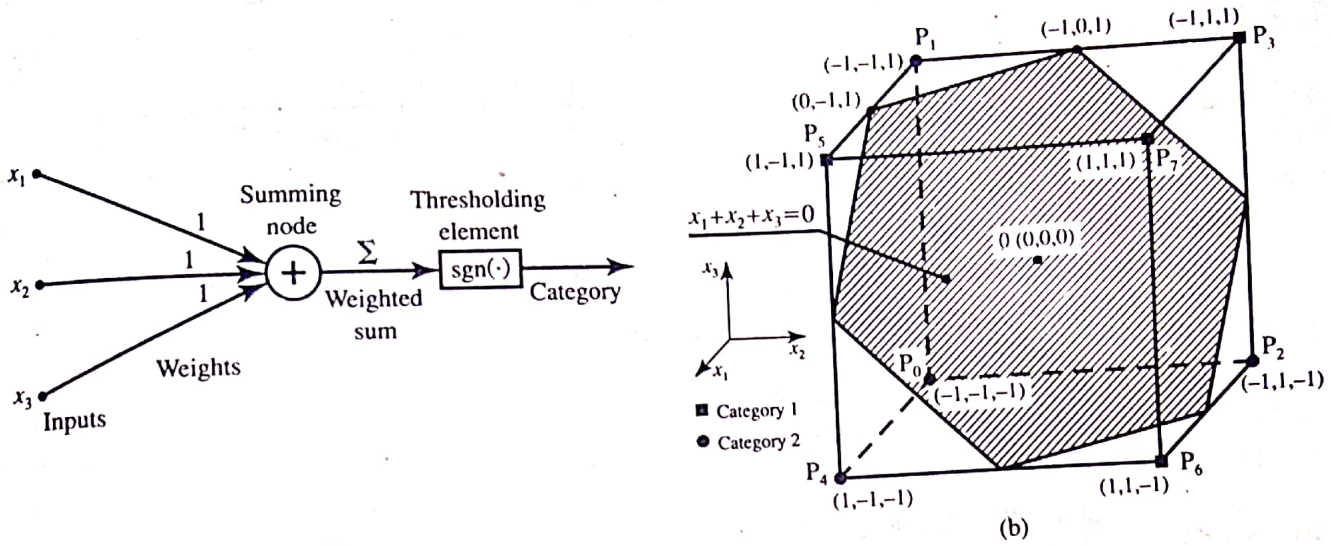


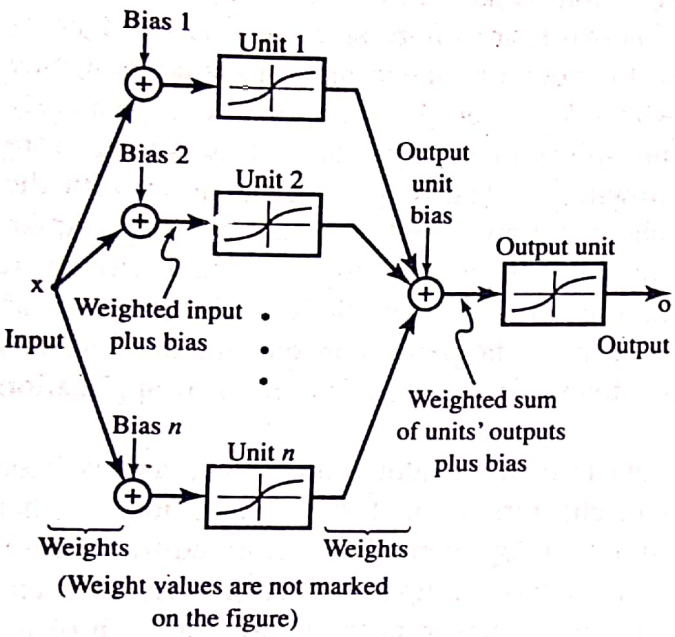
Figure 1.1 Partitioning of the set of cube vertices: (a) single-unit diagram and (b) implemented partitioning.

partitioning plane is $x_1 + x_2 + x_3 = 0$. All points above it are mapped into $+1$ and all below it are mapped into -1. Although this observation is not crucial for our understanding of classifier design, insight into the geometry of mapping will prove to be very valuable for more complex cases.

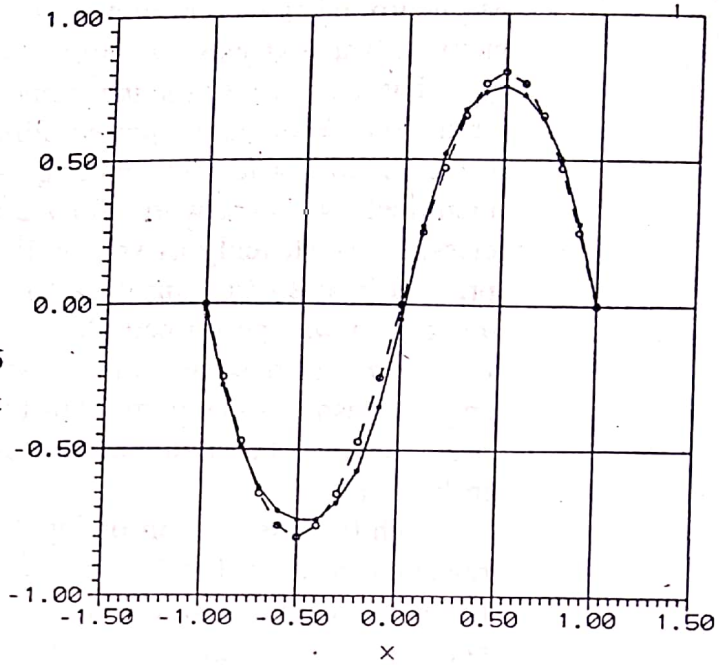
Design of neural network classifiers becomes far more involved and intriguing when requirements for membership in categories become complicated. No single-unit classifier exists, for example, that would implement assignment of P_2, P_3, and P_5 into a single category, and of the remaining five points of the set into the other category. The details of the classification discussion, however, will be postponed until Chapter 3, since they require more formal coverage.

As stated, the unit from Figure 1.1(a) maps the entire three-dimensional space into just two points, 1 and -1. A question arises as to whether a unit with a "squashed" sgn function rather than a regular sgn function could prove more advantageous. Assuming that the "squashed" sgn function has the shape as in Figure 1.2, notice that now the outputs take values in the range $(-1, 1)$ and are generally more discernible than in the previous case. Using units with continuous characteristics offers tremendous opportunities for new tasks that can be performed by neural networks. Specifically, the fine granularity of output provides more information than the binary ± 1 output of the thresholding element.

An important new feature of networks with units having continuous characteristics is that they can be trained independent of their architecture. This has not



(a)



(b)

Figure 1.3 Neural network with continuous units as function approximator: (a) block diagram and (b) example approximation of Equation (1.2).

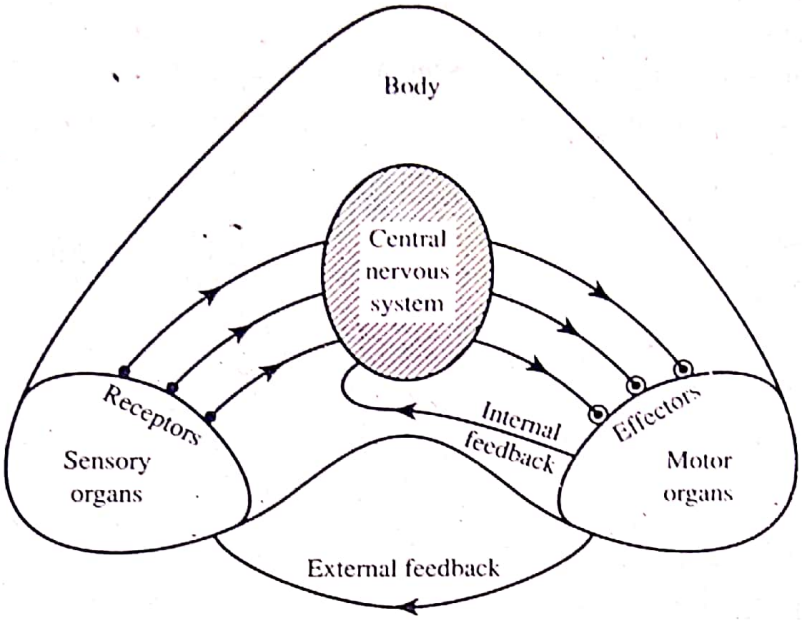


Figure 2.1 Information flow in nervous system.

A lucid, although rather approximate idea, about the information links in the nervous system is shown in Figure 2.1. As we can see from the figure, the information is processed, evaluated, and compared with the stored information in the central nervous system. When necessary, commands are generated there and transmitted to the motor organs. Notice that motor organs are monitored in the central nervous system by feedback links that verify their action. Both internal and external feedback control the implementation of commands. As can be seen, the overall nervous system structure has many of the characteristics of a closed-loop control system.

Biological Neuron

The *elementary nerve cell*, called a *neuron*, is the fundamental building block of the biological neural network. Its schematic diagram is shown in Figure 2.2. A typical cell has three major regions: the cell body, which is also called the *soma*, the *axon*, and the *dendrites*. Dendrites form a dendritic tree, which is a very fine bush of thin fibers around the neuron's body. Dendrites receive information from neurons through axons—long fibers that serve as transmission lines. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits into a fine arborization. Each branch of it terminates in a small endbulb almost touching the dendrites of neighboring neurons. The axon-dendrite contact organ is called a *synapse*. The synapse is where the neuron introduces its signal to the neighboring neuron. The signals reaching a synapse and received

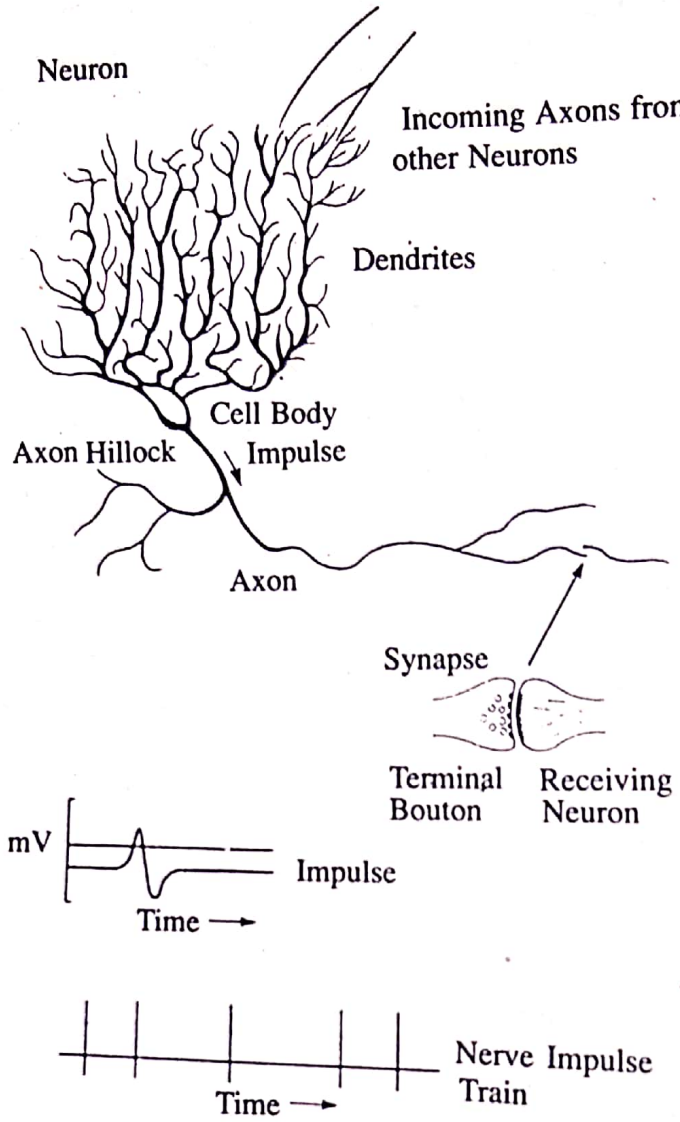


Figure 2.2 Schematic diagram of a neuron and a sample of pulse train. (Adapted from (Dayhoff 1990), © Van Nostrand Reinhold; with permission.)

by dendrites are electrical impulses. The interneuronal transmission is sometimes electrical but is usually effected by the release of chemical transmitters at the synapse. Thus, terminal boutons generate the chemical that affects the receiving neuron. The receiving neuron either generates an impulse to its axon, or produces no response.

The neuron is able to respond to the total of its inputs aggregated within a short time interval called the *period of latent summation*. The neuron's response is generated if the total potential of its membrane reaches a certain level. The membrane can be considered as a shell, which aggregates the magnitude of the incoming signals over some duration. Specifically, the neuron generates a pulse response and sends it to its axon only if the conditions necessary for firing are fulfilled.

The above discussion is extremely simplified when seen from a neurobiological point of view, though it is valuable for gaining insight into the principles of "biological computation." Our computing networks are far simpler than their biological counterparts. Let us examine an artificial neuron model that is of special, historical significance.

McCulloch-Pitts Neuron Model

The first formal definition of a synthetic neuron model based on the highly simplified considerations of the biological model described in the preceding section was formulated by McCulloch and Pitts (1943). The McCulloch-Pitts model of the neuron is shown in Figure 2.3a. The inputs x_i, for $i = 1, 2, \ldots, n$, are 0 or 1, depending on the absence or presence of the input impulse at instant k. The neuron's output signal is denoted as o. The firing rule for this model is defined as follows

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

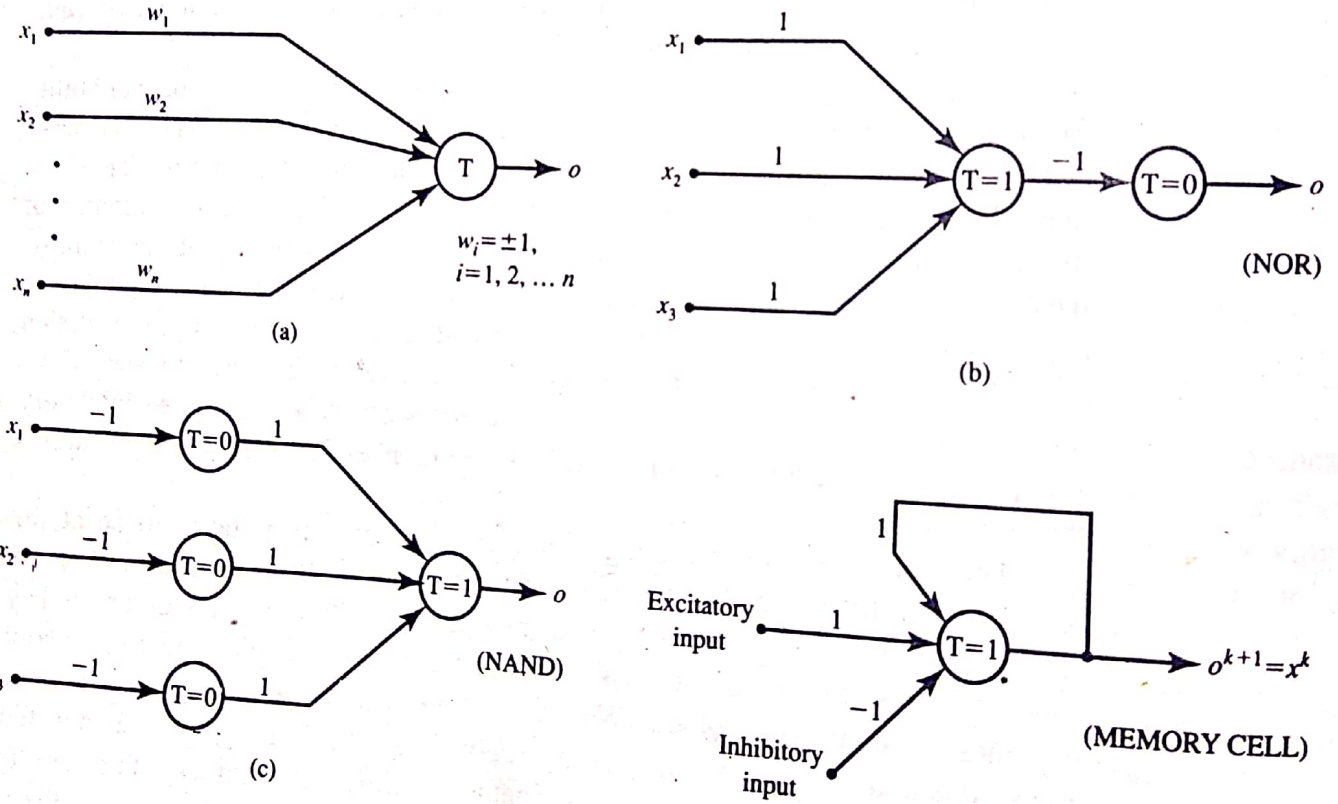


Figure 2.3 McCulloch-Pitts model neuron and elementary logic networks: (a) model diagram, (b) NOR gate, (c) NAND gate, and (d) memory cell.

2.2 MODELS OF ARTIFICIAL NEURAL NETWORKS

Our introductory definition of artificial neural networks as physical cellular networks that are able to acquire, store, and utilize experiential knowledge has been related to the network's capabilities and performance. At this point, knowing the definition of the artificial neural network neuron model, we may benefit from another definition. The neural network can also be defined as *an interconnection of neurons, as defined in (2.1) through (2.4), such that neuron outputs are connected, through weights, to all other neurons including themselves; both lag-free and delay connections are allowed.* As research efforts continue, new and extended definitions may be developed, but this definition is sufficient for the introductory study of artificial neural architectures and algorithms found in this text.

Feedforward Network

Let us consider an elementary *feedforward architecture* of m neurons receiving n inputs as shown in Figure 2.8(a). Its output and input vectors are, respectively

$$\mathbf{o} = [o_1 \quad o_2 \quad \cdots \quad o_m]^t$$
$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^t \tag{2.8}$$

Weight w_{ij} connects the i'th neuron with the j'th input. The double subscript convention used for weights in this book is such that *the first and second subscript denote the index of the destination and source nodes, respectively.* We thus can write the activation value for the i'th neuron as

$$net_i = \sum_{j=1}^{n} w_{ij}x_j, \quad \text{for } i = 1, 2, \ldots, m \tag{2.9}$$

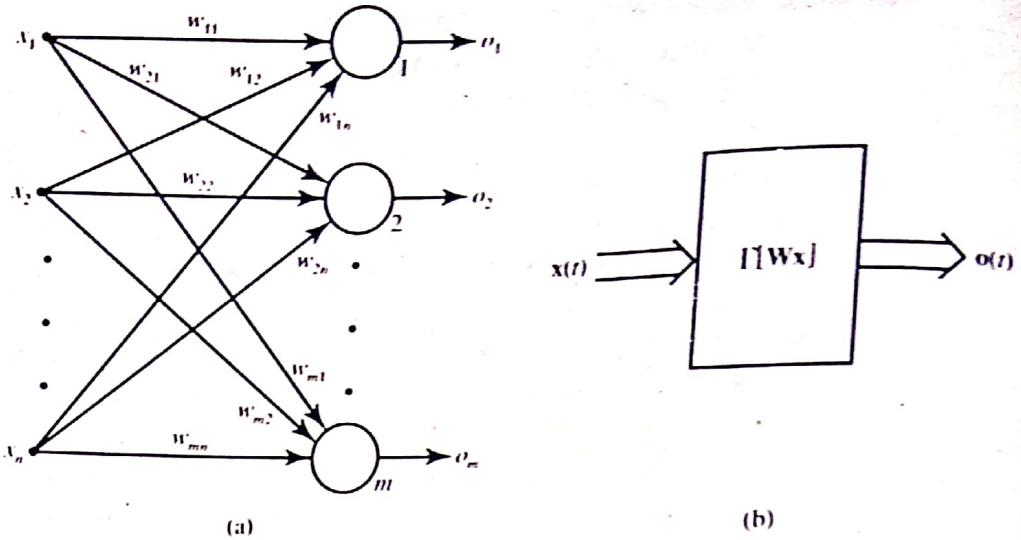
The following nonlinear transformation [Equation (2.10)] involving the activation function $f(net_i)$, for $i = 1, 2, \ldots, m$, completes the processing of $\mathbf{x}$. The transformation, performed by each of the m neurons in the network, is a strongly nonlinear mapping expressed as

$$o_i = f(\mathbf{w}_i^t\mathbf{x}), \quad \text{for } i = 1, 2, \ldots, m \tag{2.10}$$

where weight vector $\mathbf{w}_i$ contains weights leading toward the i'th output node and is defined as follows

$$\mathbf{w}_i \triangleq [w_{i1} \quad w_{i2} \quad \cdots \quad w_{in}]^t \tag{2.11}$$

Introducing the nonlinear matrix operator Γ, the mapping of input space x to



(a) (b)

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

output space o implemented by the network can be expressed as follows

$$\mathbf{o} = \Gamma[\mathbf{W}\mathbf{x}] \tag{2.12a}$$

where $\mathbf{W}$ is the *weight matrix*, also called the *connection matrix*:

$$\mathbf{W} \triangleq \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \tag{2.12b}$$

and

$$\Gamma[\cdot] \triangleq \begin{bmatrix} f(\cdot) & 0 & \cdots & 0 \\ 0 & f(\cdot) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & f(\cdot) \end{bmatrix} \tag{2.12c}$$

Note that the nonlinear activation functions $f(\cdot)$ on the diagonal of the matrix operator Γ operate componentwise on the activation values *net* of each neuron. Each activation value is, in turn, a scalar product of an input with the respective weight vector.

The input and output vectors $\mathbf{x}$ and $\mathbf{o}$ are often called *input* and *output patterns*, respectively. The mapping of an input pattern into an output pattern as shown in (2.12) is of the feedforward and instantaneous type, since it involves no time delay between the input $\mathbf{x}$ and the output $\mathbf{o}$. Thus, we can rewrite (2.12a)

in the explicit form involving time t as

$$\mathbf{o}(t) = \Gamma[\mathbf{W}\mathbf{x}(t)] \tag{2.13}$$

Figure 2.8(b) shows the block diagram of the feedforward network. As can be seen, the generic feedforward network is characterized by the lack of feedback. This type of network can be connected in cascade to create a multilayer network. In such a network, the output of a layer is the input to the following layer. Even though the feedforward network has no explicit feedback connection when $\mathbf{x}(t)$ is mapped into $\mathbf{o}(t)$, the output values are often compared with the "teacher's" information, which provides the desired output value, and also an error signal can be employed for adapting the network's weights. We will postpone more detailed coverage of feedforward network learning through adaptation to Sections 2.4 and 2.5.

EXAMPLE 2.1

This example presents the analysis of a two-layer feedforward network using neurons having the bipolar binary activation function given in (2.3b). The network to be analyzed is shown in Figure 2.9(a). Our purpose is to find output o_5 for a given network and input pattern. Each of the network layers is described by the formula (2.12a)

$$\mathbf{o} = \Gamma[\mathbf{W}\mathbf{x}] \tag{2.14}$$

By inspection of the network diagram, we obtain the output, input vectors, and the weight matrix for the first layer, respectively, as

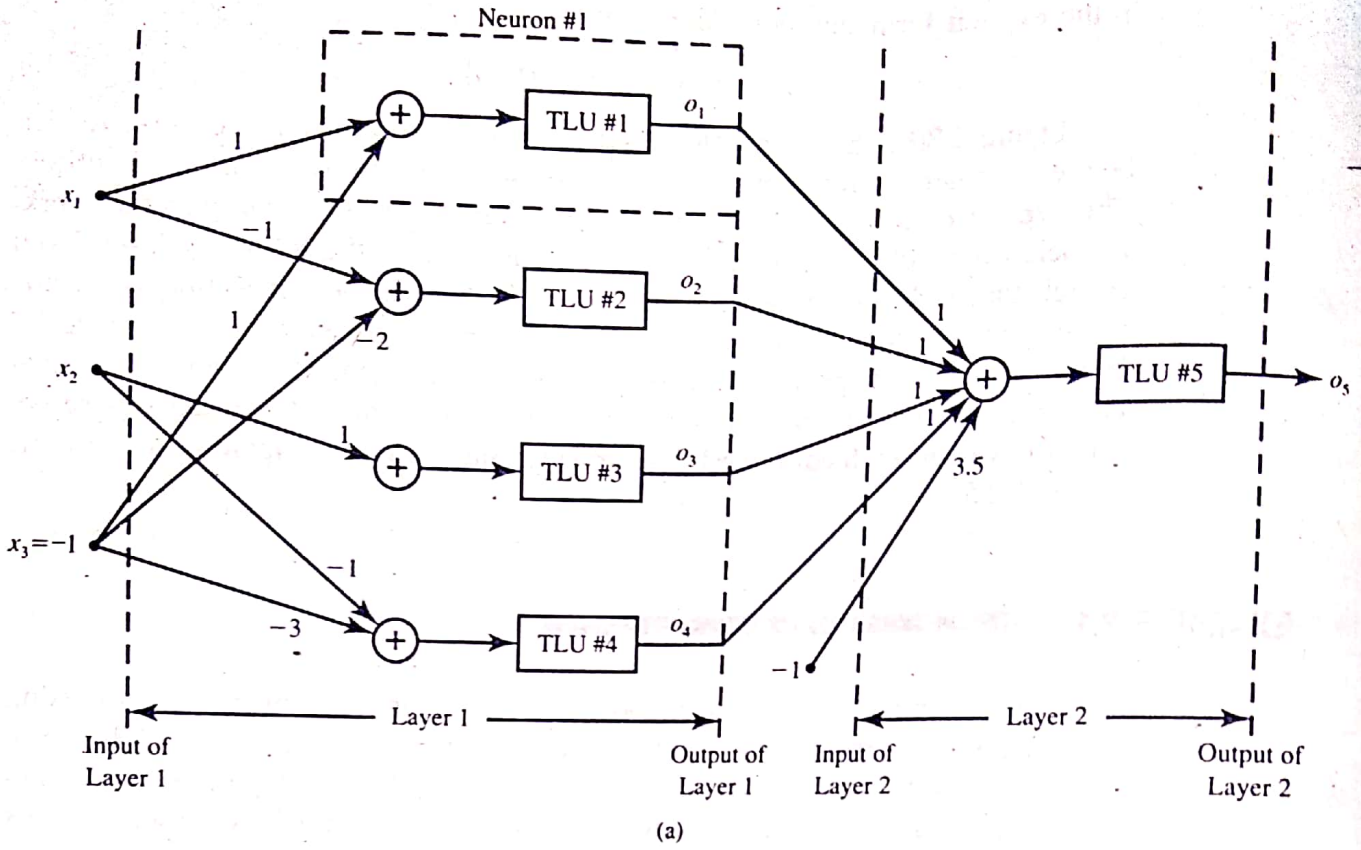
$$\mathbf{o} = [o_1 \quad o_2 \quad o_3 \quad o_4]^t$$
$$\mathbf{x} = [x_1 \quad x_2 \quad -1]^t$$
$$\mathbf{W}_1 = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & -1 & -3 \end{bmatrix}$$

Similarly, for the second layer we can write

$$\mathbf{o} = [o_5]$$
$$\mathbf{x} = [o_1 \quad o_2 \quad o_3 \quad o_4 \quad -1]^t$$
$$\mathbf{W}_2 = [1 \quad 1 \quad 1 \quad 1 \quad 3.5]$$

The response of the first layer can be computed for bipolar binary activation functions as below

$$\mathbf{o} = [\operatorname{sgn}(x_1 - 1) \quad \operatorname{sgn}(-x_1 + 2) \quad \operatorname{sgn}(x_2) \quad \operatorname{sgn}(-x_2 + 3)]^t$$



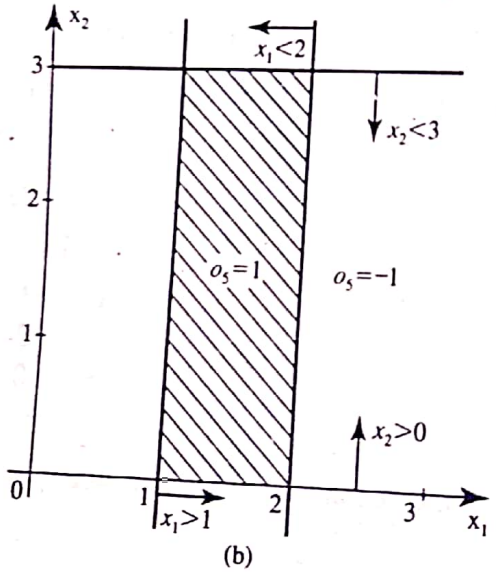
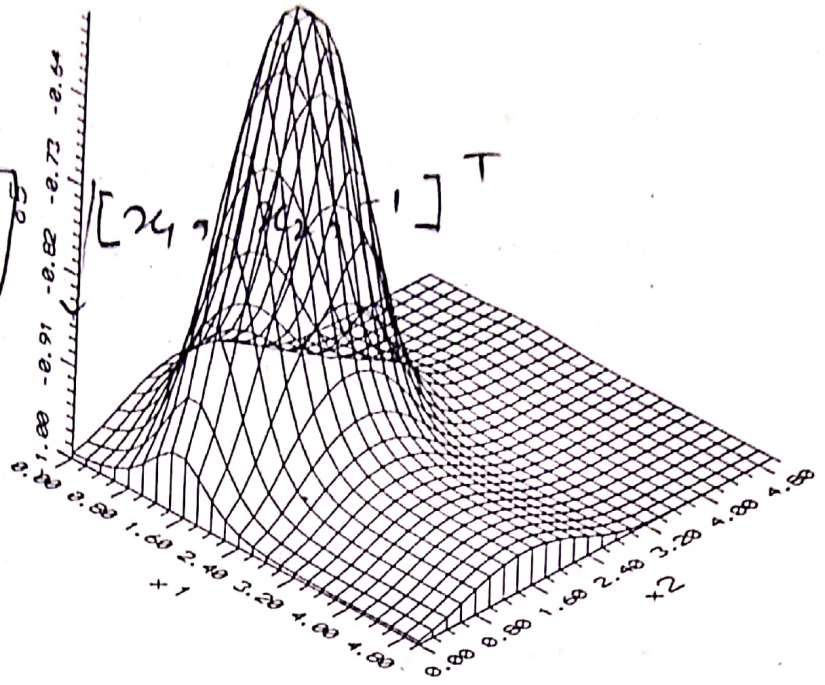


Figure 2.9a,b Example of two-layer feedforward network: (a) diagram and (b) two-dimensional space mapping (discrete activation function).

Let us discuss the mapping performed by the first layer. Note that each of the neurons 1 through 4 divides the plane x_1, x_2 into two half-planes. The half-planes where the neurons' responses are positive (+1) have been marked on Figure 2.9(b) with arrows pointing toward the positive response



(c)

Figure 2.9c Example of two-layer feedforward network *(continued)*: (c) two-dimensional space mapping (continuous activation function, $\lambda = 2.5$).

half-plane. The response of the second layer can be easily obtained as

$$o_5 = \text{sgn}\,(o_1 + o_2 + o_3 + o_4 - 3.5)$$

Note that the fifth neuron responds $+1$ if and only if $o_1 = o_2 = o_3 = o_4 = 1$. It therefore selects the intersection of four half-planes produced by the first layer and designated by the arrows. Figure 2.9(b) shows that the network maps the shaded region of plane x_1, x_2 into $o_5 = 1$, and it maps its complement into $o_5 = -1$. In summary, the network of Figure2.9(a) provides mapping of the entire x_1, x_2 plane into one of the two points ± 1 on the real number axis.

Let us look at the mapping provided by the same architecture but with neurons having sigmoidal characteristics. For the continuous bipolar activation function given in (2.3a), we obtain for the first layer

$$\mathbf{o} = \begin{bmatrix} \dfrac{2}{1 + \exp(1 - x_1)\lambda} - 1 \\ \dfrac{2}{1 + \exp(x_1 - 2)\lambda} - 1 \\ \dfrac{2}{1 + \exp(-x_2)\lambda} - 1 \\ \dfrac{2}{1 + \exp(x_2 - 3)\lambda} - 1 \end{bmatrix}$$

and for the second layer

$$o_5 = \frac{2}{1 + \exp(3.5 - o_1 - o_2 - o_3 - o_4)\lambda} - 1$$

The network with neurons having sigmoidal activation functions performs mapping as shown in Figure 2.9(c). The figure reveals that although some similarity exists with the discrete neuron case, the mapping is much more complex. The example shows how the two-dimensional space has been mapped into the segment of one-dimensional space. In summary, the network of Figure 2.9(a) with bipolar continuous neurons provides mapping of the entire x_1, x_2 plane into the interval $(-1, 1)$ on the real number axis. Similar mapping was shown earlier in Figure 1.3. In fact, neural networks with as few as two layers are capable of universal approximation from one finite dimensional space to another.

Feedback Network

A feedback network can be obtained from the feedforward network shown in Figure 2.8(a) by connecting the neurons' outputs to their inputs. The result is depicted in Figure 2.10(a). The essence of closing the feedback loop is to enable

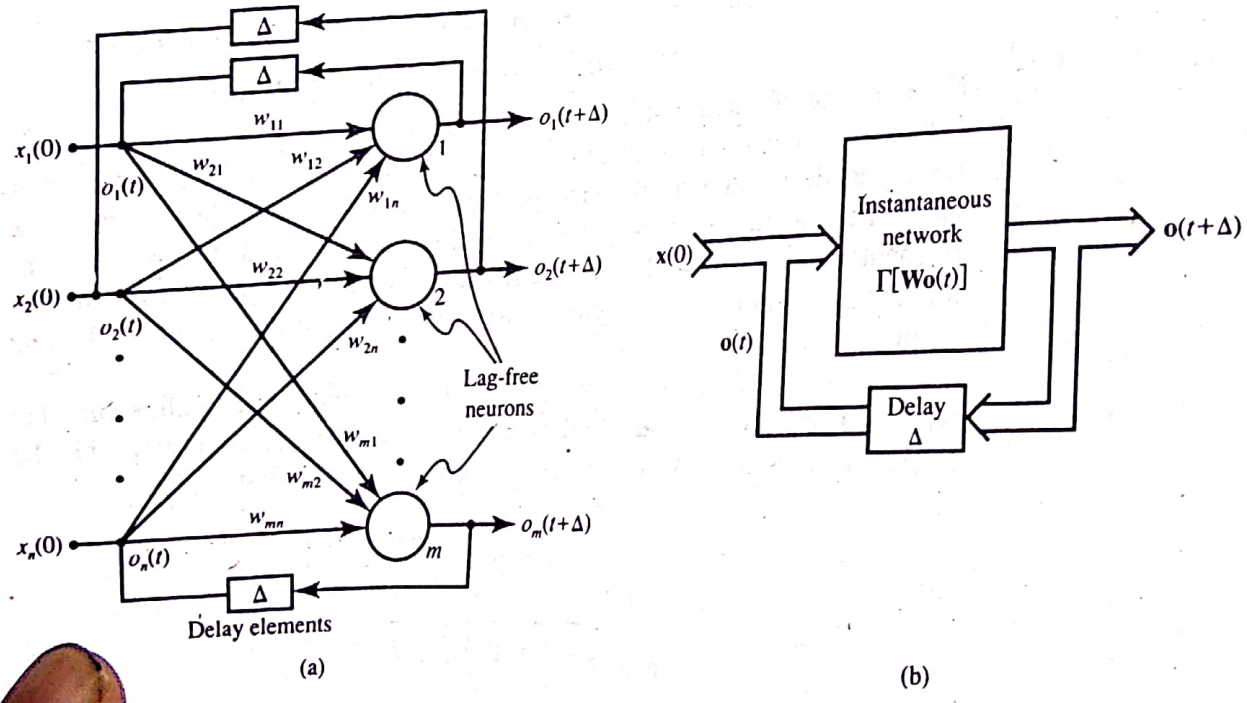


Figure 2.10 Single-layer discrete-time feedback network: (a) interconnection scheme and (b) block diagram.

2.5 NEURAL NETWORK LEARNING RULES

Our focus in this section will be artificial neural network learning rules. A neuron is considered to be an adaptive element. Its weights are modifiable depending on the input signal it receives, its output value, and the associated teacher response. In some cases the teacher signal is not available and no error information can be used, thus the neuron will modify its weights based only on the input and/or output. This is the case for unsupervised learning.

Let us study the learning of the weight vector $\mathbf{w}_i$, or its components w_{ij} connecting the j'th input with the i'th neuron. The trained network is shown in Figure 2.21 and uses the neuron symbol from Figure 2.4. In general, the j'th input can be an output of another neuron or it can be an external input. Our discussion in this section will cover single-neuron and single-layer network supervised learning and simple cases of unsupervised learning. Under different learning rules, the form of the neuron's activation function may be different. Note that the threshold parameter may be included in learning as one of the weights. This would require fixing one of the inputs, say x_n. We will assume here that x_n, if fixed, takes the value of -1.

The following *general learning rule* is adopted in neural network studies (Amari 1990): *The weight vector* $\mathbf{w}_i = [w_{i1} \quad w_{i2} \quad \cdots \quad w_{in}]^t$ *increases in proportion to the product of input* $\mathbf{x}$ *and learning signal* r. The learning signal r is in general a function of $\mathbf{w}_i, \mathbf{x}$, and sometimes of the teacher's signal d_i. We thus have for the network shown in Figure 2.21:

$$r = r(\mathbf{w}_i, \mathbf{x}, d_i) \tag{2.27}$$

The increment of the weight vector $\mathbf{w}_i$ produced by the learning step at time t according to the general learning rule is

$$\Delta \mathbf{w}_i(t) = cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right]\mathbf{x}(t) \tag{2.28}$$

where c is a positive number called the *learning constant* that determines the rate of learning. The weight vector adapted at time t becomes at the next instant, or learning step,

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right]\mathbf{x}(t) \tag{2.29a}$$

The superscript convention will be used in this text to index the discrete-time training steps as in Eq. (2.29a). For the k'th step we thus have from (2.29a) using this convention

$$\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + cr(\mathbf{w}_i^k, \mathbf{x}^k, d_i^k)\mathbf{x}^k \tag{2.29b}$$

The learning in (2.29) assumes the form of a sequence of discrete-time weight modifications. Continuous-time learning can be expressed as

$$\frac{d\mathbf{w}_i(t)}{dt} = cr\mathbf{x}(t) \tag{2.30}$$

4.2 DELTA LEARNING RULE FOR MULTIPERCEPTRON LAYER

The following discussion is focused on a training algorithm applied to the multilayer feedforward networks. The algorithm is called the *error back-propagation training* algorithm. As mentioned earlier in this chapter, the algorithm has reawakened the scientific and engineering community to the modeling of many quantitative phenomena using neural networks.

The back-propagation training algorithm allows experiential acquisition of input/output mapping knowledge within multilayer networks. Similarly, as in simple cases of the delta learning rule training studied before, input patterns are submitted during the back-propagation training sequentially. If a pattern is submitted and its classification or association is determined to be erroneous, the synaptic weights as well as the thresholds are adjusted so that the current least mean-square classification error is reduced. The input/output mapping, comparison of target and actual values, and adjustment, if needed, continue until all mapping examples from the training set are learned within an acceptable overall error. Usually, mapping error is cumulative and computed over the full training set.

During the association or classification phase, the trained neural network itself operates in a feedforward manner. However, the weight adjustments enforced by the learning rules propagate exactly backward from the output layer through the so-called "hidden layers" toward the input layer. To formulate the learning algorithm, the simple continuous perceptron network involving K neurons will be revisited first. Let us take another look at the network shown in Figure 3.23. It is redrawn again in Figure 4.6 with a slightly different connection form and notation, but both networks are identical.

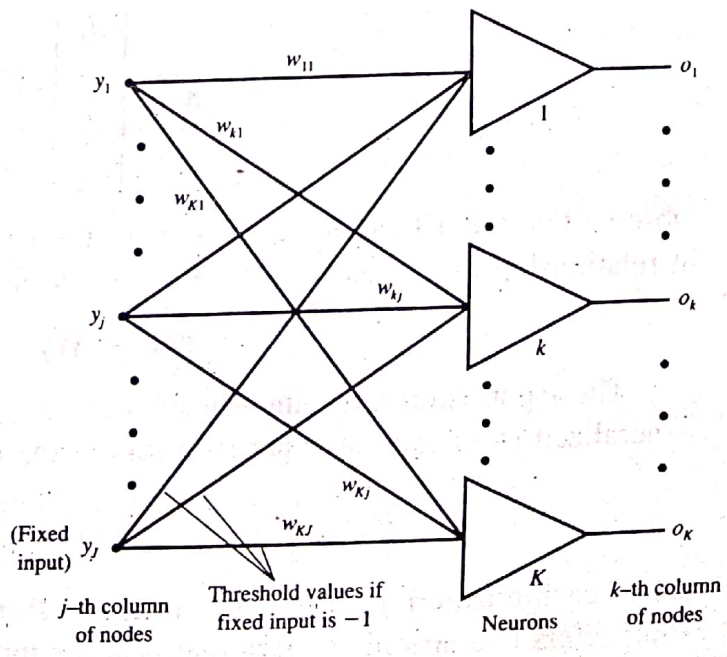
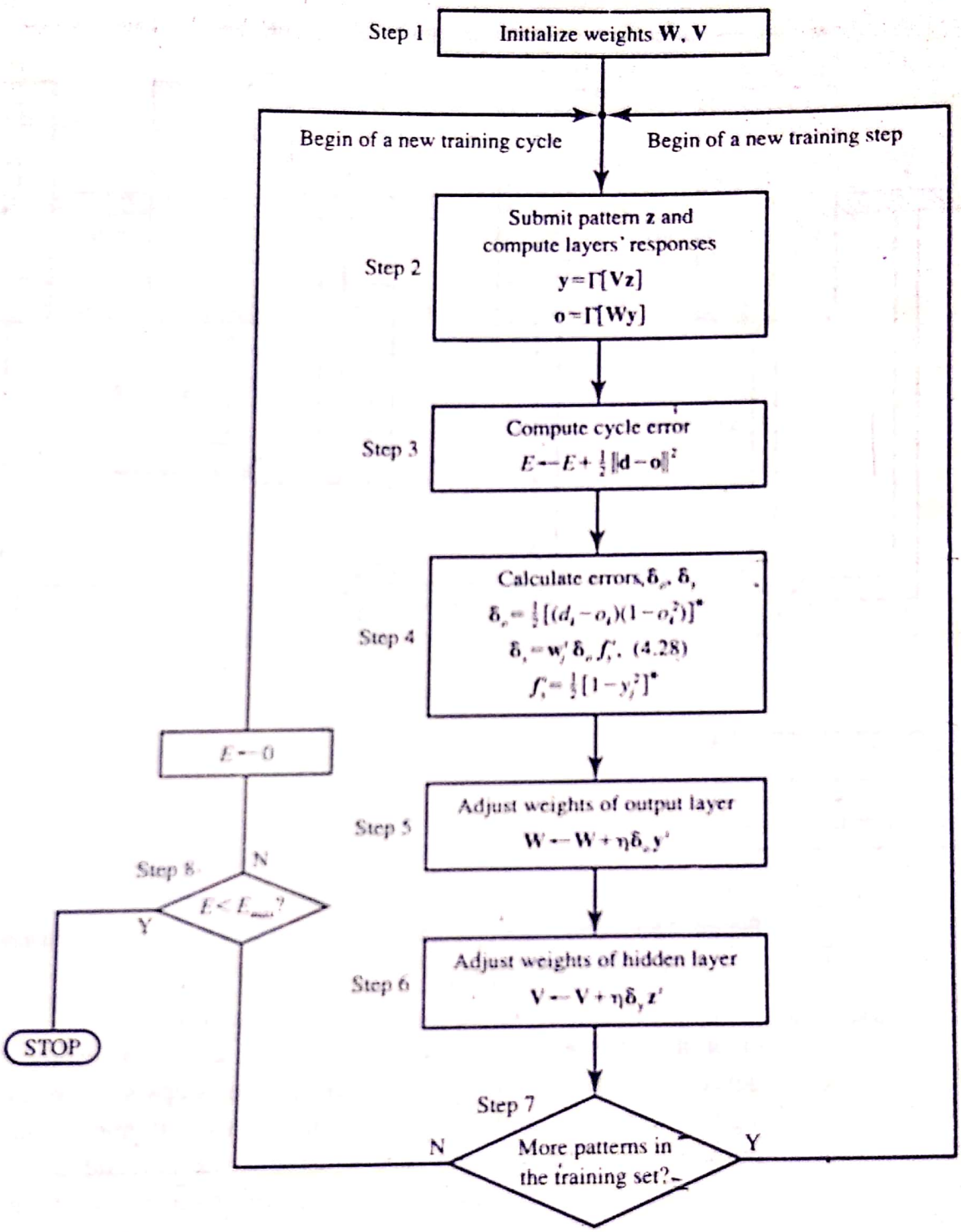


Figure 4.6 Single layer network with continuous perceptrons.



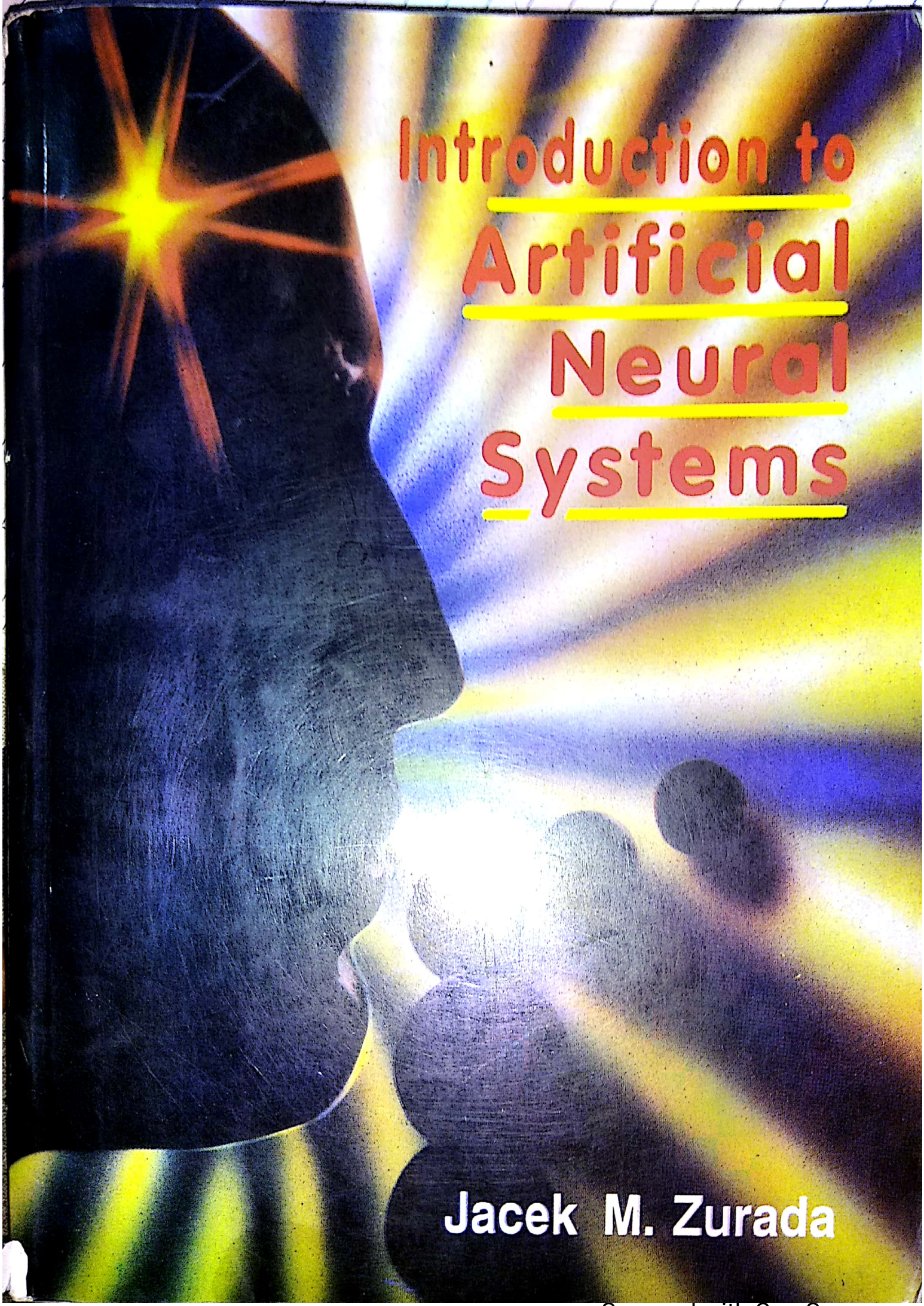
*If $f(net)$ given by (2.4a) is used in Step 2, then in Step 4 use

$$\delta_o=[(d_k-o_k)(1-o_k)o_k],\quad f_y'=[(1-y_j)y_j]$$

(a)

Figure 4.8a Error back-propagation training (EBPT algorithm): (a) algorithm flowchart.

Introduction to Artificial Neural Systems



Jacek M. Zurada

Date

Activation function :-

(i) Binary function

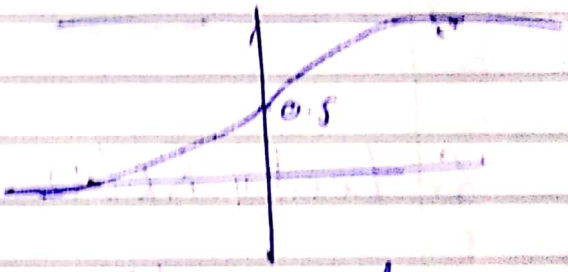
$$f(x) = \begin{cases} 1 & x >= 0 \\ 0 & x < 0 \end{cases}$$

(ii) Linear function

$$f(x) = ax$$

(iii) Sigmoid function

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



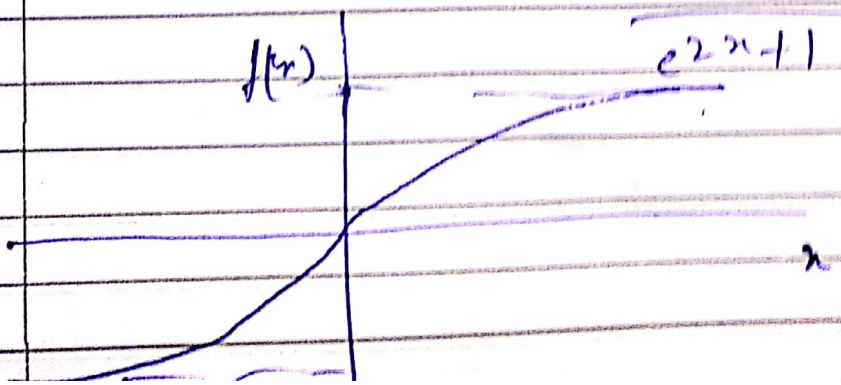
derivative of sigmoid function

$$f_D'(x) = 2(1-x)$$

(iv) tanh

$$f(x) = 2\,\text{sigmoid}(2x) - 1$$

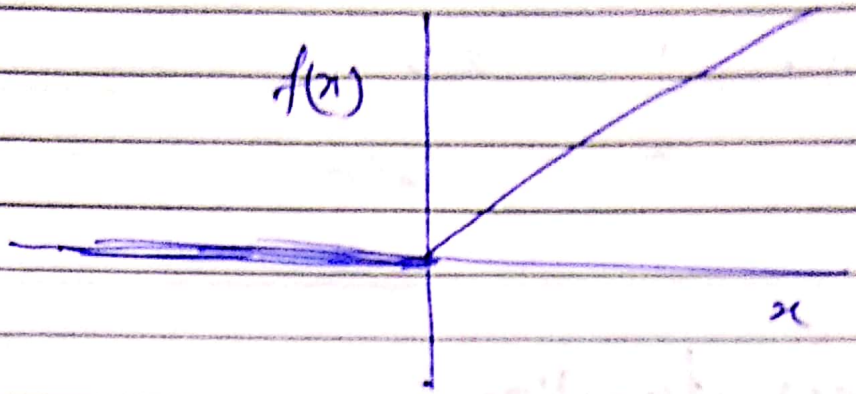
$$= \frac{e^{2x} - 1}{e^{2x} + 1}$$



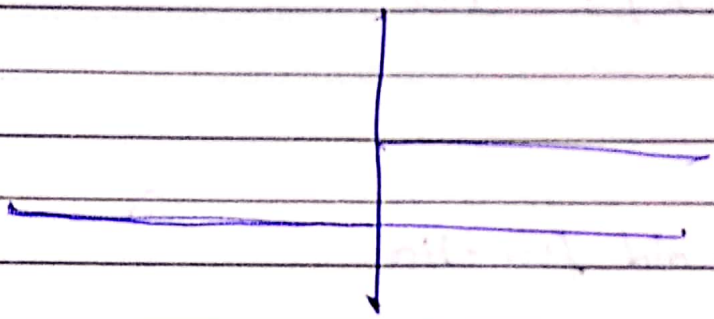
Date

5. ReLU :- Rectified Linear Unit

$f(x) = \max(0, x)$



Derivatives



6. Leaky ReLU

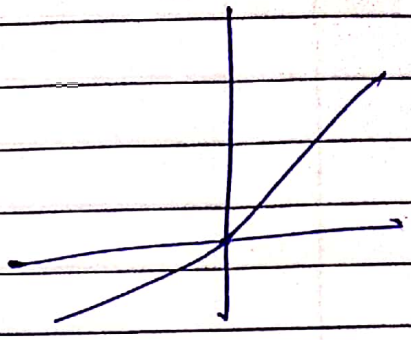
$$f(x) = \begin{cases} 0.01x & x < 0 \\ x & x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} (0.01) & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Date _____

(7) Parameterised ReLU

$$f(x) = \begin{cases} x & x >= 0 \\ ax & x < 0 \end{cases}$$



(8) Exponential Linear Unit

$$f(x) = \begin{cases} x & x \geq 0 \\ a(e^x - 1) & x < 0 \end{cases}$$