Fundamental Concepts and Models of Artificial Neural Systems

Berlin Chen, 2002

Outline: Characteristics of Artificial Neural Systems

- Activation Functions of Neurons: The Basic Definitions of Neuron and Elementary Neural Networks
  - Hard-limiting or soft-limiting thresholds
  - Bipolar or unipolar

- Architectures: a Taxonomy of the Most Important Neural Networks
  - Single neuron, single-layer or multi-layer feedforward, recurrent etc.
- Learning Modes: The Basic Learning Concepts
  - Learning algorithms
  - Supervised/unsupervised learning

Activation Functions of the Neurons

The Biological Neuron

- Neuron is the elementary nerve cell
  - Cell body (soma)
  - Axon
  - Dendrites
- Firing threshold of a neuron is about 40mV
  - Signals generated do not differ significantly in magnitude (absent or max.)
  - Information transmitted between neuron cells by means of binary signals

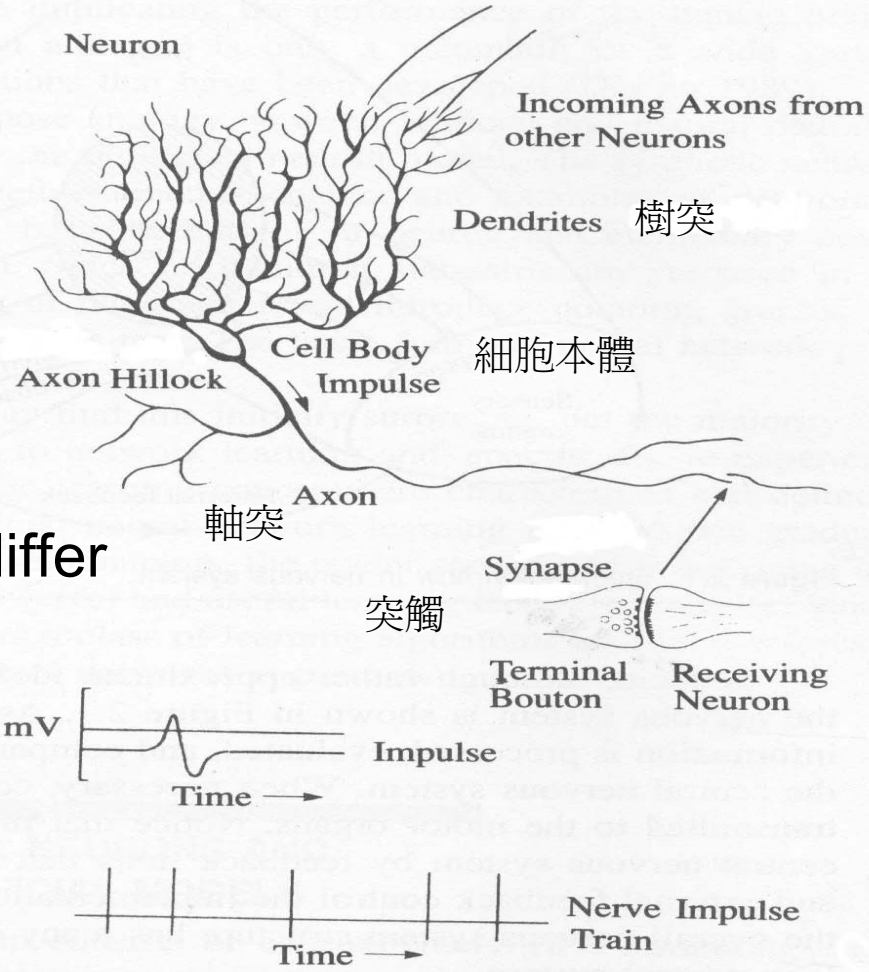


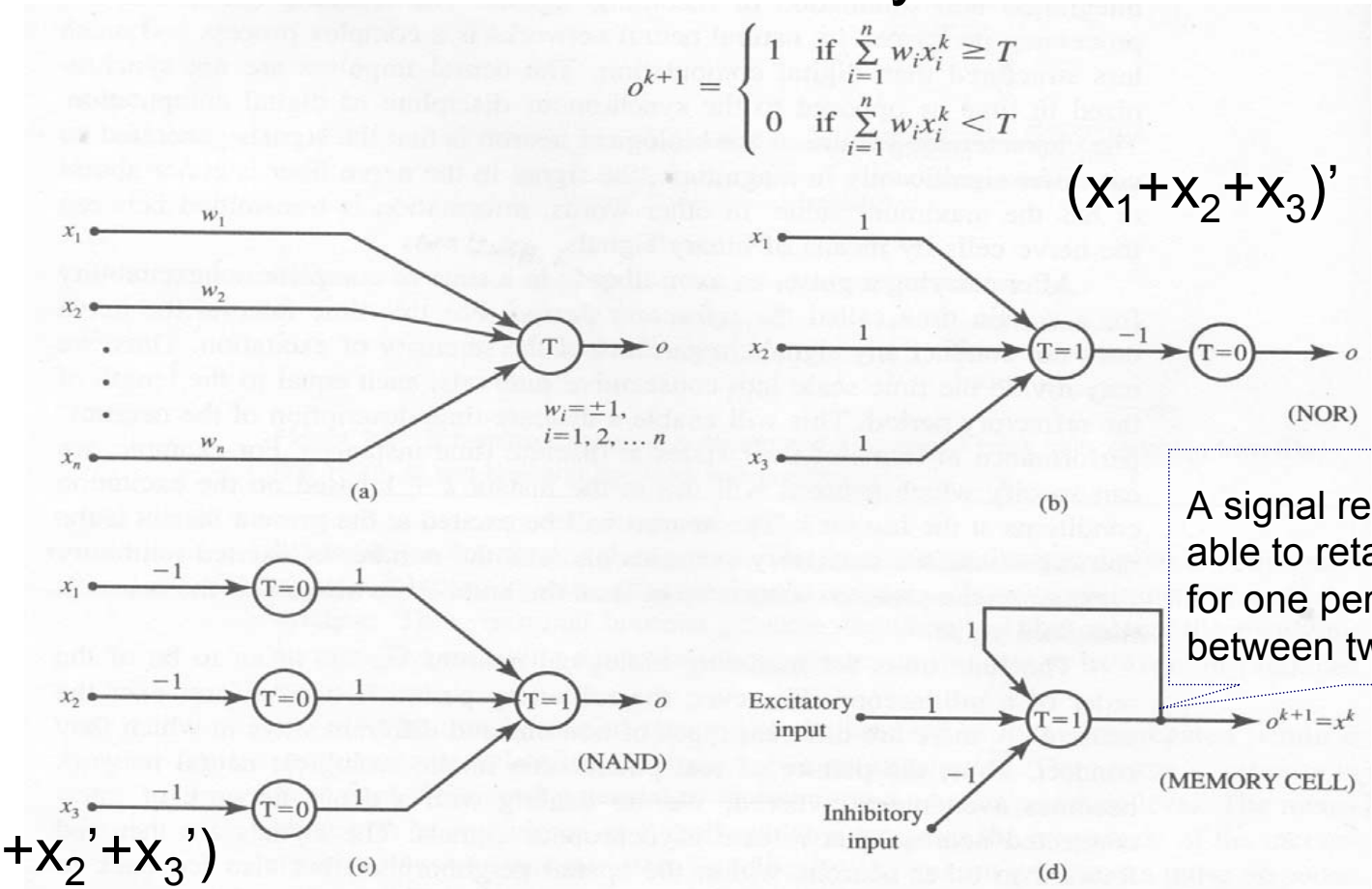
Figure 2.2 Schematic diagram of a neuron and a sample of pulse train.

McCulloch-Pitts Neuron Model

- A first formal definition of a synthetic neuron

$$o^{k+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$

$(x_1+x_2+x_3)'$



A signal register cell able to retain the input for one period elapsing between two instants

$(x_1'+x_2'+x_3')$
$=(x_1x_2x_3)'$

Figure 2.3 McCulloch-Pitts model neuron and elementary logic networks: (a) model diagram, (b) NOR gate, (c) NAND gate, and (d) memory cell.

McCulloch-Pitts Neuron Model

- A unit delay elapses between the instants *k* and *k+1*
- **Fixed weights:** w_i= +1 for excitatory synapses, w_i= -1 for inhibitory synapses
- **Fixed thresholds:** *T* is the neuron's threshold value
  - Needs to be exceeded by the weighted sum of signals for the neuron to fire
- The McCulloch-Pitts Neuron Model can be used to implement
  - Any multivariable combinational functions
  - Simple memory/register cells

General Symbol of a Neuron Model

- A processing element/unit with input connections and a single output

Neuron output signal o,

$$o = f\left(\mathbf{w}^t\mathbf{x}\right)$$

$$= f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Weight vector $\mathbf{w}$,

$$\mathbf{w} = \left[w_1\ w_2 \quad w_n\right]^t$$

Input vector $\mathbf{x}$,

$$\mathbf{x} = \left[x_1\ x_2 \quad x_n\right]^t$$

Activation function $f(.)$,

Activation value, or net, is $\sum_{i=j}^{n} w_j x_j$

$$net = \mathbf{w}^t\mathbf{x}$$

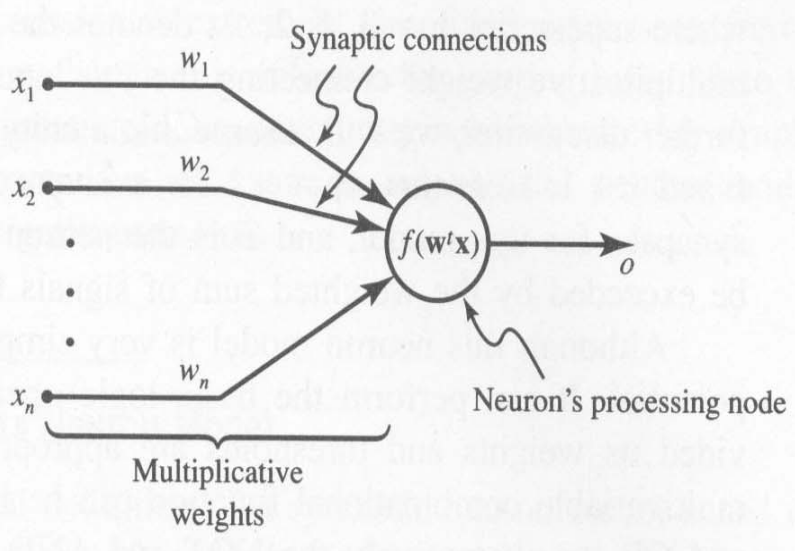


Figure 2.4 General symbol of neuron consisting of processing node and synaptic connections.

Activation Functions of Neurons

- Bipolar continuous and binary functions

$$f(net) = \frac{2}{1 + \exp(-\lambda \cdot net)} - 1, \text{ where } \lambda > 0$$

$$f(net) = \operatorname{sgn}(net) = \begin{cases} +1, & net > 0 \\ -1, & net < 0 \end{cases}$$

Bipolar means both positive and negative responses of neurons are produced

- Unipolar continuous and binary functions

$$f(net) = \frac{2}{1 + \exp(-\lambda \cdot net)}, \text{ where } \lambda > 0$$

$$f(net) = \operatorname{sgn}(net) = \begin{cases} 1, & net > 0 \\ 0, & net < 0 \end{cases}$$

Shifting and scaling of the bipolar activations functions

Activation Functions of Neurons

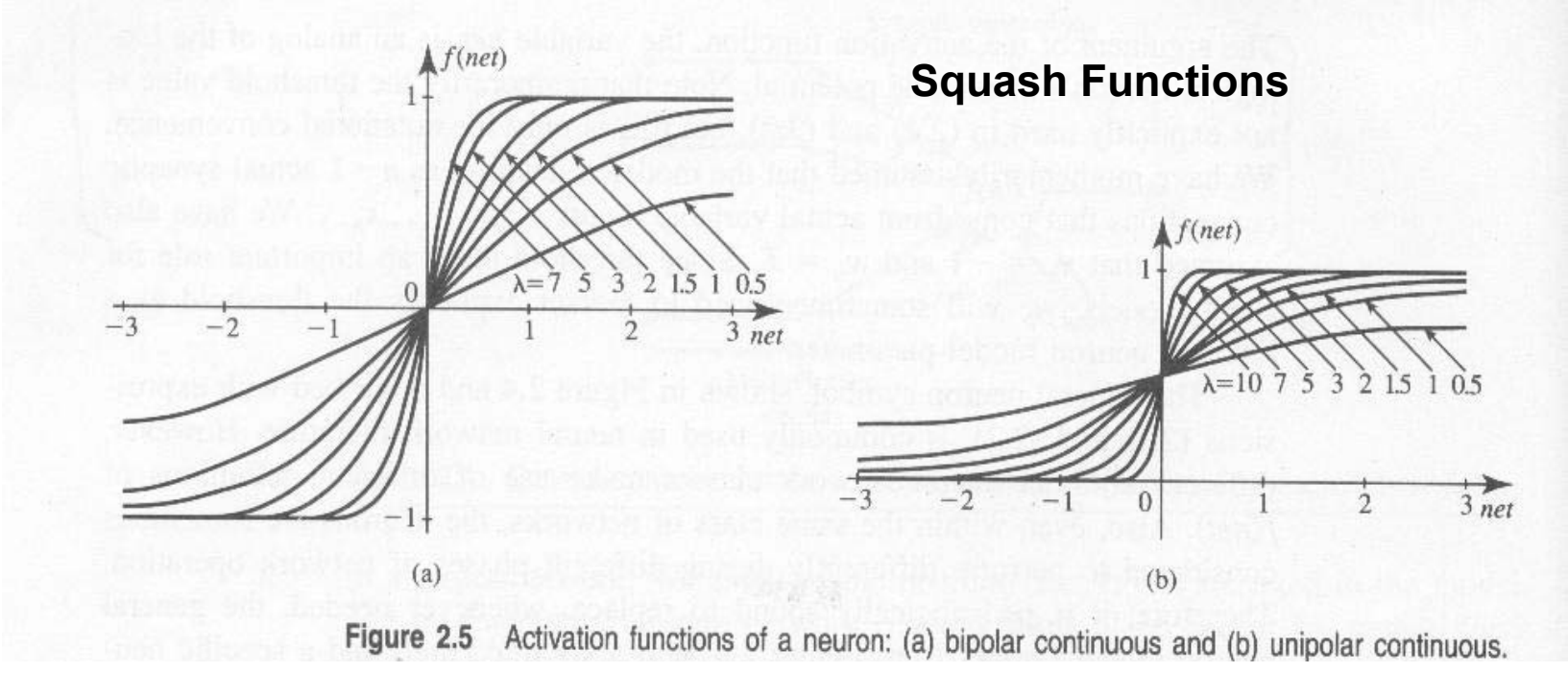


Figure 2.5 Activation functions of a neuron: (a) bipolar continuous and (b) unipolar continuous.

– The soft-limiting activation functions, the bipolar and unipolar continuous functions, are often called sigmoidal characteristics (sigmoid functions), as opposed to the hard-limiting activation functions, the bipolar and unipolar binary functions

Activation Functions of Neurons

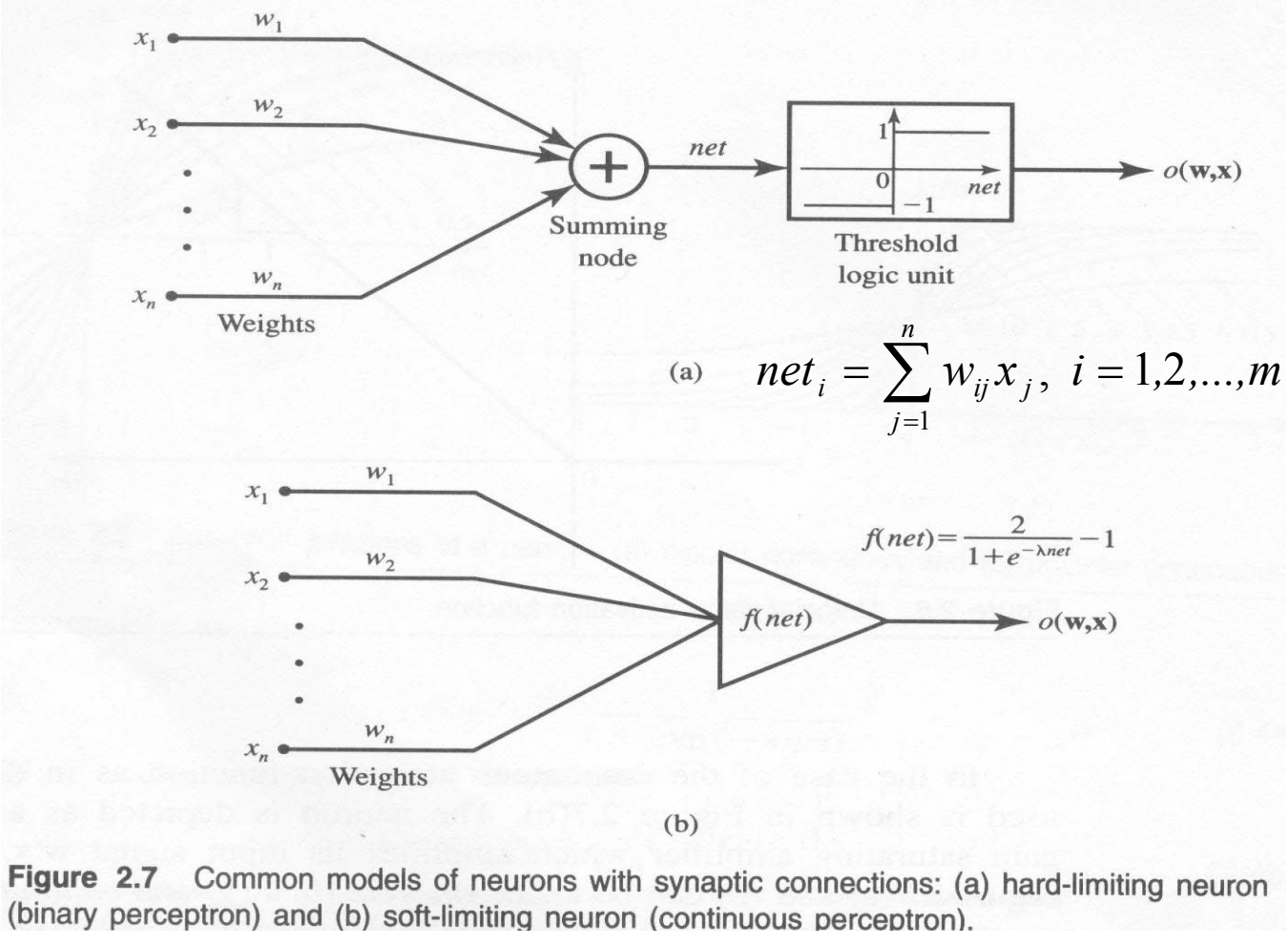
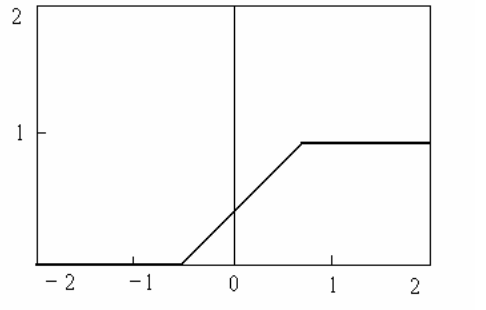


Figure 2.7 Common models of neurons with synaptic connections: (a) hard-limiting neuron (binary perceptron) and (b) soft-limiting neuron (continuous perceptron).

Other Activation Functions

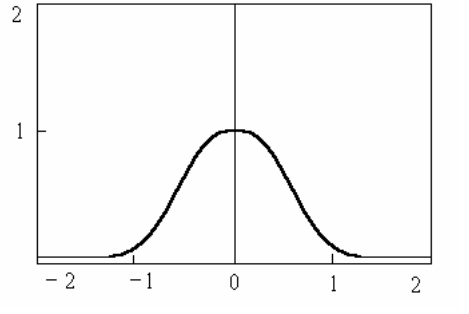
- Piecewise Linear Function (區域線性函數)

$$f(v) = \begin{cases} 1 & if\, v > v_1 \\ cv & if\, v_2 \le v \le v_1 \\ 0 & if\, v < v_2 \end{cases}$$



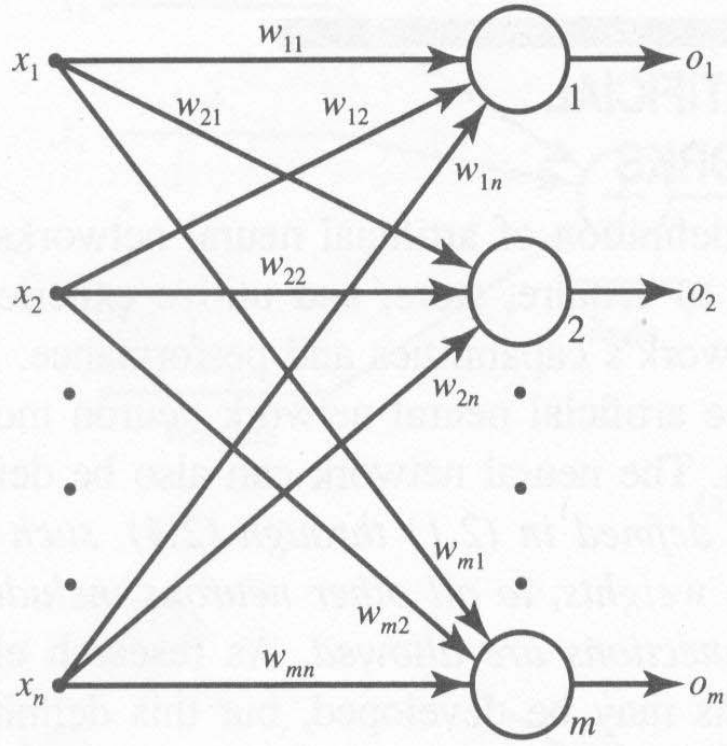
- Gaussian Function(高斯函數)

$$f(v) = \exp\left(-\frac{v^2}{2\sigma^2}\right)$$

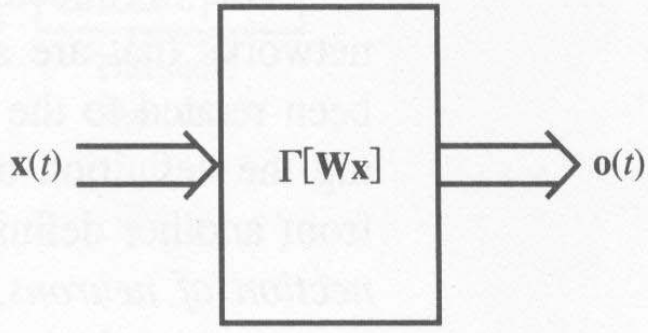


参考自蘇木村教授著作

Feedforward Network (前饋式網路)



$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^t \\ \mathbf{w}_2^t \\ . \\ . \\ \mathbf{w}_m^t \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ . & . & \dots & . \\ . & . & \dots & . \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$



$$\Gamma[\cdot] = \begin{bmatrix} f(\cdot) & 0 & \dots & 0 \\ 0 & f(\cdot) & \dots & 0 \\ . & . & \dots & . \\ . & . & \dots & . \\ 0 & 0 & \dots & f(\cdot) \end{bmatrix}$$

$$net_i = \mathbf{w}_i^t \mathbf{x} = \sum_{j=1}^{n} w_{ij} x_j, \quad i = 1,2,\dots,m \quad \text{(a)}$$

$$o_i = f\left(\mathbf{w}_i^t \mathbf{x}\right), \quad i = 1,2,\dots,m \quad \text{(b)}$$

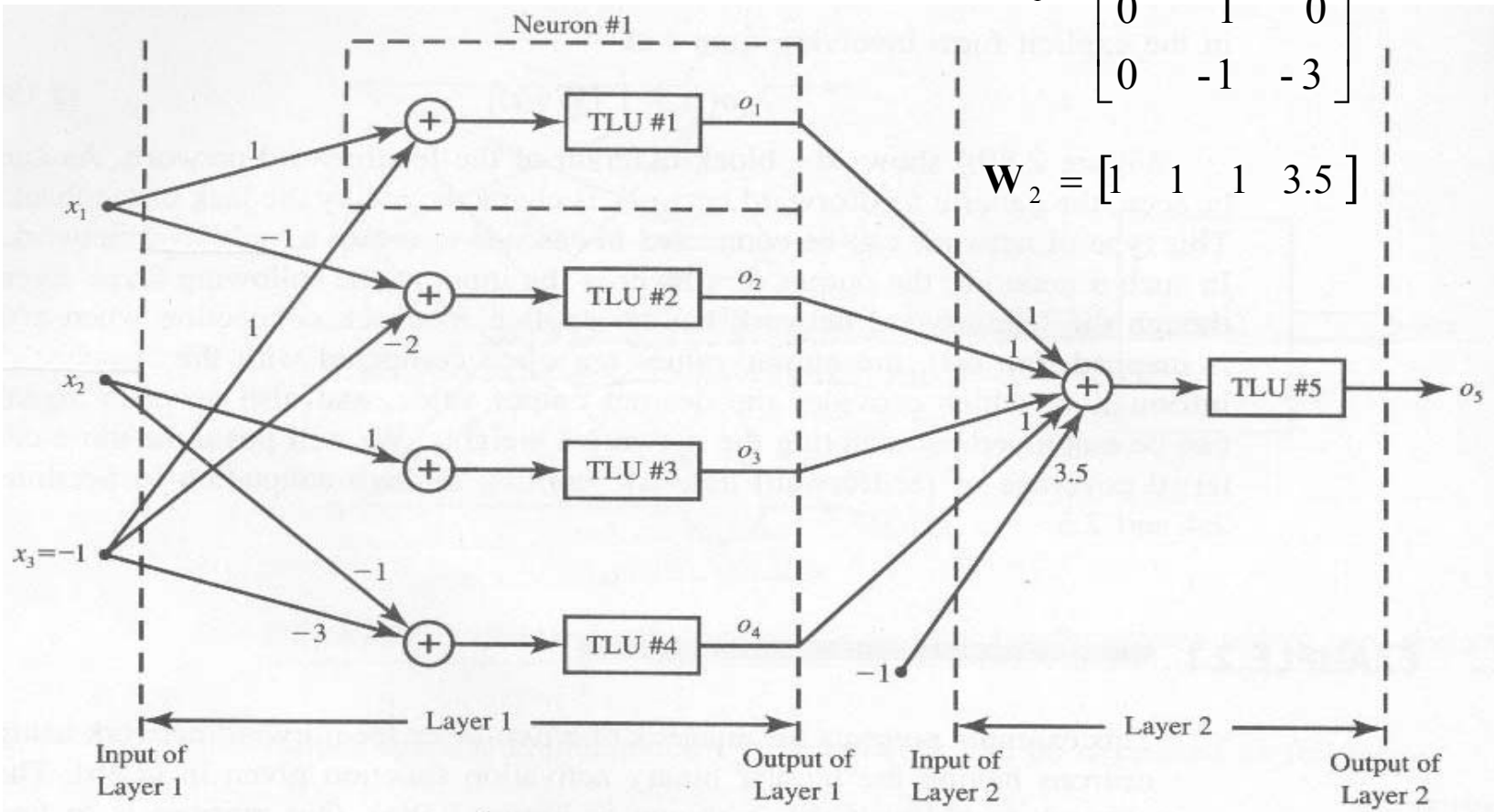
Figure 2.8 Single-layer feedforward network: (a) interconnection scheme and (b) block diagram.

Feedforward Network (cont.)

- Example 2.1

$$\mathbf{W}_1 = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & -1 & -3 \end{bmatrix}$$

$$\mathbf{W}_2 = \begin{bmatrix} 1 & 1 & 1 & 3.5 \end{bmatrix}$$



(a)

Multi-Layer Feedforward Network

Feedforward Network (cont.)

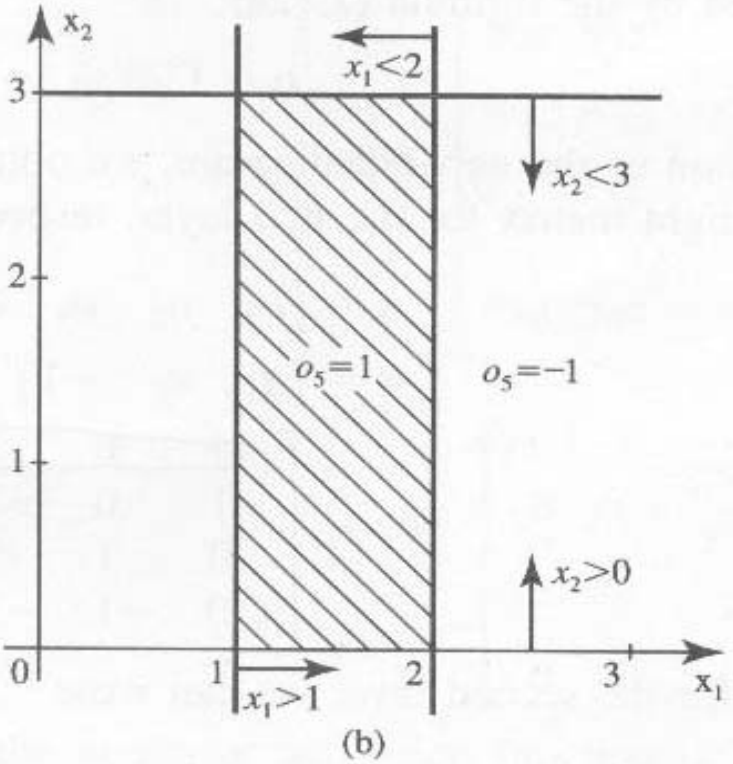


Figure 2.9a,b Example of two-layer feedforward network: (a) diagram and (b) two-dimensional space mapping (discrete activation function).

Feedforward Network (cont.)

P2.4 The feedforward network shown in Figure P2.4 using bipolar binary neurons is mapping the entire plane x_1, x_2 into a binary o value. Find the segment of the x_1, x_2 plane for which $o_4 = 1$, and its complement for which $o_4 = -1$.

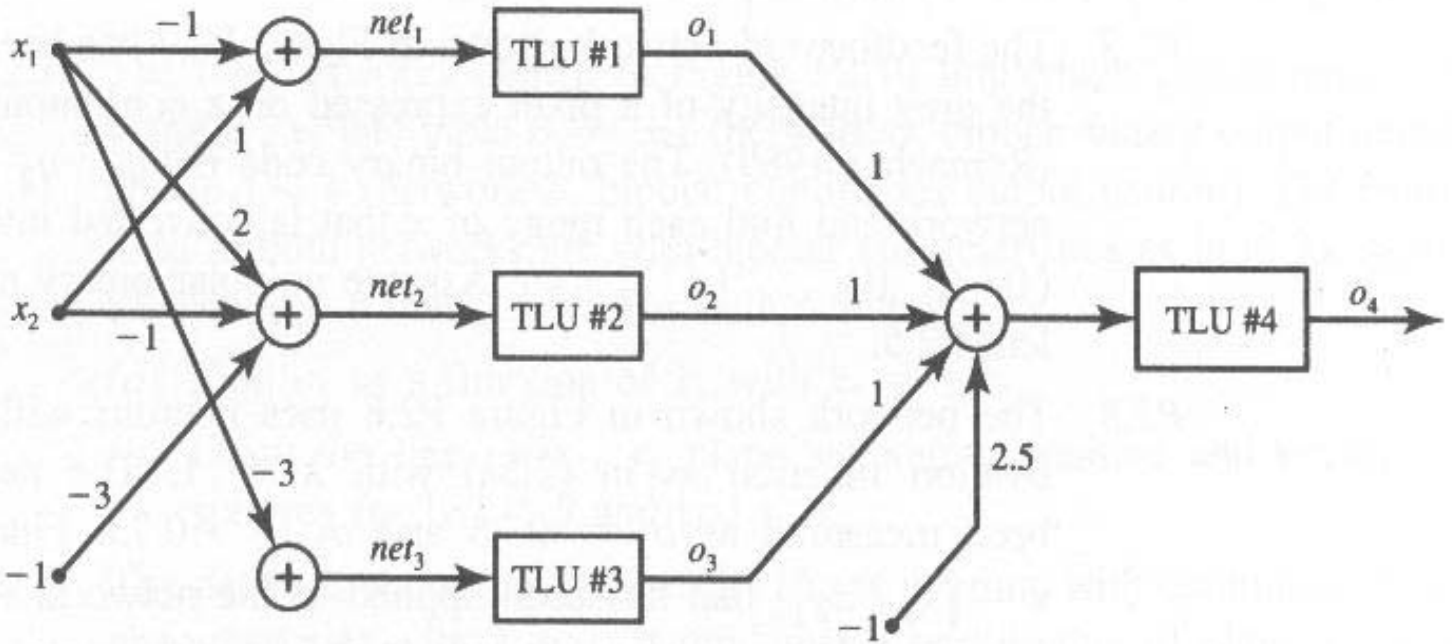


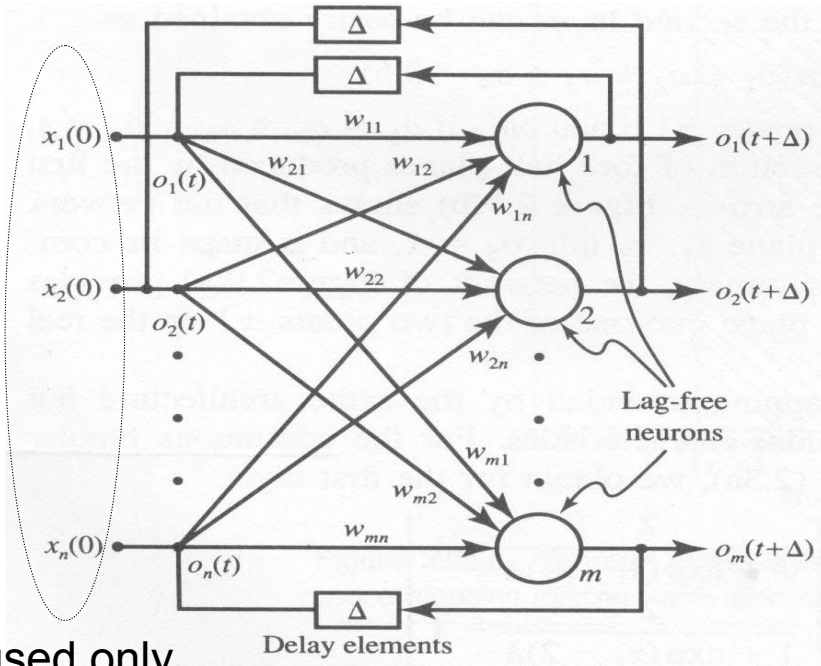
Figure P2.4 Feedforward network for Problem P2.4.

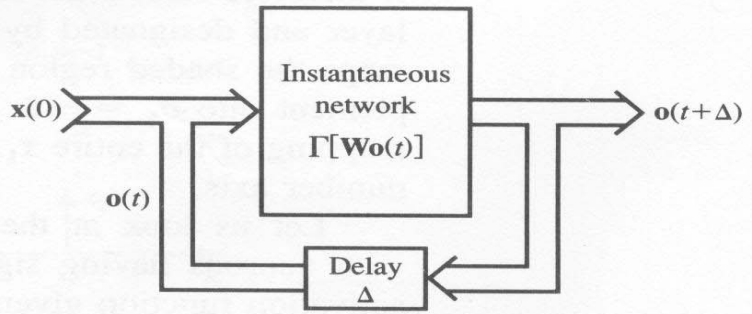
Feedback/Recurrent Network (循環式網路)

- The essence of closing the feedback loop
  - Enable the control of a specific output through the other outputs

$$\mathbf{o}^{k+1} = \Gamma\left[\mathbf{W}\mathbf{o}^{k}\right], \quad \text{for } k = 1,2,\ldots$$

$$\mathbf{o}^{k+1} = \Gamma\left[\mathbf{W}\Gamma\left[\ldots\Gamma\left[\mathbf{W}\mathbf{x}^{0}\right]\ldots\right]\right], \quad \text{where } \mathbf{x}^{0} = \mathbf{o}^{0}$$





Δ : means a unit delay

Γ : hard-limiting activation function

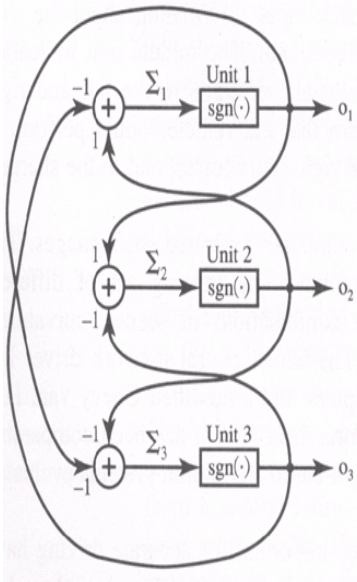
X(0) used only for initialization

Figure 2.10 Single-layer discrete-time feedback network: (a) interconnection scheme and (b) block diagram.

Feedback/Recurrent Network (cont.)

- $\mathbf{o}^{k+1} = \Gamma\left[\mathbf{W}\mathbf{o}^{k}\right]$ describes the state $\mathbf{o}^{k}$ of the network, and yields the sequence of state transitions
  - Initialized at instant 0 with $\mathbf{o}^0 = \mathbf{x}^0$
  - Goes through state transitions until it possible finds an equilibrium state (called attractor) or a limited number of equilibrium states

Feedback/Recurrent Network (cont.)



(a)

Case	Unit Number	Present Output O	Σ	sgn Σ	Next Output O
Started from P_7: 1	1	1	0	x	1
	2	1	0	x	1
	3	1	−2	−1	(−1)
2	1	1	2	1	1
	2	1	2	1	1
	3	−1	−2	−1	−1
Started from P_2: 3	1	−1	2	1	(1)
	2	1	0	x	1
	3	−1	0	x	−1
Started from P_4: 4	1	1	0	x	1
	2	−1	2	1	(1)
	3	−1	0	x	−1

x stands for sgn(0).
Encircled are updated outputs.

(b)

sgn(.)>0, 1
sgn(.)=0, don't change
sgn(.)<0, -1

Figure 1.5 Simple neural network memory: (a) network diagram and (b) listing of updates.

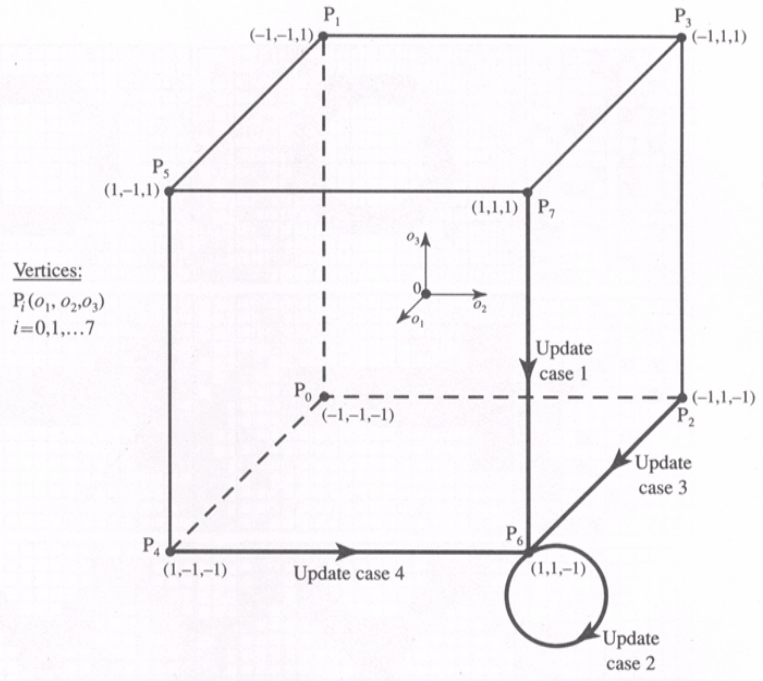
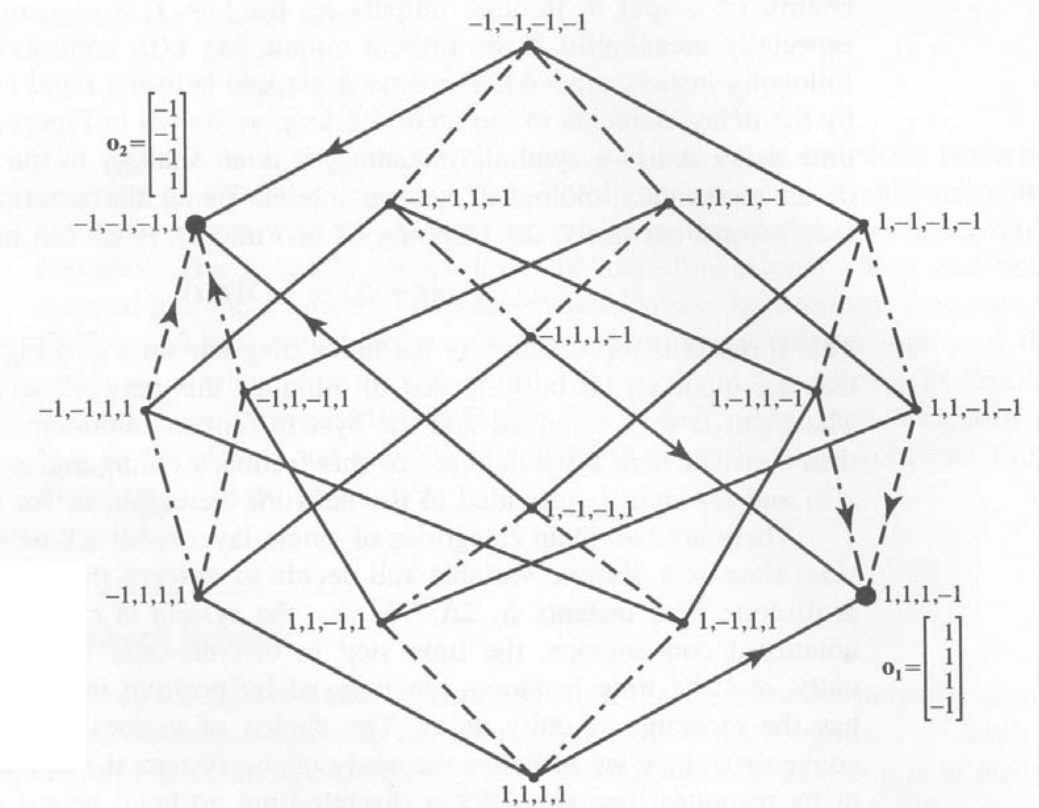


Figure 1.6 Graphical interpretation of the memory network from Figure 1.5.

Feedback/Recurrent Network (cont.)

- Example 2.2
  - Find the output that is most similar to the initial input



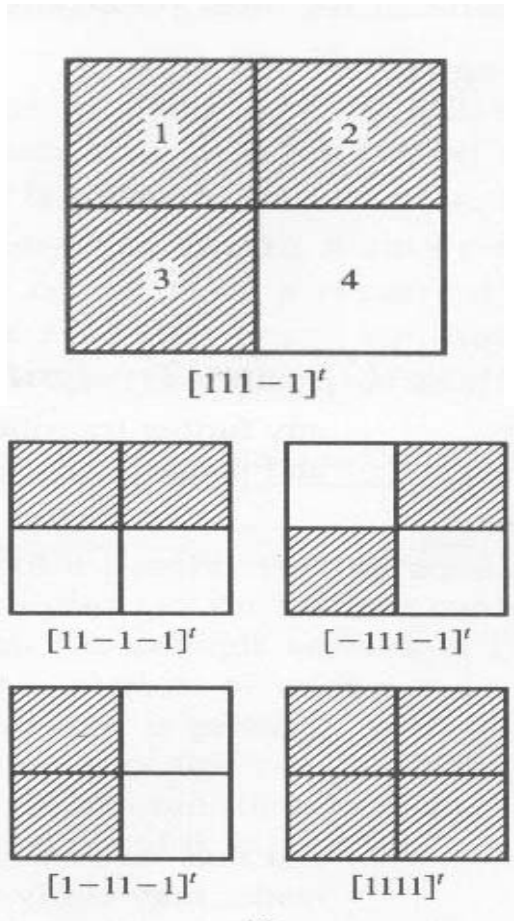
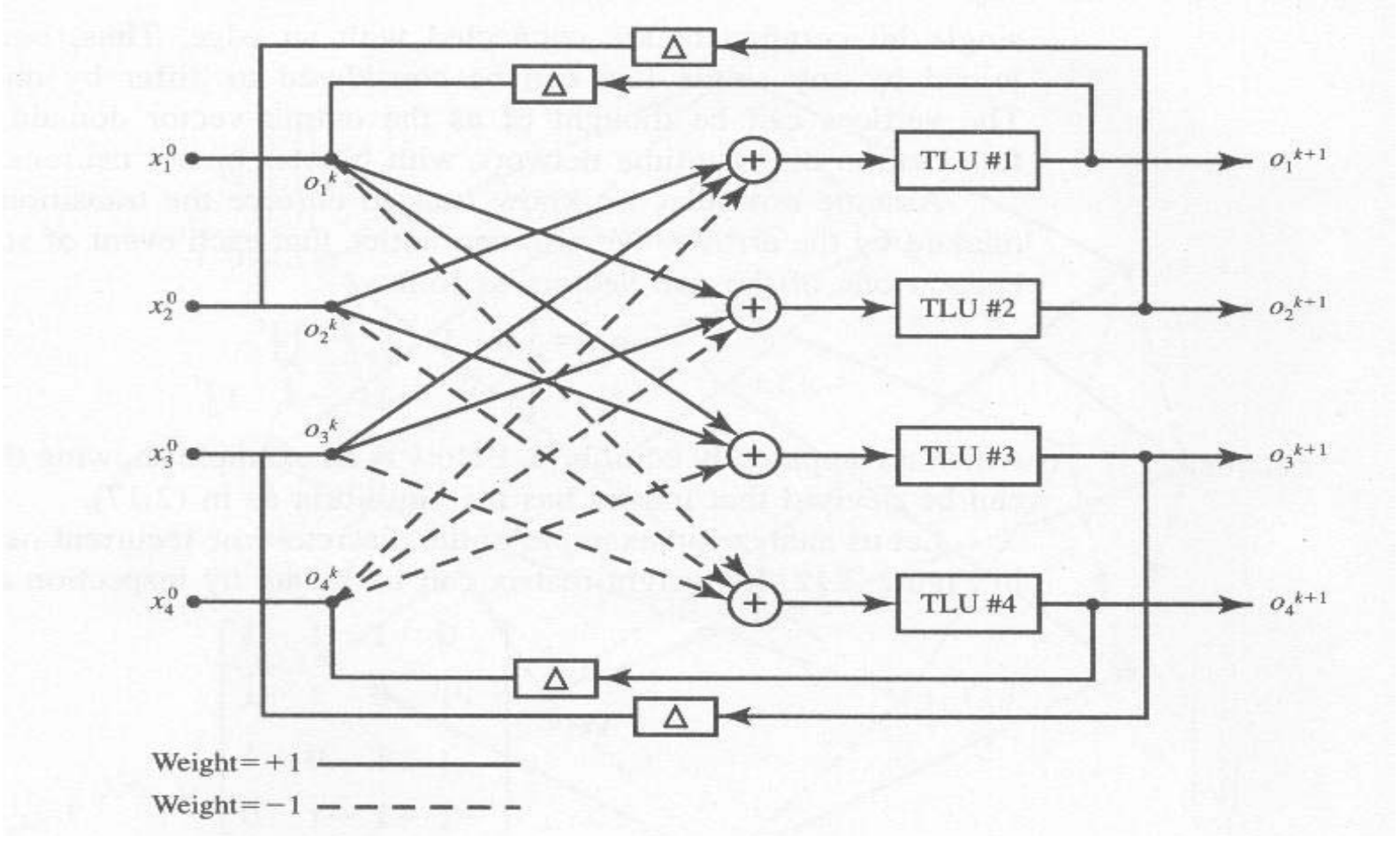


Figure 2.11 Four-dimensional hypercube with two equilibrium states in Example 2.2.

Feedback/Recurrent Network (cont.)

- Example 2.2 (cont.)



Neural Processing

- Recall
  - The process of computation of an output **o** for a given an input **x** perform by the (neural) network
  - It's objective is to retrieve the information, i.e., to decode the stored content which may been encoded in the network previously
- Autoassociation
  - A network is presented a pattern similar to a member of the stored set, it may associate the input pattern with the closest stored pattern

Neural Processing (cont.)

- Heteroassociation
  - The network associates the input pattern with pairs of patterns stored

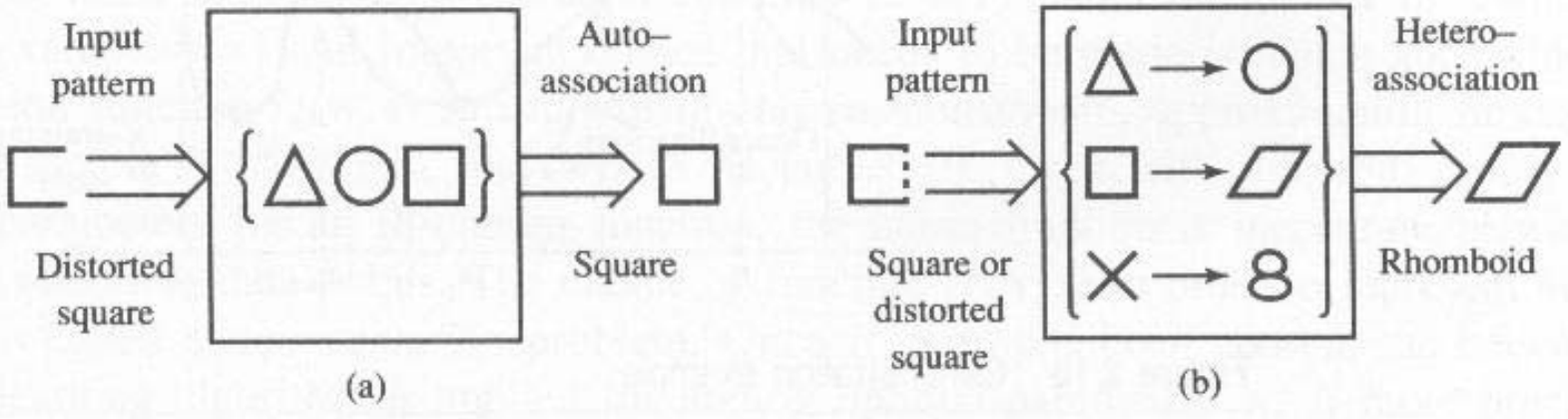


Figure 2.16 Association response: (a) autoassociation and (b) heteroassociation.

Neural Processing (cont.)

- Classification
  - A set of patterns is already divided into a number of classes, or categories
  - When an input pattern is presented, the classifier recalls the information regarding the class membership of the input pattern
  - The classes are expressed by discrete-valued output vectors, thus the output neurons of the classifier employ binary activation functions
  - A special case of heteroassociation
- Recognition
  - If the desired response is the class number, but the input pattern doesn't exactly corresponding to any of the patterns in the stored set

Neural Processing (cont.)

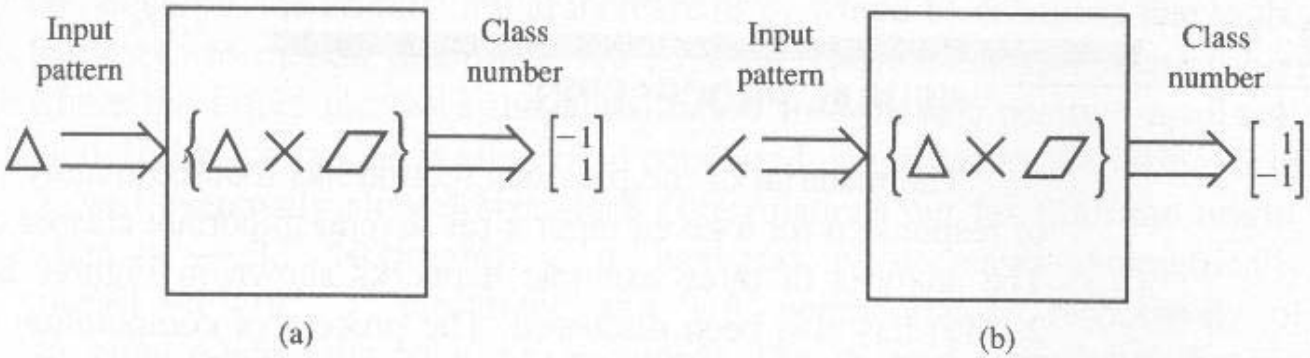


Figure 2.17 Classification response: (a) classification and (b) recognition.

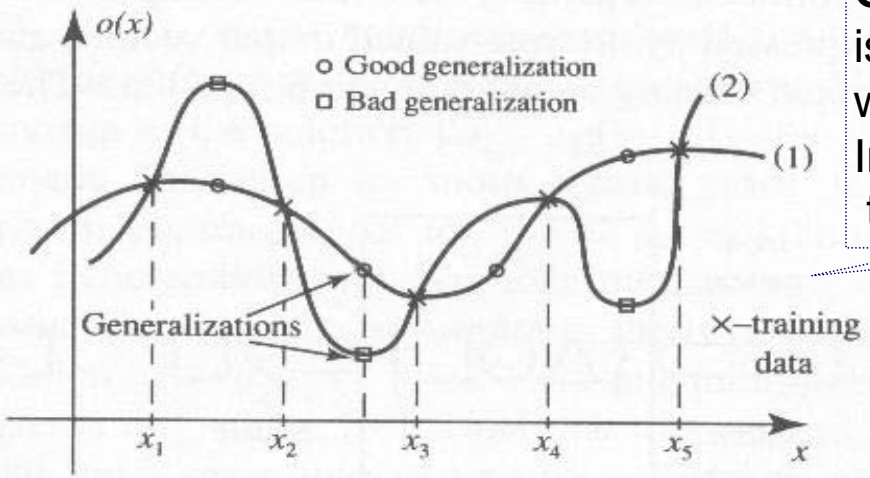


Figure 2.18 Generalization example.

Generation: The network is said to generalize well when it sensibly interpolates Input pattern that are new to the network

Neural Processing (cont.)

- Clustering
  - Unsupervised classification of patterns/objects without providing information about the actual classes
  - The network must discover for itself any existing patterns, regularities, separating properties, etc.
  - While discovering these, the network undergoes change of its parameters, which is called ***self-organization***

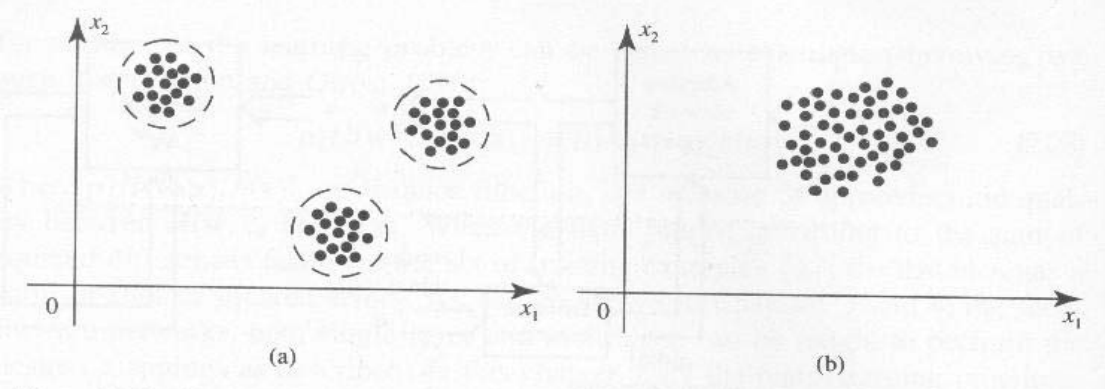


Figure 2.20 Two-dimensional patterns: (a) clustered and (b) no apparent clusters.

Learning and Adaptation

Learning

- So far, we discussed the retrieval of network responses without covering any possible methods of data storage
- Data is stored in a network as a result of learning
- For human beings, we can't see learning happening directly but assume it has occurred by finally observing changes in performance
- For artificial neural networks
  - We can capture each learning step in a distinct cause-effect relation
  - E.g., learning of input-output mapping from a set of (training) examples, especially for feedforward networks

Learning as Approximation

- Approximation theory
  - Focus on approximating a continuous, multivariable function h(**x**) by another function H(**w**,**x**)
    - a representation problem
  - The learning task is to find **w** that best approximates h(**x**) based on a set of training samples

distance function

$$\rho[H(\mathbf{w}^*, \mathbf{x}), h(\mathbf{x})] \leq \rho[H(\mathbf{w}, \mathbf{x}), h(\mathbf{x})]$$

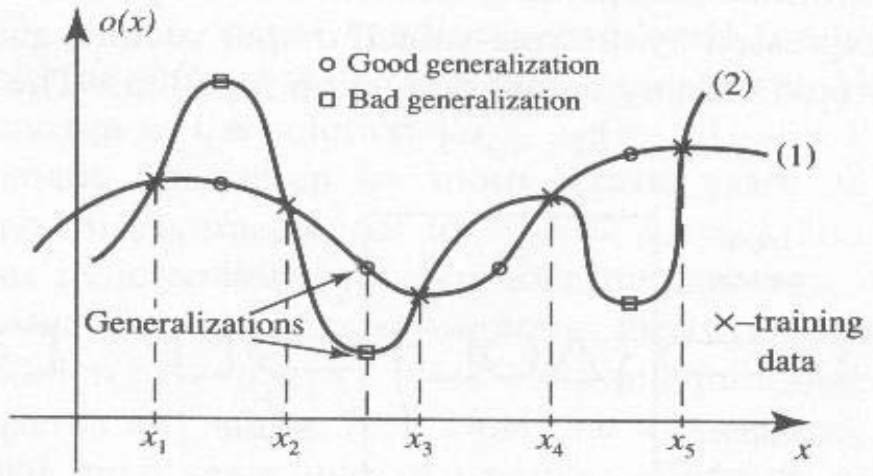


Figure 2.18 Generalization example.

Supervised and Unsupervised Learning

- Supervised vs. unsupervised
- Incremental vs. bath

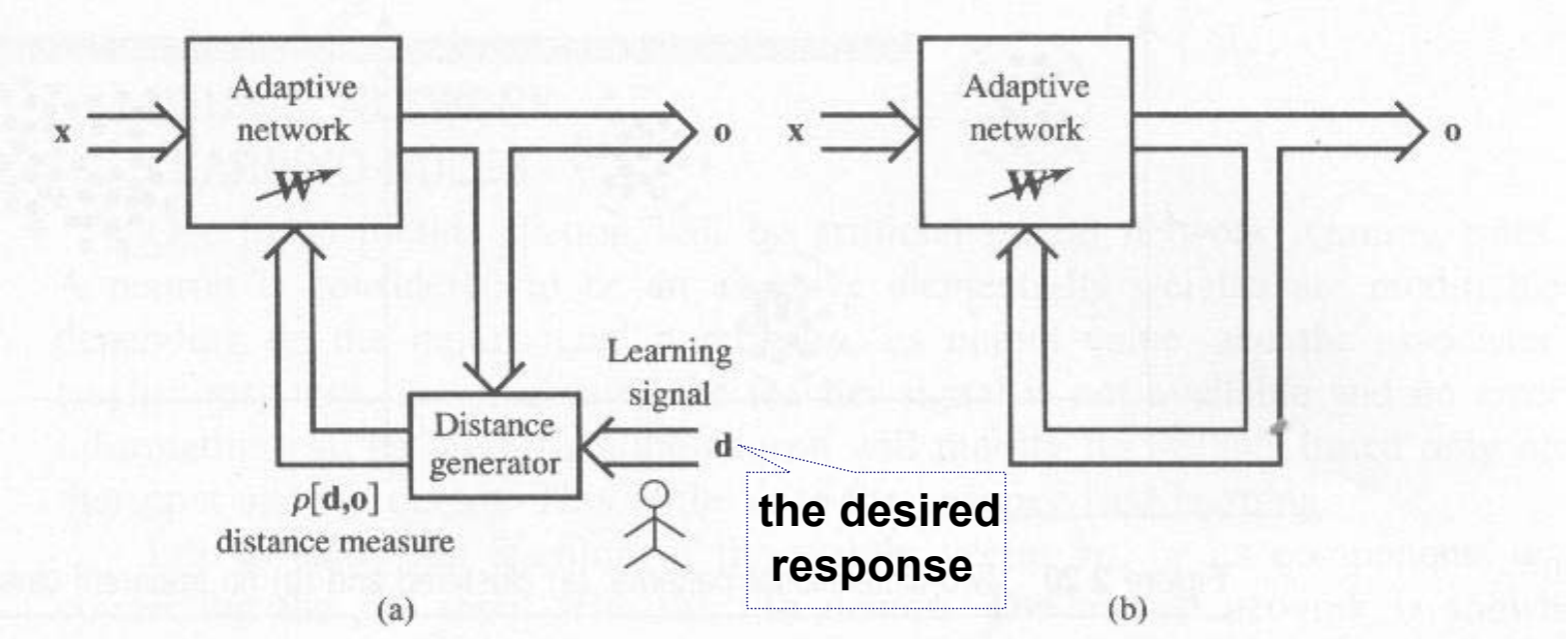


Figure 2.19 Block diagram for explanation of basic learning modes: (a) supervised learning and (b) unsupervised learning.

Supervised and Unsupervised Learning (cont.)

- Supervised learning
  - The desired response of the system is provided by a teacher, e.g., the distance ρ [**d,o**] as as error measure
  - Estimate the negative error gradient direction and reduce the error accordingly
    - Modify the synaptic weights to reduce the stochastic minimization of error in multidimensional weight space
- Unsupervised learning (Learning without a teacher)
  - The desired response is unknown, no explicit error information can be used to improve network behavior
    - E.g. finding the cluster boundaries of input patterns
  - Suitable weight self-adaptation mechanisms have to embedded in the trained network

Neural Network Learning Rules

The General Learning Rule

- The weight adjustment is in proportional to the product of input **x** and the learning signal *r*

$$\Delta \mathbf{w}_i(t) = c \cdot r\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \cdot \mathbf{x}(t), \quad c \text{ is a postive learning constant}$$

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \Delta \mathbf{w}_i(t) = \mathbf{w}_i(t) + c \cdot r\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \cdot \mathbf{x}(t)$$

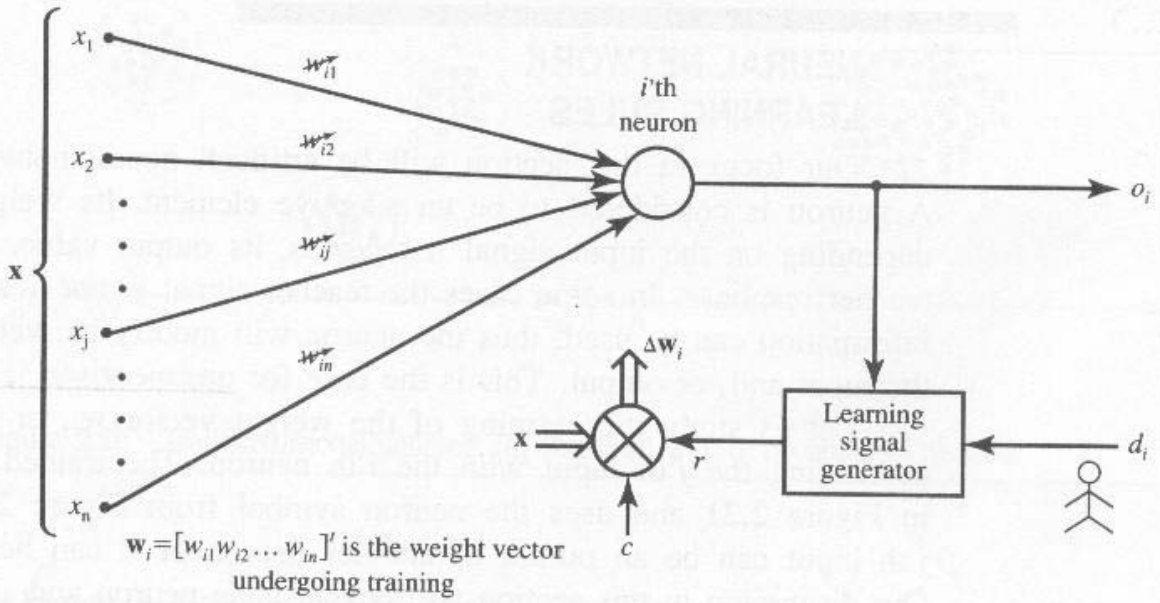


Figure 2.21 Illustration for weight learning rules (d_i provided only for supervised learning mode).

Hebbian Learning

Hebb, 1949

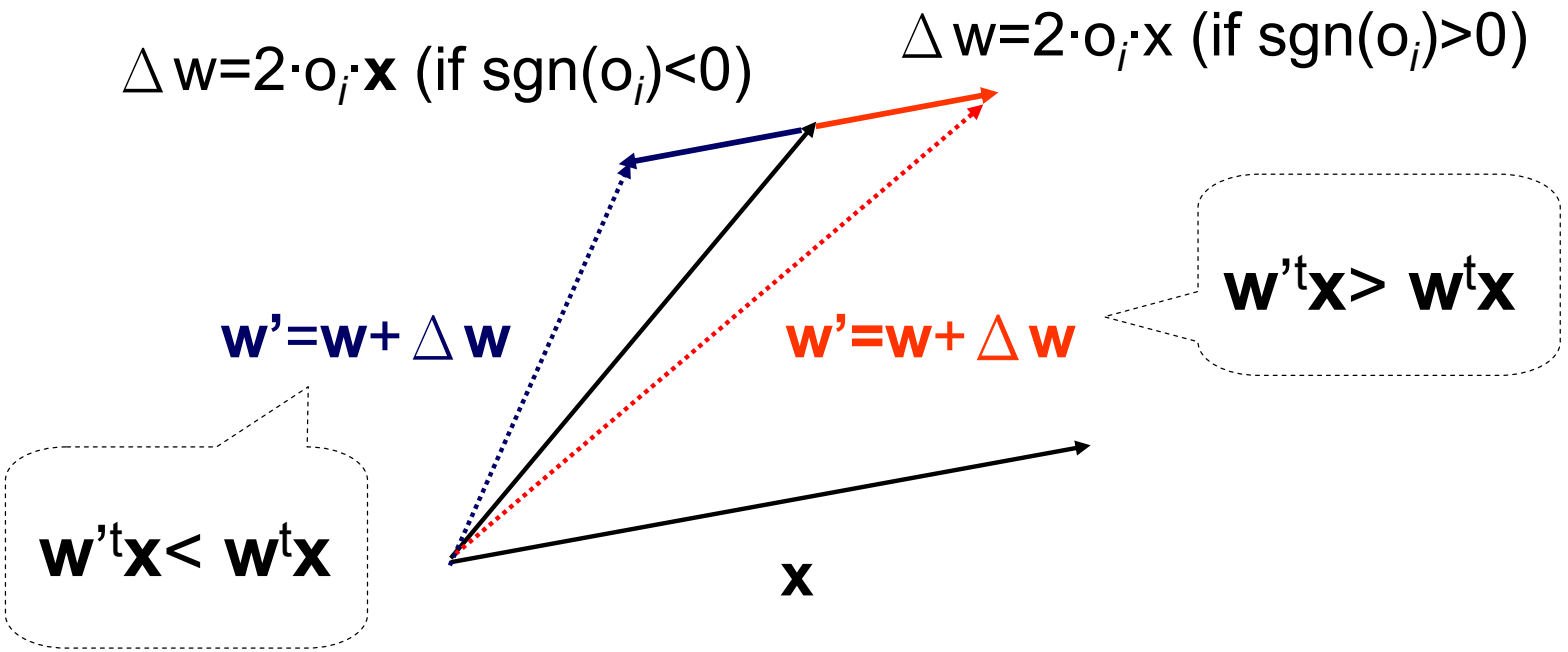
- A purely feedforward, unsupervised learning
- The learning signal is equal to the neuron's output

$$r = o_i = f\left(\mathbf{w}_i^t \mathbf{x}\right)$$

$$\Delta \mathbf{w}_i = c \cdot o_i \cdot \mathbf{x} = cf\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot \mathbf{x} \quad \textbf{or } \Delta w_{ij} = c \cdot f\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot x_j$$

- The weight initialization at small random values around $\mathbf{w}_i$=0 prior to learning
- If the crossproduct of output and input (or correlation) is positive, it results in an increase of the weight, otherwise the weight decreases

Hebbian Learning (cont.)



Hebbian Learning (cont.)

- "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently take places in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency, as one of the cell firing B, is increased" (Hebb, 1949)

當神經元 A 的軸突與神經元 B 之距離，近到足以激發它的地步時，
若神經元 A 重複地或持續地扮演激發神經元 B 的角色，則某種
增長現象或新陳代謝的改變，會發生在其中之一或兩個神經元的細胞上，
以至於神經元 A 能否激發神經元 B 的有效性會被提高。

The weight connecting to neuron B

參考自蘇木村教授著作

Perceptron Learning Rule

- Supervised learning, only applicable for binary neuron response (e.g. [-1,1])
- The learning signal is equal to:

$$r = d_i - o_i$$

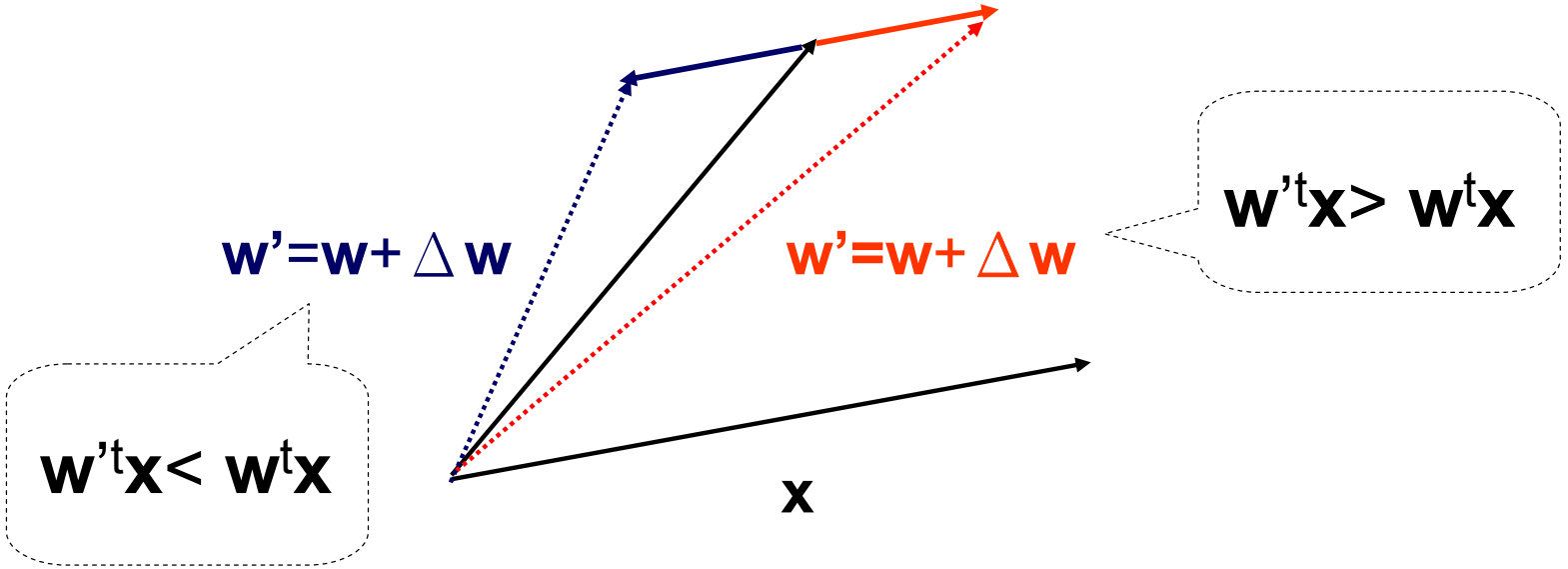
$$\Delta \mathbf{w}_i = c\left[d_i - \mathrm{sgn}\left(\mathbf{w}_i^t \mathbf{x}\right)\right] \cdot \mathbf{x}, \quad \text{if } o_i = f\left(\mathbf{w}_i^t \mathbf{x}\right) = \mathrm{sgn}\left(\mathbf{w}_i^t \mathbf{x}\right)$$

$$\Delta \mathbf{w}_i = \pm 2c\mathbf{x}, \text{ or } \Delta \mathbf{w}_i = 0 \text{ when } d_i = o_i$$

  - E.g., in classification task, the weight is adapted only when classification error occurred
- The weight initialization at any value

Perceptron Learning Rule (cont.)

$\triangle \mathbf{w}=-2\cdot c\cdot \mathbf{x}$ **(**if $\text{sgn}(d_i)<\text{sgn}(\mathbf{w}^t\mathbf{x})$**)** $\triangle \mathbf{w}=2\cdot c\cdot \mathbf{x}$ **(**if $\text{sgn}(d_i)>\text{sgn}(\mathbf{w}^t\mathbf{x})$**)**



Perceptron Learning Rule (cont.)

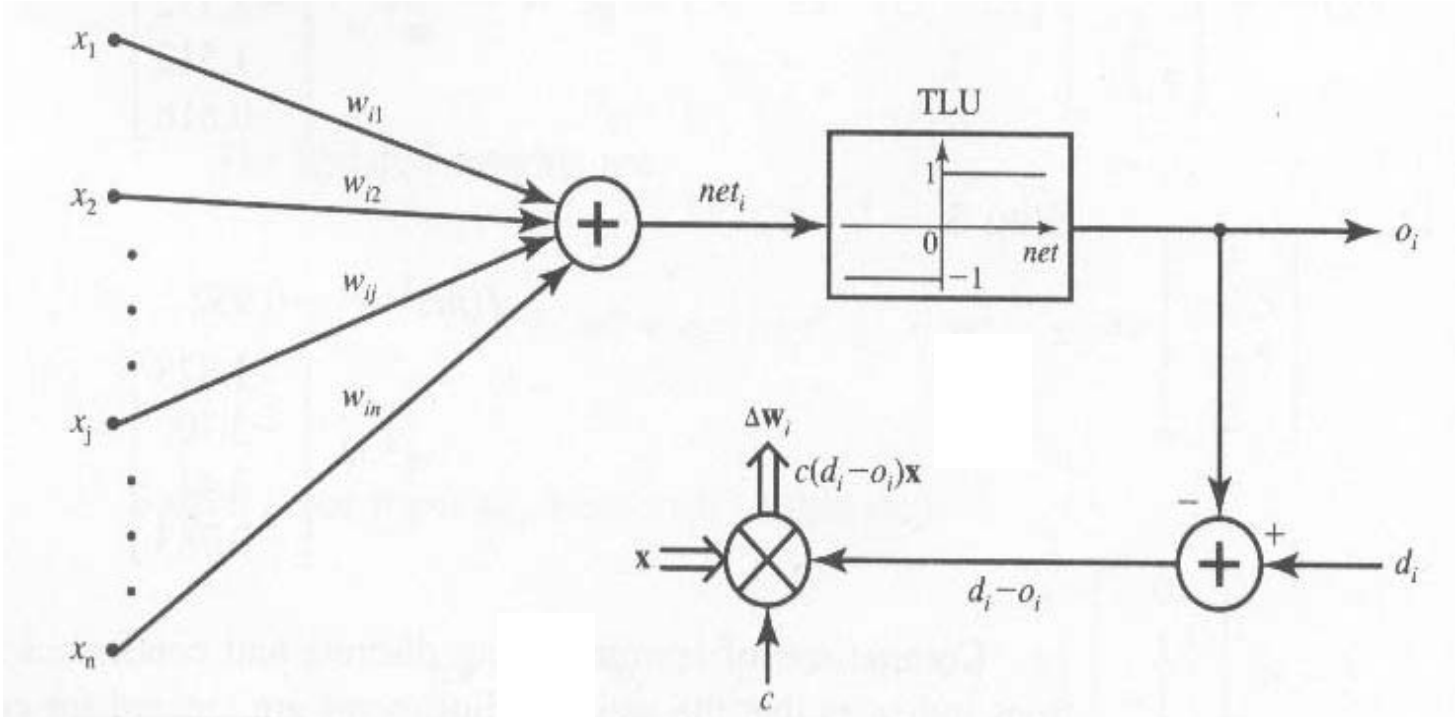


Figure 2.23 Perceptron learning rule.

Delta Learning

- Supervised learning, only applicable for continuous activation function
- The learning signal *r* is called delta and defined as:

$$r = \left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right)\right] \cdot f'\left(\mathbf{w}_i^t \mathbf{x}\right)$$

  - Derived by calculating gradient vector with respect to $\mathbf{w}_i$ of the squared error

$$E = \frac{1}{2}\left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right)\right]^2$$

$$\nabla E = -\left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right)\right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot \mathbf{x}$$

$$\frac{\partial E}{\partial w_{ij}} = -\left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right)\right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) x_j$$

$$\Delta \mathbf{w}_i = -\eta \nabla E = \eta\left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right)\right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot \mathbf{x}$$

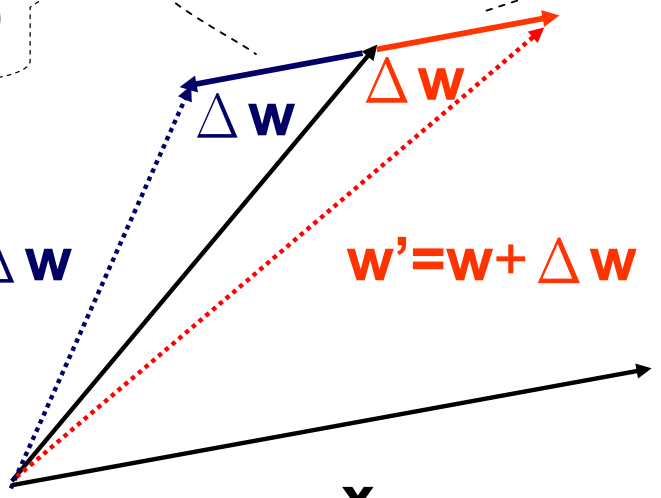
$$\Delta w_{ij} = \eta\left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right)\right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) x_j$$

Delta Learning (cont.)

$$\Delta \mathbf{w}_i = -\eta \nabla E = \eta \left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right) \right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot \mathbf{x}$$

$\eta \left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right) \right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) < 0$

$\eta \left[d_i - f\left(\mathbf{w}_i^t \mathbf{x}\right) \right] f'\left(\mathbf{w}_i^t \mathbf{x}\right) > 0$



w'^{t}x< w^{t}x

w'^{t}x> w^{t}x

Delta Learning (cont.)

- The weight initialization at any value
- Also called continuous perceptron training rule

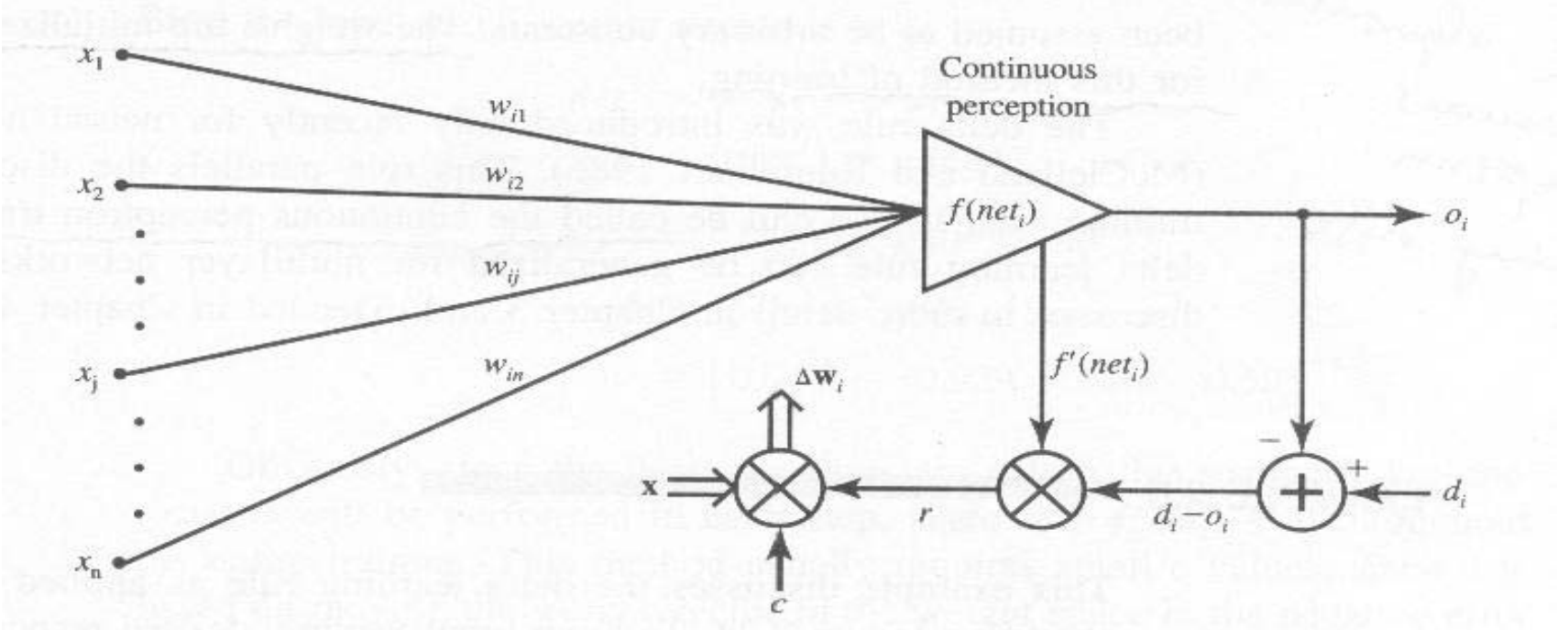


Figure 2.24 Delta learning rule.

Widrow-Hoff Learning Rule Widrow, 1962

- Supervised learning, independent of the activation function of the neuron
- Minimize the squared error between the desired output value and the neuron active value
  - Sometimes called LMS (Least Mean Square) learning rule
- The learning signal *r* is

$$r = d_i - \mathbf{w}_i^t \mathbf{x}$$

$$\Delta \mathbf{w}_i = c\left[d_i - \mathbf{w}_i^t \mathbf{x}\right]\mathbf{x}, \quad \Delta w_{ij} = c\left[d_i - \mathbf{w}_i^t \mathbf{x}\right]x_j$$

- Considered a special case of the delta learning rule when $f\left(\mathbf{w}_i^t \mathbf{x}\right) = \mathbf{w}_i^t \mathbf{x}$

Correlation Learning Rule

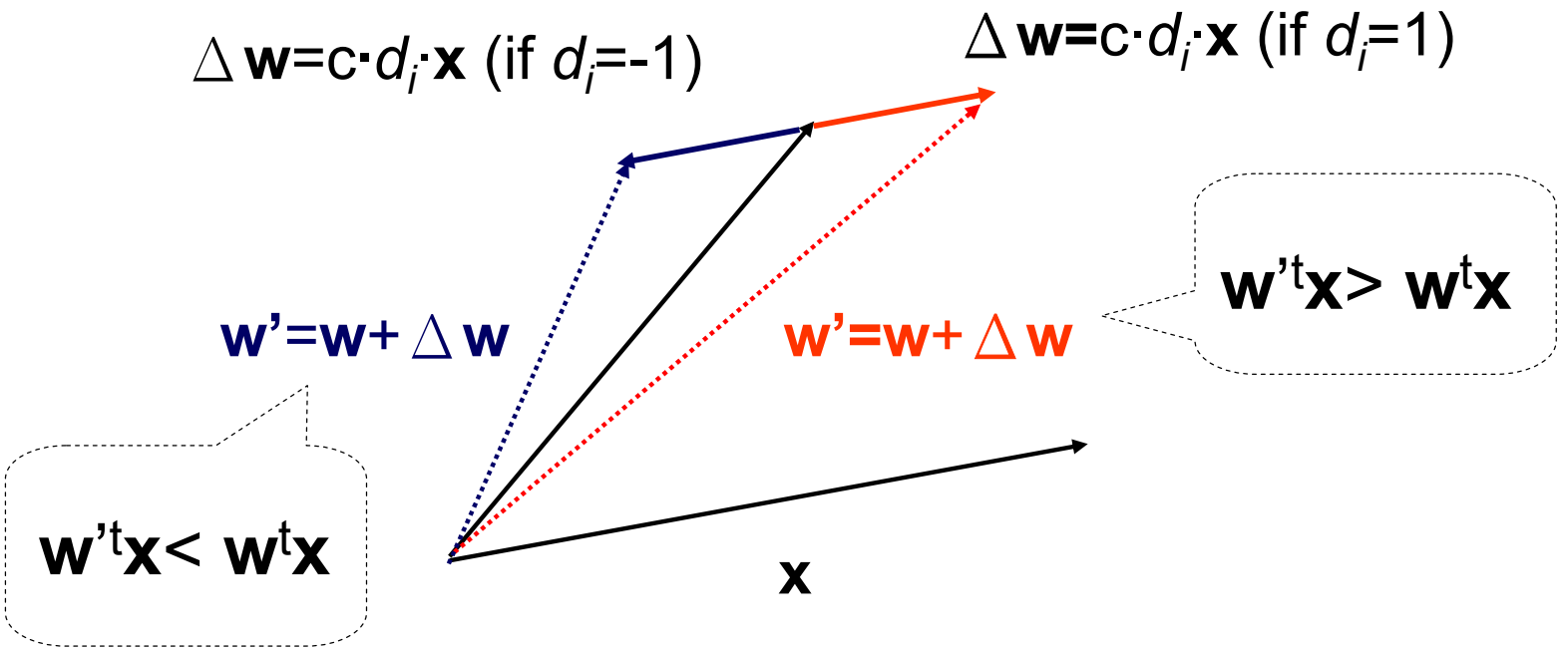
- Supervised learning, applicable for recording data in memory networks with binary response neurons
- The learning signal *r* is simply equal to the desired output d_i

$$r = d_i$$

$$\Delta \mathbf{w}_i = c \cdot d_i \cdot \mathbf{x}, \quad \Delta w_{ij} = c \cdot d_i \cdot x_j$$

- A special case of the Hebbian learning rule with a binary activation function and for $o_i = d_i$
- The weight initialization at small random values around $\mathbf{w}_i = 0$ prior to learning

Correlation Learning Rule (cont.)



Winner-Take-All Learning

- Supervised learning, and applicable for an ensemble of neurons (e.g. a layer of *p* neurons), not for a single neuron
- Adapt the neuron *m* which has the maximum response due to input **x**

$$w_m^t x = \max_{i=1,..,p} \left(w_i^t x \right)$$

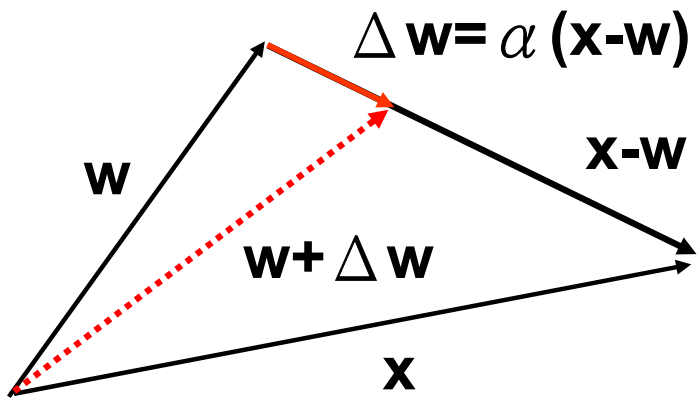
$$\Delta \mathbf{w}_i = \begin{cases} \alpha \left(\mathbf{x} - \mathbf{w}_i \right) & \text{if } i = m \\ \mathbf{0} & \text{if } i \neq m \end{cases}$$

Finding the weight vector closet to the input **x**

- Typically, it is used for learning the statistical properties of input patterns

Winner-Take-All Learning (cont.)

- Weights are typically initializing at random values and their lengths are normalized during learning
- The winner neighborhood is sometimes extended to beyond the single neuron winner to include the neighboring neurons



Summary of Learning Rules

TABLE 2.1
Summary of learning rules and their properties.

Learning rule	Single weight adjustment Δw_{ij}	Initial weights	Learning	Neuron characteristics	Neuron / Layer
Hebbian	$co_i x_j$ $j = 1, 2, \ldots, n$	0	U	Any	Neuron
Perceptron	$c\,[d_i - \text{sgn}\,(\mathbf{w}_i^t \mathbf{x})]\,x_j$ $j = 1, 2, \ldots, n$	Any	S	Binary bipolar, or Binary unipolar*	Neuron
Delta	$c(d_i - o_i) f'(net_i) x_j$ $j = 1, 2, \ldots, n$	Any	S	Continuous	Neuron
Widrow-Hoff	$c(d_i - \mathbf{w}_i^t \mathbf{x}) x_j$ $j = 1, 2, \ldots, n$	Any	S	Any	Neuron
Correlation	$cd_i x_j$ $j = 1, 2, \ldots, n$	0	S	Any	Neuron
Winner-take-all	$\Delta w_{mj} = \alpha(x_j - w_{mj})$ m-winning neuron number $j = 1, 2, \ldots, n$	Random Normalized	U	Continuous	Layer of p neurons

c, α, β are positive learning constants
S — supervised learning, U — unsupervised learning
* — Δw_{ij} not shown