

## MỤC LỤC

<b>MỞ ĐẦU .....</b>	<b>2</b>
<b>CHƯƠNG I: CÁC KHÁI NIỆM CƠ BẢN VỀ MẠNG NƠON .....</b>	<b>4</b>
<b>1.1. Sơ lược về mạng nơon .....</b>	<b>4</b>
1.1.1. Lịch sử phát triển .....	4
1.1.2. Ứng dụng .....	6
1.1.3. Căn nguyên sinh học .....	6
<b>1.2. Đơn vị xử lý .....</b>	<b>8</b>
<b>1.3. Hàm xử lý .....</b>	<b>9</b>
1.3.1. Hàm kết hợp .....	9
1.3.2. Hàm kích hoạt (hàm chuyển) .....	9
<b>1.4. Các hình trạng của mạng .....</b>	<b>12</b>
1.4.1. Mạng truyền thẳng .....	12
1.4.2. Mạng hồi quy .....	13
<b>1.5. Mạng học .....</b>	<b>13</b>
1.5.1. Học có thầy .....	13
1.5.2. Học không có thầy .....	14
<b>1.6. Hàm mục tiêu .....</b>	<b>14</b>
<b>CHƯƠNG II. MẠNG NƠON TRUYỀN THẲNG VÀ THUẬT TOÁN LAN TRUYỀN NGƯỢC ..</b>	<b>16</b>
<b>2.1. Kiến trúc cơ bản .....</b>	<b>16</b>
2.1.1. Mạng truyền thẳng .....	16
2.1.2. Mạng hồi quy .....	18
<b>2.2. Khả năng thể hiện .....</b>	<b>19</b>
<b>2.3. Vấn đề thiết kế cấu trúc mạng .....</b>	<b>19</b>
2.3.1. Số lớp ẩn .....	19
2.3.2. Số đơn vị trong lớp ẩn .....	20
<b>2.4. Thuật toán lan truyền ngược (Back-Propagation) .....</b>	<b>21</b>
2.4.1. Mô tả thuật toán .....	22
2.4.2. Sử dụng thuật toán lan truyền ngược .....	27
2.4.3. Một số biến thể của thuật toán lan truyền ngược .....	31
2.4.4. Nhận xét .....	36
<b>2.5. Các thuật toán tối ưu khác .....</b>	<b>38</b>
2.5.1. Thuật toán giả luyện kim (Simulated annealing) .....	38
2.5.2. Thuật giải di truyền (Genetic Algorithm) .....	39
<b>CHƯƠNG III. ỨNG DỤNG MẠNG NƠON TRUYỀN THẲNG TRONG DỰ BÁO DỮ LIỆU ...</b>	<b>41</b>
<b>3.1. Sơ lược về lĩnh vực dự báo dữ liệu .....</b>	<b>41</b>
<b>3.2. Thu thập, phân tích và xử lý dữ liệu .....</b>	<b>42</b>
3.2.1. Kiểu của các biến .....	43
3.2.2. Thu thập dữ liệu .....	44
3.2.3. Phân tích dữ liệu .....	45
3.2.4. Xử lý dữ liệu .....	46
3.2.5. Tổng hợp .....	48
<b>3.3. Chương trình dự báo dữ liệu .....</b>	<b>48</b>
3.3.1. Các bước chính trong quá trình thiết kế và xây dựng .....	48
3.3.2. Xây dựng chương trình .....	54
3.3.3. Chương trình dự báo dữ liệu .....	69
<b>3.4. Một số nhận xét .....</b>	<b>75</b>
<b>KẾT LUẬN .....</b>	<b>77</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>79</b>

## MỞ ĐẦU



Cùng với sự phát triển của mô hình kho dữ liệu (Dataware house), ở Việt nam ngày càng có nhiều kho dữ liệu với lượng dữ liệu rất lớn. Để khai thác có hiệu quả những dữ liệu khổng lồ này, đã có nhiều công cụ được xây dựng để thỏa mãn nhu cầu khai thác dữ liệu mức cao, chẳng hạn như công cụ khai thác dữ liệu Oracle Discoverer của hãng Oracle. Công cụ này được sử dụng như một bộ phân tích dữ liệu đa năng theo nhiều chiều dữ liệu, đặc biệt theo thời gian. Hay là việc xây dựng các hệ chuyên gia, các hệ thống dựa trên một cơ sở tri thức của các chuyên gia, để có thể dự báo được khuynh hướng phát triển của dữ liệu, thực hiện các phân tích trên các dữ liệu của tổ chức. Mặc dù các công cụ, các hệ thống trên hoàn toàn có thể thực hiện được phần lớn các công việc nêu trên, chúng vẫn yêu cầu một độ chính xác, đầy đủ nhất định về mặt dữ liệu để có thể đưa ra được các câu trả lời chính xác.

Trong khi đó, các ứng dụng của mạng nơron truyền thẳng được xây dựng dựa trên các nhân tố ảnh hưởng đến sự thay đổi của dữ liệu đã được thực tiễn chứng minh là khá mạnh và hiệu quả trong các bài toán dự báo, phân tích dữ liệu. Chúng có thể được huấn luyện và ánh xạ từ các dữ liệu vào tới các dữ liệu ra mà không yêu cầu các dữ liệu đó phải đầy đủ. Trong số các loại mạng tương đối phổ biến thì các mạng neuron truyền thẳng nhiều lớp, được huấn luyện bằng thuật toán lan truyền ngược được sử dụng nhiều nhất. Các mạng nơron này có khả năng biểu diễn các ánh xạ phi tuyến giữa đầu vào và đầu ra, chúng được coi như là các “bộ xấp xỉ đa năng”. Việc ứng dụng của loại mạng này chủ yếu là cho việc phân tích, dự báo, phân loại các số liệu thực tế. Đặc biệt đối với việc dự báo khuynh hướng thay đổi của các dữ liệu tác nghiệp trong các cơ quan, tổ chức kinh tế, xã hội,... Nếu có thể dự báo được khuynh hướng thay đổi của dữ liệu với một độ tin cậy nhất định, các nhà lãnh đạo có thể đưa ra được các quyết sách đúng đắn cho cơ quan, tổ chức của mình.

Luận văn này được thực hiện với mục đích tìm hiểu và làm sáng tỏ một số khía cạnh về mạng nơron truyền thẳng nhiều lớp, thuật toán lan truyền ngược và ứng dụng chúng trong giải quyết các bài toán trong lĩnh vực dự báo dữ liệu.

*Tác giả xin chân thành cảm ơn sự giúp đỡ về mặt khoa học cũng như sự động viên của các đồng nghiệp trong phòng Công nghệ phần mềm trong quản lý - Viện Công nghệ thông tin trong suốt quá trình thực hiện luận văn. Đặc biệt, tác giả xin chân thành cảm ơn **TS. Lê Hải Khôi**, người thầy đã giúp đỡ các ý kiến quý báu để tác giả có thể hoàn thành tốt luận văn này.*

Hà nội, tháng 12 năm 2002

**Trần Đức Minh**

## CHƯƠNG I: CÁC KHÁI NIỆM CƠ BẢN VỀ MẠNG NƠN

Chương này đề cập các vấn đề sau:

- 1.1. Sơ lược về mạng nơron
- 1.2. Đơn vị xử lý
- 1.3. Hàm xử lý
- 1.4. Các hình trạng của mạng
- 1.5. Mạng học
- 1.6. Hàm mục tiêu

---

### 1.1. Sơ lược về mạng nơron

#### 1.1.1. Lịch sử phát triển

Sự phát triển của mạng nơron trải qua cả quá trình đưa ra các khái niệm mới lẫn thực thi những khái niệm này.

Dưới đây là các mốc đáng chú ý trong lịch sử phát triển của mạng nơron.

- Cuối TK 19, đầu TK 20, sự phát triển chủ yếu chỉ là những công việc có sự tham gia của cả ba ngành Vật lý học, Tâm lý học và Thần kinh học, bởi các nhà khoa học như Hermann von Helmholtz, Ernst Mach, Ivan Pavlov. Các công trình nghiên cứu của họ chủ yếu đi sâu vào các lý thuyết tổng quát về HỌC (Learning), NHÌN (vision) và LẬP LUẬN (conditioning),... và không hề đưa ra những mô hình toán học cụ thể mô tả hoạt động của các nơron.
- Mọi chuyện thực sự bắt đầu vào những năm 1940 với công trình của Warren McCulloch và Walter Pitts. Họ chỉ ra rằng về nguyên tắc, mạng của các nơron nhân tạo có thể tính toán bất kỳ một hàm số học hay logic nào!
- Tiếp theo hai người là Donald Hebb, ông đã phát biểu rằng việc thuyết lập luận cổ điển (classical conditioning) (như Pavlov đưa ra) là hiện thực bởi do các thuộc tính của từng nơron riêng biệt. Ông cũng nêu ra một phương pháp học của các nơron nhân tạo.
- Ứng dụng thực nghiệm đầu tiên của các nơron nhân tạo có được vào cuối những năm 50 cùng với phát minh của mạng nhận thức (perceptron network) và luật học tương ứng

bởi Frank Rosenblatt. Mạng này có khả năng nhận dạng các mẫu. Điều này đã mở ra rất nhiều hy vọng cho việc nghiên cứu mạng nơron. Tuy nhiên nó có hạn chế là chỉ có thể giải quyết một số lớp hữu hạn các bài toán.

- Cùng thời gian đó, Bernard Widrow và Ted Hoff đã đưa ra một thuật toán học mới và sử dụng nó để huấn luyện cho các mạng nơron tuyến tính thích nghi, mạng có cấu trúc và chức năng tương tự như mạng của Rosenblatt. Luật học Widrow-Hoff vẫn còn được sử dụng cho đến nay.
- Tuy nhiên cả Rosenblatt và Widrow-Hoff đều cùng vấp phải một vấn đề do Marvin Minsky và Seymour Papert phát hiện ra, đó là các mạng nhận thức chỉ có khả năng giải quyết các bài toán khả phân tuyến tính. Họ cố gắng cải tiến luật học và mạng để có thể vượt qua được hạn chế này nhưng họ đã không thành công trong việc cải tiến luật học để có thể huấn luyện được các mạng có cấu trúc phức tạp hơn.
- Do những kết quả của Minsky-Papert nên việc nghiên cứu về mạng nơron gần như bị đình lại trong suốt một thập kỷ do nguyên nhân là không có được các máy tính đủ mạnh để có thể thực nghiệm.
- Mặc dù vậy, cũng có một vài phát kiến quan trọng vào những năm 70. Năm 1972, Teuvo Kohonen và James Anderson độc lập nhau phát triển một loại mạng mới có thể hoạt động như một bộ nhớ. Stephen Grossberg cũng rất tích cực trong việc khảo sát các mạng tự tổ chức (Self organizing networks).
- Vào những năm 80, việc nghiên cứu mạng nơron phát triển rất mạnh mẽ cùng với sự ra đời của PC. Có hai khái niệm mới liên quan đến sự hồi sinh này, đó là:
  1. Việc sử dụng các phương pháp thống kê để giải thích hoạt động của một lớp các mạng hồi quy (recurrent networks) có thể được dùng như bộ nhớ liên hợp (associative memory) trong công trình của nhà vật lý học John Hopfield.
  2. Sự ra đời của thuật toán lan truyền ngược (back-propagation) để luyện các mạng nhiều lớp được một vài nhà nghiên cứu độc lập tìm ra như: David Rumelhart, James McClelland,... Đó cũng là câu trả lời cho Minsky-Papert.

### ***1.1.2. Ứng dụng***

Trong quá trình phát triển, mạng nơron đã được ứng dụng thành công trong rất nhiều lĩnh vực. Dưới đây liệt kê ra một số ứng dụng chính của mạng nơron:

- ✓ Aerospace: Phi công tự động, giả lập đường bay, các hệ thống điều khiển lái máy bay, bộ phát hiện lỗi.
- ✓ Automotive: Các hệ thống dẫn đường tự động cho ô tô, các bộ phân tích hoạt động của xe.
- ✓ Banking: Bộ đọc séc và các tài liệu, tính tiền của thẻ tín dụng.
- ✓ Defense: Định vị - phát hiện vũ khí, dò mục tiêu, phát hiện đối tượng, nhận dạng nét mặt, các bộ cảm biến thể hệ mới, xử lý ảnh radar,...
- ✓ Electronics: Dự đoán mã tuần tự, sơ đồ chip IC, điều khiển tiến trình, phân tích nguyên nhân hỏng chip, nhận dạng tiếng nói, mô hình phi tuyến.
- ✓ Entertainment: Hoạt hình, các hiệu ứng đặc biệt, dự báo thị trường.
- ✓ Financial: Định giá bất động sản, cho vay, kiểm tra tài sản cầm cố, đánh giá mức độ hợp tác, phân tích đường tín dụng, chương trình thương mại qua giấy tờ, phân tích tài chính liên doanh, dự báo tỷ giá tiền tệ.
- ✓ Insurance: Đánh giá việc áp dụng chính sách, tối ưu hóa sản phẩm.
- ✓ .....

### ***1.1.3. Căn nguyên sinh học***

Bộ não con người chứa khoảng  $10^{11}$  các phần tử liên kết chặt chẽ với nhau (khoảng  $10^4$  liên kết đối với mỗi phần tử) gọi là các nơron. Dưới con mắt của những người làm tin học, một nơron được cấu tạo bởi các thành phần: tế bào hình cây (dendrite) - tế bào thân (cell body) – và sợi trục thần kinh (axon). Tế bào hình cây có nhiệm vụ mang các tín hiệu điện tới tế bào thân, tế bào thân sẽ thực hiện gộp (Sum) và phân ngưỡng (Thresholds) các tín hiệu đến. Sợi trục thần kinh làm nhiệm vụ đưa tín hiệu từ tế bào thân ra ngoài.

Điểm tiếp xúc giữa một sợi trục thần kinh của nơron này và tế bào hình cây của một nơron khác được gọi là khớp thần kinh (synapse). Sự sắp xếp của các nơron và mức độ mạnh yếu

của các khớp thần kinh được quyết định bởi các quá trình hóa học phức tạp, sẽ thiết lập chức năng của mạng noron.

Một vài noron có sẵn từ khi sinh ra, các phần khác được phát triển thông qua việc học, ở đó có sự thiết lập các liên kết mới và loại bỏ các liên kết cũ.

Cấu trúc của mạng noron luôn luôn phát triển và thay đổi. Các thay đổi sau này có khuynh hướng bao gồm chủ yếu là việc làm tăng hay giảm độ mạnh của các mối liên kết thông qua các khớp thần kinh.

Mạng noron nhân tạo không tiếp cận đến sự phức tạp của bộ não. Mặc dù vậy, có hai sự tương quan cơ bản giữa mạng noron nhân tạo và sinh học. Thứ nhất, cấu trúc khối tạo thành chúng đều là các thiết bị tính toán đơn giản (mạng noron nhân tạo đơn giản hơn nhiều) được liên kết chặt chẽ với nhau. Thứ hai, các liên kết giữa các noron quyết định chức năng của mạng.

Cần chú ý rằng mặc dù mạng noron sinh học hoạt động rất chậm so với các linh kiện điện tử ( $10^{-3}$  giây so với  $10^{-9}$  giây), nhưng bộ não có khả năng thực hiện nhiều công việc nhanh hơn nhiều so với các máy tính thông thường. Đó một phần là do cấu trúc song song của mạng noron sinh học: toàn bộ các noron hoạt động một cách đồng thời tại một thời điểm. Mạng noron nhân tạo cũng chia sẻ đặc điểm này. Mặc dù hiện nay, các mạng noron chủ yếu được thực nghiệm trên các máy tính số, nhưng cấu trúc song song của chúng khiến chúng ta có thể thấy cấu trúc phù hợp nhất là thực nghiệm chúng trên các vi mạch tích hợp lớn (VLSI: Very Large Scale Integrated-circuit), các thiết bị quang và các bộ xử lý song song.

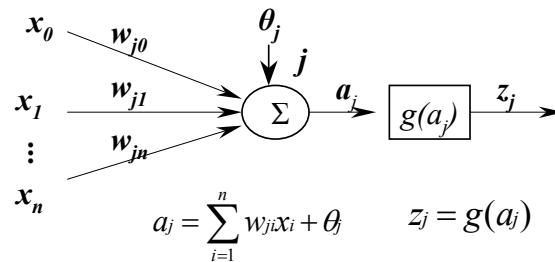
Mạng noron, đôi khi được xem như là các mô hình liên kết (connectionist models), là các mô hình phân bố song song (parallel-distributed models) có các đặc trưng phân biệt sau:

- 1) Tập các đơn vị xử lý;
- 2) Trạng thái kích hoạt hay là đầu ra của đơn vị xử lý;
- 3) Liên kết giữa các đơn vị. Xét tổng quát, mỗi liên kết được định nghĩa bởi một trọng số  $w_{jk}$  cho ta biết hiệu ứng mà tín hiệu của đơn vị  $j$  có trên đơn vị  $k$ ;
- 4) Một luật lan truyền quyết định cách tính tín hiệu ra của từng đơn vị từ đầu vào của nó;

- 5) Một hàm kích hoạt, hay hàm chuyển (activation function, transfer function), xác định mức độ kích hoạt khác dựa trên mức độ kích hoạt hiện tại;
- 6) Một đơn vị điều chỉnh (độ lệch) (bias, offset) của mỗi đơn vị;
- 7) Phương pháp thu thập thông tin (luật học - learning rule);
- 8) Môi trường hệ thống có thể hoạt động.

## 1.2. Đơn vị xử lý

Một đơn vị xử lý (Hình 1), cũng được gọi là một nơron hay một nút (node), 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 sẽ được lan truyền sang các đơn vị khác.



Hình 1: Đơn vị xử lý (Processing unit)

trong đó:

$x_i$ : các đầu vào

$w_{ji}$ : các trọng số tương ứng với các đầu vào

$\theta_j$ : độ lệch (bias)

$a_j$ : đầu vào mạng (net-input)

$z_j$ : đầu ra của nơron

$g(x)$ : hàm chuyển (hàm kích hoạt).

Trong một mạng nơron có ba kiểu đơn vị:

- 1) Các đơn vị đầu vào (Input units), nhận tín hiệu từ bên ngoài;
- 2) Các đơn vị đầu ra (Output units), gửi dữ liệu ra bên ngoài;



3) Các đơn vị ẩn (Hidden units), tín hiệu vào (input) và ra (output) của nó nằm trong mạng.

Mỗi đơn vị  $j$  có thể có một hoặc nhiều đầu vào:  $x_0, x_1, x_2, \dots, x_n$ , nhưng chỉ có một đầu ra  $z_j$ . Một đầu vào tới một đơn vị có thể là dữ liệu từ bên ngoài mạng, hoặc đầu ra của một đơn vị khác, hoặc là đầu ra của chính nó.

### 1.3. Hàm xử lý

#### 1.3.1. Hàm kết hợp

Mỗi một đơn vị trong một mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với các đơn vị khác, sinh ra một giá trị gọi là **net input**. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi một luật lan truyền cụ thể. Trong phần lớn các mạng nơron, chúng ta giả sử rằng mỗi một đơn vị cung cấp một bộ cộng như là đầu vào cho đơn vị mà nó có liên kết. Tổng đầu vào đơn vị  $j$  đơn giản chỉ là tổng trọng số của các đầu ra riêng lẻ từ các đơn vị kết nối cộng thêm ngưỡng hay độ lệch (bias)  $\theta_j$ :

$$a_j = \sum_{i=1}^n w_{ji}x_i + \theta_j$$

Trường hợp  $w_{ji} > 0$ , nơron được coi là đang ở trong trạng thái kích thích. Tương tự, nếu như  $w_{ji} < 0$ , nơron ở trạng thái kiềm chế. Chúng ta gọi các đơn vị với luật lan truyền như trên là các sigma units.

Trong một vài trường hợp người ta cũng có thể sử dụng các luật lan truyền phức tạp hơn. Một trong số đó là luật sigma-pi, có dạng như sau:

$$a_j = \sum_{i=1}^n w_{ji} \prod_{k=1}^m x_{ik} + \theta_j$$

Rất nhiều hàm kết hợp sử dụng một "độ lệch" hay "ngưỡng" để tính net input tới đơn vị. Đối với một đơn vị đầu ra tuyến tính, thông thường,  $\theta_j$  được chọn là hằng số và trong bài toán xấp xỉ đa thức  $\theta_j = 1$ .

#### 1.3.2. Hàm kích hoạt (hàm chuyển)

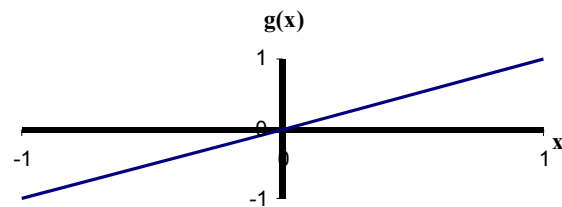
Phần lớn các đơn vị trong mạng nơron chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) gọi là hàm kích hoạt, kết quả của hàm này là một giá trị

gọi là mức độ kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing). Các hàm kích hoạt hay được sử dụng là:

1) Hàm đồng nhất (Linear function, Identity function )

$$g(x) = x$$

Nếu coi các đầu vào là một đơn vị thì chúng sẽ sử dụng hàm này. Đôi khi một hằng số được nhân với net-input để tạo ra một hàm đồng nhất.



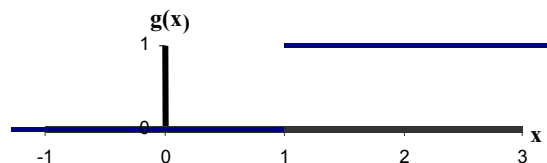
Hình 2: Hàm đồng nhất (Identity function)

2) Hàm bước nhị phân (Binary step function, Hard limit function)

Hàm này cũng được biết đến với tên "Hàm ngưỡng" (Threshold function hay Heaviside function). Đầu ra của hàm này được giới hạn vào một trong hai giá trị:

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

Dạng hàm này được sử dụng trong các mạng chỉ có một lớp. Trong hình vẽ sau,  $\theta$  được chọn bằng 1.

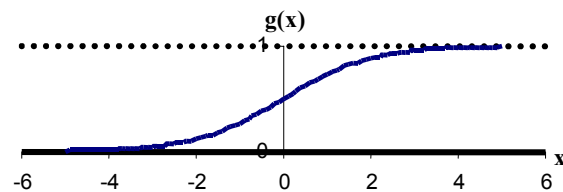


Hình 3: Hàm bước nhị phân (Binary step function)

3) Hàm sigmoid (Sigmoid function (logsig))

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

Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện (trained) bởi thuật toán **Lan truyền ngược** (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng  $[0,1]$ .

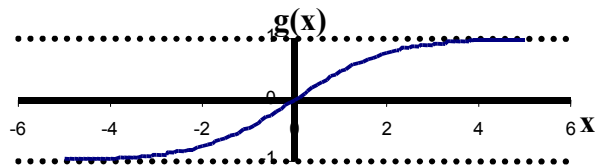


Hình 4: Hàm Sigmoid

4) Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig))

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

Hàm này có các thuộc tính tương tự hàm sigmoid. Nó làm việc tốt đối với các ứng dụng có đầu ra yêu cầu trong khoảng  $[-1,1]$ .



Hình 5: Hàm sigmoid lưỡng cực

Các hàm chuyển của các đơn vị ẩn (hidden units) là cần thiết để biểu diễn sự phi tuyến vào trong mạng. Lý do là hợp thành của các hàm đồng nhất là một hàm đồng nhất. Mặc dù vậy nhưng nó mang tính chất phi tuyến (nghĩa là, khả năng biểu diễn các hàm phi tuyến) làm cho

các mạng nhiều tầng có khả năng rất tốt trong biểu diễn các ánh xạ phi tuyến. Tuy nhiên, đối với luật học lan truyền ngược, hàm phải khả vi (differentiable) và sẽ có ích nếu như hàm được gán trong một khoảng nào đó. Do vậy, hàm sigmoid là lựa chọn thông dụng nhất.

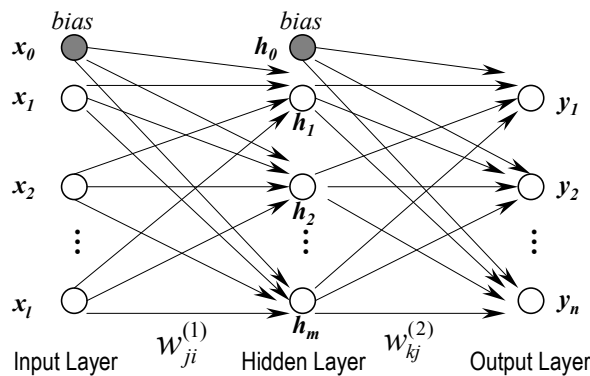
Đối với các đơn vị đầu ra (output units), các hàm chuyển cần được chọn sao cho phù hợp với sự phân phối của các giá trị đích mong muốn. Chúng ta đã thấy rằng đối với các giá trị ra trong khoảng  $[0,1]$ , hàm sigmoid là có ích; đối với các giá trị đích mong muốn là liên tục trong khoảng đó thì hàm này cũng vẫn có ích, nó có thể cho ta các giá trị ra hay giá trị đích được căn trong một khoảng của hàm kích hoạt đầu ra. Nhưng nếu các giá trị đích không được biết trước khoảng xác định thì hàm hay được sử dụng nhất là hàm đồng nhất (identity function). Nếu giá trị mong muốn là dương nhưng không biết cận trên thì nên sử dụng một hàm kích hoạt dạng mũ (exponential output activation function).

#### 1.4. Các hình trạng của mạng

Hình trạng của mạng được định nghĩa bởi: số lớp (layers), số đơn vị trên mỗi lớp, và sự liên kết giữa các lớp như thế nào. Các mạng về tổng thể được chia thành hai loại dựa trên cách thức liên kết các đơn vị:

##### 1.4.1. Mạng truyền thẳng

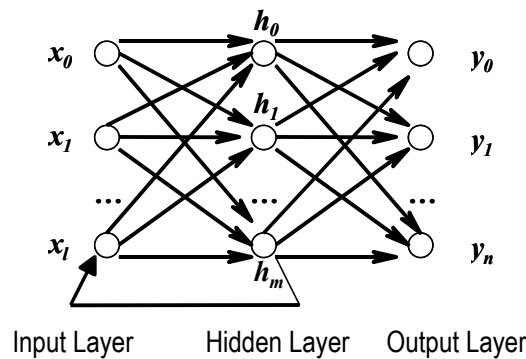
Dòng dữ liệu từ đơn vị đầu vào đến đơn vị đầu ra chỉ được truyền thẳng. Việc xử lý dữ liệu có thể mở rộng ra nhiều lớp, nhưng không có các liên kết phản hồi. Nghĩa là, các liên kết mở rộng từ các đơn vị đầu ra tới các đơn vị đầu vào trong cùng một lớp hay các lớp trước đó là không cho phép.



Hình 6: Mạng nơron truyền thẳng nhiều lớp (Feed-forward neural network)

### 1.4.2. Mạng hồi quy

Có chứa các liên kết ngược. Khác với mạng truyền thẳng, các thuộc tính động của mạng mới quan trọng. Trong một số trường hợp, các giá trị kích hoạt của các đơn vị trải qua quá trình nói lòng (tăng giảm số đơn vị và thay đổi các liên kết) cho đến khi mạng đạt đến một trạng thái ổn định và các giá trị kích hoạt không thay đổi nữa. Trong các ứng dụng khác mà cách chạy động tạo thành đầu ra của mạng thì những sự thay đổi các giá trị kích hoạt là đáng quan tâm.



Hình 7: Mạng neuron hồi quy (Recurrent neural network)

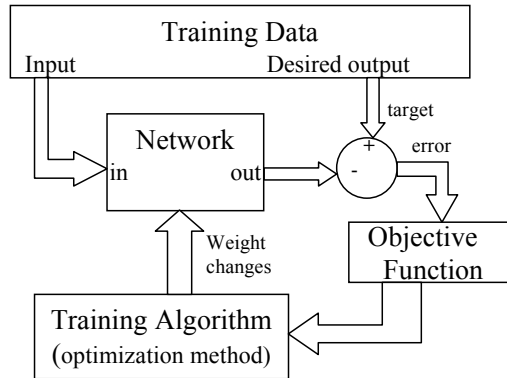
## 1.5. Mạng học

Chức năng của một mạng neuron được quyết định bởi các nhân tố như: hình trạng mạng (số lớp, số đơn vị trên mỗi tầng, và cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Hình trạng 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 đích mong muốn được gọi là học (learning) hay huấn luyện (training). Rất nhiều thuật toán học 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 thầy (Supervised learning) và Học không có thầy (Unsupervised Learning).

### 1.5.1. Học có thầy

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 được cung cấp bởi "thầy giáo", hay bởi hệ thống trên đó mạng hoạt động. Sự khác biệt giữa các đầu ra thực tế 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.



Hình 8: Mô hình Học có thầy (Supervised learning model)

### 1.5.2. Học không có thầy

Với cách học không có thầy, 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 thầy, các đích trùng với đầu vào. Nói một cách khác, học không có thầy luôn thực hiện một công việc tương tự như một mạng tự liên hợp, cô đọng thông tin từ dữ liệu vào.

## 1.6. Hàm mục tiêu

Để huấn luyện một mạng và xét xem nó thực hiện tốt đến đâu, ta cần xây dựng một hàm mục tiêu (hay hàm giá) để cung cấp cách thức đánh giá khả năng hệ thống một cách không nhập nhằng. Việc chọn hàm mục tiêu là rất quan trọng bởi vì hàm này thể hiện các mục tiêu thiết kế và quyết định thuật toán huấn luyện nào có thể được áp dụng. Để phát triển một hàm mục tiêu đo được chính xác cái chúng ta muốn không phải là việc dễ dàng. Một vài hàm cơ bản được sử dụng rất rộng rãi. Một trong số chúng là hàm tổng bình phương lỗi (sum of squares error function),

$$E = \frac{1}{NP} \sum_{p=1}^P \sum_{i=1}^N (t_{pi} - y_{pi})^2 ,$$

trong đó:

$p$ : số thứ tự mẫu trong tập huấn luyện

$i$  : số thứ tự của đơn vị đầu ra

$t_{pi}$  và  $y_{pi}$  : tương ứng là đầu ra mong muốn và đầu ra thực tế của mạng cho đơn vị đầu ra thứ  $i$  trên mẫu thứ  $p$ .

Trong các ứng dụng thực tế, nếu cần thiết có thể làm phức tạp hàm số với một vài yếu tố khác để có thể kiểm soát được sự phức tạp của mô hình.

## CHƯƠNG II. MẠNG NƠN TRUYỀN THẲNG VÀ THUẬT TOÁN LAN TRUYỀN NGƯỢC

Chương này đề cập các vấn đề sau:

- 2.1. Kiến trúc cơ bản
- 2.2. Khả năng thể hiện
- 2.3. Vấn đề thiết kế cấu trúc mạng
- 2.4. Thuật toán lan truyền ngược (Back-Propagation)
- 2.5. Các thuật toán tối ưu khác

---

### 2.1. Kiến trúc cơ bản

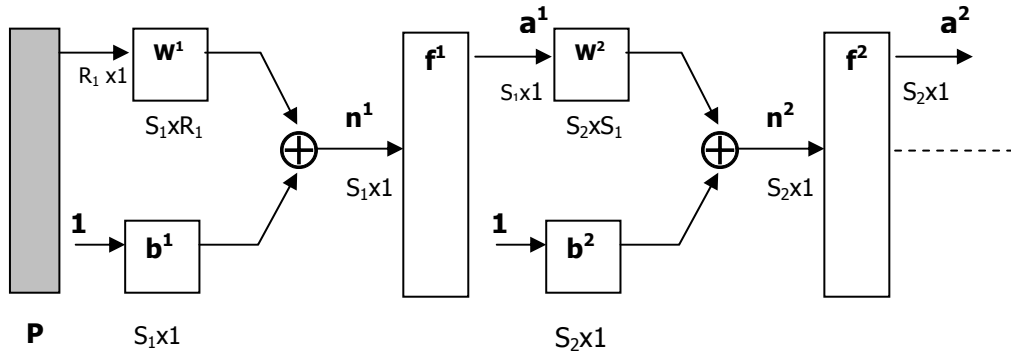
Để đơn giản và tránh hiểu nhầm, mạng truyền thẳng xét trong chương này là các mạng truyền thẳng có nhiều lớp. Kiến trúc mạng truyền thẳng nhiều lớp (Multi-layer Feed Forward - MLFF) là kiến trúc chủ đạo của các mạng nơon hiện tại. Mặc dù có khá nhiều biến thể nhưng đặc trưng của kiến trúc này là cấu trúc và thuật toán học là đơn giản và nhanh (Masters 1993).

#### 2.1.1. Mạng truyền thẳng

Một mạng truyền thẳng nhiều lớp bao gồm một lớp vào, một lớp ra và một hoặc nhiều lớp ẩn. Các nơon đầu vào thực chất không phải các nơon theo đúng nghĩa, bởi lẽ chúng không thực hiện bất kỳ một tính toán nào trên dữ liệu vào, đơn giản nó chỉ tiếp nhận các dữ liệu vào và chuyển cho các lớp kế tiếp. Các nơon ở lớp ẩn và lớp ra mới thực sự thực hiện các tính toán, kết quả được định dạng bởi hàm đầu ra (hàm chuyển). Cụm từ “truyền thẳng” (feed forward) (không phải là trái nghĩa của lan truyền ngược) liên quan đến một thực tế là tất cả các nơon chỉ có thể được kết nối với nhau theo một hướng: tới một hay nhiều các nơon khác trong lớp kế tiếp (loại trừ các nơon ở lớp ra).

Hình sau ở dạng tóm tắt biểu diễn mạng nơon một cách cô đọng và tránh gây ra sự hiểu nhầm.





Hình 9: Mạng nơron truyền thẳng nhiều lớp

trong đó:

$P$ : Vector đầu vào (vector cột)

$W^i$ : Ma trận trọng số của các nơron lớp thứ  $i$ .

( $S^i \times R^i$ :  $S$  hàng (nơron) -  $R$  cột (số đầu vào))

$b^i$ : Vector độ lệch (*bias*) của lớp thứ  $i$  ( $S^i \times 1$ : cho  $S$  nơron)

$n^i$ : net input ( $S^i \times 1$ )

$f^i$ : Hàm chuyển (hàm kích hoạt)

$a^i$ : net output ( $S^i \times 1$ )

$\oplus$ : Hàm tổng thông thường.

Mỗi liên kết gắn với một trọng số, trọng số này được thêm vào trong quá trình tín hiệu đi qua liên kết đó. Các trọng số có thể dương, thể hiện trạng thái kích thích, hay âm, thể hiện trạng thái kiềm chế. Mỗi nơron tính toán mức kích hoạt của chúng bằng cách cộng tổng các đầu vào và đưa ra hàm chuyển. Một khi đầu ra của tất cả các nơron trong một lớp mạng cụ thể đã thực hiện xong tính toán thì lớp kế tiếp có thể bắt đầu thực hiện tính toán của mình bởi vì đầu ra của lớp hiện tại tạo ra đầu vào của lớp kế tiếp. Khi tất cả các nơron đã thực hiện tính toán thì kết quả được trả lại bởi các nơron đầu ra. Tuy nhiên, có thể là chưa đúng yêu cầu, khi đó một thuật toán huấn luyện cần được áp dụng để điều chỉnh các tham số của mạng.

Trong hình 9, số nơron ở lớp thứ nhất, và lớp thứ hai tương ứng là  $S_1$  và  $S_2$ . Ma trận trọng số đối với các lớp tương ứng là  $W^1$  và  $W^2$ . Có thể thấy sự liên kết giữa các lớp mạng thể hiện trong hình vẽ 9: ở lớp thứ 2, vector đầu vào chính là **net output** của lớp thứ nhất.

Tương tự như vậy, nếu thêm vào các lớp khác nữa vào trong cấu trúc này thì lớp mạng cuối cùng thường là lớp cho ra kết quả của toàn bộ mạng, lớp đó gọi là lớp ra (OUTPUT LAYER).

Mạng có nhiều lớp có khả năng tốt hơn là các mạng chỉ có một lớp, chẳng hạn như mạng hai lớp với lớp thứ nhất sử dụng hàm sigmoid và lớp thứ hai dùng hàm đồng nhất có thể áp dụng để xấp xỉ các hàm toán học khá tốt, trong khi các mạng chỉ có một lớp thì không có khả năng này.

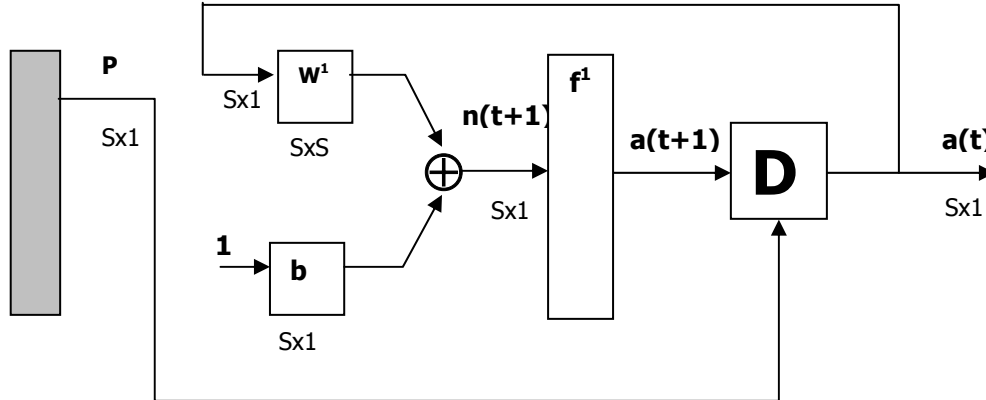
Xét trường hợp mạng có hai lớp như hình vẽ 9, công thức tính toán cho đầu ra như sau:

$$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2(\mathbf{f}^1(\mathbf{W}^1\mathbf{P} + \mathbf{b}^1)) + \mathbf{b}^2)$$

trong đó, ý nghĩa của các ký hiệu như đã nêu trong hình vẽ 9.

### 2.1.2. Mạng hồi quy

Bên cạnh mạng truyền thẳng còn có những dạng mạng khác như các mạng hồi quy. Các mạng hồi quy thường có các liên kết ngược từ các lớp phía sau đến các lớp phía trước hay giữa các nơron trong bản thân một lớp.



Hình 10: Một ví dụ của mạng hồi quy

Trong hình vẽ 10, **D** là đơn vị làm trễ đầu vào nó một bước.

Để thấy rằng, các mạng thuộc lớp các mạng truyền thẳng dễ dàng hơn cho ta trong việc phân tích lý thuyết bởi lẽ đầu ra của các mạng này có thể được biểu diễn bởi một hàm của các trọng số và các đầu vào (Sau này, khi xây dựng các thuật toán huấn luyện ta sẽ thấy điều này).

## 2.2. Khả năng thể hiện

Các mạng truyền thẳng cho ta một kiến trúc tổng quát thể hiện khả năng ánh xạ hàm phi tuyến tính giữa một tập các biến đầu vào và tập các đầu ra. Khả năng thể hiện của một mạng có thể được định nghĩa là khoảng mà nó có thể thực hiện ánh xạ khi các trọng số biến thiên. Theo [15]:

- 1) Các mạng một lớp chỉ có khả năng thể hiện các hàm khả phân tuyến tính hay các miền phân chia được (ví dụ như hàm logic AND có miền giá trị có thể phân chia được bằng một đường thẳng trong khi miền giá trị của hàm XOR thì không).
- 2) Các mạng có hai lớp ẩn có khả năng thể hiện một đường biên phân chia tùy ý với một độ chính xác bất kỳ với các hàm chuyển phân ngưỡng và có thể xấp xỉ bất kỳ ánh xạ mìn nào với độ chính xác bất kỳ với các hàm chuyển có dạng sigmoid.
- 3) Một mạng có một lớp ẩn có thể xấp xỉ tốt bất kỳ một ánh xạ liên tục nào từ một không gian hữu hạn sang một không gian hữu hạn khác, chỉ cần cung cấp số nơron đủ lớn cho lớp ẩn. Chính xác hơn, các mạng truyền thẳng với một lớp ẩn được luyện bởi các phương pháp bình phương tối thiểu (least-squares) là các bộ xấp xỉ chính xác cho các hàm hồi quy nếu như các giả thiết về mẫu, độ nhiễu, số đơn vị trong lớp ẩn và các nhân tố khác thỏa mãn. Các mạng nơron truyền thẳng với một lớp ẩn sử dụng các hàm chuyển hay hàm phân ngưỡng là các bộ xấp xỉ đa năng cho bài toán phân lớp nhị phân với các giả thiết tương tự.

## 2.3. Vấn đề thiết kế cấu trúc mạng

Mặc dù, về mặt lý thuyết, có tồn tại một mạng có thể mô phỏng một bài toán với độ chính xác bất kỳ. Tuy nhiên, để có thể tìm ra mạng này không phải là điều đơn giản. Để định nghĩa chính xác một kiến trúc mạng như: cần sử dụng bao nhiêu lớp ẩn, mỗi lớp ẩn cần có bao nhiêu đơn vị xử lý cho một bài toán cụ thể là một công việc hết sức khó khăn.

Dưới đây trình bày một số vấn đề cần quan tâm khi ta thiết kế một mạng.

### 2.3.1. Số lớp ẩn

Vì các mạng có hai lớp ẩn có thể thể hiện các hàm với đáng điệu bất kỳ, nên, về lý thuyết, không có lý do nào sử dụng các mạng có nhiều hơn hai lớp ẩn. Người ta đã xác định rằng đối với phần lớn các bài toán cụ thể, chỉ cần sử dụng một lớp ẩn cho mạng là đủ. Các bài toán sử dụng hai lớp ẩn hiếm khi xảy ra trong thực tế. Thậm chí đối với các bài toán cần sử

dùng nhiều hơn một lớp ẩn thì trong phần lớn các trường hợp trong thực tế, sử dụng chỉ một lớp ẩn cho ta hiệu năng tốt hơn là sử dụng nhiều hơn một lớp. Việc huấn luyện mạng thường rất chậm khi mà số lớp ẩn sử dụng càng nhiều. Lý do sau đây giải thích cho việc sử dụng càng ít các lớp ẩn càng tốt là:

- 1) Phần lớn các thuật toán luyện mạng cho các mạng nơron truyền thẳng đều dựa trên phương pháp gradient. Các lớp thêm vào sẽ thêm việc phải lan truyền các lỗi làm cho vector gradient rất không ổn định. Sự thành công của bất kỳ một thuật toán tối ưu theo gradient phụ thuộc vào độ không thay đổi của hướng khi mà các tham số thay đổi.
- 2) Số các cực trị địa phương tăng lên rất lớn khi có nhiều lớp ẩn. Phần lớn các thuật toán tối ưu dựa trên gradient chỉ có thể tìm ra các cực trị địa phương, do vậy chúng có thể không tìm ra cực trị toàn cục. Mặc dù thuật toán luyện mạng có thể tìm ra cực trị toàn cục, nhưng xác suất khá cao là chúng ta sẽ bị tắc trong một cực trị địa phương sau rất nhiều thời gian lặp và khi đó, ta phải bắt đầu lại.
- 3) Dĩ nhiên, có thể đối với một bài toán cụ thể, sử dụng nhiều hơn một lớp ẩn với chỉ một vài đơn vị thì tốt hơn là sử dụng ít lớp ẩn với số đơn vị là lớn, đặc biệt đối với các mạng cần phải học các hàm không liên tục. Về tổng thể, người ta cho rằng việc đầu tiên là nên xem xét khả năng sử dụng mạng chỉ có một lớp ẩn. Nếu dùng một lớp ẩn với một số lượng lớn các đơn vị mà không có hiệu quả thì nên sử dụng thêm một lớp ẩn nữa với một số ít các đơn vị.

### **2.3.2. Số đơn vị trong lớp ẩn**

Một vấn đề quan trọng trong việc thiết kế một mạng là cần có bao nhiêu đơn vị trong mỗi lớp. Sử dụng quá ít đơn vị có thể dẫn đến việc không thể nhận dạng được các tín hiệu đầy đủ trong một tập dữ liệu phức tạp, hay thiếu ăn khớp (*underfitting*). Sử dụng quá nhiều đơn vị sẽ tăng thời gian luyện mạng, có lẽ là quá nhiều để luyện khi mà không thể luyện mạng trong một khoảng thời gian hợp lý. Số lượng lớn các đơn vị có thể dẫn đến tình trạng thừa ăn khớp (*overfitting*), trong trường hợp này mạng có quá nhiều thông tin, hoặc lượng thông tin trong tập dữ liệu mẫu (training set) không đủ các dữ liệu đặc trưng để huấn luyện mạng.

Số lượng tốt nhất của các đơn vị ẩn phụ thuộc vào rất nhiều yếu tố - số đầu vào, đầu ra của mạng, số trường hợp trong tập mẫu, độ nhiễu của dữ liệu đích, độ phức tạp của hàm lỗi, kiến trúc mạng và thuật toán luyện mạng.

Có rất nhiều “luật” để lựa chọn số đơn vị trong các lớp ẩn (xem [6]), chẳng hạn:

- $m \in [l, n]$  - nằm giữa khoảng kích thước lớp vào, lớp ra
- $m = \frac{2(l+n)}{3}$  - 2/3 tổng kích thước lớp vào và lớp ra
- $m < 2l$  - nhỏ hơn hai lần kích thước lớp vào
- $m = \sqrt{l \cdot n}$  - căn bậc hai của tích kích thước lớp vào và lớp ra.

Các luật này chỉ có thể được coi như là các lựa chọn thô khi chọn lựa kích thước của các lớp. Chúng không phản ánh được thực tế, bởi lẽ chúng chỉ xem xét đến nhân tố kích thước đầu vào, đầu ra mà bỏ qua các nhân tố quan trọng khác như: số trường hợp đưa vào huấn luyện, độ nhiễu ở các đầu ra mong muốn, độ phức tạp của hàm lỗi, kiến trúc của mạng (truyền thẳng hay hồi quy), và thuật toán học.

Trong phần lớn các trường hợp, không có một cách để có thể dễ dàng xác định được số tối ưu các đơn vị trong lớp ẩn mà không phải luyện mạng sử dụng số các đơn vị trong lớp ẩn khác nhau và dự báo lỗi tổng quát hóa của từng lựa chọn. Cách tốt nhất là sử dụng phương pháp *thử-sai* (trial-and-error). Trong thực tế, có thể sử dụng phương pháp *Lựa chọn tiến* (forward selection) hay *Lựa chọn lùi* (backward selection) để xác định số đơn vị trong lớp ẩn.

Lựa chọn tiến bắt đầu với việc chọn một luật hợp lý cho việc đánh giá hiệu năng của mạng. Sau đó, ta chọn một số nhỏ các đơn vị ẩn, luyện và thử mạng; ghi lại hiệu năng của mạng. Sau đó, tăng một chút số đơn vị ẩn; luyện và thử lại cho đến khi lỗi là chấp nhận được, hoặc không có tiến triển đáng kể so với trước.

Lựa chọn lùi, ngược với lựa chọn tiến, bắt đầu với một số lớn các đơn vị trong lớp ẩn, sau đó giảm dần đi. Quá trình này rất tốn thời gian nhưng sẽ giúp ta tìm được số lượng đơn vị phù hợp cho lớp ẩn.

#### **2.4. Thuật toán lan truyền ngược (Back-Propagation)**

Cần có một sự phân biệt giữa kiến trúc của một mạng và thuật toán học của nó, các mô tả trong các mục trên mục đích là nhằm làm rõ các yếu tố về kiến trúc của mạng và cách mà mạng tính toán các đầu ra từ tập các đầu vào. Sau đây là mô tả của thuật toán học sử dụng để điều chỉnh hiệu năng của mạng sao cho mạng có khả năng sinh ra được các kết quả mong muốn.

Như đã nêu, về cơ bản có hai dạng thuật toán để luyện mạng: học có thầy và học không có thầy. Các mạng nơron truyền thẳng nhiều lớp được luyện bằng phương pháp học có thầy. Phương pháp này căn bản dựa trên việc yêu cầu mạng thực hiện chức năng của nó và sau đó trả lại kết quả, kết hợp kết quả này với các đầu ra mong muốn để điều chỉnh các tham số của mạng, nghĩa là mạng sẽ học thông qua những sai sót của nó.

Về cơ bản, thuật toán lan truyền ngược là dạng tổng quát của thuật toán trung bình bình phương tối thiểu (Least Means Square-**LMS**). Thuật toán này thuộc dạng thuật toán xấp xỉ để tìm các điểm mà tại đó hiệu năng của mạng là tối ưu. Chỉ số tối ưu (performance index) thường được xác định bởi một hàm số của ma trận trọng số và các đầu vào nào đó mà trong quá trình tìm hiểu bài toán đặt ra.

#### **2.4.1. Mô tả thuật toán**

Ta sẽ sử dụng dạng tổng quát của mạng nơron truyền thẳng nhiều lớp như trong hình vẽ 9 của phần trước. Khi đó, đầu ra của một lớp trở thành đầu vào của lớp kế tiếp. Phương trình thể hiện hoạt động này như sau:

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1}) \text{ với } m = 0, 1, \dots, M-1,$$

trong đó  $M$  là số lớp trong mạng. Các nơron trong lớp thứ nhất nhận các tín hiệu từ bên ngoài:

$$\mathbf{a}^0 = \mathbf{p},$$

chính là điểm bắt đầu của phương trình phía trên. Đầu ra của lớp cuối cùng được xem là đầu ra của mạng:

$$\mathbf{a} = \mathbf{a}^M.$$

##### **2.4.1.1. Chỉ số hiệu năng (performance index)**

Cũng tương tự như thuật toán LMS, thuật toán lan truyền ngược sử dụng chỉ số hiệu năng là trung bình bình phương lỗi của đầu ra so với giá trị đích. Đầu vào của thuật toán chính là tập các cặp mô tả hoạt động đúng của mạng:

$$\{(\mathbf{p}_1, \mathbf{t}_1), (\mathbf{p}_2, \mathbf{t}_2), \dots, (\mathbf{p}_Q, \mathbf{t}_Q)\},$$

trong đó  $\mathbf{p}_i$  là một đầu vào và  $\mathbf{t}_i$  là đầu ra mong muốn tương ứng, với  $i = 1..Q$ . Mỗi đầu vào đưa vào mạng, đầu ra của mạng đối với nó được đem so sánh với đầu ra mong muốn.

Thuật toán sẽ điều chỉnh các tham số của mạng để tối thiểu hóa trung bình bình phương lỗi:

$$F(\mathbf{x}) = E[e^2] = E[(t - a)^2],$$

trong đó  $\mathbf{x}$  là biến được tạo thành bởi các trọng số và độ lệch,  $E$  là ký hiệu kỳ vọng toán học. Nếu như mạng có nhiều đầu ra, ta có thể viết lại phương trình trên ở dạng ma trận:

$$F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})].$$

Tương tự như thuật toán LMS, xấp xỉ của trung bình bình phương lỗi như sau:

ký hiệu  $\hat{F}(\mathbf{x})$  là giá trị xấp xỉ của  $F(\mathbf{x})$  thì:

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k)) = \mathbf{e}^T(k) \mathbf{e}(k),$$

trong đó kỳ vọng toán học của bình phương lỗi được thay bởi bình phương lỗi tại bước  $k$ .

Thuật toán giảm theo hướng cho trung bình bình phương lỗi xấp xỉ là:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m}, \quad (+)$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m}, \quad (++)$$

trong đó  $\alpha$  là hệ số học.

Như vậy, mọi chuyện đến đây đều giống như thuật toán trung bình bình phương tối thiểu. Tiếp theo chúng ta sẽ đi vào phần khó nhất của thuật toán: tính các đạo hàm từng phần.

#### 2.4.1.2. Luật xích (Chain Rule)

Đối với các mạng nơron truyền thẳng nhiều lớp, lỗi không phải là một hàm của chỉ các trọng số trong các lớp ẩn, do vậy việc tính các đạo hàm từng phần này là không đơn giản. Chính vì lý do đó mà ta phải sử dụng luật xích để tính. Luật này được mô tả như sau: giả sử ta có một hàm  $f$  là một hàm của biến  $n$ , ta muốn tính đạo hàm của  $f$  có liên quan đến một biến  $w$  khác. Luật xích này như sau:

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \cdot \frac{dn(w)}{dw}$$

Ta sẽ dùng phương pháp này để tính các đạo hàm trong (+) và (++) ở phần trước.

$$\frac{\partial \hat{F}}{w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m},$$
$$\frac{\partial \hat{F}}{b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial b_i^m},$$

trong đó hạng thức thứ hai của các phương trình trên có thể dễ dàng tính toán bởi vì đầu vào của mạng tới lớp  $m$  là một hàm của trọng số và độ lệch:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m.$$

trong đó  $S^{m-1}$  là số đầu ra của lớp  $(m-1)$ . Do vậy ta có:

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \quad \frac{\partial n_i^m}{\partial b_i^m} = 1.$$

Ký hiệu

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m}$$

là *độ nhạy cảm* của  $\hat{F}$  đối với các thay đổi của phần tử thứ  $i$  của đầu vào của mạng tại lớp thứ  $m$ . Khi đó ta có:

$$\frac{\partial \hat{F}}{w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial w_{i,j}^m} = s_i^m a_j^{m-1},$$
$$\frac{\partial \hat{F}}{b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \cdot \frac{\partial n_i^m}{\partial b_i^m} = s_i^m.$$

Bây giờ, ta có thể phát biểu thuật toán giảm theo hướng (gradient descent) như sau:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1},$$
$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m$$

Ở dạng ma trận:



$$\begin{aligned}\mathbf{W}^m(k+1) &= \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T, \\ \mathbf{b}^m(k+1) &= \mathbf{b}^m(k) - \alpha \mathbf{s}^m\end{aligned}$$

trong đó:

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}$$

#### 2.4.1.3. Lan truyền ngược độ nhạy cảm

Bây giờ ta cần tính nốt ma trận độ nhạy cảm  $\mathbf{s}^m$ . Để thực hiện điều này cần sử dụng một áp dụng khác của luật xích. Quá trình này cho ta khái niệm về sự “lan truyền ngược” bởi vì nó mô tả mối quan hệ hồi quy trong đó độ nhạy cảm  $\mathbf{s}^m$  được tính qua độ nhạy cảm  $\mathbf{s}^{m+1}$  của lớp  $m+1$ .

Để dẫn đến quan hệ đó, ta sử dụng ma trận *Jacobi* sau:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix}$$

Xét phần tử  $(i, j)$  của ma trận trên:

$$\begin{aligned} \frac{\partial n_i^{m+1}}{\partial n_j^m} &= \frac{\partial \left( \sum_{l=1}^{S^m} w_{i,l}^{m+1} a_i^m + b_i^{m+1} \right)}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial a_i^m}{\partial n_j^m} \\ &= w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m(n_j^m) \end{aligned}$$

trong đó:

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}.$$

Như vậy, ma trận *Jacobi* có thể viết lại như sau:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \dot{\mathbf{F}}^m(\mathbf{n}^m),$$

trong đó:

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}^m(n_2^m) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dot{f}^m(n_{s^m}^m) \end{bmatrix}.$$

Bây giờ ta viết lại quan hệ hồi quy cho độ nhạy cảm dưới dạng ma trận:

$$\begin{aligned} \mathbf{s}^m &= \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left( \frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m) (\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} \\ &= \dot{\mathbf{F}}^m(\mathbf{n}^m) (\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}. \end{aligned}$$

Đến đây có thể thấy độ nhạy cảm được lan truyền ngược qua mạng từ lớp cuối cùng trở về lớp đầu tiên:

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \dots \rightarrow \mathbf{s}^1.$$

Cần nhấn mạnh rằng ở đây thuật toán lan truyền ngược lỗi sử dụng cùng một kỹ thuật giảm theo hướng như thuật toán LMS. Sự phức tạp duy nhất là ở chỗ để tính gradient ta cần phải lan truyền ngược độ nhạy cảm từ các lớp sau về các lớp trước như đã nêu trên.

Bây giờ ta cần biết điểm bắt đầu lan truyền ngược, xét độ nhạy cảm  $\mathbf{s}^M$  tại lớp cuối cùng:

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial \left( (t-a)^T (t-a) \right)}{\partial n_i^M} = \frac{\partial \sum_{l=1}^{S^M} (t_l - a_l)^2}{\partial n_i^M} = -2(t_i - a_i) \frac{\partial a_i}{\partial n_i^M}.$$

Bởi vì:

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M),$$

nên ta có thể viết:


$$s_i^M = -2(t_i - a_i) \dot{f}^M(n_i^M).$$

Ở dạng ma trận sẽ là:

$$\mathbf{s}^M = -2 \dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}).$$

Tóm lại, thuật toán lan truyền ngược có thể phát biểu như sau:

**THUẬT TOÁN LAN TRUYỀN NGƯỢC - BACK-PROPAGATION**



**Bước 1:** Lan truyền xuôi đầu vào qua mạng:

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1}), \text{ với } m = 0, 1, \dots, M-1.$$

$$\mathbf{a} = \mathbf{a}^M$$

**Bước 2:** Lan truyền độ nhạy cảm (lỗi) ngược lại qua mạng:

$$\mathbf{s}^M = -2 \dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}, \text{ với } m = M-1, \dots, 2, 1.$$

**Bước 3:** Cuối cùng, các trọng số và độ lệch được cập nhật bởi công thức sau:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$

#### 2.4.2. Sử dụng thuật toán lan truyền ngược

Trên đây là thuật toán lan truyền ngược cơ bản, sau đây ta sẽ bàn về các khía cạnh ứng dụng của thuật toán lan truyền ngược như chọn lựa cấu trúc mạng, sự hội tụ và khả năng tổng quát hóa.

#### 2.4.2.1. Chọn lựa cấu trúc mạng

Như ta đã biết, thuật toán lan truyền ngược có thể được sử dụng để xấp xỉ bất kỳ một hàm số học nào nếu như ta có đủ số nơron trong các lớp ẩn. Mặc dù vậy, phát biểu trên chưa cho ta được một số cụ thể các lớp và số nơron trong mỗi lớp cần sử dụng. Ta sẽ dùng một ví dụ để có được cái nhìn chi tiết hơn về vấn đề này.

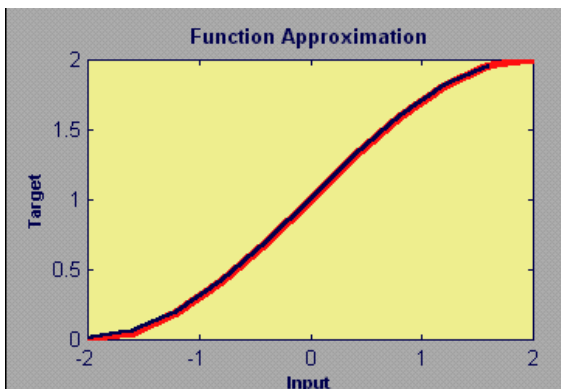
Ví dụ, ta muốn xấp xỉ hàm số sau:

$$f(x) = 1 + \sin\left(\frac{i\pi}{4}x\right) \text{ với } -2 \leq x \leq 2,$$

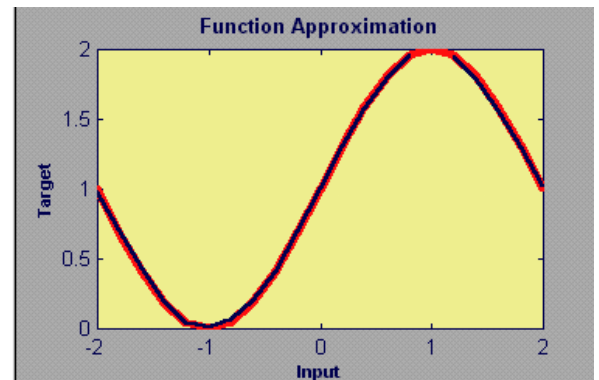
trong đó  $i$  nhận các giá trị 1, 2, 4 và 8. Khi  $i$  tăng thì hàm số cần xét sẽ trở nên phức tạp hơn do ta sẽ nhận được nhiều chu kỳ của hình sin trong phạm vi  $[-2, 2]$ . Khi đó, mạng nơron với một số nơron cố định sẽ khó có thể xấp xỉ được hàm nếu  $i$  tăng. Trong phần sau, ta sẽ sử dụng ví dụ Function Approximation trong thư viện của bộ Matlab 5.3 (file `nnd11fa.m`). Ở đây, mạng sử dụng có 1 lớp ẩn, 1 lớp ra; đương nhiên, mạng có 1 đầu vào và một đầu ra. Lớp ẩn sử dụng hàm sigmoid, lớp ra dùng hàm đồng nhất:

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

Số nơron trong lớp ẩn là 4, kết quả xấp xỉ của mạng trong các trường hợp  $i = 1, 2, 4, 8$  như trong các hình vẽ dưới đây. Các đường màu đen là trả lời của mạng, còn các đường mờ hơn là hàm cần xấp xỉ.

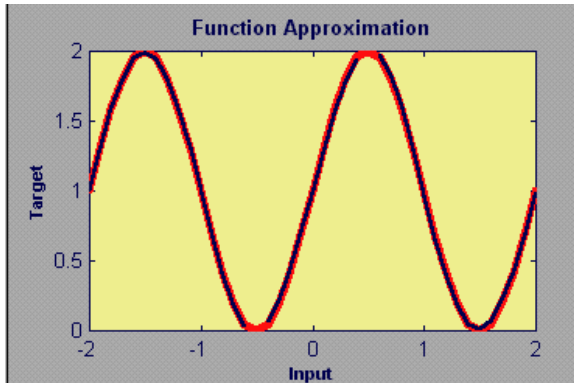


a) trường hợp  $i = 1$

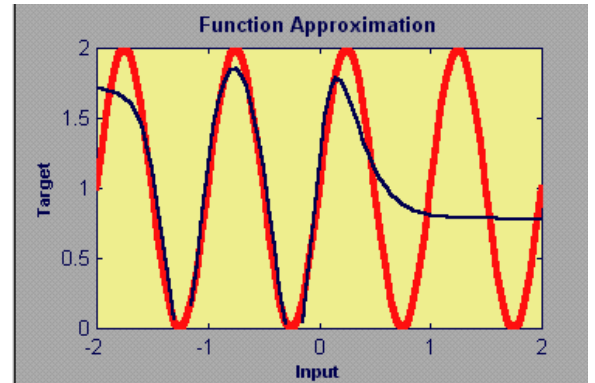


b) trường hợp  $i = 2$

Hình 11: Xấp xỉ hàm  $f(x) = 1 + \sin\left(\frac{i\pi}{4}x\right)$  với  $-2 \leq x \leq 2$ .



c) trường hợp  $i = 4$

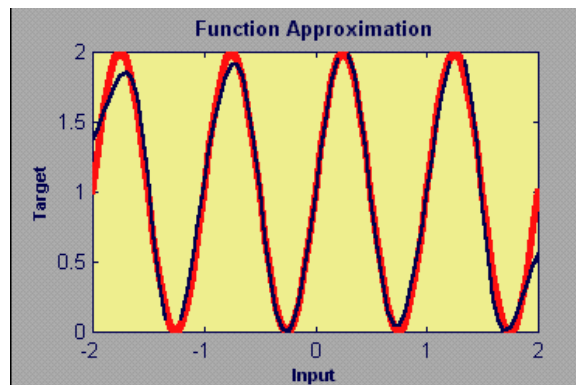


d) trường hợp  $i = 8$

Hình 11: Xấp xỉ hàm  $f(x) = 1 + \sin\left(\frac{i\pi}{4}x\right)$  với  $-2 \leq x \leq 2$ .

Khi ta tăng số nơron trong lớp ẩn lên thì khả năng xấp xỉ hàm số của mạng sẽ tốt hơn.

Chẳng hạn, xét trường hợp sử dụng 9 nơron trong lớp ẩn và  $i = 8$  ta có được kết quả sau:



trường hợp  $i = 8$  và số nơron lớp ẩn = 9

Hình 12: Xấp xỉ hàm  $f(x) = 1 + \sin\left(\frac{i\pi}{4}x\right)$  với  $-2 \leq x \leq 2$  khi tăng số nơron.

Điều đó có nghĩa là nếu ta muốn xấp xỉ một hàm số mà có số điểm cần xấp xỉ là lớn thì ta sẽ cần số nơron lớn hơn trong lớp ẩn.

#### 2.4.2.2. Sự hội tụ

Trong phần trên ta đã thấy các trường hợp mạng nơron không trả lại kết quả chính xác mặc dù thuật toán lan truyền ngược đã thực hiện tối thiểu hóa trung bình bình phương lỗi. Điều

đó là do khả năng của mạng bị giới hạn bởi số nơron trong lớp ẩn. Tuy nhiên, cũng có trường hợp mà thuật toán lan truyền ngược không cho ta các tham số có thể dẫn đến kết quả chính xác nhưng mạng vẫn có thể xấp xỉ được hàm số. Điều này xảy ra là do trạng thái khởi đầu của mạng, sau khi luyện, mạng có thể rơi vào điểm cực tiểu toàn cục hoặc rơi vào điểm cực tiểu địa phương.

Cần chú ý rằng trong thuật toán trung bình bình phương tối thiểu, điểm cực trị toàn cục là luôn tồn tại bởi lẽ hàm trung bình bình phương lỗi của thuật toán trung bình bình phương tối thiểu là một hàm bậc hai, hơn nữa, do là hàm bậc hai nên đạo hàm bậc hai của hàm lỗi sẽ là hằng số, do vậy mà độ cong của hàm theo một hướng cho trước là không thay đổi. Trong khi đó, thuật toán lan truyền ngược áp dụng cho các mạng nhiều lớp sử dụng các hàm chuyển phi tuyến sẽ có nhiều điểm cực trị địa phương và độ cong của hàm lỗi có thể không cố định theo một hướng cho trước.

#### 2.4.2.3. Sự tổng quát hóa (Generalization):

Trong phần lớn các trường hợp, mạng nơron truyền thẳng nhiều lớp được luyện bởi một số cố định các mẫu xác định sự hoạt động đúng của mạng:

$$\{(\mathbf{p}_1, \mathbf{t}_1), (\mathbf{p}_2, \mathbf{t}_2), \dots, (\mathbf{p}_Q, \mathbf{t}_Q)\},$$

trong đó,  $\mathbf{p}_i$  là các đầu vào, tương ứng với nó là các đầu ra mong muốn  $\mathbf{t}_i$ . Tập huấn luyện này thông thường là thể hiện của số lớn nhất các lớp có thể các cặp. Một điều rất quan trọng là mạng nơron có khả năng tổng quát hóa được từ những cái nó đã học. Nếu có được điều đó, mặc dù dữ liệu có nhiều thì mạng vẫn có khả năng hoạt động tốt (trả lại kết quả gần với đích mong muốn).

“Để một mạng có khả năng tổng quát hóa tốt, nó cần có số tham số ít hơn số dữ liệu có trong tập huấn luyện” ([4]). Trong các mạng nơron, cũng như các bài toán mô hình hóa, ta thường mong muốn sử dụng một mạng đơn giản nhất có thể cho kết quả tốt trên tập huấn luyện.

Một cách khác đó là dùng luyện mạng trước khi mạng xảy ra tình trạng thừa ăn khớp. Kỹ thuật này liên quan đến việc chia tập dữ liệu thu được thành ba tập: tập huấn luyện sử dụng để tính toán gradient và cập nhật các trọng số của mạng, tập kiểm định được dùng để kiểm tra điều kiện dừng của mạng và tập kiểm tra được sử dụng để so sánh khả năng tổng quát hóa của mạng đối với các bộ tham số của mạng sau các lần huấn luyện.

### 2.4.3. Một số biến thể của thuật toán lan truyền ngược

Ta đã xem xét một số đặc điểm của thuật toán lan truyền ngược sử dụng kỹ thuật giảm theo hướng. Mạng sử dụng thuật toán này tồn tại nhược điểm: rơi vào điểm cực tiểu địa phương đối với mạng nơron truyền thẳng nhiều lớp sử dụng các hàm chuyển phi tuyến. Hơn nữa, khi thực hiện luyện mạng bằng cách đưa từng mẫu vào, sau đó thực hiện cập nhật tham số, sẽ làm ảnh hưởng đến quá trình học các mẫu khác. Do đó, một phương pháp để tăng tốc độ hội tụ là sử dụng phương pháp học cả gói (batch training), nghĩa là tất cả các mẫu được đưa vào mạng, sau đó mới thực hiện cập nhật các tham số. Bây giờ ta sẽ xem xét một số biến thể của thuật toán lan truyền ngược sử dụng phương pháp học cả gói nhằm vượt qua các nhược điểm này.

#### 2.4.3.1. Sử dụng tham số bước đà (Momentum)

Đây là một phương pháp heuristic dựa trên quan sát kết quả luyện mạng nhằm làm tăng tốc độ hội tụ của thuật toán lan truyền ngược dựa trên kỹ thuật giảm nhanh nhất. Thuật toán lan truyền ngược cập nhật các tham số của mạng bằng cách cộng thêm vào một lượng thay đổi là:

$$\Delta \mathbf{W}^m(k) = -\alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T,$$

$$\Delta \mathbf{b}^m(k) = -\alpha \mathbf{s}^m.$$

Khi áp dụng thuật toán lan truyền ngược có sử dụng bước đà, phương trình trên thay đổi như sau:

$$\Delta \mathbf{W}^m(k) = \gamma \Delta \mathbf{W}^m(k-1) - (1-\gamma) \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T,$$

$$\Delta \mathbf{b}^m(k) = \gamma \Delta \mathbf{b}^m(k-1) - (1-\gamma) \alpha \mathbf{s}^m.$$

Người ta đã chứng tỏ rằng khi sử dụng tham số bước đà thì hệ số học có thể lớn hơn rất nhiều so với thuật toán lan truyền ngược chuẩn không sử dụng tham số bước đà trong khi vẫn giữ được độ tin cậy của thuật toán. Một điểm khác nữa là khi sử dụng tham số bước đà thì sự hội tụ của thuật toán sẽ được tăng tốc nếu như thuật toán đang đi theo một hướng bền vững (chỉ đi xuống trong một khoảng dài).

#### 2.4.3.2. Sử dụng hệ số học biến đổi:

Trong thực tế, các hàm hiệu năng có dạng biểu diễn hình học là không đồng đều, có lúc có dạng phẳng (hàm không thay đổi giá trị hoặc thay đổi rất ít) hoặc có dạng phễu (giá trị của hàm thay đổi rất nhanh khi thay đổi tham số đầu vào). Nếu ta chỉ sử dụng hệ số học cố định thì có thể sẽ tốn thời gian tại các vùng phẳng. Vì vậy, tư tưởng của thuật toán lan truyền ngược sử dụng hệ số học biến đổi là khi gặp vùng phẳng thì tăng hệ số học lên và ngược lại khi gặp vùng dạng phễu thì giảm hệ số học đi.

Người ta đã đưa ra rất nhiều phương pháp để thực hiện điều trên, ở đây chỉ nêu ra một cách biến đổi hệ số học dựa trên hiệu năng của mạng (có thể tham khảo ở [9]).

**Bước 1:** Nếu bình phương lỗi trên toàn bộ tập huấn luyện tăng một số phần trăm cho trước  $\xi$  (thông thường là từ 1% cho đến 5%) sau một lần cập nhật trọng số, thì bỏ qua việc cập nhật này, hệ số học được nhân với một số hạng  $\rho$  nào đó (với  $0 < \rho < 1$ ) và tham số bước đà (nếu có sử dụng) được đặt bằng 0.

**Bước 2:** Nếu bình phương lỗi giảm sau một lần cập nhật trọng số, thì cập nhật đó là chấp nhận được và hệ số học được nhân với một số hạng nào đó  $> 1$ , nếu tham số bước đà đã bị đặt bằng 0 thì đặt lại giá trị lúc đầu.

**Bước 3:** Nếu bình phương lỗi tăng một lượng  $< \xi$ , thì cập nhật trọng số là chấp nhận được, nhưng hệ số học không thay đổi và nếu tham số bước đà đã bị đặt bằng 0 thì đặt lại giá trị lúc đầu.

Các thuật toán heuristic luôn cho ta sự hội tụ nhanh hơn trong một số bài toán, tuy nhiên chúng có hai nhược điểm chính sau đây:

*Thứ nhất*, việc sửa đổi thuật toán lan truyền ngược cần có thêm một số tham số, trong khi trong thuật toán lan truyền ngược chuẩn chỉ yêu cầu có một tham số đó là hệ số học. Một số thuật toán sửa đổi cần đến năm hoặc sáu tham số, trong khi hiệu năng của thuật toán khá nhạy cảm đối với những thay đổi của các tham số này. Hơn nữa việc chọn lựa các tham số lại độc lập với bài toán đặt ra.

*Thứ hai*, các thuật toán sửa đổi có thể không hội tụ trong một số bài toán mà thuật toán lan truyền ngược chuẩn có thể hội tụ được.



Người ta đã thấy rằng cả hai nhược điểm nêu trên thường xảy ra khi sử dụng các thuật toán sửa đổi phức tạp hơn (yêu cầu nhiều tham số hơn).

#### 2.4.3.3. Sử dụng phương pháp Gradient kết hợp:

Nhắc lại phương pháp gradient kết hợp bao gồm các bước sau:

1. Chọn hướng bắt đầu ngược với gradient ( $\mathbf{p}_0 = -\mathbf{g}_0$ ).
2. Thực hiện một bước ( $\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$ ). Chọn  $\alpha_k$  để tối thiểu hàm theo hướng tìm kiếm đã chọn. Có thể chọn như sau:

$$\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$$

(phương trình trên được suy ra bằng cách chọn  $\alpha_k$  để tối thiểu hóa hàm  $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$ . Để thực hiện, ta lấy đạo hàm của  $F$  theo  $\alpha_k$ , đặt bằng 0 ta sẽ thu được phương trình trên.)

3. Chọn hướng tiếp theo dựa vào một trong ba phương trình tính  $\beta_k$ .

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \Delta \mathbf{g}_k}{\Delta \mathbf{g}_{k-1}^T \mathbf{p}_{k-1}} \text{ hoặc } \beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \text{ hoặc } \beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}.$$

4. Nếu thuật toán chưa hội tụ thì quay lại bước 2.

Phương pháp này không thể áp dụng trực tiếp trong việc luyện mạng nơron, bởi lẽ hàm chỉ số hiệu năng trong nhiều trường hợp không ở dạng bậc hai. Điều này ảnh hưởng đến thuật toán này như sau:

Thứ nhất, ta không thể sử dụng phương trình:

$$\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k}$$

để tối thiểu hóa hàm theo đường thẳng ( $\mathbf{x}_k + \alpha_k \mathbf{p}_k$ ) như trong bước thứ 2.

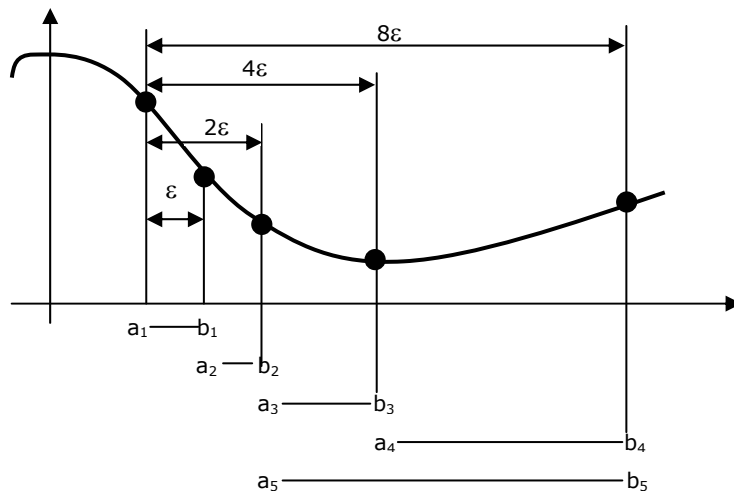
Thứ hai, điểm cực tiểu chính xác sẽ không thể đạt tới được một cách bình thường sau một số hữu hạn bước và do vậy thuật toán sẽ phải được thiết lập lại sau một số hữu hạn bước.

### Vấn đề tìm kiếm điểm cực tiểu:

Ta cần có một thuật toán tổng quát để tìm điểm cực tiểu của một hàm số theo một hướng cụ thể nào đó. Việc này liên quan đến hai thao tác: một là xác định tần số (interval location) và giảm tần số. Mục đích của bước xác định tần số là tìm kiếm tần số khởi đầu có chứa điểm cực tiểu. Bước giảm tần số sau đó giảm kích thước của tần số cho đến khi tìm ra điểm cực tiểu với một độ chính xác nào đó.

Ta sẽ sử dụng phương pháp so sánh hàm để thực hiện bước xác định tần số. Thủ tục này được mô tả trong hình vẽ 13. Ta bắt đầu bằng cách tính chỉ số hiệu năng tại một điểm khởi đầu nào đó (điểm  $a_1$  trong hình vẽ), điểm này chính là giá trị của chỉ số hiệu năng với các tham số hiện tại của mạng.

Bước tiếp theo là tính giá trị hàm chỉ số hiệu năng tại điểm thứ 2, thể hiện bởi điểm  $b_1$  trong hình vẽ cách điểm khởi đầu một đoạn là  $\varepsilon$  theo hướng tìm kiếm  $p_0$ .

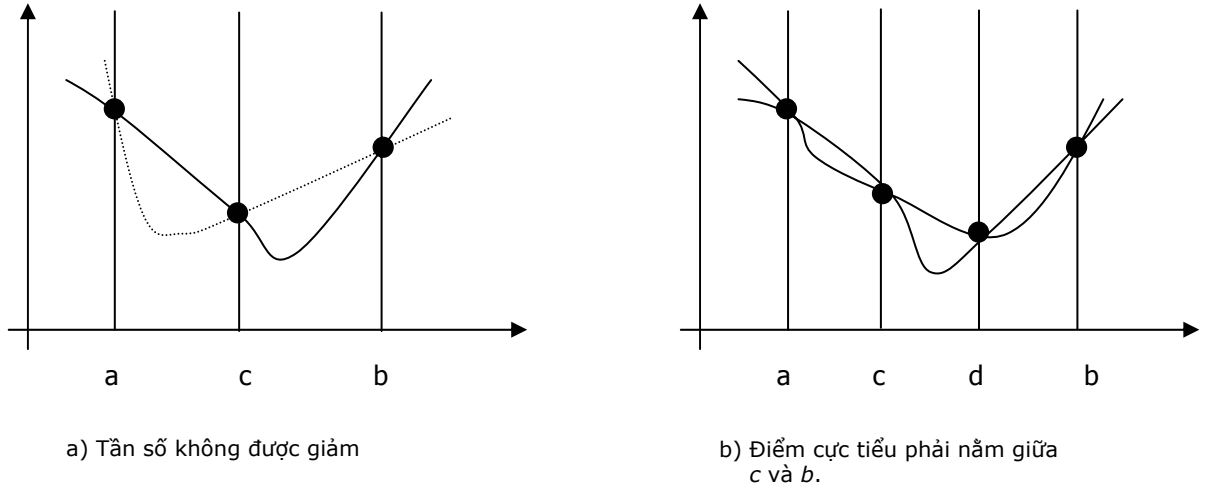


Hình 13: Xác định tần số.

Sau đó, ta tiếp tục tính giá trị của hàm hiệu năng tại các điểm  $b_i$  có khoảng cách đến điểm khởi đầu gấp đôi điểm trước. Quá trình này sẽ dừng lại nếu như giá trị của hàm tăng lên so với điểm trước đó (trong hình vẽ là điểm  $b_3$  và  $b_4$ ). Đến đây, ta biết rằng điểm cực tiểu sẽ rơi vào khoảng giữa  $[a_5, b_5]$ . Ta không thể thu hẹp thêm tần số nữa bởi lẽ điểm cực tiểu có thể rơi vào vùng  $[a_3, b_3]$  hoặc  $[a_4, b_4]$ .

Bây giờ ta tiếp tục bước thực hiện giảm tần số, ta sẽ lấy ít nhất là hai điểm  $c, d$  trong khoảng  $[a_5, b_5]$  để có thể thực hiện việc này (nếu chỉ lấy 1 điểm thì ta khó có thể xác định được liệu điểm cực tiểu sẽ nằm trong vùng nào!). Có nhiều cách để chọn các điểm trên, ở

đây ta sẽ sử dụng phương pháp gọi là: *Golden Section search*. Phương pháp này cố gắng thực hiện tìm kiếm sao cho số lần phải tính giá trị của hàm là ít nhất (tại mỗi một bước ta đều cần phải tính giá trị của hàm).



Hình 14: Giảm kích thước của tần số không chắc chắn.

Trong hình vẽ trên, điểm  $a$  sẽ được bỏ qua và điểm  $c$  sẽ trở thành cận bên trái. Sau đó, một điểm  $c$  mới sẽ được đặt vào vùng của điểm  $c$  và  $d$  cũ. Mục đích ở đây là chọn điểm  $c$  sao cho tần số của sự không chắc chắn sẽ được giảm đi càng nhanh càng tốt.

Thuật toán *Golden Section search* như sau:

**Golden Section search**

$\tau = 0.618$

Đặt  $c_1 = a_1 + (1 - \tau)(b_1 - a_1)$ ,  $F_c = F(c_1)$ .

$d_1 = b_1 - (1 - \tau)(b_1 - a_1)$ ,  $F_d = F(d_1)$ .

Với  $k = 1, 2, \dots$ , lặp lại các bước sau:

Nếu  $F_c < F_d$  thì:

Đặt  $a_{k+1} = a_k$ ;  $b_{k+1} = d_k$ ;  $d_{k+1} = c_k$ ;

$c_{k+1} = a_{k+1} + (1 - \tau)(b_{k+1} - a_{k+1})$

$F_d = F_d$ ;  $F_c = F(c_{k+1})$

Ngược lại

Đặt  $a_{k+1} = c_k$ ;  $b_{k+1} = b_k$ ;  $c_{k+1} = d_k$ ;

$d_{k+1} = b_{k+1} - (1 - \tau)(b_{k+1} - a_{k+1})$

$F_c = F_c$ ;  $F_d = F(d_{k+1})$

Kết thúc chừng nào  $(b_{k+1} - a_{k+1}) < tol$

Trong đó  $tol$  là độ chính xác chấp nhận được do người sử dụng đưa vào.

### **Thiết lập lại thuật toán**

Thuật toán gradient kết hợp còn cần phải sửa đổi thêm một chút nữa trước khi áp dụng để luyện cho mạng nơron. Đối với hàm bậc 2, thuật toán sẽ hội tụ đến điểm cực tiểu sau nhiều nhất  $n$  bước, trong đó  $n$  là số các tham số cần tối thiểu hóa. Chỉ số hiệu năng trung bình bình phương lỗi của mạng nơron truyền thẳng nhiều lớp không phải ở dạng bậc 2, do vậy thuật toán sẽ không hội tụ sau  $n$  bước lặp. Những phát triển ở phía trên không chỉ ra hướng tìm kiếm tiếp theo sau khi  $n$  bước lặp hoàn thành. Có nhiều cách để thực hiện, nhưng ta chỉ cần áp dụng một cách đơn giản nhất đó là đặt lại hướng tìm kiếm trở lại hướng ban đầu của thuật toán giảm nhanh nhất sau khi  $n$  bước lặp đã hoàn thành.

#### **2.4.4. Nhận xét**

Thuật ngữ “lan truyền ngược” được sử dụng có vẻ như không phù hợp lắm đối với thuật ngữ truyền thẳng và thường gây hiểu nhầm. Lan truyền ngược thực chất là một kỹ thuật toán học sử dụng để tính toán lỗi trong các hệ thống toán học phức tạp, chẳng hạn như một mạng nơron. Nó là một trong các thuật toán gradient tương tự như là các thuật toán theo gradient theo các cách tiếp cận của Trí tuệ nhân tạo. Các thuật toán đó ánh xạ hàm vào bề mặt ba chiều, với các mặt lồi, lõm. Phụ thuộc vào bài toán cụ thể, điểm lõm (cực tiểu) của một bề mặt thể hiện hiệu năng tốt hơn cho đầu ra.

Việc luyện mạng theo phương pháp học có thầy liên quan đến cách thức đưa các mẫu học từ miền của bài toán vào mạng, các mẫu này sẽ được phân chia thành các tập huấn luyện và tập kiểm định. Mạng được khởi tạo các trọng số là các số ngẫu nhiên, sau đó, các trọng số này sẽ được điều chỉnh cho phù hợp với tập huấn luyện. Tập kiểm định sẽ được dùng để xác định xem liệu mạng có thành công trong việc xác định đầu ra từ đầu vào mà nó chưa được luyện. Mạng sẽ được đưa vào một tập con các mẫu, mỗi mẫu một lần, sau khi nó đã được “nhìn” tất cả các mẫu, nó sẽ phải thực hiện điều chỉnh các trọng số bằng cách tính toán các lỗi xảy ra. Quá trình này được lặp lại cho đến khi mạng được luyện đủ. Kích thước của tập con được giới hạn bởi số lần lặp, có thể là trùng với kích thước của tập mẫu học, nếu không như vậy thì cần phải xác định thứ tự đưa các mẫu vào cho mạng học một cách ngẫu nhiên.

Giá trị của lỗi được tính bởi phương pháp trung bình bình phương của giá trị kích hoạt; nghĩa là nó được tính bằng cách bình phương hiệu của giá trị đầu ra mong muốn và đầu ra thực sự, sau đó tính trung bình trong tất cả các neuron đầu ra. Có thể xác định cách điều chỉnh các trọng số để có thể giảm được lỗi bằng cách tính các đạo hàm từng phần (đạo hàm theo hướng) của lỗi. Số các bước cần thực hiện theo hướng đó được gọi là mức độ học (tham số học-learning rate), nếu quá lớn, giá trị cực trị có thể bị bỏ qua, nếu quá nhỏ thì phải mất nhiều thời gian để đạt tới điểm cực trị.

Nhược điểm lớn nhất của thuật toán lan truyền ngược truyền thống đó là nó bị ảnh hưởng rất lớn của gradient địa phương, không cần thiết phải đi đường thẳng. Ví dụ, nếu như cực trị toàn cục nằm ở cuối vùng lõm và điểm hiện tại là bên cạnh, phía trên điểm lõm, khi đó thuật toán lan truyền ngược sẽ thực hiện một bước theo hướng mà gradient lớn nhất, vượt qua vùng lõm. Một khi nó phát hiện các cạnh khác của của vùng lõm, nó sẽ chạy theo đường zig zắc tiến, lùi tạo ra các bước nhỏ tới đích. Đường này sẽ lớn gấp hàng nghìn lần so với đường ngắn nhất, và do đó, thời gian học cũng sẽ lớn gấp rất nhiều lần. Thuật toán lan truyền ngược chuẩn có thể được tăng cường bằng cách thêm tham số bước đà (momentum) vào phương trình. Hiệu ứng này sẽ lọc ra ngoài các cực trị địa phương và cho phép khả năng tìm ra cực trị toàn cục lớn lên.

Khoảng bước, hay mức độ học, của thuật toán lan truyền ngược chuẩn là cố định, điều này dẫn đến việc thuật toán tìm xung quanh điểm cực tiểu trong khi đó, thuật toán không thể tìm chính xác điểm thấp nhất trong hai gradient. Nghĩa là nó đi xuống một bước, vượt qua điểm cực tiểu và đứng ở nửa trên phía bên kia. Phương pháp gradient kết hợp (Conjugate Gradient) cho phép thuật toán học thực hiện các bước nhỏ tăng dần khi nó tiếp cận điểm cực tiểu, như vậy, nó có thể đạt tới điểm gần với điểm cực tiểu thực sự rất nhanh chóng.

Mặc dù phương pháp tối ưu gradient giảm (gradient descent) dùng trong thuật toán lan truyền ngược chuẩn được sử dụng rộng rãi và được thực tế chứng minh là thành công trong rất nhiều ứng dụng, nó cũng còn tồn tại các nhược điểm:

- 1) Hội tụ rất chậm
- 2) Không đảm bảo là sẽ hội tụ tại điểm cực trị toàn cục

Rất nhiều các nhà nghiên cứu [3][9][11][12][20] đã đưa ra các cải tiến cho phương pháp gradient như là: sửa đổi động các tham số học hay điều chỉnh độ dốc của hàm sigmoid,...

Trong các hoàn cảnh thích hợp, các phương pháp tối ưu khác có thể là tốt hơn thuật toán gradient. Nhiều thuật toán hội tụ nhanh hơn là thuật toán gradient trong một số trường hợp trong khi một số khác hứa hẹn xác suất hội tụ đến điểm cực trị toàn cục lớn hơn[20].

Một trong số các phương pháp tối ưu có thể thay thế cho phương pháp gradient đó là Phương pháp gradient kết hợp (Conjugate Gradient), đó là phương pháp cực tiểu theo hướng. Tối thiểu hóa theo một hướng  $d$  đặt hàm  $E$  tới chỗ mà gradient của nó là vuông góc với  $d$ . Thay vì theo hướng gradient tại từng bước, một tập gồm  $n$  hướng được xây dựng theo cách kết hợp với các hướng khác, tối thiểu hóa theo một trong số các hướng làm hỏng giá trị tối thiểu hóa theo một trong các hướng trước đó.

Phương pháp Gradient sử dụng đạo hàm bậc hai (Ma trận Hessian), như trong phương pháp *Newton*, có thể rất hiệu quả trong một số trường hợp. Nếu sử dụng đạo hàm bậc nhất, các phương pháp đó sử dụng một xấp xỉ tuyến tính địa phương của bề mặt lỗi (error surface), Các phương pháp bậc hai, sử dụng xấp xỉ bậc hai. Do các phương pháp như vậy đều sử dụng thông tin đạo hàm bậc nhất và bậc hai theo đúng công thức, các thuộc tính hội tụ địa phương là rất tốt. Tuy vậy, chúng đều không thực tế bởi lẽ việc tính toán bộ ma trận Hessian có thể là rất tốn kém trong các bài toán có phạm vi rộng.

## **2.5. Các thuật toán tối ưu khác**

Cực trị địa phương có thể xảy ra trong trường hợp mạng không được huấn luyện một cách tối ưu, trong nhiều trường hợp, các cực trị này là chấp nhận được. Nếu ngược lại, mạng cần được huấn luyện lại cho đến khi hiệu năng tốt nhất có thể được tìm ra. Mặc dù vậy, có các kỹ thuật đã được thiết kế nhằm làm tăng hiệu quả của quá trình học của mạng, trong đó bao gồm Thuật toán giả luyện kim hoặc thuật giải di truyền (Masters 1993). Các phương pháp này có thể giúp vượt qua được cực trị địa phương đã được ứng dụng thành công trong một số vấn đề.

### **2.5.1. Thuật toán giả luyện kim (*Simulated annealing*)**

Kỹ thuật này là một quá trình luyện kim, trong đó sự sắp xếp ngẫu nhiên của các phân tử cacbon trong thép được chuyển đổi thành một kim loại có cấu trúc lớp ít giòn hơn. Quá trình này bao gồm việc nung kim loại ở một nhiệt độ rất cao và sau đó làm lạnh từ từ. Các phân tử ở nhiệt độ cao có mức năng lượng cao, là cho các phân tử này chuyển động. Khi mà nhiệt độ giảm đi, các chuyển động cũng giảm đi và chúng được sắp xếp thành các lớp.

Ý tưởng này được áp dụng vào các thuật toán huấn luyện cho mạng nơron. Nhiệt độ được coi như là hệ số học được giảm dần. Ý tưởng ở đây là nếu mạng gặp phải một điểm cực trị địa phương thì nó sẽ được “rung” (shake) để có thể thoát khỏi cực trị địa phương. Nếu như “nhiệt độ” được giữ không đổi thì hệ sẽ chỉ chuyển từ một điểm cực trị địa phương này sang một điểm cực trị địa phương khác và khó có thể ổn định. Nếu khả năng nhảy được giảm đều thì mạng sẽ có xu hướng đạt đến được điểm cực trị toàn cục. Và một khi mạng đạt đến điểm cực trị toàn cục thì mức rung sẽ không đủ để có thể khiến cho mạng bỏ qua nó.

Rõ ràng thuật toán giả luyện kim có dáng dấp của một thuật toán huấn luyện với hệ số học biến đổi, tuy nhiên, hệ số học trong thuật toán này được giảm dần trong khi, thuật toán huấn luyện sử dụng hệ số học biến đổi sẽ làm tăng hay giảm hệ số học tùy thuộc vào tình huống cụ thể khi sai số khi học là tăng hay giảm.

### 2.5.2. Thuật giải di truyền (Genetic Algorithm)

Đây thực chất là một thuật toán tìm kiếm điểm tối ưu trong không gian của các tham số. Thuật toán di truyền là kỹ thuật bắt chước sự chọn lọc tự nhiên và di truyền. Trong tự nhiên, các cá thể khỏe, có khả năng thích nghi tốt với môi trường sẽ được tái sinh và nhân bản trong các thế hệ sau.

Trong giải thuật di truyền, mỗi cá thể được mã hóa bởi một cấu trúc dữ liệu mô tả cấu trúc gen của mỗi cá thể đó, gọi là *nhịễm sắc thể*. Mỗi nhiễm sắc thể được tạo thành từ các đơn vị gọi là *gen*. Chẳng hạn như là một chuỗi nhị phân, tức là mỗi cá thể được biểu diễn bởi một chuỗi nhị phân.

Giải thuật di truyền sẽ làm việc trên các quần thể gồm nhiều cá thể. Một quần thể ứng với một giai đoạn phát triển được gọi là một *thế hệ*. Từ thế hệ đầu được tạo ra, giải thuật di truyền bắt chước chọn lọc tự nhiên và di truyền để biến đổi các thế hệ. Giải thuật di truyền sử dụng các toán tử: **tái sinh (reproduction)**: các cá thể tốt được đưa vào thế hệ sau dựa vào độ thích nghi đối với môi trường của mỗi cá thể (xác định bởi **hàm thích nghi-fitness function**); **toán tử lai ghép (crossover)**: hai cá thể cha, mẹ trao đổi các gen để tạo ra hai cá thể con; **toán tử đột biến (mutation)**: một cá thể thay đổi một số gen để tạo thành cá thể mới. Việc áp dụng các toán tử trên đối với các quần thể là ngẫu nhiên.

Thuật toán di truyền bắt đầu bằng việc khởi tạo quần thể ban đầu, sau đó thực hiện lặp lại các bước: sinh ra thế hệ mới từ thế hệ ban đầu bằng cách áp dụng các toán tử lai ghép, đột biến, tái sinh; đánh giá thế hệ mới sinh ra; cho đến khi điều kiện kết thúc được thỏa mãn. Khi thuật toán dừng, cá thể tốt nhất được lựa chọn làm nghiệm cần tìm.

Có thể thấy, thuật toán di truyền có liên quan đến kỹ thuật tìm kiếm điểm tối ưu. Thực chất, nó có thể coi như là một kỹ thuật khác để huấn luyện mạng nơron để giải quyết các bài toán. Nó liên quan đến việc mã hóa các tham số của mạng nơron bằng các nhiễm sắc thể. Các tham số ban đầu được khởi tạo ngẫu nhiên nhiều lần tạo ra quần thể ban đầu. Khi đó, hàm thích nghi của các cá thể (tập các trọng số) được xác định bằng cách tính toán lỗi đầu ra của mạng. Nếu điều kiện dừng thỏa mãn thì quá trình huấn luyện dừng lại, nếu không, sẽ thực hiện các toán tử chọn lọc, lai ghép, đột biến trên các cá thể để tạo ra quần thể mới [1] [20]. Các nghiên cứu cho thấy rằng thuật toán di truyền có thể được xem như một thuật toán tốt dùng để huấn luyện mạng.



## CHƯƠNG III. ỨNG DỤNG MẠNG NƠON TRUYỀN THẮNG TRONG DỰ BÁO DỮ LIỆU

Chương này đề cập các vấn đề sau:

- 3.1. Sơ lược về lĩnh vực dự báo dữ liệu
- 3.2. Thu thập, phân tích và xử lý dữ liệu
- 3.3. Chương trình dự báo dữ liệu
- 3.4. Một số nhận xét

---

### 3.1. Sơ lược về lĩnh vực dự báo dữ liệu

Người ta đã chứng tỏ rằng không có một phương pháp luận hoàn hảo trong tiếp cận các bài toán bằng cách sử dụng mạng nơon huấn luyện bởi thuật toán lan truyền ngược. Ta có nhiều điều cần cân nhắc, lựa chọn để có thể thiết lập các tham số cho một mạng nơon:

- ⇒ Số lớp ẩn
- ⇒ Kích thước các lớp ẩn
- ⇒ Hằng số học (beta)
- ⇒ Tham số momentum (alpha)
- ⇒ Khoảng, khuôn dạng dữ liệu sẽ đưa vào mạng
- ⇒ Dạng hàm squashing (không nhất thiết phải là hàm sigmoid)
- ⇒ Điểm khởi đầu (ma trận trọng số ban đầu)
- ⇒ Tỷ lệ nhiễu mẫu (tăng khả năng tổng quát hóa cho mạng).

Việc dự báo dữ liệu là một bài toán rất phức tạp, cả về số lượng dữ liệu cần quan tâm cũng như độ chính xác của dữ liệu dự báo. Do vậy, việc cân nhắc để có thể chọn được mô hình phù hợp cho việc dự báo dữ liệu là một việc rất khó khăn (chỉ có thể bằng phương pháp thử-sai). Tuy nhiên, thuật toán lan truyền ngược là thuật toán được ứng dụng rất rộng rãi

trong các lĩnh vực: nhận dạng, phân lớp, dự báo... đã được thực tế chứng tỏ là một công cụ tốt áp dụng cho các bài toán trong lĩnh vực dự báo dữ liệu.

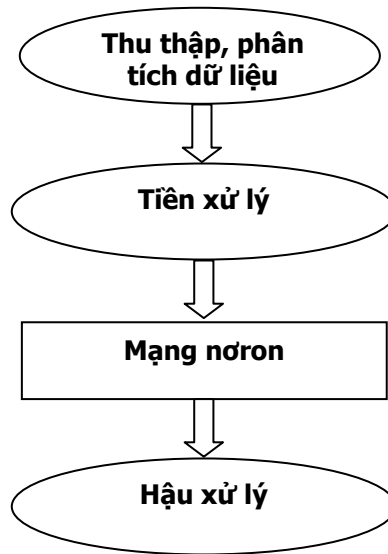
Do đặc trưng về độ phức tạp dữ liệu, các dữ liệu đầu ra thường là các con số (mảng các số) đầu phẩy động cho nên việc lựa chọn cấu trúc mạng phù hợp thường là sử dụng phương pháp thử-sai (trial and errors). Đồng thời cần phải chuẩn hóa (loại bỏ các dữ liệu sai, thừa, đưa chúng về đoạn  $[0,1]$  hoặc  $[-1,1]$ ,...) các dữ liệu đầu vào và đầu ra để mạng có khả năng học tốt hơn từ các dữ liệu được cung cấp.

Trong việc dự báo dữ liệu, nếu dữ liệu ở nhiều khoảng thời gian khác nhau được đưa vào mạng để huấn luyện thì việc dự báo chính xác là rất khó nếu như mục đích là dự báo chính xác 100% dữ liệu trong tương lai. Ta chỉ có thể có được kết quả dự báo với một mức độ chính xác nào đó chấp nhận được.

### **3.2. Thu thập, phân tích và xử lý dữ liệu**

Dữ liệu đóng một vai trò rất quan trọng trong các giải pháp sử dụng mạng nơron. Chất lượng, độ tin cậy, tính sẵn có và phù hợp của dữ liệu được sử dụng để phát triển hệ thống giúp cho các giải pháp thành công. Các mô hình đơn giản cũng có thể đạt được những kết quả nhất định nếu như dữ liệu được xử lý tốt, bộc lộ được các thông tin quan trọng. Bên cạnh đó, các mô hình tốt có thể sẽ không cho ta các kết quả mong muốn nếu dữ liệu đưa vào quá phức tạp và rắc rối.

Việc xử lý dữ liệu bắt đầu bằng việc thu thập và phân tích dữ liệu, sau đó là bước tiền xử lý. Dữ liệu sau khi qua bước tiền xử lý được đưa vào mạng nơron. Cuối cùng, dữ liệu đầu ra của mạng nơron qua bước hậu xử lý, bước này sẽ thực hiện biến đổi kết quả trả về của mạng nơron sang dạng hiểu được theo yêu cầu của bài toán (Hình 15). Sau đây, trong các mục tiếp theo, ta sẽ đi vào xem xét từng bước trong quá trình xử lý dữ liệu.



Hình 15: Xử lý dữ liệu

### 3.2.1. Kiểu của các biến

Các biến dữ liệu có thể được chia thành hai loại dựa trên các đặc điểm, tính chất của chúng (Có thể tham khảo ở [2][5][6][10][13][14]):

#### 3.2.1.1. Biến phân loại (Categorical Variables)

Các biến này thường không có thứ tự xác định, nghĩa là giữa chúng không xác định được các phép toán như: “lớn hơn” hay “nhỏ hơn”. Các biến này nằm trong các giá trị đưa vào không có giá trị số nhưng được gán các giá trị số trong đầu vào. Ví dụ, biến “kiểu màu”, có thể nhận các giá trị “đỏ”, ”xanh”, và “vàng” là một biến phân loại. Giới tính cũng là biến kiểu này. Các dữ liệu số cũng có thể thuộc loại này, ví dụ như: “mã vùng”, “mã nước”.

Các biến thuộc loại này có thể được đưa vào mạng bằng sơ đồ mã hóa *1-of-c* (*1-of-c* encoding scheme), sơ đồ này mã hóa các giá trị của biến thành các xâu nhị phân có chiều dài bằng số các giá trị mà biến có thể nhận trong phạm vi bài toán. Một bit sẽ được bật lên tùy theo giá trị của biến, các bit còn lại sẽ được đặt bằng 0. Trong ví dụ trên, biến “kiểu màu” cần ba biến vào, tương ứng với ba màu được thể hiện bằng các xâu nhị phân: (1,0,0), (0,1,0) and (0,0,1).

Một cách khác để mã hóa các biến phân loại là thể hiện tất cả các giá trị có thể vào một biến đầu vào liên tục. Ví dụ, các giá trị “đỏ”, ”xanh”, và “vàng” có thể được thể hiện bởi

các giá trị số 0.0, 0.5, và 1.0. Điểm không tốt của phương pháp này là nó tạo ra một trật tự nhân tạo trên dữ liệu mà trên thực tế, thứ tự này không hề có. Nhưng đối với các biến với một số lượng lớn các phân loại, phương pháp này có thể giảm rất nhiều số đơn vị đầu vào.

#### **3.2.1.2. Biến có thứ tự (Ordinal Variables)**

Các biến này có xác định thứ tự tự nhiên. Chúng có thể được chuyển trực tiếp thành các giá trị tương ứng của một biến liên tục với một tỷ lệ nào đó.

### **3.2.2. Thu thập dữ liệu**

Bước thực hiện thu thập các dữ liệu bao gồm ba nhiệm vụ chính:

#### **3.2.2.1. Xác định yêu cầu dữ liệu**

Điều đầu tiên cần thực hiện khi lập kế hoạch thu thập dữ liệu ta là xác định xem các dữ liệu nào là cần thiết để có thể giải quyết bài toán. Về tổng thể, có thể cần sự trợ giúp của các chuyên gia trong lĩnh vực của bài toán cần giải quyết. Ta cần phải biết: a) Các dữ liệu chắc chắn có liên quan đến bài toán; b) Các dữ liệu nào có thể liên quan; c) Các dữ liệu nào là phụ trợ. Các dữ liệu có liên quan và có thể liên quan đến bài toán cần phải được xem là các đầu vào cho hệ thống.

#### **3.2.2.2. Xác định nguồn dữ liệu**

Bước kế tiếp là quyết định nơi sẽ lấy dữ liệu, điều này cho phép ta xác định được các ước lượng thực tế về những khó khăn và phí tổn cho việc thu thập dữ liệu. Nếu ứng dụng yêu cầu các dữ liệu thời gian thực, những ước lượng này cần tính đến khả năng chuyển đổi các dữ liệu tương tự thành dạng số.

Trong một số trường hợp, ta có thể chọn lựa dữ liệu mô phỏng từ các tình huống thực tế. Tuy nhiên, cần phải quan tâm đến độ chính xác và khả năng thể hiện của dữ liệu đối với các trường hợp cụ thể.

#### **3.2.2.3. Xác định lượng dữ liệu**

Ta cần phải ước đoán số lượng dữ liệu cần thiết để có thể sử dụng trong việc xây dựng mạng. Nếu lấy quá ít dữ liệu thì những dữ liệu này sẽ không thể phản ánh toàn bộ các thuộc tính mà mạng cần phải học và do đó mạng sẽ không có được phản ứng mong đợi đối với những dữ liệu mà nó chưa được huấn luyện. Mặt khác, cũng không nên đưa vào huấn luyện

cho mạng quá nhiều dữ liệu. Về tổng thể, lượng dữ liệu cần thiết bị chi phối bởi số các trường hợp cần luyện cho mạng. Bản chất đa chiều của dữ liệu và cách giải quyết mong muốn là các nhân tố chính xác định số các trường hợp cần luyện cho mạng và kéo theo là lượng dữ liệu cần thiết.

Việc định lượng gần đúng lượng dữ liệu cần đưa vào luyện mạng là hết sức cần thiết. Thông thường, dữ liệu thường thiếu hoàn chỉnh, do đó nếu muốn mạng có khả năng thực hiện được những điều mà ta mong đợi thì nó cần phải được luyện với lượng dữ liệu lớn hơn. Đương nhiên, nếu có được độ chính xác và hoàn chỉnh của dữ liệu thì số các trường hợp cần thiết phải đưa vào mạng có thể giảm đi.

### ***3.2.3. Phân tích dữ liệu***

Có hai kỹ thuật cơ bản giúp ta có thể hiểu được dữ liệu:

#### ***3.2.3.1. Phân tích thống kê***

Mạng nơron có thể được xem như là một mở rộng của các phương pháp thống kê chuẩn. Các thử nghiệm có thể cho ta biết được khả năng mà mạng có thể thực hiện. Hơn nữa, phân tích có thể cho ta các đầu mối để xác định các đặc trưng, ví dụ, nếu dữ liệu được chia thành các lớp, các thử nghiệm thống kê có thể xác định được khả năng phân biệt các lớp trong dữ liệu thô hoặc dữ liệu đã qua tiền xử lý.

#### ***3.2.3.2. Trực quan hóa dữ liệu***

Trực quan hóa dữ liệu bằng cách vẽ biểu đồ trên các dữ liệu theo một dạng thích hợp sẽ cho ta thấy được các đặc trưng phân biệt của dữ liệu, chẳng hạn như: các điểm lệch hay các điểm đỉnh. Điều này nếu thực hiện được, có thể áp dụng thêm các thao tác tiền xử lý để tăng cường các đặc trưng đó.

Thông thường, phân tích dữ liệu bao gồm cả các kiểm tra thống kê và trực quan hóa. Các kiểm tra này sẽ được lặp đi lặp lại. Trực quan hóa cho ta sự đánh giá về dữ liệu và các khái niệm sơ khởi về các mẫu nằm sau dữ liệu. Trong khi các phương pháp thống kê cho phép ta kiểm thử những khái niệm này.

### 3.2.4. Xử lý dữ liệu

#### 3.2.4.1. Dẫn nhập về xử lý dữ liệu

Khi những dữ liệu thô đã được thu thập, chúng cần phải được chuyển đổi sang các khuôn dạng phù hợp để có thể đưa vào luyện mạng. Ở bước này, ta cần thực hiện các công việc sau:

##### *Kiểm tra tính hợp lệ dữ liệu (Data validity checks)*

Việc kiểm tra tính hợp lệ sẽ phát hiện ra các dữ liệu không thể chấp nhận được mà nếu sử dụng chúng thì sẽ cho ra các kết quả không tốt. Ví dụ, ta có thể kiểm tra khoảng hợp lệ của dữ liệu về nhiệt độ không khí của một vùng nhiệt đới chẳng hạn. Ta mong muốn các giá trị trong khoảng từ 5°C đến 40°C, do đó, các giá trị nằm ngoài khoảng này rõ ràng là không thể chấp nhận được.

Nếu có một mẫu cho một phân bố sai của dữ liệu (ví dụ, nếu phần lớn dữ liệu được thu thập ở một ngày trong tuần) ta cần xem xét nguyên nhân của nó. Dựa trên bản chất của nguyên nhân dẫn đến sai lầm, ta có thể hoặc phải loại bỏ các dữ liệu này, hoặc cho phép những thiếu sót đó. Nếu có các thành phần quyết định không mong muốn như là các xu hướng hay các biến thiên có tính chất mùa vụ, chúng cần được loại bỏ ngay.

##### *Phân hoạch dữ liệu (Partitioning data)*

Phân hoạch là quá trình chia dữ liệu thành các tập kiểm định, huấn luyện, và kiểm tra. Theo định nghĩa, tập *kiểm định* được sử dụng để xác định kiến trúc của mạng; các tập *huấn luyện* được dùng để cập nhật các trọng số của mạng; các tập *kiểm tra* được dùng để kiểm tra hiệu năng của mạng sau khi luyện. Ta cần phải đảm bảo rằng:

- a) Tập huấn luyện chứa đủ dữ liệu, các dữ liệu đó phân bố phù hợp sao cho có thể biểu diễn các thuộc tính mà ta muốn mạng sẽ học được.
- b) Không có dữ liệu trùng nhau hay tương tự nhau của các dữ liệu trong các tập dữ liệu khác nhau.

#### 3.2.4.2. Tiền xử lý

Về mặt lý thuyết, một mạng nơron có thể dùng để ánh xạ các dữ liệu thô đầu vào trực tiếp thành các dữ liệu đầu ra. Nhưng trong thực tế, việc sử dụng quá trình tiền xử lý cho dữ liệu

thường mang lại những hiệu quả nhất định trước khi những dữ liệu này được đưa vào mạng. Có rất nhiều kỹ thuật liên quan đến tiền xử lý dữ liệu. Tiền xử lý dữ liệu có thể là thực hiện lọc dữ liệu (trong dữ liệu biến thiên theo thời gian time-series) hay các phương pháp phức tạp hơn như là các phương pháp kết xuất, trích chọn các đặc trưng từ dữ liệu ảnh tĩnh (image data). Bởi lẽ việc chọn thuật toán dùng trong tiền xử lý dữ liệu là phụ thuộc vào ứng dụng và bản chất của dữ liệu, cho nên, các khả năng lựa chọn là rất lớn. Tuy nhiên, mục đích của các thuật toán tiền xử lý dữ liệu thường tương tự nhau, như sau (Xem chẳng hạn [6]):

1) Chuyển đổi dữ liệu về khuôn dạng phù hợp đối với đầu vào mạng nơron - điều này thường đơn giản hóa quá trình xử lý của mạng phải thực hiện trong thời gian ngắn hơn. Các chuyển đổi này có thể bao gồm:

- Áp dụng một hàm toán học (hàm logarit hay bình phương) cho đầu vào;
- Mã hóa các dữ liệu văn bản trong cơ sở dữ liệu;
- Chuyển đổi dữ liệu sao cho nó có giá trị nằm trong khoảng  $[0, 1]$ .
- Lấy biến đổi Fourier cho các dữ liệu thời gian.

2) Lựa chọn các dữ liệu xác đáng nhất - việc lựa chọn này có thể bao gồm các thao tác đơn giản như lọc hay lấy tổ hợp của các đầu vào để tối ưu hóa nội dung của dữ liệu. Điều này đặc biệt quan trọng khi mà dữ liệu có nhiều hoặc chứa các thông tin thừa. Việc lựa chọn cẩn thận các dữ liệu phù hợp sẽ làm cho mạng dễ xây dựng và tăng cường hiệu năng của chúng đối với các dữ liệu nhiễu.

3) Tối thiểu hóa số các đầu vào mạng - giảm số chiều của dữ liệu đầu vào và tối thiểu số các mẫu đưa vào có thể đơn giản hóa được bài toán. Trong một số trường hợp - chẳng hạn trong xử lý ảnh - ta không thể nào đưa tất cả các dữ liệu vào mạng. Ví dụ như trong ứng dụng nhận dạng ảnh, mỗi một ảnh có thể chứa hàng triệu điểm ảnh, khi đó rõ ràng là không khả thi nếu sử dụng nhiều đầu vào như vậy. Trong trường hợp này, việc tiền xử lý cần thực hiện giảm số đầu vào của dữ liệu bằng cách sử dụng các tham số đơn giản hơn chẳng hạn như sử dụng các tham số vùng ảnh và tỷ lệ chiều dài/chiều cao. Quá trình này còn gọi là trích chọn dấu hiệu (*feature extraction*) [14].

#### *3.2.4.3. Hậu xử lý*

Hậu xử lý bao gồm các xử lý áp dụng cho đầu ra của mạng. Cũng như đối với tiền xử lý, hậu xử lý hoàn toàn phụ thuộc vào các ứng dụng cụ thể và có thể bao gồm cả việc phát hiện các tham số có giá trị vượt quá khoảng cho phép hoặc sử dụng đầu ra của mạng như một đầu vào của một hệ khác, chẳng hạn như một bộ xử lý dựa trên luật. Đôi khi, hậu xử lý chỉ đơn giản là quá trình ngược lại đối với quá trình tiền xử lý.

#### **3.2.5. Tổng hợp**

Trong thực tế khi xây dựng các mạng nơron ứng dụng trong lĩnh vực dự báo dữ liệu, việc áp dụng các phương pháp tiền xử lý dữ liệu đầu vào (và sau đó áp dụng phương pháp hậu xử lý để biến đổi đầu ra về dạng phù hợp) giúp ích rất nhiều trong các ứng dụng. Như đã nêu ở trên, có rất nhiều các phương pháp có thể áp dụng cho dữ liệu ở quá trình tiền xử lý cũng như hậu xử lý. Các phương pháp này thực sự hiệu quả cho các bài toán cụ thể bởi lẽ chúng làm giảm bớt đi độ phức tạp của dữ liệu đầu vào, từ đó làm giảm thời gian học của mạng nơron.

Các phương pháp xử lý dữ liệu còn phụ thuộc vào công việc thu thập, phân tích và lựa chọn dữ liệu đầu vào cho mạng. Đây cũng là yếu tố quyết định cho sự thành công của các ứng dụng mạng nơron. Việc dữ liệu được chuẩn hóa trước khi đưa vào mạng huấn luyện có thể làm giảm bớt thời gian mạng học, làm tăng độ chính xác cho dữ liệu dự báo. Điều này rất có ý nghĩa bởi lẽ thuật toán lan truyền ngược khi thực thi rất tốn thời gian!

### **3.3. Chương trình dự báo dữ liệu**

#### **3.3.1. Các bước chính trong quá trình thiết kế và xây dựng**

Trước hết, dưới đây nêu ra các bước chính trong quá trình thiết kế và xây dựng một ứng dụng dựa trên mạng nơron. Có rất nhiều vấn đề cần phải xem xét khi xây dựng mạng nơron nhiều lớp sử dụng thuật toán lan truyền ngược:

##### *Tiền xử lý dữ liệu*

Tần số của dữ liệu: hàng ngày, hàng tuần, hàng tháng hay hàng quý.

Kiểu dữ liệu: các chỉ số kỹ thuật hay các chỉ số căn bản.



Cách thức chuẩn hóa dữ liệu: max/min hay Trung bình/Độ lệch chuẩn (standard deviation).

### *Huấn luyện*

Hệ số học.

Hệ số bước đà.

Hệ số thứ lỗi.

Số chu kỳ tối đa.

Hệ số học tối đa.

Thực hiện lấy ngẫu nhiên trọng số.

Kích thước của các tập huấn luyện, kiểm tra, và kiểm định.

### *Cấu trúc mạng (topology)*

Số đầu vào.

Số lớp ẩn.

Số nơron trong các lớp.

Số nơron đầu ra.

Hàm chuyển cho các nơron.

Hàm lỗi.

Dưới đây là các bước chính cần thực hiện khi thiết kế mô hình mạng nơron sử dụng cho bài toán dự báo:

- i) Chọn lựa các biến.
- ii) Thu thập dữ liệu.
- iii) Tiền xử lý dữ liệu.
- iv) Phân chia tập dữ liệu thành các tập: huấn luyện, kiểm tra, kiểm định.
- v) Xác định cấu trúc mạng:
  - số lớp ẩn.

- số neuron trong các lớp ẩn.
  - số neuron đầu ra.
  - các hàm chuyển.
- vi) Xác định tiêu chuẩn đánh giá (hàm lỗi)
- vii) Huấn luyện mạng.
- viii) Thực thi trong thực tế.

Trong khi thực hiện, không nhất thiết phải theo thứ tự các bước mà có thể quay lại các bước trước đó, đặc biệt là ở bước huấn luyện và lựa chọn các biến.

#### Bước 1: Chọn lựa các biến

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.

#### Bước 2: Thu thập dữ liệu

Cần xem xét khả năng có thể thu thập được các dữ liệu. Các dữ liệu kỹ thuật có thể thu thập được dễ dàng hơn là các dữ liệu cơ bản. Mặt khác, các dữ liệu sau khi thu thập cần được kiểm tra tính hợp lệ của chúng. Đồng thời, các dữ liệu bị thiếu sót cần được xử lý cẩn thận, có thể bỏ qua chúng hoặc giả sử rằng các dữ liệu bị thiếu đó không thay đổi so với dữ liệu trước nó.

#### Bước 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. Các đầu vào, đầu ra của mạng neuron hiếm khi được đưa trực tiếp vào mạng. Chúng thường được chuẩn hóa vào khoảng giữa cận trên và cận dưới của hàm chuyển (thường là giữa đoạn  $[0;1]$  hoặc  $[-1;1]$ ).

Các phương pháp phổ biến có thể là:

$$SV = ((0.9 - 0.1) / (MAX\_VAL - MIN\_VAL)) * (OV - MIN\_VAL)$$

hoặc đưa về khoảng giữa giá trị min và max:

$$SV = TFmin + ((TFmax - TFmin) / (MAX\_VAL - MIN\_VAL)) * (OV - MIN\_VAL)$$

trong đó:

SV: Giá trị sau khi biến đổi

MAX\_VAL: Giá trị lớn nhất của dữ liệu

MIN\_VAL: Giá trị nhỏ nhất của dữ liệu

TFmax: Giá trị lớn nhất của hàm chuyển

TFmin: Giá trị nhỏ nhất của hàm chuyển

OV: Giá trị ban đầu

#### Bước 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.

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.

#### Bước 5: Xác định cấu trúc mạng

Phương pháp thực hiện xây dựng mạng nơron bao gồm việc xác định sự liên kết giữa các nơron, đồng thời xác định cấu trúc của mạng bao gồm số lớp ẩn, số nơron trong từng lớp.

Tuy nhiên, các thực nghiệm cho thấy rằng, số lớp ẩn sử dụng trong mạng không nên vượt quá 4 lớp. Ngoài ra, không có phương pháp nào có thể chọn được số tối ưu các nơon sử dụng trong lớp ẩn. Mặc dù vậy cũng có một số phương pháp cho ta lựa chọn ban đầu. Nhưng để có được số tối ưu các nơon trong các lớp ẩn thì người phát triển mô hình cần phải thực hiện nhiều thí nghiệm để có được nó. Bên cạnh đó, việc chọn lựa số các đầu vào mạng cũng mang một tính chất quyết định đến cấu trúc của mạng để có được khả năng tổng quát hóa tốt.

Ta có thể thực hiện lựa chọn số nơon 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 nơon.

Bất kể phương pháp nào thì luật tổng quát nhất là thực hiện chọn cấu trúc mạng cho ta lỗi tổng quát hóa trên tập dữ liệu huấn luyện là nhỏ nhất. Khi thực hiện điều chỉnh, nên giữ các tham số còn lại không thay đổi để tránh tạo ra các cấu trúc khác có khả năng đưa lại các phức tạp không cần thiết trong quá trình lựa chọn số tối ưu các nơon trong lớp ẩn.

#### Bước 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 nhỏ nhất (least absolute deviation), hiệu phần trăm (percentage differences), bình phương nhỏ nhất bất đối xứng (asymmetric least squares),... Tuy nhiên, các hàm này có thể không phải là hàm đánh giá chất lượng cuối cùng cho mạng. Phương pháp đánh giá các giá trị dự báo hay được sử dụng là giá trị trung bình tuyệt đối phần trăm lỗi (mean absolute percentage error - MAPE).

Chẳng hạn trong các hệ thống bán hàng, các giá trị dự báo của mạng nơon sẽ được chuyển sang tín hiệu mua hoặc bán tùy thuộc vào một tiêu chuẩn xác định trước đó.

#### Bước 7: Huấn luyện mạng

Huấn luyện mạng học các 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.

Vấn đề đặt ra là khi nào thì ngừng huấn luyện. Có hai quan điểm trong vấn đề này. Quan điểm thứ nhất cho rằng chỉ nên ngừng huấn luyện chừng nào không có tiến triển nào của

hàm lỗi nữa đối với dữ liệu dựa trên một số tập các tham số của mạng được chọn ngẫu nhiên. Nói cách khác là xác định được khả năng đạt đến được điểm cực tiểu toàn cục lớn nhất. Trường phái thứ hai cho rằng cần thực hiện xem xét thường xuyên khả năng tổng quát hóa của mạng bằng cách sau một số chu kỳ nào đó thực hiện kiểm tra và kiểm tra sự tổng quát hóa của mạng, sau đó lại tiếp tục quá trình huấn luyện.

Cả hai quan điểm này đều thống nhất rằng kết quả kiểm tra trên tập kiểm định là chính xác nhất bởi lẽ nó thể hiện trực tiếp kết quả trả lời của mạng sau khi được huấn luyện.

Việc thực hiện huấn luyện mạng còn cần phải xem xét khả năng của mạng nơron với một số nào đó lần thực hiện huấn luyện mạng trên các tập khởi tạo ban đầu của các tham số. Sau khi thực hiện huấn luyện trên tất cả các tham số này cần thực hiện đánh giá lại kết quả, từ đó đưa ra kết luận về số lần tối đa thực hiện huấn luyện cho mạng cho từng bài toán cụ thể của mình.

Một phương pháp khác là thực hiện vẽ đồ thị để có thể theo dõi trạng thái lỗi của mạng, từ đó có thể quan sát được các vùng mà mạng có trạng thái không thay đổi đối với dữ liệu vào. Thông thường, số lần tối đa thực hiện huấn luyện cho mạng thường có khoảng biến thiên khá lớn: từ vài nghìn cho đến vài chục, vài trăm nghìn chu kỳ, việc theo dõi được trạng thái của mạng đối với tập huấn luyện và khả năng tổng quát hóa để có thể ngừng khi cần là khá quan trọng. Có thể thực hiện cập nhật đồ thị sau mỗi chu kỳ để có thể theo dõi được các tham số này.

#### Bước 8: Thực thi

Bước thực thi thực ra cần được xem xét trước cả bước thu thập dữ liệu. Bởi lẽ, việc xác định khả năng sẵn có của dữ liệu, xác định hàm lỗi sử dụng và thời gian huấn luyện đều là những đặc trưng của môi trường mà mạng sẽ được triển khai. Người ta thấy rằng, do mạng nơron có đặc trưng tính toán song song, do vậy mạng nơron tốt nhất nên được thực hiện cài đặt trên các vi mạch điện tử. Tuy nhiên, môi trường máy tính cá nhân lại phù hợp hơn trong quá trình huấn luyện, dễ cài đặt, đồng thời có khả năng linh hoạt đáp ứng được nhiều bài toán hơn.

Sau khi cài đặt, triển khai, khả năng hoạt động của mạng nơron sẽ giảm đi theo thời gian nếu như không có bước thực hiện huấn luyện lại, bởi lẽ không thể đảm bảo được rằng các tham biến được lựa chọn sẽ luôn đóng vai trò quyết định đối với các kết quả mà ta mong

muốn theo thời gian. Tần số thực hiện huấn luyện lại mạng cần hợp lý sao cho mạng có thể đạt được trạng thái hoạt động tốt nhất.

### **3.3.2. Xây dựng chương trình**

Về tổng thể, chương trình dự báo dữ liệu được xây dựng dựa trên các cơ sở lý thuyết đã nêu trên. Mạng sử dụng trong bài toán dự báo dữ liệu là mạng truyền thẳng nhiều lớp, được huấn luyện bởi thuật toán lan truyền ngược sửa đổi (có sử dụng tham số bước đà) để tăng khả năng tổng quát hóa và thời gian hội tụ. Về tổng thể, các mạng nơron truyền thẳng nhiều lớp được huấn luyện bởi thuật toán lan truyền ngược cần có khả năng linh hoạt đáp ứng được nhiều bài toán. (Chú ý rằng điều này có thể thực hiện được bằng cách xây dựng cấu trúc chương trình phù hợp). Điều quan trọng là xác định được các biến chi phối trong bài toán, khả năng sẵn có của dữ liệu (hàng ngày, hàng tháng hay quý, năm),...

Ở đây nêu ra một ví dụ của bài toán dự báo dữ liệu: **Bài toán dự báo khả năng sử dụng khí ga**

“Trong ngành công nghiệp ga, việc dự báo khả năng sử dụng hàng ngày hay hàng giờ là rất cần thiết đối với các công ty, giúp họ tối ưu được sự phân phối phục vụ của họ đối với khách hàng. Đối với các công ty đường ống, việc dự báo khả năng tiêu thụ có thể giúp xác định các ảnh hưởng đến hoạt động của hệ thống đường ống, từ đó có thể đáp ứng được nhu cầu và dự báo khả năng tiêu thụ trong tương lai. Nó cũng có thể giúp họ tìm ra cách tốt nhất để tối thiểu hóa chi phí điều hành, đáp ứng được nhu cầu. Một quyết định cần phải tăng thêm hay rút bớt lượng ga để có thể phù hợp với yêu cầu phải được đưa ra bất kể tình trạng lưu trữ hiện tại. Một lý do khác là lượng ga chảy trong hệ thống là không được xác định chính xác. Nói một cách khác, khách hàng có quyền để lại một lượng ga lưu trữ tại nhà mà không phải thông báo. Do vậy, khả năng này cũng cần phải được xem xét.”

Rõ ràng là từ các nguyên nhân trên, cần phải xây dựng một hệ dự báo tin cậy dựa trên các yếu tố lập kế hoạch hoạt động.

Dự đoán khả năng sử dụng ga mượn ý tưởng từ bài toán dự báo lượng tiêu thụ điện, bài toán đã áp dụng mạng nơron thành công cho việc dự báo lượng tiêu thụ trong 1 cho đến 24 giờ (Xem chẳng hạn [6][18]). Việc dự báo lượng tiêu thụ ga có một sự tương tự nhất định đối với các bài toán khác như: điện, nước, đồng thời cũng có những đặc điểm riêng: nó chứa đựng các dự báo cho các khoảng thời gian trùng với các chu kỳ kế lập hoạch cho việc

điều hành và quản lý hệ thống cung cấp ga. Thường là dự báo cho từ ba đến năm ngày sau. Ta sẽ bắt đầu xây dựng hệ thống này coi như một case study cho việc phát triển các hệ thống dự báo dữ liệu.

#### *3.3.2.1. Các yếu tố ảnh hưởng*

Phần khó nhất trong việc xây dựng mô hình là xác định và thu thập được các dữ liệu huấn luyện và kiểm tra. Người ta đã chỉ ra các yếu tố ảnh hưởng đến nhu cầu sử dụng ga như sau (Xem chẳng hạn [2][10][18]):

- ⇒ Điều kiện thời tiết: Các điều kiện này bao gồm nhiệt độ, lượng mây, điểm sương, lượng mưa. Có thể thấy rằng trong hầu hết các trường hợp có một liên hệ rõ rệt giữa thời tiết và nhu cầu sử dụng ga, đặc biệt là nhiệt độ và tốc độ gió. Trong phần lớn các tình huống, khi mà nhiệt độ hạ xuống, thì nhu cầu sử dụng ga tăng lên và ngược lại. Mặc dù vậy, quan hệ này là phi tuyến tính. Các yếu tố khác có ảnh hưởng ít hơn tới nhu cầu của khách hàng.
- ⇒ Điều kiện thời gian: Bao gồm giờ trong ngày, ngày trong tuần và tháng trong năm, những ngày cuối tuần và ngày nghỉ. Phần lớn các mẫu dữ liệu cho thấy có sự phụ thuộc rất mạnh vào các yếu tố này. Chẳng hạn, giả sử rằng tất cả các yếu tố khác giống nhau, nhu cầu sử dụng ga vào lúc 1 giờ sáng khi mà phần lớn mọi người đang ngủ sẽ khác so với lúc sáu giờ sáng khi mà mọi người đang chuẩn bị thức dậy. Có thể lấy các ví dụ tương tự đối với yếu tố ngày trong tuần. Mặc dù không thể tổng quát hóa thành một quy tắc chung, nhưng các ngày giữa tuần (Thứ Ba, Tư, Năm, Sáu) chắc chắn sẽ có nhu cầu khác so với những ngày còn lại. Tháng trong năm cho ta hiệu ứng về mùa. Những ngày nghỉ và ngày cuối tuần có xu hướng gần tương tự như nhau.
- ⇒ Thông tin kinh tế: Các thông tin như giá ga trên thị trường, tỷ suất giá ga so với giá dầu, và tỷ suất giá giữa các nhà cung cấp. Trong phần lớn các trường hợp, hiệu ứng của các nhân tố kinh tế đối với nhu cầu sử dụng ga là không tầm thường. Có thể thấy được các ảnh hưởng này khi mà khách hàng tăng hoặc giảm lượng yêu cầu. Nếu giá ga trên thị trường thấp, thậm chí nhiệt độ đang cao, có thể khách hàng sẽ có nhu cầu tiêu thụ nhiều hơn. Tương tự nếu nhiệt độ thấp nhưng giá ga cao thì khách hàng có xu hướng sử dụng ga lưu trữ hơn là mua mới, do vậy sẽ giảm nhu cầu. Sự so sánh giá ga so với giá dầu có một vai trò quan trọng trong việc xác định nhu cầu nếu khi khách

hàng sử dụng đồng thời cả hai loại nhiên liệu này. Nếu giá ga cao hơn giá dầu thì nhu cầu về ga có khuynh hướng giảm và ngược lại.

Những hiệu ứng trên là những thứ có thể xác định được số lượng và do vậy có thể là các đối tượng xem xét để sử dụng như là các đầu vào của mạng để huấn luyện và thực hiện dự báo. Có các nhân tố khác, chẳng hạn như các giao ước hợp đồng rõ ràng có một ảnh hưởng rõ rệt đối với nhu cầu sử dụng, nhưng chúng rất khó có thể được định lượng và do đó không thể coi chúng như là các tham số ảnh hưởng.

#### 3.3.2.2. Mô hình dự báo:

##### **Dữ liệu vào**

Dữ liệu vào sử dụng trong mô hình này được thu thập từ khách hàng, có thể là từ một cơ sở dữ liệu tác nghiệp của họ hay một dạng lưu trữ nào đó. Các dữ liệu lịch sử mà chúng ta quan tâm được lưu trữ dưới dạng sau:

Ngày	Giờ	Nhiệt độ	Tốc độ gió	Sử dụng
02-08-1998	00	37	3	1168
02-08-1998	01	37	9	1213
02-08-1998	02	37	6	1316
02-08-1998	03	37	3	1417
02-08-1998	04	37	3	1534
02-08-1998	05	37	5	1680
02-08-1998	06	36	5	1819
02-08-1998	07	34	6	1967

##### **Tiền xử lý**

Với các dữ liệu đã cho, có thể thiết lập mô hình phản ánh bởi sáu hiệu ứng sau:

- 1) Nhiệt độ: Chính là giá trị thực của nó.
- 2) Tốc độ gió: Thể hiện bằng giá trị thực của nó.
- 3) Giờ trong ngày: Thể hiện 24 tiếng trong ngày: 0, 1, 2... 23



4) Ngày trong tuần: Thể hiện các ngày Chủ nhật, thứ Hai, thứ Ba, thứ Tư, thứ Năm, thứ Sáu, thứ Bảy bằng các số 0, 1, 2, 3, 4, 5, và 6 tương ứng.

5) Ngày cuối tuần: thể hiện thứ Hai, thứ Ba, thứ Tư, thứ Năm, thứ Sáu bởi 0; thứ Bảy và Chủ nhật bởi 1.

6) Tháng trong năm: thể hiện 12 tháng trong năm bởi các giá trị từ 0 đến 11.

Rõ ràng là các hiệu ứng 1) và 2) là các biến có thứ tự. Giá trị của chúng có thể được đưa vào mạng như chúng vốn có. Các hiệu ứng còn lại là các biến phân loại. Ta biết rằng, đối với các biến phân loại, chúng ta có thể sử dụng phương pháp mã hóa *1-of-c* (sẽ phải dùng  $1+1+24+7+2+12=47$  đơn vị đầu vào), hoặc phương pháp one-effect-one-unit (chỉ dùng có  $1+1+1+1+1+1=6$  đơn vị đầu vào) (*Xem lại Mục 3.2.1*).

Ở đây chúng ta sử dụng cách thứ hai, mặc dù chúng gây ra một trật tự nhân tạo trên các giá trị nhưng chúng giảm đi rất nhiều số lượng các đầu vào, từ đó có thể làm đơn giản mô hình.

Tập dữ liệu có thể được tạo ra bằng cách sử dụng bảng tính, các dữ liệu theo khuôn dạng nói trên sẽ được mã hóa thành dạng dưới đây:

Nhiệt độ	Tốc độ gió	Giờ	Ngày trong tuần	Ngày cuối tuần	Tháng	Sử dụng
37	3	00	6	1	1	1168
37	9	01	6	1	1	1213
37	6	02	6	1	1	1316
37	3	03	6	1	1	1417
37	3	04	6	1	1	1534
37	5	05	6	1	1	1680
36	5	06	6	1	1	1819
34	6	07	6	1	1	1967

Ngoài ra, các dữ liệu chưa tốt cũng cần được xử lý, chẳng hạn như các giá trị nằm ngoài khoảng giá trị thực tế,... Tất cả dữ liệu đầu vào được đưa về khoảng  $[0,1]$ .

### **Kiến trúc mạng**

Mạng bao gồm một lớp ra, một lớp ẩn. Rõ ràng là chỉ có duy nhất một đơn vị ở đầu ra - lượng tiêu thụ. Số đầu vào được cố định, phụ thuộc vào số nhân tố ảnh hưởng được sử dụng. Số đơn vị trong lớp ẩn được xác định bằng cách huấn luyện với một số tập kiểm tra.

Mạng sẽ yêu cầu một số đơn vị trong lớp ẩn vừa đủ để có thể học được các đặc trưng tổng quát về mối quan hệ giữa các nhân tố đầu vào và đầu ra. Mục tiêu của chúng ta là làm sao chỉ phải sử dụng số các đơn vị trong lớp ẩn càng ít càng tốt, đồng thời vẫn duy trì được khả năng của mạng có thể học được mối quan hệ giữa các dữ liệu. Như đã nêu, sử dụng nhiều hơn một lớp ẩn không tăng đáng kể độ chính xác của các dự báo.

Các hàm kích hoạt của các đơn vị trong lớp ẩn là các hàm sigmoid. Đối với các đơn vị ở lớp ra có thể là hàm sigmoid hoặc hàm đồng nhất. Ta sẽ chọn hàm đồng nhất.

### **Cài đặt thuật toán lan truyền ngược**

Mạng được huấn luyện bằng thuật toán lan truyền ngược. Hàm lỗi trung bình bình phương được sử dụng:

$$E = \frac{1}{2} \sum_{k=1}^n (t_k - y_k)^2 .$$

Như đã nêu trên, hàm chuyển của các đơn vị lớp ẩn là hàm sigmoid:

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

Hàm này có một đặc trưng rất có ích đó là đạo hàm của nó có thể biểu diễn dưới dạng sau:

$$g'(x) = g(x)(1 - g(x)) .$$

Các hàm trên có thể dễ dàng cài đặt với ngôn ngữ Visual Basic 6.0 của hãng Microsoft:

```
Public Function sigmoid (x As Double) As Double
```

```
    If (x > 50) Then
        sigmoid = 1
    ElseIf (x < -50) Then
        sigmoid = 0
    Else
        sigmoid = 1 / (1 + Exp(-x))
    End If
```

```
End Function
```

```
Public Function sigmoidDerivative (x As Double) As Double
```

```
    sigmoidDerivative = x * (1 - x)
```

```
End Function
```

Bước đầu tiên của thuật toán lan truyền ngược đó là: truyền xuôi (*forward propagation*)

```
Public Sub forward_prop()
```

```
    Dim i As Integer, j As Integer, k As Integer
```

```
    Dim aTemp() As Double, aTmp() As Double
```

```
    ' Đặt đầu vào cho lớp ẩn
```

```
    layers(0).Set_Inputs inputs
```

```
    For i = 0 To numOfLayers - 1
```

```
        layers(i).calc_out
```

```
        layers(i).getOutputs aTemp
```

```
        If i + 1 < numOfLayers Then
```

```
            layers(i + 1).Set_Inputs aTemp
```

```
        End If
```

```
    Next
```

```
End Sub
```

Trong đó, các hàm **Set\_Inputs**, **calc\_out**, **getOutputs** là các hàm thành phần của lớp **layers**, lớp (class) dành biểu diễn cho các lớp mạng.

Các hàm đó như sau:

**Public Sub calc\_out ()**

Dim i, j, k As Integer

Dim acc As Double

acc = 0

For j = 0 To numOutputs - 1

For i = 0 To numInputs - 1

k = i \* numOutputs

If Weights(k + j) \* Weights(k + j) > 100000 Then

Debug.Print "Trọng số tăng quá lớn ...!"

Exit For

End If

outputs(j) = Weights(k + j) \* inputs(i) + bias(j)

acc = acc + outputs(j)

Next

If layerType = 0 Then

outputs(j) = acc

ReDim Preserve predicted\_values(j)

predicted\_values(j) = acc

outValue = acc

Else

outputs(j) = sigmoid(acc)

End If

acc = 0

Next

**End Sub**

**Public Sub Set\_Inputs (ByRef nIns)**

Dim i As Integer

```
For i = 0 To numInputs - 1
    inputs(i) = nIns(i)
Next

End Sub

Public Sub getOutputs(ByRef nOut)

    Dim i

    For i = 0 To numOutputs - 1

        ReDim Preserve nOut(i)

        nOut(i) = outputs(i)

    Next
```

**End Sub**

Bước thứ hai của thuật toán là lan truyền ngược lỗi (*Backward propagation*). Đối với đơn vị ở lớp ra, lỗi  $\delta$  được cho bởi phương trình:

$$\delta_k = y_k - t_k ,$$

trong khi đó, đối với các đơn vị lớp ẩn, lỗi  $\delta$  được tính bởi công thức:

$$\delta_j = z_j(1 - z_j) \sum_{k=1}^c w_{kj} \delta_k .$$

Các đạo hàm tương ứng với các trọng số của lớp ẩn và lớp ra:

$$\frac{\partial E}{\partial w_{ji}} = \delta_j x_i \text{ và } \frac{\partial E}{\partial w_{kj}} = \delta_k z_j .$$

Ta sử dụng thuật toán giảm theo gradient (gradient descent algorithm) với tham số bước đã đề cập nhật trọng số:

```
Public Sub backward_prop(ByRef tongLoi As Double)

    Dim i As Integer

    Dim OE() As Double

    ' Tính toán lỗi đầu ra

    layers(numOfLayers - 1).calc_error tongLoi
```

```
For i = numOfLayers - 2 To 0 Step -1
    ' Truyền ngược lỗi đầu ra về các lớp trước nó
    layers(i + 1).getBack_errors OE
    layers(i).setOutput_errors OE
    ' Thực hiện tính toán lỗi cho lớp đó
    layers(i).mid_calc_error
Next
End Sub
```

Các hàm **calc\_error**, **getBack\_errors**, **setOutput\_errors**, **mid\_calc\_error** là các hàm thuộc lớp (class) **layers** được cài đặt như sau:

```
Public Sub calc_error(ByRef error)
    ' Hàm này được sử dụng cho lớp ra
    Dim i, j, k As Integer
    Dim acc As Double, total_error As Double

    acc = 0
    total_error = 0
    For j = 0 To numOutputs - 1
        ReDim Preserve output_errors(j)
        output_errors(j) = expected_values(j) - outputs(j)
        total_error = total_error + output_errors(j)
    Next
    error = total_error

    For i = 0 To numInputs - 1
        ReDim Preserve back_errors(i)
        k = i * numOutputs
        For j = 0 To numOutputs - 1
            back_errors(i) = Weights(k + j) * output_errors(j)
        Next
    Next
End Sub
```

```
        acc = acc + back_errors(i)
    Next

    back_errors(i) = acc

    acc = 0

    ' Lỗi lan truyền ngược lại được nhân với đạo hàm của hàm chuyển
    back_errors(i) = back_errors(i) * sigmoidDerivative(inputs(i))
Next

End Sub

Public Sub mid_calc_error()

    ' Sử dụng cho lớp ẩn

    Dim i, j, k As Integer

    Dim acc As Double

    acc = 0

    For i = 0 To numInputs - 1

        ReDim Preserve back_errors(i)

        k = i * numOutputs

        For j = 0 To numOutputs - 1

            back_errors(i) = Weights(k + j) * output_errors(j)

            acc = acc + back_errors(i)

        Next

        back_errors(i) = acc

        acc = 0

        ' Lỗi lan truyền ngược lại được nhân với đạo hàm của hàm chuyển
        back_errors(i) = back_errors(i) * sigmoidDerivative(inputs(i))
    Next

End Sub

Public Sub setOutput_errors(OE)

    Dim i

    For i = 0 To numOutputs - 1
```

```
ReDim Preserve output_errors(i)

output_errors(i) = OE(i)

Next
```

**End Sub**

**Public Sub getBack\_errors(ByRef BE)**

```
Dim i

For i = 0 To numInputs - 1

    ReDim Preserve BE(i)

    BE(i) = back_errors(i)

Next
```

**End Sub**

Sau khi lan truyền ngược lỗi từ đầu vào về các lớp trước, ta bắt đầu thực hiện cập nhật trọng số:

**Public Sub update\_weights()**

```
Dim i As Integer

For i = 0 To numOfLayers - 1

    layers(i).updateWeights learningRate, alpha

Next
```

**End Sub**

Hàm **updateWeights** là hàm thuộc lớp (class) **layers** thực hiện cập nhật trọng số của từng lớp:

**Public Sub updateWeights(ByVal beta As Double, ByVal alpha As Double)**

```
Dim i, j, k As Integer

Dim delta As Double, deltaB As Double, avgDelta As Double

' Learning law: weight_change = beta * output_errors * input +
'                                     alpha * past_deltas

' Cập nhật trọng số

For i = 0 To numInputs - 1
```



```
k = i * numOutputs
For j = 0 To numOutputs - 1
    delta = beta * output_errors(j) * inputs(i)
           + alpha * past_deltas(k + j)
    Weights(k + j) = Weights(k + j) + delta

    ReDim Preserve cum_deltas(k + j)
    cum_deltas(k + j) = cum_deltas(k + j) + delta
Next
Next

' Cập nhật độ lệch
For j = 0 To numOutputs - 1
    deltaB = beta * output_errors(j) + alpha * past_deltas_B(j)
    bias(j) = bias(j) + deltaB

    ReDim Preserve cum_deltas_B(j)
    cum_deltas_B(j) = cum_deltas_B(j) + deltaB
Next
```

**End Sub**

Các tham số **past\_deltas** và **cum\_deltas** được cập nhật mỗi khi một chu kỳ huấn luyện mới bắt đầu bởi thủ tục **updateMomentum** của lớp (class) **network**.

**Public Sub updateMomentum()**

```
Dim i

For i = 0 To numOfLayers - 1
    layers(i).updateMomentum
Next
```

**End Sub**

Thủ tục này thực chất là trao đổi giá trị **cum\_deltas** và **past\_deltas** của từng lớp.

**Public Sub updateMomentum()**

```
Dim i, j, k
` Tráo đổi các vector deltas
swapVector past_deltas, cum_deltas
swapVector past_deltas_B, cum_deltas_B
For i = 0 To numInputs - 1
    k = i * numOutputs
    For j = 0 To numOutputs - 1
        ReDim Preserve cum_deltas(k + j)
        cum_deltas(k + j) = 0
    Next
Next
For j = 0 To numOutputs - 1
    ReDim Preserve cum_deltas_B(j)
    cum_deltas_B(j) = 0
Next
```

**End Sub**

Sau đây là toàn bộ thủ tục huấn luyện của mạng nơron:

**Public Sub train()**

```
Dim tot_err As Double, num As Integer
currentError = 0.9999
Dim MaxCycles As Integer, i As Integer, j As Double, k As Integer
MaxCycles = 15000
i = 0
j = 0
tot_err = 0
ReDim predictValue(numOfPatterns - 1)
ReDim arrMSE(50)
```

*'Hiển thị các màn hình theo dõi lỗi và đồ thị khi huấn luyện*

FrmPlot.Show

mseUpdate.Show

Do While True And Not stopTraining

    updateMomentum

*' Lấy mẫu thứ i trong tập huấn luyện đã chuẩn bị sẵn*

    get\_patterns i

*' Đặt vector thứ i cho mạng biết*

    setNumPatterns i

    setStatusText 1, "Training..."

*' Truyền xuôi*

    forward\_prop

    predictValue(i) = layers(numOfLayers - 1).getoutValue

*' Truyền ngược lỗi*

    backward\_prop currentError

*' Tính toán lỗi*

    tot\_err = tot\_err + currentError \* currentError

    Set\_currentCycle CLng(j)

*' Thực hiện cập nhật trọng số*

    update\_weights

*' Tăng số thứ tự vector mẫu sẽ đọc vào lên 1*

    i = i + 1

DoEvents

If i = numOfPatterns Then

*' Nếu đã kết thúc một chu kỳ*

    i = 0

    j = j + 1

*' Tính toán lỗi trung bình bình phương*

```
MSE = sqr(tot_err) / numOfPatterns

ReDim Preserve arrMSE(UBound(arrMSE) + 1)

arrMSE(UBound(arrMSE)) = MSE

'Thực hiện cập nhật các đồ thị mỗi 50 chu kỳ

If NumOfCycles Mod 50 = 0 Then

    FrmPlot.update

    mseUpdate.update

    DoEvents

    ReDim arrMSE(0)

    DoEvents

End If

If Abs(MSE) < errorToleranceRate Then

    'Nếu lỗi trung bình bình phương nhỏ hơn hệ số thứ lỗi

    'thì kết thúc huấn luyện

    Set_currentCycle CLng(j)

    setcurrentError currentError

    DoEvents

    setstopTraining True

End If

MSE = 0

tot_err = 0

NumOfCycles = NumOfCycles + 1

End If

Loop

setStatusText 1, "DONE!!!!!!!!!!"

'Ghi lại các ma trận trọng số

write_weights

setStatusText 2, "Cycle number: " & CStr(j)

setStatusText 3, "Total error: " & Abs(currentError)
```

```
Set_currentCycle CLng(j)  
  
FrmPlot.update  
  
DoEvents
```

**End Sub**

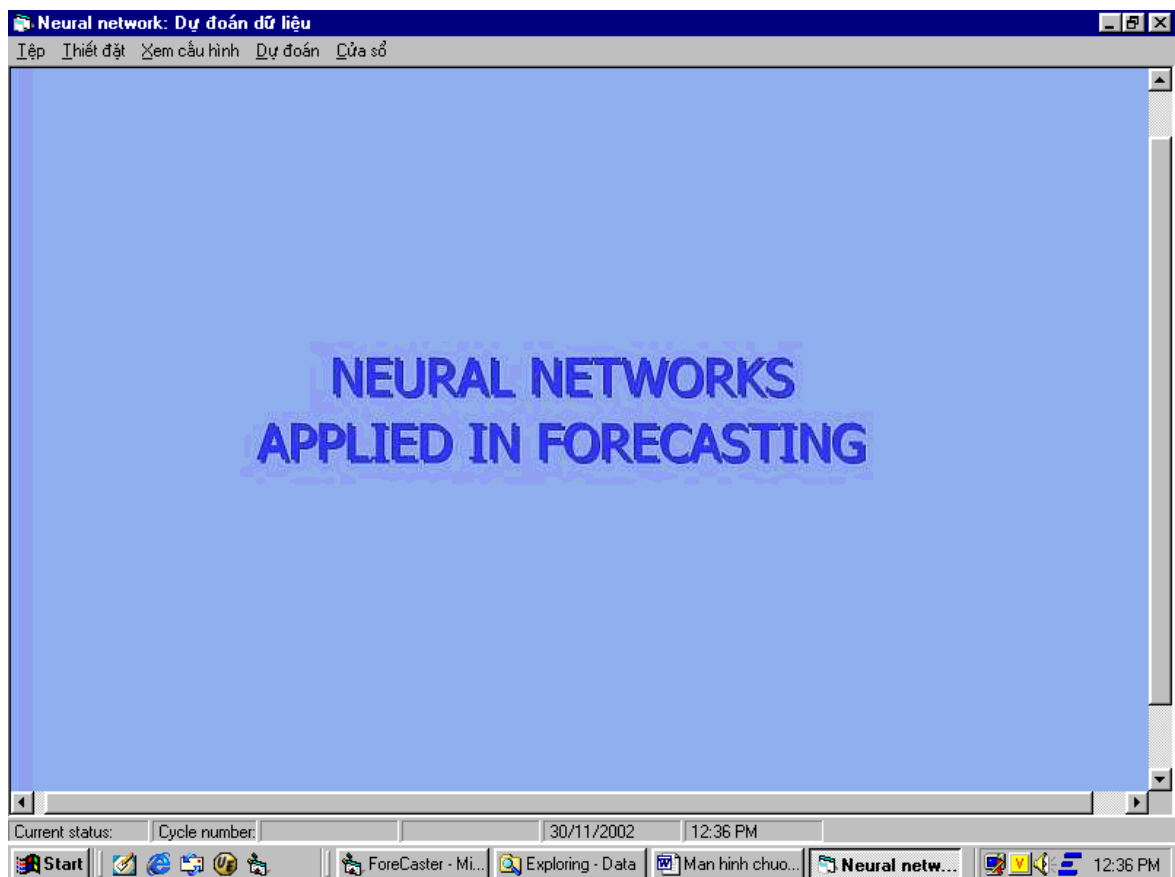
Các mẫu được tuần tự đưa vào mạng để huấn luyện.

### **Sự tổng quát hóa của mạng**

Một phần dữ liệu được sử dụng như là tập kiểm tra, tập này sẽ không được sử dụng trong quá trình huấn luyện. Trong quá trình huấn luyện trên tập dữ liệu huấn luyện, sự tổng quát hóa đối với các dữ liệu kiểm tra được hiển thị đồng thời dựa trên các tham số hiện tại của mạng.

#### **3.3.3. Chương trình dự báo dữ liệu**

Màn hình ban đầu của chương trình

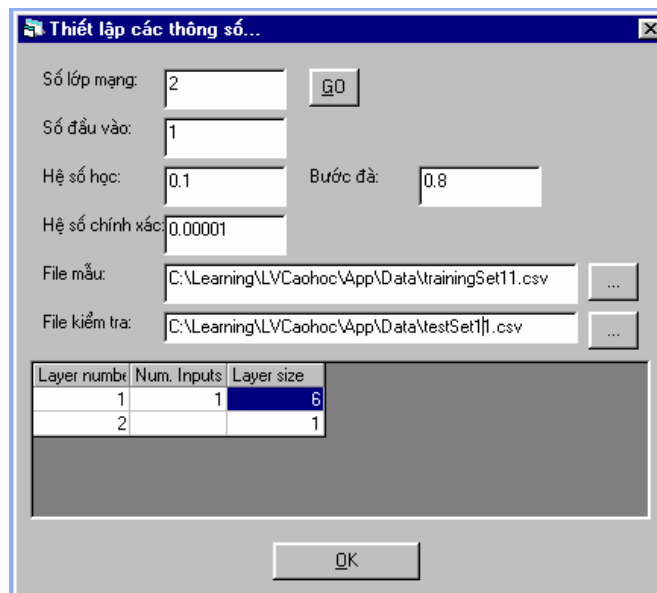


Chương trình được xây dựng bao gồm các mục thực đơn: **Tệp**, **Thiết đặt**, **Xem cấu hình**, **Dự đoán**. Sau đây, các đặc trưng chính của hệ thống sẽ được mô tả chi tiết.

### 3.3.3.1. Màn hình nhập các tham số cho mạng.

Chức năng này cho phép người sử dụng nhập các tham số đầu vào cho mạng như: Số lớp mạng, Số đầu vào, Hệ số học,... Sau khi người sử dụng đã nhập xong các mục, cần nhấn nút lệnh **GO** để thực hiện nhập cấu trúc cho mạng.

Sau khi đã nhập xong xuôi các tham số, nhấn **OK** để ghi lại các tham số vừa nhập. Tại đây, các tham số cho mạng nơron được gán các giá trị, đồng thời, các bộ dữ liệu huấn luyện và kiểm tra cũng được đọc vào bộ đệm chương trình, tiền xử lý.



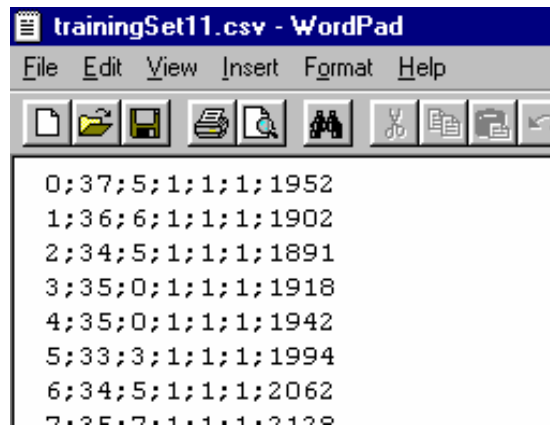
Layer number	Num. Inputs	Layer size
1	1	6
2		1

Các tệp dữ liệu là các tệp có cấu trúc:

- Các trường dữ liệu được phân cách bởi dấu “;”
- Trường dữ liệu dự báo là trường cuối cùng.
- Sau trường dữ liệu dự báo không cần phải có dấu “;”.
- Tệp dữ liệu không được có các khoảng trống ở phía cuối. Nếu có thì cần được loại bỏ.

Ví dụ:

Tệp dữ liệu có dạng như sau:



Các dữ liệu sau khi được đọc vào sẽ được chuẩn hóa về khoảng [0,1] theo phương pháp:

$$SV = ((0.9 - 0.1) / (\text{MAX\_OF\_EXP} - \text{MIN\_OF\_EXP})) * (\text{OV} - \text{MIN\_OF\_EXP}),$$

trong đó:

SV: Scaled Value - Giá trị sau khi biến đổi

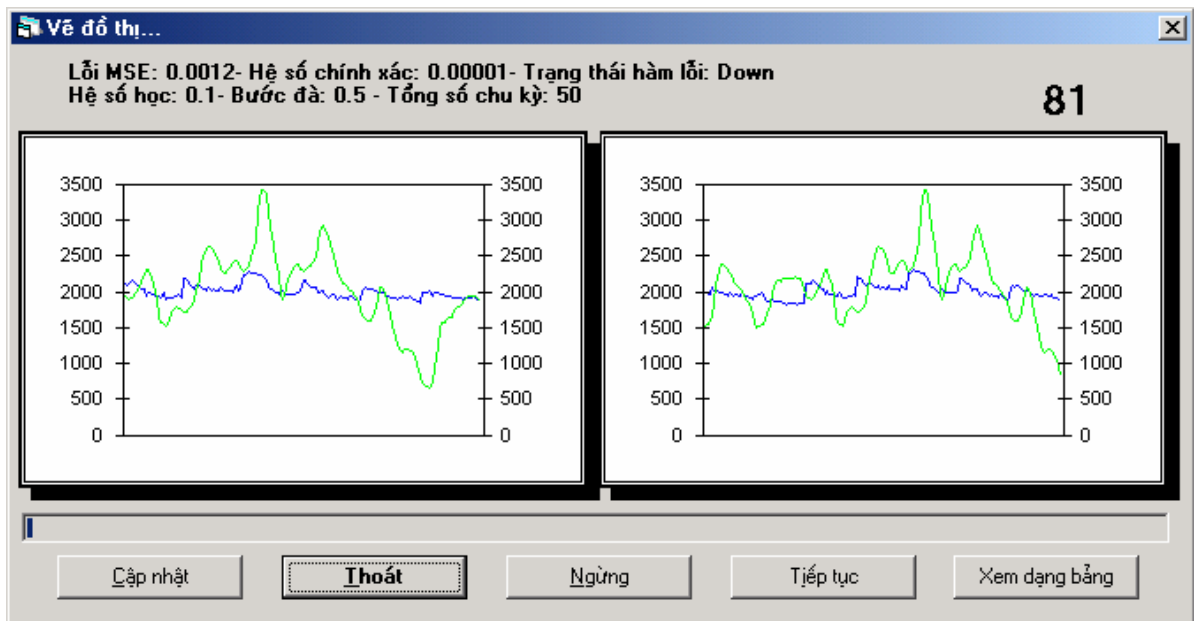
OV: original Value - Giá trị ban đầu

MAX\_OF\_EXP, MIN\_OF\_EXP: Giá trị lớn nhất vào nhỏ nhất của tập các giá trị

0.9, 0.1: Giá trị “lớn nhất” và “nhỏ nhất” của hàm sigmoid.

### 3.3.3.1. Huấn luyện mạng.

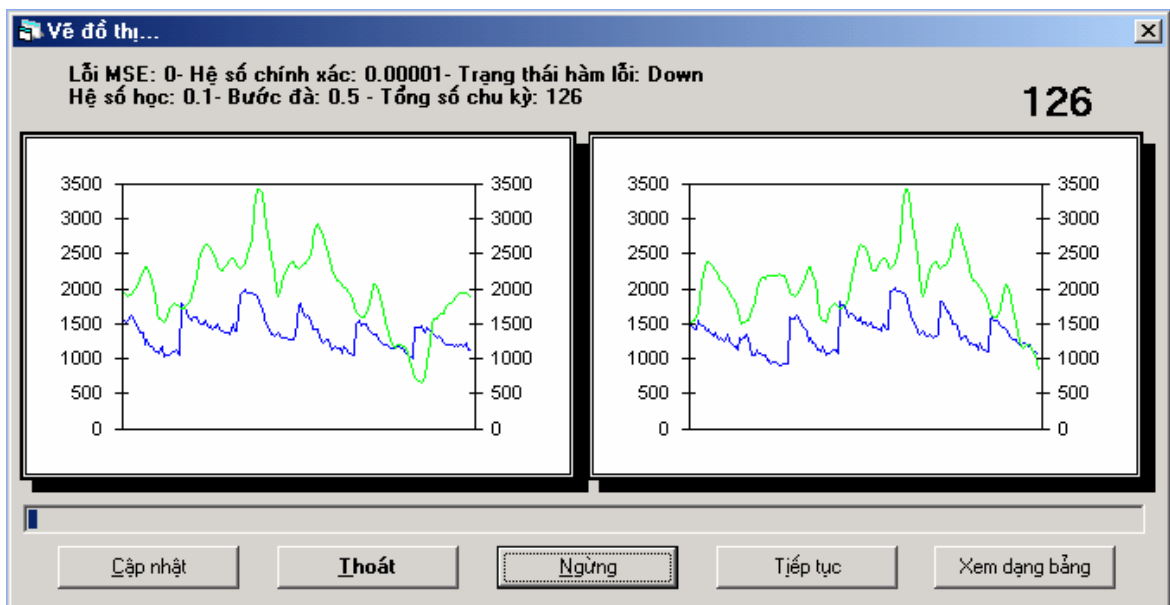
Sau khi qua bước thiết lập các thông số cho mạng, có thể bắt đầu huấn luyện mạng. để thực hiện điều này, chọn: **Thiết đặt Huấn luyện mạng (Train network)**. Màn hình ban đầu thể hiện trạng thái của việc huấn luyện có dạng sau:



Chú thích:

Đồ thị bên trái thể hiện kết quả huấn luyện mạng trên tập mẫu đưa vào. Đồ thị bên phải thể hiện trả lời của mạng đối với các mẫu kiểm tra, các mẫu chưa đưa vào mạng.

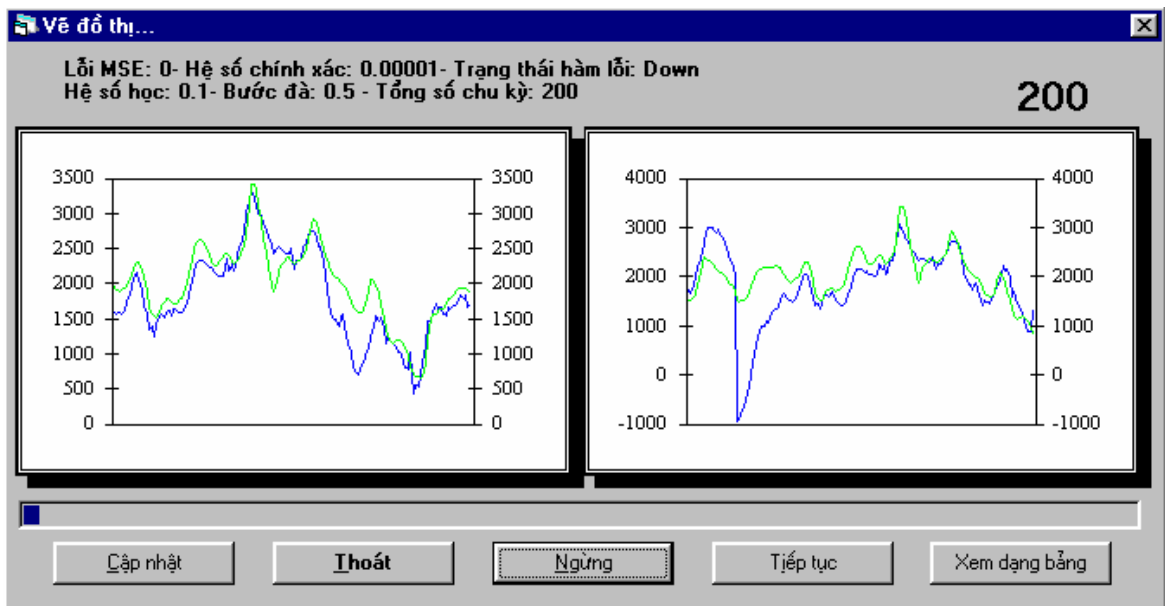
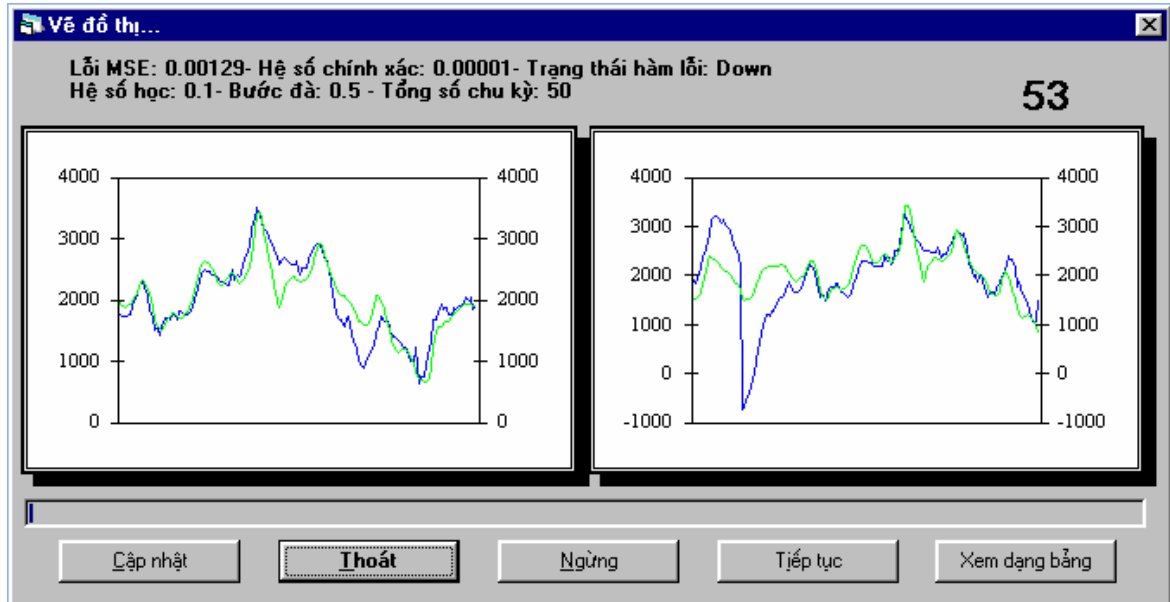
Sau một số chu kỳ huấn luyện, mạng đã có kết quả trả lời đối với tập dữ liệu huấn luyện và tập kiểm tra tốt hơn so với trạng thái ban đầu.

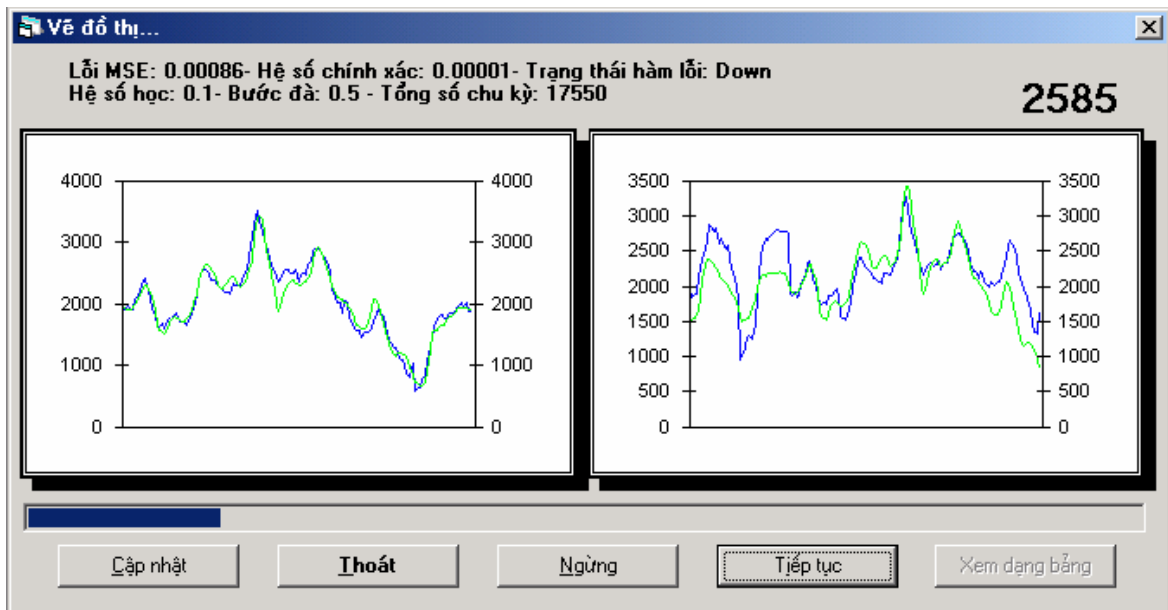


Các đường màu xanh lá cây (nhạt) là các đầu ra mong muốn đối với tập dữ liệu. Các đường màu xanh đậm (sẫm) là trả lời của mạng đối với các dữ liệu đầu vào đưa vào nó.

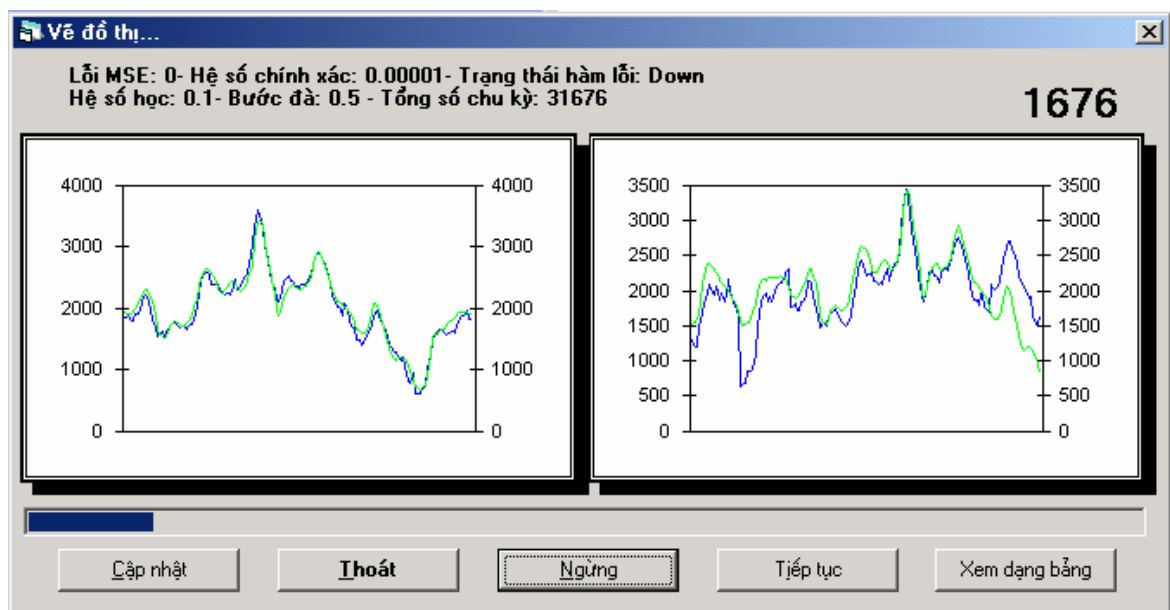


Sau một số chu kỳ tiếp sau, trả lời của mạng đối với dữ liệu huấn luyện và kiểm tra đã tốt hơn nhiều so với trạng thái ban đầu.



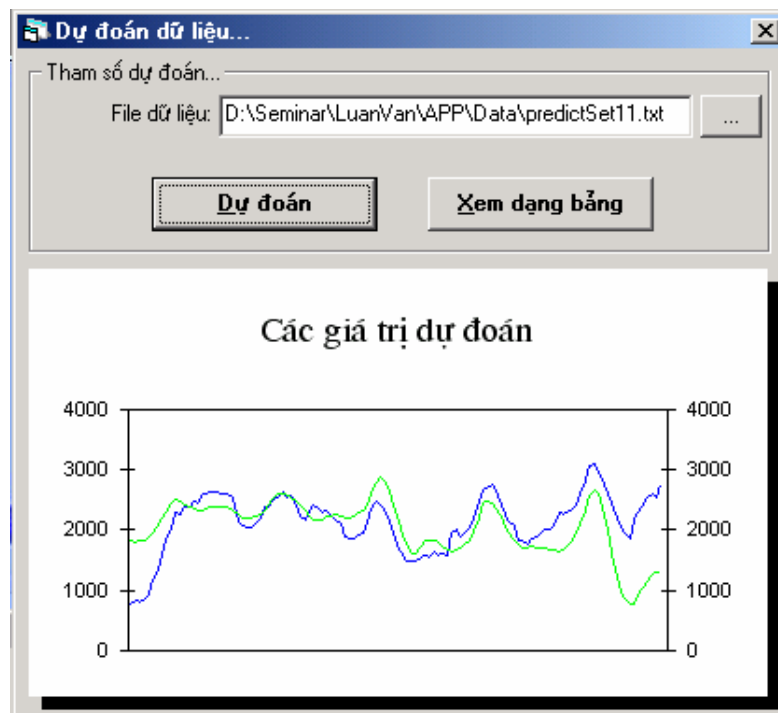


Có thể thấy, lỗi MSE được giảm sau một thời gian huấn luyện, đồng thời khả năng tổng quát hóa của mạng đối với các dữ liệu chưa được “biết” cũng đã tốt lên.



### 3.3.3.3. Dự báo dữ liệu.

Sau khi mạng đã được huấn luyện, có thể sử dụng để dự báo dữ liệu. Chỉ cần xác định tệp chứa dữ liệu và thực hiện dự báo. Màn hình như sau:



### 3.4. Một số nhận xét

- Mạng bị ảnh hưởng rất nhiều từ trạng thái khởi đầu của các tham số. Trong quá trình học, mạng cố gắng điều chỉnh các tham số sao cho tổng bình phương lỗi là nhỏ nhất. Khả năng hội tụ của mạng phụ thuộc vào các tham số khởi đầu, còn khả năng tổng quát hóa thì lại phụ thuộc rất nhiều vào dữ liệu đầu vào. Nếu dữ liệu đầu vào quá nhiều (!) thì có thể dẫn tới tình trạng luyện mạng mất rất nhiều thời gian và khả năng tổng quát hóa kém, nếu quá ít dữ liệu thì sai số sẽ tăng.
- Ngoài đặc trưng về dữ liệu, một đặc trưng khác trong quá trình huấn luyện mạng cần quan tâm là nếu số lần thực hiện điều chỉnh các tham số của mạng quá ít sẽ dẫn đến tình trạng là khả năng tổng quát hóa của mạng rất kém. Bởi vậy, số chu kỳ các mẫu đưa vào mạng cần được xem xét phải lớn hơn một ngưỡng nào đó (từ vài nghìn cho đến vài chục nghìn lần).
- Để có thể xem xét, đánh giá được khả năng tổng quát hóa của mạng, cần thực hiện phân chia tập dữ liệu thành các tập: huấn luyện (training set) và tập kiểm tra (test set). Tập các dữ liệu thử sẽ không đưa vào để kiểm tra hoạt động của mạng để đảm bảo sự khách quan.

- Một vấn đề nữa đối với mạng nơron đó là khả năng rơi vào các điểm cực trị địa phương. Như chúng ta đã biết, thuật toán Lan truyền ngược lỗi không đảm bảo sẽ cho ta điểm cực trị toàn cục. Nếu rơi vào điểm cực trị địa phương, ta sẽ phải bắt đầu huấn luyện lại, điều này sẽ khiến cho mạng nơron sẽ không thể áp dụng được trong thực tế đối với các bài toán yêu cầu độ chính xác cao trong thời gian tối thiểu. Do đó, giải pháp sử dụng hệ số học biến đổi là một trong các hướng để có thể vượt qua được nhược điểm trên. Ngoài ra, nếu dữ liệu phân bố không đều trên từng mẫu thì khả năng tổng quát hóa cũng không tốt.
- Một điều nữa, là mạng có khả năng sẽ không thể đạt được trạng thái mong muốn, mà có thể nó sẽ bỏ qua điểm cực trị. Để có thể tránh điều này, không nên đặt hệ số học quá lớn (cỡ 0.1 chẳng hạn), cũng như hệ số bước đà quá lớn (chẳng hạn  $= 0.5$ ) (do đặc trưng của thuật toán lan truyền ngược sử dụng tham số bước đà).
- Như đã nêu trên, để đảm bảo khả năng có thể đạt đến điểm cực tiểu, số các đơn vị trong lớp ẩn cần đủ lớn. Tuy nhiên, nếu số các đơn vị trong lớp ẩn vượt quá một ngưỡng nào đó thì khả năng tổng quát hóa của mạng sẽ kém, bởi lẽ sau khi huấn luyện mạng có xu hướng ghi nhớ tất cả các mẫu đã được học. Khi đó, nên xem xét đến khả năng sử dụng thêm một lớp ẩn nữa với số nơron nhỏ (vài nơron) và giảm bớt số nơron ở lớp ẩn thứ nhất.

## KẾT LUẬN



Mạng nơron có thể được huấn luyện để xấp xỉ các hàm bất kỳ mà không cần biết trước sự liên hệ của các đầu vào đối với đầu ra. Chúng có thể hoạt động như một bộ nhớ tự liên hợp bằng cách sử dụng các dữ liệu đặc thù cho các ứng dụng, bài toán trong các lĩnh vực cụ thể. Đó là đặc trưng đem lại cho mạng nơron lợi thế đối với các mô hình khác, đặc trưng thứ lỗi.

Trong luận văn này, chúng tôi xem xét các thuộc tính của mạng nơron truyền thẳng và quá trình xác định các đầu vào, kiến trúc của mạng phục vụ cho một bài toán cụ thể. Chúng tôi cũng đã xây dựng một hệ chương trình dự báo dữ liệu nhằm áp dụng các vấn đề lý thuyết đã tìm hiểu. Các thí nghiệm cho thấy, nếu như được huấn luyện tốt trên tập các dữ liệu đầy đủ và hoàn thiện với các tham số được lựa chọn cẩn thận thì kết quả dự báo có thể chính xác đến 90%. Chương trình cũng cung cấp khả năng lưu lại tập các tham số, trọng số và các độ lệch sau những lần huấn luyện thành công và nạp lại các tham số này để sử dụng khi dự báo dữ liệu.

Tuy nhiên, luận văn này mới chỉ xem xét đến các khía cạnh tổng thể về mạng nơron truyền thẳng nhiều lớp và vấn đề dự báo dữ liệu trong khoảng thời gian ngắn (short-term forecasting) và trung bình (mid-term forecasting). Tuy nhiên, ứng dụng của các vấn đề lý thuyết thể hiện trong hệ chương trình được xây dựng hoàn toàn có thể áp dụng cho các bài toán dự báo trong thời gian dài (long-term forecasting) với một số sửa đổi trong thuật toán huấn luyện.

Cần nhấn mạnh rằng, để có thể dự báo được dữ liệu, ta cần sử dụng các dữ liệu lịch sử để huấn luyện và có thể cả các dữ liệu dự báo của các đầu vào (Ví dụ như: dự báo nhiệt độ ngày hôm sau,...). Người ta cũng đã chỉ ra rằng mạng nơron truyền thẳng nhiều lớp có khả năng tốt nhất trong dự báo trong khoảng thời gian ngắn.

Mạng nơron truyền thẳng nhiều lớp có thể sử dụng trong rất nhiều bài toán dự báo trong các lĩnh vực khác: dự báo lượng sử dụng điện, nước, thị trường chứng khoán, lưu lượng giao thông và lượng sản phẩm bán ra chừng nào các mối quan hệ giữa các đầu vào và đầu

ra có thể thấy được và đưa vào trong mô hình. Tuy vậy, không tồn tại một mô hình chung thích hợp cho tất cả các bài toán dự báo trong thực tế. Đối với mỗi một bài toán, cần thực hiện phân tích cẩn kẽ, cụ thể các dữ liệu trong phạm vi và sử dụng các tri thức thu thập được để có thể xây dựng được một mô hình thích hợp. Các phân tích và các tri thức thu thập được luôn có ích trong việc lựa chọn các đầu vào, mã hóa các đầu vào này hoặc quyết định cấu trúc của mạng, đặc biệt khi mà dữ liệu trong lĩnh vực đó chỉ có giới hạn.

Thuật toán lan truyền ngược chuẩn được sử dụng trong việc huấn luyện mạng nơron truyền thẳng nhiều lớp đã chứng tỏ khả năng rất tốt thậm chí đối với cả các bài toán hết sức phức tạp. Mặc dù vậy, để có được khả năng như vậy, ta cần mất rất nhiều thời gian để huấn luyện, điều chỉnh các tham số của mạng (thậm chí cả đối với các bài toán có cấu trúc hết sức đơn giản). Điều này luôn là trở ngại đối với các bài toán trong thực tế, do vậy, các thuật toán cải tiến cần được áp dụng để tăng khả năng hội tụ của mạng khi huấn luyện.

Luận văn này được thực hiện nhằm làm sáng tỏ những vấn đề lý thuyết về mạng nơron truyền thẳng nhiều lớp, thuật toán lan truyền ngược, các bước cần thực hiện khi phân tích, thiết kế và xây dựng ứng dụng cho bài toán dự báo dữ liệu, đồng thời xây dựng một chương trình ứng dụng nhằm mục đích thể hiện các vấn đề lý thuyết đã nêu. Chắc chắn luận văn này vẫn còn những thiếu sót, chúng tôi rất mong nhận được những ý kiến đóng góp nhằm hoàn thiện hơn nữa hiểu biết của mình.

## TÀI LIỆU THAM KHẢO

- [1]. Đ. M. Tường, *Trí tuệ nhân tạo*, NXB Khoa học và Kỹ thuật, 2002.
- [2]. Dipti Srinivasan, A. C. Liew, John S., P. Chen, Short term forecasting using neural network approach, *IEEE 91TH0374-9/91/0000-0012*, pp 12-16, 1991.
- [3]. Drucker H., Cun Y. L., Improving Generalization Performance using Double Backpropagation, *IEEE Transactions on neural networks*, Vol. 3, No. 6, November 1992.
- [4]. Hagan M. T., Demuth H. B., Beale M., *Neural networks design*, PWS Publishing Company, Boston, Ma, 1996.
- [5]. Haykin, S., *Neural networks, a comprehensive foundation*, Macmillan New York, Ny 1994.
- [6]. Kaastra, I., & Boyd, M. - Designing a neural network for forecasting financial and economic time series - *Neurocomputing* **10** (1996), pp 215-236.
- [7]. Kesmir C., Nussbaum A. K., Schild H., Detours V., Brunak S., Prediction of proteasome cleavage motifs by neural networks, *Protein engineering*, Vol 15-No 4, pp 287-196, 2002.
- [8]. Kolen J. F., Pollack J. B., Back Propagation is Sensitive to Initial Condition, *Technical Report*, Laboratory for artificial intelligence Research-The ohio State university.
- [9]. Lawrence S., C. L. Giles, a. C. Tsoj, What size Neural Network Gives optimal Generalization? Convergence Properties of Backpropagation, *Technical Report*, Institute for Advanced Computer Studies - University of Maryland College Park, June 1996.
- [10]. Morioka Y., Sakurai K., Yokoyama A. Sekine Y., Next day peak load forecasting using a Multilayer neural network with an additional Learning, *IEEE*, 0-7803-1217-1/93, 1993.
- [11]. Oh S.H., Lee Yj., A modified error function to improve the error Back-Propagation algorithm for Multi-layer perceptrons, *ETRI Journal Vol 17, No 1*, April 1995
- [12]. Ooyen A. V., Nienhuis B., Improving the Convergence of the Back-Propagation algorithm, *Neural Networks*, Vol. 5, pp. 465-471, 1992.

- [13]. Poh, H. L., Yao, J. T., & Jašić T., Neural Networks for the Analysis and Forecasting of Advertising and Promotion impact - *International Journal of intelligent Systems in accounting, Finance & Management*. 7 (1998), pp 253-268.
- [14]. Rao, Valluru B. and Rao, Hayagriva V., *C++ Neural Networks and Fuzzy Logic*, MIS Press, 1993.
- [15]. Ripley B.D., *Pattern Recognition and Neural Networks*, Cambridge university Press, 1996.
- [16]. Sullivan, R., Timmermann, A. & White, H., Dangers of data-driven inference: the case of calendar effects in stock returns, *Discussion Paper*, University of California, San Diego, Department of economics, 7/1998.
- [17]. Swingler K., Financial Predictions, Some Pointers, Pitfalls, and Common errors, *Technical Report*, Center for cognitive and computational neuroscience - Stirling University, July 14, 1994.
- [18] Takashi O., Next day's peak load forecasting using an artificial neural network, *IEEE 0-7803-1217-1/93*, pp 284-289, 1993.
- [19] T. Masters, *Practical Neural Network Recipes in C++*. Academic Press, Inc., 1993.
- [20]. UdoSeiffert, Michaelis B., On the gradient descent in back-propagation and its substitution by a genetic algorithm, *Proceedings of the IASTED International Conference Applied informatics 14-17/02/2000*, Innsbruck, Austria.
- [21]. Vogl. P. T., Mangis J. K., Zigler A. K., Zink W. T. and Alkon D. L., "Accelerating the convergence of the back-propagation method", *Biological Cybernetics*, vol.59, pp 256-264, 09/1988.