

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



BÁO CÁO LUẬN VĂN TỐT NGHIỆP

Đề tài: Kết hợp giải thuật lan truyền ngược và mô phỏng luyện kim để huấn luyện mạng nơ ron cho công tác dự báo dòng chảy trên sông

GVHD: PGS.TS Dương Tuấn Anh

GVPB:

---o0o---

SVTH : Nguyễn Huy Khánh 51201637

TP. HỒ CHÍ MINH, 12/2016

LỜI CAM KẾT

Tôi xin cam đoan rằng ngoại trừ các kết quả tham khảo từ các công trình khác như đã ghi rõ trong luận văn, các công việc trình bày trong luận văn này là do chính tôi thực hiện và chưa có phần nội dung nào của luận văn này được nộp để lấy một bằng cấp ở trường đại học nào khác.

TP.Hồ Chí Minh, ngày tháng năm.....

Ký tên

LỜI CẢM ƠN

Đóng góp vào thành công trong cuộc sống của mỗi con người, không chỉ dựa vào tài năng, may mắn mà còn một phần quan trọng là từ sự giúp đỡ của những người xung quanh. Trong quá trình học tập tại trường ĐH Bách Khoa TP.HCM, tôi không chỉ nhận được sự chỉ dạy tận tình của quý thầy cô mà còn là sự động viên san sẻ từ gia đình và bạn bè.

Xin gửi lời cảm ơn sâu sắc nhất đến quý thầy cô khoa Khoa Học và Kỹ Thuật Máy Tính đã mang tri thức và tâm huyết truyền đạt vốn kiến thức quý giá của mình đến bao thế hệ sinh viên. Và đặc biệt, tôi muốn gửi lời cảm ơn chân thành và sâu sắc nhất đến PGS.TS Dương Tuấn Anh, người đã trực tiếp hướng dẫn, chỉ dạy tận tình tôi hoàn thành luận văn này. Chỉ với định hướng nghiên cứu ban đầu sơ khai mà thiếu đi sự dìu dắt chỉ bảo của thầy, chắc chắn tôi đã không thể hoàn thiện được luận văn tốt nghiệp như ngày hôm nay. Thầy truyền dạy lời khuyên, tài liệu cùng với kiến thức tích lũy vô cùng quý giá, là hành trang tôi mang theo không chỉ để thực hiện luận văn mà còn để ứng dụng trong cuộc sống sau này. Tôi cũng muốn gửi lời cảm ơn cùng tình cảm yêu quý nhất đến gia đình và bạn bè, những người đã sát cánh cùng tôi một chặng đường dài, đã động viên, chia sẻ và tạo điều kiện tốt nhất cho tôi hoàn thành luận văn này.

Do tầm nhìn của bản thân còn hạn hẹp, chắc chắn luận văn khó tránh khỏi những sai sót, rất mong nhận được những ý kiến đóng góp chuyên môn bổ ích đến từ phía Thầy cô và các bạn để Luận văn có thể hoàn thiện hơn.

Tôi xin chân thành cảm ơn.

TÓM TẮT LUẬN VĂN

Bài toán sử dụng mạng Nơ-ron nhân tạo trong công tác dự báo dữ liệu chuỗi thời gian ngày càng được quan tâm và sử dụng rộng rãi. Nhiều phương pháp huấn luyện mạng được đưa ra nhằm cải tiến phương pháp huấn luyện ban đầu đó là phương pháp lan truyền ngược.

Trong luận văn này, chúng tôi sẽ xây dựng phương pháp huấn luyện mạng Nơ-ron nhân tạo bằng sự kết hợp giải thuật lan truyền ngược và giải thuật mô phỏng luyện kim. Mô hình này sẽ giúp loại bỏ điểm yếu trước đây của giải thuật lan truyền ngược, giải thuật mô phỏng luyện kim sẽ giúp mạng neuron thoát ra khỏi điểm tối ưu cục bộ và hướng tới kết quả tốt hơn.

Trong quá trình thực nghiệm, mô hình huấn luyện mạng Nơ-ron bằng sự kết hợp giải thuật lan truyền ngược và giải thuật mô phỏng luyện kim cho ra kết quả dự báo tốt hơn so với mô hình huấn luyện bằng giải thuật lan truyền ngược đơn thuần.

ABSTRACT

Nowadays, time series forecasting gets more and more attention from the scientific community. There are several methods which have been introduced to improve training algorithm for artificial neuron networks.

In this work, we will apply a algorithm that combine Backpropagation algorithm and Simulated Annealing algorithm to train artificial neuron network. Backpropagation is a local search algorithm that is easy to be stick in local maximum. Simulated Annealing method will take it out of the local maximum and find better solution.

We also implement and use many real-world time series data sets to test the two algorithms and find that the combination gives better forecast values than neuron network with Backpropagation.

MỤC LỤC

LỜI CAM KẾT.....	ii
LỜI CẢM ƠN.....	iii
TÓM TẮT LUẬN VĂN.....	iv
ABSTRACT.....	v
DANH MỤC HÌNH.....	viii
DANH MỤC BẢNG	ix
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 CƠ SỞ LÝ THUYẾT	3
2.1. SƠ LƯỢC VỀ MẠNG NEURON NHÂN TẠO	3
2.2. NGUYÊN TẮC HOẠT ĐỘNG CỦA MẠNG NEURON NHÂN TẠO.....	6
2.3. ỨNG DỤNG MẠNG NEURON TRONG CÔNG TÁC DỰ BÁO	12
2.4. HUẤN LUYỆN MẠNG NEURON	14
2.4.1. GIẢI THUẬT LAN TRUYỀN NGƯỢC	14
2.4.2. GIẢI THUẬT TÌM KIẾM NGẪU NHIÊN CỤC BỘ	19
2.4.3. GIẢI THUẬT MÔ PHỎNG LUYỆN KIM	21
2.4.4. GIẢI THUẬT MÔ PHỎNG LUYỆN KIM HUẤN LUYỆN MẠNG NEURON	24
Chương 3 HIỆN THỰC VÀ KẾT QUẢ THỰC NGHIỆM.....	26
3.1. CÁCH ĐÁNH GIÁ KẾT QUẢ HUẤN LUYỆN	26
3.2. KẾT QUẢ THỰC NGHIỆM.....	27
3.2.1. THỰC HIỆN TRÊN CÁC BỘ DỮ LIỆU NHỎ	28
3.2.2. THỰC HIỆN TRÊN CÁC BỘ DỮ LIỆU LỚN.....	33
3.2.3. ĐÁNH GIÁ	39
Chương 4 KẾT LUẬN	40

4.1.	ĐÁNH GIÁ KẾT QUẢ	40
4.1.1.	Những công việc làm được	40
4.1.2.	Những đúc kết về mặt lý luận	40
4.1.3.	Mặt hạn chế	40
4.2.	HƯỚNG PHÁT TRIỂN.....	41
TÀI LIỆU THAM KHẢO		42
Phụ lục A Bảng thuật ngữ Anh-Việt		1
Phụ lục B Chương trình thực nghiệm.....		1
B.1 Giao diện và hướng dẫn sử dụng.....		1
B.2 Cấu trúc cơ bản các lớp và các hàm trong chương trình:		3
B.3 Cấu trúc định dạng file XML lưu trữ định dạng mạng.....		4

DANH MỤC HÌNH

Hình 1.1 Tổng số lượng hành khách trên các chuyến bay quốc tế được lưu trữ theo từng tháng từ tháng 1 năm 1949 đến tháng 12 năm 1960	1
Hình 2.1: Đơn vị mạng neuron	3
Hình 2.2: Mạng neuron truyền thẳng	5
Hình 2.3: Mạng neuron hồi quy	5
Hình 2.4: Mô hình học có giám sát	6
Hình 2.5: Đơn vị mạng Nơ-ron	7
Hình 2.6: Mặt quyết định biểu diễn bởi perceptron hai đầu nhập	8
Hình 2.7: Hàm lỗi của một đơn vị tuyến tính	10
Hình 2.8: Mô hình học với chuỗi thời gian	13
Hình 2.9: Đơn vị sigmoid	14

DANH MỤC BẢNG

Bảng 3.1	28
Bảng 3.2	29
Bảng 3.3	29

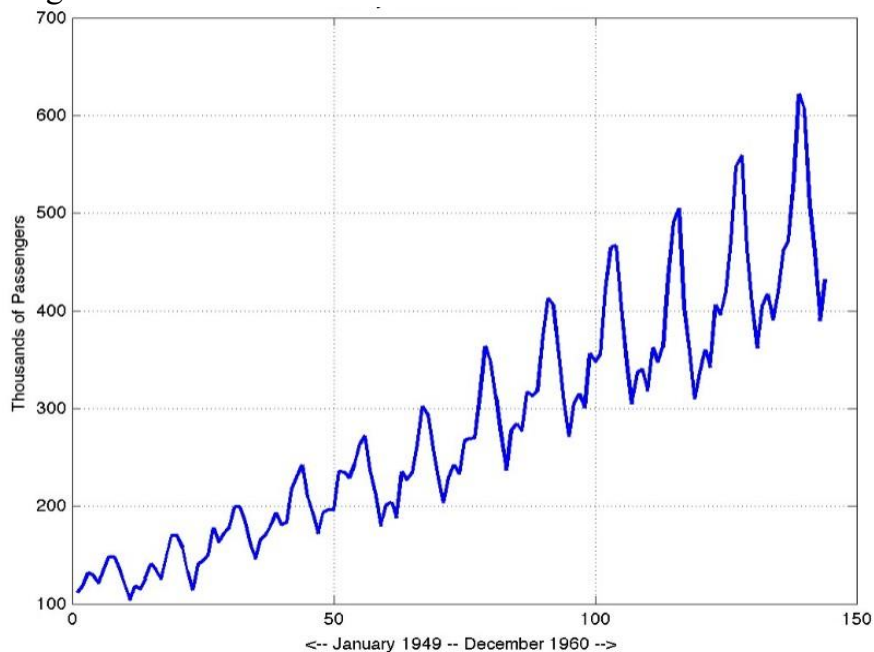
Chương 1

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 hoạt động 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* (time series data). 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, tổng số lượng hành khách trên các chuyến bay quốc tế được lưu trữ theo từng tháng (Hình 1.1), 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.



Hình 1.1 Tổng số lượng hành khách trên các chuyến bay quốc tế được lưu trữ theo từng tháng từ tháng 1 năm 1949 đến tháng 12 năm 1960

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 hàm mũ, 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, áp dụng các giải thuật huấn luyện mạng neuron: *lan truyền ngược* (backpropagation), *mô phỏng luyện kim* (simulated annealing) và sự kết hợp đồng thời 2 giải thuật trên để huấn luyện. Sau đó, đánh giá khả năng dự báo của các mạng neuron khác nhau thông qua các bộ dữ liệu thực tế.

1.3. CẤU TRÚC BÁO CÁO

Nội dung báo cáo được chia làm 4 chương như sau:

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

Chương 2: Cơ sở lý thuyết

Chương 3: Hiện thực và kết quả thực nghiệm

Chương 4: Kết luận

Chương 2

CƠ SỞ LÝ THUYẾT

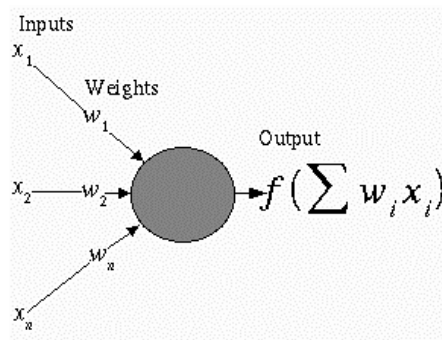
Trong quá trình thực hiện đề tài này, một số nền tảng lý thuyết cần phải được hiểu rõ. Những nền tảng lý thuyết này rất quan trọng, chúng là tiền đề để chúng ta hiểu được việc chúng ta đang làm, từ đó đem lại sự hiệu quả và chính xác trong quá trình giải quyết vấn đề sau này.

2.1. SƠ LƯỢC VỀ MẠNG NEURON NHÂN TẠO

Mạng neuron nhân tạo (Artificial Neuron Network) là một mô hình toán học định nghĩa một hàm số từ một tập đầu vào đến một tập đầu ra[1]. Mạng neuron nhân tạo được mô phỏng theo mạng neuron sinh học trong bộ não người.

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 2.1) 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 tín 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 2.1: Đơn vị mạng neuron

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:

$$g(x) = x$$

- Hàm ngưỡng:

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

- Hàm sigmoid:

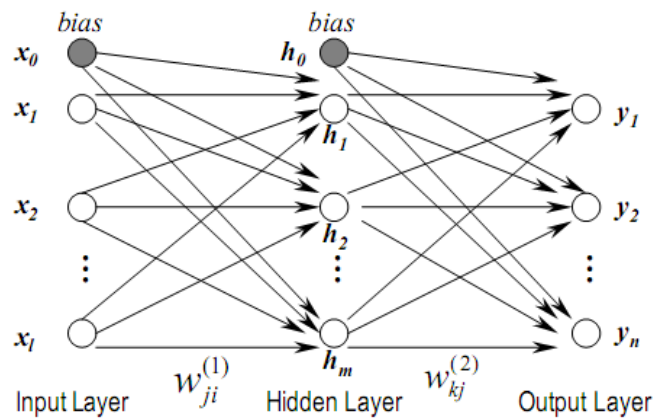
$$g(x) = \frac{1}{1 + e^{-x}}$$

- Hàm sigmoid lưỡng cực

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

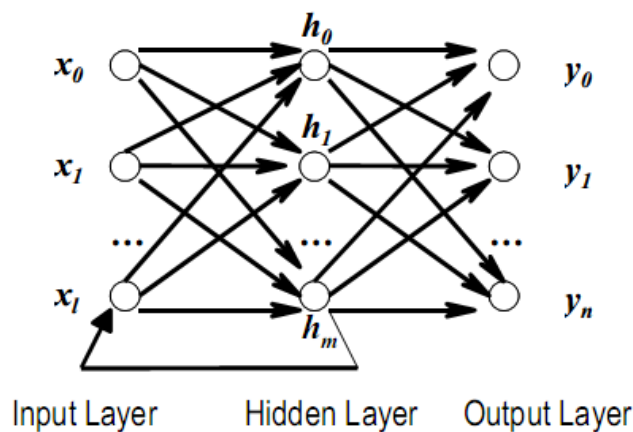
Các đơn vị liên kết với nhau qua các cạnh có trong số tạo thành mạng neuron nhân tạo. Tùy theo số lượng các đơn vị và cách thức liên kết của chúng mà tạo thành các mạng neuron khác nhau có khả năng khác nhau. Có hai loại hình dạng mạng neuron nhân tạo cơ bản là mạng truyền thẳng (Hình 2.2) và mạng hồi quy (Hình 2.3):

- *Mạng truyền thẳng* (Feed-forward neuron 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 2.2: Mạng neuron truyền thẳng

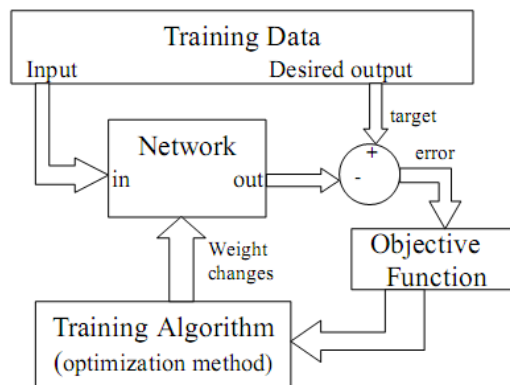
- *Mạng hồi quy* (Recurrent neuron network): Khác với mạng truyền thẳng, mạng hồi quy có chứa các liên kết ngược từ một đơn vị đến các đơn vị ở lớp trước nó.



Hình 2.3: 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) (Hình 2.4) và *Học không có giám sát* (Unsupervised Learning) [8].

- **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 $f(x)$ thoả mãn tất cả các mẫu học đầu vào [8]. 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.

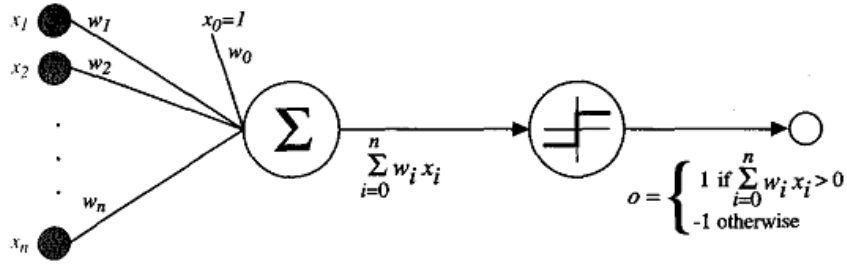


Hình 2.4: Mô hình học có giám sát

- **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 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 [8].

2.2. NGUYÊN TẮC HOẠT ĐỘNG CỦA MẠNG NEURON NHÂN TẠO

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 2.5: Đơn vị mạng Neuron

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

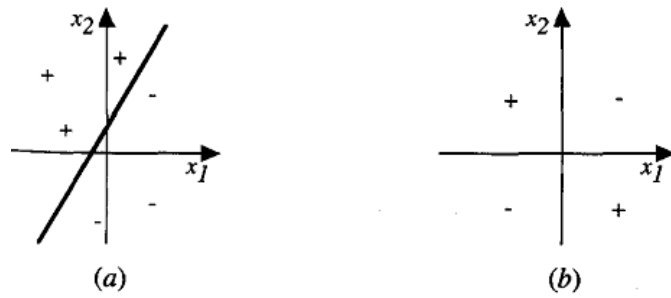
$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

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

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

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} sao 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 2.6: Mặt quyết định biểu diễn bởi perceptron hai đầu nhập

Hình 2.6 (a) là một tập mẫu khả phân tuyến tính có thể được phân ra bởi một mặt quyết định của perceptron. Hình 2.6 (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 sao 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. 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.

- 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} sao cho $E(\vec{w})$ đạt giá trị nhỏ nhất. Hình 2.7 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 $\nabla E(\vec{w})$

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

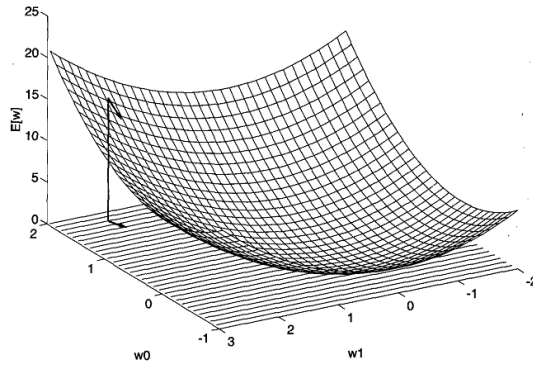
Về mặt toán học vector độ dốc biểu diễn hướng làm tăng giá trị hàm E trong không gian trọng số, do đó $-\nabla E(\vec{w})$ sẽ là hướng làm giảm giá trị hàm E . Trong hình 2.7 nó được biểu diễn bằng dấu mũi tên. Các trọng số sẽ được cập nhập theo quy luật sau:

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$$

$$\Delta \vec{w} = -\eta \nabla E(\vec{w})$$

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 2.7: 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:

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

Thay (2.2) vào (2.1) ta được giá nhập cập nhập 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 nhập các trọng số ngay khi duyệt qua một mẫu của tập dữ liệu. Giá trị cập nhập 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. Theo Tom Mitchell 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 nhập 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 nhập 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 nhập các trọng số được cập nhập 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

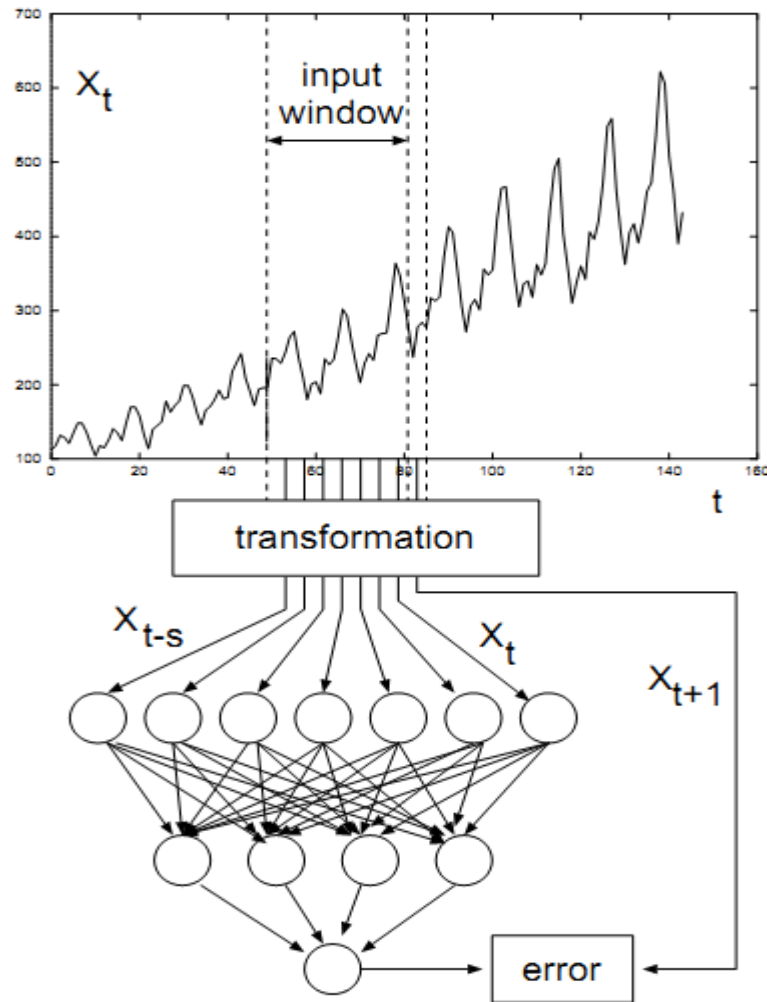
liệu huấn luyện khả phân tuyến tính trong khi giải thuật luật delta sẽ hội tụ về một điểm cực tiểu của hàm lỗi với một thời gian khá lâu (có thể là vô hạn) nhưng sự hội tụ của nó không bị ảnh hưởng bởi tính khả phân tuyến tính của tập dữ liệu huấn luyện.

2.3. ỨNG DỤNG MẠNG NEURON TRONG CÔNG TÁC DỰ BÁO

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.

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). Việc xác định một chuỗi thời gian có thành phần xu hướng hay thành phần mùa hay 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.

Việc sử dụng mạng neuron nhân tạo vào việc dự báo dữ liệu chuỗi thời gian (Hình 2.5) 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 đơn vị cho lớp ẩn có thể xấp xỉ bất kỳ hàm khả đánh giá (measurable function) tuyến tính hay phi tuyến nào[6].



Hình 2.8: 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 vectơ 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 đơn vị ở tầng nhập. Trong giai đoạn truyền tiến, những giá trị đó được truyền qua tầng ẩn rồi đến các đơn vị đầu ra. Khi truyền tới đơn vị đầu ra, giá trị lỗi được tính toán dựa vào sự khác biệt giữa giá trị 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 nhật lại trọng số của các kết nối này.

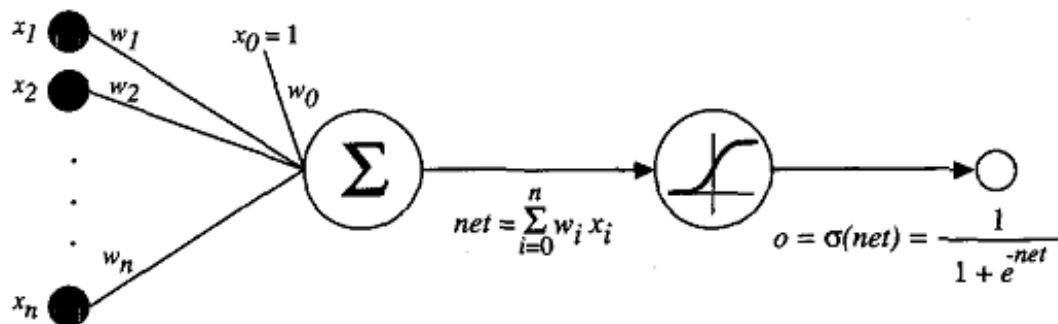
Các cửa sổ đầu vào có thể được chọn một cách ngẫu nhiên hoặc liên tiếp nhau từ dữ liệu chuỗi thời gian. Chọn cửa sổ đầu vào một cách ngẫu nhiên sẽ phức tạp hơn, tuy nhiên sẽ đảm bảo cấu hình mạng tốt hơn và tránh được lỗi tối ưu cục bộ [6].

2.4. HUẤN LUYỆN MẠNG NEURON

Bản chất của huấn luyện mạng neuron là huấn luyện từng perceptron thông qua việc thay đổi vector trọng số của chúng. Quá trình này thường thay đổi tất cả vector trọng số cùng một lúc để tạo ra một mạng mới phù hợp hơn. Trong bài này, tất cả các giải thuật huấn luyện đều là học có giám sát.

2.4.1. GIẢI THUẬT LAN TRUYỀN NGƯỢC

Giải thuật lan truyền ngược áp dụng trong một mạng neuron nhiều lớp. Các mạng Nơ-ron 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 2.9: Đơn vị sigmoid

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

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

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:

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

với
$$\frac{\partial o_i}{\partial w_{ij}} = \frac{\partial o_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}} = f'(net_i) o_j \quad (2.6)$$

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

Ở đây:

- w_{ij} là trọng số của cạnh nối đơn vị j đến đơn vị i
- o_j là kết quả xuất của đơn vị j
- $f()$ 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:

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

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

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

- 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.

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

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

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

Ở đây $\text{succ}(i)$ 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 giải 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 nhật 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 nhật.

Cơ bản có hai phương pháp cập nhật các trọng số phân loại theo thời điểm cập nhật: *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 nhật. 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 [4].

Phương pháp học theo epoch thực hiện lấy tổng tất cả thông tin về *độ dốc* (gradient) cho toàn bộ *tập mẫu* (pattern set) sau đó mới cập nhật các trọng số theo phương pháp giảm độ dốc thông thường, nghĩa là nó thực hiện việc cập nhật 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 [4]. 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 cực tiểu và các vùng phẳng của bề mặt lỗi. Nó cũng giúp tăng giá trị cập nhập 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:

Khởi tạo tất cả các trọng số bằng các số nhỏ ngẫu nhiên

While điều kiện dừng chưa thỏa

For each mỗi mẫu trong tập dữ liệu

Nhập mẫu vào mạng và tính toán giá trị đầu ra.

For each mỗi giá trị xuất của đơn vị k

$$\text{Tính } \frac{\partial E}{\partial w_{kj}}$$

For each đơn vị ẩn h , từ lớp ẩn cuối cùng đến lớp ẩn đầu tiên

$$\text{Tính } \frac{\partial E}{\partial w_{hj}}$$

For each w_{ij} trong mạng

$$\text{Tính } \Delta w_{ij}(t) = -\varepsilon \frac{\partial E}{\partial w_{ij}}(t) + \mu * \Delta w_{ij}(t-1)$$

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

Giải thuật lan truyền ngược cần hai thông số nhập vào đó là hệ số học η và hệ số quán tính μ . 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.

2.4.2. GIẢI THUẬT TÌM KIẾM NGẪU NHIÊN CỤC BỘ

Giải thuật tìm kiếm ngẫu nhiên cục bộ (randomize local search) có thể được áp dụng trong rất nhiều bài toán khác nhau, trong đó có bài toán huấn luyện mạng neuron. Giải thuật đã được vận dụng sáng tạo để phù hợp với bài toán hiện tại [1].

Giải thuật này coi một *lời giải* (solution) của mạng là một mảng trọng số w bao gồm tất cả các trọng số truyền trong mạng. Để tạo ra lời giải lân cận, giải thuật khởi tạo một giá trị gia giảm ngẫu nhiên với mỗi $w[i]$ dựa trên nhiệt độ hiện tại.

- Mã giả quá trình tạo trạng thái lân cận ngẫu nhiên dựa vào nhiệt độ hiện tại:

Khởi tạo mảng trọng số từ mạng neuron hiện tại w

For each mỗi w_i trong mảng

Ngẫu nhiên hệ số gia giảm add $[-0.5;0.5]$

Gia giảm trọng số $w[i] += \text{add} * (T_{\text{current}}/T_0)$;

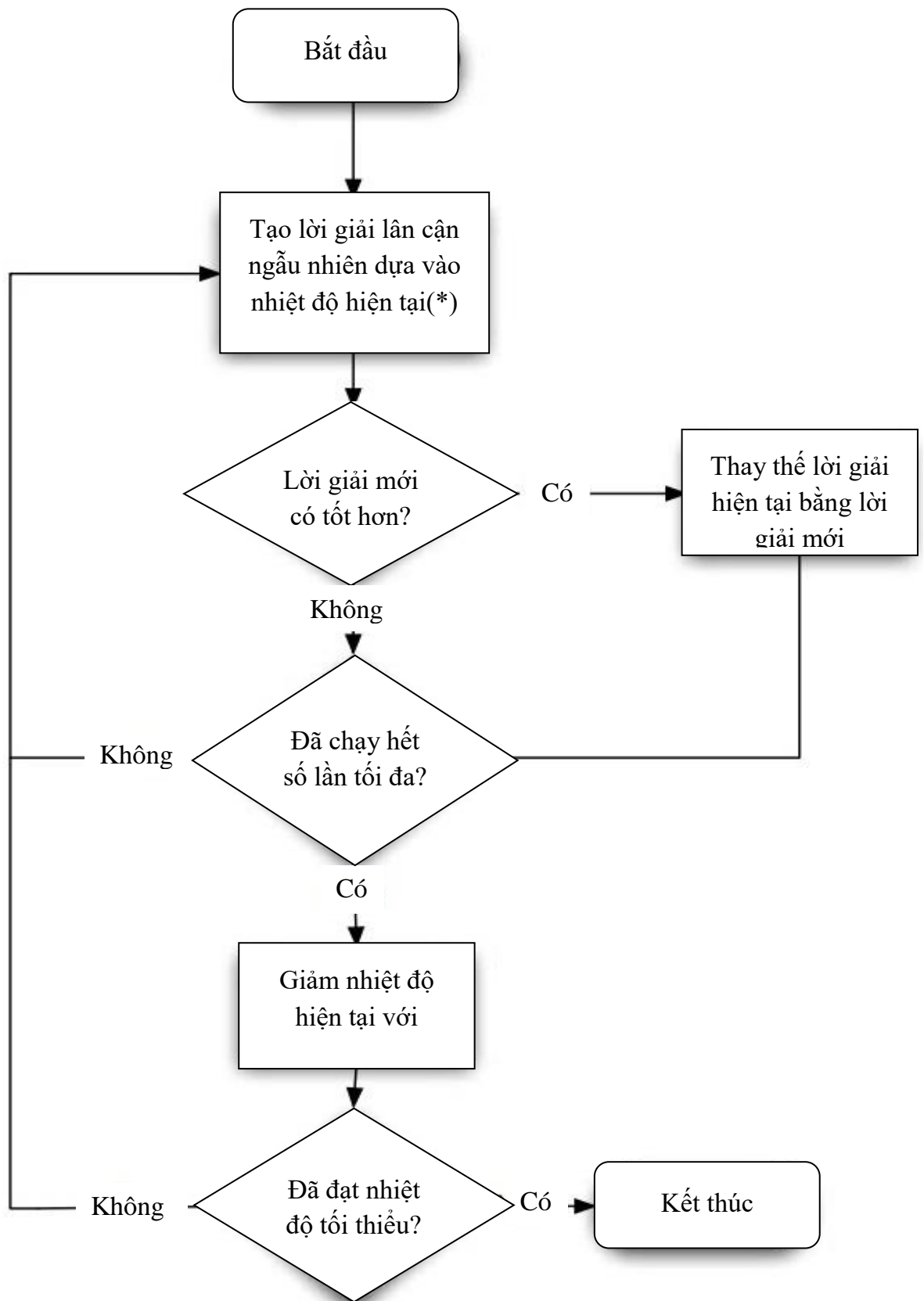
Ở đây n_{ih} và n_{ho} lần lượt là số trọng số giữa tầng nhập với tầng ẩn và số trọng số giữa tầng ẩn với tầng xuất.

Ứng với mỗi nhiệt độ, quá trình tìm kiếm lời giải lân cận sẽ được lặp n lần. Nhiệt độ hiện tại (T_{current}) trong quá trình huấn luyện sẽ được giảm dần. Cụ thể trong bài:

$$T_{\text{current}} *= \text{step}; // \left(\text{step} = e^{\frac{\ln(\frac{s}{e})}{c-1}} \right) [1]$$

Trong giải thuật này, việc chấp nhận một lời giải không tốt hơn lời giải hiện tại có xác suất là 0, đồng nghĩa với việc chỉ chấp nhận lời giải đó khi nó tốt hơn. Và tiếp tục thực hiện với lời giải tốt nhất hiện tại cho đến khi $T_{\text{current}} < T_{\text{last}}$.

- Dưới đây là sơ đồ giải thuật mô phỏng luyện kim huấn luyện mạng neuron (Hình.2.10):



Hình 2.10 Sơ đồ giải thuật mô phỏng luyện kim huấn luyện mạng neuron

2.4.3. GIẢI THUẬT MÔ PHỎNG LUYỆN KIM

Từng bước của thuật toán mô phỏng luyện kim (gọi tắt là SA) cố gắng thay thế các giải pháp hiện tại bằng một giải pháp ngẫu nhiên (lựa chọn theo một danh sách phân phối của các ứng cử viên, thường được xây dựng từ các giải pháp gần với giải pháp hiện tại đang xét), các giải pháp mới sau đó có khả năng được chấp nhận dựa trên một xác suất, quyết định bởi sự khác biệt của giá trị một hàm tương ứng và một tham số toàn cục T (gọi là nhiệt độ - temperature) giảm dần trong suốt thuật toán thực hiện. Sự phụ thuộc này cũng như việc lựa chọn giữa giải pháp trước đó và giải pháp hiện tại hầu như là ngẫu nhiên khi T lớn, tuy nhiên sẽ cho kết quả chọn lựa tốt hơn khi T giảm dần về 0, cũng như trong việc luyện kim, khi nhiệt độ cao, các nguyên tử chuyển động hỗn loạn theo một tốc độ và tần suất lớn, tuy nhiên nhiệt độ càng giảm thì các nguyên tử càng ít chuyển động dần đi.

- **Tổng quát:**

Trong thuật toán SA, mỗi điểm s của không gian tìm kiếm là tương tự với một trạng thái của một số hệ thống vật lý, và hàm $E(s)$ tượng trưng cho nội năng của hệ thống trong trạng thái đó.

Tại mỗi bước, quy tắc heuristic sẽ xem xét và đưa ra một số trạng thái kề s' của trạng thái hiện tại s , và xác suất quyết định giữa thay đổi hệ thống sang trạng thái s' hay là vẫn tiếp tục giữ trạng thái s . Những xác suất này đưa hệ thống tới trạng thái có năng lượng thấp hơn. Bước này sẽ được lặp đi lặp lại cho đến khi hệ thống đạt được một trạng thái đủ tốt, hoặc cho đến khi không thể tiếp tục.

- **Trạng thái kề**

Trạng thái kề của một trạng thái là những trạng thái mới của vấn đề được sinh ra sau khi biến đổi một trạng thái đã biết theo một số cách cụ thể. Ví dụ, ở trong *bài toán người bán hàng* (travelling salesman problem) gọi tắt là TSP, mỗi trạng thái được định nghĩa là một cách đi để có thể đi qua hết tất cả các thành phố, hay nói cách khác mỗi trạng thái là một hoán vị của các thành phố thể hiện cho mỗi cách đi. Ví dụ:

Ta xét bài toán TSP với 6 thành phố $\{1, 2, 3, 4, 5, 6\}$, thì một trạng thái được hiểu là một hoán vị của 6 thành phố trên, mà mỗi hoán vị thể hiện cho một cách đi. Chẳng hạn hoán vị $\{1, 2, 3, 4, 5, 6\}$ thể hiện cho cách đi $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 1$, tương tự hoán vị $\{1, 3, 2, 6, 4, 5\}$ thể hiện một cách đi từ $1 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 1$.

Những trạng thái kề của một hoán vị là những hoán vị được sinh ra chẳng hạn như từ việc trao đổi vị trí của một cặp thành phố kề nhau trong hoán vị đó. Hành động thay đổi giải pháp để tìm ra giải pháp lân cận được gọi là *bước di chuyển* (move) và các move khác nhau sẽ cho các giải pháp lân cận khác nhau.

Việc tìm kiếm những lân cận của một trạng thái là cơ bản cho việc tối ưu vì lời giải cuối cùng có được sau một tour lần lượt các lân cận. Các kỹ thuật dựa trên kinh nghiệm (Heuristic) đơn giản di chuyển bằng việc tìm lân cận tốt nhất và dừng lại khi tìm được một lời giải mà lời giải đó không có lân cận nào tốt hơn nó. Vấn đề ở đây là các lân cận của một trạng thái ta không biết chắc là chúng có chứa lời giải nào tốt hơn hay không, nghĩa là khi ta xét một lời giải, mặc dù lời giải đó không là tốt nhất nhưng ta không chắc chắn là trong số các lân cận của nó không chứa một giải pháp tốt hơn. Đó là lý do vì sao lời giải tốt nhất tìm được bằng thuật toán được gọi là tối ưu cục bộ, trái ngược với lời giải tốt nhất trên thực tế (tối ưu toàn cục (global optimization)). Các kỹ thuật kinh nghiệm tổng quát (metaheuristic) sử dụng các lân cận của một trạng thái như một cách để khám phá không gian lời giải và có thể chấp nhận những lời giải không tốt trong việc tìm kiếm để đạt được của chúng để thực hiện điều đó. Có nghĩa việc tìm kiếm sẽ không bị mắc kẹt và nếu thuật toán chạy một khoảng thời gian đủ lớn, kết quả tối ưu toàn cục sẽ được tìm thấy.

- **Xác suất chấp nhận:**

Xác suất của việc chuyển từ trạng thái hiện tại S sang một trạng thái kế S' của nó được định nghĩa bởi một hàm xác suất chấp nhận:

- $P(e, e', T)$

Hàm phụ thuộc vào các nguồn năng lượng $e=E(s)$ và $e'=E(s')$ của 2 trạng thái, và một tham số toàn cục T gọi là nhiệt độ. Những trạng thái có năng lượng thấp hơn sẽ tốt hơn những trạng thái có năng lượng cao hơn. Hàm xác suất P phải cho kết quả dương khi $e' > e$.

- $e' > e \Leftrightarrow P(e, e', T) > 0$

Khi xây dựng hàm P , ta cần chú ý rằng khi T dần tiến tới 0, P cũng phải dần tới 0 nếu $e' > e$ và luôn mang một giá trị dương. Có nghĩa khi T đạt giá trị càng nhỏ, hệ thống sẽ càng ưu tiên việc di chuyển xuống nơi có năng lượng thấp, và tránh việc di chuyển ngược lại, tương tự như việc giảm nhiệt độ trong luyện kim, các nguyên tử bớt chuyển động hỗn độn hơn. Khi $T=0$ thuật toán đơn thuần trở về thuật toán tham lam, với việc đơn thuần chỉ có sự di chuyển từ nơi có năng lượng cao đến nơi có năng lượng thấp. Ngoài ra hàm P thường được xây dựng sao cho tỉ lệ thuận với độ chênh lệch năng lượng giữa trạng thái tiếp theo và trạng thái hiện tại, nói cách khác khi $e' - e$ càng giảm thì P càng giảm, vì vậy xác suất chuyển từ trạng thái hiện tại tới trạng thái có độ chênh lệch năng lượng thấp thì khó thấp hơn so với độ trạng thái có độ chênh lệch năng lượng cao.

Tham số T có vai trò quyết định với việc điều khiển quá trình tiến triển của trạng thái s . Vì khi T càng nhỏ thì P càng nhỏ, cho nên tham số T quyết định độ “hỗn

độ” của trạng thái s , nghĩa là khi T càng nhỏ thì khả năng trạng thái s được di chuyển càng thấp, tương tự như tham số nhiệt độ trong kỹ thuật luyện kim.

Đối với bất một vấn đề cụ thể nào, thuật toán cho xác suất để tìm thấy lời giải tối ưu của vấn đề dần tiến tới 1 khi mà thời gian thực hiện thuật toán đủ lớn. Tuy nhiên ta không nên quan tâm tới việc này vì thời gian cần thiết để có thể tìm thấy lời giải tối ưu thường sẽ vượt quá thời gian thực hiện của phương pháp vét cạn trong không gian lời giải.

- **Quá trình tổng quát của giải thuật:**

Thuật toán bắt đầu với việc khởi tạo tham số T một giá trị lớn (càng lớn càng cho lời giải chính xác, hoặc có thể là vô tận), say đó nó được giảm dần sau mỗi vòng lặp, độ giảm của T có thể do ta tự định nghĩa, tuy nhiên thuật toán phải kết thúc khi $T=0$. Trong suốt quá trình, lời giải sẽ lặp đi lặp lại việc di chuyển ngẫu nhiên với một xác suất P nhất định và ngày càng mở rộng không gian tìm kiếm chứa các lời giải tốt, và cứ như vậy lời giải trôi về vùng không gian lời giải có năng lượng thấp, cuối cùng dừng lại khi $T=0$.

- **Mã giả:**

Khởi tạo lời giải tốt nhất s_{best} , e_{best} ;

Foreach số lần giảm nhiệt độ n **as** k

 Tính nhiệt độ $T = \text{temperature}((n - k) / n)$;

 Tìm trạng thái kề của s_{new}

 Tính năng lượng của trạng thái mới e_{new}

if ($P(e, e_{new}, T) > \text{random}()$) /* kiểm tra xem có nên di chuyển sang trạng thái mới */

 Cập nhật trạng thái mới $s = s_{new}$; $e = e_{new}$;

 }

if ($e < e_{best}$) /* nếu trạng thái mới tốt hơn trạng thái tốt nhất */

$s_{best} = s_{new}$; $e_{best} = e_{new}$; /* cập nhật trạng thái tốt nhất */

2.4.4. GIẢI THUẬT MÔ PHỎNG LUYỆN KIM HUẤN LUYỆN MẠNG NEURON

Để áp dụng giải thuật SA, trước hết ta cần biết về bài toán tìm kiếm cục bộ với các thành phần chính sau:

- Một không gian trạng thái
- Cấu trúc trạng thái cận
- Một hàm đánh giá

Điều này về cơ bản xác định một giải pháp bằng cách di chuyển từ giải pháp hiện tại đến một giải pháp lân cận. Mã giả của thuật toán SA như sau (Thompson & Dowsland):

```
Chọn một trạng thái ban đầu  $s_0$ 
Chọn một nhiệt độ ban đầu  $t_0 > 0$ 
Chọn một hàm giảm nhiệt độ  $\beta$ ;
While điều kiện dừng chưa thỏa
    For each nrep as iteration_count
        Randomly select  $s \in N(s_0)$ ; /*  $s$  is a neighbor solution of  $s_0$  */
         $\delta = f(s) - f(s_0)$  /* compute the change in cost function */
        if ( $\delta < 0$ )
             $s_0 = s$ 
        else
            generate random  $x \in [0,1]$  /*  $x$  is a random number in range 0 to 1 */
            if  $x < \exp(-\delta/t)$ 
                 $s_0 = s$ 
         $t = \beta(t)$ 
    /*  $s_0$  is the approximation to the optimal solution */
```

Di chuyển tiềm năng được lấy mẫu ngẫu nhiên và tất cả các động thái cải thiện được chấp nhận tự động. Các di chuyển khác được chấp nhận với xác suất $\exp(-\delta/t)$, với δ là sự thay đổi của hàm đánh giá và t là tham số điều khiển.

Chất lượng của giải pháp nhạy cảm với cách thức mà các thông số nhiệt độ được điều chỉnh - lịch trình làm mát. Điều này được xác định bởi

- Nhiệt độ ban đầu t_0 ,
- Điều kiện dừng,
- Hàm giảm nhiệt độ β
- Số lượng vòng lặp ở mỗi nhiệt độ, $nrep$.

Các giá trị trên phải được lựa chọn phù hợp với từng bài toán.

+ Cách tạo trạng thái lân cận

Trong không gian trạng thái S là tập các vector $w=(w_1, w_2 \dots w_{(n+1)m})$, trạng thái v được xem là trạng thái lân cận của w nếu mọi trọng số v_i thuộc v có thể sinh ra từ trọng số w_i thuộc w bằng cách thêm hoặc bớt một giá số αw_i . Do đó, trạng thái cận N_α của w sẽ được sinh như sau:

$$N_\alpha(w) = \{((1+z_1)w_1, \dots, (1+z_{(n+1)m})w_{(n+1)m}) \mid z_1, \dots, z_{(n+1)m} \in \{-\alpha \mid \alpha\}\}$$

Thí dụ, ta có thể chọn $\alpha = 0.01$

+ Cách đánh giá một trạng thái của mạng nơ-ron

Và hàm dùng để đánh giá trạng thái của mạng nơ-ron là

$$f = \sqrt{\frac{\sum_{k=1}^n (t_k - o_k)^2}{n}}$$

Với t_k là giá trị dự báo và o_k là giá trị quan sát được
 n là chiều dài chuỗi huấn luyện.

Chương 3

HIỆN THỰC VÀ KẾT QUẢ THỰC NGHIỆM

Với yêu cầu của đề tài, việc thực hiện một chương trình demo đã được hoàn thiện. Chương trình có dựa trên một số hướng dẫn trong một số tài liệu tham khảo [7] ở giai đoạn trước để củng cố kinh nghiệm cho lần chạy thử này. Mục đích chính của chương trình demo là để kiểm tra và đánh giá kết quả huấn luyện trong thực nghiệm với các bộ dữ liệu dòng chảy.

3.1. CÁCH ĐÁNH GIÁ KẾT QUẢ HUẤN LUYỆN

Chương trình hiện thực cho phép người dùng tiến hành đánh giá mạng neuron vừa huấn luyện trên một tập dữ liệu nằm ngoài tập dữ liệu huấn luyện. Giá trị các hàm lỗi dự báo sau đây sẽ được tính và xuất ra cho người dùng:

- Hàm *trung bình tuyệt đối lỗi* (mean absolute error):

$$MAE = \frac{\sum_{k=1}^n |t_k - o_k|}{n}$$

- Hàm *phần trăm trung bình tuyệt đối lỗi* (mean absolute percentage error):

$$MAPE = \frac{\sum_{k=1}^n \left(\frac{|t_k - o_k|}{|t_k|} \right)}{n} \times 100$$

- Hàm *tổng bình phương lỗi* (sum of squared error):

$$SSE = \sum_{k=1}^n (t_k - o_k)^2$$

- Hàm *trung bình bình phương lỗi* (mean squared error)

$$MSE = \frac{\sum_{k=1}^n (t_k - o_k)^2}{n}$$

Ở đây n là số điểm trong tập kiểm tra, t_k và o_k lần lượt là giá trị thực trong tập dữ liệu và giá trị xuất của mạng neuron.

Dựa trên các giá trị này người dùng sẽ quyết định mạng có phù hợp không, có cần được huấn luyện hay xây dựng lại không.

Các kết quả này được chạy thực nghiệm với chương trình demo được bằng C# trên hệ điều hành Window 10, cài đặt sẵn .NET framework với cấu hình máy: Ram 4GB, CPU Intel Core i5 1.7 GHz.

3.2. KẾT QUẢ THỰC NGHIỆM

Trong giai đoạn này tôi thực hiện chạy các giải thuật trên các bộ dữ liệu *lưu lượng dòng chảy* (runoff) trên sông, một bộ dữ liệu qmax và một bộ dữ liệu thiên văn. Quá trình chạy bao gồm một lần giải thuật lan truyền ngược, một lần giải thuật lai và một lần giải thuật mô phỏng luyện kim để so sánh và đánh giá. Các dữ liệu được thực nghiệm là:

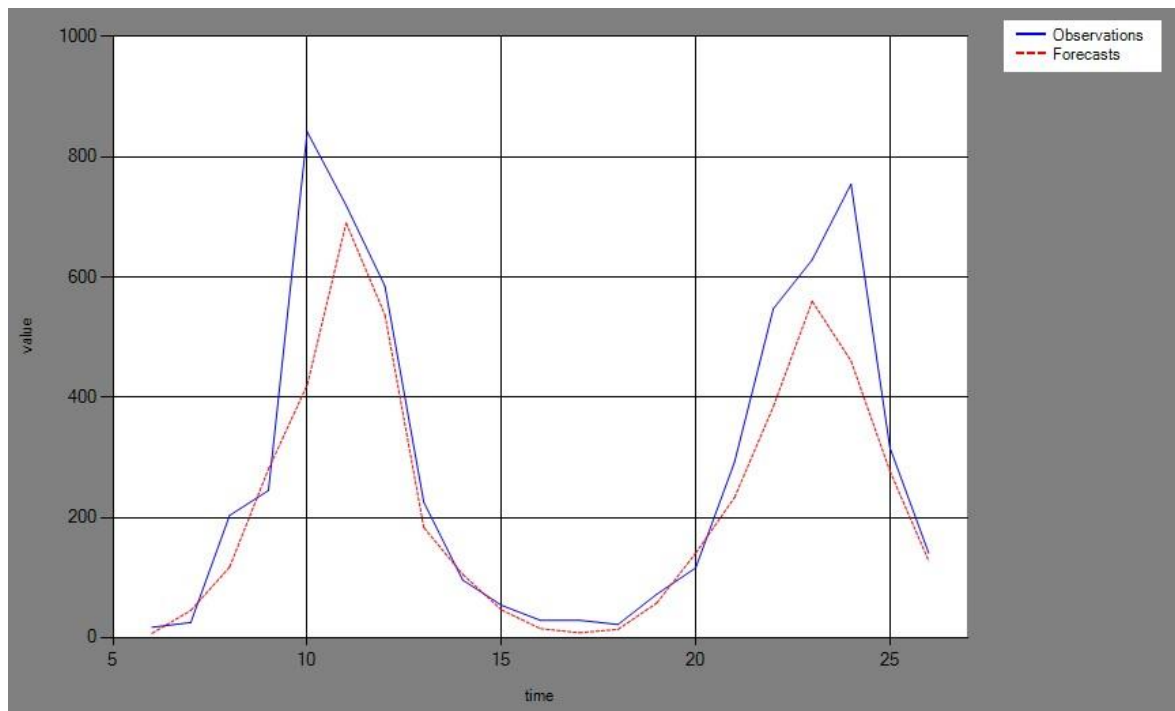
- (1) Runoff Phước Hòa, 204 điểm
- (2) Runoff Ghềnh Ga, 239 điểm
- (3) Runoff Chiêm Hóa, 239 điểm
- (4) Runoff Phước Long, 204 điểm
- (5) Qmax Châu Đốc, 365 điểm
- (6) Runoff Buôn Hồ, 4000 điểm
- (7) Runoff Cầu 14, 4000 điểm
- (8) Runoff Đức Xuyên, 3000 điểm
- (9) Dữ liệu thiên văn Sunspot, 2899 điểm

Các bộ dữ liệu được chia thành hai nhóm dựa vào số điểm trong bộ. Trong đó, năm bộ (1) tới (5) được coi là bộ dữ liệu nhỏ, còn bốn bộ còn lại là dữ liệu lớn.

3.2.1. THỰC HIỆN TRÊN CÁC BỘ DỮ LIỆU NHỎ

Đối với các bộ dữ liệu nhỏ, tôi chọn sử dụng mạng neuron 6-6-1. Mỗi giải thuật sẽ được chạy 5 lần để giảm bớt sai số.

- Kết quả trên bộ dữ liệu Runoff Phước Hòa:



LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	727	2	0,07389174	29,93%	0,347554481	0,018112118
0,2	0,2	410	1	0,069463573	35,37%	0,296298921	0,017347568
0,01	0,3	847	2	0,07808959	38,88%	0,334094961	0,015909284
0,3	0,25	107	1	0,071741771	39,20%	0,235486075	0,011213623
0,05	0,15	967	2	0,059809321	29,65%	0,247133507	0,011768262
0,132	0,2	611,6	1,6	0,070599199	34,61%	0,292113589	0,014870171

Bảng 3.1

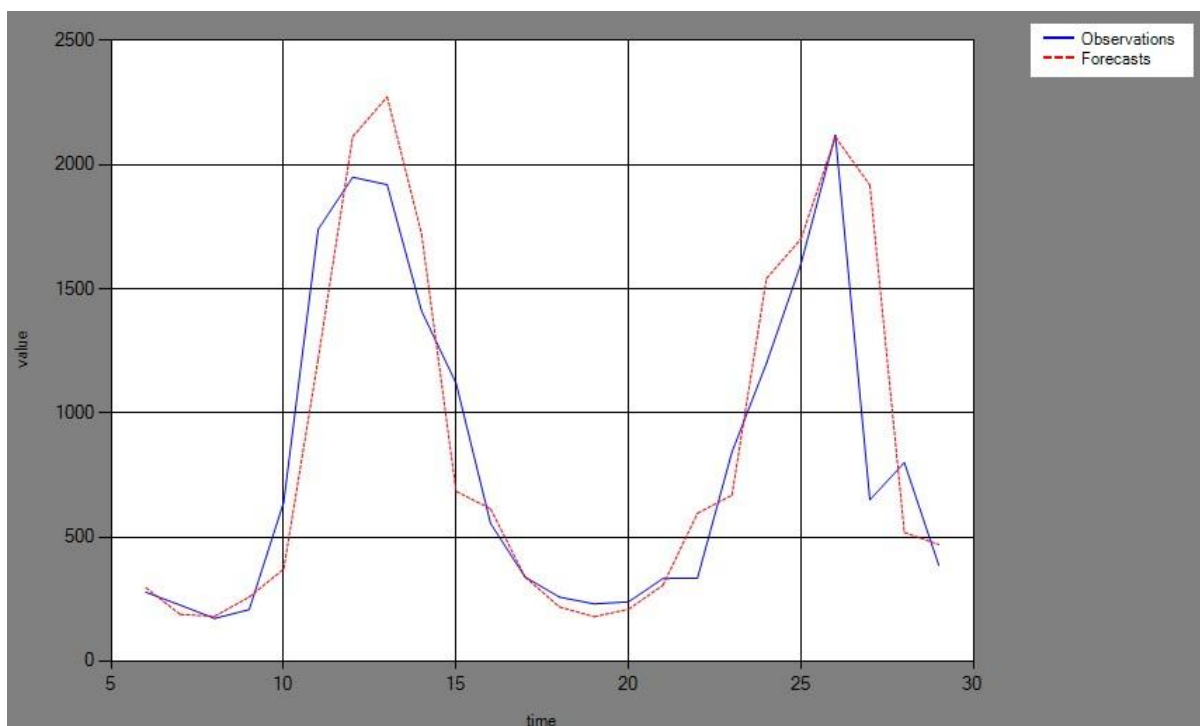
Time	MAE	MAPE	SSE	MSE
1	0,06950055	25,79%	0,309563353	0,014741112
1	0,067241916	23,09%	0,297351488	0,014159595
1	0,065852722	22,62%	0,318396686	0,015161747
1	0,067481545	20,47%	0,317028502	0,015096595
1	0,067552078	23,41%	0,2990079	0,014238471
1	0,067525762	23,08%	0,308269586	0,014679504

Bảng 3.2

Time	MAE	MAPE	SSE	MSE
15	0,06193869	20,49%	0,224979358	0,010713303
19	0,068008203	23,03%	0,268373087	0,012779671
21	0,069514716	25,66%	0,277844603	0,013230695
42	0,065675183	20,38%	0,254096617	0,012099839
25	0,073546062	25,36%	0,285673022	0,013603477
24,4	0,067736571	22,98%	0,262193337	0,012485397

Bảng 3.3

- Kết quả trên bộ dữ liệu Runoff Ghềnh Ga:

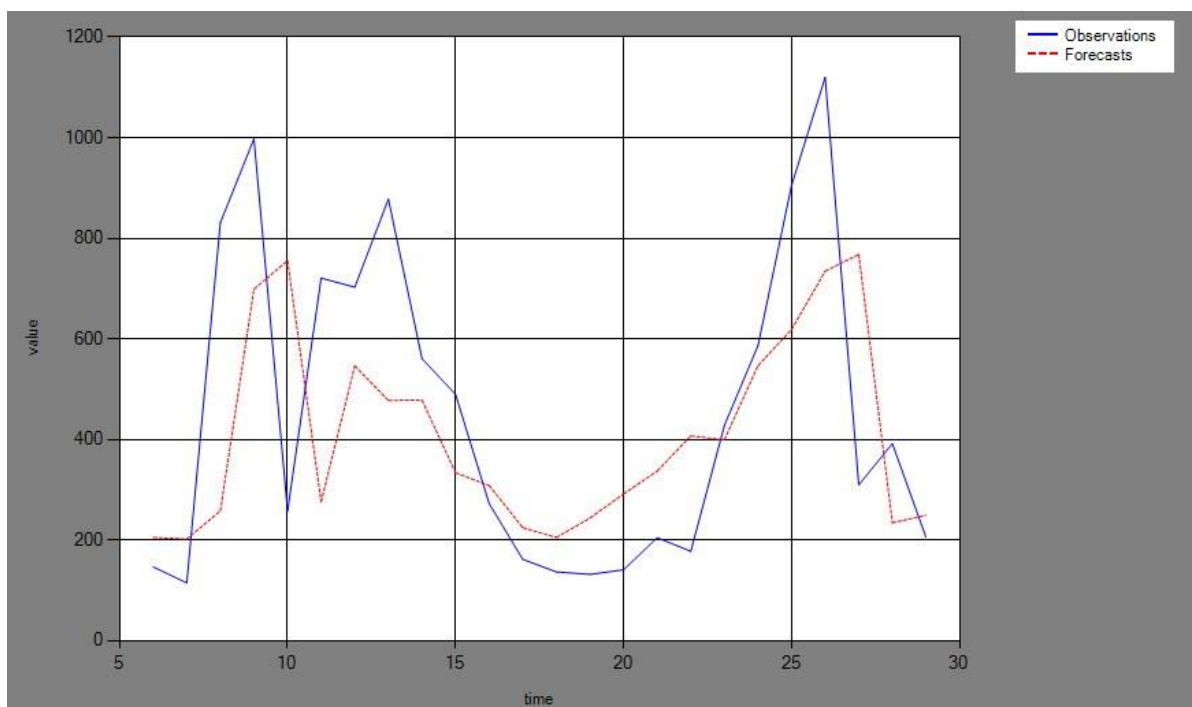


LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	190	1	0,053999507	35,14%	0,144244078	0,00601017
0,2	0,2	630	2	0,043419792	29,94%	0,105317717	0,004388238
0,01	0,3	101	1	0,126165232	99,66%	0,54270884	0,022612868
0,3	0,25	591	2	0,043768089	29,56%	0,10087792	0,004203247
0,05	0,15	383	1	0,052275241	32,53%	0,144993083	0,006041378
0,132	0,2	379	1,4	0,063925572	45,37%	0,207628328	0,00865118

Time	MAE	MAPE	SSE	MSE
1	0,059006859	29,40%	0,178700504	0,007445854
1	0,046133828	25,42%	0,129442371	0,005393432
1	0,056764851	27,05%	0,17588841	0,007328684
1	0,046623387	25,88%	0,120506208	0,005021092
1	0,057973287	28,51%	0,17746352	0,007394313
1	0,053300442	27,25%	0,156400203	0,006516675

Time	MAE	MAPE	SSE	MSE
18	0,047805516	33,26%	0,114827302	0,004784471
23	0,052124061	37,84%	0,128470032	0,005352918
14	0,050583092	35,86%	0,150760923	0,006281705
11	0,048798242	32,81%	0,139501976	0,005812582
16	0,046241519	33,26%	0,110771114	0,004615463
16,4	0,049110486	34,61%	0,128866269	0,005369428

- Kết quả trên bộ dữ liệu Runoff Chiêm Hóa:

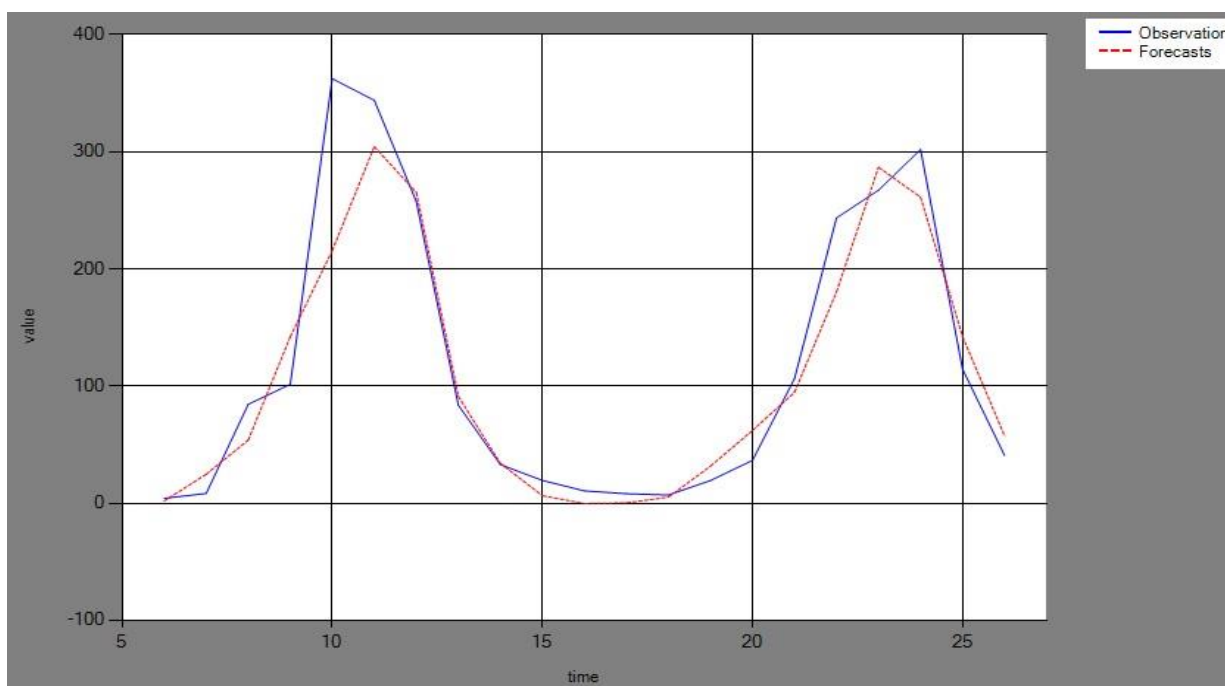


LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	341	1	0,054283815	40,67%	0,144685761	0,006028573
0,2	0,2	624	2	0,052975765	36,24%	0,131718174	0,005488257
0,01	0,3	101	1	0,119787238	120,64%	0,493034322	0,020543097
0,3	0,25	573	2	0,048532983	31,19%	0,11509927	0,004795803
0,05	0,15	494	1	0,055549281	41,92%	0,154575085	0,006440629
0,132	0,2	426,6	1,4	0,066225816	54,13%	0,207822522	0,008659272

Time	MAE	MAPE	SSE	MSE
1	0,053539669	30,22%	0,14823762	0,006176567
1	0,057684401	34,00%	0,149799581	0,006241649
1	0,055622874	29,71%	0,161176778	0,006715699
1	0,055585813	30,64%	0,145331447	0,006055477
1	0,056349542	33,58%	0,152288903	0,006345371
1	0,05575646	31,63%	0,151366866	0,006306953

Time	MAE	MAPE	SSE	MSE
10	0,058295368	35,52%	0,188384593	0,007849358
13	0,057632968	34,22%	0,148268925	0,006177872
9	0,057701392	33,76%	0,164748142	0,006864506
9	0,056860335	33,04%	0,154567214	0,006440301
14	0,060720548	36,04%	0,195081217	0,008128384
11	0,058242122	34,52%	0,170210018	0,007092084

- Kết quả trên bộ dữ liệu Runoff Phước Long:



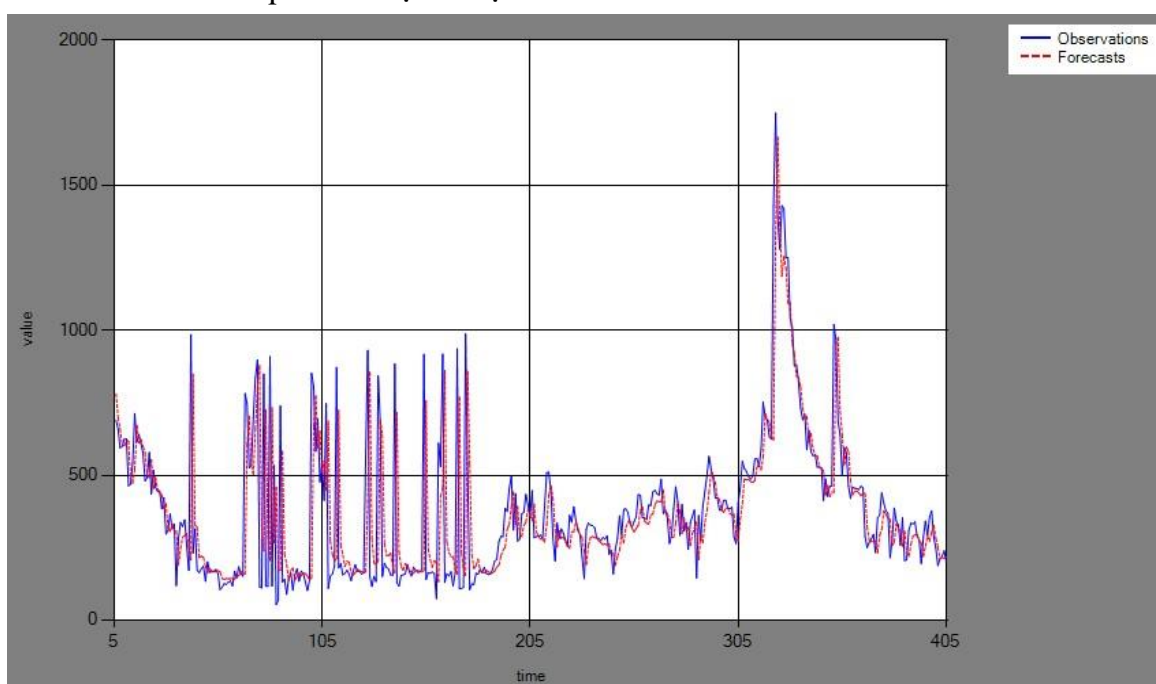
LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	852	2	0,059680377	49,29%	0,135969631	0,006474744
0,2	0,2	420	1	0,063144214	53,18%	0,153465718	0,007307891
0,01	0,3	771	2	0,082168204	70,60%	0,263516533	0,012548406
0,3	0,25	278	1	0,069448568	58,53%	0,182001366	0,008666732
0,05	0,15	988	2	0,061236255	49,17%	0,153625069	0,007315479
0,132	0,2	661,8	1,6	0,067135524	56,15%	0,177715663	0,00846265

Time	MAE	MAPE	SSE	MSE
1	0,049700723	25,80%	0,128327091	0,006110814
1	0,049739688	29,36%	0,13554563	0,006454554
1	0,053530913	33,25%	0,161162833	0,007674421
1	0,049511693	34,04%	0,120717052	0,005748431
1	0,050822707	25,57%	0,143866997	0,006850809
1	0,050661145	29,60%	0,137923921	0,006567806

Time	MAE	MAPE	SSE	MSE
9	0,045443274	26,59%	0,091373124	0,004351101
25	0,048323439	26,31%	0,090940016	0,004330477
10	0,049745919	28,38%	0,118261779	0,005631513
5	0,045508655	27,63%	0,110997254	0,005285584
9	0,047343036	28,39%	0,117951749	0,00561675
11,6	0,047272865	27,46%	0,105904784	0,005043085

3.2.2. THỰC HIỆN TRÊN CÁC BỘ DỮ LIỆU LỚN

- Kết quả trên bộ dữ liệu Runoff Buôn Hồ:

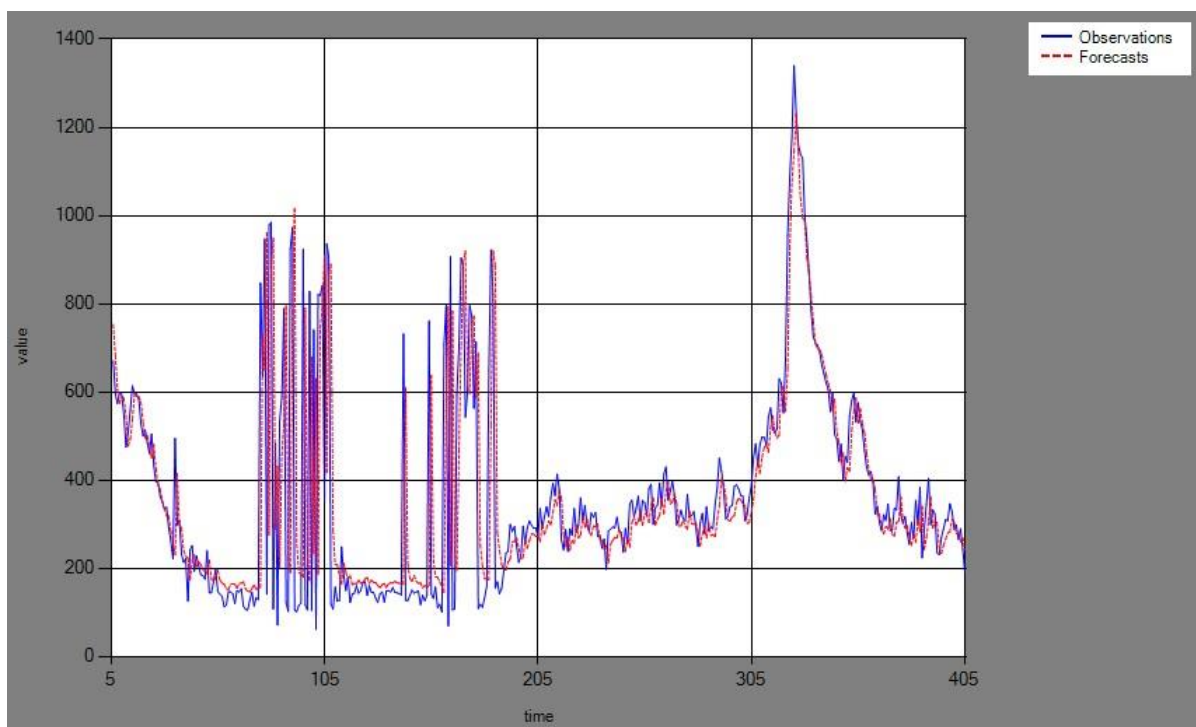


LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	110	9	0,043228788	40,83%	2,046955423	0,005117389
0,2	0,2	101	9	0,054572083	54,10%	2,432962281	0,006082406
0,01	0,3	370	28	0,042339616	37,98%	1,94590072	0,004864752
0,3	0,25	511	40	0,056505333	56,43%	2,638305985	0,006595765
0,05	0,15	161	13	0,041696647	37,61%	2,003758915	0,005009397
0,132	0,2	250,6	19,8	0,047668493	45,39%	2,213576665	0,005533942

Time	MAE	MAPE	SSE	MSE
26	0,040084831	33,61%	2,169534403	0,005423836
24	0,040728557	34,75%	2,279895159	0,005699738
23	0,040568582	33,91%	2,098451434	0,005246129
25	0,041940338	36,74%	2,240178507	0,005600446
23	0,041073624	34,39%	2,215118203	0,005537796
24,2	0,040879186	34,68%	2,200635541	0,005501589

Time	MAE	MAPE	SSE	MSE
55	0,042076208	38,11%	2,144375494	0,005360939
23	0,044565263	40,95%	2,427800216	0,006069501
40	0,041932693	37,97%	2,106495285	0,005266238
14	0,04196773	38,13%	2,124353276	0,005310883
53	0,042051196	38,42%	2,1375729	0,005343932
37	0,042518618	38,72%	2,188119434	0,005470299

- Kết quả trên bộ dữ liệu Runoff Cầu 14:

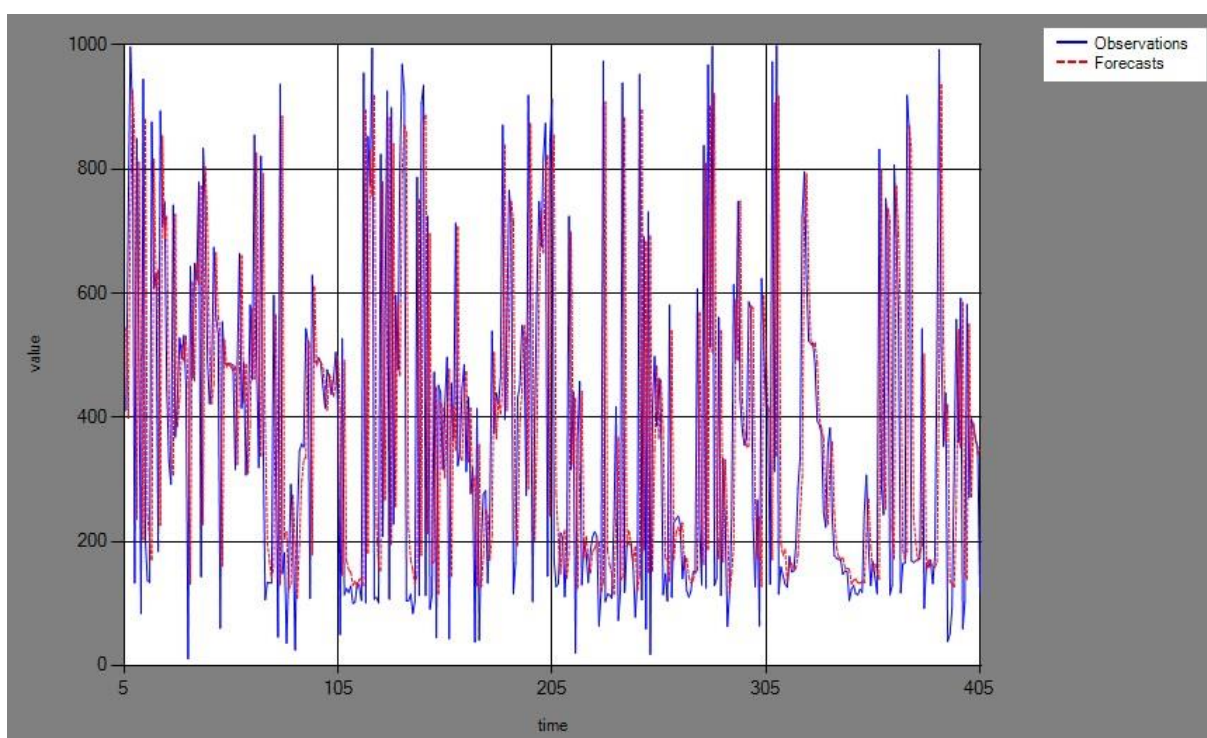


LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	101	8	0,045633421	43,25%	2,495447518	0,006238619
0,2	0,2	562	41	0,049940362	49,59%	2,708476614	0,006771192
0,01	0,3	322	22	0,044311555	39,84%	2,38882558	0,005972064
0,3	0,25	365	26	0,052964217	52,29%	2,759643456	0,006899109
0,05	0,15	166	12	0,044056574	39,99%	2,438421402	0,006096054
0,132	0,2	303,2	21,8	0,047381226	44,99%	2,558162914	0,006395408

Time	MAE	MAPE	SSE	MSE
25	0,041342642	37,55%	2,789012427	0,006972531
23	0,04148048	39,19%	2,851256214	0,007128141
22	0,041658453	36,43%	2,538065366	0,006345163
22	0,040548315	36,53%	2,752839926	0,0068821
23	0,042076575	37,35%	2,736011871	0,00684003
23	0,041421293	37,41%	2,733437161	0,006833593

Time	MAE	MAPE	SSE	MSE
32	0,042981096	38,95%	2,608269355	0,006520673
5	0,043537269	39,72%	2,62469751	0,006561744
48	0,042631053	38,59%	2,506131528	0,006265329
29	0,042311228	38,95%	2,460975507	0,006152439
53	0,042829099	38,88%	2,53443561	0,006336089
33,4	0,042857949	39,02%	2,546901902	0,006367255

- Kết quả trên bộ dữ liệu Runoff Đức Xuyên:

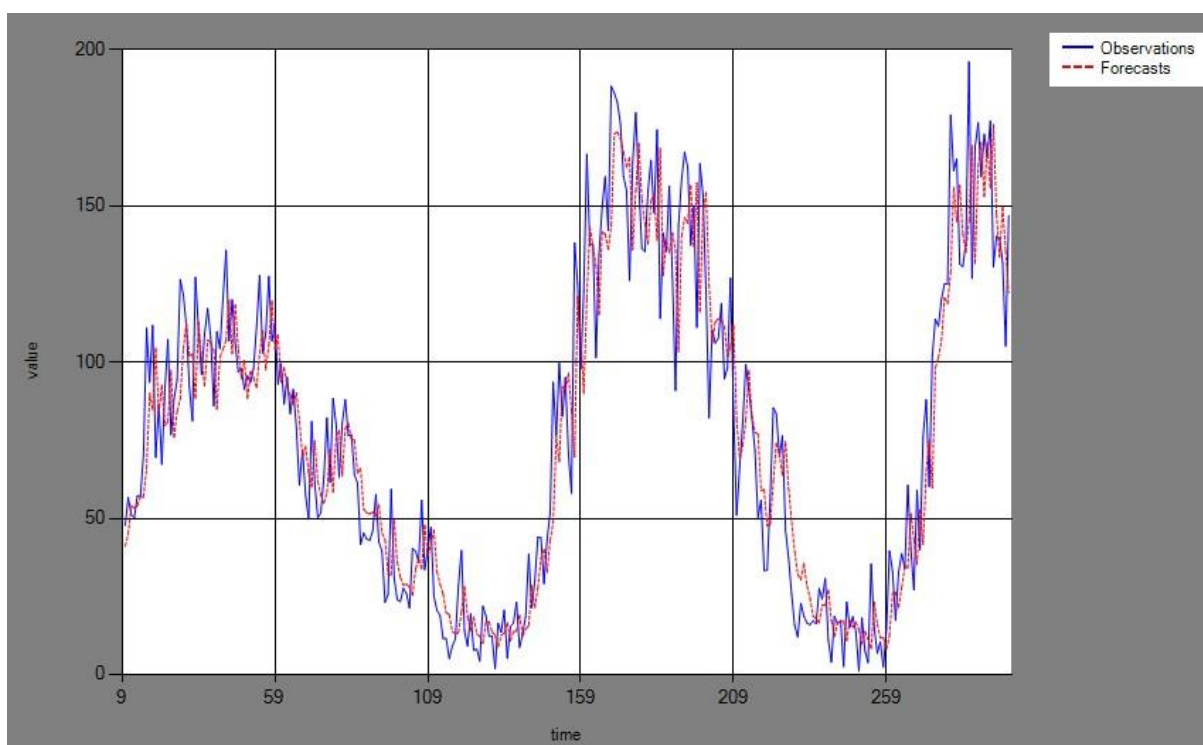


LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	249	13	0,027246091	35,30%	0,494266003	0,001647553
0,2	0,2	186	10	0,028344031	34,03%	0,557347261	0,001857824
0,01	0,3	589	32	0,032211726	41,96%	0,735087097	0,00245029
0,3	0,25	184	10	0,02974042	34,55%	0,631415899	0,00210472
0,05	0,15	317	17	0,028054896	36,59%	0,512652975	0,001708843
0,132	0,2	305	16,4	0,029119433	36,49%	0,586153847	0,001953846

Time	MAE	MAPE	SSE	MSE
17	0,027229736	35,76%	0,492169171	0,001640564
17	0,027271238	34,63%	0,503215351	0,001677385
17	0,029268094	36,94%	0,562346506	0,001874488
17	0,025933393	34,04%	0,449760785	0,001499203
17	0,027803392	34,97%	0,518621626	0,001728739
17	0,027501171	35,27%	0,505222688	0,001684076

Time	MAE	MAPE	SSE	MSE
38	0,025883721	35,79%	0,473023287	0,001576744
39	0,025656421	34,73%	0,464891792	0,001549639
2	0,025830908	35,35%	0,470911546	0,001569705
25	0,025899052	34,81%	0,474097523	0,001580325
10	0,025549589	34,79%	0,460106727	0,001533689
22,8	0,025763938	35,09%	0,468606175	0,00156202

- Kết quả trên bộ dữ liệu Runoff Dữ liệu thiên văn Sunspot:



LearnRate	Momen	Epochs	Time	MAE	MAPE	SSE	MSE
0,1	0,1	143	5	0,060154316	27,74%	1,765888422	0,00608927
0,2	0,2	110	4	0,082750045	30,42%	3,779549234	0,013032928
0,01	0,3	372	13	0,056427516	28,42%	1,487848295	0,005130511
0,3	0,25	704	23	0,068435036	27,48%	2,450930886	0,008451486
0,05	0,15	276	9	0,05554689	27,83%	1,464129249	0,005048722
0,132	0,2	321	10,8	0,06466276	28,3790574%	2,189669217	0,007550584

Time	MAE	MAPE	SSE	MSE
10	0,054069397	24,08%	1,45542108	0,005018693
10	0,054993187	23,95%	1,529554004	0,005274324
10	0,055228262	24,52%	1,525055709	0,005258813
10	0,054043622	24,21%	1,460338779	0,005035651
10	0,0552208	24,16%	1,49462578	0,005153882
10	0,054711054	24,186915%	1,49299907	0,005148273

Time	MAE	MAPE	SSE	MSE
49	0,054063562	25,43%	1,431906902	0,00493761
46	0,054376966	25,26%	1,445832861	0,004985631
27	0,05469825	25,09%	1,470000843	0,005068968
7	0,053970847	25,14%	1,424334037	0,004911497
28	0,054456579	25,38%	1,437478493	0,004956822
31,4	0,054313241	25,2607%	1,441910627	0,004972106

3.2.3. ĐÁNH GIÁ

Chương 4

KẾT LUẬN

4.1. ĐÁNH GIÁ KẾT QUẢ

4.1.1. Những công việc làm được

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

- Tìm hiểu về cấu trúc và nguyên lý hoạt động của mạng neuron nhân tạo.
- Tìm hiểu về các giải thuật huấn luyện mạng neuron nhân tạo: giải thuật lan truyền ngược, giải thuật tìm kiếm ngẫu nhiên cục bộ và giải thuật mô phỏng luyện kim.
- Tìm hiểu về dữ liệu chuỗi thời gian và các tính chất của nó.
- Hiện thực một chương trình dự báo chuỗi thời gian bằng mạng neuron nhân tạo cho phép người dùng lựa chọn một trong các giải thuật huấn luyện: lan truyền ngược, tìm kiếm ngẫu nhiên cục bộ và mô phỏng luyện kim hoặc giải thuật lai kết hợp hai giải thuật với nhau.
- Thực hiện chạy thử nghiệm chương trình trên 9 bộ dữ liệu mẫu.

4.1.2. 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.

Kết quả thực nghiệm cho thấy, giải thuật lai có khả năng vượt trội hơn giải thuật lan truyền ngược khi huấn luyện một mạng lớn với bộ dữ liệu lớn.

4.1.3. Mặt hạn chế

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

4.2. HƯỚNG PHÁT TRIỂN

Trong tương lai, giải thuật mô phỏng luyện kim có thể kết hợp với giải thuật tabu search để đem lại kết quả huấn luyện tốt hơn.

TÀI LIỆU THAM KHẢO

- [1] Jeff Heaton. *Introduction to Neural Networks with C#*, ISBN 1-60439-009-3, Second Edition, 2008.
- [2] T. M. Mitchell. *Machine Learning*, McGraw-Hill Science/ Engineering/ Math, ISBN 0070428077, 1997.
- [3] Dương Tuấn Anh. Slide bài giảng *Data Mining: Classification and Prediction*, Đại học Bách khoa TP.Hồ Chí Minh, 2011.
- [4] M. Riedmiller. *Advanced Supervised Learning In Multi-layer Perceptrons – From Backpropagation To Adaptive Learning Algorithms*. Int. Journal of Computer Standards and Interfaces, 1994.
- [5] I. Kaastra, M. Boyd. **Designing A Neural Network For Forecasting Financial And Economic Time Series**. Neurocomputing, vol. 10, pages 215-236, 1996.
- [6] T. Kolarik, G. Rudorfer. *Time Series Forecasting Using Neural Networks*. ACM Sigapl Apl Quote Quad, vol. 25, no. 1, pages 86-94, 1994.
- [7] Eamonn Keogh. *URC Time series*. www.cs.ucr.edu/~eamonn/time_series_data University of California - Riverside, 2015.
- [8] 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

Phụ lục A
Bảng thuật ngữ Anh-Việt

Activation Function	Hàm kích hoạt
Artificial Neural Network	Mạng neuron nhân tạo
Autocorrelation	Hệ số tự tương quan
Autocorrelation Function	Hàm tự tương quan
Backpropagation Algorithm	Giải thuật lan truyền ngược
Batch Learning	Học theo bó
Bias	Độ lệch
Cyclical	Tính chu kỳ
Decision Surface	Mặt quyết định
Delta Rule	Luật delta
Differencing	Lấy hiệu
Feed-forward neural network	Mạng truyền thẳng
Global Minimum	Cực tiểu toàn cục
Global Optimization	Tối ưu toàn cục
Global Strategy	Chiến lược toàn cục
Gradient	Độ dốc
Gradient Descent	Giảm độ dốc
Heuristic	Kỹ thuật dựa trên kinh nghiệm
Hyperplane	Siêu phẳng
Incremental Gradient Descent	Giảm độ dốc tăng cường

Irregular	Tính bất quy tắc
Learning By Epoch	Học theo epoch
Learning By Pattern	Học theo mẫu
Learning Rate	Hệ số học
Least Absolute Deviation	Độ lệch tuyệt đối nhỏ nhất
Linear Unit	Đơn vị tuyến tính
Linearly Separable	Khả phân tuyến tính
Local Minimum	Cực tiểu cục bộ
Local Optimization	Tối ưu cục bộ
Local Strategy	Chiến lược cục bộ
Mean Absolute Errors	Trung bình tuyệt đối lỗi
Mean Squared Errors	Trung bình bình phương lỗi
Measurable Function	Hàm khả đánh giá
Metaheuristic	Kỹ thuật kinh nghiệm tổng quát
Momentum Term	Hệ số quán tính
Move	Bước di chuyển
Online Learning	Học trực tuyến
Oscillation	Giao động
Overall Error	Lỗi tổng thể
Overfitting	Quá khớp
Pattern Set	Tập mẫu
Percentage Differences	Hiệu phần trăm

Perceptron Training Rule	Luật huấn luyện perceptron
Probability Distribution	Phân bố xác suất
Random Variable	Biến ngẫu nhiên
Recurrent Neural Network	Mạng hồi quy
Seasonal	Tính mùa
Sigmoid Unit	Đơn vị sigmoid
Simulated Annealing	Mô phỏng luyện kim
Sum Of Squared Errors	Tổng bình phương lỗi
Supervised Learning	Học có giám sát
Synapse	Khớp thần kinh
Thresholds	Phân ngưỡng
Time Series	Chuỗi thời gian
Training Algorithm	Giải thuật huấn luyện
Training Error Function	Hàm lỗi
Transfer Function	Hàm truyền
Trend	Tính xu hướng
Unsupervised Learning	Học không có giám sát
Unthresholded Perceptron	Perceptron không phân ngưỡng

Phụ lục B

Chương trình thực nghiệm

Chương trình thực nghiệm được hiện thực bằng ngôn ngữ C# trên nền tảng .NET framework với IDE hỗ trợ là Microsoft Visual Studio 2013.

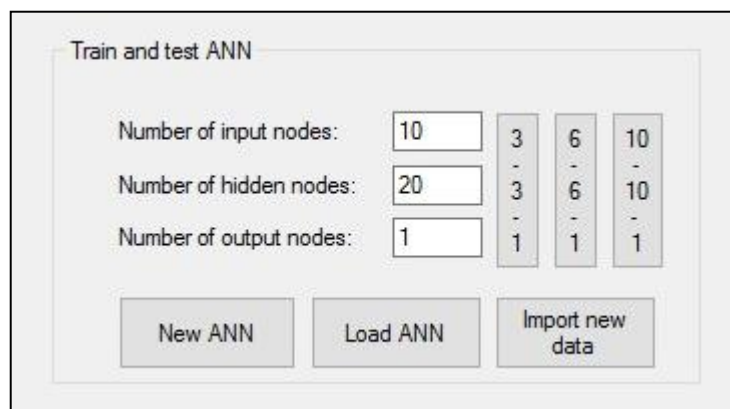
Chương trình hỗ trợ các chức năng:

- Khởi tạo mạng neuron và huấn luyện bằng 3 giải thuật (BP, SA, HY)
- Kiểm thử và hiển thị các thông số
- Save/Load mạng dưới dạng xml
- Chuẩn hóa data (time series) trước khi đưa vào huấn luyện

B.1 Giao diện và hướng dẫn sử dụng

Giao diện chính (Hình 3.1):

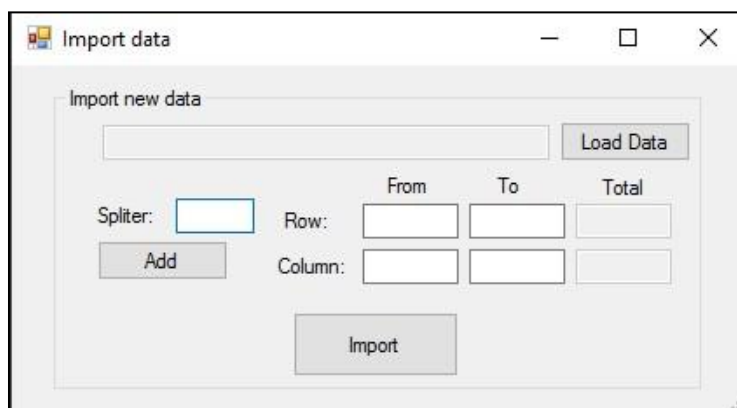
- New ANN: dựa vào số node 3 tầng mà user nhập để khởi tạo 1 mạng neuron mới
- Load ANN: cho phép user load một mạng đã được lưu trước đó
- Import new data: mở ra chức năng chuẩn hóa data phục vụ cho việc huấn luyện



Hình 3.1 Giao diện khởi đầu của chương trình demo

Giao diện chức năng chuẩn hóa dữ liệu đầu vào (Hình 3.2):

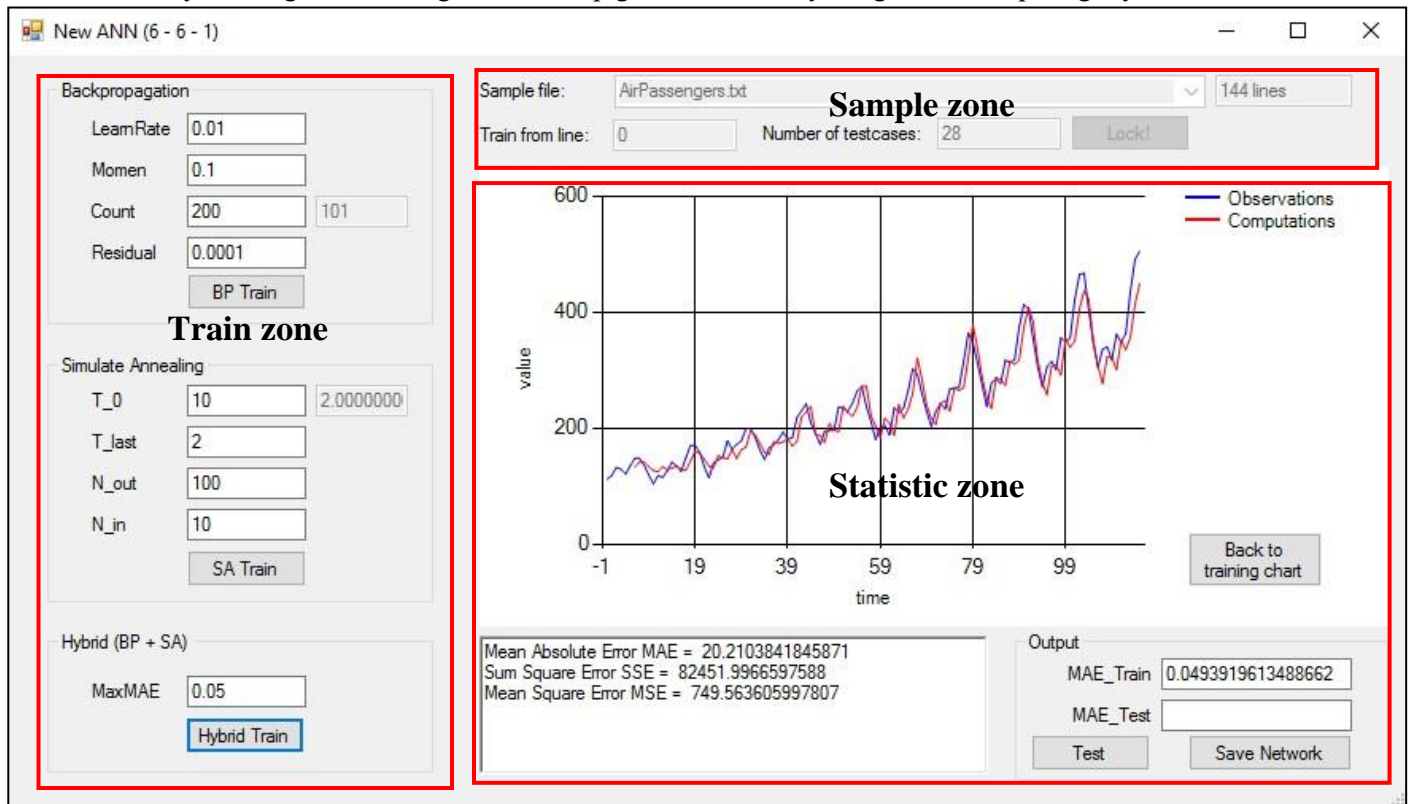
- Load data: lấy đường dẫn đến file cần thêm
- Splitter – Add: định dạng ngăn cách giữa 2 giá trị trên cùng một hàng
- Row – Column: xác định khoảng giá trị cần lấy
- Import: tiến hành xử lý để đưa file đã chuẩn hóa vào database



Hình 3.2 Giao diện chức năng chuẩn hóa dữ liệu đầu vào

Giao diện huấn luyện và kiểm thử (Hình 3.3):

- Sample zone:
 - Chọn các mẫu có sẵn trong database
 - Hiển thị tổng số lines trong mẫu
 - Chọn line bắt đầu để train
 - Chọn số lượng lines dành ra để thực hiện test
 - Khóa các lựa chọn và tiến hành train
- Train zone:
 - Chỉnh các thông số của 3 giải thuật
 - Hiển thị trạng thái tiến hành của giải thuật BP và giải thuật SA
 - Tiến hành train
- Statistic zone:
 - Biểu đồ hiển thị kết quả train và test mạng
 - Hiển thị các thông số kết quả khi train và test
 - Lưu mạng khi kết thúc train



Hình 3.3 Giao diện huấn luyện và kiểm thử

B.2 Cấu trúc cơ bản các lớp và các hàm trong chương trình:

Có 2 lớp (class) chính trong chương trình:

- Lớp Perceptron gồm:
 - Thuộc tính hàm truyền
 - Hàm CalOutput(double Input): dựa vào input và hàm truyền để tính ra output
- Lớp ArtificialNeuronNetwork:
 - Thuộc tính số lượng perceptron mỗi tầng (n_input, n_hidden, n_output)
 - Hàm khởi tạo ANN(int n_input, int n_hidden, int n_output): dựa vào số lượng perceptron mỗi tầng khởi tạo ra các mảng trọng số cần cho mạng neuron
 - Hàm huấn luyện BP, SA, HY: dựa vào thông số đưa vào để thực hiện huấn luyện thay đổi mạng hiện tại

B.3 Cấu trúc định dạng file XML lưu trữ định dạng mạng

Dưới đây là một bản rút gọn của một mạng neuron khi được lưu lại:

```
<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.