Học Máy
(IT 4862)

Nguyễn Nhật Quang

quangnn-fit@mail.hut.edu.vn

Trường Đại học Bách Khoa Hà Nội

Viện Công nghệ thông tin và truyền thông

Năm học 2011-2012

Nội dung môn học:

- Giới thiệu chung
- Đánh giá hiệu năng hệ thống học máy
- Các phương pháp học dựa trên xác suất
- **Các phương pháp học có giám sát**
  - **Học mạng nơron nhân tạo (Artificial neural network)**
- Các phương pháp học không giám sát
- Lọc cộng tác
- Học tăng cường

Mạng nơ-ron nhân tạo – Giới thiệu (1)

- Mạng nơ-ron nhân tạo (Artificial neural network – ANN)
  - Mô phỏng các hệ thống nơ-ron sinh học (các bộ não con người)
  - ANN là một cấu trúc (structure/network) được tạo nên bởi một số lượng các nơ-ron (artificial neurons) liên kết với nhau
- Mỗi nơ-ron
  - Có một đặc tính vào/ra
  - Thực hiện một tính toán cục bộ (một hàm cục bộ)
- Giá trị đầu ra của một nơ-ron được xác định bởi
  - Đặc tính vào/ra của nó
  - Các liên kết của nó với các nơ-ron khác
  - (Có thể) các đầu vào bổ sung

Mạng nơ-ron nhân tạo – Giới thiệu (2)

- ANN có thể được xem như một cấu trúc xử lý thông tin một cách phân tán và song song ở mức cao
- ANN có khả năng học (learn), nhớ lại (recall), và khái quát hóa (generalize) từ các dữ liệu học –bằng cách gán và điều chỉnh (thích nghi) các giá trị trọng số (mức độ quan trọng) của các liên kết giữa các nơ-ron
- Chức năng (hàm mục tiêu) của một ANN được xác định bởi
  - Kiến trúc (topology) của mạng nơ-ron
  - Đặc tính vào/ra của mỗi nơ-ron
  - Chiến lược học (huấn luyện)
  - Dữ liệu học

ANN – Các ứng dụng điển hình (1)

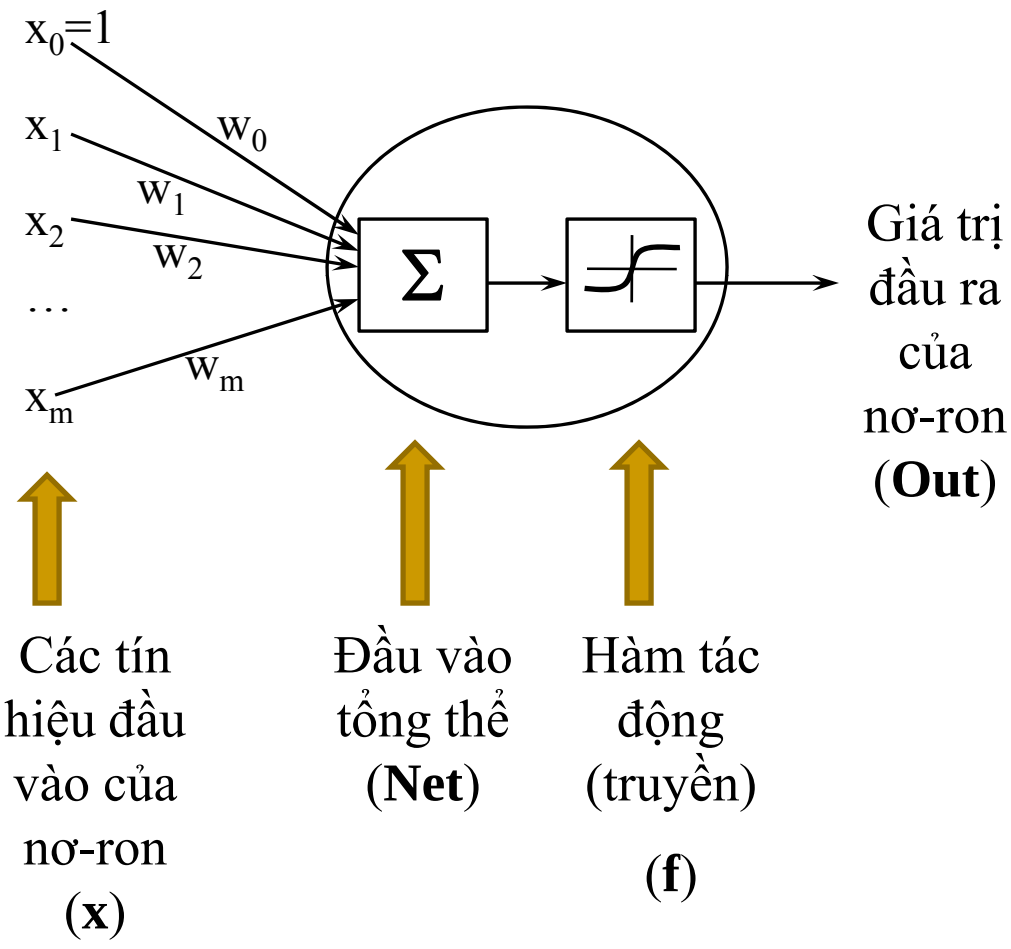
- Xử lý ảnh và Computer vision
  - Ví dụ: So khớp, tiền xử lý, phân đoạn và phân tích ảnh, computer vision, nén ảnh, xử lý và hiểu các ảnh thay đổi theo thời gian
- Xử lý tín hiệu
  - Ví dụ: Phân tích tín hiệu và hình thái địa chấn, động đất
- Nhận dạng mẫu
  - Ví dụ: Trích chọn thuộc tính, phân loại và phân tích tín hiệu ra-đa, nhận dạng và hiểu giọng nói, nhận dạng dấu vân tay, nhận dạng ký tự (chữ hoặc số), nhận dạng mặt người, và phân tích chữ viết tay
- Y tế
  - Ví dụ: Phân tích và hiểu tín hiệu điện tim, chẩn đoán các loại bệnh, và xử lý các ảnh trong lĩnh vực y tế

ANN – Các ứng dụng điển hình (2)

- Các hệ thống quân sự
  - Ví dụ: Phát hiện thủy lôi, phân loại nhiễu ra-đa
- Các hệ thống tài chính
  - Ví dụ: Phân tích thị trường chứng khoán, đánh giá giá trị bất động sản, kiểm tra truy cập thẻ tín dụng, kinh doanh cổ phiếu
- Lập kế hoạch, điều khiển, và tìm kiếm
  - Ví dụ: Cài đặt song song các bài toán thỏa mãn ràng buộc, tìm lời giải cho bài toán người đưa hàng, điều khiển và khoa học nghiên cứu về người máy (robotics)
- Các hệ thống năng lượng
  - Ví dụ: Đánh giá trạng thái hệ thống, phát hiện và khắc phục sự cố, dự đoán tải (khối lượng) công việc, và đánh giá mức độ an toàn
- ...(và nhiều lĩnh vực bài toán khác!)

Cấu trúc và hoạt động của một nơ-ron

- **Các tín hiệu đầu vào (input signals)** của nơ-ron (x_i, i=1..m)
  - Mỗi tín hiệu đầu vào x_i gắn với một trọng số w_i
- Trọng số **điều chỉnh (bias)** w_0 (với x_0=1)
- **Đầu vào tổng thể (Net input)** là một hàm tích hợp của các tín hiệu đầu vào – `Net(w,x)`
- **Hàm tác động/truyền (Activation/transfer function)** tính giá trị đầu ra của nơ-ron – `f(Net(w,x))`
- **Giá trị đầu ra (Output)** của nơ-ron: `Out=f(Net(w,x))`

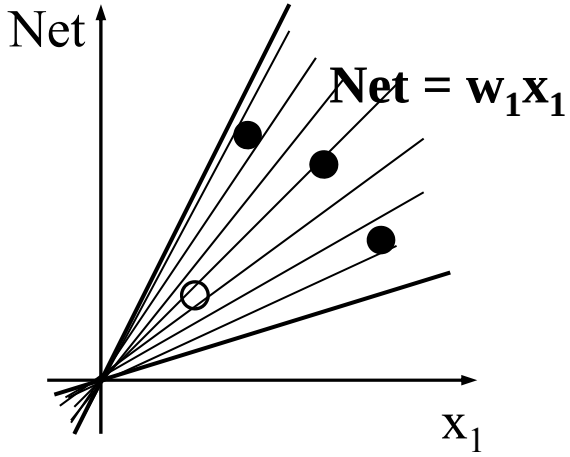


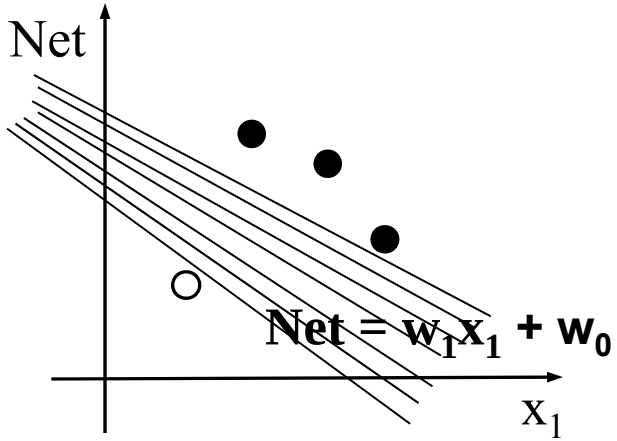
Đầu vào tổng thể và dịch chuyển

- Đầu vào tổng thể (net input) thường được tính toán bởi một hàm tuyến tính

$$Net = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = w_0 .1 + \sum_{i=1}^{m} w_i x_i = \sum_{i=0}^{m} w_i x_i$$

- Ý nghĩa của tín hiệu dịch chuyển (bias) w_0
  - → Họ các hàm phân tách $\texttt{Net}=w_1 x_1$ *không thể phân tách được* các ví dụ thành 2 lớp (two classes)
  - → Nhưng: họ các hàm $\texttt{Net}=w_1 x_1 + w_0$ có thể!



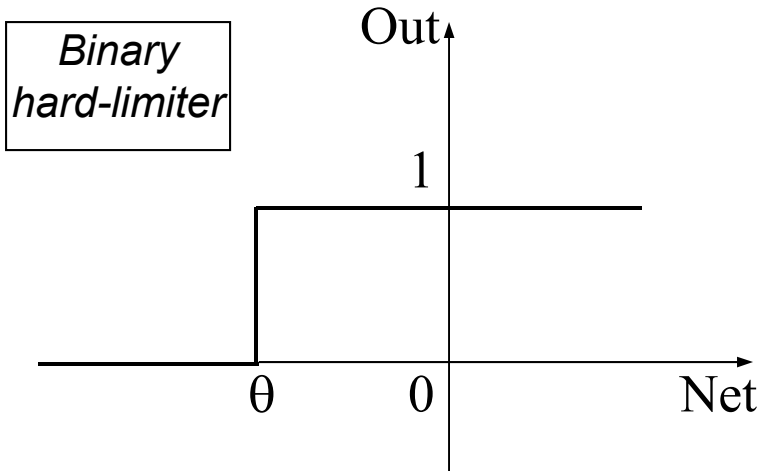


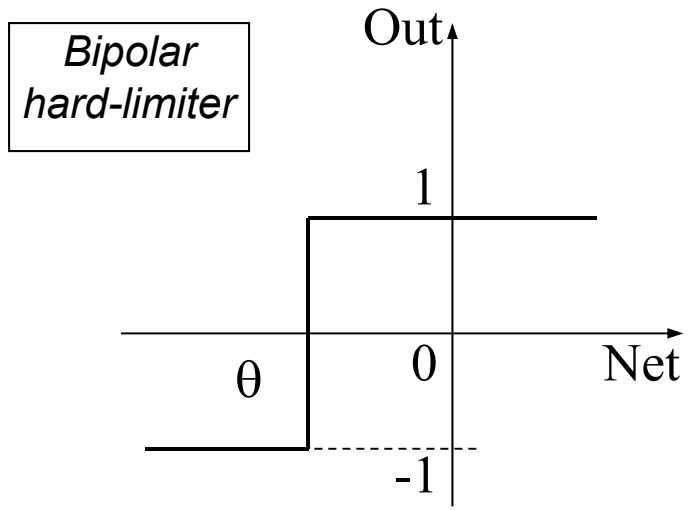
Hàm tác động: Giới hạn cứng (Hard-limiter)

- Còn được gọi là hàm ngưỡng (threshold function)
- Giá trị đầu ra lấy một trong 2 giá trị
- θ là giá trị ngưỡng
- **Nhược điểm**: không liên tục, đạo hàm không liên tục

$$Out(Net) = hl1(Net, \theta) = \begin{cases} 1, \text{nêu} & Net \geq \theta \\ 0, \text{nêu nguoc lai} & \end{cases}$$

$$Out(Net) = hl2(Net, \theta) = sign(Net, \theta)$$

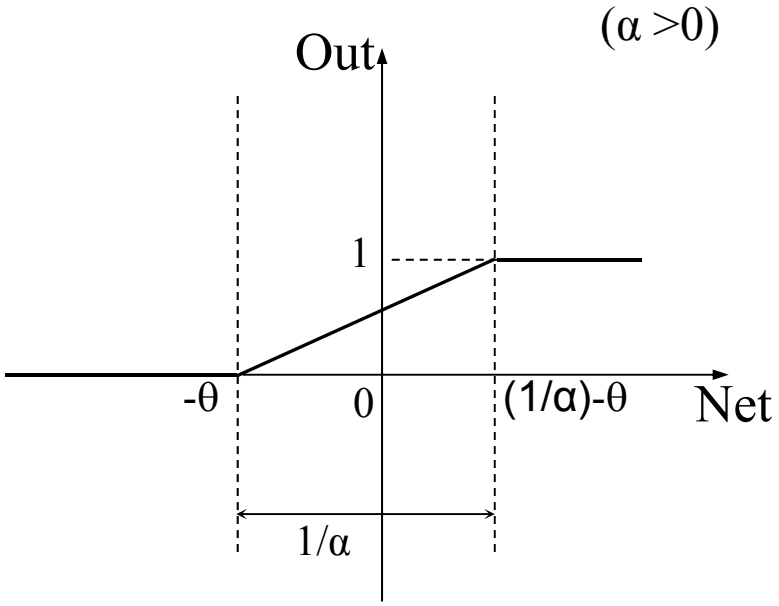




Hàm tác động: Logic ngưỡng (Threshold logic)

$$Out(Net) = tl(Net, \alpha, \theta) = \begin{cases} 0, & \text{if } Net < -\theta \\ \alpha(Net + \theta), & \text{if } -\theta \le Net \le \frac{1}{\alpha} - \theta \\ 1, & \text{if } Net > \frac{1}{\alpha} - \theta \end{cases}$$

$$= \max(0, \min(1, \alpha(Net + \theta)))$$

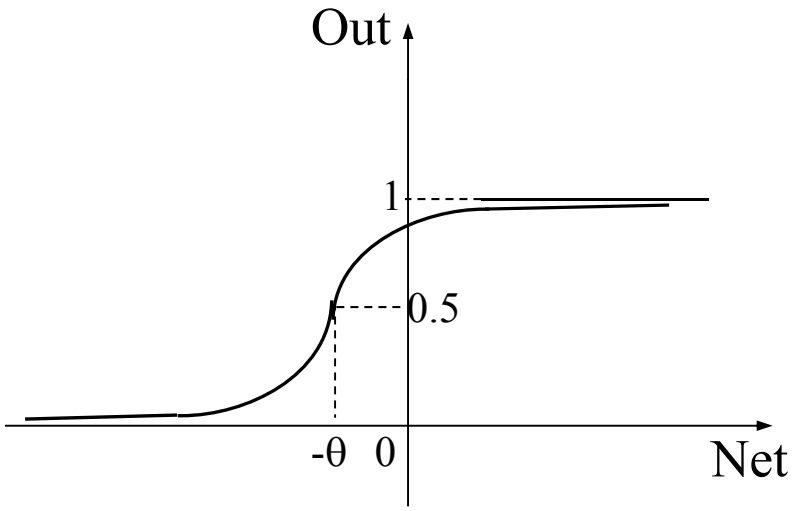


- Còn được gọi là hàm tuyến tính bão hòa (saturating linear function)
- Kết hợp của 2 hàm tác động: tuyến tính và giới hạn chặt
- α xác định độ dốc của khoảng tuyến tính
- **Nhược điểm**: Liên tục – nhưng đạo hàm không liên tục

Hàm tác động: Xích-ma (Sigmoidal)

$$Out(Net) = sf(Net, \alpha, \theta) = \frac{1}{1 + e^{-\alpha(Net+\theta)}}$$

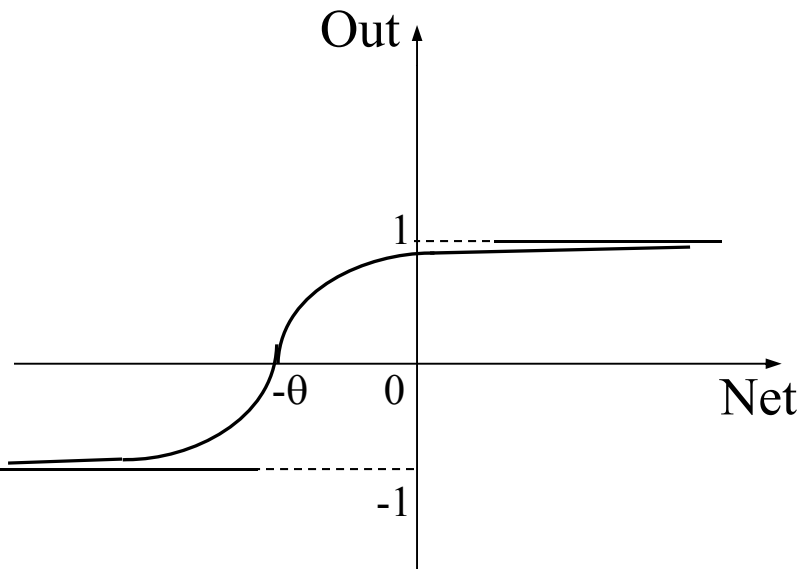
- Được dùng phổ biến nhất
- Tham số α xác định độ dốc
- Giá trị đầu ra trong khoảng (0,1)
- **Ưu điểm**
  - Liên tục, và đạo hàm liên tục
  - Đạo hàm của một hàm xích-ma được biểu diễn bằng một hàm của chính nó



Hàm tác động: Hyperbolic tangent

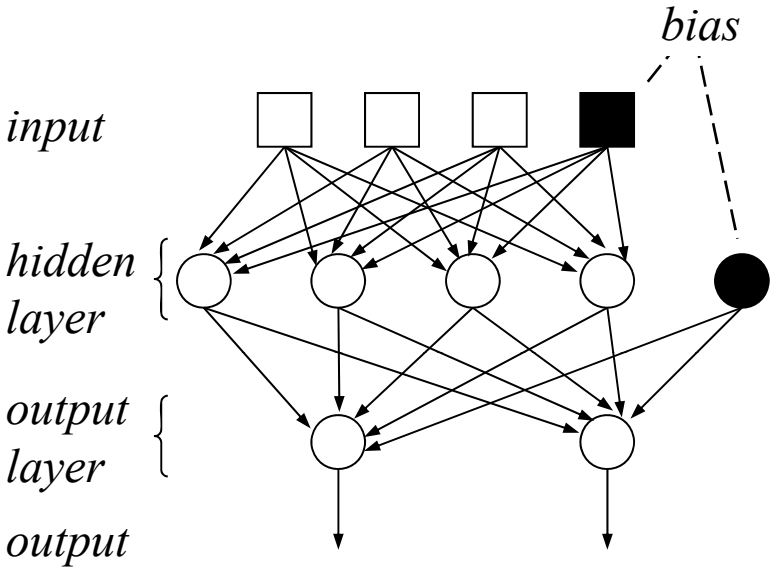
$$Out(Net) = \tanh(Net, \alpha, \theta) = \frac{1 - e^{-\alpha(Net+\theta)}}{1 + e^{-\alpha(Net+\theta)}} = \frac{2}{1 + e^{-\alpha(Net+\theta)}} - 1$$

- Cũng hay được sử dụng
- Tham số α xác định độ dốc
- Giá trị đầu ra trong khoảng (-1,1)
- **Ưu điểm**
  - Liên tục, và đạo hàm liên tục
  - Đạo hàm của một hàm `tanh` có thể được biểu diễn bằng một hàm của chính nó



ANN – Kiến trúc mạng (1)

- Kiến trúc của một ANN được x/đ bởi:
  - Số lượng các tín hiệu đầu vào và đầu ra
  - Số lượng các tầng
  - Số lượng các nơ-ron trong mỗi tầng
  - Số lượng các trọng số (các liên kết) đối với mỗi nơ-ron
  - Cách thức các nơ-ron (trong một tầng, hoặc giữa các tầng) liên kết với nhau
  - Những nơ-ron nào nhận các tín hiệu điều chỉnh lỗi
- Một ANN phải có
  - Một tầng đầu vào (input layer)
  - Một tầng đầu ra (output layer)
  - Không, một, hoặc nhiều tầng ẩn (hidden layer(s))



Ví dụ: Một ANN với một tầng ẩn

- Đầu vào: 3 tín hiệu
- Đầu ra: 2 giá trị
- Tổng cộng, có 6 neurons
  - 4 ở tầng ẩn
  - 2 ở tầng đầu ra

ANN – Kiến trúc mạng (2)

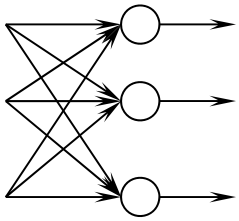
- Một tầng (layer) chứa một nhóm các nơ-ron
- Tầng ẩn (hidden layer) là một tầng nằm ở giữa tầng đầu vào (input layer) và tầng đầu ra (output layer)
- Các nút ở tầng ẩn (hidden nodes) không tương tác trực tiếp với môi trường bên ngoài (của mạng nơ-ron)
- Một ANN được gọi là ***liên kết đầy đủ (fully connected)*** nếu mọi đầu ra từ một tầng liên kết với mọi nơ-ron của tầng kế tiếp

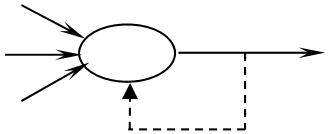
ANN – Kiến trúc mạng (3)

- Một ANN được gọi là ***mạng lan truyền tiến (feed-forward network)*** nếu không có bất kỳ đầu ra của một nút là đầu vào của một nút khác thuộc cùng tầng (hoặc thuộc một tầng phía trước)
- Khi các đầu ra của một nút liên kết ngược lại làm các đầu vào của một nút thuộc cùng tầng (hoặc thuộc một tầng phía trước), thì đó là một ***mạng phản hồi (feedback network)***
  - Nếu phản hồi là liên kết đầu vào đối với các nút thuộc cùng tầng, thì đó là ***phản hồi bên (lateral feedback)***
- Các mạng phản hồi có các vòng lặp kín (closed loops) được gọi là **các mạng hồi quy (*recurrent networks)***

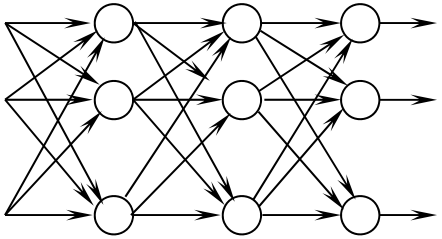
Kiến trúc mạng – Ví dụ

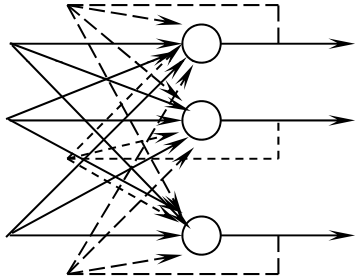
Mạng lan truyền tiến một tầng





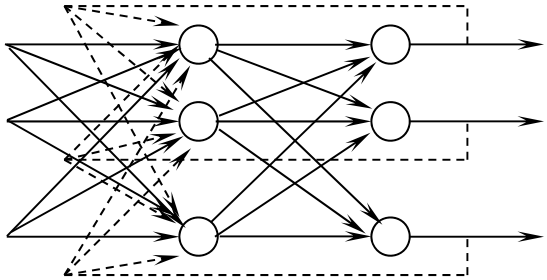
Một nơ-ron với phản hồi đến chính nó





Mạng hồi quy một tầng

Mạng lan truyền tiến nhiều tầng



Mạng hồi quy nhiều tầng

ANN – Các quy tắc học

- 2 kiểu học trong các mạng nơ-ron nhân tạo
  - *Học tham số (Parameter learning)*
    - → Mục tiêu là thay đổi thích nghi các trọng số (weights) của các liên kết trong mạng nơ-ron
  - *Học cấu trúc (Structure learning)*
    - → Mục tiêu là thay đổi thích nghi cấu trúc mạng, bao gồm số lượng các nơ-ron và các kiểu liên kết giữa chúng
- 2 kiểu học này có thể được thực hiện đồng thời hoặc riêng rẽ
- Phần lớn các quy tắc học trong ANN thuộc kiểu học tham số
- Trong bài học này, chúng ta sẽ chỉ xét việc học tham số

Quy tắc học trọng số tổng quát

- Tại bước học (*t*), mức độ điều chỉnh vec-tơ trọng số ***w*** tỷ lệ thuận với tích của tín hiệu học $r^{(t)}$ và đầu vào $\boldsymbol{x}^{(t)}$

  $\Delta \mathbf{w}^{(t)} \sim r^{(t)}.\mathbf{x}^{(t)}$

  $\Delta \mathbf{w}^{(t)} = \eta.r^{(t)}.\mathbf{x}^{(t)}$

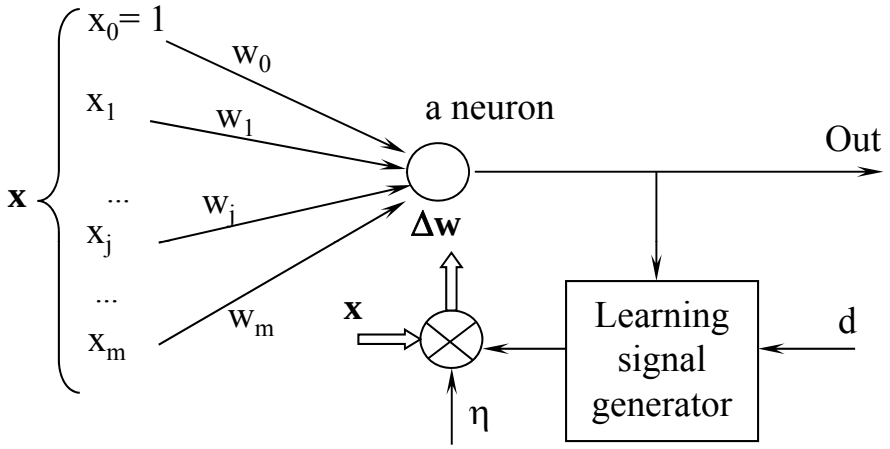
  trong đó η **(>0) là tốc độ học** (learning rate)

- Tín hiệu học *r* là một hàm của ***w***, ***x***, và giá trị đầu ra mong muốn *d*

  $r = g(\mathbf{w},\mathbf{x},d)$

- Quy tắc học trọng số tổng quát

  $\Delta \mathbf{w}^{(t)} = \eta.g(\mathbf{w}^{(t)},\mathbf{x}^{(t)},d^{(t)}).\mathbf{x}^{(t)}$



Lưu ý: x_j có thể là:

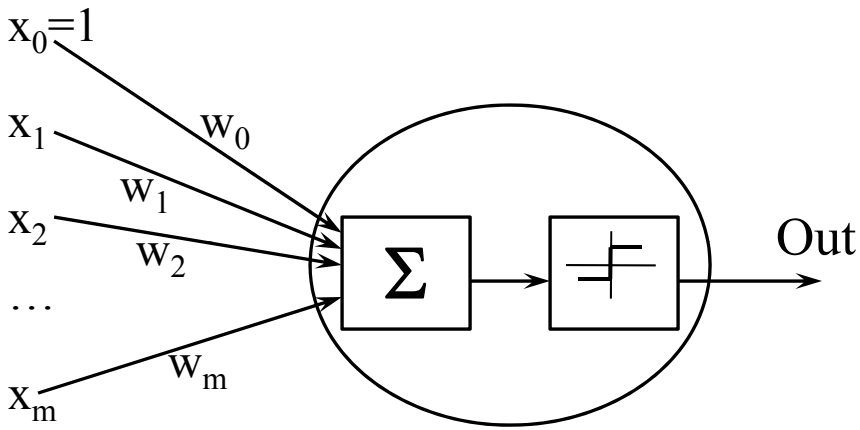
- một tín hiệu đầu vào, hoặc
- một giá trị đầu ra của một nơ-ron khác

Perceptron

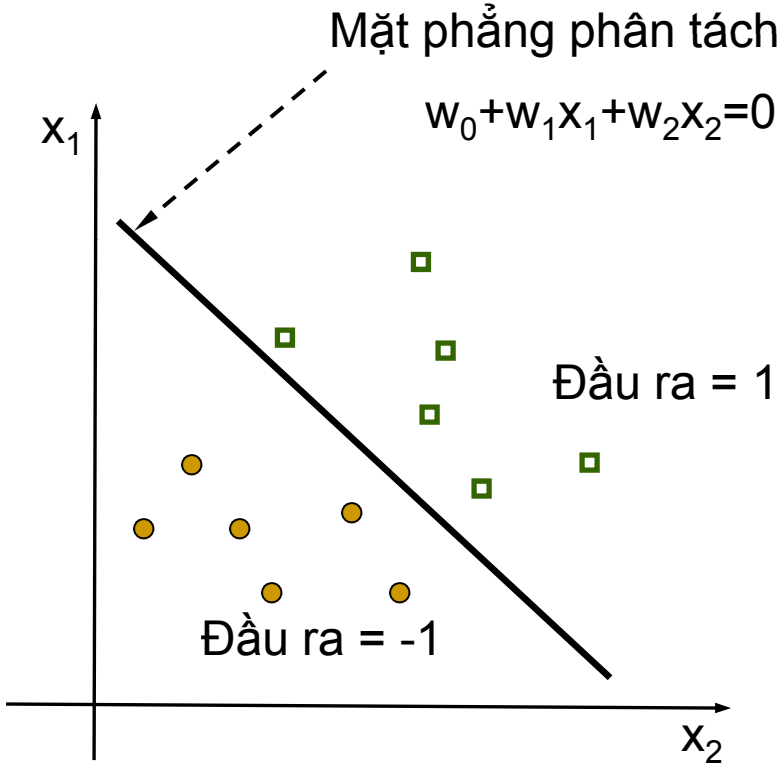
- Một perceptron là một kiểu đơn giản nhất của ANNs (chỉ gồm duy nhất một nơ-ron)
- Sử dụng hàm tác động giới hạn chặt

$$Out = sign\big(Net(w,x)\big) = sign\left(\sum_{j=0}^{m} w_j x_j\right)$$

- Đối với một ví dụ ***x***, giá trị đầu ra của perceptron là
  - 1, nếu Net(**w**,**x**)>0
  - -1, nếu ngược lại



Perceptron – Minh họa



Perceptron – Giải thuật học

- Với một tập các ví dụ học D= {(**x**,d)}
  - ***x*** là vectơ đầu vào
  - *d* là giá trị đầu ra mong muốn (-1 hoặc 1)
- Quá trình học của perceptron nhằm xác định một vectơ trọng số cho phép perceptron sinh ra giá trị đầu ra chính xác (-1 hoặc 1) cho mỗi ví dụ học
- Với một ví dụ học ***x*** được perceptron phân lớp chính xác, thì vectơ trọng số ***w*** không thay đổi
- Nếu d=1 nhưng perceptron lại sinh ra -1 (Out=-1), thì ***w*** cần được thay đổi sao cho giá trị Net(**w**,**x**) tăng lên
- Nếu d=-1 nhưng perceptron lại sinh ra 1 (Out=1), thì ***w*** cần được thay đổi sao cho giá trị Net(**w**,**x**) giảm đi

Perceptron_incremental(D, η)

Initialize **w** ($w_i \leftarrow$ an initial (small) random value)

do

  for each training instance $(\mathbf{x}, d) \in D$

    Compute the real output value Out

    if (Out≠d)

      $\mathbf{w} \leftarrow \mathbf{w} + \eta(d - Out)\mathbf{x}$

  end for

until all the training instances in D are correctly classified

return **w**

Perceptron_batch(D, η)

Initialize **w** ($w_i \leftarrow$ an initial (small) random value)

do

  $\Delta\mathbf{w} \leftarrow 0$

  for each training instance $(\mathbf{x}, d) \in D$

    Compute the real output value Out

    if (Out≠d)

      $\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \eta(d-Out)\mathbf{x}$

  end for

  $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$

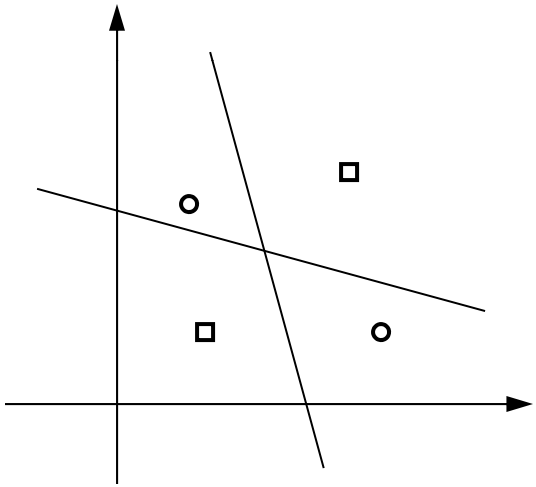
until all the training instances in D are correctly classified

return **w**

Perceptron – Giới hạn

- Giải thuật học cho perceptron được chứng minh là hội tụ (converge) nếu:
  - Các ví dụ học là có thể phân tách tuyến tính (linearly separable)
  - Sử dụng một tốc độ học η đủ nhỏ
- Giải thuật học perceptron có thể không hội tụ nếu như các ví dụ học không thể phân tách tuyến tính (not linearly separable)
- Khi đó, áp dụng **quy tắc delta (delta rule)**
  - Đảm bảo hội tụ về một xấp xỉ phù hợp nhất của hàm mục tiêu
  - Quy tắc delta sử dụng chiến lược **gradient descent** để tìm trong không gian giả thiết (các vectơ trọng số) một vectơ trọng số phù hợp nhất với các ví dụ học

Một perceptron không thể phân lớp chính xác đối với tập học này!



Hàm đánh giá lỗi (Error function)

- Xét một ANN có *n* nơ-ron đầu ra
- Đối với một ví dụ học (**x**,d), giá trị **lỗi học (training error)** gây ra bởi vectơ trọng số (hiện tại) **w**:

$$E_{\mathbf{x}}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(d_i - Out_i\right)^2$$

- **Lỗi học** gây ra bởi vectơ trọng số (hiện tại) **w** đối với toàn bộ tập học *D*:

$$E_D(\mathbf{w}) = \frac{1}{|D|}\sum_{\mathbf{x}\in D} E_{\mathbf{x}}(\mathbf{w})$$

Gradient descent

- **Gradient** của *E* (ký hiệu là ∇E) là một vectơ
  - Có hướng chỉ đi lên (dốc)
  - Có độ dài tỷ lệ thuận với độ dốc
- Gradient ∇E xác định hướng gây ra việc **tăng nhanh nhất (steepest increase)** đối với giá trị lỗi *E*

$$\nabla E(\mathbf{w}) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_N} \right)$$

trong đó *N* là tổng số các trọng số (các liên kết) trong mạng

- Vì vậy, hướng gây ra việc **giảm nhanh nhất (steepest decrease)** là giá trị phủ định của gradient của *E*

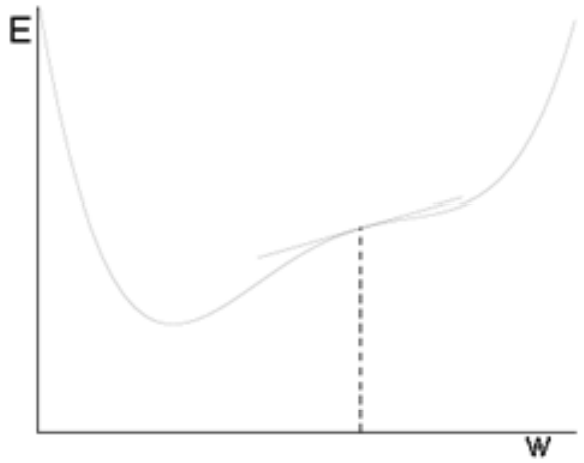
$$\Delta \mathbf{w} = -\eta . \nabla \mathrm{E}(\mathbf{w}); \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \quad \forall i = 1..N$$

- Yêu cầu: Các hàm tác động được sử dụng trong mạng phải là các hàm liên tục đối với các trọng số, và có đạo hàm liên tục

Gradient descent – Minh họa

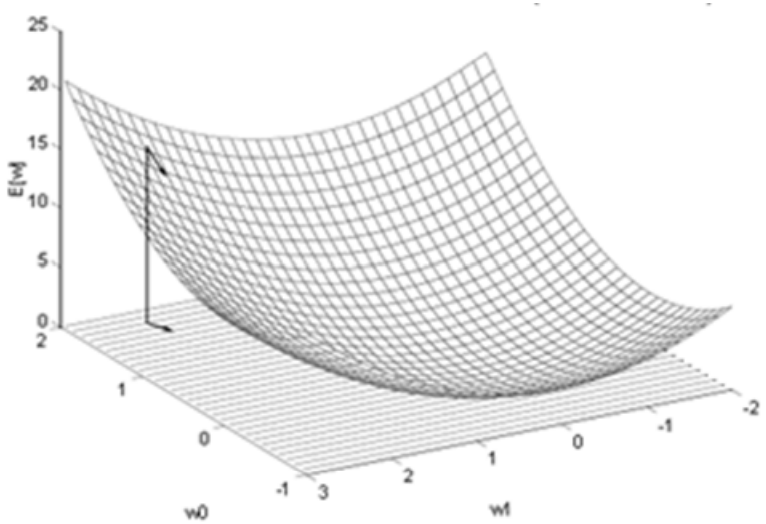
Không gian một chiều

E(w)



Không gian 2 chiều

$E(w_1,w_2)$



Gradient_descent_incremental (D, η)

Initialize $\mathbf{w}$ ($w_i \leftarrow$ an initial (small) random value)

do

  for each training instance $(\mathbf{x}, d) \in D$

    Compute the network output

    for each weight component w_i

      $w_i \leftarrow w_i - \eta(\partial E_{\mathbf{x}}/\partial w_i)$

    end for

  end for

until (stopping criterion satisfied)

return $\mathbf{w}$

Stopping criterion: Số chu kỳ học (epochs), Ngưỡng lỗi, ...

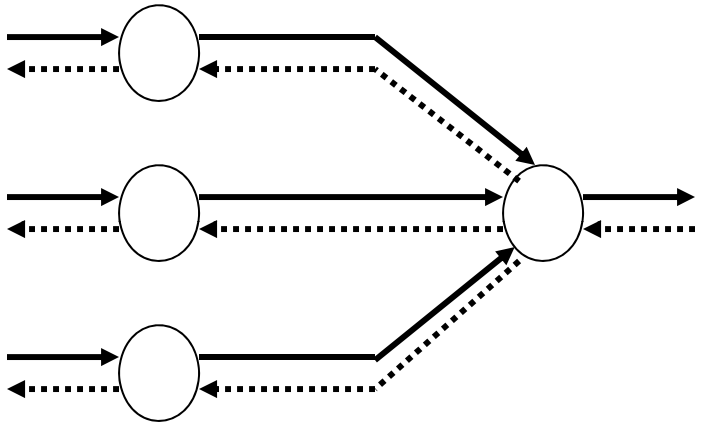
ANN nhiều tầng và giải thuật lan truyền ngược

- Một perceptron chỉ có thể biểu diễn một hàm phân tách tuyến tính (linear separation function)
- Một mạng nơ-ron nhiều tầng (multi-layer NN) được học bởi giải thuật lan truyền ngược (back-propagation -BP- algorithm) có thể biểu diễn một hàm phân tách phi tuyến phức tạp (highly non-linear separation function)
- Giải thuật học BP được sử dụng để học các trọng số của một mạng nơ-ron nhiều tầng
  - *Cấu trúc mạng cố định* (các nơ-ron và các liên kết giữa chúng là cố định)
  - Đối với mỗi nơ-ron, *hàm tác động phải có đạo hàm liên tục*
- Giải thuật BP áp dụng chiến lược *gradient descent* trong quy tắc cập nhật các trọng số
  - Để cực tiểu hóa lỗi (khác biệt) giữa các giá trị đầu ra thực tế và các giá trị đầu ra mong muốn, đối với các ví dụ học

Giải thuật học lan truyền ngược (1)

- Giải thuật học lan truyền ngược tìm kiếm một vectơ các trọng số (weights vector) giúp **cực tiểu hóa lỗi tổng thể** của hệ thống đối với tập học
- Giải thuật BP bao gồm 2 giai đoạn (bước)
  - Giai đoạn **lan truyền tiến tín hiệu (Signal forward)**. Các tín hiệu đầu vào (vectơ các giá trị đầu vào) được lan truyền tiến từ tầng đầu vào đến tầng đầu ra (đi qua các tầng ẩn)
  - Giai đoạn **lan truyền ngược lỗi (Error backward**)
    - Căn cứ vào giá trị đầu ra mong muốn của vectơ đầu vào, hệ thống tính toán giá trị lỗi
    - Bắt đầu từ tầng đầu ra, giá trị lỗi được lan truyền ngược qua mạng, từ tầng này qua tầng khác (phía trước), cho đến tầng đầu vào
    - Việc lan truyền ngược lỗi (error back-propagation) được thực hiện thông qua việc tính toán (một cách truy hồi) giá trị gradient cục bộ của mỗi nơ-ron

Giải thuật học lan truyền ngược (2)



Giai đoạn lan truyền tiến tín hiệu:
- Kích hoạt (truyền tín hiệu qua) mạng

Giai đoạn lan truyền ngược lỗi:
- Tính toán lỗi ở đầu ra
- Lan truyền (ngược) lỗi

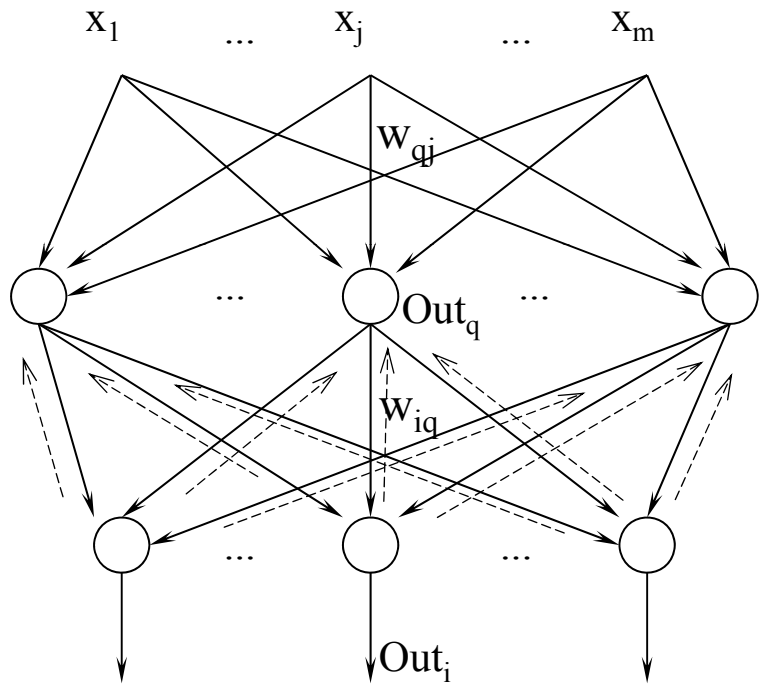
Giải thuật BP – Cấu trúc mạng

- Xét mạng nơ-ron 3 tầng (trong hình vẽ) để minh họa giải thuật học BP
- m tín hiệu đầu vào x_j (j=1..m)
- l nơ-ron tầng ẩn z_q (q=1..l)
- n nơ-ron đầu ra y_i (i=1..n)
- w_{qj} là trọng số của liên kết từ tín hiệu đầu vào x_j tới nơ-ron tầng ẩn z_q
- w_{iq} là trọng số của liên kết từ nơ-ron tầng ẩn z_q tới nơ-ron đầu ra y_i
- Out_q là giá trị *đầu ra (cục bộ)* của nơ-ron tầng ẩn z_q
- Out_i là giá trị *đầu ra của mạng* tương ứng với nơ-ron đầu ra y_i

Input x_j (j=1..m)

Hidden neuron z_q (q=1..l)

Output neuron y_i (i=1..n)



Giải thuật BP – Lan truyền tiến (1)

- Đối với mỗi ví dụ học ***x***
  - Vectơ đầu vào ***x*** được *lan truyền* từ tầng đầu vào đến tầng đầu ra
  - Mạng sẽ sinh ra một giá trị đầu ra thực tế (actual output) ***Out*** (là một vectơ của các giá trị Out_i, i=1..n)
- Đối với một vectơ đầu vào ***x***, một nơ-ron z_q ở tầng ẩn sẽ nhận được giá trị đầu vào tổng thể (net input) bằng:

$$Net_q = \sum_{j=1}^{m} w_{qj} x_j$$

…và sinh ra một giá trị đầu ra (cục bộ) bằng:

$$Out_q = f(Net_q) = f\left(\sum_{j=1}^{m} w_{qj} x_j\right)$$

trong đó f(.) là hàm tác động (activation function) của nơ-ron z_q

Giải thuật BP – Lan truyền tiến (2)

- Giá trị đầu vào tổng thể (net input) của nơ-ron y_i ở tầng đầu ra

$$Net_i = \sum_{q=1}^{l} w_{iq} Out_q = \sum_{q=1}^{l} w_{iq} f\left(\sum_{j=1}^{m} w_{qj} x_j \right)$$

- Nơ-ron y_i sinh ra giá trị đầu ra (là một giá trị đầu ra của mạng)

$$Out_i = f(Net_i) = f\left(\sum_{q=1}^{l} w_{iq} Out_q \right) = f\left(\sum_{q=1}^{l} w_{iq} f\left(\sum_{j=1}^{m} w_{qj} x_j \right) \right)$$

- Vectơ các giá trị đầu ra Out_i (i=1..n) chính là giá trị đầu ra thực tế của mạng, đối với vectơ đầu vào ***x***

Giải thuật BP – Lan truyền ngược (1)

- Đối với mỗi ví dụ học ***x***
  - Các tín hiệu lỗi (error signals) do sự khác biết giữa giá trị đầu ra mong muốn ***d*** và giá trị đầu ra thực tế ***Out*** được tính toán
  - Các tín hiệu lỗi này được *lan truyền ngược (back-propagated)* từ tầng đầu ra tới các tầng phía trước, để cập nhật các trọng số (weights)

- Để xét các tín hiệu lỗi và việc lan truyền ngược của chúng, cần định nghĩa một hàm đánh giá lỗi

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}\left(d_i - Out_i\right)^2 = \frac{1}{2}\sum_{i=1}^{n}\left[d_i - f\left(Net_i\right)\right]^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}\left[d_i - f\left(\sum_{q=1}^{l} w_{iq} Out_q\right)\right]^2$$

Giải thuật BP – Lan truyền ngược (2)

- Theo phương pháp gradient-descent, các trọng số của các liên kết **từ tầng ẩn tới tầng đầu ra** được cập nhật bởi

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

- Sử dụng quy tắc chuỗi đạo hàm đối với $\partial E/\partial w_{iq}$, ta có

$$\Delta w_{iq} = -\eta \left[\frac{\partial E}{\partial Out_i}\right]\left[\frac{\partial Out_i}{\partial Net_i}\right]\left[\frac{\partial Net_i}{\partial w_{iq}}\right] = \eta\left[d_i - Out_i\right]\left[f'\left(Net_i\right)\right]\left[Out_q\right] = \eta\delta_i Out_q$$

(Lưu ý: dấu "–" đã được kết hợp với giá trị $\partial E/\partial Out_i$)

- δ_i là **tín hiệu lỗi (error signal)** của nơ-ron y_i ở tầng đầu ra

$$\delta_i = -\frac{\partial E}{\partial Net_i} = -\left[\frac{\partial E}{\partial Out_i}\right]\left[\frac{\partial Out_i}{\partial Net_i}\right] = \left[d_i - Out_i\right]\left[f'\left(Net_i\right)\right]$$

trong đó Net_i là đầu vào tổng thể (net input) của nơ-ron y_i ở tầng đầu ra, và $f'(\text{Net}_i)=\partial f(\text{Net}_i)/\partial \text{Net}_i$

Giải thuật BP – Lan truyền ngược (3)

- Để cập nhật các trọng số của các liên kết **từ tầng đầu vào tới tầng ẩn**, chúng ta cũng áp dụng phương pháp gradient-descent và quy tắc chuỗi đạo hàm

$$\Delta w_{qj} = -\eta \frac{\partial E}{\partial w_{qj}} = -\eta \left[\frac{\partial E}{\partial Out_q} \right] \left[\frac{\partial Out_q}{\partial Net_q} \right] \left[\frac{\partial Net_q}{\partial w_{qj}} \right]$$

- Từ công thức tính hàm lỗi $E(\mathbf{w})$, ta thấy rằng mỗi thành phần lỗi $(d_i - y_i)$ $(i=1..n)$ là một hàm của Out_q

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \left[d_i - f\left(\sum_{q=1}^{l} w_{iq} Out_q \right) \right]^2$$

Giải thuật BP – Lan truyền ngược (4)

- Áp dụng quy tắc chuỗi đạo hàm, ta có

$$\Delta w_{qj} = \eta \sum_{i=1}^{n} \left[(d_i - Out_i) f'(Net_i) w_{iq} \right] f'(Net_q) x_j$$

$$= \eta \sum_{i=1}^{n} \left[\delta_i w_{iq} \right] f'(Net_q) x_j = \eta \delta_q x_j$$

- δ_q là **tín hiệu lỗi (error signal)** của nơ-ron z_q <u>ở tầng ẩn</u>

$$\delta_q = -\frac{\partial E}{\partial Net_q} = -\left[\frac{\partial E}{\partial Out_q} \right] \left[\frac{\partial Out_q}{\partial Net_q} \right] = f'(Net_q) \sum_{i=1}^{n} \delta_i w_{iq}$$

trong đó Net_q là đầu vào tổng thể (net input) của nơ-ron z_q ở tầng ẩn, và $f'(\text{Net}_q) = \partial f(\text{Net}_q)/\partial \text{Net}_q$

Giải thuật BP – Lan truyền ngược (5)

- Theo các công thức tính các tín hiệu lỗi δ_i và δ_q đã nêu, thì *tín hiệu lỗi của một nơ-ron ở tầng ẩn* **khác với** *tín hiệu lỗi của một nơ-ron ở tầng đầu ra*
- Do sự khác biệt này, thủ tục cập nhật trọng số trong giải thuật BP được gọi là *quy tắc học delta tổng quát*
- Tín hiệu lỗi δ_q của nơ-ron z_q ở tầng ẩn được xác định bởi
  - Các tín hiệu lỗi δ_i của các nơ-ron y_i ở tầng đầu ra (mà nơ-ron z_q liên kết tới), và
  - Các hệ số chính là các trọng số w_{iq}
- Đặc điểm quan trọng của giải thuật BP: **Quy tắc cập nhật trọng số có tính cục bộ**
  - Để tính toán thay đổi (cập nhật) trọng số của một liên kết, hệ thống chỉ cần sử dụng các giá trị ở 2 đầu của liên kết đó!

Giải thuật BP – Lan truyền ngược (6)

- Quá trình tính toán tín hiệu lỗi (error signals) như trên có thể được mở rộng (khái quát) dễ dàng đối với mạng nơ-ron có nhiều hơn 1 tầng ẩn
- Dạng tổng quát của quy tắc cập nhật trọng số trong giải thuật BP là:

$$\Delta w_{ab} = \eta \delta_a x_b$$

  - *b* và *a* là 2 chỉ số tương ứng với 2 đầu của liên kết ($b \rightarrow a$) (từ một nơ-ron (hoặc tín hiệu đầu vào) *b* đến nơ-ron *a*)
  - x_b là giá trị đầu ra của nơ-ron ở tầng ẩn (hoặc tín hiệu đầu vào) *b*,
  - δ_a là tín hiệu lỗi của nơ-ron *a*

Back_propagation_incremental(D, η)

Mạng nơ-ron gồm Q tầng, *q = 1,2,...,Q*

$^{q}Net_i$ và $^{q}Out_i$ là đầu vào tổng thể (net input) và giá trị đầu ra của nơ-ron *i* ở tầng *q*

Mạng có *m* tín hiệu đầu vào và *n* nơ-ron đầu ra

$^{q}w_{ij}$ là trọng số của liên kết từ nơ-ron *j* ở tầng *(q-1)* đến nơ-ron *i* ở tầng *q*

Bước 0 (Khởi tạo)

Chọn ngưỡng lỗi $E_{threshold}$ (giá trị lỗi có thể chấp nhận được)

Khởi tạo giá trị ban đầu của các trọng số với các giá trị nhỏ ngẫu nhiên

Gán E=0

Bước 1 (Bắt đầu một chu kỳ học)

Áp dụng vectơ đầu vào của ví dụ học *k* đối với tầng đầu vào (*q*=1)

$^{q}Out_i = {}^{1}Out_i = x_i^{(k)}, \forall i$

Bước 2 (Lan truyền tiến)

Lan truyền tiến các tín hiệu đầu vào qua mạng, cho đến khi nhận được các giá trị đầu ra của mạng (ở tầng đầu ra) $^{Q}Out_i$

$$^{q}Out_i = f\left(^{q}Net_i\right) = f\left(\sum_j {}^{q}w_{ij}\, {}^{q-1}Out_j\right)$$

Bước 3 (Tính toán lỗi đầu ra)

Tính toán lỗi đầu ra của mạng và tín hiệu lỗi ${}^{Q}\delta_i$ của mỗi nơ-ron ở tầng đầu ra

$$E = E + \frac{1}{2}\sum_{i=1}^{n}(d_i^{(k)} - {}^{Q}Out_i)^2$$

$${}^{Q}\delta_i = (d_i^{(k)} - {}^{Q}Out_i) f'({}^{Q}Net_i)$$

Bước 4 (Lan truyền ngược lỗi)

Lan truyền ngược lỗi để cập nhật các trọng số và tính toán các tín hiệu lỗi ${}^{q-1}\delta_i$ cho các tầng phía trước

$$\Delta^{q}w_{ij} = \eta.({}^{q}\delta_i).({}^{q-1}Out_j); \qquad {}^{q}w_{ij} = {}^{q}w_{ij} + \Delta^{q}w_{ij}$$

$${}^{q-1}\delta_i = f'({}^{q-1}Net_i)\sum_j {}^{q}w_{ji}\,{}^{q}\delta_j; \quad \text{for all } q = Q, Q-1, ..., 2$$

Bước 5 (Kiểm tra kết thúc một chu kỳ học – epoch)

Kiểm tra xem toàn bộ tập học đã được sử dụng (đã xong một chu kỳ học – epoch)

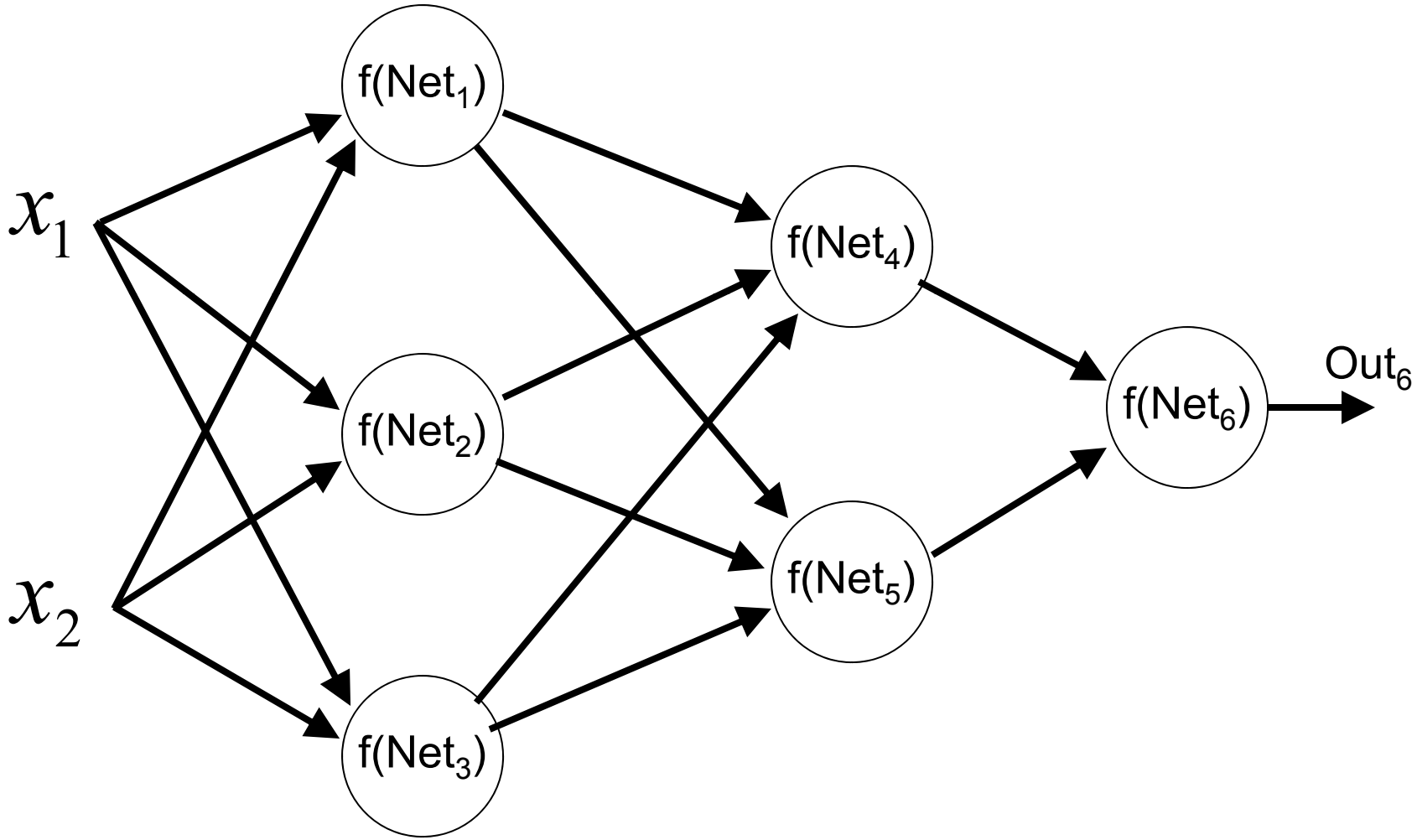
Nếu toàn bộ tập học đã được dùng, chuyển đến Bước 6; ngược lại, chuyển đến Bước 1

Bước 6 (Kiểm tra lỗi tổng thể)

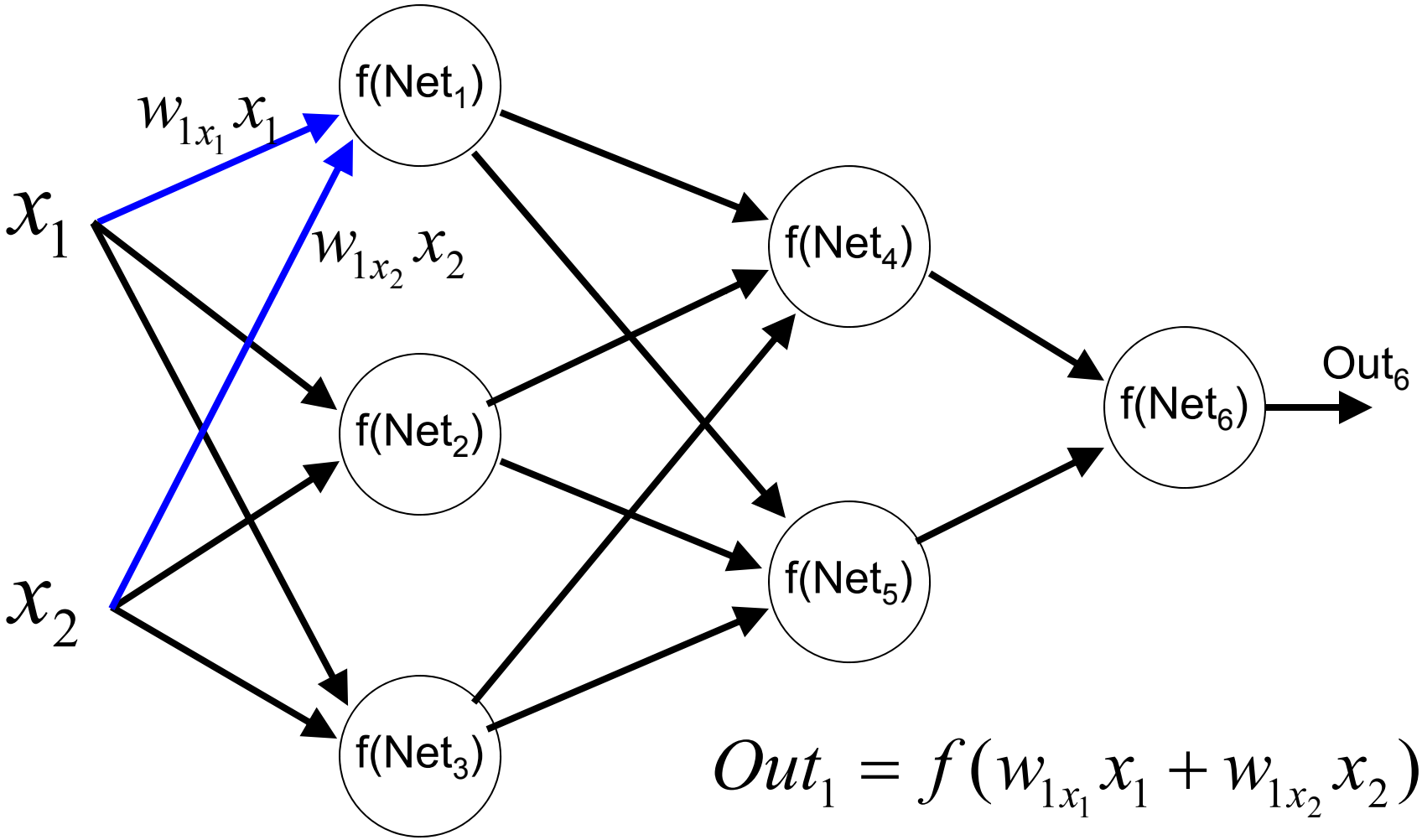
Nếu lỗi tổng thể E nhỏ hơn ngưỡng lỗi chấp nhận được ($<E_{threshold}$), thì quá trình học kết thúc và trả về các trọng số học được;

Ngược lại, gán lại E=0, và bắt đầu một chu kỳ học mới (quay về Bước 1)

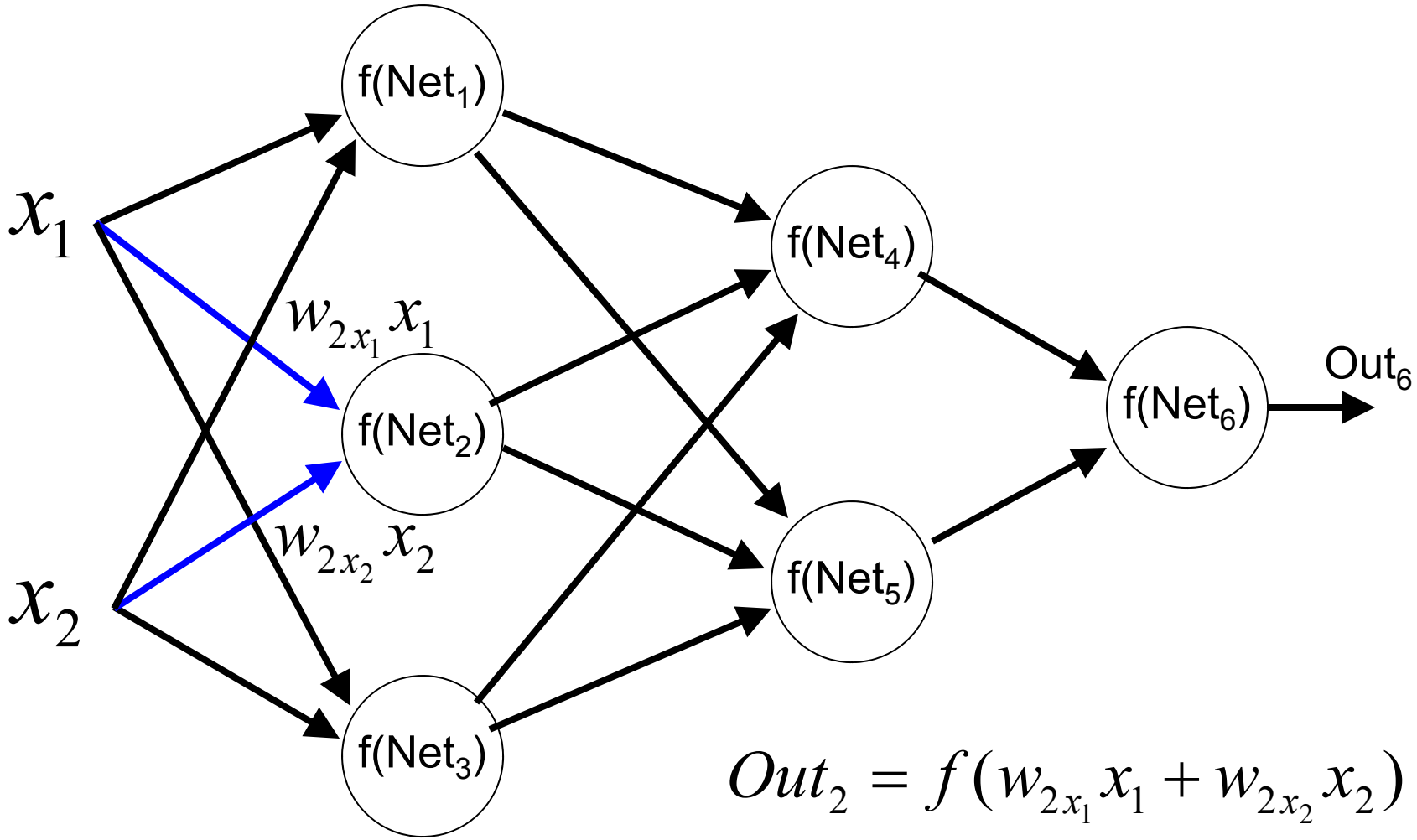
Giải thuật BP – Lan truyền tiến (1)



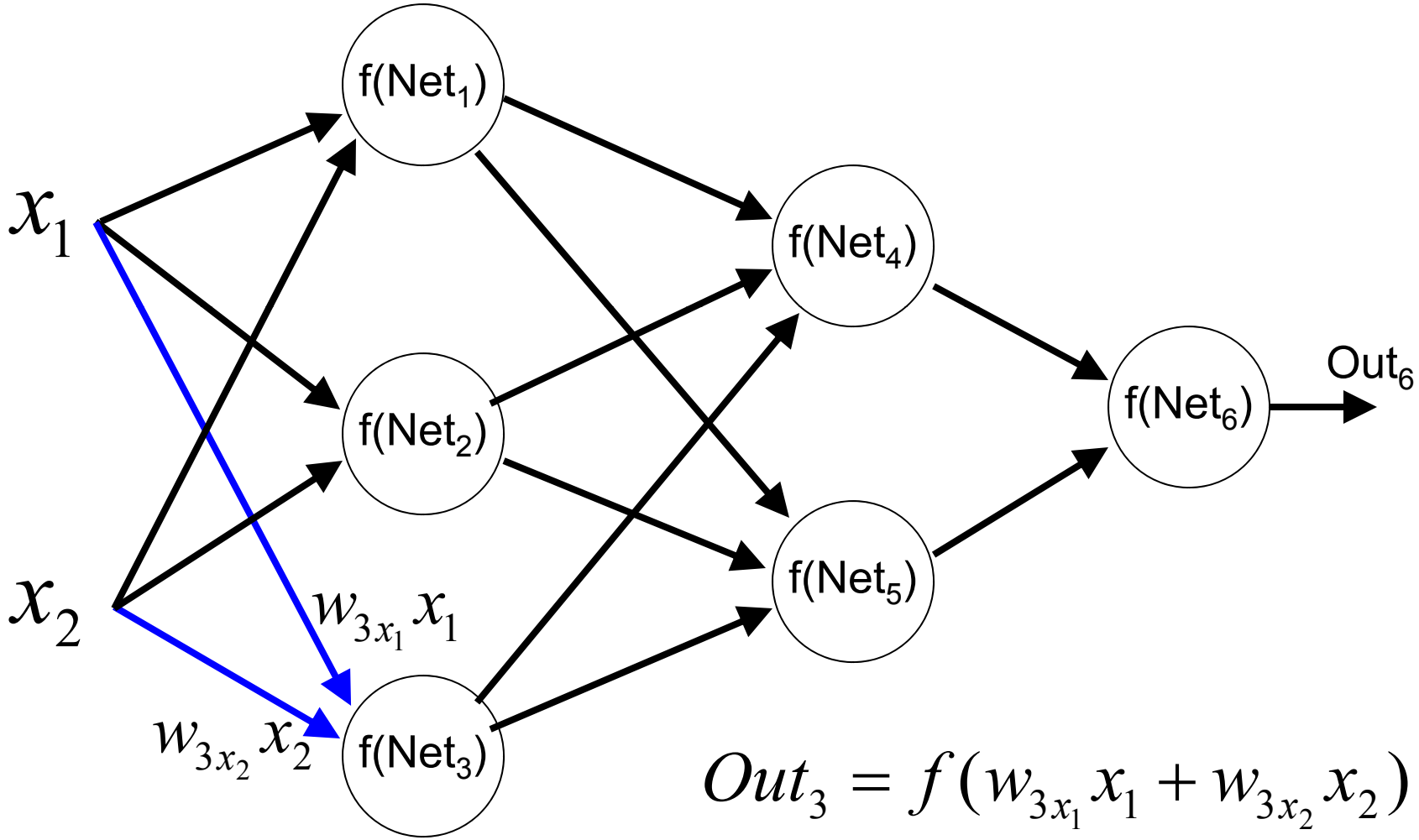
Giải thuật BP – Lan truyền tiến (2)



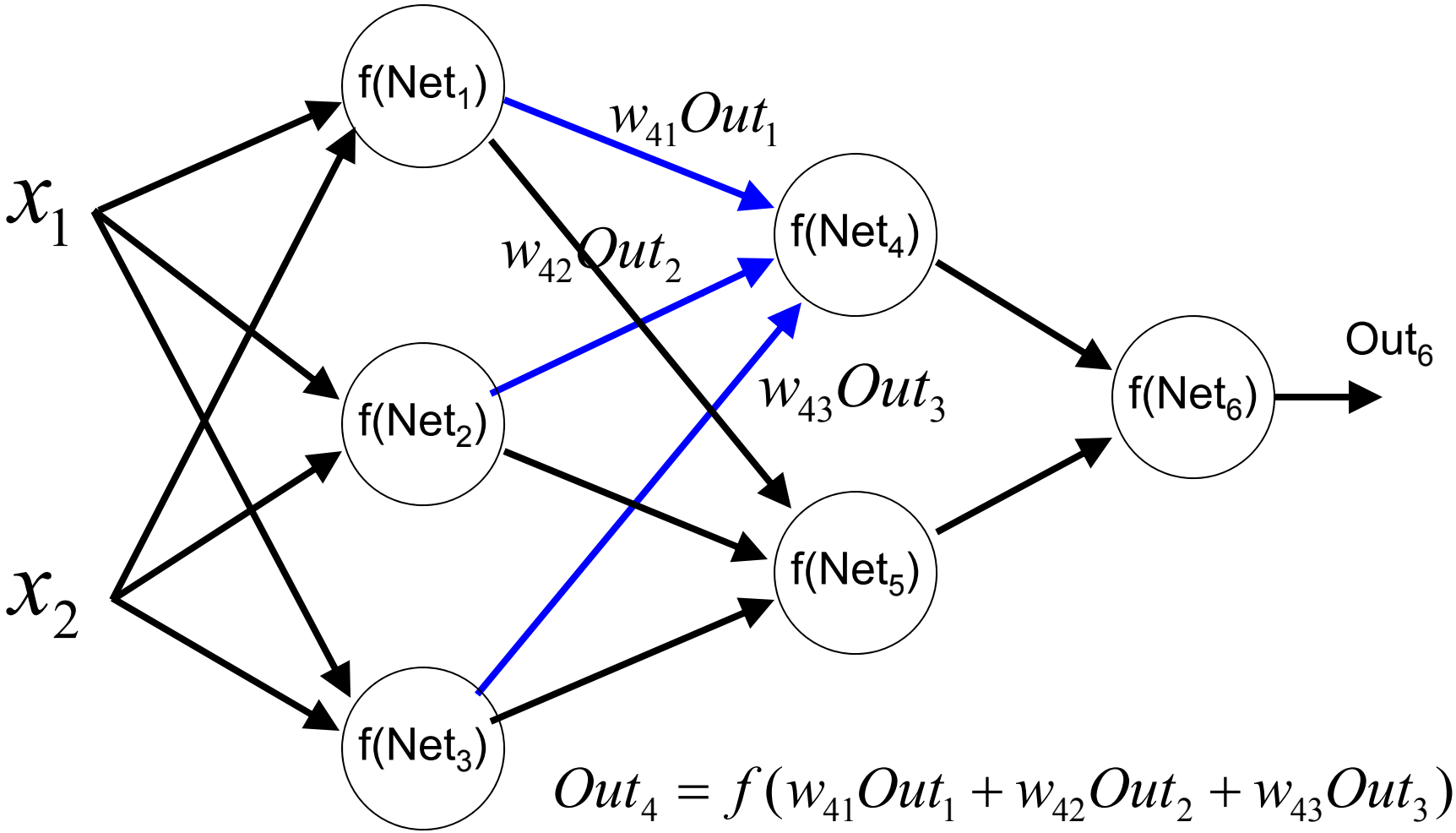
Giải thuật BP – Lan truyền tiến (3)



Giải thuật BP – Lan truyền tiến (4)

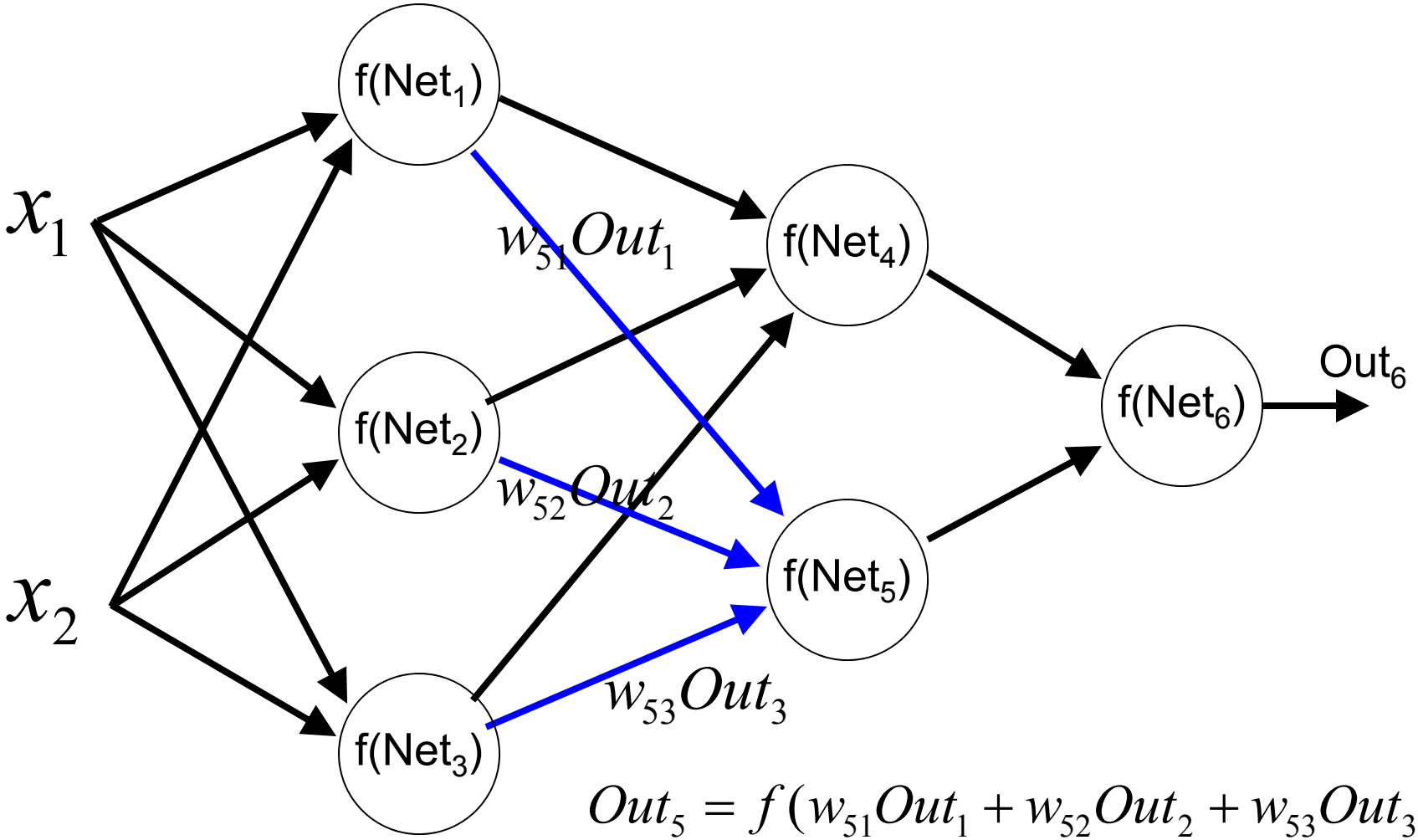


Giải thuật BP – Lan truyền tiến (5)



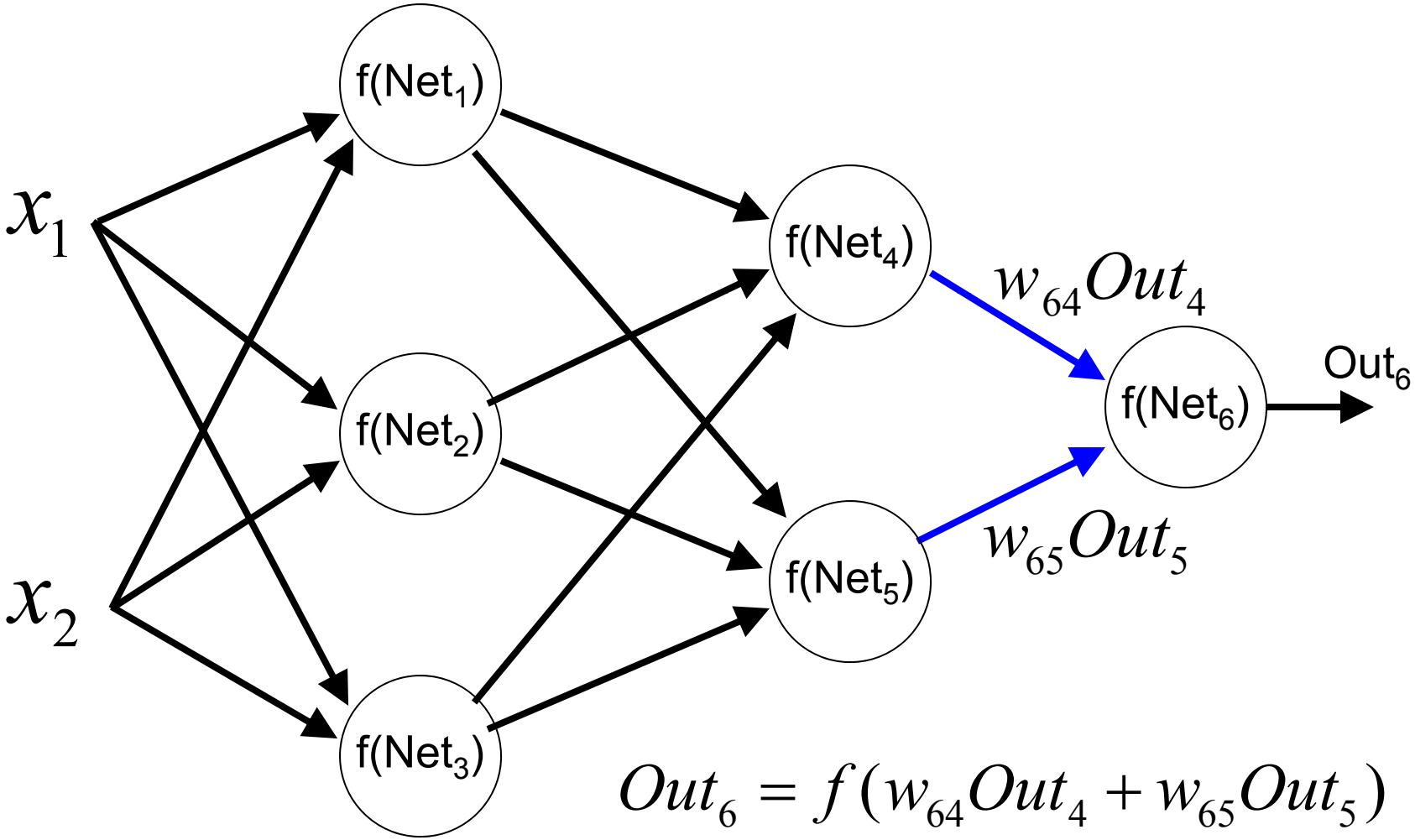
$$Out_4 = f(w_{41}Out_1 + w_{42}Out_2 + w_{43}Out_3)$$

Giải thuật BP – Lan truyền tiến (6)

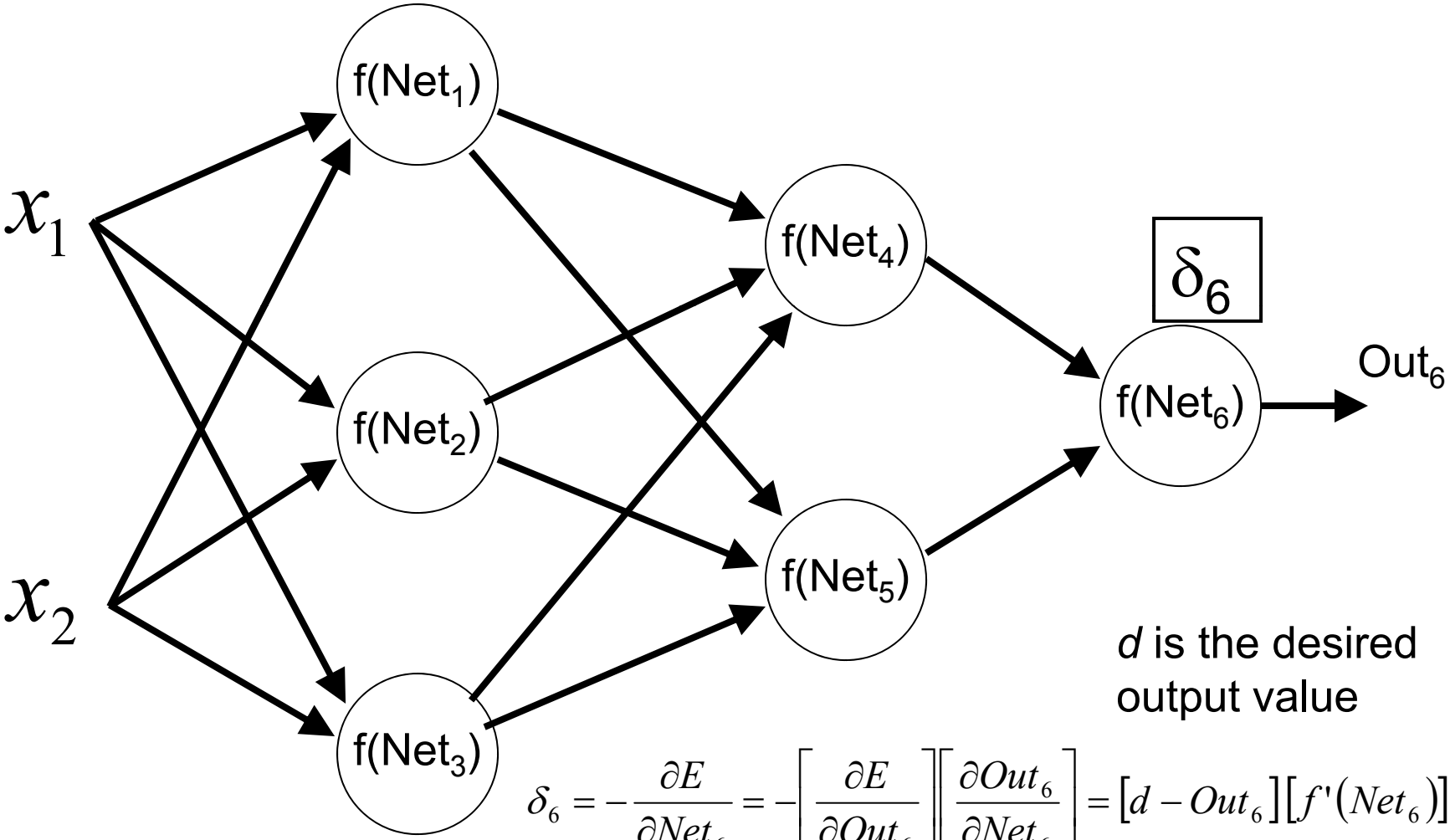


$$Out_5 = f(w_{51}Out_1 + w_{52}Out_2 + w_{53}Out_3)$$

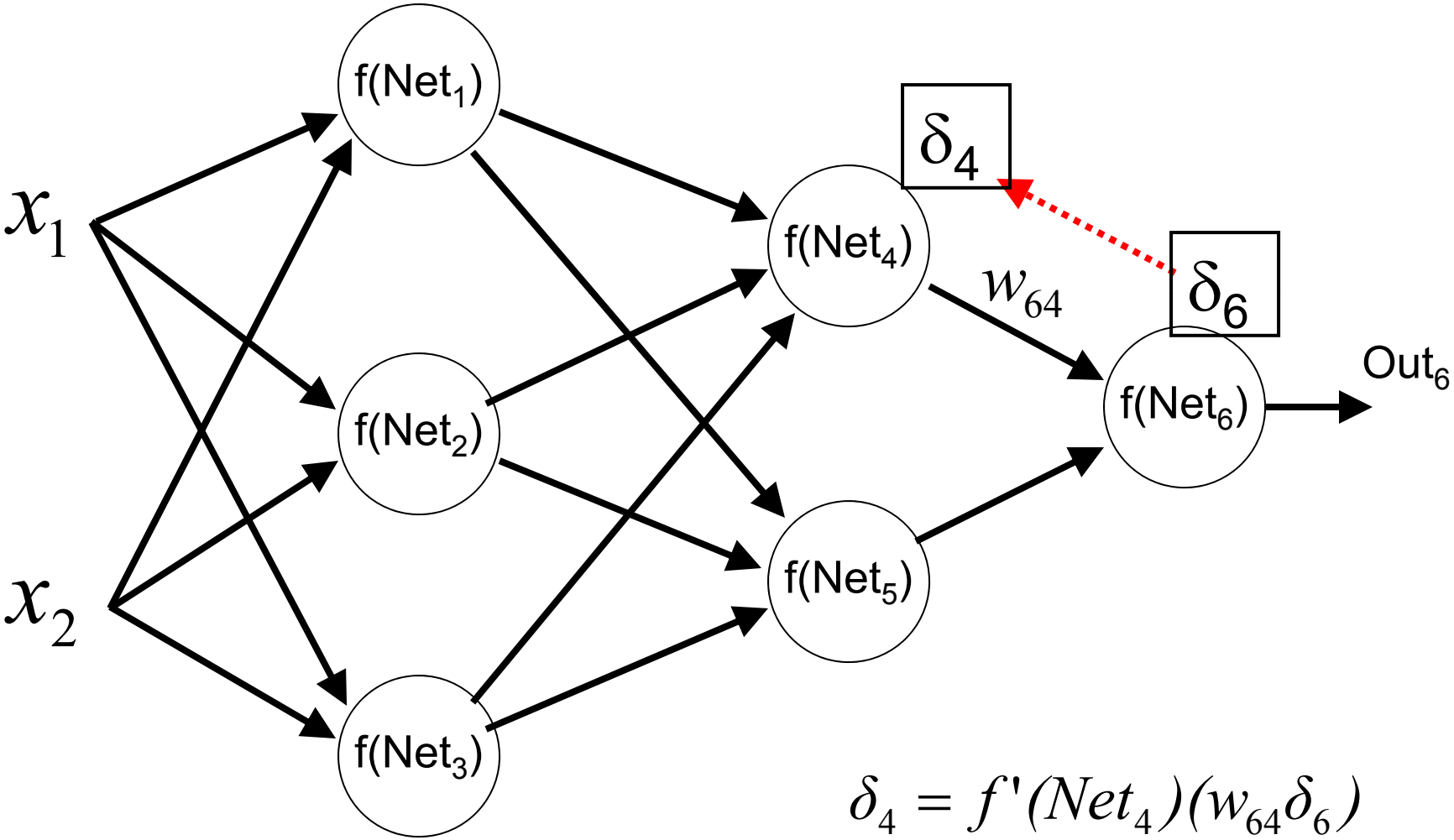
Giải thuật BP – Lan truyền tiến (7)



Giải thuật BP – Tính toán lỗi

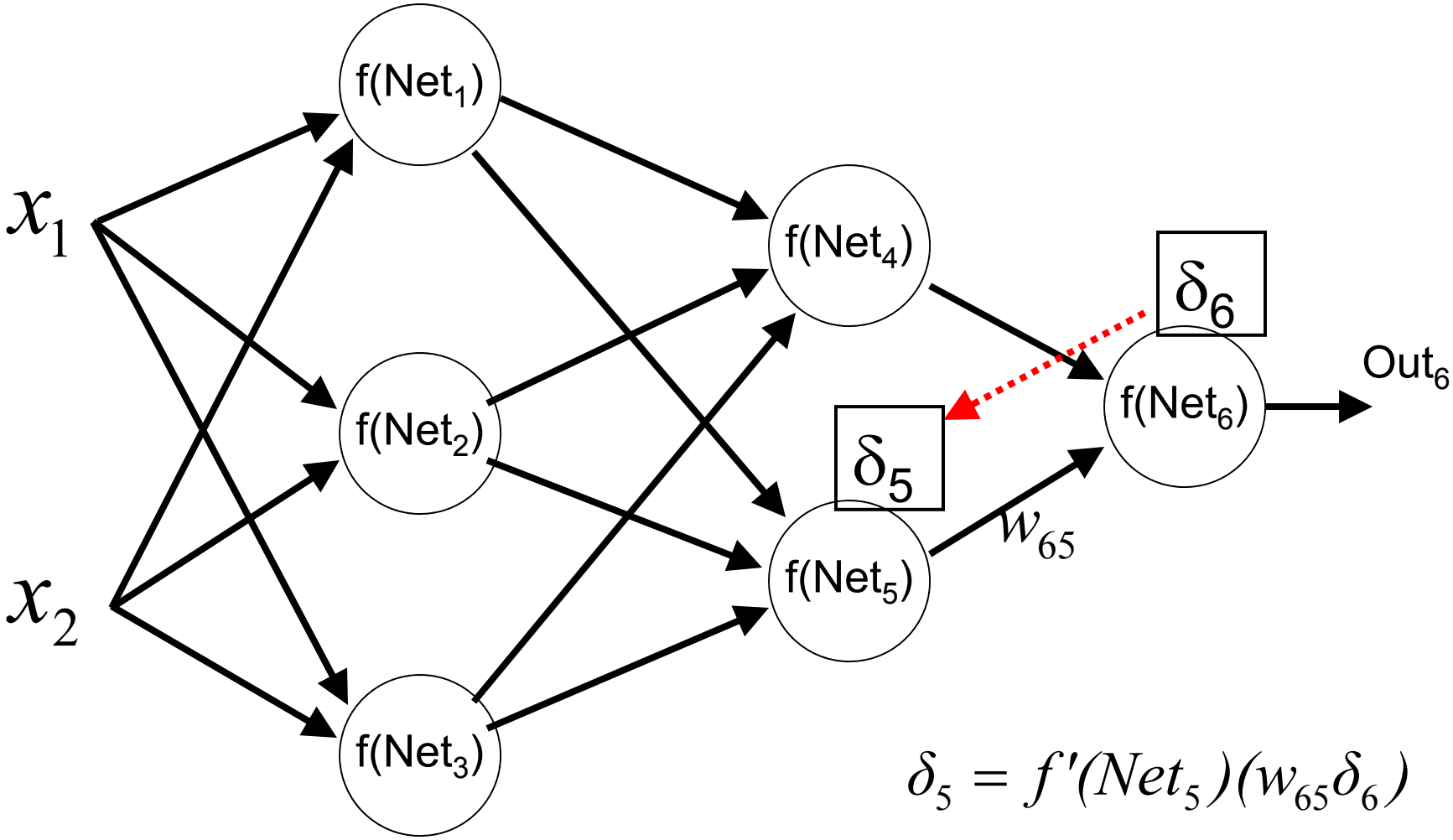


Giải thuật BP – Lan truyền ngược (1)

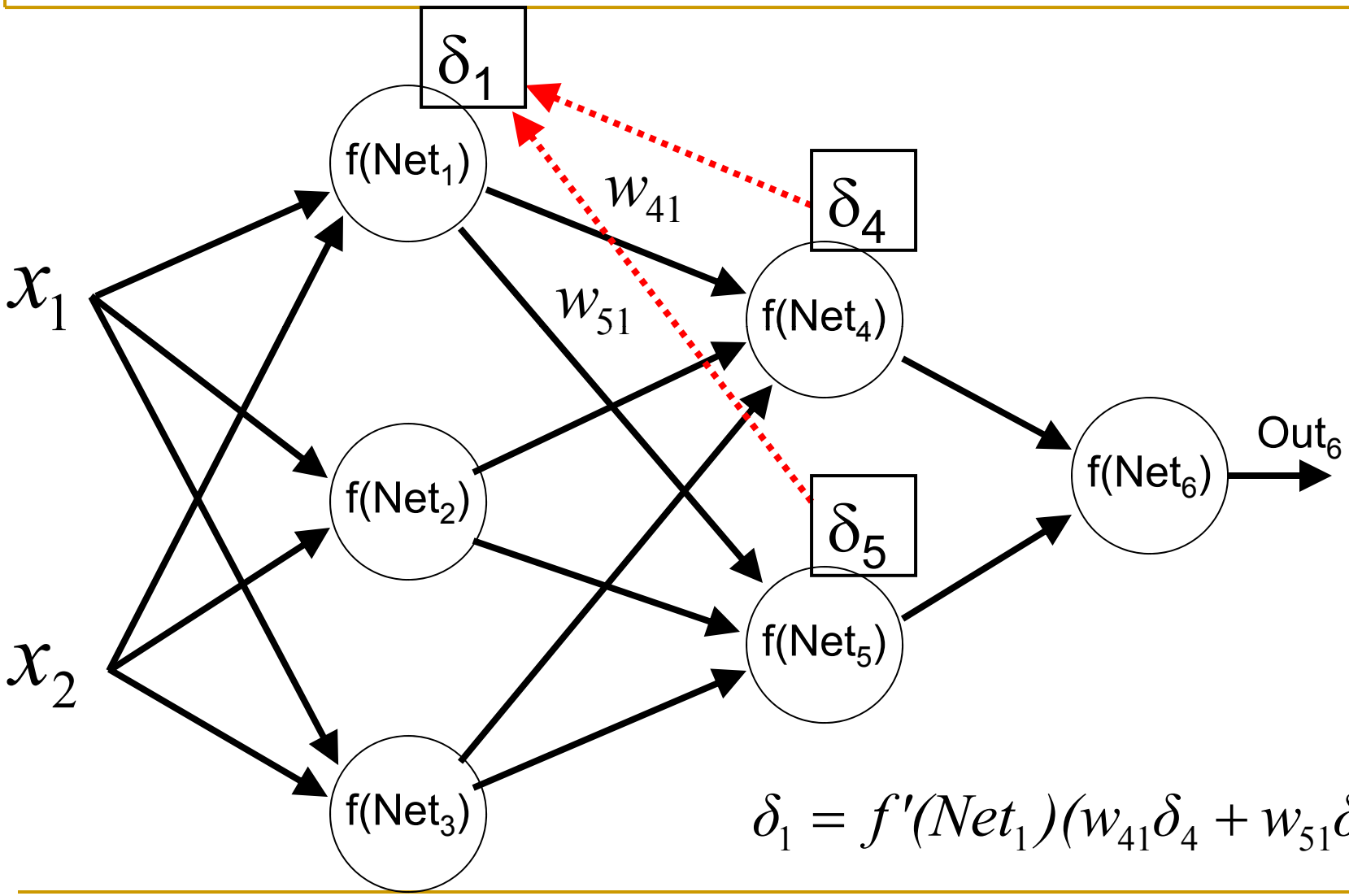


$$\delta_4 = f'(Net_4)(w_{64}\delta_6)$$

Giải thuật BP – Lan truyền ngược (2)

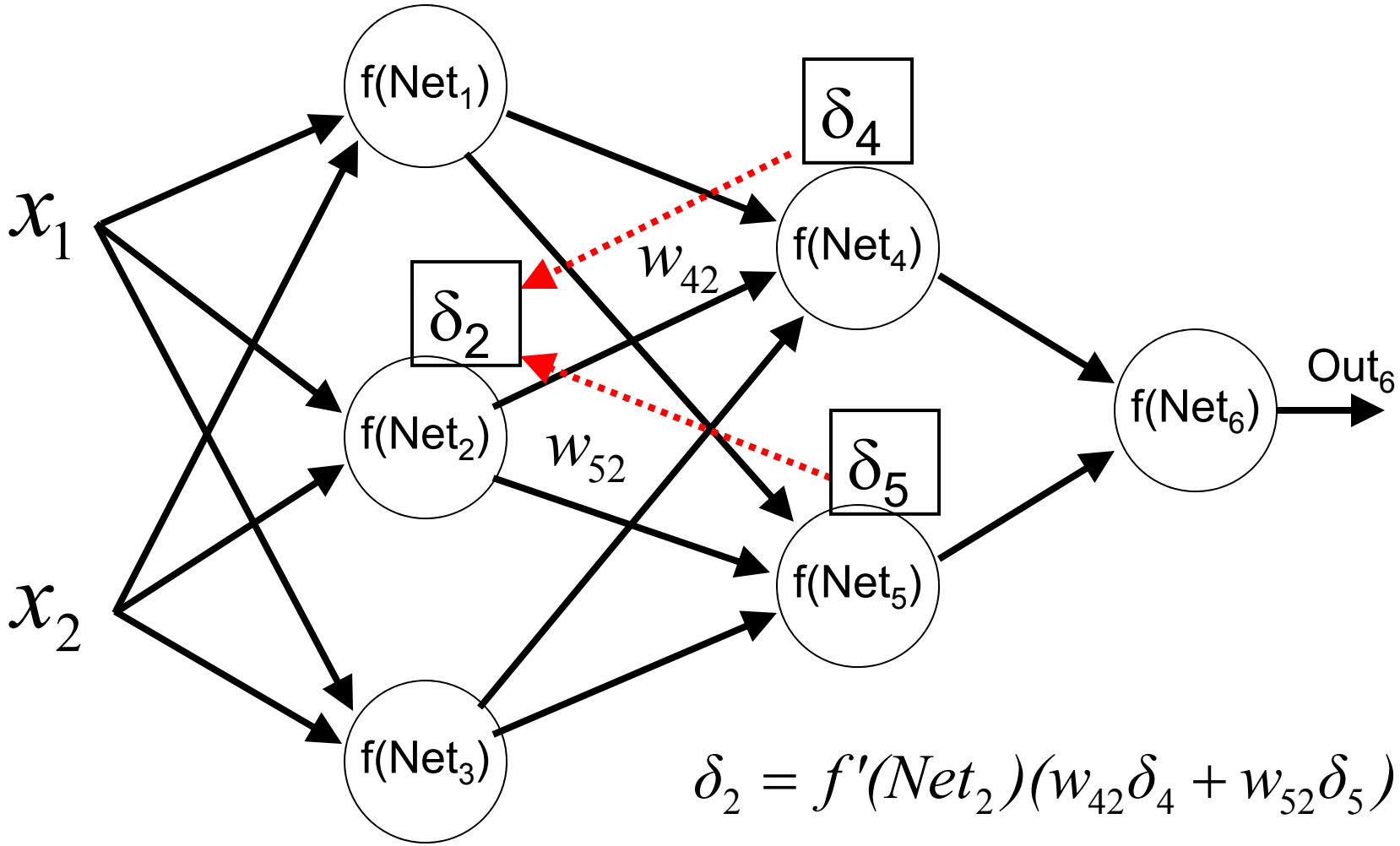


Giải thuật BP – Lan truyền ngược (3)

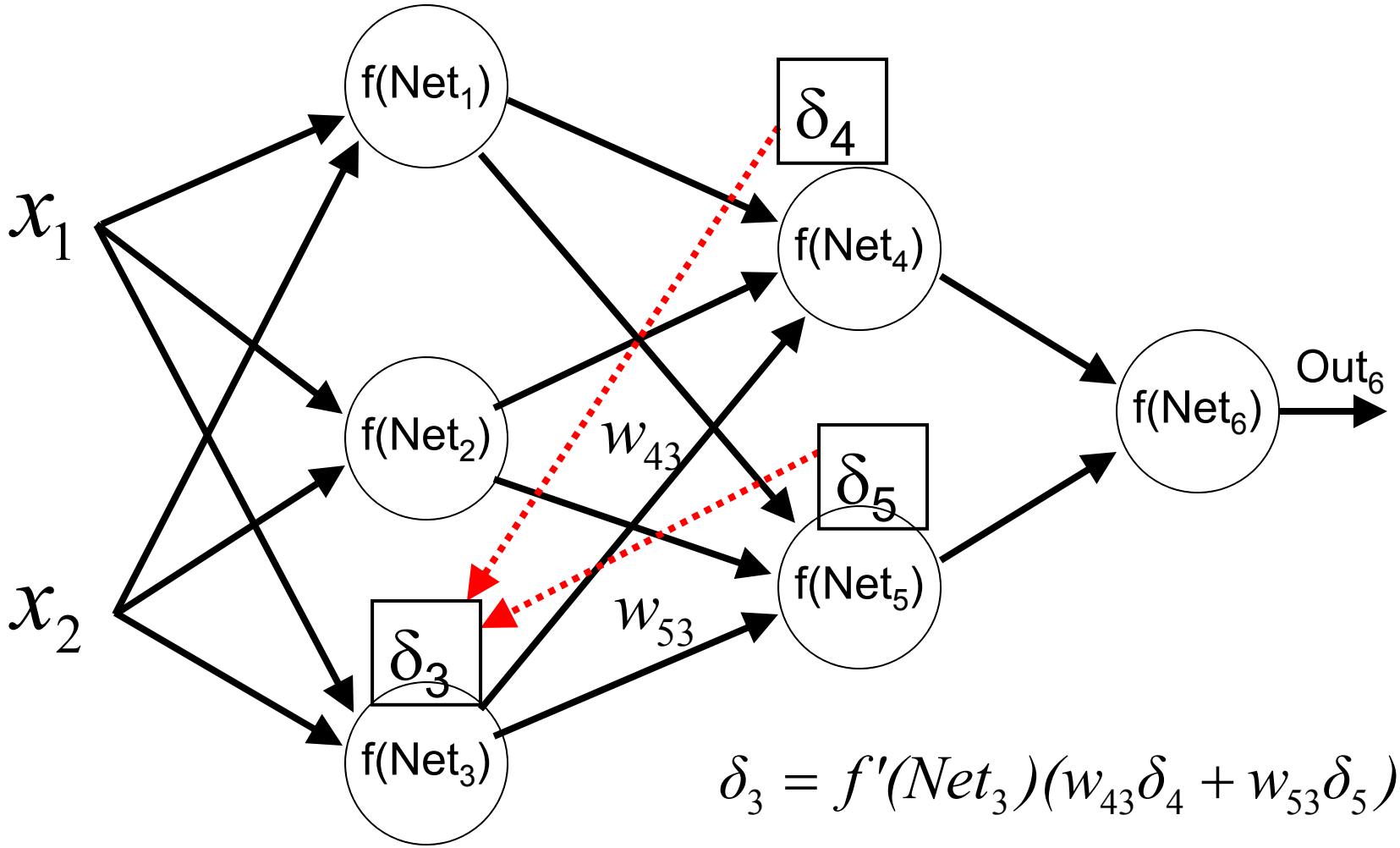


$$\delta_1 = f'(Net_1)(w_{41}\delta_4 + w_{51}\delta_5)$$

Giải thuật BP – Lan truyền ngược (4)

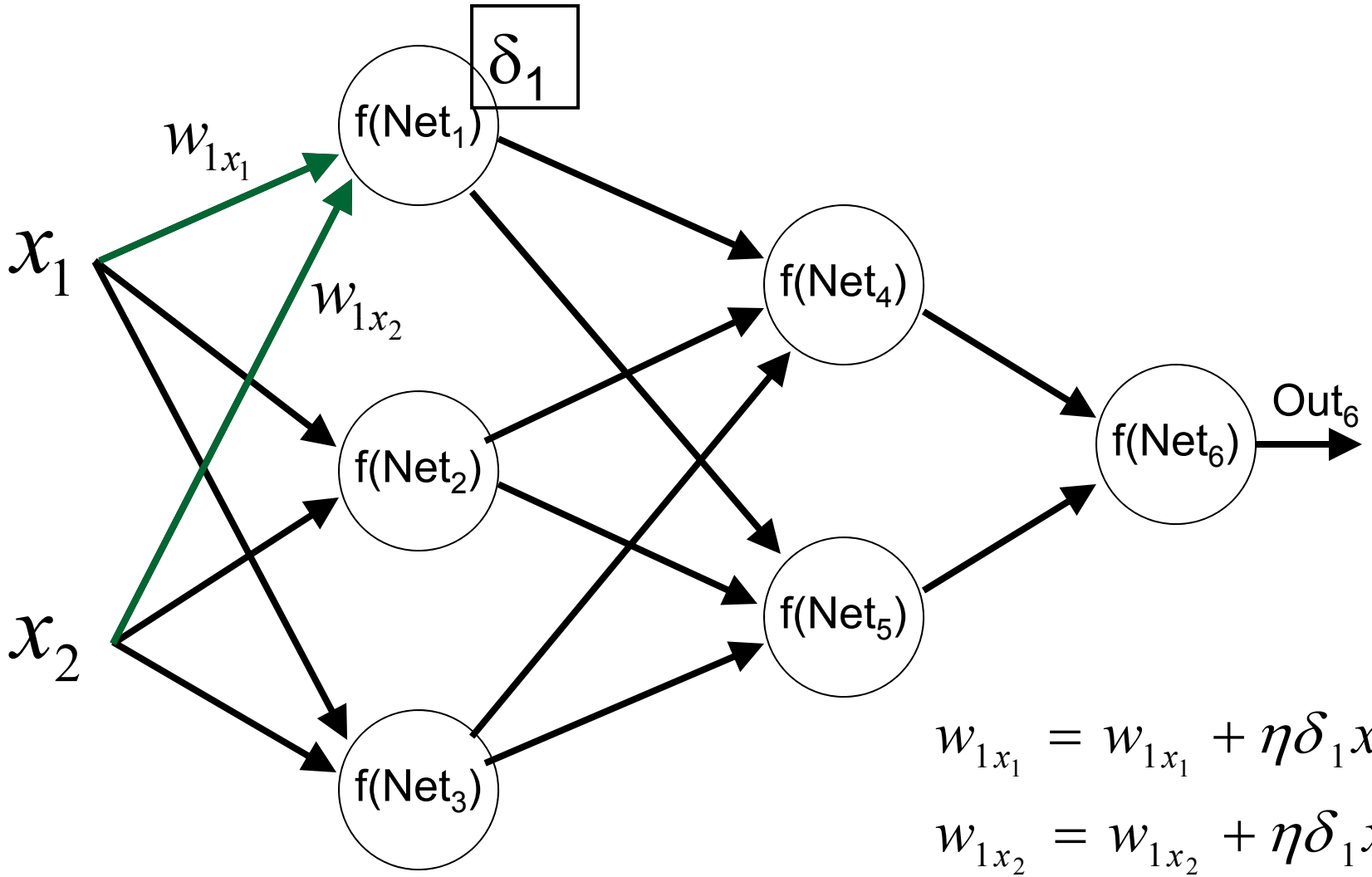


Giải thuật BP – Lan truyền ngược (5)



$$\delta_3 = f'(Net_3)(w_{43}\delta_4 + w_{53}\delta_5)$$

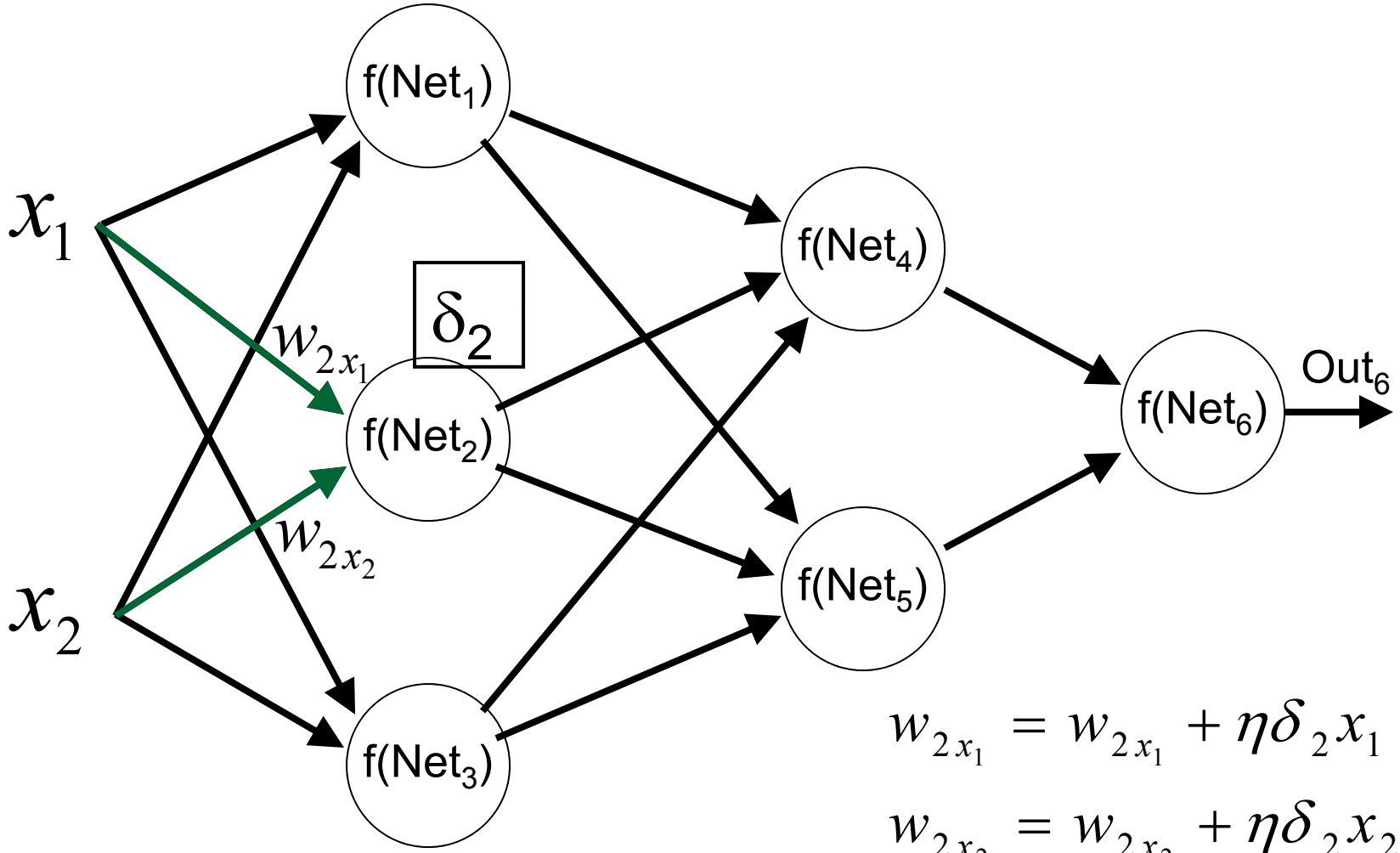
Giải thuật BP – Cập nhật trọng số (1)



$$w_{1x_1} = w_{1x_1} + \eta\delta_1 x_1$$

$$w_{1x_2} = w_{1x_2} + \eta\delta_1 x_2$$

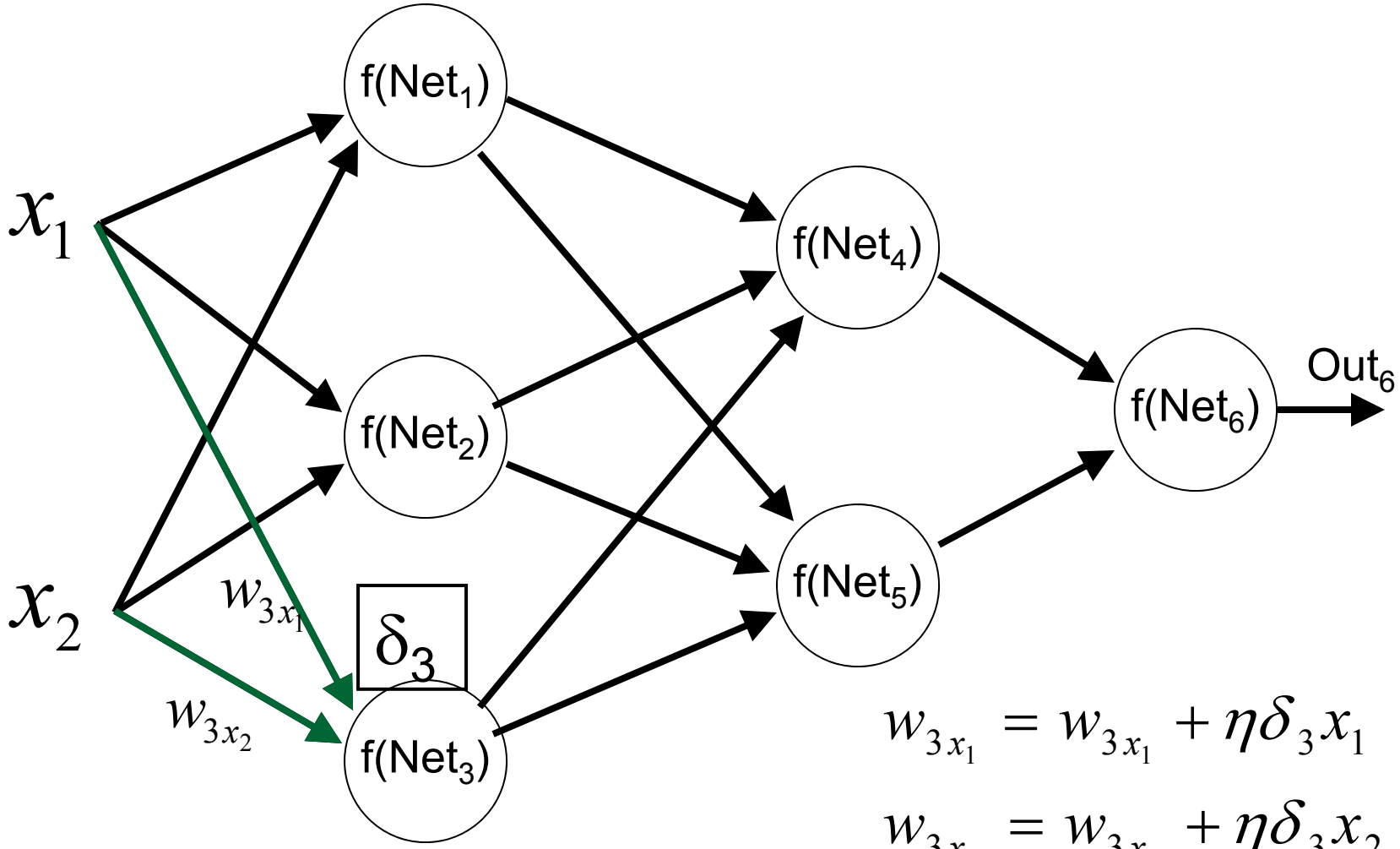
Giải thuật BP – Cập nhật trọng số (2)



$$w_{2x_1} = w_{2x_1} + \eta\delta_2 x_1$$

$$w_{2x_2} = w_{2x_2} + \eta\delta_2 x_2$$

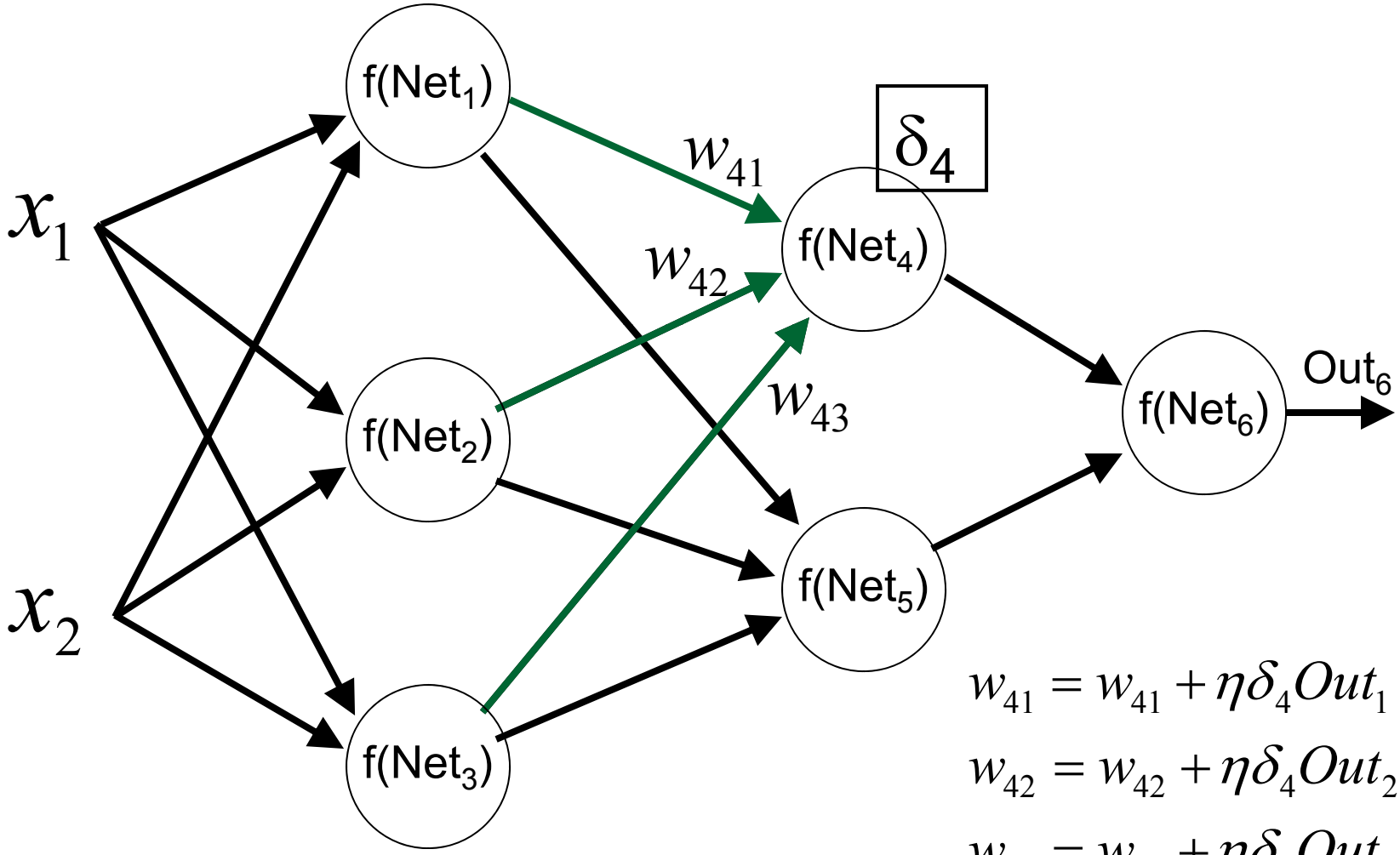
Giải thuật BP – Cập nhật trọng số (3)



$$w_{3x_1} = w_{3x_1} + \eta\delta_3 x_1$$

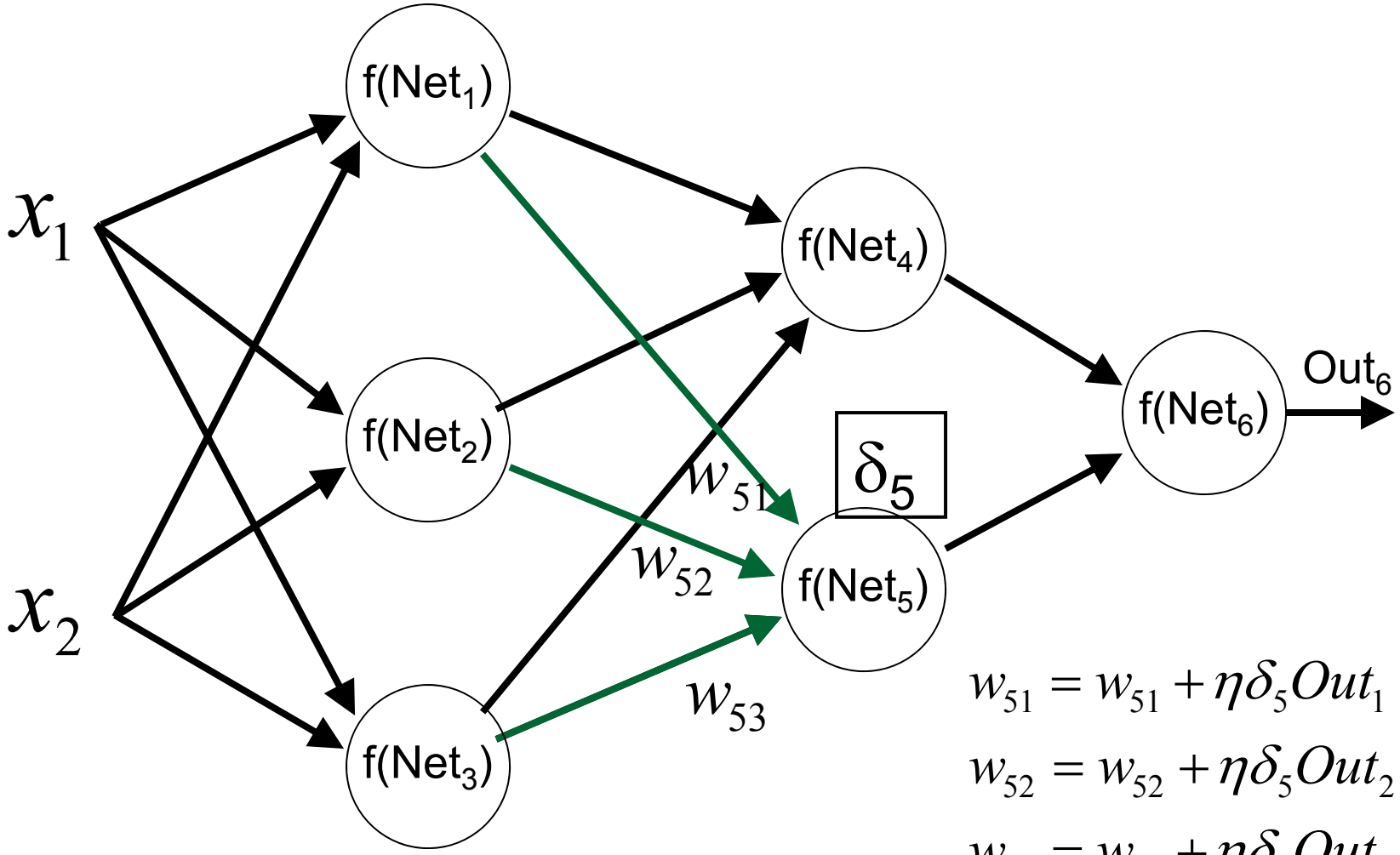
$$w_{3x_2} = w_{3x_2} + \eta\delta_3 x_2$$

Giải thuật BP – Cập nhật trọng số (4)



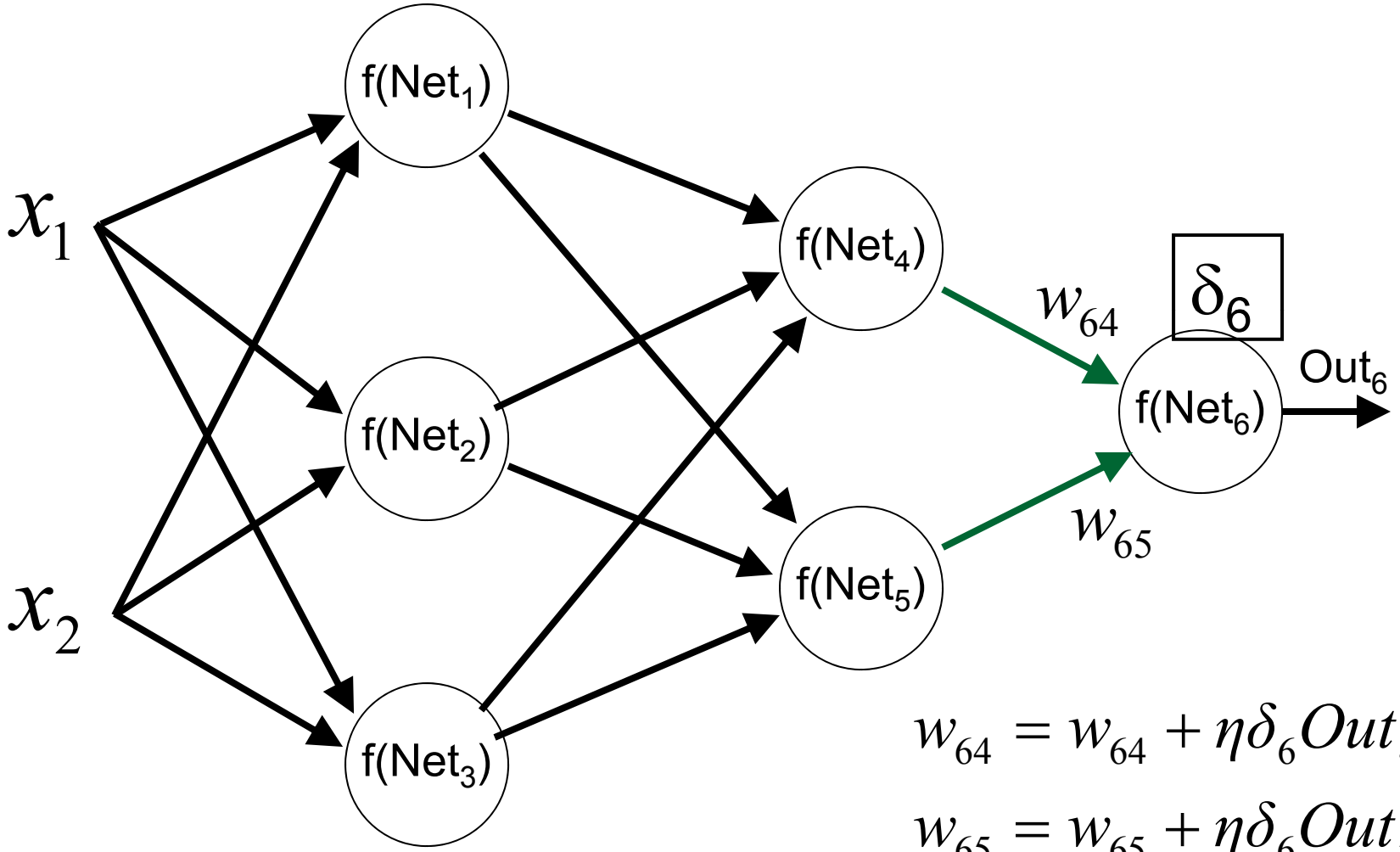
$$w_{41} = w_{41} + \eta\delta_4 Out_1$$
$$w_{42} = w_{42} + \eta\delta_4 Out_2$$
$$w_{43} = w_{43} + \eta\delta_4 Out_3$$

Giải thuật BP – Cập nhật trọng số (5)



$$w_{51} = w_{51} + \eta\delta_5 Out_1$$
$$w_{52} = w_{52} + \eta\delta_5 Out_2$$
$$w_{53} = w_{53} + \eta\delta_5 Out_3$$

Giải thuật BP – Cập nhật trọng số (6)



$$w_{64} = w_{64} + \eta\delta_6 Out_4$$

$$w_{65} = w_{65} + \eta\delta_6 Out_5$$

BP: Khởi tạo giá trị của các trọng số

- Thông thường, các trọng số được khởi tạo với các giá trị nhỏ ngẫu nhiên
- Nếu các trọng số có các giá trị ban đầu lớn
  - Các hàm xích-ma (sigmoid functions) sẽ đạt trạng thái bão hòa sớm
  - Hệ thống sẽ tắc ở một điểm cực tiểu cục bộ (local minimum) hoặc ở một trạng thái không đổi (very flat plateau) gần điểm bắt đầu
- Các gợi ý cho w^0_{ab} (liên kết từ nơ-ron *b* tới nơ-ron *a*)
  - Ký hiệu n_a là số lượng các nơ-ron ở cùng tầng với nơ-ron *a*

$$w^0_{ab} \in [-1/n_a, 1/n_a]$$

  - Ký hiệu k_a là số lượng các nơ-ron có liên kết (tiến) đến nơ-ron *a* (=số lượng các liên kết đầu vào của nơ-ron *a*)

$$w^0_{ab} \in [-3/\sqrt{k_a}, 3/\sqrt{k_a}]$$

BP: Tốc độ học (Learning rate)

- Ảnh hưởng quan trọng đến hiệu quả và khả năng hội tụ của giải thuật học BP
  - Một giá trị η lớn có thể đẩy nhanh sự hội tụ của quá trình học, nhưng có thể làm cho hệ thống bỏ qua điểm tối ưu toàn cục hoặc rơi vào điểm tối ưu cục bộ
  - Một giá trị η nhỏ có thể làm cho quá trình học kéo dài rất lâu
- Thường được chọn theo thực nghiệm (experimentally) đối với mỗi bài toán
- Các giá trị tốt của tốc độ học ở lúc bắt đầu (quá trình học) có thể không tốt ở một thời điểm sau đấy
  - Nên sử dụng một tốc độ học thích nghi (động)
- Sau khi cập nhật các trọng số, kiểm tra xem việc cập nhật các trọng số có giúp làm giảm giá trị lỗi

$$\Delta\eta = \begin{cases} a & \text{, if } \Delta E < 0 \text{ consistently} \\ -b\eta & \text{, if } \Delta E > 0 \\ 0 & \text{, otherwise.} \end{cases} \qquad (a, b > 0)$$

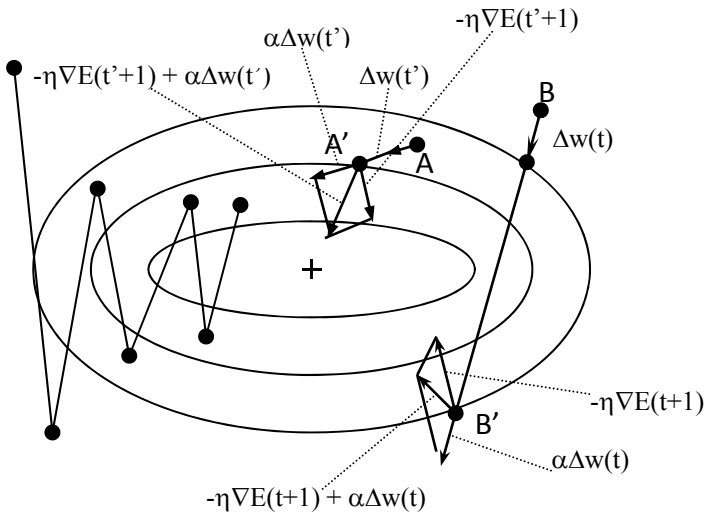
BP: Momentum

- Phương pháp *Gradient descent* có thể rất chậm nếu η nhỏ, và có thể dao động mạnh nếu η quá lớn
- Để giảm mức độ dao động, cần đưa vào một thành phần momentum

$$\Delta w^{(t)} = -\eta \nabla E^{(t)} + \alpha \Delta w^{(t-1)}$$

trong đó α ($\in[0,1]$) là một tham số momentum (thường lấy =0.9)

- Một quy tắc, dựa trên các thí nghiệm, để chọn các giá trị hợp lý cho tốc độ học và momentum là: $(\eta+\alpha) \gtrapprox 1$; trong đó $\alpha > \eta$ để tránh dao động



Gradient descent đối với một hàm lỗi bậc 2 đơn giản.

Quỹ đạo bên trái không sử dụng momentum.

Quỹ đạo bên phải có sử dụng momentum.

BP: Số lượng các nơ-ron ở tầng ẩn

- Kích thước (số nơ-ron) của tầng ẩn là một câu hỏi quan trọng đối với việc áp dụng các mạng nơ-ron lan truyền tiến nhiều tầng để giải quyết các bài toán thực tế
- Trong thực tế, rất khó để xác định chính xác số lượng các nơ-ron cần thiết để đạt được một độ chính xác mong muốn của hệ thống
- Kích thước của tầng ẩn thường được xác định qua thí nghiệm (experimentally/trial and test)
- Gợi ý
  - Bắt đầu với số lượng nhỏ các nơ-ron ở tầng ẩn (=tỷ lệ nhỏ so với số lượng các tín hiệu đầu vào)
  - Nếu mạng không thể hội tụ, bổ sung thêm các nơ-ron vào tầng ẩn
  - Nếu mạng hội tụ, có thể xem xét sử dụng ít hơn các nơ-ron tầng ẩn

ANNs – Giới hạn học

- Các hàm nhị phân (Boolean functions)
  - Bất kỳ hàm nhị phân nào cũng có thể học được bởi một ANN sử dụng 1 tầng ẩn
- Các hàm liên tục (Continuous functions)
  - Bất kỳ một hàm liên tục bị giới hạn (bounded continuous function) nào cũng có thể học được (xấp xỉ) bởi một ANN sử dụng 1 tầng ẩn [Cybenko, 1989; Hornik et al., 1989]
  - Bất kỳ một hàm mục tiêu nào cũng có thể học được (xấp xỉ) bởi một ANN sử dụng 2 tầng ẩn [Cybenko, 1988]

ANNs – Ưu điểm, Nhược điểm

- Các ưu điểm
  - Bản chất (về cấu trúc) hỗ trợ tính toán song song ở mức (rất) cao
  - Khả năng chịu nhiễu/lỗi, nhờ kiến trúc tính toán song song
  - Có thể được thiết kế để tự thích nghi (các trọng số, cấu trúc mạng)
- Các nhược điểm
  - Không có quy tắc tổng quát để xác định cấu trúc mạng và các tham số học tối ưu cho một (lớp) bài toán nhất định
  - Không có phương pháp tổng quát để đánh giá hoạt động bên trong của ANN (vì vậy, hệ thống ANN bị xem như một "hộp đen")
  - Rất khó (không thể) đưa ra giải thích cho người dùng
  - Rất khó để dự đoán hiệu năng của hệ thống trong tương lai (khả năng khái quát hóa của hệ thống học)

ANNs – Khi nào?

- Mỗi ví dụ được biểu diễn bởi một tập gồm (rất) nhiều thuộc tính kiểu rời rạc hoặc kiểu số
- Miền giá trị đầu ra của hàm mục tiêu có kiểu số thực, hoặc kiểu rời rạc, hoặc kiểu vectơ
- Tập dữ liệu có thể chứa nhiễu/lỗi
- Dạng của hàm mục tiêu không xác định (biết) trước
- Không cần thiết (hoặc không quan trọng) phải đưa ra giải thích cho người dùng đối với các kết quả
- Chấp nhận thời gian (khá) lâu cho quá trình huấn luyện
- Yêu cầu thời gian (khá) nhanh cho việc phân loại/dự đoán