Chương 4

MẠNG PERCEPTRONS

4.1. MỞ ĐẦU

Chương này với mục tiêu đầu tiên là: Giới thiệu về các luật học, các phương pháp để diễn giải những sự thay đổi tiếp theo mà nó có thể được làm trong một mạng, vì sự huấn luyện là một thủ tục mà nhờ đó mạng được điều chỉnh để làm một công việc đặc biệt. Tiếp theo đó ta tìm hiểu về các hàm công cụ để thiết lập mạng Perceptron đơn giản đồng thời chúng ta cũng khảo sát các hàm để khởi tạo và mô phỏng các mạng tương tự. Ta sử dụng mạng Perceptron như là một phương tiện biểu lộ của các khái niệm cơ bản.

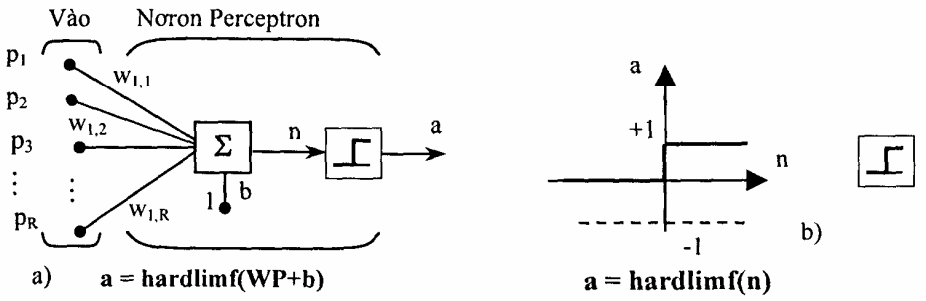
Rosenblatt đã thiết lập nhiều biến thể của mạng perceptron. Một trong các dạng đơn giản nhất là mạng lớp đơn mà hàm trọng và độ dốc của nó có thể được huấn luyện để đưa ra một véc tơ đích chính xác khi có véc tơ vào tương ứng được gửi tới. Kỹ thuật huấn luyện được gọi là luật học perceptron. Perceptron làm phát sinh nhiều cơ hội quan trọng cho khả năng khái quát hoá từ các véc tơ huấn luyện chúng và sự học từ điều kiện đầu phân bổ các mối quan hệ một cách ngẫu nhiên. Perceptron đặc biệt phù hợp cho những vấn đề đơn giản trong phân loại sản phẩm. Chung là những mạng nhanh và tin cậy cho những vấn đề chúng có thể giải quyết. Hơn nữa, sự thông hiểu hoạt động của Perceptron sẽ tạo cơ sở cho sự hiểu biết các mạng phức tạp hơn. Trong chương này, ta sẽ định nghĩa luật học, giải thích mạng Perceptron và luật học của nó, làm thế nào để khởi tạo và mô phỏng mạng Perceptron. Các vấn đề nêu ra ở đây chỉ là những vấn đề tóm lược cơ bản, để hiểu sâu hơn ta cần đọc trong [10].

4.1.1. Mô hình nơron perceptron

Một nơron Perceptron sử dụng hàm chuyển hardlim được chỉ ra trên hình 4.1**.**

Mỗi đầu p_i có hàm trọng với trọng liên kết w_{lj} và tổng các đầu vào kể cả độ dốc b là $n = \sum wl_j + b$ được gửi đến hàm chuyển bước nhảy (**hard-limit**) (Hình 4.1b). Đầu ra của nơron perceptron có giá trị 1 nếu n lớn hơn hoặc bằng 0 và có giá trị bằng 0 nếu n nhỏ hơn không:

$$a = \begin{cases} 1 \text{ khi } n \geq 0 \\ 0 \text{ khi } n < 0. \end{cases}$$



Hình 4.1a,b. Nơron với R đầu vào

a) Mô hình nơron, b) Hàm chuyển bước nhảy

Với hàm chuyển **hard-limit** cho phép Perceptron có khả năng phân loại véc tơ vào bằng cách phân chia không gian vào thành 2 vùng, phân cách với nhau bằng đường biên giới L ứng với phương trình: **W.p** + b = 0.

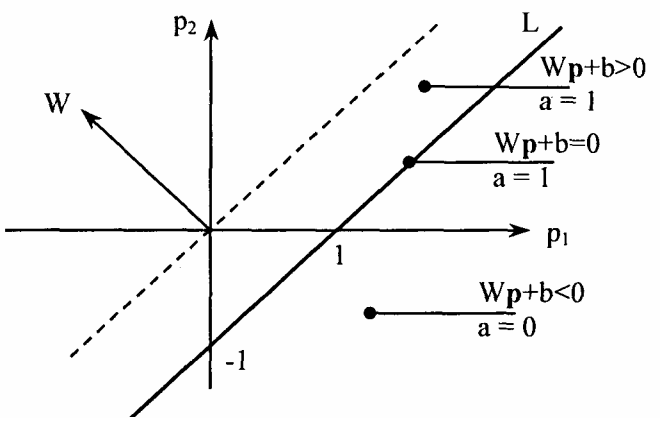
Ví dụ: Xét của nơron Perceptron có 2 đầu vào với các hàm trọng $w_{1,1}$= -1, $w_{1,2}$ = 1 và độ gốc b = 1. Ta có:

$n = \mathbf{W.p} + b = w_{1.1}p_1 + w_{1,2}.p_2 + b$

$= -p_1 + p_2 + 1.$

Đường biên giới L được chỉ ra trên hình 4. 1**.** Đường này vuông góc với ma trận trọng W và di chuyển dọc theo độ dốc b.

Các véc tơ vào ở phía trên và bên trái đường L có giá trị đầu vào mạng lớn hơn 0, vì vậy, nơron **hard-limit** đưa ra 1. Đường biên giới có thể chuyển hướng và di chuyển đến bất cứ chỗ nào để phân loại không gian vào mong muốn bằng cách lựa chọn hàm trọng và giá trị độ dốc. Nơron **hard-limit** không có độ dốc sẽ luôn có đường biên giới đi qua gốc toạ độ. Cộng thêm độ dốc sẽ cho phép nơron giải quyết bài toán ở đó 2 tập véc tơ vào không nằm trên 2 cạnh khác nhau của gốc toạ độ. Độ dốc cho phép đường biên giới thay đổi rời xa khỏi gốc như trên hình 4.2. Ta có thể thay đổi hướng của đường phân cách, chọn các đầu vào mới để phân loại và quan sát quá trình lặp của các luật học.

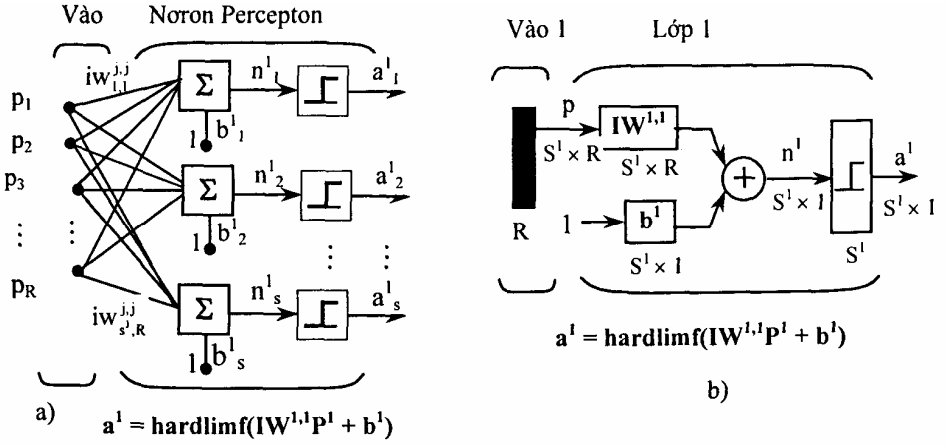


Hình 4.2. Sự phân loại của nơron Perceptron 2 đầu vào

4.1.2. Kiến trúc mạng perceptron

Hình 4.3a,b biểu diễn cấu trúc của mạng Perceptron bao hàm một lớp với S nơron Perceptron nối với R đầu vào thông qua tập các hàm trọng W_{ij}^{l}.

Luật học của perceptron được miêu tả ngắn gọn là khả năng huấn luyện chỉ của lớp đơn. Do vậy, ta chỉ coi là mạng một lớp, và nó chỉ có khả năng giải quyết được những bài toán đơn giản. Những hạn chế của Perceptron sẽ được đề cập đến phần cuối của chương.



Hình 4.3a,b. Kiến trúc một lớp mạng Perceptron

a) Kiến trúc đầy đủ, b) Ký hiệu tắt

4.2. THIẾT LẬP VÀ MÔ PHỎNG PERCEPTRON TRONG MATLAB

4.2.1 Thiết lập

Để thiết lập mạng perceptron ta dùng hàm **newp** với cú pháp:

newp net = newp(PR,S)

trong đó PR là min và max của các giá trị của R phần tử vào, S là số nơron. Hàm chuyển mặc định của perceptron là hardlim.

Ví dụ: để thiết lập một mạng perceptron với một phần tử, một lớp, giới hạn véc tơ vào từ 0 - 2 ta dùng lệnh:

net = newp([0 2],1);

Ta có thể thấy mạng đã được thiết lập thế nào bằng cách thực hiện chuỗi lệnh:

inputweights = net.inputweights{1,1}.

Kết quả cho ra

inputweights =

delays: 0

initFcn: 'initzero'

learn: 1

learnFcn: 'learnp'

learnParam: [ ]

size: 11 11

userdata: [IXI structl

weightFcn: 'dotprod'.

Hàm học mặc định là cho mạng Perceptron là hàm **learnp** (sẽ được đề cập ở phần sau). Tích số của véc tơ vào với ma trận trong liên kết cộng với độ dốc được đưa đến hàm chuyển hardlim. Hàm khởi tạo mặc định **initzero** được sử dụng để thiết lập giá trị ban đầu của trọng liên kết (thiết lập giá trị ban đầu bằng zero). Mô phỏng mạng ta được:

biases = net.biases{1}

gives biases =

initFcn: 'initzero'

learn: 1

learnFcn: 'learnp'

learnparam: [ ]

size: 1

userdata: [1x1 struct]

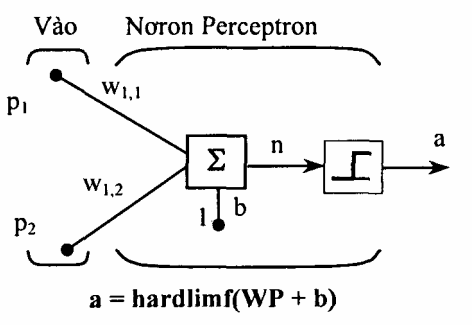
Ta cũng thấy rằng giá trị mặc định ban đầu của độ dốc là 0.

4.2.2. Mô phỏng (sim)

Để thấy sự làm việc của **sim**, ta xét ví dụ cần tạo ra một Perceptron một lớp có 2 đầu vào (hình 4.4). Ta định nghĩa một mạng với:

net = newp([-2 2;-2 +2],1);

Như đã biết ở trên, hàm trọng và độ dốc ban đầu lấy giá trị mặc định bằng 0, vì vậy nếu ta muốn một tập khác 0, ta cần phải thiết lập chúng. Ví dụ để thiết lập 2 hàm trọng và một độ dốc là: $w_{1,1}$ = -1, $w_{1,2}$ = 1 và b = 1 ta thực hiện 2 dòng lệnh:



Hình 4.4. Nơron với 2 đầu vào

net.IW{1,1}= [-1 1];

net.b{1} = [1];

Để đảm bảo chắc chắn rằng các tham số được thiết lập là chính xác, ta kiểu tra chúng với lệnh:

net.IW{1,1}

ans -

-1 1

net.b{1}

ans =

Bây giờ ta thấy nếu mạng đưa ra 2 tín hiệu, một trong mỗi cạnh của đường biên giới perceptron.

P1 = [1;1];

a1 = sim(net,p1)

a1 =

1

and for p2 = [1;-1]

a2 = sim(net,p2)

a2 =

0

Như vậy perceptron đã phân loại 2 đầu vào một cách chính xác.

Chú ý: Ta cũng có thể đưa đến 2 đầu vào một chuỗi dữ liệu, khi đó ở đầu ra ông nhận được một chuỗi dữ liệu. Ví dụ:

p3 = {[1;1] [1;-1]};

a3 = sim(net,p3)

a3 =

[1] [0]

4.2.3. Khởi tạo

Ta có thể sử dụng hàm **init** để thiết lập lại (reset) hàm trọng và độ dốc về giá rị ban đầu. Để làm điều đó, chúng ta hãy bắt đầu với mạng:

net = newp(1-2 2;-2 +2],1);

Để kiểm tra hàm trọng của nó, ta dùng lệnh:

wts = net.IW{1,1}

Kết quả ta được:

wts =

0 0

Tương tự, ta có thể kiểm tra độ dốc b = 0

bias = net.b{1}

Kết quả:

hias =

0

Bây giờ ta thiết lập hàm trọng có giá trị 3 và 4, độ dốc bằng 5

net.IW{1,1} = [3,4];

net.b{1} = 5;

Kiểm tra lại kết quả

wts =

3 4

bias =

Sử dụng init để reset the hàm trọng và độ dốc về giá trị ban đầu

net - init(net);

Kiểm tra

wts -

0 0

bias =

0

Ta có thể thay đổi cách thức của perceptron được khởi tạo với **init**. Ví dụ, ta ó thể định nghĩa lại các hàm trọng đầu vào mạng và độ dốc **intFcn** như sự ngẫu nhiên và sau đó áp dụng inh như chỉ ra như sau:

net.inputweights{1,1}.initFcn = 'rands';

net.biases{1}.initFcn = 'rands';

net = init(net);

Kiểm tra hàm trọng và độ dốc

wts =

0 2309 0.5839

biases =

-0.1106

Ta thấy rằng hàm trọng và độ dốc được lấy các số ngẫu nhiên.

4.3. CÁC LUẬT HỌC

4.3.1. Khái niệm

Luật học là một thủ tục nhằm sửa đổi hàm trọng và độ dốc của mạng (thủ tục này cũng có thế coi như một thuật toán huấn luyện). Luật học được áp dụng để huấn luyện mạng thực hiện một vài nhiệm vụ cụ thể nào đó. Các luật học trong mạng được phân thành 2 loại: học có giám sát và học không có giám sát.

+ **Đối với học có giám sá**t các luật học được cung cấp cùng với một tập các ví dụ (tập huấn luyện) đặc trưng của mạng:

$$\{P_1,t_1\};\ \{p_2,t_2\}...\ ;\{p_Q,t_Q\}$$

Trong đó P_Q là đầu vào mạng vả từ đầu ra đáp ứng chính xác tương ứng (đích). Giống như các đầu vào áp dụng cho mạng, ở các đầu ra mạng được so sánh với đích. Luật học được sử dụng để điều chỉnh hàm trọng và độ dốc của mạng nhằm dịch chuyển dần các đầu ra mạng tiến dần đến đích.

+ **Đối với học không giám sát**, hàm trọng và độ dốc của mạng được thay

đổi tương ứng giá trị ở đầu vào mà không có sẵn đích ở đầu ra. Phần lớn các thuật toán này biểu diễn thành một tập bó. Người ta chia mẫu vào thành con số cụ thể của hạng (loại). Điều này đặc biệt hữu ích trong các ứng dụng cụ thể như một véc tơ lượng tử hoá. Trong chương này ta chỉ đề cập đến các thuật toán huấn luyện mạng perceptron theo kiểu học có giám sát. Trong Matlab người ta sử dụng 2 hàm để huấn luyện mạng là hàm **learnp** và hàm **train**.

4.3.2. Luật học Perceptron (learnp)

Perceptron được huấn luyện theo mẫu mong muốn cho trước. Mẫu mong muốn có thể tập hợp thành một tập các cặp đầu vào, đầu ra:

$$P_1, t_1; p_2, t_2;...; p_Q, t_Q$$

trong đó: p là đầu vào mạng, t là đáp ứng tương ứng ở đầu ra. Mục đích là giảm sai lệch e giữa đáp ứng của nơron và hàm mục tiêu **t (e = t - a)**. Luật học perceptron (leranp) tính trước sự thay đổi mong muốn đối với hàm trọng và độ dốc gửi tới véc tơ vào p và kết hợp với sai lệch e. Do véc tơ đích t chỉ có thể có giá trị 0 hoặc 1, đối với nơron perceptron (với các hàm truyền dạng hardlim) đầu ra chỉ có thể có 1 trong 2 giá trị. Mỗi lần lệnh **learnp** được thực hiện, mạng sẽ có một giá trị kết quả đầu ra chính xác hơn. Luật học perceptron sẽ hội tụ đến kết quả cuối cùng sau một số hữu hạn các lần lặp nếu như có tồn tại đáp án. Nếu không sử dụng độ dốc, learnp làm việc để tìm đáp án bằng việc thay đổi véc tơ trọng liên kết W để chỉ rõ véc tơ đầu vào thuộc lớp 1 hay lớp 0. Kết quả này dùng để quyết định đường biên giới là đường trực giao với W và nó phân loại chính xác véc tơ vào.

Trong quá trình huấn luyện mạng có thể xảy ra 3 trường hợp mỗi khi véc tơ đầu vào (**p**) xuất hiện và đáp ứng mạng (**a**) được tính toán:

+ Trường hợp 1: nếu véc tơ đầu vào xuất hiện và đáp ứng đầu ra của nó là đúng (**a = t, và e = t - a = 0**) thì véc tơ hàm trọng W không thay đổi.

+ Trường hợp 2: nếu đầu ra của nơron bằng 0 còn trước đó bằng 1 (**a = 0; t = 1** và **e = t - a = 1**) véc tơ đầu vào P làm tăng véc tơ hàm trọng **W**. Điều này làm cho véc tơ trọng tiến gần tới véc tơ vào, dần dần khả năng véc tơ vào sẽ được phân loại a = 1 trong tương lai.

+ Trường hợp 3: nếu đầu ra của nơron bằng 1 trước đó có thể là 0 (**a = 1; t = 0** và **e = t - a = -l**) véc tơ vào P được trừ đi véc tơ hàm trọng W. Điều đó làm cho véc tơ trọng ngày càng xa véc tơ vào, dần dần véc tơ vào được phân loại a = 0 trong tương lai.

Luật học perceptron có thể được viết cô đọng trong mối quan hệ của sai lệch e = t - a và sự thay đổi của véc tơ trọng $\Delta\mathbf{W}$ như sau:

Trường hợp 1: Nếu **e** = 0 thì sự thay đổi giá trị của $\Delta\mathbf{W}$ bằng 0.

Trường hợp 2: Nếu **e** = 1 thì sự thay đổi giá trị của **ΔW** bằng 0.

Trường hợp 3: Nếu **e** = -1 thì sự thay đổi giá trị của **ΔW** bằng 0 cả 3 trường hợp trên có thể viết dưới dạng biểu thức đơn giản:

$$\Delta W = (t - a)p^T = ep^T.$$

Ta có thể nhận được biểu thức để thay đổi độ dốc của nơron với chú ý rằng, độ dốc chỉ đơn giản là hàm trọng có đầu vào là 1:

$$\Delta b = (t - a).(1) = e.$$

Đối với trường hợp của một lớp nơron ta có:

$$\Delta W = (t - a)PT = ep^T \text{ và}$$

$$\Delta b = (t - a).(1) = e.$$

Luật học perceptron có thể tóm tắt như sau:

$$W^{mới} = W^{cũ} + eP^T \text{ và}$$

$$b^{mới} = b^{cũ} + e \text{ Trongđó } e = t\text{-}a.$$

Ví dụ: xét nơron đơn giản có véc tơ vào với 2 phần tử

net = newp([-2 2;-2 +2],1);

Để đơn giản ta thiết lập độ dốc bằng 0, các hàm trọng là 1 và 0,8.

net.b{1} = [0];

w = [1 -0 8];

net.IW{1,1} = w;

Cặp vào đích được xác định bởi:

p = [t; 2];

t = [1];

Ta có thể tính toán đầu ra và sai lệch với các lệnh:

a = sim(net,p)

Kết quả: **a = 0**

e = t- a = 1

Và cuối cùng sử dụng hàm learnp để nhận được sự thay đổi hàm trọng.

dw - learnp(w, p, [ ], [ ], [ ], [ ], e, [ ], [ ], [ ])

dw : 1 2

Hàm trọng mới thu được là:

W = W + dW

W = 2.0000 1.2000

Quá trình tìm hàm trọng mới (và các độ dốc mới) có thể được lặp đi lặp lại cho đến khi không còn sai lệch.

Chú ý: Luật học perceptron đảm bảo để hội tụ sau một số hữu hạn các bước của tất cả các bài toán có thể được giải quyết bằng perceptron. Nó bao hàm tất cả các bài toán phân loại "tách rời tuyến tính" (linearly separable). Các đối tượng để phân loại trong mọi trường hợp đều có thể cách li bằng đường đơn.

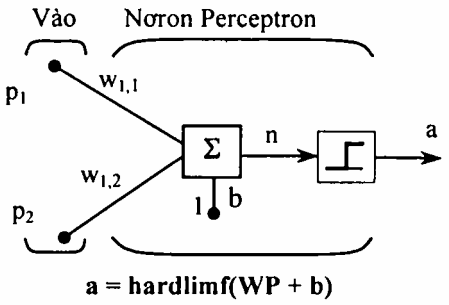
4.3.3. Huấn luyện mạng (train)

Lệnh **sim** và hàm **learnp** được sử dụng lặp để đưa các dữ liều đầu vào đến perceptron và để thay đổi hàm trọng và độ dốc của perceptron theo sai lệch giữa đầu ra của mạng và đích, perceptron sẽ tìm được hàm trọng và độ dốc cuối cùng để giải quyết bài toán đặt ra (với điều kiện bài toán đó perceptron có thể giải quyết được) Mỗi đường ngang xuyên qua tất cả các đầu vào huấn luyện và véc tơ đích được gọi bằng lệnh pass.

Hàm **train** đưa ra như vòng lặp của sự tính toán. Trong mỗi một pass hàm trình thu được thông qua chuỗi các tín hiệu vào, tính toán đầu ra, sai số và điều chỉnh mạng để mỗi véc tơ tín hiệu vào trong chuỗi giống như các đầu vào đã có.

Chú ý rằng huấn luyện trình không đảm bảo kết quả mạng làm được công việc của nó. Giá trị mới của W và b cần được **kiểm tra** bằng việc tính toán đầu ra mạng theo mỗi véc tơ vào để thấy được nếu tất cả các đích đã đạt được. Nếu mạng thực hiện không thành công, nó cần được huấn luyện thểm nữa bằng cách gọi lại train với hàm trọng và độ dốc mới cho các lần huấn luyện thểm, hoặc có thể phân tích để thấy rằng bài toán không phù hợp cho perceptron. (Các bài toán không thể giải được bằng mạng perceptron được trình bày trong mục cuối của chương này).

Để minh hoạ cho thủ tục huấn luyện **train**, ta xét một nơron perceptron với véc tơ vào có 2 phần tử như hình vẽ. Đây là một mạng rất đơn giản cho phép ta có thể thực hiện tính toán bằng tay (nếu cần).



Hình 4.5. Nơron với 2 đầu vào

Giả thiết ta sử dụng mạng trên để giải quyết bài toán phân loại với các cặp véc tơ đích như sau:

$$\left\{P_1 = \begin{bmatrix}2\\2\end{bmatrix}, t_1 = 0\right\}\left\{P_2 = \begin{bmatrix}1\\-2\end{bmatrix}, t_2 = 1\right\}\left\{P_3 = \begin{bmatrix}-2\\2\end{bmatrix}, t_3 = 0\right\}\left\{P_4 = \begin{bmatrix}-1\\1\end{bmatrix}, t_4 = 1\right\}.$$

Sử dụng hàm trọng và độ dốc ban đầu. Ta biểu diễn các biến ở mỗi bước tính bằng cách sử dụng con số trong dấu ngoặc đơn sau biến. Theo cách đó ta có giá trị ban đầu là W(0) và b(0).

$$W(0) = [0\ 0]; \quad b(0) = 0$$

Ta bắt đầu bằng việc tính đầu ra của perceptron a đối với đầu vào thứ nhất là véc tơ p_1, sử dụng điều kiện về hàm trọng và độ dốc,

$$a = \text{hardlim}(W(0)\mathbf{p_1} + b(0)) = \text{hard lim}\left([0\ 0]\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = \text{hard lim}(0) = 1.$$

Đầu ra a không bằng giá trị đích t_1, vì vậy ta sử dụng luật perceptron để tìm sự thay đổi của hàm trọng và độ dốc trên cơ sở của sai lệch.

$$\mathbf{e = t1 - a = 0 - 1 = -1}$$

$$\mathbf{\Delta W = e\ P_1^T = (-1)[2\ 2] - [-2\ -2]}$$

$$\mathbf{\Delta b = e = (-1) = -1.}$$

Ta có thể tính hàm trọng và độ dốc mới nhờ sử dụng các quy tắc cập nhật perceptron đã chỉ ra trước đây:

$$\mathbf{W^{mới} = W^{cũ} + eP^T = [0\ \ 0] + [-2\ -2] = [-2\ -2\ ] = w(1)}$$

$$\mathbf{b^{mới} = b^{cũ} + e = 0 + (-1) = -1 = b(1).}$$

Với véc tơ vào P_2 tiếp theo, đầu ra được tính:

$$a = \text{hardlim}(W(1)\mathbf{p_2} + b(1)) = \text{hard lim}\left([-2\ \ -2]\begin{bmatrix}-2\\-2\end{bmatrix} - 1\right) = \text{hard lim}(1) = 1.$$

Trong trường hợp này đích là 1 vì vậy sai lệch bằng 0. Do không có sự thay đổi hàm trọng và độ dốc:

$$W(2) = W(1) = [-2\ -2] \text{ và } b(2) = b(1) = -1.$$

Tương tự ta tiếp tục xem xét sự có mặt của P3' tính toán đầu ra, sai lệch và tìm sự thay đổi của hàm trọng và độ dốc... Sau khi thực hiện một lượt qua cả 4 giá trị vào, ta nhận được:

$$W(4) = [-3\ -1] \text{ và } b(4) = 0.$$

Để kết thúc khi đã thu được đáp án thỏa đáng, ta cần phải làm một lượt

qua tất cả các véc tơ vào để thấy được kết quả của tất cả các giá trị đích mong muốn. Điều này không đúng cho đầu vào thứ 4, nhưng thuật toán hội tụ trong lần thứ 6. Giá trị cuối cùng là:

$$W(6) = [-2\ -3] \text{ và } b(6) = 1.$$

Đến đây kết thúc sự tính toán bằng tay. Bây giờ ta cần làm thế nào để sử dụng hàm huấn luyện? Theo mã định nghĩa perceptron như đã chỉ ra trên hình vẽ trước, với giá trị ban đầu của hàm trọng và độ dốc bằng 0, ta có:

net = newp(l-2 2;-2 +2],1);

Quan sát giá trị của đầu vào đơn.

p = [2; 2];

ta có đích

t = (0);

Đặt kỳ huấn luyện epochs = 1, như vậy **train** sẽ đi qua các véc tơ vào ở một lần.

net.trainparam.epochs = 1;

net = train(net,p,t);

Hàm trọng mới và độ dốc mới là:

w =

-2 -2

b =

- 1

Vậy với giá trị ban đầu của hàm trọng và độ dốc = 0, sau khi huấn luyện với chỉ véc tơ thứ nhất, chúng có giá trị [-2 -2] và -1 giống như khi ta tính bằng tay. Bây giờ áp dụng cho véc tơ vào thứ 2 (p_2). Đầu ra là 1, hàm trọng và độ dốc sẽ được thay đổi, nhưng bây giờ đích là 1, sai lệch sẽ bằng 0 nên sự thay đổi sẽ bằng 0. Ta có thể đi theo cách này, bắt đầu từ kết quả trước và áp dụng véc tơ đầu vào mới ở các lần sau. Tuy nhiên ta có thể làm công việc đó một cách tự động với hàm **train**. Sau đây ta sẽ áp dụng hàm **train** cho một khóa huấn luyện từng đầu vào lần lượt thông qua chuỗi của tất cả 4 véc tơ vào. Đầu tiên ta định nghĩa mạng:

net : newp([-2 2;-2 +2[,1);

net.trainParam.epochs = 1;

Các véc tơ vào và đích là:

p = [[2;2] [1;-2] 1-2;2] [-1;1]]

t - [0 1 0 1]

Để huấn luyện ta sử dụng:

net = train(net,p,t);

Hàm trọng và độ dốc mới là:

w =

-3 -1

b =

0

Kết quả này tương tự như kết quả ta đã tính bằng tay trước đây. Mô phỏng cuối cùng sự huấn luyện mạng cho mỗi đầu vào là:

a = sim(net,p)

a =

[0] [0] [1] [1]

Đầu ra mạng không bằng giá trị đích. Vì vậy cần huấn luyện mạng thểm một số lần nữa. Ta sẽ thử 4 khóa huấn luyện. Các kết quả cho ra như sau:

TRAINC, Epoch 0/20

TRAINC, Epoch 3/20

TRAINC, Performance goal met.

Như vậy, mạng đã được huấn luyện vào lúc các đầu vào có mặt trong 3 khóa (Như đã biết từ việc tính bằng tay, mạng hội tụ với sự xuất hiện của véc tơ vào thứ 6. Điều này xuất hiện ở giữa của khóa 2 nhưng đến khóa huấn luyện thứ 3 ta mới nhận ra sự hội tụ của mạng). Hàm trọng và độ dốc cuối cùng là:

w =

-2 -3

b =

1

Kết quả mô phỏng ở đầu ra và sai số của các đầu vào riêng biệt là:

a =

0 1.00 0 1.00

error = [a(1) – t(1) a(2) - t(2) a(3) - t(3) a(4) - t(4)]

error =

0 0 0 0

Vậy ta thấy rằng thủ tục huấn luyện đã thành công. Mạng hội tụ và kết quả đúng với đích đầu ra của 4 véc tơ đầu vào.

Chú ý: Hàm huấn luyện mặc định của mạng được thiết lập với lệnh newp là **train**s (bạn đọc có thể tìm hiểu thêm bằng cách gõ lệnh **net.trainFcn** từ cửa sổ lệnh của Matlab). Hàm huấn luyện này áp dụng cho các luật học perceptron dưới dạng thuần tuý. Trong đó, mỗi thành viên của véc tơ vào được áp dụng riêng lẻ thành chuỗi và sự hiệu chỉnh hàm trọng và độ dốc được tiến hành sau mỗi lần xuất hiện của 1 véc tơ vào. Vậy, huấn luyện perceptron với hàm trình sẽ hội tụ ở một số hữu hạn các bước, ngoại trừ bài toán không thể giải quyết được với perceptron đơn giản.

Hàm trạm có thể được sử dụng trong các trường hợp khác nhau cho các mạng khác đều cho kết quả tốt.

4.4. CÁC HẠN CHẾ CỦA PERCEPTRON

Mạng perceptron có thể được huấn luyện với hàm **Adapt**, nó đưa lần lượt các véc tơ vào đến mạng và tiến hành hiệu chỉnh mạng dựa trên kết quả của mỗi lần thực hiện. Sử dụng **Adapt** đảm bảo rằng một bài toán độc lập tuyến tính bất kỳ sẽ được giải quyết trong một số hữu hạn các bước huấn luyện. Perceptron cũng có thể được huấn luyện với hàm **train**. Khi trình được sử dụng cho perceptron, nó gửi véc tơ vào đến mạng theo gói và tiến hành hiệu chỉnh mạng trên cơ sở tổng của tất cả các hiệu chỉnh thành phần. Tuy nhiên đến nay ta chưa chứng minh được sự hội tụ thuật toán huấn luyện của perceptron.

Mạng perceptron có một vài hạn chế sau:

- Đầu ra của perceptron chỉ có thể nhận 1 trong 2 giá trị 0 hoặc 1 do hàm chuyển **hard-limit**.

- Perceptron chỉ có thể phân loại, cho tập các véc tơ độc lập tuyến tính. Nếu là đường thẳng hoặc mặt phẳng ta có thể vẽ để tách rời các véc tơ vào thành các loại chính xác, các véc tơ vào là độc lập tuyến tính. Nếu các véc tơ vào không độc lập tuyến tính sử học sẽ không bao giờ đạt tới mức tất cả các véc tơ được phân loại chính xác. Tuy nhiên điều đó cũng chứng minh rằng nếu các véc tơ là độc lập tuyến tính, perceptron huấn luyện thích nghi sẽ luôn tìm được đáp án trong thời gian hữu hạn. Ta cũng có thể sử dụng nhiều nơron perceptron có thể được để giải quyết các bài toán phức tạp hơn.

Ví dụ: Giả thiết có tập 4 véc tơ ta cần phân chia chúng thành các nhóm

riêng biệt với 2 đường thẳng được vẽ để tách rời chúng. Khi đó, ta có thể sử dụng mạng 2 nơron perceprton, mạng được thiết lập sao cho 2 đường biên giới của nó phân chia đầu vào thành 4 loại. *(Bạn đọc có thể đọc [HDB1996] để hiểu thểm về perceptron và các bài toán perceptron phức tạp)*.

Những sự bất thường và luật pcrceptron mở rộng

Thời gian huấn luyện dài có thể do sự hiện diện của véc tơ vào bên ngoài, chúng có kích thước quá rộng hoặc quá nhỏ so với các véc tơ vào khác. Việc áp dụng luật học perceptron bao gồm việc cộng hay trừ véc tơ vào dựa vào hàm trọng và độ dốc hiện thời ở đáp ứng đến sai số. Do vậy một véc tơ vào với phần tử lớn có thể làm cho sự thay đổi hàm trọng và độ dốc lâu hơn nhiều lần véc tơ vào nhỏ.

Bằng việc thay đổi luật học perceptron chút ít, thời gian huấn luyện có thể thích hợp cho các véc tơ vào rất lớn hoặc rất nhỏ.

Luật gốc để cập nhật hàm trọng là:

$$\Delta W = (t-a)p^T = ep^T$$

Như đã chỉ ra ở trên, độ rộng của véc tơ vào **p**, có tác động lên véc tơ hàm trọng **W.** Do vậy, nếu một véc tơ vào lớn hơn nhiều so với các véc tơ vào khác, các véc tơ vào nhỏ chỉ cần một thời gian ngắn để có kết quả. Để khắc phục nhược điểm này, ta đưa ra luật học mở rộng. Khi đó, tác động của mỗi véc tơ vào lên hàm trọng:

$$\Delta W = (t-a)\frac{p^T}{\|p\|} = e\frac{p^T}{\|p\|}.$$

Luật perceptron mở rộng được thực hiện nhờ hàm **learnp**. Nó làm giảm bớt thời gian thực hiện nhưng không làm giảm số lần huấn luyện một cách đáng kể nếu có véc tơ vào bất thường (outlier).

4.5. SỬ DỤNG GIAO DIỆN ĐỒ HỌA ĐỂ KHẢO SÁT MẠNG NƠRON

(Graphical User Interface - GUI)

4.5.1. Giới thiệu về GUI

Giao diện đồ họa (Graphical User Interface - GUI) được thiết kế để đơn giản và thuận tiện cho mgười sử dụng. Cửa sổ giao diện đồ họa có một vùng làm việc của nó tách rời khỏi các dòng lệnh của vùng làm việc. Vì vậy khi sử dụng GUI ta cần phải xuất kết quả GUI sang (dòng lệnh) vùng làm việc. Tương tự ta có thể nhận kết quả từ dòng lệnh làm việc đến GUI.

Mỗi lần Network/Data Manager được đưa ra và chạy, ta có thể thiết lập mạng, quan sát, huấn luyện, mô phỏng nó và cất kết quả cuối cùng vào vùng

làm việc.

Tương tự, ta có thể lấy dữ liệu từ vùng làm việc để sử dụng trong GUI.

Ví dụ sau đây với mạng perceptron, ta sẽ đi qua tất cả các bước để thiết lập mạng và chỉ rõ ta có thể làm gì để được những điều mong muốn.

4.5.2. Thiết lập mạng Perceptron (nntool)

Giả thiết cần thiết lập mạng perceptron thực hiện cổng logic AND. Nó có véc tơ vào:

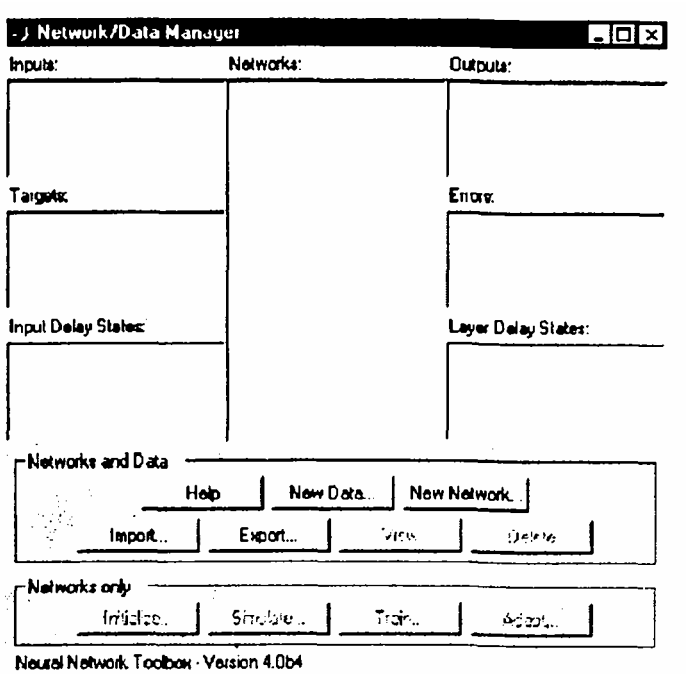
p = [0 0 1 1; 0 1 0 1] và véc tơ đích là:

t - [0 0 0 1]

Ta gọi mạng là **ANDNet**. Một lần thiệt lập mạng sẽ dược huấn luyện. Sau đó ta có thể cất mạng, đầu ra của nó; v.v... bằng lệnh "exporting" trong cửa sổ dòng lệnh.

a/ Thiết lập các giá trị vào - ra

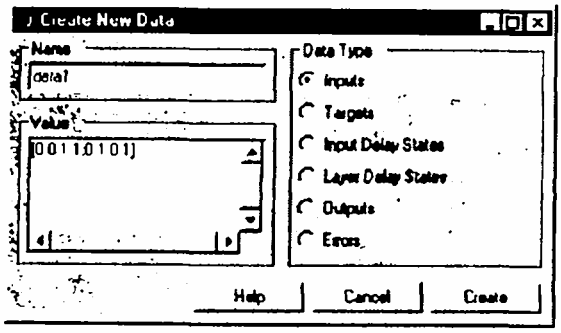
Để bắt đầu ta gõ nntool, xuất hiện cửa sổ hình 4.6.



Hình 4.6

Kích vào **help** để bắt đầu vào bài toán mới và để thấy ý nghĩa của các nút. Trước tiên, để định nghĩa đầu vào mạng ta gọi p, có giá trị cụ thể [0 0 1 0; 0 1 0 1].

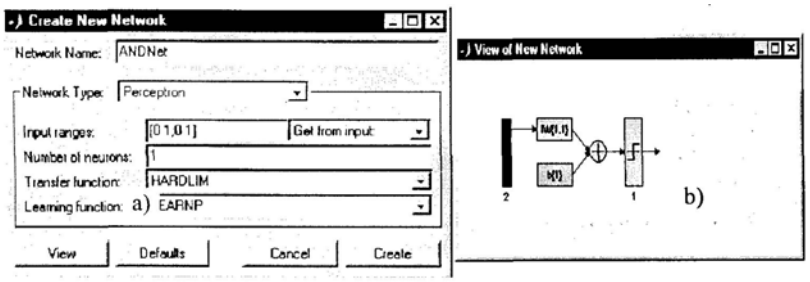
Như vậy, mạng có 2 phần tử vào và 4 tập của 2 phần tử véc tơ đó được đưa đến để huấn luyện. Để đinh nghĩa dữ liệu này, kích vào **new data** sẽ xuất hiện cửa sổ **Create New Data**. Đặt tên cho p, giá trị là **[0 0 1 1; 0 1 0 -1]** và xác định kiểu dữ liệu (data type) là tập dữ liệu vào (inputs). Cửa sổ thiết lập dữ liệu mới như hình 4.7. Bây giờ kịch **Create** để thiết lập **file** đầu vào p. Cửa sổ **Network/Data Manager** hiện lên và p chỉ rõ là đầu vào. Tiếp theo ta thiết lập đích của mạng. Kích **new data** một lần nữa rồi đưa vào biến t với giá trị [0 0 0 1], sau đó kích target để ấn định kiểu dữ liệu. Sau đó lại kịch Create ta sẽ thấy cửa sổ **Network/Data Mangaer** xuất hiện với t là đích và p là các đầu vào.



Hình 4.7

b. Thiết lập mạng

Giả thiết ta muốn thiết lập mạng mới có tên là **ANDNet**. Để làm điều đó ta kích **New Network**. cửa sổ **CreateNew Network** xuất hiện với tên **ANDNet** trong khung **Network Name** (hình 4.8), thiết lập kiêu mạng **Netword Type** là **Perceptron**, khi đó kiểu mạng ta mong muốn được thiết lập. Phạm vi đầu vào có thể được cài dặt bằng con số trong vùng đó. song ta cũng rất dễ đàng nhận được chúng từ 1 đầu dữ liệu riêng biệt ta cần sử dụng. Để làm điều này ta kích vào mũi lên di xuống ở phần bên phải của phạm vi đầu vào (Input Range) menu này trải xuống chỉ ra rằng ta có thể có được phạm vi đầu vào từ file p nếu ta muốn. Nếu kích vào p phạm vi đầu vào sẽ là [0 1 ; 0 1].



Hình 4.8a,b

Hình 4.8a, b

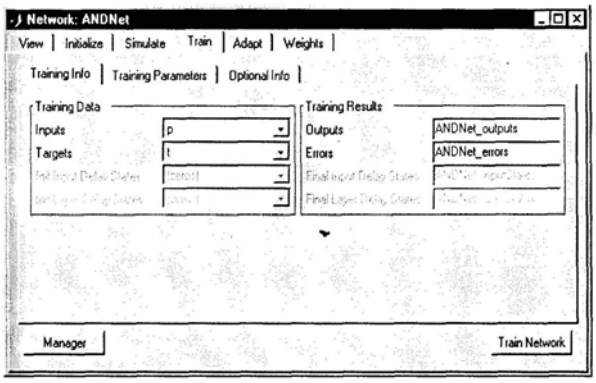
Ta chọn **hardlim** trong menu hàm chuyển **transfer function** và **learnp** trong menu hàm học **learning function**. Đến đây ta có cửa sổ Create New Netword như hình 4.8a.Ta có thể quan sát cấu trúc mạng bằng cách kích vào **Wiew** (hifnh 4.8b).

Như vậy ta đã thiết lập được một mạng nơron đầu vào đơn (bao gồm 2 phần tử) hàm chuyển hardlim và 1 đầu ra. Đó là mạng perceptron ta mong muốn.

Bây giờ kích vào Create để mọi ra mạng vừa thiết lập, ta sẽ nhận được cửa sổ **Netword/Data Manager**. Chú ý rằng **ANDNet** bây giờ được liệt kê như một mạng (hình 4.9).

4.5.3. Huấn luyện mạng

Để huấn luyện mạng ta kích vào **ANDNet** để mở chúng, sau đó kích vào **Train**, xuất hiện cửa sổ mới với nhãn: **Netword:ANDNet**. Ở đây ta có thể nhìn lại mạng bằng cách kích vào **Train**. Để kiểm tra điều kiện đầu ta kích vào nhãn **Initialize**. Bây giờ ấn vào nhãn **Train**, định rõ đầu vào, đầu ra bằng cách kích vào nhăn **Training Info**, chọn P trong hộp thoại **Inputs** và t trong hộp thoại **targets**. Khi đó cửa sổ **Netword:ANDNet** như hình 4.9.

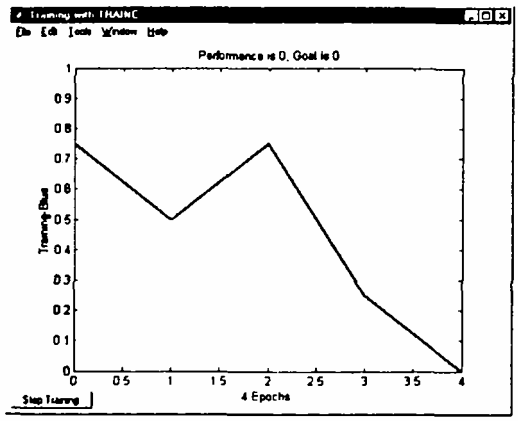


Hình 4.9

Chú ý rằng kết quả huấn luyện của các đầu ra và sai số có ở **ANDNet** gắn vào chúng. Việc làm này của chúng dễ dàng nhận ra sau khi chúng được đưa ra từ dòng lệnh.

Sau khi kích vào nhãn **Training Parameter**, nó cho ta biết các thông số như số lần huấn luyện, sai số đích. Ta có thể thay đổi các thông số này nếu ta muốn.

Kích chuột vào **Train Network** để huấn luyện mạng pcrceptron, ta được kết quả như hình 4.10.



Hình 4.10

Vậy, mạng đã được huấn luyện dễ sai lệch bằng 0 ở chu kỳ thứ 4 (chú ý rằng các dạng mạng khác thường không thể huấn luyện để được sai lệch bằng 0 mà sai lệch của chúng thường bao hàm trong một phạm vi rộng. Theo bản miêu tả đó chúng ta vẽ đồ thị sai lệch của chúng trên tọa độ loga đúng hơn trên tọa độ tuyến tính ví dụ nó đã dùng ở trên cho mạng perceptron).

Ta có thể **kiểm tra** răng mạng được huấn luyện cho sai lệch bằng 0 bằng việc sử dụng đầu vào p và mô phỏng mạng. Để làm điều này, ta vào cửa sổ **Network/Data Manager** và kích vào **Network Only**: **Simulate**, khi đó xuất hiện **cửa sổ**. **Netword: ANDNet** kích vào Simulate. Lúc này menu **Input pull-down** trải xuống chỉ rõ p là đầu vào và nhãn ra là **ANDNet_outputsSim** để phân biệt nó từ đầu ra huấn luyện. Kích vào **Simulate Network** ở góc dưới bên phải, quan sát **Network/Data Manager** ta sẽ thấy giá trị mới của đầu ra: **ANDNet_outputsSim**. Kich đúp vào nó, một cửa sổ dữ liệu nhỏ: **ANDnet_outputsSim** mở ra với trị số [0 0 0 1].

Vậy, mạng thực hiện cổng logic AND các đầu vào, nó cho ra giá trị 1 ở đầu ra chỉ trong trường hợp cuối cùng, khi cả 2 đầu vào là 1.

4.5.4. Xuất kết quả Perceptron ra vùng làm việc

Để xuất các đầu ra và sai số của mạng ra cửa sổ vùng làm việc của MATLAB, ta kích vào nút thấp hơn bên trái của cửa sổ **Network:ANDNet** để đi đến phần sau **Network/Data Manager**. Chú ý đầu ra và sai số của **ANDNet** được liệt kê trong bản liệt kê các đầu ra và sai số (**Outputs and Error**) ở phần bên phải. Kích tiếp Export ta được cửa sổ **Export** hoặc **Save from Network/Data Magager**. Kích vào **ANDNet_outputs** và **ANDNet-crrors** để làm nổi rõ chúng, sau đó kích vào nút **Export**. Bây giờ 2 biến đó có thể có ở vùng làm việc dòng lệnh. Để kiểm tra điều này, từ cửa sổ lệnh ta gõ **who** để thấy tất cả các biến đã định nghĩa. Kết quả như sau:

who

Các biến là:

ANDNet_errors ANDNet outputs

Ta có thể gô **ANDNe_toutputs** và **ANDNet_errors** để nhận được kết quả sau: **ANDNet_outputs** =

0 0 0 1

and ANDNet_errors =

0 0 0 0

Ta có thể xuất p, t và **ANDnet** ra đường mô phỏng. Ta có thể làm điều này và kiểm tra lại với lệnh **who** để chắc chắn rằng chúng có ở cửa sổ lệnh.

Bây giờ **ANDNet** đó được xuất ra ta có thể nhìn được mô tả mạng và khảo sát ma trận trọng của mạng. Ví dụ:

ANDNet.iw{1,1}

gives ans =

2 1

Similarly, ANDNet.b{1} yiclds ans =

4.5.5. Xoá cửa sổ dữ liệu mạng (Network/Data Window)

Ta có thể xoá cử sổ dữ liệu mạng bằng cách làm sáng biến (ví dụ p) rồi kích nút **Delete** cho tới khi tất cả các mục trong hộp liệt kê biến mất, bằng cách làm này, chúng ta bắt đầu từ việc xoá danh sách.

Một cách khác là ta có thể thoát MATLAB, khởi động lại MATLAB, đi vào nntool được cửa sổ **Netword Data Manager** đã xoá.

Tuy nhiên việc gọi lại những dữ liệu ta đã xuất ra cửa sổ dòng lệnh như p, t... từ ví dụ perceptron, chúng không thay đổi khi ta xoá **Netword/Data Manager.**

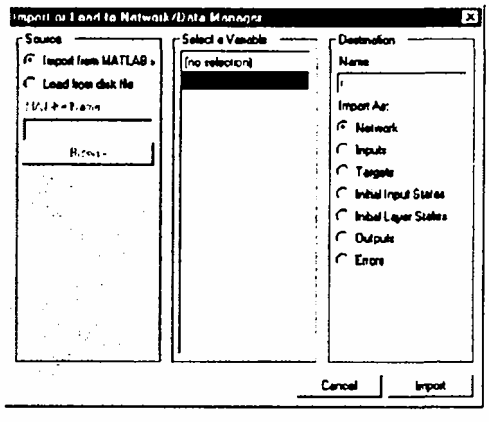
4.5.6 Nhập từ dòng lệnh

Đề đơn giản, ta thoát khỏi MATLAB, khởi động lại lần nữa và gõ bệnh nntool đề bắt đầu một trang mới.

Thiết lập véc tơ mới:

r = [0; 1; 2; 3]

r =



Hình 4.11

0

1

2

3

Kích vào Import và đặt tên nơi gửi đến là R (để phân biệt với tên biến từ dòng lệnh và biến trong GUI). Ta sẽ có cửa sổ như hình 4.11.

Bây giờ kích vào **Import** và kiểm tra lại bằng cách nhìn vào **Network/Data Manager** để thấy biến R như là một đầu vào.

4.5.7. Cất biến vào file và nạp lại nó

Đưa ra **Network/Data Manager** và kích vào **New Netword** dặt tên cho mạng là **mynet**. Kích vào **Create**, tên mạng **mynet** có thể xuất hiện trong cửa sổ **Network/Data Manager**. Tương tự như cửa sổ **Manager** kích vào **Export**. Chọn **mynet** trong danh sách biến của cửa sổ **Export or Saye** và kích vào **Save**. Các hướng dẫn này để cất vào cửa sổ **Save to a MAT file**. Cất **file mynetfile.**

Bây giờ, rời khỏi **mynet** trong GUI và tìm lại nó từ **file** đã cất. Đầu tiên, chuyển đến **Data/Netword Manager**, **mynet** nổi lên và kích vào **Delete**. Sau đó kích vào **Import**, cửa sổ **Import or Load to Network/Data Manager** mở ra. Chọn nút **Load from Disk** và gõ **mynetfile** như ở **MAT-file Name**. Bây giờ kích vào **Browse** để mở ra cửa sổ **Select MAT file** với file **mynetfile** như một sự lựa chọn rằng ta có thể chọn như là một biến để nhập. **Mynetfile** nổi lên, ấn vào **Open** và ta trở về cửa sổ **Import or Load to Netword/Data Manager**. Trong danh sách **Import As**, chọn **Netwrork, mynet** nổi lên và kích vào **Load** để đưa mynet đến GUI. Bây giờ ta đã có ở trong của cửa sổ GUI **Netword/Data Manager.**

Chương 5

MẠNG TUYẾN TÍNH

5.1. MỞ ĐẦU

5.1.1. Khái niệm

Mạng tuyến tính có cấu trúc tương tự như mạng perceptron, nhưng hàm chuyển của nó là hàm tuyến tính (khác với hàm chuyển **hard-limiting** của perceptron). Vì vậy cho phép đầu ra của mạng nhận được giá trị bất kỳ, trong khi đó đầu ra của perceptron chỉ nhận giá trị 0 hoặc 1.

Khi đưa vào mạng tuyến tính một tập véc tơ vào nó sẽ đưa ra vcc tơ đáp ứng tương ứng. Đối với mỗi véc tơ vào, ta có thể tính toán véc tơ ra của mạng. Sự sai khác giữa véc tơ vào và véc tơ đích của nó là sai lệch. Ta có thể tìm giá trị của hàm trọng và độ dốc sao cho tổng của các bình phương sai lệch là cực tiểu hoặc nhỏ hơn một giá trị xác định nào đó. Điều này hoàn toàn có thể làm được bởi vì hệ tuyến tính có sai lệch đơn cực tiểu. Trong đa số các trường hợp, ta có thể tính toán mạng tuyến tính một cách trực tiếp sao cho sai lệch là các tiêu đôi với các véc tơ vào và véc tơ đích định sẵn. Một số trường hợp khác các bài toán số không cho phép tính trực tiếp. Tuy nhiên, ta luôn luôn có thể huấn luyện mạng để có sai lệch cực tiểu bằng việc sử dụng thuật toán bình phương trung bình nhỏ nhất (Widrow-Hoff).

Trong chương này, Sau khi tìm hiểu cấu trúc mạng lọc tuyến tính, chúng ta sẽ tìm hiểu 2 hàm sử dụng trong Matlab: Hàm **Newlin** dùng để thiết lập lớp mạng tuyến tính và hàm **newlind** dùng để thiết kế lớp tuyến tính cho một mục đích cụ thể.

5.1.2. Mô hình nơron

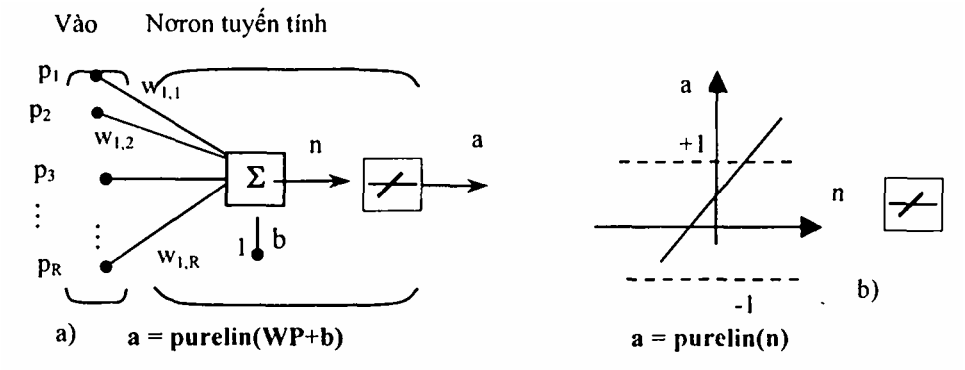
Một nơron tuyến tính với R đầu vào được chi ra trên hình 5.1. Mạng tuyến tính có cấu trúc cơ bản tương tự như perceptron, chỉ có điểm khác là ở đây dùng dùng hàm chuyển tuyến tính, ta gọi nó là hàm **purelin**. Hàm chuyển tuyến tính tính toán đầu ra của nơron bằng cách điều chỉnh giá trị đưa vào:

$$\mathbf{a = purelin(n) = purelin(Wp + b) = Wp + b.}$$

Nơron này có thể được huấn luyện để học một hàm xác định ở đầu ra hoặc để xấp xỉ tuyến tính một hàm phi tuyến.

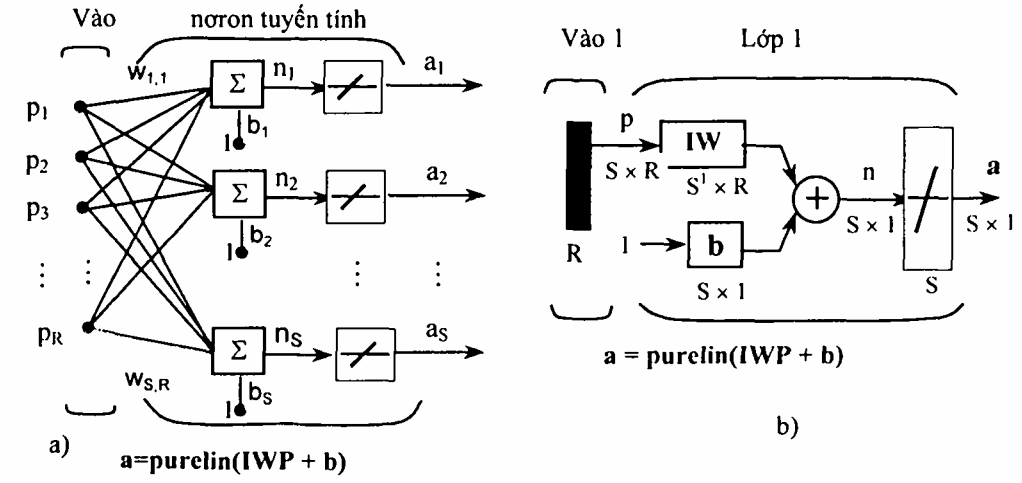
Mạng tuyến tính đương nhiên không phù hợp bác thực hiện các tính toán

cho hệ phi tuyến.



Hình 5.1a,b. Nơron với R đầu vào

a) Mô hình nơron, b) Hàm chuyển tuyến tính



Hình 5.2a,b. Kiến trúc một lớp mạng tuyến tính

a) Kiến trúc đầy đủ, b) Ký hiệu tắt

5.2. CẤU TRÚC MẠNG

5.2.1. Cấu trúc

Mạng tuyến tính như hình 5.2, có một lớp, S nơron liên hệ với R đầu vào thông qua ma trận trọng liên kết W. Trong sơ đồ S là độ dài của véc tơ đầu ra a.

Ta biểu diễn mạng tuyến tính lớp đơn, tuy nhiên mạng này cũng có năng lực như mạng tuyến tính nhiều lớp. Thay thế cho mỗi mạng tuyến tính nhiều lớp có mạng tuyến tính lớp đơn tương đương.

5.2.2. Khởi tạo nơron tuyến tính (Newlin)

Xét một nơron đơn giản với 2 đầu vào có sơ đồ như hình 5.3a. Ma trận trọng liên kết trong trường hợp này chỉ có 1 dòng. Đầu ra của mạng là:

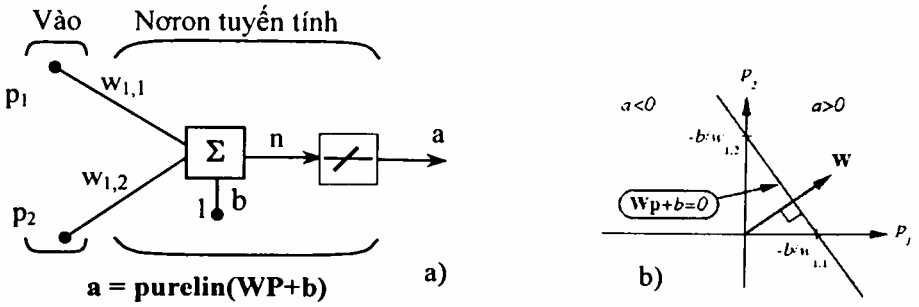
$$\mathbf{a = purelin(n) = purelin(wp + b) = Wp + b}$$

hoặc $\mathbf{a = w_{1,1}p_1 + w_{1,2}P_2 + b.}$

Giống như perccptron, mạng tuyến tính có đường phân chia biên giới dược xác định bằng các véc tơ vào đối với nó mạng vào **n** bằng 0. Để **n** - 0 thì biểu thức **Wp + b** = 0. Hình 5.3b chỉ rõ ví dụ về đường phân chia biên giới như sau: Các véc tơ vào phía trên, bên phải có mẫu sẫm sẽ dẫn đến đầu ra lớn hơn 0.

Các véc tơ vào phía dưới bên trái có mẫu sẫm sẽ dẫn đến đầu ra nhỏ hơn 0. Như vậy mạng tuyến tính có thể dùng để phân loại đối tượng thành 2 loại. Tuy nhiên nó chỉ có thể phân loại theo cách này nếu như đối tượng là tuyến tính tách rời. Như vậy mạng tuyến tính có hạn chế giống như mạng perceptron. Ta có thể khởi tạo mạng với lệnh:

net = Newlin([-1 1; -1 1],l);



Hình 5.3a,b. Nơron với 2 đầu vào

Ma trận thứ nhất của đối số chỉ rõ giới hạn của 2 đầu vào vô hướng. Đối số cuối cùng, '1' nói lên mạng có một đầu ra. Trong liên kết và độ dốc được thiết lập mặc định bằng 0. Ta có thể quan sát giá trị hiện thời của chúng với lệnh:

W = net.IW{1,1}

W =

0 0

và

b = net.b{1}

b =

0

Tuy nhiên ta có thể cho hàm trọng giá trị bất kỳ nếu ta muốn, chẳng hạn bằng 2 và 3 theo thứ tự định sẵn:

net.IW{1,1} = [2 31;

W = net.IW{1,1}

W =

Độ dốc cũng có thể cho trước và kiểm tra tương tự như vậy:

net.b{1} =[-41;

b = net.b{1}

b =

4

Ta có thể mô phỏng mạng tuyến tính đối với véc tơ vào cụ thể, ví dụ P = [5;6]; ta có thể tìm được đầu ra mạng với hàm sim.

a = sim(net,p)

a =

24

Tóm lại, ta có thể khởi tạo mạng tuyến tính với hàm **newlin,** điều chỉnh các phần tử của mạng nếu ta muốn và mô phỏng mạng với hàm **sim**.

5.3. THUẬT TOÁN CỰC TIỂU TRUNG BÌNH BÌNH PHƯƠNG SAI LỆCH

Giống như luật học perceptron, thuật toán cực tiểu trung bình bình phương sai lệch (LMS) được làm mẫu để giám sát huấn luyện mạng tuyên tính, trên chúng luật huấn luyện được chuẩn bị đầy đủ với tập mẫu các hành vi của mạng mong muốn:

$$\{p_1, t_1\}, \{P_2, t_2\},...,\{P_Q, t_Q\}$$

Trong đó P_q là đầu vào, t_q là đáp ứng đích ở đầu ra. Khi mới đầu vào được đưa tới mạng, đầu ra mạng được so sánh với đích. Sai số được tính toán như là hiệu giữa đích ra và đầu ra mạng. Ta muốn giá trị trung bình của tổng các sai số này đạt cực tiểu mse

$$\text{mse} = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 \quad = (t(k) - a(k))^2$$

Thuật toán các tiêu trung bình bình phương sai lệch sẽ điều chỉnh hàm

trọng và độ dốc của mạng tuyến tính sao cho giá trị trung bình bình phương sai số đạt cực tiểu.

Do chỉ số biểu diễn sai số trung bình bình phương là một hàm toàn phương nên chỉ số biểu diễn sẽ có một cực tiểu toàn cục, gần cực tiểu hoặc không cực tiểu tuỳ thuộc đặc điểm của véc tơ vào.

5.4. THIẾT KẾ HỆ TUYẾN TÍNH

Khác với các kiến trúc mạng khác, mạng tuyến tính có thể được thiết kế trực tiếp nếu ta đã biết từng cặp véc tơ vào/đích. Đặc biệt giá trị của hàm trọng và độ dốc mạng có thể thu được từ cực tiểu hóa trung bình bình phương sai lệch bằng cách sử dụng hàm **newlind**.

Giả thiết các đầu vào và đích của mạng là:

P = [1 2 3];

T= 12.0 4.1 5.9];

Để thể thiết kế mạng ta dùng lệnh:

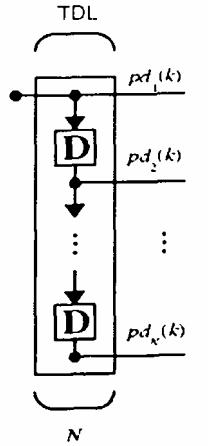
net = Newlind(p,T);

Ta có thể mô phỏng hành vi mạng để kiểm tra kết quả thiết kế bằng lệnh:

Y = sim(net,P)

Y =

2.0500 4.0000 5.9500



Hình 5.4

5.5. MẠNG TUYẾN TÍNH CÓ TRỄ

5.5.1 Mắt trễ

Ta cần một khâu mới là mắt trễ để tạo nên năng lực sử dụng đầy đủ cho mạng tuyến tính, ví dụ một mắt trễ được chỉ ra như sau, có một đầu vào tín hiệu đi vào từ trái và qua N-1 khâu trễ. Đầu ra của TDL là véc tơ kích thước N tạo ra từ tín hiệu vào ở thời điểm hiện tại, tín hiệu vào trước đó v.v...

5.5.2. Thuật toán LMS (learnwh)

Thuật toán LMS hay thuật toán học Widrow-Hoff được xây dựng dựa trên thủ tục hạ thấp độ dốc gần đúng. Ở đây, một lần nữa mạng tuyến tính được huấn luyện trên các mẫu của trạng thái chính xác.

Widrow và Hoff cho rằng họ có thể ước lượng sai số trung bình bình phương bằng việc sử dụng bình phương sai số ở mỗi lần tính lặp. Nếu ta lấy một phần đạo hàm của bình phương sai trọng và độ dốc ở lần lặp thứ k ta có:

lệch theo hàm

$$\frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} \quad \text{với } j = 1, 2, \ldots, R$$

và

$$\frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}$$

Sau đó quan sát phần đạo hàm đối với sai lệch:

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial[t(k) - a(k)]}{\partial w_{1,j}} = \frac{\partial}{\partial w_{i,j}}[t(k) - (Wp(k) + b)]$$

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial}{\partial w_{i,j}}\left[t(k) - \left(\sum_{i=1}^{R} w_{1,i}p_i(k) + b\right)\right]$$

trong đó $p_i(k)$ là phần tử thứ i của véc tơ vào trong lần lặp thứ k.

Tương tự:
$$\frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k)$$

điều đó có thể đơn giản hoá:

$$\frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k)$$

$$\frac{\partial e(k)}{\partial b} = -1.$$

cuối cùng sự thay đổi của ma trận trọng và độ dốc sẽ là:

$$\mathbf{2\alpha e(k)p(k) \text{ và } 2\alpha e(k)}$$

đây là 2 biểu thức dạng cơ bản của thuật toán học Widrow-Hoff (LMS). Kết quả trên có thể mở rộng cho trường hợp có nhiều nơron, khi đó la viết dưới dạng ma trận như sau:

$$\mathbf{W(k+1) = W(k) + 2\alpha e(k)P^T(k)}$$

$$\mathbf{B(k+1) = b(k) + 2\alpha e(k)}$$

Ở đây sai lệch e và độ dốc b là các véc tơ còn α là tốc độ học, nếu α lớn sự hội tụ học nhanh, song nếu α lớn quá có thể dẫn đến mất ổn định và sai số có thể tăng. Để đảm bảo học ổn định, tốc độ học cần nhỏ hơn nghịch đảo của giá trị riêng lớn nhất của ma trận tương quan $\mathbf{P^TP}$ của véc tơ vào.

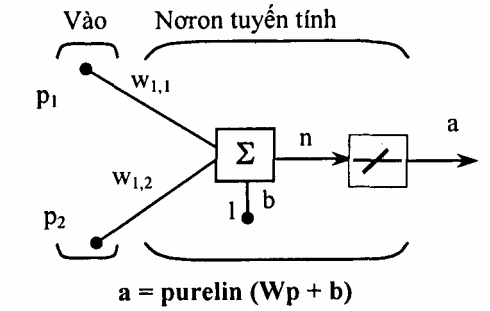
Hàm **learnwh** trong Matlab thực.hiện tất cả các công việc tính toán. Nó tính toán sự thay đổi của hàm trọng và độ dốc theo công thức:

dw = lr*e*p' và db = lr*e.

Hằng số 2 trong các công thức trên được thêm vào mã của tốc độ học lr. Hàm **maxlinlr** tính toán tốc độ học ổn định cực đại là: $0,999.p^Tp$.

5.5.3. Sự phân loại tuyến tính (train)

Mạng tuyến tính có thể được huấn luyện để thực hiện việc phân loại tuyến tính với hàm **train**. Hàm này đưa ra mỗi véc tơ của tập các véc tơ vào và tính toán sự thay đổi hàm trọng và độ dốc của mạng tương ứng với mỗi đầu vào theo **learnp**. Sau đó mạng được đặt lại cho đúng với tổng của tất cả các điều chỉnh đó. Ta gọi mỗi một lần thông qua các véc tơ vào là một khóa (epoch). Cuối cùng **train** áp dụng các đầu vào với mại mới, tính toán các đầu ra, so sánh chúng với đích và tính toán sai lệch bình quân phương. Nếu sai số đích là phù hợp hoặc nếu đã đạt tới số chu kỳ huấn luyện đặt trước thì số huấn luyện dừng. **Train** trả về mạng mới và ghi lại kết quả huấn luyện. Nếu không thì **train** chuyển sang khóa huấn luyện khác. Người ta chứng minh được rằng thuật toán LMS hội tụ khi các thủ tục này được thực hiện.



Hình 5.5. Nơron với 2 đầu vào

Ví dụ: Xét mạng tuyến tính đơn giản có 2 đầu vào, ta cần huấn luyện mạng để được cặp véc tơ vào-đích như sau:

$$\left\{p_1=\begin{bmatrix}2\\2\end{bmatrix}, t_1=0\right\}\quad\left\{p_2=\begin{bmatrix}1\\-2\end{bmatrix}, t_2=1\right\}\quad\left\{p_3=\begin{bmatrix}-2\\2\end{bmatrix}, t_3=0\right\}\left\{p_4=\begin{bmatrix}-1\\1\end{bmatrix}, t_4=1\right\}$$

Ở đây có 4 véc tơ vào, ta muốn mạng đưa ra đầu ra tương ứng với mỗi véc tơ vào khi véc tơ này xuất hiện.

Ta sẽ sử dụng hàm thun để nhận được hàm trọng và độ dốc để mạng đưa ra đích đúng cho mỗi véc tơ vào. Giá trị ban đầu của hàm trọng và độ dốc được mặc định bằng 0. Ta sẽ đặt đích sai số là 0,1 so với giá trị chấp nhận (mặc định của nó là 0)

P = [2 1 -2 -1;2 -2 2 1];

t = [0 1 0 1];

net = newlin([-2 2; -2 2],1);

net.trainParam.goal= 0.1;

[net, tr] = train(net,P,t);

Bài toán chạy đưa ra bảng ghi huấn luyện sau đây:

TRAINB, Epoch 0/100, MSE 0.510.1.

TRAINB, Epoch 251100, MSE 0.181122/0.1.

TRAINB, Epoch 501100, MSE 0.111233/0.1.

TRAINB, Epoch 64/100, MSE 0.0999066/0.1.

TRAINB, Performance goal met.

Như vậy, sau 64 kỳ huấn luyện ta đạt được mục tiêu đề ra. Hàm trọng và độ dốc mới là:

weights = net.IW{1,1}

weights =

-0 0615 -0.2194

bias = net.b(1)

bias =

[0.5899]

Ta có thể mô phỏng mạng như sau:

A = sim(net, p)

A =

0.0282 0.9672 0.2741 0.4320,

Sai số được tính toán:

err = t - sim(net,P)

err =

0 0282 0.0328 -0.2741 0.5680

Chú ý: Ta có thể huấn luyện thểm một số chu kỳ nữa, song sai số vẫn khác không và không thể đạt được sai số đích bằng 0. Điều này nói lên hạn chế về năng lực của mạng tuyến tính.

5.6. MỘT SỐ HẠN CHẾ CỦA MẠNG TUYẾN TÍNH

Mạng tuyến tính chỉ có thể học mối quan hệ tuyến tính giữa các véc tơ vào và ra. Do vậy, nó không thể tìm được lời giải cho một số bài toán. Tuy nhiên, trong lúc lời giải thực tế không tồn tại, mạng tuyến tính sẽ cực tiểu hóa tổng của bình phương các sai lệch nếu như tốc độ học (lr) của nó nhỏ. Mạng sẽ tìm được càng gần lời giải càng tốt dựa vào sự tuyến tính tự nhiên của kiến trúc mạng. Thuộc tính này tồn tại là do bề mặt sai số của mạng tuyến tính có nhiều đường parabol, các parabol chỉ có một cực tiểu và thuật toán hạ thấp độ dốc cần phải đưa ra lời giải từ cực tiểu đó.

Mạng tuyến tính có một số nhược điểm sau:

- **Đối với các hệ thống đã xác định**

Xét một hệ thống xác định. Giả thiết rằng mạng được huấn luyện với bộ 4 phần tử véc tơ vào và 4 đích. Lời giải đầy đủ thỏa mãn wp + b = t đối với mỗi véc tơ vào có thể không tồn tại do có 4 biểu thức ràng buộc mà chỉ có 1 hàm trọng và 1 độ dốc để điều chỉnh. Tuy nhiên sẽ làm cho cực tiểu sai số.

- **Các hệ thống không xác định**

Khảo sát một nơron tuyến tính đơn giản với 1 đầu vào. Lần này ta sẽ huấn luyện nó chỉ một lần, một phần tử véc tơ vào và một phần tử véc tơ đích

P = [+1.0];

T = [+0.5];

Chú ý rằng khi chỉ có một sự ràng buộc xuất hiện từ cặp vào/đích đơn giản có 2 sự biến thiên là hàm trọng và độ dốc. Có nhiều biến thiên hơn so với kết quả bắt buộc trong bài toán không xác định với số bước giải vô hạn.

Chương 6

HỆ MỜ - NƠRON (FUZZY-NEURAL)

6.1 SỰ KẾT HỢP GIỮA LOGIC MỜ VÀ MẠNG NƠRON

6.1.1 Khái niệm

Khi khảo sát mạng nơron và lôgíc mờ, ta thấy mỗi loại đều có điểm mạnh, điểm yếu riêng của nó.

Đối với logic mờ, ta dễ dàng thiết kế một hệ thống mong muốn chỉ bằng các luật **Nếu - thì (If-Then)** gần với việc xử lý của con người. Với đa số ứng dụng thì điều này cho phép tạo ra lời giải đơn giản hơn, trong khoảng thời gian ngắn hơn. Thêm nữa, ta dễ dàng sử dụng những hiểu biết của mình về đối tượng để tối ưu hệ thống một cách trực tiếp.

Tuy nhiên, đi đôi với các ưu điểm hệ điều khiển mờ còn tồn tại một số khuyết như việc thiết kế và tối ưu hóa hệ logic mờ đòi hỏi phải có một số kinh nghiệm về điều khiển đối tượng, đối với những người mới thiết kế lần đầu điều đó hoàn toàn không đơn giản. Mặt khác còn hàng loạt những câu hỏi khác đặt ra cho người thiết kế mà nếu chỉ dừng lại ở tư duy logic mờ thì hầu như chưa có lời giải, ví dụ: Số tập mờ trong mỗi biến ngôn ngữ cần chọn bao nhiêu là tôi ưu? Hình dạng các tập mờ thế nào? Vi trí mỗi tập mờ ở đâu? Việc kết hợp các tập mờ như thế nào? Trọng số của mỗi luật điều khiển bằng bao nhiêu? Nếu như tri thức cần đưa vào hệ được thể hiện dưới dạng các tập dữ liệu (điều này thường gặp khi thu thập và xử lý dữ liệu để nhận dạng đối tượng) thì làm thế nào?...

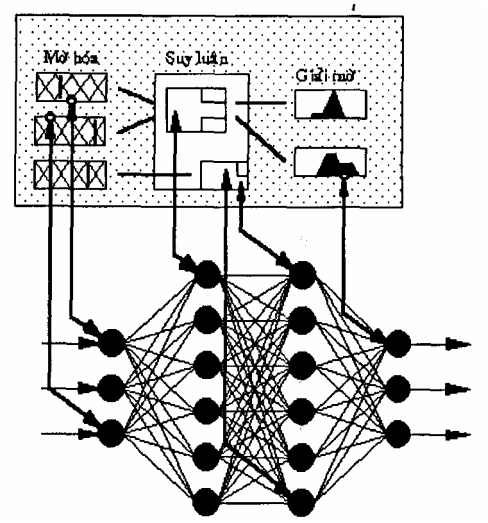
Đối với mạng nơron, chúng có một số ưu điểm như xử lý song song nên tốc độ xử lý rất nhanh; Mạng nơron có khả năng học hỏi; Ta có thể huấn luyện mạng để xấp xỉ một hàm phi tuyến bất kỳ, đặc biệt khi đã biết một tập dữ liệu vào/ra... Song nhược điểm cơ bản của mạng nơron là khó giải thích rõ ràng hoạt động của mạng nơron như thế nào. Do vậy việc chỉnh sửa trong mạng nơron rất khó khăn.

Hai tiêu chí cơ bản trợ giúp cho người thiết kế ở logic mờ và ở mạng nơron thể hiện trái ngược nhau (bảng 6.1).

Bảng 6.1

Tiêu chí	Mạng nơron	Logic mờ
Thể hiện tri thức	Không tường minh, khó giải thích và khó sửa đổi.	Tường minh, dễ kiểm_ chứng hoạt động và dễ sửa đổi.
Khả năng học	Có khả năng học thông qua các tập dữ liệu.	Không có khả năng học, người thiết kế phải tự thiết kế tất cả.

Từ những phân tích trên, ta thấy nếu kết hợp logic mờ và mạng nơron, ta sẽ có một hệ lai với ưu điểm của cả hai: logic mờ cho phép thiết kế hệ dễ dàng, tường minh trong khi mạng nơron cho phép học những gì mà ta yêu cầu về bộ điều khiển. Nó sửa đổi các hàm phụ thuộc về hình dạng, vị trí và sự kết hợp,... hoàn toàn tự động. Điều này làm giảm bớt thời gian cũng như giảm bớt chi phí khi phát triển hệ (hình 6.1).



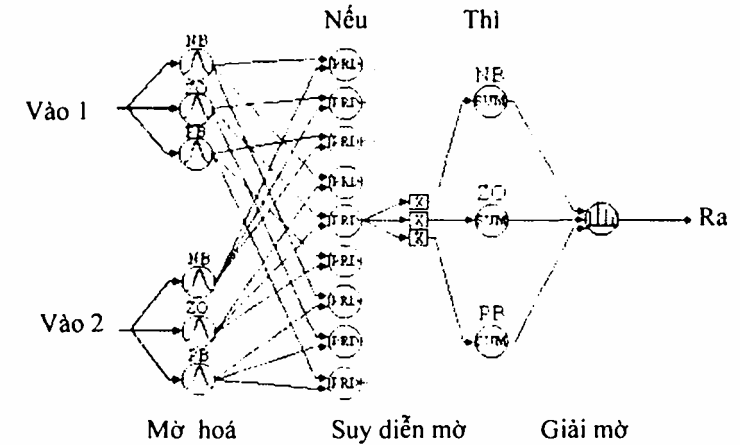
Hình 6.1. Mô hình hệ mờ - nơron

6.1.2. Kết hợp điều khiển mờ và mạng nơron

a. Cấu trúc chung của hệ mờ - nơron

Có nhiều cách kết khác nhau để hợp mạng nơron với logic mờ. Cấu trúc chung của hệ Mờ - Nơron (fuzzyneuro) như hình 6.2.

Sử dụng các nơron RBF mô tả dưới đây, sự mờ hoá có thể đạt được rất dễ dàng. Mỗi biến ngôn ngữ được xây dựng bằng 1 nơron. Chú ý rằng kiểu hàm của nơron không nhất thiết phải là hàm Gaus mà có thể là hàm khác. Trong phần này hàm liên thuộc kiểu tam giác có thể không được sử dụng vì chúng không trơn. Các nơron mờ hoá đóng vai trò lớp vào của mạng.



Hình 6.2. Cấu trúc chung của hệ mờ-nơron

Tiếp theo, lớp ẩn là toán từ MIN. Đôi khi hàm này được thay bằng toán tử PROD. Mỗi nơron nhân với giá trị đầu vào của nó và sử dụng số này như đầu ra của nó. Lớp thứ 3 được xây dựng bởi các nơron MAX (ta có thể sử dụng SUM thay vào đó). Lớp này tương tự lớp trước nhưng chúng cộng các đầu vào.

Nếu các luật đã biết, ta sẽ chỉ có mối liên hệ nơron PROD được sử dụng với các khối tổng tương ứng, nói cách khác là xây dựng đường liên lạc giữa mỗi nơron của 2 lớp này và sử dụng phép nhân cho mỗi kết nối. Việc thực hiện từng quy tắc như vậy được định nghĩa ở thời điểm đầu. Khi tối ưu mạng, giá trị của mỗi quy tắc là 1 hoặc 0 (luật hợp lệ hoặc không hợp lệ). Như vậy, các luật cơ sở như là một nhân tố bổ sung để hoàn thiện mạng.

Cuối cùng, tất cả các nơron tổng được liên kết với nơron đơn tạo thành lớp ra. Khối này xác định một giá trị cứng bằng việc xây dựng tích của mỗi vị trí MAX của nơron với giá trị tương ứng của nó và phân chia tổng này theo vị trí nơron. Đây chính là phương pháp singleton để xác định giá trị rõ ở đầu ra.

Mạng có tham số sau để thay đổi các đặc trưng của nó:

- Giá trị trung bình của mỗi hàm liên thuộc (vi là giá trị cực đại của nó).

- Chiều rộng của mỗi hàm liên thuộc.

- Tính hợp lệ (giá trị) của mỗi quy tắc.

Nhìn chung, giá trị của mỗi quy tắc không nhất thiết phải là 1 hoặc 0, chủ yếu chúng nằm giữa 2 giá trị này. Nếu bằng 0 ta coi luật đó bị mất, bình thường ta coi một luật bằng 1 hoặc bằng 0 với một mức độ nhất định.

b. Biểu diễn luật If-Then theo cấu trúc mạng nơron

Xét hệ SISO, luật điều khiển có dạng:

$$\mathbf{R_i = Nếu\ x\ là\ A_i\ Thì\ y\ là\ B_i} \qquad \mathbf{(6.1)}$$

với A_i, B_i là các tập mờ, i = 1,..., n.

Mỗi luật của (6.1) có thể chuyển thành một mẫu dữ liệu cho mạng nơron đa tầng bằng cách lấy phần "Nếu" làm đầu vào và phần "Thì" làm đầu ra của mạng. Từ đó ta chuyển khối luật thành tập dữ liệu sau:

$$\mathbf{\{(A_1,B_1),...,(A_n,B_n)\}.}$$

Đối với hệ MISO, việc biểu diễn khối luật dưới dạng tập dữ liệu cũng tương tự như đối với hệ SISO.

Ví dụ: Luật Ri :

$$\mathbf{Nếu\ x\ là\ A_i\ và\ y\ là\ B_i\ Thì\ z\ là\ C_i} \qquad \mathbf{(6.2)}$$

với A_i, B_i, C_i là các tập mờ, i = 1,..., n.

Tập dữ liệu của khối luật là:

$$\{(A_i,B_i),C_i\},\ 1 \le i \le n.$$

Còn đối với hệ MIMO thì khối luật :

$$\mathbf{R_i : Nếu\ x\ là\ A_i\ và\ y\ là\ B_i\ Thì\ r\ là\ C_i\ và\ s\ là\ D_i} \qquad \mathbf{(6.3)}$$

với A_i, B_i, C_i, D_i là các tập mờ, i = 1,..., n.

Tập dữ liệu của khối luật là:

$$\{(A_i,B_i),(C_i,D)\},\ 1 \le i \le n.$$

Có hai cách để thực hiện luật "Nếu...Thì" (If...Then) dựa trên giải thuật lan truyền ngược sai lệch :

Phương pháp Umano - Ezawa

Theo phương pháp này, một tập mờ được biểu diễn bởi một số xác định các giá trị của hàm liên thuộc của nó. Ta thực hiện theo các bước sau:

- Đặt $[\alpha_1,\alpha_2]$ chứa miền xác định của biến ngôn ngữ đầu vào (tức miền xác định của tất cả A_i).

- Đặt $[\beta_1,\beta_2]$ chứa miền xác định của biến ngôn ngữ đầu ra (tức miền xác định của tất cả B_i).

- Với M, N nguyên dương, $M \ge 2$ và $N \ge 2$ ta đặt:

$$x_i = \alpha_i + (i - 1)(\alpha_2 - \alpha_1)/(N - 1)$$

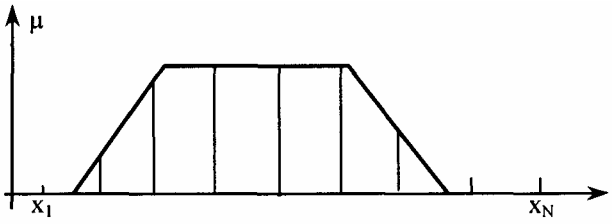
$$y_j = \beta_1 + (j - 1)(\beta_2 - \beta_1)/(M - 1)$$

với $1 \le i \le N$ và $1 \le j \le M$.

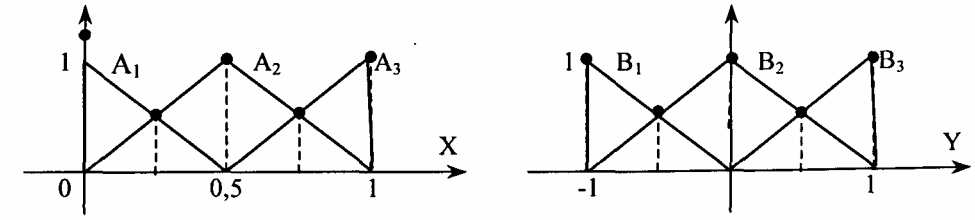
- Rời rạc hóa các tập mờ thành tập các cặp vào-ra (hình 6.3).

$$\{(A_i(x_1),..., A_i(X_N)),(B_i(y_1),...,B_i(y_M))\}, \text{ với } 1 \le i \le n.$$

Đặt $a_{ij} = A_i(x_j)$, $b_{ij} = B_i(y_j)$, khi đó mạng nơron mờ sẽ chuyển thành mạng nơron rõ với N đầu vào và M đầu ra. Từ đó có thể cho mạng học bằng giải thuật huấn luyện mạng nơron đã biết.



Hình 6.3. Rời rạc hoá hàm liên thuộc



Hình 6.4. Hàm liên thuộc các tập mờ vào và ra

Xét một hệ có 3 luật mờ với các tập mờ vào và ra như hình 6.4:

R1 : Nếu x là A_1 Thì y là B_1;

R2 : Nếu x là A_2 Thì y là B_2;

R3 : Nếu x là A_3 Thì y là B_3;

với các hàm phụ thuộc:

$\mu_{A1}(u) = 1- 2x \qquad 0 \le x \le -\frac{1}{2}$

$\mu_{A2}(u) = 1 - 2|x - 0,5| \qquad 0 \le x \le 1$

$\mu_{A3}(u) = 2x -1 \qquad \frac{1}{2} \le x \le 1$

$\mu_{B1} = -y \qquad -1 \le y \le 0$

$\mu_{B2} = 1 - 2|y| \qquad -\frac{1}{2} \le y \le \frac{1}{2}$

$\mu_{B3} = y \qquad 0 \le y \le 1.$

+ Tập dữ liệu được rút ra từ các luật này có dạng:

$$\{(A_1,B_1), (A_2,B_2), (A_3,B_3)\}.$$

+ Đặt $[\alpha_1, \alpha_2] = [0\ \ 1]$ là miền xác định của biến ngôn ngữ đầu vào.

+ Đặt $[\beta_1, \beta_2] = [-1\ \ 1]$ là miền xác định của biến ngôn ngữ đầu ra.

+ Đặt M = N = 5, Ta có:

$$x_i = (i - 1)/4, \text{ với } 1 \le i \le 5$$

$\Rightarrow x_1 = 0; x_2 = 0{,}25; x_3 = 0{,}5; x_4 = 0{,}75; x_5 = 1$

và $y_j = 1 + (j - 1)2/4 = -3/2 + j/2$, với $1 \le j \le 5$

$\Rightarrow y_1 = -1; y_2 = -0{,}5; y_3 = 0; y_4 - 0{,}5; y_5 = 1.$

+ Tập dữ liệu gồm 3 cặp vào-ra là:

$$\{(a_{11},...,a_{15}),(b_{11},...,b_{15})\}$$
$$\{(a_{21},...,a_{25}),(b_{21},...,b_{25})\}$$
$$\{(a_{31},...,a_{35}),(b_{31},...,b_{35})\}$$

với

$a_{1i} = \mu_{small}(x_i)$ $\quad$ $b_{1j} = \mu_{negative}(y_j)$

$a_{2i} = \mu_{medium}(x_i)$ $\quad$ $b_{2j} = \mu_{zem}(y_j)$

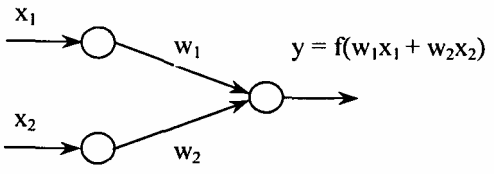
$a_{3i} = \mu_{big}(x_i)$ $\quad$ $b_{3j} = \mu_{positive}(y_j)$

Như vậy ta có:

$$\{(1;0,5;0;0;0),(1;0,5;0;0;0)\}$$
$$\{10;0,5;1;0,5;0),(0;0;1;0;0)\}$$
$$\{10;0;0;0,5;1),(0;0;0;0,5;1)\}.$$

6.2. NƠRON MỜ

Xét mạng nơron như hình 6.5. Trong đó: các tín hiệu vào-ra và các trọng số đều là số thực; Hai nơron ở đầu vào không làm thay đổi tín hiệu nên đầu ra của nó cũng là đầu vào.



Hình 6.5

Tín hiệu x_i kết hợp với trọng số w_i tạo thành tích:

$$p_i = w_i x_i, \qquad i = 1,2.$$

Đầu vào của nơron ở tầng ra là sự kết hợp của các p_i theo phép cộng:

$$p_1 + p_2 = w_1x_1 + w_2x_2.$$

- Nơron này dùng một hàm chuyển f để tạo đầu ra.

Ví dụ hàm chuyển là hàm dạng chữ S đơn cực: $f(x) = \dfrac{1}{1+e^{-x}}$

$$y = f(w_1x_1 + w_2x_2), \qquad f(x) = \frac{1}{1+e^{-x}}$$

Mạng nơron dùng phép nhân, phép cộng và hàm dạng chữ S được gọi là mạng nơron chuẩn.

Nếu mạng nơron dùng các phép toán khác như t-norm, t-conorm để kết hợp dữ liệu được gọi là mạng nơron lai. Mạng nơron lai là cơ sở để tạo ra cấu trúc nơron mờ dựa trên các phép toán mờ. Để có mạng nơron mờ ta thực hiện: Biểu diễn các đầu vào (thường là các độ phụ thuộc) x_1, x_2 và trọng số w_1, w_2 trên khoảng [0, 1].

- Mạng nơron lai có thể không dùng các phép toán nhân, phép toán cộng hoặc hàm dạng chữ S bởi vì kết quả của các phép toán này có thể không nằm trong khoảng [0, 1].

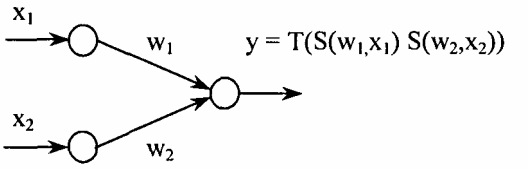
Định nghĩa: Mạng nơron lai là mạng nơron sử dụng tín hiệu rõ và hàm truyền rõ, song sự kết hợp x_1 và w_1 dùng các phép toán t-norm, t-conorm hay các phép toán liên tục khác và sự liên kết p_1 và p_2 dùng các hàm t-norm, t-conorm hay các hàm liên tục khác, hàm chuyển f có thể là một hàm liên tục bất kỳ.

Chú ý: đối với mạng nơron mờ thì giá trị vào, giá trị ra, và trọng số là những số thực nằm trong khoảng [0, 1].

Nơron mờ AND (hình 6.6)

Tín hiệu x_i và trọng số w_i được kết hợp bởi conorm S tạo thành:

$$p_i = S(w_i,x_i), \; i = 1,2$$



Hình 6.6. Nơ ron mờ AND

Các p_i được tính bởi norm T để tạo đầu ra của nơron.

$$y = AND(p_1,P_2) = T(p_1,p_2) = T(S(w_1,x_1),S(w_2,x_2)).$$

Nếu T = min và S = max thì nơron mờ AND chính là luật hợp thành min-max

$$y = \min\{w_1 \vee v\, x_1, w_2 \vee x_2\}.$$

Nơron mờ OR (Hình 6.7)

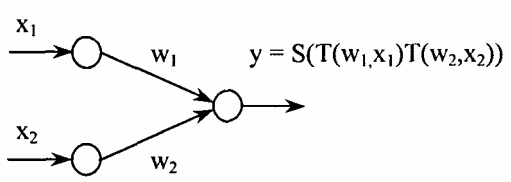
Tín hiệu x_i và trọng số w_i được kết hợp bởi norm T tạo thành :

$p_i = T(w_i,x_i)$, i = 1,2.

Các p_i được tính bởi conorm S tạo đầu ra của nơron:

$y = OR(p_1,p_2) = S(p_1,p_2)$

$= S(T(w_1,x_1),T(w_2,x_2))$.



Hình 6.7. Nơ ron mờ OR

Nếu T = min và S = max thì nơron mờ OR chính là hợp thành max-min

$$y = \max\{w_1 \wedge x_1, w_2 \wedge x_2\}.$$

6.3. HUẤN LUYỆN MẠNG NƠRON-MỜ

Đối với mô hình mờ, mối quan hệ phi tuyến vào-ra phụ thuộc rất nhiều vào các phân vùng mờ của không gian vào-ra. Do đó việc chỉnh định hàm liên thuộc trong các mô hình mờ trở nên rất quan trọng. Trong mạng nơron mờ việc chỉnh định này có thể xem như là vấn đề tối ưu dùng giải thuật học để giải quyết.

Đầu tiên ta giả định các hàm liên thuộc có một hình dạng nhất định. Sau đó ta thay đổi các thông số của hình dạng đó qua quá trình học bằng mạng nơron.

Như vậy ta cần một tập dữ liệu ở dạng các cặp vào-ra mong muốn để cho mạng nơron học và cũng cần phải có một bảng các luật sơ khởi dựa trên các hàm phụ thuộc đó.

Giả sử cần thực hiện ánh xạ:

$$y^k = f(x^k) = f(x_1^k,...,x_n^k), \text{ với } k = 1,..., K.$$

Ta có tập dữ liệu : $\{(x^1,y^1),...,(x^k,y^k)\}$.

Dùng luật If-Then (nếu - thì) để thực hiện ánh xạ này:

Ri : Nếu x_1 là A_{il} và... và x_n là A_{in} thì $y = z_i$, $1 \le i \le m$

với A_{if} là các tập mờ có dạng hình tam giác và z_i là số thực.

Đặt o^k là giá trị ra của hệ khi ta đưa vào x^k.

Ký hiệu α_1 là giá trị ra của luật thứ i, được định nghĩa theo tích Larsen:

$$\alpha_i = \prod_{j=1}^{n} A_{ij}(x_j^k)$$

(cũng có thể định nghĩa các t-norm khác).

Giải mờ theo phương pháp trung bình trọng tâm ta có:

$$o^k = \frac{\sum_{i=1}^{m} \alpha_i z_i}{\sum_{i=1}^{m} \alpha_i}.$$

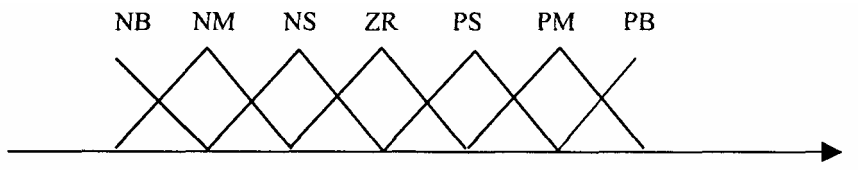
Sai lệch của mẫu thứ k là:

$$e^k = \frac{1}{2} (o^k - y^k)^2.$$

Dùng phương thức giảm để học zi trong phần kết quả của luật **Ri**:

$$z_i(t+1) = z_i(t) - \eta \frac{\partial E_k}{\partial z_i} = z_i(t) - \eta(o^k - y^k) \frac{\alpha_i}{\alpha_1 + ... + \alpha_m}, \qquad i = \overline{1,m}.$$

Cho rằng mỗi biến ngôn ngữ có 7 tập mờ như hình 6.8: {NB, NM, NS, ZE, PS, PM, PB}.



Hình 6.8

Các hàm liên thuộc có hình dạng tam giác được đặc trưng bởi 3 tham số: tâm, độ rộng trái, độ rộng phải. Các tham số này của tam giác cũng được học bằng phương thức giảm.

Ví dụ: Xét 2 luật mờ SISO

R_1: Nếu x là A_1 Thì $y = z_1$

R_2: Nếu x là A_2 Thì $y = z_2$

Giả sử A_1 và A_2 được định nghĩa bởi :

$$A_1 = \frac{1}{1 + \exp[b_1(x - a_1)]} \qquad A_2 = \frac{1}{1 + \exp[b_2(x - a_2)]}$$

với a_1, a_2, b_1, b_2 là các giá trị khởi tạo ban đầu.

Vậy giá trị ra của luật là:

$$\alpha_1 = A_1 = \frac{1}{1+\exp[b_1(x-a_1)]} \qquad \alpha_2 = A_2 = \frac{1}{1+\exp[b_2(x-a_2)]}.$$

Giá trị ra của hệ mờ:

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \frac{A_1(x) z_1 + A_2(x) z_2}{A_1(x) + A_2(x)}.$$

Giả sử chúng ta có tập dữ liệu cần học:

$$\{(x_1, y_1),...,(x_k, y_k)\}.$$

Nhiệm vụ của chúng ta là xây dựng 2 luật mờ dựa trên các tập mờ đã sửa đổi, sao cho kết quả tạo ra tương thích với các cặp vào-ra cho trước.

Định nghĩa sai lệch cho mẫu thứ k:

$$E_k = E_k(a_1,b_1,a_2,b_2,z_1,z_2) = \frac{1}{2}[o^k(a_1,b_1,a_2,b_2,z_1,z_2) - y^k]2.$$

Dùng phương thức giảm để học:

$$\begin{aligned} z_1(t+1) &= z_1(t) - \eta\frac{\partial E_k}{\partial z_1} = z_1(t) - \eta(o^k - y^k)\frac{\alpha_1}{\alpha_1+\alpha_2} \\ &= z_1(t) - \eta(o^k - y^k)\frac{A_1(x^k)}{A_1(x^k)+A_2(x^k)} \end{aligned}$$

tương tự:

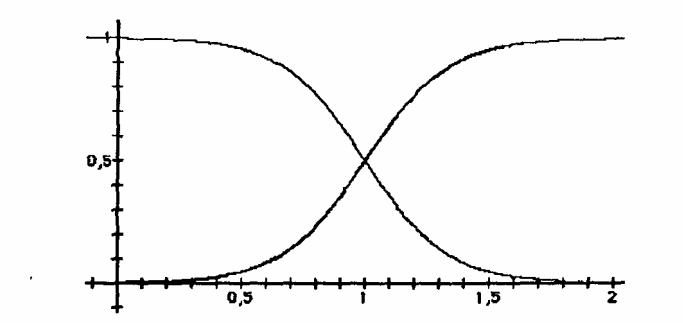
$$z_2(t+1) = z_2(t) - \eta(o^k - y^k)\frac{A_1(x^k)}{A_1(x^k)+A_2(x^k)}$$

$$a_1(t+1) = a_1(t) - \eta\frac{\partial E_k}{\partial a_1}$$

$$b_1(t+1) = b_1(t) - \eta\frac{\partial E_k}{\partial b_1}$$

$$a_2(t+1) = a_2(t) - \eta\frac{\partial E_k}{\partial a_2}$$

$$b_2(t+1) = b_2(t) - \eta\frac{\partial E_k}{\partial b_2}.$$



Hình 6.9

Luật học sẽ đơn giản hơn nếu ta dùng các hàm liên thuộc có dạng hình 6.9:

$$A_1(x) = \frac{1}{1+\exp[-b(x-a)]} \quad A_2(x) = \frac{1}{1+\exp[b(x-a)]}.$$

khi đó $A_1(x) + A_2(X) = 1$, $\forall x$.

Việc sửa đổi được thực hiện như sau :

$$z_1(t+1) = z_1(t) - \eta\frac{\partial E_k}{\partial z_1} = z_1(t) - \eta\left(o^k - y^k\right)A_1(x^k)$$

$$z_2(t+1) = z_2(t) - \eta\frac{\partial E_k}{\partial z_2} = z_2(t) - \eta\left(o^k - y^k\right)A_2(x^k)$$

$$a(t+1) = a(t) - \mu\frac{\partial E_k(a,b)}{\partial a}$$

$$\frac{\partial E_k(a,b)}{\partial a} = \left(o^k - y^k\right)\frac{\partial o^k}{\partial a} =$$

$$\left(o^k - y^k\right)\frac{\partial}{\partial a}\left[z_1A_1(x^k) + z_2A_2(x^k)\right] =$$

$$\left(o^k - y^k\right)\frac{\partial}{\partial a}\left[z_1A_1(x^k) + z_2\left(1 - A_1(x^k)\right)\right] =$$

$$\left(o^k - y^k\right)\left(z_1 - z_2\right)\frac{\partial A_1(x^k)}{\partial a} =$$

$$\left(o^k - y^k\right)\left(z_1 - z_2\right)b\frac{\exp\left(b(x^k - a)\right)}{\left[1+\exp\left(b(x^k - a)\right)\right]^2} =$$

$$\left(o^k - y^k\right)\left(z_1 - z_2\right)bA_1(x^k)\left(1 - A_1(x^k)\right) =$$

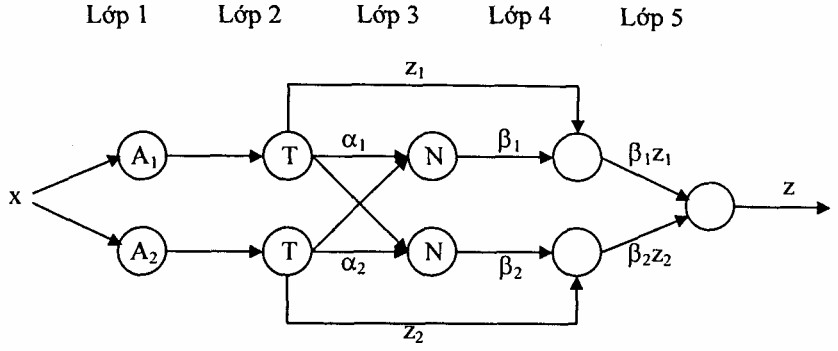
$$\left(o^k - y^k\right)\left(z_1 - z_2\right)bA_1(x^k)A_2(x^k)$$

và

$$\frac{\partial E_k(a,b)}{\partial b} = (o^k - y^k)(z_1 - z_2)\frac{\partial}{\partial}\frac{1}{1+\exp(b(x^k - a))} =$$
$$-(o^k - y^k)(z_1 - z_2)(x^k - a)A_1(x^k)(1 - A_1(x^k)) =$$
$$-(o^k - y^k)(z_1 - z_2)(x^k - a)A_1(x^k)A_2(x^k).$$

Mạng nơron cho hệ mờ này sẽ như hình 6.10, mạng này gồm 5 lớp: Lớp 1: Giá trị ra từ nút chính là độ phụ thuộc của biến đối với tập mờ. Lớp 2: Tạo giá trị ra của luật: $\alpha_1 = A_1$; $\alpha_2 = A_2$.

Những nút này được gán nhãn T bởi vì chúng ta có thể chọn nhiều phép toán t-norm khác nhau cho phép AND (VÀ).



Hình 6.10

Lớp 3: Lấy trung bình: $\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}$ $\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}$

Lớp 4: Giá trị ra của nơron là tích của z_i và β_1.

Lớp 5: Kết hợp tạo giá trị ra cuối cùng của hệ: $z = \beta_1 z_1 + \beta_2 z_2$.

6.4. SỬ DỤNG CÔNG CỤ ANFIS TRONG MATLAB ĐỂ THIẾT KẾ HỆ MỜ - NƠRON (ANFIS and the ANFIS Editor GUI)

6.4.1. Khái niệm

Cấu trúc cơ bản của hệ thống suy luận mờ như chúng ta đã thấy là mô hình thực hiện sự ánh xạ các thuộc tính vào đến các hàm liên thuộc vào, hàm liên thuộc vào đến các luật, các luật đến tập các thuộc tính ra, các thuộc tính ra đến hàm liên thuộc ra và hàm liên thuộc ra đến giá trị ra đơn trị hoặc quyết định kết hợp với đầu ra. Chúng ta mới chỉ đề cập đến các hàm liên thuộc được bố trí trước và ở mức độ nào đó việc chọn còn tuỳ tiện. Đồng thời chúng ta

cũng mới chỉ áp dụng các suy diễn mờ để mô hình hoá hệ thống mà cấu trúc luật về cơ bản được định trước bằng việc sử dụng sự thể hiện của thuộc tính của các biến trong mô hình.

Trong phần này, ta sẽ việc sử dụng hàm **anfis** và **ANFIS Editor GUI** trong bộ công cụ **Fuzzy Logic Toolbox** của Matlab. Công cụ này áp dụng kỹ thuật suy diễn mờ để mô hình hoá đối tượng. Như ta đã biết ở phần suy diễn mờ GUIS hình dạng của hàm liên thuộc phụ thuộc vào các tham số, khi thay đổi các tham số sẽ thay đổi hình dạng của hàm liên thuộc. Thay vì nhìn vào dữ liệu để chọn tham số hàm liên thuộc chúng ta thấy các hàm liên thuộc có thể được chọn một cách tự động.

Giả thiết ta muốn áp dụng suy diễn mờ cho hệ thống mà đối với nó ta đã có một tập dữ liệu vào/ra, ta có thể sử dụng để mô hình hoá, mô hình sắp tới hoặc một vài phương pháp tương tự. Không nhất thiết phải có cấu trúc mô hình định trước làm cơ sở cho thuộc tính của các biến trong hệ thống. Có một vài mô hình trạng thái trên nó chúng ta không thể nhận thấy dữ liệu và không thể hình dung được hình dạng của hàm lên thuộc. Đúng hơn là việc chọn các thông số liên kết với các hàm liên thuộc định sẵn là tuỳ tiện, các thông số này được chọn sao cho làm biến đổi tập dữ liệu vào/ra đến bậc được miêu tả cho dạng đó của các biến trong các giá trị dữ liệu. Do đó được gọi là kỹ thuật học neuro-**Adaptive** hợp thành **anfis.**

6.4.2. Mô hình học và suy diễn mờ thông qua ANFIS (Model Learning and Inferencc Through ANFIS)

Ý tưởng cơ bản của kỹ thuật học **neuro-adaptive** rất đơn giản. Kỹ thuật này đưa ra cơ chế cho mô hình mờ có thủ tục để học thông tin về tập dữ liệu theo thứ tự ước tính các tham số của hàm liên thuộc mà nó cho phép kết hợp với hệ thống suy diễn mờ theo hướng dữ liệu vào/ra nhất định. Phương pháp học này làm việc tương tự như mạng nơron. Bộ công cụ lôgic mờ dùng để thực hiện việc điều chỉnh tham số của hàm liên thuộc được gọi là anfis. Ta có thể mở anrs từ dòng lệnh hoặc từ giao diện đồ hoạ (ANFIS Editor GUI). Hai cách này tương tự nhau, chúng được sử dụng hoán đổi nhau. Tuy nhiên, giữa chúng cũng có đối chút khác biệt (chúng ta sẽ bàn đến ở phần sau).

a/ Tìm hiểu về ANFIS

ANFIS xuất phát từ tiến Anh là **Adaptive neuro-fuzzy infercnce system**. Sử dụng tập dữ liệu vào/ra có sẵn, hàm **anfis** xây dựng nên hệ thống suy diễn mờ (FIS), các thông số hàm liên thuộc của nó được điều chỉnh nhờ sử dụng các thuật toán huấn luyện của mạng nơron như thuật toán lan truyền ngược hoặc kết hợp lan truyền với phương pháp bình phương cực tiểu. Điều đó cho phép hệ mờ của ta "học" từ tập dữ liệu chúng được mô hình.

b) Cấu trúc và sự điều chỉnh tham số của FIS

Một kiểu mạng có cấu trúc tương tự mạng nơron, nó ánh xạ các đầu vào qua các hàm liên thuộc vào với các thông số tương ứng và sau đó là thông qua các hàm ra với các tham số tương ứng tạo nên các đầu ra có thể được sử dụng để giải thích ánh xạ vào/ra. Các thông số tương ứng với hàm liên thuộc sẽ thay đổi thông qua quá trình học. Việc tính toán các tham số này (hoặc việc điều chỉnh chúng) thực hiện dễ dàng bằng véc tơ gradient nó đưa ra giới hạn theo cách tốt cho hệ thống suy diễn mờ được mô hình hoá dữ liệu vào/ra theo tập các tham số nhất định. Ta đã biết, véc tơ gradient được áp dụng cho một vài thủ tục tối ưu cốt để điều chỉnh các tham số sao cho giảm nhỏ giá trị sai số (thường được định nghĩa bằng tổng bình phương sai lệch giữa đầu ra hiện thời và đầu ra mong muốn). **Anfis** sử dụng điều đó theo giải thuật lan truyền ngược hoặc kết hợp sự ước lượng bình phương cực tiểu và sự lan truyền ngược cho sự ước lượng tham số hàm liên thuộc.

6.4.3. Xác nhận dữ liệu huấn luyện (Familiarity Brecds Validation)

a. Tìm hiểu dữ liệu

Phương thức tạo mẫu được sử dụng bởi anfis giống như các kỹ thuật nhận dạng hệ thống khác. Đầu tiên ta đưa ra một cấu trúc tham số mẫu (liên kết các đầu vào tới các hàm liên thuộc với các luật tới các đầu ra tới các hàm liên thuộc...). Kế đến, là thu thập dữ liệu vào/ra vào một dạng sao cho tiện lợi cho sự huấn luyện của anfis. Ta có thể sử dụng anfis để huấn luyện mô hình FIS nhằm mô phỏng dữ liệu huấn luyện đưa vào để nó sửa đổi các tham số của hàm liên thuộc theo tiêu chuẩn sai số dã lựa chọn. Nói chung, kiểu mô hình này sẽ làm việc tốt nếu dữ liệu đưa vào anfis cho sự huấn luyện tham số các hàm liên thuộc đại diện đầy đủ cho các đặc tính của tập dữ liệu mà nó được FIS huấn luyện giành cho mô hình. Điều này không phải luôn luôn xảy ra, tuy nhiên, trong một vài trường hợp trong quá trình thu thập dữ liệu, do ảnh hưởng của nhiễu đo lường mà dữ liệu huấn luyện không thể đại diện cho tất cả các thuộc tính của dữ liệu sẽ có mặt ở mô hình.

b. Xác định mô hình bằng cách sử dụng các phần dữ liệu thử và kiểm tra (Model Validation Using Checking and Testing Data Sets)

Công nhận giá trị mẫu (xác định mẫu) là quá trình trong đó các vectơ vào từ dữ liệu vào/ra được đặt tại nơi mà FIS chưa được huấn luyện, mẫu được đưa tới huấn luyện FIS để mẫu FIS đón trước giá trị dữ liệu đầu ra tương ứng có tốt hay không. Nó được thực hiện bởi bộ soạn thảo ANFIS GUI. Ta có thể sử dụng một loại dữ liệu khác để công nhận giá trị mẫu trong anfis. Hình thức công nhận dữ liệu này được hình dung như một hệ thống dữ liệu kiểm tra được sử dụng để điều chỉnh sự công nhận giá trị dữ liệu. Khi dữ liệu kiểm tra được đưa tới **anfis** cũng giống như dữ liệu huấn luyện, mẫu FIS lựa chọn để

các tham số liên quan có sai số mẫu dữ liệu nhỏ nhất.

Một vấn đề đặt ra là việc công nhận giá trị dữ liệu để tạo mẫu sử dụng các kỹ thuật thích nghi là lựa chọn tập dữ liệu tiêu biểu cho dữ liệu mẫu huấn luyện, nhưng khác biệt với dữ liệu huấn luyện được thiết lập không phải để phản hồi cho quá trình hợp thức hoá thiếu hiệu quả. Nếu ta thu thập một lượng lớn các dữ liệu, thì dữ liệu này chứa đựng đầy đủ các đặc tính tiêu biểu vì vậy quá trình thu thập dữ liệu để phục vụ mục đích kiểm tra hoặc thử sẽ dễ dàng hơn. Tuy nhiên nếu ta muốn thực hiện các phép đo ở mẫu, có thể dữ liệu huấn luyện không bao gồm tất cả các đặc tính tiêu biểu mà ta muốn.

Ý tưởng cơ bản đằng sau việc sử dụng dữ liệu kiểm tra cho hợp thức hoá là sau một điểm nhất định trong quá trình huấn luyện, mẫu.bắt đầu vượt quá phần dữ liệu huấn luyện đã được thiết lập. Theo nguyên tắc, sai số mẫu cho thiết lập dữ liệu kiểm tra dường như giảm khi việc huấn luyện xảy ra tại điểm mà việc điều chỉnh quá mức bắt đầu, và sau đó sai số mẫu cho dữ liệu kiểm tra đột ngột tăng. Trong ví dụ đầu ở phần dưới đây, hai dữ liệu giống nhau được sử dụng để huấn luyện và kiểm tra, nhưng phần dữ liệu kiểm tra bị sửa đổi bởi một lượng tiếng ồn nhỏ. Bằng việc kiểm tra chuỗi sai số trong quá trình huấn luyện, rõ ràng là dữ liệu kiểm tra không được tốt cho các mục đích hợp thức hoá mẫu. Ví dụ này minh hoạ cách sử dụng bộ soạn thảo ANFIS GUI để so sánh các dữ liệu.

c. Một số hạn chế của Anfis

Anfis phức tạp hơn các hệ thống suy luận mờ mà chúng ta đã đề cập ở chương 1 rất nhiều, và cũng không sẵn có như các tuỳ chọn của hệ thống suy luận mờ. Đặc biệt, anfis chỉ hỗ trợ cho các hệ thống mờ theo mô hình Sugèno và chúng cần có những ràng buộc sau:

- Là các hệ thống loại Sugeno ở vị trí 0 hoặc 1**.**
- Có một đầu ra đơn, giải mờ bằng phương pháp trung bình trọng tâm. Tất cả các hàm liên thuộc đầu ra phải cùng loại, hoặc tuyến tính hoặc bất biến.
- Không chia sẻ luật điều khiển. Các luật khác nhau không thể chia sẻ cùng một hàm liên thuộc đầu ra, cụ thể là số lượng các hàm liên thuộc đầu ra phải bằng số lượng các luật.

Có một trọng lượng nhất định (đồng nhất) cho mỗi một nguyên tắc.

Khi không **train** thủ đủ những ràng buộc trên, cấu trúc FIS sẽ bị sai số. Hơn nữa, anfis không thể chấp nhận các tuỳ chọn thông thường mà suy luận mờ cơ bản cho phép. Vì vậy chúng ta không thể tùy ý tạo ra các hàm liên thuộc và các phương pháp giải mờ của mình mà phải sử dụng những chức năng đã cho.

6.5. SỬ DỤNG BỘ SOẠN THẢO ANFIS GUI

6.5.1. Các chức năng của ANFIS GUI

Trong phần này, chúng ta cùng tìm hiểu cách khai thác bộ soạn thảo hệ mờ - nơron thông qua giao diện đồ họa. Để khởi động bộ soạn thảo ANFIS GUI, gõ: **anfisedit**. Cửa sổ thảo GUI sau đây xuất hiện trên màn hình (hình 6.11). Từ bộ soạn thảo GUI này ta có thể:

Tải dữ liệu (huân luyện, thử và kiểm tra) bằng cách lựa chọn những nút thích hợp trong phần Load thừa của GUI và bấm vào Load Data. Dữ liệu tải về được vẽ trong phần đồ thị.

Tạo một mô hình FIS ban đầu hoặc tải một mô hình FIS ban đầu bằng cách sử dựng các lựa chọn trong Generate FIS của GUI.

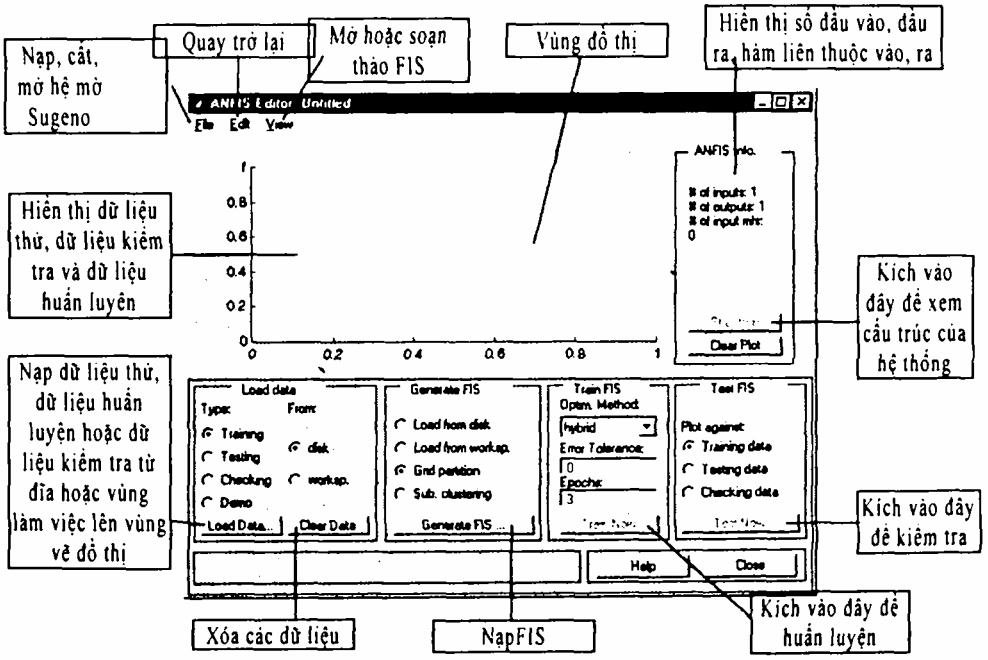
Xem câu trúc mẫu FIS khi FIS ban đầu đã được tạo hoặc tải bằng cách nháy vào nút Structure.

Chọn phương pháp tôi ưu tham số mô hình FIS: truyền ngược hoặc kết hợp truyền ngược với binh phương nhỏ nhất (phương pháp lai).

Chọn số kỳ huấn luy huấn luyện mô hình FIS bằng cách nhấn vào nút **Train** Now.

Huấn luyện này điều chỉnh các tham số hàm liên thuộc và các sơ đồ huấn luyện (và/ hoặc dữ liệu kiểm tra) các sơ đồ sai số trong phần sơ đồ.

Quan sát mô hình FIS để thấy dược dữ liệu huấn luyện, **kiểm tra** hoặc thử dữ liệu đầu ra bằng cách ấn nút Test Now.



Hình 6.11

Chức năng này vẽ dữ liệu thử tương phản với đầu ra FIS trong phần sơ đồ.

Ta có thể sử dụng thanh thực đơn bộ soạn thảo ANFIS GUI để tải một huấn luyện FIS ban đầu, ghi FIS huấn luyện, mở một hệ thống Sugeno hoặc mở bất kỳ một GUI nào để phân tích sự huấn luyện của mô hình FIS.

6.5.2. Khuôn dạng dữ liệu và bộ soạn thảo ANFIS GUI: kiểm tra và huấn luyện (Data Formalities and the ANFIS Editor GUI: Checking and Training)

Để khởi động một FIS sử dụng anfis hoặc bộ soạn thảo ANFIS GUI, đầu tiên ta cần có một dữ liệu huấn luyện chứa các cặp dữ liệu đầu vào/đầu ra mong muốn của hệ thống đích. Đôi khi cũng cần tập dữ liệu thử tuỳ chọn có thể kiểm tra được khả năng khái quát hoá của hệ thống suy luận mờ, đồng thời tập dữ liệu kiểm tra có thể giúp đỡ việc điều chỉnh trong suốt quá trình huấn luyện. Như đã đề cập từ phần trước, việc điều chỉnh được tính để thử nghiệm huấn luyện FIS trên một dữ liệu huấn luyện đối lập dữ liệu kiểm tra, và chọn hàm tham số hàm liên thuộc nối kết với sai số kiểm tra nhỏ nhất nếu những sai số này chỉ ra việc điều chỉnh mẫu quá mức. Ta sẽ phải kiểm tra sơ đồ sai số huấn luyện nhỏ nhất để quyết định điều này. Những vấn đề này sẽ được bàn đến ở một ví dụ phần sau. Thường thì những phần dữ liệu huấn luyện và kiểm tra được thu thập dựa trên các quan sát của hệ thống đích và

sau đó được lưu lại trong các tệp tin tách biệt.

Chú ý: Bất cứ tập dữ liệu nào mà ta tải vào bộ soạn thảo ANFIS GUI, (hoặc là cái được ứng dụng vào hàm lệnh anfis) phải là một ma trận với các dữ liệu đầu vào được sắp xếp như các vecto trong tất cả trừ cột cuối cùng. Dữ liệu đầu ra phải được đặt trong cột cuối cùng.

6.5.3. Một số ví dụ

a. Ví d ụ 1:

Dữ liệu kiểm tra giúp công nhận giá trị mẫu (Checking Data Hetps Model Validation)

Trong phần này chúng ta sẽ xem xét một ví dụ tải tập dữ liệu huấn luyện và kiểm tra, trong đó chỉ có dữ liệu kiểm tra bị thay đổi bởi nhiễu.

+ Tải dữ liệu

Để thực hiện cho cả hai ví dụ, sau vào cửa sổ soạn thảo ANFIS GUI, ta tải tập dữ liệu huấn luyện có tên: **(fuzex1trnData and fuzex2trnData)** và tập dữ liệu kiểm tra có tên: (**fuzexlchkData and fuzex2chkData**) vào vùng làm việc bộ soạn thảo ANFIS GUI.

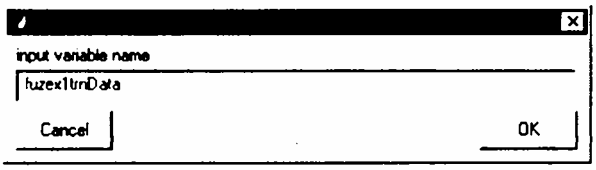
Để tải các tập dữ liệu từ thư mục fuzzydemos của Matlab vào vùng làm việc của MATLAB, từ cửa sổ dòng lệnh ta gõ:

- load fuzex 1 trnData.dat
- load fuzex2trnData.dat
- load fuzex1chkData.dat
- load fuzex2chkData.dat

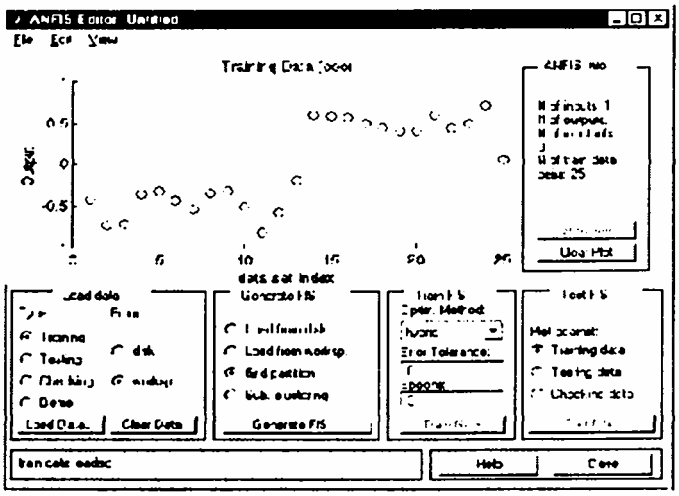
Chú ý: Ta có thể muôn tải dữ liệu từ fuzzydemos hoặc bất cứ thư mục nào trên đĩa, sử dụng bộ soạn thảo ANFIS Gõ một cách trực tiếp.

Mở bộ soạn thảo ANFIS GUI bằng cách gõ **anrscdit**. Đề tài dữ liệu huấn luyện, nhấn vào **Training worksp** và sau đó là **Load Data**...

Cửa sổ nhờ GUI mở ra cho phép ta gõ một tên biến từ vùng làm việc. Gõ vào **fuzex1tmDat**a như trong hình 6.12.



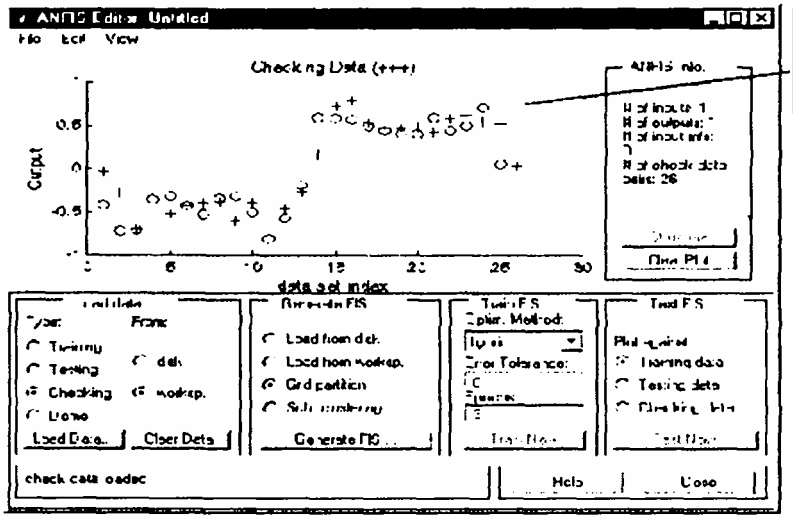
Hình 6.12



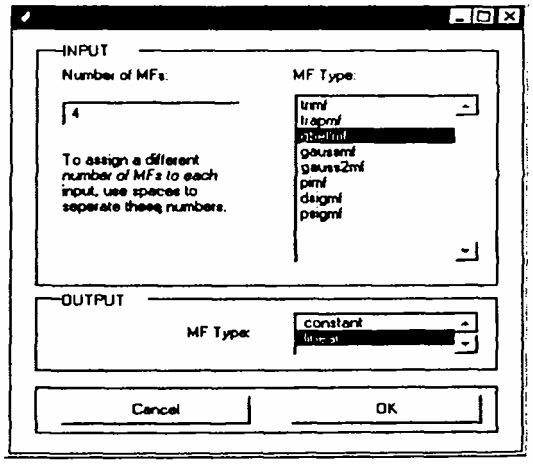
Hình 6.13

Dữ liệu huấn luyện xuất hiện trong sơ đồ ở khu đồ thị của GUI là tập các hình tròn (hình 6.13).

Chú ý: Thanh ngang được đánh dấu là tập chi số dữ liệu. Chỉ số này chỉ rõ hàng để nhập giá trị dữ liệu đầu vào (dù dầu vào là một vectơ hay một đại lượng vô hướng). Tiếp đến lựa chọn hộp **Checking** trong cột Type của **Load Data** của GUI để tải fuzex1chkData từ nơi làm việc. Dữ liệu này xuất hiện trong sơ đồ GUI như các phần thêm vào dữ liệu huấn luyện (hình 6.14).



Hình 6.14



Hình 6.15

Dữ liệu này sẽ được sử dụng để huấn luyện một hệ thống suy luận mờ bằng cách điều chỉnh các tham số của hàm liên thuộc sao cho tốt nhất với dữ liệu này. Bước tiếp theo là cụ thể hoá một hệ thống suy luận mờ ban đầu cho anfis nhằm mục đích huấn luyện.

+ Khởi tạo FIS ban đầu

Ta thể khởi tạo cấu trúc FIS theo sở thích của mình, hoặc có thể tạo ra mô hình FIS ban đầu một cách tự động. Để khởi tạo FIS sử dụng anr's theo các bước sau:

☞ Chọn Giựt partition, phương pháp phân chia mặc định.

Bấm vào nút Generate FIS. Một thực đơn hiện ra (hình 6.15) cho phép

chọn số lượng hàm liên thuộc (MFS), và kiểu hàm liên thuộc đầu vào, đầu ra. Lưu ý chỉ có hai lựa chọn cho hàm liên thuộc đầu ra: hằng số và tuyến tính. Giới hạn của các lựa chọn hàm liên thuộc đầu ra là vì anfis chỉ có thể hoạt động trên hệ thống kiểu Sugeno.

☞ Điền đầy đủ các thông số như dưới đây rồi nhấn OK.

☞ Ta cũng có thể tạo FIS từ dòng lệnh có sử dụng lệnh **genfis1** (dành cho phân chia lưới) hoặc **genfis2** (Cho tập hợp các phép trừ). Ví dụ ngôn ngữ một dòng lệnh minh hoạ cách sử dụng của genfis1 và anfis sẽ được cung cấp sau.

+ Ấn định các hàm liên thuộc cho ANFIS

Để định nghĩa cấu trúc FIS và các tham số của nó theo quan điểm của riêng mình, ta thực hiện theo các bước sau:

☞ Mở thực đơn **Edit membership funtions** từ thực đơn **View**.

☞ Thêm các hàm liên thuộc mong muốn (việc lựa chọn các hàm liên thuộc theo ý muốn sẽ làm vô hiệu hóa anrs). Các hàm liên thuộc đầu ra cần phải tuyến tính hoặc hằng số.

☞ Chọn thực đơn **Edit rule**s trong thực đơn **View**. Sử dụng Rule Edìtor để tạo ra các luật (xem The Rule Editor).

☞ Chọn thực đơn **Edit FIS Properties** từ thực đơn View. Đặt tên cho FIS và ghi vào vùng làm việc hoặc vào đĩa.

☞ Sử dụng thực đơn View để quay trở lại bộ soạn thảo AN FIS GUI nhằm huấn luyện FIS.

Để tải một FIS hiện có cho ANFIS ban đầu, trong mục Gcnerate FIS của GUI, bấm vào **Load from worksp** hoặc **Load from disk**. Ta sẽ tải FIS mà trước đó đã được lưu vào đĩa. Ta cũng có thể tải FIS từ vùng làm việc, hoặc bật nút Generate FIS để tải FIS bằng cách bấm vào nút này.

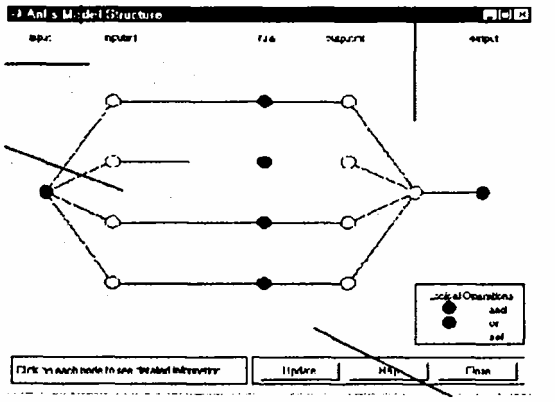
+ Xem cấu trúc của FIS

Sau khi tạo một mô hình FIS, ta có thể xem câu trúc của mô hình bằng cách bấm vào nút **Structure** ở phần giữa bên phải của GUI. Một GUI mới xuất hiện như trong hình 6.16.

Trong các nhánh nút mạch là nút màu để biểu thị nguyên tắc AND, NOT hoặc OR được sử dụng. Bấm vào các nút sẽ hiển thị thông tin về cấu trúc.

Ta có thể quan sát các hàm liên thuộc cũng như các luật bằng cách mở bộ soạn thảo hàm liên thuộc, hoặc bộ soạn thảo nguyên tắc từ thực đơn View.

+ Huấn luyện ANFIS



Hình 6.16

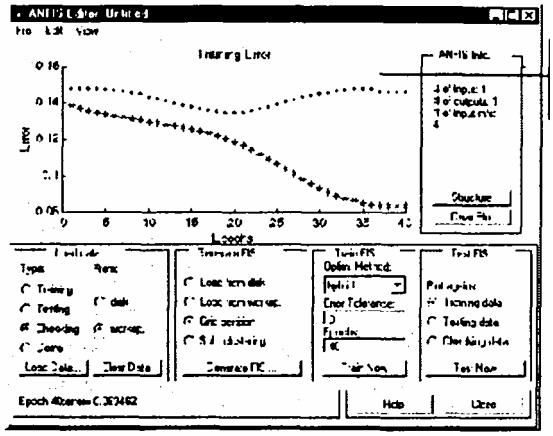
Ta có thể chọn một trong hai phương pháp tối ưu hóa các tham số của **anfis** để huấn luyện FIS là **hybris** (hỗn hợp truyền ngược và bình phương nhỏ nhất) và **backpropa** (lan truyền ngược). Dung sai được sử dụng để tạo một tiêu chuẩn dùng huấn luyện. Việc huấn luyện sẽ ngừng lại sau khi sai số dữ liệu huấn luyện nằm trong dung sai cho phép. Thường ta đặt sai số cho phép cuối cùng bằng 0 khi chưa biết rõ sai số bằng bao nhiêu.

Để bắt đầu huấn luyện ta thec hiện theo các bước sau:

☞ Chọn phương pháp tối ưu, ví dụ: **hybrid**.

☞ Đặt số kỳ huấn luyện là 40, trong ô **Epochs** của GUI (giá trị mặc định là 3).

☞ Lựa chọn **Train now**. Hình vẽ sau xuất hiện trên màn hình (hình 6.17).

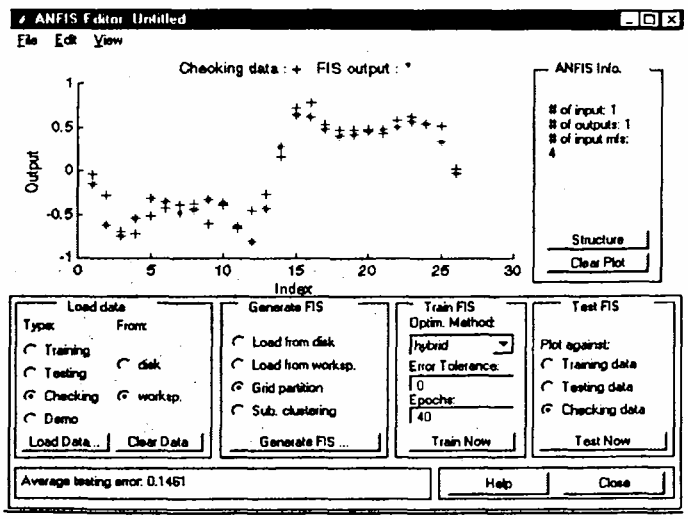


Hình 6.17

Chú ý: Các sai số kiểm tra giảm tới một mức độ nhất định trong quá trình huấn luyện và sau đó lại tăng lên. Sự lăng này thể hiện điểm mà lại đó việc

điều chỉnh mẫu đã vượt quá giới hạn. Anr's chọn các tham số mô hình có liên quan đến sai số kiểm tra tối thiểu. Trong ví dụ này, tuỳ chọn kiểm tra của anrs có hữu ích.

+ Thử dữ liệu tương phản với FIS huấn luyện (Testing Your Data Against thể Traincd FIS)



Hình6.18

Để thử FIS tương phản với dữ liệu kiểm tra, chọn hộp kiểm tra **Checking data** trong phần **Tesst FIS** của GUI, và bấm **Test now**. Ta kiểm tra được dữ liệu kiểm tra tương phản với FIS (hình 6.18).

Chú ý:

☞ **Khi tải thêm dữ liệu với anfis**: Nếu tải dữ liệu mới vào anfis sau khi đã xoá dữ liệu đã tải trước đó, ta phải chắc chắn rằng dữ liệu mới tải có cùng số đầu vào như dữ liệu trước. Nếu không ta cần phải khởi động lại một **anfisedit** mới từ dòng lệnh.

☞ **Khi lựa chọn dữ liệu kiểm tra và xoá dữ liệu**: Nếu không muốn sử dụng phần lựa chọn dữ liệu kiểm tra của anfis, ta không cần tải bất cứ một dữ liệu kiểm tra nào trước khi huấn luyện FIS. Nếu ta quyết định huấn luyện lại FIS mà không cần dữ liệu kiểm tra, ta có thể loại bỏ dữ liệu kiểm tra bằng một trong hai cách:

- Cách 1: Lựa chọn nút **Checking** trong phần **Load data** của GUI và sau đó bấm vào nút **Clear Data** để bỏ dữ liệu kiểm tra.

- Cách 2: Đóng GUI và đi tới dòng lệnh rồi gõ lại anfisedit. Trong trường hợp này có thể phải tải lại dữ liệu kiểm tra. Sau khi xóa dữ liệu, ta sẽ cần tạo FIS.

☞ **Khi một FIS đã được tạo ra ta** có thể sử dụng kinh nghiệm huấn luyện kỳ đầu tiên để quyết định số kỳ huấn luyện nếu muốn huấn luyện kỳ hai.

b. **Ví dụ 2: Dữ liệu kiểm tra không hợp với mô hình**

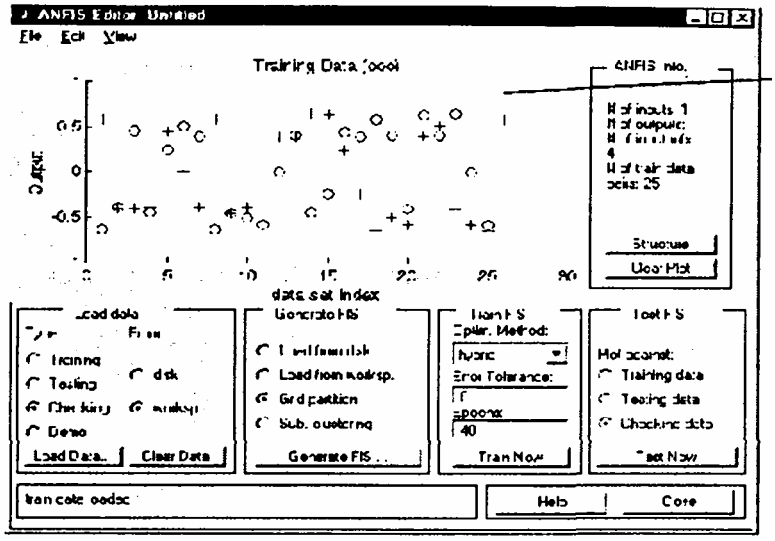
Trong ví dụ này, chúng ta sẽ khảo sát điều gì xảy ra khi dữ liệu huấn luyện và kiểm tra khác nhau hoàn toàn. Để xem cách sử dụng bộ soạn thảo ANFIS GUI để nghiên cứu một số điều về các tập dữ liệu và sự khác nhau giữa chúng. Ta thực hiện theo các bước sau:

☞ Xoá tất cả dữ liệu huấn luyện và dữ liệu kiểm tra. Để làm điều đó có thể bấm vào nút **Clear Plot** bên phải của cửa sổ soạn thảo.

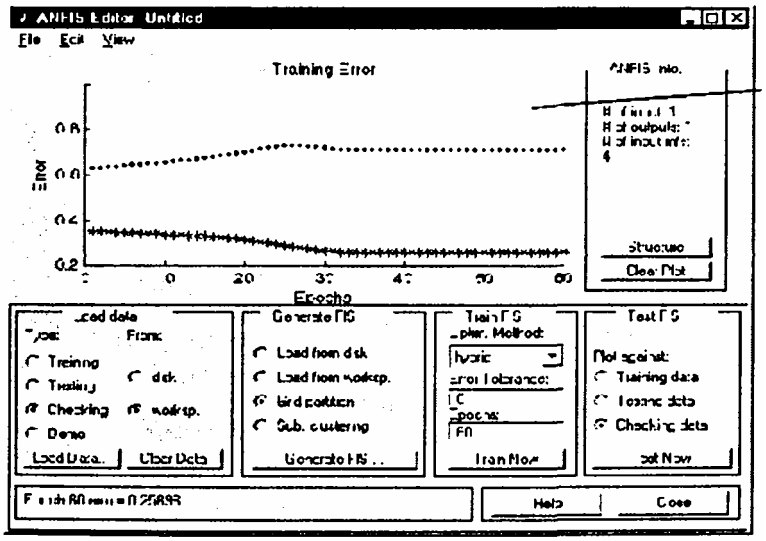
☞ Tải dữ liệu huấn luyện và dữ liệu kiểm tra theo thứ tự định sẵn **fuzex2trnData** và **fuzex2chkData** vào vùng làm việc của MATLAB tương tự như đã làm ở ví dụ trước.

Trên màn hình ta nhận được những dữ liệu như hình 6.19.

Huấn luyện FIS cho hệ thống này tương tự như ở ví dụ trước, chỉ khác là chọn kỳ huấn luyện là 60 trước khi huấn luyện. Ta nhận được sai số huấn luyện và sai số kiểm tra (hình 6.20).



Hình 6.19

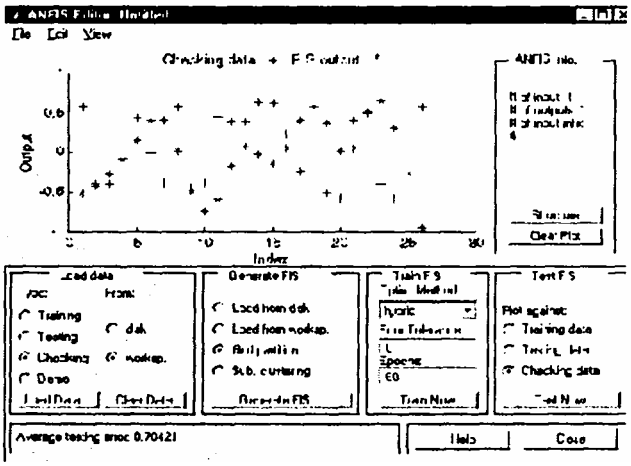


Hình 6.20

Ta thấy rằng sai số kiểm tra là khá lớn. Dường như là sai số kiểm tra nhỏ nhất lấy ra trong kỳ đầu tiên. Chúng ta hãy nhớ lại việc sử dụng tuỳ chọn dữ liệu kiểm ra bằng anfis một cách tự động để thiết lập các tham số có liên kết với sai số kiểm tra nhỏ nhất. Rõ ràng là tập các hàm liên thuộc không phải là lựa chọn tốt nhất để tạo mẫu dữ liệu kiểm tra.

Vậy, vấn đề ở đây là gì? Ví dụ này làm sáng tỏ vấn óc đã được đề cập ở trên, trong đó dữ liệu kiểm tra đưa đến anfis để huấn luyện khác hoàn toàn với dữ liệu huấn luyện. Nó thể hiện tầm quan trọng của việc.hiểu biết rõ các đặc điểm dữ liệu khi ta lựa chọn dữ liệu huấn luyện và kiểm tra. Nếu không đúng, ta có thể phân tích sơ đồ sai số kiểm tra để xem dữ liệu kiểm tra có hoạt động hiệu quả với mô hình huấn luyện hay không. Trong ví dụ này, sai số kiểm tra đủ lớn để cho thấy cần phải có nhiều dữ liệu hơn để lựa chọn cho việc huấn luyện hoặc thay đổi các lựa chọn hàm liên thuộc (cả số lượng và kiêu hàm liên thuộc). Hoặc là huấn luyện lại hệ thống mà không cân dữ liệu kiểm tra nếu ta thấy dữ liệu huấn luyện đủ mô tả những nét đặc trưng mà ta đang cố gắng thực hiện.

Sau đây ta hãy thử huấn luyện mô hình FIS đối lập với dữ liệu kiểm tra. Để làm việc này, lựa chọn **Checking data** trong mục **Test FIS** của GUI, và bấm vào **Test Now**. Hình 6.21 dưới đây thể hiện sự khác nhau giữa đầu ra dữ liệu kiểm tra với đầu ra FIS.



Hình 6.21

6.6. SOẠN THẢO ANFIS TỪ DÒNG LỆNH

Ở trên ta đã thấy rõ số tiện lợi và đơn giản khi sử dụng bộ soạn thảo ANFIS GUI để xây dựng một mô hình mờ - nơron (FIS). Tuy nhiên, như đã thấy trongví dụ 2, ta cần phái thận trọng khi thực hiện chức năng công nhận giá trị dữ liệu kiểm tra của anfis và phải kiểm tra xem sai số dữ liệu kiểm tra có đảm bảo không. Nếu không ta cần huấn luyện lại FIS.

Sau đây chúng ta sẽ tìm hiểu cách thức sử dụng dòng lệnh anfis thông qua ví dụ dự đoán chuỗi thời gian hỗn độn.

Chuỗi thời gian hỗn độn được mô tả bởi phương trình Mackey- Glass (MG) như sau:

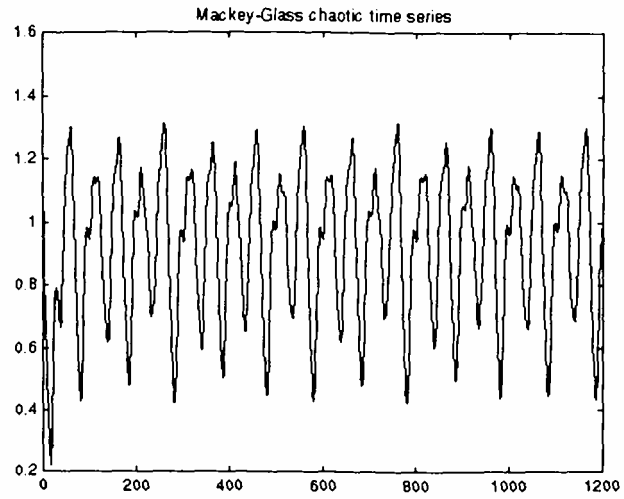
$$\dot{x}(t) = \frac{0{,}2x(-\tau)}{1+x^{10}(t-\tau)} - 0{,}1x(t)$$

Chuỗi thời gian này là hỗn độn, vì không có khoảng thời gian xác định rõ ràng. Chuỗi này sẽ không hội tụ hay phân kỳ, và đường cong có độ nhạy cao với các điều kiện ban đầu. Đây là một bài toán điển hình trong số các nghiên cứu mô hình mạng nơron và logic mờ.

Để thu được giá trị chuỗi thời gian tại những giá trị nguyên, chúng ta áp dụng phương pháp Runge-kuna bậc 4 để tìm ra giải pháp số cho công thức (Mackey- Glass) MG ở trên, kết quả được lưu lại trong file **mgdata.dat**. Ở đây, ta giả thiết x(0) = 1,2, T = 17 và x(t) = 0 khi t < 0. Để vẽ đồ thị chuỗi thời gian MG (hình 6.22) ta thực hiện các lệnh sau:

load mgdata.dai

t = mgdata (: , 1); x - mgdata (:, 2); plot(t, x);



Hình 6.22. Đồ thị chuỗi MG

Để dự đoán chuỗi thời gian, ta mong muốn dùng các giá trị của chuỗi thời gian đến điểm thời điểm t, để dự đoán giá trị của chuỗi tại một số thời điểm trong tương lai (t + P). Phương pháp chuẩn cho loại dự đoán này là tạo ra một bản đồ từ các điểm lấy mẫu dữ liệu D, thời gian trích mẫu là Δ. Tại thời điểm, (x(t - (D-1) Δ),... x(t- Δ x(t)) để dự đoán các giá trị x(t + P). Theo quy ước để dự đoán chuỗi thời gian MG, chúng ta đặt D = 4 và Δ = P = 6. Với mỗi t, dữ liệu huấn luyện đầu vào cho anfls là một vecto bốn chiều theo công thức sau:

$$w(t) = [x(t- 18)\ x(t- 12)\ x(t-6)\ x(t)].$$

Dữ liệu huấn luyện đầu ra tương ứng với đường cong dự đoán:

$$s(t) = x(t + 6).$$

Với mỗi t, theo giá trị từ 118 đến 1117, dữ liệu huấn luyện đầu vào/ đầu ra sẽ là một cấu trúc có thành phần đầu tiên là đầu vào w bốn hướng, và thành phần thứ hai là đầu ra s. Sẽ có 1000 giá trị dữ liệu vào/ ra. Chúng ta sử dụng 500 giá trị dữ liệu đầu tiên để huấn luyện anfis (những giá trị này trở thành dữ liệu huấn luyện) trong khi đó những giá trị khác được sử dụng như dữ liệu kiểm tra để nhận dạng mô hình mờ. Đây là kết quả trong 2 cấu trúc dữ liệu 500 điểm, tmData và chkData.

Dưới đây là các lệnh để tạo dữ liệu này.

For t = 118: 1117,

Data (t- 117, :) = [x(t-18) x(t-12) x(t-6) x(t) x(t + 6)];

End

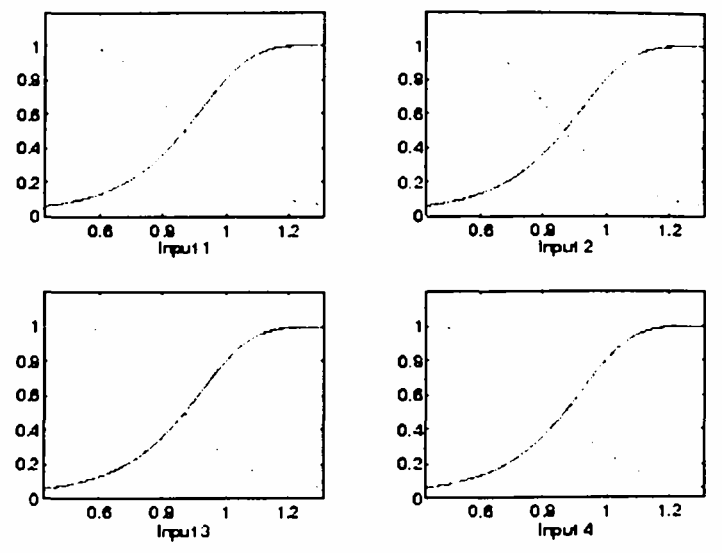
trnData : Data (1:500, :);

chkData = Data (501 : end, :);

Để bắt đầu việc huấn luyện, chúng ta cần một cấu trúc FIS chỉ rõ các tham số ban đầu và cấu trúc của FIS để học. Đây là công việc của genrsl:

fismat = genfisl (trnData).

Vì chúng ta không xác định rõ số lượng và kiểu hàm liên thuộc sử dụng trong FIS nên các giá trị mặc định không có thật. Những giá trị mặc định này cung cấp hai hàm liên thuộc hình chuông ở mỗi đầu vào, tổng cộng là 8. Cấu trúc tạo FIS có 16 luật mờ với 140 tham số. Theo thứ tự, để đạt khả năng phát tốt, quan trọng là phải có các điểm dữ liệu huấn luyện lớn gấp vài lần số các tham số ước tính. Trong trường hợp này, tỷ lệ giữa dữ liệu và các tham số vào khoảng 5 (500/104).



Hình 6.23. Các hàm liên thuộc trước khi huấn luyện

Hàm **genfisl** tạo ra các hàm liên thuộc ban đầu được sắp xếp cân bằng và bao phủ tất cả khoảng đầu vào. T có thể vẽ các hàm liên thuộc đầu bằng các lệnh sau:

subplot(2,2,1)

plotmf(rsmat, 'input', 1)

suhplot(2,2,2)

plotmf(rsmat, 'input', 2)

subplot(2,2,3)

plotmf(rsmat, 'input', 3)

subplot(2,2,4)

plotmf(rsmat, 'input', 4)

Các hàm liên thuộc ban đầu như hình 6.23.

Để bắt đầu việc huấn luyện, gõ:

[rsmat1, error1,ss, rsmat2, error2]=...

anrs (trnData, fismat, [],[], chkData);

Vì tuỳ chọn dữ liệu kiểm tra anrs được gọi ra, FIS chọn cuối cùng sẽ thường là một liên kết với sai số kiểm tra nhỏ nhất. Nó được lưu trong rsmat2. Các lệnh sau sẽ vẽ các hàm liên thuộc mới:

subplot(2,2,1)

plotmf(fismat2, 'input', 1)

subplot(2,2,2)

plotmf(rsmat2, 'input', 2)

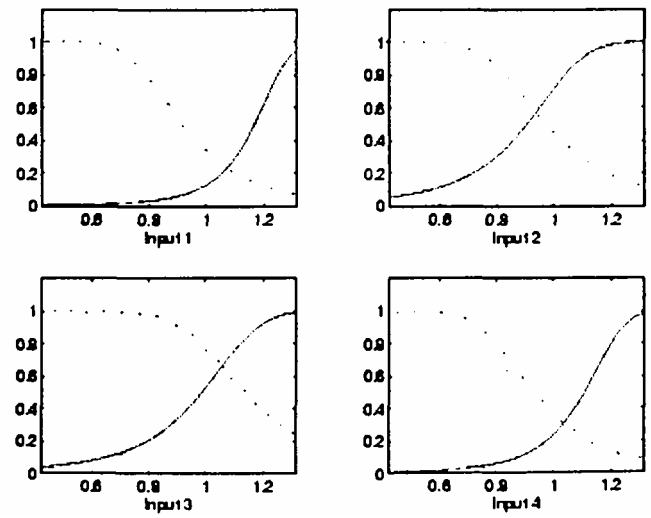
subplot(2,2,3)

plotmí(rsmat2, 'input', 3)

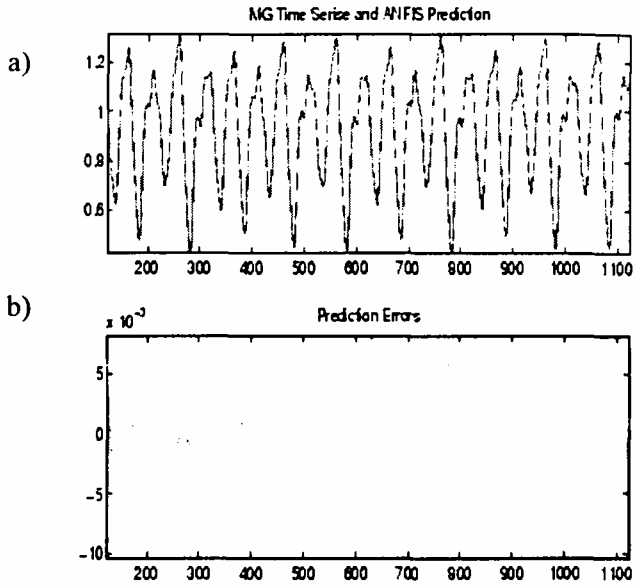
subplot(2,2,4)

plotmf(rsmat2, 'input', 4)

Hàm liên thuộc mới được chi ra trên hình 6.24.



Hình 6.24. Các hàm liên thuộc sau khi huấn luyện



Hình 6.25a,b. Tín hiệu ra và sai số sau khi huấn luyện

Để vẽ đồ thị sai số tín hiệu, gõ lệnh:

plot ([error1; error2]);

Ở đây **error1** và **error2** là bình phương trung bình sai số dữ liệu huấn luyện và kiểm tra theo thứ tự định sẵn.

Thêm vào các đồ thị sai số này, ta có thể muốn vẽ đồ thị đầu ra FIS trái với dữ liệu huấn luyện và kiểm tra. Để so sánh chuỗi thời gian MG gốc và dự đoán logic mờ từng mặt một, hãy thử

anfis_output = evamS([trnData; chkData1], fismat2);

index = 125:1124;

subplot(211), plot(t(index), [x(index) anfisoutput]);

subplot(212), plot(t(index), x(index) - anfis_output);

Chú ý: Sự khác nhau giữa chuỗi thời gian MG gốc và các giá trị anrs ước tính là rất nhỏ. Hai đồ thị gần như trùng khít lên nhau (hình 6.sa). Sai số giữa chúng được chi ra trên đồ thị hình 6.25b với tỉ lệ mịn hơn nhiều. Trong ví dụ này, ta mới chi huấn luyện cho 10 kỳ. Nếu thêm số kỳ huấn luyện, chúng ta sẽ thu được kết quả tốt hơn.

6.7. THÔNG TIN THÊM VỀ ANFIS VÀ BỘ SOẠN THẢO ANFIS EDITOR GUI

Lệnh anfis có ít nhất là 2 và nhiều nhất là 6 đối số đầu vào, dạng tổng

quát là:

[fismat1,trnError,ss,fismat2,chkError] = ...

anfis(trnData,fismat,trnOpt,dispOpt,chkData,method);

Ở đây **trnOpt** (các tuỳ chọn huấn luyện), dispopt (các tuỳ chọn hiển thị), **chkData** (dữ liệu kiểm tra) và method (phương pháp huấn luyện) là tuỳ chọn. Tất cả các đối số đầu ra đều là tuỳ chọn. Trong phần này chúng ta bàn đến các đối số và phạm vi các thành phần của hàm dòng lệnh **anfis**, cũng như hàm tương tự của bộ soạn thảo ANFIS GUI.

Khi bộ soạn thảo ANFIS GUI được gọi ra sử dụng anfsedit, chỉ có tập dữ liệu huấn luyện được ưu tiên hơn việc thực hiện anrs. Thêm vào đó, bước tính sẽ được cố định khi hệ thống nơron mờ thích nghi được huấn luyện sử dụng dụng cụ này.

6.7.1. Dữ liệu huấn luyện (Training Data)

Dữ liệu huấn luyện, t**rnData**, là một đối số yêu cầu với **anfis**. Cũng như bộ soạn thảo ANFIS GUI. Mỗi dòng của **trnData** là một cặp vào/ra của hệ thống đích đã được mô hình hóa. Mỗi dòng bắt đầu bằng một vecto đầu vào và theo sau bởi một giá trị đầu ra. Vì thế số dòng của **trnData** bằng với số cặp dữ liệu huấn luyện và do chỉ có một đầu ra nên số cột của **trnData** bằng với số đầu vào cộng 1**.**

6.7.2. Cấu trúc đầu vào FIS (Input FIS Structure)

Cấu trúc đầu vào FIS, **fismat**, có thể có được từ một trong số các bộ soạn thảo logic mờ bất kỳ: Bộ soạn thảo FIS, bộ soạn thảo hàm liên thuộc và bộ soạn thảo luật từ bộ soạn tháo ANFIS GUI, (cho phép tải một cấu trúc FIS từ đĩa hoặc vùng làm việc), hoặc từ một hàm dòng lệnh, **genfis1** (chỉ cần đưa số lượng và kiểu hàm liên thuộc). Cấu trúc FIS chứa cả cấu trúc mô hình (như số luật trong FIS, số hàm liên thuộc cho mỗi đầu vào,...) và các tham số, (như dạng hàm liên thuộc).

Có hai phương pháp mà **anfis** huấn luyện để cập nhật các tham số của liên thuộc: truyền ngược đối với tất cả tham số (Hạ thấp độ dốc - a steepest descent method) và phương pháp lai (hybrid) bao gồm cả việc truyền ngược đối với các tham số có liên quan đến các hàm liên thuộc đầu vào, và ước lượng bình phương nhỏ nhất với các tham số có liên quan với các hàm liên thuộc đầu ra. Kết quả là, sai số huấn luyện giảm, ít nhất là trong khu vực, qua quá trình luyện. Vì vậy, càng nhiều hàm liên thuộc ban đầu giống các hàm tối ưu thì càng dễ hội tụ các tham số huấn luyện mô hình. Sự hiểu biết của con người về hệ thống đích được dùng làm mẫu có thể giúp cho việc thiết lập các hàm tham số của hàm liên thuộc ban đầu trong cấu trúc FIS.

genfis1 tạo ra một cấu trúc FIS dựa trên nhiều hàm liên thuộc cố định. Điều này gây phiền phức và tạo ra rất nhiều nguyên tắc khi số đầu vào tương đối nhiều (4 đến 5 đầu vào). Để giảm kích cỡ trong hệ thống suy luận mờ ta có thể tạo ra một cấu trúc FIS sử dụng thuật toán hồi (tham khảo mục Subtractive Clustering) từ bộ soạn thảo ANFIS GUI, thuật toán này được lựa chọn trước khi tạo FIS. Phương pháp hồi quy phân chia dữ liệu thành các nhóm gọi là clusters và tạo ra một FIS với các nguyên tắc tối thiểu theo yêu cầu để phân biệt các tính chất mờ có liên quan đến mỗi cluster.

6.7.3. Các tùy chọn huấn luyện (Training Options)

ANFIS Editor GUI cho phép ta lựa chọn dung sai sai số và số kỳ huấn luyện theo ý muốn.

Tuỳ chọn huấn luyện **trnOpt** cho dòng lệnh **anfis** là một vecto định rõ tiêu chuẩn dừng và cỡ bước tính của quá trình thích nghi như sau:

trnOpt(1): number of training epochs, default =10.

trnOpt(2): error tolerance, default = 0.

trnOpt(3): initial step-size, default = 0.01.

trnOpt(4): step-size decrease nhe, default = 0.9.

trnOpt(5): step-size increase nhe, default = 1.1.

Nếu một thành phần bất kỳ của trnopt là một NaN hoặc thiếu, giá trị mặc định sẽ được lấy. Quá trình huấn sẽ luyện dừng khi đã chạy đủ kỳ huấn luyện thiết kế hoặc khi sai số nằm trong giới hạn cho trước sau một số kỳ huấn luyện.

Thông thường ta muốn mô tả sơ lược cỡ của bước tính là một đường cong tăng vào lúc đầu, đạt tối đa, và sau đó giảm trong thời gian huấn luyện còn lại. Mô tả sơ lược cỡ bước lý tưởng đạt được bằng cách điều chỉnh cỡ bước ban đầu và tỷ lệ tăng, giảm (tmopt (3) - tmopt (5)). Giá trị mặc định được đặt cho một vùng rộng các nhiệm vụ nghiên cứu. Để ứng dụng đặc biệt, ta có thể muốn sửa đổi những tuỳ chọn cỡ bước để đánh giá tích cực việc huấn luyện. Tuy nhiên, như đã được nói đến ở phần trước, có những tuỳ chọn cỡ bước không được sử dụng đặc biệt để huấn luyện hệ thống noron suy luận mờ thích ứng được tạo để sử dụng bộ soạn thảo ANFIS GUI.

6.7.4 Tuỳ chọn hiển thị Display Options

Tuỳ chọn hiển thị chỉ ứng dụng cho hàm lệnh anrs. Đối với dòng lệnh anrs, đối số tuỳ chọn hiển thị, disopt, là một vectơ bằng 1 hoặc bằng 0 chỉ rõ thông tin hiển thị (in trong cửa sổ dòng lệnh MATLAB), trước, trong và sau

quá trình huấn luyện. Số '1' được sử dụng để biểu thị in phần lựa chọn trong khi 0 biểu thị không in phần chọn.

disOpt(1): hiển thị thông tin, mặc định = 1

dissOpt(2): hiển thị sai số (mỗi kỳ), mặc định = 1

disOpt(3): hiển thị cỡ bước (mỗi kỳ), mặc định = 1

disOpt(4): hiển thị các kết quả cuối cùng, mặc định = 1

Mẫu mặc định hiển thị tất cả các thông tin có sẵn. Nếu một phụ của **disOpt** là NaN hoặc bị mất đi thì giá trị mặc định sẽ được lấy.

6.7.5. Phương pháp huấn luyện (Method)

Cả bộ soạn thảo ANFIS GUI và dòng lệnh **anfis** đều áp dụng dạng thức truyền ngược của phương pháp hạ thấp độ dốc để ước tính các tham số hàm liên thuộc hoặc kết hợp giữa dạng truyền ngược và phương pháp bình phương nhỏ nhất để tính các hàm tham số hàm liên thuộc. Các đối số chọn là hybrid hoặc backpropagation. Các lựa chọn phương pháp này được thiết kế trong hàm dòng lệnh, anfis, = 1 và 0, theo thứ tự mặc định.

6.7.6. Cấu trúc đầu ra FIS cho dữ liệu huấn 1uyện

Fismat1 là cấu trúc đầu ra FIS tương ứng với sai số huấn luyện nhỏ nhất. Đây là cấu trúc FIS mà ta sẽ sử dụng để trình bày hệ thống suy luận mờ khi không có dữ liệu kiểm tra để công nhận giá trị của mẫu. Dữ liệu này cũng trình bày cấu trúc FIS được lưu bởi bộ soạn thảo ANFIS GUI khi tuỳ chọn dữ liệu kiểm tra không được sử dụng. Khi dữ liệu kiểm tra được sử dụng, đầu ra được lưu là đầu ra có liên hệ với sai số kiểm tra nhỏ nhất.

6.7.7. Sai số huấn luyện

Sai số huấn luyện là sự khác nhau giữa giá trị dữ liệu đầu ra, và đầu ra của hệ thống suy luận mờ tương ứng với giá trị dữ liệu đầu vào (có liên hệ với giá trị dữ liệu huấn luyện đầu ra). Sai số huấn luyện **trnError** ghi lại sai số bình phương nhỏ nhất ban đầu (RMSE) của dữ liệu ở mỗi kỳ huấn luyện. **Fismat1** là ghi nhận nhanh của cấu trúc FIS khi phép đo sai số huấn luyện nhỏ nhất. Bộ soạn thảo ANFIS GUI sẽ vẽ sơ đồ sai số huấn luyện và các đường cong ở mỗi kỳ khi huấn luyện hệ thống.

6.7.8. Bước tính (Step-size)

Trong bộ soạn thảo ANFIS GUI ta không thể tùy ý chọn bước tính. Sử dụng dòng lệnh **anfis**, chuỗi các cỡ bước **ss** ghi lại cỡ nước trong suốt quá trình huấn luyện. Vẽ đồ thị **ss** mô tả sơ bộ cỡ bước như một phần tham khảo để điều chỉnh cỡ bước theo tỷ lệ tăng, giảm tương ứng. Cỡ bước **ss** cho hàm

lệnh **anfis** được cập nhật theo hướng dẫn sau:

+ Nếu sai số giảm 4 lần liên tiếp, ta tăng bước tính bằng cách nhân nó với một hằng số (ssinc) lớn hơn 1.

+ Nếu sai số tăng và giảm 1 lần liên tiếp, giảm bước tính bằng cách nhân nó với một hằng số (ssdec) nhỏ hơn 1.

Giá trị mặc định cho bước ban đầu là 0.01, các giá trị mặc định cho ssinc và ssdec là 1.1 và 0.9 theo thứ tự mặc định. Tất cả các giá trị mặc định có thể được thay đổi theo tuỳ chọn huấn luyện của lệnh **anfis**.

6.7.9. Dữ liệu kiểm tra (Checking Data)

Dữ liệu kiểm tra, **chkData**, được sử dụng để kiểm năng lực tổng quát của hệ thống suy luận mờ ở mỗi kỳ huấn luyện. Dữ liệu kiểm tra có cùng dạng với dữ liệu huấn luyện, nhưng nó có các phần tử nói chung khác biệt với dữ liệu huấn luyện.

Dữ liệu kiểm tra rất quan trọng đối với việc nghiên cứu hệ thống có số đầu vào lớn, hoặc với dữ liệu bị nhiễu. Nói chung chúng ta muốn hệ thống suy luận mờ **train** theo một dữ liệu vào/ra đã có thật tốt. Vì cấu trúc mô hình sử dụng sử dụng cho **anfis** là cố định nên có có thể xảy ra trường hợp khả năng của mô hình mô hình vượt quá dữ liệu huấn luyện, đặc biệt là số kỳ huấn luyện lớn. Nếu việc đó xảy ra, hệ thống suy luận mờ sẽ không thể đáp ứng tốt với các phần dữ liệu độc lập khác, đặc biệt khi chúng bị tác động của nhiễu. Một dữ liệu kiểm tra hoặc được công nhận giá trị có thể hữu ích trong những tình huống này. Dữ liệu này được sử dụng để công nhận giá trị của mô hình suy luận mờ. Nó được thực hiện bằng cách áp dụng dữ liệu kiểm tra cho mô hình, và xem việc mô hình phản hồi tới dữ liệu tốt như thế nào.

Khi tuỳ chọn dữ liệu kiểm tra được sử dụng với **anfis**, hoặc là theo dòng lệnh hoặc là sử dụng bộ soạn thảo ANFIS GUI, dữ liệu kiểm tra được áp dụng cho mẫu ở mỗi kỳ huấn luyện. Khi dòng lệnh **anfis** được gọi ra, các tham số mô hình tương ứng với sai số kiểm tra nhỏ nhất trở lại theo đối số đầu ra **fismat2**. Các hàm tham số hàm liên thuộc FIS được tính bằng cách sử dụng bộ soạn thảo ANFIS GUI khi hai dữ liệu huấn luyện và kiểm tra đã được nạp vào kết hợp với kỳ huấn luyện có sai số kiểm tra nhỏ nhất.

Kỳ sai số dữ liệu kiểm tra nhỏ nhất dùng để thiết lập các hàm tham số của hàm liên thuộc thừa nhận:

- Dữ liệu kiểm tra đủ giống dữ liệu huấn luyện để sai số dữ liệu kiểm tra giảm khi quá trình huấn luyện bắt đầu.

- Dữ liệu kiểm tra tăng tại một vài điểm của quá trình huấn luyện, sau đó xảy ra việc điều chỉnh quá mức.

Dựa vào đường cong sai số của dữ liệu kiểm tra để kết luận FIS có thể được sử dụng hoặc không được sử dụng.

6.7.10. Cấu trúc đầu ra FIS cho dữ liệu kiểm tra (Output FIS Structure for Checking Data)

Đầu ra của dòng lệnh **anfis**, **fismat2** là cấu trúc đầu ra FIS có sai số kiểm tra nhỏ nhất. Đây là cấu trúc FIS được sử dụng để tính toán thẩm nếu dữ liệu kiểm tra được sử dụng cho quá trình công nhận giá trị.

6.7.11. Sai số kiểm tra (Checking Error)

Sai số kiểm tra là sự khác nhau giữa giá trị dữ liệu kiểm tra đầu ra và đầu ra của hệ thống suy luận mờ tương ứng với giá trị dữ liệu kiểm tra đầu vào (có kết hợp với giá trị dữ liệu kiểm tra đầu ra). Sai số kiểm tra **chkError** ghi lại RMSE cho dữ liệu kiểm tra ở mỗi kỳ. **Fismat2** là một minh họa của cấu trúc FIS khi sai số kiểm tra ở mức nhỏ nhất. Bộ soạn thảo ANFIS GUI sẽ vẽ sơ đồ sai số kiểm tra cùng với đường cong các kỳ khi huấn luyện hệ thống.

TÀI LIỆU THAM KHẢO

[1] Phan Xuân Minh & Nguyễn Doãn Phước (1999), *"Lý thuyết điều khiển mờ"*, Nhà xuất bản khoa học và kỹ thuật.

[2] Nguyễn Thương Ngô (1998), *"Lý thuyết điều khiển tự động hiện đại",* Nhà xuất bản khoa học và kỹ thuật.

[3] Nguyễn Như Hiển (2003) *"Nghiên cơ nâng cao chất lượng hệ chuyển động nhiều trục* ", Luận án tiến sĩ kỹ thuật.

[4] Lại Khắc Lãi (2003), *"Một số phương pháp tổng hợp tối ưu bộ điều khiển trên cơ sở logic mờ và thích nghi"*, Luận án tiến sĩ kỹ thuật.

[5] Phan Xuân Minh, Nguyễn Tiến Hiếu *"Điều khiển thích nghi tay máy trên* cơ sở *hệ mờ"* Tuyển tập các báo cáo khoa học hội nghị tự động hóa toàn quốc lần thứ 6 (2005), trang 370-375.

[6] Phạm Hữu Đức Dục *"Nghiên cứu ứng dụng tương nơron mờ điều khiển thích nghi rôbôt hai khâu"* Tuyển tập các báo cáo khoa học hội nghị tự động hóa toàn quốc lần thứ 6 (2005), trang 107- 112.

[7] H. Yamamoto & T. Furuhashi (1999), *"New fuzzy Inference methodfor system Using symbolic stability analysis offuzzy control"*, The fourth Asian Fuzzy System Synposium, May 31 - June, Tsukuba, Japan. pp.450-455.

[8] Kenji IKEDA and Seiichi SHIN (2000), *"Autonomous Decentralizrd Adaptive Control System Using Parameter Projection",* The fourth Asian Fuzzy System Synposium, May 31 - June, Tsukuba, Japan. pp.293-298.

[9] Huganghan & Shuta Murakami (2000), *"Adaptive Fuzzy Control of Nonlinear system Witht Various Shapes of Membership Funetions",* The fouth Asian Fuzzy Systems Symosium, may 31 -June 3.2000 Tshkuba, Japan. pp.426-467.

[10] Kosko, B (1991), *"Neuralnetworks andfuzzy control",* Prentice Hall,.

[11] L.A. Zadeh (1965), *"fuzzy sét",* Inf. contr. vol. 8, pp. 338-353.

[12] Yih-Guang Leu, Wei-Yen Wang and Tsu-Tian lee (1999), *"Robust Adaptive Fuzzy-Neural controllers for Uncertain Nonlinear Systems",* IEEE Transaction on roboties and automation. Vol. 15. No. 5, pp. 805 - 817.

[13] Matlab 7.4

[14] research reports Esprit, I.F.Croall, J.P.Mason) Industrial Applications of Neural Networks.

NHÀ XUẤT BẢN KHOA HỌC TỰ NHIÊN VÀ CÔNG NGHỆ
18 đường Hoàng Quốc Việt, Cầu Giấy, Hà Nội
Điện thoại
Phòng phát thanh: 04.2149040; Biên tập: 04.2149034
Quản lý tổng hợp:04.2149041
Fax: 04.7910147, Email: nxb@vap.ac.vn; www.vap.ac.vn

HỆ MỜ VÀ NƠRON TRONG KỸ THUẬT ĐIỀU KHIỂN

Tác giả:
TS. Nguyễn Như Hiển và TS. Lại Khắc Lãi

Chịu trách nhiệm xuất bản:
GS.TSKH Nguyễn Khoa Sơn

Tổng biên tập:
GS.TSKH Nguyễn Khoa Sơn

Kỹ thuật vi tính: **Lê Thị Thiên Hương**
Trình bày bìa: **Vương Quốc Cường**

In 1000 cuốn khổ 16 x 24 tại Nhà in Khoa Học và Công Nghệ
Số đăng ký KHXB: 124-2007/CXB/012-10/KHTNVCN
Cấp ngày 9 tháng 2 năm 2007.