết quả dự đoán [7].

- Số lượng perceptron trong mỗi lớp cũng là một vấn đề cần phải xem xét vì nó cũng ảnh hưởng nhiều đến chất lượng của công tác dự báo. Số lượng các perceptron ở tầng xuất luôn là 1 cho bài toán dự báo chuỗi thời gian. Tuy nhiên việc chọn số perceptron cho tầng ẩn và tầng nhập là việc không dễ. Số perceptron ở tầng nhập bằng số giá trị trong cửa sổ nhập, việc lựa chọn này dựa trên giả định của nhà dự báo về giá trị tại thời điểm hiện tại của chuỗi thời gian sẽ bị chi phối chủ yếu bởi giá trị của bao nhiêu thời điểm trước nó. Việc lựa chọn thông số này phụ thuộc vào kinh nghiệm và sự hiểu biết của nhà dự báo vào chuỗi thời gian đang xét. Số lượng perceptron ở tầng ẩn cũng là một thông số cần phải lựa chọn cẩn thận và cũng không có một thủ tục hình thức nào giúp ta xác định được một cách tối ưu thông số này. Việc lựa chọn sao cho phù hợp phải dựa vào thực nghiệm. Thông thường có hai cách chủ yếu để tìm giá trị tối ưu cho số perceptron ở lớp ẩn. Cách thứ nhất ta chuẩn bị một nhóm các mạng neuron chỉ khác nhau số perceptron ở lớp ẩn (số lượng perceptron có thể tăng dần theo một, hai hoặc ba), sau đó ta thực hiện huấn luyện và kiểm tra các mạng này trên tập dữ liệu đã chuẩn bị. Mạng neuron có sai số nhỏ nhất là mạng có cấu hình tốt nhất. Phương pháp này khá tốn thời gian nhưng khá hiệu quả. Cách thứ hai là thay đổi số perceptron trong lớp ẩn ngay trong quá trình huấn luyện. Cách này không cần phải tạo ra nhiều mạng neuron riêng biệt nhưng lại rất phức tạp. Rất ít các hệ thống thương mại cho phép việc thay đổi số perceptron trong quá trình huấn luyện[7].
- Nhiều mô hình mạng neuron tầng vào-tầng ẩn-tầng ra đã được sử dụng hiệu quả như: 8-8-1, 6-6-1, 5-5-1 [6].

3.3.6. Xác định tiêu chuẩn đánh giá

Hàm được sử dụng để đánh giá mạng thường là hàm trung bình bình phương lỗi. Các hàm khác có thể là hàm độ lệch tuyệt đối nhỏ nhất (least absolute deviation), hiệu phần trăm (percentage differences), bình phương nhỏ nhất bất đối xứng (asymetric least squares).

3.3.7. Huấn luyện mạng

- Huấn luyện mạng để học các mẫu từ dữ liệu bằng cách lần lượt đưa các mẫu vào cùng với những giá trị mong muốn. Mục tiêu của việc huấn luyện mạng đó là tìm ra tập các trọng số cho ta giá trị nhỏ nhất toàn cục của chỉ số hiệu năng hay hàm lỗi.
- Một vấn đề quan trọng trong quá trình huấn luyện mạng neuron là xác định điều kiện dừng của quá trình huấn luyện. Có ba cách thường dùng để dừng một quá trình huấn luyện. Cách thứ nhất nhấn mạnh vào việc tránh bị rơi vào điểm tối ưu cục bộ, nhà dự báo chỉ dừng quá trình học khi không có một sự cải thiện đáng kể nào của hàm lỗi. Điểm mà mạng neuron không còn cải thiện được nữa gọi là điểm hội tụ. Cách thứ hai là sử dụng một thông số cố định là số lần lặp tối đa, quá trình huấn luyện sẽ dừng nếu số số lần lặp (epoches) vượt quá thông số này. Mạng neuron sẽ được kiểm tra, nếu kết quả không tốt thì quá trình học sẽ được tiếp tục lại. Cách thứ ba là ta sử dụng một tập dữ liệu ngoài dữ liệu huấn luyện gọi là *tập dữ liệu xác thực* (validation set). Trong quá trình huấn luyện, cứ mỗi lần vector trọng số của mạng neuron thay đổi, tập dữ liệu xác thực này sẽ được đưa vào mạng và tính ra sai số. Giải thuật huấn luyện sẽ dừng khi sai số này nhỏ hơn một ngưỡng mà nhà dự đoán mong muốn. Phương pháp này có khả năng tránh được quá khớp.

3.3.8. Dự đoán

Áp dụng mô hình mạng vừa học được để dự đoán.

3.3.9. Cải tiến

Sau một thời gian, có thể mô hình mạng không còn đúng nữa, ta cần phải tiến hành hành học lại và xây dựng mạng mới. Do đó cần phải theo dõi các sai số của mạng.

Chương 4. CHƯƠNG TRÌNH THỰC NGHIỆM

4.1. CẤU TRÚC CHƯƠNG TRÌNH

Chương trình thực nghiệm được nhóm hiện thực bằng ngôn ngữ C# trên nền tảng .NET framework với IDE hỗ trợ là Microsoft Visual Studio 2010.

Cấu trúc chương trình gồm hai lớp chính: *MainWindow* và *NeuronNetwork*

- **MainWindow:** chứa các đối tượng giao diện và một đối tượng của class *NeuronNetwork*. Công việc: xử lý các input từ phía người dùng, sau đó truyền các thông số cho *NeuronNetwork* xử lý.
- **NeuronNetwork:** đây là lớp hiện thực chính của hai giải thuật huấn luyện: giải thuật lan truyền ngược và giải thuật RPROP. Ngoài ra lớp này còn hiện thực một số chức năng khác như:
 - Hỗ trợ kiểm tra kết quả huấn luyện với kết quả quan sát thực tế
 - Hỗ trợ dự báo với mô hình mạng sau khi huấn luyện
 - Hỗ trợ import/export mô hình mạng đã huấn luyện ra file XML
 - Hỗ trợ nhập và duyệt các file dữ liệu nhằm trích xuất các giá trị cho huấn luyện, kiểm tra và dự báo
- Điều kiện dừng của quá trình huấn luyện được hiện thực dựa vào số epoch và sai số học. Quá trình học sẽ dừng khi không có sự thay đổi đáng kể của sai số qua mỗi lần lặp hay số epochs đã vượt quá một ngưỡng cố định do người dùng nhập vào (trên textbox có nhãn là Maximum Number of Epoches). Sai số được đánh giá là không thay đổi đáng kể khi độ lệch của sai số hiện thời so với sai số trong vòng lặp trước nhỏ hơn một giá trị được người dùng nhập vào (trên textbox có nhãn là Residual of Errors).

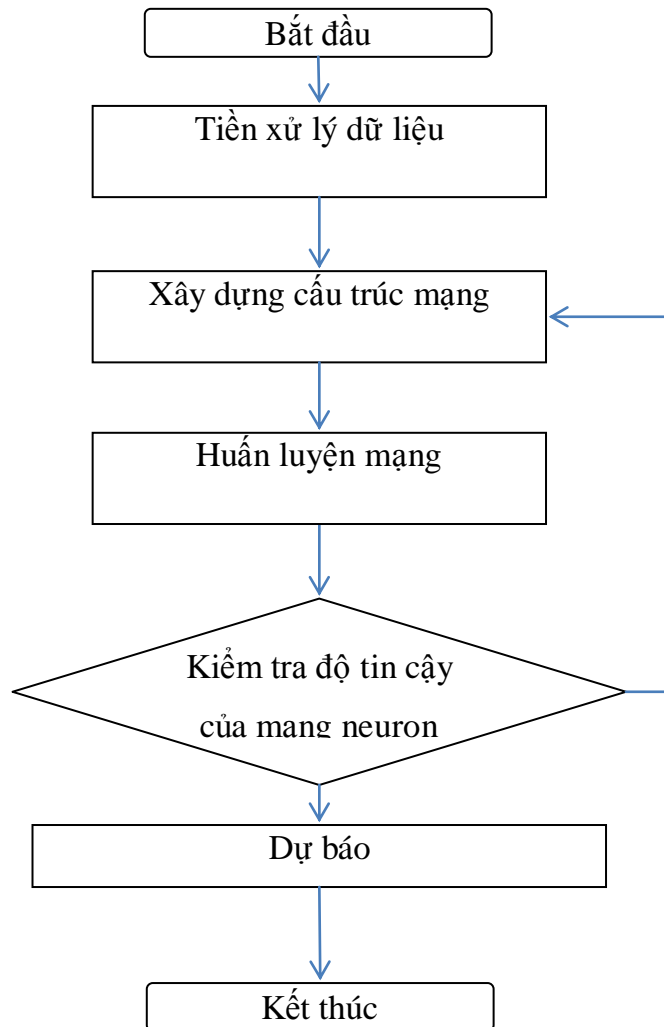
4.1.1. Cấu trúc lớp NeuronNetwork

Function	Input	Description
BasicInitForNetwork	nInputNodes, nHiddenNodes, nOutputNodes	Khởi tạo cấu trúc mạng neuron
Network	nInputNodes, nHiddenNodes, nOutputNodes	Khởi tạo cấu trúc mạng neuron và gán giá trị ngẫu nhiên cho các trọng số kết nối giữa các neuron.
Import	pathFile	Đọc cấu trúc mạng neuron đã được lưu dưới dạng tập tin xml.
Export	network, pathFile	Ghi cấu trúc mạng neuron dưới dạng tập tin xml.
Bp_Train	sample	Chuẩn hóa dữ liệu trước khi huấn luyện bằng giải thuật lan truyền ngược.
Bp_run	sample, learnRate, momentum, theEpoche, residual	Thực hiện giải thuật lan truyền ngược.
Rprop_Train	sample	Chuẩn hóa dữ liệu trước khi huấn luyện bằng giải thuật RPROP.

Ứng dụng mạng Neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian

Rprop_Train	sample, min, max, theEpoches, residual	Thực hiện giải thuật RPROP.
test	sample	Kiểm tra độ tin cậy của mạng neuron
forecast	Sample, nahead	Dự đoán dựa vào mạng neuron đã huấn luyện.
drawResult	Sample, result	Vẽ đồ thị kết quả dự đoán

4.1.2. Luồng thực thi chương trình



Hình 15: Lược đồ thực hiện đối với chương trình thực nghiệm

4.1.3. Cấu trúc định dạng file XML lưu trữ định dạng mạng

```
<Network>
  <numInputNodes>8</numInputNodes>
  <numHiddenNodes>8</numHiddenNodes>
  <numOutputNodes>1</numOutputNodes>
  <InputNodes>
    <Input1>
      <activateFunc>SIGMOID_FUNCTION</activateFunc>
      <InHid11>1.91530435885674</InHid11>
      <InHid12>-0.181711934665587</InHid12>
      ...
    </Input1>
    <Input2>
      <activateFunc>SIGMOID_FUNCTION</activateFunc>
      <InHid21>0.763952149661041</InHid21>
      <InHid22>-0.212135847035231</InHid22>
      ...
    </Input2>
    ...
  </InputNodes>
  <HiddenNodes>
    <Hidden1>
      <activateFunc>SIGMOID_FUNCTION</activateFunc>
      <bias>0.784919710264038</bias>
      <HidOut11>-3.60266904821375</HidOut11>
    </Hidden1>
    ...
  </HiddenNodes>
  <OutputNodes>
```



```
<Output1>  
  <activateFunc>SIGMOID_FUNCTION</activateFunc>  
  <bias>0.293281876152978</bias>  
</Output1>  
</OutputNodes>  
</Network>
```

Trong đó:

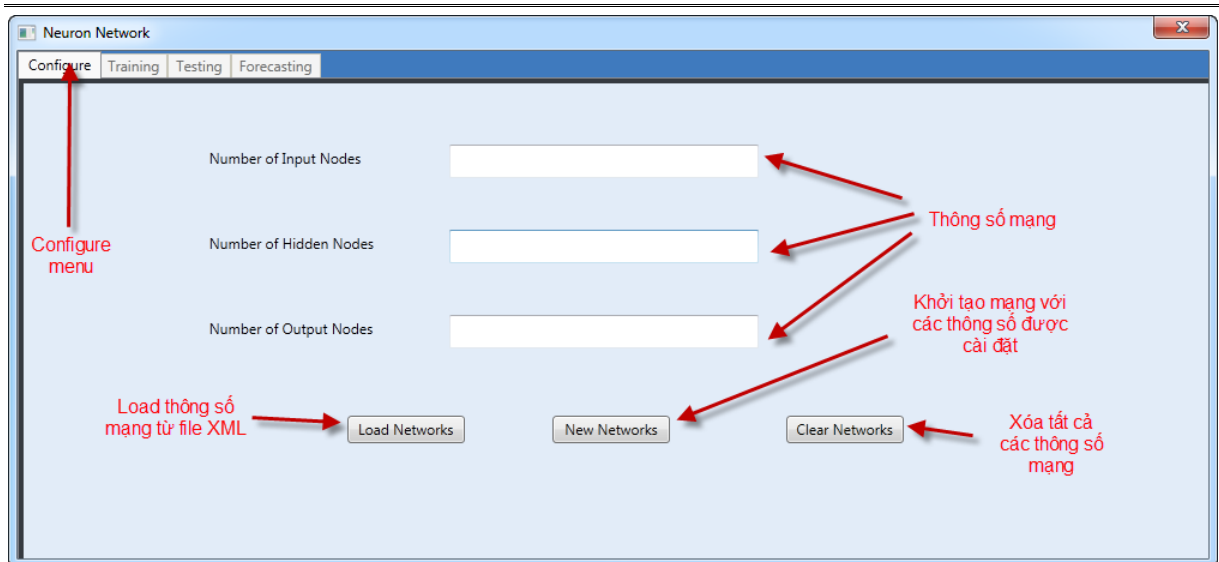
- **numInputNodes**: Số lượng perception ở tầng vào.
- **numHiddenNodes**: Số lượng perception ở tầng ẩn.
- **numOutputNodes**: Số lượng perception ở tầng ra.
- **InputNodes**: Cấu trúc của các perception ở tầng vào.
- **Input(i)**: Cấu trúc của 1 perception ở tầng vào.
- **activateFunc**: loại hàm kích hoạt.
- **InHid(I,j)**: trọng số của kết nối từ một perception tầng vào sang tầng ẩn.
- **HiddenNodes**: Cấu trúc của các perception ở tầng ẩn.
- **Hidden(i)**: Cấu trúc của 1 perception ở tầng ẩn.
- **Bias**: Độ lệch
- **HidOut(I,j)**: trọng số của kết nối từ một perception tầng ẩn sang tầng ra.
- **OutputNodes**: Cấu trúc của các perception ở tầng ra.
- **Output(i)**: Cấu trúc của 1 perception ở tầng ra.

4.2. HƯỚNG DẪN SỬ DỤNG

Quá trình sử dụng chương trình để thực nghiệm gồm các bước sau:

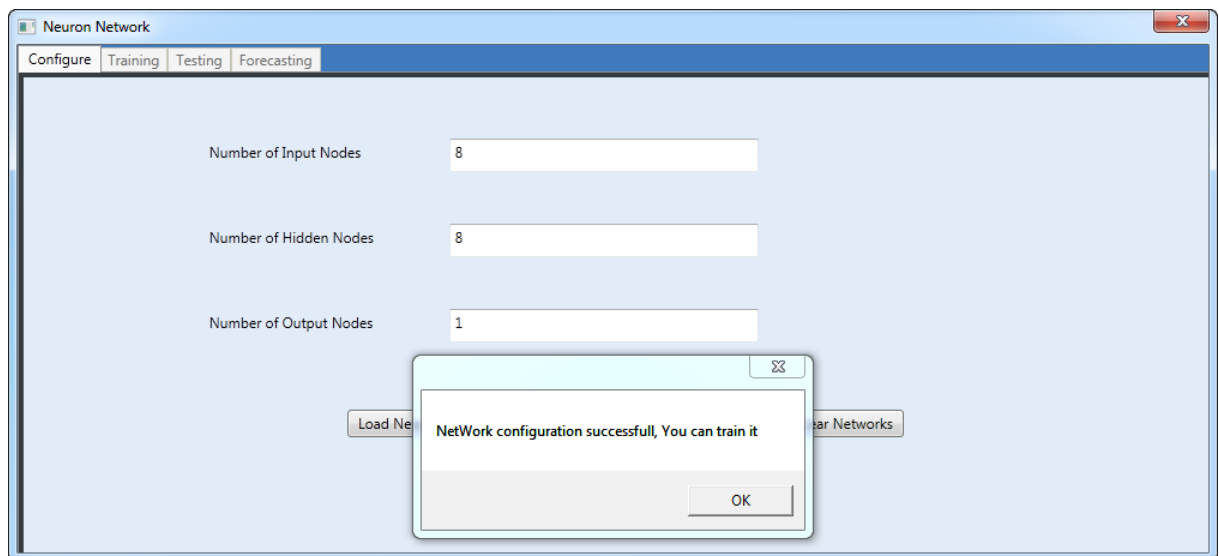
- Khởi tạo mạng với các thông số (số lượng các lớp: input, hidden, output) hoặc load các thông số mạng từ một mạng được huấn luyện sẵn.

Ứng dụng mạng Neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian



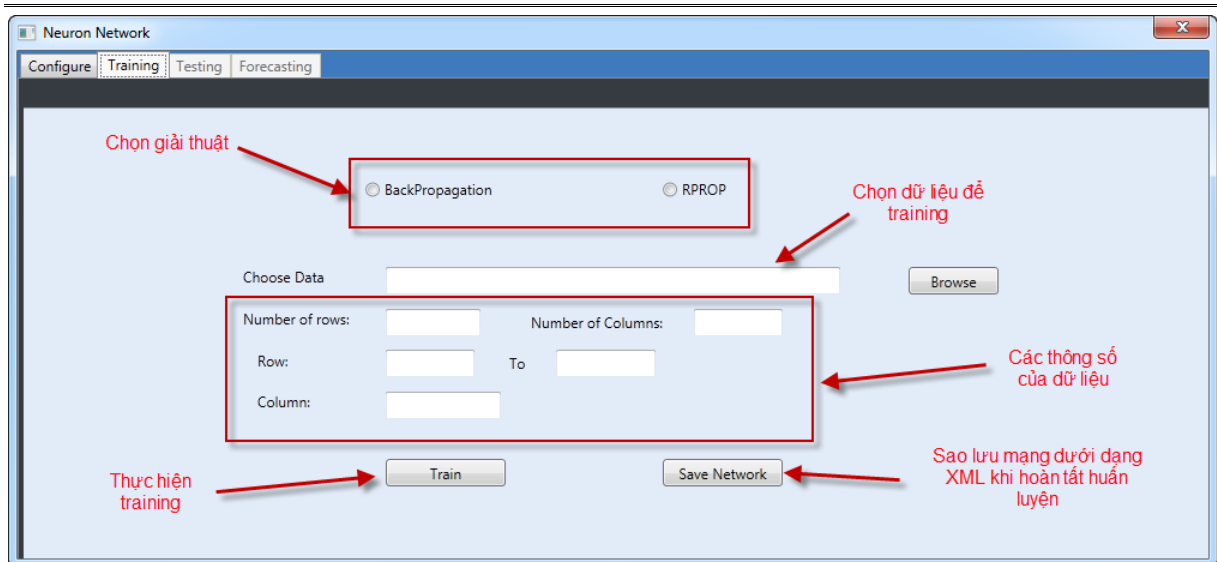
Hình 16: Giao diện configure của chương trình

Kết quả sau khi đã khởi tạo mạng thành công

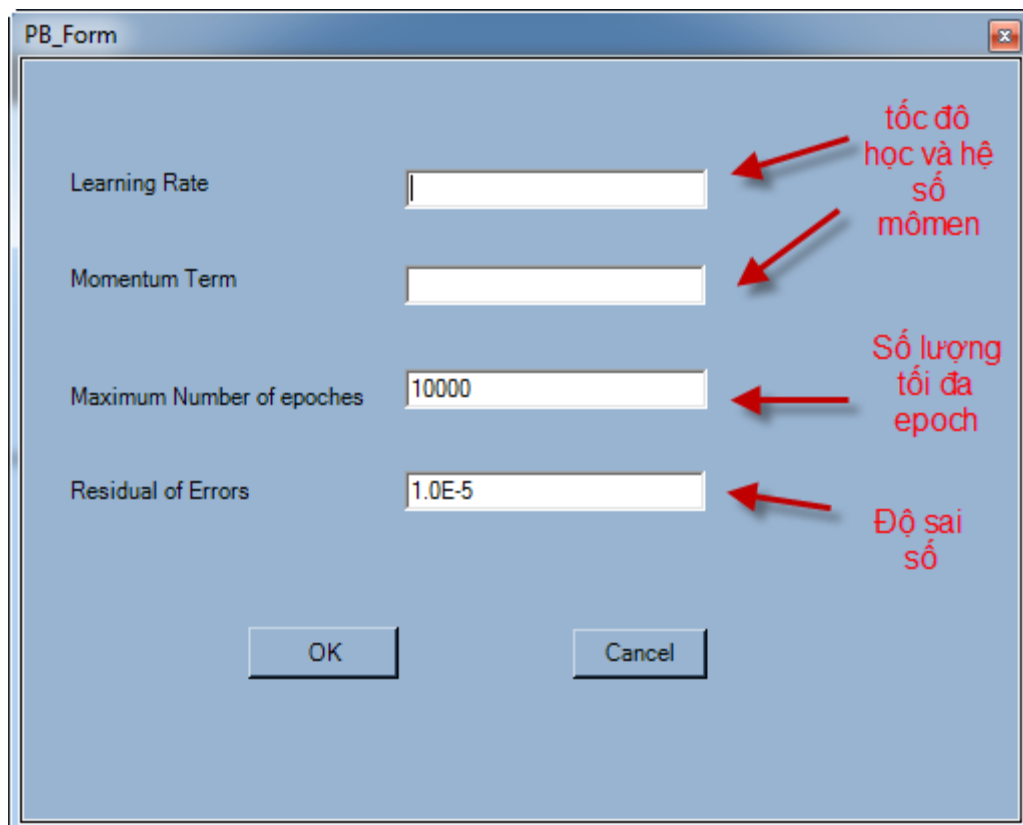


Hình 17: Giao diện khởi tạo thành công

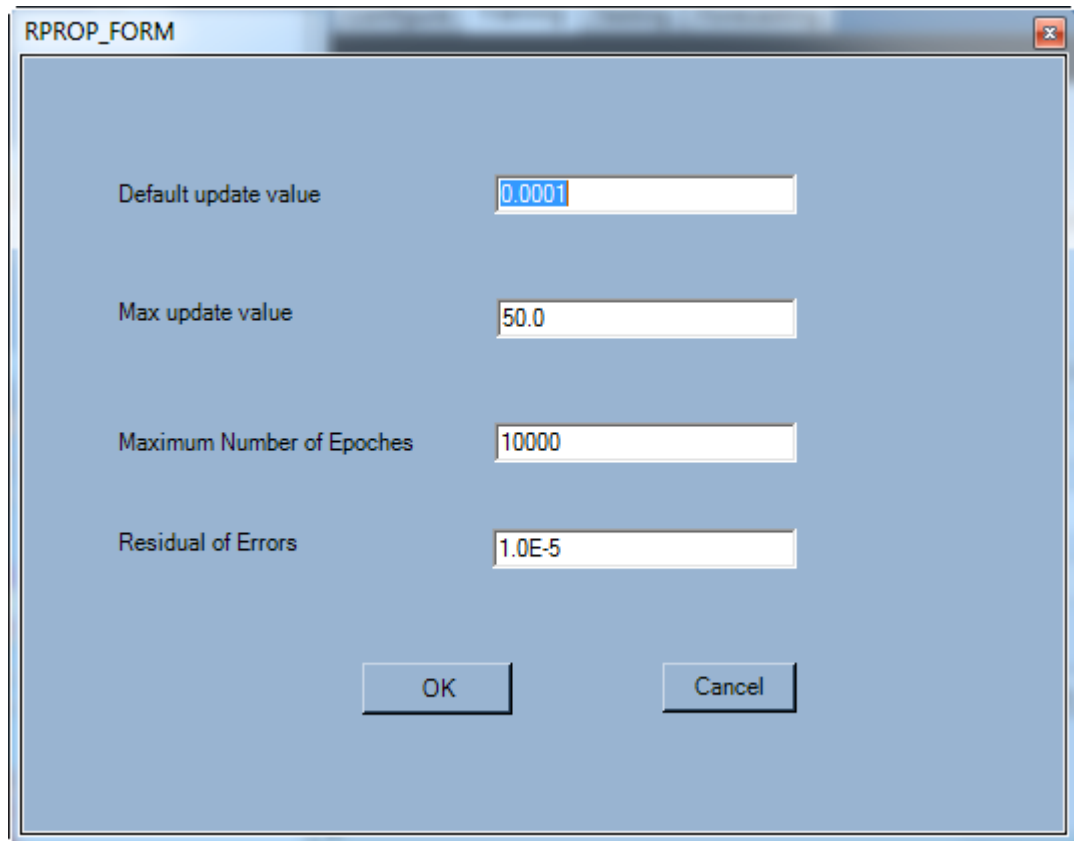
- Chọn giải thuật huấn luyện, nhập các thông số cần thiết và tiến hành huấn luyện



Hình 18: Giao diện configure cho huấn luyện



Hình 19: Giao diện cài đặt huấn luyện sử dụng giải thuật lan truyền ngược

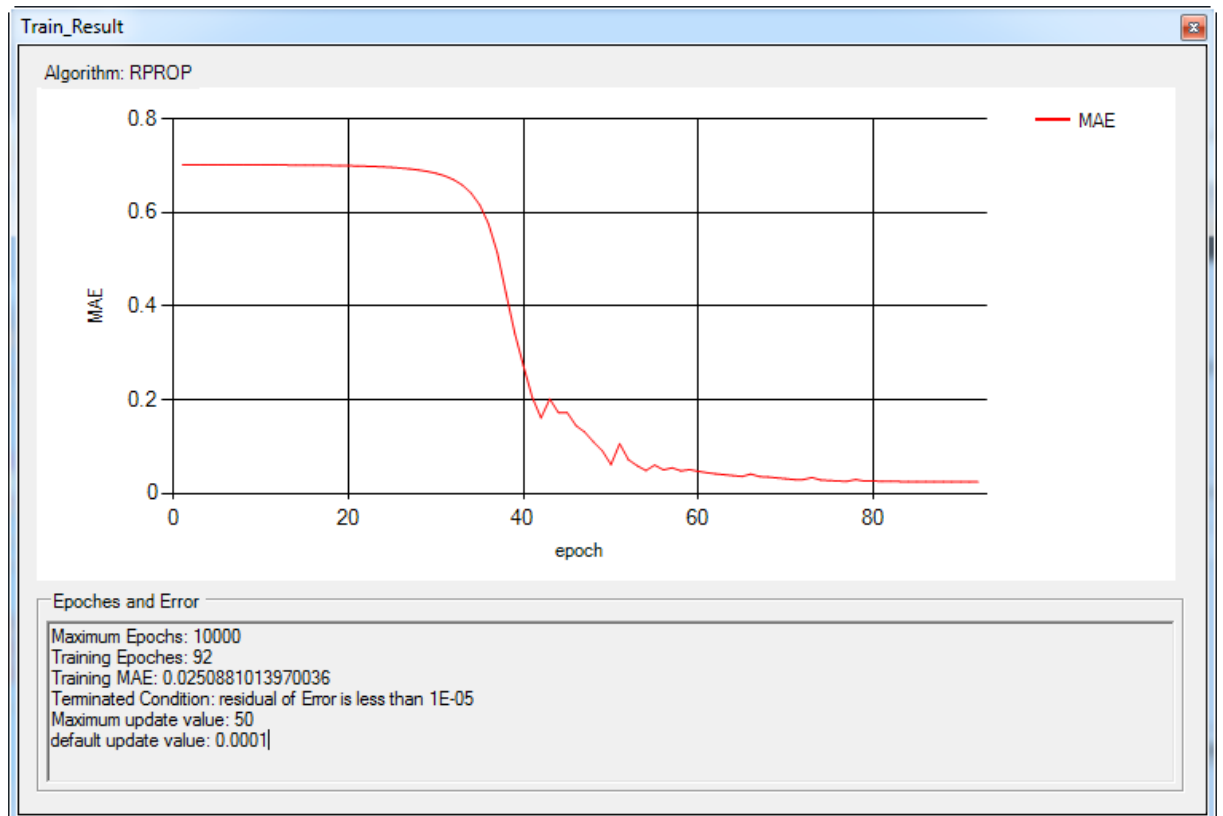


The image shows a software dialog box titled "RPROP_FORM". It contains four input fields for configuring training parameters:

- Default update value:** 0.0001
- Max update value:** 50.0
- Maximum Number of Epoches:** 10000
- Residual of Errors:** 1.0E-5

At the bottom of the dialog are two buttons: "OK" and "Cancel".

Hình 20: Giao diện cài đặt huấn luyện sử dụng giải thuật RPROP



Hình 21: Giao diện kết quả của quá trình huấn luyện

- Kiểm tra kết quả huấn luyện

Neuron Network

Configure Training Testing Forecasting

Choose Data: D:\HK112\Do An\Application\trunk\Test\Power_data\power_data.txt

Number of Rows: 35040

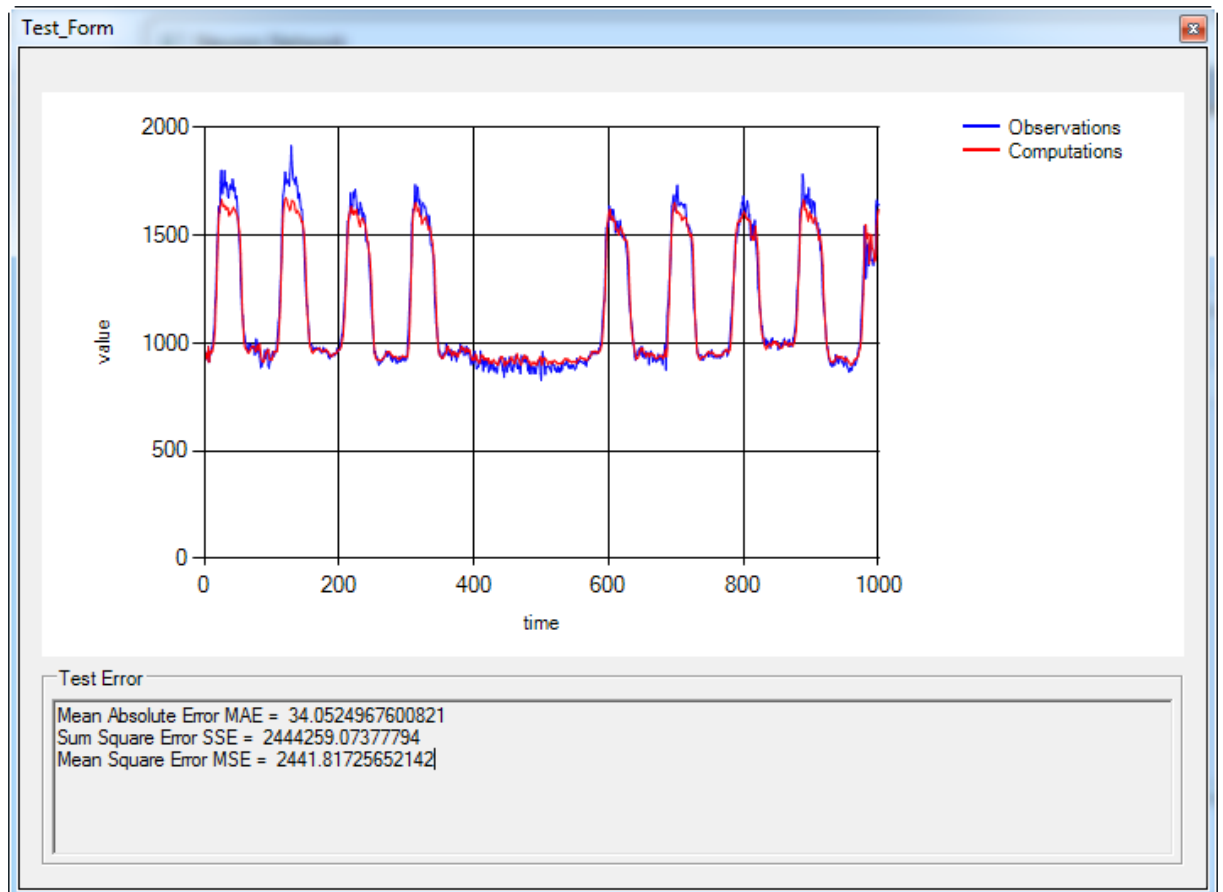
Number of Columns: 1

Row: 10000 To: 11000

Column: 1

Test

Hình 22: Giao diện kiểm tra kết quả huấn luyện



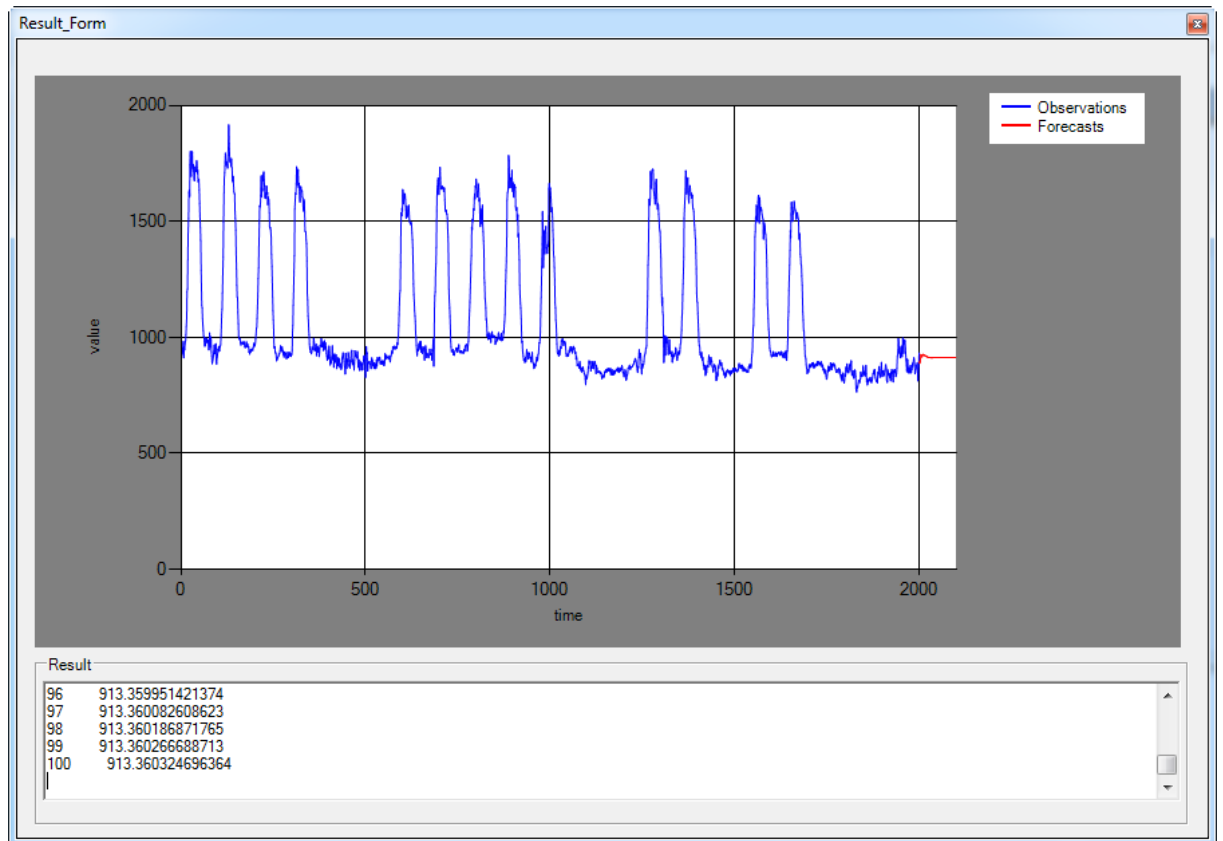
Hình 23: Giao diện kết quả kiểm tra huấn luyện

- Dự báo với mạng sau khi đã huấn luyện

The screenshot shows a window titled "Neuron Network" with a tabbed interface. The "Forecasting" tab is selected. The window contains the following configuration options:

- Choose Data:** A text box containing the file path "D:\HK112\Do An\Application\trunk\Test\Power_data\power_data.txt" and a "Browse" button.
- Number of Rows:** A text box containing "35040".
- Number of Columns:** A text box containing "1".
- Row:** A text box containing "10000".
- To:** A text box containing "12000".
- Column:** A text box containing "1".
- N ahead:** A text box containing "100".
- Forecast:** A blue button at the bottom center.

Hình 24: Giao diện cài đặt cho việc dự báo



Hình 25: Giao diện kết quả của dự báo

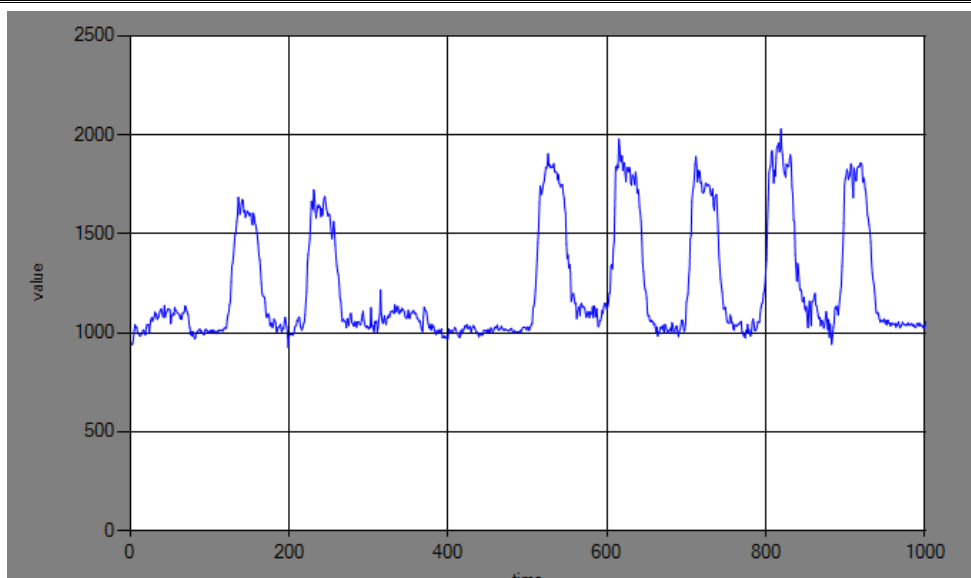
Chương 5. KẾT QUẢ THỰC NGHIỆM

Các kết quả nghiên cứu của Riedmiller[4][5] khẳng định giải thuật RPROP so với giải thuật lan truyền ngược có độ hội tụ nhanh chóng hơn và ít bị ảnh hưởng bởi các tham số của giải thuật. Trong đề tài này chúng tôi cũng thực hiện chạy các giải thuật trên các bộ dữ liệu khác nhau để so sánh trên hai tiêu chí: tốc độ hội tụ và khả năng tổng quát hóa (thể hiện qua sai số trong quá trình kiểm tra). Các dữ liệu được thực nghiệm là: dữ liệu về Nhu cầu năng lượng ở Italia, Tỷ giá giữa đồng euro và đồng đô la, Chỉ số tiêu dùng xăng dầu của người dân thành thị ở Mỹ, Tổng số người sinh theo tháng ở New York. Các bộ dữ liệu trên được lấy từ nguồn internet.

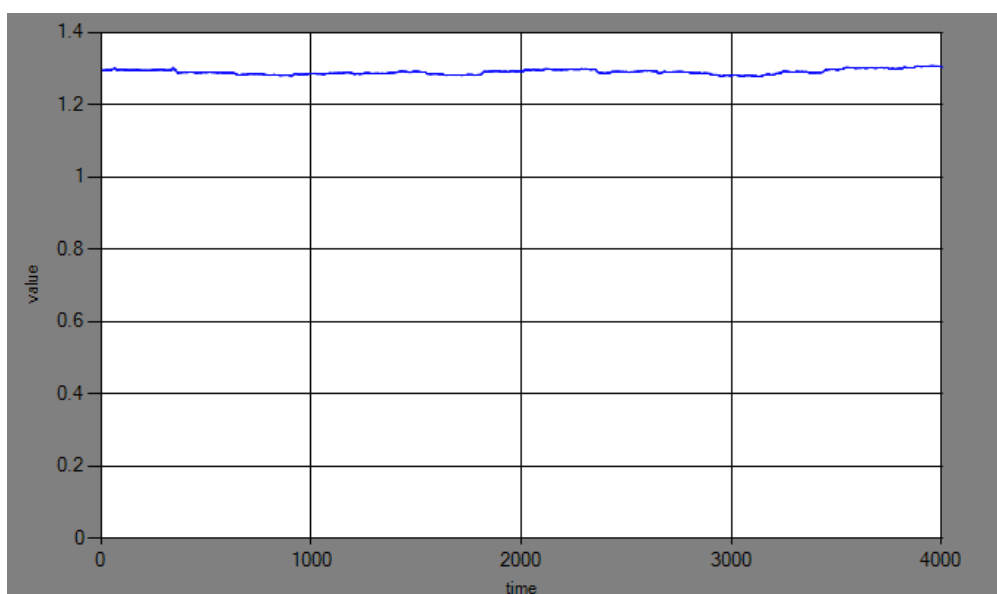
Các kết quả này được các thành viên trong nhóm chạy thực nghiệm với chương trình trên hệ điều hành Window 7, cài đặt sẵn .NET framework với cấu hình máy: Ram 3GB, CPU Core 2 Duo 2.4 GHz.

5.1. THỰC HIỆN VỚI DỮ LIỆU LỚN

Ở đây chúng tôi dùng cấu hình mạng neuron 6-6-1 để chạy trên bộ dữ liệu về Nhu cầu năng lượng ở italia và dùng cấu hình 8-8-1 để chạy trên bộ dữ liệu về Tỷ giá giữa đồng euro và đồng đô la. Đây là hai bộ dữ liệu lớn (kích cỡ khoảng vài nghìn dòng)



Hình 26: Chuỗi thời gian nhu cầu năng lượng ở italia.



Hình 27: chuỗi thời gian về tỉ giá giữa đồng euro và đồng đô la.

Đối với chuỗi Nhu cầu năng lượng ở Italia, ta được kết quả sau:

Giải thuật lan truyền ngược						
lần chạy thử	learning rate	momentum	epoches	MAE	SSE	MSE
1	0.1	0.3	221	34.19671658	6.45E+05	2.14E+03
2	0.2	0.4	257	36.72345698	7.76E+05	2.58E+03
3	0.3	0.5	173	47.44182087	1.45E+06	4.83E+03
4	0.4	0.3	166	48.04692055	1.46E+06	4.86E+03
5	0.5	0.4	140	53.23645894	1798885.885	5.98E+03
AVG			191.4	43.92907478	1.23E+06	4.08E+03

Ứng dụng mạng Neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian

Giải thuật RPROP						
lần chạy thử	Default Update	Max Update	epoches	MAE	SSE	MSE
1	0.0001	1	107	36.77817084	8.34E+05	2.77E+03
2	0.01	50	108	31.96824666	6.65E+05	2.21E+03
3	0.001	20	72	43.61600058	1.25E+06	4.16E+03
4	0.02	5	93	35.6093785	7.75E+05	2.58E+03
5	0.05	10	129	31.45905778	6.29E+05	2.09E+03
AVG			101.8	35.88617087	8.31E+05	2.76E+03

Đối với chuỗi Tỉ giá giữa đồng euro và đồng đô la, ta được kết quả là

Giải thuật lan truyền ngược						
lần chạy thử	learning rate	momentum	epoches	MAE	SSE	MSE
1	0.1	0.3	160	0.001565542	5.82E-03	4.85E-06
2	0.2	0.4	158	0.00243476	1.37E-02	1.14E-05
3	0.3	0.5	108	0.00378153	3.43E-02	2.86E-05
4	0.4	0.3	137	0.003286123	2.30E-02	1.92E-05
5	0.5	0.4	98	0.004790869	0.050568828	4.21E-05
AVG			132.2	0.003171765	2.55E-02	2.12E-05

Giải thuật RPROP						
lần chạy thử	Default Update	Max Update	epoches	MAE	SSE	MSE
1	0.0001	1	73	0.001852678	8.46E-03	7.04E-06
2	0.01	50	78	0.001757992	6.28E-03	5.23E-06
3	0.001	20	83	0.001070099	2.96E-03	2.47E-06
4	0.02	5	48	0.001522628	5.76E-03	4.80E-06
5	0.05	10	115	0.001307399	0.00430871	3.59E-06
AVG			79.4	0.001502159	5.55E-03	4.62E-06

Ở đây epoches là số epochs trong quá trình học còn các thông số MAE, SSE, MSE tương ứng là trung bình sai số tuyệt đối, tổng bình phương sai số và trung bình bình phương sai số trong quá trình kiểm tra tính tổng quát hóa của mạng đã huấn luyện

Qua kết quả trên ta thấy rằng giải thuật RPROP nhìn chung tốt hơn giải thuật lan truyền ngược về cả tốc độ hội tụ và khả năng tổng quát hóa. Đối với chuỗi nhu cầu năng lượng, số epoches và trung bình sai số tuyệt đối của giải thuật lan truyền ngược trung bình là 191 và 44, trong khi ở giải thuật RPROP là 102 và 36. Đối với chuỗi tỉ giá số, số epochs và trung bình sai số tuyệt đối của giải thuật lan truyền ngược là 132 và 0.003, trong khi ở giải thuật RPROP là 79 và 0.001.

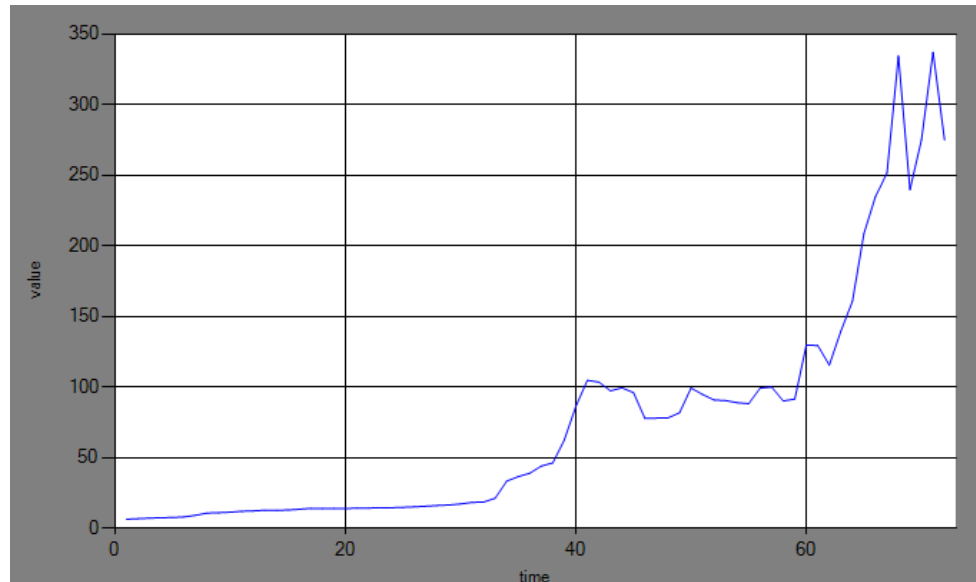
5.2. THỰC HIỆN VỚI DỮ LIỆU NHỎ

Trong trường hợp áp dụng hai giải thuật trên với bộ dữ liệu nhỏ (khoảng vài chục tới vài trăm dòng), ta sử dụng mạng neuron có cấu hình 10-10-1.

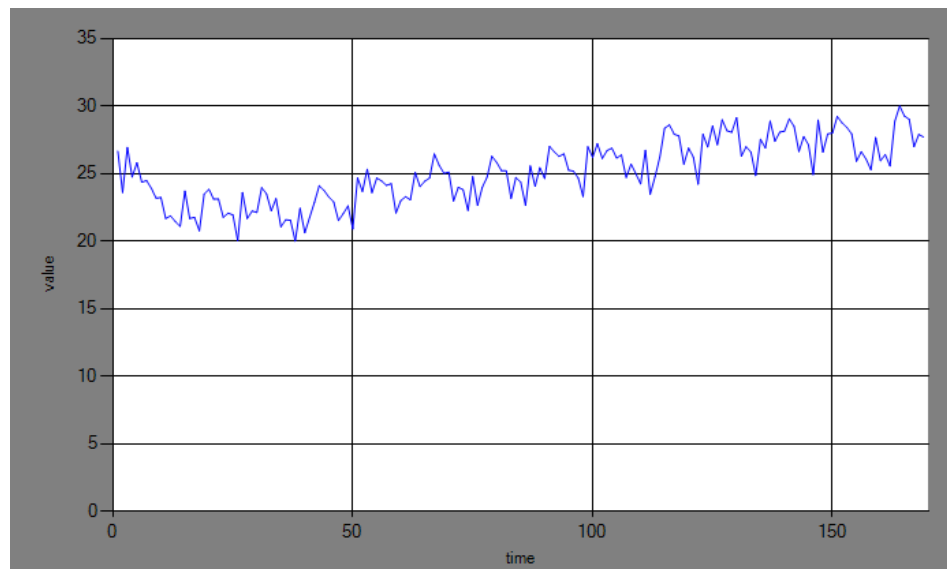
Với trường hợp này ta sử dụng 2 bộ dữ liệu:

- Chỉ số tiêu dùng xăng dầu của người dân thành thị ở Mỹ

- Tập dữ liệu thống kê tổng số người sinh theo tháng ở New-York (Mỹ) từ tháng 1-1946 đến tháng 12-1959



Hình 28: Chỉ số tiêu dùng xăng dầu của người dân thành thị ở Mỹ.



Hình 29: Số người sinh ra theo tháng

Thực nghiệm với cấu trúc mạng nơ-ron 10-10-1 với dữ liệu Số người sinh trong tháng tại New York, ta được kết quả sau:

Ứng dụng mạng Neuron nhân tạo vào công tác dự báo dữ liệu chuỗi thời gian

Giải thuật lan truyền ngược						
lần chạy thử	learning rate	momentum	epoches	MAE	SSE	MSE
1	0.01	0.5	906	0.545534173	1.93E+01	1.02E+00
2	0.1	0.5	255	0.555911222	1.65E+01	9.16E-01
3	0.2	0.5	5	0.51432318	1.42E+01	7.87E-01
4	0.2	0	13	0.510156632	1.39E+01	7.69E-01
5	0.25	0.2	2	0.511245663	13.92114155	7.73E-01
AVG	0.152	0.34	236.2	0.527434174	1.56E+01	8.53E-01

Giải thuật RPROP						
lần chạy thử	Default Update	Max Update	epoches	MAE	SSE	MSE
1	0.0001	1	14	0.507341169	1.37E+01	7.59E-01
2	0.01	50	31	0.508480913	1.37E+01	7.63E-01
3	0.01	20	21	0.506990315	1.36E+01	7.57E-01
4	0.02	5	20	0.388749689	9.85E+00	5.47E-01
5	0.1	30	29	0.388683611	9.83E+00	5.46E-01
AVG	0.02802	21.2	23	0.460049139	1.21E+01	6.74E-01

Trong trường hợp này, dữ liệu huấn luyện và dữ liệu kiểm tra là rất ít so với những trường hợp trên, tuy vậy ta vẫn thấy rất rõ tốc độ hội tụ của giải thuật RPROP là lớn hơn hẳn so với giải thuật BP. Giải thuật RPROP chỉ cần khoảng 23 epoch là hội tụ trong khi giải thuật lan truyền ngược là 236. Về sai số khi kiểm tra thì giải thuật RPROP cũng tốt hơn giải thuật lan truyền ngược tuy không đáng kể (0.46 và 0.53). Tuy nhiên ở đây ta thấy thời gian hội tụ cho các lần chạy khác nhau của giải thuật lan truyền ngược biến đổi trong một khoảng rộng. Cụ thể là khi hệ số học và hệ số quán tính là 0.1, 0.5 thì thời gian hội tụ là 255 epoches, nhưng khi hệ số học được thay đổi thành 0.2, giữ nguyên hệ số quán tính thì tốc độ hội tụ chỉ còn 5 epoch. Trong khi đó giải thuật RPROP, với sự thay đổi trong một khoảng rộng của các tham số học nhưng tốc độ hội tụ không thay đổi đáng kể. Điều đó cho thấy giải thuật lan truyền ngược nhạy cảm với các tham số đầu vào của quá trình học hơn giải thuật RPROP

Kết quả thực nghiệm với dữ liệu chỉ số tiêu dùng xăng dầu:

Giải thuật lan truyền ngược						
lần chạy thử	learning rate	momentum	epoches	MAE	SSE	MSE
1	0.01	0.5	18	5.372777063	3.49E+03	3.18E+02
2	0.2	0.4	2	5.069759497	3.11E+03	2.83E+02
3	0.0001	0.8	1	5.069702982	3.11E+03	2.83E+02
4	0.5	0.5	427	7.026527971	5.97E+03	5.43E+02
5	0.5	0.1	33	6.508909121	5126.273651	4.66E+02
AVG	0.24202	0.46	96.2	5.809535327	4.16E+03	3.78E+02

Giải thuật RPROP						
lần chạy thử	Default Update	Max Update	epoches	MAE	SSE	MSE
1	0.0001	1	32	6.421845294	4.99E+03	4.54E+02
2	0.01	50	36	6.072949861	4.42E+03	4.02E+02
3	0.5	10	27	9.306836846	1.05E+04	9.53E+02
4	0.02	5	3	8.984628711	9.77E+03	8.88E+02
5	0.05	10	47	6.842627347	5665.407431	5.15E+02
AVG	0.11602	15.2	29	7.525777612	7.06E+03	6.42E+02

Ở đây ta thấy giải thuật RPROP hội tụ nhanh hơn so với giải thuật lan truyền ngược (29 epoches so với 96 epoches) tuy nhiên sai số khi kiểm tra của mạng được huấn luyện bằng RPROP cao hơn mạng được huấn luyện bằng lan truyền ngược. Điều đó cho thấy RPROP đã rơi nhanh vào điểm hội tụ cục bộ mà không đạt được điểm tối ưu toàn cục. Như vậy không phải lúc nào giải thuật RPROP cũng tốt hơn giải thuật lan truyền ngược, khi thực hiện một công việc dự báo bằng mạng neuron, ta phải xem xét lựa chọn cẩn thận các cấu hình mạng và giải thuật học cho phù hợp với chuỗi thời gian đang quan tâm. Việc lựa chọn này được dựa trên kinh nghiệm và một thử tục lặp thử và sửa sai để tìm ra cấu hình và giải thuật tối ưu.

Qua việc chạy thử nghiệm trên bốn bộ dữ liệu trên, ta thấy chương trình hiện thực mạng neuron nhân tạo dự đoán khá chính xác. Giải thuật RPROP nhìn chung có tốc độ hội tụ và độ chính xác dự báo tốt hơn giải thuật lan truyền ngược. Tuy nhiên, đối với các tập dữ liệu có tính xu hướng như Chỉ số tiêu dùng xăng dầu hay có tính mùa như tập dữ liệu về Nhu cầu năng lượng của Italia thì sai số dự đoán khá lớn. Điều đó chứng tỏ cần phải có nhiều cải tiến cho chương trình hiện thực mạng neuron này để đưa ra các dự báo chính xác hơn.

Chương 6. KẾT LUẬN

6.1. ĐÁNH GIÁ KẾT QUẢ

6.1.1. Những công việc làm được

Trong quá trình thực hiện đề tài, chúng tôi đã làm được những công việc sau:

Tìm hiểu về dữ liệu chuỗi thời gian và các tính chất của nó

Tìm hiểu về cấu trúc và nguyên lý hoạt động của mạng neuron nhân tạo.

Tìm hiểu về hai giải thuật huấn luyện mạng neuron nhân tạo là giải thuật lan truyền ngược và giải thuật RPROP.

Hiện thực một chương trình dự báo chuỗi thời gian bằng mạng neuron nhân tạo cho phép người dùng lựa chọn một trong hai giải thuật huấn luyện: lan truyền ngược và RPROP.

Thực hiện chạy thử nghiệm chương trình trên các bộ dữ liệu mẫu

6.1.2. Những đúc kết về mặt lý luận

Chất lượng dự báo của mạng neuron nhân tạo phụ thuộc nhiều vào cấu hình của mạng (số lớp, số đơn vị mỗi lớp) và các tham số của giải thuật huấn luyện

Về mặt lý thuyết mạng neuron nhân tạo có khả năng xấp xỉ bất cứ hàm liên tục nào, điều này đã làm cho mạng neuron trở thành một công cụ mạnh trong công tác dự báo chuỗi thời gian. Tuy nhiên để tìm một cấu hình tối ưu cho một mạng neuron nhân tạo trong công tác dự báo một chuỗi thời gian nào đó là một việc khó khăn. Ta phải tiến hành việc lựa chọn bằng việc xây dựng nhiều cấu hình khác nhau, qua một quá trình lặp các công đoạn huấn luyện và kiểm tra lựa chọn một cấu hình tốt nhất. Trong quá trình áp dụng mô hình mạng để dự báo, khi các giá trị mới được thu nhập sai số nhiều so với kết quả dự báo, ta cần phải tiến hành huấn luyện lại mạng với các dữ liệu mới.

Giải thuật RPROP tốc độ hội tụ và độ chính xác nhìn chung tốt hơn giải thuật lan truyền ngược và cũng ít bị chi phối bởi việc lựa chọn các tham số huấn luyện. Tuy nhiên điều này không luôn đúng cho mọi tập dữ liệu, vì vậy để thực hiện một công tác dự báo chuỗi thời gian bằng mạng neuron nhân tạo ta nên xem xét lựa chọn và thử nghiệm trên nhiều cấu hình và giải thuật khác nhau để tìm ra một mô hình tốt nhất.

6.1.3. Mặt hạn chế

Đối với những chuỗi thời gian có xu hướng và tính mùa chương trình dự đoán với độ chính xác chưa cao.

Giao diện chương trình hiện thực chưa thân thiện với người dùng.

6.2. HƯỚNG PHÁT TRIỂN

Trong thời gian tới, chúng tôi sẽ tìm hiểu thêm về các phương pháp cải tiến cho mạng neuron nhân tạo để có thể đưa ra những dự báo chính xác cho các chuỗi có xu hướng và tính mùa

TÀI LIỆU THAM KHẢO

- [1] John E. Hanke, Dean W. Wichern. *Business Forecasting*, Pearson Prentice Hall, ISBN 0-13-141290-6, 2005.
- [2] Tom M. Mitchell. *Machine Learning*, McGraw-Hill Science/ Engineering/ Math, ISBN 0070428077, 1997.
- [3] Trần Đức Minh. Luận văn thạc sĩ *Mạng Neural Truyền Thẳng Và Ứng Dụng Trong Dự Báo Dữ Liệu*. Đại học quốc gia Hà Nội, 2002
- [4] Martin Riedmiller, Heinrich Braun. *A Direct Adaptive Method For Faster Backpropagation Learning: The RPROP Algorithm*. Proceedings of the IEEE International Conference on Neural Networks, pages 586-591, 1993.
- [5] Martin Riedmiller. *Advanced Supervised Learning In Multi-layer Perceptrons – From Backpropagation To Adaptive Learning Algorithms*. Int. Journal of Computer Standards and Interfaces, 1994.
- [6] Thomas Kolarik, Gottfried Rudorfer. *Time Series Forecasting Using Neural Networks*. ACM Sigapl Apl Quote Quad, vol. 25, no. 1, pages 86-94, 1994.
- [7] Iebling Kaastra, Milton Boyd. *Designing A Neural Network For Forecasting Financial And Economic Time Series*. Neurocomputing, 10, pages 215-236, 1996.
- [8] Gioqinang Zhang, Michael Y. Hu. *Neural Network Forecasting Of The British Pound/US Dollar Exchange Rate*. Omega, International Journal of Management Science, 26, pages 495-506, 1998.