

TS. NGUYỄN NHƯ HIỀN & TS. LẠI KHẮC LÃI

HỆ MỜ & NƠN

TRONG KỸ THUẬT ĐIỀU KHIỂN

Sách Chuyên khảo dùng cho đào tạo Sau đại học ngành Điều khiển & Tự động hoá

NHÀ XUẤT BẢN KHOA HỌC TỰ NHIÊN VÀ CÔNG NGHỆ

HÀ NỘI – 2007

MỤC LỤC

LỜI NÓI ĐẦU	6
Chương 1: LÔGIC MỜ.....	1
1.1. TỔNG QUAN VỀ LÔGIC MỜ.....	1
1.1.1. Quá trình phát triển của lôgic mờ	1
1.1.2. Cơ sở toán học của lôgic mờ.....	1
1.1.3. Lôgic mờ là lôgic của con người	2
1.2. KHÁI NIỆM VỀ TẬP MỜ.....	3
1.2.1. Tập kinh điển	3
1.2.3. Các thông số đặc trưng cho tập mờ	4
1.2.4. Các dạng hàm liên thuộc của tập mờ.....	5
1.3. CÁC PHÉP TOÁN TRÊN TẬP MỜ.....	5
1.3.1. Phép hợp hai tập mờ	5
1.3.2. Phép giao của hai tập mờ.....	6
1.3.3. Phép bù của một tập mờ	8
1.4. BIẾN NGÔN NGỮ VÀ GIÁ TRỊ CỦA BIẾN NGÔN NGỮ.....	8
1.5. LUẬT HỢP THÀNH MỜ.....	9
1.5.1. Mệnh đề hợp thành	9
1.5.2. Mô tả mệnh đề hợp thành	9
1.5.3. Luật hợp thành mờ.....	10
1.5.4. Các cấu trúc cơ bản của luật hợp thành	11
1.5.5. Luật hợp thành đơn có cấu trúc SISO.....	12
1.5.7. Luật của nhiều mệnh đề hợp thành.....	19
1.5.7. Luật hợp thành SUM-MIN và SUM-PROD.....	22
1.6. GIẢI MỜ	23
2.6.1. Phương pháp cực đại	24
Chương 2: ĐIỀU KHIỂN MỜ.....	29
2.1. CẤU TRÚC CỦA BỘ ĐIỀU KHIỂN MỜ.....	29
2.1.1. Sơ đồ khối bộ điều khiển mờ.....	29
2.1.2. Phân loại bộ điều khiển mờ	30
2.1.3. Các bước tổng hợp bộ điều khiển mờ.....	31
2.2. BỘ ĐIỀU KHIỂN MỜ TĨNH.....	32

2.2.1. Khái niệm.....	32
2.2.2. Thuật toán tổng hợp một bộ điều khiển mờ tĩnh	32
2.2.3. Tổng hợp bộ điều khiển mờ tuyến tính từng đoạn.....	33
2.3. BỘ ĐIỀU KHIỂN MỜ ĐỘNG.....	35
2.4. THIẾT KẾ HỆ ĐIỀU KHIỂN MỜ BẰNG PHẦN MỀM MATLAB .	37
2.4.1. Giới thiệu hộp công cụ logic mờ	37
2.3.2. Ví dụ thiết kế hệ mờ	41
2.5. HỆ ĐIỀU KHIỂN MỜ LAI (F-PID)	45
2.6. HỆ ĐIỀU KHIỂN THÍCH NGHI MỜ	46
2.6.1. Khái niệm.....	46
2.6.2. Tổng hợp bộ điều khiển thích nghi mờ ổn định.....	48
2.7. TỔNG HỢP BỘ ĐIỀU KHIỂN MỜ THÍCH NGHI TRÊN CƠ SỞ LÝ THUYẾT THÍCH NGHI KINH ĐIỂN.....	58
2.7.1. Đặt vấn đề	58
2.7.2. Mô hình toán học của bộ điều khiển mờ	60
2.7.3. Xây dựng cơ cấu thích nghi cho bộ điều khiển mờ	66
2.7.4. Một số ứng dụng điều khiển các đối tượng công nghiệp.....	70
Chương 3: TỔNG QUAN VỀ MẠNG NƠRON	75
3.1. NƠRON SINH HỌC	75
3.1.1. Chức năng, tổ chức và hoạt động của bộ não con người	75
3.1.2. Mạng nơron sinh học	76
3.2. MẠNG NƠRON NHÂN TẠO	77
3.2.1. Khái niệm.....	77
3.2.2. Mô hình nơron	80
3.3. CẤU TRÚC MẠNG	83
3.3.1. Mạng một lớp.....	83
3.3.2. Mạng nhiều lớp.....	84
3.4. CẤU TRÚC DỮ LIỆU VÀO MẠNG	87
3.4.1. Mô tả vectơ vào đối với mạng tĩnh.....	88
3.4.2. Mô tả vectơ vào liên tiếp trong mạng động	89
3.5. HUẤN LUYỆN MẠNG	92
3.5.1. Huấn luyện gia tăng	92
3.5.2 Huấn luyện mạng theo gói.....	94
Chương 4: MẠNG PERCEPTRONS	98
4.1. MỞ ĐẦU	98

4.1.1. Mô hình nơron perceptron	98
4.1.2. Kiến trúc mạng perceptron	100
4.2. THIẾT LẬP VÀ MÔ PHỎNG PERCEPTRON TRONG MATLAB	100
4.2.1 Thiết lập	100
4.2.2. Mô phỏng (sim)	102
4.2.3. Khởi tạo	103
4.3. CÁC LUẬT HỌC	104
4.3.1. Khái niệm.....	104
4.3.2. Luật học Perceptron (learnp)	105
4.3.3. Huấn luyện mạng (train).....	107
4.4. CÁC HẠN CHẾ CỦA PERCEPTRON	111
4.5. SỬ DỤNG GIAO DIỆN ĐỒ HỌA ĐỂ KHẢO SÁT MẠNG NƠRON	112
4.5.1. Giới thiệu về GUI	112
4.5.2. Thiết lập mạng Perceptron (nntool).....	113
4.5.3. Huấn luyện mạng	115
4.5.4. Xuất kết quả Perceptron ra vùng làm việc.....	116
4.5.5. Xoá cửa sổ dữ liệu mạng (Network/Data Window)	117
4.5.6 Nhập từ dòng lệnh	117
4.5.7. Cắt biến vào file và nạp lại nó	118
Chương 5: MẠNG TUYẾN TÍNH.....	119
5.1. MỞ ĐẦU	119
5.1.1. Khái niệm.....	119
5.1.2. Mô hình nơron	119
5.2. CẤU TRÚC MẠNG	120
5.2.1. Cấu trúc.....	120
5.2.2. Khởi tạo nơron tuyến tính (Newlin)	121
5.3. THUẬT TOÁN CỤC TIỂU TRUNG BÌNH BÌNH PHƯƠNG SAI LỆCH.....	122
5.4. THIẾT KẾ HỆ TUYẾN TÍNH	123
5.5. MẠNG TUYẾN TÍNH CÓ TRỄ	123
5.5.1 Mất trễ.....	123
5.5.2. Thuật toán LMS (learnwh)	123
5.5.3. Sự phân loại tuyến tính (train)	125
5.6. MỘT SỐ HẠN CHẾ CỦA MẠNG TUYẾN TÍNH.....	126

Chương 6: HỆ MỜ - NƠON (FUZZY-NEURAL).....	128
6.1 SỰ KẾT HỢP GIỮA LOGIC MỜ VÀ MẠNG NƠON	128
6.1.1 Khái niệm.....	128
6.1.2. Kết hợp điều khiển mờ và mạng nơon	129
6.2. NƠON MỜ.....	133
6.3. HUẤN LUYỆN MẠNG NƠON-MỜ	135
6.4. SỬ DỤNG CÔNG CỤ ANFIS TRONG MATLAB ĐỂ THIẾT KẾ HỆ MỜ - NƠON (ANFIS and the ANFIS Editor GUI).....	139
6.4.1. Khái niệm.....	139
6.4.2. Mô hình học và suy diễn mờ thông qua ANFIS (Model Learning and Inference Through ANFIS).....	140
6.4.3. Xác nhận dữ liệu huấn luyện (Familiarity Breeds Validation)...	141
6.5. SỬ DỤNG BỘ SOẠN THẢO ANFIS GUI	143
6.5.1. Các chức năng của ANFIS GUI	143
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)	144
6.5.3. Một số ví dụ	145
6.6. SOẠN THẢO ANFIS TỪ DÒNG LỆNH	153
6.7. THÔNG TIN THÊM VỀ ANFIS VÀ BỘ SOẠN THẢO ANFIS EDITOR GUI.....	157
6.7.1. Dữ liệu huấn luyện (Training Data).....	158
6.7.2. Cấu trúc đầu vào FIS (Input FIS Structure).....	158
6.7.3. Các tùy chọn huấn luyện (Training Options)	159
6.7.4 Tùy chọn hiển thị Display Options	159
6.7.5. Phương pháp huấn luyện (Method)	160
6.7.6. Cấu trúc đầu ra FIS cho dữ liệu huấn luyện.....	160
6.7.7. Sai số huấn luyện.....	160
6.7.8. Bước tính (Step-size).....	160
6.7.9. Dữ liệu kiểm tra (Checking Data).....	161
6.7.10. Cấu trúc đầu ra FIS cho dữ liệu kiểm tra (Output FIS Structure for Checking Data)	162
6.7.11. Sai số kiểm tra (Checking Error).....	162
 TÀI LIỆU THAM KHẢO	 163

LỜI NÓI ĐẦU

Ngày nay, các hệ thống mờ và mạng nơ ron ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực của đời sống xã hội. Đặc biệt, trong lĩnh vực điều khiển và tự động hoá, hệ mờ và mạng nơ ron ngày càng chiếm ưu thế và đã mang lại nhiều lợi ích to lớn. Với ưu điểm cơ bản là có thể xử lý với độ chính xác cao những thông tin "không chính xác" hệ mờ và mạng nơ ron là cơ sở của hệ "điều khiển thông minh" và "trí tuệ nhân tạo".

Để đáp ứng nhu cầu tìm hiểu và ứng dụng logic mờ và mạng nơ ron của đông đảo bạn đọc, được sự cổ vũ và động viên của BGH trường Đại học Kỹ thuật Công nghiệp, chúng tôi đã mạnh dạn viết cuốn sách "Hệ mờ và nơ ron trong kỹ thuật điều khiển".

Cuốn sách được viết dựa trên các bài giảng về hệ thống điều khiển thông minh cho học viên cao học ngành Tự động hoá trường Đại học Kỹ thuật Công nghiệp. Cuốn sách không phân tích quá sâu những vấn đề lý thuyết phức tạp mà chỉ cung cấp cho bạn đọc những nội dung rất cơ bản về Hệ mờ, mạng nơ ron nhân tạo và hệ Mờ-nơ ron. Mục tiêu cao hơn là giúp bạn đọc biết cách khai thác những công cụ sẵn có của phần mềm MATLAB để phân tích, thiết kế các bộ điều khiển mờ, nơ ron nhằm điều khiển các đối tượng trong công nghiệp. Mỗi phần đều có các ví dụ cụ thể để hướng dẫn thiết kế.

Cuốn sách là tài liệu tham khảo cho học viên cao học, sinh viên ngành Điều khiển, các kỹ sư ngành Điện, Công nghệ thông tin và các nghiên cứu sinh quan tâm đến lĩnh vực điều khiển mờ và mạng nơ ron.

Trong quá trình biên soạn, không tránh khỏi còn nhiều sai sót. Chúng tôi mong nhận được sự đóng góp ý kiến các của đồng nghiệp và bạn đọc gần, xa.

Xin chân thành cảm ơn!

Thái Nguyên, ngày 01 tháng 12 năm 2006

Các tác giả

Chương 1

LÔGIC MỜ

1.1. TỔNG QUAN VỀ LÔGIC MỜ

1.1.1. Quá trình phát triển của lôgic mờ

Từ năm 1965 đã ra đời một lý thuyết mới đó là lý thuyết tập mờ (Fuzzy set theory) do giáo sư Lofti A. Zadeh ở trường đại học California - Mỹ đưa ra. Từ khi lý thuyết đó ra đời nó được phát triển mạnh mẽ qua các công trình khoa học của các nhà khoa học như: Năm 1972 GS Terano và Asai thiết lập ra cơ sở nghiên cứu hệ thống điều khiển mờ ở Nhật, năm 1980 hãng Smith Co. bắt đầu nghiên cứu điều khiển mờ cho lò hơi... Những năm đầu thập kỷ 90 cho đến nay hệ thống điều khiển mờ và mạng nơron (Fuzzy system and neural network) được các nhà khoa học, các kỹ sư và sinh viên trong mọi lĩnh vực khoa học kỹ thuật đặc biệt quan tâm và ứng dụng trong sản xuất và đời sống. Tập mờ và lôgic mờ đã dựa trên các thông tin "không đầy đủ, về đối tượng để điều khiển đầy đủ về đối tượng một cách chính xác.

Các công ty của Nhật bắt đầu dùng lôgic mờ vào kỹ thuật điều khiển từ năm 1980. Nhưng do các phần cứng chuẩn tính toán theo giải thuật lôgic mờ rất kém nên hầu hết các ứng dụng đều dùng các phần cứng chuyên về lôgic mờ. Một trong những ứng dụng dùng lôgic mờ đầu tiên tại đây là nhà máy xử lý nước của Fuji Electric vào năm 1983, hệ thống xe điện ngầm của Hitachi vào năm 1987.

1.1.2. Cơ sở toán học của lôgic mờ

Lôgic mờ và xác suất thông kê đều nói về sự không chắc chắn. Tuy nhiên mỗi lĩnh vực định nghĩa một khái niệm khác nhau về đối tượng.

Trong xác suất thông kê sự không chắc chắn liên quan đến sự xuất hiện của một sự kiện chắc chắn" nào đó.

Ví dụ: *Xác suất viên đạn trúng đích là 0,*

Bản thân của sự kiện "trúng đích" đã được định nghĩa rõ ràng, sự không

chắc chắn ở đây là có trúng đích hay không và được định lượng bởi mức độ xác suất (trong trường hợp này là 0,8). Loại phát biểu này có thể được xử lý và kết hợp với các phát biểu khác bằng phương pháp thống kê, như là xác suất có điều kiện chẳng hạn.

Sự không chắc chắn trong ngữ nghĩa, liên quan đến ngôn ngữ của con người, đó là sự không chính xác trong các từ ngữ mà con người dùng để ước lượng vấn đề và rút ra kết luận. Ví dụ như các từ mô tả nhiệt độ "nóng", "lạnh", "ấm" sẽ không có một giá trị chính xác nào để gán cho các từ này, các khái niệm này cũng khác nhau đối với những người khác nhau (là lạnh đối với người này nhưng không lạnh đối với người khác). Mặc dù các khái niệm không được định nghĩa chính xác nhưng con người vẫn có thể sử dụng chúng cho các ước lượng và quyết định phức tạp. Bằng sự trừu tượng và óc suy nghĩ, con người có thể giải quyết câu nói mang ngữ cảnh phức tạp mà rất khó có thể mô hình bởi toán học chính xác.

Sự không chắc chắn theo ngữ vựng: Như đã nói trên, mặc dù dùng những phát biểu không mang tính định lượng nhưng con người vẫn có thể thành công trong các ước lượng phức tạp. Trong nhiều trường hợp, con người dùng sự không chắc chắn này để tăng thêm độ linh hoạt. Như trong hầu hết xã hội, hệ thống luật pháp bao gồm một số luật, mỗi luật mô tả một tình huống. Ví dụ một luật quy định tội trộm xe phải bị tù 2 năm, một luật khác lại giảm nhẹ trách nhiệm. Và trong một phiên tòa, chánh án phải quyết định số ngày phạt tù của tên trộm dựa trên mức độ rượu trong người, trước đây có tiền án hay tiền sự không,... từ đó kết hợp lại đưa ra một quyết định công bằng.

1.1.3. Logic mờ là logic của con người

Trong thực tế, ta không định nghĩa một luật cho một trường hợp mà định nghĩa một số luật cho các trường hợp nhất định. Khi đó những luật này là những điểm rời rạc của một tập các trường hợp liên tục và con người xấp xỉ chúng. Gặp một tình huống cụ thể, con người sẽ kết hợp những luật mô tả các tình huống tương tự. Sự xấp xỉ này dựa trên sự linh hoạt của các từ ngữ cấu tạo nên luật, cũng như sự trừu tượng và sự suy nghĩ dựa trên sự linh hoạt trong logic của con người.

Để thực thi logic của con người trong kỹ thuật cần phải có một mô hình toán học của nó. Từ đó logic mờ ra đời như một mô hình toán học cho phép

mô tả các quá trình quyết định và ước lượng của con người theo dạng giải thuật. Dĩ nhiên cũng có giới hạn, đó là logic mờ không thể bắt chước trí tưởng tượng và khả năng sáng tạo của con người. Tuy nhiên, logic mờ cho phép ta rút ra kết luận khi gặp những tình huống không có mô tả trong luật nhưng có sự tương đương. Vì vậy, nếu ta mô tả những mong muốn của mình đối với hệ thống trong những trường hợp cụ thể vào luật thì logic mờ sẽ tạo ra giải pháp dựa trên tất cả những mong muốn đó.

1.2. KHÁI NIỆM VỀ TẬP MỜ

1.2.1. Tập kinh điển

Khái niệm tập hợp được hình thành trên nền tảng logic và được định nghĩa như là sự sắp xếp chung các đối tượng có cùng tính chất, được gọi là phần tử của tập hợp đó.

Cho một tập hợp A, một phần tử x thuộc A được ký hiệu: $x \in A$. Thông thường ta dùng hai cách để biểu diễn tập hợp kinh điển, đó là:

Liệt kê các phần tử của tập hợp, ví dụ tập $A_1 = \{\text{xe đạp, xe máy, xe ca, xe tải}\}$;

- Biểu diễn tập hợp thông qua tính chất tổng quát của các phần tử, ví dụ: tập các số thực (R), Tập các số tự nhiên (N).

Để biểu diễn một tập hợp A trên tập nền X, ta dùng hàm thuộc $\mu_A(x)$, với:

$$\mu_A(x) = \begin{cases} 1 & \text{khi } x \in A \\ 0 & \text{khi } x \notin A \end{cases} \quad \begin{array}{l} \mu_A(x) \text{ chỉ nhận một trong 2 giá trị "1"} \\ \text{hoặc "0"} \end{array}$$

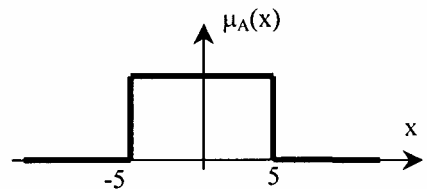
ký hiệu = $\{x \in X | x \text{ thoả mãn một số tính chất nào đó}\}$. Ta nói: Tập A được định nghĩa trên tập nền X.

Hình 1.1 mô tả hàm phụ thuộc $\mu_A(x)$ của tập các số thực từ -5 đến 5.

$$A = \{x \in \mathbb{R} | 5 \leq x \leq 5\}$$

1.2.2. Định nghĩa tập mờ

Trong khái niệm tập hợp kinh điển hàm phụ thuộc $\mu_A(x)$ của tập A, chỉ có một trong hai giá trị là "1" nếu $x \in A$ hoặc "0" nếu $x \notin A$.

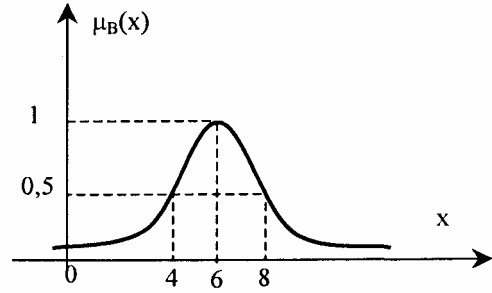


Hình 1.1. Hàm phụ thuộc $\mu_A(x)$ của tập kinh điển A

Cách biểu diễn hàm phụ thuộc như trên sẽ không phù hợp với những tập được mô tả "mờ" như tập B gồm các số thực gần bằng 5:

$$B = \{x \in \mathbb{R} \mid x \approx 5\}.$$

Khi đó ta không thể khẳng định chắc chắn số 4 có thuộc B hay không? mà chỉ có thể nói nó thuộc B bao nhiêu phần trăm. Để trả lời được câu hỏi này, ta phải coi hàm phụ thuộc $\mu_B(x)$ có giá trị trong khoảng từ 0 đến 1 tức là: $0 \leq \mu_B(x) \leq 1$.



Hình 1.2. Hàm liên thuộc $\mu_B(x)$ của tập "mờ" B

Từ phân tích trên ta có định nghĩa: *Tập mờ B xác định trên tập kinh điển M là một tập mà một phần tử của nó được biểu diễn bởi một cặp giá trị $(x, \mu_B(x))$. Trong đó $x \in M$ và $\mu_B(x)$ là ánh xạ.*

Ánh xạ $\mu_B(x)$ được gọi là hàm liên thuộc của tập mờ B. Tập kinh điển M được gọi là cơ sở của tập mờ B.

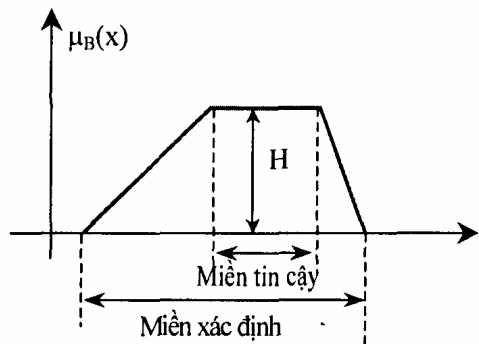
1.2.3. Các thông số đặc trưng cho tập mờ

Các thông số đặc trưng cho tập mờ là độ cao, miền xác định và miền tin cậy (hình 1.3)

+ **Độ cao** của một tập mờ B (Định nghĩa trên cơ sở M) là giá trị lớn nhất trong các giá trị của hàm liên thuộc:

$$H = \sup_{x \in M} \mu_B(x)$$

Một tập mờ có ít nhất một phần tử có độ phụ thuộc bằng 1 được gọi là tập mờ chính tắc ($H = 1$). Ngược lại, một tập mờ B với $H < 1$ gọi là tập mờ không chính tắc.



Hình 1.3. Độ cao, miền xác định, miền tin cậy của tập mờ

+ **Miền xác định** của tập mờ B (Định nghĩa trên cơ sở M) được ký hiệu bởi S là tập con của M có giá trị hàm liên thuộc khác không:

$$S = \{x \in M \mid \mu_B(x) > 0\}.$$

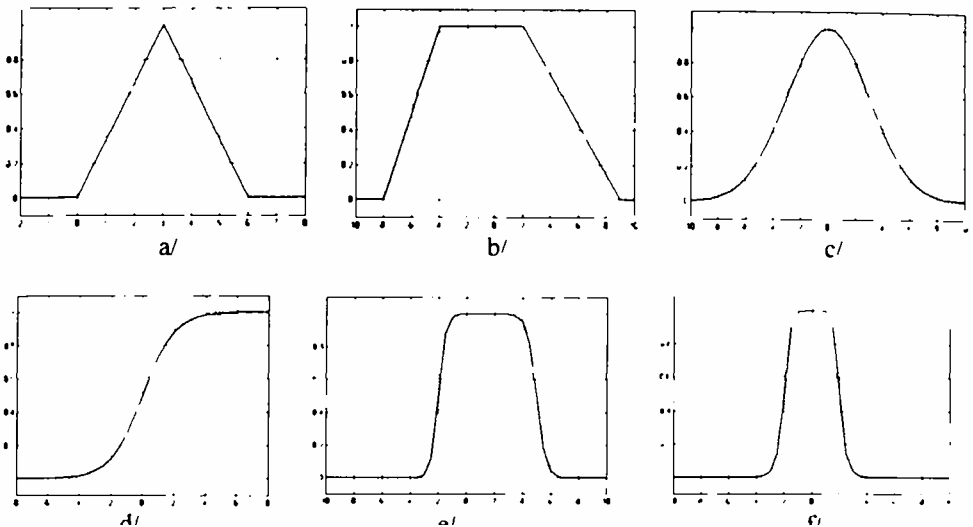
+ **Miền tin cậy** của tập mờ B (định nghĩa trên cơ sở M) được ký hiệu bởi T, là tập con của M có giá trị hàm liên thuộc bằng 1:

$$T = \{x \in M \mid \mu_B(x) = 1\}.$$

1.2.4. Các dạng hàm liên thuộc của tập mờ

Có rất nhiều cách khác nhau để biểu diễn hàm liên thuộc của tập mờ. Dưới đây là một số dạng hàm liên thuộc thông dụng:

- + Hàm liên thuộc hình tam giác (hình 1.4a);
- + Hàm liên thuộc hình thang (hình 1.4b);
- + Hàm liên thuộc dạng Gauss (hình 1.4c);
- + Hàm liên thuộc dạng Sign (hình 1.4d);
- + Hàm Sigmoidal (hình 1.4e);
- + Hàm hình chuông (hình 1.4f).



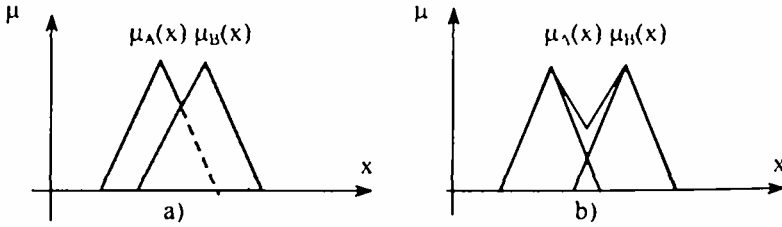
Hình 1.4. Các dạng hàm liên thuộc của tập mờ

1.3. CÁC PHÉP TOÁN TRÊN TẬP MỜ

Trên tập mờ có 3 phép toán cơ bản là phép hợp, phép giao, và phép bù.

1.3.1. Phép hợp hai tập mờ

a/ Hợp của hai tập mờ có cùng cơ sở



Hình 1.5. Hợp của hai tập mờ có cùng cơ sở
theo quy tắc Max (a), theo Lukasiewicz (b)

Hợp của hai tập mờ A và B có cùng cơ sở M là một tập mờ cùng xác định trên cơ sở M với hàm liên thuộc được xác định theo một trong các công thức sau:

1. $\mu_{A \cup B}(x) = \text{Max} \{ \mu_A(x), \mu_B(x) \};$
2. $\mu_{A \cup B}(x) = \min \{ 1, \mu_A(x) + \mu_B(x) \}$ (phép hợp Lukasiewicz);
3. $\mu_{A \cup B}(x) = \begin{cases} \max \{ \mu_A(x), \mu_B(x) \} & \text{khi } \min \{ \mu_A(x), \mu_B(x) \} = 0 \\ 1 & \text{khi } \min \{ \mu_A(x), \mu_B(x) \} \neq 0 \end{cases}$
4. $\mu_{A \cup B}(x) = \frac{\mu_A(x) + \mu_B(x)}{1 + \mu_A(x) + \mu_B(x)}$ (Tổng Einstein)
5. $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$ (tổng trực tiếp).

Chú ý: Có nhiều công thức khác nhau được dùng để tính hàm liên thuộc $\mu_{A \cup B}(x)$ của hai tập mờ. Song trong kỹ thuật điều khiển mờ ta chủ yếu dùng 2 công thức hợp, đó là lấy Max và phép hợp Lukasiewicz.

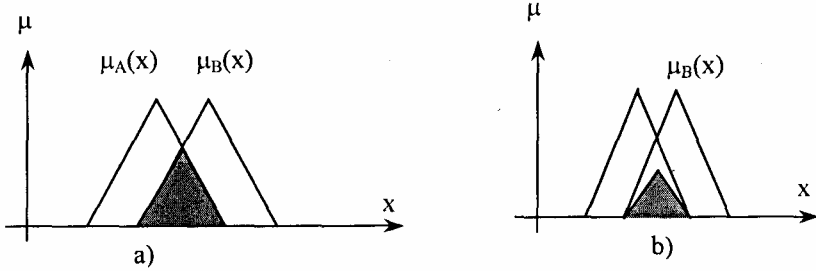
b/ Hợp hai tập mờ khác cơ sở

Để thực hiện phép hợp 2 tập mờ khác cơ sở, về nguyên tắc ta phải đưa chúng về cùng một cơ sở. Xét tập mờ A với hàm liên thuộc $\mu_A(x)$ được định nghĩa trên cơ sở M và B với hàm liên thuộc $\mu_B(x)$ được định nghĩa trên cơ sở N, hợp của 2 tập mờ A và B là một tập mờ xác định trên cơ sở $M \times N$ với hàm liên thuộc: $\mu_{A \cup B}(x, y) = \text{Max} \{ \mu_A(x, y), \mu_B(x, y) \}$

Với $\mu_A(x, y) = \mu_A(x)$ với mọi $y \in N$ và $\mu_B(x, y) = \mu_B(y)$ với mọi $x \in M$.

1.3.2. Phép giao của hai tập mờ

a/ Giao hai tập mờ cùng cơ sở



Hình 1.6. Giao của hai tập mờ có cùng cơ sở theo quy tắc Min (a) và theo tích đại số (b)

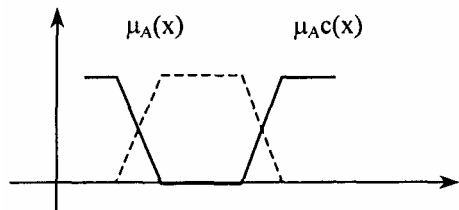
Giao của hai tập mờ A và B có cùng cơ sở M là một tập mờ cũng xác định trên cơ sở M với hàm liên thuộc $\mu_{A \cap B}(x)$ được tính:

1. $\mu_{A \cap B}(x) = \text{Min} \{ \mu_A(x), \mu_B(x) \};$
2. $\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x)$ (tích đại số);
3. $\mu_{A \cap B}(x) = \begin{cases} \min\{\mu_A(x), \mu_B(x)\} & \text{khi } \min\{\mu_A(x), \mu_B(x)\} = 1; \\ 0 & \text{khi } \max\{\mu_A(x), \mu_B(x)\} \neq 1 \end{cases};$
4. $\mu_{A \cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$ (Phép giao Lukasiewicz);
5. $\mu_{A \cap B}(x) = \frac{\mu_A(x) \cdot \mu_B(x)}{2 - (\mu_A(x) + \mu_B(x)) - \mu_A(x)\mu_B(x)}$ (tích Einstein).

cũng giống như trong phép hợp, trong kỹ thuật điều khiển chủ yếu ta sử dụng công thức 1 và công thức 2 để thực hiện phép giao 2 tập mờ.

b/ Giao hai tập mờ khác cơ sở

Để thực hiện phép giao 2 tập mờ khác cơ sở, ta cần phải đưa về cùng cơ sở. Khi đó, giao của tập mờ A có hàm liên thuộc $\mu_A(x)$ định nghĩa trên cơ sở M với tập mờ B có hàm liên thuộc $\mu_B(x)$ định nghĩa trên cơ sở N là một tập mờ xác định trên cơ sở M x N có hàm liên thuộc được tính:



Hình 1.7. Bù của tập mờ

$$\mu_{A \cap B}(x, y) = \text{MIN} \{ \mu_A(x, y), \mu_B(x, y) \}$$

Trong đó: $\mu_A(x, y) = \mu_A(x)$ với mọi $y \in N$ và $\mu_B(x, y) = \mu_B(x)$ với mọi $x \in M$.

1.3.3. Phép bù của một tập mờ

Bù của tập mờ A có cơ sở M và hàm liên thuộc $\mu_A(x)$ là một tập mờ A_C xác định trên cùng cơ sở M với hàm liên thuộc: $\mu_{A_C}(x) = 1 - \mu_A(x)$

1.4. BIẾN NGÔN NGỮ VÀ GIÁ TRỊ CỦA BIẾN NGÔN NGỮ

Thực tế hàng ngày chúng ta luôn dùng các từ ngữ, lời nói để mô tả các biến. Ví dụ khi ta nói: "Điện áp cao quá", "xe chạy nhanh quá",... như vậy biến "Điện áp", biến "Tốc độ xe",... nhận các giá trị từ "nhanh" đến "chậm", từ "cao" đến "thấp". Ở dạng tường minh, các biến này nhận các giá trị cụ thể (rõ) như điện áp bằng 200 V, 250 V...; tốc độ xe bằng 60 km/h, 90 km/h... Khi các biến nhận các giá trị không rõ ràng như "cao", "rất cao" "nhanh", "hơi nhanh"... ta không thể dùng các giá trị rõ để mô tả được mà phải sử dụng một số khái niệm mới để mô tả gọi là biến ngôn ngữ.

Một biến có thể gán bởi các từ trong ngôn ngữ tự nhiên làm giá trị của nó gọi là biến ngôn ngữ.

Một biến ngôn ngữ thường bao gồm 4 thông số: X, T, U, M . Với:

- + X : Tên của biến ngôn ngữ;
- + T : Tập của các giá trị ngôn ngữ;
- + U : Không gian nền mà trên đó biến ngôn ngữ X nhận các giá trị rõ;
- + M : Chỉ ra sự phân bố của T trên U .

Ví dụ: Biến ngôn ngữ "Tốc độ xe" có tập các giá trị ngôn ngữ là rất chậm, chậm, trung bình, nhanh, rất nhanh, không gian nền của biến là tập các số thực dương. Vậy biến tốc độ xe có 2 miền giá trị khác nhau:

- Miền các giá trị ngôn ngữ $N = [\text{rất chậm, chậm, trung bình, nhanh, rất nhanh}]$.

- Miền các giá trị vật lý $V = \{x \in \mathbb{R} (x \geq 0)\}$.

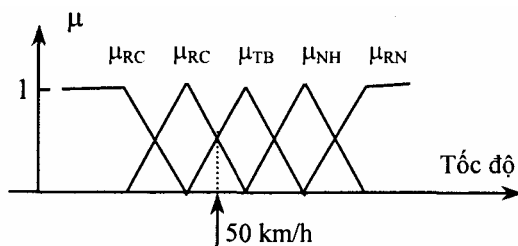
Mỗi giá trị ngôn ngữ (mỗi phần tử của N) có tập nền là miền giá trị vật lý V . Từ một giá trị vật lý của biến ngôn ngữ ta có được một vectơ $\underline{\mu}$ gồm các độ phụ thuộc của x :

$$X \rightarrow \underline{\mu}^T = [\mu_{\text{rất chậm}} \ \mu_{\text{chậm}} \ \mu_{\text{trung bình}} \ \mu_{\text{nhanh}} \ \mu_{\text{rất nhanh}}]$$

ánh xạ trên được gọi là quá trình fuzzy hoá giá trị rõ x.

Ví dụ: ứng với tốc độ 50 km/h ta có

$$\text{véc tơ: } \mu(50) = \begin{cases} 0 \\ 0,5 \\ 0,5 \\ 0 \\ 0 \end{cases}$$



Hình 2.8. Mờ hoá biến “Tốc độ”

1.5. LUẬT HỢP THÀNH MỜ

1.5.1. Mệnh đề hợp thành

Xét hai biến ngôn ngữ χ và γ ; Biến χ nhận giá trị (mờ) A có hàm liên thuộc $\mu_A(x)$ và γ nhận giá trị (mờ) B có hàm liên thuộc $\mu_B(x)$ thì hai biểu thức:

$\chi = A; \gamma = B$ được gọi là hai mệnh đề.

Luật Điều khiển: nếu $\chi = A$ thì $\gamma = B$ được gọi là mệnh đề hợp thành. Trong đó $\chi = A$ gọi là mệnh đề điều kiện và $\gamma = B$ gọi là mệnh đề kết luận. Một mệnh đề hợp

thành có thể có nhiều mệnh đề điều kiện và nhiều mệnh đề kết luận, các mệnh đề liên kết với nhau bằng toán tử "và". Dựa vào số mệnh đề điều kiện và số mệnh đề kết luận trong một mệnh đề hợp thành mà ta phân chúng thành các cấu trúc khác nhau:

- Cấu trúc SISO (một vào, một ra): Chỉ có một mệnh đề điều kiện và một mệnh đề kết luận. Ví dụ: nếu $\chi = A$ thì $\gamma = B$.

- Cấu trúc MISO (Nhiều vào, một ra): Có từ 2 mệnh đề điều kiện trở lên và một mệnh đề kết luận. Ví dụ: nếu $\chi_1 = A_1$ và $\chi_2 = A_2$ thì $\gamma = B$.

- Cấu trúc MIMO (Nhiều vào, nhiều ra): Có ít nhất 2 mệnh đề điều kiện và 2 mệnh đề kết luận. Ví dụ: nếu $\chi_1 = A_1$ và $\chi_2 = A_2$ thì $\gamma_1 = B_1$ và $\gamma_2 = B_2$

1.5.2. Mô tả mệnh đề hợp thành

Xét mệnh đề hợp thành: nếu $\chi = \mathbf{A}$ thì $\gamma - \mathbf{B}$; Từ một giá trị x_0 có độ phụ thuộc $\mu_A(x_0)$ đối với tập mờ A của mệnh đề điều kiện, ta xác định được độ thỏa mãn mệnh đề kết luận. Biểu diễn độ thỏa mãn của mệnh đề kết luận như một tập mờ B' cùng cơ sở với B thì mệnh đề hợp thành chính là ánh xạ: $\mu_A(x_0) \rightarrow \mu_B(y)$.

Ánh xạ này chỉ ra rằng mệnh đề hợp thành là một tập mà mỗi phần tử là một giá trị $(\mu_A(x_0), \mu_{B'}(y))$ tức là mỗi phần tử là một tập mờ. Mô tả mệnh đề hợp thành tức là mô tả ánh xạ trên. Ánh xạ $(\mu_A(x_0), \mu_{B'}(y))$ được gọi là hàm liên thuộc của luật hợp thành. Để xây dựng $\mu_{B'}(y)$ đã có rất nhiều ý kiến khác nhau. Trong kỹ thuật điều khiển ta thường sử dụng nguyên tắc của Mamdani "**Độ phụ thuộc của kết luận không được lớn hơn độ phụ thuộc của điều kiện**". Từ nguyên tắc đó ta có hai công thức xác định hàm liên thuộc cho mệnh đề hợp thành $A \Rightarrow B$:

1. công thức MIN: $\mu_{A \Rightarrow B}(x, y) = \text{MIN}\{\mu_A(x), \mu_B(y)\}$
2. công thức PROD: $\mu_{A \Rightarrow B}(x, y) = \mu_A(x)\mu_B(y)$

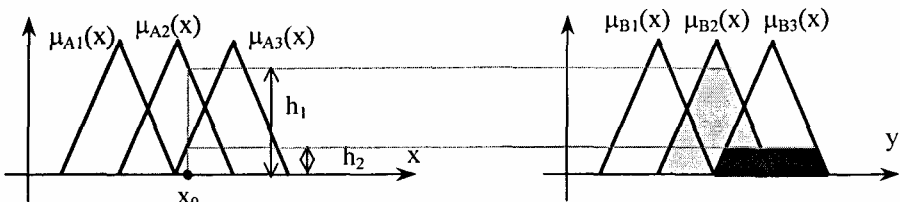
1.5.3. Luật hợp thành mờ

Luật hợp thành là tên chung gọi mô hình R biểu diễn (một hay nhiều) hàm liên thuộc $\mu_{A \Rightarrow B}(x, y)$ cho (một hay nhiều) mệnh đề hợp thành $A \Rightarrow B$.

Một luật hợp thành chỉ có 1 mệnh đề hợp thành gọi là luật hợp thành đơn, có từ 2 mệnh đề hợp thành trở lên gọi là luật hợp thành phức.

Xét luật hợp thành R gồm 3 mệnh đề hợp thành:

- R₁: Nếu $x = A_1$ Thì $y = B_1$ hoặc
- R₂: Nếu $x = A_2$ Thì $y = B_2$ hoặc
- R₃: Nếu $x = A_3$ Thì $y = B_3$ hoặc



Hình 1.9. Mô tả hàm liên thuộc của luật hợp thành

Với mỗi giá trị rõ x_0 của biến ngôn ngữ đầu vào, ta có 3 tập mờ ứng với 3 mệnh đề hợp thành R_1, R_2, R_3 của luật hợp thành R . Gọi hàm liên thuộc của các tập mờ đầu ra là: $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ thì giá trị của luật hợp thành R ứng với x_0 là tập mờ B' thu được qua phép hợp 3 tập mờ: $B' = B'_1 \cup B'_2 \cup B'_3$.

Tùy theo cách thu nhận các hàm liên thuộc $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ và phương pháp thực hiện phép hợp để nhận tập mờ B' mà ta có tên gọi các luật hợp thành khác nhau:

- Luật hợp thành MAX-MIN nếu $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ thu được qua phép lấy Min còn phép hợp thực hiện theo luật Max;

- Luật hợp thành MAX-PROD nếu $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ thu được qua phép PROD còn phép hợp thực hiện theo luật Max;

- Luật hợp thành SUM-MIN nếu $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ thu được qua phép lấy Min còn phép hợp thực hiện theo luật SUM;

- Luật hợp thành SUM - PROD nếu $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ thu được qua phép lấy PROD còn phép hợp thực hiện theo Lukasiewicz.

Vậy, để xác định hàm liên thuộc $\mu_{B'}(y)$ của giá trị đầu ra B' của luật hợp thành có n mệnh đề hợp thành R_1, R_2, \dots ta thực hiện theo các bước sau:

+ Xác định độ thỏa mãn h_j .

+ Tính $\mu_{B'_1}(y); \mu_{B'_2}(y); \mu_{B'_3}(y)$ theo quy tắc min hoặc Prod

$$\mu_{B'_j}(y) = \text{Min}\{\mu_A(x_0), \mu_{B_j}(y)\} = \text{Min}\{h_j, \mu_{B_j}(y)\}$$

$$\text{hoặc } \mu_{B'_j}(y) = \mu_A(x_0) \cdot \mu_{B_j}(y) = h_j \cdot \mu_{B_j}(y).$$

+ xác định $\mu_{B'}(y)$ bằng cách thực hiện phép hợp các $\mu_{B'_j}(y)$

1.5.4. Các cấu trúc cơ bản của luật hợp thành

Ta sẽ khảo sát hai cấu trúc cơ bản của luật hợp thành, đó là cấu trúc SISO và cấu trúc MISO.

+ **Cấu trúc SISO** là cấu trúc trong đó luật hợp thành có các mệnh đề điều kiện và mệnh đề kết luận là các mệnh đề đơn.

Ví dụ: R_1 : nếu $\chi = A_1$ thì $\gamma = B_1$ hoặc

R_2 : nếu $\chi = A_2$ thì $\gamma = B_2$.

+ **Cấu trúc MISO** là cấu trúc trong đó luật hợp thành có các mệnh đề điều kiện là mệnh đề phức và mệnh đề kết luận là mệnh đề đơn.

Ví dụ: R_1 : nếu $\chi_1 = A_1$ và $\chi_2 = B_1$ thì $\gamma = C_1$ hoặc

R_2 : nếu $\chi_1 = A_2$ và $\chi_2 = B_2$ thì $\gamma = C_2$.

1.5.5. Luật hợp thành đơn có cấu trúc SISO

a) Luật hợp thành MIN

Luật hợp thành MIN là tên gọi mô hình (ma trận) R của mệnh đề hợp thành $A \Rightarrow B$ khi hàm liên thuộc $\mu_{A \Rightarrow B}(x, y)$ của nó được xây dựng theo quy tắc MIN.

Xét luật hợp thành chỉ có 1 mệnh đề: **Nếu $\chi = A$ thì $\gamma = B$**

Để xây dựng R, trước tiên hai hàm liên thuộc $\mu_A(x)$ và $\mu_B(y)$ được rời rạc hoá với tần số rời rạc đủ nhỏ để không bị mất thông tin.

Ví dụ: $\mu_A(x)$, $\mu_B(y)$ được rời rạc hoá tại các điểm:

$x \in \{10, 20, 30, 40, 50\}$

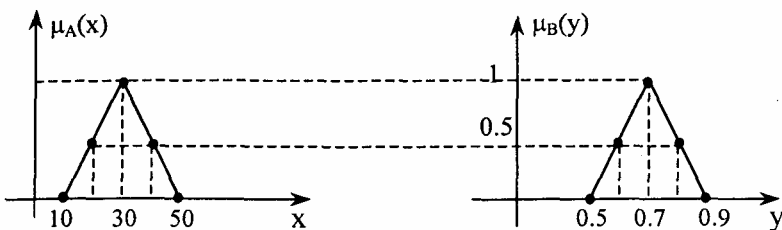
$y \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$.

Với các điểm rời rạc này thì theo

$$\mu_{A \Rightarrow B}(20; 0.7) = \mu_R(20; 0.7) = \text{MIN}\{\mu_A(20), \mu_B(0.7)\} = \text{MIN}\{0.5; 1\} = 0.5$$

$$\mu_{A \Rightarrow B}(30; 0.7) = \mu_R(30; 0.7) = \text{MIN}\{\mu_A(30), \mu_B(0.7)\} = \text{MIN}\{1; 1\} = 1$$

.....



Hình 1.10. Rời rạc hoá các hàm liên thuộc

Nhóm tất cả các giá trị $\mu_{A \Rightarrow B}(x, y) = \mu_R(x, y)$ gồm $5 \times 5 = 25$ giá trị, thành ma trận R (được gọi là ma trận hợp thành MIN) gồm 5 hàng 5 cột.

		y				
R		0.5	0.6	0.7	0.8	0.9
X	10	0	0	0	0	0
	20	0	0.5	0.5	0.5	0
	30	0	0.5	1	0.5	0
	40	0	0.5	0.5	0.5	0
	50	0	0	0	0	0

Khi tín hiệu đầu vào là một giá trị rõ $x_0 = 20$, tín hiệu đầu ra B' có hàm liên thuộc:

$$\mu_{B'}(y) = \mu_R(20, y) = \{0; 0.5; 0.5; 0.5; 0\}.$$

Để thuận tiện cho việc xác định hàm liên thuộc của tín hiệu ra dưới dạng nhân ma trận, ta định nghĩa một ma trận $T = \{a_1 a_2 \dots\}$ ma trận này chỉ có một phần tử bằng 1 còn các phần tử khác đều bằng 0. Ví dụ với tập 5 phần tử cho tín hiệu đầu vào xử $\{10; 20; 30; 40; 50\}$ thì ứng với $x_0 = 20$ (phần tử thứ hai) ta có:

$$\underline{a} = (0 \ 1 \ 0 \ 0 \ 0)$$

Và khi đó

$$\mu_{B'}(y) = \mu_R(x_0, y) = \underline{a}^T \cdot R = \{0 \ 0.5 \ 0.5 \ 0.5 \ 0\}.$$

Tổng quát cho một giá trị rõ x_0 bất kỳ

$$x_0 \in X = \{10 \ 20 \ 30 \ 40 \ 50\}$$

tại đầu vào vectơ chuyển vị có dạng:

$$\underline{a}^T = (a_1, a_2, a_3, a_4, a_5)$$

trong đó chỉ có một phần tử a_i duy nhất có chỉ số i là chỉ số của x_0 trong X có giá trị bằng 1, các phần tử còn lại đều bằng 0. Hàm liên thuộc $\mu_{B'}(y)$ dưới dạng rời rạc được xác định:

$$\mu_{B'}(y) = \underline{a}^T \cdot R = (a_1, a_2, a_3, a_4, a_5) \begin{pmatrix} r_{11} \dots\dots\dots r_{15} \\ \dots\dots\dots \\ r_{51} \dots\dots\dots r_{55} \end{pmatrix} =$$

$$= (l_1, l_2, l_3, l_4, l_5) \text{ với } l_k = \sum_{i=1}^5 a_i \cdot r_{ik} . \quad (1.1)$$

Chú ý: Trong biểu thức (1.1) để tính $\mu_{B'}(y)$ ta cần cài đặt thuật toán nhân ma trận của đại số tuyến tính, do đó tốc độ xử lý chậm. Để khắc phục nhược điểm này, phép nhân ma trận (1.1) được thay bởi luật MAX-MIN của Zadeh với MAX (phép lấy cực đại) thay vào vị trí phép cộng và MIN (phép lấy cực tiểu) thay vào vị trí phép nhân. Khi đó:

$$l_k = \max_{1 \leq i \leq 5} \min \{a_i r_{ik}\}$$

Kết quả hai phép tính (1.1) và (1.2) với đầu vào là một giá trị rõ hoàn toàn giống nhau. Cũng từ lý do trên mà luật hợp thành MIN còn có tên gọi là luật hợp thành MAX-MIN.

b/ Luật hợp thành PROD

Tương tự như đã làm với luật hợp thành MIN, ma trận R của luật hợp thành PROD được xây dựng gồm các hàng là m giá trị rời rạc của đầu ra $\mu_{B'}(y_1), \mu_{B'}(y_2), \mu_{B'}(y_m)$ cho n giá trị rõ đầu vào x_1, x_2, \dots, x_n . Như vậy ma trận R sẽ có n hàng và m cột. Xét ví dụ trên cho 5 giá trị đầu vào:

$$\{x_1, x_2, x_3, x_4, x_5\} = \{10, 20, 30, 40, 50\}$$

thì với từng giá trị x_i , 5 giá trị của hàm liên thuộc đầu ra tương ứng $\mu_{B'}(0.5), \mu_{B'}(0.6), \mu_{B'}(0.7), \mu_{B'}(0.8), \mu_{B'}(0.9)$ được liệt kê trong ma trận R được gọi là ma trận hợp thành PROD.

Từ ma trận R trên, hàm liên thuộc $\mu_{B'}(y)$ của giá trị đầu ra khi đầu vào là giá trị rõ x_4 cũng được xác định bằng công thức:

$$\underline{a}^T = (0, 0, 0, 1, 0)$$

$$\mu_{B'}(y) = \mu_{R}(x_4, y) = \underline{a}^T \cdot R = \{0, 0.25, 0.5, 0.25, 0\}.$$

Để rút ngắn thời gian tính và cũng để mở rộng công thức trên cho trường hợp đầu vào là giá trị mờ, phép nhân ma trận T.R cũng được thay bằng luật MAX-PROD của Zadeh như đã làm cho luật hợp thành MIN. Trong đó phép nhân được thực hiện bình thường còn phép lấy cực đại thay vào vị trí của phép cộng.

$$\mu_{B'}(y) = \underline{a}^T \cdot \mathbf{R} = (a_1, a_2, a_3, a_4, a_5) \begin{pmatrix} r_{11} \dots \dots \dots r_{15} \\ \dots \dots \dots \dots \dots \dots \\ r_{51} \dots \dots \dots r_{55} \end{pmatrix} =$$

$$= (l_1, l_2, l_3, l_4, l_5)$$

với $l_k = \sum_{i=1}^5 a_i r_{ik}$; $l_k = \max_{1 \leq i \leq 5} \text{Pr od}\{a_i r_{ki}\}$.

	R	0.5	0.6	0.7	0.8	0.9
i = 1	10	0	0	0	0	0
i = 2	20	0	0.25	0.5	0.25	0
i = 3	30	0	0.5	1	0.5	0
i = 4	40	0	0.25	0.5	0.25	0
i = 5	50	0	0	0	0	0

c) Thuật toán xây dựng R

Từ các phân tích trên, ta rút ra thuật toán xây dựng R cho luật hợp thành đơn có cấu trúc SISO (**Nếu $\chi = \mathbf{A}$ Thì $\gamma = \mathbf{B}$**) như sau:

1- Rời rạc hoá $\mu_A(x)$ tại n điểm x_1, x_2, \dots, x_n tại m điểm y_1, y_2, \dots, y_m (n có thể khác m)

2- Xây dựng ma trận R gồm n hàng và m cột:

$$\mathbf{R} = \begin{pmatrix} \mu_R(x_1, y_1) \dots \dots \dots \mu_R(x_1, y_m) \\ \dots \dots \dots \dots \dots \dots \\ \mu_R(x_n, y_1) \dots \dots \dots \mu_R(x_n, y_m) \end{pmatrix} = \begin{pmatrix} r_{11} \dots \dots \dots r_{1m} \\ \dots \dots \dots \dots \dots \dots \\ r_{n1} \dots \dots \dots r_{nm} \end{pmatrix}$$

3- Xác định hàm liên thuộc $\mu_{B'}(y)$ của đầu ra ứng với giá trị rõ đầu vào x_k theo biểu thức:

$$\mu_B(y) = \underline{a}^T \cdot R = (a_1, a_2, \dots, a_n) \begin{pmatrix} r_{11} \dots r_{1m} \\ \dots \dots \dots \\ r_{n1} \dots r_{nm} \end{pmatrix} = (l_1, l_2, \dots, l_m)$$

với $l_k = \sum_{i=1}^n a_i r_{ik}$.

Với: $\underline{a}^T = (0, 0, \dots, 0, 1, 0, \dots, 0)$
 \uparrow vị trí thứ k

trong đó: $l_k = \max_{1 \leq i \leq n} \min \{a_i r_{ki}\}$, $k = 1, 2, \dots, m$ nếu sử dụng công thức MAX-MIN và $l_k = \max_{1 \leq i \leq n} \prod \{a_i r_{ki}\}$, $k = 1, 2, \dots, m$ nếu sử dụng công thức MAX-PROD.

4- Xác định $\mu_B(y)$ theo công thức: $\mu_B(y) = (l_1, l_2, \dots, l_m)$.

Chú ý:

☞ Trong trường hợp đầu vào là giá trị mờ A' với hàm liên thuộc $\mu_{A'}(y)$ thì hàm liên thuộc $\mu_{B'}(y)$ của giá trị đầu ra B' : $\mu_{B'}(y) = (l_1, l_2, \dots, l_m)$ cũng được tính theo công thức (2.4) và

$$l_k = \max_{1 \leq i \leq n} \min \{a_i r_{ki}\}, k = 1, 2, \dots, m$$

trong đó \underline{a} là véc tơ gồm các giá trị rời rạc của hàm liên thuộc $\mu_{A'}(x)$ của A' tại các điểm:

$$x \in X = \{x_1, x_2, \dots, x_n\} \text{ tức là } \underline{a}^T = (\mu_{A'}(x_1), \mu_{A'}(x_2), \dots, \mu_{A'}(x_n)).$$

☞ Giả thiết có n điểm rời rạc x_1, x_2, \dots, x_n của cơ sở A và m điểm rời rạc y_1, y_2, \dots, y_m của cơ sở B ta có hai véc tơ:

$$\underline{\mu}_A^T = \{\mu_A(x_1), \mu_A(x_2), \dots, \mu_A(x_n)\} \text{ và } \underline{\mu}_B^T = \{\mu_B(y_1), \mu_B(y_2), \dots, \mu_B(y_m)\}$$

theo Zadeh ta có thể xác định ngay được R thông qua tích dyadic, tức là tích của một véc tơ với một véc tơ chuyển vị:

$$R = \underline{\mu}_A \underline{\mu}_B^T$$

Trong đó nếu quy tắc áp dụng là MAX - MIN thì phép nhân phải được thay bằng phép tính lấy cực tiểu (min), với quy tắc MAX - PROD thì thực hiện phép nhân như bình thường.

Ví dụ: Luật điều khiển: **Nếu $\chi = A$ Thì $\gamma = B$.** Hãy xây dựng ma trận R của luật $\mu_{A \Rightarrow B}(x, y)$.

Với 5 điểm rời rạc của X (cơ sở của A) ta có:

$\{x_1, x_2, x_3, x_4, x_5\} = \{10, 20, 30, 40, 50\}$ tương ứng $\underline{\mu}_A^T = \{0; 0.5; 1; 0.5; 0\}$ Và Với 5 điểm rời rạc của Y (cơ sở của B)

$\{y_1, y_2, y_3, y_4, y_5\} = \{0.5, 0.6, 0.7, 0.8, 0.9\}$ Tương ứng $\underline{\mu}_B^T = \{0; 0.5; 1; 0.5; 0\}$.

Nếu sử dụng quy tắc MAX-MIN (phép nhân được thay bằng min) ma trận hợp thành R sẽ như sau:

$$R = \begin{pmatrix} 0 \\ 0.5 \\ 1 \\ 0.5 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 0.5 & 1 & 0.5 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Nếu sử dụng quy tắc MAX-PROD (phép nhân thực hiện bình thường) ta có ma trận hợp thành R là:

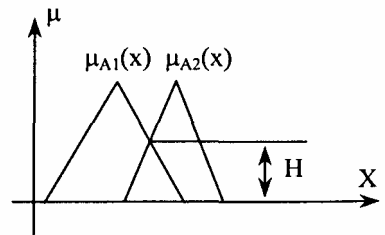
$$R = \begin{pmatrix} 0 \\ 0.5 \\ 1 \\ 0.5 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 0.5 & 1 & 0.5 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

1.5.6. Luật hợp thành đơn có cấu trúc MISO

Xét một mệnh đề hợp thành với d mệnh đề điều kiện:

Nếu $\chi_1 = A_1$ và $\chi_2 = A_2$ và ... và $\chi_d = A_d$ thì $\gamma = B$

Bao gồm d biến ngôn ngữ đầu vào $\chi_1, \chi_2, \dots, \chi_d$ và một biến đầu ra γ .



Hình 1.12. Xác định độ thỏa mãn H

Việc mô hình hoá mệnh đề trên cũng được thực hiện tương tự như việc mô hình hoá mệnh đề hợp thành có một điều kiện, trong đó liên kết và giữa các mệnh đề (hay giá trị mờ) được thực hiện bằng phép giao các tập mờ $A_1,$

A_2, \dots, A_n Với nhau theo công thức:

$$\mu_{A1} \cap A_2(x) = \min \{ \mu_{A1}(x), \mu_{A2}(x) \}.$$

Kết quả của phép giao sẽ là độ thỏa mãn H của luật (hình 1-12).

Các bước xây dựng luật hợp thành R như sau:

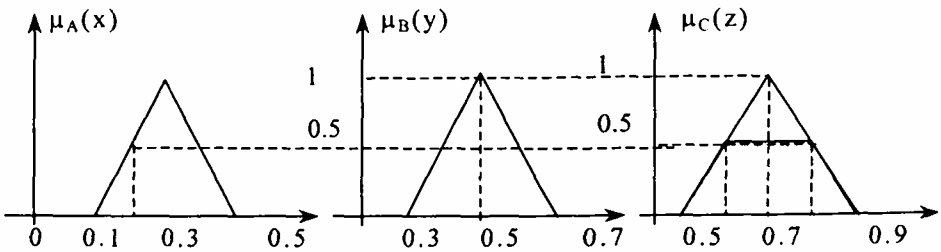
1- Rời rạc hoá miền xác định hàm liên thuộc $\mu_{A1}(x_1), \mu_{A2}(x_2), \dots, \mu_{Ad}(x_d), \mu_B(y)$ của các mệnh đề điều kiện và mệnh đề kết luận.

2- Xác định độ thỏa mãn H cho từng véctơ các giá trị rõ đầu vào là véctơ tổ hợp d điểm mẫu thuộc miền xác định của các hàm liên thuộc $\mu_{Ai}(x)$, ($i = 1, 2, \dots, d$).

Chẳng hạn với một véctơ các giá trị rõ đầu vào:

$$\underline{x} = \begin{pmatrix} c_1 \\ \dots \\ c_d \end{pmatrix} \text{ trong đó } c_i \text{ (} i=1, 2, \dots, d \text{) là một trong các điểm mẫu trong miền xác định của } \mu_{Ai}(x) \text{ thì:}$$

$$H = \text{MIN} \{ \mu_{A1}(c_1), \mu_{A2}(c_2), \dots, \mu_{Ad}(c_d) \}$$



Hình 1.13. Xây dựng R cho luật hợp thành hai mệnh đề điều kiện

3- Lập R gồm các hàm liên thuộc giá trị mờ đầu ra cho từng véctơ các giá trị đầu vào theo nguyên tắc:

$$\mu_{B'}(y) = \text{MIN} \{ H, \mu_B(y) \} \text{ Nếu sử dụng quy tắc MAX-MIN}$$

$$\mu_{B'}(y) = H, \mu_B(y) \text{ Nếu sử dụng quy tắc MAX-PROD.}$$

Chú ý: Đối với luật hợp thành R có d mệnh đề điều kiện không thể biểu diễn dưới dạng ma trận được nữa mà thành một lưới trong không gian d + 1 chiều.

Thật vậy, xét một mệnh đề hợp thành với hai mệnh đề điều kiện:

Nếu $\chi = A$ và $\gamma = B$ thì $\zeta = C$

Luật hợp thành R của nó có dạng như hình 2.12:

$$R: A \wedge B \Rightarrow C$$

Các bước xây dựng R như sau:

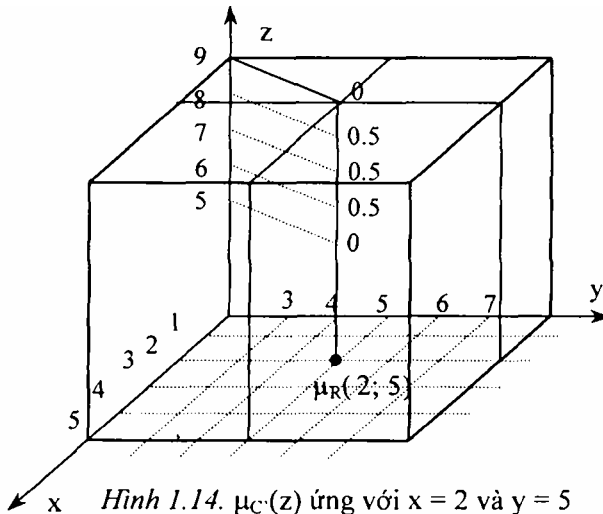
1. Rời rạc hoá các hàm liên thuộc:

- Hàm liên thuộc $\mu_A(x)$ được rời rạc hoá tại 5 điểm: $x \in \{1; 2; 3; 4; 5\}$.

- Hàm liên thuộc $\mu_B(y)$ được rời rạc hoá tại 5 điểm: $y \in \{3; 4; 5; 6; 7\}$.

- Hàm liên thuộc $\mu_C(z)$ được rời rạc hoá tại 5 điểm: $z \in \{5; 6; 7; 8; 9\}$.

2. Lập R gồm các hàm liên thuộc cho từng vector giá trị đầu vào và ứng với từng cặp điểm đầu vào là một hàm liên thuộc $\mu_{C'}(z)$ của biến mờ đầu ra C' (hình 1.14).



1.5.7. Luật của nhiều mệnh đề hợp thành

Trong thực tế hầu như không bộ Điều khiển mờ nào chỉ làm việc với một mệnh đề hợp thành mà thông thường với nhiều mệnh đề hợp thành? hay còn gọi là một tập các luật điều khiển R_k . sau đây ta sẽ trình bày cách liên kết các luật điều khiển riêng rẽ R_k lại với nhau trong một bộ điều khiển chung và qua đó mà nêu bật được ý nghĩa của ký hiệu "MAX" sử dụng trong tên gọi luật hợp thành như MAX- MIN hay MAX-PROD.

a) Luật hợp thành của hai mệnh đề hợp thành

Xét luật điều khiển gồm hai mệnh đề hợp thành:

R1: Nếu $\chi = A_1$ thì $\gamma = B_1$ hoặc

R2: Nếu $\chi = A_2$ thì $\gamma = B_2$

Hàm liên thuộc của các tập mờ được mô tả trong hình 2.15.

Ký hiệu R là luật hợp thành chung của bộ điều khiển, ta có: $R = R_1 \cup R_2$
 Ký hiệu hàm liên thuộc của R1 là $\mu_{R1}(x, y)$ và của R2 là $\mu_{R2}(x, y)$, thì theo công thức $\mu_{A \cup B}(x) = \max \{ \mu_A(x), \mu_B(x) \}$.

Hàm liên thuộc của R sẽ được xác định: $\mu_R(x, y) = \max \{ \mu_{R1}(x, y), \mu_{R2}(x, y) \}$. Với một giá trị rõ x_0 tại đầu vào, ta có độ thỏa mãn của các mệnh đề điều kiện như sau:

Đối với luật điều khiển R1:

- Độ thỏa mãn: $H_1 = \mu_{A1}(x_0)$

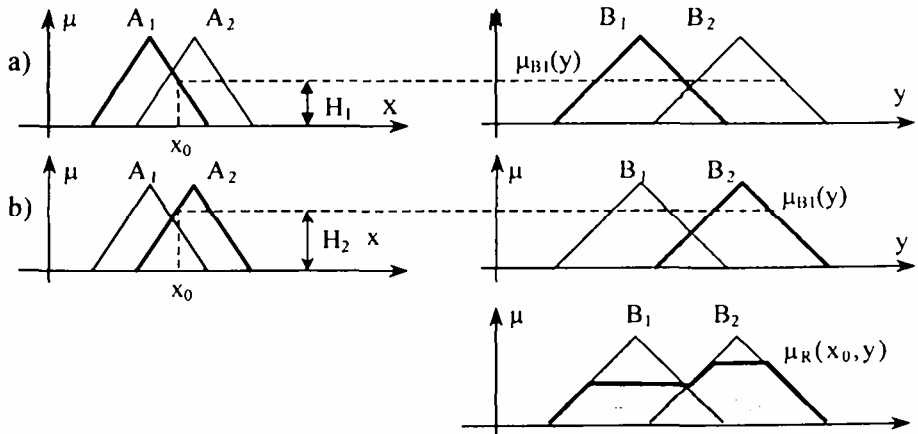
- Giá trị mờ đầu ra B1: $\mu_{B1}(y) = \min \{ H_1, \mu_{B1}(y) \}$ (hình 2.15a).

Đối với luật điều khiển R2:

- Độ thỏa mãn: $H_2 = \mu_{A2}(x_0)$

- Giá trị mờ đầu ra B2: $\mu_{B2}(y) = \min \{ H_2, \mu_{B2}(y) \}$ (hình 2.15b).

Từ đây ta có: $\mu_R(x_0, y) = \max \{ \mu_{B1}(y), \mu_{B2}(y) \}$



Hình 2.15. hàm liên thuộc của luật Điều khiển theo quy tắc MAX-MIN

a) Xác định hàm liên thuộc đầu ra của luật Điều khiển thứ nhất.

b) Xác định hàm liên thuộc đầu ra của luật điều khiển thứ hai.

c) Hàm liên thuộc đầu ra của luật hợp thành.

Đó chính là hàm liên thuộc của giá trị mờ đầu ra B' của bộ điều khiển gồm hai luật điều khiển $R = R_1 \cup R_2$ khi đầu vào là một giá trị rõ x_0 (hình 2.15c).

Để xác định luật hợp thành chung R , trước hết hai cơ sở X và Y của các giá trị A_1, A_2 và B_1, B_2 được rời rạc hoá, giả sử tại các điểm:

$$X = \{x_1, x_2, x_3, \dots, x_n\} \text{ (n điểm mẫu)}$$

$$Y = \{y_1, y_2, y_3, \dots, y_m\} \text{ (m điểm mẫu)}.$$

Giá trị của các hàm liên thuộc $\mu_{A_1}(x), \mu_{A_2}(x), \mu_{B_1}(y), \mu_{B_2}(y)$ sau khi rời rạc hoá là

$$\underline{\mu}_{A_1}^T = (\mu_{A_1}(x_1), \mu_{A_1}(x_2), \dots, \mu_{A_1}(x_n))$$

$$\underline{\mu}_{A_2}^T = (\mu_{A_2}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_2}(x_n))$$

$$\underline{\mu}_{B_1}^T = (\mu_{B_1}(y_1), \mu_{B_1}(y_2), \dots, \mu_{B_1}(y_m))$$

$$\underline{\mu}_{B_2}^T = (\mu_{B_2}(y_1), \mu_{B_2}(y_2), \dots, \mu_{B_2}(y_m)).$$

Từ đây suy ra:

$$R_1 = \underline{\mu}_{A_1} \cdot \underline{\mu}_{B_1}^T = \begin{pmatrix} r_{11}^1 & \dots & r_{1m}^1 \\ \dots & \dots & \dots \\ r_{n1}^1 & \dots & r_{nm}^1 \end{pmatrix}$$

$$R_2 = \underline{\mu}_{A_2} \cdot \underline{\mu}_{B_2}^T = \begin{pmatrix} r_{11}^2 & \dots & r_{1m}^2 \\ \dots & \dots & \dots \\ r_{n1}^2 & \dots & r_{nm}^2 \end{pmatrix}$$

và do đó luật hợp thành chung sẽ là:

$$R = R_1 \cup R_2 = \begin{pmatrix} \max\{r_{11}^1, r_{11}^2\} & \dots & \max\{r_{1m}^1, r_{1m}^2\} \\ \dots & \dots & \dots \\ \max\{r_{n1}^1, r_{n1}^2\} & \dots & \max\{r_{nm}^1, r_{nm}^2\} \end{pmatrix}.$$

b) Luật hợp thành của nhiều mệnh đề hợp thành

Xét luật điều khiển R gồm p mệnh đề hợp thành:

R_1 : Nếu $\chi = A_1$ Thì $Y = B_1$ hoặc

R_2 : Nếu $\chi = A_2$ Thì $Y = B_2$ hoặc

.....

R_p : Nếu $\chi = A_p$ Thì $Y = B_p$

trong đó các giá trị mờ A_1, A_2, \dots, A_p có cùng cơ sở X và B_1, B_2, \dots, B_p có cùng cơ sở Y.

Gọi hàm liên thuộc của A_k và B_k là $\mu_{A_k}(x)$ và $\mu_{B_k}(y)$ với $k = 1, 2, \dots, p$.

Thuật toán triển khai: $R = R_1 \cup R_2 \cup \dots \cup R_p$ được thực hiện theo các bước sau:

Bước 1: Rời rạc hoá X tại n điểm $(x_1, x_2, x_3, \dots, x_n)$ Và Y tại m điểm $(y_1, y_2, y_3, \dots, y_m)$

Bước 2: Xác định các vectơ $\underline{\mu}_{A_k}$ và $\underline{\mu}_{B_k}$ ($k = 1, 2, \dots, p$) tại các điểm rời rạc theo biểu thức:

$$\underline{\mu}_{A_k}^T = \{\mu_{A_k}(x_1), \mu_{A_k}(x_2), \dots, \mu_{A_k}(x_n)\}$$

$$\underline{\mu}_{B_k}^T = \{\mu_{B_k}(y_1), \mu_{B_k}(y_2), \dots, \mu_{B_k}(y_m)\}$$

Bước 3: Xác định mô hình (ma trận) R_k cho mệnh đề thứ k

$$R_k = \underline{\mu}_{A_k} \cdot \underline{\mu}_{B_k}^T = (r_{ij}^k), \quad i = 1, 2, \dots, n \text{ và } j = 1, 2, \dots, m$$

trong đó phép $(.)$ được thay bằng phép tính lấy cực tiểu min khi sử dụng nguyên tắc MAX-MIN và sử dụng phép nhân bình thường khi sử dụng nguyên tắc MAX-PROD.

Bước 4: Xác định luật hợp thành $R = \text{Max}(r_{ij}^k)$ với $k = 1, 2, \dots, p$.

1.5.7. Luật hợp thành SUM-MIN và SUM-PROD

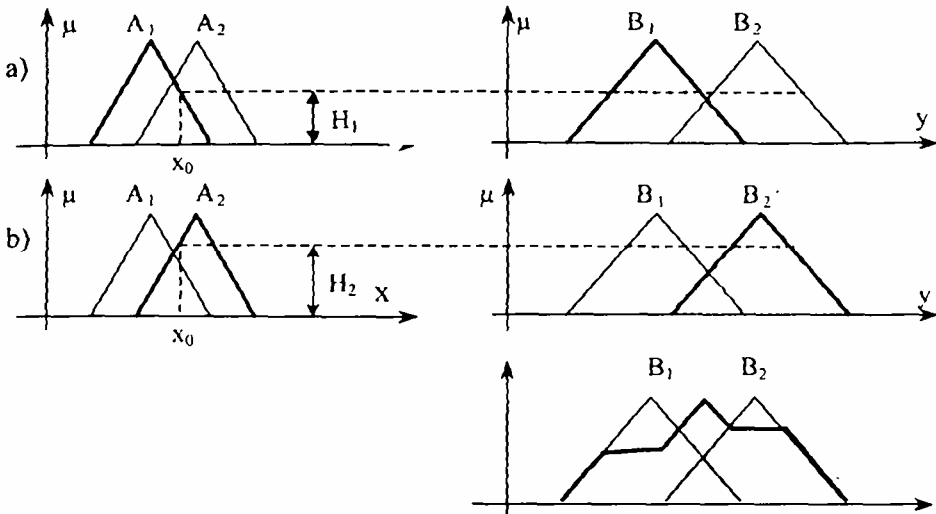
Ở phần trên, chúng ta đã tìm hiểu phương pháp xây dựng luật hợp thành chung R cho một tập gồm nhiều mệnh đề hợp thành R_k được liên kết với nhau bằng phép hợp theo biểu thức: $\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$. Kiểu liên kết này không có tính thống kê. Ví dụ khi đa số các mệnh đề hợp thành R_k có cùng một giá trị đầu ra nhưng không phải là giá trị lớn nhất sẽ không được đề ý tới và bị mất trong kết quả chung. Để khắc phục nhược điểm này phép hợp Lukasiewicz theo biểu:

$$\mu_{A \cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\} \text{ thay cho } \mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$$

để liên kết các luật điều khiển Rk lại với nhau thành luật hợp thành chung R

$$R = \min\left\{1, \sum_{k=1}^p R_k\right\}$$

trong đó phép lấy cực tiểu min được thực hiện giữa số 1 và từng phần tử của ma trận tổng. Ở công thức này, R được xác định bằng cách cộng các Rk của các mệnh đề hợp thành nên luật hợp thành chung R theo liên kết Lukasiewicz sẽ có tên gọi là SUM-MIN hoặc SUM-PROD.



Hình 2.16. Hàm liên thuộc của hợp hai luật điều khiển theo quy tắc SUM-MIN

Thuật toán triển khai R theo quy tắc SUM-MIN hay SUM-PROD cũng bao gồm các bước như khi triển khai với quy tắc MAX-MIN hoặc MAX-PROD đã trình bày ở mục trên chỉ khác ở bước 4 ta sử dụng công thức: $R =$

$$\min\left\{1, \sum_{k=1}^n R_k\right\}$$

Hình 1.16 là một ví dụ về mô hình hoá R gồm hai mệnh đề hợp thành theo quy tắc SUM-MIN.

1.6. GIẢI MỜ

Từ một giá trị rõ x_0 ở đầu vào, sau khi qua khối luật hợp thành ta có tập

mở đầu ra B' . Vấn đề đặt ra là cần phải xác định giá trị rõ y_0 từ tập mờ đầu ra đó. Muốn vậy ta cần thực hiện việc giải mờ.

Giải mờ là quá trình xác định một giá trị rõ y_0 nào đó có thể chấp nhận được từ hàm liên thuộc $\mu_{B'}(y)$ của giá trị mờ B' (tập mờ B').

Có hai phương pháp giải mờ chính là phương pháp cực đại và phương pháp điểm trọng tâm.

2.6.1. Phương pháp cực đại

Để giải mờ theo phương pháp cực đại, ta cần thực hiện 2 bước:

- **Xác định miền chứa giá trị rõ y_0 (miền G):** Đó là miền mà tại đó hàm liên thuộc $\mu_{B'}(y)$ đạt giá trị cực đại (độ cao H của tập mờ B'), tức là miền:

$$G = \{y \in Y \mid \mu_{B'}(y) = H\}$$

- **Xác định y_0 có thể chấp nhận được từ G .**

Hình 1.17 là tập mờ đầu ra của một luật hợp thành gồm 2 mệnh đề hợp thành:

$$R_1: \text{Nếu } \chi = A_1 \text{ Thì } \gamma = B_1$$

$$R_2: \text{Nếu } \chi = A_2 \text{ Thì } \gamma = B_2$$

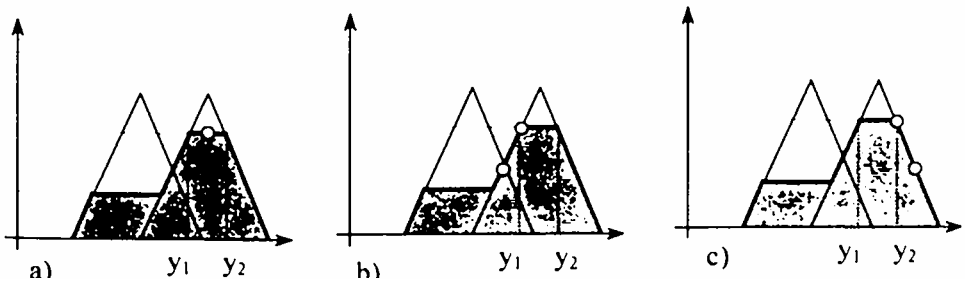
Miền chứa giá trị rõ G là khoảng $[y_1, y_2]$ của miền giá trị của tập mờ đầu ra B_2 của luật điều khiển:

$$R_2: \text{Nếu } \chi = A_2 \text{ Thì } \gamma = B_2$$

với y_1 là điểm cận trái của G $\left(y_1 = \inf_{y \in G}(y) \right)$ và y_2 là điểm cận phải của G $\left(y_2 = \sup_{y \in G}(y) \right)$. Khi đó, luật R_2 được gọi là luật Điều khiển quyết định.

Vậy luật điều khiển quyết định là luật $R_k, k \in \{1, 2, \dots, p\}$ mà giá trị mở đầu ra của nó có độ cao lớn nhất (Bằng độ cao H của B').

Để xác định y_0 trong khoảng $[y_1, y_2]$ ta có thể áp dụng theo một trong ba nguyên lý: Nguyên lý trung bình; nguyên lý cận trái và nguyên lý cận phải.



Hình 1.17a.b.c. Các nguyên lý giải mờ theo phương pháp cực đại

a) Nguyên lý trung bình

Giá trị rõ y_0 sẽ là trung bình cộng của y_1 và y_2

$$y_0 = \frac{y_1 + y_2}{2}$$

b) Nguyên lý cận trái

Giá trị rõ y_0 được lấy bằng cận trái y_1 của G

$$\left(y_1 = \inf_{y \in G}(y) \right).$$

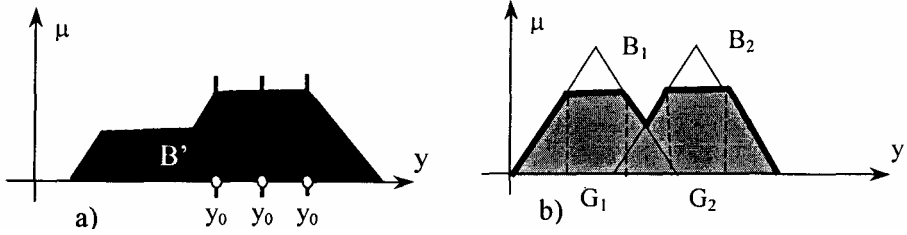
c) Nguyên lý cận phải

Giá trị rõ y_0 được lấy bằng cận phải y_2 của G

$$\left(y_2 = \sup_{y \in G}(y) \right).$$

Nhận xét:

+ Giá trị rõ y_0 lấy theo nguyên lý trung bình sẽ không phụ thuộc vào độ thoả mãn của luật điều khiển quyết định nếu tập mờ B' là tập đều (hình 1.17a), còn theo nguyên lý cận trái và cận phải, giá trị rõ y_0 Phụ thuộc tuyến tính vào độ thoả mãn của luật điều khiển quyết định (hình 1.17b,c).

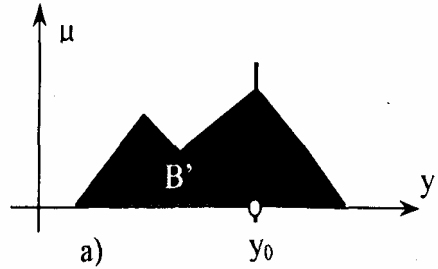


Hình 1.18. a) y_0 với các nguyên tắc chọn khác nhau
 b) Hàm liên thuộc B' có miền G không liên thông

+ Sai lệch của ba giá trị rõ, xác định theo nguyên lý trung bình, cận trái hay cận phải sẽ càng lớn nếu độ thoả mãn H của luật điều khiển càng nhỏ (hình 1.18a).

+ Khi miền G là miền không liên thông sử dụng phương pháp cực đại sẽ không chính xác (hình 2.18b).

+ Đối với luật hợp thành MAX-PROD, miền G chỉ có một điểm duy nhất, do đó kết quả giải mờ theo cả 3 nguyên lý đề giống nhau (hình 1.19).



Hình 1.19. Hàm liên thuộc B' đối với luật hợp thành MAX-PROD

1.6.2. Phương pháp điểm trọng tâm

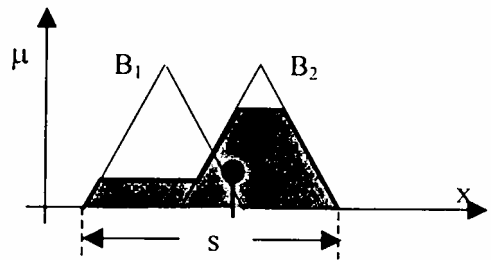
Giải mờ theo phương pháp điểm trọng tâm sẽ cho ra kết quả y' là hoành độ của điểm trọng tâm miền được bao bởi trục hoành và đường $\mu_{B'}(y)$ (hình 1.20). Công thức xác định y_0 theo phương pháp điểm trọng tâm như sau:

$$y' = \frac{\int_s \mu_{B'}(y) dy}{\int_s \mu_{B'}(y) dy}$$

Với s là miền xác định của tập mờ B' .

a) Phương pháp điểm trọng tâm cho luật hợp thành SUM-MIN

Giả sử có q luật điều khiển được triển khai. Khi đó mỗi giá trị mờ B' tại đầu ra của bộ điều khiển sẽ là tổng của



Hình 2.20. Giá trị rõ y' là hoành độ của điểm trọng tâm

q giá trị mờ đầu ra của từng luật hợp thành. Ký hiệu giá trị mờ đầu ra của luật điều khiển thứ k là $\mu_{B^k}(y)$ với $k = 1, 2, \dots, q$. Với quy tắc SUM- MIN, hàm liên thuộc $\mu_{B^k}(x)$ sẽ là:

$$\mu_{B^k}(y) = \sum_{k=1}^q \mu_{B^k}(y).$$

sau khi biên đổi, ta có:

$$y' = \frac{\int_S [y \sum_{k=1}^q \mu_{B^k}(y)] dy}{\int_S \sum_{k=1}^q \mu_{B^k}(y) dy} = \frac{\sum_{k=1}^q [\int_S y \mu_{B^k}(y) dy]}{\sum_{k=1}^q [\int_S \mu_{B^k}(y) dy]} = \frac{\sum_{k=1}^q M_k}{\sum_{k=1}^q A_k}$$

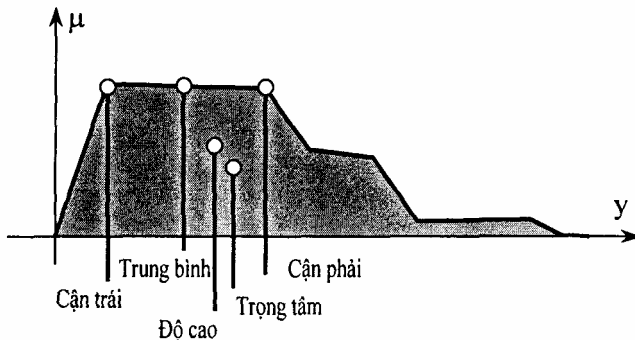
trong đó: $M_k = \int_S y \mu_{B^k}(y) dy$ và $A_k = \int_S \mu_{B^k}(y) dy$.

b) Phương pháp độ cao Sử dụng công thức:

$$y' = \frac{\int_S [y \sum_{k=1}^q \mu_{B^k}(y)] dy}{\int_S \sum_{k=1}^q \mu_{B^k}(y) dy} = \frac{\sum_{k=1}^q [\int_S y \mu_{B^k}(y) dy]}{\sum_{k=1}^q [\int_S \mu_{B^k}(y) dy]} = \frac{\sum_{k=1}^q M_k}{\sum_{k=1}^q A_k}$$

Cho cả hai luật hợp thành MAX-MIN và SUM-MIN với thêm một giả thiết là mỗi tập mờ $\mu_{B^k}(y)$ được xấp xỉ bằng một cặp giá trị (y_k, H_k) duy nhất (singleton), trong đó H_k là độ cao của $\mu_{B^k}(y)$ và y_k là một điểm mẫu trong miền giá trị của $\mu_{B^k}(y)$

Ta có: $\mu_{B^k}(y) = H_k$ và $y' = \frac{\sum_{k=1}^q y_k H_k}{\sum_{k=1}^q H_k}$.



Hình 1.21. So sánh các phương pháp giải mờ

Chú ý: Tùy hình dạng hàm liên thuộc B' mà sai khác giữa các phương pháp giải mờ có khác nhau. Hình 1.21 cho biết kết quả các phương pháp giải mờ ứng với một hàm liên thuộc B' cụ thể.

Chương 2

ĐIỀU KHIỂN MỜ

2.1. CẤU TRÚC CỦA BỘ ĐIỀU KHIỂN MỜ

2.1.1. Sơ đồ khối bộ điều khiển mờ

Hoạt động của một bộ điều khiển mờ phụ thuộc vào kinh nghiệm và phương pháp rút ra kết luận theo tư duy của con người sau đó được cài đặt vào máy tính trên cơ sở logic mờ.

Một bộ điều khiển mờ bao gồm 3 khối cơ bản: Khối mờ hoá, thiết bị hợp thành và khối giải mờ. Ngoài ra còn có khối giao diện vào và giao diện ra (hình 2.1).



Hình 2.1. Các khối chức năng của bộ Điều khiển mờ

- **Khối mờ hoá** có chức năng chuyển mỗi giá trị rõ của biến ngôn ngữ đầu vào thành vectơ μ có số phần tử bằng số tập mờ đầu vào.

- **Thiết bị hợp thành** mà bản chất của nó sự triển khai luật hợp thành R được xây dựng trên cơ sở luật điều khiển.

- **Khối giải mờ** có nhiệm vụ chuyển tập mờ đầu ra thành giá trị rõ y_0 (ứng với mỗi giá trị rõ x_0 để điều khiển đối tượng).

- **Giao diện đầu vào** thực hiện việc tổng hợp và chuyển đổi tín hiệu vào (từ tương tự sang số), ngoài ra còn có thể có thêm các khâu phụ trợ để thực hiện bài toán động như tích phân, vi phân....

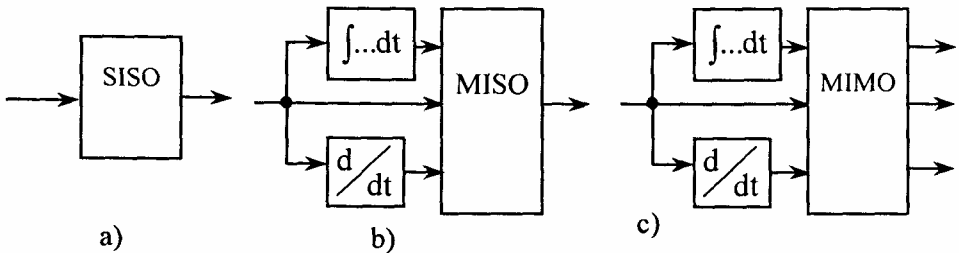
- **Giao diện đầu ra** thực hiện chuyển đổi tín hiệu ra (từ số sang tương tự) để điều khiển đối tượng.

Nguyên tắc tổng hợp một bộ điều khiển mờ hoàn toàn dựa vào những phương pháp toán học trên cơ sở định nghĩa các biến ngôn ngữ vào/ra và sự lựa chọn những luật điều khiển. Do các bộ điều khiển mờ có khả năng xử lý các giá trị vào/ra biểu diễn dưới dạng dấu phẩy động với độ chính xác cao nên chúng hoàn toàn đáp ứng được các yêu cầu của một bài toán điều khiển "rõ ràng" và "chính xác"

2.1.2. Phân loại bộ điều khiển mờ

Cũng giống như điều khiển kinh điển, bộ điều khiển mờ được phân loại dựa trên các quan điểm khác nhau:

Theo số lượng đầu vào và đầu ra ta phân ra bộ Điều khiển mờ "Một vào - một ra" (SISO); "Nhiều vào - một ra" (MISO); "Nhiều vào - nhiều ra" (MIMO) (hình 2.2a,b,c).



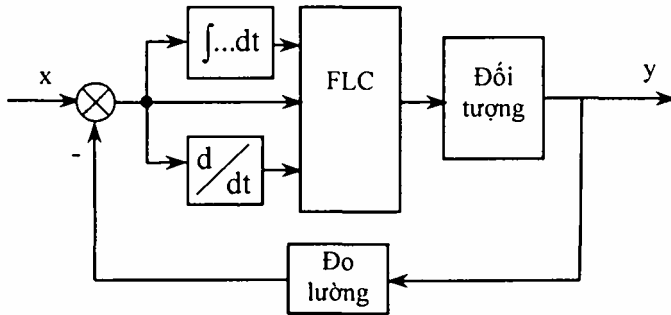
Hình 2.2a,b,c. Các bộ điều khiển mờ

Bộ điều khiển mờ MIMO rất khó cài đặt thiết bị hợp thành. Mặt khác, một bộ điều khiển mờ có m đầu ra dễ dàng cài đặt thành m bộ điều khiển mờ chỉ có một đầu ra vì vậy bộ điều khiển mờ MIMO chỉ có ý nghĩa về lý thuyết, trong thực tế không dùng.

- Theo bản chất của tín hiệu đưa vào bộ điều khiển ta phân ra bộ điều khiển mờ tĩnh và bộ điều khiển mờ động. Bộ điều khiển mờ tĩnh chỉ có khả năng xử lý các tín hiệu hiện thời, bộ điều khiển mờ động có sự tham gia của các giá trị đạo hàm hay tích phân của tín hiệu, chúng được ứng dụng cho các bài toán điều khiển động. Bộ điều khiển mờ tĩnh chỉ có khả năng xử lý các giá trị tín hiệu hiện thời. Để mở rộng miền ứng dụng của chúng vào các bài toán điều khiển động, các khâu động học cần thiết sẽ được nối thêm vào bộ điều khiển mờ tĩnh nhằm cung cấp cho bộ điều khiển các giá trị đạo hàm hay tích phân của tín hiệu. Cùng với những khâu động học bổ sung này, bộ điều khiển tĩnh sẽ trở thành bộ Điều khiển mờ động.

2.1.3. Các bước tổng hợp bộ điều khiển mờ

Cấu trúc tổng quát của một hệ điều khiển mờ được chỉ ra trên hình 2.3.



Hình 2.3. Cấu trúc tổng quát một hệ mờ

Với một miền compact $X \subset \mathbb{R}_n$ (n là số đầu vào) các giá trị vật lý của biến ngôn ngữ đầu vào và một đường phi tuyến $g(x)$ tùy ý nhưng liên tục cùng các đạo hàm của nó trên X thì bao giờ cũng tồn tại một bộ điều khiển mờ cơ bản có quan hệ:

$$\sup_{x \in X} |y(x) - g(x)| < \varepsilon \text{ với } \varepsilon \text{ là một số thực dương bất kỳ cho trước.}$$

Điều đó cho thấy kỹ thuật điều khiển mờ có thể giải quyết được một bài toán tổng hợp điều khiển (tính) phi tuyến bất kỳ.

Để tổng hợp được các bộ Điều khiển mờ và cho nó hoạt động một cách hoàn thiện ta cần thực hiện qua các bước sau:

1- Khảo sát đối tượng, từ đó định nghĩa tất cả các biến ngôn ngữ vào, ra và miền xác định của chúng. Trong bước này chúng ta cần chú ý một số đặc điểm cơ bản của đối tượng điều khiển như: Đối tượng biến đổi nhanh hay chậm? có trễ hay không? tính phi tuyến nhiều hay ít?,... Đây là những thông tin rất quan trọng để quyết định miền xác định của các biến ngôn ngữ đầu vào, nhất là các biến động học (vận tốc, gia tốc,...). Đối với tín hiệu biến thiên nhanh cần chọn miền xác định của vận tốc và gia tốc lớn và ngược lại.

2- Mờ hoá các biến ngôn ngữ vào/ra: Trong bước này chúng ta cần xác định số lượng tập mờ và hình dạng các hàm liên thuộc cho mỗi biến ngôn ngữ. Số lượng các tập mờ cho mỗi biến ngôn ngữ được chọn tùy ý. Tuy nhiên nếu chọn ít quá thì việc điều chỉnh sẽ không mịn, chọn nhiều quá sẽ khó khăn

khi cài đặt luật hợp thành, quá trình tính toán lâu, hệ thống dễ mất ổn định. Hình dạng các hàm liên thuộc có thể chọn hình tam giác, hình thang, hàm Gaus,...

3- Xây dựng các luật điều khiển (mệnh đề hợp thành): Đây là bước quan trọng nhất và khó khăn nhất trong quá trình thiết kế bộ điều khiển mờ. Việc xây dựng luật điều khiển phụ thuộc rất nhiều vào tri thức và kinh nghiệm vận hành hệ thống của các chuyên gia. Hiện nay ta thường sử dụng một vài nguyên tắc xây dựng luật hợp thành đủ để hệ thống làm việc, sau đó mô phỏng và chỉnh định dần các luật hoặc áp dụng một số thuật toán tối ưu (được trình bày ở phần sau).

4- Chọn thiết bị hợp thành (MAX-MIN hoặc MAX-PROD hoặc SUM-MIN hoặc SUM-PRROD) và chọn nguyên tắc giải mờ (Trung bình, cận trái, cận phải, điểm trọng tâm, độ cao).

5- Tối ưu hệ thống: Sau khi thiết kế xong bộ điều khiển mờ, ta cần mô hình hoá và mô phỏng hệ thống để kiểm tra kết quả, đồng thời chỉnh định lại một số tham số để có chế độ làm việc tối ưu. Các tham số có thể điều chỉnh trong bước này là. Thêm, bớt luật điều khiển; Thay đổi trọng số các luật; Thay đổi hình dạng và miền xác định của các hàm liên thuộc.

2.2. BỘ ĐIỀU KHIỂN MỜ TÍNH

2.2.1. Khái niệm

Bộ điều khiển tính là bộ điều khiển mờ có quan hệ vào/ra $y(x)$, với x là đầu vào và y là đầu ra, theo dạng một phương trình đại số (tuyến tính hoặc phi tuyến). Bộ điều khiển mờ tính không xét tới các yếu tố "động" của đối tượng (vận tốc, gia tốc,...). Các bộ điều khiển tính điển hình là bộ khuếch đại P, bộ điều khiển re lay hai vị trí, ba vị trí,...

2.2.2. Thuật toán tổng hợp một bộ điều khiển mờ tính

Các bước tổng hợp bộ điều khiển mờ tính về cơ bản giống các bước chung để tổng hợp bộ điều khiển mờ như đã trình bày ở trên. Để hiểu kỹ hơn ta xét ví dụ cụ thể sau:

Ví dụ: Hãy thiết kế bộ điều khiển mờ tính SISO có hàm truyền đạt $y = f(x)$ trong khoảng $x = [a_1, a_2]$ tương ứng với y trong khoảng $y = [\beta_1, \beta_2]$.

Bước 1: Định nghĩa các tập mờ vào, ra

- Định nghĩa N tập mờ đầu vào: A_1, A_2, \dots, A_n trên khoảng $[a_1, a_2]$ của x có hàm liên thuộc $\mu_{A_i}(x)$ ($i = 1, 2, \dots, N$) dạng hình tam giác cân.

- Định nghĩa N tập mờ đầu ra: B_1, B_2, \dots, B_N trên khoảng $[\beta_1, \beta_2]$ của y có hàm liên thuộc $\mu_{B_j}(x)$ ($j = 1, 2, \dots, N$) dạng hình tam giác cân.

Bước 2: Xây dựng luật điều khiển

Với N hàm liên thuộc đầu vào ta sẽ xây dựng được N luật điều khiển theo cấu trúc:

$$R^i: \text{nếu } \chi = A_i; \text{ thì } \gamma = B_i.$$

Bước 3: Chọn thiết bị hợp thành

Giả thiết chọn nguyên tắc triển khai SUM-PROD cho mệnh đề hợp thành, và công thức Lukasiewicz cho phép hợp thì tập mờ đầu ra B' khi đầu vào là một giá trị rõ x_0 sẽ là:

$$\mu_{B'}(y) = \text{MIN} \left\{ 1, \sum_{i=1}^N \mu_{B_i}(y) \mu_{A_i}(x_0) \right\}. \quad (2.1)$$

vì $\mu_{B_i}(y)$ là một hàm Kronecker $\mu_{B_i}(y) \mu_{A_i}(x_0) = \mu_{A_i}(x_0)$ khi đó:

$$\mu_{B'}(y) = \text{MIN} \left\{ 1, \sum_{i=1}^N \mu_{A_i}(x_0) \right\}. \quad (2.2)$$

Bước 4: Chọn phương pháp giải mờ

Chọn phương pháp độ cao để giải mờ, ta có:

$$y(x_0) = \frac{\sum_{i=1}^N y_i H_i}{\sum_{i=1}^N H_i} = \frac{\sum_{i=1}^N y_i \mu_{A_i}(x_0)}{\sum_{i=1}^N \mu_{A_i}(x_0)}. \quad (2.3)$$

Quan hệ truyền đạt của bộ điều khiển mờ có dạng:

$$y(x) = \frac{\sum_{i=1}^N y_i \mu_{A_i}(x)}{\sum_{i=1}^N \mu_{A_i}(x)}. \quad (2.4)$$

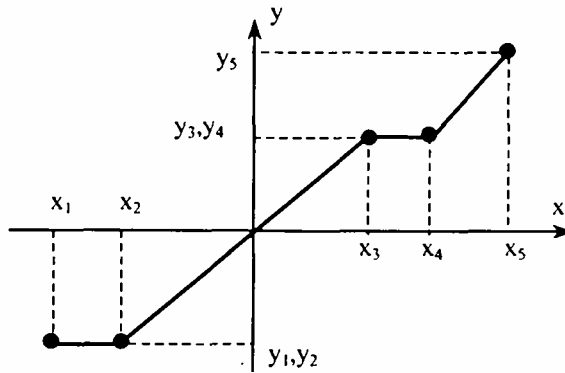
2.2.3. Tổng hợp bộ điều khiển mờ tuyến tính từng đoạn

Trong kỹ thuật nhiều khi ta cần phải thiết kế bộ điều khiển mờ với đặc tính vào - ra cho trước tuyến tính từng đoạn. Chẳng hạn, cần thiết kế bộ điều khiển mờ có đặc tính vào - ra như hình 2.4.

Thuật toán tổng hợp bộ điều khiển này giống như thuật toán tổng hợp bộ điều khiển mờ với hàm truyền đạt $y(x)$ bất kỳ. Tuy nhiên, để các đoạn đặc tính thẳng và nối với nhau một cách liên tục tại các nút thì cần tuân thủ một số nguyên tắc sau:

+ Mỗi giá trị rõ đầu vào phải làm tích cực 2 luật điều khiển.

+ Các hàm liên thuộc đầu vào có dạng hình tam giác có đỉnh là một điểm ở nút k , có miền xác định là khoảng $[x_{k-1}, x_{k+1}]$ (hình 2.5a).



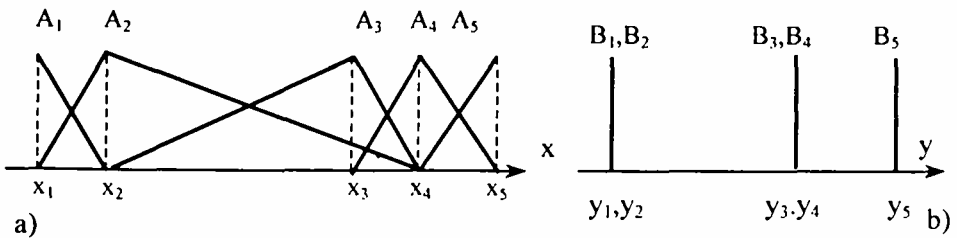
Hình 2.4. Đặc tính vào - ra cho trước

+ Các hàm liên thuộc đầu ra có dạng singleton tại các điểm nút y_k (hình 2.5b).

+ Cài đặt luật hợp thành Max-Min với luật điều khiển tổng quát:

$$R_k: \text{nếu } \chi = A_k; \text{ thì } \gamma = B_k.$$

+ Giải mờ bằng phương pháp độ cao.



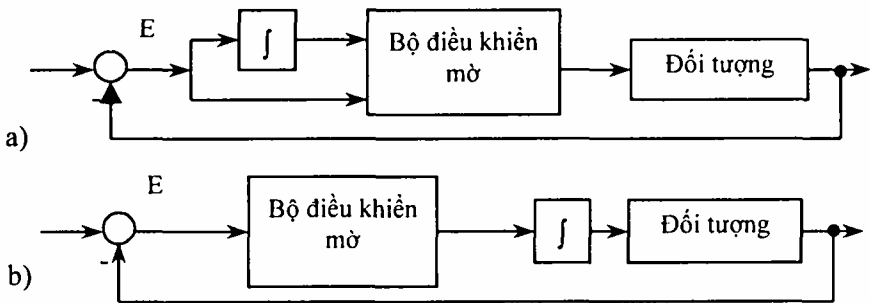
Hình 2.5 a.b. hàm liên thuộc của các biến ngôn ngữ vào, ra

2.3. BỘ ĐIỀU KHIỂN MỜ ĐỘNG

Bộ Điều khiển mờ động là bộ điều khiển mờ mà đầu vào có xét tới các trạng thái động của đối tượng như vận tốc, gia tốc, đạo hàm của gia tốc,... Ví dụ đối với hệ điều khiển theo sai lệch thì đầu vào của bộ điều khiển mờ ngoài tín hiệu sai lệch e theo thời gian còn có các đạo hàm của sai lệch giúp cho bộ điều khiển phản ứng kịp thời với các biến động đột xuất của đối tượng.

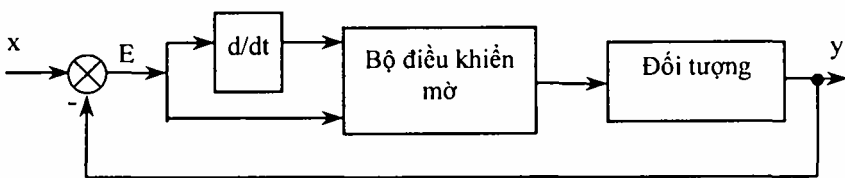
Các bộ điều khiển mờ động hay được dùng hiện nay là bộ điều khiển mờ theo luật tỉ lệ tích phân (PI), tỉ lệ vi phân (PD) và tỉ lệ vi tích phân (PID).

Một bộ điều khiển mờ theo luật I có thể thiết kế từ một bộ mờ theo luật P (bộ Điều khiển mờ tuyến tính) bằng cách mắc nối tiếp một khâu tích phân vào trước hoặc sau khối mờ đó. Do tính phi tuyến của hệ mờ, nên việc mắc khâu tích phân trước hay sau hệ mờ hoàn toàn khác nhau (hình 3.2 a,b).



Hình 2.6a,b. hệ điều khiển mờ theo luật PI

Khi mắc thêm một khâu vi phân ở đầu vào của một bộ điều khiển mờ theo luật tỉ lệ sẽ có được một bộ điều khiển mờ theo luật tỉ lệ vi phân PD (hình 2.4).



Hình 2.7. hệ điều khiển mờ theo luật PD

Các thành phần của bộ điều khiển này cũng giống như bộ điều khiển theo luật PD thông thường bao gồm sai lệch giữa tín hiệu chủ đạo và tín hiệu ra của hệ thống e và đạo hàm của sai lệch e' . Thành phần vi phân giúp cho hệ

thống phản ứng chính xác hơn với những biến đổi lớn của sai lệch theo thời gian.

Trong kỹ thuật Điều khiển kinh điển, bộ Điều khiển PID được biết đến như là một giải pháp đa năng và có miền ứng dụng rộng lớn. Định nghĩa về bộ điều khiển theo luật PID kinh điển trước đây vẫn có thể sử dụng cho một bộ điều khiển mờ theo luật PID. Bộ điều khiển mờ theo luật PID được thiết kế theo hai thuật toán:

- Thuật toán chỉnh định PID;
- Thuật toán PID tốc độ.

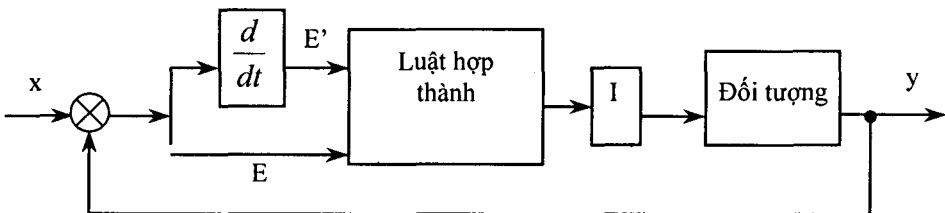
Bộ điều khiển mờ được thiết kế theo thuật toán chỉnh định PID có 3 đầu vào gồm sai lệch e giữa tín hiệu chủ đạo và tín hiệu ra, đạo hàm và tích phân của sai lệch. Đầu ra của bộ điều khiển mờ chính là tín hiệu điều khiển rút).

$$u(t) = K \left[e + \frac{1}{T_I} \int_0^t e \cdot dt + T_D \frac{d}{dt} e \right]. \quad (2.5)$$

Với thuật toán PID tốc độ, bộ điều khiển PID có 3 đầu vào: sai lệch e giữa tín hiệu đầu vào và tín hiệu chủ đạo, đạo hàm bậc nhất e' và đạo hàm bậc hai e'' của sai lệch. Đầu ra của hệ mờ là đạo hàm $\frac{du}{dt}$ của tín hiệu điều khiển $u(t)$.

$$\frac{du}{dt} = K \left[\frac{d}{dt} e + \frac{1}{T_I} e + \frac{d^2}{(dt)^2} e \right]. \quad (2.6)$$

Do trong thực tế thường có một hoặc hai thành phần trong (3.6), (3.7) được bỏ qua, nên thay vì thiết kế một bộ điều khiển PID hoàn chỉnh người ta lại thường tổng hợp các bộ điều khiển PI hoặc PD.



Hình 2.8. Hệ điều khiển mờ theo luật PID

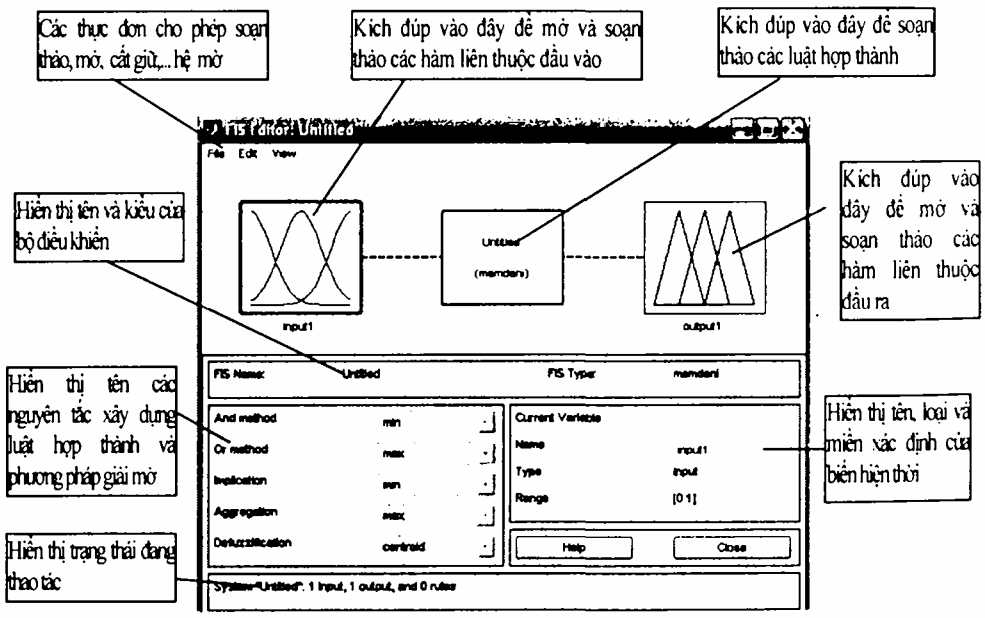
Bộ điều khiển PID mờ được thiết kế trên cơ sở của bộ điều khiển PD mờ bằng cách mắc nối tiếp ở đầu ra của bộ điều khiển PD mờ một khâu tích phân (hình 2.6).

Hiện nay đã có rất nhiều dạng cấu trúc khác nhau của PID mờ đã được nghiên cứu. Các dạng cấu trúc này thường được thiết lập trên cơ sở tách bộ điều chỉnh PID thành hai bộ điều chỉnh PD và PI (hoặc I). Việc phân chia này chỉ nhằm mục đích thiết lập các hệ luật cho PD và PI (hoặc I) gồm hai (hoặc 1) biến vào, một biến ra, thay vì phải thiết lập 3 biến vào. Hệ luật cho bộ điều chỉnh PID mờ kiểu này thường dựa trên ma trận do Mac Vicar-whelan đề xuất. Cấu trúc này không làm giảm số luật mà chỉ đơn giản cho việc tính toán.

2.4. THIẾT KẾ HỆ ĐIỀU KHIỂN MỜ BẰNG PHẦN MỀM MATLAB

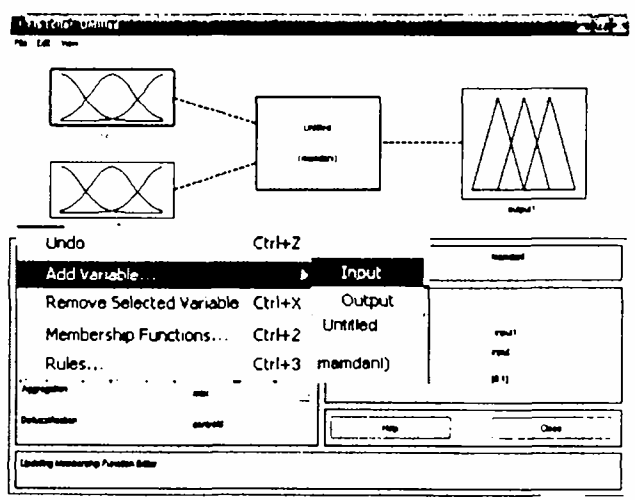
2.4.1. Giới thiệu hộp công cụ logic mờ

Hộp công cụ Logic mờ (The Fuzzy Logic Toolbox) là tổ hợp các hàm được xây dựng trên nền Matlab giúp cho việc thiết kế, mô phỏng, kiểm tra và hiệu chỉnh bộ điều khiển mờ một cách dễ dàng. Để thiết kế bộ điều khiển mờ trong hộp công cụ này, ta có thể thực hiện thông qua dòng lệnh hoặc thông qua giao diện đồ họa. Trong khuôn khổ cuốn sách này chỉ giới thiệu những thao tác cơ bản để thiết kế bộ điều khiển mờ thông qua giao diện đồ họa. Phần thiết kế thông qua dòng lệnh, ta có thể đọc trong phần "**Fuzzy Logic Toolbox**" của Malab.

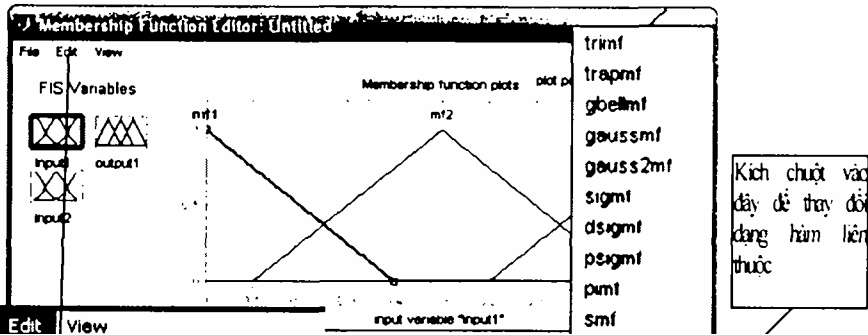


Hình 2.9

Sau khi đã có cấu trúc của bộ Điều khiển mờ, ta tiến hành soạn thảo các hàm liên thuộc vào, hàm liên thuộc ra, các luật điều khiển.

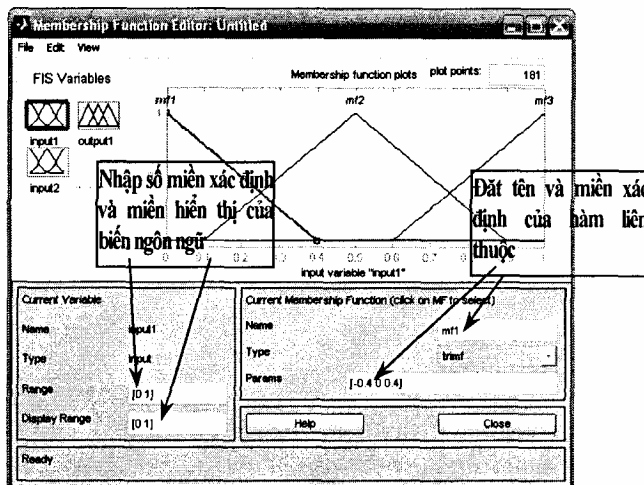


Hình 2.10

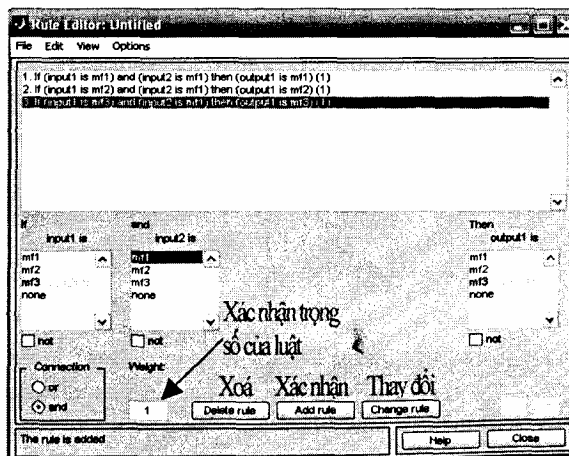


Hình 2.11

Kích đúp chuột vào biểu tượng Input (Hình 2.11) Chọn **Edit**, và chọn **Add MFs** hoặc **Add Custom MF** thêm hàm liên thuộc, chọn Remov Select MF để gỡ bỏ một hàm liên thuộc nào đó, nếu chọn Remov All MFs sẽ gỡ bỏ tất cả các hàm liên thuộc của biến đã chọn. Theo mặc định, số hàm liên thuộc là 3 có dạng tam giác, ta có thể thay đổi số lượng cũng như hình dạng hàm liên thuộc. Để thay đổi hình dạng một hàm liên thuộc nào đó, ta kích chuột vào hàm đó, nó sẽ chuyển sang màu đỏ, sau đó kích chuột vào hộp thoại như chỉ ra ở hình 2.12 để chọn hàm liên thuộc mong muốn. Trên ô Range và Display Range ta có nhập các giá trị về miền xác định và miền hiển thị của biến ngôn ngữ, mặc định của các miền đó là từ 0 đến 1. Trên ô Name và ô Params (hình 2.12) ta có thể đặt tên và miền xác định cho từng tập mờ.



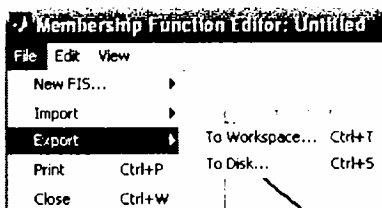
Hình 2.12



Hình 2.13. Cửa sổ soạn thảo luật

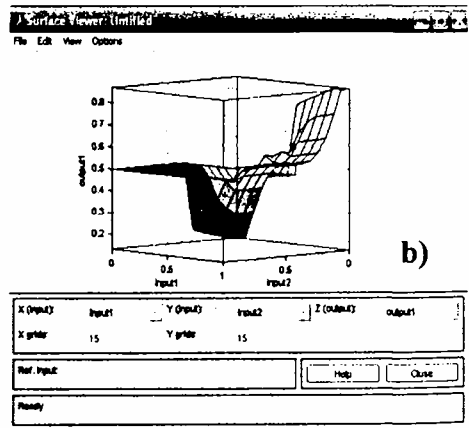
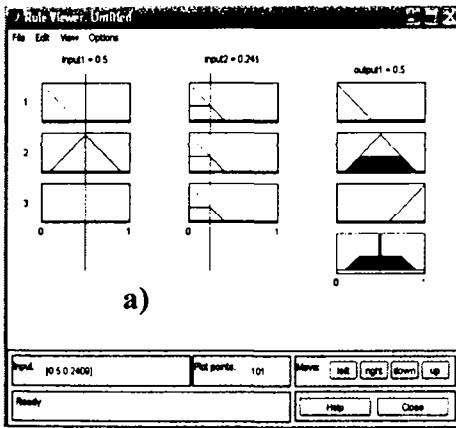
Để soạn thảo luật hợp thành, ta ấn **Edit, Rules** trên màn hình hiện ra cửa sổ hình 2.13. Sau mỗi lần soạn xong một luật ta ấn **Add rule** để xác nhận. Để thay đổi một luật hợp thành ta ấn **Change rule**. Để xoá một luật điều khiển ta ấn **Delete rules**. Muốn quan sát hoạt động của các luật ta ấn **View Rules**. Ấn **View Surface** để quan sát quan hệ vào – ra của bộ điều khiển (hình 2.14a, b).

Sau khi thiết kế xong bộ điều khiển, ta cần đặt tên và lưu chúng bằng cách ấn **File, Export To Disk** để cất vào đĩa hoặc to **Workspase** để lưu vào vùng làm việc của Matlab.



Muốn mở một bộ Điều khiển mờ đã lưu trên đĩa, ấn **File, Export To Disk** sau đó ấn **Import from disk**, chọn file cần mở.

Sau khi thiết kế xong bộ điều khiển mờ bằng cửa sổ Edit GUI, ta chuyển về cửa sổ mô phỏng SIMULINK, mở một file mới với đuôi '.mat', xây dựng mô hình mô phỏng cho hệ, tiến hành chạy mô phỏng và hiệu chỉnh hệ thống.



Hình 2. 14a.b. a) Quan sát hoạt động của các luật

b) Quan hệ vào-ra của bộ điều khiển

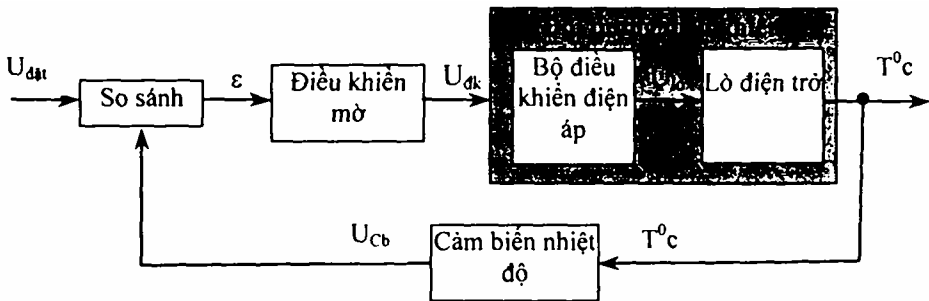
2.3.2. Ví dụ thiết kế hệ mờ

Để minh họa cho những vấn đề đã trình bày ở trên, sau đây chúng ta tiến hành phân tích, thiết kế bộ điều khiển mờ để điều khiển đối tượng nhiệt độ lò điện trở có hàm số truyền là:

$$W(s) = \frac{k}{T_s + 1} e^{-\tau_s s} = \frac{6,52}{150s + 1} e^{-25s} \quad (2.7)$$

Biết điện áp cấp cho lò có giá trị định mức là 230 V.

Sơ đồ khối của hệ được chỉ ra trên hình 2.15.



Hình 2.15. Sơ đồ khối hệ điều khiển nhiệt độ lò điện trở

Bước 1: Tìm hiểu hệ thống

Lò điện trở dùng để gia nhiệt chi tiết bằng kim loại cho các công đoạn như tô, ram... Lò điện trở được nung nóng bằng dây điện trở, nguồn điện

cung cấp cho lò là nguồn áp có thể điều chỉnh được. Việc điều khiển nhiệt độ lò được thực hiện thông qua điều khiển điện áp cung cấp cho lò. Trong kỹ thuật điều khiển, người ta mô tả lò bằng một khâu quán tính bậc nhất có trễ có hàm số truyền:

$$W(s) = \frac{k}{Ts + 1} e^{-\tau s}$$

Trong đó, hằng số thời gian T và thời gian trễ T có giá trị tùy vào loại lò và công suất lò.

☞ Bộ Điều khiển điện áp có điện áp điều chỉnh được và biến thiên trong khoảng từ 100V: 230V, được mô tả gần đúng bằng một khâu có hàm số truyền:

$$w(s) = ke^{-2s} \text{ với } k = 23, \tau = 0,05(s).$$

☞ Cảm biến nhiệt độ được coi là 1 khâu tỉ lệ với hệ số:

$$k = \frac{10V}{1500^{\circ}C} = 0,0067 (\text{V}/^{\circ}C).$$

☞ Điện áp đặt có giá trị lớn nhất là 10 V.

☞ Khâu so sánh làm nhiệm vụ so sánh điện áp đặt và điện áp phản hồi lấy từ đầu ra của khối cảm biến, đầu ra của khâu so sánh là sai lệch $e = U - u_{cb}$. Lò điện trở nói riêng, cũng như đối tượng nhiệt nói chung thường không cho phép có độ quá điều chỉnh, do đó e biến thiên trong khoảng từ 10 đến 0.

Bước 2: Chọn các biến ngôn ngữ vào, ra

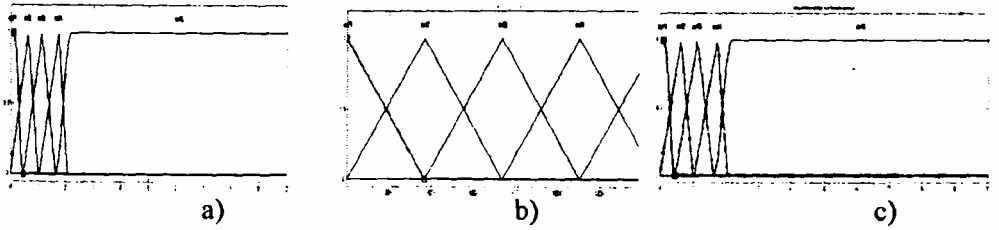
Giả thiết ta điều khiển lò điện trở theo quy luật PI, khi đó biến ngôn ngữ đầu vào bộ điều khiển mờ là sai lệch (ký hiệu là E) và tích phân sai lệch (ký hiệu là TE). Đầu ra bộ Điều khiển mờ là điện áp (ký hiệu là U). Miền giá trị của các biến ngôn ngữ được chọn như sau:

$E = [0 \div 10]$; $TE = [0 \div 1500]$; $U = [0 \div 20]$; hàm liên thuộc của các biến ngôn ngữ được chọn như hình 2.16a,b,c

$$\underline{\mu}_E^T = [\mu_{E1}(x) \mu_{E2}(x) \mu_{E3}(x) \mu_{E4}(x) \mu_{E5}(x)] \text{ (hình 2.16a);}$$

$$\underline{\mu}_{TE}^T = [\mu_{TE1}(x) \mu_{TE2}(x) \mu_{TE3}(x) \mu_{TE4}(x) \mu_{TE5}(x)] \text{ (hình 2.16b);}$$

$$\underline{\mu}_U^T = [\mu_{U1}(x) \mu_{U2}(x) \mu_{U3}(x) \mu_{U4}(x) \mu_{U5}(x)] \text{ (hình 2.16a);}$$



Hình 2.16a,b,c. Hình dạng các hàm liên thuộc đầu vào và đầu ra

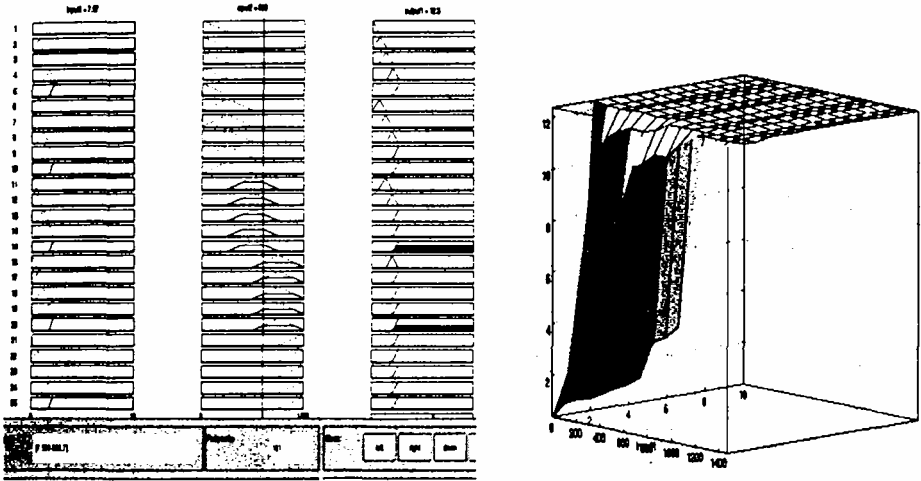
Bước 3: Xây dựng luật hợp thành: Với 5 tập mờ của mỗi đầu vào, ta xây dựng được $5 \times 5 = 25$ luật điều khiển. Các luật điều khiển này được xây dựng theo 2 nguyên tắc sau:

- Sai lệch càng lớn thì tác động điều khiển càng lớn.
- Tích phân sai lệch càng lớn thì tác động điều khiển càng lớn.

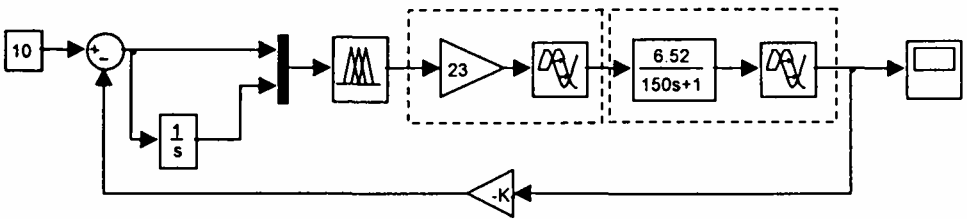
R ₁ :	Nếu	E	=	E ₁	và	TE	=	TE ₁	thì	U	=	U ₁	hoặc
R ₂ :	Nếu	E	=	E ₂	và	TE	=	TE ₁	thì	U	=	U ₂	hoặc
R ₃ :	Nếu	E	=	E ₃	và	TE	=	TE ₁	thì	U	=	U ₃	hoặc
R ₄ :	Nếu	E	=	E ₄	và	TE	=	TE ₁	thì	U	=	U ₄	hoặc
R ₅ :	Nếu	E	=	E ₅	và	TE	=	TE ₁	thì	U	=	U ₅	hoặc
R ₆ :	Nếu	E	=	E ₁	và	TE	=	TE ₂	thì	U	=	U ₂	hoặc
R ₇ :	Nếu	E	=	E ₂	và	TE	=	TE ₂	thì	U	=	U ₃	hoặc
R ₈ :	Nếu	E	=	E ₃	và	TE	=	TE ₂	thì	U	=	U ₄	hoặc
R ₉ :	Nếu	E	=	E ₄	và	TE	=	TE ₂	thì	U	=	U ₅	hoặc
R ₁₀ :	Nếu	E	=	E ₅	và	TE	=	TE ₂	thì	U	=	U ₅	hoặc
R ₁₁ :	Nếu	E	=	E ₁	và	TE	=	TE ₃	thì	U	=	U ₃	hoặc
R ₁₂ :	Nếu	E	=	E ₂	và	TE	=	TE ₃	thì	U	=	U ₄	hoặc
R ₁₃ :	Nếu	E	=	E ₃	và	TE	=	TE ₃	thì	U	=	U ₅	hoặc
R ₁₄ :	Nếu	E	=	E ₄	và	TE	=	TE ₃	thì	U	=	U ₅	hoặc
R ₁₅ :	Nếu	E	=	E ₅	và	TE	=	TE ₃	thì	U	=	U ₅	hoặc
R ₁₆ :	Nếu	E	=	E ₁	và	TE	=	TE ₄	thì	U	=	U ₄	hoặc
R ₁₇ :	Nếu	E	=	E ₂	và	TE	=	TE ₄	thì	U	=	U ₅	hoặc
R ₁₈ :	Nếu	E	=	E ₃	và	TE	=	TE ₄	thì	U	=	U ₅	hoặc
R ₁₉ :	Nếu	E	=	E ₄	và	TE	=	TE ₄	thì	U	=	U ₅	hoặc
R ₂₀ :	Nếu	E	=	E ₅	và	TE	=	TE ₄	thì	U	=	U ₅	hoặc
R ₂₁ :	Nếu	E	=	E ₁	và	TE	=	TE ₅	thì	U	=	U ₅	hoặc
R ₂₂ :	Nếu	E	=	E ₂	và	TE	=	TE ₅	thì	U	=	U ₅	hoặc
R ₂₃ :	Nếu	E	=	E ₃	và	TE	=	TE ₅	thì	U	=	U ₅	hoặc
R ₂₄ :	Nếu	E	=	E ₄	và	TE	=	TE ₅	thì	U	=	U ₅	hoặc
R ₂₅ :	Nếu	E	=	E ₅	và	TE	=	TE ₅	thì	U	=	U ₅	

Bước 4: Chọn luật hợp thành Max-Min, giải mờ bằng phương pháp trọng tâm, ta quan sát được sự tác động của các luật và quan hệ vào - ra của bộ điều khiển như hình 2.17a,b.

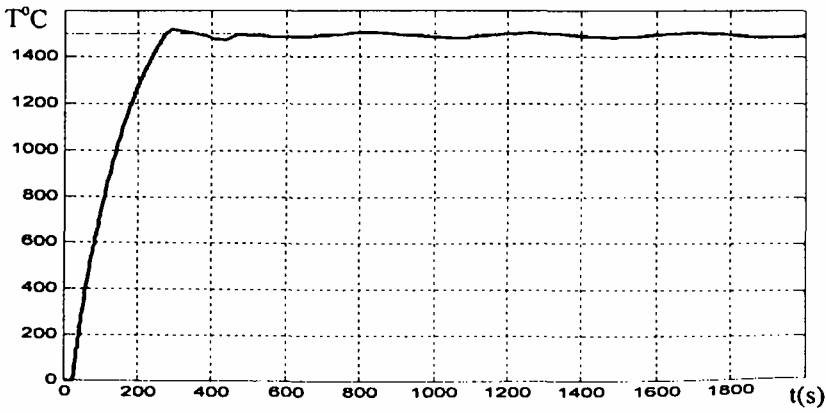
Bước 5: Mô phỏng hệ thống: Sơ đồ mô phỏng hệ thống được chỉ ra trên hình 2.18. Kết quả mô phỏng được chỉ ra trên hình 2.19.



Hình 2.17a, b. Quan hệ vào - Ra của bộ Điều khiển

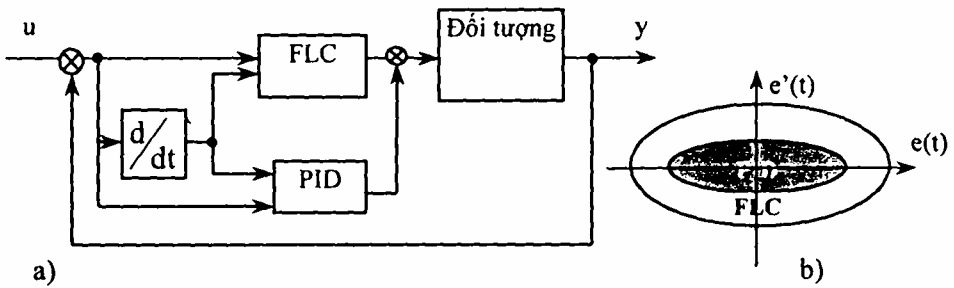


Hình 2.18. Sơ đồ mô phỏng hệ thống

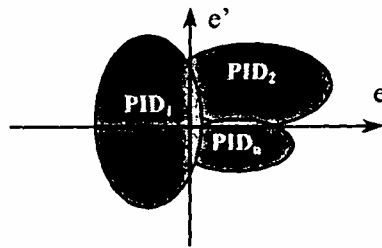


2.5. HỆ ĐIỀU KHIỂN MỜ LAI (F-PID)

Hệ mờ lai viết tắt là F-PID là hệ điều khiển trong đó thiết bị điều khiển gồm 2 thành phần: Thành phần điều khiển kinh điển và thành phần điều khiển mờ. Bộ Điều khiển F-PID có thể thiết lập dựa trên hai tín hiệu là sai lệch $e(t)$ và đạo hàm của nó $e'(t)$. Bộ Điều khiển mờ có đặc tính rất tốt ở vùng sai lệch lớn, ở đó với đặc tính phi tuyến của nó có thể tạo ra phản ứng động rất nhanh. Khi quá trình của hệ tiến gần đến điểm đặt (sai lệch $e(t)$ và đạo hàm của nó $e'(t)$ xấp xỉ bằng 0) vai trò của bộ điều khiển mờ (FLC) bị hạn chế nên bộ điều khiển sẽ làm việc như một bộ điều chỉnh PID bình thường. Trên hình 2.20 thể hiện ý tưởng thiết lập bộ điều khiển mờ lai F-PID và phân vùng tác động của chúng.



Hình 2.20. a) Nguyên lý điều khiển mờ lai;
b) Vùng tác động của các bộ điều khiển



Hình 2.21. Vùng tác động của các bộ điều khiển

Sự chuyển đổi giữa các vùng tác động của FLC và PID có thể thực hiện nhờ khoá mờ hoặc dùng chính FLC. Nếu sự chuyển đổi dùng FLC thì ngoài nhiệm vụ là bộ điều chỉnh FLC còn làm nhiệm vụ giám sát hành vi của hệ thống để thực hiện sự chuyển đổi. Việc chuyển đổi tác động giữa FLC và PID có thể thực hiện nhờ luật đơn giản sau:

if $|e(t)|$ dương lớn và $|\dot{e}(t)|$ dương lớn thì u là FLC (2.8)

if $|e(t)|$ dương nhỏ và $|\dot{e}(t)|$ dương nhỏ thì u là PID (2.9)

Để thực hiện chuyển đổi mờ giữa các mức FLC và bộ chuyển đổi PID, ta có thể thiết lập nhiều bộ điều chỉnh PID_i ($i = 1, 2, \dots, n$) mà mỗi bộ được chọn để tối ưu chất lượng theo một nghĩa nào đó đề tạo ra đặc tính tốt trong 1 vùng giới hạn của biến vào (hình 2.21). Các bộ điều chỉnh này có chung thông tin ở đầu vào và sự tác động của chúng phụ thuộc vào giá trị đầu vào. Trong trường hợp này, luật chuyển đổi có thể viết theo hệ mờ như sau:

Nếu (trạng thái của hệ) là E_i thì (tín hiệu điều khiển) = u_i

Trong đó $i = 1, 2, \dots, n$; E_i là biến ngôn ngữ của tín hiệu vào, u_i là các hàm với các tham số của tác động điều khiển. Nếu tại mỗi vùng điều chỉnh, tác động điều khiển là do bộ điều chỉnh PID_i với:

$$u_i = K_{p_i}e + K_{i_i} \int_0^t e(t)dt + K_{d_i} \frac{de}{dt} \quad (i = 1, 2, \dots, n). \quad (2.10)$$

Như vậy, các hệ số của bộ điều chỉnh PID_i mới phụ thuộc các tín hiệu đầu vào tổng quát hơn là phụ thuộc vào trạng thái của hệ. Nếu coi các hệ số K_{p_i} , K_{d_i} và K_{i_i} chính là kết quả giải mờ theo phương pháp trung bình trọng tâm từ ba hệ mờ hàm:

☞ Hệ mờ hàm tính hệ số K_p với hệ luật:

$$Ru(i): \text{if } E \text{ is } E_i \text{ and } DE \text{ is } DE_i \text{ then } K_p = K_{p_i}. \quad (2.11)$$

☞ Hệ mờ hàm tính hệ số K_D với hệ luật:

$$Ru(i): \text{if } E \text{ is } E_i \text{ and } DE \text{ is } DE_i \text{ then } K_D = K_{D_i}. \quad (2.12)$$

☞ Hệ mờ hàm tính hệ số K_I với hệ luật:

$$Ru(i): \text{if } E \text{ is } E_i \text{ and } DE \text{ is } DE_i \text{ then } K_I = K_{I_i}. \quad (2.13)$$

2.6. HỆ ĐIỀU KHIỂN THÍCH NGHI MỜ

2.6.1. Khái niệm

a/ Định nghĩa: Hệ điều khiển thích nghi mờ là hệ điều khiển thích nghi được xây dựng trên cơ sở của hệ mờ

So với hệ điều khiển thích nghi kinh điển, hệ điều khiển thích nghi mờ có miền tham số chỉnh định rất lớn. Bên cạnh các tham số K_p , T_I , T_D giống như bộ điều khiển PID thông thường, ở bộ điều khiển mờ ta còn có thể chỉnh định các tham số khác như hàm liên thuộc, các luật hợp thành, các phép toán OR, AND, NOT, nguyên lý giải mờ v.v...

Trong thực tế, hệ điều khiển thích nghi được sử dụng ngày càng nhiều vì nó có các ưu điểm nổi bật so với hệ thông thường. Với khả năng tự chỉnh định lại các tham số của bộ điều chỉnh cho phù hợp với đối tượng chưa biết rõ đã đưa hệ thích nghi mờ trở thành một hệ điều khiển thông minh.

b/ Phân loại

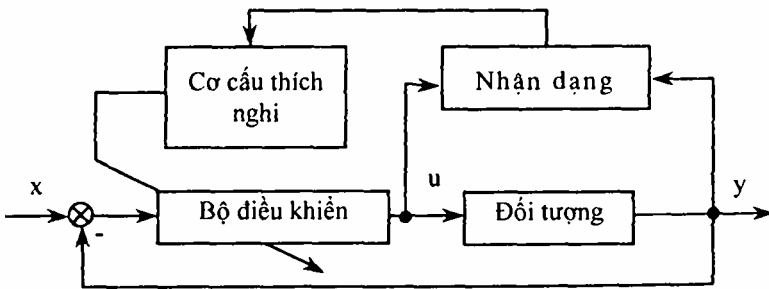
Một cách tổng quát, hệ điều khiển thích nghi mờ có thể phân thành 2 loại:

- Bộ Điều khiển mờ tự chỉnh là bộ điều khiển mờ có khả năng chỉnh định các tham số của các tập mờ (các hàm liên thuộc);

- Bộ điều khiển mờ tự thay đổi cấu trúc là bộ điều khiển mờ có khả năng chỉnh định lại các luật điều khiển. Đối với loại này hệ thống có thể bắt đầu làm việc với một vài luật điều khiển đã được chỉnh định trước hoặc chưa đủ các luật.

c/ Các phương pháp điều khiển thích nghi mờ

Các bộ điều khiển thích nghi rõ và mờ đều có mạch vòng thích nghi được xây dựng trên cơ sở của 2 phương pháp:



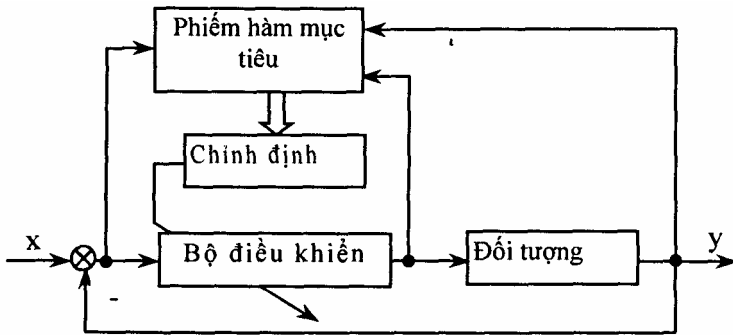
Hình 2.22. Cấu trúc phương pháp điều khiển thích nghi trực tiếp

☞ Phương pháp trực tiếp (hình 2.22) thực hiện thông qua việc nhận dạng thường xuyên các tham số của đối tượng trong hệ kín. Quá trình nhận dạng

thông số của đối tượng có thể thực hiện bằng cách thường xuyên đo trạng thái của các tín hiệu vào/ra của đối tượng và chọn 1 thuật toán nhận dạng hợp lý, trên cơ sở mô hình đối tượng đã biết trước hoặc mô hình mờ;

☞ Phương pháp gián tiếp (hình 2.23) thực hiện thông qua phiếm hàm mục tiêu của hệ kín xây dựng trên các chỉ tiêu chất lượng.

Phiếm hàm mục tiêu có thể được xây dựng trên cơ sở các chỉ tiêu chất lượng động của hệ thống như độ quá điều chỉnh, thời gian quá độ hay các chỉ tiêu tích phân sai lệch... Bộ điều khiển thích nghi mờ có thể chia thành 2 loại:



Hình 2.23. Cấu trúc phương pháp điều khiển thích nghi gián tiếp

2.6.2. Tổng hợp bộ điều khiển thích nghi mờ ổn định

a. Cơ sở lý thuyết

Xét 1 hệ phi tuyến SISO được mô tả bởi phương trình:

$$y^{(n)} = f(y, y', \dots, y^{(n-1)}) + bu; \quad y = x \text{ là biến trạng thái.}$$

$$y_{(n)} = f(y) + bu \quad (2.14)$$

Trong đó u là đầu vào, y là đầu ra, hàm phi tuyến $f(\cdot)$ và hằng số b được giả thiết chưa biết, $y = [y, y', \dots, y^{(n-1)}]^T$. Mục tiêu là thiết kế bộ điều khiển mờ để tạo ra tín hiệu điều khiển u sao cho tín hiệu ra $y(t)$ của hệ thống bám theo quỹ đạo y_d cho trước nào đó.

Nếu biết trước $f(y)$ và b , ta có thể tổng hợp được bộ điều khiển theo các phương pháp kinh điển [9], [55], bộ điều khiển đó có tín hiệu đầu ra là:

$$u(t) = \frac{1}{b} \left[-f(\underline{y}) + \frac{d^n y_d}{dt^n} + K^T E \right] \quad (2.15)$$

trong đó:

$$K^T = \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{bmatrix}, \quad E = \begin{bmatrix} e \\ \frac{de}{dt} \\ \vdots \\ \frac{d^{n-1}e}{dt^{n-1}} \end{bmatrix}. \quad (2.16)$$

Các hệ số k_1, k_2, \dots, k_n được chọn sao cho tất cả các nghiệm của phương trình: $p^n + k_n p^{n-1} + \dots + k_1 = 0$ nằm ở nửa trái mặt phẳng phức. Tức là các nghiệm p_k có phần thực âm:

$$\text{Re}(pk) < 0. \quad (2.17)$$

Thay (2.15) vào (2.14) ta có:

$$\frac{d^n e}{dt^n} + k_n \frac{d^{n-1} e}{dt^{n-1}} + \dots + k_1 e = 0. \quad (2.18)$$

Do có điều kiện (2.17) nên nghiệm của $e(t)$ chắc chắn thoả mãn điều kiện:

$$\lim_{t \rightarrow \infty} e(t) = 0.$$

Ta thấy rằng bài toán tổng hợp trên chỉ có ý nghĩa khi đã biết chính xác mô hình toán học của hệ thống, hay nói cách khác là trong (2.1) ta đã biết $f(y)$ và b . Điều này không phù hợp với nhiều bài toán thực tế. Vì vậy mục tiêu điều khiển đề ra là phải xác định bộ điều khiển mờ $u = u(\underline{x}, \underline{\theta})$ và luật Điều khiển véctơ tham số $\underline{\theta}$ sao cho thoả mãn các điều kiện sau:

- Hệ kín phải ổn định toàn cục trong phạm vi của các biến $\underline{y}(t), \underline{\theta}(t)$ và $u(\underline{x}, \underline{\theta})$.

Tức là: $|\underline{x}(t)| \leq M_x < \infty; |\underline{\theta}(t)| \leq M_0 < \infty; |u(\underline{x}, \underline{\theta})| \leq M_u < \infty$ với mọi $t \geq 0$. Trong đó M_x, M_0, M_u là các tham số do người thiết kế đặt ra.

- Độ sai lệch $e = y_d - y$ càng nhỏ càng tốt.

Khi $f(\cdot)$ và b đã biết thì ta dễ dàng tổng hợp được bộ điều khiển:

$$\mathbf{u}^* = \frac{1}{\mathbf{b}} \left[-\mathbf{f}(\underline{\mathbf{y}}) + \frac{d^n \mathbf{y}_d}{dt^n} + \mathbf{K}^T \underline{\mathbf{e}} \right] \quad (2.19)$$

Trong đó, \mathbf{u}^* được coi là tối ưu. Nhưng vì $\mathbf{f}(\cdot)$ và \mathbf{b} chưa biết nên \mathbf{u}^* không thể thực hiện được, ta sẽ thiết kế bộ điều khiển mờ để xấp xỉ hoá điều khiển tối ưu này.

Giả thiết bộ điều khiển \mathbf{u} là tổ hợp 2 bộ điều khiển: Bộ điều khiển mờ $u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}})$ và bộ điều khiển giám sát $u_s(\underline{\mathbf{x}})$:

$$\mathbf{u} = u_f + u_s \quad (2.20)$$

Trong đó $u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}})$ là bộ điều khiển mờ được đề cập trong tổng kết 2.1.

Tổng kết 2.1: Xét một hệ logic mờ MISO có n đầu vào $\underline{\mathbf{x}}$ và 1 đầu ra y ($\underline{\mathbf{x}} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ và $y \in \mathbb{R}$). Định nghĩa N_j tập mờ $A_{i_j}^j$ với các hàm liên thuộc $\mu_{A_{i_j}^j}$ bao phủ miền xác định của các biến ngôn ngữ đầu vào ($j = 1, \dots, n$ là số đầu vào). Luật điều khiển $R_u^{i_1 \dots i_n}$ có dạng:

$$\text{if } e_1 = A_{i_1}^1 \text{ and } e_2 = A_{i_2}^2 \text{ and } \dots \text{ and } e_n = A_{i_n}^n \text{ then } u = B_{i_1 \dots i_n} \quad (2.21)$$

trong đó $i_1 = 1, 2, \dots, N_1; \dots; i_n = 1, 2, \dots, N_n$ là số hàm liên thuộc cho mỗi biến đầu vào, $B_{i_1 \dots i_n}$ là tập mờ đầu ra.

Sử dụng luật hợp thành PROD, mờ hoá theo đường singleton và giải mờ bằng phương pháp trung bình trọng tâm, ta thu được bộ điều khiển mờ:

$$u = u(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}}) = \frac{\sum_{i_1=1}^{N_1} \dots \sum_{i_n=1}^{N_n} y_{i_1 \dots i_n} \left[\prod_{j=1}^n \mu_{A_{i_j}^j}(x_j) \right]}{\sum_{i_1=1}^{N_1} \dots \sum_{i_n=1}^{N_n} \left[\prod_{j=1}^n \mu_{A_{i_j}^j}(x_j) \right]}, \quad (2.22)$$

$$\mathbf{u} = \boldsymbol{\theta}^T \boldsymbol{\xi}(\underline{\mathbf{x}}) \quad (2.23)$$

trong đó $\boldsymbol{\xi}(\underline{\mathbf{x}})$ là vectơ hàm mờ cơ sở.

$$\boldsymbol{\xi}(\underline{\mathbf{x}}) = \frac{\prod_{j=1}^n \mu_{A_{i_j}^j}(x_j)}{\sum_{i_1=1}^{N_1} \dots \sum_{i_n=1}^{N_n} \left[\prod_{j=1}^n \mu_{A_{i_j}^j}(x_j) \right]}. \quad (2.24)$$

Thay (2.20) vào (2.14) ta được:

$$\mathbf{y}^{(n)} = \mathbf{f}(\underline{\mathbf{x}}) + \mathbf{b}[u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}}) + u_s(\underline{\mathbf{x}})]. \quad (2.25)$$

Từ (3.29) ta rút ra: $\mathbf{f}(\underline{\mathbf{x}}) = -\mathbf{b}u^* + \frac{d^n y_d}{dt^n} + \mathbf{K}^T \underline{\mathbf{e}}$ thay vào (3.35)

$\mathbf{y}^{(n)} = -\mathbf{b}u^* + y_d^{(n)} + \mathbf{K}^T \underline{\mathbf{e}} + \mathbf{b}[u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}}) + u_s(\underline{\mathbf{x}})]$. Sau khi biến đổi ta được:

$$\underline{\mathbf{e}}^{(n)} = -\mathbf{K}^T \underline{\mathbf{e}} + \mathbf{b} [u^* - u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}}) - u_s(\underline{\mathbf{x}})]. \quad (2.26)$$

Hoặc viết dưới dạng phương trình trạng thái:

$$\dot{\underline{\mathbf{e}}} = \mathbf{A}\underline{\mathbf{e}} + \mathbf{B}[u^* - u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}}) - u_s(\underline{\mathbf{x}})] \quad (2.27)$$

Trong đó:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ & & & \dots & & \\ -k_n & -k_{n-1} & -k_{n-2} & \dots & -k_1 & \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ \mathbf{b} \end{bmatrix}. \quad (2.28)$$

Chọn hàm Lyapunov $V = \frac{1}{2} \underline{\mathbf{e}}^T \mathbf{P} \underline{\mathbf{e}}$. Trong đó \mathbf{P} là ma trận dương đối xứng được xác định từ phương trình Lyapunov:

$$\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A} = -\mathbf{Q} \quad (\mathbf{Q} > 0). \quad (2.29)$$

Đạo hàm V ta được:

$$\dot{V} = \frac{1}{2} \dot{\underline{\mathbf{e}}}^T \mathbf{P} \underline{\mathbf{e}} + \frac{1}{2} \underline{\mathbf{e}}^T \mathbf{P} \dot{\underline{\mathbf{e}}}. \quad (2.30)$$

Thay (2.27), (2.29) vào (2.30) ta được:

$$\begin{aligned} \dot{V} &= -\frac{1}{2} \underline{\mathbf{e}}^T \mathbf{Q} \underline{\mathbf{e}} + \underline{\mathbf{e}}^T \mathbf{P} \mathbf{B} [u^* - u_f(\underline{\mathbf{x}}, \underline{\boldsymbol{\theta}}) - u_s(\underline{\mathbf{x}})] \\ &\leq -\frac{1}{2} \underline{\mathbf{e}}^T \mathbf{Q} \underline{\mathbf{e}} + |\underline{\mathbf{e}}^T \mathbf{P} \mathbf{B}| (|u^*| + |u_f|) - \underline{\mathbf{e}}^T \mathbf{P} \mathbf{B} u_s \end{aligned} \quad (2.31)$$

ta cần phải tìm hàm u_s sao cho $V \leq 0$.

Giả thiết ta xác định được hàm $f^u(\underline{\mathbf{x}})$ và hằng số b_L thỏa mãn điều kiện: $|f(\underline{\mathbf{x}})| \leq f^u(\underline{\mathbf{x}})$ và $0 < b_L < b$ thì hàm điều khiển giám sát $u_s(\underline{\mathbf{x}})$ được xây dựng

như sau:

$$\mathbf{u}_s(\mathbf{x}) = I_1^* \operatorname{sgn}(\mathbf{e}^T \mathbf{P} \mathbf{B} \left[|\mathbf{u}_f| + \frac{1}{b_L} (f^U + |y_d^{(n)}| + |\mathbf{K}^T \mathbf{e}|) \right])$$

Trong đó:

$$I_1^* = \begin{cases} 1 & \text{Khi } V > \bar{V} \\ 0 & \text{khi } V \leq \bar{V} \end{cases}$$

(\bar{V} là 1 hằng số được chọn bởi người thiết kế).

Vì $b > 0$, $\operatorname{sgn}(\mathbf{e}^T \mathbf{P} \mathbf{B})$ có thể xác định, hơn nữa tất cả các thành phần trong (2.32) có thể xác định được, vì vậy bộ điều khiển giám sát \mathbf{u}_s là hoàn toàn xác định. Thay (2.32) và (2.19) vào (2.31) và xét cho trường hợp $I_1^* = 1$ ta có:

$$\begin{aligned} \dot{V} &\leq -\frac{1}{2} \mathbf{e}^T \mathbf{Q} \mathbf{e} + |\mathbf{e}^T \mathbf{P} \mathbf{B}| \left[\frac{1}{b} (|f| + |y_d^{(n)}| + |\mathbf{K}^T \mathbf{e}|) + |\mathbf{u}_f| - |\mathbf{u}_f| \right. \\ &\quad \left. - \frac{1}{b_L} (f^U + |y_d^{(n)}| + |\mathbf{K}^T \mathbf{e}|) \right] \\ &\leq -\frac{1}{2} \mathbf{e}^T \mathbf{Q} \mathbf{e} \leq 0 \end{aligned}$$

vậy sử dụng \mathbf{u}_s theo (2.32) ta luôn nhận được $V \leq \bar{V}$.

Từ (2.32) ta thấy rằng \mathbf{u}_s chỉ xuất hiện khi không thoả mãn điều kiện: $V \leq \bar{V}$.

Do vậy trong khoảng sai số nhỏ (nghĩa là $V \leq \bar{V}$) thì chỉ có bộ điều khiển mờ \mathbf{u}_f làm việc còn bộ điều khiển giám sát không làm việc ($\mathbf{u}_s = 0$). Khi hệ thống có khuynh hướng mất ổn định ($V > \bar{V}$) thì bộ điều khiển giám sát bắt đầu làm việc để hướng cho $V \leq \bar{V}$.

Nếu chọn $I_1^* \equiv 1$ thì từ (2.33) ta cần phải đảm bảo không chỉ giới hạn của véctơ trạng thái mà còn phải đảm bảo cho \mathbf{e} hội tụ về 0. Ta không chọn phương án này vì \mathbf{u}_s thường rất lớn.

Thật vậy, từ (2.33) ta thấy \mathbf{u}_s tỉ lệ với giới hạn trên của f_u mà giới hạn này thường rất lớn. Tín hiệu điều khiển lớn có thể gây phiền phức do có làm tăng

thêm chi phí phụ. Bởi vậy ta chọn u_s làm việc theo kiểu giám sát.

Để tìm luật điều khiển thích nghi véctơ tham số $\underline{\theta}$ ta thay $u_f(\underline{x}, \underline{\theta}) = \underline{\theta} \zeta(\underline{x})$. Đặt $\underline{\theta}^*$ là véctơ tham số tối ưu:

$$\underline{\theta}^* = \arg \min_{|\underline{\theta}| \leq M_\theta} \left[\sup_{|\underline{x}| \leq M_x} |u_c(\underline{x}, \underline{\theta}) - u^*| \right]$$

và đặt $w = u_f(\underline{x}, \underline{\theta}^*) - u^*$ biểu thức (2.27) có thể viết:

$$\begin{aligned} \underline{\dot{e}} &= A\underline{e} + B[u^* - u_f(\underline{x}, \underline{\theta}) - u_s(\underline{x})] \\ &= A\underline{e} + B[u_f(\underline{x}, \underline{\theta}^*) - u_f(\underline{x}, \underline{\theta})] - B u_s(\underline{x}) - B w \\ &= A\underline{e} + B\underline{\varphi}^T \underline{\xi}(\underline{x}) - B u_s - B w \end{aligned} \quad (2.34)$$

trong đó $\underline{\varphi} = \underline{\theta}^* - \underline{\theta}$; $\underline{\xi}(\underline{x})$ là hàm mờ cơ sở.

Chọn hàm Lyapunov dạng:

$$V = \frac{1}{2} \underline{e}^T P \underline{e} + \frac{b}{2\gamma} \underline{\varphi}^T \underline{\varphi}. \quad (2.35)$$

Với γ là một hằng số dương, ta có:

$$\dot{V} = -\frac{1}{2} \underline{e}^T Q \underline{e} + \underline{e}^T P B \left[\underline{\varphi}^T \underline{\xi}(\underline{x}) - u_s - w \right] + \frac{b}{\gamma} \underline{\varphi}^T \dot{\underline{\varphi}}. \quad (2.36)$$

Gọi P_n là cột cuối cùng của ma trận P , từ (2.28) ta có:

$$\underline{e}^T P B = \underline{e}^T P_n b. \quad (2.37)$$

$\underline{e}^T P B = T_0 b$. (2.37) Thay (2.37) vào (2.36) ta được:

$$\dot{V} = -\frac{1}{2} \underline{e}^T Q \underline{e} + \frac{b}{\gamma} \underline{\varphi}^T \left[\underline{\gamma} \underline{e}^T P_n \underline{\xi}(\underline{x}) + \dot{\underline{\varphi}} \right] - \underline{e}^T P B u_s - \underline{e}^T P B w. \quad (2.38)$$

Chọn luật thích nghi:

$$\underline{\hat{\theta}} = \gamma \underline{e}^T \underline{P}_n \underline{\xi}(\underline{x}) \quad (2.39)$$

thì (2.38) trở thành:

$$\dot{V} \leq -\frac{1}{2} \underline{e}^T \underline{Q} \underline{e} - \underline{e}^T \underline{P} \underline{B} \underline{w} \quad (2.40)$$

trong đó: $\underline{e}^T \underline{P} \underline{B} u_s \geq 0$.

trong đó: $\underline{e}^T \underline{P} \underline{B} u_s \geq 0$

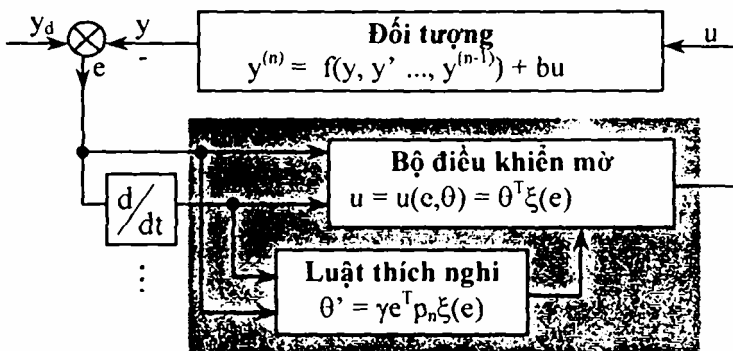
Đây là điều tốt nhất ta có thể đạt được.

b) Thuật toán tổng hợp bộ điều khiển mờ thích nghi

Để tổng hợp bộ điều khiển mờ thích nghi, ta có thể tiến hành theo 2 bước: Bước 1 là chọn cấu trúc của bộ điều khiển mờ, bước 2 là xác định thích nghi các véctơ tham số.

+ Chọn cấu trúc của bộ điều khiển mờ

Cấu trúc của bộ điều khiển mờ thích nghi như hình 2.24. trong đó đối tượng điều khiển là 1 hệ phi tuyến bất kỳ được mô tả tổng quát bằng biểu thức (2.1). Bộ điều khiển mờ thích nghi có thể có nhiều đầu vào gồm sai lệch và các đạo hàm của chúng. Mục đích của việc thiết kế bộ điều khiển mờ là tạo ra tín hiệu điều khiển u , sao cho quỹ đạo đầu ra của đối tượng (y) bám theo quỹ đạo cho trước (y_d), cho dù có sự thay đổi thông số và cấu trúc của đối tượng.



Hình 2.24: Sơ đồ cấu trúc bộ điều khiển mờ thích nghi

+ Các bước thực hiện thuật toán

Trong trường hợp tổng quát, bộ điều khiển mờ có n đầu vào, thuật toán tổng hợp được tóm tắt theo các bước sau:

- **Bước 1.** Xác định hàm liên thuộc của các biến ngôn ngữ đầu vào.

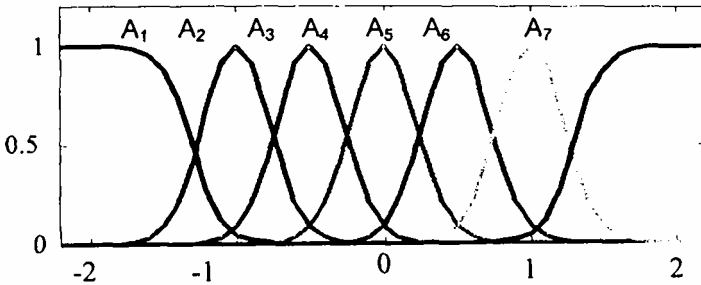
$$\text{Đặt } E = (e_1, e_2, \dots, e_n)^T = (y_d - y, \dot{y}_d - \dot{y} \dots y_d^{(n-1)} - y^{(n-1)})^T.$$

Định nghĩa miền xác định của các thành phần e_j là:

$$\left[\alpha_{\min}^j, \alpha_{\max}^j \right] \quad (j = 1, 2, \dots, n \text{ là số đầu vào}).$$

Chú ý rằng, giá trị thức của e_j có thể ở bên ngoài khoảng $\left[\alpha_{\min}^j, \alpha_{\max}^j \right]$

đã chọn, ở đây $\left[\alpha_{\min}^j, \alpha_{\max}^j \right]$ là khoảng mà e_j rơi vào nhiều nhất.



Hình 2.25. Hàm liên thuộc với 7 tập mờ

Định nghĩa N_j tập mờ $A_{1j} \dots A_{nj}$ trên miền $\left[\alpha_{\min}^j, \alpha_{\max}^j \right]$, hàm liên thuộc

của các tập mờ có thể chọn là hình tam giác, hình thang, hàm Gaus, hàm sigmoid v.v... Chọn hàm liên thuộc kiểu hình tam giác và hình thang có ưu điểm là đơn giản, song có nhược điểm là độ điều chỉnh không trơn. Hình 2.25 là ví dụ về hàm liên thuộc kiểu Gaus ở giữa và kiểu sigmoid ở 2 bên đối với 1 biến ngôn ngữ đầu vào.

$$\mu_{A_i^j}(e_j) = \mu(e_j; \delta_i^j, \alpha_i^j) = 1 - \frac{1}{1 + e^{-\delta_i^j(e_j + \alpha_i^j)}} \quad (2.41)$$

$$\mu_{A_p^j}(e_j) = \mu(e_j; \delta_p^j, \alpha_p^j) = e^{-\delta_p^j(e_j - \alpha_p^j)^2} \quad (2.42)$$

với $p = 2, 3 \dots, N_j - 1$, còn

$$\mu_{A_{N_j}^j}(e_j) = \mu(e_j; \delta_{N_j}^j, \alpha_{N_j}^j) = \frac{1}{1 + e^{-\delta_{N_j}^j(e_j - \alpha_{N_j}^j)}} \quad (2.43)$$

trong đó: $\alpha_{\min}^j = \alpha_1^j < \alpha_2^j < \dots < \alpha_{N-1}^j < \alpha_N^j = \alpha_{\max}^j$.

- **Bước 2.** Xây dựng bộ điều khiển mờ u từ tích $N_1 \dots N_n$ luật sau đây:

Luật $Ru^{i_1 \dots i_n}$

if $e_1 = A_{i_1}^1$ and $e_2 = A_{i_2}^2$ and...and $e_n = A_{i_n}^n$ then $u = B_{i_1 \dots i_n}$ (2.44)

Trong đó $i_1 = 1, 2, \dots, N_1; \dots i_n = 1, 2, \dots, N_n$ là số hàm liên thuộc cho mỗi biến đầu vào

$B_{i_1 \dots i_n}$ là tập mờ đầu ra sẽ được xác định.

Việc thiết kế bộ điều khiển mờ bây giờ chuyển sang việc xác định các thông số $B_{i_1 \dots i_n}$

Sử dụng luật hợp thành PROD, mờ hoá theo đường singleton và giải mờ bằng phương pháp trung bình trọng tâm ta thu được bộ điều khiển mờ:

$$u = u(e, \theta) = \frac{\sum_{i_1=1}^{N_1} \dots \sum_{i_n=1}^{N_n} y_{i_1 \dots i_n} \left[\prod_{j=1}^n \mu_{A_{i_j}^j}(e_j) \right]}{\sum_{i_1=1}^{N_1} \dots \sum_{i_n=1}^{N_n} \left[\prod_{j=1}^n \mu_{A_{i_j}^j}(e_j) \right]} \quad (2.45)$$

$$u = \theta^T \xi(e) \quad (2.46)$$

trong đó: $\zeta(e)$ là tập hợp hàm mờ cơ sở đã biết.

$$\xi(e) = \frac{\prod_{j=1}^n \mu_{A_{i_j}^j}(e_j)}{\sum_{i_1=1}^{N_1} \dots \sum_{i_n=1}^{N_n} \left[\prod_{j=1}^n \mu_{A_{i_j}^j}(e_j) \right]} \quad (2.47)$$

lưu đồ thuật toán tổng hợp hàm mờ cơ sở xe) như hình 2.26.

$y_{i_1 \dots i_n}$ là điểm trọng tâm của $B_{i_1 \dots i_n}$ chúng sẽ được chỉnh định theo luật thích nghi cho phù hợp với đối tượng.

$\underline{\theta}$ là một véctơ gồm tập hợp các $y_{i_1 \dots i_n}$ với $i_1 = 1 \dots N_1; \dots; i_n = 1 \dots N_n$

$$\text{Đặt: } \underline{\theta} = [y_{1 \dots 1}, y_{1 \dots 2}, \dots, y_{2 \dots 1}, \dots, y_{N_1 \dots 1}, \dots, y_{N_1 \dots N_n}]. \quad (2.48)$$

Các thông số e được chỉnh định nhờ sử dụng luật thích nghi sau:

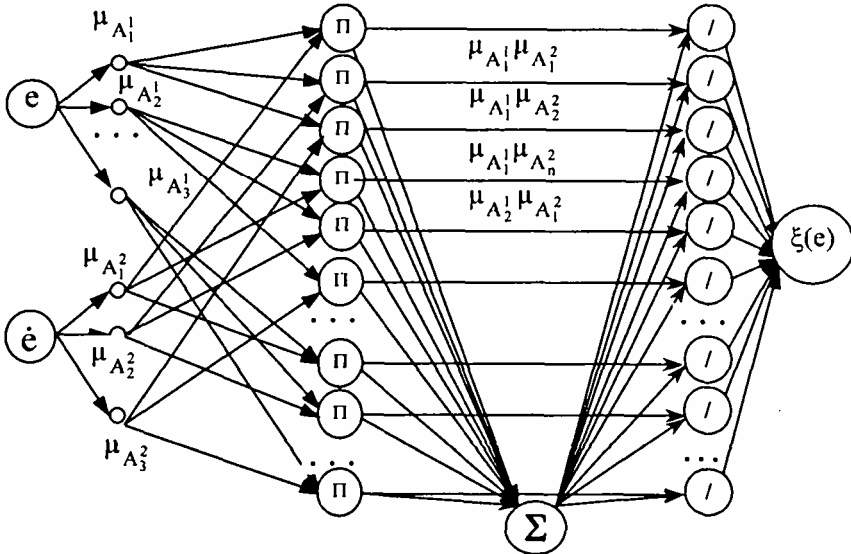
$$\dot{\underline{\theta}} = \gamma e^T P_n \xi(e) \quad (2.49)$$

Trong đó γ là 1 hằng số dương xác định tốc độ của thuật toán còn p_n là cột cuối cùng của ma trận P , với P là nghiệm của phương trình Lyapunov.

$$A^T P + p a = -Q \quad (2.50)$$

trong đó Q là ma trận dương xác định tùy ý, A là ma trận $(n \times n)$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ & & & & \dots & \\ -k_n & -k_{n-1} & -k_{n-2} & & \dots & -k_1 \end{bmatrix}. \quad (2.51)$$



Hình 2.26. Lưu đồ thuật toán tổng hợp hàm mờ cơ sở $\zeta(e)$

với các hằng số k_1, k_2, \dots được chọn sao cho tất cả các nghiệm của phương trình: $P_n + k_n P^{n-1} + \dots + k_1 = 0$ nằm bên nửa trái mặt phẳng phức. Với cách tổng hợp như vậy hệ thống chắc chắn thỏa mãn điều kiện $\lim_{t \rightarrow \infty} e(t) = 0$.

$t \rightarrow \infty$

Từ các tập mờ đầu vào (2.41)... (2.43) và các thông số γ . P_n được xác định ở trên ta tiến hành xây dựng bộ điều khiển mờ theo trình tự sau:

- Định nghĩa các hàm liên thuộc (2.41)... (2.43).
- Xây dựng hàm mờ cơ sở sơ (2.47).
- Xác định luật thích nghi $\underline{\theta} = \gamma \mathbf{e}^T \mathbf{P}_n \xi(\mathbf{e})$
- Xây dựng bộ điều khiển (2.46).

Chú ý:

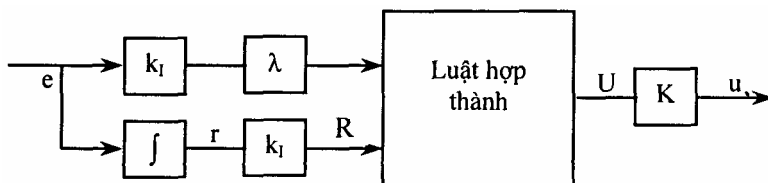
- Hệ số y trong (2.49) nói lên tốc độ hội tụ của thuật toán thích nghi. Nó được chọn và sau đó được kiểm nghiệm thông qua mô phỏng, nếu y chọn quá nhỏ như thuật toán thích nghi hội tụ chậm, y chọn lớn, quá trình hội tụ nhanh nhưng nếu y chọn quá lớn hệ thống sẽ mất ổn định.

- Các giá trị P_1, P_2 được Xác định từ phương trình Lyapunov (2.40), Tuy nhiên độ lớn của nó cũng ảnh hưởng đáng kể đến chất lượng của hệ thống. Vì vậy sau khi thiết kế xong cần chỉnh định lại các giá trị của chúng sao cho đảm bảo chất lượng tốt trong toàn dải thay đổi của các thông số của đối tượng.

2.7. TỔNG HỢP BỘ ĐIỀU KHIỂN MỜ THÍCH NGHI TRÊN CƠ SỞ LÝ THUYẾT THÍCH NGHI KINH ĐIỂN

2.7.1. Đặt vấn đề

Một cấu trúc thông dụng nhất của hệ điều khiển logic mờ (FLC - Fuzzy Logic Control) là cấu trúc kiểu phản hồi sai lệch. Sơ đồ như hình 2.27. Trong đó k_1, λ là các hệ số khuếch đại đầu vào, K là hệ số khuếch đại đầu ra. Thực tiễn cho thấy việc chỉnh định FLC khó khăn hơn nhiều so với chỉnh định bộ điều khiển kinh điển, một trong những lý do chính là tính mềm dẻo của vùng nhận biết cơ bản của bộ điều khiển mờ và sự móc nối các thông số của chúng. Tuy nhiên không có một cách hệ thống hoá nào để đưa ra tất cả những thông số này.



Hình 2.27. Cấu trúc cơ bản của bộ điều khiển mờ 2 đầu vào

Hiện nay trong công nghiệp các bộ điều khiển logic mờ (FLC) thường được thiết kế theo kinh nghiệm và sự hiểu biết định tính đối tượng của các chuyên gia. Việc chỉnh định FLC được thực hiện thông qua chỉnh định các hàm liên thuộc đầu vào và đầu ra và mang nhiều tính chất "mờ mẫm". Do đó không phù hợp cho việc chuẩn hoá chất lượng và khó trở thành một phương pháp luận có hệ thống. Trong mục này chúng ta sẽ tiếp cận kiểu thiết kế hỗn hợp theo hướng kết hợp cả hai cách tiếp cận định tính và tiếp cận định lượng. Đầu tiên ta xây dựng mô hình cơ bản của bộ điều khiển mờ bao gồm các hàm liên thuộc và các luật hợp thành, chúng có thể tạo ra một đáp ứng hợp lý ở một mức độ nào đó. Luật hợp thành cơ bản được chọn là một luật hợp thành tuyến tính, còn hàm liên thuộc có thể được xác định theo hình tam giác, hình thang hoặc hàm Gaus. Sau khi xác định được hàm liên thuộc và luật hợp thành cơ bản, ta sử dụng chúng để tìm ra hệ số khuếch đại tỷ lệ. Có thể sử dụng nhiều phương pháp định lượng khác nhau, việc xác định các hệ số khuếch đại tỷ lệ đúng rất quan trọng đối với sự hoạt động của FLC.

Trong điều khiển kinh điển, ta đã biết một Algorithm điều khiển thích nghi theo mô hình mẫu sử dụng phương pháp gradient hay phương pháp Lyapunov rất thích hợp cho việc điều khiển một quá trình không nhận biết được, đặc biệt đối với hệ phi tuyến. Một bộ điều khiển mờ với một luật hợp thành tuyến tính và các hàm liên thuộc tam giác có thể xấp xỉ tuyến tính xung quanh trạng thái cân bằng. Do đó ta sử dụng ý tưởng của bộ điều khiển thích nghi kinh điển để áp dụng cho bộ điều khiển mờ thích nghi với một vài sự xấp xỉ nào đó. Mục tiêu chính của mục này là:

- ☛ Tìm ra cách tiếp cận định lượng để xác định mô hình toán học của bộ điều khiển mờ với một vài sự xấp xỉ nào đó.

- ☛ Xây dựng bộ điều khiển mờ thích nghi cho những hệ thống phi tuyến và hệ thống biến đổi theo thời gian trên cơ sở lý thuyết thích nghi kinh điển. Bộ điều khiển này có thể sử dụng để điều khiển đối tượng như là bộ thích

nghi trực tuyến, hoặc dùng làm cơ sở cho việc tổng hợp bộ điều khiển mờ thông thường.

Để đơn giản ta tiến hành xây dựng cơ chế thích nghi cho bộ điều khiển mờ hai đầu vào từ kết quả đó có thể dễ dàng mở rộng cho những bộ điều khiển mờ có nhiều đầu vào khác. Cấu trúc của các bộ điều khiển mờ thích nghi dựa trên cơ sở lý thuyết Lyapunov và phương pháp Gradient kinh điển.

2.7.2. Mô hình toán học của bộ điều khiển mờ

Xét bộ Điều khiển mờ hai đầu vào như hình 2.27. Để xây dựng mô hình toán học của nó ta thực hiện theo các bước sau:

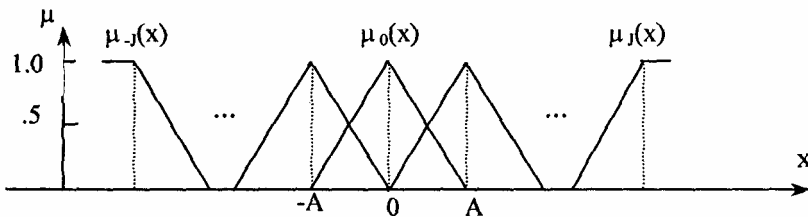
a/ Chọn các hàm liên thuộc

Các tập mờ đầu vào được chọn để mờ hoá là E và R. Ta chọn số lượng các tập mờ vào và ra bằng nhau và bằng N, các hàm liên thuộc sơ bộ chọn hình tam giác với mỗi hàm liên thuộc bao phủ không gian trạng thái 2A cho mỗi đầu vào và 2B cho đầu ra. Giả sử chọn j hàm liên thuộc âm cho E, R, U, chọn j hàm liên thuộc dương cho E, R, U và 1 hàm liên thuộc bằng zero cho E, R, U (hình 2.28). Như vậy số lượng các hàm liên thuộc của mỗi biến vào/ra là: $N = 2j + 1$.

Để đơn giản cho việc xây dựng luật hợp thành, thay vì sử dụng các ngôn ngữ như "âm nhiều", "dương nhiều" v.v... ta sử dụng các chỉ số là số, ví dụ

$$\mu_{-1}(x), \mu_{-2}(x), \mu_0(x), \mu_1(x)$$

Ta thấy rằng, mặc dầu sử dụng các hàm liên thuộc giống nhau để mô tả 2 tập mờ đầu vào nhưng thông qua các hệ số k_1 và λ (hình 2.27) chúng thực sự là các hàm liên thuộc khác nhau.



Hình 2.28. Minh họa việc định nghĩa hàm liên thuộc cho các biến đầu vào và đầu ra

b/ Chọn luật điều khiển

Với bộ điều khiển mờ 2 đầu vào, mỗi đầu vào có N tập mờ ta sẽ có N² luật điều khiển miêu tả tất cả các khả năng kết hợp của E_i và R_j Dạng tổng quát của luật hợp thành là:

Nếu E = E_i và R = R_j thì U = u_k Với k = f(i, j)

Định nghĩa 1: Các luật điều khiển của một bộ điều khiển mờ được gọi là tuyến tính nếu f(i,j) là 1 hàm tuyến tính đối với i và j.

Ví dụ: $f = i + j; f = I + j + 1$ vv

Trong đó f(i, j) là quy luật để sinh ra các luật điều khiển. Với các f(i, j) khác nhau sẽ cho các luật điều khiển khác nhau. Việc chọn luật điều khiển có thể coi là một nghệ thuật và phụ thuộc rất nhiều vào kiến thức và kinh nghiệm của các chuyên gia. Trong mục này tác giả đề cập đến việc chuẩn hóa và đơn giản hóa việc chọn luật điều khiển nhằm tạo điều kiện thuận lợi cho người thiết kế hệ điều khiển mờ.

		R						
	E	-3	-2	-1	0	1	2	3
R	3	0	1	2	3	3	3	3
	2	-1	0	1	2	3	3	3
	1	-2	-1	0	1	2	3	3
	0	-3	-2	-1	0	1	2	3
	-1	-3	-3	-2	-1	0	1	2
	-2	-3	-3	-3	-2	-1	0	1
	-3	-3	-3	-3	-3	-2	-1	0

Hình 2.29. Luật hợp thành tuyến tính

Hình 2.29 minh họa luật điều khiển tuyến tính với f(i,j) = i + j cho bộ điều khiển mờ 2 đầu vào 1 đầu ra với 7 hàm liên thuộc cho mỗi biến vào và ra. Bảng 2.1 và Hình 2.30 là quan hệ vào-ra của luật hợp thành tuyến tính.

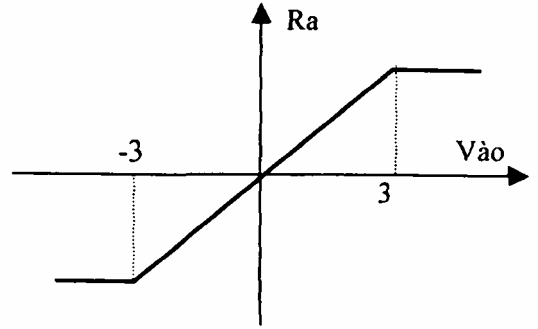
Bảng 2.1

I + j	-3	-2	1	0	1	2	3
U _{k-1}	3	2	-1	0	1	2	3

Định nghĩa 2: Bộ điều khiển mờ cơ sở (Basis Fuzzy Control - BFC) là bộ điều khiển mờ có 2 đầu vào và 1 đầu ra, số tập mờ của các đầu vào và đầu ra bằng nhau, luật hợp thành được sử dụng là luật hợp thành tuyến tính

c/ Phân tích luật cơ sở thành ô suy luận

Các luật cơ sở chia vùng làm việc của bộ điều khiển mờ cơ bản thành nhiều ô vuông, với đầu ra của luật ở trên 4 góc như hình 2.29. Vì tất cả các thao tác mờ đều có thể được tính toán trên các ô này nên chúng được gọi là ô suy luận [33], [55].



Hình 2.28. Quan hệ Vào-ra của luật hợp thành tuyến tính

Một cách tổng quát ta có thể chọn ô suy luận $IC(i, j)$ để phân tích. Ô này được tạo bởi các hàm liên thuộc $\mu_i(E)$, $\mu_{i+1}(E)$, $\mu_j(R)$ và $\mu_{j+1}(R)$ các đường chéo của ô chia chúng ra thành 4 vùng ($IC1... IC4$) (hình 2.3 1).

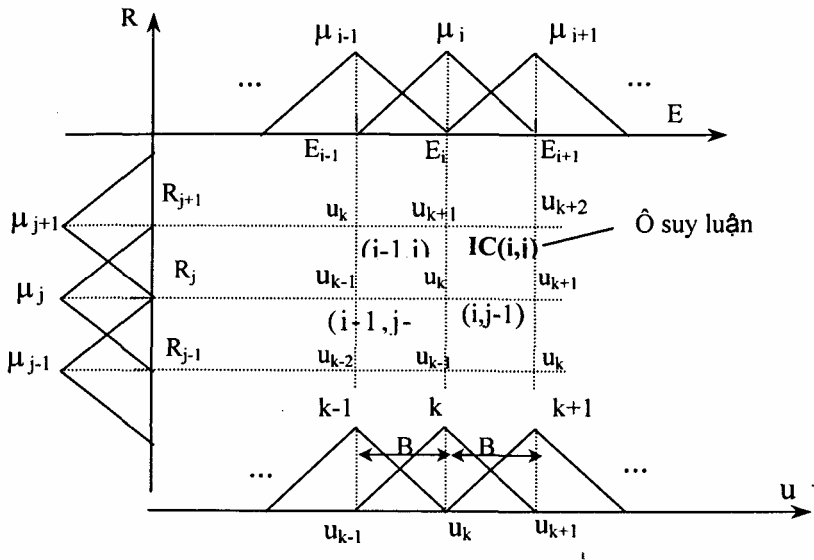
Vị trí tuyệt đối của 1 ô suy luận $IC(i, j)$ trong luật cơ bản là từ $[iA, jA]$ đến $[(i+1)A, (j + 1)A]$, vị trí tương đối của mỗi vùng trong ô $IC(i,j)$ là từ $[0, 0]$ đến $[A, A]$.

Các dữ liệu vào (E, R) trong luật cơ bản luôn luôn được ánh xạ đến dữ liệu vào tương đối (e^*, r^* trong $IC(i, j)$) theo công thức [22]:

$$E = iA + e^* \quad (i = \dots, -1, 0, 1, \dots) \tag{2.52}$$

$$R = jA + r^* \quad (j = \dots, -1, 0, 1, \dots). \tag{2.53}$$

Tất cả những thao tác mờ bao gồm "Mờ hoá", "suy diễn mờ" và "giải mờ" đều có thể được thực hiện trong ô suy luận IC .



Hình 2.29. Sự hình thành ô suy luận từ luật hợp thành

d/ Các thao tác mờ trong ô suy luận

Trong ô suy luận ta có thể thực hiện các thao tác mờ như: Mờ hoá, suy diễn mờ và giải mờ. Sử dụng phương pháp suy luận Max-Min của Mamdani, các thao tác đó được trình bày như sau:

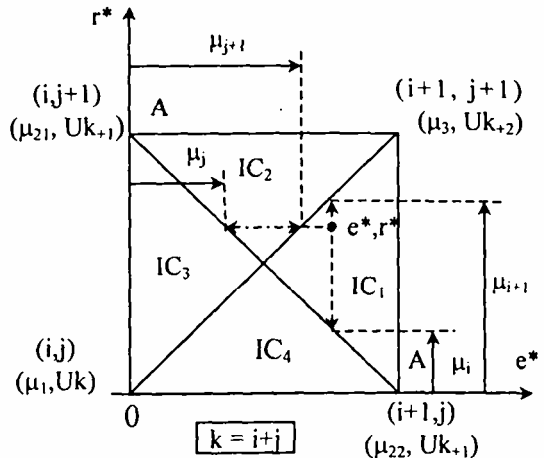
+ Mờ hoá: Từ các biểu thức (2.52) và (2.53) ta thấy trong một ô $IC(i,j)$ các đầu vào (E, R) được xác định bởi (e^*, r^*) với các giá trị hàm liên thuộc của e^* là μ_i và μ_{i+1} , các giá trị hàm liên thuộc của r^* là μ_j và μ_{j+1}

Vì luôn tồn tại quan hệ: $\mu_i + \mu_{i+1} = 1$; μ_j và $\mu_{j+1} = 1$ do đó giá trị các hàm liên thuộc đầu vào trong ô suy luận:

$$\begin{aligned} \mu_i &= 1 - \frac{e^*}{A}; & \mu_{i+1} &= \frac{e^*}{A} \\ \mu_j &= 1 - \frac{r^*}{A}; & \mu_{j+1} &= \frac{r^*}{A} \end{aligned} \quad (2.54)$$

	μ_1	μ_2	μ_3
IC1	μ_i	μ_j	μ_{j+1}
IC2	μ_j	μ_j	μ_{i+1}
IC3	μ_j	μ_{j+1}	μ_{i+1}
IC4	μ_i	μ_{i+1}	μ_{j+1}

Bảng 2.2. Kết quả của phép lấy Max-min trong ô suy luận



Hình 2.30. Các vùng trong ô suy luận

+ Suy diễn mờ

Từ luật hợp thành cơ sở: Nếu $E = E_i$ và $R = R_j$ thì $U = u_k$

với $k = f(i, j) = i + j$. (2.55)

Hàm liên thuộc của các tập mờ đầu ra được biểu diễn trong Hình 2.29 với giá trị đầu ra là:

$$u_k = k.B. \tag{2.56}$$

Tại mỗi vùng của ô suy luận ta thu được các giá trị μ_1, μ_2, μ_3 (bảng 2.2) thông qua phép lấy Max-min [21] với:

$$\left. \begin{aligned} \mu_1 &= \min(\mu_i, \mu_j) \text{ cho đầu ra } u_k \\ \mu_{21} &= \min(\mu_i, \mu_{j+1}) \text{ cho đầu ra } u_{k+1} \\ \mu_{22} &= \min(\mu_{i+1}, \mu_j) \text{ cho đầu ra } u_{k+1} \\ \mu_3 &= \min(\mu_{i+1}, \mu_{j+1}) \text{ cho đầu ra } u_{k+2} \\ \mu_2 &= \text{Max}(\mu_{21}, \mu_{22}) \end{aligned} \right\} \tag{2.57}$$

+ Giải mờ

Dùng phương pháp trung bình trọng tâm [20] ta được tín hiệu ra:

$$u = \frac{\sum_{l=1}^3 \mu_l u_{k+l-l}}{\sum_{l=1}^3 \mu_l} \quad (2.58)$$

trong đó $l = 1, 2, 3, 4$ là các vùng tương ứng của ô suy luận.

e/ Xây dựng biểu thức toán học của bộ điều khiển mờ

Qua các phân tích trên ta thấy rằng các tín hiệu vào khác nhau (e^* , r^*) có thể rơi trên các vùng khác nhau của ô suy luận từ IC1 - IC4, đó là do kết quả của phép lấy Max- min.

+ Xét vùng IC1:

Từ (2.54) và bảng 2.2 ta có:

$$\sum_{l=1}^3 \mu_l = \mu_i + \mu_j + \mu_{j+1} = \mu_{i+1} = 2 - \frac{e^*}{A} = \gamma_1^{-1} \quad (2.59)$$

Từ bảng (2.2), (2.54) và (2.58) ta có:

$$\begin{aligned} \sum_{l=1}^3 \mu_l u_{k+l-1} &= \mu_i u_{k-1} + \mu_j u_k + \mu_{j+1} u_{k+1} \\ &= (1 - e^*/A)(k-1)B + (1 - r^*/A)kB + r^*/A(k+1)B \\ &= B[(k-1)A + e^* + r^*]/A + kB(1 - e^*/A). \end{aligned}$$

Với $S = E + R = (k-1)A + e^* + r^*$ ta có:

$$\sum_{l=1}^3 \mu_l u_{k+l-2} = \frac{B}{A} S + kB(\gamma_1^{-1} - 1)$$

Từ đó ta rút ra:

$$u_1 = \frac{\sum_{l=1}^3 \mu_l u_{k+l-2}}{\sum_{l=1}^3 \mu_l} = \frac{\frac{B}{A} S + kB(\gamma_1^{-1} - 1)}{\gamma_1^{-1}}$$

$$u_1 = \frac{B}{A} \gamma_1 S + kB(1 - \gamma_1)$$

$$u_1 = kB + \frac{B}{A} \gamma_1 (S - kA).$$

Tương tự với các ô suy luận khác, cuối cùng ta thu được [10]:

$$u_i = \frac{B}{A} \gamma_i S + kB(1 - \gamma_i) \quad (i = 1, 2, 3, 4) \quad (2.60)$$

hoặc

$$u_i = B/A \gamma_i(S - kA) + kB \quad (2.61)$$

$$\left. \begin{aligned} \gamma_1 &= (1 + \mu_i)^{-1} = (2 - e^*/A)^{-1} \\ \gamma_2 &= (1 + \mu_j)^{-1} = (2 - e^*/A)^{-1} \\ \gamma_3 &= (1 + \mu_{i+1})^{-1} = (1 + e^*/A)^{-1} \\ \gamma_4 &= (1 + \mu_{j+1})^{-1} = (1 + e^*/A)^{-1} \end{aligned} \right\} \quad (2.62)$$

$$S = E + R = KI(\lambda e + r)$$

$$= (k - 1)A + e^* + r^*, \quad k = i + j$$

γ_i ($i = 1, 2, 3, 4$) là tham số phi tuyến trong vùng IC1.

Ta thấy điều khiển mờ với luật hợp thành tuyến tính thực sự là điều khiển phi tuyến như biểu thức (2.61). Nó sẽ trở thành điều khiển tuyến tính ở trạng thái cân bằng. Trong biểu thức (2.61) ta cần phải xác định các hệ số khuếch đại tỷ lệ đầu vào k_1 , λ và đầu ra K . Giá trị danh định của các hệ số khuếch đại đầu vào k_1 và λ có thể được xác định theo phương pháp của H.X. Li [10]. Thông thường việc xác định hệ số khuếch đại đầu ra K đúng là rất khó khăn.

2.7.3. Xây dựng cơ cấu thích nghi cho bộ điều khiển mờ

a/ Hệ điều khiển thích nghi theo mô hình mẫu (MRAS) dùng lý thuyết thích nghi kinh điển

Xét một đối tượng điều khiển được mô tả bởi phương trình:

$$\frac{dy}{dt} = -ay + bu. \quad (2.63)$$

Mô hình mẫu có phương trình:

$$\frac{dy_m}{dt} = -a_m y_m + b_m u_c. \quad (2.64)$$

Tín hiệu điều khiển: $u = \theta_1 u_c - \theta_2 y$ với sai số: $\varepsilon = y - y_m$

Biểu thức ε chứa tham số điều chỉnh. Ta cần tìm ra cơ cấu thích nghi để điều chỉnh các tham số θ_1 và θ_2 tới giá trị mong muốn sao cho sai số ε tiến tới 0. Để tìm ra cơ cấu thích nghi này ta có thể dùng lý thuyết ổn định Lyapunov

hoặc phương pháp Gradient theo các bổ đề sau:

Bổ đề 2.1: (luật thích nghi theo Lyapunov)

Giả thiết $b\eta > 0$ và chọn hàm Lyapunov có dạng:

$$V(\varepsilon, \theta_1, \theta_2) = \frac{1}{2} \left[\varepsilon^2 + \frac{1}{b\eta} (b\theta_2 + a - a_m)^2 + \frac{1}{b\eta} (b\theta_1 - b_m)^2 \right]$$

thì quy luật điều chỉnh các tham số θ_1, θ_2 để cho $\varepsilon \rightarrow 0$ là:

$$\frac{d\theta_1}{dt} = -\eta u_c \varepsilon; \quad \frac{d\theta_2}{dt} = \eta y \varepsilon. \quad (2.65)$$

Nếu chỉ có 1 tham số biến thiên, luật điều chỉnh thích nghi tham số trở thành:

$$\frac{d\theta_1}{dt} = -\eta u_c \varepsilon. \quad (2.66)$$

Bổ đề 2.2: (Luật thích nghi theo Gradient)

Giả thiết $\underline{\theta}$ là một vectơ tham số cần được xác định, và phụ thuộc độ sai lệch giữa đầu ra của đối tượng (y) và đầu ra của mô hình (y_m). Tiêu chuẩn sai lệch đáp ứng của hệ được chọn:

$$J(\theta) = \frac{1}{2} \varepsilon^2 \rightarrow 0. \quad (2.67)$$

thì quy luật điều chỉnh θ theo hướng của gradient của J là:

$$\frac{d\underline{\theta}}{dt} = -\eta \frac{\partial J}{\partial \underline{\theta}} = -\eta \varepsilon \frac{\partial \varepsilon}{\partial \underline{\theta}} = -\eta \varepsilon \frac{\partial y}{\partial \underline{\theta}}. \quad (2.68)$$

Trong điều khiển thích nghi kinh điển, nói chung không cần một mô hình mẫu hoàn hảo, tuy nhiên sự sai khác giữa mô hình và đối tượng cũng như tính phi tuyến của nó chỉ nằm trong giới hạn nào đó, nếu quá giới hạn này bộ điều chỉnh sẽ không làm việc hiệu quả nữa. Để khắc phục nhược điểm đó, trong cuốn sách này các tác giả đề xuất sử dụng hệ điều khiển mờ thích nghi theo mô hình.

b/ Điều chỉnh thích nghi hệ số khuếch đại đầu ra của bộ điều khiển mờ

Tín hiệu đầu ra của bộ điều khiển mờ (2.60) được viết:

$$U_1 = \frac{B}{A} \gamma_1 S + kB(1 - \gamma_1), \quad \text{với } \gamma \text{ là thông số phi tuyến.}$$

Thay $S = E + R = KI(\lambda + I)e$ (với $I = \int dt$) ta có:

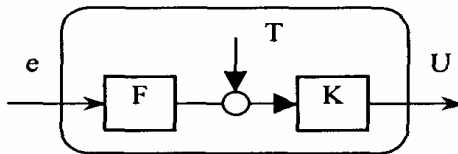
$$U_1 = \frac{B}{A} \gamma_1 K_1 (\lambda + I)e + kB(1 - \gamma_1) = F \cdot e + T$$

trong đó:
$$F = \frac{B}{A} \gamma_1 S = \frac{B}{A} \gamma_1 K_1 (\lambda + I)$$

$$T = kB(1 - \gamma_1).$$

Bộ điều khiển mờ 2 đầu vào trong biểu thức (2.60) với hệ số khuếch đại đầu ra K , có thể được biểu diễn như là $F \cdot e$ cộng thêm 1 giới hạn trễ T như biểu thức (2.69) (hình 2.31) giới hạn trễ T sẽ tiến tới zero khi hệ thống tiến đến điểm cân bằng [11], [12].

$$U = K(T + F \cdot e) \tag{2.69}$$



Hình 2.31. Sơ đồ khối bộ điều khiển mờ với hệ số khuếch đại đầu ra K

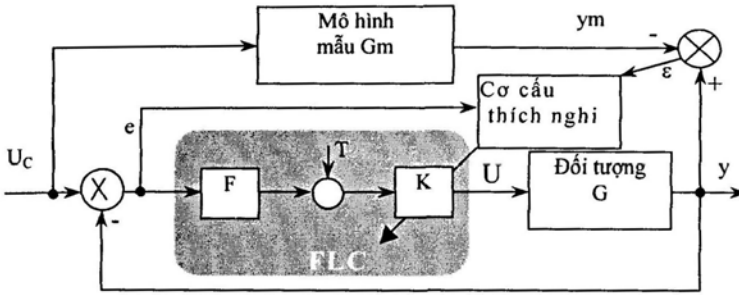
Ta sẽ áp dụng phương pháp Lyapunov và phương pháp Gradient để chỉnh định thích nghi hệ số khuếch đại đầu ra K của bộ điều khiển mờ. Quá trình điều chỉnh được thực hiện theo 2 cấu trúc chính được gọi chung là điều khiển thích nghi mờ theo mô hình mẫu (MRAFC) (Model Reference Adaptive Fuzzy Controller). Ta tiến hành khảo sát 2 sơ đồ là sơ đồ phản hồi đầu ra và sơ đồ điều khiển thích nghi mờ theo mô hình hiệu chỉnh trước (FMRAFC) (Feedforward Model Reference Adaptive Fuzzy Controller).

c/ Sơ đồ điều khiển thích nghi mờ theo mô hình mẫu (MRAFC)

Xét một cấu trúc điều khiển mờ thích nghi theo mô hình được biểu diễn trên hình 2.32 [19], [20].

Trong đó: Đối tượng Điều khiển có hàm số truyền G , mô hình mẫu có hàm truyền G_m , bộ điều khiển mờ bao gồm bộ điều khiển mờ cơ bản kết hợp với bộ khuếch đại K . Cần phải tìm ra quy luật chỉnh định hệ số K sao cho sai

lệch giữa mô hình và đối tượng tiến đến 0 ($\epsilon \rightarrow 0$).



Hình 2.32. MRAC điều chỉnh hệ số khuếch đại đầu ra

Xấp xỉ γ_1 trong (2.61) thành một hằng số, hệ thống vòng kín xung quanh trạng thái cân bằng trở thành tuyến tính với phương trình của vòng kín là:

$$y = \frac{KFG}{1 + KFG} \cdot U_c$$

Và

$$\frac{\partial \epsilon}{\partial K} = \frac{\partial y}{\partial K} = \frac{KFG}{1 + KFG} \cdot \frac{e}{K} \approx G_m \cdot \frac{e}{K}. \quad (2.70)$$

Giả thiết y tiến đến y_m thì ta có thể xấp xỉ $\frac{KFG}{1 + KFG} \approx G_m$. Khi đó quy luật điều chỉnh thích nghi cho hệ số khuếch đại đầu ra của FLC có thể xác định từ (2.68) là:

$$\frac{dK}{dt} = -\eta G_m \frac{\epsilon e}{K}. \quad (2.71)$$

Để xét ổn định của sơ đồ trên, ta chọn hàm Lyapunov:

$$V = 1/2 \epsilon^2$$

$$\dot{V} = \epsilon \left(\frac{\partial \epsilon}{\partial K} \dot{K} + \frac{\partial \epsilon}{\partial t} \right) = -\eta \left(\frac{\epsilon e}{K} G_m \right)^2 + \epsilon \frac{\partial \epsilon}{\partial t}$$

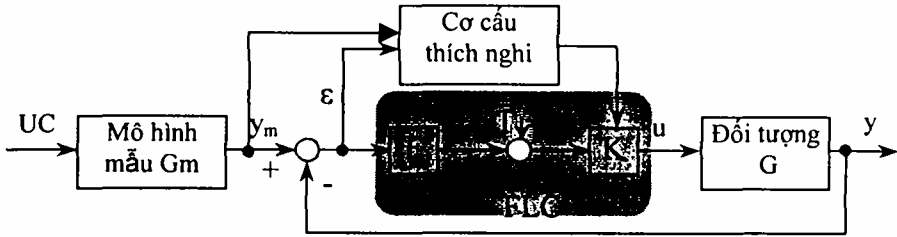
Hệ thống sẽ ổn định $\epsilon \frac{\partial \epsilon}{\partial t} < 0$.

d/ Sơ đồ điều khiển thích nghi mờ kiểu truyền thẳng (FMRAFC)

Một cấu trúc khác của bộ điều khiển thích nghi mờ được biểu diễn trên

hình 2.33 sơ đồ này gọi là sơ đồ thích nghi mờ truyền thẳng (Feedforward Model Reference Adaptive Fuzzy Controller - FMRAFC) [11].

Trong sơ đồ này sai lệch giữa tín hiệu đặt và tín hiệu đầu ra của đối tượng được thay thế bằng giá trị sai lệch giữa đối tượng và mô hình ε



Hình 2.33. Cấu trúc hệ FMRAFC

Hệ số khuếch đại thích nghi đầu ra:
$$y = \frac{KFG}{1 + KFG} y_m$$

$$\frac{\partial \varepsilon}{\partial K} = -\frac{\partial y}{\partial K} = -\frac{KFG}{(1 + KFG)^2} \cdot \frac{y_m}{K} = -\frac{KFG}{1 + KFG} \cdot \frac{\varepsilon}{K} \approx -\frac{\varepsilon}{K} \quad (2.72)$$

trong đó: $\frac{y_m}{1 + KFG} = \varepsilon$ và giả thiết rằng khi y tiến đến y_m thì $KFG/(1 + KFG) \approx 1$.

Từ (2.68) ta rút ra quy luật thích nghi cho hệ số khuếch đại đầu ra là:

$$\frac{dK}{dt} = \eta \frac{\varepsilon^2}{k} \quad (\text{theo gradient}) [21] \quad (2.73)$$

$$\frac{dK}{dt} = \eta y_m u_c \quad (\text{theo Lyapunov}). \quad (2.74)$$

Ta thấy do hàm truyền của mô hình không có mặt ở luật thích nghi (2.73) và (2.74) nên cấu trúc thích nghi này chịu đựng tốt đối với giới hạn lớn độ sai lệch giữa mô hình và đối tượng. Trong thực tế nó chỉ cần một mô hình xấp xỉ gần đúng ví dụ mô hình mẫu bậc nhất: $G_m = \frac{b_m}{a_m + S}$ cũng có thể áp dụng cho phần lớn các đối tượng điều khiển.

2.7.4. Một số ứng dụng điều khiển các đối tượng công nghiệp

Mục đích của phần này là thông qua mô phỏng trình bày tính hiệu quả của bộ điều khiển thích nghi mờ được tổng hợp trên cơ sở lý thuyết điều khiển thích nghi kinh điển. Đồng thời thông qua đó (MRAFC) ta xác định được hệ số khuếch đại đầu ra cho bộ điều khiển mờ, làm cơ sở cho việc xây dựng thuật toán tổng hợp bộ điều khiển mờ. Các ứng dụng được xây dựng cho 3 lớp đối tượng điển hình trong công nghiệp:

☛ Đối tượng tuyến tính bậc hai trong đó có khâu tích phân được mô tả bởi:

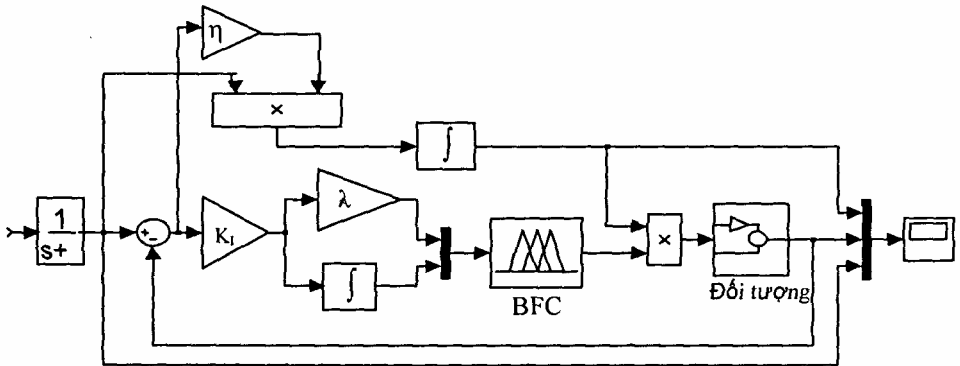
$$G_1(s) = \frac{1}{S} \frac{K}{TS + 1} \quad (2.75)$$

☛ Đối tượng tuyến tính bậc 3 với những tham số không biết, được cho bởi cấu trúc gần đúng sau?

$$G_2(s) = \frac{1}{S+1} \left(\frac{K}{T_1 S^2 + T_2 S + 1} \right) \quad (2.76)$$

☛ Một đối tượng phi tuyến với các thông số biến thiên theo thời gian được mô tả gần đúng bằng phương trình:

$$0.4 \dot{y} + (1 + 0,2 \sin(0,1t))y = 2u - \sin y \quad (2.77)$$



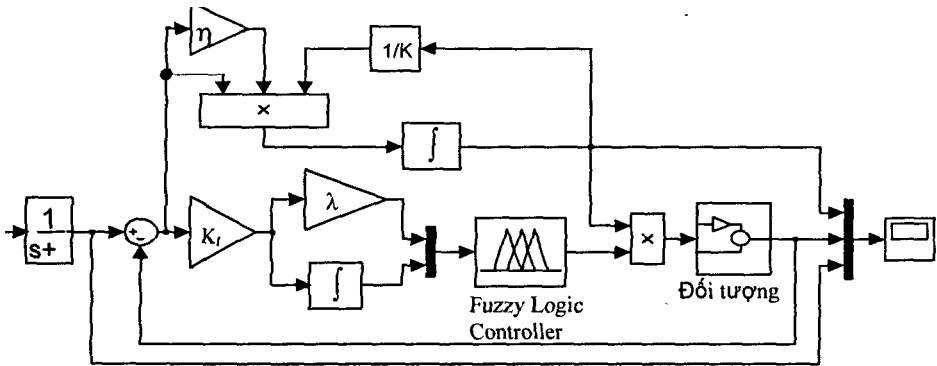
Hình 2.34. Sơ đồ cấu trúc hệ MRAFC với luật điều khiển theo Lyapunov

Mô hình mẫu là khâu quán tính bậc nhất có hàm truyền:

$$G_m = \frac{a_m}{b_m S + 1} = \frac{1}{S + 1}$$

với $a_m = b_m = 1$. Tín hiệu đặt U_C là sóng hình vuông.

Sơ đồ cấu trúc hệ MRAFC với luật điều khiển theo Lyapunov được biểu diễn trên hình 2.34 và theo Gradient được biểu diễn trên hình 2.35.



Hình 2.35. Sơ đồ cấu trúc hệ MRAFC với luật điều khiển theo Gradient

a/ Kết quả mô phỏng

Các kết quả mô phỏng được chỉ ra trên các hình từ hình 2.36 đến hình 2.44.

Để tiện so sánh ta đưa ra đáp ứng tương ứng với 3 cấu trúc MRAC, FMRAFC theo Lyapunov và FMRAFC theo Phương pháp Gradient.

b/ Nhận xét

Từ kết quả mô phỏng ở trên ta rút ra một số nhận xét sau:

- ☛ Đáp ứng của hệ FMRAFC theo phương pháp Lyapunov và phương pháp Gradient gần giống nhau và được biểu diễn trên các hình từ hình 2.36 đến hình 2.41. Ta thấy:

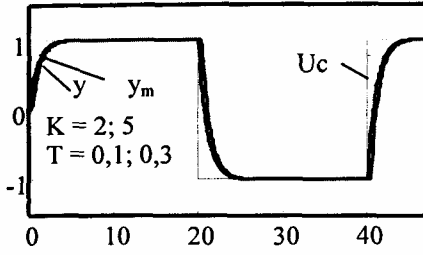
- ☛ Đối với đối tượng tuyến tính bậc hai có khâu tích phân đáp ứng của FMRAFC trong hình 2.36 và hình 2.37 đạt chất lượng động tốt, quá trình làm việc sẽ bám theo mô hình một cách nhanh chóng.

- ☛ Đối với đối tượng tuyến tính bậc 3 đáp ứng của FMRAFC trong hình 2.38 và hình 2.39 gần giống với đối tượng bậc nhất.

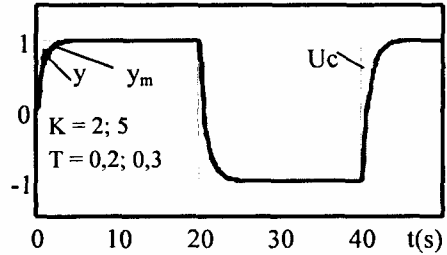
- ☛ Đối với đối tượng không tuyến tính biến đổi theo thời gian, đáp ứng của FMRAFC hình 2.40 và hình 2.41 không thay đổi nhiều so với đối tượng bậc 2.

Vậy hệ điều khiển thích nghi mờ (MRAFC) có thể đạt được đáp ứng tốt hơn rất nhiều so với hệ điều khiển thích nghi kinh điển (MRAC), đặc biệt cho

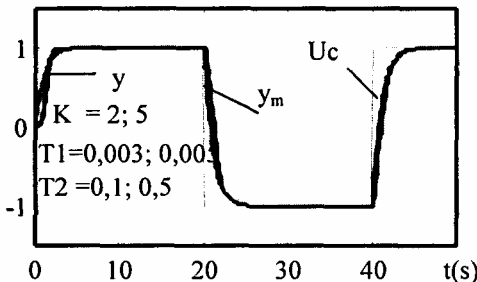
những đối tượng biến đổi theo thời gian và không mô hình hoá được. Bên cạnh đó chỉ ra khả năng to lớn của bộ điều khiển mờ thích nghi làm việc với các quá trình không nhận biết được. Từ những kết quả trên, ta có thể tiếp tục phát triển theo hướng này để xây dựng các bộ điều khiển mờ tự chỉnh trực tuyến mà có thể đạt được đáp ứng tối ưu một cách tự động cho một giới hạn rộng hơn các quá trình.



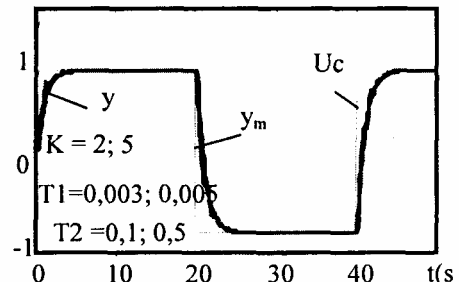
Hình 2.36: Đáp ứng của FMRAFC với lớp đối tượng bậc hai trong đó có khâu tích phân theo Liapunov ứng với 2 giá trị của $K=2; 5$ và $T=0,1; 0,3$



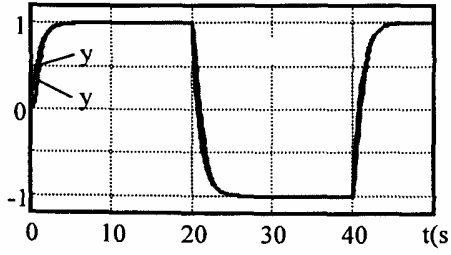
Hình 2.37: Đáp ứng của FMRAFC với lớp đối tượng bậc hai trong đó có khâu tích phân theo Gradient ứng với $K=2; 5$ và $T=0,2; 0,3$



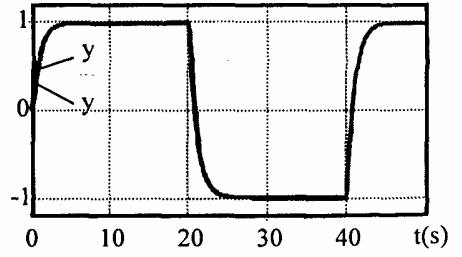
Hình 2.38: Đáp ứng của FMRAFC với lớp đối tượng bậc 3 theo Liapunov ứng với $K=2; 5$; $T_1=0,003; 0,005$ và $T_2=0,1; 0,5$



Hình 2.39: Đáp ứng của FMRAFC với lớp đối tượng bậc 3 theo Gradient ứng với $K=2; 5$; $T_1=0,003; 0,005$ và $T_2=0,1; 0,5$



Hình 2.40: Đáp ứng hệ FMRAFC với đối tượng phi tuyến theo Liapunov



Hình 2.41: Đáp ứng của FMRAFC với đối tượng phi tuyến theo Gradient

Chương 3

TỔNG QUAN VỀ MẠNG NƠN

3.1. NƠN SINH HỌC

3.1.1. Chức năng, tổ chức và hoạt động của bộ não con người

Bộ não người có chức năng hết sức quan trọng trong đời sống của con người. Nó gần như kiểm soát hầu hết mọi hành vi của con người từ các hoạt động cơ bắp đơn giản đến những hoạt động phức tạp như học tập, nhớ, suy luận, tư duy, sáng tạo,...

Bộ não người được hình thành từ sự liên kết của khoảng 10^{11} phần tử (tế bào), trong đó có khoảng 10^{10} phần tử là noron, số còn lại khoảng $9 \cdot 10^{10}$ phần tử là các tế bào thần kinh đệm và chúng có nhiệm vụ phục vụ cũng như hỗ trợ cho các noron. Thông thường một bộ não trung bình cân nặng khoảng 1,5 kg và có thể tích là 235 cm^3 , cho đến nay người ta vẫn chưa thực sự biết rõ cấu tạo chi tiết của bộ não. Tuy vậy về đại thể thì cấu tạo não bộ được phân chia ra thành nhiều vùng khác nhau. Mỗi vùng có thể kiểm soát một hay nhiều hoạt động của con người.

Bộ não có cấu trúc nhiều lớp. Lớp bên ngoài thường thấy như là các nếp nhăn, là lớp có cấu tạo phức tạp nhất. Đây là nơi kiểm soát và phát sinh các hành động phức tạp như nghe, nhìn, tư duy,...

Hoạt động của bộ não nói riêng và của hệ thần kinh nói chung đã được con người quan tâm nghiên cứu từ lâu nhưng cho đến nay người ta vẫn chưa hiểu rõ thực sự về hoạt động của bộ não và hệ thần kinh. Đặc biệt là trong các hoạt động liên quan đến trí óc như suy nghĩ, nhớ, sáng tạo,... Tuy thế cho đến nay, người ta cũng có những hiểu biết căn bản về hoạt động cấp thấp của não.

Mỗi noron liên kết với khoảng 10^4 noron khác, cho nên khi hoạt động thì bộ não hoạt động một cách tổng lực và đạt hiệu quả cao. Nói một cách khác là các phần tử của não hoạt động một cách song song và tương tác hết sức tinh vi phức tạp, hiệu quả hoạt động thường rất cao, nhất là trong các vấn đề phức tạp, về tốc độ xử lý của bộ não người rất nhanh mặc dù tốc độ xử lý của

mỗi noron (có thể xem như phần tử xử lý hay phần tử tính) là rất chậm so với xử lý của các cổng logic silicon trong các chip vi xử lý (10^{-3} giây so với 10^{-10} giây).

Hoạt động của cả hệ thống thần kinh bao gồm não bộ và các giác quan như sau: Trước hết con người bị kích thích bởi giác quan từ bên ngoài hoặc trong cơ thể. Sự kích thích đó được biến thành các xung điện bởi chính các giác quan tiếp nhận kích thích. Những tín hiệu này được chuyển về trung ương thần kinh là não bộ để xử lý. Trong thực tế não bộ liên tục nhận thông tin xử lý, đánh giá và so sánh với thông tin lưu trữ để đưa ra các quyết định thích đáng.

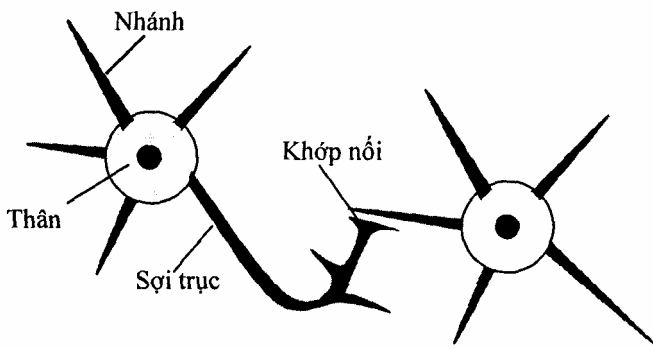
Những mệnh lệnh cần thiết được phát sinh và gửi đến những bộ phận thi hành thích hợp như các cơ tay, chân,... Những bộ phận thi hành biến những xung điện thành dữ liệu xuất của hệ thống.

Tóm lại: bộ não người có chức năng hết sức quan trọng đối với đời sống của con người. Cấu tạo của nó rất phức tạp, tinh vi bởi được tạo thành từ mạng noron có hàng chục tỉ tế bào với mức độ liên kết giữa các noron là rất cao. Hơn nữa, nó còn được chia thành các vùng và các lớp khác nhau. Bộ não hoạt động dựa trên cơ chế hoạt động song song của các nghìn tạo nên nó.

3.1.2. Mạng noron sinh học

a/ Cấu tạo

Noron là phần tử cơ bản tạo nên bộ não con người. Sơ đồ cấu tạo của một noron sinh học được chỉ ra như trong hình 3.1. Một noron điển hình có 3 phần chính:



Hình 3.1. Mô hình 2 noron sinh học

- **Thân nơron** (soma): Nhân của nơron được đặt ở đây.

- **Các nhánh (dendrite)**: Đây chính là các mạng dạng cây của các dây thần kinh để nối các soma với nhau.

- **Sợi trục (Axon)**: Đây là một nối kết, hình trụ dài và mang các tín hiệu từ đó ra ngoài. Phần cuối của axon được chia thành nhiều nhánh nhỏ (cả của dendrite và axon) kết thúc trong một cơ quan nhỏ hình củ hành được gọi là synapse mà tại đây các nơron đưa các tín hiệu của nó vào các nơron khác. Những điểm tiếp nhận với các synapse trên các nơron khác có thể ở các dendrite hay chính soma.

b/ Hoạt động

Các tín hiệu đưa ra bởi một synapse và được nhận bởi các dendrite là các kích thích điện tử. Việc truyền tín hiệu như trên liên quan đến một quá trình hóa học phức tạp mà trong đó các chất truyền đặc trưng được giải phóng từ phía gửi của nơi tiếp nối. Điều này làm tăng hay giảm điện thế bên trong thân của nơron nhận. Nơron nhận tín hiệu sẽ kích hoạt (fire) nếu điện thế vượt khỏi một ngưỡng nào đó và một xung (hoặc điện thế hoạt động) với độ mạnh (cường độ) và thời gian tồn tại cố định được gửi ra ngoài thông qua axon tới phần nhánh của nó rồi tới các chỗ nối synapse với các nơron khác. Sau khi kích hoạt, nơron sẽ chờ trong một khoảng thời gian được gọi là chu kỳ, trước khi nó có thể được kích hoạt lại. Synapses là Hưng phần (excitatory) nếu chúng cho phép các kích thích truyền qua gây ra tình trạng kích hoạt (fire) đối với nơron nhận. Ngược lại, chúng là ức chế (inhibitory) nếu các kích thích truyền qua làm ngăn trở trạng thái kích hoạt (fire) của nơron nhận.

3.2. MẠNG NƠRON NHÂN TẠO

3.2.1. Khái niệm

Nơron nhân tạo là sự sao chép nơron sinh học của não người, nó có những đặc tính sau:

- Mỗi nơron có một số đầu vào, những kết nối (Synaptic) và một đầu ra (axon)

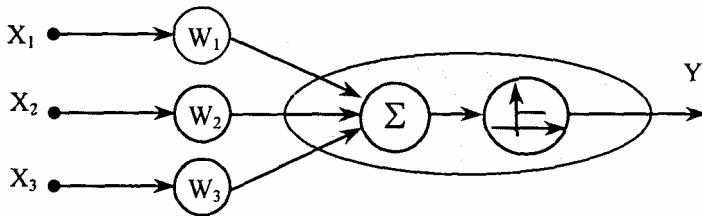
- Một nơron có thể hoạt động (+35 mV) hoặc không hoạt động (-0,75 mV)

- Chỉ có một đầu ra duy nhất của một nơron được nối với các đầu vào khác nhau của nơron khác. Điều kiện để nơron được kích hoạt hay không kích hoạt chỉ phụ thuộc những đầu vào hiện thời của chính nó.

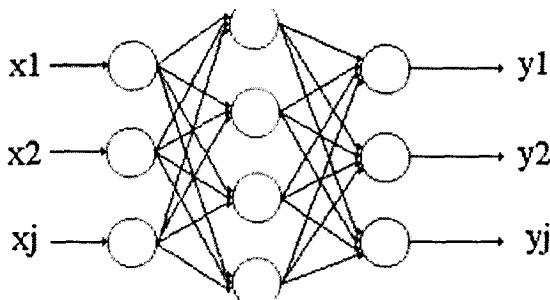
Một nơron trở nên tích cực nếu đầu vào của nó vượt qua ngưỡng ở một mức nhất định..

Có nhiều kiểu nơron nhân tạo khác nhau. Hình 3.2 biểu diễn một kiểu rất đơn giản.

Các đầu vào có hàm trọng W_j và bộ tổng. Đầu ra của bộ tổng được sử dụng để quyết định một giá trị của đầu ra thông qua hàm chuyển. Có nhiều kiểu hàm chuyển khác nhau (sẽ được đề cập ở phần sau). Tương tự nơron sinh học của con người, nơron sẽ được kích hoạt nếu tổng giá trị vào vượt quá ngưỡng và không được kích hoạt nếu tổng giá trị vào thấp hơn ngưỡng. Sự làm việc như vậy của nơron gọi là sự kích hoạt nhảy bậc.



Hình 3.2. Mô hình nơron đơn giản



Hình 3.3. Mạng nơron 3 lớp

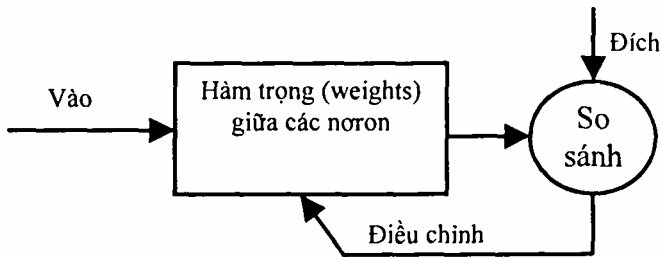
Kết nối một vài nơron ta được mạng nơron. Hình 3.3 là một mạng nơron gồm 3 lớp: lớp vào, lớp ẩn và lớp ra.

Các nơron lớp vào trực tiếp nhận tín hiệu ở đầu vào, ở đó mỗi nơron chỉ có một tín hiệu vào. Mỗi nơron ở lớp ẩn được nối với tất cả các nơron lớp vào và lớp ra. Các nơron ở lớp ra có đầu vào được nối với tất cả các nơron ở

lớp ẩn, chúng là đầu ra của mạng. Cần chú ý rằng một mạng nơron cũng có thể có nhiều lớp ẩn. Các mạng nơron trong mỗi nơron chỉ được liên hệ với tất cả các nơron ở lớp kế tiếp và tất cả các mối liên kết chỉ được xây dựng từ trái sang phải được gọi là mạng nhiều lớp truyền thẳng (perceptrons).

Thông thường mạng nơron được điều chỉnh hoặc được huấn luyện để hướng các đầu vào riêng biệt đến đích ở đầu ra. Cấu trúc huấn luyện mạng được chỉ ra trên hình 3.4. Ở đây, hàm trọng của mạng được điều chỉnh trên cơ sở so sánh đầu ra với đích mong muốn (taget) cho tới khi đầu ra mạng phù hợp với đích. Những cặp vào/đích (input/taget) được dùng để giám sát cho sự huấn luyện mạng.

Để có được một số cặp vào/ra, ở đó mỗi giá trị vào được gửi đến mạng và giá trị ra tương ứng được thực hiện bằng mạng là sự xem xét và so sánh với giá trị mong muốn. Bình thường tồn tại một sai số bởi lẽ giá trị mong muốn không hoàn toàn phù hợp với giá trị thực. Sau một lần chạy, ta có tổng bình phương của tất cả các sai số. Sai số này được sử dụng để xác định các hàm trọng mới.



Hình 3.4. Cấu trúc huấn luyện mạng nơron

Sau mỗi lần chạy, hàm trọng của mạng được sửa đổi với đặc tính tốt hơn tương ứng với đặc tính mong muốn. Từng cặp giá trị vào/ra phải được kiểm tra và trọng lượng được điều chỉnh một vài lần. Sự thay đổi các hàm trọng của mạng được dừng lại nếu tổng các bình phương sai số nhỏ hơn một giá trị đặt trước hoặc đã chạy đủ một số lần chạy xác định (trong trường hợp này mạng có thể không thoả mãn yêu cầu đặt ra do sai lệch còn cao).

Có 2 phương pháp cơ bản để huấn luyện mạng nơron: Huấn luyện gia tăng (tiên dần) và huấn luyện theo gói. Sự huấn luyện theo gói của mạng nhận được bằng việc thay đổi hàm trọng và độ dốc trong một tập (batch) của véctor đầu vào. Huấn luyện tiến dần là thay đổi hàm trọng và độ dốc của

mạng sau mỗi lần xuất hiện của một phần tử véctor đầu vào. Huấn luyện tiến dần đôi khi được xem như huấn luyện trực tuyến hay huấn luyện thích nghi.

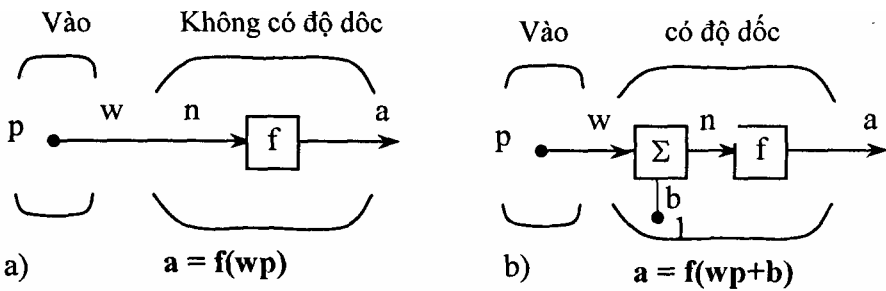
Mạng nơron đã được huấn luyện để thực hiện những hàm phức tạp trong nhiều lĩnh vực ứng dụng khác nhau như trong nhận dạng, phân loại sản phẩm, xử lý tiếng nói, chữ viết và điều khiển hệ thống.

Thông thường để huấn luyện mạng nơron, người ta sử dụng phương pháp huấn luyện có giám sát, nhưng cũng có mạng thu được từ sự huấn luyện không có giám sát. Mạng huấn luyện không giám sát có thể được sử dụng trong trường hợp riêng để xác định nhóm dữ liệu.

Mạng nơron bắt đầu xuất hiện từ 50 năm nhưng mới chỉ tìm thấy các ứng dụng từ khoảng 10 năm trở lại đây và vẫn đang phát triển nhanh chóng. Như vậy, rõ ràng có sự khác biệt với những hệ thống điều khiển hoặc tối ưu hoá, nơi mà các thuật ngữ, cơ sở toán học và thủ tục thiết kế đã được thiết lập chắc chắn và được ứng dụng từ nhiều năm.

3.2.2. Mô hình nơron

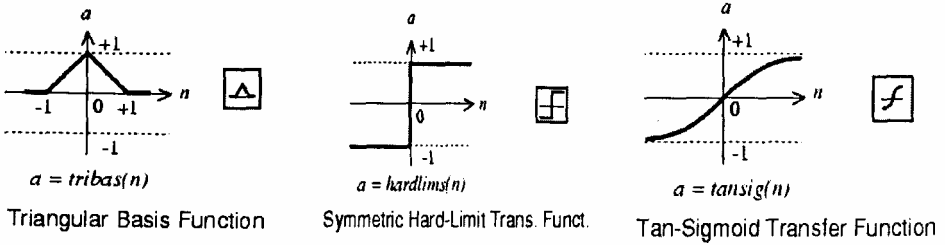
a/ Nơron đơn giản: một nơron với một đầu vào vô hướng và không có độ dốc độc được chỉ ra trên hình 1.5a,b.



Hình 3.5a,b. Mô hình nơron đơn giản

Tín hiệu vào vô hướng p thông qua trọng liên kết vô hướng w trở thành wp cũng là đại lượng vô hướng. Ở đây wp là đối số duy nhất của hàm truyền f , tín hiệu đầu ra là đại lượng vô hướng a . Hình 1.5b là nơron có độ dốc b . Ta có thể hiểu b như là phép cộng đơn giản vào tích wp hoặc như là một sự thăng giáng của hàm f ở hình a đi một lượng b . Độ dốc được xem như một trọng lượng, chỉ có điều đầu vào là một hằng số bằng 1. Tín hiệu vào hàm truyền mạng là n là tổng của trọng đầu vào wp và độ dốc b , đáp ứng ra a

được coi là đối số của hàm chuyển f . Hàm chuyển f có thể là hàm bước nhảy, hàm sigmoid... Hình 3.6 dưới đây giới thiệu một số dạng hàm chuyển của neuron.



Hình 3.6. Một số dạng hàm chuyển của mạng neuron

Chú ý rằng w và b đều là các tham số điều chỉnh vô hướng của neuron. Ý tưởng cơ bản của mạng neuron điều chỉnh các tham số này như thế nào đó để mạng đạt được một đích mong muốn hay một hành vi nào đó. Như vậy ta có thể huấn luyện mạng làm một công việc nào đó bằng cách điều chỉnh các trọng liên kết và độ dốc, hoặc mạng có thể tự điều chỉnh các tham số này để đạt được các kết quả mong muốn.

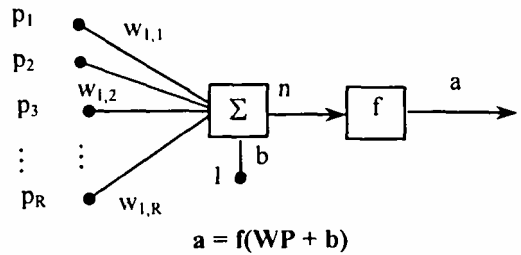
Chú ý:

- Tất cả các neuron đều cho sẵn một độ dốc (b), tuy nhiên chúng ta có thể bỏ đi khi cần thiết.

- Độ dốc b là một tham số điều chỉnh vô hướng của neuron, nó không phải là một đầu vào, song hằng số 1 phải được xem như đầu vào và nó cần được coi như vậy khi xem xét độ phụ thuộc tuyến tính của các véc tơ đầu vào.

b/ Neuron với nhiều đầu vào (véc tơ vào)

Neuron với véc tơ vào gồm R phần tử được chỉ ra trên hình 3.7. Trong đó các đầu vào là p_1, p_2, \dots, p_R được nhân với các trọng liên kết $w_{1,1}, w_{1,2}, \dots, w_{1,R}$ các trọng liên kết được biểu diễn bằng ma trận hàng, véc tơ p là ma trận cột, khi đó ta có:



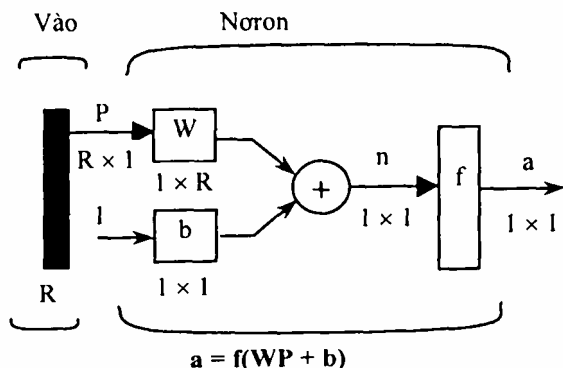
Hình 3.7. Neuron với R đầu vào

$$n = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,3}p_3 + \dots + w_{1,R}p_R + b$$

$$n = W \cdot P + b$$

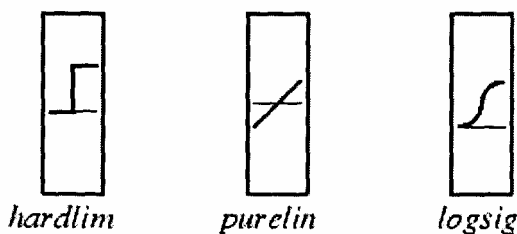
Trong đó W là ma trận trọng liên kết có kích thước $1 \times R$, P là véctor vào gồm R phần tử.

Cách biểu diễn trên sẽ rất khó khăn khi mô tả mạng gồm nhiều nơron và có nhiều lớp. Để đơn giản ta sử dụng ký hiệu như hình 3.8.



Hình 3.8. Ký hiệu nơron với R đầu vào

Trong đó véctor đầu vào được biểu diễn bởi thanh đậm bên trái. Kích thước của p được chỉ ra bên dưới ký hiệu p là $R \times 1$. (ta sử dụng chữ viết hoa R để chỉ kích thước của một véctor). Như vậy p là một véctor gồm R phần tử vào, các đầu vào này nhân với ma trận W ($1 \times R$). Giống như phần trên, ở đây hằng số 1 đưa vào nơron như một đầu vào và được nhân với độ dốc b . Hàm chuyển của mạng là f . Đầu vào hàm chuyển là n bằng tổng của độ dốc b và tích Wp . Tổng này được đi qua hàm chuyển f để có đầu ra của nơron là a . Trong trường hợp này a là một đại lượng vô hướng. Chú ý rằng nếu có từ 2 nơron trở lên thì đầu ra sẽ là một véctor.



Hình 3.9. một số hàm chuyển thông dụng

Một lớp mạng đã được định nghĩa như hình 3.8, đó là sự kết hợp giữa các trọng liên kết, phép nhân, phép cộng, độ dốc b và hàm chuyển f . Trong đó kích thước của ma trận được chỉ rõ ở bên dưới tên biến ma trận của chúng. Khi một hàm chuyển cụ thể được sử dụng thì trên hình vẽ biểu tượng của hàm chuyển đó sẽ thay thế f ở trên. Hình 3.9 là một vài ví dụ về các hàm

chuyển thông dụng.

3.3. CẤU TRÚC MẠNG

Nhiều nơron kết hợp với nhau tạo thành mạng nơron, mạng nơron có thể có một lớp hoặc nhiều lớp.

3.3.1. Mạng một lớp

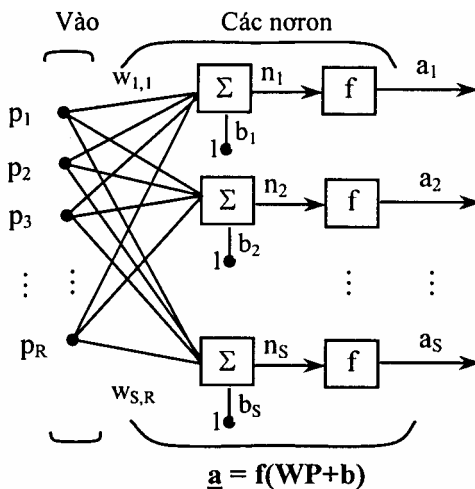
Một cấu trúc mạng 1 lớp với R đầu vào và S nơron được chỉ ra trên hình 3.10.

Trong đó:

- Véc tơ vào \mathbf{p} có R phần tử $\mathbf{p}^T = [p_1 \ p_2 \dots \ p_R]$.
- Véc tơ vào \mathbf{n} có s phần tử $\mathbf{n}^T = [n_1 \ n_2 \dots \ n_s]$.
- Véc tơ vào \mathbf{a} có s phần tử $\mathbf{a}^T = [a_1 \ a_2 \dots \ a_s]$.

Trong mạng này mỗi phần tử của véc tơ vào \mathbf{p} liên hệ với đầu vào mỗi nơron thông qua ma trận trọng liên kết \mathbf{W} . Bộ cộng của nơron thứ i thu thập các trọng liên kết đầu vào và độ dốc để tạo thành một đầu ra vô hướng n_i . Các n_i tập hợp với nhau tạo thành s phần tử của véc tơ vào \mathbf{n} . Cuối cùng ở lớp ra nơron ta thu được véc tơ \mathbf{a} gồm s phần tử.

Chú ý: Nhìn chung số đầu vào của một lớp khác với số nơron, tức là $R \neq S$. Trong một lớp, không bắt buộc phải có số đầu vào bằng số nơron của nó.



Hình 3.10. Cấu trúc mạng nơron 1

Ta có thể thiết lập lớp đơn của các nơron có các hàm chuyển khác nhau một cách dễ dàng bởi lẽ hai mạng được đặt song song. Tất cả các mạng có thể có chung đầu vào và mỗi mạng có thể thiết lập một vài đầu ra.

Các phần tử của vectơ đầu vào được đưa vào mạng thông qua ma trận trọng \mathbf{W} , với:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \cdots & & & \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{bmatrix}$$

Trong đó: Chỉ số hàng trong các phần tử của ma trận \mathbf{W} cho biết nơron nơi đến còn chỉ số cột cho biết nơi xuất phát của trọng liên kết. Ví dụ: w_{12} nói lên sự có mặt của tín hiệu vào từ phần tử thứ hai đến nơron thứ nhất với trọng liên kết là w_{12} .

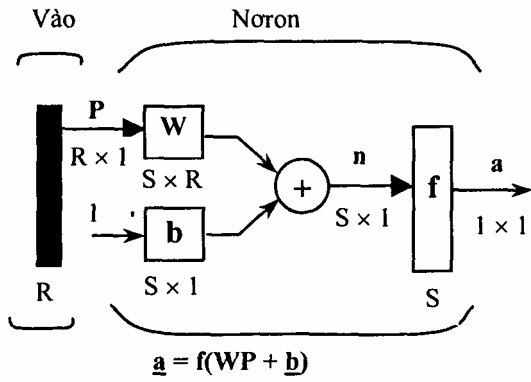
Tương tự như đã trình bày với 1 nơron, để đơn giản ta ký hiệu mạng một lớp gồm S nơron, R đầu vào như hình vẽ 3.11. Trong đó: vectơ vào \mathbf{P} có kích thước R , ma trận trọng liên kết \mathbf{W} có kích thước $S \times R$ còn a và b là các vectơ có kích thước S . Như chúng ta đã biết, một lớp mạng bao gồm ma trận trọng liên kết, toán tử nhân, vectơ độ dốc b , bộ tổng và hộp hàm truyền.

3.3.2. Mạng nhiều lớp

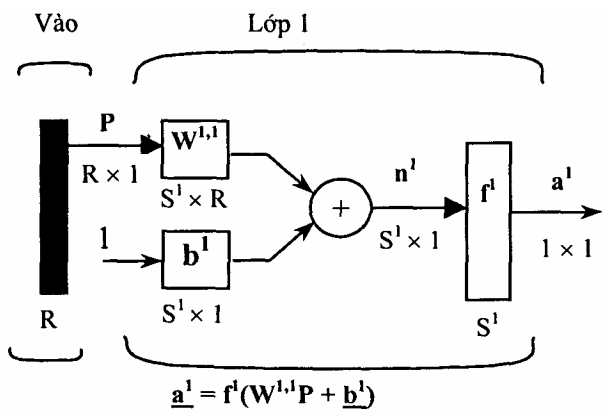
a/ Ký hiệu quy ước cho một lớp mạng

Để khảo sát mạng nhiều lớp trước hết chúng ta cần đưa ra các ký hiệu quy ước cho một lớp mạng. Đặc biệt ta cần phải phân biệt sự khác nhau giữa ma trận trọng liên kết ở đầu vào và các ma trận trọng liên kết giữa các lớp và nắm vững ký hiệu nguồn và đích của ma trận trọng liên kết.

Ta gọi ma trận trọng liên kết nối với đầu vào là các trọng vào (input weights) và các ma trận đến từ lớp ra là trọng liên kết lớp (layer weights). Ta sẽ dùng các chỉ số viết bên trên để phân biệt nguồn (chỉ số thứ hai) và đích (chỉ số thứ nhất) cho các trọng liên kết và các phần tử khác của mạng.



Hình 3.11. Ký hiệu mạng R đầu vào và S nơron



Hình 3.12. Ký hiệu một lớp mạng

Để minh họa, ta xét một lớp mạng có nhiều đầu vào như hình 3.12. Trong đó R là số phần tử lớp vào và S¹ là số nơron của lớp 1. Ta thấy ma trận trọng liên kết với vectơ vào **P** là ma trận trọng vào (**IW^{1,1}**) có nguồn là 1 (chỉ số thứ 2) và đích là 1 (chỉ số thứ nhất). Đồng thời các phần tử của 1 lớp như độ dốc, tín hiệu vào hàm chuyển, đầu ra có chỉ số viết trên là 1 để nói rằng chúng được liên kết với lớp thứ nhất (**b¹**, **n¹**, **a¹**). Ở phần sau ta sẽ sử dụng ma trận trọng liên kết lớp (**LW**) giống như ma trận trọng vào (**IW**).

Với một mạng cụ thể có ma trận trọng **IW^{1,1}** được ký hiệu:

$$\mathbf{IW}^{1,1} \rightarrow \mathbf{net.IW}\{1, 1\}$$

Như vậy, ta có thể viết ký hiệu để thu được mạng nhập vào cho hàm chuyển như sau:

$$\mathbf{net}\{1\} = \mathbf{net.IW}\{1, 1\} * \mathbf{p} + \mathbf{net.b}\{1\};$$

Một mạng nơron có thể có một vài lớp. Mỗi lớp có ma trận trọng liên kết W , véctơ độ dốc b và đầu ra a . Để phân biệt các ma trận trọng liên kết véctơ vào cho mỗi lớp mạng trong sơ đồ, ta thêm con số chỉ lớp viết ở phía trên cho biến số quan tâm.

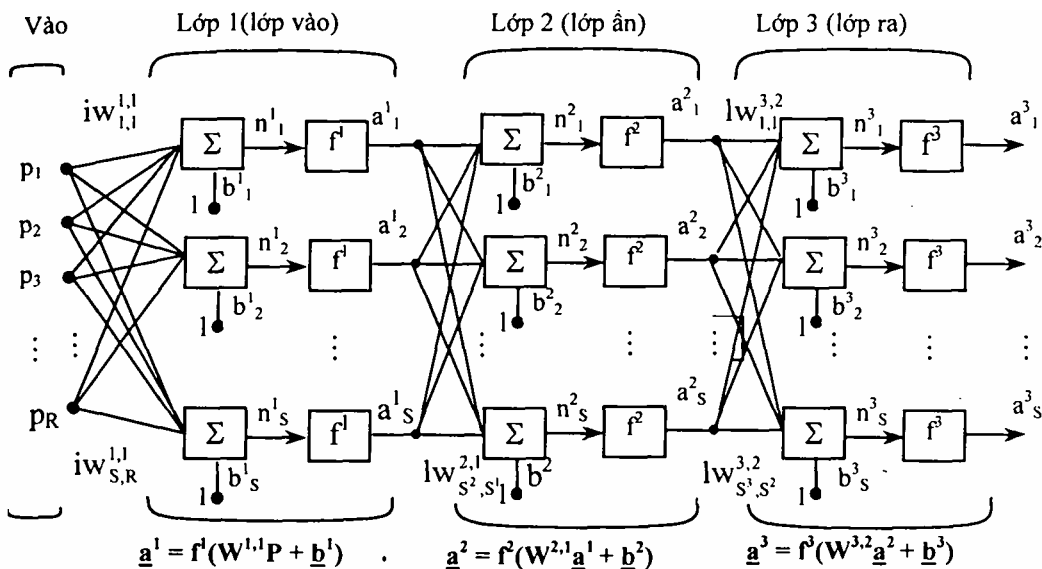
Hình 3.13 là ký hiệu sơ đồ mạng 3 lớp. Trong đó có R^1 đầu vào, S^1 nơron ở lớp 1, S^2 nơron ở lớp 2... Thông thường, các lớp khác nhau có số nơron khác nhau.

Chú ý rằng đầu ra của mỗi lớp trung gian là đầu vào của lớp tiếp theo. Như vậy lớp 2 có thể được xem như mạng 1 lớp với S^1 đầu vào, S^2 nơron và $S^2 \times S^1$ trọng liên kết của ma trận W^2 . Đầu vào của lớp 2 là véctơ a^1 , đầu ra là véctơ a^2 . Khi đã có ký hiệu của tất cả các véctơ và ma trận của lớp 2 ta có thể coi nó như là mạng 1 lớp. Cách tiếp cận này được dùng cho một lớp bất kỳ của mạng. Các lớp của mạng nhiều lớp đóng vai trò khác nhau. Lớp cuối cùng là kết quả ở đầu ra của mạng, được gọi là lớp ra. Tất cả các lớp khác được gọi là lớp ẩn. Mạng 3 lớp ở trên có 1 lớp ra (lớp 3) và 2 lớp ẩn (lớp 1 và lớp 2).

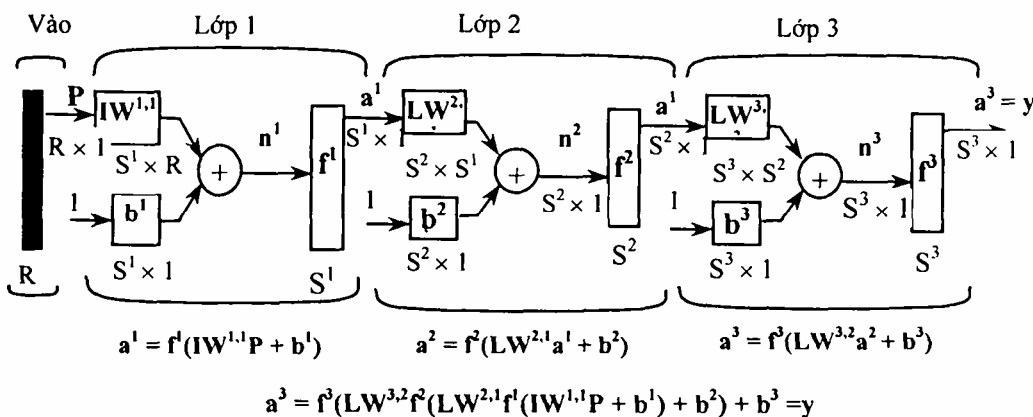
(Một vài tài liệu coi lớp vào như là lớp thứ tư ở đây ta không sử dụng quan điểm này).

Đối với mạng 3 lớp ta cũng có thể sử dụng ký hiệu tắt để biểu diễn (hình 3.14). Mạng nhiều lớp rất mạnh, ví dụ có mạng 2 lớp, trong đó lớp 1 có hàm chuyển sigmoid, lớp 2 có hàm chuyển linear có thể được huấn luyện để làm xấp xỉ một hàm bất kỳ (với số điểm gián đoạn có hạn chế). Loại mạng 2 lớp này sẽ được sử dụng rộng rãi ở chương 5 (mạng lan truyền ngược).

Trong đó a^3 là đầu ra của mạng, ta ký hiệu đầu ra này là y . Ta sẽ sử dụng ký hiệu này để định rõ đầu ra của mạng nhiều lớp.



Hình 3.13. Cấu trúc mạng nơron 3 lớp

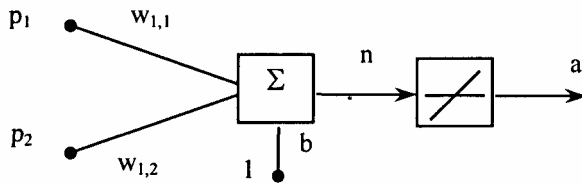


Hình 3.14. Ký hiệu tắt của mạng nơron 3 lớp

3.4. CẤU TRÚC DỮ LIỆU VÀO MẠNG

Để mô phỏng mạng nơron ta cần phải định rõ khuôn dạng của cấu trúc dữ liệu được dùng trong mạng. Dữ liệu đưa vào mạng được biểu diễn dưới 2 dạng cơ bản: một dạng xuất hiện đồng thời (tại cùng một thời điểm hoặc chuỗi thời điểm cụ thể) và một dạng xuất hiện liên tiếp theo thời gian. Đối với vectơ vào đồng thời, ta không cần quan tâm đến thứ tự của các phần tử, kiểu dữ liệu này được áp dụng cho mạng tĩnh. Đối với kiểu vectơ vào nối tiếp thì thứ tự xuất hiện của các phần tử vectơ rất quan trọng, nó được áp dụng

cho mạng động.



$$a = \text{purelin}(WP + b)$$

Hình 3.15. Một noron với 2 đầu vào

3.4.1. Mô tả vectơ vào đối với mạng tĩnh

Đối với mạng tĩnh (không có phản hồi và trễ), ta không cần quan tâm tới việc có hay không vectơ vào xuất hiện trong một chuỗi thời điểm cụ thể, vì vậy ta có thể xem như các đầu vào là đồng thời. Trong phép cộng, ta giải quyết bài toán đơn giản bằng tổng của mạng chỉ có một vectơ vào:

$$n = W_{1,1} * p_1 + W_{1,2} * p_2 + b.$$

Ví dụ: Mạng truyền thẳng có 2 đầu vào (hình 3.15) với các thông số: $W = [1 \ 2]$ và $b = [0]$; tập dữ liệu mô phỏng mạng có 4 vectơ vào đồng thời ($Q = 4$):

$$P_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad P_3 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 3 \\ 1 \end{bmatrix},$$

Các vectơ vào đồng thời được trình bày trong mạng như một ma trận đơn giản:

$$P = [1 \ 2 \ 2 \ 3; \ 2 \ 1 \ 3 \ 1];$$

Sau khi chạy mô phỏng ta thu được các giá trị ở đầu ra

$$a_1 = W_{1,1} * p_1 + W_{1,2} * p_2 + b = 1 * 1 + 2 * 2 + 0 = 5$$

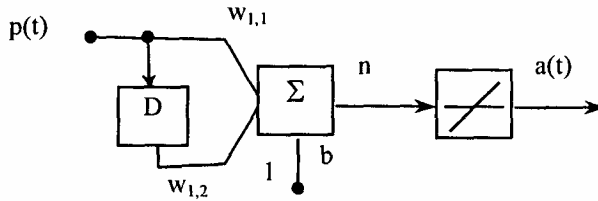
$$a_2 = W_{1,1} * p_1 + W_{1,2} * p_2 + b = 1 * 2 + 2 * 1 + 0 = 4$$

$$a_3 = W_{1,1} * p_1 + W_{1,2} * p_2 + b = 1 * 2 + 2 * 3 + 0 = 8$$

$$a_4 = W_{1,1} * p_1 + W_{1,2} * p_2 + b = 1 * 3 + 2 * 1 + 0 = 5$$

Vậy vectơ vectơ đầu ra là: $A = [5 \ 4 \ 8 \ 5]$.

Một ma trận đơn của véctơ đồng thời được đưa tới mạng và mạng đưa ra một ma trận đơn của véctơ đồng thời ở đầu ra. Kết quả tương tự như 4 mạng làm việc song song, mỗi mạng có một véctơ vào và 1 véctơ ra. Thứ tự của các véctơ vào không quan trọng do chúng không ảnh hưởng lẫn nhau.



$$a(t) = w_{1,1}p(t) + w_{1,2}p(t-1)$$

Hình 3.16. Noron có chứa khâu trễ

3.4.2. Mô tả véctơ vào liên tiếp trong mạng động

Khi mạng có chứa khâu trễ, ở đầu vào mạng thường sẽ có một chuỗi các véctơ vào mà chúng xuất hiện theo thứ tự thời gian nào đó. Để minh họa cho trường hợp này ta sử dụng một mạng đơn bao gồm một khâu trễ (hình 3.16). Ta đưa vào mạng gồm dãy liên tiếp các dữ liệu vào thì mạng sinh ra một mảng bao gồm chuỗi liên tiếp các dữ liệu ra. Chú ý rằng thứ tự của dữ liệu vào rất quan trọng khi chúng được đưa vào như một sự nối tiếp. Trong trường hợp này dữ liệu ra thu được bằng cách nhân dữ liệu vào hiện thời với $w_{1,1}$, dữ liệu vào trước đó với $w_{1,2}$ rồi cộng kết quả lại nếu thay đổi thứ tự các dữ liệu vào nó có thể làm thay đổi những số thu được ở đầu ra.

Ví dụ: Mạng hình 3.16 có các thông số: $W = [1 \ 2]$; $b = 0$; Chuỗi vào nối tiếp là:

$p_1 = [1]$ $p_2 = [2]$, $p_3 = [3]$, $p_4 = [4]$, được biểu diễn dưới dạng mảng:

$$P = \{1 \ 2 \ 3 \ 4\}.$$

Sau khi chạy mô phỏng ta thu được một mảng dữ liệu ra với các phần tử có giá trị:

$$a_1 = W_{1,1} * p_1 + W_{1,2} * p_2 = 1 * 1 + 2 * 0 = 1 \text{ (giá trị đầu vào 2 vẫn là 0)}$$

$$a_2 = W_{1,1} * p_1 + W_{1,2} * p_2 = 1 * 2 + 2 * 1 = 4 \text{ (giá trị đầu vào 2 là 1)}$$

$$a_3 = W_{1,1} * p_1 + W_{1,2} * p_2 = 1 * 3 + 2 * 2 = 7 \text{ (giá trị đầu vào 2 là 2)}$$

$$a_4 = W_{1,1} * p_1 + W_{1,2} * p_2 = 1 * 4 + 2 * 3 + 0 = 10 \text{ (giá trị đầu vào 2 là 3)}$$

$$\mathbf{Vây A} = [1] \quad [4] \quad [7] \quad [10].$$

3.4.3. Mô tả các dữ liệu vào đồng thời trong mạng động

Khi đưa vào mạng động đã xét ở trên một tập các dữ liệu đồng thời thay cho các dữ liệu liên tiếp, ta có thể thu được kết quả khác nhau hoàn toàn. Ví dụ có tập các dữ liệu vào đồng thời:

$$P_1 = [1], P_2 = [2], P_3 = [3], P_4 = [4] \text{ được thiết lập theo mã sau:}$$

$$\mathbf{P} = [1 \ 2 \ 3 \ 4];$$

Sau khi chạy mô phỏng với các dữ liệu vào đồng thời ta thu được:

$$\mathbf{A} = [1 \ 2 \ 3 \ 4].$$

Kết quả này giống như khi ta áp dụng đồng thời mỗi đầu vào tới một mạng riêng biệt và tính toán một đầu ra.

Chú ý: Một khi ta không ấn định bất kỳ điều kiện đầu nào cho mạng có trễ thì chúng được coi bằng zero. Trong trường hợp này đầu ra chỉ đơn giản là 1 nhân với đầu vào vì hàm trọng nhân với đầu vào hiện thời là 1.

Trong trường hợp đặc biệt, ta có thể cần phải mô phỏng đáp ứng của mạng với một vài chuỗi số khác nhau trong cùng một thời gian, ta cần đưa tới mạng với một tập đồng thời của chuỗi. Ví dụ ta cần đưa tới mạng hai dữ liệu liên tiếp sau:

$$p_1(1) = [1], p_1(2) = [2], p_1(3) = [3], p_1(4) = [4]$$

$$p_2(1) = [4], p_2(2) = [3], p_2(3) = [2], p_2(4) = [1].$$

Đầu vào P cần phải là một mảng, trong đó mỗi phần tử của mảng bao gồm 2 phần tử liên tiếp mà chúng xuất hiện cùng một lúc.

$$\mathbf{P} = \{[1 \ 4] \ [2 \ 3] \ [3 \ 2] \ [4 \ 1]\};$$

Chạy mô phỏng mạng:

$$\mathbf{A} = \mathbf{sim}(\mathbf{net}, \mathbf{P});$$

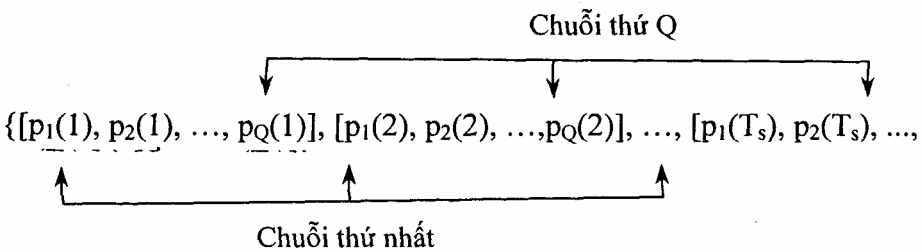
Kết quả đầu ra của mạng sẽ là:

$$\mathbf{A} = \{[1 \ 4] \ [4 \ 11] \ [7 \ 8] \ [10 \ 5]\} = \{[a_{11} \ a_{21}] \ [a_{12} \ a_{22}] \ [a_{13} \ a_{23}] \ [a_{14} \ a_{24}]\} \text{ trong đó:}$$

$$\begin{aligned}
a_{11} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 1 + 2 * 0 + 0 = 1; \\
a_{21} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 4 + 2 * 0 + 0 = 4; \\
a_{12} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 2 + 2 * 1 + 0 = 4; \\
a_{22} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 3 + 2 * 4 + 0 = 11; \\
a_{13} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 3 + 2 * 2 + 0 = 7; \\
a_{23} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 2 + 2 * 3 + 0 = 8; \\
a_{14} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 4 + 2 * 3 + 0 = 7; \\
a_{24} &= W_{1,1}.p_1 + W_{1,2}.p_2 + b = 1 * 1 + 2 * 2 + 0 = 8;
\end{aligned}$$

Ta có thể thấy cột đầu tiên của mỗi ma trận kết quả, chuỗi ra được tạo ra từ chuỗi vào đầu tiên mà chúng ta đã làm quen trong ví dụ trước. Cột thứ hai của mỗi ma trận kết quả chuỗi ra được tạo ra từ chuỗi vào thứ hai. Không có sự tương tác giữa hai chuỗi đồng thời. Nó giống như khi mỗi ứng dụng của các mạng riêng biệt được chạy song song.

Sơ đồ dưới đây chỉ ra khuôn dạng chung của đầu vào P khi ta có Q chuỗi vào đồng thời qua những bước thời gian T_s , nó bao hàm cả trường hợp khi có 1 véctor vào. Mỗi phần tử của mảng là một ma trận của các véctor đồng quy mà nó ứng với cùng một thời điểm cho mỗi chuỗi. Nếu có nhiều véctor vào sẽ có nhiều hàng của ma trận trên mảng.



Trong mục này chúng ta đã áp dụng các dữ liệu vào liên tiếp và đồng thời cho mạng động.

Chú ý: ở mục 3.4.1 ta đã áp dụng dữ liệu vào đồng thời cho mạng tĩnh. Ta cũng có thể áp dụng dữ liệu vào liên tiếp cho mạng tĩnh, nó sẽ không làm thay đổi kết quả mô phỏng của mạng, nhưng nó có thể ảnh hưởng tới cách thức huấn luyện mạng.

3.5. HUẤN LUYỆN MẠNG

Trong phần này, chúng ta đề cập đến 2 kiểu huấn luyện mạng: Huấn luyện gia tăng (tiến dần) và huấn luyện theo gói. Đối với sự huấn luyện gia tăng, hàm trọng và độ dốc của mạng được cập nhật mỗi khi dữ liệu được đưa vào mạng. Đối với sự huấn luyện theo gói, hàm trọng và độ dốc chỉ được cập nhật sau khi tất cả các dữ liệu được đưa vào mạng.

3.5.1. Huấn luyện gia tăng

Sự huấn luyện gia tăng (huấn luyện tiến dần) có thể được áp dụng cho cả mạng tĩnh và mạng động. Tuy nhiên, trong thực tế nó được sử dụng nhiều hơn cho mạng động, ví dụ các bộ lọc thích nghi. Trong mục này, chúng ta sẽ giải thích sự huấn luyện gia tăng được thực hiện như thế nào trên mạng tĩnh và mạng động.

a/ Huấn luyện gia tăng đối với mạng tĩnh

Xét mạng tĩnh học, ta muốn huấn luyện nó gia tăng, sao cho hàm trọng và độ dốc của nó được cập nhật mỗi khi đầu vào có mặt. Trong trường hợp này chúng ta sử dụng hàm "**Adapt**" và ta coi các giá trị đầu vào và đích là các chuỗi nối tiếp.

Giả thiết ta muốn huấn luyện mạng để tạo ra hàm tuyến tính:

$$\mathbf{t} = 2\mathbf{p}_1 + \mathbf{P}_2$$

Các dữ liệu vào ban đầu được sử dụng:

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{p}_4 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

Đích của mạng là: $\mathbf{t}_1 = [4]$ $\mathbf{t}_2 = [5]$ $\mathbf{t}_3 = [7]$ $\mathbf{t}_4 = [7]$

Trước hết ta thiết lập mạng với những hàm trọng và độ dốc ban đầu bằng zero. Ta cũng đặt mức học xuất phát từ zero, để cho thấy hiệu ứng của sự huấn luyện gia tăng.

```
net = newlin([-1 1;-1 1], 1,0,0);
```

```
net.IW{1,1} = [0 0];
```

```
net.b{1} = 0;
```

Để huấn luyện gia tăng, ta cần coi các đầu vào và đích là các chuỗi:

$$P = \{[1;2] [2;1] [2;3] [3;1]\}$$

$$T = \{4 \ 5 \ 7 \ 7\};$$

Như đã đề cập ở mục trước, đối với mạng tính kết quả sự mô phỏng của mạng ở đầu ra liệu có giống như đầu vào đã được đưa ra như là một ma trận của véctơ đồng thời hay như là một mảng của các véctơ liên tiếp. Điều này là không đúng khi huấn luyện mạng. Tuy vậy khi sử dụng hàm **Adapt**, nếu một mảng các véctơ liên tục được đưa đến đầu vào thì hàm trọng được cập nhật như với mỗi đầu vào được đưa đến. Như chúng ta sẽ thấy ở phần sau, nếu một ma trận của véctơ đồng thời được đưa đến đầu vào thì hàm trọng chỉ được cập nhật sau khi tất cả các tín hiệu vào được đưa đến.

Để huấn luyện gia tăng ta sử dụng dòng lệnh:

$$[\mathbf{net}, \mathbf{a}, \mathbf{e}, \mathbf{p}, \mathbf{f}] = \mathbf{adapt}(\mathbf{net}, \mathbf{P}, \mathbf{T});$$

Đầu ra của mạng vẫn là zero bởi lẽ tốc độ học bằng zêro và hàm trọng không được cập nhật. Các giá trị sai lệch sẽ bằng các giá trị đích:

$$\mathbf{a} = [0] \quad [0] \quad [0] \quad [0]$$

$$\mathbf{e} = [4] \quad [5] \quad [7] \quad [7]$$

Nếu bây giờ ta đặt tốc độ học bằng 0, 1 ta có thể thấy mạng sẽ được điều chỉnh mỗi khi đầu vào có tín hiệu:

$$\mathbf{net.inputWeights}\{1,1\}.\mathbf{learnParam.Ir}=0.1;$$

$$\mathbf{net.biases}\{1,1\}.\mathbf{learnParam.Ir}=0.1;$$

$$[\mathbf{net}, \mathbf{a}, \mathbf{e}, \mathbf{p}, \mathbf{f}] = \mathbf{adapt}(\mathbf{net}, \mathbf{P}, \mathbf{T});$$

$$\mathbf{a} = [0] \quad [2] \quad [6.0] \quad [5.8]$$

$$\mathbf{e} = [4] \quad [3] \quad [1.0] \quad [1.2]$$

Dữ liệu ra thứ nhất tương tự như dữ liệu ra với tốc độ học bằng 0, do không có sự cập nhật nào cho tới khi dữ liệu vào thứ nhất xuất hiện. Dữ liệu ra thứ hai là khác do hàm trọng đã được cập nhật. Các hàm trọng liên tục được sửa đổi theo mỗi sai lệch được tính toán. Nếu mạng có năng lực và tốc độ huấn luyện chọn hợp lý thì các sai lệch sẽ dần tiến tới zêro.

b/ Huấn luyện gia tăng đối với mạng động

Đối với mạng động, ta cũng có thể huấn luyện gia tăng (đây là kiểu huấn luyện chung nhất). Xét mạng tuyến tính với một trễ ở đầu vào mà ta đã đề cập ở phần trước. Ta cho giá trị ban đầu của hàm trọng bằng 0 và đặt tốc độ học là 0,1.

```
net = newlin([-1 1],1,[0 1], 0.1);
```

```
net.IW{1, 1} = [0 0];
```

```
net.biasconnect = 0;
```

Để huấn luyện gia tăng mạng này, ta biểu diễn dữ liệu vào và dữ liệu đích như là các phần tử của mảng.

```
Pi = {1};
```

```
P = {2 3 4};
```

```
T = {3 5 7};
```

Ở đây ta thử huấn luyện mạng thực hiện phép cộng dữ liệu vào hiện thời và dữ liệu vào trước để tạo ra dữ liệu ra hiện thời. Điều này giống như sự nối tiếp dữ liệu vào ta đã sử dụng ở ví dụ trước của sự sử dụng hàm **Sim**, Chỉ có điều chúng ta gán giới hạn đầu tiên trong sự nối tiếp như điều kiện ban đầu cho sự trì hoãn. Bây giờ ta có thể sử dụng hàm **Adapt** để huấn luyện mạng:

```
[net,a,e,pf] = adapt(net,P,T,Pi);
```

```
a = [0] [2.4] [7.98]
```

```
e = [3] [2.6] [-0.98]
```

Dữ liệu ra đầu tiên bằng 0 do hàm trọng chưa được cập nhật. Hàm trọng sẽ thay đổi tại mỗi bước thời gian kế tiếp.

3.5.2 Huấn luyện mạng theo gói

Huấn luyện theo gói trong đó các hàm trọng và độ dốc chỉ được cập nhật sau khi tất cả các dữ liệu vào và đích đã được đưa tới, có thể được áp dụng cho cả mạng tĩnh và mạng động. Trong mục này, chúng ta sẽ thảo luận kỹ cả hai loại mạng này.

a/ Huấn luyện theo gói đối với mạng tĩnh

Để huấn luyện theo gói, ta có thể sử dụng hàm `adapt` hoặc hàm `train`, song nói chung trai là tùy chọn tốt nhất, vì nó đặc trưng cho sự truy nhập có hiệu quả hơn của giải thuật huấn luyện. Như vậy, sự huấn luyện gia tăng chỉ có thể làm việc với hàm `adapt`, còn hàm `train` chỉ có thể thực hiện để huấn luyện theo gói. Trước hết ta hãy bắt đầu huấn luyện theo gói đối với mạng tĩnh đã đề cập trong ví dụ trước, tốc độ học đặt bằng 0,1.

```
net = newlin([-1 1;-1 1],1,0,0.1);
```

```
net.IW{1,1} = [0 0];
```

```
net.b{1} = 0;
```

Để huấn luyện theo gói mạng tĩnh các véc tơ dữ liệu vào cần được đặt trong ma trận của các véc tơ đồng thời.

```
P = [1 2 2 3; 2 1 3 1];
```

```
T = [4 5 7 7];
```

Khi ta gọi lệnh `Adapt`, nó sẽ kéo theo `trains` (là các hàm thích nghi mặc định của mạng tuyến tính) và `learnwh` (là các hàm huấn luyện mặc định của hàm trọng và độ dốc).

```
[net,a,e,pf] = Adapt(net,P,T);
```

```
a = 0 0 0 0
```

```
e = 4 5 7 7.
```

Chú ý rằng tất cả các đầu ra của mạng đều bằng zero, bởi lẽ các hàm trọng chưa được cập nhật cho tới khi tất cả tập hợp huấn luyện được đưa tới. Nếu hiển thị trên màn hình ta thấy:

```
»net.IW{1,1}
```

```
ans = 4.9000    4.1000
```

```
»net.b{1}
```

```
ans =
```

```
2.3000.
```

Đây là sự khác nhau về kết quả ta nhận được sau một lần thực hiện hàm `Adapt` với sự cập nhật gia tăng. Bây giờ chúng ta hãy thực hiện việc huấn luyện theo gói sử dụng hàm `train`. Do luật Widrow-Hoff có thể sử dụng cho kiểu gia tăng và kiểu gói, nó có thể được gọi bằng `Adapt` hoặc `train`. Có một vài thuật toán huấn luyện chỉ có thể sử dụng trong kiểu gói (ví dụ Levenberg-Marquardt) và do đó các thuật toán này chỉ có thể gọi bằng lệnh `train`. Mạng

sẽ được cài đặt bằng cách tương tự.

```
net = newlin([-1 1;-1 1],0,0.1);
```

```
net.IW{1,1} = [0 0];
```

```
net.b{1} = 0;
```

Trong trường hợp này véc tơ dữ liệu vào có thể đặt dưới dạng ma trận của các véc tơ đồng thời (concurrent vectors) hoặc dưới dạng mảng của các véc tơ liên tiếp. Trong **Train**, mảng của các véc tơ liên tiếp bất kỳ được chuyển đổi thành ma trận của các véc tơ đồng thời. Đó là do mạng là tĩnh và do lệnh **train** luôn luôn hoạt động theo kiểu gói.

```
P = [1 2 2 3; 2 1 3 1];
```

```
T = [4 5 7 7];
```

Bây giờ ta sẵn sàng để huấn luyện mạng. Ta sẽ huấn luyện nó chỉ trong một kỳ vì ta chỉ sử dụng một lần hàm **Adapt**. Hàm huấn luyện mặc định cho mạng tuyến tính là **train** và hàm huấn luyện mặc định cho hàm trọng và độ dốc là **learnwh**, vì vậy ta có thể nhận được các kết quả tương tự kết quả sử dụng **Adapt** trong ví dụ trước, khi ta sử dụng hàm thích nghi mặc định là **trains**.

```
net.inputWeights{1,1}.learnParam.Ir = 0,1;
```

```
net.biases{1}.learnParam.Ir = 0,1;
```

```
net.trainparam.epochs : 1;
```

```
net = train(net,P,T);
```

Nếu cho hiển thị hàm trọng sau một kỳ huấn luyện ta thấy:

```
»net.IW{1,1}
```

```
ans = 4.9000 4.1000
```

```
»net.b{1}
```

```
ans =
```

```
2.3000.
```

Kết quả này tương tự với kết quả huấn luyện theo gói sử dụng **Adapt**. Đối với mạng tĩnh, hàm **Adapt** có thể thực hiện sự huấn luyện gia tăng hoặc theo gói tùy thuộc vào khuôn dạng dữ liệu vào. Nếu dữ liệu được đưa tới mạng dưới dạng ma trận của các véc tơ đồng thời thì huấn luyện theo gói sẽ xảy ra. Nếu dữ liệu được đưa tới dưới dạng chuỗi thì huấn luyện gia tăng sẽ xảy ra. Điều này không đúng với hàm **train**, nó luôn luôn huấn luyện theo gói mà không phụ thuộc vào khuôn dạng của dữ liệu vào.

b/ Huấn luyện theo gói đối với mạng động

Huấn luyện mạng tĩnh học tương đối dễ hiểu. Nếu ta sử dụng thun để huấn luyện mạng theo gói và dữ liệu vào được chuyển đổi thành véc tơ đồng thời (các cột của ma trận) cho dù khuôn dạng trước đây của chúng là chuỗi. Nếu ta sử dụng **Adapt** thì khuôn dạng dữ liệu vào quyết định phương pháp huấn luyện. Nếu khuôn dạng dữ liệu vào là chuỗi thì mạng được huấn luyện kiểu gia tăng, nếu khuôn dạng dữ liệu vào là véc tơ đồng thời thì mạng được huấn luyện kiểu gói.

Đối với mạng động, kiểu huấn luyện theo gói chỉ được thực hiện với hàm **train**. Để minh họa điều này ta lại xét mạng tuyến tính có trễ. Ta sử dụng tốc độ học là 0,02 để huấn luyện. Khi sử dụng giải thuật giảm độ dốc ta chọn tốc độ học cho kiểu huấn luyện gói nhỏ hơn kiểu huấn luyện gia tăng.

Ví dụ:

```
net = newlin([-1 1],1,[0 1],0.02);
```

```
net.IW{1,1}=[0 0];
```

```
net.biasConnect 0;
```

```
net.trainparam.epochs = 1;
```

```
Pi = {1};
```

```
P = {2 3 4};
```

```
T = {3 5 6};
```

Ta muốn huấn luyện mạng với chuỗi tương tự như đã sử dụng cho sự huấn luyện gia tăng trước đây thế nhưng thời điểm cần thiết để cập nhật các hàm trọng chỉ xảy ra sau khi tất cả dữ liệu vào được áp dụng (kiểu gói). Mạng được coi như tuần tự vì đầu vào là tuần tự, song các hàm trọng được cập nhật theo kiểu gói.

```
net=train(net,P,T,Pi);
```

```
»net.IW{1,1}
```

```
ans = 0.9000 0.6200.
```

Kết quả này khác với kết quả ta đã thu được bằng huấn luyện gia tăng, ở đó các hàm trọng được cập nhật 3 lần trong thời gian một tập huấn luyện. Đối với huấn luyện theo gói các hàm trọng chỉ được cập nhật một lần trong một khóa huấn luyện.

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.

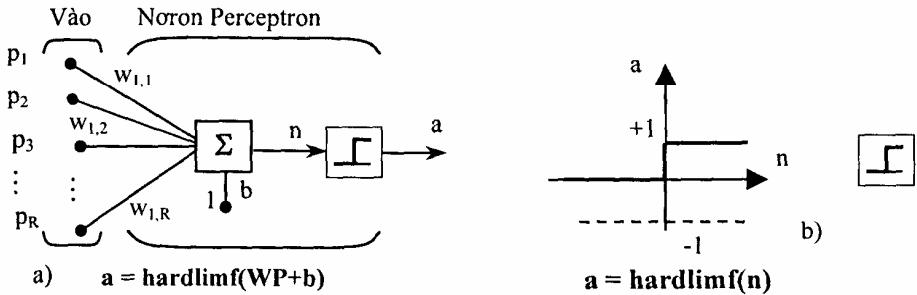
Rosenblatt đã thiết lập nhiều biến thể của mạng perceptron. Một trong các dạng đơn giản nhất là mạng lớp đơn mà hàm trọng và độ dốc của nó có thể được huấn luyện để đưa ra một véc tơ đích chính xác khi có véc tơ vào tương ứng được gửi tới. Kỹ thuật huấn luyện được gọi là luật học perceptron. Perceptron 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 bố 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_{ij} và tổng các đầu vào kể cả độ dốc b là $n = \sum w_{ij} + b$ 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. Noron với R đầu vào

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

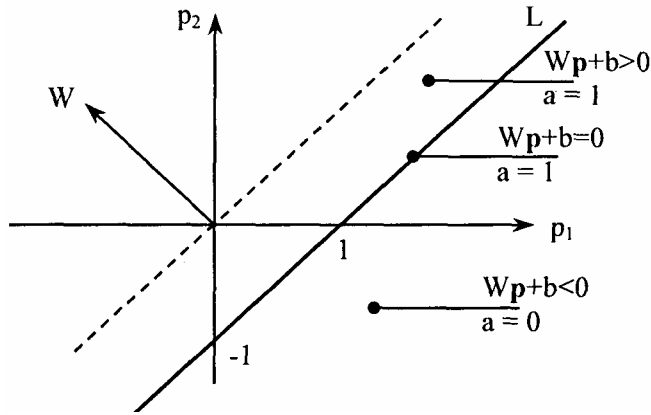
Với hàm chuyển **hard-limit** cho phép Perceptron có khả năng phân loại véc tơ vào bằng cách phân chia không gian vào thành 2 vùng, phân cách với nhau bằng đường biên giới L ứng với phương trình: $\mathbf{W} \cdot \mathbf{p} + b = 0$.

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

$$\begin{aligned} n &= \mathbf{W} \cdot \mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b \\ &= -p_1 + p_2 + 1. \end{aligned}$$

Đường biên giới L được chỉ ra trên hình 4. 1. Đường này vuông góc với ma trận trọng \mathbf{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, noron **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. Noron **hard-limit** không có độ dốc sẽ luôn có đường biên giới đi qua gốc toạ độ. Cộng thêm độ dốc sẽ cho phép noron 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 lập của các luật học.

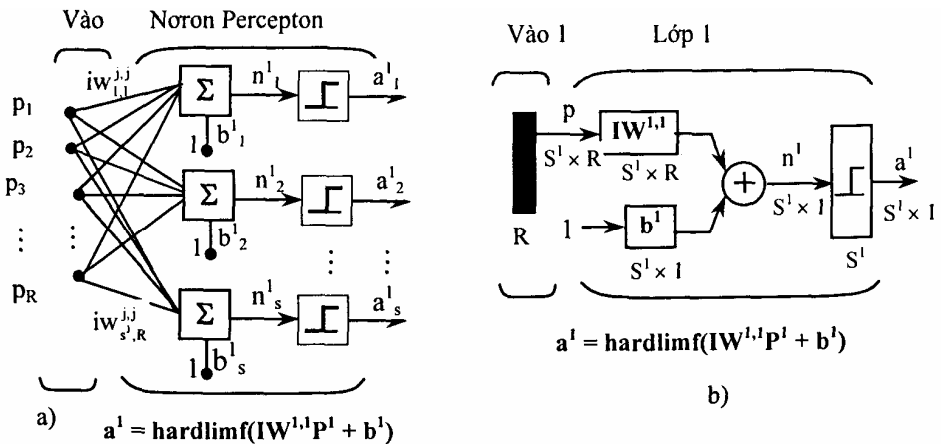


Hình 4.2. Sự phân loại của nơron Perceptron 2 đầu vào

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}^1 .

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 phân tử vào, S là số neuron. Hàm chuyển mặc định của perceptron là hardlim.

Ví dụ: để thiết lập một mạng perceptron với một phân 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: [1x1 struct]
```

```
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:

```
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 kiểm 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 =
```

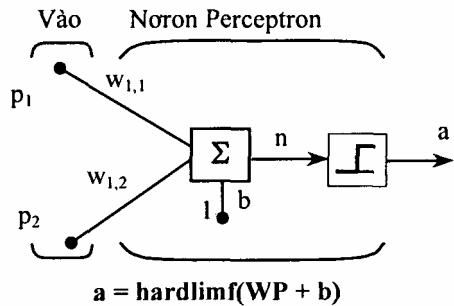
```
1
```

```
and for p2 = [1;-1]
```

```
a2 = sim(net,p2)
```

```
a2 =
```

```
0
```



Hình 4.4. Nơron với 2 đầu vào

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 trị 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:

bias =

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 có thể định nghĩa lại các hàm trọng đầu vào mạng và độ dốc **intFcn** như sự ngẫu nhiên và sau đó áp dụng in 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 thể 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 sát** các luật học được cung cấp cùng với một tập các ví dụ (tập huấn luyện) đặc trưng của mạng:

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

Trong đó P_Q là đầu vào mạng và 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 bố. 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:

$$P_1, t_1; p_2, t_2; \dots; p_Q, t_Q$$

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 neuron 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 neuron 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 neuron 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 neuron bằng 1 trước đó có thể là 0 ($a = 1$; $t = 0$ và $e = t - a = -1$) 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 ΔW như sau:

Trường hợp 1: Nếu $e = 0$ thì sự thay đổi giá trị của ΔW 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 neuron với chú ý rằng, độ dốc chỉ đơn giản là hàm trọng có đầu vào là 1:

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

Đối với trường hợp của một lớp neuron ta có:

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

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

Luật học perceptron có thể tóm tắt như sau:

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

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

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

$$\text{net} = \text{newp}([-2 \ 2; -2 \ 2], 1);$$

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

$$\text{net.b}\{1\} = [0];$$

$$w = [1 \ 0 \ 8];$$

$$\text{net.IW}\{1,1\} = w;$$

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:

$$a = \text{sim}(\text{net}, p)$$

$$\text{Kết quả: } a = 0$$

$$e = t - a = 1$$

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

$$dw = \text{learnp}(w, p, [], [], [], [], e, [], [], [])$$

$$dw : 1 \quad 2$$

Hàm trọng mới thu được là:

$$W = W + dW$$

$$W = 2.0000 \quad 1.2000$$

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)

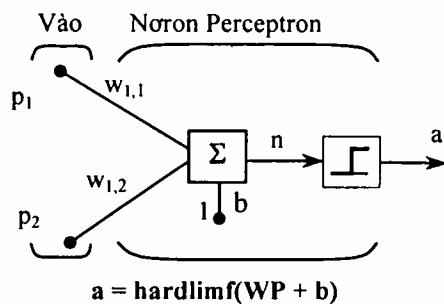
Lệnh **sim** và hàm **learnp** được sử dụng lặp để đưa các dữ liệu đầu vào đến perceptron và để thay đổi hàm trọng và độ dốc của perceptron theo sai lệch giữa đầu ra của mạng và đích, perceptron sẽ tìm được hàm trọng và độ dốc cuối cùng để giải quyết bài toán đặt ra (với điều kiện bài toán đó perceptron 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 thêm nữa bằng cách gọi lại train với hàm trọng và độ dốc mới cho các lần huấn luyện thêm, 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 họa cho thủ tục huấn luyện **train**, ta xét một neuron 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).

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



Hình 4.5. Neuron với 2 đầu vào

$$\left\{ \mathbf{P}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0 \right\} \left\{ \mathbf{P}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1 \right\} \left\{ \mathbf{P}_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0 \right\} \left\{ \mathbf{P}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 1 \right\}.$$

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 \mathbf{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.

$$\mathbf{e} = t_1 - a = 0 - 1 = -1$$

$$\Delta W = \mathbf{e} \mathbf{P}_1^T = (-1)[2 \ 2] = [-2 \ -2]$$

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

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

$$W^{\text{mới}} = W^{\text{cũ}} + \mathbf{e} \mathbf{P}_1^T = [0 \ 0] + [-2 \ -2] = [-2 \ -2] = w(1)$$

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

Với véc tơ vào \mathbf{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 \mathbf{P}_3 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:

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

Đền 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:

$$\mathbf{net} = \mathbf{newp}(\mathbf{[-2 \ 2; -2 \ +2]}, \mathbf{1});$$

Quan sát giá trị của đầu vào đơn.

$$\mathbf{p} = \mathbf{[2; 2]};$$

ta có đích

$$\mathbf{t} = \mathbf{(0)};$$

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

$$\mathbf{net.trainparam.epochs} = \mathbf{1};$$

$$\mathbf{net} = \mathbf{train}(\mathbf{net}, \mathbf{p}, \mathbf{t});$$

Hàm trọng mới và độ dốc mới là:

$$\mathbf{w} =$$

$$\mathbf{-2 \ -2}$$

$$\mathbf{b} =$$

$$\mathbf{-1}$$

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:

$$\mathbf{net} : \mathbf{newp}(\mathbf{[-2 \ 2; -2 \ +2]}, \mathbf{1});$$

$$\mathbf{net.trainParam.epochs} = \mathbf{1};$$

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 thêm 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

$$\text{error} = [a(1) - t(1) \quad a(2) - t(2) \quad a(3) - t(3) \quad a(4) - t(4)]$$

$$\text{error} =$$

$$0 \quad 0 \quad 0 \quad 0$$

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 `trains` (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 túy. 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 dụng 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 neuron 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 neuron perceptron, 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 thêm về perceptron và các bài toán perceptron phức tạp.*)

Những sự bất thường và luật perceptron 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:

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

Như đã chỉ ra ở trên, độ rộng của véc tơ vào \mathbf{p} , có tác động lên véc tơ hàm trọng \mathbf{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:

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

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 NƠON

(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 người 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 công logic AND. Nó có véc tơ vào:

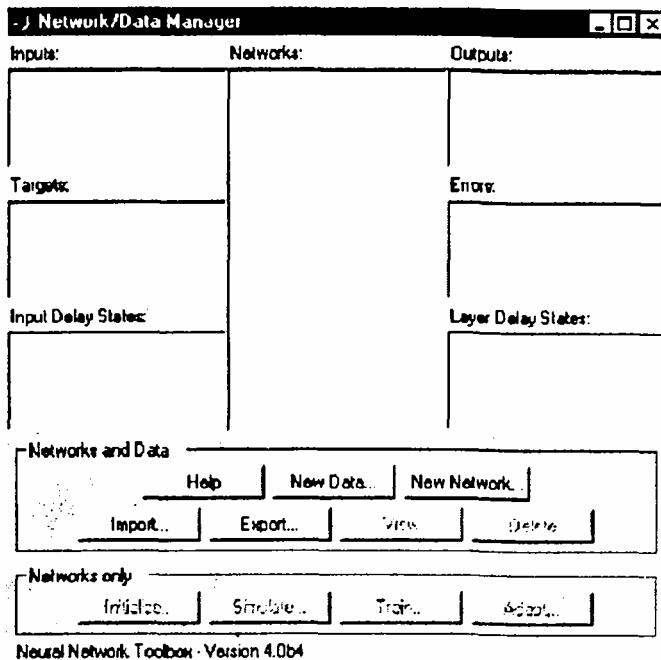
$$p = [0 \ 0 \ 1 \ 1; \ 0 \ 1 \ 0 \ 1] \text{ và véc tơ đích là:}$$

$$t = [0 \ 0 \ 0 \ 1]$$

Ta gọi mạng là **ANDNet**. Một lần thiết lập mạng sẽ được 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

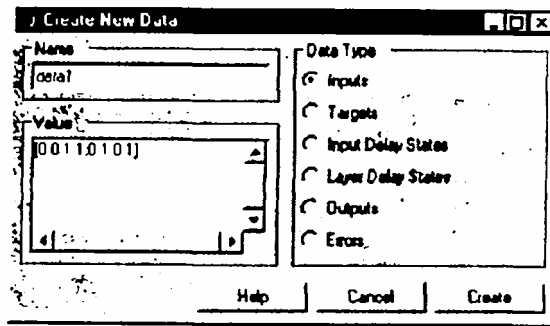
Để bắt đầu ta gõ nntool, xuất hiện cửa sổ hình 4.6.



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].

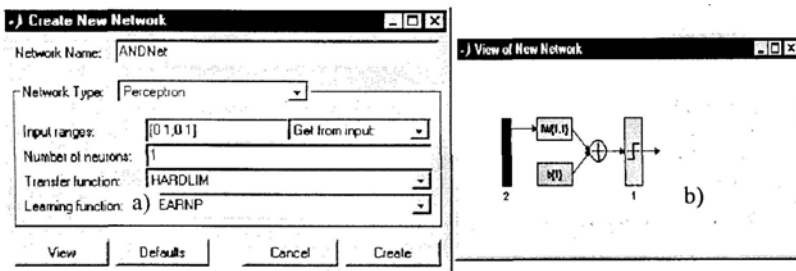
Như vậy, mạng có 2 phần tử vào và 4 tập của 2 phần tử véc tơ đó được đưa đến để huấn luyện. Để định nghĩa dữ liệu này, kích vào **new data** sẽ xuất hiện cửa sổ **Create New Data**. Đặt tên cho p, giá trị là $[0 \ 0 \ 1 \ 1; \ 0 \ 1 \ 0 \ -1]$ và xác định kiểu dữ liệu (data type) là tập dữ liệu vào (inputs). Cửa sổ thiết lập dữ liệu mới như hình 4.7. Bây giờ kích **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 kích **Create** ta sẽ thấy cửa sổ **Network/Data Manager** 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 kiểu mạng **Network Type** là **Perceptron**, khi đó kiểu mạng ta mong muốn được thiết lập. Phạm vi đầu vào có thể được cài đặt bằng con số trong vùng đó. song ta cũng rất dễ dàng nhận được chúng từ 1 đầu dữ liệu riêng biệt ta cần sử dụng. Để làm điều này ta kích vào mũi lên đi 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

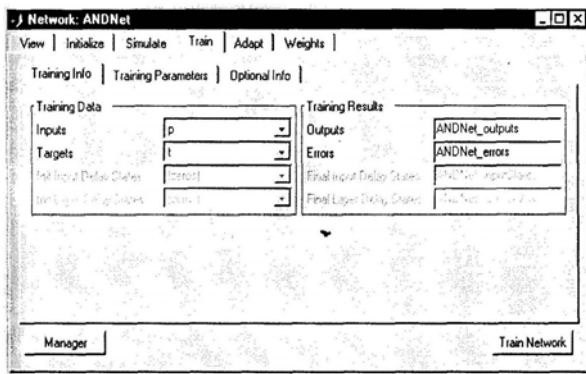
Ta chọn **hardlim** trong menu hàm chuyển **transfer function** và **learnp** trong menu hàm học **learning function**. Đến đây ta có cửa sổ Create New Network như hình 4.8a. Ta có thể quan sát cấu trúc mạng bằng cách kích vào **View** (hình 4.8b).

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 **Network/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: **Network: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 nhãn **Training Info**, chọn P trong hộp thoại **Inputs** và t trong hộp thoại **targets**. Khi đó cửa sổ **Network:ANDNet** như hình 4.9.

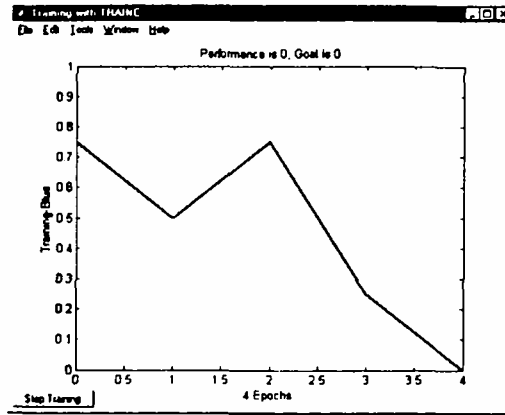


Hình 4.9

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 perceptron, ta được kết quả như hình 4.10.



Hình 4.10

Vậy, mạng đã được huấn luyện để 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 dừng ở trên cho mạng perceptron).

Ta có thể **kiểm tra** rằng 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 **Network: ANDNet** kích vào **Simulate**. Lúc này menu **Input pull-down** trái xuống chỉ rõ p là đầu vào và nhận 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**. Kích đúp vào nó, một cửa sổ dữ liệu nhỏ: **ANDnet_outputsSim** mở ra với trị số [0 0 0 1].

Vậy, mạng thực hiện công 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 Manager**. Kích vào **ANDNet_outputs** và **ANDNet-errors** 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 gõ **ANDNet_outputs** 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} yields ans =

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

Ta có thể xóa cửa 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 xóa danh sách.

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

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 xóa **Network/Data Manager**.

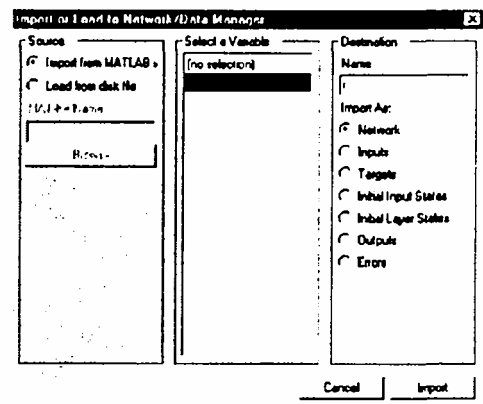
4.5.6 Nhập từ dòng lệnh

Đơn giản, ta thoát khỏi MATLAB, khởi động lại lần nữa và gõ lệnh nntool để bắt đầu một trang mới.

Thiết lập véc tơ mới:

r = [0; 1; 2; 3]

r =



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 Network** đặt 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 Save** 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/Network 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 Network/Data Manager**. Trong danh sách **Import As**, chọn **Network**, **mynet** nổi lên và kích vào **Load** để đưa **mynet** đến GUI. Bây giờ ta đã có ở trong cửa sổ GUI **Network/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 véc 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 đối 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

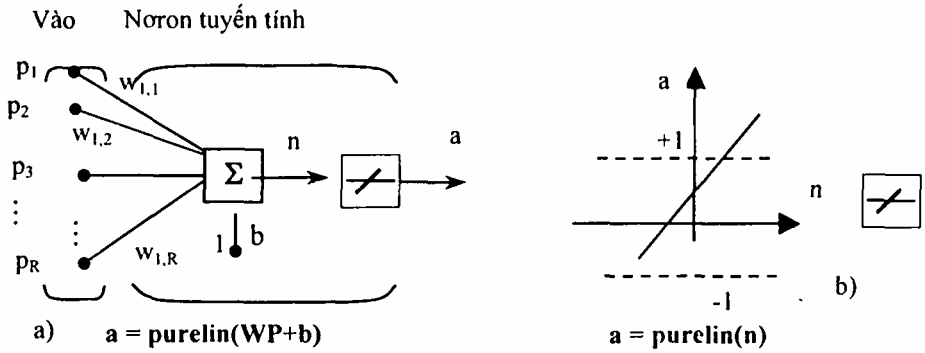
Một nơron tuyến tính với R đầu vào được chỉ ra trên hình 5.1. Mạng tuyến tính có cấu trúc cơ bản tương tự như perceptron, chỉ có điểm khác là ở đây dùng dùng hàm chuyển tuyến tính, ta gọi nó là hàm **purelin**. Hàm chuyển tuyến tính tính toán đầu ra của nơron bằng cách điều chỉnh giá trị đưa vào:

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

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.

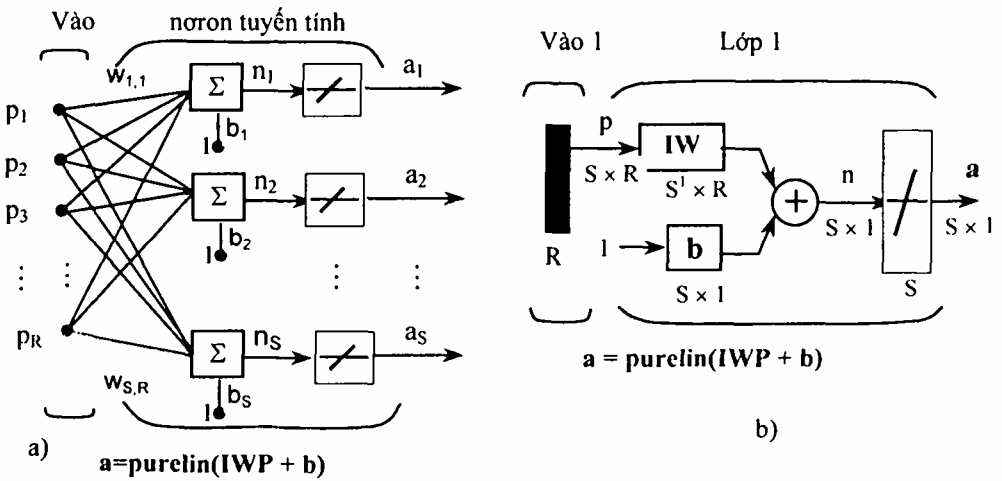
Mạng tuyến tính đương nhiên không phù hợp bác thực hiện các tính toán

cho hệ phi tuyến.



Hình 5.1a,b. Neuron với R đầu vào

a) Mô hình neuron, 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 neuron 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:

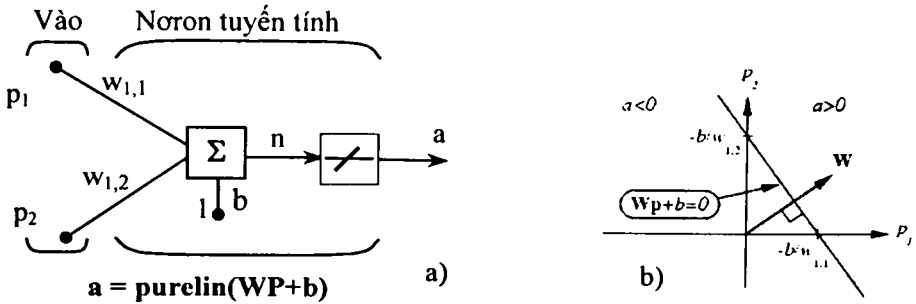
$$\mathbf{a} = \text{purelin}(\mathbf{n}) = \text{purelin}(\mathbf{w}\mathbf{p} + \mathbf{b}) = \mathbf{W}\mathbf{p} + \mathbf{b}$$

hoặc
$$\mathbf{a} = w_{1,1}p_1 + w_{1,2}p_2 + b.$$

Giống như perceptron, mạng tuyến tính có đường phân chia biên giới được xác định bằng các véc tơ vào đối với nó mạng vào \mathbf{n} bằng 0. Để $\mathbf{n} = 0$ thì biểu thức $\mathbf{W}\mathbf{p} + \mathbf{b} = 0$. 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:

$$\text{net} = \text{Newlin}([-1 \ 1; -1 \ 1], 1);$$



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:

$$\mathbf{W} = \text{net.IW}\{1,1\}$$

$$\mathbf{W} =$$

$$\mathbf{0 \ 0}$$

và

$$\mathbf{b} = \text{net.b}\{1\}$$

$$\mathbf{b} =$$

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:

$$\text{net.IW}\{1,1\} = [2 \ 3];$$

$$W = \text{net.IW}\{1,1\}$$

$$W =$$

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

$$\text{net.b}\{1\} = [-4];$$

$$b = \text{net.b}\{1\}$$

$$b =$$

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**.

$$a = \text{sim}(\text{net}, p)$$

$$a =$$

24

Tóm lại, ta có thể khởi tạo mạng tuyến tính với hàm **newlin**, điều chỉnh các phân 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 tuyến tính, trên chúng luật huấn luyện được chuẩn bị đầy đủ với tập mẫu các hành vi của mạng mong muốn:

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

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

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

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 tùy 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:

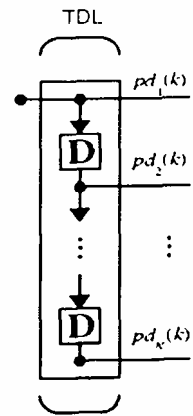
$$\text{net} = \text{Newlind}(p,T);$$

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

$$Y = \text{sim}(\text{net},P)$$

$$Y =$$

$$2.0500 \quad 4.0000 \quad 5.9500$$



Hình 5.4

5.5. MẠNG TUYẾN TÍNH CÓ TRỄ

5.5.1 Mất trễ

Ta cần một khâu mới là mất trễ để tạo nên năng lực sử dụng đầy đủ cho mạng tuyến tính, ví dụ một mất 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

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

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:

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

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

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:

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

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 neuron, khi đó la viết dưới dạng ma trận như sau:

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

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

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 $\mathbf{P}^T\mathbf{P}$ của véc tơ vào.

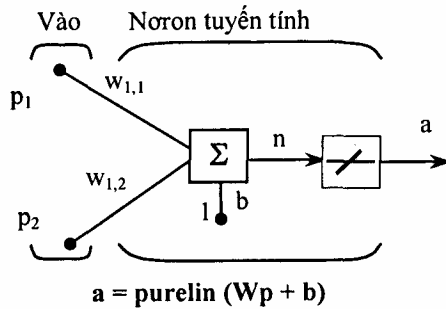
Hàm **learnwh** trong Matlab thực hiện tất cả các công việc tính toán. Nó tính toán sự thay đổi của hàm trọng và độ dốc theo công thức:

$$dw = lr * e * p' \text{ và } db = lr * e.$$

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: $0,999.p^T p$.

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. Neuron 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:

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

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 thêm 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 phân 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 phân tử véc tơ vào và một phân tử véc tơ đích

$$\mathbf{P} = [+1.0];$$

$$\mathbf{T} = [+0.5];$$

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 logic 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 tối ưu? Hình dạng các tập mờ thế nào? Vị 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?...

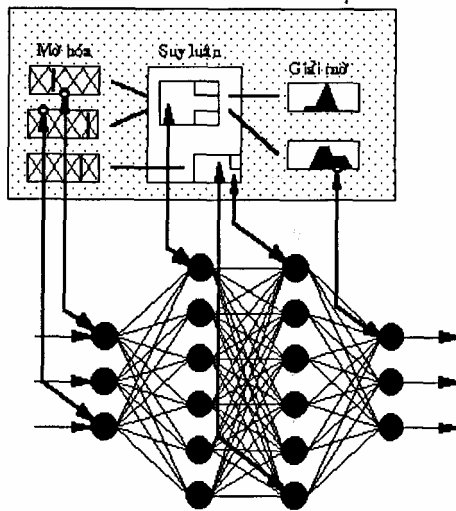
Đối với mạng nơron, chúng có một số ưu điểm như xử lý song song nên tốc độ xử lý rất nhanh; Mạng nơron có khả năng học hỏi; Ta có thể huấn luyện mạng để xấp xỉ một hàm phi tuyến bất kỳ, đặc biệt khi đã biết một tập dữ liệu vào/ra... Song nhược điểm cơ bản của mạng nơron là khó giải thích rõ ràng hoạt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



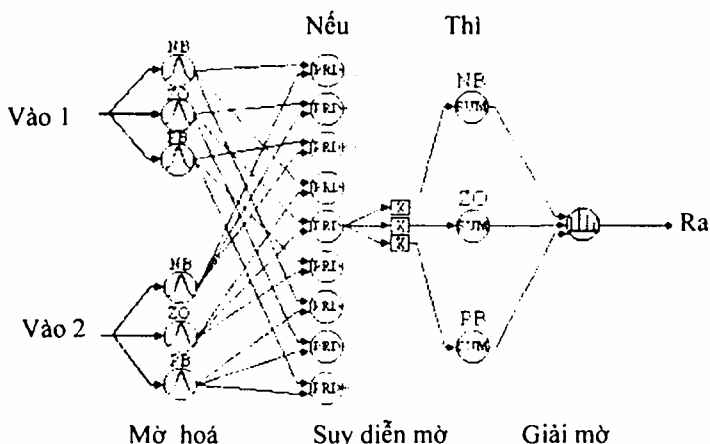
Hình 6.1. Mô hình hệ mờ - nơron

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 tử 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 (vì 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:

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

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

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:

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

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 :

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

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

Tập dữ liệu của khối luật là:

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

Còn đối với hệ MIMO thì khối luật :

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

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

Tập dữ liệu của khối luật là:

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

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, $M \geq 2$ và $N \geq 2$ ta đặt:

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

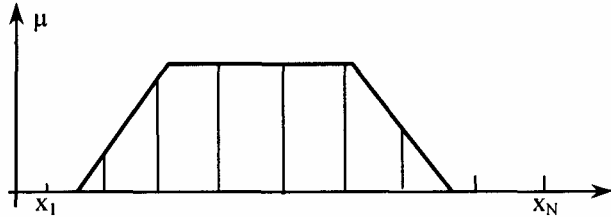
$$y_j = \beta_1 + (j - 1)(\beta_2 - \beta_1)/(M - 1)$$

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

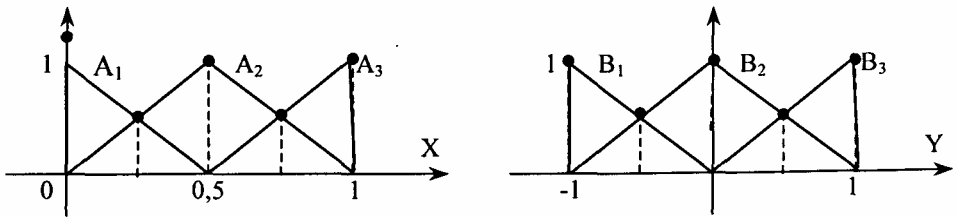
- Rời rạc hóa các tập mờ thành tập các cặp vào-ra (hình 6.3).

$$\{(A_i(x_1), \dots, A_i(x_N)), (B_i(y_1), \dots, B_i(y_M))\}, \text{ với } 1 \leq i \leq n.$$

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:

$$\mu_{A_1}(u) = 1 - 2x \quad 0 \leq x \leq \frac{1}{2}$$

$$\mu_{A_2}(u) = 1 - 2|x - 0,5| \quad 0 \leq x \leq 1$$

$$\mu_{A_3}(u) = 2x - 1 \quad \frac{1}{2} \leq x \leq 1$$

$$\mu_{B_1} = -y \quad -1 \leq y \leq 0$$

$$\mu_{B_2} = 1 - 2|y| \quad -\frac{1}{2} \leq y \leq \frac{1}{2}$$

$$\mu_{B_3} = y \quad 0 \leq y \leq 1.$$

+ 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:

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

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

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

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

+ Tập dữ liệu gồm 3 cặp vào-ra là:

$$\{(a_{11}, \dots, a_{15}), (b_{11}, \dots, b_{15})\}$$

$$\{(a_{21}, \dots, a_{25}), (b_{21}, \dots, b_{25})\}$$

$$\{(a_{31}, \dots, a_{35}), (b_{31}, \dots, b_{35})\}$$

với

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

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

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

Như vậy ta có:

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

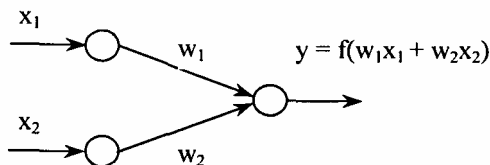
$$\{10;0,5;1;0,5;0), (0;0;1;0;0)\}$$

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

6.2. NƠN MỜ

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

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



Hình 6.5

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

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

$$p_1 + p_2 = w_1 x_1 + w_2 x_2.$$

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

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

$$y = f(w_1 x_1 + w_2 x_2), \quad f(x) = \frac{1}{1 + e^{-x}}$$

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

Nếu mạng noron dùng các phép toán khác như t-norm, t-conorm để kết hợp dữ liệu được gọi là mạng noron lai. Mạng noron lai là cơ sở để tạo ra cấu trúc noron mờ dựa trên các phép toán mờ. Để có mạng noron 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 $[0, 1]$.

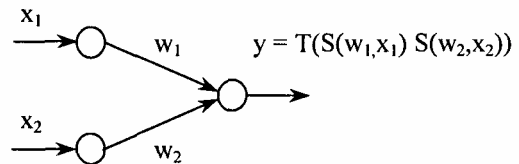
- Mạng noron lai có thể không dùng các phép toán nhân, phép toán cộng hoặc hàm dạng chữ S bởi vì kết quả của các phép toán này có thể không nằm trong khoảng $[0, 1]$.

Định nghĩa: Mạng noron lai là mạng noron 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 noron mờ thì giá trị vào, giá trị ra, và trọng số là những số thực nằm trong khoảng $[0, 1]$.

Noron mờ AND (hình 6.6)

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



Hình 6.6. Noron mờ AND

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

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

$$y = \text{AND}(p_1, p_2) = T(p_1, p_2) = T(S(w_1, x_1), S(w_2, x_2)).$$

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

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

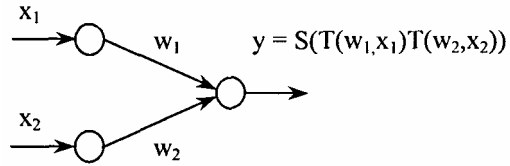
Nơ ron mờ OR (Hình 6.7)

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

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

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

$$\begin{aligned} y &= \text{OR}(p_1, p_2) = S(p_1, p_2) \\ &= S(T(w_1, x_1), T(w_2, x_2)). \end{aligned}$$



Hình 6.7. Nơ ron mờ OR

Nếu $T = \min$ và $S = \max$ thì nơ ron 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 NƠ RON-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 nơ ron 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 nơ ron.

Như vậy ta cần một tập dữ liệu ở dạng các cặp vào-ra mong muốn để cho mạng nơ ron 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:

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

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

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

Ri : Nếu x_1 là A_{i1} và... và x_n là A_{in} thì $y = z_i$, $1 \leq i \leq m$

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

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

Ký hiệu α_i 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:

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

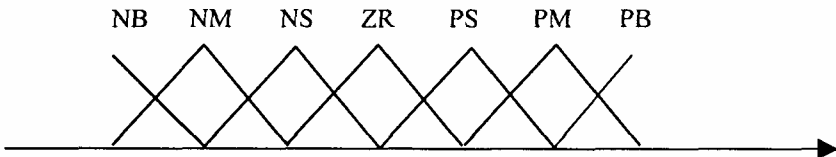
Sai lệch của mẫu thứ k là:

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

Dùng phương thức giảm để học z_i trong phần kết quả của luật **R_i**:

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

Cho rằng mỗi biến ngôn ngữ có 7 tập mờ như hình 6.8: {NB, NM, NS, ZR, 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

R₁: Nếu x là A₁ Thì y = z₁

R₂: Nếu x là A₂ Thì y = z₂

Giả sử A₁ và A₂ được định nghĩa bởi :

$$A_1 = \frac{1}{1 + \exp[\mathbf{b}_1(\mathbf{x} - \mathbf{a}_1)]} \quad A_2 = \frac{1}{1 + \exp[\mathbf{b}_2(\mathbf{x} - \mathbf{a}_2)]}$$

với a₁, a₂, b₁, b₂ là các giá trị khởi tạo ban đầu.

Vậy giá trị ra của luật là:

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

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:

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

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:

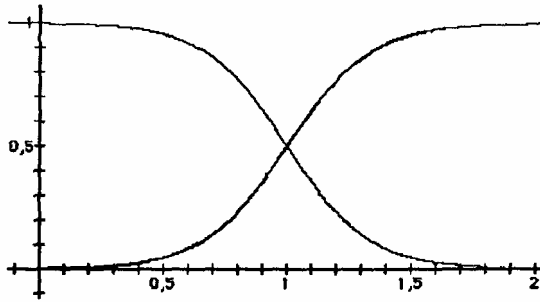
$$z_2(t+1) = z_2(t) - \eta(o^k - y^k) \frac{A_2(x^k)}{A_1(x^k) + A_2(x^k)}$$

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

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

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

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



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 :

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

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

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

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

$$(o^k - y^k) \frac{\partial}{\partial a} [z_1 A_1(x^k) + z_2 A_2(x^k)] =$$

$$(o^k - y^k) \frac{\partial}{\partial a} [z_1 A_1(x^k) + z_2 (1 - A_1(x^k))] =$$

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

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

$$(o^k - y^k) (z_1 - z_2) b A_1(x^k) (1 - A_1(x^k)) =$$

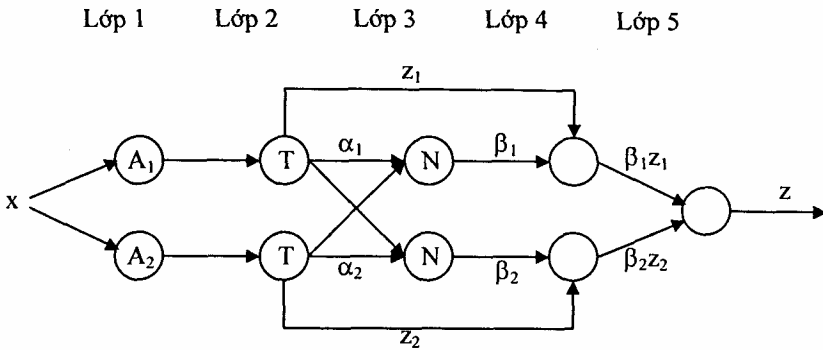
$$(o^k - y^k) (z_1 - z_2) b A_1(x^k) A_2(x^k)$$

và

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

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 đổi 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_1 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 - NƠN (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 tùy 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 GUI 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 liên thuộc. Đúng hơn là việc chọn các thông số liên kết với các hàm liên thuộc định sẵn là tùy 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 Inference 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 logic mờ dùng để thực hiện việc điều chỉnh tham số của hàm liên thuộc được gọi là **anfis**. Ta có thể mở **anfis** 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 đôi chút khác biệt (chúng ta sẽ bàn đến ở phần sau).

a/ Tìm hiểu về ANFIS

ANFIS xuất phát từ tiếng Anh là **Adaptive neuro-fuzzy inference 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 Breeds 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 đã 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 phân 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 tùy chọn của hệ thống suy luận mờ. Đặc biệt, anfis chỉ hỗ trợ cho các hệ thống mờ theo mô hình Sugeno 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 tùy 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 (huấn 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 dụng các lựa chọn trong Generate FIS của GUI.

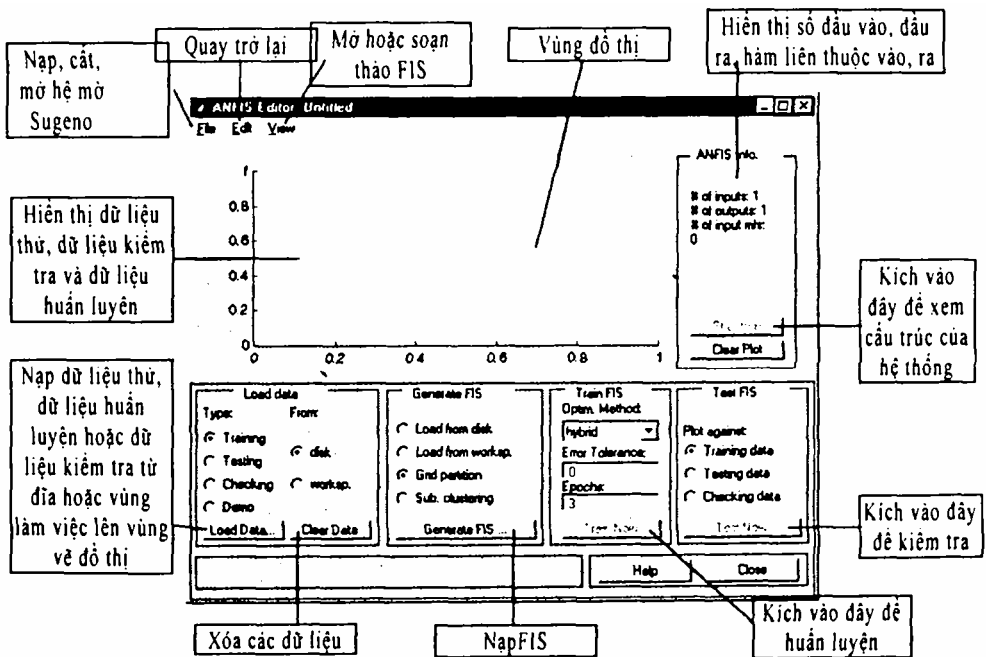
Xem cấu trúc mẫu FIS khi FIS ban đầu đã được tạo hoặc tải bằng cách nhấp vào nút Structure.

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

Chọn số kỳ 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 được 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 tùy 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 vectơ 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 dụ 1:

Dữ liệu kiểm tra giúp công nhận giá trị mẫu (Checking Data Helps 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: (**fuzex1chkData and fuzex2chkData**) vào vùng làm việc bộ soạn thảo ANFIS GUI.

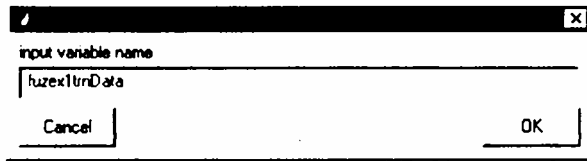
Để tải các tập dữ liệu từ thư mục `fuzzydemos` của Matlab vào vùng làm việc của MATLAB, từ cửa sổ dòng lệnh ta gõ:

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

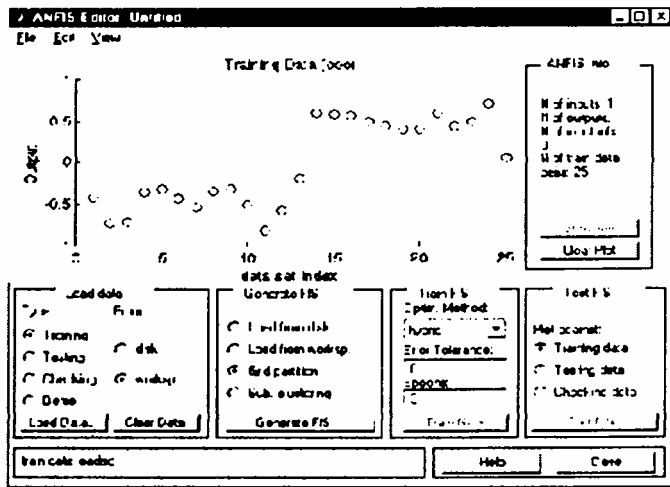
Chú ý: Ta có thể muốn tải dữ liệu từ `fuzzydemos` hoặc bất cứ thư mục nào trên đĩa, sử dụng bộ soạn thảo ANFIS Gõ một cách trực tiếp.

Mở bộ soạn thảo ANFIS GUI bằng cách gõ `anrscdit`. Để tải 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 **fuzex1tmData** như trong hình 6.12.



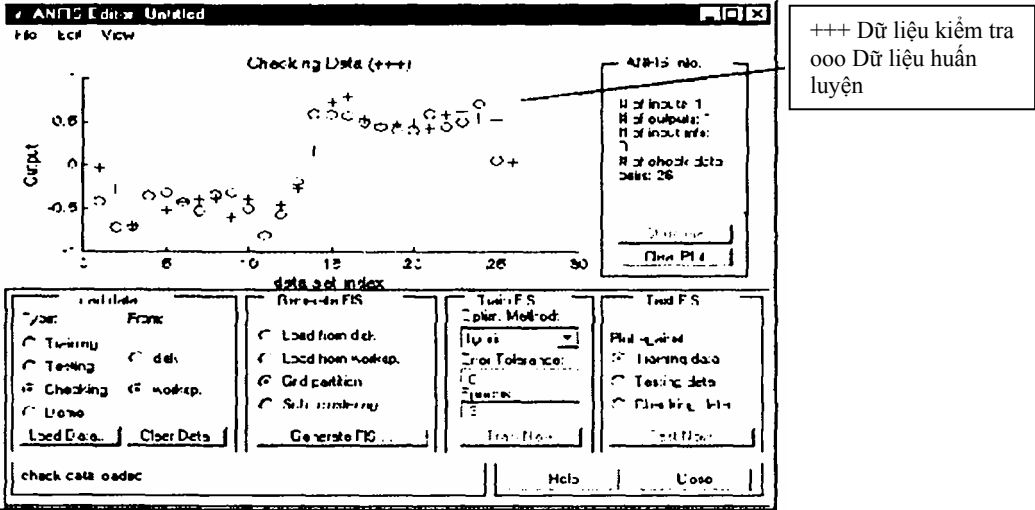
Hình 6.12



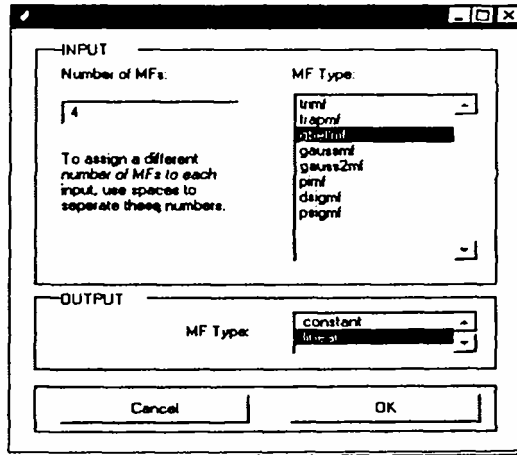
Hình 6.13

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 chỉ số dữ liệu. Chỉ số này chỉ rõ hàng để nhập giá trị dữ liệu đầu vào (dù đầu 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 sẽ 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 anfis theo các bước sau:

- ☞ Chọn Giọt 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 họa cách sử dụng của genfis1 và anfis sẽ được cung cấp sau.

+ **Án 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 functions** 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 rules** trong thực đơn **View**. Sử dụng Rule Editor 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 Generate 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 bấm nút Generate FIS để tải FIS bằng cách bấm vào nút này.

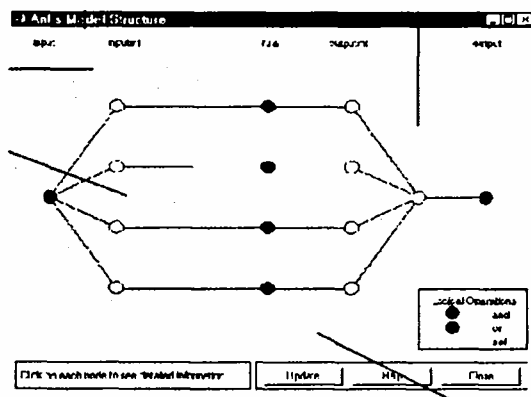
+ **Xem cấu trúc của FIS**

Sau khi tạo một mô hình FIS, ta có thể xem cấu 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**

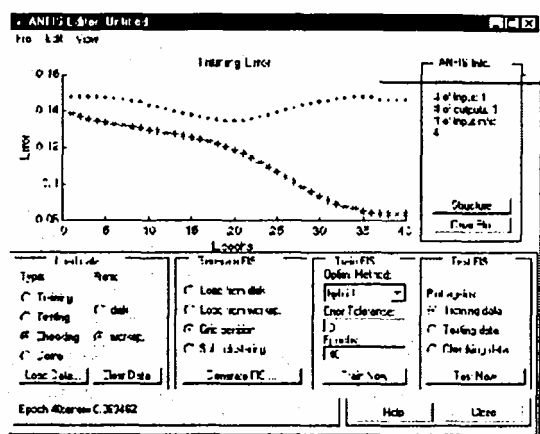


Hình 6.16

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 dừng 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 thực 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).

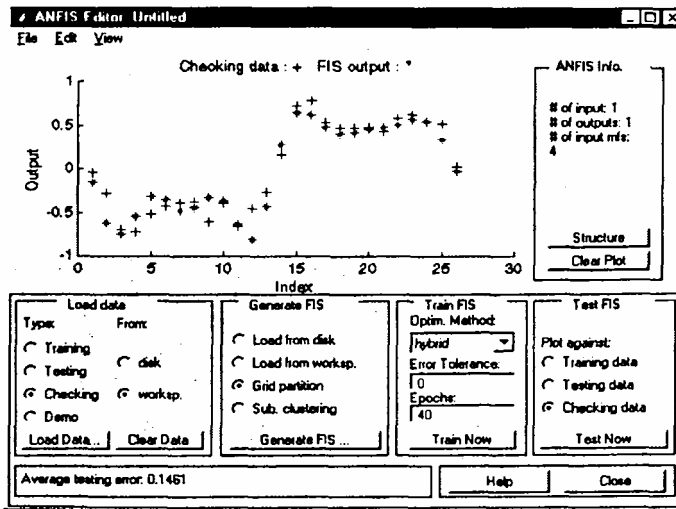


Hình 6.17

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 lằng lằng này thể hiện điểm mà lại đo 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, tùy chọn kiểm tra của anrs có hữu ích.

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



Hình 6.18

Để thử FIS tương phản với dữ liệu kiểm tra, chọn hộp kiểm tra **Checking data** trong phần **Test 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 xóa 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 xóa 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

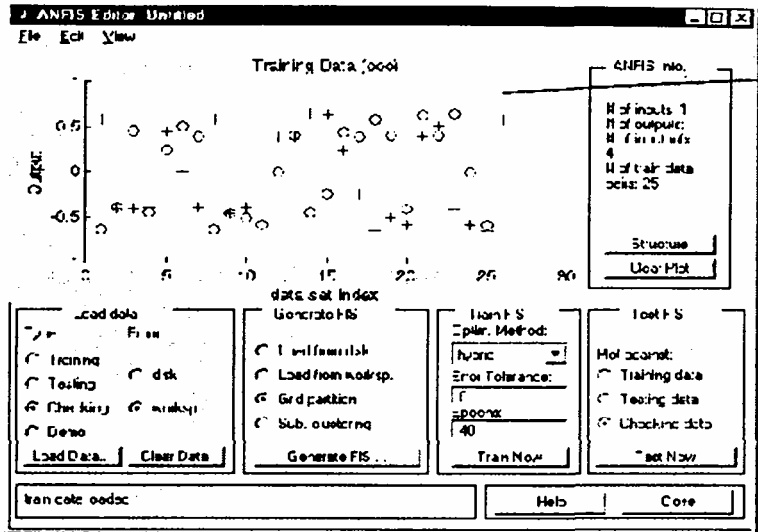
Trong ví dụ này, chúng ta sẽ khảo sát điều gì xảy ra khi dữ liệu huấn luyện và kiểm tra khác nhau hoàn toàn. Để xem cách sử dụng bộ soạn thảo ANFIS GUI để nghiên cứu một số điều về các tập dữ liệu và sự khác nhau giữa chúng. Ta thực hiện theo các bước sau:

☞ 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.

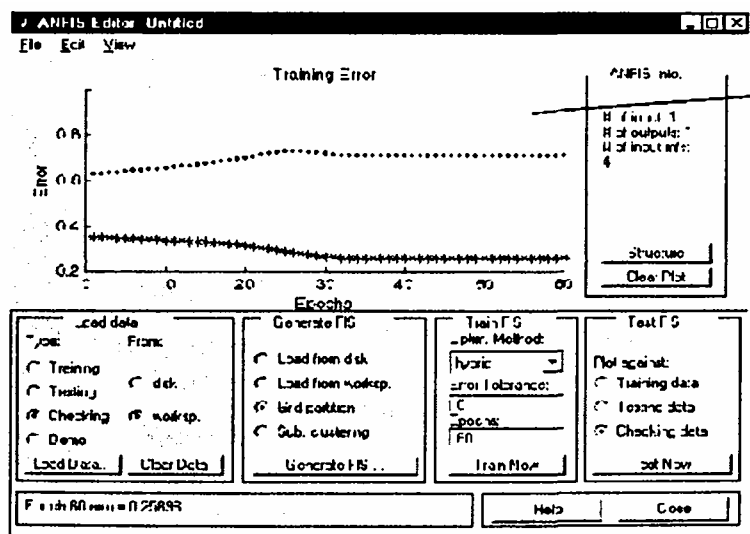
☞ Tải dữ liệu huấn luyện và dữ liệu kiểm tra theo thứ tự định sẵn **fuzex2trnData** và **fuzex2chkData** vào vùng làm việc của MATLAB tương tự như đã làm ở ví dụ trước.

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

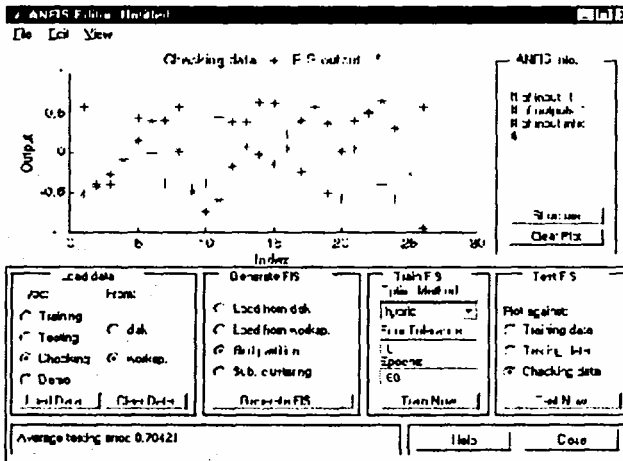


Hình 6.20

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 tùy chọn dữ liệu kiểm tra 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 đề 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 kiểu hàm liên thuộc). Hoặc là huấn luyện lại hệ thống mà không cần 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 - neuron (FIS). Tuy nhiên, như đã thấy trong ví dụ 2, ta cần phải 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:

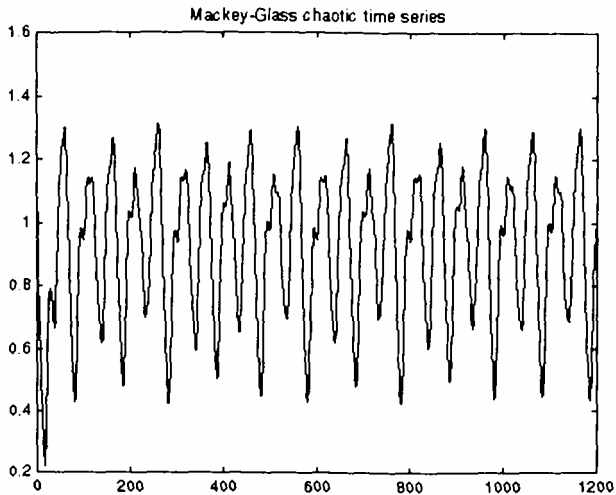
$$\dot{x}(t) = \frac{0,2x(-\tau)}{1 + x^{10}(t-\tau)} - 0,1x(t)$$

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 neuron 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.dat
```

```
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, $(x(t - (D-1)\Delta), \dots, x(t - \Delta), x(t))$ 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 $\Delta = P = 6$. 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 anfls (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

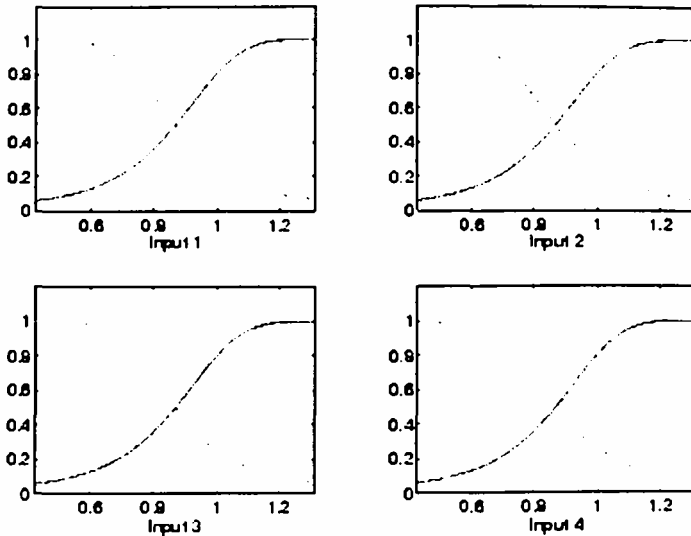
trnData : Data (1:500, :);


```
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 `genfis`:

```
fismat = genfis (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 `genfis` 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)  
subplot(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 tùy 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)

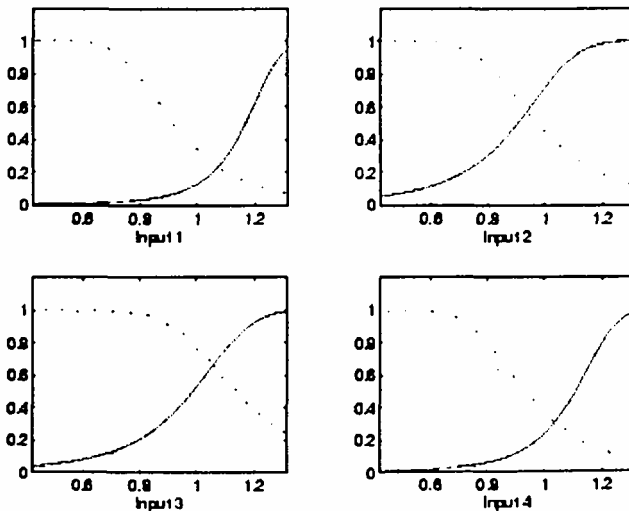
subplot(2,2,3)

plotmf(rsmat2, 'input', 3)

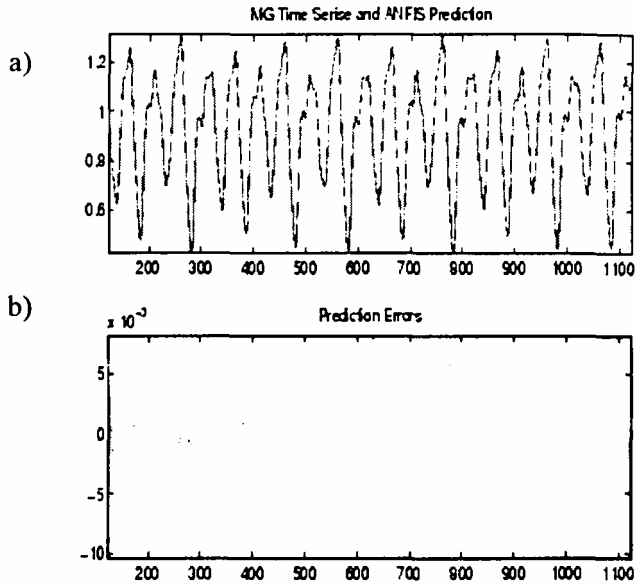
subplot(2,2,4)

plotmf(rsmat2, 'input', 4)

Hàm liên thuộc mới được chỉ 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 = evans([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 6.25a). Sai số giữa chúng được chỉ ra trên đồ thị hình 6.25b với tỉ lệ mịn hơn nhiều. Trong ví dụ này, ta mới chỉ 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 tùy chọn huấn luyện), **dispopt** (các tùy chọn hiển thị), **chkData** (dữ liệu kiểm tra) và **method** (phương pháp huấn luyện) là tùy chọn. Tất cả các đối số đầu ra đều là tùy 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 **ansedit**, 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, **trnData**, 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 vectơ 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 thảo 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.

Tùy chọn huấn luyện **trnOpt** cho dòng lệnh **anfis** là một vectơ 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 luyện sẽ 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 (**trnOpt(3)** - **trnOpt(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 tùy chọn cỡ bước để đánh giá tích cực việc huấn luyện. Tuy nhiên, như đã được nói đến ở phần trước, có những tùy chọn cỡ bước không được sử dụng đặc biệt để huấn luyện hệ thống neuron suy luận mờ thích ứng được tạo để sử dụng bộ soạn thảo ANFIS GUI.

6.7.4 Tùy chọn hiển thị Display Options

Tùy chọn hiển thị chỉ ứng dụng cho hàm lệnh **anrs**. Đối với dòng lệnh **anrs**, đối số tùy 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 luyện

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 tùy 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 tùy 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 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 tùy 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 thêm 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 Hiền (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 Tiên 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 noron 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 method for system Using symbolic stability analysis of fuzzy control*", The fourth Asian Fuzzy System Synposium, May 31 - June, Tsukuba, Japan. pp.450-455.
- [8] Kenji IKEDA and Seiichi SHIN (2000), "*Autonomous Decentralized Adaptive Control System Using Parameter Projection*", The fourth Asian Fuzzy System Synposium, May 31 - June, Tsukuba, Japan. pp.293-298.
- [9] Huganganhan & Shuta Murakami (2000), "*Adaptive Fuzzy Control of Nonlinear system With Various Shapes of Membership Functions*", The fourth Asian Fuzzy Systems Symosium, may 31 -June 3.2000 Tshkuba, Japan. pp.426-467.
- [10] Kosko, B (1991), "*Neural networks and fuzzy control*", Prentice Hall,.
- [11] L.A. Zadeh (1965), "*fuzzy set*", 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 robotics 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 Hiến 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.