CHAPTER 1

Introduction

1.1 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 (Marr, 1982; Levine, 1985; Churchland and Sejnowski, 1992). 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 may take days on a conventional computer.

For another example, consider the *sonar* of a bat. Sonar is an active echo-location system. In addition to providing information about how far away a target (e.g., a flying insect) is, a 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 (Suga, 1990a, b). 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 echo-locating 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 has great structure and the ability to build up its own rules through what we usually refer to as "experience." Indeed, experience is built up over time, with the most dramatic development (i.e., hard-wiring) of the human brain taking place during the first two years from birth; but the development continues well beyond that stage.

A "developing" neuron 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. Our interest in this book is confined largely to 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, which 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, 1996). 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 that new synaptic connections can grow.

Neural networks are also referred to in literature as *neurocomputers*, *connectionist networks*, *parallel distributed processors*, etc. Throughout the book we use the term "neural networks"; occasionally the term "neurocomputer" or "connectionist network" is used.

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 producing reasonable outputs for inputs not encountered during training (learning). These two information-processing capabilities make it possible for neural networks to solve complex (large-scale) problems that are currently 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 a human brain.

The use of neural networks offers 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 samples* or *task examples*. Each example consists of a unique *input signal* and a corresponding *desired 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 can 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, make 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, 1988b).

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 a 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 hard-wired design techniques. These two viewpoints are illustrated by the following two respective examples:

- In Anastasio (1993), linear system models of the vestibulo-ocular reflex are compared to neural network models based on *recurrent networks* that are described in Section 1.6 and discussed in detail in Chapter 15. The *vestibulo-ocular reflex (VOR)* 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 that is 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 (Anastasio, 1993).
- 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. A few researchers have built electronic chips that mimic the structure of the retina (Mahowald and Mead, 1989; Boahen and Ardreou, 1992; Boahen, 1996). 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.

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.

1.2 HUMAN BRAIN

The human nervous system may be viewed as a three-stage system, as depicted in the block diagram of Fig. 1.1 (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 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 (10^{-9} s) range, whereas neural events happen in the millisecond (10^{-3} s) 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 in use today is about 10^{-6} joules per operation per second (Faggin, 1991).

Synapses 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 signal and then

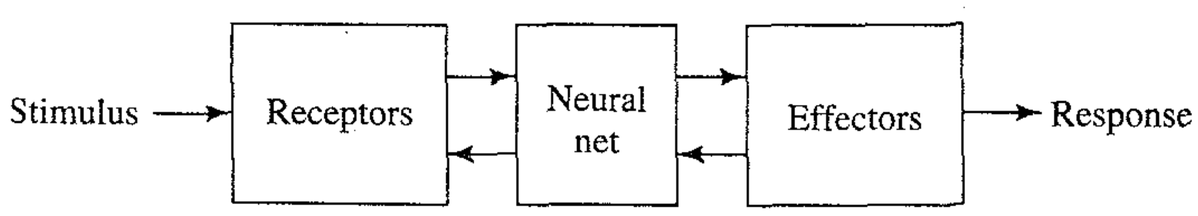


FIGURE 1.1 Block diagram representation of nervous system.

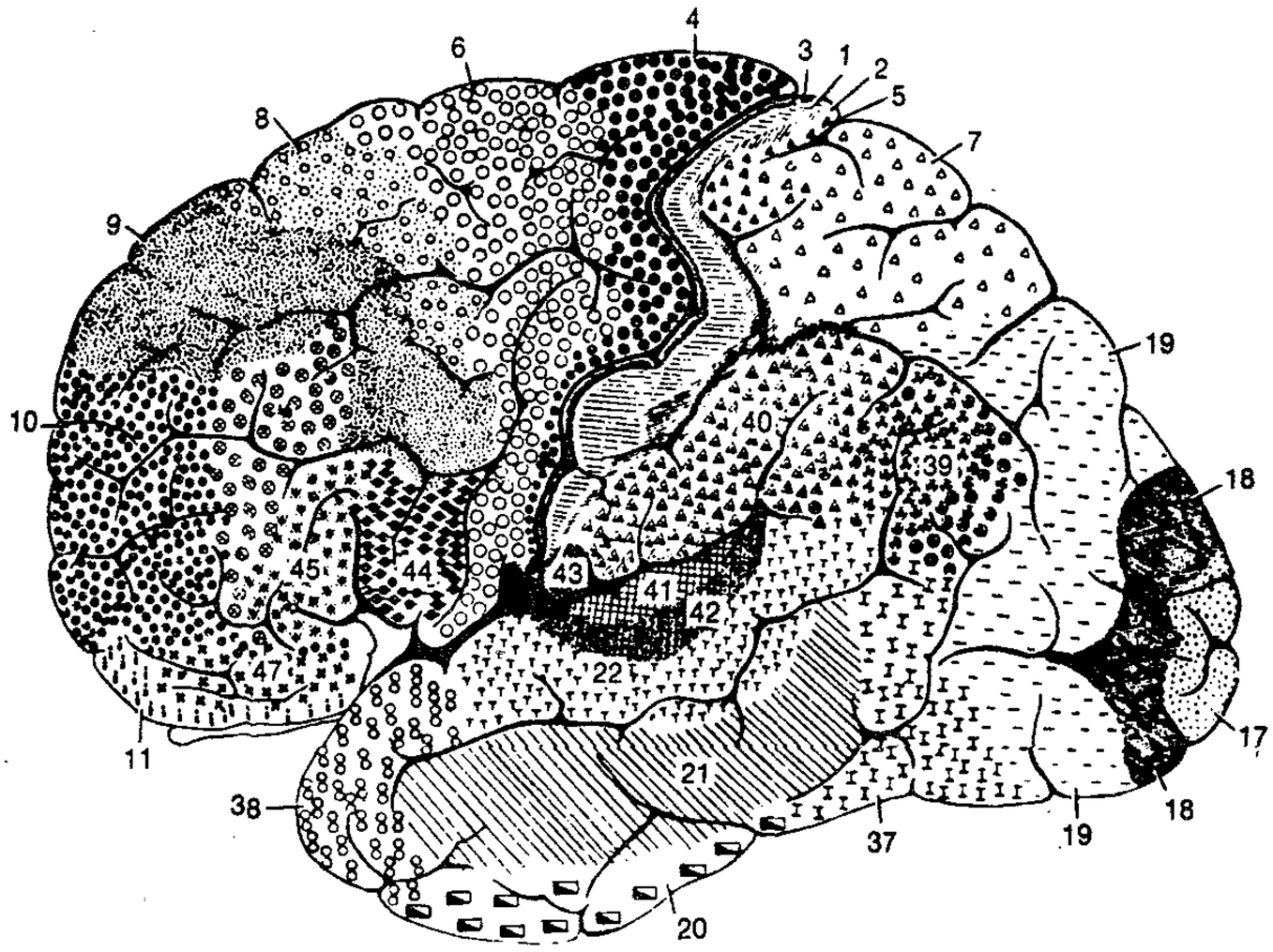


FIGURE 1.4 Cytoarchitectural map of the cerebral cortex. The different areas are identified by the thickness of their layers and types of cells within them. Some of the most important specific areas are as follows. Motor cortex: motor strip, area 4; premotor area, area 6; frontal eye fields, area 8. Somatosensory cortex: areas 3, 1, 2. Visual cortex: areas 17, 18, 19. Auditory cortex: area 41 and 42. (From A. Brodal, 1981; with permission of Oxford University Press.)

1.3 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. 1.5 shows the *model* of a neuron, which forms the basis for designing (artificial) neural networks. Here we identify three basic elements of the neuronal 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 refers to the neuron in question and the second subscript refers to the input end of the synapse to which the weight refers. Unlike 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 synapses 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.

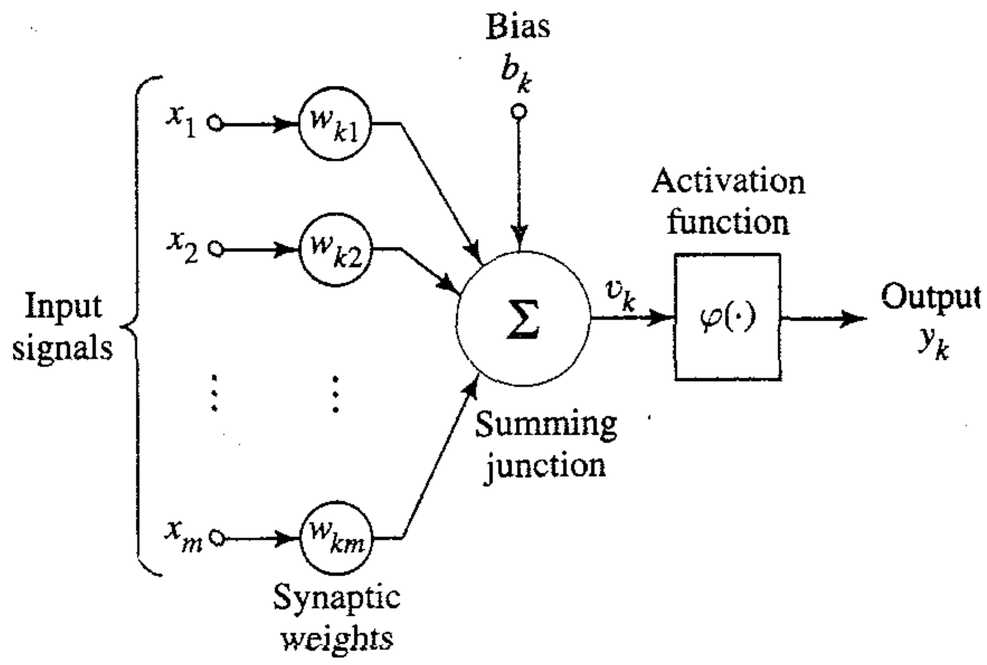


FIGURE 1.5 Nonlinear model of a neuron.

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 neuronal model of Fig. 1.5 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 a neuron k by writing the following pair of equations:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1.1}$$

and

$$y_k = \varphi(u_k + b_k) \tag{1.2}$$

where $x_1, x_2, \ldots, x_m$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{km}$ are the synaptic weights of neuron k; u_k 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. 1.5, as shown by

$$v_k = u_k + b_k \tag{1.3}$$

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. 1.6; hereafter the term "induced local field" is used. 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 artificial neuron k. We may account for its presence as in Eq. (1.2). Equivalently, we may formulate the combination of Eqs. (1.1) to (1.3) as follows:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{1.4}$$

and

$$y_k = \varphi(v_k) \tag{1.5}$$

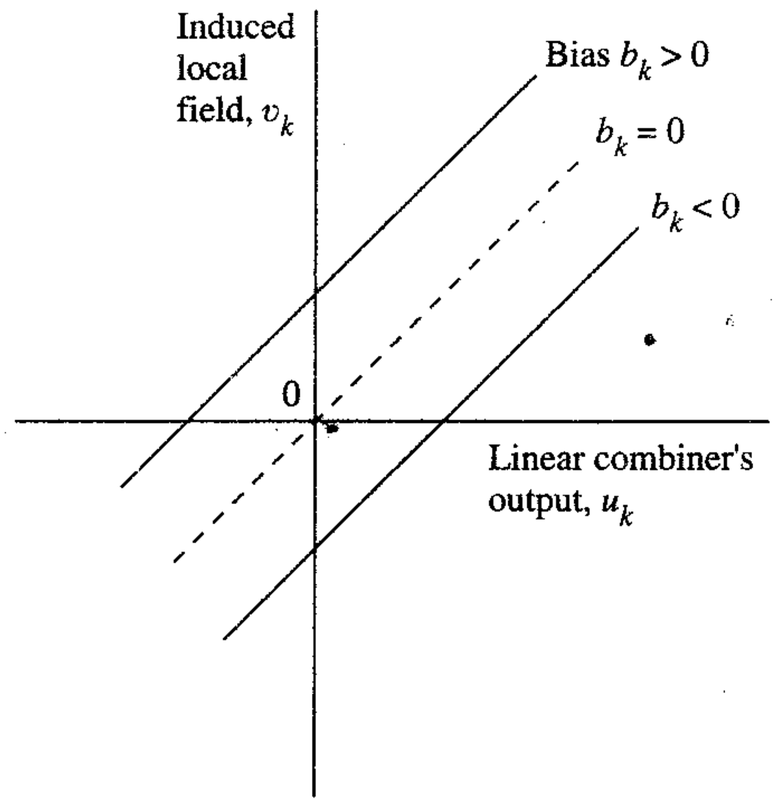


FIGURE 1.6 Affine transformation produced by the presence of a bias; note that $v_k = b_k$ at $u_k = 0$.

In Eq. (1.4) we have added a new synapse. Its input is

$$x_0 = +1 \tag{1.6}$$

and its weight is

$$w_{k0} = b_k \tag{1.7}$$

We may therefore reformulate the model of neruon k as in Fig. 1.7. 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. 1.5 and 1.7 are different in appearance, they are mathematically equivalent.

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. Here we identify three basic types of activation functions:

1. *Threshold Function*. For this type of activation function, described in Fig. 1.8a, we have

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \tag{1.8}$$

In engineering literature, 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

$$y_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases} \tag{1.9}$$

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

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{1.10}$$

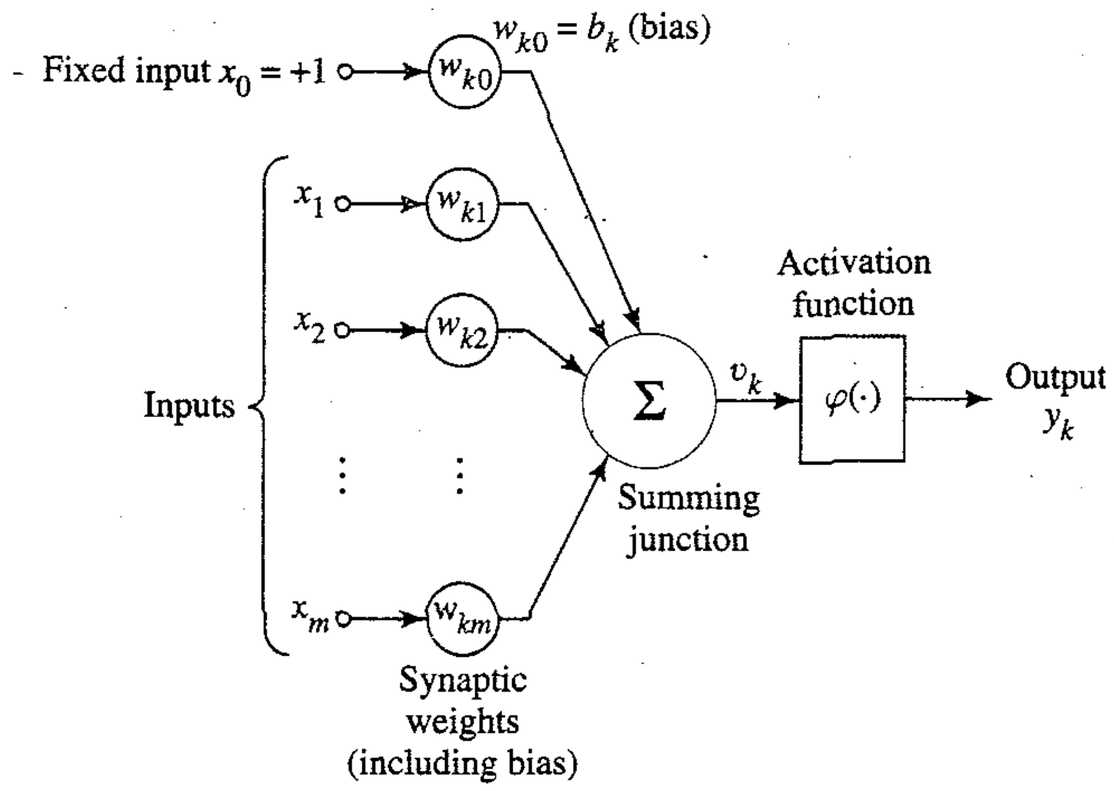


FIGURE 1.7 Another nonlinear model of a neuron.

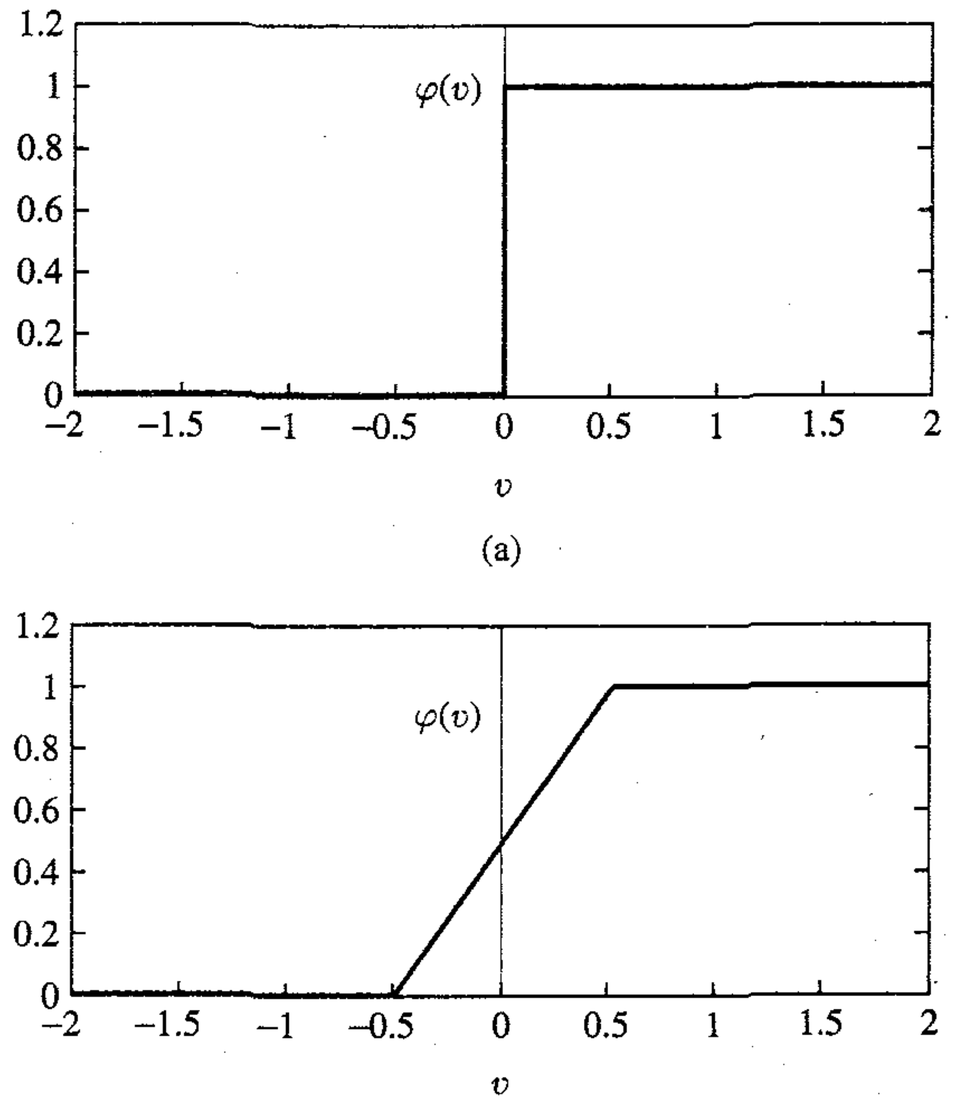


FIGURE 1.8 (a) Threshold function. (b) Piecewise-linear function. (c) Sigmoid function for varying slope parameter a.

Such a neuron is referred to in the literature 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. *Piecewise-Linear Function.* For the piecewise-linear function described in Fig. 1.8b we have

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{1.11}$$

where the amplification factor inside the linear region of operation is assumed to be unity. This form of an activation function may be viewed as an *approximation* to a nonlinear amplifier. The following two situations may be viewed as special forms of the piecewise-linear function:

- A *linear combiner* arises if the linear region of operation is maintained without running into saturation.
- The piecewise-linear function reduces to a *threshold function* if the amplification factor of the linear region is made infinitely large.

3. *Sigmoid Function.* The sigmoid function, whose graph is s-shaped, is by far the most common form of activation function used in the construction of artifical neural networks. It is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behavior.[3] An example of the sigmoid function is the *logistic function*,[4] defined by

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{1.12}$$

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. 1.8c. 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. (1.8), (1.11), and (1.12) 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 assumes an antisymmetric form with respect to the origin; that is, the activation function is an odd function of the induced local field. Specifically, the threshold function of Eq. (1.8) is now defined as

$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases} \tag{1.13}$$

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

$$\varphi(v) = \tanh(v) \tag{1.14}$$

Allowing an activation function of the sigmoid type to assume negative values as prescribed by Eq. (1.14) has analytic benefits (as shown in Chapter 4).

Stochastic Model of a Neuron

The neuronal model described in Fig. 1.7 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 neuronal 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

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

A standard choice for $P(v)$ is the sigmoid-shaped function (Little, 1974):

$$P(v) = \frac{1}{1 + \exp(-v/T)} \tag{1.15}$$

where T is a *pseudotemperature* that is used to control the noise level and therefore the uncertainty in firing. 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 $T \to 0$, the stochastic neuron described by Eq. (1.15) reduces to a noiseless (i.e., deterministic) form, namely the McCulloch–Pitts model.

1.4 NEURAL NETWORKS VIEWED AS DIRECTED GRAPHS

The *block diagram* of Fig. 1.5 or that of Fig. 1.7 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. 1.9a.
- *Activation links*, whose behavior is governed in general by a *nonlinear* input–output relation. This form of relationship is illustrated in Fig 1.9b, where $\varphi(\cdot)$ is the nonlinear activation function.

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. 1.9c 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.

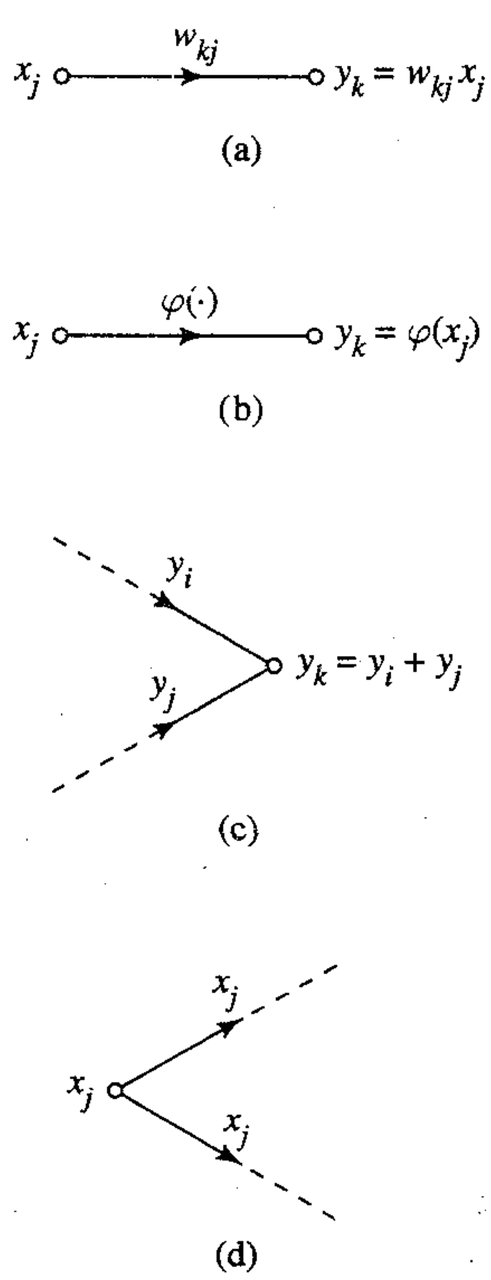


FIGURE 1.9 Illustrating basic rules for the construction of signal-flow graphs.

This third rule is illustrated in Fig. 1.9d for the case of *synaptic divergence* or *fan-out*.

For example, using these rules we may construct the signal-flow graph of Fig 1.10 as the model of a neuron, corresponding to the block diagram of Fig. 1.7. The representation shown in Fig. 1.10 is clearly simpler in appearance than that of Fig. 1.7, 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. 1.10 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.*
> 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.*

The state of the neuron may be defined in terms of its induced local field or its output signal.

A directed graph so defined 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.

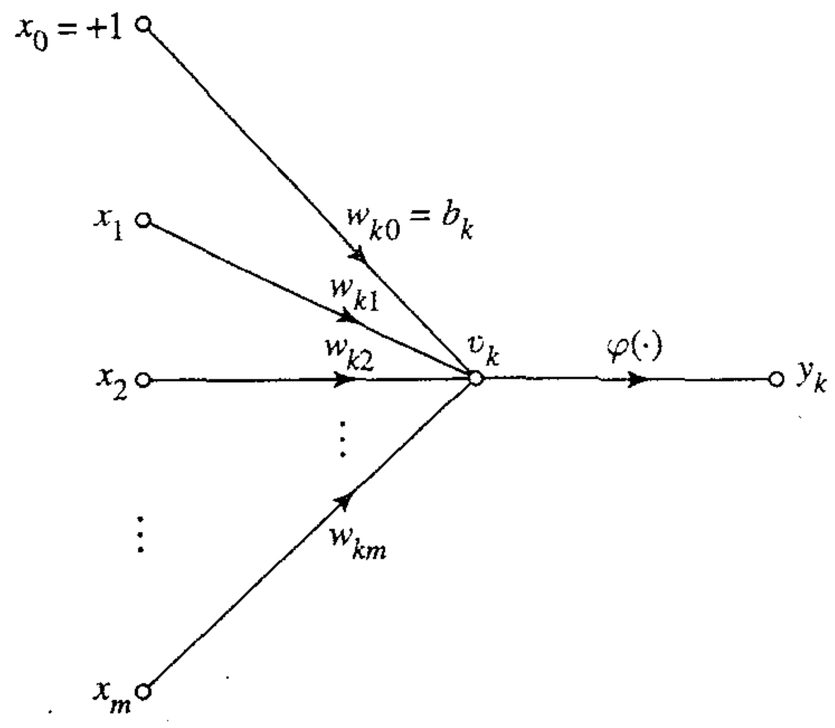


FIGURE 1.10 Signal-flow graph of a neuron.

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. 1.11 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 in Section 1.6.

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

- Block diagram, providing a functional description of the network.
- Signal-flow graph, providing a complete description of signal flow in the network.
- Architectural graph, describing the network layout.

1.5 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 1.12 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 1.12 we readily note the following input–output relationships:

$$y_k(n) = A[x_j'(n)] \tag{1.16}$$

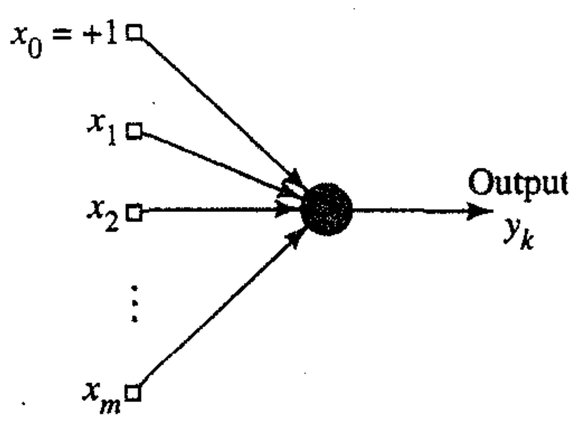


FIGURE 1.11 Architectural graph of a neuron.

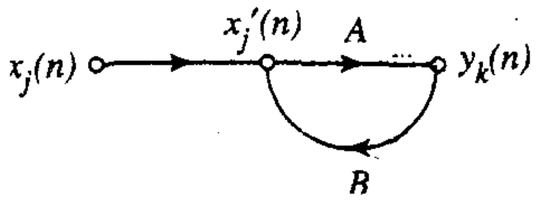


FIGURE 1.12 Signal-flow graph of a single-loop feedback system.

$$x_j'(n) = x_j(n) + B[y_k(n)] \tag{1.17}$$

where the square brackets are included to emphasize that A and B act as operators. Eliminating $x_j'(n)$ between Eqs. (1.16) and (1.17), we get

$$y_k(n) = \frac{A}{1 - AB}[x_j(n)] \tag{1.18}$$

We refer to $A/(1 - AB)$ 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 $BA \neq AB$.

Consider, for example, the single-loop feedback system shown in Fig. 1.13, 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

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

Using the binomial expansion for $(1 - wz^{-1})^{-1}$, we may rewrite the closed-loop operator of the system as

$$\frac{A}{1 - AB} = w\sum_{l=0}^{\infty} w^l z^{-l} \tag{1.19}$$

Hence, substituting Eq. (1.19) in (1.18), we get

$$y_k(n) = w\sum_{l=0}^{\infty} w^l z^{-l}[x_j(n)] \tag{1.20}$$

where again we have included square brackets to emphasize the fact that z^{-1} is an operator. In particular, from the definition of z^{-1} we have

$$z^{-l}[x_j(n)] = x_j(n - l) \tag{1.21}$$

where $x_j(n - l)$ is a sample of the input signal delayed by l time units. Accordingly, we may express the output signal $y_k(n)$ as an infinite weighted summation of present and past samples of the input signal $x_j(n)$, as shown by

$$y_k(n) = \sum_{l=0}^{\infty} w^{l+1} x_j(n - l) \tag{1.22}$$

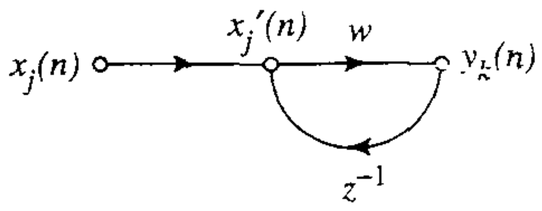


FIGURE 1.13 Signal-flow graph of a first-order, infinite-duration impulse response (IIR) filter.

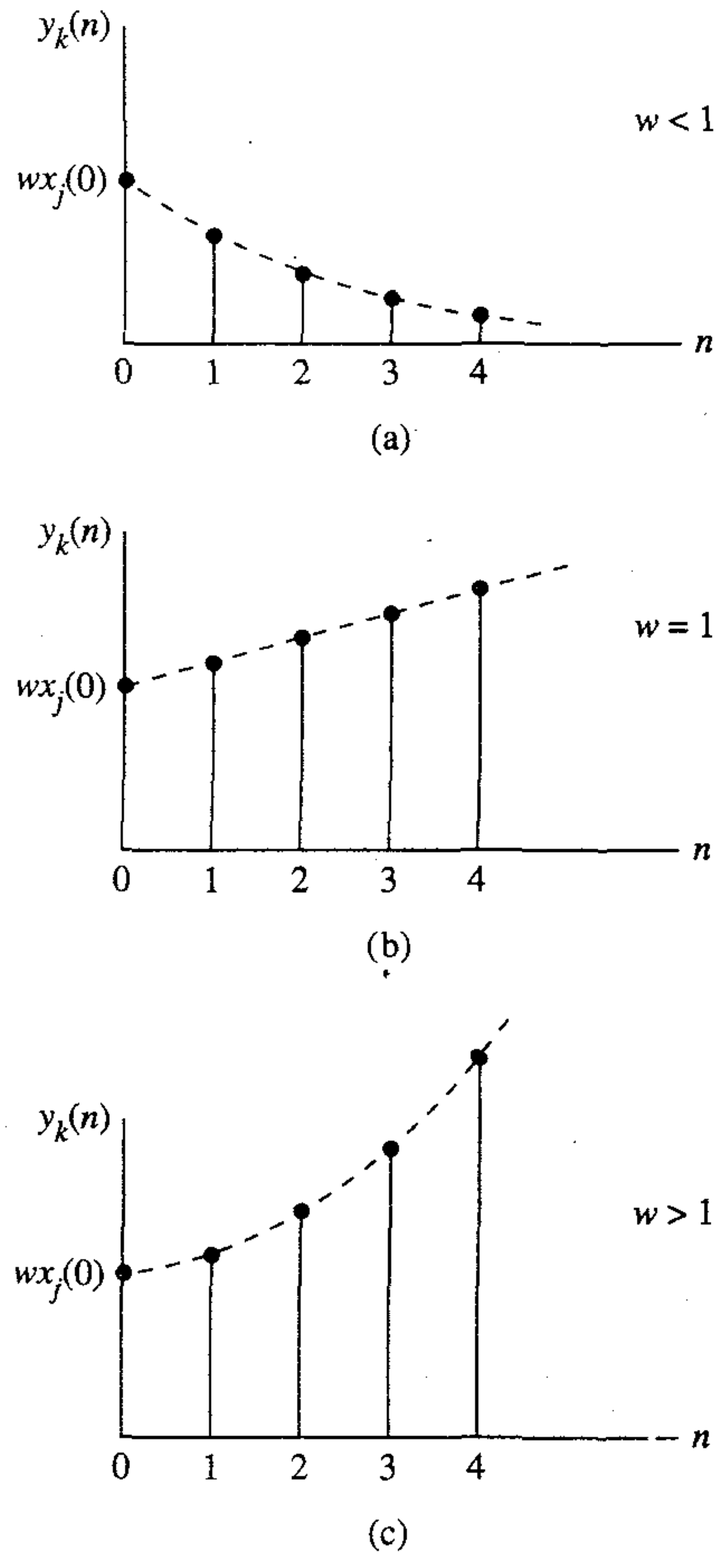


FIGURE 1.14 Time response of Fig. 1.13 for three different values of forward weight w. (a) Stable. (b) Linear divergence. (c) Exponential divergence.

We now see clearly that the dynamic behavior of the system is controlled by the weight w. In particular, we may distinguish two specific cases:

1. $|w| < 1$, for which the output signal $y_k(n)$ is exponentially *convergent*; that is, the system is *stable*. This is illustrated in Fig. 1.14a for a positive w.
2. $|w| \geq 1$, for which the output signal $y_k(n)$ is *divergent*; that is, the system is *unstable*. If $|w| = 1$ the divergence is linear as in Fig. 1.14b, and if $|w| > 1$ the divergence is exponential as in Fig 1.14c.

Stability features prominently in the study of feedback systems.

The case of $|w| < 1$ corresponds to a system with *infinite memory* in the sense that the output of the system depends on samples of the input extending into the infinite past. Moreover, the memory is *fading* in that the influence of a past sample is reduced exponentially with time n.

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

1.6 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 the next chapter, and the development of different learning algorithms is taken up in subsequent chapters of the book. In this section we focus our attention on network architectures (structures).

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

1. 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 onto an *output layer* of neurons (computation nodes), but not vice versa. In other words, this network is strictly a *feedforward* or *acyclic* type. It is illustrated in Fig. 1.15 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.

2. 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 function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or

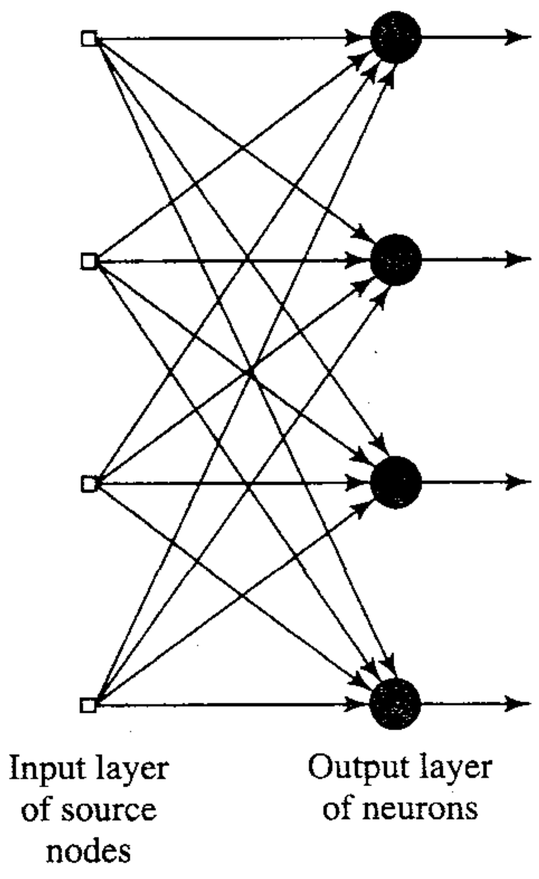


FIGURE 1.15 Feedforward or acyclic network with a single layer of neurons.

more hidden layers, the network is enabled to extract higher-order statistics. 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 ability of hidden neurons to extract higher-order statistics is particularly valuable when the size of the input layer is large.

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. 1.16 illustrates the layout of a multilayer feedforward neural network for the case of a single hidden layer. For brevity the network in Fig. 1.16 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.

The neural network in Fig. 1.16 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*.

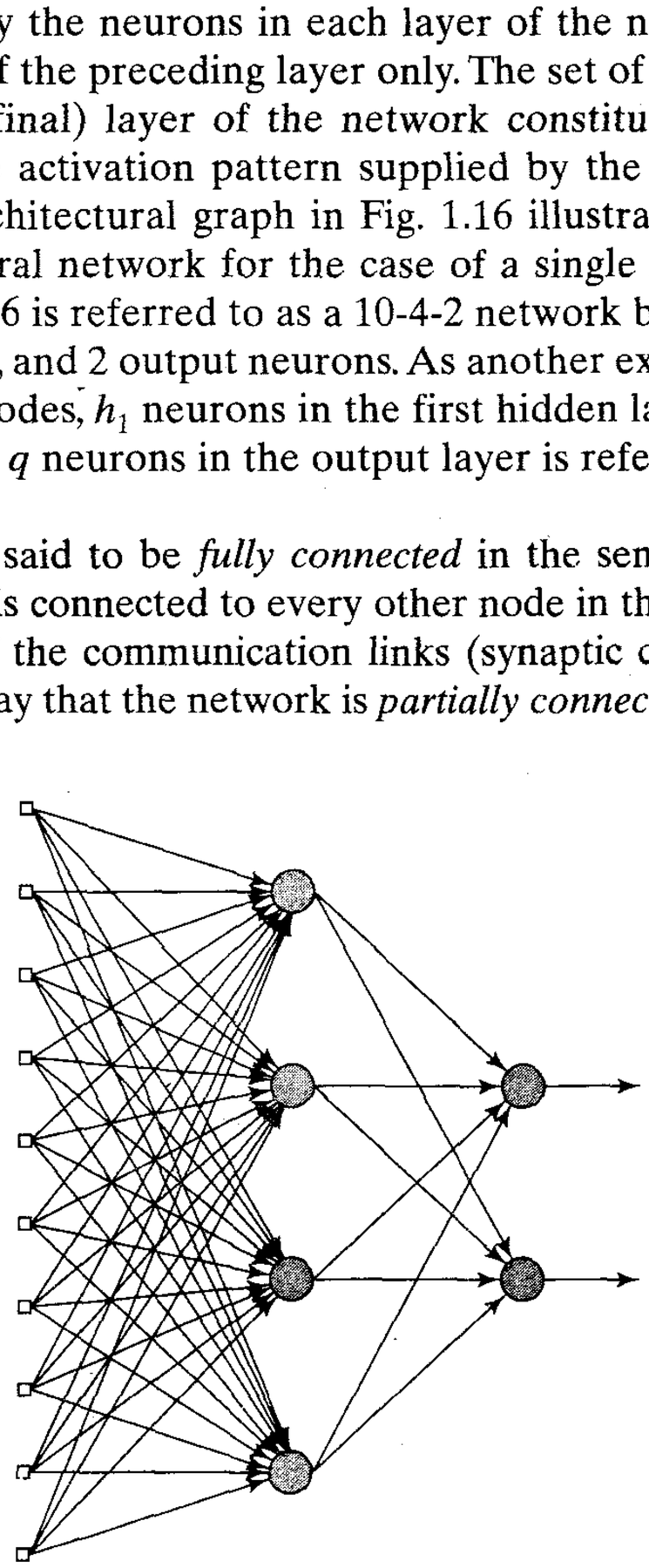


FIGURE 1.16 Fully connected feedforward or acyclic network with one hidden layer and one output layer.

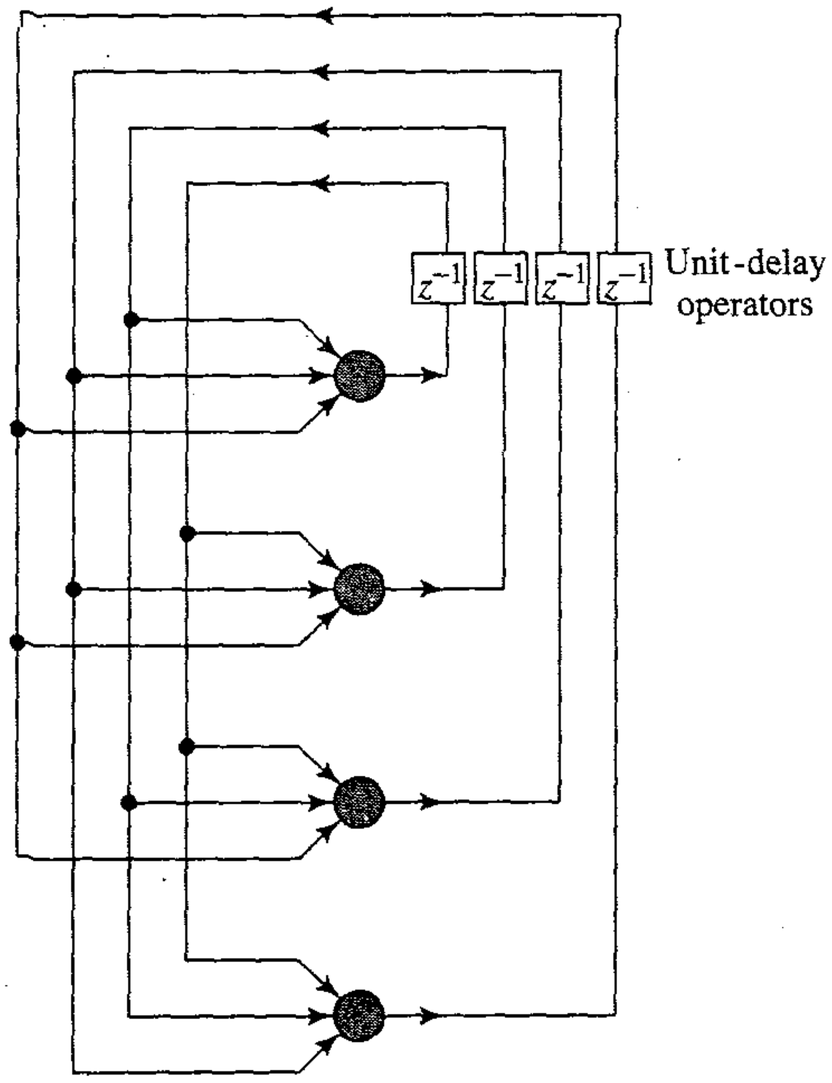


FIGURE 1.17 Recurrent network with no self-feedback loops and no hidden neurons.

3. 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. 1.17. 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. 1.17 also has *no* hidden neurons. In Fig. 1.18 we illustrate another class of recurrent networks with hidden neurons. The feedback connections shown in Fig. 1.18 originate from the hidden neurons as well as from the output neurons.

The presence of feedback loops, whether in the recurrent structure of Fig. 1.17 or that of Fig. 1.18, 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-delay elements* (denoted by z^{-1}), which result in a nonlinear dynamical behavior, assuming that the neural network contains nonlinear units.

1.7 KNOWLEDGE REPRESENTATION

In Section 1.1 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.*

NOTES AND REFERENCES

1. This definition of a neural network is adapted from Aleksander and Morton (1990).
2. For a complementary perspective on neural networks with emphasis on neural modeling, cognition, and neurophysiological considerations, see Anderson (1995). For a highly readable account of the computational aspects of the brain, see Churchland and Sejnowski (1992). For more detailed descriptions of neural mechanisms and the human brain, see Kandel and Schwartz (1991), Shepherd (1990a, b), Koch and Segev (1989), Kuffler et al. (1984), and Freeman (1975).
3. For a thorough account of sigmoid functions and related issues, see Menon et al. (1996).
4. The logistic function, or more precisely the *logistic distribution function*, derives its name from a transcendental "law of logistic growth" that resulted in a huge literature. Measured in appropriate units, all growth processes were supposed to be represented by the logistic distribution function

$$F(t) = \frac{1}{1 + e^{\alpha t - \beta}}$$

where t represents time, and α and β are constants. It turned out, however, that not only the logistic distribution but also the Gaussian and other distributions can apply to the same data with the same or better goodness of fit (Feller, 1968).

5. According to Kuffler et al. (1984), the term "receptive field" was coined originally by Sherrington (1906) and reintroduced by Hartline (1940). In the context of a visual system, the receptive field of a neuron refers to the restricted area on the retinal surface, which influences the discharges of that neuron due to light.
6. It appears that the weight-sharing technique was originally described in Rumelhart et al. (1986b).
7. The historical notes presented here are largely (but not exclusively) based on the following sources: (1) the paper by Saarinen et al. (1992); (2) the chapter contribution by Rall (1990); (3) the paper by Widrow and Lehr (1990); (4) the papers by Cowan (1990) and Cowan and Sharp (1988); (5) the paper by Grossberg (1988c); (6) the two-volume book on neurocomputing (Anderson et al., 1990; Anderson and Rosenfeld, 1988); (7) the chapter contribution of Selfridge et al. (1988); (8) the collection of papers by von Neumann on computing and computer theory (Aspray and Burks, 1986); (9) the handbook on brain theory and neural networks edited by Arbib (1995); (10) Chapter 1 of the book by Russell and Norvig (1995); and (11) the article by Taylor (1997).

PROBLEMS

Models of a neuron

1.1 An example of the logistic function is defined by

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

whose limiting values are 0 and 1. Show that the derivative of $\varphi(v)$ with respect to v is given by

$$\frac{d\varphi}{dv} = a\varphi(v)[1 - \varphi(v)]$$

What is the value of this derivative at the origin?

1.2 An odd sigmoid function is defined by

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

$$= \tanh\left(\frac{av}{2}\right)$$

where tanh denotes a hyperbolic tangent. The limiting values of this second sigmoid function are -1 and $+1$. Show that the derivative of $\varphi(v)$ with respect to v is given by

$$\frac{d\varphi}{dv} = \frac{a}{2}[1 - \varphi^2(v)]$$

What is the value of this derivative at the origin? Suppose that the slope parameter a is made infinitely large. What is the resulting form of $\varphi(v)$?

1.3 Yet another odd sigmoid function is the algebraic sigmoid:

$$\varphi(v) = \frac{v}{\sqrt{1 + v^2}}$$

whose limiting values are -1 and $+1$. Show that the derivative of $\varphi(v)$ with respect to v is given by

$$\frac{d\varphi}{dv} = \frac{\varphi^3(v)}{v^3}$$

What is the value of this derivative at the origin?

1.4 Consider the following two functions:

(i) $\varphi(v) = \dfrac{1}{\sqrt{2\pi}} \displaystyle\int_{-\infty}^{v} \exp\left(-\frac{x^2}{2}\right) dx$

(ii) $\varphi(v) = \dfrac{2}{\pi} \tan^{-1}(v)$

Explain why both of these functions fit the requirements of a sigmoid function. How do these two functions differ from each other?

1.5 Which of the five sigmoid functions described in Problems 1.1 to 1.4 would qualify as a cumulative (probability) distribution function? Justify your answer.

1.6 Consider the pseudolinear activation function $\varphi(v)$ shown in Fig. P1.6.

(a) Formulate $\varphi(v)$ as a function of v.

(b) What happens to $\varphi(v)$ if a is allowed to approach zero?

1.7 Repeat Problem 1.6 for the pseudolinear activation function $\varphi(v)$ shown in Fig. P1.7.

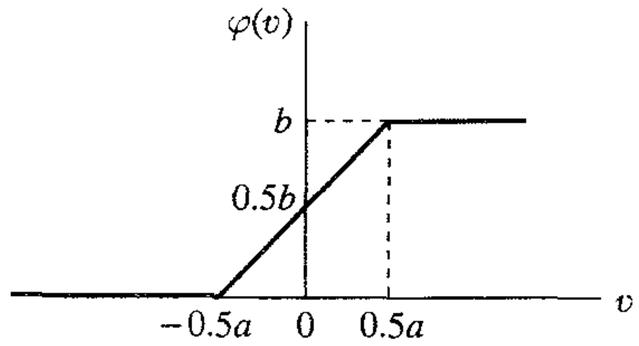


FIGURE P1.6

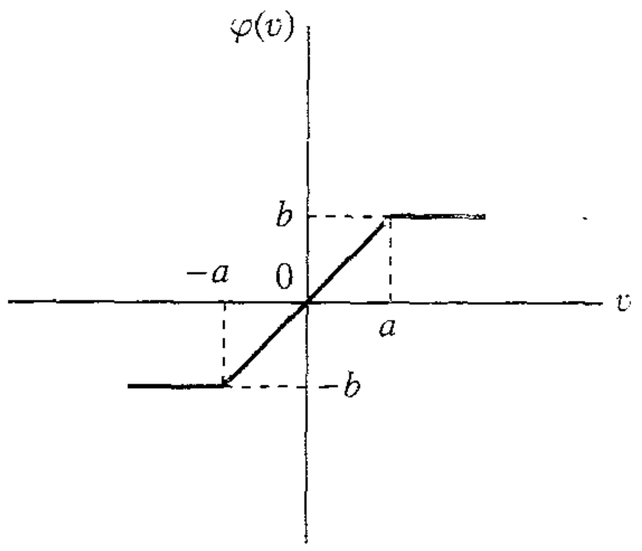


FIGURE P1.7

1.8 A neuron has an activation function $\varphi(v)$ defined by the logistic function of Problem 1.1, where v is the induced local field, and the slope parameter a is available for adjustment. Let $x_1, x_2, ..., x_m$ denote the input signals applied to the source nodes of the neuron, and b denote the bias. For convenience of presentation, we would like to absorb the slope parameter a in the induced local field v by writing

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

How would you modify the inputs $x_1, x_2, ..., x_m$ to produce the same output as before? Justify your answer.

1.9 A neuron j receives inputs from four other neurons whose activity levels are 10, −20, 4, and −2. The respective synaptic weights of neuron j are 0.8, 0.2, −1.0, and −0.9. Calculate the output of neuron j for the following two situations:

(a) The neuron is linear.

(b) The neuron is represented by a McCulloch–Pitts model.

Assume that the bias applied to the neuron is zero.

1.10 Repeat Problem 1.9 for a neuron model based on the logistic function

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

1.11 **(a)** Show that the McCulloch–Pitts formal model of a neuron may be approximated by a sigmoidal neuron (i.e., neuron using a sigmoid activation function with large synaptic weights.

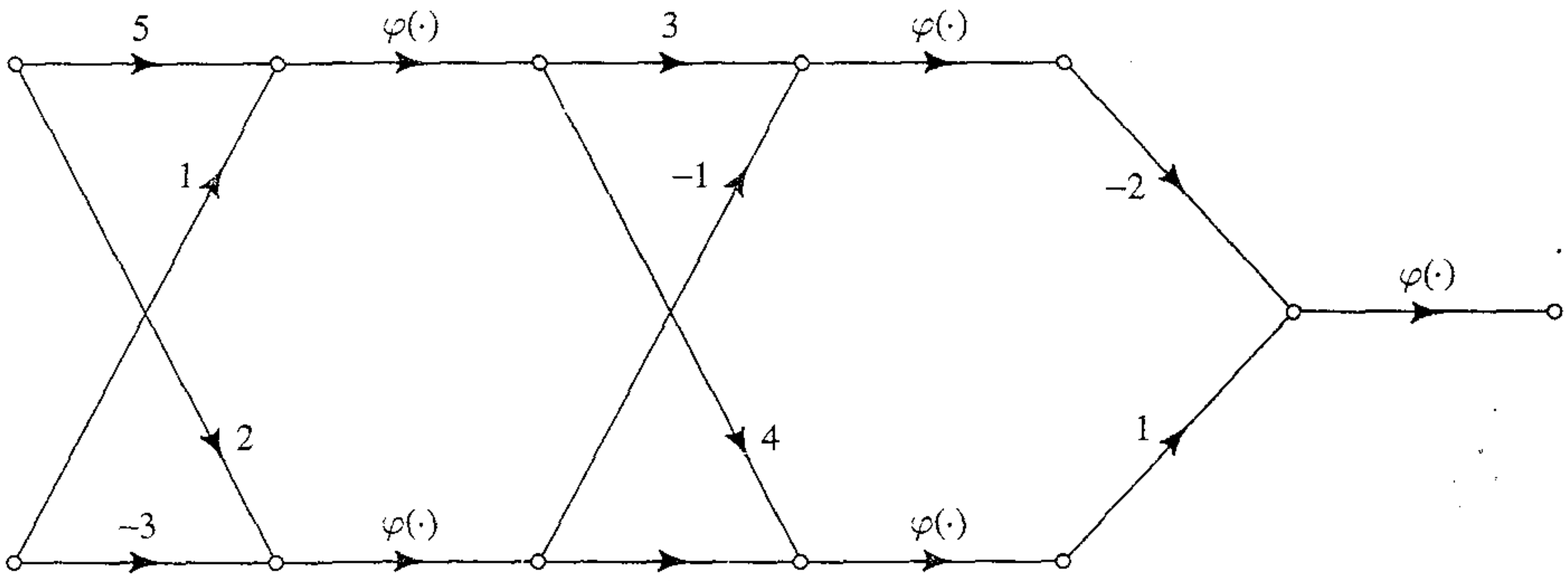
(b) Show that a linear neuron may be approximated by a sigmoidal neuron with small synaptic weights.

Network architectures

1.12 A fully connected feedforward network has 10 source nodes, 2 hidden layers, one with 4 neurons and the other with 3 neurons, and a single output neuron. Construct an architectural graph of this network.

1.13 **(a)** Figure P1.13 shows the signal-flow graph of a 2-2-2-1 feedforward network. The function $\varphi(\cdot)$ denotes a logistic function. Write the input–output mapping defined by this network.

(b) Suppose that the output neuron in the signal-flow graph of Fig. P1.13 operates in its linear region. Write the input–output mapping defined by this new network.



6

FIGURE P1.13

1.14 The network described in Fig. P1.13 has no biases. Suppose that biases equal to -1 and $+1$ are applied to the top and bottom neurons of the first hidden layer, and biases equal to $+1$ and -2 are applied to the top and bottom neurons of the second hidden layer. Write the new form of the input–output mapping defined by the network.

1.15 Consider a multilayer feedforward network, all the neurons of which operate in their linear regions. Justify the statement that such a network is equivalent to a single-layer feedforward network.

1.16 Construct a fully recurrent network with 5 neurons, but with no self-feedback.

1.17 Figure P1.17 shows the signal-flow graph of a recurrent network made up of two neurons. Write the nonlinear difference equation that defines the evolution of $x_1(n)$ or that of $x_2(n)$. These two variables define the outputs of the top and bottom neurons, respectively. What is the order of this equation?

1.18 Figure P1.18 shows the signal-flow graph of a recurrent network consisting of two neurons with self-feedback. Write the coupled system of two first-order nonlinear difference equations that describe the operation of the system.

1.19 A recurrent network has 3 source nodes, 2 hidden neurons, and 4 output neurons. Construct an architectural graph that describes such a network.

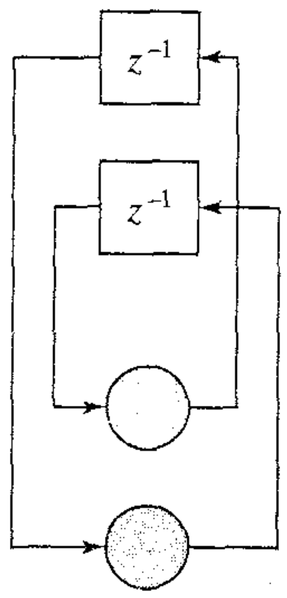


FIGURE P1.17

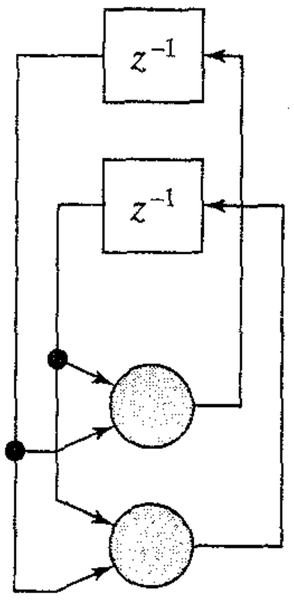


FIGURE P1.18

Knowledge representation

1.20 A useful form of preprocessing is based on the *autoregressive* (AR) *model* described by the difference equation (for real-valued data)

$$y(n) = w_1 y(n-1) + w_2 y(n-2) + \cdots + w_M y(n-M) + v(n)$$

where $y(n)$ is the model output; $v(n)$ is a sample drawn from a white-noise process of zero mean and some prescribed variance; $w_1, w_2, \ldots, w_M$ are the AR model coefficients; and M is the model order. Show that the use of this model provides two forms of geometric invariance: (a) scale, and (b) time translation. How could these two invariances be used in neural networks?

1.21 Let $\mathbf{x}$ be an input vector, and $\mathbf{s}(\alpha, \mathbf{x})$ be a transformation operator acting on $\mathbf{x}$ and depending on some parameter α. The operator $\mathbf{s}(\alpha, \mathbf{x})$ satisfies two requirements:

- $\mathbf{s}(0, \mathbf{x}) = \mathbf{x}$
- $\mathbf{s}(\alpha, \mathbf{x})$ is differentiable with respect to α.

The *tangent vector* is defined by the partial derivative $\partial \mathbf{s}(\alpha, \mathbf{x})/\partial \alpha$ (Simard et al., 1992).

Suppose that $\mathbf{x}$ represents an image, and α is a rotation parameter. How would you compute the tangent vector for the case when α is small? The tangent vector is locally invariant with respect to rotation of the original image; why?