Lý Thuyết Mạng Nơron

Tổng quan về mạng nơron

1. Mạng nơron nhân tạo

Mạng noron nhân tạo (Artifical Neural Networks) mô phỏng lại mạng noron sinh học là một cấu trúc khối gồm các đơn vị tính toán đơn giản được liên kết chặt chẽ với nhautrong đó các liên kết giữa các noron quyết định chức năng của mạng.

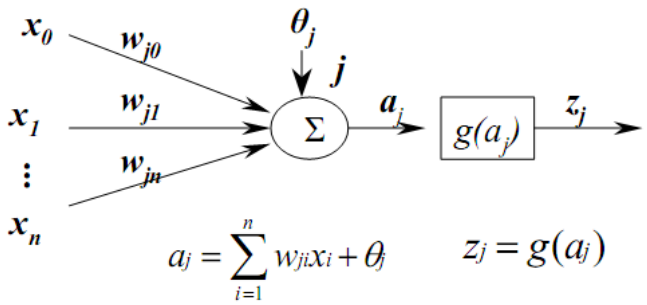
2. Các đặc trưng cơ bản của mạng nơron

- Gồm một tập các đơn vị xử lý (các noron nhân tạo)
- Trạng thái kích hoạt hay đầu ra của đơn vị xử lý
- Liên kết giữa các đơn vị. Xét tổng quát, mỗi liên kết được định nghĩa bởi một trọng số W_{jk} cho ta biết hiệu ứng mà tín hiệu của đơn vị j có trên đơn vị k
- Một luật lan truyền quyết định cách tính tín hiệu ra của từng đơn vị từ đầu vào của nó
- Một hàm kích hoạt, hay hàm chuyển (activation function, transfer function), xác định mức độ kích hoạt khác dựa trên mức độ kích hoạt hiện tại
- Một đơn vị điều chỉnh (độ lệch) (bias, offset) của mỗi đơn vị
- Phương pháp thu thập thông tin (luật học - learning rule)
- Môi trường hệ thống có thể hoạt động.

3. Các thành phần cơ bản của mạng nơron nhân tạo

3.1 Đơn vị xử lý

Còn được gọi là một nơron hay một nút (node), thực hiện một công việc rất đơn giản: nó nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu ra sẽ được lan truyền sang các đơn vị khác.



Hình 2.6 – Đơn vị xử lý (Processing Unit)

Trong đó:

xi : các đầu vào

wji : các trọng số tương ứng với các đầu vào

θj : độ lệch (bias)

aj : đầu vào mạng (net-input)

zj : đầu ra của nơron

g(x): hàm chuyển (hàm kích hoạt).

Trong một mạng nơron có ba kiểu đơn vị:

- Các đơn vị đầu vào (Input units), nhận tín hiệu từ bên ngoài.
- Các đơn vị đầu ra (Output units), gửi dữ liệu ra bên ngoài.
- Các đơn vị ẩn (Hidden units), tín hiệu vào (input) và ra (output) của nó nằm trong mạng.

Mỗi đơn vị j có thể có một hoặc nhiều đầu vào: x0, x1, x2, … xn, nhưng chỉ có một đầu ra zj. Một đầu vào tới một đơn vị có thể là dữ liệu từ bên ngoài mạng, hoặc đầu ra của một đơn vị khác, hoặc là đầu ra của chính nó.

3.2 Hàm kết hợp

Mỗi một đơn vị trong một mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với các đơn vị khác, sinh ra một giá trị gọi là net input. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi một luật lan truyền cụ thể. Trong phần lớn các mạng nơron, chúng ta giả sử rằng mỗi một đơn vị cung cấp một bộ cộng như là đầu vào cho đơn vị mà nó có liên kết. Tổng đầu vào đơn vị j đơn giản chỉ

là tổng trọng số của các đầu ra riêng lẻ từ các đơn vị kết nối cộng thêm ngưỡng hay độ lệch (bias) θj :

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

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

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

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

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

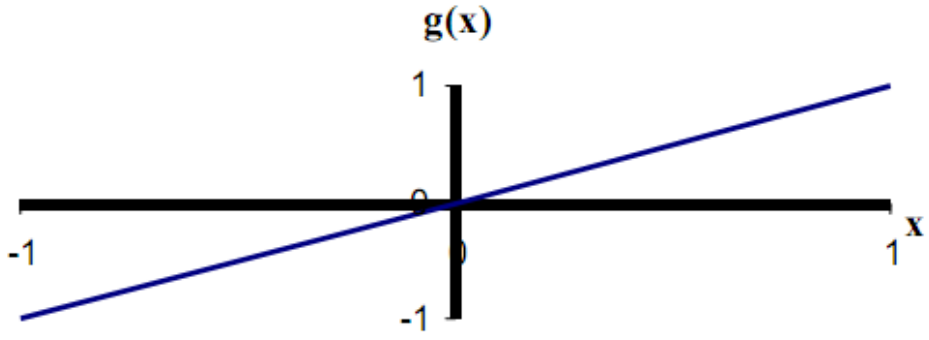
3.3 Hàm kích hoạt

Phần lớn các đơn vị trong mạng nơron chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing). Các hàm kích hoạt hay được sử dụng là:

- **Hàm đồng nhất (Linear function, Identity function)**

$$g(x) = x$$

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



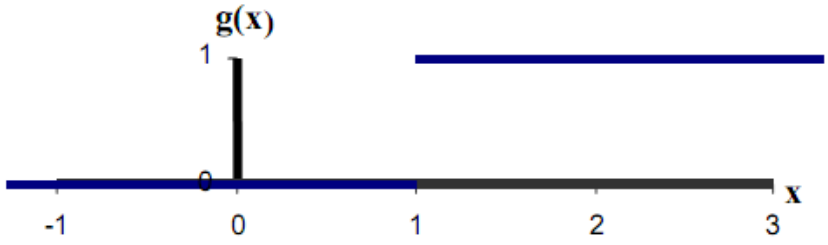
Hình 2.7 – Hàm đồng nhất (Identity function)

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

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

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

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

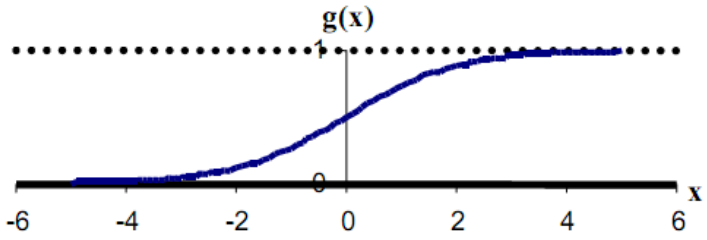


Hình 2.8 – Hàm bước nhị phân (Binary step function)

- **Hàm sigmoid (Sigmoid function (logsig))**

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

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

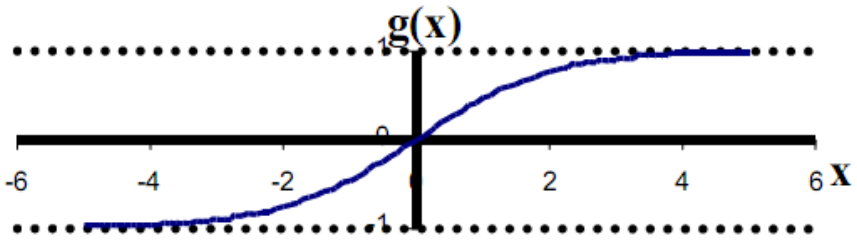


Hình 2.9 – Hàm Sigmoid

- **Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig))**

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

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



Hình 2.10 – Hàm sigmoid lưỡng cực

Các hàm chuyển của các đơn vị ẩn (hidden units) là cần thiết để biểu diễn sự phi tuyến vào trong mạng. Lý do là hợp thành của các hàm đồng nhất là một hàm đồng nhất. Mặc dù vậy nhưng nó mang tính chất phi tuyến (nghĩa là, khả năng biểu diễn các hàm phi tuyến) làm cho các mạng nhiều tầng có khả năng rất tốt trong biểu diễn các ánh xạ phi tuyến. Tuy nhiên, đối với luật học lan truyền ngược, hàm phải khả vi (differentiable) và sẽ có ích nếu như hàm được gắn trong một khoảng nào đó. Do vậy, hàm sigmoid là lựa chọn thông dụng nhất.

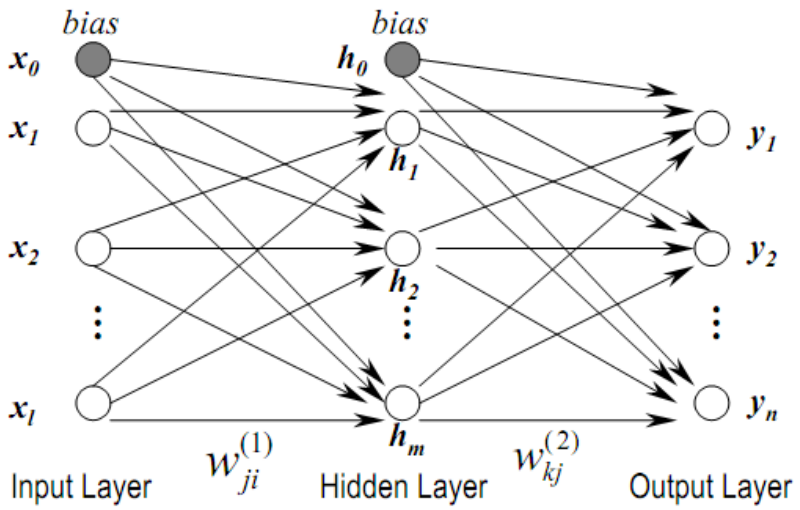
Đối với các đơn vị đầu ra (output units), các hàm chuyển cần được chọn sao cho phù hợp với sự phân phối của các giá trị đích mong muốn. Chúng ta đã thấy rằng đối với các giá trị ra trong khoảng [0,1], hàm sigmoid là có ích; đối với các giá trị đích mong muốn là liên tục trong khoảng đó thì hàm này cũng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

4. Các hình trạng của mạng

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

4.1 Mạng truyền thẳng (Feed-forward neural network):

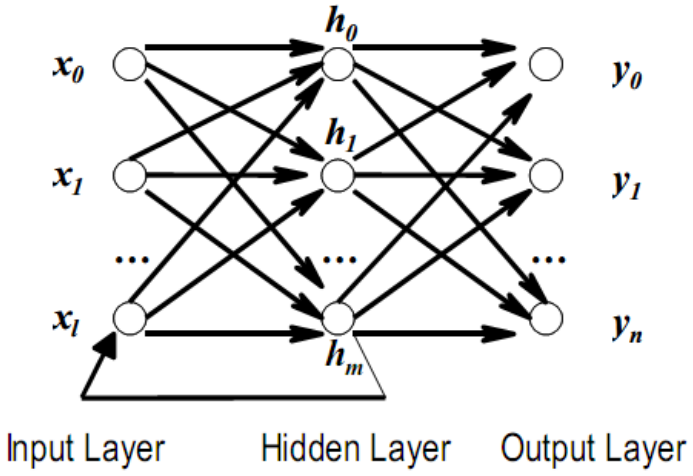
Dòng dữ liệu từ đơn vị đầu vào đến đơn vị đầu ra chỉ được truyền thẳng. Việc xử lý dữ liệu có thể mở rộng ra nhiều lớp, nhưng không có các liên kết phản hồi. Nghĩa là, các liên kết mở rộng từ các đơn vị đầu ra tới các đơn vị đầu vào trong cùng một lớp hay các lớp trước đó là không cho phép.



Hình 2.11 – Mạng nơron truyền thẳng nhiều lớp (Feed-forward neural network)

4.2 Mạng hồi quy (Recurrent neural network):

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



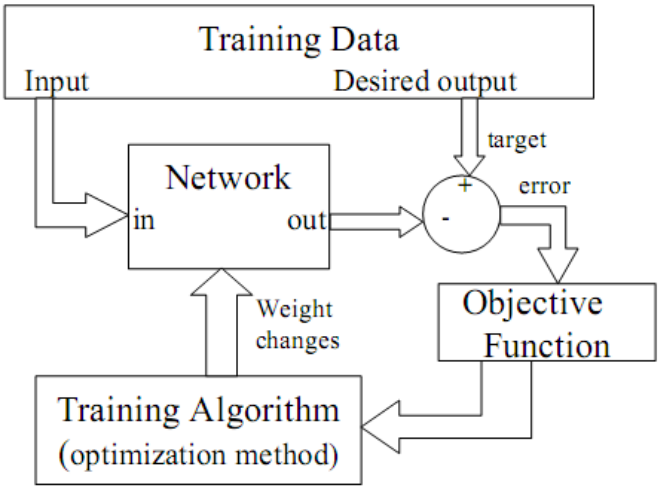
Hình 2.12 – Mạng nơron hồi quy (Recurrent neural network)

5. Huấn luyện mạng

Chức năng của một mạng nơron được quyết định bởi các nhân tố như: hình trạng mạng (số lớp, số đơn vị trên mỗi tầng, và cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Hình trạng của mạng thường là cố định, và các trọng số được quyết định bởi một thuật toán huấn luyện (training algorithm). Tiến trình điều chỉnh các trọng số để mạng "nhận biết" được quan hệ giữa đầu vào và đích mong muốn được gọi là học (learning) hay huấn luyện (training). Rất nhiều thuật toán học đã được phát minh để tìm ra tập trọng số tối ưu làm giải pháp cho các bài toán. Các thuật toán đó có thể chia làm hai nhóm chính: Học có thầy (Supervised learning) và Học không có thầy (Unsupervised Learning).

5.1 Học có thầy (Supervised learning):

Mạng được huấn luyện bằng cách cung cấp cho nó các cặp mẫu đầu vào và các đầu ra mong muốn (target values). Các cặp được cung cấp bởi "thầy giáo", hay bởi hệ thống trên đó mạng hoạt động. Sự khác biệt giữa các đầu ra thực tế so với các đầu ra mong muốn được thuật toán sử dụng để thích ứng các trọng số trong mạng. Điều này thường được đưa ra như một bài toán xấp xỉ hàm số - cho dữ liệu huấn luyện bao gồm các cặp mẫu đầu vào x, và một đích tương ứng t, mục đích là tìm ra hàm f(x) thoả mãn tất cả các mẫu học đầu vào.



Hình 2.13 – Mô hình Học có thầy (Supervised learning model)

5.2 Học không có thầy (Unsupervised Learning):

Với cách học không có thầy, không có phản hồi từ môi trường để chỉ ra rằng đầu ra của mạng là đúng. Mạng sẽ phải khám phá các đặc trưng, các điều chỉnh, các mối tương quan, hay các lớp trong dữ liệu vào một cách tự động. Trong thực tế, đối với phần lớn các biến thể của học không có thầy, các đích trùng với đầu vào. Nói một cách khác, học không có thầy luôn thực hiện một công việc tương tự như một mạng tự liên hợp, cô đọng thông tin từ dữ liệu vào.

6. Hàm mục tiêu

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

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

Trong đó

p: số thứ tự mẫu trong tập huấn luyện

i : số thứ tự của đơn vị đầu ra

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

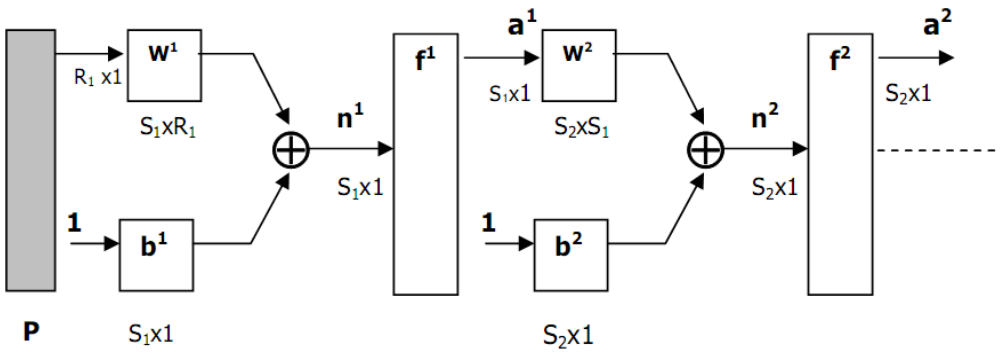
Trong các ứng dụng thực tế, nếu cần thiết có thể làm phức tạp hàm số với một vài yếu tố khác để có thể kiểm soát được sự phức tạp của mô hình.

7. Mạng truyền thẳng và thuật toán lan truyền ngược

Để đơn giản và tránh hiểu nhầm, mạng truyền thẳng trình bày trong phần này là mạng truyền thẳng có nhiều lớp (MLP - MultiLayer Perceptron). Đây là một trong những mạng truyền thẳng điển hình, thường được sử dụng trong các hệ thống nhận dạng.

Mạng truyền thẳng MLP

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



Hình 2.14 – Mạng nơron truyền thẳng nhiều lớp

Trong đó

P: Vector đầu vào (vector cột)

$\mathbf{W^i}$: Ma trận trọng số của các nơron lớp thứ i.

($\mathbf{S^i x R^i}$: S hàng (nơron) - R cột (số đầu vào))

$\mathbf{b^i}$: Vector độ lệch (bias) của lớp thứ i (S^ix1: cho S nơron)

$\mathbf{n^i}$: net input (S^ix1)

$\mathbf{f^i}$: Hàm chuyển (hàm kích hoạt)

$\mathbf{a^i}$: net output (S^ix1)

$\oplus$: Hàm tổng thông thường.

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

Xét trường hợp mạng có hai lớp như hình 2.14, công thức tính toán cho đầu ra như sau:

$$a^2 = f^2(W^2(f^1(W^1P + b^1)) + b^2)$$

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

Thiết kế cấu trúc mạng

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

Số lớp ẩn:

Vì các mạng có hai lớp ẩn có thể thể hiện các hàm với dáng điệu bất kỳ, nên, về lý thuyết, không có lý do nào sử dụng các mạng có nhiều hơn hai lớp ẩn. Người ta đã xác định rằng đối với phần lớn các bài toán cụ thể, chỉ cần sử dụng một lớp ẩn cho mạng là đủ. Các bài toán sử dụng hai lớp ẩn hiếm khi xảy ra trong thực tế. Thậm chí đối với các bài toán cần sử dụng nhiều hơn một lớp ẩn thì trong phần lớn các trường hợp trong thực tế, sử dụng chỉ một lớp ẩn cho ta hiệu năng tốt hơn là sử dụng nhiều hơn một lớp. Việc huấn luyện mạng thường rất chậm khi mà số lớp ẩn sử dụng càng nhiều.

Số đơn vị trong lớp ẩn:

Một vấn đề quan trọng trong việc thiết kế một mạng là cần có bao nhiêu đơn vị trong mỗi lớp. Sử dụng quá ít đơn vị có thể dẫn đến việc không thể nhận dạng được các tín hiệu đầy đủ trong một tập dữ liệu phức tạp, hay thiếu ăn khớp (underfitting). Sử dụng quá nhiều đơn vị sẽ tăng thời gian luyện mạng, có lẽ là quá

nhiều để luyện khi mà không thể luyện mạng trong một khoảng thời gian hợp lý. Số lượng lớn các đơn vị có thể dẫn đến tình trạng thừa ăn khớp (overfitting), trong trường hợp này mạng có quá nhiều thông tin, hoặc lượng thông tin trong tập dữ liệu mẫu (training set) không đủ các dữ liệu đặc trưng để huấn luyện mạng.

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

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

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

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

Thuật toán lan truyền ngược (Back-Propagation)

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

Về cơ bản có hai dạng thuật toán để luyện mạng: học có thầy và học không có thầy. Các mạng nơron truyền thẳng nhiều lớp được luyện bằng phương pháp học có

thầy. Phương pháp này căn bản dựa trên việc yêu cầu mạng thực hiện chức năng của nó và sau đó trả lại kết quả, kết hợp kết quả này với các đầu ra mong muốn để điều chỉnh các tham số của mạng, nghĩa là mạng sẽ học thông qua những sai sót của nó.

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

Bỏ qua sự phức tạp về mặt toán học, thuận toán có thể phát biểu đơn giản như sau:

- **Bước 1**: ***Lan truyền xuôi các tính toán trong mạng truyền thẳng***
  - Khi đó, đầu ra của một lớp trở thành đầu vào của lớp kế tiếp. Phương trình thể hiện hoạt động này như sau (trong đó M là số lớp trong mạng) :

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

  - Các nơron trong lớp thứ nhất nhận các tín hiệu từ bên ngoài (với p chính là điểm bắt đầu của phương trình hình 2.14)

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

  - Đầu ra của lớp cuối cùng được xem là đầu ra của mạng:

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

- **Bước 2**: ***Lan truyền lỗi (hay độ nhạy cảm) ngược lại qua mạng***
  - Thuật toán lan truyền ngược sử dụng chỉ số hiệu năng là trung bình bình phương lỗi của đầu ra so với giá trị đích. Đầu vào của thuật toán chính là tập các cặp mô tả hoạt động đúng của mạng:

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

Trong đó $\mathbf{p_i}$ là một đầu vào và $\mathbf{t_i}$ là đầu ra mong muốn tương ứng, với i=1..Q.

- Mỗi đầu vào đưa vào mạng, đầu ra của mạng đối với nó được đem so sánh với đầu ra mong muốn.Thuật toán sẽ điều chỉnh các tham số của mạng để tối thiểu hóa trung bình bình phương lỗi.

➢ **Bước 3**: ***Cập nhật lại các trọng số và độ lệch tương ứng***