

Nội dung môn học

- Lecture 1: Giới thiệu về Học máy và khai phá dữ liệu
- Lecture 2: Thu thập và tiền xử lý dữ liệu
- Lecture 3: Hồi quy tuyến tính (Linear regression)
- Lecture 4+5: Phân cụm
- Lecture 6: Phân loại và Đánh giá hiệu năng
- Lecture 7: dựa trên láng giềng gần nhất (KNN)
- Lecture 8: Cây quyết định và Rừng ngẫu nhiên
- Lecture 9: Học dựa trên xác suất
- **Lecture 10: Mạng nơron (Neural networks)**
- Lecture 11: Máy vector hỗ trợ (SVM)
- Lecture 12: Khai phá tập mục thường xuyên và các luật kết hợp
- Lecture 13: Thảo luận ứng dụng học máy và khai phá dữ liệu trong thực tế

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
- Khả năng của một ANN phụ thuộc vào
 - Kiến trúc (topology) của mạng nơ-ron
 - Đặc tính đầu vào/ra của mỗi nơ-ron
 - Thuật toán học (huấn luyện)
 - Dữ liệu học

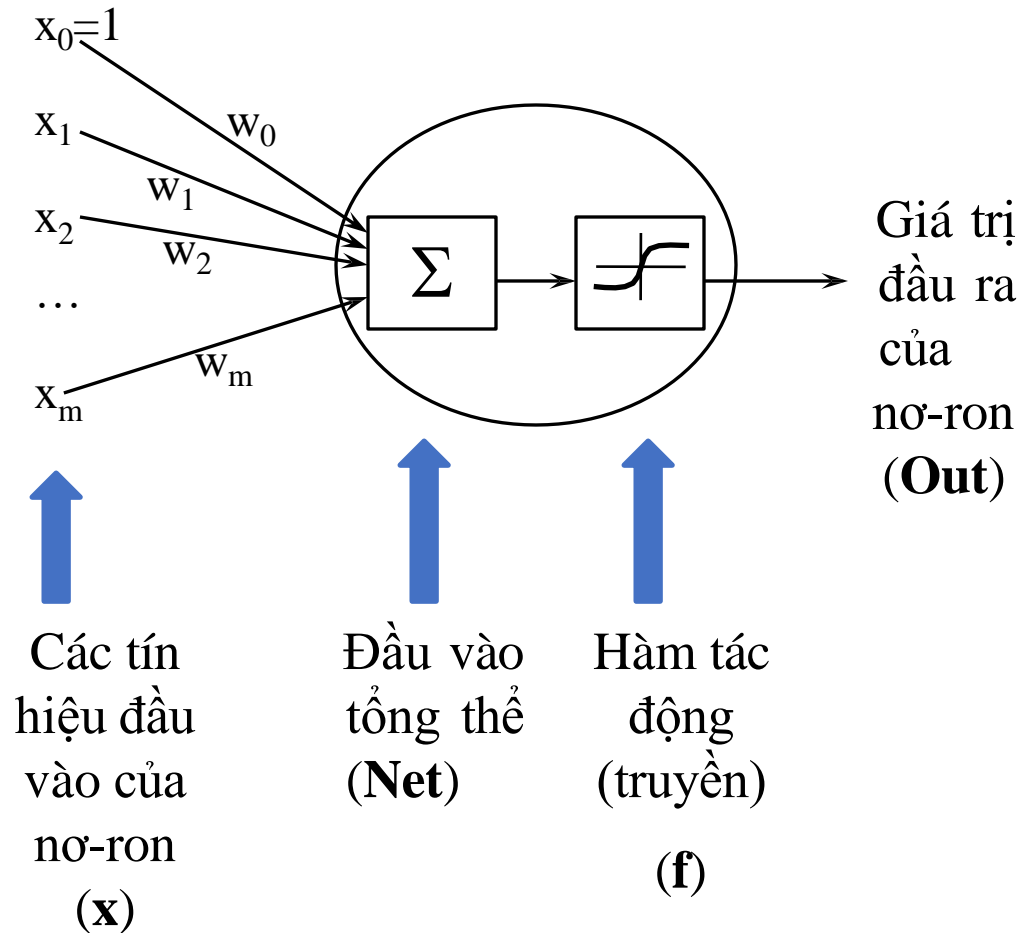
ANN: a huge breakthrough

- AlphaGo of Google the world champion at Go (cờ vây), 3/2016
 - Go is a 2500 year-old game.
 - Go is one of the most complex games.
- AlphaGo learns from 30 millions human moves, and plays itself to find new moves.
- It beat Lee Sedol (World champion)
 - <http://www.wired.com/2016/03/two-i-redefined-future/>
 - <http://www.nature.com/news/google-game-of-go-1.19234>



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 (θ -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 (\mathbf{w}, \mathbf{x})
- Hàm tác động/truyền (Activation/transfer function) tính giá trị đầu ra của nơ-ron – $f(\text{Net}(\mathbf{w}, \mathbf{x}))$
- Giá trị đầu ra (Output) của nơ-ron: $\text{Out} = f(\text{Net}(\mathbf{w}, \mathbf{x}))$

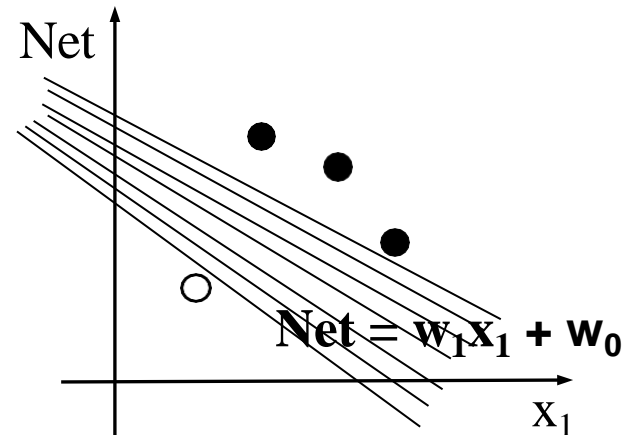
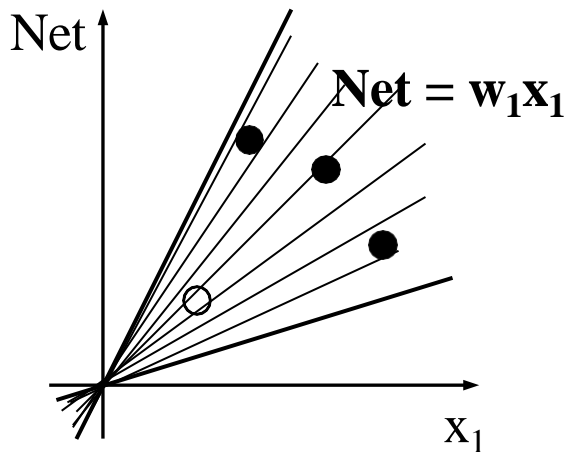


Đầu vào tổng thể

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

$$Net = w_0 + w_1x_1 + w_2x_2 + \dots + w_mx_m = w_0 \cdot 1 + \sum_{i=1}^m w_ix_i = \sum_{i=0}^m w_ix_i$$

- Ý nghĩa của trọng số điều chỉnh (bias) w_0
 - Họ các hàm $Net = w_1x_1$ *không thể phân tách được* các ví dụ thành 2 lớp (two classes)
 - Nhưng: họ các hàm $Net = w_1x_1 + w_0$ *có thể!*



Các hàm kích hoạt

Tạo ra tính phi tuyến giữa đầu vào và đầu ra của một nơ-ron. Nhờ đó, mô hình có thể học được các mối quan hệ phức tạp giữa dữ liệu đầu vào và nhãn đầu ra.

Hàm tác động: Giới hạn cứng

Còn được gọi là hàm ngưỡng

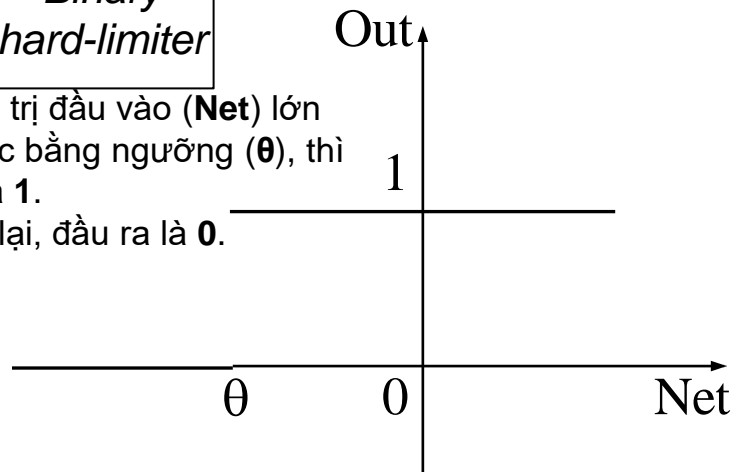
θ là giá trị ngưỡng (threshold function) vì nó đưa ra đầu ra dựa trên việc **so sánh với một ngưỡng θ** . Dùng trong logic rời rạc hoặc **Perceptron nguyên thủy**

$$Out(Net) = HL(Net, \theta) = \begin{cases} 1, & \text{if } Net \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

- Giá trị đầu ra lấy một trong 2 giá trị
- Nhược điểm:** không liên tục, không có đạo hàm

*Binary
hard-limiter*

- Nếu giá trị đầu vào (**Net**) lớn hơn hoặc bằng ngưỡng (**θ**), thì đầu ra là **1**.
- Ngược lại, đầu ra là **0**.



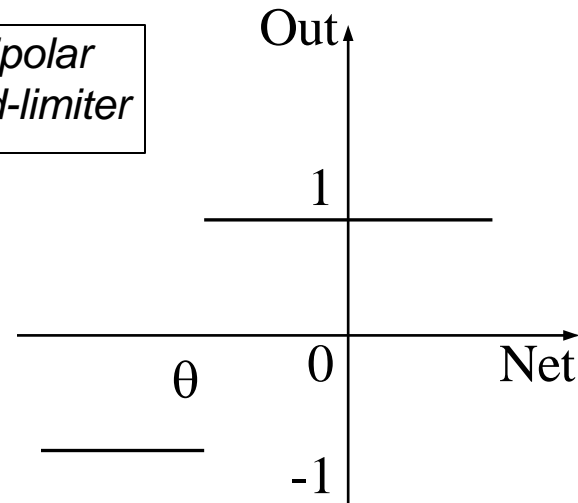
- Gồm 2 mức: 0 và 1.
- Tại điểm θ , đầu ra nhảy từ 0 lên 1

$$Out(Net) = HL2(Net, \theta) = sign(Net, \theta)$$

Ý nghĩa:

- Nếu **Net $\geq \theta$** , đầu ra là **1**.
- Nếu **Net $< \theta$** , đầu ra là **-1**.

*Bipolar
hard-limiter*



- Gồm 2 mức: -1 và 1.
- Đổi dấu tại ngưỡng θ .

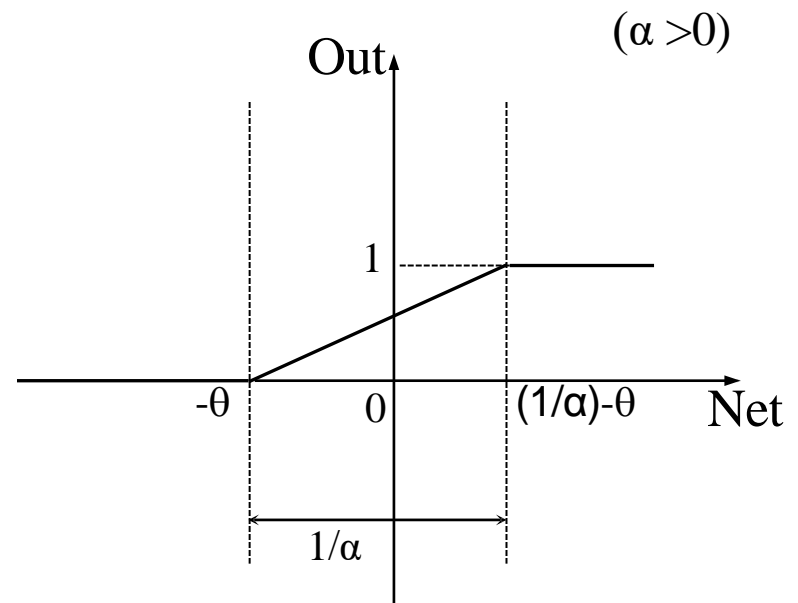
Hàm tác động: Logic ngưỡng

Dùng trong mô hình cổ điển như perceptron đơn lớp.

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

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

- 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 không có đạo hàm

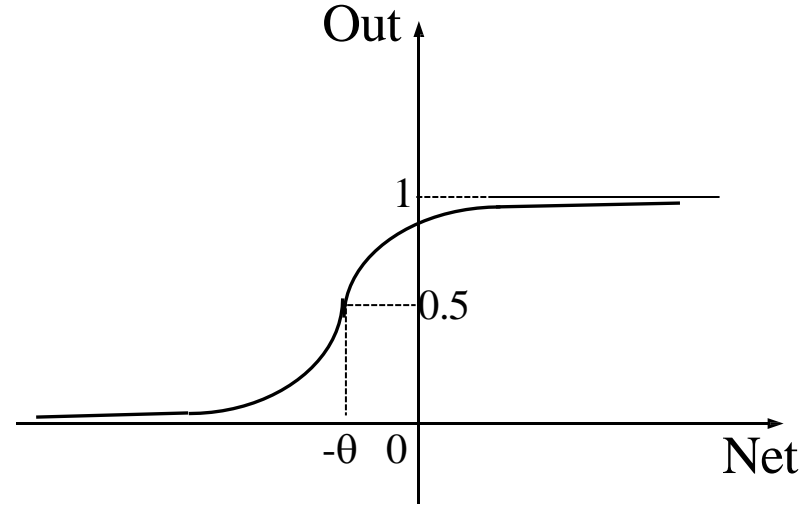


Hàm tác động: Sigmoid

Dùng nhiều trong các mạng nơ-ron hồi tiếp (RNN), hoặc lớp ẩn.

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

- Được dùng phổ biến
- 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 sigmoid được biểu diễn bằng một hàm của chính nó

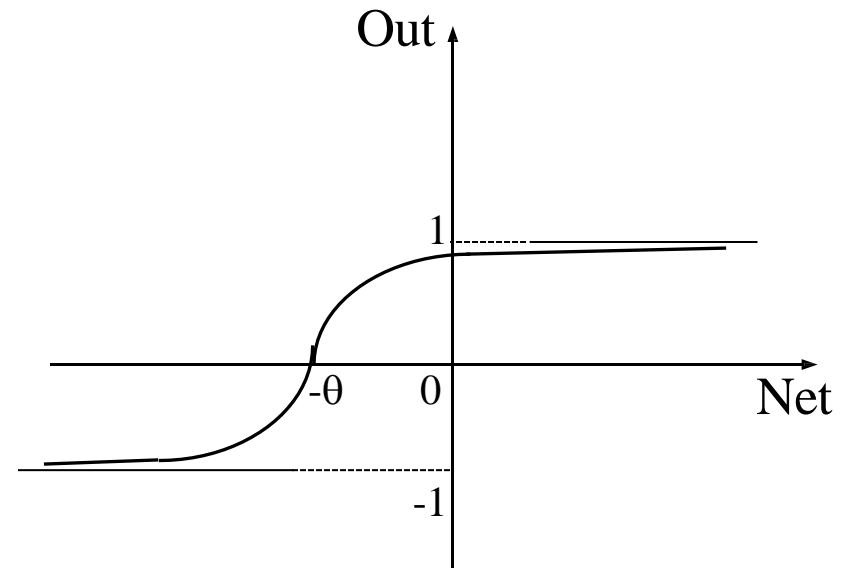


Hàm tác động: Hyperbolic tangent

Là hàm kích hoạt phổ biến nhất trong các mạng sâu (CNN, DNN, v.v.).

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

- 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

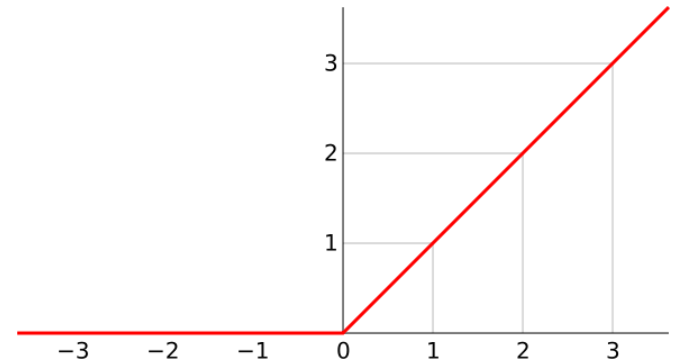


Hàm tác động: rectified linear unit (ReLU)

Được dùng nhiều nhất hiện nay trong các mạng sâu như CNN, DNN.

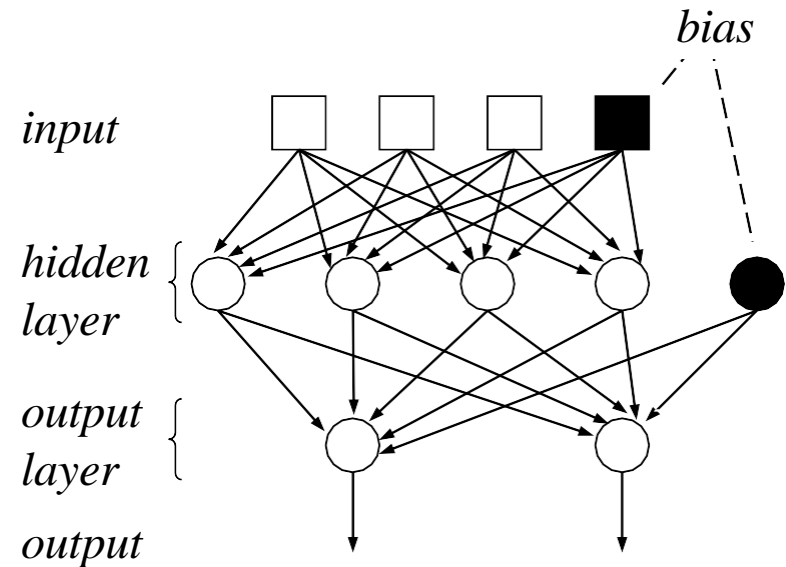
$$Out(net) = \max(0, net)$$

- Được sử dụng nhiều nhất hiện nay
- Giá trị đầu ra luôn không âm
- Ưu điểm
 - Liên tục
 - Không có đạo hàm tại điểm 0 duy nhất.
 - Dễ tính toán



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 liên kết đối với mỗi nơ-ron
 - Cách thức các nơ-ron (trong một tầng, hoặc giữa các tầng) liên kết với nhau
- 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))

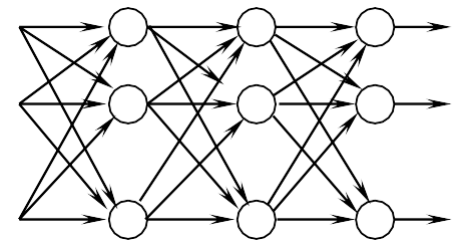


Ví dụ: Một ANN với một tầng ẩn

- 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

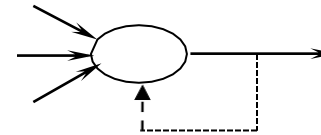
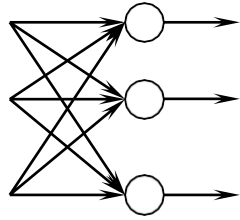


ANN: Kiến trúc mạng (3)

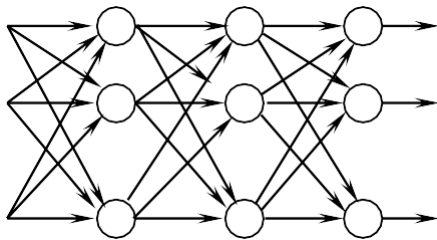
- Một ANN được gọi là ***mạng lan truyền tiến (feed-forward network)*** nếu không có bất kỳ đầu ra của một nút là đầu vào của một nút khác thuộc cùng tầng (hoặc thuộc một tầng phía trước)
- Khi các đầu ra của một nút liên kết ngược lại làm các đầu vào của một nút thuộc cùng tầng (hoặc thuộc một tầng 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)***

ANN: Kiến trúc mạng (4)

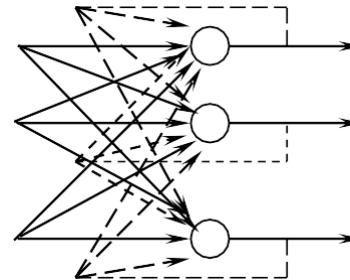
Mạng lan truyền tiến một tầng



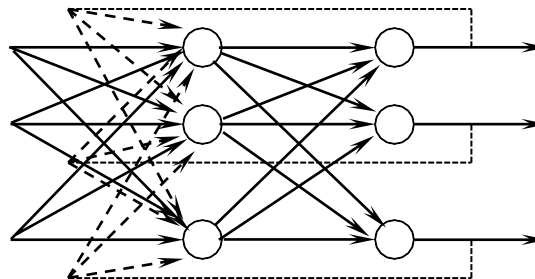
Một nơ-ron với phản hồi đến chính nó



Mạng lan truyền tiến nhiều tầng



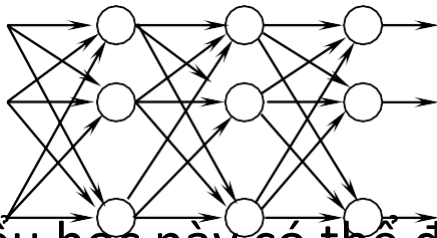
Mạng hồi quy một tầng



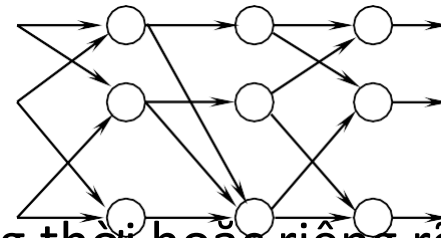
Mạng hồi quy nhiều tầng

ANN: cách huấn luyện

- 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



Or



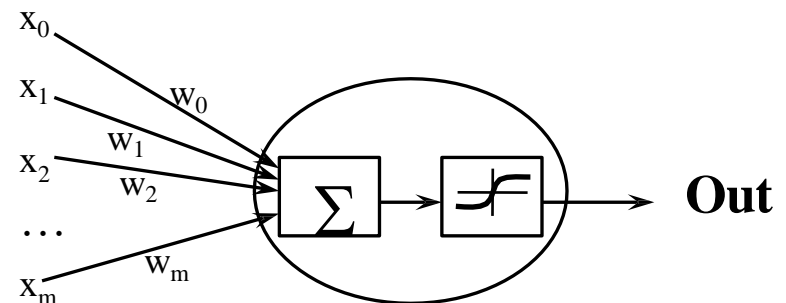
- 2 kiểu học này có thể được thực hiện đồng thời hoặc riêng rẽ
- Trong bài học này, chúng ta sẽ chỉ xét việc học tham số

ANN: ý tưởng

- Huấn luyện một mạng nơron (khi cố định kiến trúc) chính là việc học các trọng số \mathbf{w} của mạng từ tập học \mathbf{D} .
- Đưa việc học về bài toán cực tiểu hoá một hàm lỗi thực nghiệm:

$$L(\mathbf{w}) = \frac{1}{|\mathbf{D}|} \sum_{x \in \mathbf{D}} \text{loss}(d_x, \text{out}(x))$$

- Trong đó $\text{out}(x)$ là đầu ra của mạng, với đầu vào x có nhãn tương ứng là d_x ; loss là một hàm đo lỗi phán đoán.
- Nhiều phương pháp lặp dựa trên Gradient:
 - Backpropagation
 - SGD
 - Adam
 - AdaGrad

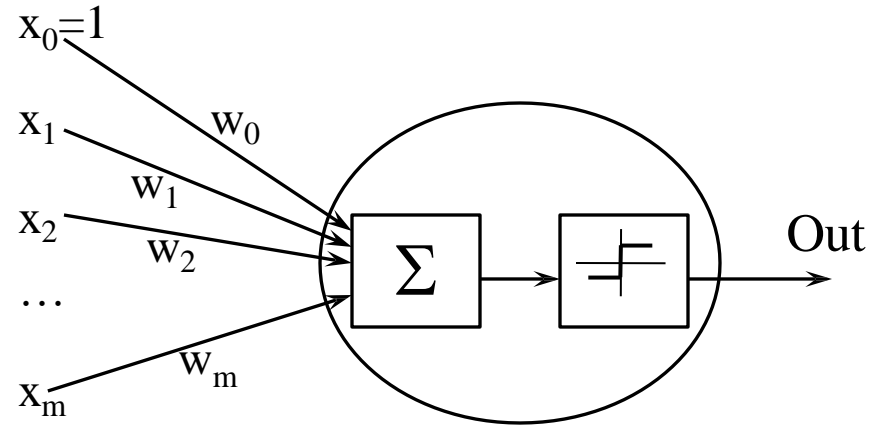


Perceptron

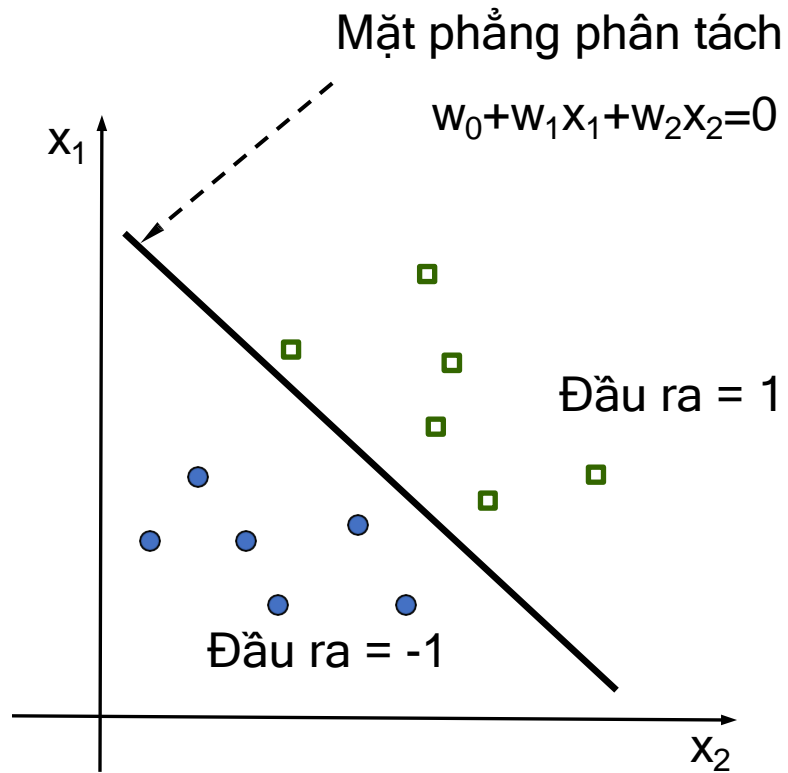
- Một perceptron là một kiểu đơn giản nhất của ANNs (chỉ gồm duy nhất một nơ-ron)
- Sử dụng hàm tác động giới hạn chặt

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

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



Perceptron: Minh họa



Perceptron: Giải thuật học

- Với một tập các ví dụ học $D = \{(\mathbf{x}, d)\}$
 - \mathbf{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 \mathbf{x} được perceptron phân lớp chính xác, thì vectơ trọng số \mathbf{w} không thay đổi
- Nếu $d=1$ nhưng perceptron lại sinh ra -1 ($\text{Out}=-1$), thì \mathbf{w} cần được thay đổi sao cho giá trị $\text{Net}(\mathbf{w}, \mathbf{x})$ tăng lên
- Nếu $d=-1$ nhưng perceptron lại sinh ra 1 ($\text{Out}=1$), thì \mathbf{w} cần được thay đổi sao cho giá trị $\text{Net}(\mathbf{w}, \mathbf{x})$ giảm đi

D : Tập huấn luyện, gồm các cặp (\mathbf{x}, d) – đầu vào \mathbf{x} và nhãn mong muốn d

Perceptron_batch(D, η) η : learning rate (tốc độ học)

Initialize \mathbf{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

 ($Out \neq d$)

$\Delta \mathbf{w} \leftarrow \Delta \mathbf{w} + \eta (d - Out) \mathbf{x}$ (*Trong đó* $Out = \text{sign}(\mathbf{w} \cdot \mathbf{x})$)

end for

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

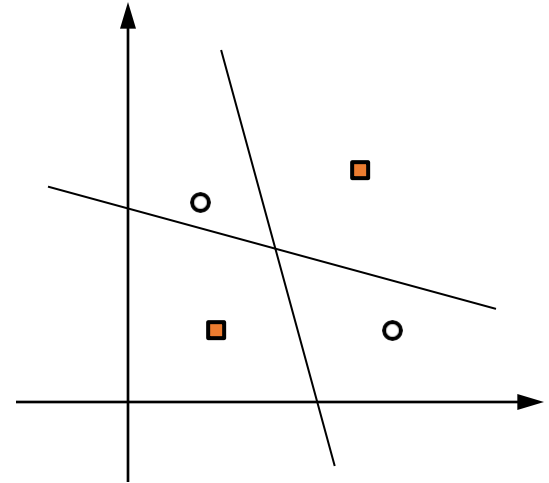
until all the training instances in D are correctly classified

return \mathbf{w}

Perceptron: Giới hạn (VD_L10:Excel)


- 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)

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



Perceptron cổ điển

Chỉ kiểm đúng/sai
Cập nhật theo lỗi nhị phân
Không cần đạo hàm
Đơn giản, tuyến tính



Hàm đánh giá lỗi (Loss function)

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

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

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

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

Tối thiểu hoá lỗi với Gradient

- **Gradient** của E (ký hiệu là ∇E) là một vectơ

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

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

- Gradient ∇E xác định hướng gây ra việc **tăng nhanh nhất (steepest increase)** đối với giá trị lỗi E
- Vì vậy, hướng gây ra việc **giảm nhanh nhất (steepest decrease)** là hướng ngược với gradient của E

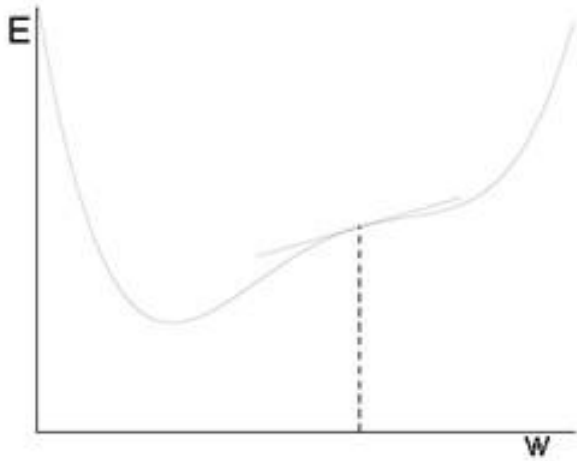
$$\Delta \mathbf{w} = -\eta \cdot \nabla E(\mathbf{w}); \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \quad \forall i = 1..N$$

- Yêu cầu: Các hàm tác động được sử dụng trong mạng phải có đạo hàm liên tục

Gradient descent: Minh họa

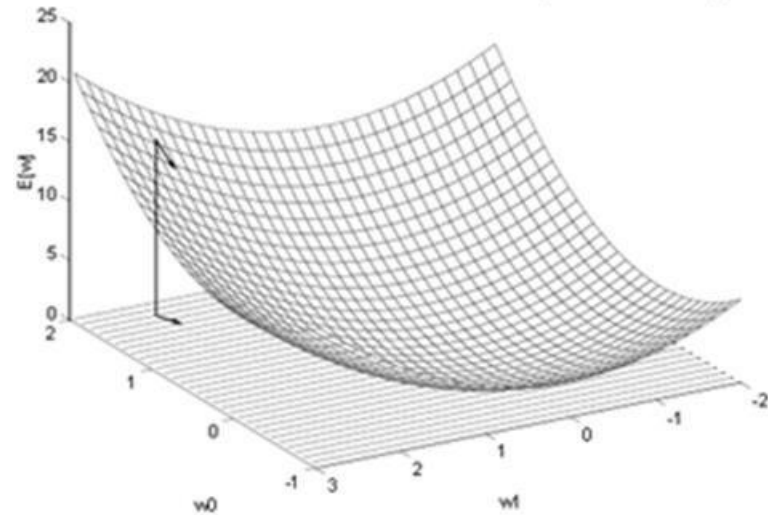
Không gian một chiều

$$E(w)$$



Không gian 2 chiều

$$E(w_1, w_2)$$



Gradient_descent_incremental (\mathbf{D}, η)

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

do

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

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}

If we take a small *subset* (mini-batch) *randomly* from \mathbf{D} to update the weights, we will have mini-batch training.

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

Thuật toán Perceptron_batch(\mathcal{D}, η) được suy ra từ Gradient vì:

- Hàm mất mát bình phương sai số (Mean Squared Error cho 1 mẫu):

$$E = \frac{1}{2}(d - y)^2$$

- Đầu ra của mô hình tuyến tính:

$$y = \sum w_i x_i + b = \mathbf{w} \cdot \mathbf{x} + b$$

$$\frac{\partial E}{\partial w_i} = -(d - y) \cdot x_i$$

1. Thuật toán Gradient Descent Incremental (SGD)

Tóm tắt lại quy trình:

- Khởi tạo các trọng số w_1 , w_2 , $bias$ (ngẫu nhiên nhỏ).
- Với mỗi mẫu (x, d) trong tập dữ liệu:
 - Tính đầu ra y
 - Tính sai số: $e = d - y$
 - Cập nhật trọng số:

$$w_i = w_i + \eta \cdot e \cdot x_i$$

$$bias = bias + \eta \cdot e$$

- Lặp lại cho đến khi đạt tiêu chí dừng (epoch hoặc lỗi nhỏ).

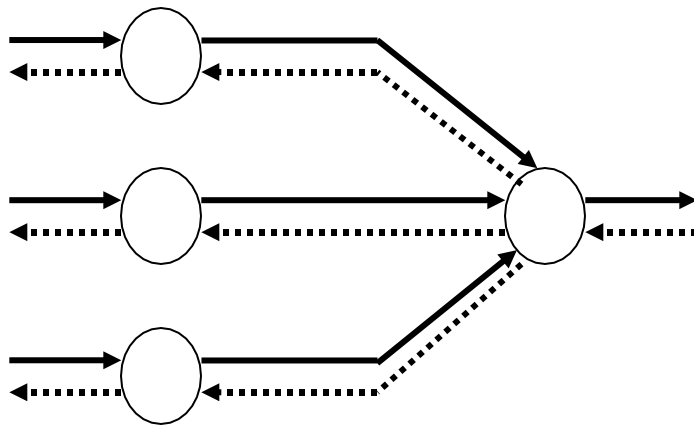
ANN nhiều tầng và giải thuật lan truyền ngược

- 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) 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

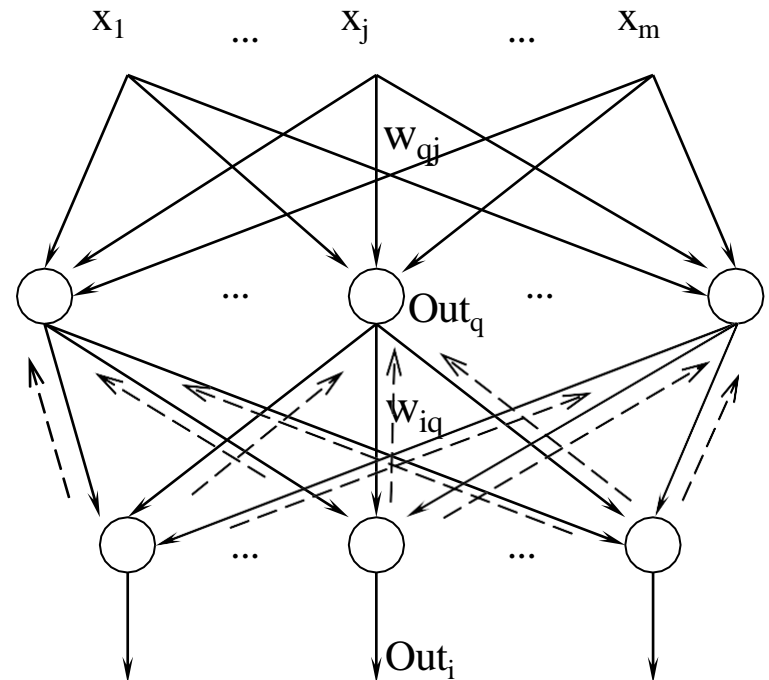
Giải thuật BP: Cấu trúc mạng

- 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 \mathbf{x}
 - Vectơ đầu vào \mathbf{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 \mathbf{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 \mathbf{x}

Giải thuật BP: Lan truyền ngược (1)

- Đối với mỗi ví dụ học x

- Các tín hiệu lỗi (error signals) do sự khác biệt 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 lỗi

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

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

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

(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

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

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

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

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

$$= \eta \sum_{i=1}^n [\delta w] f'(Net) x = \eta \delta_q x_j$$

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

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

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

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 còn 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}

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, $q = 1, 2, \dots, Q$

qNet_i và qOut_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

${}^qW_{ij}$ 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 ($q=1$)

$${}^qOut_i = {}^1Out_i = x_{i(k)}, \forall i$$

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) QOut_i

$${}^qOut_i = f({}^qNet_i) = f\left(\sum_j {}^qW_{ij} {}^{q-1}Out_j\right)$$

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

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

$${}^Q\delta_i = (d_i^{(k)} - {}^QOut_i) f'({}^QNet_i)$$

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

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

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

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)

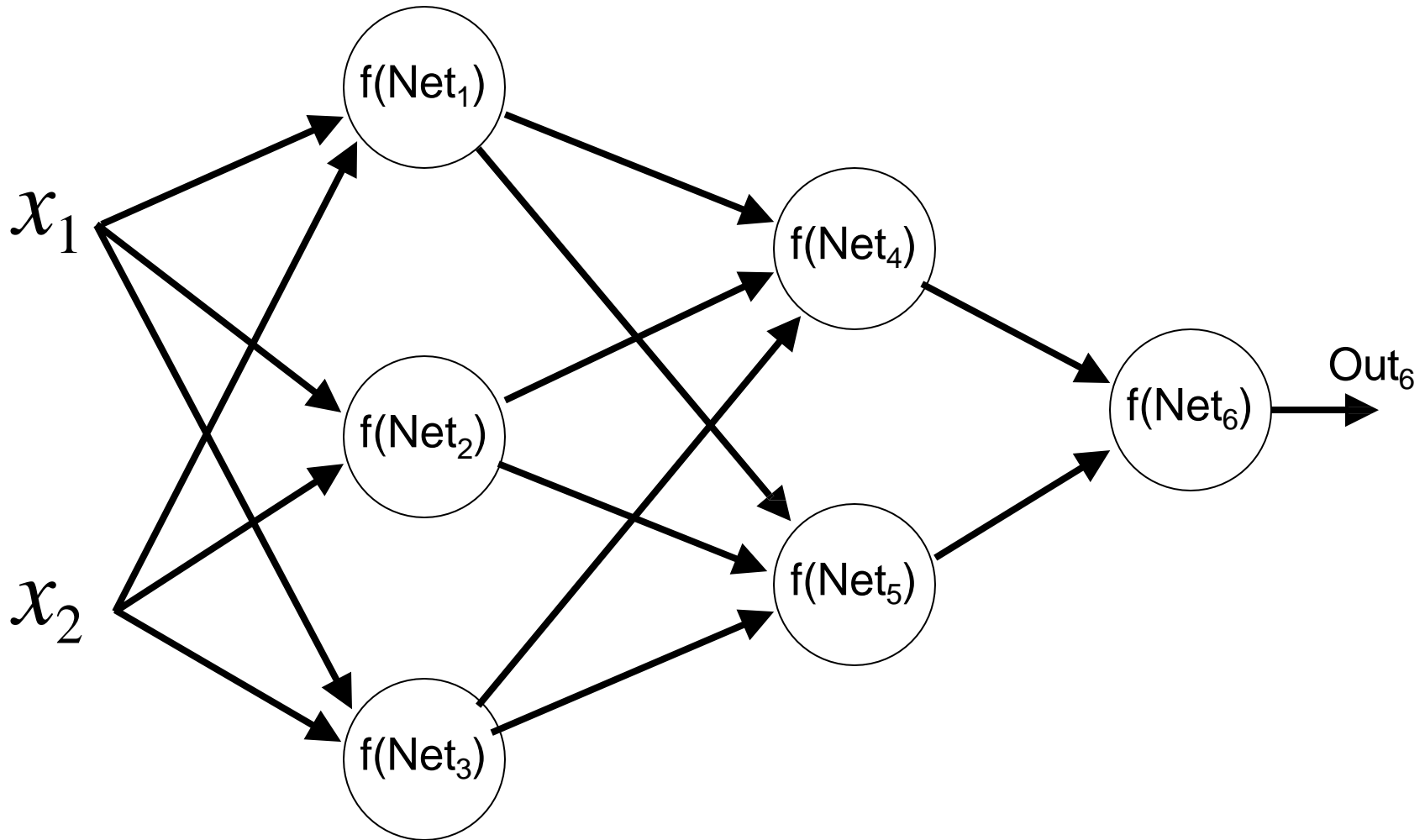
Nếu toàn bộ tập học đã được dùng, chuyển đến Bước 6; ngược lại, chuyển đến Bước 1

Bước 6 (Kiểm tra lỗi tổng thể)

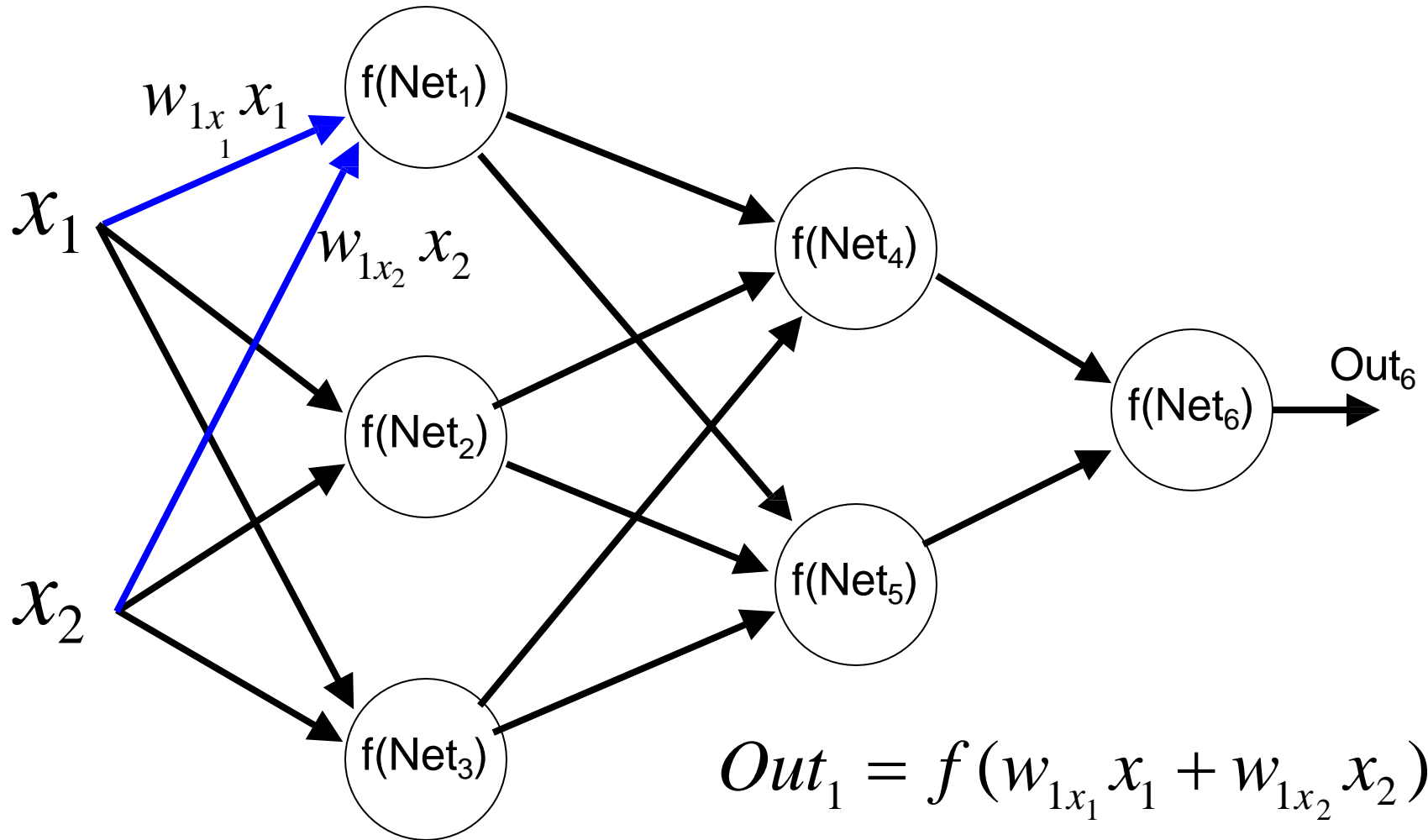
Nếu lỗi tổng thể E nhỏ hơn ngưỡng lỗi chấp nhận được ($< E_{\text{threshold}}$), thì quá trình học kết thúc và trả về các trọng số học được;

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

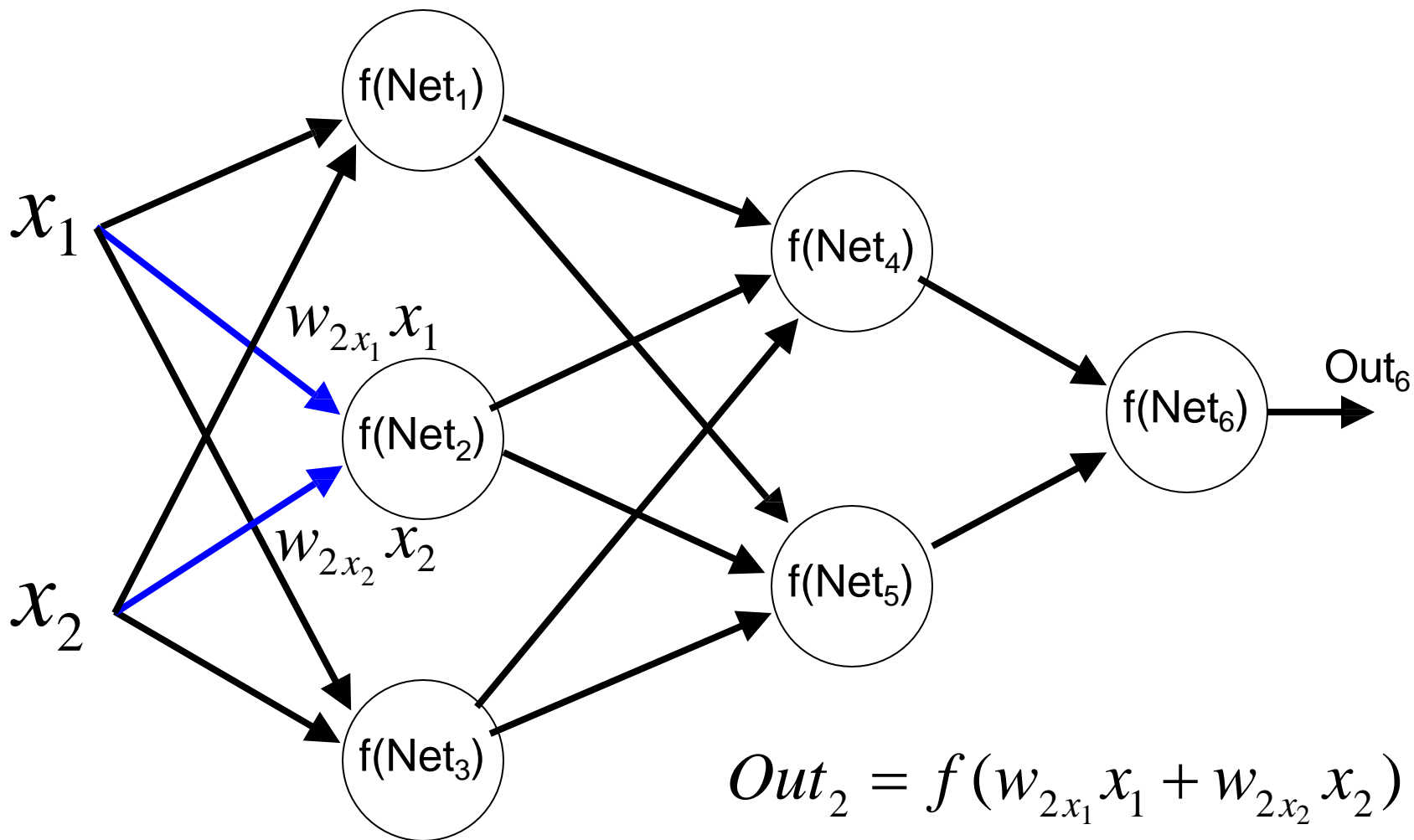
Giải thuật BP: Lan truyền tiến (1)



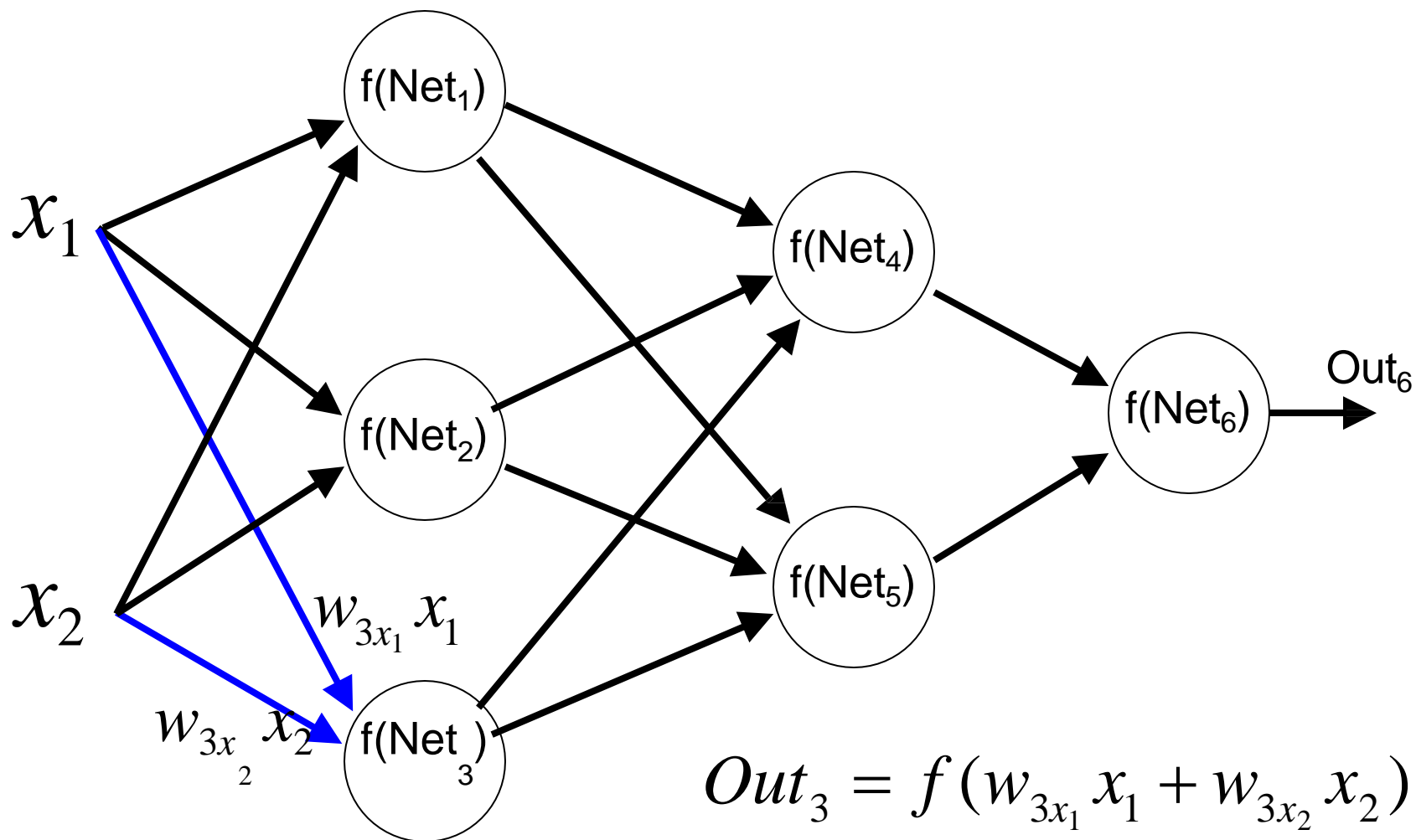
Giải thuật BP: Lan truyền tiến (2)



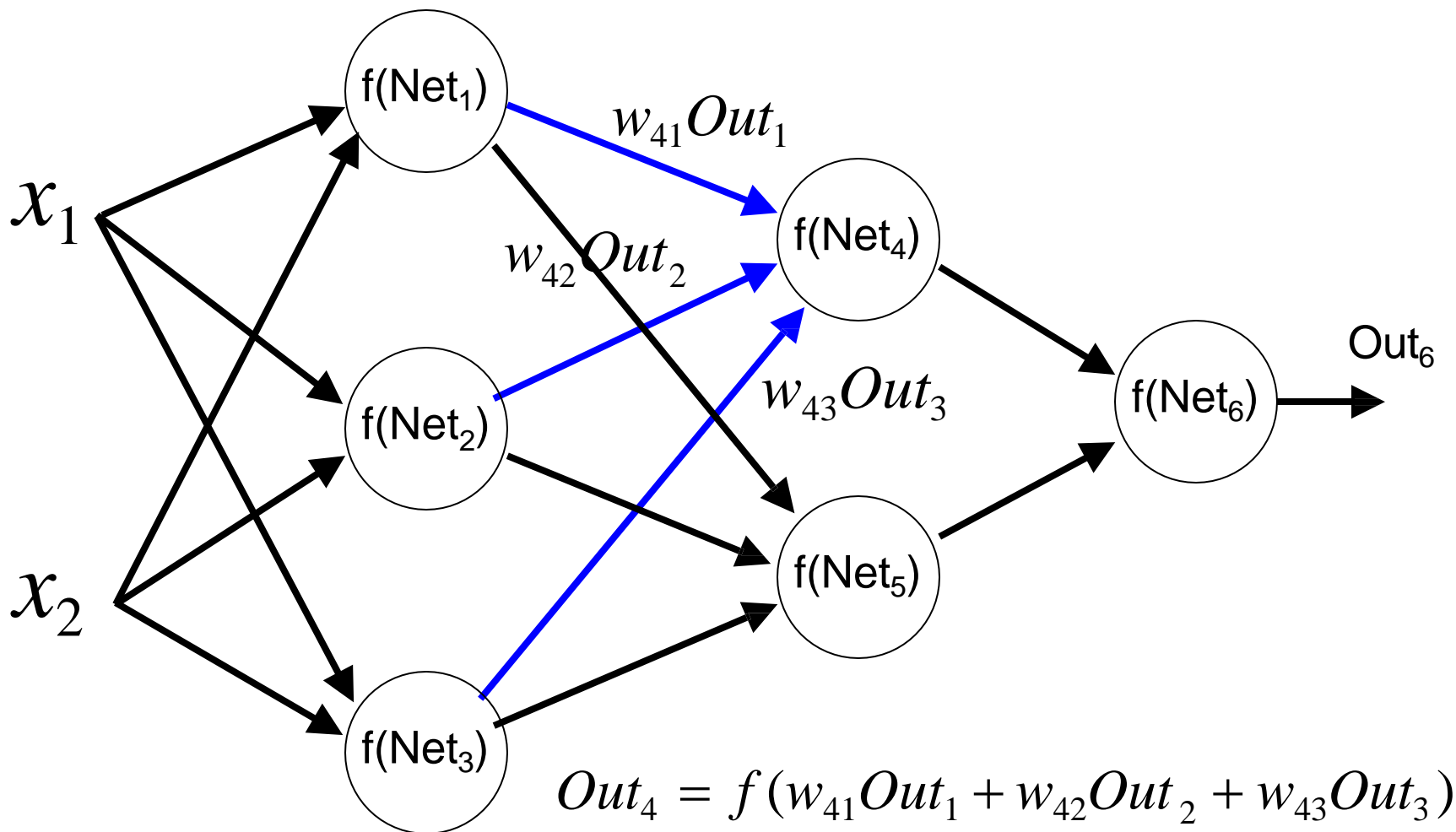
Giải thuật BP: Lan truyền tiến (3)



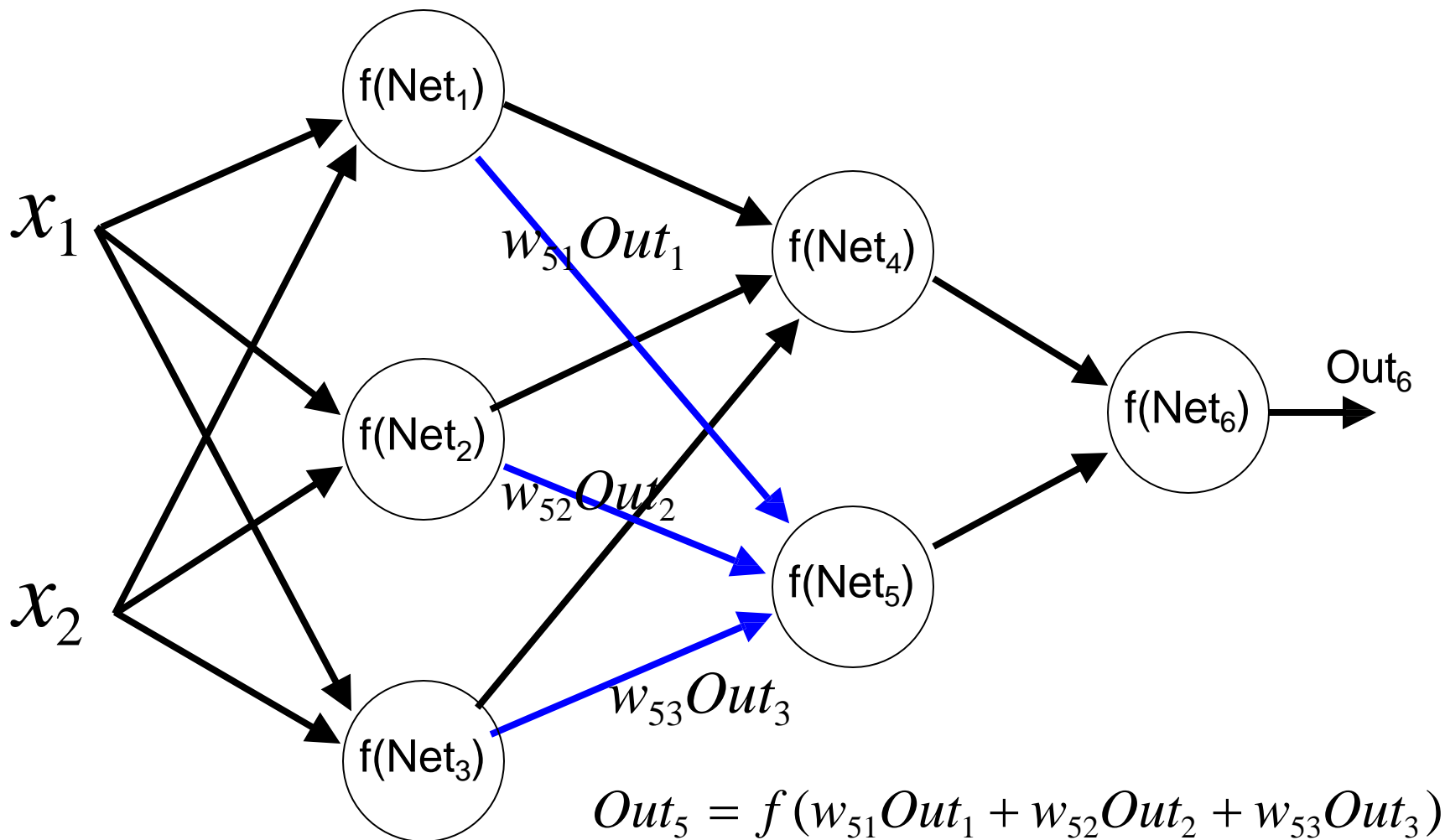
Giải thuật BP: Lan truyền tiến (4)



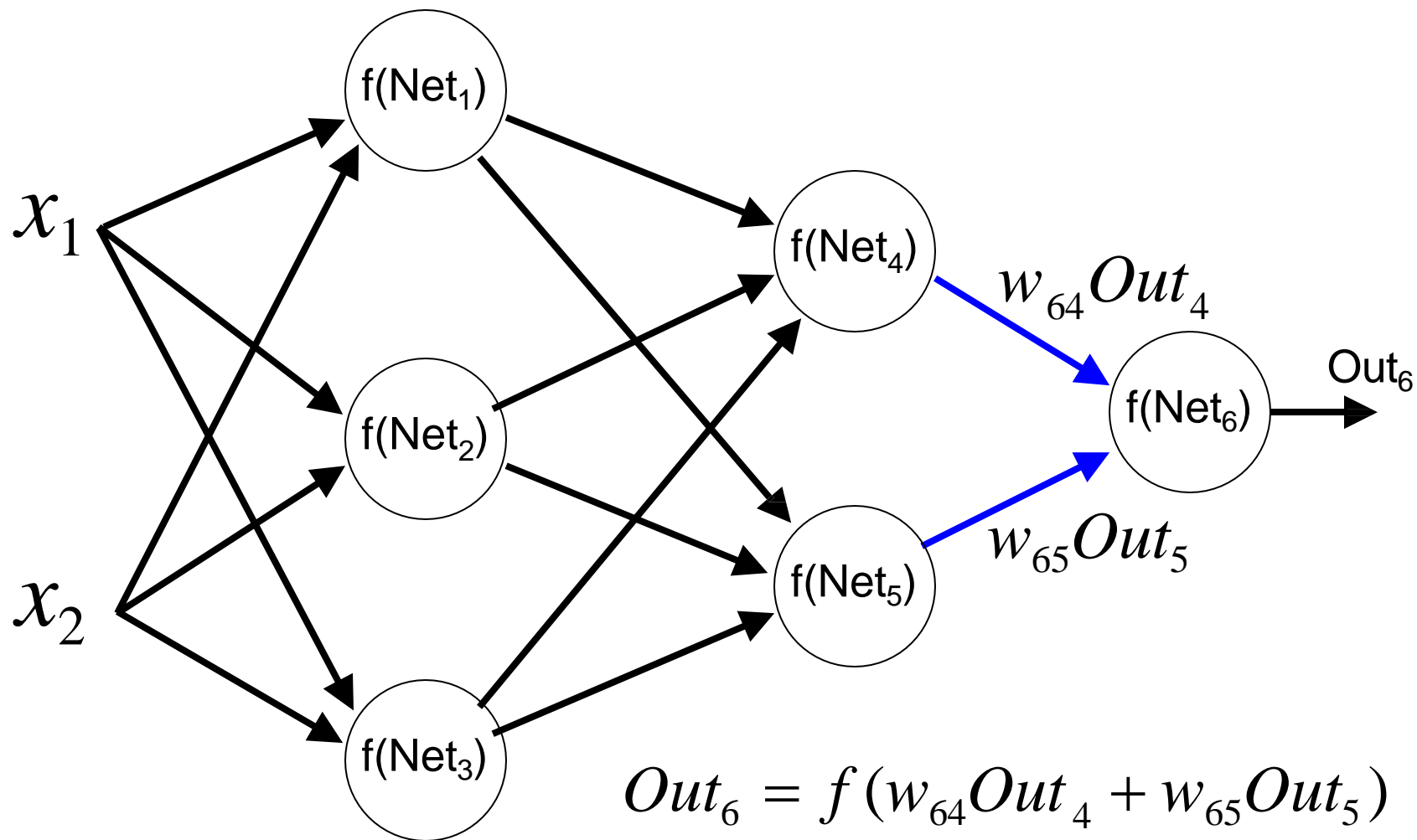
Giải thuật BP: Lan truyền tiến (5)



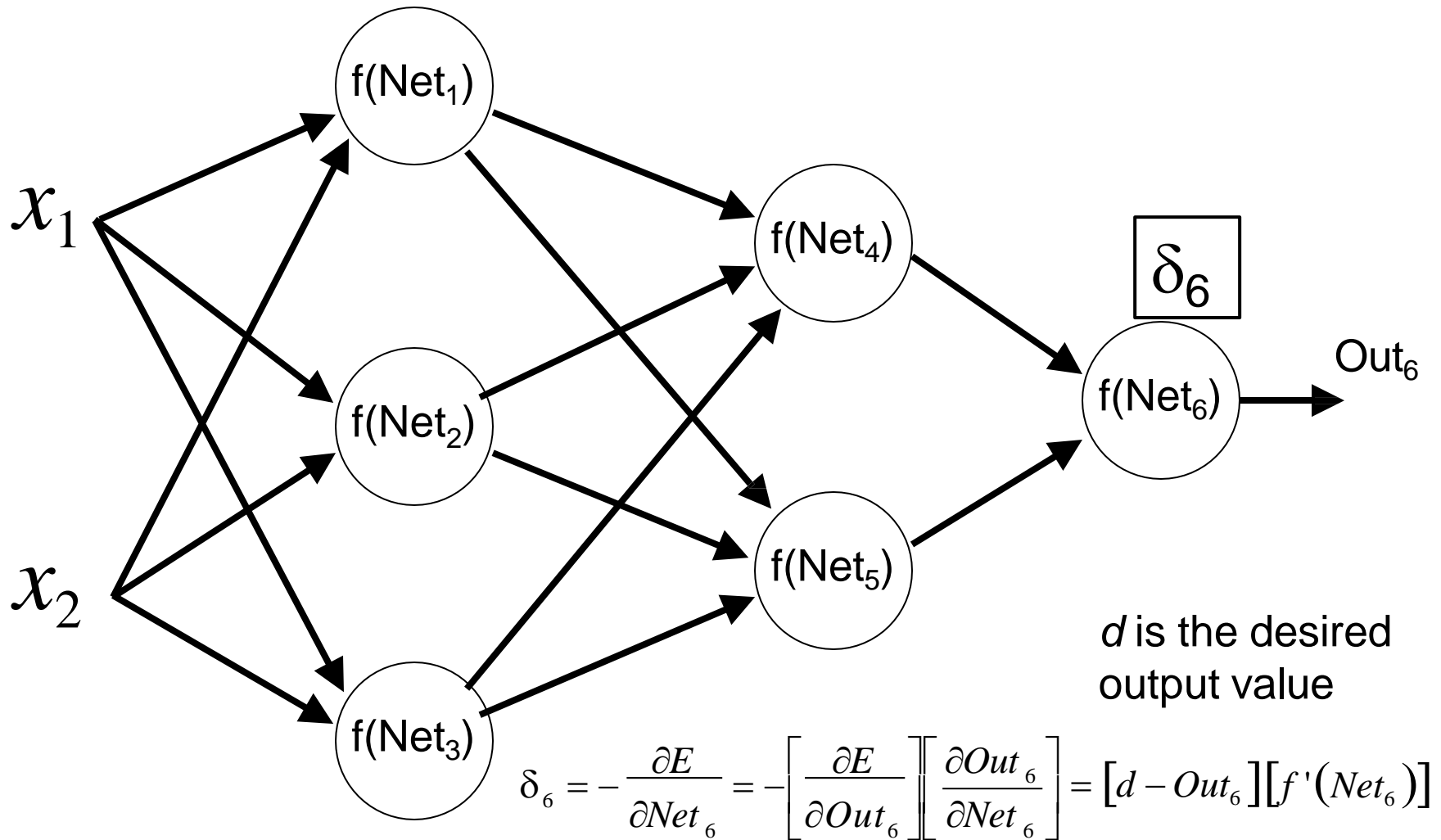
Giải thuật BP: Lan truyền tiến (6)



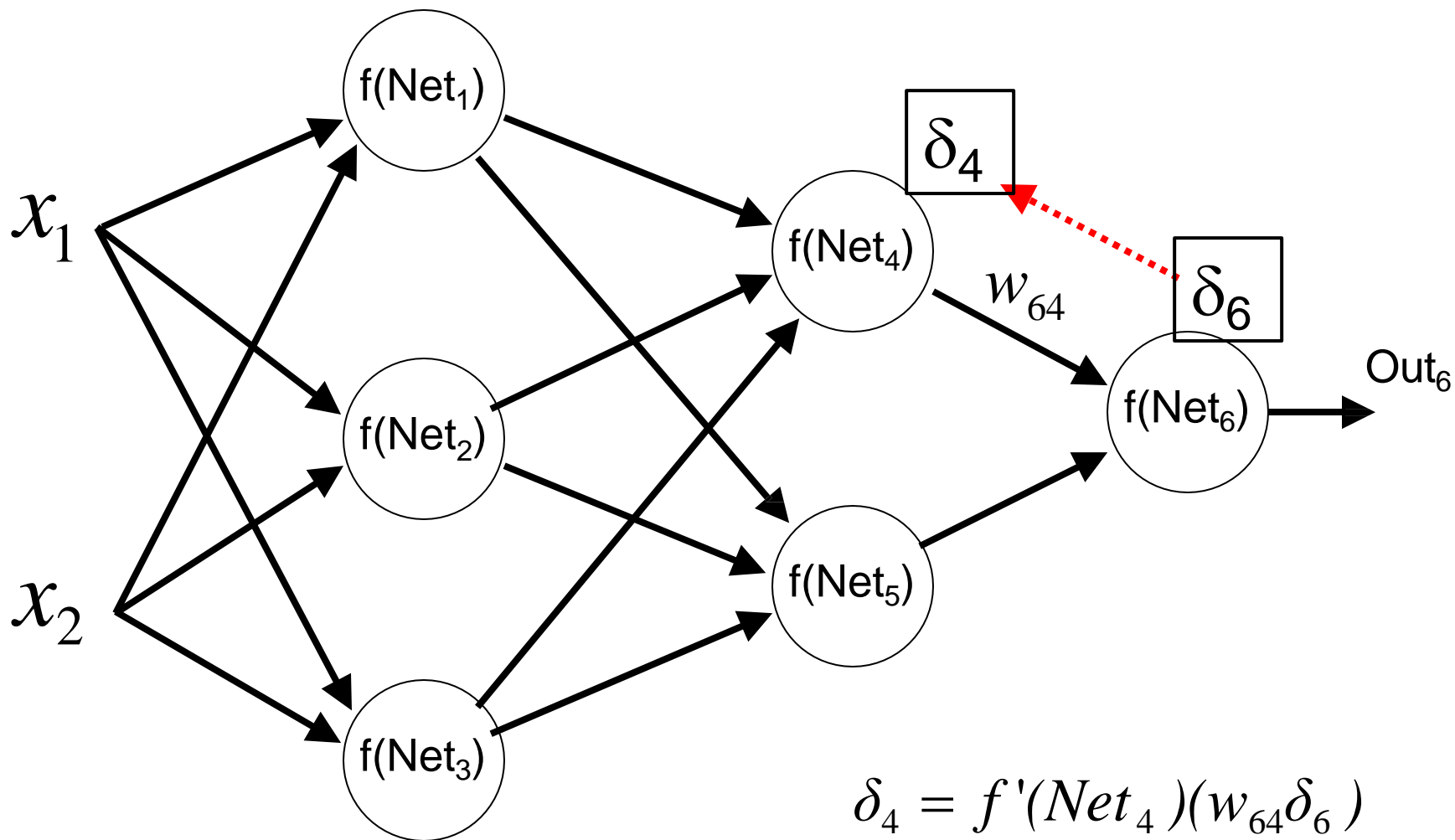
Giải thuật BP: Lan truyền tiến (7)



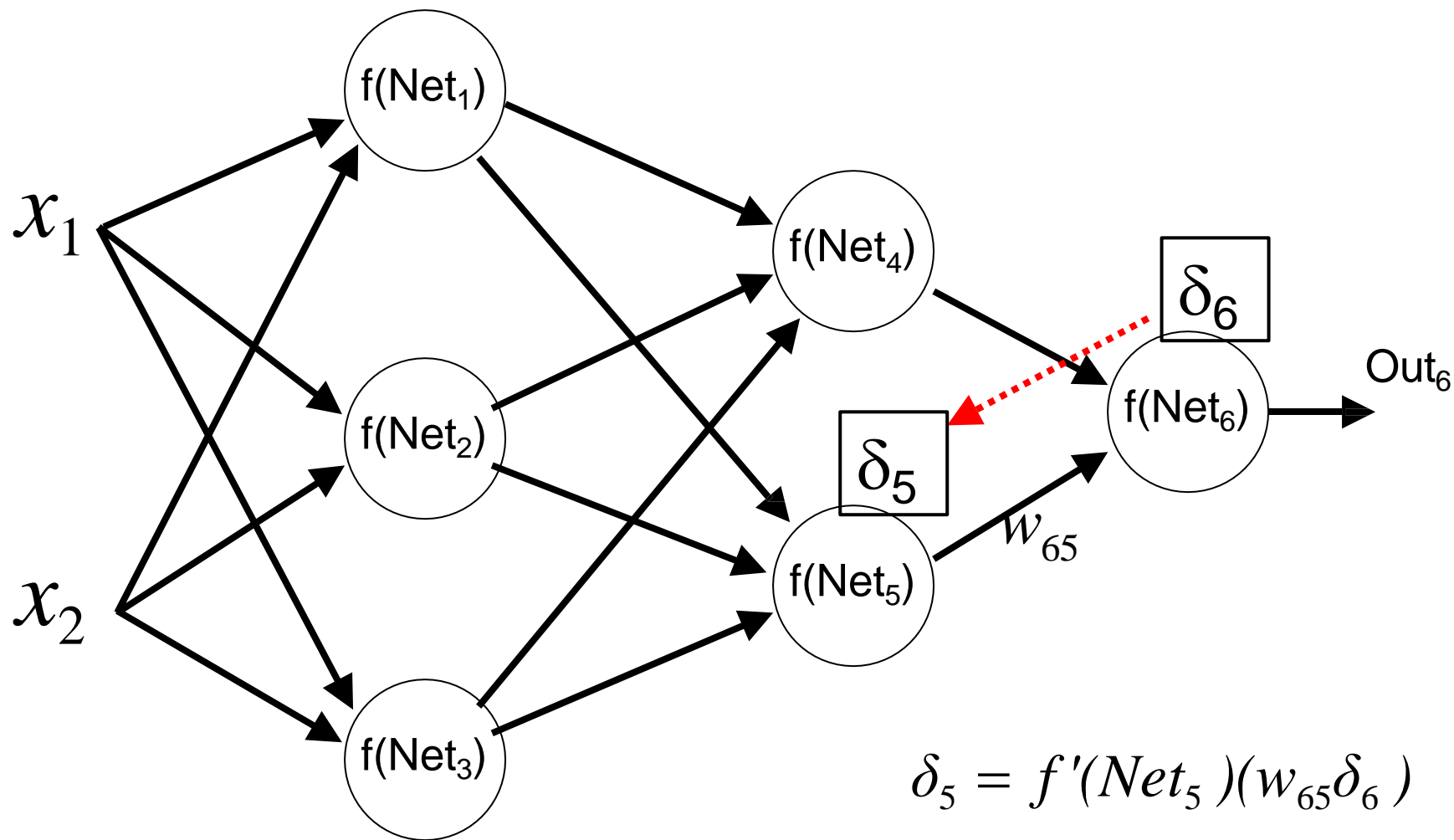
Giải thuật BP: Tính toán lỗi



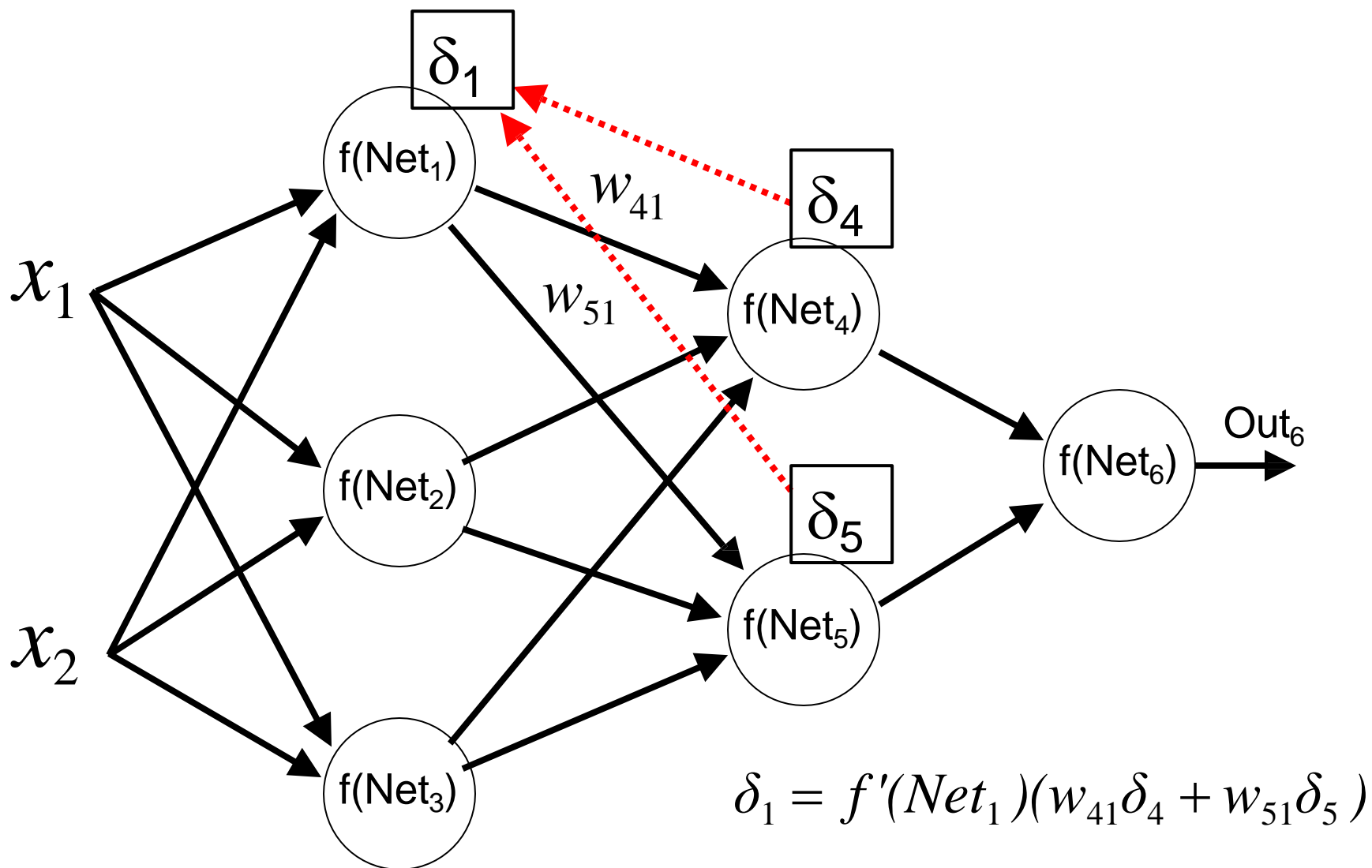
Giải thuật BP: Lan truyền ngược (1)



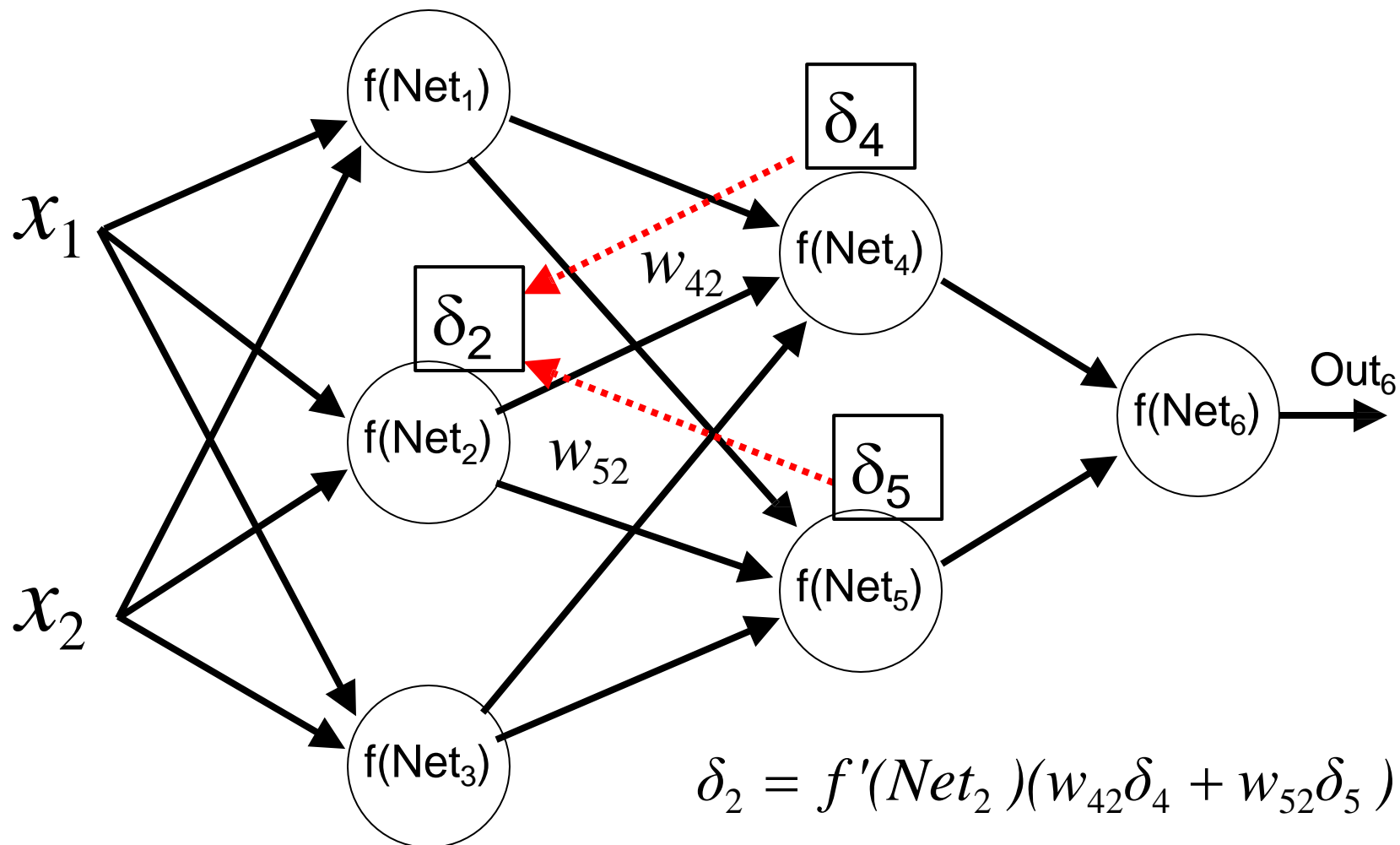
Giải thuật BP: Lan truyền ngược (2)



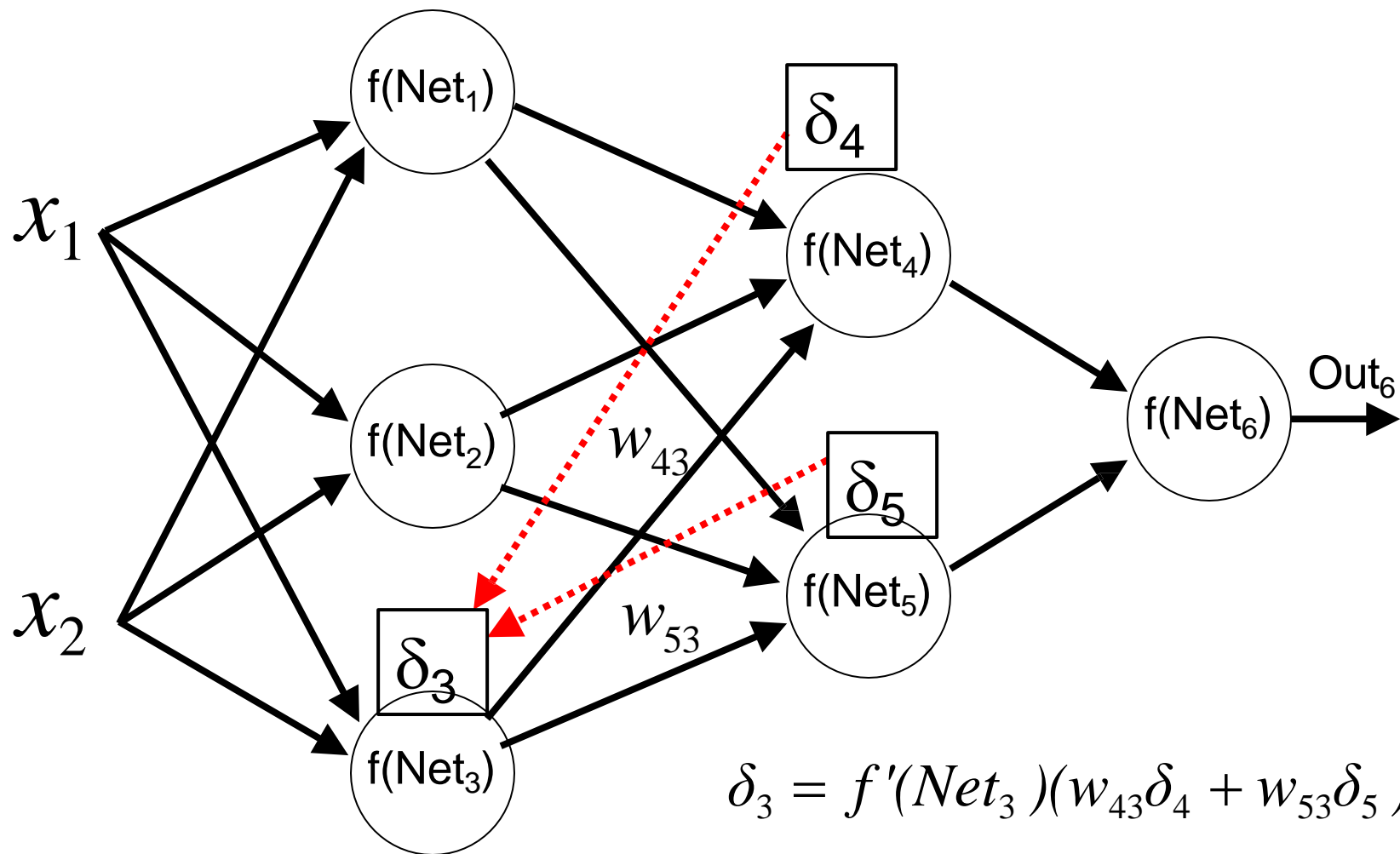
Giải thuật BP: Lan truyền ngược (3)



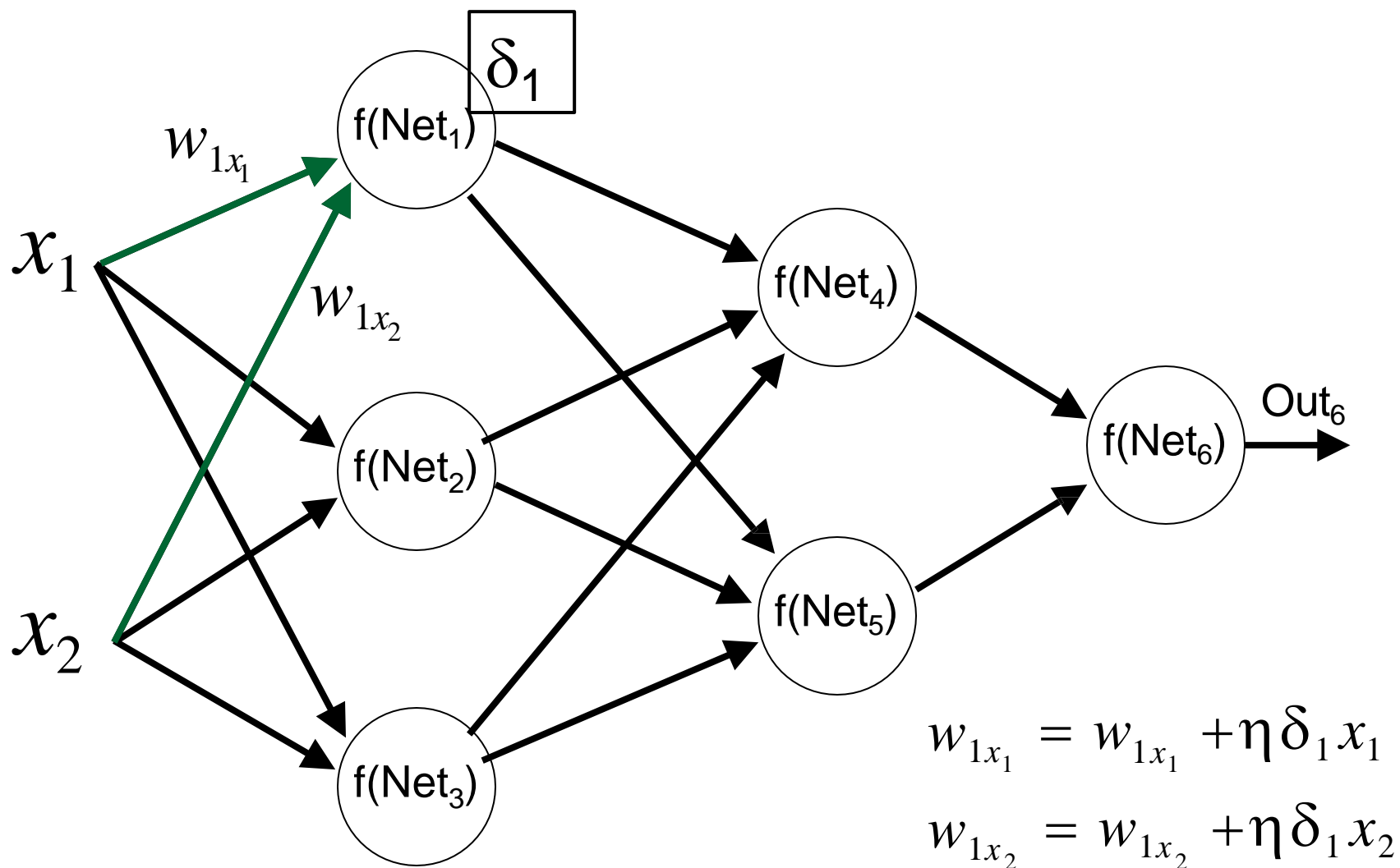
Giải thuật BP: Lan truyền ngược (4)



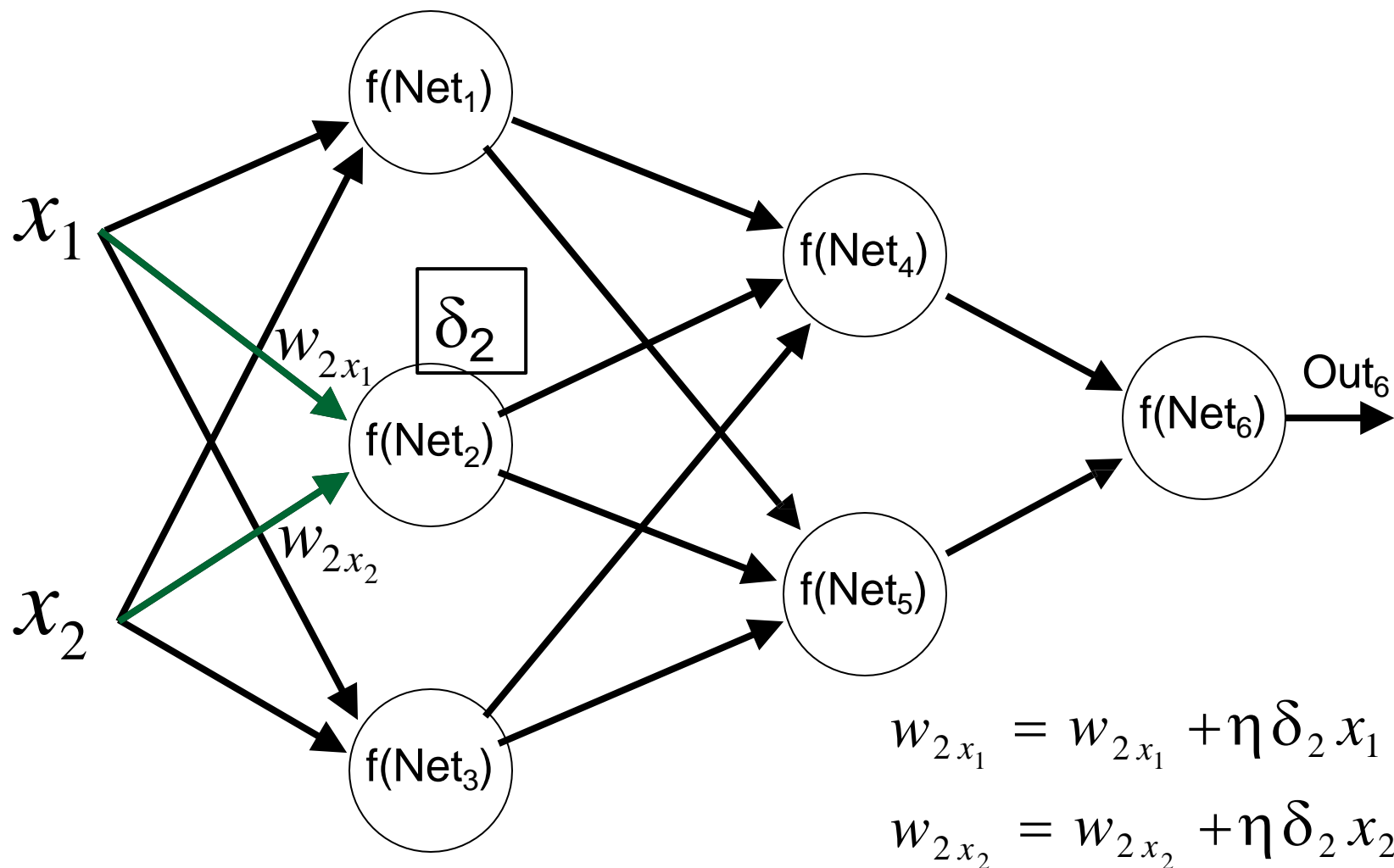
Giải thuật BP: Lan truyền ngược (5)



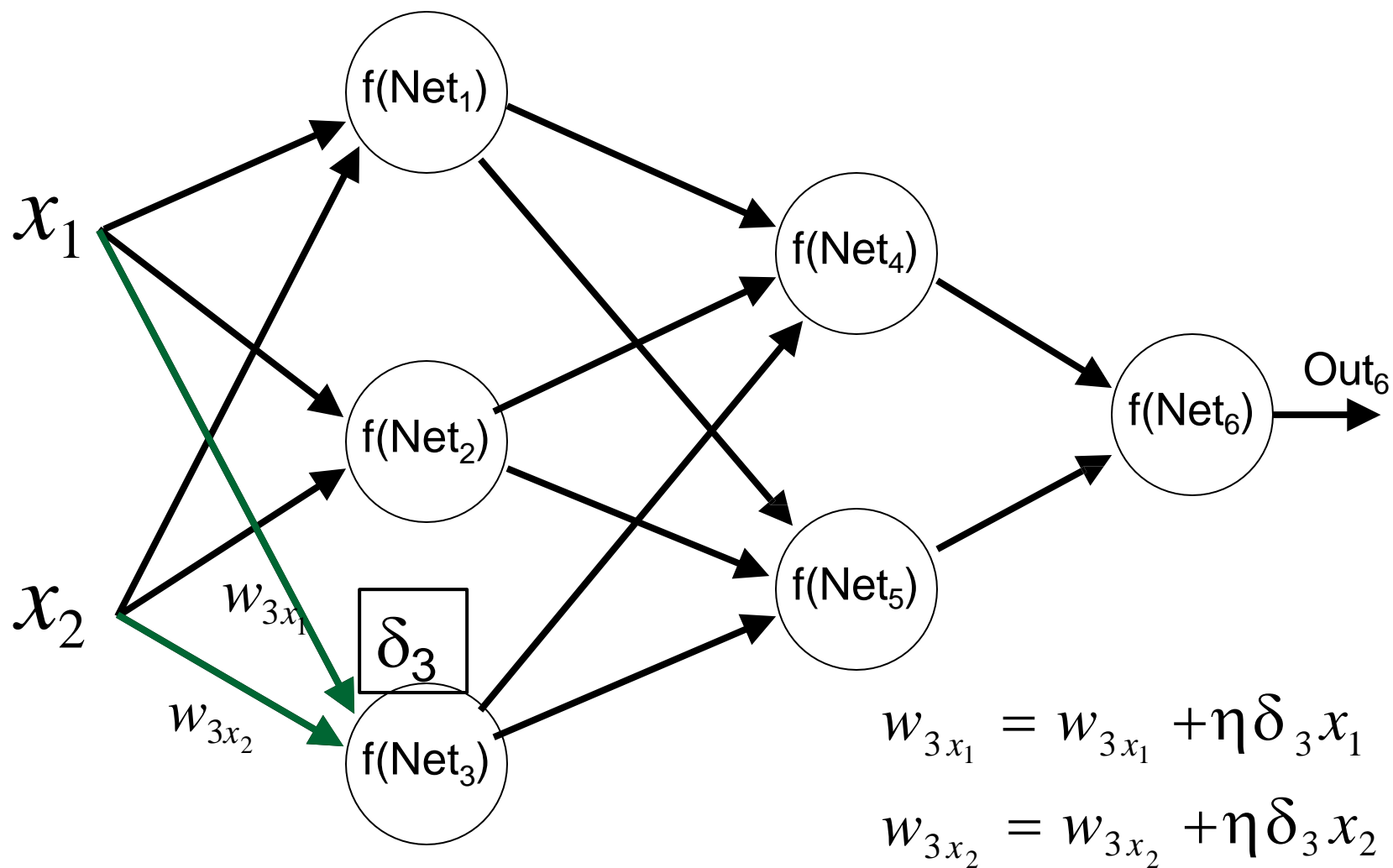
Giải thuật BP: Cập nhật trọng số (1)



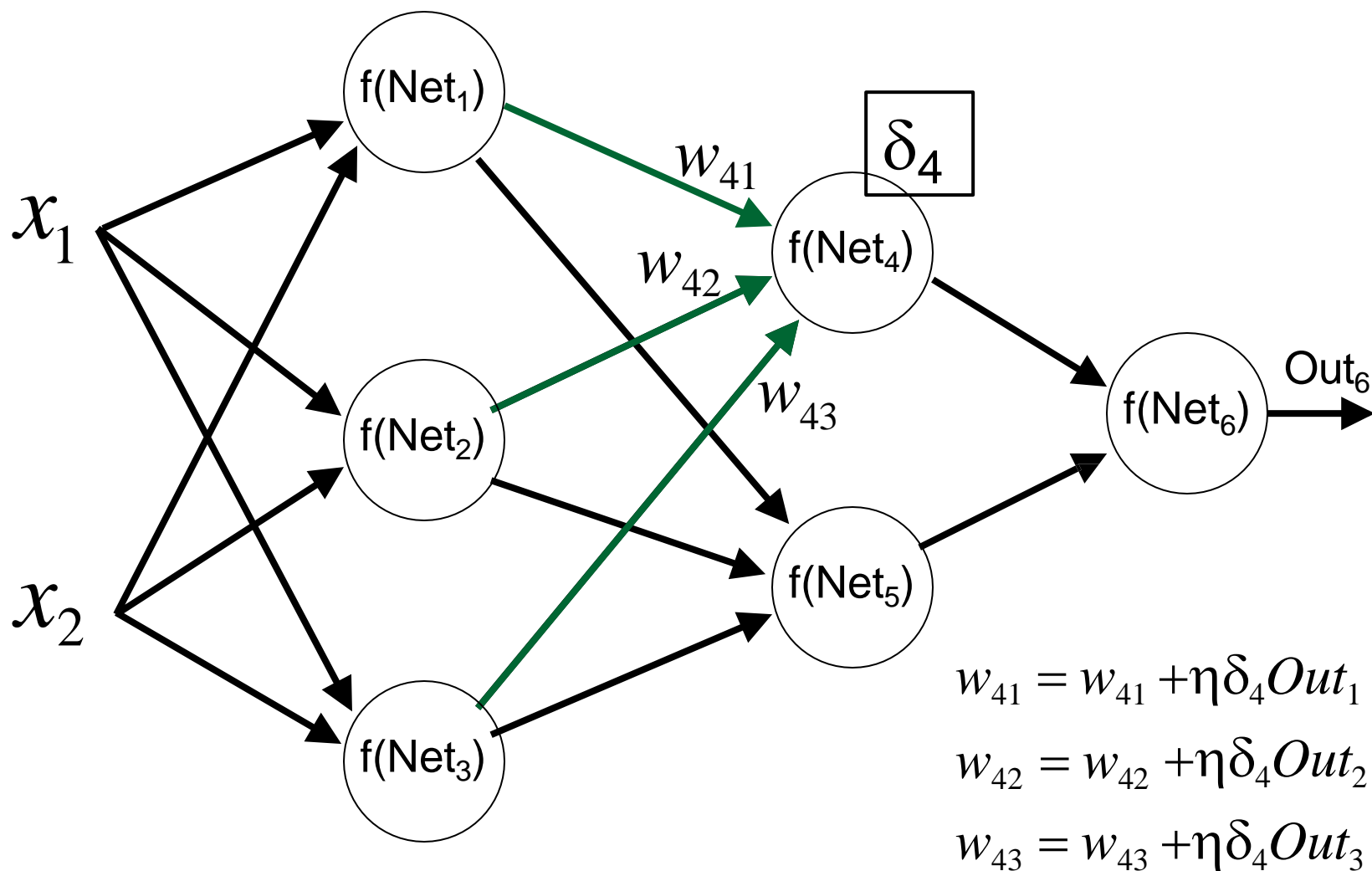
Giải thuật BP: Cập nhật trọng số (2)



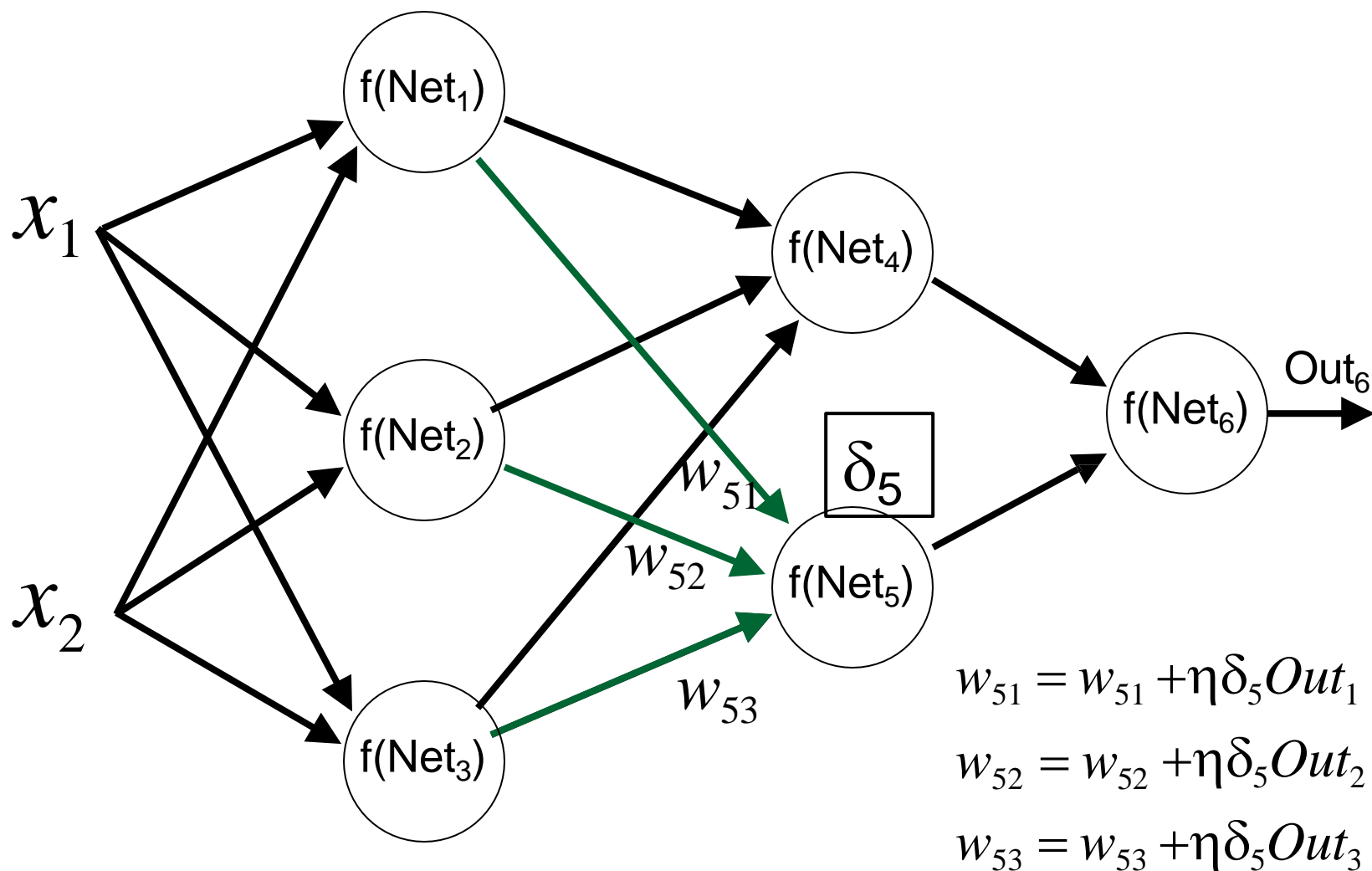
Giải thuật BP: Cập nhật trọng số (3)



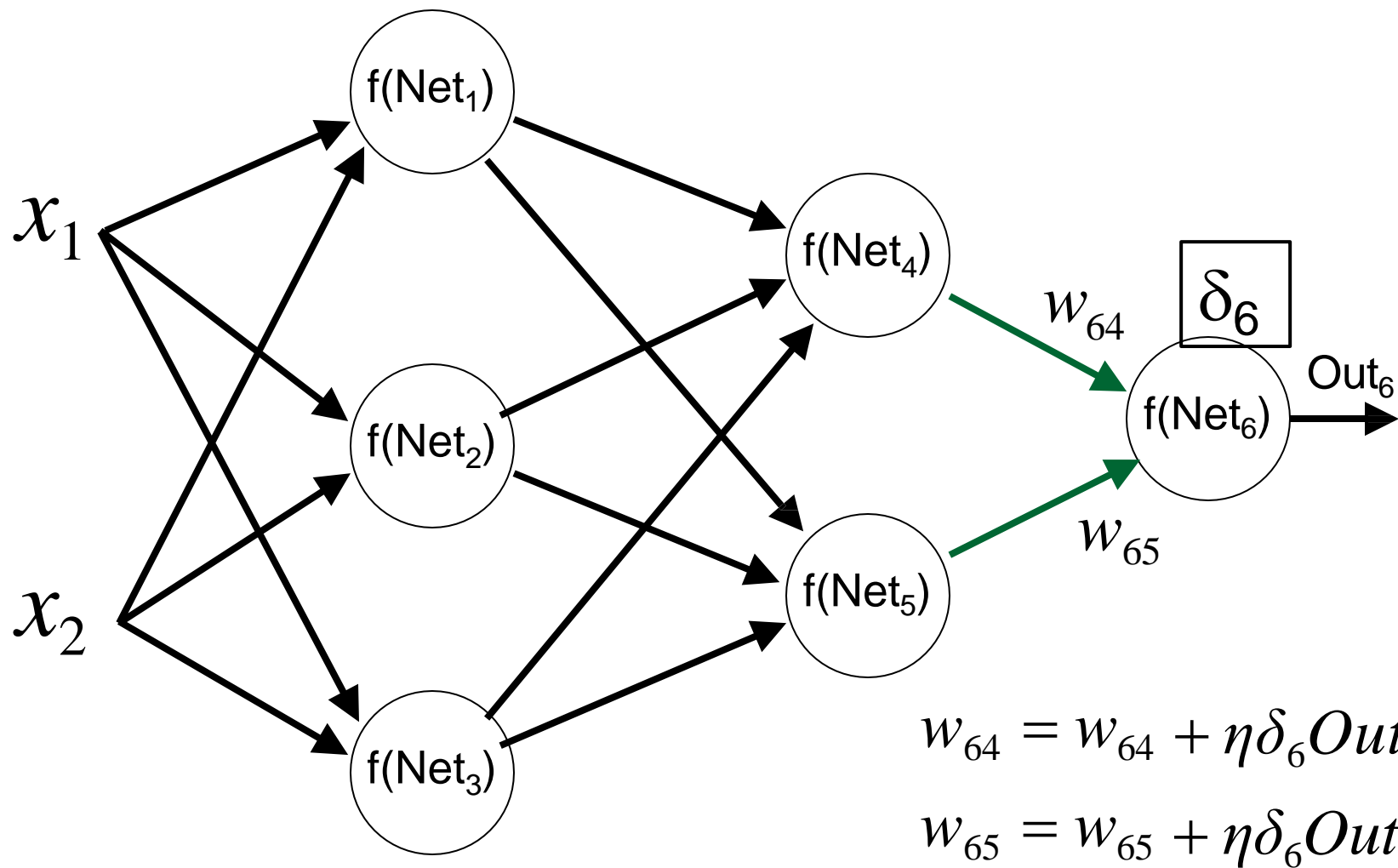
Giải thuật BP: Cập nhật trọng số (4)



Giải thuật BP: Cập nhật trọng số (5)



Giải thuật BP: Cập nhật trọng số (6)



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 sigmoid sẽ đạt trạng thái bão hòa sớm
 - Hệ thống sẽ tắc ở một điểm yên ngựa (saddle/stationary points)

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 hội tụ vào điểm không tốt (saddle points)
 - 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 đây
 - Sử dụng một tốc độ học thích nghi (động)?

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

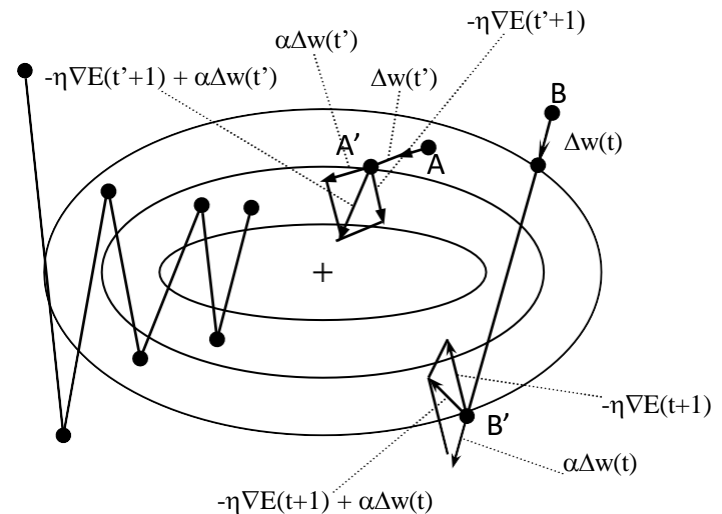
$$\Delta w^{(t)} = -\eta \nabla E^{(t)} + \alpha \Delta w^{(t-1)}$$

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

- Dựa trên các kinh nghiệm, ta nên chọn các giá trị hợp lý cho tốc độ học và momentum thoả mãn

$$(\eta + \alpha) \gtrsim 1$$

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 (experiment/trial and test)

ANN: Giới hạn học

- Các hàm nhị phân (Boolean functions)
 - Bất kỳ hàm nhị phân nào cũng có thể học được (xấp xỉ tốt) 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., 1991]

ANN: Ưu điểm, Nhược điểm

- Các ưu điểm
 - Bản chất (về cấu trúc) hỗ trợ tính toán song song ở mức cao
 - Đạt độ chính xác cao trong nhiều bài toán (ảnh, video, âm thanh, văn bản)
 - Rất linh động trong kiến 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
 - Lý thuyết nền tảng còn ít, để giúp giải thích được những thành công trong thực tế

ANN: Khi nào?

- Dạng của hàm học không xác định được 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
- Có thể thu thập một lượng lớn các nhãn cho dữ liệu.
- Các miền liên quan đến: image, video, speech, text

ANN (Artificial Neural Network - Mạng Nơ-ron Nhân tạo)

- |— Perceptron (Mô hình ANN đơn giản nhất)
- |— Shallow Neural Networks (Mạng nơ-ron nông - 1 lớp ẩn)
- └─ DNN (Deep Neural Networks - Mạng nơ-ron sâu - nhiều hơn 1 lớp ẩn)
 - |— MLP (Multi-Layer Perceptron - khi có nhiều lớp ẩn)
 - |— CNN (Convolutional Neural Network)
 - └─ RNN (Recurrent Neural Network)
 - └─ LSTM, GRU (Biến thể của RNN)

Tài liệu tham khảo

- Cybenko, G. (1989) "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2 (4), 303-314
- Kurt Hornik (1991) "Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, 4(2), 251–257