Introduction

WHAT IS A NEURAL NETWORK?

Work on artificial neural networks, commonly referred to as "neural networks," has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. The brain is a highly *complex, nonlinear, and parallel computer* (information-processing system). It has the capability to organize its structural constituents, known as *neurons*, so as to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today. Consider, for example, human *vision*, which is an information-processing task. It is the function of the visual system to provide a *representation* of the environment around us and, more important, to supply the information we need to *interact* with the environment. To be specific, the brain routinely accomplishes perceptual recognition tasks (e.g., recognizing a familiar face embedded in an unfamiliar scene) in approximately 100–200 ms, whereas tasks of much lesser complexity take a great deal longer on a powerful computer.

For another example, consider the *sonar* of a bat. Sonar is an active echolocation system. In addition to providing information about how far away a target (e.g., a flying insect) is, bat sonar conveys information about the relative velocity of the target, the size of the target, the size of various features of the target, and the azimuth and elevation of the target. The complex neural computations needed to extract all this information from the target echo occur within a brain the size of a plum. Indeed, an echolocating bat can pursue and capture its target with a facility and success rate that would be the envy of a radar or sonar engineer.

How, then, does a human brain or the brain of a bat do it? At birth, a brain already has considerable structure and the ability to build up its own rules of behavior through what we usually refer to as "experience." Indeed, experience is built up over time, with much of the development (i.e., hardwiring) of the human brain taking place during the first two years from birth, but the development continues well beyond that stage.

A "developing" nervous system is synonymous with a plastic brain: *Plasticity* permits the developing nervous system to *adapt* to its surrounding environment. Just as plasticity appears to be essential to the functioning of neurons as information-processing units in the human brain, so it is with neural networks made up of artificial neurons. In

its most general form, a *neural network* is a machine that is designed to *model* the way in which the brain performs a particular task or function of interest; the network is usually implemented by using electronic components or is simulated in software on a digital computer. In this book, we focus on an important class of neural networks that perform useful computations through a process of *learning*. To achieve good performance, neural networks employ a massive interconnection of simple computing cells referred to as "neurons" or "processing units." We may thus offer the following definition of a neural network viewed as an adaptive machine[1]:

> *A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*
>
> 1. *Knowledge is acquired by the network from its environment through a learning process.*
> 2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

The procedure used to perform the learning process is called a *learning algorithm*, the function of which is to modify the synaptic weights of the network in an orderly fashion to attain a desired design objective.

The modification of synaptic weights provides the traditional method for the design of neural networks. Such an approach is the closest to linear adaptive filter theory, which is already well established and successfully applied in many diverse fields (Widrow and Stearns, 1985; Haykin, 2002). However, it is also possible for a neural network to modify its own topology, which is motivated by the fact that neurons in the human brain can die and new synaptic connections can grow.

Benefits of Neural Networks

It is apparent that a neural network derives its computing power through, first, its massively parallel distributed structure and, second, its ability to learn and therefore generalize. *Generalization* refers to the neural network's production of reasonable outputs for inputs not encountered during training (learning). These two information-processing capabilities make it possible for neural networks to find good approximate solutions to complex (large-scale) problems that are *intractable*. In practice, however, neural networks cannot provide the solution by working individually. Rather, they need to be integrated into a consistent system engineering approach. Specifically, a complex problem of interest is *decomposed* into a number of relatively simple tasks, and neural networks are assigned a subset of the tasks that *match* their inherent capabilities. It is important to recognize, however, that we have a long way to go (if ever) before we can build a computer architecture that mimics the human brain.

Neural networks offer the following useful properties and capabilities:

1. *Nonlinearity.* An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear. Moreover, the nonlinearity is of a special kind in the sense that it is *distributed* throughout the network. Nonlinearity is a highly important property, particularly if the underlying physical

mechanism responsible for generation of the input signal (e.g., speech signal) is inherently nonlinear.

2. *Input–Output Mapping.* A popular paradigm of learning, called *learning with a teacher*, or *supervised learning*, involves modification of the synaptic weights of a neural network by applying a set of labeled *training examples*, or *task examples*. Each example consists of a unique *input signal* and a corresponding *desired (target) response*. The network is presented with an example picked at random from the set, and the synaptic weights (free parameters) of the network are modified to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate statistical criterion. The training of the network is repeated for many examples in the set, until the network reaches a steady state where there are no further significant changes in the synaptic weights. The previously applied training examples may be reapplied during the training session, but in a different order. Thus the network learns from the examples by constructing an *input–output mapping* for the problem at hand. Such an approach brings to mind the study of *nonparametric statistical inference*, which is a branch of statistics dealing with model-free estimation, or, from a biological viewpoint, *tabula rasa* learning (Geman et al., 1992); the term "nonparametric" is used here to signify the fact that no prior assumptions are made on a statistical model for the input data. Consider, for example, a *pattern classification* task, where the requirement is to assign an input signal representing a physical object or event to one of several prespecified categories (classes). In a nonparametric approach to this problem, the requirement is to "estimate" arbitrary decision boundaries in the input signal space for the pattern-classification task using a set of examples, and to do so *without* invoking a probabilistic distribution model. A similar point of view is implicit in the supervised learning paradigm, which suggests a close analogy between the input–output mapping performed by a neural network and nonparametric statistical inference.

3. *Adaptivity.* Neural networks have a built-in capability to *adapt* their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in a specific environment can be easily *retrained* to deal with minor changes in the operating environmental conditions. Moreover, when it is operating in a *nonstationary* environment (i.e., one where statistics change with time), a neural network may be designed to change its synaptic weights in real time. The natural architecture of a neural network for pattern classification, signal processing, and control applications, coupled with the adaptive capability of the network, makes it a useful tool in adaptive pattern classification, adaptive signal processing, and adaptive control. As a general rule, it may be said that the more adaptive we make a system, all the time ensuring that the system remains stable, the more robust its performance will likely be when the system is required to operate in a nonstationary environment. It should be emphasized, however, that adaptivity does not always lead to robustness; indeed, it may do the very opposite. For example, an adaptive system with short-time constants may change rapidly and therefore tend to respond to spurious disturbances, causing a drastic degradation in system performance. To realize the full benefits of adaptivity, the principal time constants of the system should be long enough for the system to ignore spurious disturbances, and yet short enough to respond to meaningful changes in the

environment; the problem described here is referred to as the *stability–plasticity dilemma* (Grossberg, 1988).

4. *Evidential Response.* In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to *select*, but also about the *confidence* in the decision made. This latter information may be used to reject ambiguous patterns, should they arise, and thereby improve the classification performance of the network.

5. *Contextual Information.* Knowledge is represented by the very structure and activation state of a neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.

6. *Fault Tolerance.* A neural network, implemented in hardware form, has the potential to be inherently *fault tolerant*, or capable of robust computation, in the sense that its performance degrades gracefully under adverse operating conditions. For example, if a neuron or its connecting links are damaged, recall of a stored pattern is impaired in quality. However, due to the distributed nature of information stored in the network, the damage has to be extensive before the overall response of the network is degraded seriously. Thus, in principle, a neural network exhibits a graceful degradation in performance rather than catastrophic failure. There is some empirical evidence for robust computation, but usually it is uncontrolled. In order to be assured that the neural network is, in fact, fault tolerant, it may be necessary to take corrective measures in designing the algorithm used to train the network (Kerlirzin and Vallet, 1993).

7. *VLSI Implementability.* The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks. This same feature makes a neural network well suited for implementation using *very-large-scale-integrated* (VLSI) technology. One particular beneficial virtue of VLSI is that it provides a means of capturing truly complex behavior in a highly hierarchical fashion (Mead, 1989).

8. *Uniformity of Analysis and Design.* Basically, neural networks enjoy universality as information processors. We say this in the sense that the same notation is used in all domains involving the application of neural networks. This feature manifests itself in different ways:

- Neurons, in one form or another, represent an ingredient *common* to all neural networks.
- This commonality makes it possible to *share* theories and learning algorithms in different applications of neural networks.
- Modular networks can be built through a *seamless integration of modules.*

9. *Neurobiological Analogy.* The design of a neural network is motivated by analogy with the brain, which is living proof that fault-tolerant parallel processing is not only physically possible, but also fast and powerful. Neurobiologists look to (artificial) neural networks as a research tool for the interpretation of neurobiological phenomena. On the other hand, engineers look to neurobiology for new ideas to solve problems more complex than those based on conventional hardwired design

techniques. These two viewpoints are illustrated by the following two respective examples:

- In Anastasio (1993), linear system models of the *vestibulo-ocular reflex* (VOR) are compared to neural network models based on *recurrent networks*, which are described in Section 6 and discussed in detail in Chapter 15. The vestibulo-ocular reflex is part of the oculomotor system. The function of VOR is to maintain visual (i.e., retinal) image stability by making eye rotations that are opposite to head rotations. The VOR is mediated by premotor neurons in the vestibular nuclei that receive and process head rotation signals from vestibular sensory neurons and send the results to the eye muscle motor neurons. The VOR is well suited for modeling because its input (head rotation) and its output (eye rotation) can be precisely specified. It is also a relatively simple reflex, and the neurophysiological properties of its constituent neurons have been well described. Among the three neural types, the premotor neurons (reflex interneurons) in the vestibular nuclei are the most complex and therefore most interesting. The VOR has previously been modeled using lumped, linear system descriptors and control theory. These models were useful in explaining some of the overall properties of the VOR, but gave little insight into the properties of its constituent neurons. This situation has been greatly improved through neural network modeling. Recurrent network models of VOR (programmed using an algorithm called real-time recurrent learning, described in Chapter 15) can reproduce and help explain many of the static, dynamic, nonlinear, and distributed aspects of signal processing by the neurons that mediate the VOR, especially the vestibular nuclei neurons.
- The *retina*, more than any other part of the brain, is where we begin to put together the relationships between the outside world represented by a visual sense, its *physical image* projected onto an array of receptors, and the first *neural images*. The retina is a thin sheet of neural tissue that lines the posterior hemisphere of the eyeball. The retina's task is to convert an optical image into a neural image for transmission down the optic nerve to a multitude of centers for further analysis. This is a complex task, as evidenced by the synaptic organization of the retina. In all vertebrate retinas, the transformation from optical to neural image involves three stages (Sterling, 1990):

  (i) photo transduction by a layer of receptor neurons;
  (ii) transmission of the resulting signals (produced in response to light) by chemical synapses to a layer of bipolar cells;
  (iii) transmission of these signals, also by chemical synapses, to output neurons that are called ganglion cells.

  At both synaptic stages (i.e., from receptor to bipolar cells, and from bipolar to ganglion cells), there are specialized laterally connected neurons called *horizontal cells* and *amacrine cells*, respectively. The task of these neurons is to modify the transmission across the synaptic layers. There are also centrifugal elements called *inter-plexiform cells*; their task is to convey signals from the inner synaptic layer back to the outer one. Some researchers have built electronic chips that mimic the structure of the retina. These electronic chips are called *neuromorphic* integrated circuits, a term coined by Mead (1989). A neuromorphic imaging sensor

consists of an array of photoreceptors combined with analog circuitry at each picture element (pixel). It emulates the retina in that it can adapt locally to changes in brightness, detect edges, and detect motion. The neurobiological analogy, exemplified by neuromorphic integrated circuits, is useful in another important way: It provides a hope and belief, and to a certain extent an existence of proof, that physical understanding of neurobiological structures could have a productive influence on the art of electronics and VLSI technology for the implementation of neural networks.

With inspiration from neurobiology in mind, it seems appropriate that we take a brief look at the human brain and its structural levels of organization.[2]

2 THE HUMAN BRAIN

The human nervous system may be viewed as a three-stage system, as depicted in the block diagram of Fig. 1 (Arbib, 1987). Central to the system is the *brain*, represented by the *neural (nerve) net*, which continually receives information, perceives it, and makes appropriate decisions. Two sets of arrows are shown in the figure. Those pointing from left to right indicate the *forward* transmission of information-bearing signals through the system. The arrows pointing from right to left (shown in red) signify the presence of *feedback* in the system. The *receptors* convert stimuli from the human body or the external environment into electrical impulses that convey information to the neural net (brain). The *effectors* convert electrical impulses generated by the neural net into discernible responses as system outputs.

The struggle to understand the brain has been made easier because of the pioneering work of Ramón y Cajál (1911), who introduced the idea of *neurons* as structural constituents of the brain. Typically, neurons are five to six orders of magnitude slower than silicon logic gates; events in a silicon chip happen in the nanosecond range, whereas neural events happen in the millisecond range. However, the brain makes up for the relatively slow rate of operation of a neuron by having a truly staggering number of neurons (nerve cells) with massive interconnections between them. It is estimated that there are approximately 10 billion neurons in the human cortex, and 60 trillion synapses or connections (Shepherd and Koch, 1990). The net result is that the brain is an enormously efficient structure. Specifically, the *energetic efficiency* of the brain is approximately 10^{-16} joules (J) per operation per second, whereas the corresponding value for the best computers is orders of magnitude larger.

Synapses, or *nerve endings*, are elementary structural and functional units that mediate the interactions between neurons. The most common kind of synapse is a *chemical synapse*, which operates as follows: A presynaptic process liberates a *transmitter* substance that diffuses across the synaptic junction between neurons and then acts on a postsynaptic process. Thus a synapse converts a presynaptic electrical signal into a chemical

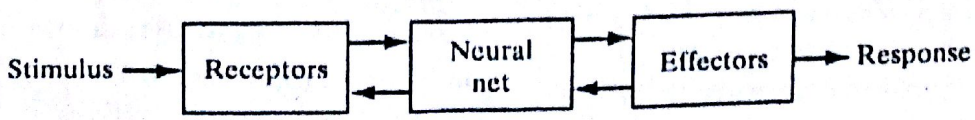

FIGURE 1 Block diagram representation of nervous system.

signal and then back into a postsynaptic electrical signal (Shepherd and Koch, 1990). In electrical terminology, such an element is said to be a *nonreciprocal two-port device.* In traditional descriptions of neural organization, it is assumed that a synapse is a simple connection that can impose *excitation* or *inhibition*, but not both on the receptive neuron.

Earlier we mentioned that plasticity permits the developing nervous system to adapt to its surrounding environment (Eggermont, 1990; Churchland and Sejnowski, 1992). In an adult brain, plasticity may be accounted for by two mechanisms: the creation of new synaptic connections between neurons, and the modification of existing synapses. *Axons*, the transmission lines, and *dendrites*, the receptive zones, constitute two types of cell filaments that are distinguished on morphological grounds; an axon has a smoother surface, fewer branches, and greater length, whereas a dendrite (so called because of its resemblance to a tree) has an irregular surface and more branches (Freeman, 1975). Neurons come in a wide variety of shapes and sizes in different parts of the brain. Figure 2 illustrates the shape of a *pyramidal cell*, which is one of the most common types of cortical neurons. Like many other types of neurons, it receives most of its inputs through dendritic spines; see the segment of dendrite in the insert in Fig. 2 for detail. The pyramidal cell can receive 10,000 or more synaptic contacts, and it can project onto thousands of target cells.

The majority of neurons encode their outputs as a series of brief voltage pulses. These pulses, commonly known as *action potentials*, or *spikes*,[3] originate at or close to the cell body of neurons and then propagate across the individual neurons at constant velocity and amplitude. The reasons for the use of action potentials for communication among neurons are based on the physics of axons. The axon of a neuron is very long and thin and is characterized by high electrical resistance and very large capacitance. Both of these elements are distributed across the axon. The axon may therefore be modeled as resistance-capacitance (RC) transmission line, hence the common use of "cable equation" as the terminology for describing signal propagation along an axon. Analysis of this propagation mechanism reveals that when a voltage is applied at one end of the axon, it decays exponentially with distance, dropping to an insignificant level by the time it reaches the other end. The action potentials provide a way to circumvent this transmission problem (Anderson, 1995).

In the brain, there are both small-scale and large-scale anatomical organizations, and different functions take place at lower and higher levels. Figure 3 shows a hierarchy of interwoven levels of organization that has emerged from the extensive work done on the analysis of local regions in the brain (Shepherd and Koch, 1990; Churchland and Sejnowski, 1992). The *synapses* represent the most fundamental level, depending on molecules and ions for their action. At the next levels, we have neural microcircuits, dendritic trees, and then neurons. A *neural microcircuit* refers to an assembly of synapses organized into patterns of connectivity to produce a functional operation of interest. A neural microcircuit may be likened to a silicon chip made up of an assembly of transistors. The smallest size of microcircuits is measured in micrometers (μm), and their fastest speed of operation is measured in milliseconds. The neural microcircuits are grouped to form *dendritic subunits* within the *dendritic trees* of individual neurons. The whole *neuron*, about 100 μm in size, contains several dendritic subunits. At the next level of complexity, we have *local circuits* (about 1 mm in size) made up of neurons with similar or different properties; these neural

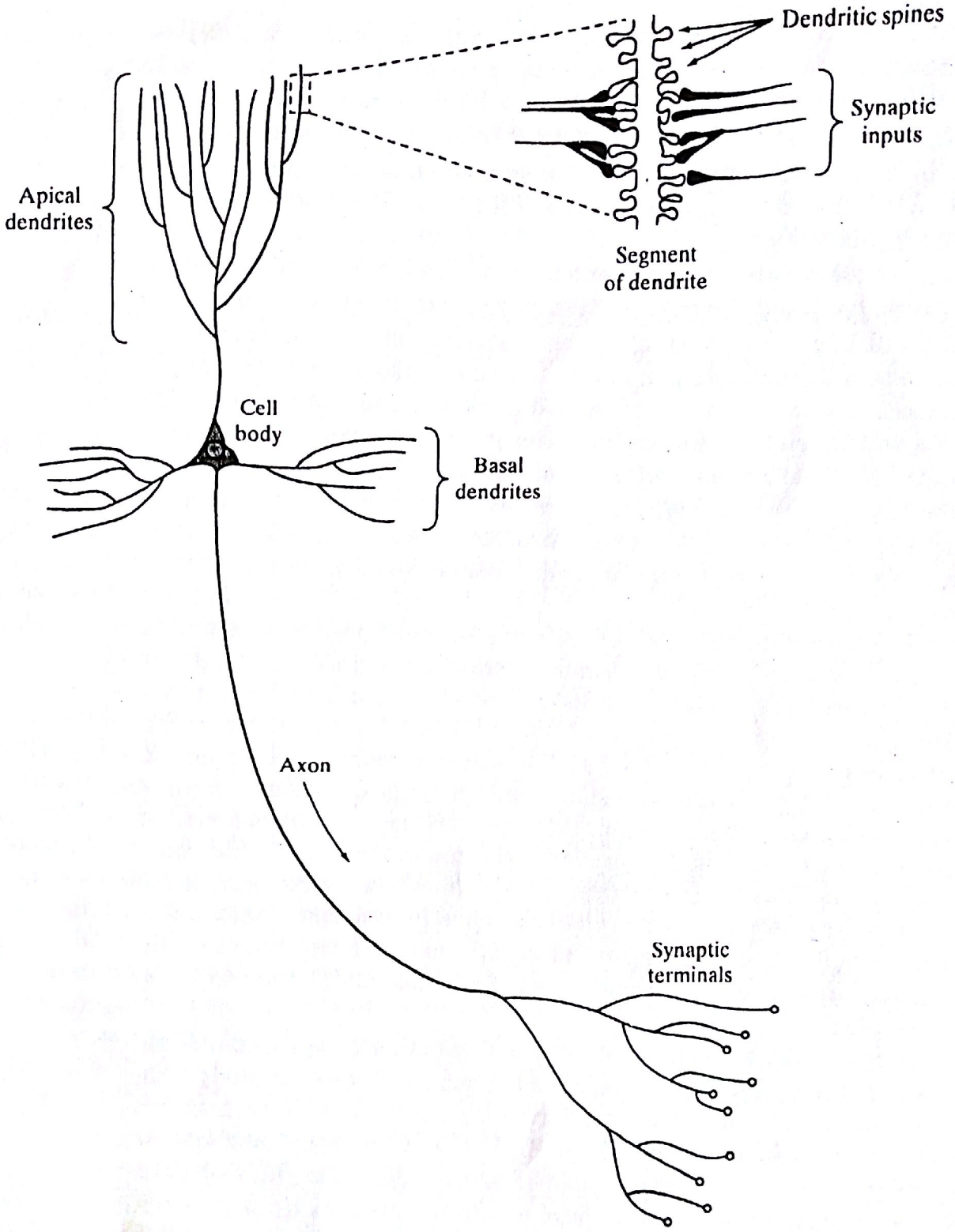

FIGURE 2 The pyramidal cell.

assemblies perform operations characteristic of a localized region in the brain. They are followed by *interregional circuits* made up of pathways, columns, and topographic maps, which involve multiple regions located in different parts of the brain.

Topographic maps are organized to respond to incoming sensory information. These maps are often arranged in sheets, as in the *superior colliculus*, where the visual,

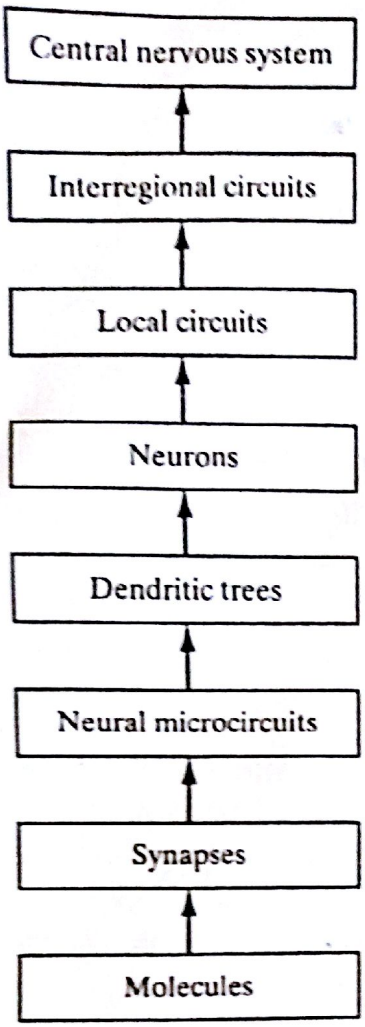

FIGURE 3 Structural organization of levels in the brain.

auditory, and somatosensory maps are stacked in adjacent layers in such a way that stimuli from corresponding points in space lie above or below each other. Figure 4 presents a cytoarchitectural map of the cerebral cortex as worked out by Brodmann (Brodal, 1981). This figure shows clearly that different sensory inputs (motor, somatosensory, visual, auditory, etc.) are mapped onto corresponding areas of the cerebral cortex in an orderly fashion. At the final level of complexity, the topographic maps and other interregional circuits mediate specific types of behavior in the *central nervous system.*

It is important to recognize that the structural levels of organization described herein are a unique characteristic of the brain. They are nowhere to be found in a digital computer, and we are nowhere close to re-creating them with artificial neural networks. Nevertheless, we are inching our way toward a hierarchy of computational levels similar to that described in Fig. 3. The artificial neurons we use to build our neural networks are truly primitive in comparison with those found in the brain. The neural networks we are presently able to design are just as primitive compared with the local circuits and the interregional circuits in the brain. What is really satisfying, however, is the remarkable progress that we have made on so many fronts. With neurobiological analogy as the source of inspiration, and the wealth of theoretical and computational tools that we are bringing together, it is certain that our understanding of artificial neural networks and their applications will continue to grow in depth as well as breadth, year after year.

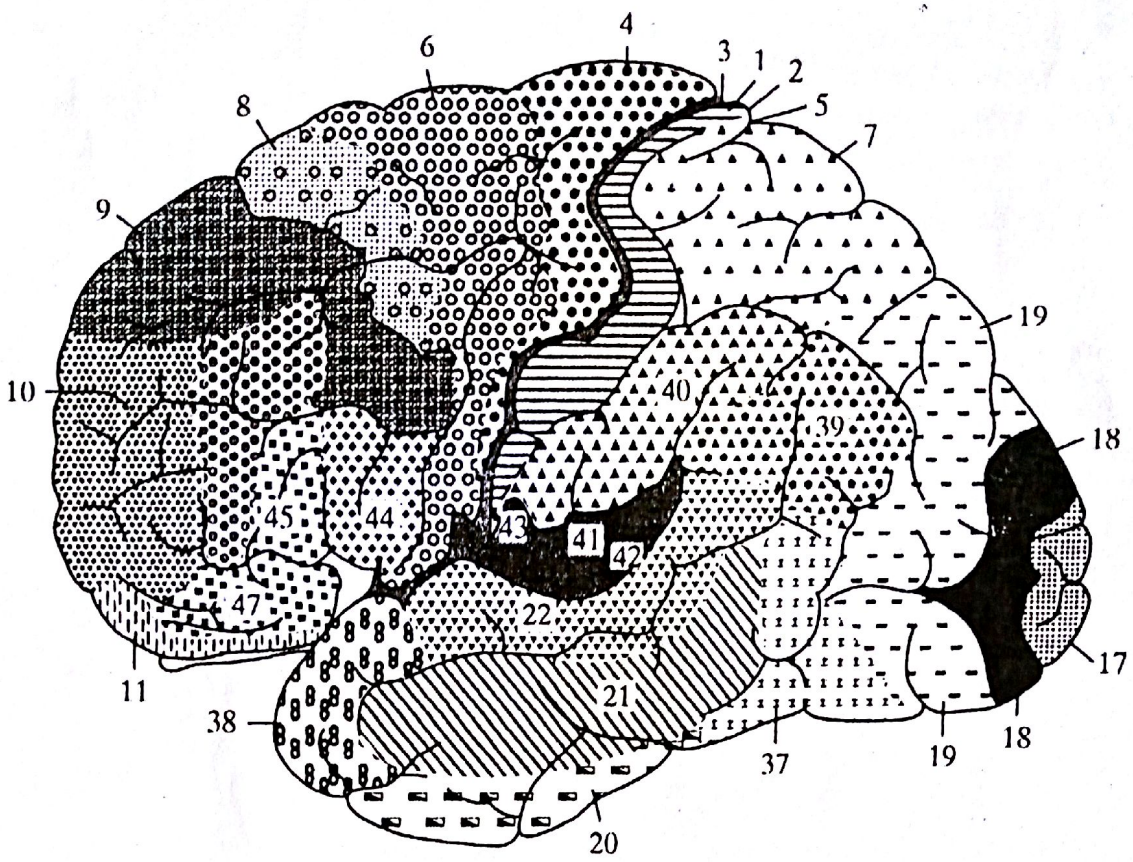

FIGURE 4 Cytoarchitectural map of the cerebral cortex. The different areas are identified by the thickness of their layers and types of cells within them. Some of the key sensory areas are as follows: Motor cortex: motor strip, area 4; premotor area, area 6; frontal eye fields, area 8. Somatosensory cortex: areas 3, 1, and 2. Visual cortex: areas 17, 18, and 19. Auditory cortex: areas 41 and 42. *(From A. Brodal, 1981; with permission of Oxford University Press.)*

3 MODELS OF A NEURON

A *neuron* is an information-processing unit that is fundamental to the operation of a neural network. The block diagram of Fig. 5 shows the *model* of a neuron, which forms the basis for designing a large family of neural networks studied in later chapters. Here, we identify three basic elements of the neural model:

1. A set of *synapses*, or *connecting links*, each of which is characterized by a *weight* or *strength* of its own. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. It is important to make a note of the manner in which the subscripts of the synaptic weight w_{kj} are written. The first subscript in w_{kj} refers to the neuron in question, and the second subscript refers to the input end of the synapse to which the weight refers. Unlike the weight of a synapse in the brain, the synaptic weight of an artificial neuron may lie in a range that includes negative as well as positive values.
2. An *adder* for summing the input signals, weighted by the respective synaptic strengths of the neuron; the operations described here constitute a *linear combiner*.
3. An *activation function* for limiting the amplitude of the output of a neuron. The activation function is also referred to as a *squashing function*, in that it squashes (limits) the permissible amplitude range of the output signal to some finite value.

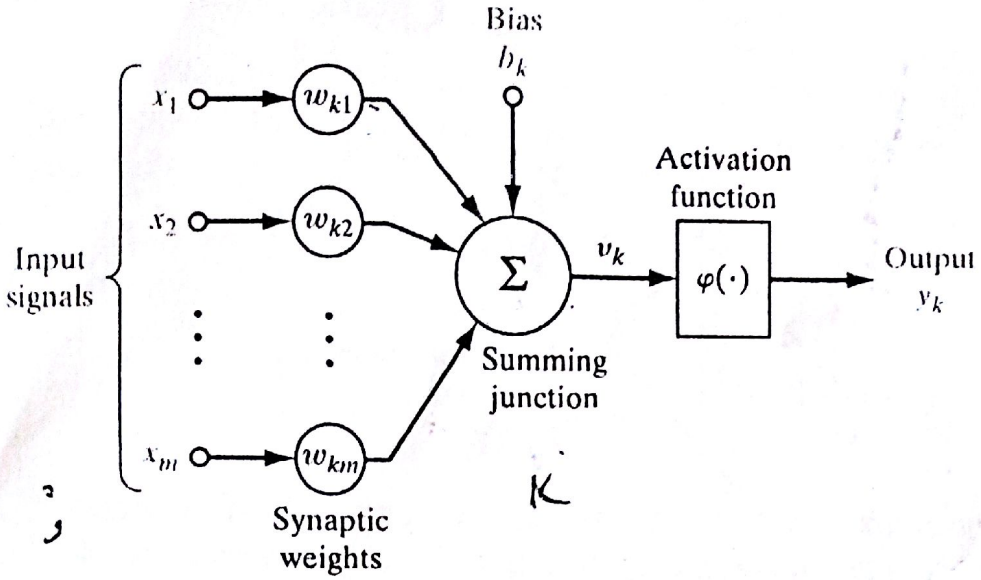

FIGURE 5 Nonlinear model of a neuron, labeled k.

Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1], or, alternatively, [−1,1].

The neural model of Fig. 5 also includes an externally applied *bias*, denoted by b_k. The bias b_k has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively.

In mathematical terms, we may describe the neuron k depicted in Fig. 5 by writing the pair of equations:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1}$$

and

$$y_k = \varphi(u_k + b_k) \tag{2}$$

where $x_1, x_2, \ldots, x_m$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{km}$ are the respective synaptic weights of neuron k; u_k (not shown in Fig. 5) is the *linear combiner output* due to the input signals; b_k is the bias; $\varphi(\cdot)$ is the *activation function*; and y_k is the output signal of the neuron. The use of bias b_k has the effect of applying an *affine transformation* to the output u_k of the linear combiner in the model of Fig. 5, as shown by

$$v_k = u_k + b_k \tag{3}$$

In particular, depending on whether the bias b_k is positive or negative, the relationship between the *induced local field*, or *activation potential*, v_k of neuron k and the linear combiner output u_k is modified in the manner illustrated in Fig. 6; hereafter, these two terms are used interchangeably. Note that as a result of this affine transformation, the graph of v_k versus u_k no longer passes through the origin.

The bias b_k is an external parameter of neuron k. We may account for its presence as in Eq. (2). Equivalently, we may formulate the combination of Eqs. (1) to (3) as follows:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{4}$$

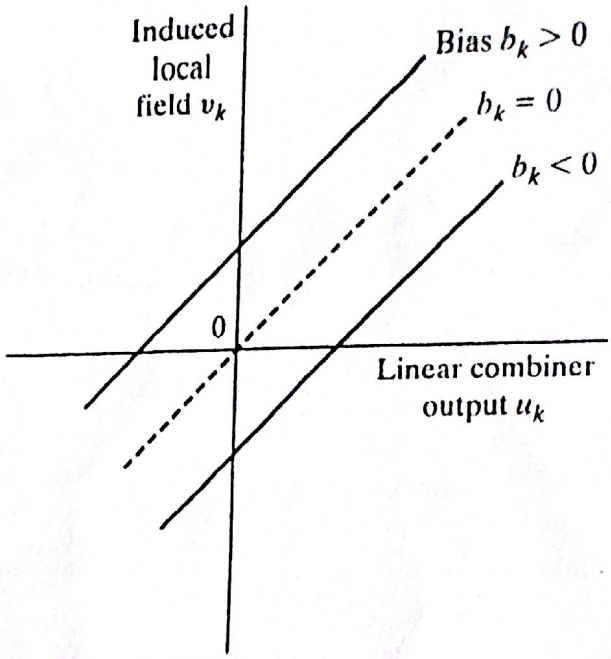

FIGURE 6 Affine transformation produced by the presence of a bias; note that $v_k = b_k$ at $u_k = 0$.

and

$$y_k = \varphi(v_k) \tag{5}$$

In Eq. (4), we have added a new synapse. Its input is

$$x_0 = +1 \tag{6}$$

and its weight is

$$w_{k0} = b_k \tag{7}$$

We may therefore reformulate the model of neuron k as shown in Fig. 7. In this figure, the effect of the bias is accounted for by doing two things: (1) adding a new input signal fixed at +1, and (2) adding a new synaptic weight equal to the bias b_k. Although the models of Figs. 5 and 7 are different in appearance, they are mathematically equivalent.

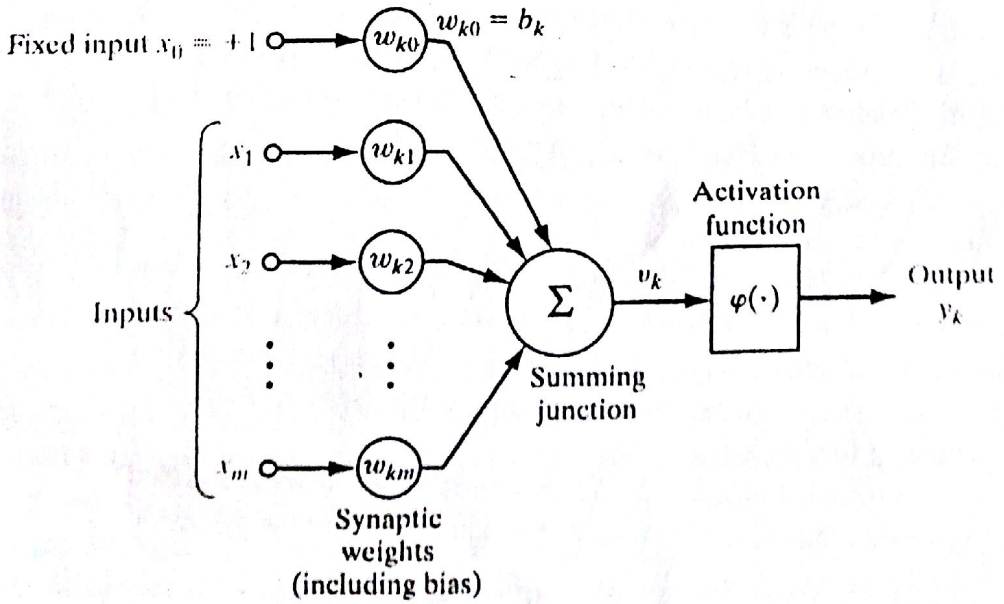

FIGURE 7 Another nonlinear model of a neuron; w_{k0} accounts for the bias b_k.

Types of Activation Function

The activation function, denoted by $\varphi(v)$, defines the output of a neuron in terms of the induced local field v. In what follows, we identify two basic types of activation functions:

1. *Threshold Function.* For this type of activation function, described in Fig. 8a, we have

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \tag{8}$$

In engineering, this form of a threshold function is commonly referred to as a *Heaviside function.* Correspondingly, the output of neuron k employing such a threshold function is expressed as

$$y_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases} \tag{9}$$

where v_k is the induced local field of the neuron; that is,

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{10}$$

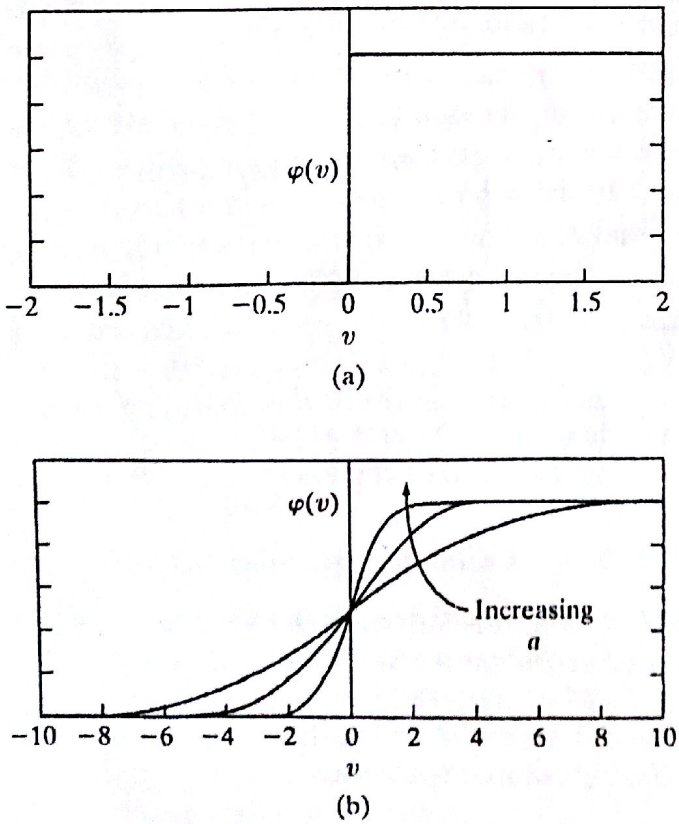

FIGURE 8 (a) Threshold function. (b) Sigmoid function for varying slope parameter a.

In neural computation, such a neuron is referred to as the *McCulloch–Pitts model*, in recognition of the pioneering work done by McCulloch and Pitts (1943). In this model, the output of a neuron takes on the value of 1 if the induced local field of that neuron is nonnegative, and 0 otherwise. This statement describes the *all-or-none property* of the McCulloch–Pitts model.

2. *Sigmoid Function.*[4] The sigmoid function, whose graph is "S"-shaped, is by far the most common form of activation function used in the construction of neural networks. It is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behavior. An example of the sigmoid function is the *logistic function*,[5] defined by

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{11}$$

where a is the *slope parameter* of the sigmoid function. By varying the parameter a, we obtain sigmoid functions of different slopes, as illustrated in Fig. 8b. In fact, the slope at the origin equals $a/4$. In the limit, as the slope parameter approaches infinity, the sigmoid function becomes simply a threshold function. Whereas a threshold function assumes the value of 0 or 1, a sigmoid function assumes a continuous range of values from 0 to 1. Note also that the sigmoid function is differentiable, whereas the threshold function is not. (Differentiability is an important feature of neural network theory, as described in Chapter 4).

The activation functions defined in Eqs. (8) and (11) range from 0 to +1. It is sometimes desirable to have the activation function range from −1 to +1, in which case, the activation function is an odd function of the induced local field. Specifically, the threshold function of Eq. (8) is now defined as

$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases} \tag{12}$$

which is commonly referred to as the *signum function*. For the corresponding form of a sigmoid function, we may use the *hyperbolic tangent function*, defined by

$$\varphi(v) = \tanh(v) \tag{13}$$

Allowing an activation function of the sigmoid type to assume negative values as prescribed by Eq. (13) may yield practical benefits over the logistic function of Eq. (11).

Stochastic Model of a Neuron

The neural model described in Fig. 7 is deterministic in that its input–output behavior is precisely defined for all inputs. For some applications of neural networks, it is desirable to base the analysis on a stochastic neural model. In an analytically tractable approach, the activation function of the McCulloch–Pitts model is given a probabilistic interpretation. Specifically, a neuron is permitted to reside in only one of two states: +1

or −1, say. The decision for a neuron to *fire* (i.e., switch its state from "off" to "on") is probabilistic. Let x denote the state of the neuron and $P(v)$ denote the *probability* of firing, where v is the induced local field of the neuron. We may then write

$$x = \begin{cases} +1 & \text{with probability } P(v) \\ -1 & \text{with probability } 1 - P(v) \end{cases} \tag{14}$$

A standard choice for $P(v)$ is the sigmoid-shaped function

$$P(v) = \frac{1}{1 + \exp(-v/T)} \tag{15}$$

where T is a *pseudotemperature* used to control the noise level and therefore the uncertainty in firing (Little, 1974). It is important to realize, however, that T is *not* the physical temperature of a neural network, be it a biological or an artificial neural network. Rather, as already stated, we should think of T merely as a parameter that controls the thermal fluctuations representing the effects of synaptic noise. Note that when $T \rightarrow 0$, the stochastic neuron described by Eqs. (14) and (15) reduces to a noiseless (i.e., deterministic) form, namely, the McCulloch–Pitts model.

4 NEURAL NETWORKS VIEWED AS DIRECTED GRAPHS

The *block diagram* of Fig. 5 or that of Fig. 7 provides a functional description of the various elements that constitute the model of an artificial neuron. We may simplify the appearance of the model by using the idea of signal-flow graphs without sacrificing any of the functional details of the model. Signal-flow graphs, with a well-defined set of rules, were originally developed by Mason (1953, 1956) for linear networks. The presence of nonlinearity in the model of a neuron limits the scope of their application to neural networks. Nevertheless, signal-flow graphs do provide a neat method for the portrayal of the flow of signals in a neural network, which we pursue in this section.

A *signal-flow graph* is a network of directed *links* (*branches*) that are interconnected at certain points called *nodes*. A typical node j has an associated *node signal* x_j. A typical directed link originates at node j and terminates on node k; it has an associated *transfer function*, or *transmittance*, that specifies the manner in which the signal y_k at node k depends on the signal x_j at node j. The flow of signals in the various parts of the graph is dictated by three basic rules:

Rule 1. A signal flows along a link only in the direction defined by the arrow on the link.

Two different types of links may be distinguished:

- *Synaptic links*, whose behavior is governed by a *linear* input–output relation. Specifically, the node signal x_j is multiplied by the synaptic weight w_{kj} to produce the node signal y_k, as illustrated in Fig. 9a.
- *Activation links*, whose behavior is governed in general by a *nonlinear* input–output relation. This form of relationship is illustrated in Fig. 9b, where $\varphi(\cdot)$ is the nonlinear activation function.

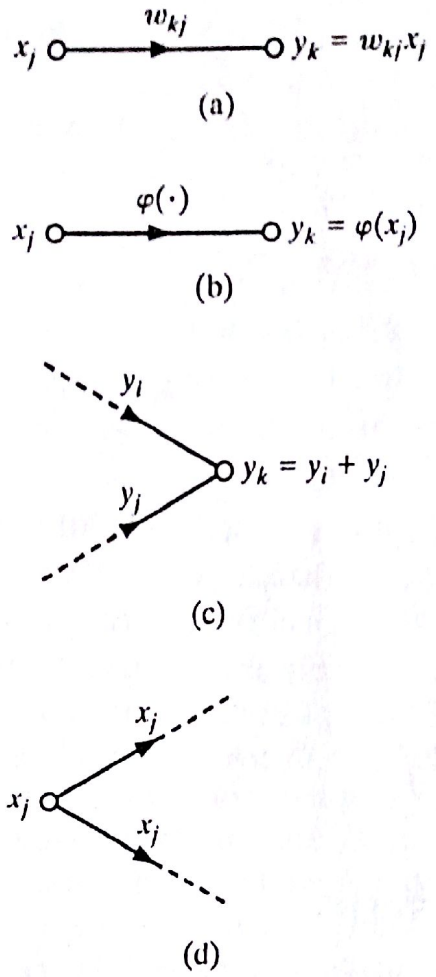

FIGURE 9 Illustrating basic rules for the construction of signal-flow graphs.

Rule 2. A node signal equals the algebraic sum of all signals entering the pertinent node via the incoming links.

This second rule is illustrated in Fig. 9c for the case of *synaptic convergence*, or *fan-in*.

Rule 3. The signal at a node is transmitted to each outgoing link originating from that node, with the transmission being entirely independent of the transfer functions of the outgoing links.

This third rule is illustrated in Fig. 9d for the case of *synaptic divergence*, or *fan-out*.

For example, using these rules, we may construct the signal-flow graph of Fig. 10 as the model of a neuron, corresponding to the block diagram of Fig. 7. The representation shown in Fig. 10 is clearly simpler in appearance than that of Fig. 7, yet it contains all the functional details depicted in the latter diagram. Note that in both figures, the input $x_0 = +1$ and the associated synaptic weight $w_{k0} = b_k$, where b_k is the bias applied to neuron k.

Indeed, based on the signal-flow graph of Fig. 10 as the model of a neuron, we may now offer the following mathematical definition of a neural network:

A neural network is a directed graph consisting of nodes with interconnecting synaptic and activation links and is characterized by four properties:

1. *Each neuron is represented by a set of linear synaptic links, an externally applied bias, and a possibly nonlinear activation link. The bias is represented by a synaptic link connected to an input fixed at +1.*
2. *The synaptic links of a neuron weight their respective input signals.*

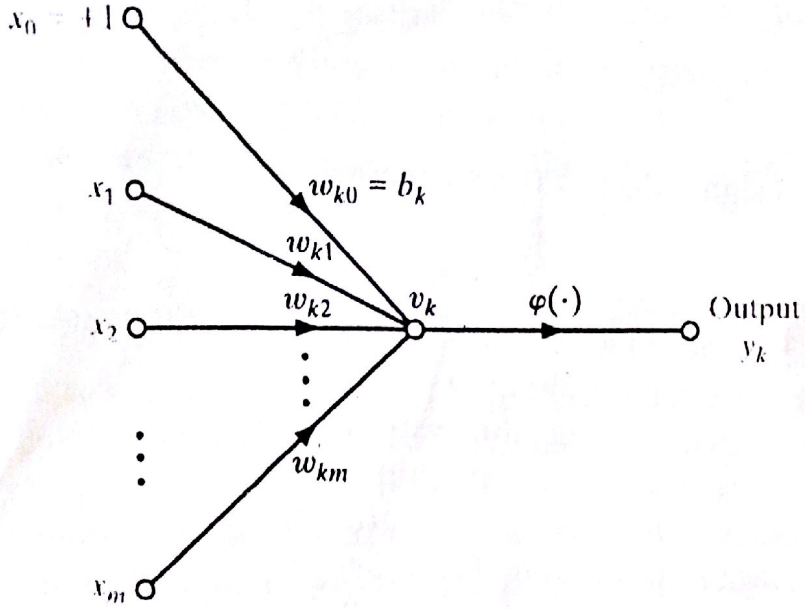

FIGURE 10 Signal-flow graph of a neuron.

3. *The weighted sum of the input signals defines the induced local field of the neuron in question.*
4. *The activation link squashes the induced local field of the neuron to produce an output.*

A directed graph, defined in this manner is *complete* in the sense that it describes not only the signal flow from neuron to neuron, but also the signal flow inside each neuron. When, however, the focus of attention is restricted to signal flow from neuron to neuron, we may use a reduced form of this graph by omitting the details of signal flow inside the individual neurons. Such a directed graph is said to be *partially complete.* It is characterized as follows:

1. *Source nodes* supply input signals to the graph.
2. Each neuron is represented by a single node called a *computation node.*
3. The *communication links* interconnecting the source and computation nodes of the graph carry no weight; they merely provide directions of signal flow in the graph.

A partially complete directed graph defined in this way is referred to as an *architectural graph*, describing the layout of the neural network. It is illustrated in Fig. 11 for the simple case of a single neuron with m source nodes and a single node fixed at +1 for the bias. Note that the computation node representing the neuron is shown shaded, and the source node is shown as a small square. This convention is followed throughout the book. More elaborate examples of architectural layouts are presented later in Section 6.

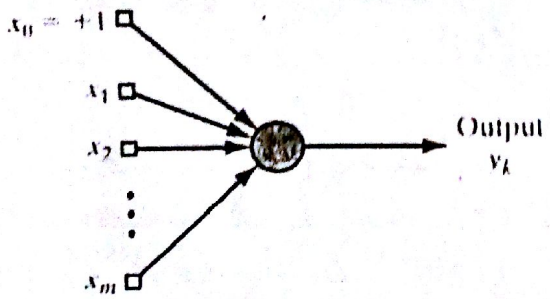

FIGURE 11 Architectural graph of a neuron.

To sum up, we have three graphical representations of a neural network:

- block diagram, providing a functional description of the network;
- architectural graph, describing the network layout;
- signal-flow graph, providing a complete description of signal flow in the network.

5 FEEDBACK

Feedback is said to exist in a dynamic system whenever the output of an element in the system influences in part the input applied to that particular element, thereby giving rise to one or more closed paths for the transmission of signals around the system. Indeed, feedback occurs in almost every part of the nervous system of every animal (Freeman, 1975). Moreover, it plays a major role in the study of a special class of neural networks known as *recurrent networks*. Figure 12 shows the signal-flow graph of a *single-loop feedback system*, where the input signal $x_j(n)$, internal signal $x_j'(n)$, and output signal $y_k(n)$ are functions of the discrete-time variable n. The system is assumed to be *linear*, consisting of a forward path and a feedback path that are characterized by the "operators" **A** and **B**, respectively. In particular, the output of the forward channel determines in part its own output through the feedback channel. From Fig. 12, we readily note the input–output relationships

$$y_k(n) = \mathbf{A}[x_j'(n)] \tag{16}$$

and

$$x_j'(n) = x_j(n) + \mathbf{B}[y_k(n)] \tag{17}$$

where the square brackets are included to emphasize that **A** and **B** act as *operators*. Eliminating $x_j'(n)$ between Eqs. (16) and (17), we get

$$y_k(n) = \frac{\mathbf{A}}{1 - \mathbf{AB}}[x_j(n)] \tag{18}$$

We refer to $\mathbf{A}/(1 - \mathbf{AB})$ as the *closed-loop operator* of the system, and to **AB** as the *open-loop operator*. In general, the open-loop operator is noncommutative in that $\mathbf{BA} \neq \mathbf{AB}$.

Consider, for example, the single-loop feedback system shown in Fig. 13a, for which **A** is a fixed weight w and **B** is a *unit-delay operator* z^{-1}, whose output is delayed with respect to the input by one time unit. We may then express the closed-loop operator of the system as

$$\begin{aligned}\frac{\mathbf{A}}{1 - \mathbf{AB}} &= \frac{w}{1 - wz^{-1}} \\ &= w(1 - wz^{-1})^{-1}\end{aligned}$$

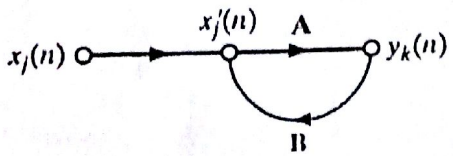

FIGURE 12 Signal-flow graph of a single-loop feedback system.

The analysis of the dynamic behavior of neural networks involving the application of feedback is unfortunately complicated by the fact that the processing units used for the construction of the network are usually *nonlinear*. Further consideration of this important issue is deferred to the latter part of the book.

6 NETWORK ARCHITECTURES

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network. We may therefore speak of learning algorithms (rules) used in the design of neural networks as being *structured*. The classification of learning algorithms is considered in Section 8. In this section, we focus attention on network architectures (structures).

In general, we may identify three fundamentally different classes of network architectures:

(i) Single-Layer Feedforward Networks

In a *layered* neural network, the neurons are organized in the form of layers. In the simplest form of a layered network, we have an *input layer* of source nodes that projects directly onto an *output layer* of neurons (computation nodes), but not vice versa. In other words, this network is strictly of a *feedforward* type. It is illustrated in Fig. 15 for the case of four nodes in both the input and output layers. Such a network is called a *single-layer network*, with the designation "single-layer" referring to the output layer of computation nodes (neurons). We do not count the input layer of source nodes because no computation is performed there.

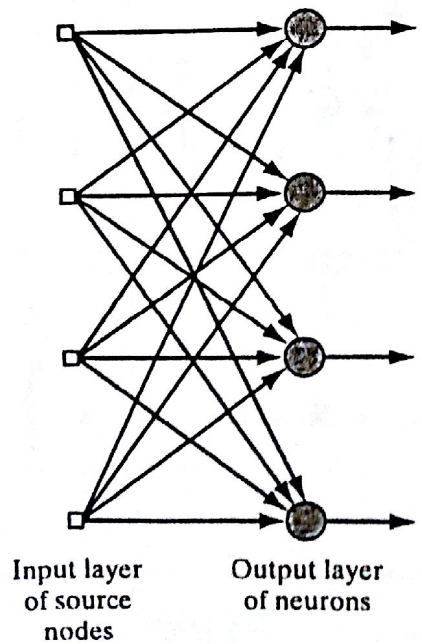

FIGURE 15 Feedforward network with a single layer of neurons.

(ii) Multilayer Feedforward Networks

The second class of a feedforward neural network distinguishes itself by the presence of one or more *hidden layers*, whose computation nodes are correspondingly called *hidden neurons* or *hidden units*; the term "hidden" refers to the fact that this part of the neural network is not seen directly from either the input or output of the network. The function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or more hidden layers, the network is enabled to extract higher-order statistics from its input. In a rather loose sense, the network acquires a *global* perspective despite its local connectivity, due to the extra set of synaptic connections and the extra dimension of neural interactions (Churchland and Sejnowski, 1992).

The source nodes in the input layer of the network supply respective elements of the activation pattern (input vector), which constitute the input signals applied to the neurons (computation nodes) in the second layer (i.e., the first hidden layer). The output signals of the second layer are used as inputs to the third layer, and so on for the rest of the network. Typically, the neurons in each layer of the network have as their inputs the output signals of the preceding layer only. The set of output signals of the neurons in the output (final) layer of the network constitutes the overall response of the network to the activation pattern supplied by the source nodes in the input (first) layer. The architectural graph in Fig. 16 illustrates the layout of a multilayer feedforward neural network for the case of a single hidden layer. For the sake of brevity, the network in Fig. 16 is referred to as a 10–4–2 network because it has 10 source nodes, 4 hidden neurons, and 2 output neurons. As another example, a feedforward network with m source nodes, h_1 neurons in the first hidden layer, h_2 neurons in the second hidden layer, and q neurons in the output layer is referred to as an m–h_1–h_2–q network.

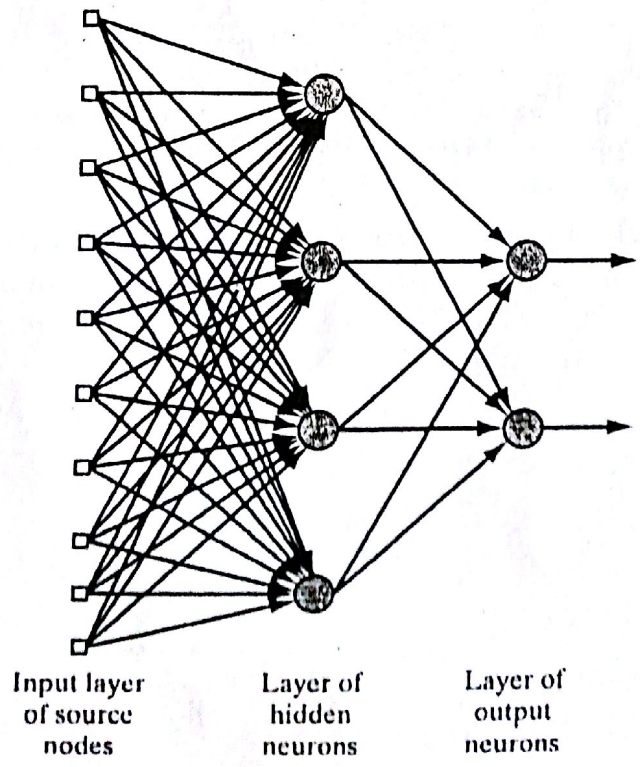

FIGURE 16 Fully connected feedforward network with one hidden layer and one output layer.

The neural network in Fig. 16 is said to be *fully connected* in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer. If, however, some of the communication links (synaptic connections) are missing from the network, we say that the network is *partially connected.*

(iii) Recurrent Networks

A *recurrent neural network* distinguishes itself from a feedforward neural network in that it has at least one *feedback* loop. For example, a recurrent network may consist of a single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons, as illustrated in the architectural graph in Fig. 17. In the structure depicted in this figure, there are *no* self-feedback loops in the network; self-feedback refers to a situation where the output of a neuron is fed back into its own input. The recurrent network illustrated in Fig. 17 also has *no* hidden neurons.

In Fig. 18 we illustrate another class of recurrent networks with hidden neurons. The feedback connections shown in Fig. 18 originate from the hidden neurons as well as from the output neurons.

The presence of feedback loops, be it in the recurrent structure of Fig. 17 or in that of Fig. 18, has a profound impact on the learning capability of the network and on its performance. Moreover, the feedback loops involve the use of particular branches composed of unit-time delay elements (denoted by z^{-1}), which result in a nonlinear dynamic behavior, assuming that the neural network contains nonlinear units.

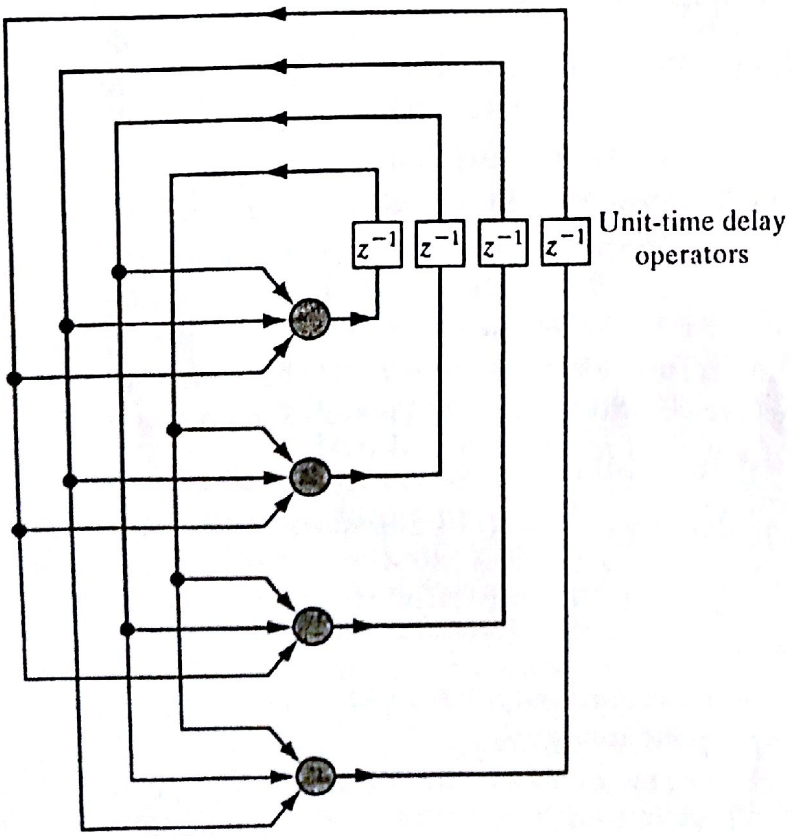

FIGURE 17 Recurrent network with no self-feedback loops and no hidden neurons.

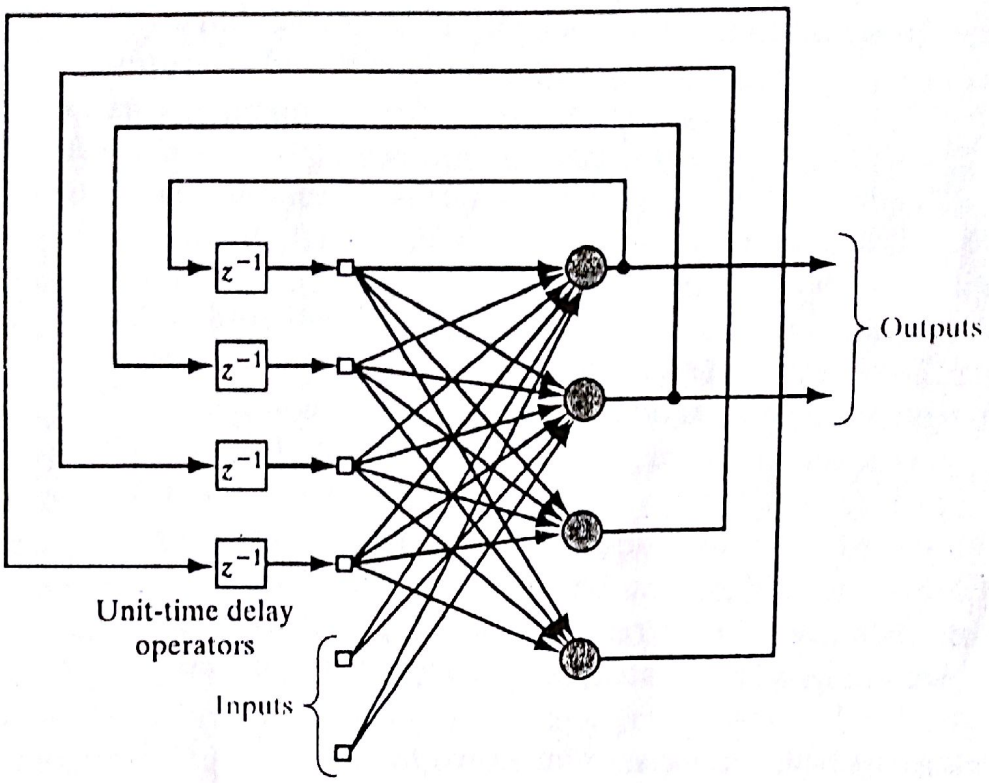

FIGURE 18 Recurrent network with hidden neurons.

7 KNOWLEDGE REPRESENTATION

In Section 1, we used the term "knowledge" in the definition of a neural network without an explicit description of what we mean by it. We now take care of this matter by offering the following generic definition (Fischler and Firschein, 1987):

> *Knowledge refers to stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world.*

The primary characteristics of *knowledge representation* are twofold: (1) what information is actually made explicit; and (2) how the information is physically encoded for subsequent use. By the very nature of it, therefore, knowledge representation is goal directed. In real-world applications of "intelligent" machines, it can be said that a good solution depends on a good representation of knowledge (Woods, 1986). So it is with neural networks. Typically, however, we find that the possible forms of representation from the inputs to internal network parameters are highly diverse, which tends to make the development of a satisfactory solution by means of a neural network a real design challenge.

A major task for a neural network is to learn a model of the world (environment) in which it is embedded, and to maintain the model sufficiently consistently with the real world so as to achieve the specified goals of the application of interest. Knowledge of the world consists of two kinds of information:

1. The known world state, represented by facts about what is and what has been known; this form of knowledge is referred to as *prior information*.
2. Observations (measurements) of the world, obtained by means of sensors designed to probe the environment, in which the neural network is supposed to operate.

Ordinarily, these observations are inherently noisy, being subject to errors due to sensor noise and system imperfections. In any event, the observations so obtained provide the pool of information, from which the *examples* used to train the neural network are drawn.

The examples can be *labeled* or *unlabeled*. In labeled examples, each example representing an *input signal* is paired with a corresponding *desired response* (i.e., target output). On the other hand, unlabeled examples consist of different realizations of the input signal all by itself. In any event, a set of examples, labeled or otherwise, represents knowledge about the environment of interest that a neural network can learn through training. Note, however, that labeled examples may be expensive to collect, as they require the availability of a "teacher" to provide a desired response for each labeled example. In contrast, unlabeled examples are usually abundant as there is no need for supervision.

A set of input–output pairs, with each pair consisting of an input signal and the corresponding desired response, is referred to as a *set of training data*, or simply *training sample*. To illustrate how such a data set can be used, consider, for example, the *handwritten-digit recognition problem*. In this problem, the input signal consists of an image with black or white pixels, with each image representing one of 10 digits that are well separated from the background. The desired response is defined by the "identity" of the particular digit whose image is presented to the network as the input signal. Typically, the training sample consists of a large variety of handwritten digits that are representative of a real-world situation. Given such a set of examples, the design of a neural network may proceed as follows:

- An appropriate architecture is selected for the neural network, with an input layer consisting of source nodes equal in number to the pixels of an input image, and an output layer consisting of 10 neurons (one for each digit). A subset of examples is then used to train the network by means of a suitable algorithm. This phase of the network design is called *learning*.
- The recognition performance of the trained network is *tested* with data not seen before. Specifically, an input image is presented to the network, but this time the network is not told the identity of the digit which that particular image represents. The performance of the network is then assessed by comparing the digit recognition reported by the network with the actual identity of the digit in question. This second phase of the network operation is called *testing*, and successful performance on the test patterns is called *generalization*, a term borrowed from psychology.

Herein lies a fundamental difference between the design of a neural network and that of its classical information-processing counterpart: the pattern classifier. In the latter case, we usually proceed by first formulating a mathematical model of environmental observations, validating the model with real data, and then building the design on the basis of the model. In contrast, the design of a neural network is based directly on real-life data, with the *data set being permitted to speak for itself*. Thus, the neural network not only provides the implicit model of the environment in which it is embedded, but also performs the information-processing function of interest.

The examples used to train a neural network may consist of both *positive* and *negative* examples. For instance, in a passive sonar detection problem, positive examples pertain to input training data that contain the target of interest (e.g., a submarine). Now,

- *Echo amplitude*, which is encoded by other neurons with different dynamic ranges; it is manifested both as amplitude tuning and as the number of discharges per stimulus.
- *Echo delay*, which is encoded through neural computations (based on cross-correlation) that produce delay-selective responses; it is manifested as target-range tuning.

The two principal characteristics of a target echo for image-forming purposes are *spectrum* for target shape and *delay* for target range. The bat perceives "shape" in terms of the arrival time of echoes from different reflecting surfaces (glints) within the target. For this to occur, *frequency* information in the echo spectrum is converted into estimates of the *time* structure of the target. Experiments conducted by Simmons and coworkers on the big brown bat, *Eptesicus fuscus*, critically identify this conversion process as consisting of parallel time-domain and frequency-to-time-domain transforms whose converging outputs create the common delay of range axis of a perceived image of the target. It appears that the unity of the bat's perception is due to certain properties of the transforms themselves, despite the separate ways in which the auditory time representation of the echo delay and frequency representation of the echo spectrum are initially performed. Moreover, feature invariances are built into the sonar image-forming process so as to make it essentially independent of the target's motion and the bat's own motion. ■

Some Final Remarks

The issue of knowledge representation in a neural network is directly related to that of network architecture. Unfortunately, there is no well-developed theory for optimizing the architecture of a neural network required to interact with an environment of interest, or for evaluating the way in which changes in the network architecture affect the representation of knowledge inside the network. Indeed, satisfactory answers to these issues are usually found through an exhaustive experimental study for a specific application of interest, with the designer of the neural network becoming an essential part of the structural learning loop.

8 LEARNING PROCESSES

Just as there are different ways in which we ourselves learn from our own surrounding environments, so it is with neural networks. In a broad sense, we may categorize the learning processes through which neural networks function as follows: learning with a teacher and learning without a teacher. By the same token, the latter form of learning may be subcategorized into unsupervised learning and reinforcement learning. These different forms of learning as performed on neural networks parallel those of human learning.

Learning with a Teacher

Learning with a teacher is also referred to as *supervised learning*. Figure 24 shows a block diagram that illustrates this form of learning. In conceptual terms, we may think of the teacher as having knowledge of the environment, with that knowledge being represented by a set of *input–output examples*. The environment is, however, *unknown* to the neural network. Suppose now that the teacher and the neural network are both exposed to a training vector (i.e., example) drawn from the same environment. By virtue of built-in

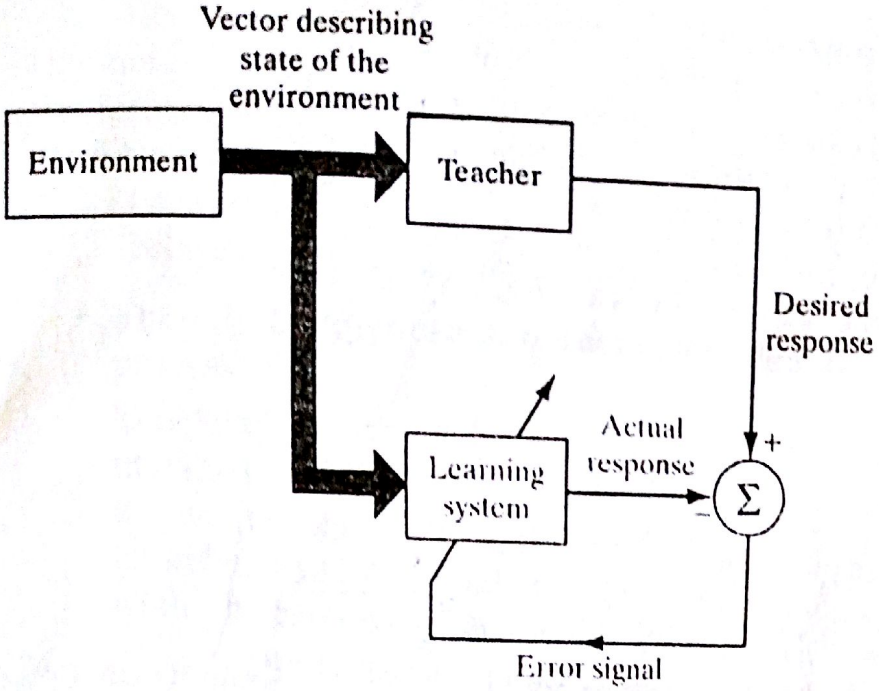

FIGURE 24 Block diagram of learning with a teacher; the part of the figure printed in red constitutes a feedback loop.

knowledge, the teacher is able to provide the neural network with a desired response for that training vector. Indeed, the desired response represents the "optimum" action to be performed by the neural network. The network parameters are adjusted under the combined influence of the training vector and the error signal. The *error signal* is defined as the difference between the desired response and the actual response of the network. This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network *emulate* the teacher; the emulation is presumed to be optimum in some statistical sense. In this way, knowledge of the environment available to the teacher is transferred to the neural network through training and stored in the form of "fixed" synaptic weights, representing *long-term memory*. When this condition is reached, we may then dispense with the teacher and let the neural network deal with the environment completely by itself.

The form of supervised learning we have just described is the basis of *error-correction learning*. From Fig. 24, we see that the supervised-learning process constitutes a closed-loop feedback system, but the unknown environment is outside the loop. As a performance measure for the system, we may think in terms of the *mean-square error*, or the *sum of squared errors* over the training sample, defined as a function of the free parameters (i.e., synaptic weights) of the system. This function may be visualized as a multidimensional *error-performance surface*, or simply *error surface*, with the free parameters as coordinates. The true error surface is *averaged* over all possible input–output examples. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher, the operating point has to move down successively toward a minimum point of the error surface; the minimum point may be a local minimum or a global minimum. A supervised learning system is able to do this with the useful information it has about the *gradient* of the error surface corresponding to the current behavior of the system. The gradient

of the error surface at any point is a vector that points in the direction of *steepest descent*. In fact, in the case of supervised learning from examples, the system may use an *instantaneous estimate* of the gradient vector, with the example indices presumed to be those of time. The use of such an estimate results in a motion of the operating point on the error surface that is typically in the form of a "random walk." Nevertheless, given an algorithm designed to minimize the cost function, an adequate set of input–output examples, and enough time in which to do the training, a supervised learning system is usually able to approximate an unknown input–output mapping reasonably well.

Learning without a Teacher

In supervised learning, the learning process takes place under the tutelage of a teacher. However, in the paradigm known as *learning without a teacher*, as the name implies, there is *no teacher* to oversee the learning process. That is to say, there are no labeled examples of the function to be learned by the network. Under this second paradigm, two subcategories are identified:

1. Reinforcement Learning

In *reinforcement learning*, the learning of an input–output mapping is performed through continued interaction with the environment in order to minimize a scalar index of performance. Figure 25 shows the block diagram of one form of a reinforcement-learning system built around a *critic* that converts a *primary reinforcement signal* received from the environment into a higher quality reinforcement signal called the *heuristic reinforcement signal*, both of which are scalar inputs (Barto et al., 1983). The system is designed to learn under *delayed reinforcement*, which means that the system observes a temporal sequence of stimuli also received from the environment, which eventually result in the generation of the heuristic reinforcement signal.

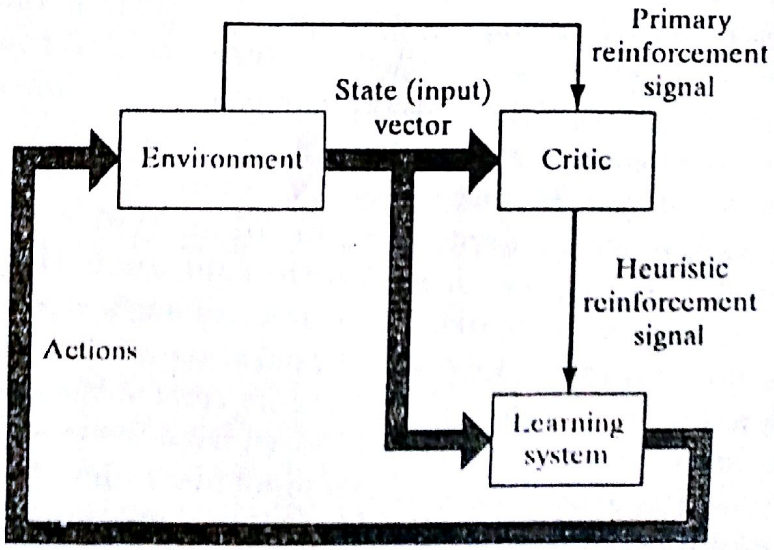

FIGURE 25 Block diagram of reinforcement learning; the learning system and the environment are both inside the feedback loop.

The goal of reinforcement learning is to minimize a *cost-to-go function*, defined as the expectation of the cumulative cost of *actions* taken over a sequence of steps instead of simply the immediate cost. It may turn out that certain actions taken earlier in that sequence of time steps are in fact the best determinants of overall system behavior. The function of the *learning system* is to *discover* these actions and feed them back to the environment.

Delayed-reinforcement learning is difficult to perform for two basic reasons:

- There is no teacher to provide a desired response at each step of the learning process.
- The delay incurred in the generation of the primary reinforcement signal implies that the learning machine must solve a *temporal credit assignment problem*. By this we mean that the learning machine must be able to assign credit and blame individually to each action in the sequence of time steps that led to the final outcome, while the primary reinforcement may only evaluate the outcome.

Notwithstanding these difficulties, delayed-reinforcement learning is appealing. It provides the basis for the learning system to interact with its environment, thereby developing the ability to learn to perform a prescribed task solely on the basis of the outcomes of its experience that result from the interaction.

2. Unsupervised Learning

In *unsupervised*, or *self-organized*, *learning*, there is no external teacher or critic to oversee the learning process, as indicated in Fig. 26. Rather, provision is made for a *task-independent measure* of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. For a specific task-independent measure, once the network has become tuned to the statistical regularities of the input data, the network develops the ability to form internal representations for encoding features of the input and thereby to create new classes automatically (Becker, 1991).

To perform unsupervised learning, we may use a competitive-learning rule. For example, we may use a neural network that consists of two layers—an input layer and a competitive layer. The input layer receives the available data. The competitive layer consists of neurons that compete with each other (in accordance with a learning rule) for the "opportunity" to respond to features contained in the input data. In its simplest form, the network operates in accordance with a "winner-takes-all" strategy. In such a strategy, the neuron with the greatest total input "wins" the competition and turns on; all the other neurons in the network then switch off.

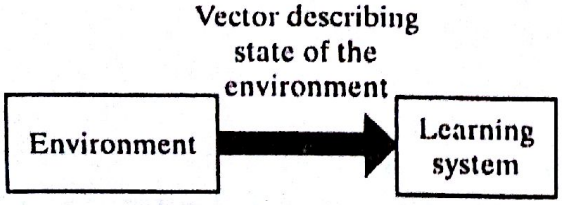

FIGURE 26 Block diagram of unsupervised learning.