Mạng nơ ron nhân tạo

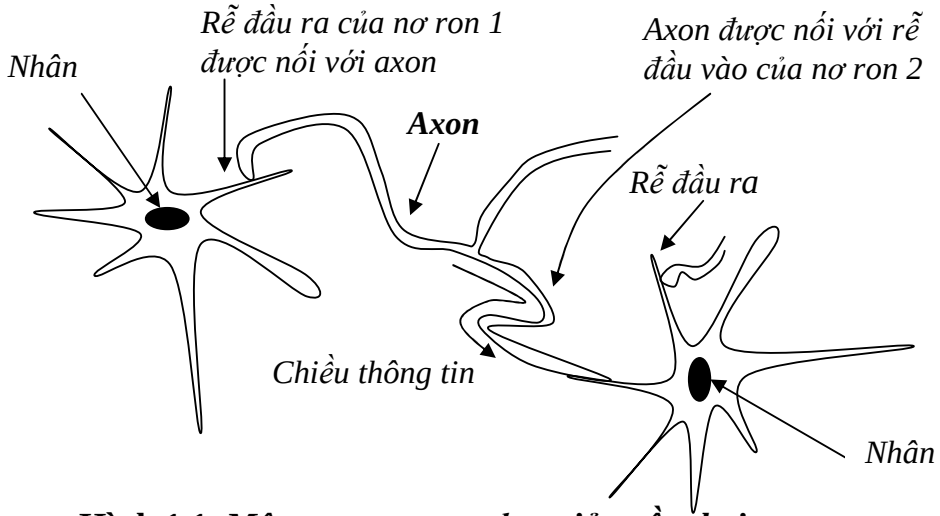
Mở đầu

Các mô hình tính toán mô phỏng bộ não người đã được nghiên cứu trong nửa đầu thế kỷ 20. Mặc dù có nhiều mô hình khác nhau được đề xuất, song tất cả đều dùng một cấu trúc mạng trong đó các đỉnh được gọi là các nơ ron. Các nơ ron này xử lý các tín hiệu số được gửi tới từ môi trường bên ngoài hoặc từ các nơ ron khác trong mạng thông qua các kết nối và sau đó gửi tín hiệu đến các nơ ron khác hoặc môi trường. Mạng nơ ron nhân tạo, gọi tắt là mạng nơ ron, là một lớp các mô hình tính toán như vậy.

1. Tổng quan về mạng nơ ron

1.1. Cấu trúc và mô hình của một nơ ron

Mạng nơ ron là sự tái tạo bằng kỹ thuật những chức năng của hệ thần kinh con người. Trong quá trình tái tạo không phải tất cả các chức năng của bộ não con người đều được tái tạo, mà chỉ có những chức năng cần thiết. Bên cạnh đó còn có những chức năng mới được tạo ra nhằm giải quyết một bài toán định trước.

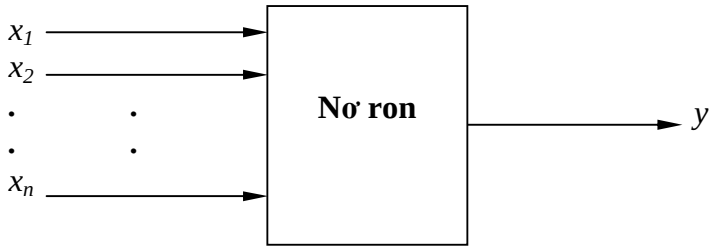


Hình 1.1. ***Một mạng nơ ron đơn giản gồm hai nơ ron.***

Mạng nơ ron bao gồm vô số các nơ ron được liên kết truyền thông với nhau trong mạng. Hình 1.1 là một phần của mạng nơ ron bao gồm hai nơ ron.

Nơ ron còn có thể liên kết với các nơ ron khác qua các rễ. Chính vì cách liên kết đa dạng như vậy nên mạng nơ ron có độ liên kết rất cao.

Các rễ của nơ ron được chia làm hai loại: loại nhận thông tin từ nơ ron khác qua *axon*, ta gọi là rễ đầu vào và loại đưa thông tin qua *axon* tới nơ ron khác gọi là rễ đầu ra. Một nơ ron có thể có nhiều rễ đầu vào, nhưng chỉ có một rễ đầu ra như vậy có thể xem nơ ron như một mô hình nhiều đầu vào một đầu ra MISO, (hình 1.1).



Hình 1.2. ***Nơ ron là mô hình MISO***

Một tính chất rất cơ bản của mạng nơ ron sinh học là các đáp ứng theo kích thích có khả năng thay đổi theo thời gian. Các đáp ứng có thể tăng lên, giảm đi hoặc hoàn toàn biến mất. Qua các nhánh axon liên kết tế bào nơ ron này với các nơ ron khác, sự thay đổi trạng thái của một nơ ron cũng kéo theo sự thay đổi trạng thái của những nơ ron khác và do đó làm thay đổi toàn bộ mạng nơ ron. Việc thay đổi trạng thái của mạng nơ ron có thể thực hiện qua một quá trình "*dạy*" hoặc do khả năng "*học*" tự nhiên.

Sự thay thế những tính chất này bằng một mô hình toán học tương đương được gọi là mạng nơ ron nhân tạo. Mạng nơ ron nhân tạo có thể được chế tạo bằng nhiều cách khác nhau vì vậy trong thực tế tồn tại rất nhiều kiểu mạng nơ ron nhân tạo.

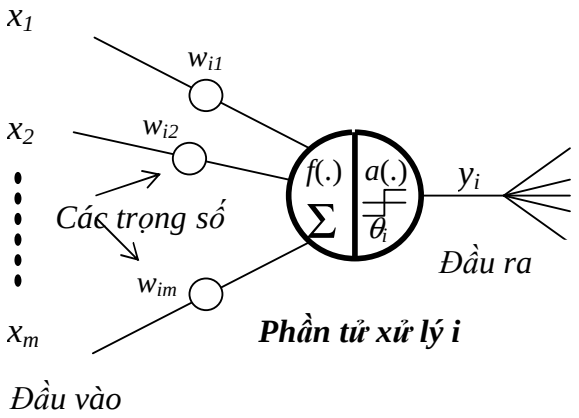
Mô hình nơ ron có m đầu vào $x_1, x_2, ...x_m$ và một đầu ra y (hình 1.2)

Mô hình này gồm có ba thành phần cơ bản:

Các kích thích đầu vào của tế bào nơ ron có thế năng tác động vào màng membran khác nhau được biểu diễn qua trọng lượng w_i, $i = 1, .., m$ tương ứng với cường độ kích thích của từng đầu vào. Tổng giá trị của các kích thích đầu vào được thực hiện qua một bộ tổng, đó là giá trị đo kích thích đầu vào tác động vào tế bào nơ ron.

Đầu ra của bộ tổng được đưa đến phần đáp ứng *a*(.). phần này không chỉ có chức năng tạo ra đáp ứng tức thời mà còn có khả năng lưu giữ các đáp ứng theo thời gian. Thành phần này hoạt động theo nguyên lý " nhớ " động.

Nơ ron bị kích thích trong thời gian thế năng của màng membran vượt quá ngưỡng. Quan hệ này được thực hiện nhờ hàm *a*(.), nó có chức năng của khâu tạo tín hiệu ngưỡng, xác định phụ thuộc của tín hiệu ra *y* vào các kích thích đầu vào.



Hình 1.3. ***Cấu trúc của một phần tử xử lý nơ ron thứ i (PE).***

Cách thành lập nơ ron nhân tạo như vậy tạo ra một độ tự do trong thiết kế. Việc lựa chọn phép cộng tín hiệu đầu vào và đáp ứng *a*(.) sẽ cho ra các kiểu mạng nơ ron nhân tạo khác nhau và tương ứng là các mô hình mạng khác nhau.

Theo hình 1.3 thì tín hiệu đầu ra y_i là:

$$y_i(t+1) = a\left(\sum_{j=1}^{m} w_{ij} x_j(t) - \theta_i\right) \tag{1.1}$$

trong đó hàm kích hoạt *a*(*f*) ở dạng hàm bước nhẩy:

$$a(f) = \begin{cases} 1 & khi \quad f \geq 0 \\ 0 & khi \quad f < 0 \end{cases} \tag{1.2}$$

Như vậy y_i chỉ có thể có 2 giá trị hoặc bằng 0, hoặc bằng 1.

$$f_i \triangleq net_i = \sum_{j=1}^{m} w_{ij} x_j - \theta_i \tag{1.3}$$

ở đây θ_i là ngưỡng đặt vào phần tử nơ ron thứ *i*.

Các hàm *f*(.) thường được sử dụng

+ Hàm bình phương (*Quadratic function*):

$$f_i = \sum_{j=1}^{m} w_{ij} x_j^2 - \theta_i \tag{1.4}$$

+ Hàm hình cầu (*Spherical function*):

$$f_i = \rho^{-2} \sum_{j=1}^{m} \left(x_j - w_{ij} \right)^2 - \theta_i \tag{1.5}$$

trong đó ρ và w_{ij} là bán kính và tâm của hình cầu.

+ Hàm đa thức (*Polynomial function*):

$$f_i = \sum_{j=1}^{m} \sum_{k=1}^{m} w_{ijk} x_j x_k + x_j^{\alpha_j} + x_k^{\alpha_k} - \theta_i \tag{1.6}$$

trong đó w_{ijk} là trọng số kết nối phần tử *PE j* và *PE k* đến *PE i*; α_j *và* α_k là các hệ số thực không đổi.

Các hàm kích hoạt thường sử dụng:

+ Hàm bước nhẩy (*Step function*):

$$a(f) = \begin{cases} 1 & if \quad f \geq 0 \\ 0 & if \quad f < 0 \end{cases} \tag{1.7}$$

+ Hàm dấu (*Hard limiter – threshold function*):

$$a(f) = \text{sgn}(f) = \begin{cases} 1 & if \quad f \geq 0 \\ -1 & if \quad f < 0 \end{cases} \tag{1.8}$$

+ Hàm dốc (*Ramp function*): $$a(f) = \begin{cases} 1 & if \quad f > 1 \\ f & if \quad 0 \leq f \leq 1 \\ 0 & if \quad f < 0 \end{cases}, \tag{1.9}$$

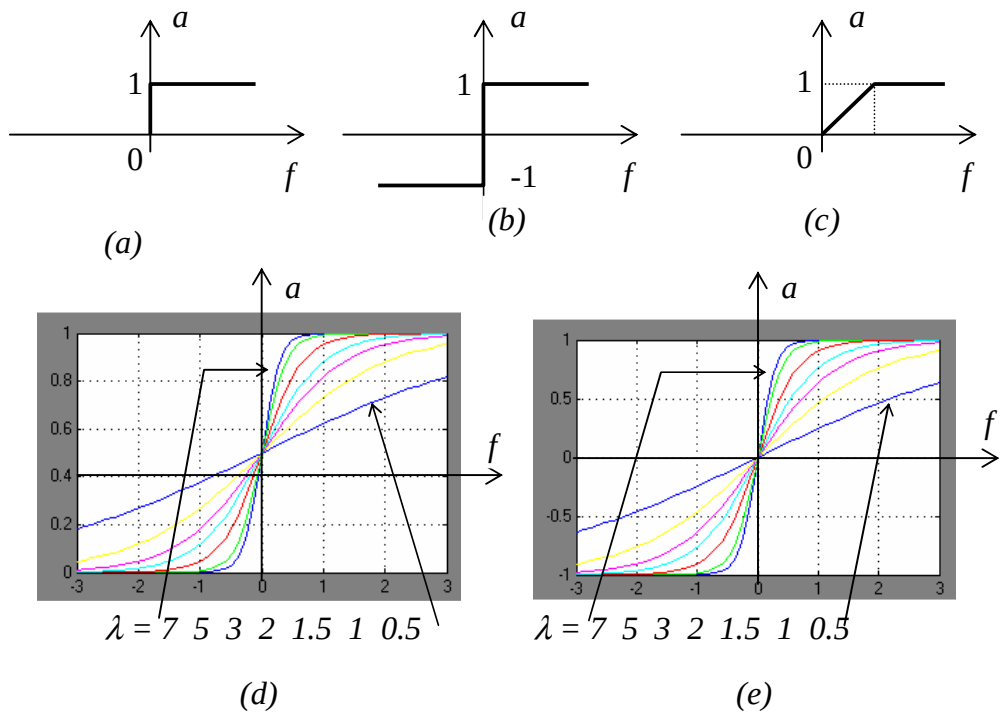
+ Hàm sigmoid đơn cực (*Unipolar sigmoid function*):

$$a(f) = \frac{1}{1 + e^{-\lambda f}} \tag{1.10}$$

+ Hàm sigmoid lưỡng cực (*Bipolar sigmoid function*):

$$a(f) = \frac{2}{1 + e^{-\lambda f}} - 1 \quad \text{trong đó } \lambda > 0 \tag{1.11}$$

Hình 1.4 là thể hiện dưới dạng đồ thị quan hệ *a*(*f*) của 5 hàm kích hoạt trên



Hình 1.4. ***Các hàm kích hoạt: (a) hàm bước nhẩy; (b) hàm dấu; (c) hàm dốc; (d) hàm sigmoid đơn cực; (e) hàm sigmoid lưỡng***

1.2. Phân loại theo cấu trúc mạng nơ ron

1.2.1. Mạng nơ ron 1 lớp:

Hình 1.5 là một loại liên kết đặc thù của mạng nơ ron. Nơ ron có các mối liên hệ đến các nơ ron khác nhờ các trọng số. Một lớp nơ ron là một nhóm các nơ ron mà chúng đều có cùng các trọng số, nhận cùng số tín hiệu đầu vào đồng thời (hình 1.5.1)

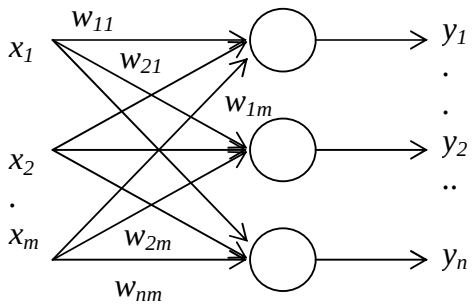
Trong ma trận trọng số, các hàng thể hiện các nơ ron, mỗi hàng thứ j có thể đặt nhãn như một véc tơ $\boldsymbol{w}_j$ của nơ ron thứ j gồm m trọng số w_{ji}. Các trọng số trong cùng một cột thứ j (j = 1, 2, ..., n) đồng thời cùng nhận một tín hiệu đầu vào x_j:

$$\boldsymbol{w}_j = [\, w_{j1}, w_{j2}, ..., w_{jm} \,]. \qquad (1.12)$$

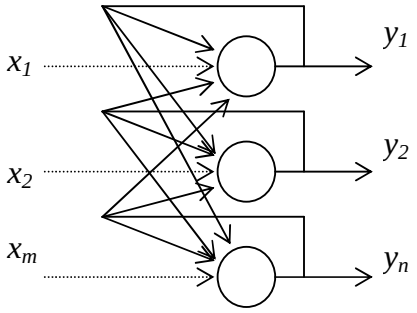
Tại cùng một thời điểm, véc tơ đầu vào:

$$\underline{x} = [\, x_1, x_2, ..., x_m \,] \qquad (1.13)$$

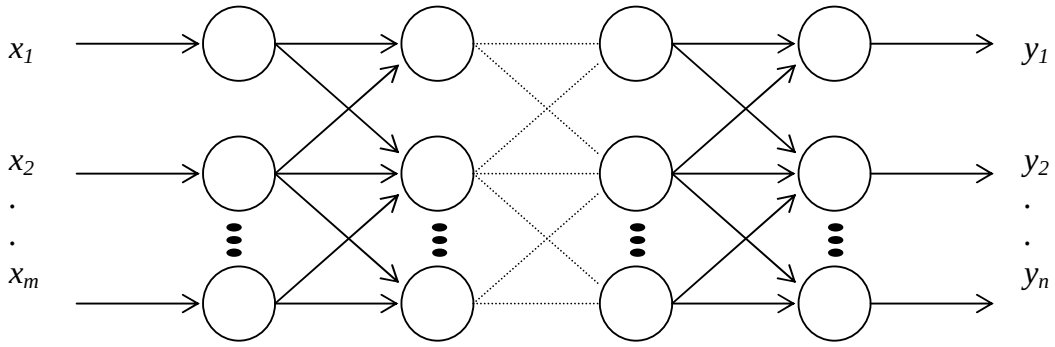
có thể là một nguồn bên ngoài đưa tới mạng.



Hình 1.5.1. Mạng nơ ron 1 lớp.



Hình 1.5.2. Mạng nơ ron hồi quy



Hình 1.5.3. Mạng nơ ron nhiều lớp.

Hình 1.5. ***Một số liên kết đặc thù của mạng nơ ron.***

1.2.2. Mạng nơ ron truyền thẳng nhiều lớp:

Mạng nơ ron nhiều lớp (hình 1.5.3) có các lớp được phân chia thành 3 loại sau:

- Lớp vào là lớp nơ ron đầu tiên nhận tín hiệu vào x_i. Mỗi tín hiệu x_i được đưa đến tất cả các nơ ron của lớp đầu vào, chúng được phân phối trên các trọng số đúng bằng số nơ ron của lớp này. Thông thường, các nơ ron đầu vào không làm biến đổi các tín hiệu vào $x_{i,}$ tức là chúng không có các trọng số hoặc không có các loại hàm chuyển đổi nào, chúng chỉ đóng vai trò phân phối các tín hiệu và không đóng vai trò sửa đổi chúng.

- Lớp ẩn là lớp nơ ron dưới lớp vào, chúng không trực tiếp liên hệ với thế giới bên ngoài như các lớp nơ ron vào và ra.

- Lớp ra là lớp nơ ron tạo các tín hiệu ra cuối cùng.

1.2.3 Mạng nơ ron hồi quy:

Mạng nơ ron hồi quy còn được gọi là mạng phản hồi, là loại mạng tự liên kết thành các vòng và liên kết hồi quy giữa các nơ ron. Mạng nơ ron hồi quy có

trọng số liên kết đối xứng như mạng Hopfield luôn hội tụ về trạng thái ổn định (hình 1.5.2). Mạng BAM thuộc nhóm mạng nơ ron hồi quy, gồm 2 lớp liên kết 2 chiều, không được gắn với tín hiệu vào-ra. Nghiên cứu mạng nơ ron hồi quy có trọng số liên kết không đối xứng sẽ gặp phức tạp nhiều hơn so với mạng truyền thẳng và mạng hồi quy đối xứng.

Đặc điểm cấu trúc mạng nơ ron mà người ta quan tâm đến là: số lượng đầu vào, đầu ra, số lượng các lớp, số lượng nơ ron có trong mỗi lớp, trọng số liên kết trong mỗi lớp và giữa các lớp với nhau.

Căn cứ vào yêu cầu của tín hiệu học, đối với mỗi cấu trúc mạng, mạng nơ ron cần được đánh giá lại giá trị của trọng số liên kết bằng cách thực hiện bài toán tối ưu thông qua các điều kiện thực hiện được gọi là luật học. Mỗi luật học chỉ phù hợp với từng dạng tín hiệu học và cũng chỉ phù hợp với từng kiểu cấu trúc mạng.

1.3. Các luật học:

Học tham số (Parameter Learning): là các tham số về trọng số cập nhật kết nối giữa các nơ ron.

Học cấu trúc (Structure Learning): trọng tâm là sự biến đổi cấu trúc của mạng nơ ron gồm số lượng nút và các mẫu liên kết.

Giả sử ma trận trọng số bao gồm tất cả các phần tử thích ứng của mạng nơ ron. Nhiệm vụ của việc học thông số là bằng cách nào đó, tìm được ma trận chính xác mong muốn từ ma trận giả thiết ban đầu với cấu trúc của mạng nơ ron có sẵn. Để làm được việc đó, mạng nơ ron sử dụng các trọng số điều chỉnh, với nhiều phương pháp học khác nhau có thể tính toán gần đúng ma trận W cần tìm đặc trưng cho mạng. Có 3 phương pháp học:

- Học có giám sát (Supervised Learning): Là quá trình học có tín hiệu chỉ đạo bên ngoài d (hình 1.6).

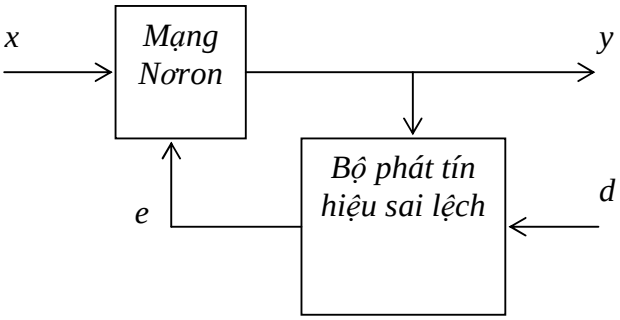
- Học củng cố (Reinforcement Learning): Tín hiệu chỉ đạo d có thể lấy từ bên ngoài môi trường (hình 1.6), nhưng tín hiệu này không được đưa đầy đủ, mà chỉ đưa đại diện một vài *bit* để có tính chất kiểm tra quá trình đúng hay sai. Phương pháp này chỉ là một trường hợp của phương pháp học có giám sát.

- Học không có giám sát (Unsupervised Learning): Là quá trình học không có tín hiệu chỉ đạo từ bên ngoài (hình 1.7). Hình 1.8 mô tả cấu trúc chung của quá trình học của ba phương pháp học đã được nêu trên. Trong đó tín hiệu vào x_j (j = 1, 2, 3, ..., m) có thể được lấy từ đầu ra của các nơ ron khác hoặc có thể được lấy từ bên ngoài. Trọng số của nơ ron thứ i được thay đổi tuỳ theo tín hiệu ở đầu vào mà nó thu nhận, giá trị đầu ra của nó. Dạng tổng quát của luật học trọng số của mạng nơ ron cho biết là gia số của véc tơ w_i là Δw_i tỷ lệ với tín hiệu học r và tín hiệu đầu vào $x(t)$:

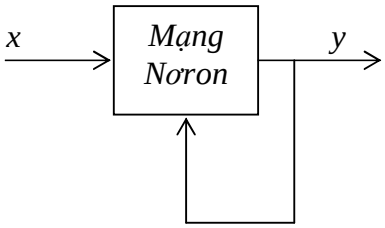
$$\Delta w_i(t) = \eta.r.x(t) \qquad (1.14)$$

η là một số dương còn gọi là hằng số học, xác định tốc độ học, r là tín hiệu học, nó phụ thuộc :

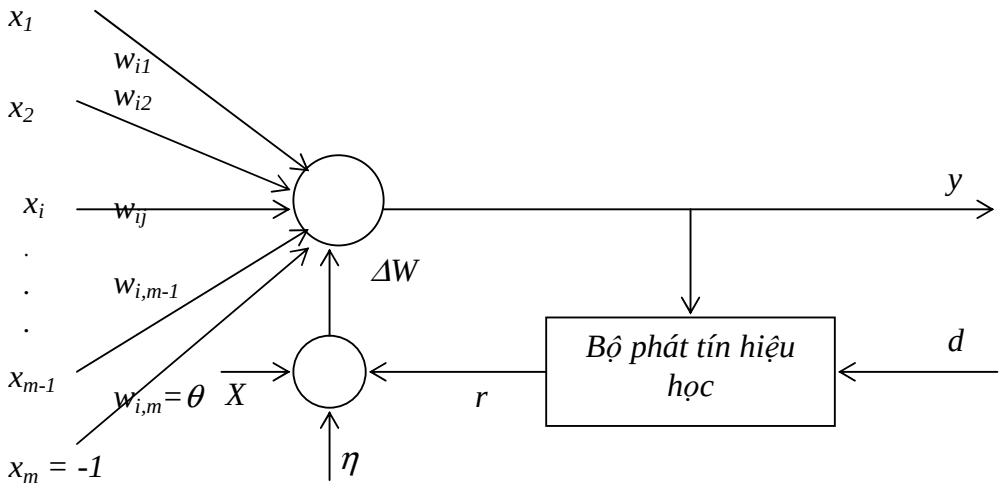
$$r = f_r(w_i, x, d_i). \qquad (1.15)$$



Hình 1.6. ***Học có giám sát.***



Hình 1.7. ***Học không giám sát.***



Hình 1.8. ***Cấu trúc chung của 3 quá trình học.***

Từ (1.14) ta thấy véc tơ trọng số w_i = [w_{i1}, w_{i2}, ..., w_{im}]^T có số gia tỷ lệ với tín hiệu vào x và tín hiệu học r. Véc tơ trọng số ở thời điểm $(t+1)$ được tính:

$$w_i(t+1) = w_i(t) + \eta f_r(w_i(t), x(t), d(t))\, x(t) \qquad (1.16)$$

Phương trình liên quan đến sự biến đổi trọng số trong mạng nơ ron rời rạc và tương ứng với sự thay đổi trọng số trong mạng nơ ron liên tục theo biểu thức:

$$\frac{dw_i(t)}{dt} = \eta.r.x(t) \qquad (1.17)$$

Một số thuật toán học có giám sát và không giám sát được phát triển dựa vào luật cập nhật trọng số 1.17, sự khác biệt chính giữa các thuật toán học có giám sát và không giám sát này là tín hiệu học r sẽ được phát ra để cập nhật trọng số như thế nào.

Luật Hebb là một ví dụ điển hình, với tiếp cận trọng số được điều chỉnh phù hợp với quan hệ trước –sau. Cụ thể luật học Hebb định nghĩa:

$$r \triangleq a(w_i^T x) = y_i \qquad (1.18)$$

với a(.) là hàm kích hoạt. Tóm lại trong luật Hebb tín hiệu lỗi r đơn giản là đầu ra của nơ ron hiện thời. Theo công thức 1.13 ta có:

$$\Delta w_i = \eta.a(w_i^T x)x = \eta.y_i.x \qquad (1.19)$$

Từ đó các thành phần của của véc tơ trọng số w_i được cập nhật như sau

$$\Delta w_{ij} = \eta.a(w_i^T x)x_j = \eta.y_i.x_j \qquad i = 1,\ldots,n;\ j = 1,\ldots, m \qquad (1.20)$$

Luật Hebb là luật học không giám sát cho mạng truyền thẳng, nó chỉ sử dụng các đầu vào và các đầu ra hiện thời để cập nhật trọng số. Trong luật học này các trọng số cần được khởi tạo ngẫu nhiên với giá trị xấp xỉ bằng không trước khi học

Ví dụ 1.1: Xét mạng chỉ có một đơn vị xử lý như hình 1.3 với hàm kích hoạt *Sign*(). Có 4 đầu vào x_1, x_2, x_3, x_4. Véc tơ trọng số w = (w_1, w_2, w_3, w_4). Giả sử ta huấn luyện cho mạng với 3 véc tơ mẫu sau:

$$x^{(1)} = \begin{pmatrix} 1 \\ 1.5 \\ 0.5 \\ 0 \end{pmatrix},\ x^{(2)} = \begin{pmatrix} -0.5 \\ 1 \\ 0 \\ 1.5 \end{pmatrix},\ x^{(3)} = \begin{pmatrix} -1 \\ 0 \\ -1 \\ -0.5 \end{pmatrix}$$

Véc tơ trọng số được khởi tạo như sau: $w^{(1)} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}$ và hằng số học được cho là $\eta = 1$. Theo luật Hebb ta có các bước kết quả sau:

Bước 1: Đưa véc tơ mẫu $x^{(1)}$ được đưa vào để cập nhật véc trọng số

$$w^{(2)} = w^{(1)} + Sign((w^{(1)})^T x^{(1)})x^{(1)} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix} + Sign(0.5)\begin{pmatrix} 1 \\ 1.5 \\ 0.5 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1.5 \\ -0.5 \\ 0 \end{pmatrix}$$

Bước 2: Đưa véc tơ mẫu $x^{(2)}$ được đưa vào để cập nhật véc trọng số

$$w^{(3)} = w^{(2)} + Sign((w^{(2)})^T x^{(2)})x^{(2)} = \begin{pmatrix} 2 \\ 1.5 \\ -0.5 \\ 0 \end{pmatrix} + Sign(0.5)\begin{pmatrix} 0.5 \\ 1 \\ 0 \\ 1.5 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 2.5 \\ -0.5 \\ 1.5 \end{pmatrix}$$

Bước 3: Đưa véc tơ mẫu $x^{(3)}$ được đưa vào để cập nhật véc trọng số

$$w^{(4)} = w^{(3)} + Sign((w^{(3)})^T x^{(3)})x^{(3)} = \begin{pmatrix} 1.5 \\ 2.5 \\ -0.5 \\ 1.5 \end{pmatrix} + Sign(-0.75)\begin{pmatrix} -1 \\ 0 \\ -1 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 2.5 \\ 2.5 \\ 0.5 \\ 2 \end{pmatrix}$$

Có thể nhận thấy $Sign((w^{(4)})^T x^{(1)} = Sign((w^{(4)})^T x^{(2)} = 1$ và $Sign((w^{(4)})^T x^{(3)} = -1$

Các luật cập nhật trọng số cho bởi công thức (1.13), như luật ….. sẽ được bàn tới trong phần sau

Mạng nơ ron nhân tạo có các tính chất sau đây:

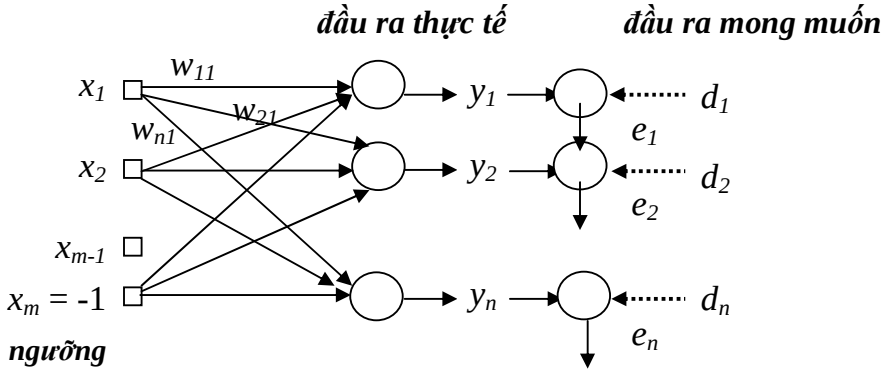
- *Là hệ phi tuyến.*
- *Là hệ xử lý song song.*
- *Là hệ học và thích nghi:* mạng được huấn luyện từ số liệu quá khứ, có khả năng tự chỉnh khi số liệu đầu vào bị mất.
- *Là hệ nhiều biến, nhiều đầu vào, nhiều đầu ra (MIMO)*

2. Mạng nơ ron truyền thẳng

Trong phần này chúng ta sẽ nghiên cứu mạng một lớp và mạng nhiều lớp truyền thẳng với luật học có giám sát

2.1. Mạng Perceptron một lớp đơn

Hình 2.1 là một ví dụ về một mạng perceptron đơn. Cho các véc tơ mẫu đầu ra $d^{(k)} = [d_1^{(k)}, d_2^{(k)}, \ldots, d_n^{(k)}]^T$ tương ứng với các véc tơ mẫu đầu vào đầu vào là $x^{(k)} = [x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}]^T$, $k = 1, \ldots, p$ trong đó m là số đầu vào, n là số đầu ra, p là số cặp mẫu



Hình 2.1. ***Mạng perceptron đơn***

Đầu ra thực tế theo cấu trúc:

$$y_i^{(k)} = a(w_i^T x^{(k)}) = a\left(\sum_{j=1}^{m} w_{ij} x_j^k\right) = d_i^k,\ i = 1, \ldots, n;\ k = 1, \ldots, p \quad (2.1)$$

với $w_i^T = [w_{i1}, x_{i2}, \ldots, w_{im}]^T$ là véc tơ trọng số liên kết với nơ ron thứ i và $a(.)$ hàm dấu hoặc hàm *sigmond*

2.1.2. Luật học Perceptron

Với các perceptron cùng với .. đầu ra mong muốn $d_i^{(k)}$ có thể lấy các giá trị -1, +1. Khi đó công thức (2.1) trở thành:

$$y_i^{(k)} = Sign(w_i^T x^{(k)}) = d_i^{(k)} \text{ với } i = 1, \ldots, n;\ k = 1, \ldots, p \quad (2.2)$$

Điều này cho thấy véc tơ trọng số w_i của nơ ron thứ i phải được chọn sao cho tích vô hướng của mẫu $x^{(k)}$ và nó có giá trị giống như $d_i^{(k)}$. Siêu phẳng $w^T x = 0$ chia tập dữ liệu thành 2 phần một phần gồm các dữ liệu có đầu ra mong muốn dương và một phần gồm các dữ liệu có đầu ra mong muốn âm. Như vậy nó có khả năng phân lớp dữ liệu. Khái niệm phân lớp sẽ được làm rõ thông qua ví dụ sau:

Ví dụ 2.1: Chúng ta thiết kế một mạng perceptron đơn với đơn vị xử lý có hàm kích hoạt tuyến tính có thể phân chia sáu mẫu trong không gian đầu vào vào hai lớp như sau:

$\{[-1, 0]^T, [-1.5, -1]^T, [-1, -2]^T\}$: lớp 1

$\{[2, 0]^T, [2.5, -1]^T, [1, -2]^T\}$: lớp 2

Với sáu mẫu trên ta luôn chỉ ra được phương trình ranh giới:

$g(x) = -2x_1 + x_2 + 2 = 0$

Đường này chia không gian đầu vào thành hai miền kề nhau, mỗi miền là một nửa mặt phẳng thoả $g(x) > 0$ và $g(x) < 0$

Chúng ta sẽ xây dựng một perceptron đơn sao cho nó đáp ứng là a+1 khi mẫu ở lớp 1 được đưa qua mạng và a-2 khi mẫu ở lớp 2 được đưa qua

Giả sử đơn vị xử lý nhận tín hiệu ngưỡng, ta có: $y^{(k)} = sign(w_1 x_1^{(k)} + w_2 x_2^{(k)} - \theta)$, với θ là ngưỡng và $(x_1^{(k)}, x_2^{(k)})$, k = 1, .., 6 là sáu mẫu đầu vào. Từ phương trình $g(x)$ trên ta có $w_1 = -2$, $w_2 = 1$ và θ = -2 và mạng perceptron đơn được thiết kế như hình vẽ

Khi đó các mẫu đầu vào hai chiều được gia tăng thành ba chiều:

$$x^{(1)} = \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix},\ x^{(2)} = \begin{pmatrix} -1.5 \\ -1 \\ -1 \end{pmatrix},\ x^{(3)} = \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix},\ x^{(4)} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix},\ x^{(5)} = \begin{pmatrix} 2.5 \\ -1 \\ -1 \end{pmatrix},\ x^{(6)} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$ và

đầu ra mong muốn: $d^{(1)} = d^{(2)} = d^{(3)} = +1$, $d^{(4)} = d^{(5)} = d^{(6)} = -1$

Mạng perceptron đơn như thiết kế sẽ hoạt động như sau: $y^{(k)} = sign(w^T x^{(k)}) = d^{(k)}$, với $w = [w_1, w_2, w_3] = [-2, 1, -2]$

Bài toán phân lớp tuyến tính như trên đã minh hoạ thiết kế một mạng perceptron đơn, nhiệm vụ tiếp theo là phải xây dựng một thủ tục huấn luyện để có thể tìm ra các trọng số của mạng, từ đó tìm ra mặt phẳng phân lớp, thủ tục này được xây dựng như sau:

Xét luật học tổng quát (9.14) với tham số học:

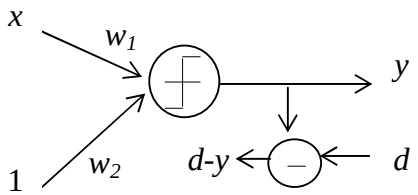
$$r \triangleq d_i - y_i \qquad (2.3)$$

với $y_i = Sign(w_i^T x)$ và d_i là đầu ra thực tế và đầu ra mong muốn của nơ ron thứ i. Giả sử d_i nhận các giá trị -1, +1, sử dụng (1.13) ta có

$$\Delta w_{ij} = \eta[d_i - sign(w_i^T x)]x_j = \begin{cases} 2\eta d_i x_j & if\ y_i \neq d_i \\ 0 & otherwise \end{cases} \text{ với } j = 1,\ldots, m \qquad (2.4)$$

Từ đây ta thấy trọng số chỉ được điều chỉnh khi đầu ra thực tế y_i khác d_i. Các trọng số ban đầu được khởi tạo bất kỳ

Mạng perception có khả năng phân lớp dữ liệu, ví dụ: Cho mạng perceptron đơn giản chỉ có một nơ ron, hình 2.2



Hình 2.2. ***Mạng perceptron đơn giản dùng cho bài toán phân lớp***

Không gian đầu vào gồm các giá trị:

$x^{(1)} = 0.5$, $x^{(2)} = 2$, $x^{(3)} = -1$, $x^{(4)} = -2$

$d^{(1)} = d^{(3)} = +1$, $d^{(2)} = d^{(4)} = -1$

Như vậy các mẫu tương ứng được dùng để huấn luyện mạng là:

$$x^{(1)} = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix},\ x^{(2)} = \begin{pmatrix} 2 \\ -1 \end{pmatrix},\ x^{(3)} = \begin{pmatrix} -1 \\ -1 \end{pmatrix},\ x^{(4)} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

Để thuận tiện ta chọn $\eta = 0.5$ và khởi tạo véc tơ trọng số $w = [-2, 1.5]$

Quá trình huấn luyện diễn ra như sau

Bước 1: Khi $x^{(1)}$ được đưa vào, đầu ra thực tế và các trọng số được xác định:

$$y^{(1)} = sign\left([-2, 1.5]\begin{pmatrix} 0.5 \\ -1 \end{pmatrix}\right) = -1 \neq d^{(1)},\ w^{(2)} = w^{(1)} + x^{(1)} = \begin{pmatrix} -1.5 \\ 0.5 \end{pmatrix}$$

Bước 2: Khi $x^{(2)}$ được đưa vào, đầu ra thực tế và các trọng số được xác định:

$$y^{(2)} = sign\left([-1.5, 0.5]\begin{pmatrix} -1 \\ -1 \end{pmatrix}\right) = +1 \neq d^{(2)},\ w^{(3)} = w^{(2)} - x^{(2)} = \begin{pmatrix} -0.5 \\ 1.5 \end{pmatrix}$$

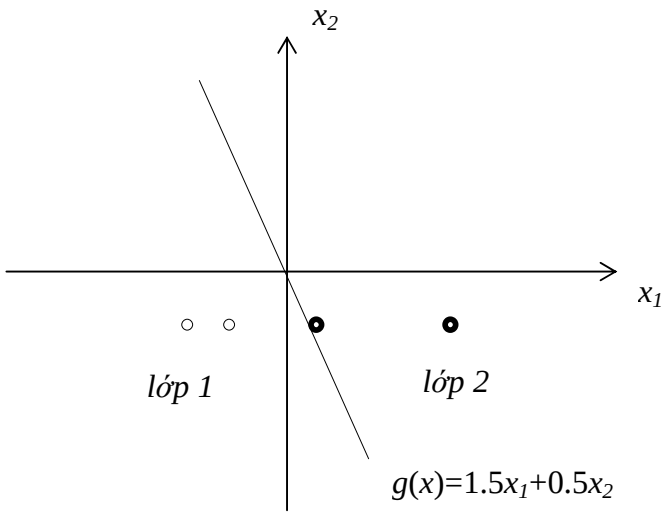
Bước 3: Khi $x^{(3)}$ được đưa vào, đầu ra thực tế và các trọng số được xác định:

$$y^{(3)} = sign\left([-0.5, 1.5]\begin{pmatrix} 2 \\ -1 \end{pmatrix}\right) = -1 \neq d^{(3)},\ w^{(4)} = w^{(3)} + x^{(3)} = \begin{pmatrix} 1.5 \\ 0.5 \end{pmatrix}$$

Bước 4: Cuối cùng khi $x^{(4)}$ được đưa vào, đầu ra thực tế và các trọng số được xác định:

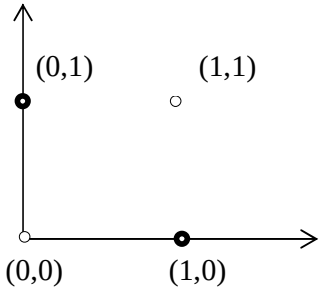
$$y^{(4)} = sign\left([1.5, 0.5]\begin{pmatrix}-2\\-1\end{pmatrix}\right) = -1 = d^{(4)},\ w^{(5)} = w^{(4)}$$

Tiếp tục sử dụng các dữ liệu: $x^{(5)} = x^{(1)}$, $x^{(6)} = x^{(2)}$, $x^{(7)} = x^{(3)}$, $x^{(8)} = x^{(4)}$, đưa vào mạng và cập nhật lại trọng số ta thấy $w^{(8)} = w^{(7)} = w^{(6)} = w^{(4)}$, tóm lại $w^{(4)}$, là kết quả của bài toán, hình ?? cho thấy đường phân lớp được được xác định nhờ mạng nơ ron:



Hình 2.3. ***Đường thẳng phân lớp được xác định nhờ mạng***

Ở trên chúng ta đã đề cập tới khả năng phân lớp của mạng perceptron, tuy nhiên khả năng này phụ thuộc vào tập mẫu, ví dụ: xét bài toán XOR cho bởi hình 2.4



Hình 2.4. ***Bài toán XOR***

Rõ ràng ở đây không thể tồn tại một đường thẳng chia tập dữ liệu thành 2 phần thoả điều kiện các phần tử ở phần 1 có đáp ứng bằng 1, và các phần tử ở phần 2 có đáp ứng bằng 0. Do đó ta không thể biểu diễn hàm XOR bằng một mạng perceptron đơn cùng với hàm kích hoạt tuyến tính

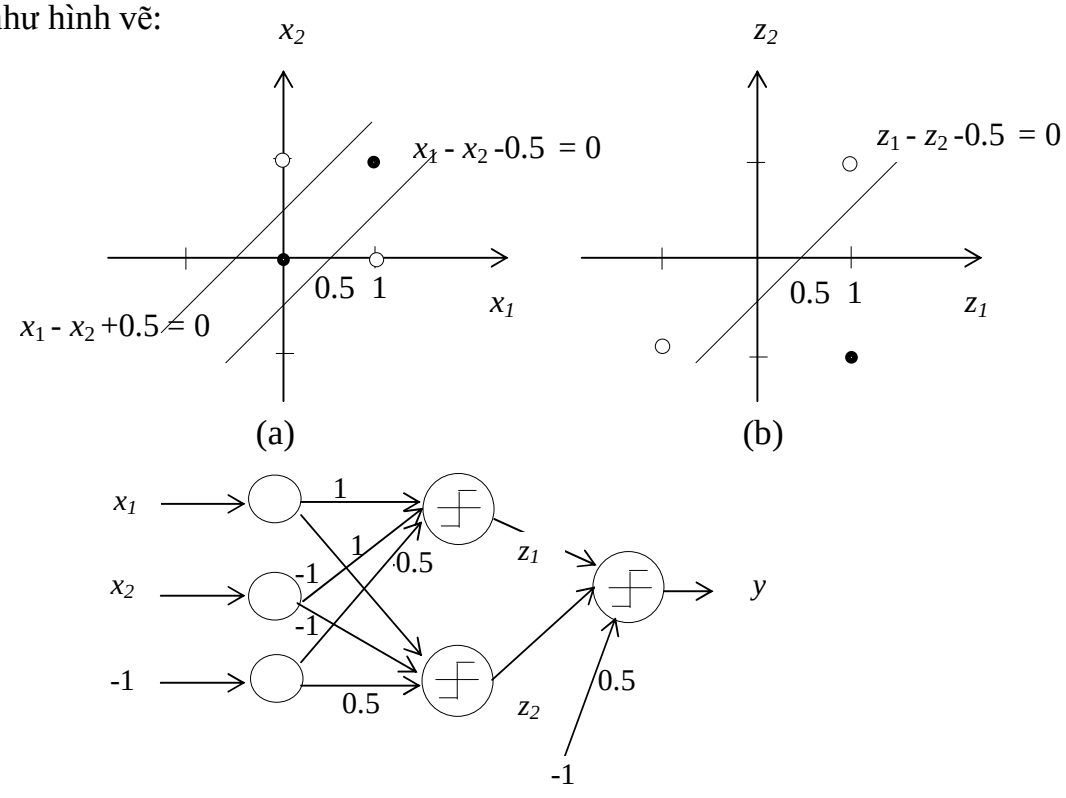
2.2. Các mạng truyền thẳng nhiều lớp

Mạng một lớp perceptron được dùng để phân lớp và xấp xỉ các hàm, tuy nhiên nó phụ thuộc vào dữ liệu mẫu của bài toán, bài toán XOR là một ví dụ (không có mạng perceptron tương ứng). Để giải quyết điều đó người ta đưa ra một cấu trúc mạng truyền thẳng nhiều lớp, ví dụ sau sẽ minh chứng điều đó

Ví dụ 2.3: Xét bài toán XOR, với các mẫu đầu vào và đầu ra mong muốn

$$\left(x^{(1)}=\begin{pmatrix}0\\0\end{pmatrix}, d^{(1)}=1\right);\ \left(x^{(2)}=\begin{pmatrix}0\\1\end{pmatrix}, d^{(2)}=-1\right);\left(x^{(3)}=\begin{pmatrix}1\\0\end{pmatrix}, d^{(1)}=-1\right);\ \left(x^{(4)}=\begin{pmatrix}1\\1\end{pmatrix}, d^{(2)}=1\right)$$

Rõ ràng bốn mẫu vào này không thể phân chia tuyến tính trong không gian đầu vào. Để phân chia tập mẫu này ta phải sử dụng đến hai đường thẳng như hình vẽ:



Hình 2.5. ***Mạng perceptron đa lớp cho bài toán XOR***

Như vậy mỗi không gian con bao gồm các mẫu các mẫu của các lớp tương ứng của bài toán, hai đường thẳng được chọn là:

$x_1 - x_2 + 0.5 = 0$ và $x_1 - x_2 - 0.5 = 0$

Ta sử dụng hai LTU để thực hiện việc trên, các đầu ra tương ứng của chúng là:

$z_1 = Sign(x_1 - x_2 + 0.5)$ và $z_1 = Sign(x_1 - x_2 - 0.5)$

Các đơn vị này sẽ được đặt trong tầng ẩn của một mạng 3 lớp hình 2.5(c), chú ý lớp một chỉ tiếp nhận đầu vào chứ không đóng vai trò xử lý

Như vậy các mẫu gốc sẽ có các ảnh tương ứng sau khi đi qua lớp ẩn này, các ảnh này là:

$$\left(z^{(1)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, d^{(1)} = 1\right); \left(z^{(2)} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, d^{(1)} = -1\right)$$

$$\left(z^{(3)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, d^{(1)} = -1\right); \left(z^{(4)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, d^{(1)} = 1\right)$$

Sự thay đổi của các mẫu huấn luyện trong không gian ảnh thể hiện trong hình ???, các mẫu $x^{(1)}$, $x^{(4)}$ trở nên trùng khớp trong không gian ảnh, nhờ đó có thể tiến hành quá trình phân lớp tuyến tính trong không gian ảnh. Đường biên chia ba mẫu trên thành 2 lớp có thể chọn tuỳ ý, ví dụ $z_1 - z_2 - 0.5 = 0$. Đây chính là đầu vào của nơ ron ở tầng xuất, và đầu ra tương ứng là:

$y = Sign(z_1 - z_2 - 0.5)$

Cuối cùng ta có mạng nơ ron 3 lớp để giải bài toán phân lớp XOR, hình vẽ 2.5(c)

2.1. Lan truyền ngược sai số (Back Propagation)

Thuật toán học lan truyền ngược là một thuật toán trong lịch sử phát triển mạng nơ ron nhân tạo, thuật toán được ứng dụng cho mạng truyền thẳng nhiều lớp

Mạng nơ ron sử dụng thuật toán huấn luyện lan truyền ngược lan truyền ngược được gọi là mạng BP (Back Propagation Network)

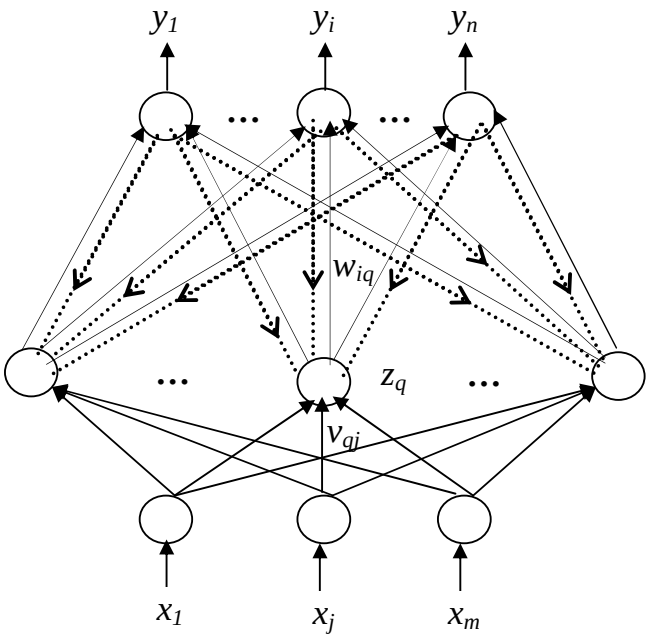
Giả sử tập mẫu dùng để huấn luyện mạng gồm p mẫu $\{x^k, d^k\}$, k=1, 2,.., p thủ tục BP xây dựng một chuỗi p các hàm sai số

$$E_k(W) = \frac{1}{2}\left|y^k - d^k\right|^2 = \frac{1}{2}\sum_{j=1}^{m}\left(d^k{}_j - y^k{}_j\right)^2,$$

trong đó: m - số rơ ron đầu ra; $d^k{}_j$ - thành phần thứ j của véc tơ ra mong muốn d^k, $y^k{}_j$ - thành phần thứ j của véc tơ ra y^k do lan truyền đầu vào x^k. Các hàm sai số này lần lượt được tối thiểu trong không gian các trọng số. Giá trị hàm sai số E_k đối với một mẫu được tính toán dựa trên giá trị các trọng số hiện tại. Các giá trị trọng số này sau đó được hiệu chỉnh và trở thành các giá trị trọng số hiện tại để tính giá trị hàm sai số tiếp theo E_{k+1}. Dễ nhận thấy, cách làm này có khả năng tạo ra sự dao động trong quá trình hiệu chỉnh các trọng số. Kết quả hiệu chỉnh hiện tại có thể làm hỏng kết quả hiệu chỉnh ở các lần trước đó.

Giải pháp thay thế là xây dựng một hàm sai số duy nhất bằng cách tích luỹ các sai số $E = \sum_{i=1}^{n} E_k(W)$. Các trọng số của mạng nơ ron chỉ được hiệu chỉnh sau khi tất cả các véc tơ tín hiệu vào có trong tập mẫu được lan truyền qua mạng nơ ron.

Xét một mạng truyền thẳng ba lớp như hình vẽ



Hình 2.6. ***Mạng truyền thẳng ba lớp lan truyền ngược sai số***

Quá trình huấn luyện tiến hành như sau:

+ Trước tiên ta xét lan truyền tín hiệu qua từng lớp mạng: lớp nhập, lớp ẩn và lớp xuất

Giả sử tín hiệu đưa vào mạng nơ ron $x = (x_1, x_2, \ldots, x_m)$ với m là số nơ ron trong lớp nhập, l là số nơ ron trong lớp ẩn và n là số nơ ron trong lớp xuất. Các nơ ron trong lớp nhập nhận và truyền tín hiệu sang lớp ẩn. Đầu vào các nơ ron (nút) trong lớp ẩn thứ nhất có dạng:

$$net_q = \sum_{j=1}^{m} v_{qj} x_j \tag{2.1}$$

trong đó w_{qj} là trọng số liên kết từ nút j của lớp nhập đến nút q của lớp ẩn. Đầu ra z_q của lớp ẩn nhận được:

$$z_q = a(net_q) = a(\sum_{j=1}^{m} v_{qj} x_j) \tag{2.2}$$

với $a(net_q)$ là hàm kích hoạt của các nơ ron trong lớp ẩn. Đầu vào của nút i ở lớp xuất sẽ nhận được:

$$net_i = \sum_{q=1}^{l} w_{iq} z_q = \sum_{q=1}^{l} w_{iq} a(\sum_{j=1}^{n} v_{qj} x_j) \tag{2.3}$$

Đầu ra của nút i ở lớp xuất nhận được

$$y_i = a(net_i) = a(\sum_{q=1}^{l} w_{iq} a(\sum_{j=1}^{m} v_{qj} x_j)) \tag{2.4}$$

+ Sau khi lan truyền thẳng tín hiệu qua mạng, sai số của tín hiệu ra sẽ được lan truyền ngược để điều chỉnh các trọng số, mục tiêu ở đây là cực tiểu sai số giữa tín hiệu ra $y = (y_1, y_2, \ldots, y_n)$ của mạng nơ ron và tín hiệu mong muốn $d = (d_1, d_2, \ldots, d_n)$, hàm mục tiêu có dạng:

$$\begin{aligned} E(w) &= \frac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}[d_i - a(net_i)]^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\left[d_i - a(\sum_{q=1}^{l} w_{iq} z_q)\right]^2 \end{aligned} \tag{2.5}$$

Theo phương pháp giảm gradient, trọng số liên kết giữa lớp ẩn và lớp xuất được điều chỉnh như sau:

$$\Delta w_{iq} = -\eta \frac{\partial E}{\partial w_{iq}} \tag{2.6}$$

ở đây η là hệ số học của mạng nơ ron

Sử dụng (2.3), (2.5) ta có:

$$\Delta w_{iq} = -\eta \left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial w_{iq}}\right] = \eta[d_i - y_i][a'(net_i)][z_q] = \eta \delta_{oi} z_q \tag{2.7}$$

trong đó δ_{oi} là tín hiệu sai số, *i* là số thứ tự của nút *i* của lớp xuất, tín hiệu sai số được định nghĩa như sau:

$$\delta_{oi} = -\frac{\partial E}{\partial net_i} = -\left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right] = [d_i - y_i][a'(net_i)] \tag{2.8}$$

với net_i là tổng vào của nơ ron thứ *i* của lớp xuất và $a'(net_i) = \partial a(net_i) / net_i$

Trọng số liên kết giữa lớp ẩn và lớp nhập cũng được điều chỉnh theo phương pháp giảm gradient, như sau:

$$\begin{aligned} \Delta v_{qj} &= -\eta\left[\frac{\partial E}{\partial v_{qj}}\right] = -\eta\left[\frac{\partial E}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qj}}\right] \\ &= -\eta\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qj}}\right] \end{aligned} \tag{2.9}$$

Từ (2.5) ta thấy với mỗi thành phần sai số $[d_i - y_i]$, i = 1, 2, …, *n* là một hàm của z_q và ta có:

$$\Delta v_{qj} = \eta \sum_{i=1}^{n} [(d_i - y_i) a'(net_i) w_{iq}] a'(net_q) x_j \tag{2.10}$$

Sử dụng (2.8), (2.10) được viết lại như sau:

$$\Delta v_{qj} = \eta \sum_{i=1}^{n} [\delta_{oi} w_{iq}] a'(net_q) x_j = \eta \delta_{hq} x_j \tag{2.11}$$

trong đó δ_{hq} là tín hiệu lỗi của nơ ron thứ *q* ở tầng ẩn và được định nghĩa:

$$\delta_{hq} = -\frac{\partial E}{\partial net_q} = -\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right] = a'(net_q) \sum_{i=1}^{n} \delta_{oi} w_{iq} \tag{2.12}$$

Khi hàm kích hoạt là hàm *sigmoid* các công thức (2.8), (2.12) trở thành:

$$\delta_{oi} = \frac{1}{2}(1 - y_i^2)[d_i - y_i] \quad (2.13)$$

$$\delta_{hq} = \frac{1}{2}(1 - z_q^2)\sum_{i=1}^{n} \delta_{oi} w_{qi} \quad (2.14)$$

Thuật toán có thể khái quát như sau:

Algorithm BP: Back-propagation Learning Rule

Xét mạng nơ ron với Q lớp lan truyền thẳng, q = 1, 2, …, Q và ${}^{q}net_i$ và ${}^{q}y_i$ là đầu vào và đầu ra của nơ ron thứ i trong lớp q tương ứng. Mạng có m nút nhập, n nút xuất. Gọi ${}^{q}w_{ij}$ là trọng số liên kết từ nơ ron j lớp q-1 đến nơ ron i lớp q.

Input: Tập mẫu huấn luyện $\{(x^{(k)}, d^{(k)})\}$, k = 1, 2, …, p, ở đây véc tơ đầu vào được gia tăng với thành phần trước là –1, nghĩa là $x_{m+1}^{k} = -1$.

***Bước* 0.** (khởi tạo): chọn $\eta > 0$ và E*max* (sai số lớn nhất), khởi tạo trọng số với giá trị ngẫu nhiên nhỏ. Đặt E = 0 và k = 1.

***Bước* 1**. (vòng luyện): sử dụng mẫy đầu vào thứ k cho lớp nhập (q = 1)

$${}^{q}y_i = {}^{1}y_i = x_i^{(k)} \quad (2.15)$$

***Bước* 2**. (Lan truyền thẳng tín hiệu): lan truyền thẳng tín hiệu qua mạng theo công thức:

$${}^{q}y_i = a({}^{q}net_i) = f\left(\sum_{j=1}^{m} {}^{q}w_{ij} \times {}^{q-1}y_{ij}\right) \quad (2.16)$$

***Bước* 3**. (Tính sai số đầu ra): tính sai số và tín hiệu sai số ${}^{Q}\delta_i$ cho lớp xuất:

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(d_i^{(k)} - {}^{Q}y_i\right)^2 + E \quad (2.17)$$

$${}^{Q}\delta_i = \frac{1}{2}\left(d_i^{(k)} - {}^{Q}y_i\right)^2 a\left({}^{Q}net_i\right) \quad (2.18)$$

***Bước* 4**. (Lan truyền ngược sai số): lan truyền ngược sai số để điều chỉnh trọng số và tính tín hiệu sai số ${}^{q-1}\delta_i$ cho lớp trước.

$$\Delta^{q}w_{ij} = \eta\, {}^{q}\delta_i\, {}^{q-1}y_j \text{ và } {}^{q}w_{ij}^{new} = {}^{q}w_{ij}^{old} + \Delta^{q}w_{ij} \quad (2.19)$$

$${}^{q-1}\delta_i = a'({}^{q-1}net_i)\sum_{j=1}^{m} {}^{q-1}w_{ji}\, {}^{q}\delta_j \text{ với } q = Q, Q\text{-}1, \ldots, 2. \quad (2.20)$$

***Bước* 5**. (Lặp một chu kỳ): kiểm tra tập dữ liệu huấn luyện đã quay hết một vòng. Nếu $k < p$ thì $k = k+1$ và quay lại bước 1; trường hợp khác về bước 6.

***Bước* 5**. (Kiểm tra tín hiệu sai số): kiểm tra tín hiệu sai số, nếu $E < E_{max}$ thì kết thúc vòng luyện và đưa ra bộ trọng số cuối cùng; trường hợp khác cho $E = 0$, $k = 1$ và quay lại bước 1 tiến hành chu kỳ luyện mới.

End BP

Thuật toán trên là phương pháp huấn luyện gia tăng (incremental training) khi điều chỉnh trọng số, nghĩa là các trọng số được điều chỉnh ngay sau khi có một mẫu luyện. Một phương pháp khác là luyện dạng gói (batch training), ở đó các trọng số chỉ thay đổi sau khi đã có tất cả các mẫu luyện.

Mạng nơ ron truyền thẳng có từ một lớp ẩn trở lên có thể xấp xỉ một hàm, trong [Hornik et al., 1989] đã phát biểu và chứng minh định lý dưới đây:

Định lý: Mạng nơ ron truyền thẳng với các lớp ẩn sử dụng hàm kích hoạt nén, tuyến tính hoặc hàm đa thức mở rộng có thể xấp xỉ gần đúng một hàm bất kỳ với độ chính xác tuỳ ý miễn là có đủ số nơ ron ẩn. Hàm a: R→[0, 1] (hoặc [-1, 1]) là hàm nén nếu nó đồng biến và $\lim_{\lambda\to\infty} a(\lambda) = 1,\ \lim_{\lambda\to-\infty} a(\lambda) = 0$ (hoặc –1) với λ là tham số của hàm

Ví dụ: Xét một mạng BP đơn giản như hình vẽ 2.7, mỗi thành phần xử lý của mạng sử dụng một hàm kích hoạt sigmond lưỡng cực $y = a(net) = 1/(1+e^{-\lambda net})$, với net là tổng đầu vào của nơ ron, khi $\lambda = 1$ ta có: $a'(net) = y(1-y)$

$$\delta_9 = a'(net_9)(d - y_9) = y_9(1-y_9)(d-y_9)$$

$$\delta_6 = a'(net_6)\sum_{i=9}^{9} w_{i6}\delta_i = y_6(1-y_6)w_{96}\delta_9$$

$$\delta_7 = a'(net_7)\sum_{i=9}^{9} w_{i7}\delta_i = y_7(1-y_7)w_{97}\delta_9$$

$$\delta_3 = a'(net_3)\sum_{r=6}^{7} w_{r3}\delta_r = y_3(1-y_3)(w_{63}\delta_6 + w_{73}\delta_7)$$

$$\delta_4 = a'(net_4)\sum_{r=6}^{7} w_{r4}\delta_r = y_4(1-y_4)(w_{64}\delta_6 + w_{74}\delta_7)$$

Luật cập nhật trọng số

$$\Delta w_{96} = \eta\delta_9 y_6,\ \Delta w_{97} = \delta_9 y_6,\ \Delta w_{98} = \eta\delta_9 y_8 = -\eta\delta_9$$

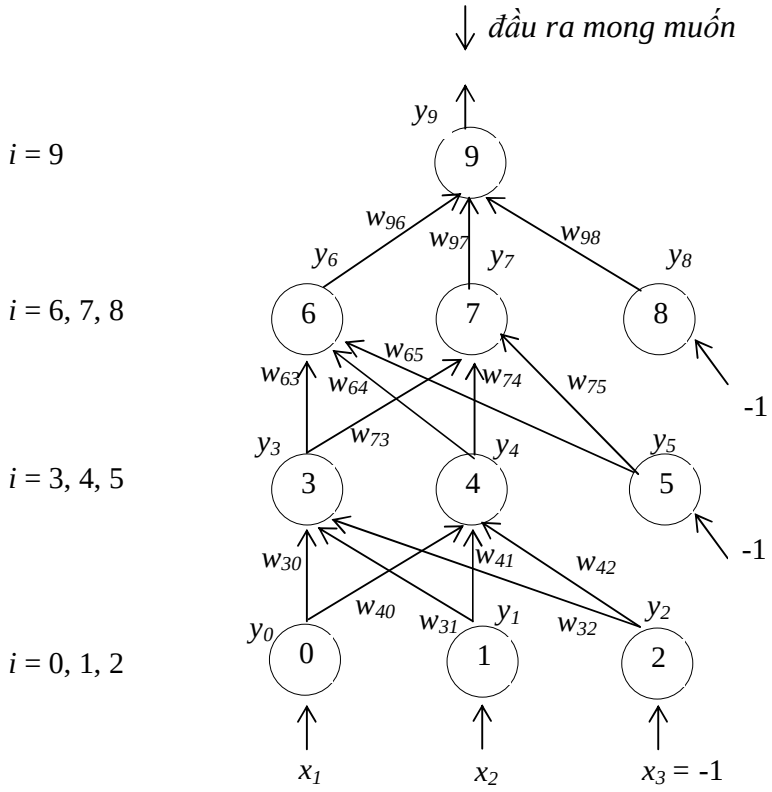
$$\Delta w_{63} = \eta\delta_6 y_3,\ \Delta w_{64} = \delta_6 y_4,\ \Delta w_{65} = \eta\delta_6 y_5 = -\eta\delta_6$$

$$\Delta w_{73} = \eta\delta_7 y_3,\ \Delta w_{74} = \delta_7 y_4,\ \Delta w_{75} = \eta\delta_7 y_5 = -\eta\delta_7$$

$$\Delta w_{30} = \eta\delta_3 y_0,\ \Delta w_{31} = \delta_3 y_1,\ \Delta w_{32} = \eta\delta_3 y_2 = -\eta\delta_3$$

$$\Delta w_{40} = \eta\delta_4 y_0,\ \Delta w_{41} = \delta_4 y_1,\ \Delta w_{42} = \eta\delta_4 y_2 = -\eta\delta_4$$

Ở trên các trọng số được điều chỉnh theo từng mẫu đơn, gọi là *learning step*. Sau khi các mẫu được đưa vào hết thì một thời kỳ được thiết lập (epoch). Các thời kỳ huấn luyện được tiếp tục cho đến khi đạt tới điều kiện dừng của mạng



Hình 2.7. ***Một ví dụ về mạng nơ ron truyền thẳng***

2.3. Một số vấn đề cần quan tâm

2.3.1.Khởi tạo trọng số

Việc khởi tạo trọng số có ý nghĩa rất quan trọng trong quá trình huấn luyện mạng. Khi các trọng số được gán các giá trị khởi đầu lớn, rất có thể các nơ ron của mạng sẽ đi đến trạng thái bão hoà. Nếu điều này xảy ra các gradient cục bộ trọng thuật toán back – propagation đạt các giá trị nhỏ và sẽ làm cho quá

trình học chậm đi. Tuy nhiên nếu các trọng số được gán các giá trị khởi đầu nhỏ, thuật toán có xu hướng hoạt động trên một vùng bằng phẳng xung quanh gốc của bề mặt lỗi. Bởi vậy việc sử dụng các giá trị lớn hay nhỏ cho việc khởi đầu các trọng số đều nên tránh, giá trị thích hợp ở đây là những giá trị nằm giữa hai thái cực đó

2.3.3.Tốc độ học

Một trong những yếu tố tác động đến mạng là tham số tốc độ học η, nếu tham số tốc độ học η lớn thì tốc độ hội tụ sẽ tăng tuy nhiên có thể kết quả sẽ vượt ra đích mong muốn, trong khi tham số tốc độ học nhỏ thì tốc độ hội tụ sẽ chậm. Việc lựa chọn tham số này thường được dựa theo kinh nghiệm

Một vấn đề khác đặt ra là một tham số tốc độ học là tốt khi bắt đầu huấn luyện nhưng lại là không tốt ở cuối quá trình huấn luyện, để khắc phục điều này người ta sử dụng một tham số tốc độ học thích nghi, tham số này có thể được điều chỉnh như sau:

$$\Delta\eta = \begin{cases} +a & if\ \Delta E < 0 \\ -b\eta & if\ \Delta E > 0 \\ 0 & otherwise \end{cases} \qquad (2.21)$$

trong đó a, b là các hằng số dương

2.3.3.Moment

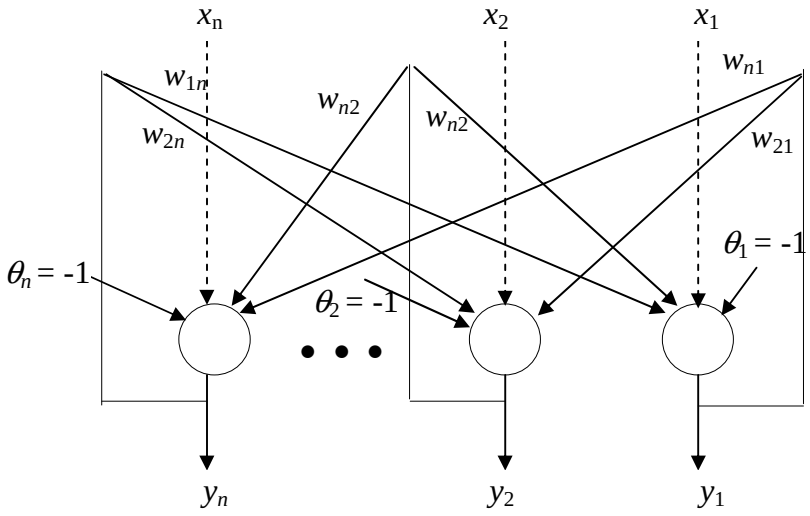
Có một điều đáng quan tâm là tham số tốc độ học càng nhỏ thì sử thay đổi của trọng số sẽ càng nhỏ và quỹ đạo trong không gian trọng số sẽ càng trơn, tuy nhiên điều này làm cho tốc độ học chậm đi. Trái lại nếu tham số tốc độ học quá lớn, sự thay đổi lớn của các trọng số có thể làm mạng trở nên không ổn định. Một phương pháp đơn giản để tăng tốc độ học mà tránh được nguy cơ không ổn định trên là thay đổi quy tắc delta trong công thức(??) bằng cách sử dụng thêm một toán hạng moment như sau

$$\Delta w(t) = -\eta\nabla E(t) + \alpha\Delta w(t-1) \qquad (2.22)$$

với α là tham số moment và giá trị thường được sử dụng là 0.9 [chin teng lin]. Công thức ?? được gọi là công thức delta tổng quát mà công thực … là một trờng hợp đặc biệt với α = 0

3. Mạng một lớp đơn hồi quy và bộ nhớ kết hợp

3.1 Mạng Hopfield rời rạc



Hình 3.1. ***Cấu trúc của mạng Hopfield***

Hình 3.1 là cấu trúc mạng Hopfield là mạng phản hồi đơn lớp hình. Với chế độ hoạt động rời rạc theo thời gian, nó đợc gọi là mạng Hopfied rời rạc. Cơ chế hoạt động của mạng như sau:

Các mẫu đầu vào đi qua mạng và sản sinh các đầu ra tương ứng, sau đó các đầu ra này lại được phản hồi trở lại mạng (xem hình vẽ). Quá trình được lặp đi lặp lại cho đến khi mạng đạt trạng thái ổn định.

Xét mạng có cấu trúc như hình 3.1 Mỗi nút mạng có một đầu vào bên ngoài x_j và một giá trị ngưỡng θ,với j = 1, 2, …, n. Các nút đầu ra không phản hồi lại chính nó. Các đầu ra có thứ tự j của mạng được nối với các đầu vào có thứ tự i ($i \neq j$) khác của mạng bằng một liên kết có trọng số w_{ij}, điều đó có nghĩa w_{ii} = 0 với i = 1, 2, …, n. Hơn nữa mạng đòi hỏi các trọng số của mạng có tính chất đối xứng, có nghĩa là $w_{ij} = w_{ji}$ với i, j = 1, 2, …, n. Luật tiến hoá (cập nhật) cho mỗi nút trong mạng Hopfield rời rạc là:

$$y_i^{(k+1)} = sign\left(\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i \right), \; i = 1, 2, \ldots, n \qquad (3.1)$$

với *sign*(.) là hàm dấu được định nghĩa trong?? và chỉ số *k* chỉ lần cập nhật thứ *k* của mạng. Một yêu cầu rất quan trọng là quá trình cập nhật của mạng được thực hiện theo chế độ không đồng bộ, điều này có nghĩa là tại lần cập nhật thứ *k* chỉ có một nút được cập nhật đầu ra của nó. Lần cập nhật tiếp theo sẽ chọn ngẫu nhiên một nút trong số các nút của mạng. Nói cách khác ở chế độ hoạt động không đồng bộ, đầu ra của mỗi nút được cập nhật riêng biệt, ví dụ sau sẽ cho thấy điều đó:

Ví dụ: Giả sử ta có mạng Hopfield rời rạc có 2 nút với $w_{12} = w_{21} = -1$, $w_{11} = w_{22} = 0$, $x_1 = x_2 = 0$ và $\theta_1 = \theta_2 = 0$. Véc tơ đầu ra được khởi tạo như sau $y^{(0)} = [-1, -1]^T$. Theo chế độ cập nhật không đồng bộ chỉ có một nút được cập nhật tại một thời điểm, do đó ta giả sử nút đầu tiên được chọn để cập nhật, kết quả là:

$$y_1^{(1)} = sign(w_{12} y_2^{(0)}) = sign[(-1)(-1)] = 1$$

Do đó, $y^{(1)} = [1, -1]^T$. Sau đó, nút thứ 2 được chọn để cập nhật, kết quả là:

$$y_2^{(2)} = sign(w_{21} y_1^{(1)}) = sign[(-1)(1)] = -1$$

Do đó, $y^{(2)} = [1, -1]^T$. Tiếp tục như vậy ta sẽ thấy trạng thái [1, -1] là trạng thái ổn định của mạng. Sử dụng các khởi tạo khác, chúng ta sẽ tìm thấy trạng thái ổn định thứ hai của mạng là [-1, 1]

Xét chế độ cập nhật đồng bộ với véc tơ đầu ra được khởi tạo là [-1, -1], sử dụng công thức () ta có:

$$y^{(1)} = \begin{pmatrix} sign(w_{12} y_2^0) \\ sign(w_{21} y_1^0) \end{pmatrix} = \begin{pmatrix} sign((-1)(-1)) \\ sign((-1)(-1)) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Lần cập nhật thứ 2 ta có

$$y^{(2)} = \begin{pmatrix} sign(w_{12} y_2^1) \\ sign(w_{21} y_1^1) \end{pmatrix} = \begin{pmatrix} sign((-1)(1)) \\ sign((-1)(1)) \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

Ta thấy kết quả quay lại giống như véc tơ $y^{(0)}$. Tóm lại, chế độ cập nhật đồng bộ sinh ra một chu trình 2 trạng thái thay vì đi đến trạng thái ổn định. Ví dụ này đã xác định rằng chế độ cập nhật đồng bộ có thể là nguyên nhân dẫn tới mạng không hội tụ tới điểm ổn định

Bây giờ chúng ta sẽ đánh giá đặc tính ổn định của mạng Hopfield rời rạc. Với mục đích này, chúng ta có thể mô tả hoạt động của mạng này bằng một hàm năng lượng như sau:

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} y_i y_j - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i \qquad (3.2)$$

Ý tưởng này thể hiện rằng nếu mạng ổn định thì hàm năng lượng ở trên luôn giảm. Giả sử nút *i* thay đổi từ trạng thái $y_i^{(k)}$ sang $y_i^{(k+1)}$. Nói cách khác đầu ra của nó thay đổi từ +1 sang –1, hoặc ngược lại. Thay đổi năng lượng ΔE là:

$$\Delta E = E(y_i^{k+1}) - E(y_i^k) = -\left(\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i\right)\left(y_i^{(k+1)} - y_i^{(k)}\right) \qquad (3.3)$$

hoặc viết vắn tắt

$$\Delta E = -(net_i)\Delta y_i \qquad (3.4)$$

với $\Delta y_i = (y_i^{(k+1)} - y_i^{(k)})$. Phương trình ??? lợi dụng yếu tố $y_i^{(k+1)} = y_i^{(k)}$ khi $j \neq i$, và $w_{ij} = w_{ji}$ và $w_{ii} = 0$

Từ công thức 3.4, nếu y_i thay đổi từ $y_i^{(k)} = -1$ sang $y_i^{(k+1)} = +1\,(\Delta y_i = 2)$ thì net_i phải dương và ΔE sẽ âm. Tương tự nếu y_i thay đổi từ $y_i^{(k)} = +1$ sang $y_i^{(k+1)} = -1\,(\Delta y_i = -2)$ thì net_i phải âm và ΔE sẽ dương. Nếu y_i không thay đổi thì $\Delta y_i = (y_i^{(k+1)} - y_i^{(k)}) = 0$, trường hợp này ΔE, do đó ta luôn có:

$$\Delta E \leq 0 \qquad (3.5)$$

Từ đó hàm năng lượng E trong 3.2 là một dạng bậc hai và bị ràng buộc trong một không gian n chiều bao gồm 2^n đỉnh của siêu lập phương *n* chiều, *E* phải có giá trị cực tiểu tuyệt đối. Thêm nữa, cực tiểu của *E* phải nằm tại góc của siêu lập phương. Tóm lại, hàm năng lượng theo luật cập nhật 3.1 phải đạt tới giá trị cực tiểu của nó. Như vậy với bất kỳ trạng thái khởi đầu nào, mạng Hopfield luôn hội tụ về trạng thái ổn định sau một số bước cập nhật nhất định.

3.2. Bộ nhớ kết hợp

Bộ nhớ kết hợp có thể lưu giữ được một tập các mẫu, khi hoạt động cùng với một *mẫu khoá*, nó sẽ sinh ra một một trong những mẫu đã được lưu giữ gần

giống hoặc có quan hệ gần với mẫu khoá nhất, đó chính là sự hồi tưởng của bộ nhớ. Mạng Hopfield cũng là một bộ nhớ kết hợp như vậy

Có hai loại bộ nhớ kết hợp. Đó là bộ nhớ kiểu *autoassociative*, và *heteroassociative*

Giả sử ta có p cặp véc tơ $\{(x^1, y^1), (x^2, y^2), \ldots,(x^p, y^p)\}$, với $x^i \in R^n$ và $y^i \in R^m$. Trong bộ nhớ kiểu *autoassociative* các véc tơ $x^i = y^i$ và mạng được cài đặt như một ánh xạ ϕ từ R^n vào R^n sao cho $\phi(x^i) = x^i$. Nếu một mẫu tuỳ ý x gần với x^i hơn x^j, $j = 1, 2, \ldots, n$, $j \neq i$, thì $\phi(x) = x^i$. Theo đó mạng sẽ sản sinh ra mẫu lưu giữ x^i khi chính x^i đóng vai trò đầu vào. Trong bộ nhớ kiểu *heteroassociative* mạng được cài đặt như một ánh xạ ϕ từ R^n vào R^m sao cho $\phi(x^i) = y^i$. Nếu một mẫu tuỳ ý x gần với x^i hơn x^j, $j = 1, 2, \ldots, n$, $j \neq i$, thì $\phi(x) = y^i$. Khái niệm gần (*closer*) được hiểu theo khía cạnh độ đo khoảng cách, ví dụ như khoảng cách Euclidean hay Hamming, cho hai véc tơ $(x_1, x_2, \ldots, x_n)$ và $(x'_1, x'_2, \ldots, x'_n)$ các khoảng cách này được định nghĩa như sau

$$ED(x, x') = \left(\sum_{i=1}^{n} |x_i - x'_i| \right)^{\frac{1}{2}} \tag{3.6}$$

$$HD(x, x') = \begin{cases} \sum_{i=1}^{n} |x_i - x'_i| & if\ x_i, x'_i \in \{0,1\} \\ \frac{1}{2} \sum_{i=1}^{n} |x_i - x'_i| & if\ x_i, x'_i \in \{-1,1\} \end{cases} \tag{3.7}$$

Trong trường hợp đặc biệt khi các x^i, $i = 1, 2, \ldots, n$ là tập các véc tơ trực giao, bộ nhớ kết hợp được xác định như sau:

$$\phi(x) = Wx \underline{\underline{\Delta}} [y^1 (x^1)^T + y^2 (x^2)^T + \ldots + y^p (x^p)^T] x \tag{3.8}$$

với W có thể được coi như một ma trận trọng số của mạng. Có thể dễ dàng nhận thấy $\phi(x^i) = Wx^i = y^i$ khi tập các véc tơ x trực giao. Mạng kết hợp cùng ma trận trọng số được định nghĩa trong công thức (3.8) được gọi là một bộ kết hợp tuyến tính.

3.2.1. Bộ nhớ Hopfield

Trước tiên ta giới thiệu Hopfield autoassociative memory, Mạng Hopfield được sử dụng như một bộ nhớ tự kết hợp. Mạng Hopfield có thể tìm lại được các

véc tơ gốc đã được lưu giữ kể cả khi nó có một số sai lệch, điều đó có nghĩa là mạng có thể phục hồi được dữ liệu khi đưa vào. Công thức (3.1) được gọi là luật phục hồi dữ liệu hoạt động trong chế độ không đồng bộ. Vấn đề còn lại là lưu giữ dữ liệu như thế nào trong bộ nhớ. Giả sử các véc tơ lưỡng cực cần được lưu giữ là x^k, $k = 1, 2, \ldots, p$. Việc lưu giữ các véc tơ trên chính là tìm ma trận trọng số W như sau:

$$W = \sum_{k=1}^{p} x^k (x^k)^T - pI, \tag{3.9}$$

hoặc

$$w_{ij} = \sum_{k=1}^{p} x_i^k x_j^k \ , \ i \neq j; \ w_{ii} = 0 \tag{3.10}$$

với $x^k = (x_1^k, x_2^k, \ldots, x_n^k)$ và I là ma trận đơn vị. Nếu x^k là véc tơ đơn cực, tức là $x_i^k \in \{0, 1\}$ thì luật lưu giữ là:

$$w_{ij} = \sum_{k=1}^{p} (2x_i^k - 1)(2x_j^k - 1) \ , \ i \neq j; \ w_{ii} = 0 \tag{3.11}$$

Luật cập xác định trọng số trên dựa vào luật học Hebb trong công thức (9.20) với các trọng số được khởi tạo bằng 0. Tóm lại luật này được gọi là luật học kiểu Hebb.

Ví dụ: Giả sử bộ nhớ Hopfield lưu hai véc tơ x^1 và x^2:

$x^1 = [1, -1, -1, 1]^T$ và $x^2 = [-1, 1, -1, 1]^T$

Từ công thức .. ta có ma trận trọng số

$$W = \begin{pmatrix} 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{pmatrix}$$

và từ công thức …, hàm năng lượng là

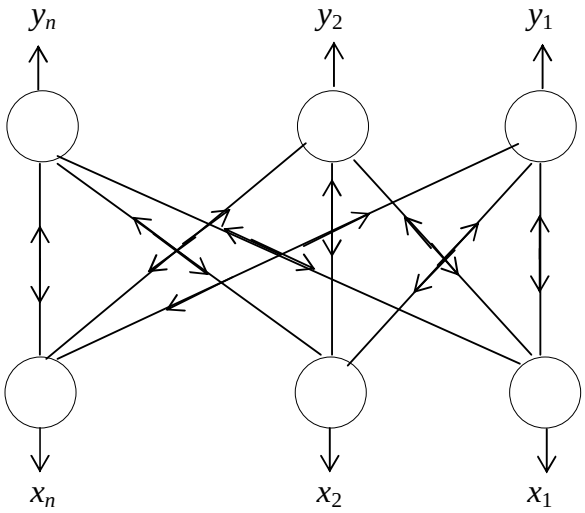
$$E(x) = -\frac{1}{2} x^T W x = 2(x_1 x_2 + x_3 x_4)$$

Bây giờ ta sẽ xem xét một số ví dụ của sự chuyển tiếp các trạng thái khi mạng hoạt động. Trạng thái được xem xét trước tiên là $[1, 1, 1, 1]^T$, các nút lần lượt được cập nhật theo thứ tự tăng, chúng ta có:

$[1, 1, 1, 1]^T \rightarrow [-1, 1, 1, 1]^T \rightarrow [-1, 1, 1, 1]^T \rightarrow [-1, 1, -1, 1]^T$… tóm lại trạng thái sẽ hội tụ đến mẫu x^2. Tuy nhiên khi ta cập nhật theo các thứ tự khác trạng thái $[1, 1, 1, 1]^T$ sẽ hội tụ đến $x^1, \bar{x}^1$ hoặc $\bar{x}^2$, điều này xảy ra là do khoảng cách Hamming giữa trạng thái ban đầu và các trạng thái khác đều bằng 2

Khả năng nhớ mẫu của Hopfield đã được phân tích khá kỹ trong [??], theo đó một mạng có n nút có thể lưu giữ được số mẫu $p \leq 0.14n$

3.2.2. Bộ nhớ liên kết hai chiều (BAM)



Hình 3.2. ***Bộ nhớ liên kết hai chiều***

Bộ nhớ liên kết hai chiều là một bộ nhớ kết hợp heteroassociative bao gồm hai lớp, có thể coi nó như là sử mở rộng của mạng Hopfield. Cũng giống như mạng Hopfield ta có bộ nhớ liên kết hai chiều rời rạc và liên tục, trong khuôn khổ chuyên đề này ta chỉ đề cập tới bộ nhớ liên kết hai chiều rời rạc, hình 3.2 là cấu trúc của BAM rời rạc, và chế độ hoạt động của bộ nhớ này được xác định như sau:

Một véc tơ x đã được khởi tạo được coi như các đầu ra của tầng X sẽ là đầu vào của các nơ ron tầng Y, và các đầu ra của tầng Y được xác định theo công thức sau đây:

$$\lambda_\iota = \alpha(\aleph x) \text{ hoặc } y'_i = a(\sum_{j=1}^{m} w_{ij} x_j), j = 1, 2, \ldots, n \qquad (3.12)$$

với a(.) là hàm ngưỡng. Sau đó véc tơ y’ xem như đầu vào của các nơ ron tầng X và các đầu ra của các nơ ron tầng X được xác định theo công thức sau:

$$x' = a(W^T y) \text{ hoặc } x'_j = a(\sum_{i=1}^{n} w_{ji} y'_i),\ i = 1, 2, \ldots, m \qquad (3.13)$$

Sau đó *x’* được xem như đầu vào của tầng Y và tầng này sinh ra *y’’* theo công thức 3.12 rồi *y’’* được xem như đầu vào của tầng X và tầng này sinh ra *x’’* theo công thức 3.13 Quá trình xử lý này tiếp tục cho đến khi việc cập nhật *x*, *y* dừng. Quá trình xử lý đệ quy trên bao gồm các bước sau:

$y^{(1)} = a(Wx^{(0)})$

$x^{(2)} = a(Wy^{(1)})$

$y^{(3)} = a(Wx^{(1)})$

……..

……..

$y^{(k-1)} = a(Wx^{(k-2)})$

$x^{(k)} = a(Wy^{(k-1)})$

Lưu ý rằng chế độ cập nhật trạng thái trong công thức 3.12, 3.13 là đồng bộ. Tuy nhiên vẫn có thể sử dụng chế độ cập nhật không đồng bộ cho các công thức bằng cách chọn ngẫu nhiên các nút xử lý *i*, *j*. Tuy nhiên hoạt động đồng bộ thường được sử dụng trong bộ nhớ liên kết hai chiều vì năng lượng của mạng sẽ thay đổi lớn dẫn đến mạng hội tụ nhanh

Giả sử có p cặp véc tơ mẫu được lưu giữ trong BAM:

$\{(x^1, y^1), (x^2, y^2), \ldots, (x^p, y^p)\}$

với $x^k = (x_1^k, x_2^k, \ldots, x_m^k)$ và $y^k = (y_1^k, y_2^k, \ldots, y_n^k)$ là các véc tơ lưỡng cực. Sử dụng luật ??, luật học của BAM là:

$$W = \begin{cases} \sum_{k=1}^{p} y^k (x^k)^T & \textit{for bipolar vectors} \\ \sum_{k=1}^{p} (2y^k - 1)(2x^k - 1) & \textit{for unipolar vectors} \end{cases}$$

hoặc

$$W = \begin{cases} \sum_{k=1}^{p} y_i^k x_j^k & \textit{for bipolar vectors} \\ \sum_{k=1}^{p} (2y_i^k - 1)(2x_j^k - 1) & \textit{for unipolar vectors} \end{cases}$$

Giả sử $x^{k'}$ là một trong những véc tơ được lưu giữ trong BAM, cho mạng hoạt động với đầu vào là véc tơ này, theo công thức ?? ta có:

$$y = a\left(\sum_{k=1}^{p} (y^k (x^k)^T x^{k'}\right) = a\left(ny^{k'} + \sum_{\substack{k=1 \\ k \neq k'}}^{p} (y^k (x^k)^T x^{k'}\right) =$$

$$= a(ny^{k'} + \eta)$$

với η là nhiễu . Do đó, nếu các véc tơ x^k là trực giao ($HD(x^k, x^{kl}) = n/2$ với $k, l = 1, 2,\ldots,p, k \neq l$) thì nhiễu $\eta = 0$ và $y = a(ny^{k'}) = y^{k'}$, a(.) là hàm ?? , điều này có nghĩa là sự ổn định xảy ra ngay lập tức khi dữ liệu đi qua tầng ra. Nếu véc tơ vào x_{k_1} không thuộc tập véc tơ được lưu giữ, sự ổn định tại y_{k_1} phụ thuộc vào các yếu tố như khoảng cách khoảng cách Hamming giữa véc tơ x_{k_1} và các véc tơ được lưu giữ x^k và tính trực giao của tập véc tơ y^k và khoảng cách Hamming giữa các véc tơ y^k

Phân tích tính ổn định của BAM được giới thiệu trong [], cụ thể như sau: Bộ nhớ liên kết hai chiều được nói là ổn định khi các trạng thái của nó hội tụ đến điểm ổn định: $y^{(k)} \to y^{(k+1)} \to y^{(k+1)}$ và $y^{(k)} = y^{(k+1)}$, Điều này tương ứng với cực tiểu của hàm năng lượng. Hàm năng lượng của BAM được định nghĩa như sau:

$$E(x, y) \equiv -\frac{1}{2} x^T W^T y - -\frac{1}{2} y^T W x = -y^T W x$$

Năng lượng thay đổi giảm theo các bít đơn trong Δy_i và Δx_i như sau:

$$\Delta E_{y_i} = \nabla_y E \Delta y_i = -Wx\Delta y_i = -\left(\sum_{j=1}^{m} w_{ij} x_j\right)\Delta y_i, \; i = 1, 2,\ldots, n \qquad ()$$

$$\Delta E_{x_j} = \nabla_x E \Delta x_j = -W^T y\Delta x_j = -\left(\sum_{i=1}^{n} w_{ji} y_i\right)\Delta x_j, \; j = 1, 2,\ldots, m \qquad ()$$

Từ (11.20),() chúng ta có

$$\Delta y_i = \begin{cases} 2 & for \quad \sum_{j=1}^{m} w_{ij} x_j > 0 \\ 0 & for \quad \sum_{j=1}^{m} w_{ij} x_j = 0 \\ -2 & for \quad \sum_{j=1}^{m} w_{ij} x_j < 0 \end{cases} \qquad \Delta x_j = \begin{cases} 2 & for \quad \sum_{i=1}^{n} w_{ji} y_i > 0 \\ 0 & for \quad \sum_{i=1}^{n} w_{ji} y_i = 0 \\ -2 & for \quad \sum_{i=1}^{n} w_{ji} y_i < 0 \end{cases}$$

Điều này chứng tỏ $\Delta E \le 0$ cho cả trường hợp cập nhật đồng bộ và không đồng bộ. Mặt khác hàm năng lượng cũng có giới hạn dưới $E(x, y) \ge -\sum_{i=1}^{n}\sum_{j=1}^{m} | w_{ij} |$. Bộ nhớ kết hợp BAM hội tụ về trạng thái ổn định.

Ví dụ: Cho hai cặp mẫu được lưu giữ trong bộ nhơ liên kết hai chiều

$x^1 = [1, -1, -1, 1, 1, -1, 1, -1, -1, -1]^T$, $y^1 = [-1, -1, 1, 1, 1, 1]^T$

$x^2 = [1, 1, 1, -1, 1, 1, 1, -1, -1, -1]^T$, $y^2 = [-1, -1, 1, 1, -1, -1]^T$

Từ phương trình ?? ta có:

$$W = \begin{bmatrix} -2 & 0 & 0 & 0 & -2 & 0 & -2 & 2 & 0 & 0 \\ -2 & 0 & 0 & 0 & -2 & 0 & -2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & 0 & 2 & -2 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & 0 & 2 & -2 & 0 & 0 \\ 0 & -2 & -2 & 2 & 0 & -2 & 0 & 0 & -2 & 2 \\ 0 & -2 & -2 & 2 & 0 & -2 & 0 & 0 & -2 & 2 \end{bmatrix}$$

Ta chọn véc tơ x sao cho $HD(x, x^1) = 2$:

$x^{(0)} = [-1, 1, -1, 1, 1, -1, 1, -1, -1, 1]^T$

Từ công thức … chúng ta có:

$y^{(1)} = a([-4, -4, 4, 4, 8, 8]^T) = [-1, -1, 1, 1, 1, 1]^T$

$x^{(2)} = a([8, -4, -4, 4, 8, -4, 8, -8, -4, 4]^T) =$

$= [1, -1, -1, 1, 1, -1, 1, -1, -1, 1]^T$

$y^{(3)} = [-1, -1, 1, 1, 1, 1]^T = y^{(1)}$

Đây là trạng thái ổn định của bộ nhớ và là sự nhớ lại của BAM về cặp mẫu đầu tiên đã lưu giữ.

Chọn $x^{(0)} = [-1, 1, 1, -1, 1, -1, -1, 1, 1, 1]^T$, ta thấy $HD(x^{(0)}, x^{(1)}) = 7$ và $HD(x^{(0)}, x^{(2)}) = 5$, chúng ta hy vọng cặp mẫu thứ hai sẽ được BAM nhớ lại. Sử dụng công thức … ta có

$y^{(1)} = a([4, 4, -4, -4, -4, -4]^T) = [1, 1, -1, -1, -1, -1]^T$

$x^{(2)} = a([8, 4, 4, -4, -8, 4, -8, 8, 4, -4]^T) =$

$= [1, 1, 1, -1, -1, 1, -1, 1, 1, -1]^T$

$y^{(3)} = [1, 1, -1, -1, -1, -1]^T = y^{(1)}$

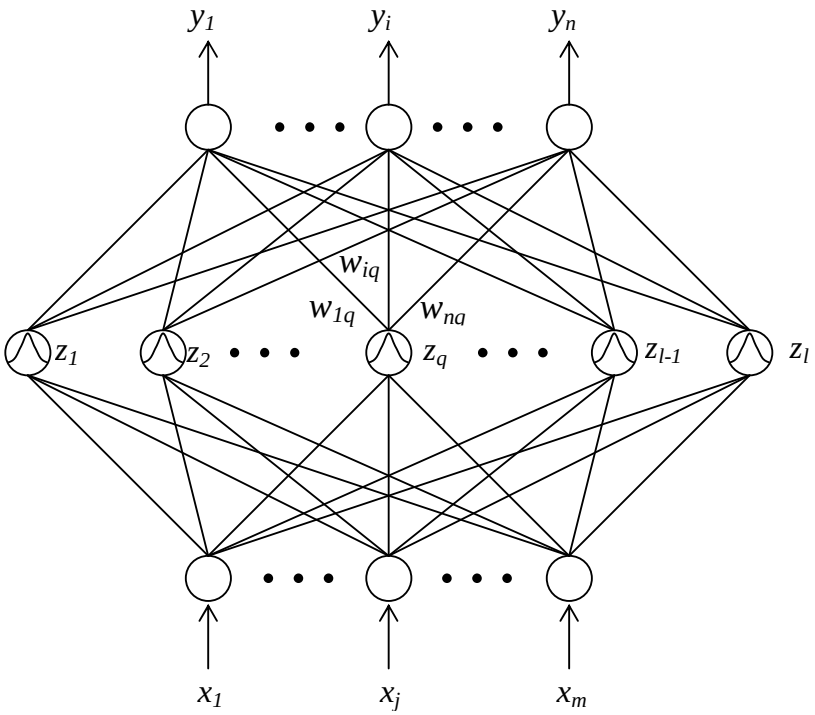
Đây là trạng thái ổn định của bộ nhớ, tuy nhiên bộ nhớ đã không nhớ lại bất kỳ cặp mẫu nào đã được lưu giữ, trên thực tế đây $(x^{(2)}, y^{(3)})$ là phần bù của cặp (x^1, y^1). Như vậy nếu như bộ nhớ đã lưu giữ cặp (x, y) thì cũng lưu giữ luôn $(\overline{x}, \overline{y})$

Khả năng nhớ mẫu của BAM đã được phân tích khá kỹ ở trong [], kết quả cho thấy với một bộ nhớ có m đầu vào và n đầu ra, có thể lưu giữ được $p = \sqrt{\min(m,n)}$ cặp mẫu

4. Mạng nơ ron RBF

4.1.Cấu trúc mạng nơ ron (RBFN- Radial Basic Function Network)

Sơ đồ cấu trúc của RBF như hình 4.1. Trong đó x_j là tín hiệu vào của mạng RBF với $i = 1, 2,…, m$ (còn gọi x là véc tơ đầu vào của mạng); y_i là tín hiệu ra của RBF với $j = 1,2,…, n$; z_q là số phần tử nơ ron lớp ẩn của mạng RBF với $q = 1, 2, …, l$; w_q là các trọng số kết nối giữa lớp ẩn và lớp đầu ra của mạng RBF. Đặc điểm của cấu trúc mạng nơ ron RBF có cấu trúc khác với cấu trúc của mạng nơ ron lan truyền ngược



Hình 4.1. ***Cấu trúc RBF*****.**

Giá trị đầu ra tại mỗi nút của lớp ẩn của mạng nơ ron RBF thông thường là ở dạng hàm Gaussian và có dạng như sau:

$$z_q = g_q(x) \triangleq \frac{R_q(x)}{\sum_k R_k(x)} = \frac{\exp\left[-\left|x-m_q\right|^2 \Big/ 2\sigma_q^2\right]}{\sum_k \exp\left[-\left|x-m_k\right|^2 \Big/ 2\sigma_k^2\right]}, \qquad (4.1)$$

trong đó m_q là véc tơ tâm của hàm cơ sở thứ q, σ_q là bán kính (độ rộng) của hàm cơ sở thứ q.

Giá trị đầu ra thứ i của mạng nơ ron RBF là y_i:

$$y_i = a_i\left(\sum_{q=1}^{l} w_{iq} z_q + \theta_i\right), \qquad (4.2)$$

trong đó $a_i(.)$ là hàm kích hoạt đầu ra của phần tử nơ ron thứ i, θ_i là giá trị ngưỡng (threshold) của phần tử nơ ron thứ i và thông thường lấy $\theta_i = 0$. Như vậy dạng hàm kích hoạt đầu ra của phần tử nơ ron là dạng hàm tuyến tính.

Như vậy RBF chỉ có một lớp ẩn q được kích hoạt và tương ứng với véc tơ trọng số $w_q = (w_{1q}, w_{2q}, \ldots, w_{nq})^T$. Giá trị đầu ra tuyến tính thứ i của RBF được tính theo tổng của tích véc tơ trọng số w_q với véc tơ giá trị đầu ra của lớp ẩn z_q. Kể từ đây thì RBF mới giống như mạng nơ ron lan truyền thẳng.

2.2.2. Huấn luyện RBF.

Mẫu vào ra để huấn luyện RBF là (x^k, d^k), $k = 1, 2, \ldots, p$. RBF được huấn luyện luật học lai: học không giám sát trong lớp đầu vào và lớp đầu ra. Các trọng số trong lớp đầu ra có thể được cập nhật một cách đơn giản bằng cách sử dụng luật học delta như sau:

$$\Delta w_{iq} = \eta(d_i - y_i)z_q \qquad (4.3)$$

Tính tổng trung bình bình phương sai số tính cho p cặp mẫu vào ra của mạng, và huấn luyện mạng sao cho tổng trung bình bình phương sai số là nhỏ nhất. Tổng trung bình bình phương sai số được tính như sau:

$$E(w_{iq}) = \frac{1}{2}\sum_k\sum_i\left[d_i^k - y_i^k\right]^2 = \frac{1}{2}\sum_k\sum_i\left[d_i^k - \sum_{q=1}^{l} w_{iq} z_q^k\right]^2$$

$$= \frac{1}{2}\sum_k\sum_i\left[d_i^k - \sum_{q=1}^{l} w_{iq} g_q(x^k)\right]^2 \qquad (4.4)$$

Tiếp theo cần phải xác định phạm vi của các tâm hàm cơ sở m_q và các độ rộng của hàm cơ sở σ_q. Các tâm m_q có thể tìm được bằng các luật học không giám sát giống như phương pháp lượng tử hoá véc tơ, các luật học cạnh tranh, hoặc luật học Kohonen đơn giản; đó là:

$$\Delta m_{closest} = \eta(x - m_{closest}), \qquad (4.5)$$

ở đây $m_{closest}$ là tâm gần với véc tơ đầu vào x nhất và các tâm khác được giữ không đổi. Chúng ta cũng có thể sử dụng luật cạnh tranh Kohonen phụ thuộc vào tất cả các tâm với giá trị cập nhật được điều chỉnh theo mối quan hệ *"khoảng cách"* giữa một tâm m_q và một đầu vào véc tơ x. Theo đó, mỗi lần thu được phạm vi các tâm m_q, các độ rộng (phương sai) có thể xác định được bằng cách giảm thiểu sai số theo σ_q sau:

$$E = \frac{1}{2}\sum_{q=1}^{l}\left[\sum_{r=1}^{l} R_q(m_r)\left|\frac{m_q - m_r}{2\sigma_q}\right|^2 - \gamma\right]^2 \quad (4.6)$$

trong đó γ là tham số thể hiện phần chồng chéo lên nhau của các hàm cơ sở. Ảnh hưởng của việc giảm thiểu này là hàm tại các nút phải đảm bảo chắc chắn là dạng hàm trơn, phép nội suy dựa trên các miền của không gian đầu vào. Trên thực tế, các độ rộng σ_q thường được xác định bằng phương pháp giống như ý nghĩa khoảng cách, có liên hệ gần gũi nhất với tâm *m*. Trong trường hợp đơn giản nhất là theo [39] như sau:

$$\sigma_q = \frac{\left|m_q - m_{closest}\right|}{\gamma} \quad (4.7)$$

ở đây $m_{closest}$ là véc tơ gần m_q nhất.

RBF có thể chọn là mạng nơ ron 2 lớp trong nhiều ứng dụng xử lý tín hiệu, nhận dạng mẫu, và xấp xỉ hàm.

RBF cũng có thể được huấn luyện bằng luật học lan truyền ngược sai số và trở thành mạng học giám sát hoàn hảo. Khi đó cần tối thiểu hoá hàm tổng trung bình bình phương sai số (4.4). Giá trị đầu ra của mạng trong suốt quá trình huấn luyện vẫn được tính theo biểu thức (4.3). Sai số đầu ra chính là giá trị độ lệch giữa đầu ra thực tế của mạng và đầu ra mong muốn, sau đó lan truyền ngược tới lớp vào để cập nhật các tâm và các độ rộng của các hàm cơ sở của chúng. Theo đó để thay đổi luật, luật học giám sát cho RBF như sau:

$$\Delta w_{iq} = \eta_w (d_i - y_i) z_q, \Delta m_q = \eta_m \sum_i (d_i - y_i)\frac{\partial y_i}{\partial m_q}, \Delta\sigma_q = \eta_\sigma \sum_i (d_i - y_i)\frac{\partial y_i}{\partial \sigma_q} \quad (4.8)$$

các đạo hàm $\partial y_i / \partial m_q$ và $\partial y_i / \partial \sigma_q$ có thể tính dựa theo biểu thức (4.1) và (4.2). Với cách này thì cả tâm và độ rộng của các hàm cơ sở đều thay đổi động. Với RBF khi huấn luyện theo thuật toán lan truyền ngược không thấy nhanh hơn rõ ràng so với mạng nơ ron lan truyền ngược. Ngoài ra, sẽ là ngẫu nhiên các chiều rộng của các hàm Gaussian học là lớn và mất vị trí mong muốn trong RBF, có nghĩa là khi đó mạng bị mất thông tin.

Kết luận: Chuyên đề đã nghiên cứu các khái niệm cơ bản về các mô hình mạng nơ ron nhân tạo, các phương pháp huấn luyện, đặt biệt đã đi sâu vào nghiên cứu

mạng nơ ron nhiều lớp truyền thẳng với giải thuật huấn luyện *"lan truyền ngược sai số"* với mục đích sử dụng lớp mạng này để xấp xỉ các hàm phi tuyến bất kỳ. Việc nghiên cứu cũng nhằm mục đích nắm vững các kiến thức cơ bản để xây dựng các mạng nơ ron và áp dụng vào phát triển phương pháp lập luận mờ sử dụng đại số gia tử, nội dung chính của luận án

Tài liệu tham khảo

1. Neural Fuzzy Systems, Chin –Teng Lin, George Lee, Prentice – Hall international, Inc, 1996
2. Mạng nơ ron - Kỹ thuật lan truyền ngược, Nguyễn Đình Thúc, Nhà xuất bản khoa học kỹ thuật, 2000