CHAPTER 2

Learning Processes

2.1 INTRODUCTION

The property that is of primary significance for a neural network is the ability of the network to *learn* from its environment, and to *improve* its performance through learning. The improvement in performance takes place over time in accordance with some prescribed measure. A neural network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias levels. Ideally, the network becomes more knowledgeable about its environment after each iteration of the learning process.

There are too many activities associated with the notion of "learning" to justify defining it in a precise manner. Moreover, the process of learning is a matter of viewpoint, which makes it all the more difficult to agree on a precise definition of the term. For example, learning as viewed by a psychologist is quite different from learning in a classroom sense. Recognizing that our particular interest is in neural networks, we use a definition of learning that is adapted from Mendel and McClaren (1970).

We define learning in the context of neural networks as:

> *Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place.*

This definition of the learning process implies the following sequence of events:

1. The neural network is *stimulated* by an environment.
2. The neural network *undergoes changes* in its free parameters as a result of this stimulation.
3. The neural network *responds in a new way* to the environment because of the changes that have occurred in its internal structure.

A prescribed set of well-defined rules for the solution of a learning problem is called a *learning algorithm.*[1] As one would expect, there is no unique learning algorithm for the design of neural networks. Rather, we have a "kit of tools" represented by a diverse variety of learning algorithms, each of which offers advantages of its own. Basically, learning algorithms differ from each other in the way in which the adjust-

ment to a synaptic weight of a neuron is formulated. Another factor to be considered is the manner in which a neural network (learning machine), made up of a set of interconnected neurons, relates to its environment. In this latter context we speak of a *learning paradigm* that refers to a *model* of the environment in which the neural network operates.

Organization of the Chapter

The chapter is organized in four interrelated parts. In the first part, consisting of Sections 2.2 through 2.6, we discuss five basic learning rules: error-correction learning, memory-based learning, Hebbian learning, competitive learning, and Boltzmann learning. Error-correction learning is rooted in optimum filtering. Memory-based learning operates by memorizing the training data explicitly. Hebbian learning and competitive learning are both inspired by neurobiological considerations. Boltzmann learning is different because it is based on ideas borrowed from statistical mechanics.

The second part of the chapter explores learning paradigms. Section 2.7 discusses the credit-assignment problem, which is basic to the learning process. Sections 2.8 and 2.9 present overviews of the two fundamental learning paradigms: (1) learning *with* a teacher, and (2) learning *without* a teacher.

The third part of the chapter, consisting of Sections 2.10 through 2.12, examines the issues of learning tasks, memory, and adaptation.

The final part of the chapter, consisting of Sections 2.13 through 2.15, deals with probabilistic and statistical aspects of the learning process. Section 2.13 discusses the bias/variance dilemma. Section 2.14 discusses statistical learning theory, based on the notion of VC-dimension that provides a measure of machine capacity. Section 2.14 introduces another important concept: probably approximately correct (PAC) learning, which provides a conservative model for the learning process.

The chapter concludes with some final remarks in Section 2.16.

2.2 ERROR-CORRECTION LEARNING

To illustrate our first learning rule, consider the simple case of a neuron k constituting the only computational node in the output layer of a feedforward neural network, as depicted in Fig. 2.1a. Neuron k is driven by a *signal vector* $\mathbf{x}(n)$ produced by one or more layers of hidden neurons, which are themselves driven by an input vector (stimulus) applied to the source nodes (i.e., input layer) of the neural network. The argument n denotes discrete time, or more precisely, the time step of an iterative process involved in adjusting the synaptic weights of neuron k. The *output signal* of neuron k is denoted by $y_k(n)$. This output signal, representing the only output of the neural network, is compared to a *desired response* or *target output,* denoted by $d_k(n)$. Consequently, an *error signal,* denoted by $e_k(n)$, is produced. By definition, we thus have

$$e_k(n) = d_k(n) - y_k(n) \tag{2.1}$$

The error signal $e_k(n)$ actuates a *control mechanism,* the purpose of which is to apply a sequence of corrective adjustments to the synaptic weights of neuron k. The corrective adjustments are designed to make the output signal $y_k(n)$ come closer to the desired

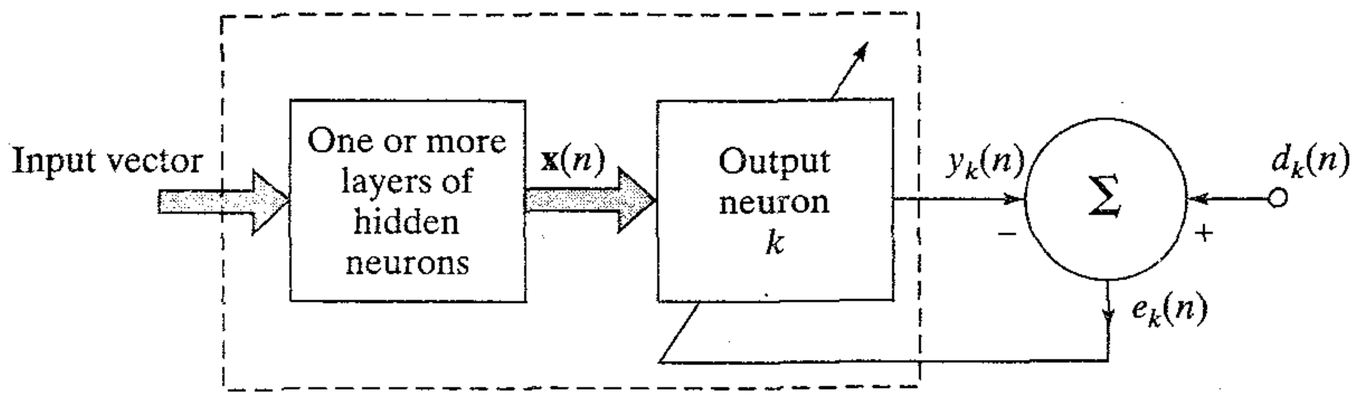


FIGURE 2.1 Illustrating error-correction learning.

response $d_k(n)$ in a step-by-step manner. This objective is achieved by minimizing a *cost function* or *index of performance*, $\mathscr{E}(n)$, defined in terms of the error signal $e_k(n)$ as:

$$\mathscr{E}(n) = \frac{1}{2} e_k^2(n) \tag{2.2}$$

That is, $\mathscr{E}(n)$ is the *instantaneous value of the error energy*. The step-by-step adjustments to the synaptic weights of neuron k are continued until the system reaches a *steady state* (i.e., the synaptic weights are essentially stabilized). At that point the learning process is terminated.

The learning process described herein is obviously referred to as *error-correction learning*. In particular, minimization of the cost function $\mathscr{E}(n)$ leads to a learning rule commonly referred to as the *delta rule* or *Widrow–Hoff rule*, named in honor of its originators (Widrow and Hoff, 1960). Let $w_{kj}(n)$ denote the value of synaptic weight w_{kj} of neuron k excited by element $x_j(n)$ of the signal vector $\mathbf{x}(n)$ at time step n. According to the delta rule, the adjustment $\Delta w_{kj}(n)$ applied to the synaptic weight w_{kj} at time step n is defined by

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{2.3}$$

where η is a positive constant that determines the *rate of learning* as we proceed from one step in the learning process to another. It is therefore natural that we refer to η as the *learning-rate parameter*. In other words, the delta rule may be stated as:

The adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse in question.

Keep in mind that the delta rule, as stated herein, presumes that the error signal is *directly measurable*. For this measurement to be feasible we clearly need a supply of desired response from some external source, which is directly accessible to neuron k. In other words, neuron k is *visible* to the outside world, as depicted in Fig. 2.1a. From this figure we also observe that error-correction learning is in fact *local* in nature. This is merely saying that the synaptic adjustments made by the delta rule are localized around neuron k.

Having computed the synaptic adjustment $\Delta w_{kj}(n)$, the updated value of synaptic weight w_{kj} is determined by

$$w_{kj}(n + 1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{2.4}$$

In effect, $w_{kj}(n)$ and $w_{kj}(n + 1)$ may be viewed as the *old* and *new* values of synaptic weight w_{kj}, respectively. In computational terms we may also write

$$w_{kj}(n) = z^{-1}[w_{kj}(n + 1)] \tag{2.5}$$

where z^{-1} is the *unit-delay operator*. That is, z^{-1} represents a *storage element*.

Figure 2.1b shows a signal-flow graph representation of the error-correction learning process, focusing on the activity surrounding neuron k. The input signal x_j and induced local field v_k of neuron k are referred to as the *presynaptic* and *postsynaptic signals* of the jth synapse of neuron k, respectively. From Fig. 2.1b we see that error-correction learning is an example of a *closed-loop feedback system*. From control theory we know that the stability of such a system is determined by those parameters that constitute the feedback loops of the system. In our case we only have a single feedback loop, and one of those parameters of particular interest is the learning-rate parameter η. It is therefore important that η is carefully selected to ensure that the stability or convergence of the iterative learning process is achieved. The choice of η also has a profound influence on the accuracy and other aspects of the learning process. In short, the learning-rate parameter η plays a key role in determining the performance of error-correction learning in practice.

Error-correction learning is discussed in much greater detail in Chapter 3, which discusses single-layer feedforward networks and in Chapter 4, which details multilayer feedforward networks.

2.3 MEMORY-BASED LEARNING

In *memory-based learning*, all (or most) of the past experiences are explicitly stored in a large memory of correctly classified input–output examples: $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ denotes an input vector and d_i denotes the corresponding desired response. Without loss of generality, we have restricted the desired response to be a scalar. For example, in a binary pattern classification problem there are two classes/hypotheses, denoted by $\mathscr{C}_1$ and $\mathscr{C}_2$, to be considered. In this example, the desired response d_i takes the value 0 (or -1) for class $\mathscr{C}_1$ and the value 1 for class $\mathscr{C}_2$. When classification of a test vector $\mathbf{x}_{\text{test}}$ (not seen before) is required, the algorithm responds by retrieving and analyzing the training data in a "local neighborhood" of $\mathbf{x}_{\text{test}}$.

All memory-based learning algorithms involve two essential ingredients:

- Criterion used for defining the local neighborhood of the test vector $\mathbf{x}_{test}$.
- Learning rule applied to the training examples in the local neighborhood of $\mathbf{x}_{test}$.

The algorithms differ from each other in the way in which these two ingredients are defined.

In a simple yet effective type of memory-based learning known as the *nearest neighbor rule,*[2] the local neighborhood is defined as the training example that lies in the immediate neighborhood of the test vector $\mathbf{x}_{test}$. In particular, the vector

$$\mathbf{x}'_N \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \tag{2.6}$$

is said to be the nearest neighbor of $\mathbf{x}_{test}$ if

$$\min_i d(\mathbf{x}_i, \mathbf{x}_{test}) = d(\mathbf{x}'_N, \mathbf{x}_{test}) \tag{2.7}$$

where $d(\mathbf{x}_i, \mathbf{x}_{test})$ is the Euclidean distance between the vectors $\mathbf{x}_i$ and $\mathbf{x}_{test}$. The class associated with the minimum distance, that is, vector $\mathbf{x}'_N$, is reported as the classification of $\mathbf{x}_{test}$. This rule is independent of the underlying distribution responsible for generating the training examples.

Cover and Hart (1967) have formally studied the nearest neighbor rule as a tool for pattern classification. The analysis presented therein is based on two assumptions:

- The classified examples $(\mathbf{x}_i, d_i)$ are *independently and identically distributed (iid),* according to the joint probability distribution of the example $(\mathbf{x}, d)$.
- The sample size N is infinitely large.

Under these two assumptions, it is shown that the probability of classification error incurred by the nearest neighbor rule is bounded above by twice the *Bayes probability of error,* that is, the minimum probability of error over all decision rules. Bayes probability of error is discussed in Chapter 3. In this sense, it may be said that half the classification information in a training set of infinite size is contained in the nearest neighbor, which is a surprising result.

A variant of the nearest neighbor classifier is the *k-nearest neighbor classifier,* which proceeds as follows:

- Identify the k classified patterns that lie nearest to the test vector $\mathbf{x}_{test}$ for some integer k.
- Assign $\mathbf{x}_{test}$ to the class (hypothesis) that is most frequently represented in the k nearest neighbors to $\mathbf{x}_{test}$ (i.e., use a majority vote to make the classification).

Thus the k-nearest neighbor classifier acts like an averaging device. In particular, it discriminates against a single outlier, as illustrated in Fig. 2.2 for $k = 3$. An *outlier* is an observation that is improbably large for a nominal model of interest.

In Chapter 5 we discuss another important type of memory-based classifier known as the radial-basis function network.

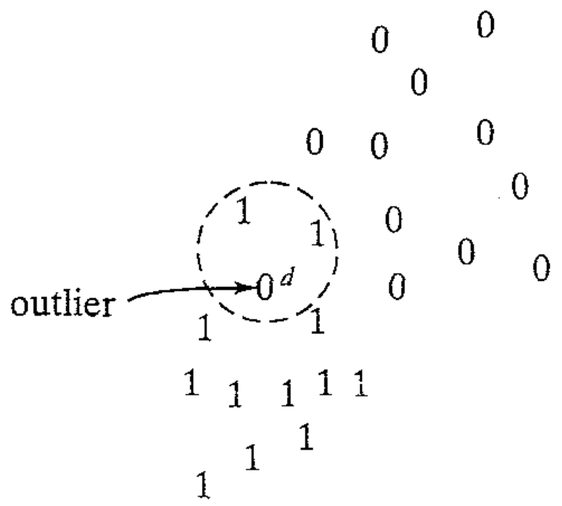


FIGURE 2.2 The area lying inside the dashed circle includes two points pertaining to class 1 and an outlier from class 0. The point *d* corresponds to the test vector $\mathbf{x}_{test}$. With $k = 3$, the *k-nearest* neighbor *classifier* assigns class 1 to point *d* even though it lies closest to the *outlier.*

2.4 HEBBIAN LEARNING

Hebb's postulate of learning is the oldest and most famous of all learning rules; it is named in honor of the neuropsychologist Hebb (1949). Quoting from Hebb's book, *The Organization of Behavior* (1949, p.62):

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

Hebb proposed this change as a basis of associative learning (at the cellular level), which would result in an enduring modification in the activity pattern of a spatially distributed "assembly of nerve cells."

This statement is made in a neurobiological context. We may expand and rephrase it as a two-part rule (Stent, 1973; Changeux and Danchin, 1976):

1. *If two neurons on either side of a synapse (connection) are activated simultaneously (i.e., synchronously), then the strength of that synapse is selectively increased.*
2. *If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.*

Such a synapse is called a *Hebbian synapse.*[3] (The original Hebb rule did not contain part 2.) More precisely, we define a Hebbian synapse as a synapse that uses a *time-dependent, highly local, and strongly interactive mechanism to increase synaptic efficiency as a function of the correlation between the presynaptic and postsynaptic activities.* From this definition we may deduce the following four key mechanisms (properties) that characterize a Hebbian synapse (Brown et al., 1990):

1. *Time-dependent mechanism.* This mechanism refers to the fact that the modifications in a Hebbian synapse depend on the exact time of occurrence of the presynaptic and postsynaptic signals.

2. *Local mechanism.* By its very nature, a synapse is the transmission site where information-bearing signals (representing ongoing activity in the presynaptic and postsynaptic units) are in *spatiotemporal* contiguity. This locally available information is used by a Hebbian synapse to produce a local synaptic modification that is input specific.

3. *Interactive mechanism.* The occurrence of a change in a Hebbian synapse depends on signals on both sides of the synapse. That is, a Hebbian form of learning depends on a "true interaction" between presynaptic and postsynaptic signals in the sense that we cannot make a prediction from either one of these two activities by itself. Note also that this dependence or interaction may be deterministic or statistical in nature.

4. *Conjunctional* or *correlational mechanism.* One interpretation of Hebb's postulate of learning is that the condition for a change in synaptic efficiency is the conjunction of presynaptic and postsynaptic signals. Thus, according to this interpretation, the co-occurrence of presynaptic and postsynaptic signals (within a short interval of time) is sufficient to produce the synaptic modification. It is for this reason that a Hebbian synapse is sometimes referred to as a *conjunctional synapse.* For another interpretation of Hebb's postulate of learning, we may think of the interactive mechanism characterizing a Hebbian synapse in statistical terms. In particular, the correlation over time between presynaptic and postsynaptic signals is viewed as being responsible for a synaptic change. Accordingly, a Hebbian synapse is also referred to as a *correlational synapse.* Correlation is indeed the basis of learning (Eggermont, 1990).

Synaptic Enhancement and Depression

The definition of a Hebbian synapse presented here does not include additional processes that may result in weakening of a synapse connecting a pair of neurons. Indeed, we may generalize the concept of a Hebbian modification by recognizing that positively correlated activity produces synaptic strengthening, and that either uncorrelated or negatively correlated activity produces synaptic weakening (Stent, 1973). Synaptic depression may also be of a noninteractive type. Specifically, the interactive condition for synaptic weakening may simply be noncoincident presynaptic or postsynaptic activity.

We may go one step further by classifying synaptic modifications as *Hebbian, anti-Hebbian, and non-Hebbian* (Palm, 1982). According to this scheme, a Hebbian synapse increases its strength with positively correlated presynaptic and postsynaptic signals, and decreases its strength when these signals are either uncorrelated or negatively correlated. Conversely, an anti-Hebbian synapse weakens positively correlated presynaptic and postsynaptic signals, and strengthens negatively correlated signals. In both Hebbian and anti-Hebbian synapses, however, the modification of synaptic efficiency relies on a mechanism that is time-dependent, highly local, and strongly interactive in nature. In that sense, an anti-Hebbian synapse is still Hebbian in nature, though not in function. A non-Hebbian synapse, on the other hand, does not involve a Hebbian mechanism of either kind.

Mathematical Models of Hebbian Modifications

To formulate Hebbian learning in mathematical terms, consider a synaptic weight w_{kj} of neuron k with presynaptic and postsynaptic signals denoted by x_j and y_k, respectively. The adjustment applied to the synaptic weight w_{kj} at time step n is expressed in the general form

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n)) \tag{2.8}$$

where $F(\cdot,\cdot)$ is a function of both postsynaptic and presynaptic signals. The signals $x_j(n)$ and $y_k(n)$ are often treated as dimensionless. The formula of Eq. (2.8) admits many forms, all of which qualify as Hebbian. In what follows, we consider two such forms.

Hebb's hypothesis. The simplest form of Hebbian learning is described by

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n) \tag{2.9}$$

where η is a positive constant that determines the *rate of learning*. Equation (2.9) clearly emphasizes the correlational nature of a Hebbian synapse. It is sometimes referred to as the *activity product rule*. The top curve of Fig. 2.3 shows a graphical representation of Eq. (2.9) with the change Δw_{kj} plotted versus the output signal (postsynaptic activity) y_k. From this representation we see that the repeated application of the input signal (presynaptic activity) x_j leads to an increase in y_k and therefore *exponential growth* that finally drives the synaptic connection into saturation. At that point no information will be stored in the synapse and selectivity is lost.

Covariance hypothesis. One way of overcoming the limitation of Hebb's hypothesis is to use the *covariance hypothesis* introduced in Sejnowski (1977a, b). In this hypothesis, the presynaptic and postsynaptic signals in Eq. (2.9) are replaced by the departure of presynaptic and postsynaptic signals from their respective average values over a certain time interval. Let $\bar{x}$ and $\bar{y}$ denote the *time-averaged values* of the presynaptic signal x_j and postsynaptic signal y_k, respectively. According to the covariance hypothesis, the adjustment applied to the synaptic weight w_{kj} is defined by

$$\Delta w_{kj} = \eta(x_j - \bar{x})(y_k - \bar{y}) \tag{2.10}$$

where η is the learning-rate parameter. The average values x and y constitute presynaptic and postsynaptic thresholds, which determine the sign of synaptic modification. In particular, the covariance hypothesis allows for the following:

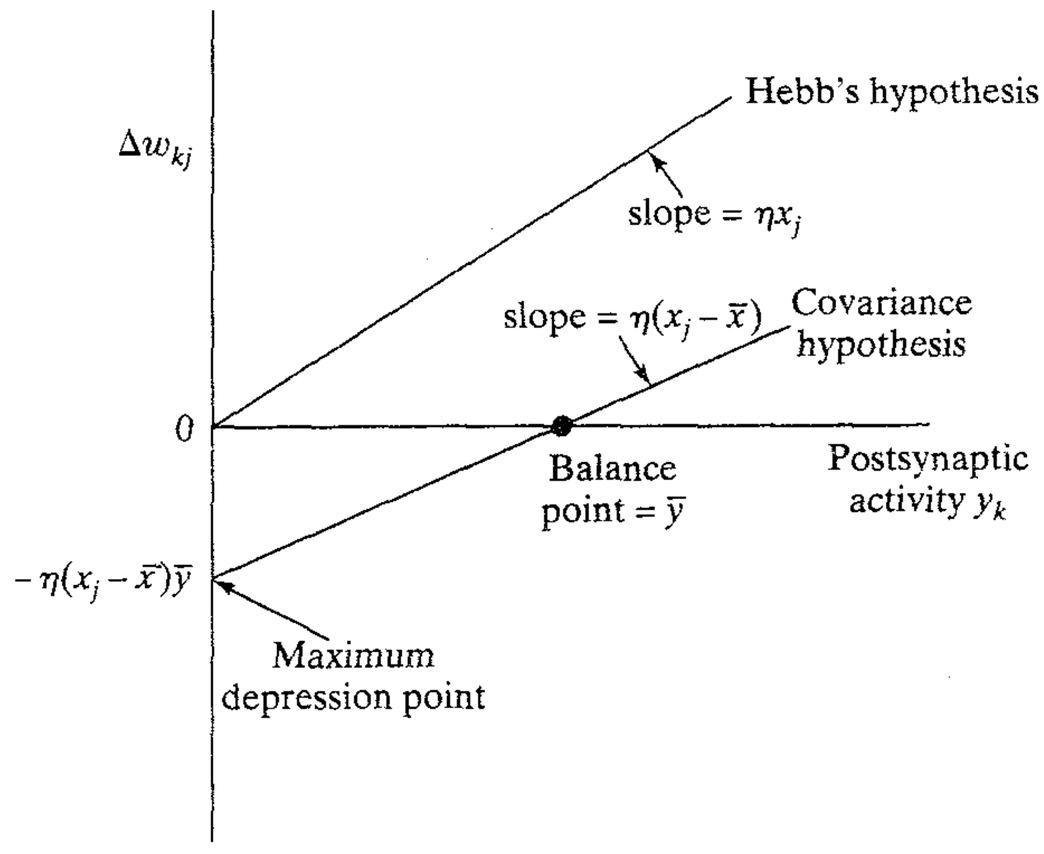


FIGURE 2.3 Illustration of Hebb's hypothesis and the covariance hypothesis.

- Convergence to a nontrivial state, which is reached when $x_k = \bar{x}$ or $y_j = \bar{y}$.
- Prediction of both synaptic *potentiation* (i.e., increase in synaptic strength) and synaptic *depression* (i.e., decrease in synaptic strength).

Figure 2.3 illustrates the difference between Hebb's hypothesis and the covariance hypothesis. In both cases the dependence of Δw_{kj} on y_k is linear; however, the intercept with the y_k-axis in Hebb's hypothesis is at the origin, whereas in the covariance hypothesis it is at $y_k = \bar{y}$.

We make the following important observations from Eq. (2.10):

1. Synaptic weight w_{kj} is enhanced if there are sufficient levels of presynaptic and postsynaptic activities, that is, the conditions $x_j > \bar{x}$ and $y_k > \bar{y}$ are both satisfied.
2. Synaptic weight w_{kj} is depressed if there is either
   - a presynaptic activation (i.e., $x_j > \bar{x}$) in the absence of sufficient postsynaptic activation (i.e., $y_k < \bar{y}$), or
   - a postsynaptic activation (i.e., $y_k > \bar{y}$) in the absence of sufficient presynaptic activation (i.e., $x_j < \bar{x}$).

This behavior may be regarded as a form of temporal competition between the incoming patterns.

There is strong physiological evidence[4] for Hebbian learning in the area of the brain called the *hippocampus*. The hippocampus plays an important role in certain aspects of learning or memory. This physiological evidence makes Hebbian learning all the more appealing.

2.5 COMPETITIVE LEARNING

In *competitive learning,*[5] as the name implies, the output neurons of a neural network compete among themselves to become active (fired). Whereas in a neural network based on Hebbian learning several output neurons may be active simultaneously, in competitive learning only a single output neuron is active at any one time. It is this feature that makes competitive learning highly suited to discover statistically salient features that may be used to classify a set of input patterns.

There are three basic elements to a competitive learning rule (Rumelhart and Zipser, 1985):

- A set of neurons that are all the same except for some randomly distributed synaptic weights, and which therefore *respond differently* to a given set of input patterns.
- A *limit* imposed on the "strength" of each neuron.
- A mechanism that permits the neurons to *compete* for the right to respond to a given subset of inputs, such that only *one* output neuron, or only one neuron per group, is active (i.e., "on") at a time. The neuron that wins the competition is called a *winner-takes-all neuron.*

Accordingly the individual neurons of the network learn to specialize on ensembles of similar patterns; in so doing they become *feature detectors* for different classes of input patterns.

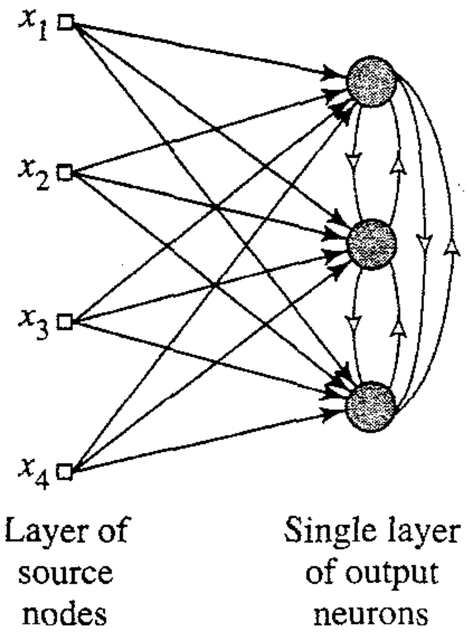


FIGURE 2.4 Architectural graph of a simple competitive learning network with feedforward (excitatory) connections from the source nodes to the neurons, and lateral (inhibitory) connections among the neurons; the lateral connections are signified by open arrows.

In the simplest form of competitive learning, the neural network has a single layer of output neurons, each of which is fully connected to the input nodes. The network may include feedback connections among the neurons, as indicated in Fig. 2.4. In the network architecture described herein, the feedback connections perform *lateral inhibition,*[6] with each neuron tending to inhibit the neuron to which it is laterally connected. In contrast, the feedforward synaptic connections in the network of Fig. 2.4 are all *excitatory*.

For a neuron k to be the winning neuron, its induced local field v_k for a specified input pattern $\mathbf{x}$ must be the largest among all the neurons in the network. The output signal y_k of winning neuron k is set equal to one; the output signals of all the neurons that lose the competition are set equal to zero. We thus write

$$y_k = \begin{cases} 1 & \text{if } v_k > v_j \text{ for all } j, j \neq k \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

where the induced local field v_k represents the combined action of all the forward and feedback inputs to neuron k.

Let w_{kj} denote the synaptic weight connecting input node j to neuron k. Suppose that each neuron is allotted a *fixed* amount of synaptic weight (i.e., all synaptic weights are positive), which is distributed among its input nodes; that is,

$$\sum_j w_{kj} = 1 \qquad \text{for all } k \tag{2.12}$$

A neuron then learns by shifting synaptic weights from its inactive to active input nodes. If a neuron does not respond to a particular input pattern, no learning takes place in that neuron. If a particular neuron wins the competition, each input node of that neuron relinquishes some proportion of its synaptic weight, and the weight relinquished is then distributed equally among the active input nodes. According to the standard *competitive learning rule,* the change Δw_{kj} applied to synaptic weight w_{kj} is defined by

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \text{if neuron } k \text{ wins the competition} \\ 0 & \text{if neuron } k \text{ loses the competition} \end{cases} \tag{2.13}$$

where η is the learning-rate parameter. This rule has the overall effect of moving the synaptic weight vector $\mathbf{w}_k$ of winning neuron k toward the input pattern $\mathbf{x}$.

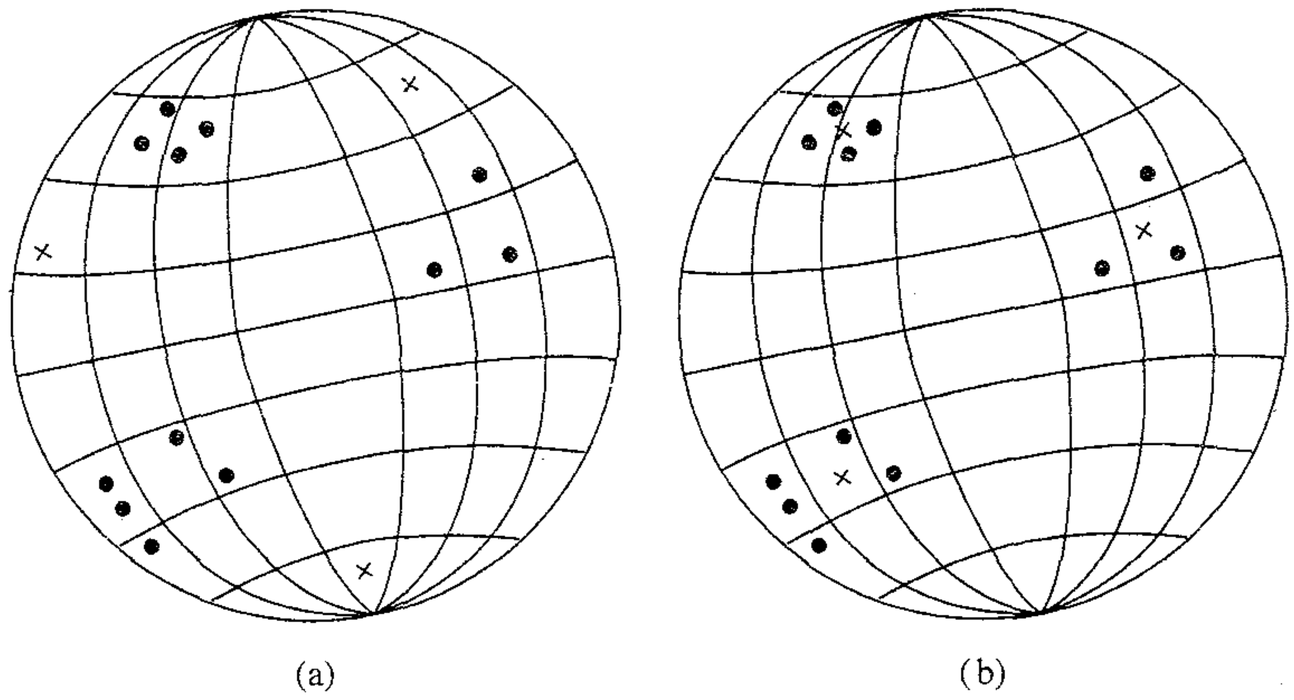


FIGURE 2.5 Geometric interpretation of the competitive learning process. The dots represent the input vectors, and the crosses represent the synaptic weight vectors of three output neurons. (a) Initial state of the network. (b) Final state of the network.

We may use the geometric analogy depicted in Fig. 2.5 to illustrate the essence of competitive learning (Rumelhart and Zipser, 1985). It is assumed that each input pattern (vector) $\mathbf{x}$ has some constant Euclidean length so that we may view it as a point on an N-dimensional unit sphere where N is the number of input nodes. N also represents the dimension of each synaptic weight vector $\mathbf{w}_k$. It is further assumed that all neurons in the network are constrained to have the same Euclidean length (norm), as shown by

$$\sum_j w_{kj}^2 = 1 \qquad \text{for all } k \tag{2.14}$$

When the synaptic weights are properly scaled they form a set of vectors that fall on the same N-dimensional unit sphere. In Fig. 2.5a we show three natural groupings (clusters) of the stimulus patterns represented by dots. This figure also includes a possible initial state of the network (represented by crosses) that may exist before learning. Figure 2.5b shows a typical final state of the network that results from the use of competitive learning. In particular, each output neuron has discovered a cluster of input patterns by moving its synaptic weight vector to the center of gravity of the discovered cluster (Rumelhart and Zipser, 1985; Hertz et al., 1991). This figure illustrates the ability of a neural network to perform *clustering* through competitive learning. However, for this function to be performed in a "stable" fashion the input patterns must fall into sufficiently distinct groupings to begin with. Otherwise the network may be unstable because it will no longer respond to a given input pattern with the same output neuron.

2.6 BOLTZMANN LEARNING

The Boltzmann learning rule, named in honor of Ludwig Boltzmann, is a stochastic learning algorithm derived from ideas rooted in statistical mechanics.[7] A neural net-

work designed on the basis of the Boltzmann learning rule is called a *Boltzmann machine* (Ackley et al., 1985; Hinton and Sejnowski, 1986).

In a Boltzmann machine the neurons constitute a recurrent structure, and they operate in a binary manner since, for example, they are either in an "on" state denoted by $+1$ or in an "off" state denoted by -1. The machine is characterized by an *energy function, E*, the value of which is determined by the particular states occupied by the individual neurons of the machine, as shown by

$$E = -\frac{1}{2}\sum_{j}\sum_{\substack{k \\ j \neq k}} w_{kj} x_k x_j \tag{2.15}$$

where x_j is the state of neuron j, and w_{kj} is the synaptic weight connecting neuron j to neuron k. The fact that $j \neq k$ means simply that none of the neurons in the machine has self-feedback. The machine operates by choosing a neuron at random—for example, neuron k—at some step of the learning process, then flipping the state of neuron k from state x_k to state $-x_k$ at some temperature T with probability

$$P(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E_k / T)} \tag{2.16}$$

where ΔE_k is the *energy change* (i.e., the change in the energy function of the machine) resulting from such a flip. Notice that T is not a physical temperature, but rather a *pseudotemperature,* as explained in Chapter 1. If this rule is applied repeatedly, the machine will reach *thermal equilibrium*.

The neurons of a Boltzmann machine partition into two functional groups: *visible* and *hidden*. The visible neurons provide an interface between the network and the environment in which it operates, whereas the hidden neurons always operate freely. There are two modes of operation to be considered:

- *Clamped condition,* in which the visible neurons are all clamped onto specific states determined by the environment.
- *Free-running condition,* in which all the neurons (visible and hidden) are allowed to operate freely.

Let ρ_{kj}^{+} denote the *correlation* between the states of neurons j and k, with the network in its clamped condition. Let ρ_{kj}^{-} denote the *correlation* between the states of neurons j and k with the network in its free-running condition. Both correlations are averaged over all possible states of the machine when it is in thermal equilibrium. Then, according to the *Boltzmann learning rule,* the change Δw_{kj} applied to the synaptic weight w_{kj} from neuron j to neuron k is defined by (Hinton and Sejnowski, 1986)

$$\Delta w_{kj} = \eta(\rho_{kj}^{+} - \rho_{kj}^{-}), \qquad j \neq k \tag{2.17}$$

where η is a learning-rate parameter. Note that both ρ_{kj}^{+} and ρ_{kj}^{-} range in value from -1 to $+1$.

A brief review of statistical mechanics is presented in Chapter 11; in that chapter we also present a detailed treatment of the Boltzmann machine and other stochastic machines.

2.7 CREDIT-ASSIGNMENT PROBLEM

When studying learning algorithms for distributed systems, it is useful to consider the notion of *credit assignment* (Minsky, 1961). Basically, the credit-assignment problem is the problem of assigning *credit* or *blame* for overall outcomes to each of the internal decisions made by a learning machine and which contributed to those outcomes. (The credit assignment problem is also referred to as the *loading problem,* the problem of "loading" a given set of training data into the free parameters of the network.)

In many cases the dependence of outcomes on internal decisions is mediated by a sequence of actions taken by the learning machine. In other words, internal decisions affect which particular actions are taken, and then the actions, not the internal decisions, directly influence overall outcomes. In these situations, we may decompose the credit-assignment problem into two subproblems (Sutton, 1984):

1. The assignment of credit for outcomes to actions. This is called the *temporal credit-assignment problem* in that it involves the instants of time *when* the actions that deserve credit were actually taken.
2. The assignment of credit for actions to internal decisions. This is called the *structural credit-assignment problem* in that it involves assigning credit to the *internal structures* of actions generated by the system.

The structural credit-assignment problem is relevant in the context of a multicomponent learning machine when we must determine precisely which particular component of the system should have its behavior altered and by how much in order to improve overall system performance. On the other hand, the temporal credit-assignment problem is relevant when there are many actions taken by a learning machine that result in certain outcomes, and we must determine which of these actions were responsible for the outcomes. The combined temporal and structural credit-assignment problem faces any distributed learning machine that attempts to improve its performance in situations involving temporally extended behavior (Williams, 1988).

The credit-assignment problem, for example, arises when error-correction learning is applied to a multilayer feedforward neural network. The operation of each hidden neuron, as well as that of each output neuron in such a network is important to its correct overall operation on a learning task of interest. That is, in order to solve the prescribed task the network must assign certain forms of behavior to all of its neurons through the specification of error-correction learning. With this background in mind, consider the situation described in Fig. 2.1a. Since the output neuron k is visible to the outside world, it is possible to supply a desired response to this neuron. As far as the output neuron is concerned, it is a straightforward matter to adjust the synaptic weights of the output neuron in accordance with error-correction learning, as outlined in Section 2.2. But how do we assign credit or blame for the action of the hidden neurons when the error-correction learning process is used to adjust the respective synaptic weights of these neurons? The answer to this fundamental question requires more detailed attention; it is presented in Chapter 4, where algorithmic details of the design of multilayer feedforward neural networks are described.

2.8 LEARNING WITH A TEACHER

We now turn our attention to learning paradigms. We begin by considering *learning with a teacher,* which is also referred to as *supervised learning*. Figure 2.6 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 of interest. Suppose now that the teacher and the neural network are both exposed to a training vector (i.e., example) drawn from the environment. By virtue of built-in 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 as fully as possible. 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 error-correction learning discussed previously in Section 2.2. It is a closed-loop feedback system, but the unknown environment is not in 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 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

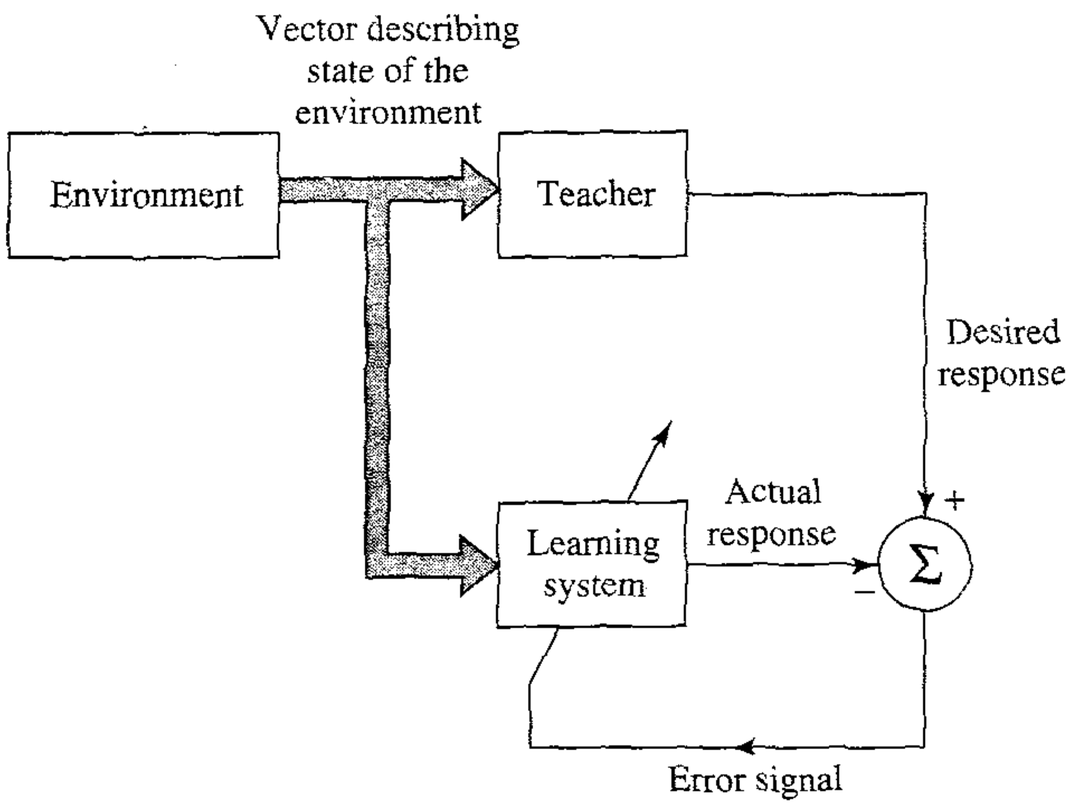


FIGURE 2.6 Block diagram of learning with a teacher.

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 an 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 permitted to do the training, a supervised learning system is usually able to perform such tasks as pattern classification and function approximation.

2.9 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 subdivisions are identified:

1. Reinforcement learning/Neurodynamic programming

In *reinforcement learning,*[8] 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 2.7 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

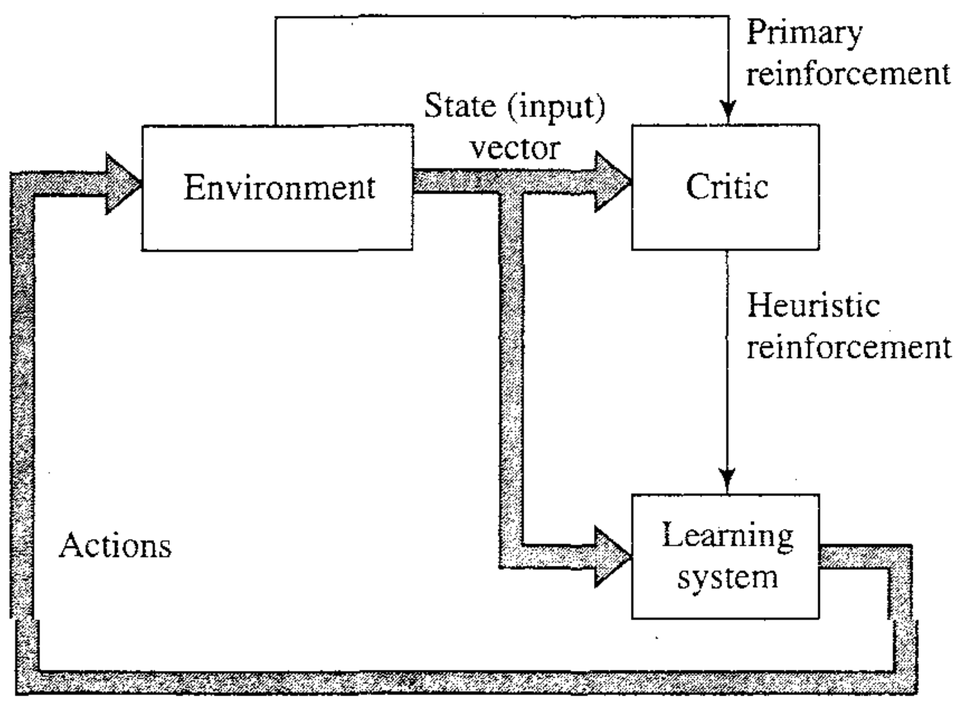


FIGURE 2.7 Block diagram of reinforcement learning.

system is designed to learn under *delayed reinforcement,* which means that the system observes a temporal sequence of stimuli (i.e., state vectors) also received from the environment, which eventually result in the generation of the heuristic reinforcement signal. The goal of 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 machine,* which constitutes the second component of the system, is to *discover* these actions and to 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 very appealing. It provides the basis for the 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.

Reinforcement learning is closely related to *dynamic programming,* which was developed by Bellman (1957) in the context of optimal control theory. Dynamic programming provides the mathematical formalism for sequential decision making. By casting reinforcement learning within the framework of dynamic programming, the subject matter becomes all the richer for it, as demonstrated in Bertsekas and Tsitsiklis (1996). An introductory treatment of dynamic programming and its relationship to reinforcement learning is presented in Chapter 12.

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. 2.8. 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. Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input and thereby to create new classes automatically (Becker, 1991).

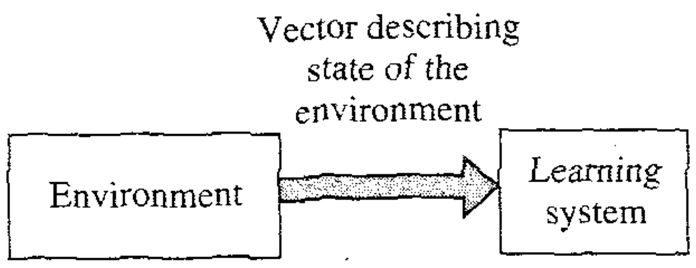


FIGURE 2.8 Block diagram of unsupervised learning.

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. As described in Section 2.5, in such a strategy the neuron with the greatest total input "wins" the competition and turns on; all the other neurons then switch off.

Different algorithms for unsupervised learning are described in Chapters 8 through 11.

2.10 LEARNING TASKS

In previous sections of this chapter we have discussed different learning algorithms and learning paradigms. In this section, we describe some basic learning tasks. The choice of a particular learning algorithm is influenced by the learning task that a neural network is required to perform. In this context we identify six learning tasks that apply to the use of neural networks in one form or another.

Pattern Association

An *associative memory* is a brainlike distributed memory that learns by *association*. Association has been known to be a prominent feature of human memory since Aristotle, and all models of cognition use association in one form or another as the basic operation (Anderson, 1995).

Association takes one of two forms: *autoassociation* or *heteroassociation*. In autoassociation, a neural network is required to *store* a set of patterns (vectors) by repeatedly presenting them to the network. The network is subsequently presented a partial description or distorted (noisy) version of an original pattern stored in it, and the task is to *retrieve* (*recall*) that particular pattern. Heteroassociation differs from autoassociation in that an arbitrary set of input patterns is *paired* with another arbitrary set of output patterns. Autoassociation involves the use of unsupervised learning, whereas the type of learning involved in heteroassociation is supervised.

Let $\mathbf{x}_k$ denote a *key pattern* (vector) applied to an associative memory and $\mathbf{y}_k$ denote a *memorized pattern* (vector). The pattern association performed by the network is described by

$$\mathbf{x}_k \rightarrow \mathbf{y}_k, \qquad k = 1, 2, \ldots, q \tag{2.18}$$

where q is the number of patterns stored in the network. The key pattern $\mathbf{x}_k$ acts as a stimulus that not only determines the storage location of memorized pattern $\mathbf{y}_k$, but also holds the key for its retrieval.

In an autoassociative memory, $\mathbf{y}_k = \mathbf{x}_k$, so the input and output (data) spaces of the network have the same dimensionality. In a heteroassociative memory, $\mathbf{y}_k \neq \mathbf{x}_k$; hence, the dimensionality of the output space in this second case may or may not equal the dimensionality of the input space.

There are two phases involved in the operation of an associative memory:

these learning tasks are all problems of learning a *mapping* from (possibly noisy) examples of the mapping. Without the imposition of prior knowledge, each of the tasks is in fact *ill posed* in the sense of nonuniqueness of possible solution mappings. One method of making the solution well posed is to use the theory of regularization as described in Chapter 5.

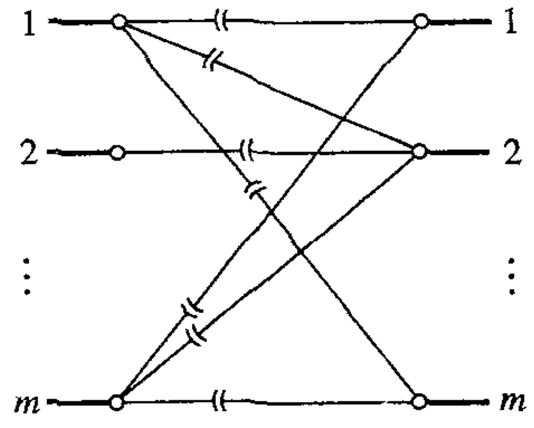
2.11 MEMORY

Discussion of learning tasks, particularly the task of pattern association, leads us naturally to think about *memory*. In a neurobiological context, memory refers to the relatively enduring neural alterations induced by the interaction of an organism with its environment (Teyler, 1986). Without such a change there can be no memory. Furthermore, for the memory to be useful it must be accessible to the nervous system in order to influence future behavior. However, an activity pattern must initially be stored in memory through a *learning process*. Memory and learning are intricately connected. When a particular activity pattern is learned, it is stored in the brain where it can be recalled later when required. Memory may be divided into "short-term" and "long-term" memory, depending on the retention time (Arbib, 1989). *Short-term memory* refers to a compilation of knowledge representing the "current" state of the environment. Any discrepancies between knowledge stored in short-term memory and a "new" state are used to update the short-term memory. *Long-term memory,* on the other hand, refers to knowledge stored for a long time or permanently.

In this section we study an associative memory that offers the following characteristics:

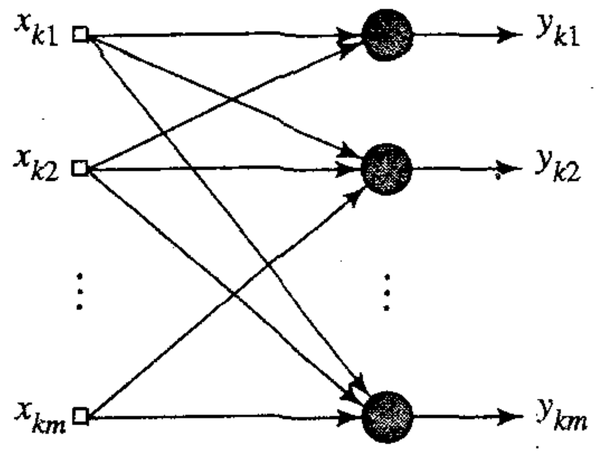
- The memory is distributed.
- Both the stimulus (key) pattern and the response (stored) pattern of an associative memory consist of data vectors.
- Information is stored in memory by setting up a spatial pattern of neural activities across a large number of neurons.
- Information contained in a stimulus not only determines its storage location in memory but also an address for its retrieval.
- Although neurons do not represent reliable and low-noise computing cells, the memory exhibits a high degree of resistance to noise and damage of a diffusive kind.
- There may be interactions between individual patterns stored in memory. (Otherwise the memory would have to be exceptionally large for it to accommodate the storage of a large number of patterns in perfect isolation from each other.) There is therefore the distinct possibility for the memory to make *errors* during the recall process.

In a *distributed memory,* the basic issue of interest is the simultaneous or near-simultaneous activities of many different neurons, which are the result of external or internal stimuli. The neural activities form a spatial pattern inside the memory that contains information about the stimuli. The memory is therefore said to perform a distributed mapping that transforms an activity pattern in the input space into another activity pattern in the output space. We may illustrate some important properties of a



Input layer of neurons | Synaptic junctions | Output layer of neurons

(a) Associative memory model component of a nervous system



Input layer of source nodes | Output layer of neurons

(b) Associative memory model using artificial neurons

FIGURE 2.17 Associative memory models.

distributed memory mapping by considering an idealized neural network that consists of two layers of neurons. Figure 2.17a illustrates a network that may be regarded as a *model component of a nervous system* (Cooper, 1973; Scofield and Cooper, 1985). Each neuron in the input layer is connected to every one of the neurons in the output layer. The actual synaptic connections between the neurons are complex and redundant. In the model of Fig. 2.17a, a single ideal junction is used to represent the integrated effect of all the synaptic contacts between the dendrites of a neuron in the input layer and the axon branches of a neuron in the output layer. The level of activity of a neuron in the input layer may affect the level of activity of every other neuron in the output layer.

The corresponding situation for an artificial neural network is depicted in Fig. 2.17b. Here we have an input layer of source nodes and an output layer of neurons acting as computation nodes. In this case, the synaptic weights of the network are included as integral parts of the neurons in the output layer. The connecting links between the two layers of the network are simply wires.

In the following mathematical analysis, the neural networks in Figs. 2.17a and 2.17b are both assumed to be *linear*. The implication of this assumption is that each neuron acts as a linear combiner, as depicted in the signal-flow graph of Fig. 2.18. To proceed with the analysis, suppose that an activity pattern $\mathbf{x}_k$ occurs in the input layer of the network and that an activity pattern $\mathbf{y}_k$ occurs simultaneously in the output layer. The issue we wish to consider here is that of learning from the association

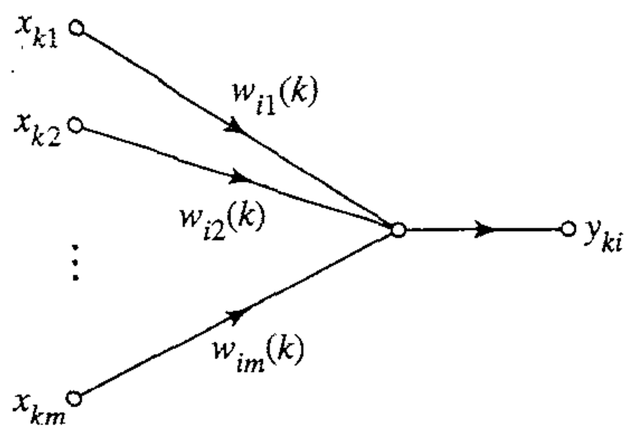


FIGURE 2.18 Signal-flow graph model of a linear neuron labeled *i*.

between the patterns $\mathbf{x}_k$ and $\mathbf{y}_k$. The patterns $\mathbf{x}_k$ and $\mathbf{y}_k$ are represented by vectors, written in their expanded forms as:

$$\mathbf{x}_k = [x_{k1}, x_{k2}, \ldots, x_{km}]^T$$

and

$$\mathbf{y}_k = [y_{k1}, y_{k2}, \ldots, y_{km}]^T$$

For convenience of presentation we have assumed that the input space dimensionality (i.e., the dimension of vector $\mathbf{x}_k$) and the output space dimensionality (i.e., the dimension of vector $\mathbf{y}_k$) are the same, equal to m. From here on we refer to m as *network dimensionality* or simply *dimensionality*. Note that m equals the number of source nodes in the input layer or neurons in the output layer. For a neural network with a large number of neurons, which is typically the case, the dimensionality m can be large.

The elements of both $\mathbf{x}_k$ and $\mathbf{y}_k$ can assume positive and negative values. This is a valid proposition in an artificial neural network. It may also occur in a nervous system by considering the relevant physiological variable to be the difference between an actual activity level (e.g., firing rate of a neuron) and a nonzero spontaneous activity level.

With the networks of Fig. 2.17 assumed to be linear, the association of key vector $\mathbf{x}_k$ with memorized vector $\mathbf{y}_k$ may be described in matrix form as:

$$\mathbf{y}_k = \mathbf{W}(k)\mathbf{x}_k, \qquad k = 1, 2, \ldots, q \tag{2.27}$$

where $\mathbf{W}(k)$ is a weight matrix determined solely by the input–output pair $(\mathbf{x}_k, \mathbf{y}_k)$.

To develop a detailed description of the weight matrix $\mathbf{W}(k)$, consider Fig. 2.18 that shows a detailed arrangement of neuron i in the output layer. The output y_{ki} of neuron i due to the combined action of the elements of the key pattern $\mathbf{x}_k$ applied as stimulus to the input layer is given by

$$y_{ki} = \sum_{j=1}^{m} w_{ij}(k)x_{kj}, \qquad i = 1, 2, \ldots, m \tag{2.28}$$

where the $w_{ij}(k), j = 1, 2, \ldots, m$, are the synaptic weights of neuron i corresponding to the kth pair of associated patterns. Using matrix notation, we may express y_{ki} in the equivalent form

$$y_{ki} = [w_{i1}(k), w_{i2}(k), \ldots, w_{im}(k)]\begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}, \qquad i = 1, 2, \ldots, m \tag{2.29}$$

The column vector on the right-hand side of Eq. (2.29) is recognized as the key vector $\mathbf{x}_k$. By substituting Eq. (2.29) in the definition of the m-by-1 stored vector $\mathbf{y}_k$, we get

$$\begin{bmatrix} y_{k1} \\ y_{k2} \\ \vdots \\ y_{km} \end{bmatrix} = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \dots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \dots & w_{2m}(k) \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1}(k) & w_{m2}(k) & \dots & w_{mm}(k) \end{bmatrix} \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix} \tag{2.30}$$

Equation (2.30) is the expanded form of the matrix transformation or mapping described in Eq. (2.27). In particular, the m-by-m weight matrix $\mathbf{W}(k)$ is defined by

$$\mathbf{W}(k) = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \dots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \dots & w_{2m}(k) \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1}(k) & w_{m2}(k) & \dots & w_{mm}(k) \end{bmatrix} \tag{2.31}$$

The individual presentations of the q pairs of associated patterns $\mathbf{x}_k \rightarrow \mathbf{y}_k$, $k = 1, 2, \dots, q$, produce corresponding values of the individual matrix, namely, $\mathbf{W}(1), \mathbf{W}(2), \dots, \mathbf{W}(q)$. Recognizing that this pattern association is represented by the weight matrix $\mathbf{W}(k)$, we may define an *m–by–m memory matrix* that describes the summation of the weight matrices for the entire set of pattern associations as follows:

$$\mathbf{M} = \sum_{k=1}^{q} \mathbf{W}(k) \tag{2.32}$$

The memory matrix $\mathbf{M}$ defines the overall connectivity between the input and output layers of the associative memory. In effect, it represents the *total experience* gained by the memory as a result of the presentations of q input–output patterns. Stated in another way, the memory matrix $\mathbf{M}$ contains a piece of every input–output pair of activity patterns presented to the memory.

The definition of the memory matrix given in Eq. (2.32) may be restructured in the form of a recursion as shown by

$$\mathbf{M}_k = \mathbf{M}_{k-1} + \mathbf{W}(k), \qquad k = 1, 2, \dots, q \tag{2.33}$$

where the initial value $\mathbf{M}_0$ is zero (i.e., the synaptic weights in the memory are all initially zero), and the final value $\mathbf{M}_q$ is identically equal to $\mathbf{M}$ as defined in Eq. (2.32). According to the recursive formula of Eq. (2.33), the term $\mathbf{M}_{k-1}$ is the old value of the memory matrix resulting from $(k-1)$ pattern associations, and $\mathbf{M}_k$ is the updated value in light of the increment $\mathbf{W}(k)$ produced by the kth association. Note, however, that when $\mathbf{W}(k)$ is added to $\mathbf{M}_{k-1}$, the increment $\mathbf{W}(k)$ loses its distinct identity among the mixture of contributions that form $\mathbf{M}_k$. In spite of the synaptic mixing of different associations, information about the stimuli may not have been lost, as demonstrated in the sequel. Notice also that as the number q of stored patterns increases, the influence of a new pattern on the memory as a whole is progressively reduced.

Correlation Matrix Memory

Suppose that the associative memory of Fig. 2.17b has learned the memory matrix $\mathbf{M}$ through the associations of key and memorized patterns described by $\mathbf{x}_k \rightarrow \mathbf{y}_k$, where $k = 1, 2, \ldots, q$. We may postulate $\hat{\mathbf{M}}$, denoting an *estimate* of the memory matrix $\mathbf{M}$ in terms of these patterns as (Anderson, 1972, 1983; Cooper, 1973):

$$\hat{\mathbf{M}} = \sum_{k=1}^{q} \mathbf{y}_k \mathbf{x}_k^T \tag{2.34}$$

The term $\mathbf{y}_k \mathbf{x}_k^T$ represents the *outer product* of the key pattern $\mathbf{x}_k$ and the memorized pattern $\mathbf{y}_k$. This outer product is an "estimate" of the weight matrix $\mathbf{W}(k)$ that maps the output pattern $\mathbf{y}_k$ onto the input pattern $\mathbf{x}_k$. Since the pattern $\mathbf{x}_k$ and $\mathbf{y}_k$ are both m-by-1 vectors by assumption, it follows that their output product $\mathbf{y}_k \mathbf{x}_k^T$, and therefore the estimate $\hat{\mathbf{M}}$, is an m-by-m matrix. This dimensionality is in perfect agreement with that of the memory matrix $\mathbf{M}$ defined in Eq. (2.32). The format of the summation of the estimate $\hat{\mathbf{M}}$ bears a direct relation to that of the memory matrix defined in that equation.

A typical term of the outer product $\mathbf{y}_k \mathbf{x}_k^T$ is written as $y_{ki} x_{kj}$, where x_{kj} is the output of source node j in the input layer, and y_{ki} is the output of neuron i in the output layer. In the context of synaptic weight $w_{ij}(k)$ for the kth association, source node j acts as a presynaptic node and neuron i in the output layer acts as a postsynaptic node. Hence, the "local" learning process described in Eq. (2.34) may be viewed as a *generalization of Hebb's postulate of learning.* It is also referred to as the *outer product rule* in recognition of the matrix operation used to construct the memory matrix $\hat{\mathbf{M}}$. Correspondingly, an associative memory so designed is called a *correlation matrix memory*. Correlation, in one form or another, is indeed the basis of learning, association, pattern recognition, and memory recall in the human nervous system (Eggermont, 1990.)

Equation (2.34) may be reformulated in the equivalent form

$$\begin{aligned} \hat{\mathbf{M}} &= [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q] \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_q^T \end{bmatrix} \\ &= \mathbf{Y}\mathbf{X}^T \end{aligned} \tag{2.35}$$

where

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q] \tag{2.36}$$

and

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q] \tag{2.37}$$

The matrix $\mathbf{X}$ is an m-by-q matrix composed of the entire set of key patterns used in the learning process; it is called the *key matrix.* The matrix $\mathbf{Y}$ is an m–by–q matrix composed of the corresponding set of memorized patterns; it is called the *memorized matrix.*

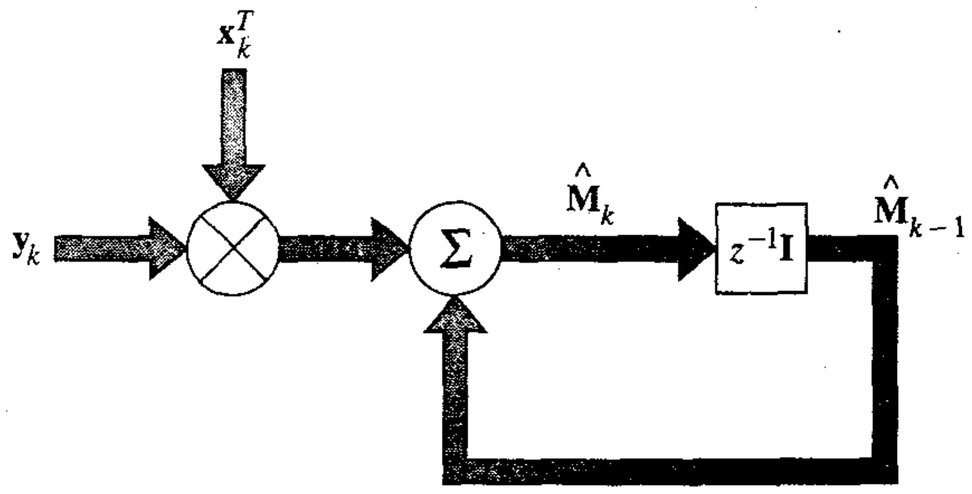


FIGURE 2.19 Signal-flow graph representation of Eq. (2.38).

Equation (2.35) may also be restructured in the form of a recursion as follows:

$$\hat{\mathbf{M}}_k = \hat{\mathbf{M}}_{k-1} + \mathbf{y}_k\mathbf{x}_k^T, \qquad k = 1, 2, \ldots, q \tag{2.38}$$

A signal-flow graph representation of this recursion is depicted in Fig. 2.19. According to this signal-flow graph and the recursive formula of Eq. (2.38), the matrix $\hat{\mathbf{M}}_{k-1}$ represents an old estimate of the memory matrix; and $\hat{\mathbf{M}}_k$ represents its updated value in the light of a new association performed by the memory on the patterns $\mathbf{x}_k$ and $\mathbf{y}_k$. Comparing the recursion of Eq. (2.38) with that of Eq. (2.33), we see that the outer product $\mathbf{y}_k\mathbf{x}_k^T$ represents an estimate of the weight matrix $\mathbf{W}(k)$ corresponding to the kth association of key and memorized patterns, $\mathbf{x}_k$ and $\mathbf{y}_k$.

Recall

The fundamental problem posed by the use of an associative memory is the address and recall of patterns stored in memory. To explain one aspect of this problem, let $\hat{\mathbf{M}}$ denote the memory matrix of an associative memory, which has been completely learned through its exposure to q pattern associations in accordance with Eq. (2.34). Let a key pattern $\mathbf{x}_j$ be picked at random and reapplied as *stimulus* to the memory, yielding the *response*

$$\mathbf{y} = \hat{\mathbf{M}}\mathbf{x}_j \tag{2.39}$$

Substituting Eq. (2.34) in (2.39), we get

$$\begin{aligned} \mathbf{y} &= \sum_{k=1}^{m} \mathbf{y}_k\mathbf{x}_k^T\mathbf{x}_j \\ &= \sum_{k=1}^{m} (\mathbf{x}_k^T\mathbf{x}_j)\mathbf{y}_k \end{aligned} \tag{2.40}$$

where, in the second line, it is recognized that $\mathbf{x}_k^T\mathbf{x}_j$ is a scalar equal to the *inner product* of the key vectors $\mathbf{x}_k$ and $\mathbf{x}_j$. We may rewrite Eq. (2.40) as

$$\mathbf{y} = (\mathbf{x}_j^T\mathbf{x}_j)\mathbf{y}_j + \sum_{\substack{k=1 \\ k \neq j}}^{m} (\mathbf{x}_k^T\mathbf{x}_j)\mathbf{y}_k \tag{2.41}$$

Let each of the key patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q$ be normalized to have unit energy; that is,

$$\begin{aligned} E_k &= \sum_{l=1}^{m} x_{kl}^2 \\ &= \mathbf{x}_k^T \mathbf{x}_k \\ &= 1, \qquad k = 1, 2, \ldots, q \end{aligned} \tag{2.42}$$

Accordingly, we may simplify the response of the memory to the stimulus (key pattern) $\mathbf{x}_j$ as

$$\mathbf{y} = \mathbf{y}_j + \mathbf{v}_j \tag{2.43}$$

where

$$\mathbf{v}_j = \sum_{\substack{k=1 \\ k \neq j}}^{m} (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k \tag{2.44}$$

The first term on the right-hand side of Eq. (2.43) represents the "desired" response $\mathbf{y}_j$; it may therefore be viewed as the "signal" component of the actual response $\mathbf{y}$. The second term $\mathbf{v}_j$ is a "noise vector" that arises because of the *crosstalk* between the key vector $\mathbf{x}_j$ and all the other key vectors stored in memory. The noise vector $\mathbf{v}_j$ is responsible for making errors on recall.

In the context of a linear signal space, we may define the *cosine of the angle* between a pair of vectors $\mathbf{x}_j$ and $\mathbf{x}_k$ as the inner product of $\mathbf{x}_j$ and $\mathbf{x}_k$ divided by the product of their individual Euclidean *norms* or *lengths* as shown by

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = \frac{\mathbf{x}_k^T \mathbf{x}_j}{\|\mathbf{x}_k\| \|\mathbf{x}_j\|} \tag{2.45}$$

The symbol $\|\mathbf{x}_k\|$ signifies the Euclidean norm of vector $\mathbf{x}_k$, defined as the square root of the energy of $\mathbf{x}_k$:

$$\begin{aligned} \|\mathbf{x}_k\| &= (\mathbf{x}_k^T \mathbf{x}_k)^{1/2} \\ &= E_k^{1/2} \end{aligned} \tag{2.46}$$

Returning to the situation, note that the key vectors are normalized to have unit energy in accordance with Eq. (2.42). We may therefore reduce the definition of Eq. (2.45) to

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = \mathbf{x}_k^T \mathbf{x}_j \tag{2.47}$$

We may then redefine the noise vector of Eq. (2.44) as

$$\mathbf{v}_j = \sum_{\substack{k=1 \\ k \neq j}}^{m} \cos(\mathbf{x}_k, \mathbf{x}_j) \mathbf{y}_k \tag{2.48}$$

We now see that if the key vectors are *orthogonal* (i.e., perpendicular to each other in a Euclidean sense), then

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = 0, \qquad k \neq j \tag{2.49}$$

and therefore the noise vector $\mathbf{v}_j$ is identically zero. In such a case, the response $\mathbf{y}$ equals $\mathbf{y}_j$. The *memory associates perfectly* if the key vectors from an *orthonormal set;* that is, if they satisfy the following pair of conditions:

$$\mathbf{x}_k^T \mathbf{x}_j = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases} \tag{2.50}$$

Suppose now that the key vectors do form an orthonormal set, as prescribed in Eq. (2.50). What is then the limit on the *storage capacity* of the associative memory? Stated in another way, what is the largest number of patterns that can be reliably stored? The answer to this fundamental question lies in the rank of the memory matrix $\hat{\mathbf{M}}$. The *rank* of a matrix is defined as the number of independent columns (rows) of the matrix. That is, if r is the rank of such a rectangular matrix of dimensions l-by-m, we then have $r \leq \min(l, m)$. In the case of a correlation memory, the memory matrix $\hat{\mathbf{M}}$ is an m-by-m matrix, where m is the dimensionality of the input space. Hence the rank of the memory matrix $\mathbf{M}$ is limited by the dimensionality m. We may thus formally state that the number of patterns that can be reliably stored in a correlation matrix memory can never exceed the input space dimensionality.

In real-life situations, we often find that the key patterns presented to an associative memory are neither orthogonal nor highly separated from each other. Consequently, a correlation matrix memory characterized by the memory matrix of Eq. (2.34) may sometimes get confused and make *errors.* That is, the memory occasionally recognizes and associates patterns never seen or associated before. To illustrate this property of an associative memory, consider a set of key patterns.

$$\{\mathbf{x}_{\text{key}}\}: \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q$$

and a corresponding set of memorized patterns,

$$\{\mathbf{y}_{\text{mem}}\}: \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q$$

To express the closeness of the key patterns in a linear signal space, we introduce the concept of *community.* We define the community of the set of patterns $\{\mathbf{x}_{\text{key}}\}$ as the lower bound on the inner products $\mathbf{x}_k^T \mathbf{x}_j$ of any two patterns $\mathbf{x}_j$ and $\mathbf{x}_k$ in the set. Let $\hat{\mathbf{M}}$ denote the memory matrix resulting from the training of the associative memory on a set of key patterns represented by $\{\mathbf{x}_{\text{key}}\}$ and a corresponding set of memorized patterns $\{\mathbf{y}_{\text{mem}}\}$ in accordance with Eq. (2.34). The response of the memory, $\mathbf{y}$, to a stimulus $\mathbf{x}_j$ selected from the set $\{\mathbf{x}_{\text{key}}\}$ is given by Eq. (2.39), where it is assumed that each pattern in the set $\{\mathbf{x}_{\text{key}}\}$ is a unit vector (i.e., a vector with unit energy). Let it be further assumed that

$$\mathbf{x}_k^T \mathbf{x}_j \geq \gamma \qquad \text{for } k \neq j \tag{2.51}$$

If the lower bound γ is large enough, the memory may fail to distinguish the response $\mathbf{y}$ from that of any other key pattern contained in the set $\{\mathbf{x}_{\text{key}}\}$. If the key patterns in this set have the form

$$\mathbf{x}_j = \mathbf{x}_0 + \mathbf{v} \tag{2.52}$$

where $\mathbf{v}$ is a stochastic vector, it is likely that the memory will recognize $\mathbf{x}_0$ and associate with it a vector $\mathbf{y}_0$ rather than any of the actual pattern pairs used to train it in the

first place; $\mathbf{x}_0$ and $\mathbf{y}_0$ denote a pair of patterns never seen before. This phenomenon may be termed *animal logic,* which is not logic at all (Cooper, 1973).

2.12 ADAPTATION

In performing a task of interest, we often find that *space* is one fundamental dimension of the learning process; *time* is the other. The *spatiotemporal* nature of learning is exemplified by many of the learning tasks (e.g., control, beamforming) discussed in Section 2.10. Species ranging from insects to humans have an inherent capacity to represent the temporal structure of experience. Such a representation makes it possible for an animal to *adapt* its behavior to the temporal structure of an event in its behavioral space (Gallistel, 1990).

When a neural network operates in a *stationary* environment (i.e., an environment whose statistical characteristics do not change with time), the essential statistics of the environment can, in theory, be *learned* by the network under the supervision of a teacher. In particular, the synaptic weights of the network can be computed by having the network undergo a training session with a set of data that is representative of the environment. Once the training process has completed, the synaptic weights of the network should capture the underlying statistical structure of the environment, which would justify "freezing" their values thereafter. Thus a learning system relies on *memory,* in one form or another, to recall and exploit past experiences.

Frequently, however, the environment of interest is *nonstationary,* which means that the statistical parameters of the information-bearing signals generated by the environment vary with time. In situations of this kind, the traditional methods of supervised learning may prove to be inadequate because the network is not equipped with the necessary means to *track* the statistical variations of the environment in which it operates. To overcome this shortcoming, it is desirable for a neural network to continually *adapt* its free parameters to variations in the incoming signals in a *real-time* fashion. Thus an *adaptive system* responds to every distinct input as a novel one. In other words the learning process encountered in an adaptive system never stops, with learning going on while signal processing is being performed by the system. This form of learning is called *continuous learning* or *learning-on-the-fly*.

Linear adaptive filters, built around a linear combiner (i.e., a single neuron operating in its linear mode), are designed to perform continuous learning. Despite their simple structure (and perhaps because of it), they are widely used in such diverse applications as radar, sonar, communications, seismology, and biomedical signal processing. The theory of linear adaptive filters has reached a highly mature stage of development (Haykin, 1996; Widrow and Stearns, 1985). However, the same cannot be said about nonlinear adaptive filters.[11]

With continuous learning as the property of interest and a neural network as the vehicle for its implementation, the question we need to address is: How can a neural network adapt its behavior to the varying temporal structure of the incoming signals in its behavioral space? One way of addressing this fundamental issue is to recognize that statistical characteristics of a nonstationary process usually change slowly enough for the process to be considered *pseudostationary* over a window of short enough duration. Examples include:

achieve good overall performance, the bias $B(\mathbf{w})$ and the variance $V(\mathbf{w})$ of the approximating function $F(\mathbf{x}, \mathbf{w}) = F(\mathbf{x}, \mathcal{T})$ would both have to be small.

Unfortunately, we find that in a neural network that learns by example and does so with a training sample of fixed size, the price for achieving a small bias is a large variance. For a single neural network, it is only when the size of the training sample becomes infinitely large that we can hope to eliminate both bias and variance at the same time. We then have a *bias/variance dilemma,* and the consequence is prohibitively slow convergence (Geman et al., 1992). The bias/variance dilemma may be circumvented if we are willing to *purposely* introduce bias, which then makes it possible to eliminate the variance or to reduce it significantly. Needless to say, we must be sure that the bias built into the network design is harmless. In the context of pattern classification, for example, the bias is said to be "harmless" in the sense that it will contribute significantly to mean-square error only if we try to infer regressions that are not in the anticipated class. In general, bias must be *designed* for each specific application of interest. A practical way of achieving such an objective is to use a *constrained* network architecture, which usually performs better than a general-purpose architecture. For example, the constraints and therefore the bias may take the form of prior knowledge built into the network design using (1) *weight-sharing* where several synapses of the network are controlled by a single weight, and/or (2) *local receptive fields* assigned to individual neurons in the network, as demonstrated in the application of a multilayer perceptron to the optical character recognition problem (LeCun et al., 1990a). These network design issues were briefly discussed in Section 1.7.

2.14 STATISTICAL LEARNING THEORY

In this section we continue the statistical characterization of neural networks by describing a *learning theory* that addresses the fundamental issue of how to control the generalization ability of a neural network in mathematical terms. The discussion is presented in the context of supervised learning.

A model of supervised learning consists of three interrelated components, illustrated in Fig. 2.22 and abstracted in mathematical terms as follows (Vapnik, 1992, 1998):

1. *Environment.* The environment is stationary, supplying a vector $\mathbf{x}$ with a fixed but unknown cumulative (probability) distribution function $F_{\mathbf{X}}(\mathbf{x})$.
2. *Teacher.* The teacher provides a desired response d for every input vector $\mathbf{x}$ received from the environment, in accordance with a conditional cumulative distribution function $F_{\mathbf{X}}(\mathbf{x}|d)$ that is also fixed but unknown. The desired response d and input vector $\mathbf{x}$ are related by

$$d = f(\mathbf{x}, v) \tag{2.69}$$

   where v is a noise term, permitting the teacher to be "noisy."
3. *Learning machine (algorithm).* The learning machine (neural network) is capable of implementing a set of input–output mapping functions described by

$$y = F(\mathbf{x}, \mathbf{w}) \tag{2.70}$$

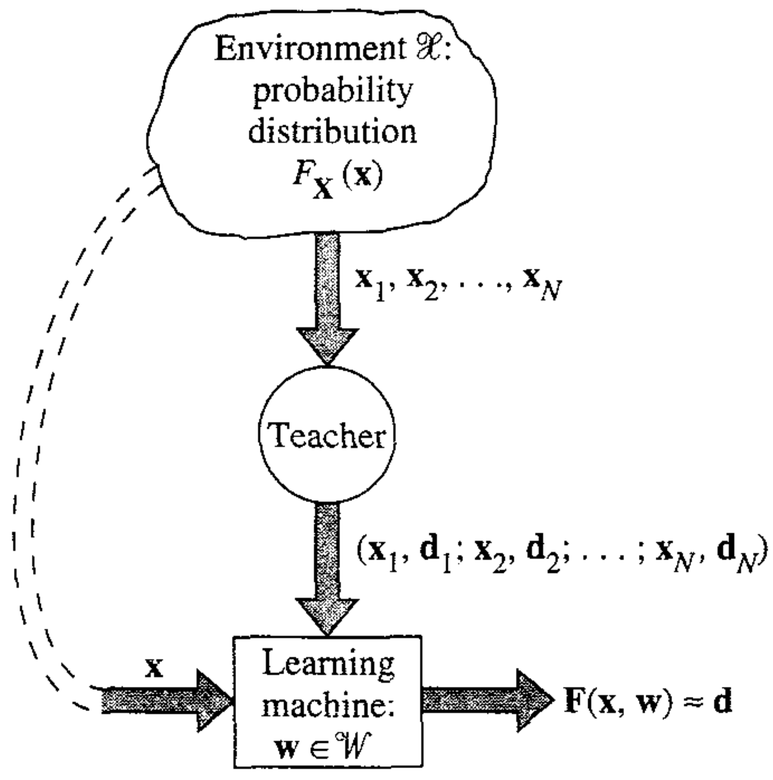


FIGURE 2.22 Model of the supervised learning process.

where y is the actual response produced by the learning machine in response to the input $\mathbf{x}$, and $\mathbf{w}$ is a set of free parameters (synaptic weights) selected from the parameter (weight) space $\mathscr{W}$.

Equations (2.69) and (2.70) are written in terms of the examples used to perform the training.

The supervised learning problem is that of selecting the particular function $F(\mathbf{x}, \mathbf{w})$ that approximates the desired response d in an optimum fashion, with "optimum" being defined in some statistical sense. The selection itself is based on the set of N *independent, identically distributed (iid)* training examples described in Eq. (2.53) and reproduced here for convenience of presentation:

$$\mathscr{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$$

Each example pair is drawn by the learning machine from $\mathscr{T}$ with a joint cumulative (probability) distribution function $F_{\mathbf{X},D}(\mathbf{x}, d)$, which, like the other distribution functions, is also fixed but unknown. The feasibility of supervised learning depends on this question: Do the training examples $\{(\mathbf{x}_i, d_i)\}$ contain sufficient information to construct a learning machine capable of good generalization performance? An answer to this fundamental question lies in the use of tools pioneered by Vapnik and Chervonenkis (1971). Specifically, we proceed by viewing the supervised learning problem as an *approximation problem,* which involves finding the function $F(\mathbf{x}, \mathbf{w})$ that is the best possible approximation to the desired function $f(\mathbf{x})$.

Let $L(d, F(\mathbf{x}, \mathbf{w}))$ denote a measure of the *loss* or *discrepancy* between the desired response d corresponding to an input vector $\mathbf{x}$ and the actual response $F(\mathbf{x}, \mathbf{w})$ produced by the learning machine. A popular definition for the loss $L(d, F(\mathbf{x}, \mathbf{w}))$ is the *quadratic loss function* defined as the squared distance between $d = f(\mathbf{x})$ and the approximation $F(\mathbf{x}, \mathbf{w})$ as shown by[12]

$$L(d, F(\mathbf{x}, \mathbf{w})) = (d - F(\mathbf{x}, \mathbf{w}))^2 \tag{2.71}$$

The squared distance of Eq. (2.64) is the ensemble-averaged extension of $L(d, F(\mathbf{x}, \mathbf{w}))$, with the averaging being performed over all the example pairs $(\mathbf{x}, d)$.

Most of the literature on statistical learning theory deals with a specific loss. The strong point of the statistical learning theory presented here is that it does *not* depend critically on the form of the loss function $L(d, F(\mathbf{x}, \mathbf{w}))$. Later in the section we do restrict the discussion to a specific loss function.

The expected value of the loss is defined by the *risk functional*

$$R(\mathbf{w}) = \int L(d, F(\mathbf{x}, \mathbf{w}))dF_{\mathbf{X},D}(\mathbf{x}, d) \tag{2.72}$$

where the integral is a multi-fold integral taken over all possible values of the example pair $(\mathbf{x}, d)$. The goal of supervised learning is to minimize the risk functional $R(\mathbf{w})$ over the class of approximating functions $\{F(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \mathcal{W}\}$. However, evaluation of the risk functional $R(\mathbf{w})$ is complicated because the joint cumulative distribution function $F_{\mathbf{X},D}(\mathbf{x}, d)$ is usually unknown. In supervised learning, the only information available is contained in the training data set $\mathcal{T}$. To overcome this mathematical difficulty, we use the inductive principle of empirical risk minimization (Vapnik, 1982). This principle relies entirely on availability of the training data set $\mathcal{T}$, which makes it perfectly suited to the design philosophy of neural networks.

Some Basic Definitions

Before proceeding further, we digress briefly to introduce some basic definitions that we use in the material to follow.

Convergence in probability. Consider a sequence of random variables $a_1, a_2, \ldots, a_N$. This sequence of random variables is said to *converge in probability* to a random variable a_0 if for any $\delta > 0$, the probabilistic relation

$$P(|a_N - a_0| > \delta) \xrightarrow{P} 0 \qquad \text{as } N \to \infty \tag{2.73}$$

holds.

Supremum and infimum. The supremum of a nonempty set $\mathcal{A}$ of scalars, denoted by $\sup \mathcal{A}$, is defined as the smallest scalar x such that $x \geq y$ for all $y \in \mathcal{A}$. If no such scalar exists, we say that the supremum of the nonempty set $\mathcal{A}$ is ∞. Similarly, the infimum of set $\mathcal{A}$, denoted by $\inf \mathcal{A}$, is defined as the largest scalar x such that $x \leq y$ for all $y \in \mathcal{A}$. If no such scalar exists, we say that the infimum of the nonempty set $\mathcal{A}$ is ∞.

Empirical risk functional. Given the training sample $\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$, the empirical risk functional is defined in terms of the loss function $L(d_i, F(\mathbf{x}_i, \mathbf{w}))$ as

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(d_i, F(\mathbf{x}_i, \mathbf{w})) \tag{2.74}$$

Strict Consistency. Consider the set $\mathcal{W}$ of functions $L(d, F(\mathbf{x}, \mathbf{w}))$ whose underlying distribution is defined by the joint cumulative distribution function $F_{\mathbf{X},D}(\mathbf{x}, d)$. Let $\mathcal{W}(c)$ be any nonempty subset of this set of functions, such that

$$\mathcal{W}(c) = \left\{ \mathbf{w}: \int L(d, F(\mathbf{x}, \mathbf{w})) \geq c \right\} \tag{2.75}$$

where $c \in (-\infty, \infty)$. The empirical risk functional is said to be *strictly (nontrivially) consistent* if for any subset $\mathcal{W}(c)$ the following convergence in probability

$$\inf_{\mathbf{w} \in \mathcal{W}(c)} R_{\text{emp}}(\mathbf{w}) \xrightarrow{P} \inf_{\mathbf{w} \in \mathcal{W}(c)} R(\mathbf{w}) \qquad \text{as } N \to \infty \tag{2.76}$$

holds.

With these definitions we may resume the discussion of Vapnik's statistical learning theory.

Principle of Empirical Risk Minimization

The basic idea of the principle of *empirical risk minimization* is to work with the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ defined in Eq. (2.74). This new functional differs from the risk functional $R(\mathbf{w})$ of Eq. (2.72) in two desirable ways:

1. It does *not* depend on the unknown distribution function $F_{\mathbf{X},D}(\mathbf{x}, d)$ in an explicit sense.
2. In theory it can be minimized with respect to the weight vector $\mathbf{w}$.

Let $\mathbf{w}_{\text{emp}}$ and $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ denote the weight vector and the corresponding mapping that minimize the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ in Eq. (2.74). Similarly, let $\mathbf{w}_o$ and $\mathbf{F}(\mathbf{x}, \mathbf{w}_o)$ denote the weight vector and the corresponding mapping that minimize the actual risk functional $R(\mathbf{w})$ in Eq. (2.72). Both $\mathbf{w}_{\text{emp}}$ and $\mathbf{w}_o$ belong to the weight space $\mathcal{W}$. The problem we must now consider is the conditions under which the approximate mapping $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ is "close" to the desired mapping $F(\mathbf{x}, \mathbf{w}_o)$ as measured by the mismatch between $R(\mathbf{w}_{\text{emp}})$ and $R(\mathbf{w}_o)$.

For some fixed $\mathbf{w} = \mathbf{w}^*$, the risk functional $R(\mathbf{w}^*)$ determines the *mathematical expectation* of a random variable defined by

$$Z_{\mathbf{w}^*} = L(d, F(\mathbf{x}, \mathbf{w}^*)) \tag{2.77}$$

In contrast, the empirical risk functional $R_{\text{emp}}(\mathbf{w}^*)$ is the *empirical (arithmetic) mean* of the random variable $Z_{\mathbf{w}^*}$. According to the *law of large numbers,* which constitutes one of the main theorems of probability theory, in general cases we find that as the size N of the training sample $\mathcal{T}$ is made infinitely large, the empirical mean of the random variable $Z_{\mathbf{w}^*}$ converges to its expected value. This observation provides theoretical justification for the use of the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ in place of the risk functional $R(\mathbf{w})$. However, just because the empirical mean of $Z_{\mathbf{w}^*}$ converges to its expected value, there is no reason to expect that the weight vector $\mathbf{w}_{\text{emp}}$ that minimizes the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ will also minimize the risk functional $R(\mathbf{w})$.

We may satisfy this requirement in an approximate fashion by proceeding as follows. If the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ approximates the original risk functional

$R(\mathbf{w})$ *uniformly* in $\mathbf{w}$ with some *precision* ϵ, then the minimum of $R_{\text{emp}}(\mathbf{w})$ deviates from the minimum of $R(\mathbf{w})$ by an amount not exceeding 2ϵ. Formally, this means that we must impose a stringent condition, such that for any $\mathbf{w} \in \mathcal{W}$ and $\epsilon > 0$, the probabilistic relation

$$P\left(\sup_{\mathbf{w}} |R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})| > \epsilon\right) \to 0 \qquad \text{as } N \to \infty \tag{2.78}$$

holds (Vapnik, 1982). When Eq. (2.78) is satisfied, we say that a *uniform convergence in the weight vector* $\mathbf{w}$ *of the empirical mean risk to its expected value occurs.* Equivalently, provided that for any prescribed precision ϵ we can assert the inequality

$$P\left(\sup_{\mathbf{w}} |R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})| > \epsilon\right) < \alpha \tag{2.79}$$

for some $\alpha > 0$, then as a consequence the following inequality also holds:

$$P(R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) > 2\epsilon) < \alpha \tag{2.80}$$

In other words, if the condition (2.79) holds, then with probability at least $(1 - \alpha)$, the solution $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ that minimizes the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ will give an actual risk $R(\mathbf{w}_{\text{emp}})$ that deviates from the true minimum possible actual risk $R(\mathbf{w}_o)$ by an amount not exceeding 2ϵ. Indeed, the condition (2.79) implies that with probability $(1 - \alpha)$ the following two inequalities are satisfied simultaneously (Vapnik, 1982):

$$R(\mathbf{w}_{\text{emp}}) - R_{\text{emp}}(\mathbf{w}_{\text{emp}}) < \epsilon \tag{2.81}$$

$$R_{\text{emp}}(\mathbf{w}_o) - R(\mathbf{w}_o) < \epsilon \tag{2.82}$$

These two equations define the differences between the true risk and empirical risk functionals at $\mathbf{w} = \mathbf{w}_{\text{emp}}$ and $\mathbf{w} = \mathbf{w}_o$, respectively. Furthermore, since $\mathbf{w}_{\text{emp}}$ and $\mathbf{w}_o$ are the minimum points of $R_{\text{emp}}(\mathbf{w})$ and $R(\mathbf{w})$, respectively, it follows that

$$R_{\text{emp}}(\mathbf{w}_{\text{emp}}) \leq R_{\text{emp}}(\mathbf{w}_o) \tag{2.83}$$

By adding the inequalities (2.81) and (2.82), and then using (2.83), we may write the following inequality

$$R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) < 2\epsilon \tag{2.84}$$

Also, since the inequalities (2.81) and (2.82) are both satisfied simultaneously with probability $(1 - \alpha)$, so is the inequality (2.84). We may also state that with probability α the inequality

$$R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) > 2\epsilon$$

holds, which is a restatement of (2.80).

We are now ready to make a formal statement of the *principle of empirical risk minimization* in three interrelated parts (Vapnik, 1982, 1998):

1. In place of the risk functional $R(\mathbf{w})$, construct the empirical risk functional

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(d_i, F(\mathbf{x}_i, \mathbf{w}))$$

on the basis of the training set of i.i.d. examples

$$(\mathbf{x}_i, d_i), \qquad i = 1, 2, \ldots, N$$

2. Let $\mathbf{w}_{emp}$ denote the weight vector that minimizes the empirical risk functional $R_{emp}(\mathbf{w})$ over the weight space $\mathcal{W}$. Then $R(\mathbf{w}_{emp})$ converges in probability to the minimum possible values of the actual risk $R(\mathbf{w})$, $\mathbf{w} \in \mathcal{W}$, as the size N of the training sample is made infinitely large, provided that the empirical risk functional $R_{emp}(\mathbf{w})$ converges uniformly to the actual risk functional $R(\mathbf{w})$.
3. Uniform convergence as defined by

$$P\left(\sup_{\mathbf{w}\in\mathcal{W}} |R(\mathbf{w}) - R_{emp}(\mathbf{w})| > \epsilon\right) \to 0 \qquad \text{as } N \to \infty$$

is a necessary and sufficient condition for the consistency of the principle of empirical risk minimization.

For a physical interpretation of this important principle, we offer the following observation. Prior to the training of a learning machine, all approximating functions are equally likely. As the training of the learning machine progresses, the likelihood of those approximating functions $F(\mathbf{x}_i, \mathbf{w})$ that are consistent with the training data set $\{\mathbf{x}_i, d_i)\}_{i=1}^N$ is increased. As the size N of the training data set grows, and the input space is thereby "densely" populated, the minimum point of the empirical risk functional $R_{emp}(\mathbf{w})$ converges in probability to the minimum point of the true risk functional $R(\mathbf{w})$.

VC Dimension

The theory of uniform convergence of the empirical risk functional $R_{emp}(\mathbf{w})$ to the actual risk functional $R(\mathbf{w})$ includes bounds on the rate of convergence, which are based on an important parameter called the *Vapnik–Chervonenkis dimension,* or simply the *VC dimension,* named in honor of its originators, Vapnik and Chervonenkis (1971). The VC dimension is a measure of the *capacity* or *expressive power* of the family of classification functions realized by the learning machine.

To describe the concept of VC dimension in a manner suitable for our purposes, consider a binary pattern classification problem, for which the desired response is written as $d \in \{0, 1\}$. We use the term *dichotomy* to refer to a binary classification function or decision rule. Let $\mathcal{F}$ denote the ensemble of dichotomies implemented by a learning machine, that is,

$$\mathcal{F} = \{F(\mathbf{x}, \mathbf{w}): \mathbf{w} \in \mathcal{W}, F: \mathbb{R}^m \mathcal{W} \to \{0, 1\}\} \tag{2.85}$$

Let $\mathcal{L}$ denote the set of N points in the m-dimensional space $\mathcal{X}$ of input vectors, that is,

$$\mathcal{L} = \{\mathbf{x}_i \in \mathcal{X}; i = 1, 2, \ldots, N\} \tag{2.86}$$

A dichotomy implemented by the learning machine partitions $\mathcal{L}$ into two disjoint subsets $\mathcal{L}_0$ and $\mathcal{L}_1$, such that we may write

$$F(\mathbf{x},\mathbf{w}) = \begin{cases} 0 & \text{for } \mathbf{x} \in \mathcal{L}_0 \\ 1 & \text{for } \mathbf{x} \in \mathcal{L}_1 \end{cases} \tag{2.87}$$

Let $\Delta_{\mathcal{F}}(\mathcal{L})$ denote the number of distinct dichotomies implemented by the learning machine, and $\Delta_{\mathcal{F}}(l)$ denote the maximum of $\Delta_{\mathcal{F}}(\mathcal{L})$ over all $\mathcal{L}$ with $|\mathcal{L}| = l$, where $|\mathcal{L}|$ is the number of elements of $\mathcal{L}$. We say that $\mathcal{L}$ is *shattered* by $\mathcal{F}$ if $\Delta_{\mathcal{F}}(\mathcal{L}) = 2^{|\mathcal{L}|}$, that is, if all possible dichotomies of $\mathcal{L}$ can be induced by functions in $\mathcal{F}$. We refer to $\Delta_{\mathcal{F}}(l)$ as the *growth function.*

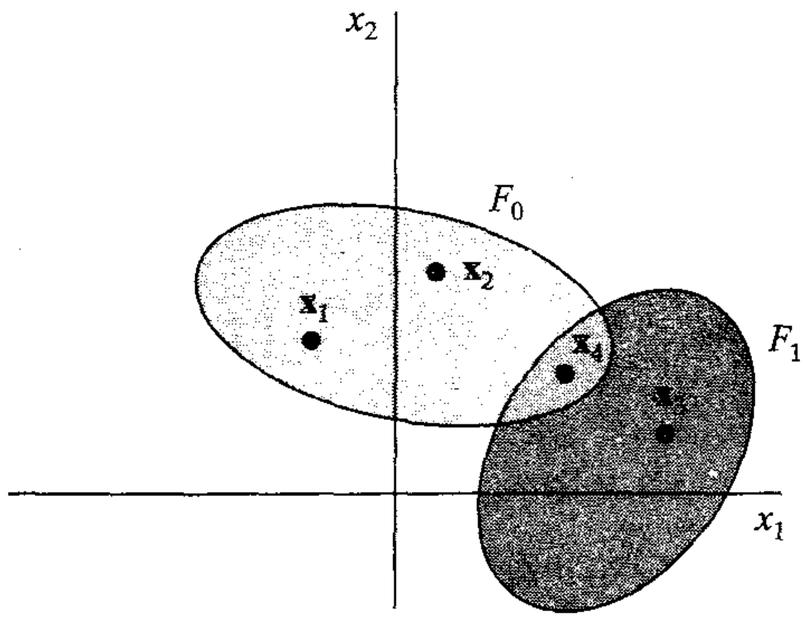


FIGURE 2.23 Diagram for Example 2.1

Example 2.1

Figure 2.23 illustrates a two-dimensional input space $\mathscr{X}$ consisting of four points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$. The decision boundaries of functions F_0 and F_1, indicated in the figure, correspond to the classes (hypotheses) 0 and 1 being true, respectively. From Fig. 2.23 we see that the function F_0 induces the dichotomy

$$\mathscr{D}_0 = \{\mathscr{S}_0 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4\}, \mathscr{S}_1 = \{\mathbf{x}_3\}\}$$

On the other hand, the function F_1 induces the dichotomy

$$\mathscr{D}_1 = \{\mathscr{S}_0 = \{\mathbf{x}_1, \mathbf{x}_2\}, \mathscr{S}_1 = \{\mathbf{x}_3, \mathbf{x}_4\}\}$$

With the set $\mathscr{S}$ consisting of four points, the cardinality $|\mathscr{S}| = 4$. Hence,

$$\Delta_{\mathscr{F}}(\mathscr{S}) = 2^4 = 16$$

■

Returning to the general discussion delineated by the ensemble $\mathscr{F}$ of dichotomies in Eq. (2.85) and the corresponding set of points $\mathscr{L}$ in Eq. (2.86), we may now formally define the VC dimension as (Vapnik and Chervonenkis, 1971; Kearns and Vazirani, 1994; Vidyasagar, 1997; Vapnik, 1998):

> The VC dimension of an ensemble of dichotomies $\mathscr{F}$ is the cardinality of the largest set $\mathscr{L}$ that is shattered by $\mathscr{F}$.

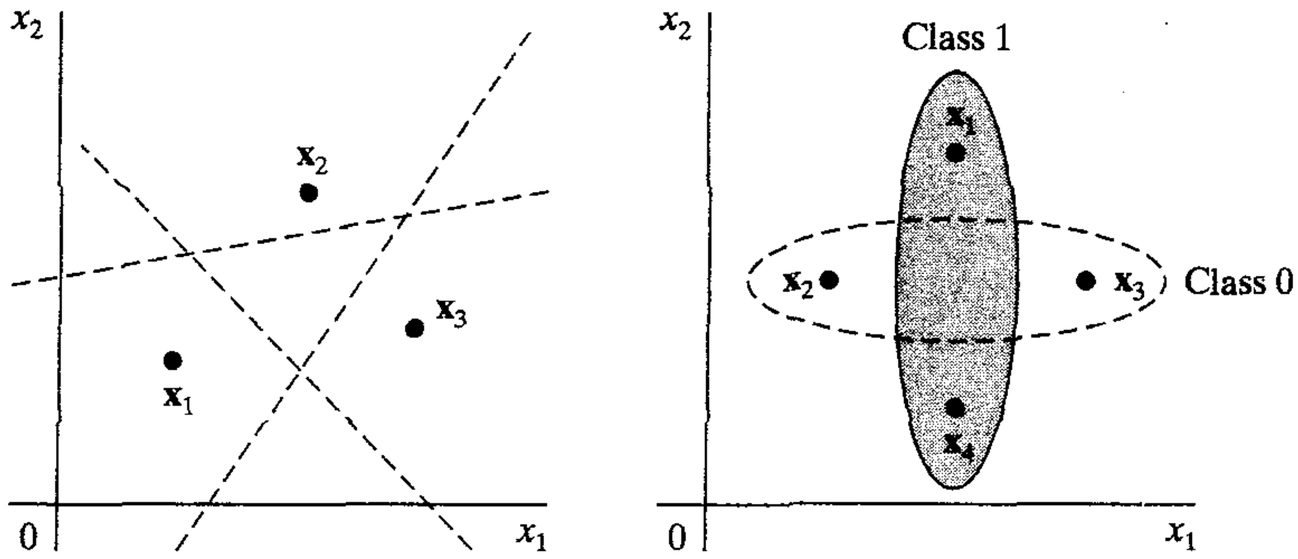
In other words, the VC dimension of $\mathscr{F}$, written as VCdim($\mathscr{F}$), is the largest N such that $\Delta_{\mathscr{F}}(N) = 2^N$. Stated in more familiar terms, the VC dimension of the set of classification functions $\{F(\mathbf{x}, \mathbf{w}): \mathbf{w} \in \mathscr{W}\}$ is the maximum number of training examples that can be learned by the machine without error for all possible binary labelings of the classification functions.

Example 2.2

Consider a simple decision rule in an m-dimensional space $\mathscr{X}$ of input vectors, which is described by

$$\mathscr{F}: y = \varphi(\mathbf{w}^T\mathbf{x} + b) \tag{2.88}$$

where $\mathbf{x}$ is an m-dimensional weight vector and b is a bias. The activation function φ is a threshold function; that is,



(a) (b)

FIGURE 2.24 A pair of two-dimensional data distributions for Example 2.2.

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

The VC dimension of the decision rule in Eq. (2.88) is given by

$$\text{VC dim}(\mathcal{F}) = m + 1 \tag{2.89}$$

To demonstrate this result, consider the situations described in Fig. 2.24 pertaining to a two-dimensional input space (i.e., $m = 2$). In Fig. 2.24a, we have three points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$. Three different possible labelings of these points are included in Fig. 2.24a, from which we readily see that a maximum of three lines can shatter these points. In Fig. 2.24b we have points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$, with points $\mathbf{x}_2$ and $\mathbf{x}_3$ labeled as 0 and points $\mathbf{x}_1$ and $\mathbf{x}_4$ labeled as 1. This time, however, we see that points $\mathbf{x}_1$ and $\mathbf{x}_4$ cannot be shattered from $\mathbf{x}_2$ and $\mathbf{x}_3$ by a line. The VC dimension of the decision rule described in Eq. (2.88) with $m = 2$ is therefore 3, which is in accord with the formula of Eq. (2.89).

■

Example 2.3

With the VC dimension providing a measure of the capacity of a set of classification (indicator) functions, we may be led to expect that a learning machine with many free parameters would have a high VC dimension, whereas a learning machine with few free parameters would have a low VC dimension. We now present a counterexample[13] to this statement.

Consider the one parameter family of indicator functions, defined by

$$f(x, a) = \text{sgn}(\sin(ax)), \qquad a \in \mathbb{R}$$

where $\text{sgn}(\cdot)$ is the signum function. Suppose we choose any number N and the requirement is to find N points that can be shattered. This requirement is satisfied by the set of functions $f(x, a)$ by choosing

$$x_i = 10^{-i}, \qquad i = 1, 2, \ldots, N$$

To separate these data points into two classes determined by the sequence

$$d_1, d_2, \ldots, d_N, \qquad d_i \in \{-1, 1\}$$

it is sufficient that we choose the parameter a according to the formula:

$$a = \pi\left(1 + \sum_{i=1}^{N} \frac{(1 - d_i)10^i}{2}\right)$$

We thus conclude that the VC dimension of the family of indicator functions $f(x, a)$ with a single free paramenter a is infinite.

■

Importance of the VC dimension and its Estimation

The VC dimension is a purely *combinatorial concept* that has no connection with the geometric notion of dimension. It plays a central role in statistical learning theory as will be shown in the material presented in the next two subsections. The VC dimension is also important from a design point of view. Roughly speaking, the number of examples needed to learn a class of interest reliably is proportional to the VC dimension of that class. Therefore, an estimate of the VC dimension is of primary concern.

In some cases the VC dimension is determined by the free parameters of a neural network. In most practical cases, however, it is difficult to evaluate the VC dimension by analytic means. Nevertheless, *bounds* on the VC dimension of neural networks are often *tractable*. In this context, the following two results are of special interest:[14]

1. *Let $\mathcal{N}$ denote an arbitrary feedforward network built up from neurons with a threshold (Heaviside) activation function:*

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

The VC dimension of $\mathcal{N}$ is $O(W \log W)$ where W is the total number of free parameters in the network.

This first result is due to Cover (1968) and Baum and Haussler (1989).

2. *Let $\mathcal{N}$ denote a multilayer feedforward network whose neurons use a sigmoid activation function*

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

The VC dimension of $\mathcal{N}$ is $O(W^2)$, where W is the total number of free parameters in the network.

This second result is due to Koiran and Sontag (1996). They arrived at this result by first showing that networks consisting of two types of neurons, one linear and the other using a threshold activation function, already have a VC dimension proportional to W^2. This is a rather surprising result, since a purely linear network has a VC dimension proportional to W as shown in Example 2.2, while a purely threshold neural network has a VC dimension proportional to $W \log W$ by virtue of result 1. The desired result pertaining to a sigmoid neural network is then obtained by invoking two approximations. First, neurons with threshold activation functions are approximated by sigmoidal ones with large synaptic weights. Second, linear neurons are approximated by sigmoidal neurons with small synaptic weights.

The important point to note here is that multilayer feedforward networks have a *finite* VC dimension.

Constructive Distribution-free Bounds on the Generalization Ability of Learning Machines

At this point in the discussion we find it instructive to consider the specific case of binary pattern classification, for which the desired response is defined by $d \in \{0, 1\}$. In a corresponding way the loss function has only two possible values as shown by

$$L(d, F(\mathbf{x}, \mathbf{w})) = \begin{cases} 0 & \text{if } F(\mathbf{x}, \mathbf{w}) = d \\ 1 & \text{otherwise} \end{cases} \tag{2.90}$$

Under these conditions the risk functional $R(\mathbf{w})$ and the empirical risk functional $R_{\text{emp}}(\mathbf{w})$ defined in Eqs. (2.72) and (2.74) respectively, assume the following interpretations:

- The risk functional $R(\mathbf{w})$ is the *probability of classification error* (i.e., error rate), denoted by $P(\mathbf{w})$.
- The empirical risk functional $R_{\text{emp}}(\mathbf{w})$ is the *training error* (i.e., frequency of errors made during the training session), denoted by $\nu(\mathbf{w})$.

Now, according to the *law of large numbers* (Gray and Davisson, 1986), the empirical frequency of occurrence of an event converges almost surely to the actual probability of that event as the number of trials (assumed to be independent and identically distributed) is made infinitely large. In the context of the discussion presented here, this result means that for any weight vector $\mathbf{w}$, which does not depend on the training set, and for any precision $\epsilon > 0$, the following condition holds (Vapnik, 1982):

$$P\big(|P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon\big) \to 0 \qquad \text{as } N \to \infty \tag{2.91}$$

where N is the size of the training set. Note, however, that the condition (2.91) does not imply that the classification rule (i.e., a particular weight vector $\mathbf{w}$) that minimizes the training error $\nu(\mathbf{w})$ will also minimize the probability of classification error $P(\mathbf{w})$. For a training set of sufficiently large size N, the proximity between $\nu(\mathbf{w})$ and $P(\mathbf{w})$ follows from a stronger condition, which stipulates that the following condition holds for any $\epsilon > 0$ (Vapnik, 1982):

$$P(\sup_{\mathbf{w}} |P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon) \to 0 \qquad \text{as } N \to \infty \tag{2.92}$$

In such a case, we speak of *the uniform convergence of the frequency of training errors to the probability* that $\nu(\mathbf{w}) = P(\mathbf{w})$.

The notion of VC dimension provides a bound on the rate of uniform convergence. Specifically, for the set of classification functions with VC dimension h, the following inequality holds (Vapnik, 1982, 1998):

$$P\Big(\sup_{\mathbf{w}} |P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon\Big) < \left(\frac{2eN}{h}\right)^h \exp(-\epsilon^2 N) \tag{2.93}$$

where N is the size of the training sample and e is the base of the natural logarithm. We want to make the right-hand side of the inequality (2.93) small for large N in order to achieve uniform convergence. The factor $\exp(-\epsilon^2 N)$ is helpful in this regard, since it decays exponentially with increasing N. The remaining factor $(2eN/h)^h$ represents a *bound* on the growth function $\Delta_{\mathcal{F}}(l)$ for the family of functions $\mathcal{F} = \{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathcal{W}\}$ for $l \geq h \geq 1$ as obtained by *Sauer's lemma*.[15] Provided that this function does *not* grow too fast, the right-hand side will go to zero as N goes to infinity; this requirement is satisfied if the VC dimension h is finite. In other words, a finite VC dimension is a necessary and sufficient condition for uniform convergence of the principle of empirical risk minimization. If the input space $\mathcal{X}$ has finite cardinality, any family of dichotomies $\mathcal{F}$ will have finite VC dimension with respect to $\mathcal{X}$, although the reverse is not necessarily true.

Let α denote the probability of occurrence of the event

$$\sup_{\mathbf{w}} |P(\mathbf{w}) - \nu(\mathbf{w})| \geq \epsilon$$

Then, with probability $1 - \alpha$, we may state that for all weight vectors $\mathbf{w} \in \mathcal{W}$ the following inequality holds:

$$P(\mathbf{w}) < \nu(\mathbf{w}) + \epsilon \tag{2.94}$$

Using the bound described in Eq. (2.93) and the definition for the probability α, we may thus set

$$\alpha = \left(\frac{2eN}{h}\right)^h \exp(-\epsilon^2 N) \tag{2.95}$$

Let $\epsilon_0(N, h, \alpha)$ denote the special value of ϵ that satisfies Eq. (2.95). Hence, we readily obtain the following important result (Vapnik, 1992):

$$\epsilon_0(N, h, \alpha) = \sqrt{\frac{h}{N}\left[\log\left(\frac{2N}{h}\right) + 1\right] - \frac{1}{N}\log\alpha} \tag{2.96}$$

We refer to $\epsilon_0(N, h, \alpha)$ as a *confidence interval,* the value of which depends on the size N of the training sample, the VC dimension h, and the probability α.

The bound described in (2.93) with $\epsilon = \epsilon_0(N, h, \alpha)$ is achieved for the worst case $P(\mathbf{w}) = \frac{1}{2}$, but not, unfortunately, for small $P(\mathbf{w})$, which is the case of interest in practice. For small $P(\mathbf{w})$, a more useful bound is obtained by considering a modification of the inequality (2.93) as follows (Vapnik, 1982, 1998):

$$P\left(\sup_{\mathbf{w}} \frac{|P(\mathbf{w}) - \nu(\mathbf{w})|}{\sqrt{P(\mathbf{w})}} > \epsilon\right) < \left(\frac{2eN}{h}\right)^h \exp\left(-\frac{\epsilon^2 N}{4}\right) \tag{2.97}$$

In the literature, different results are reported for the bound in (2.97), depending on which particular form of inequality is used for its derivation. Nevertheless, they all have a similar form. From (2.97) it follows that with probability $1 - \alpha$, and simultaneously for all $\mathbf{w} \in \mathcal{W}$ (Vapnik, 1992, 1998),

$$P(\mathbf{w}) \leq \nu(\mathbf{w}) + \epsilon_1(N, h, \alpha, \nu) \tag{2.98}$$

where $\epsilon_1(N, h, \alpha, \nu)$ is a new confidence interval defined in terms of the former confidence interval $\epsilon_0(N, h, \alpha)$ as follows (see Problem 2.25):

$$\epsilon_1(N, h, \alpha, \nu) = 2\epsilon_0^2(N, h, \alpha)\left(1 + \sqrt{1 + \frac{\nu(\mathbf{w})}{\epsilon_0^2(N, h, \alpha)}}\right) \tag{2.99}$$

This second confidence interval depends on the training error $\nu(\mathbf{w})$. For $\nu(\mathbf{w}) = 0$ it reduces to the special form

$$\epsilon_1(N, h, \alpha, 0) = 4\epsilon_0^2(N, h, \alpha) \tag{2.100}$$

We may now summarize the two bounds we have derived for the rate of uniform convergence:

1. In general, we have the following bound on the rate of uniform convergence:

$$P(\mathbf{w}) \leq \nu(\mathbf{w}) + \epsilon_1(N, h, \alpha, \nu)$$

where $\epsilon_1(N, h, \alpha, \nu)$ is as defined in Eq. (2.99).

2. For a small training error $\nu(\mathbf{w})$ close to zero, we have

$$P(\mathbf{w}) \lesssim \nu(\mathbf{w}) + 4\epsilon_0^2(N, h, \alpha)$$

which provides a fairly precise bound for real-case learning.

3. For a large training error $\nu(\mathbf{w})$ close to unity, we have the bound

$$P(\mathbf{w}) \lesssim \nu(\mathbf{w}) + \epsilon_0(N, h, \alpha)$$

Structural Risk Minimization

The *training error* is the frequency of errors made by a learning machine of some weight vector $\mathbf{w}$ during the training session. Similarly, the *generalization error* is defined as the frequency of errors made by the machine when it is tested with examples not seen before. Here it is assumed that the test data are drawn from the same population as the training data. Let these two errors be denoted by $\nu_{\text{train}}(\mathbf{w})$ and $\nu_{\text{gene}}(\mathbf{w})$, respectively. Note that $\nu_{\text{train}}(\mathbf{w})$ is the *same* as the $\nu(\mathbf{w})$ used in the previous subsection; we used $\nu(\mathbf{w})$ there to simplify the notation. Let h be the VC dimension of a family of classification functions $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathcal{W}\}$ with respect to the input space $\mathcal{X}$. Then, in light of the theory on the rate of uniform convergence, we may state that with probability $1 - \alpha$, for a number of training examples $N > h$, and simultaneously for all classification functions $F(\mathbf{x}, \mathbf{w})$, the generalization error $\nu_{\text{gene}}(\mathbf{w})$ is lower than a *guaranteed risk* defined by the sum of a pair of competing terms (Vapnik, 1992, 1998)

$$\nu_{\text{guarant}}(\mathbf{w}) = \nu_{\text{train}}(\mathbf{w}) + \epsilon_1(N, h, \alpha, \nu_{\text{train}}) \tag{2.101}$$

where the confidence interval $\epsilon_1(N, h, \alpha, \nu_{\text{train}})$ is itself defined by Eq. (2.99). For a fixed number of training examples N, the training error decreases monotonically as the capacity or VC dimension h is increased, whereas the confidence interval increases monotonically. Accordingly, both the guaranteed risk and the generalization error go through a minimum. These trends are illustrated in a generic way in Fig. 2.25. Before the minimum point is reached, the learning problem is *overdetermined* in the sense that the machine capacity h is too small for the amount of training detail. Beyond the mini-

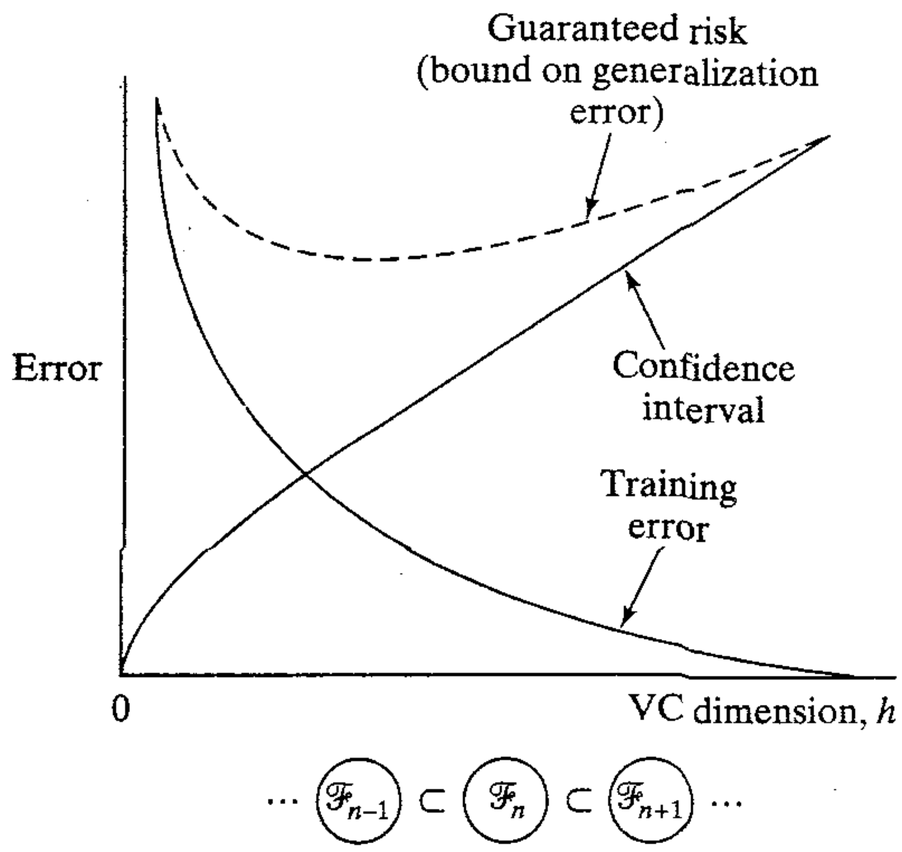


FIGURE 2.25 Illustration of the relationship between training error, confidence interval, and guaranteed risk.

mum point, the learning problem is *underdetermined* because the machine capacity is too large for the amount of training data.

The challenge in solving a supervised learning problem is therefore to realize the best generalization performance by matching the machine capacity to the available amount of training data for the problem at hand. The *method of structural risk minimization* provides an inductive procedure for achieving this goal by making the VC dimension of the learning machine a *control variable* (Vapnik, 1992, 1998). To be specific, consider an ensemble of pattern classifiers $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathcal{W}\}$, and define a nested structure of n such machines

$$\mathcal{F}_k = \{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathcal{W}_k\}, \qquad k = 1, 2, \ldots, n \tag{2.102}$$

such that we have (see Fig. 2.25)

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_n \tag{2.103}$$

where the symbol $\subset$ signifies "is contained in." Correspondingly, the VC dimensions of the individual pattern classifiers satisfy the condition

$$h_1 \leq h_2 \leq \cdots \leq h_n \tag{2.104}$$

which implies that the VC dimension of each pattern classifier is finite. Then, the method of structural risk minimization may proceed as follows:

- The empirical risk (i.e., training error) for each pattern classifier is minimized.
- The pattern classifier $\mathcal{F}^*$ with the smallest guaranteed risk is identified; this particular machine provides the best compromise between the training error (i.e., quality of approximation of the training data) and the confidence interval, (i.e., complexity of the approximating function) which compete with each other.

Our goal is to find a network structure such that decreasing the VC dimension occurs at the expense of the smallest possible increase in training error.

The principle of structural risk minimization may be implemented in a variety of ways. For example, we may vary the VC dimension h by varying the number of hidden

neurons. Specifically, we evaluate an ensemble of fully connected multilayer feedforward networks, in which the number of neurons in one of the hidden layers is increased in a monotonic fashion. The principle of structural risk minimization states that the best network in this ensemble is the one for which the guaranteed risk is the minimum.

The VC dimension is not only central to the principle of structural risk minimization but also to an equally powerful learning model called probably approximately correct (PAC). This latter model, discussed in the next section, completes the last part of the chapter dealing with probabilistic and statistical aspects of learning.

2.15 PROBABLY APPROXIMATELY CORRECT MODEL OF LEARNING

The *probably approximately correct (PAC)* learning model is credited to Valiant (1984). As the name implies, the PAC model is a probabilistic framework for the study of learning and generalization in binary classification systems. It is closely related to supervised learning.

We begin with an environment $\mathscr{X}$. A set of $\mathscr{X}$ is called a *concept,* and a set of subsets of $\mathscr{X}$ is called a *concept class*. An *example* of a concept is an object in the domain of interest, together with a class label. If the example is a member of the concept, we refer to it as a *positive example*; if the object is *not* a member of the concept, we refer to it as a *negative example*. A concept for which examples are provided is called a *target concept*. We may acquire a sequence of training data of length N for a target concept c as shown by

$$\mathscr{T} = \{(\mathbf{x}_i, c(\mathbf{x}_i)\}_{i=1}^{N} \tag{2.105}$$

which may contain repeated examples. The examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are drawn from the environment $\mathscr{X}$ at random, according to some fixed but unknown probability distribution. The following points are also noteworthy in Eq. (2.105):

- The target concept $c(\mathbf{x}_i)$ is treated as a function from $\mathscr{X}$ to $\{0, 1\}$. Moreover, $c(\mathbf{x}_i)$ is assumed to be unknown.
- The examples are usually assumed to be statistically independent, which means that the joint probability density function of any two examples, $\mathbf{x}_i$ and $\mathbf{x}_j$, say, is equal to the product of their individual probability density functions.

In the context of our previous terminology, the environment $\mathscr{X}$ may be identified with the input space of a neural network, and the target concept may be identified with the desired response for the network.

The set of concepts derived from the environment $\mathscr{X}$ is referred to as a *concept space* $\mathscr{C}$. For example, the concept space may contain "the letter A," "the letter B," and so on. Each of these concepts may be coded differently to generate a set of positive examples and a set of negative examples. In the framework of supervised learning, however, we have another set of concepts. A learning machine typically represents a set of functions, with each function corresponding to a specific state. For example, the machine may be designed to recognize "the letter A," "the letter B," and so on. The set of all functions (i.e., concepts) determined by the learning machine is referred to as a *hypothesis space* $\mathscr{G}$. The hypothesis space may or may not be equal to the concept

ship between the generalization performance of a neural network and the *distribution-free, worst-case bound* derived from Vapnik's statistical learning theory. The bound considered therein is defined by Vapnik (1982)

$$\nu_{\text{gene}} \geq O\left(\frac{h}{N} \log\left(\frac{h}{N}\right)\right) \tag{1}$$

where ν_{gene} is the generalization error, h is the VC dimension, and N is the size of the training set. The results presented by Cohn and Tesauro show that the average generalization performance is significantly better than that predicted from Eq. (1).

Second, Holden and Niranjan (1995) extend the earlier study of Cohn and Tesauro by addressing a similar question. However, there are three important differences that should be pointed out:

- All the experiments were performed on neural networks with known exact results or very good bounds on the VC dimension.
- Specific account of the learning algorithm was taken.
- The experiments were based on real-life data.

Although the results reported were found to provide sample complexity predictions of a significantly more practical value than those provided by earlier theories, there are still significant shortcomings in the theory that need to be overcome.

Third, Baum and Haussler (1989) report on the size N of the training sample needed to train a single-layer feedforward network of linear-threshold neurons for good generalization. It is assumed that the training examples are chosen from an arbitrary probability distribution, and that the test examples for evaluating the generalization performance are also drawn from the same distribution. Then, according to Baum and Haussler, the network will almost certainly provide generalization, provided two conditions are satisfied:

(1) The number of errors made on the training set is less than $\epsilon/2$.

(2) The number of examples, N, used in training is

$$N \geq O\left(\frac{W}{\epsilon} \log\left(\frac{W}{\epsilon}\right)\right) \tag{2}$$

where W is the number of synaptic weights in the network. Equation (2) provides a *distribution-free, worst-case bound* on the size N. Here again there can be a huge numerical gap between the actual size of the training sample needed and that calculated from the bound of Eq. (2).

Finally, Bartlett (1997) addresses the issue that in pattern-classification tasks using large neural networks we often find that a network is able to perform successfully with training samples that are considerably smaller in size than the number of weights in the network, as reported in Cohn and Tesauro (1992). In Bartlett's paper it is shown that for such tasks on which neural networks generalize well and if the synaptic weights are not too large, it is the size of the weights rather than the number of weights that determines the generalization performance of the network.

PROBLEMS

Learning Rules

2.1 The delta rule described in Eq. (2.3) and Hebb's rule described in Eq. (2.9) represent two different methods of learning. List the features that distinguish these two rules from each other.

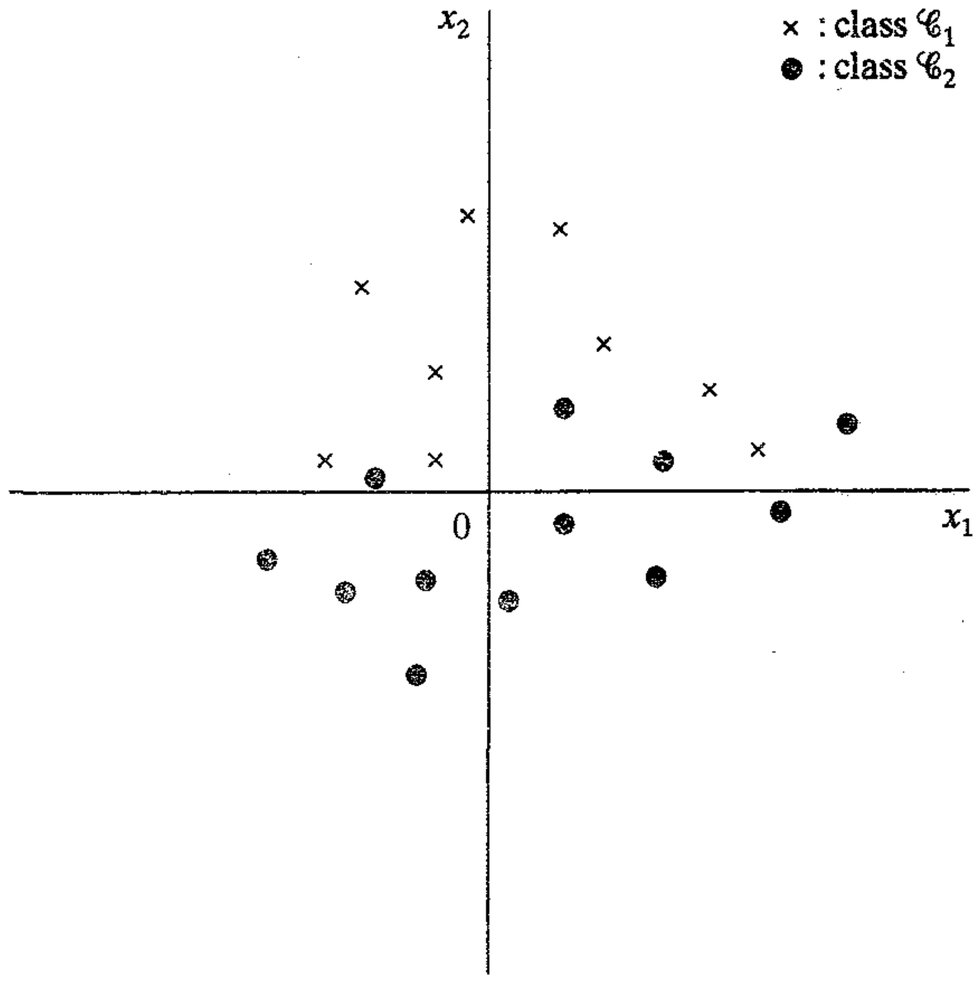


FIGURE P2.3

2.2 The error-correction learning rule may be implemented by using inhibition to subtract the desired response (target value) from the output, and then applying the anti-Hebbian rule (Mitchison, 1989). Discuss this interpretation of error-correction learning.

2.3 Figure P2.3 shows a two-dimensional set of data points. Part of the data points belongs to class $\mathscr{C}_1$ and the other part belongs to class $\mathscr{C}_2$. Construct the decision boundary produced by the nearest neighbor rule applied to this data sample.

2.4 Consider a group of people whose collective opinion on a topic of interest is defined as the weighted average of the individual opinions of its members. Suppose that if, over the course of time, the opinion of a member in the group tends to agree with the collective opinion of the group, the opinion of that member is given more weight. If, on the other hand, the particular member consistently disagrees with the collective opinion of the group, that member's opinion is given less weight. This form of weighting is equivalent to positive-feedback control, which has the effect of producing a consensus of opinion among the group (Linsker, 1988a).

Discuss the analogy between the situation described and Hebb's postulate of learning.

2.5 A generalized form of Hebb's rule is described by the relation

$$\Delta w_{kj}(n) = \alpha F(y_k(n))G(x_j(n)) - \beta w_{kj}(n)F(y_k(n))$$

where $x_j(n)$ and $y_k(n)$ are the presynaptic and postsynaptic signals, respectively; $F(\cdot)$ and $G(\cdot)$ are functions of their respective arguments; and $\Delta w_{kj}(n)$ is the change produced in the synaptic weight w_{kj} at time n in response to the signals $x_j(n)$ and $y_j(n)$. Find (a) the balance point and (b) the maximum depression that are defined by this rule.

2.6 An input signal of unit amplitude is applied repeatedly to a synaptic connection whose initial value is also unity. Calculate the variation in the synaptic weight with time using the following two rules:

(a) The simple form of Hebb's rule described in Eq. (2.9) assuming the learning-rate parameter $\eta = 0.1$.

(b) The covariance rule described in Eq. (2.10), assuming that the presynaptic activity $\bar{x} = 0$ and the postsynaptic activity $\bar{y} = 1.0$.

2.7 The Hebbian synapse described in Eq. (2.9) involves the use of positive feedback. Justify the validity of this statement.

2.8 Consider the covariance hypothesis for self-organized learning described in Eq. (2.10). Assuming ergodicity (i.e., time averages can be substituted for ensemble averages), show that the expected value of Δw_{kj} in Eq. (2.10) may be expressed as

$$E[\Delta w_{kj}] = \eta(E[y_k x_j] - \bar{y}\,\bar{x})$$

How would you interpret this result?

2.9 According to Linsker (1986), Hebb's postulate of learning may be formulated as:

$$\Delta w_{ki} = \eta(y_k - y_o)(x_i - x_o) + a_1$$

where x_j and y_k are the presynaptic and postsynaptic signals, respectively and a_1, η, x_o, and y_o are all constants. Assume that neuron k is linear, as shown by

$$y_k = \sum_j w_{kj} x_j + a_2$$

where a_2 is another constant. Assume the same probability distribution for all the input signals, that is, $E[x_i] = E[x_j] = \mu$. Let the matrix $\mathbf{C}$ denote the covariance matrix of the input signals with its ij-th element defined by

$$c_{ij} = E[(x_i - \mu)(x_j - \mu)]$$

Determine Δw_{ki}.

2.10 Formulate the expression for the output y_j of neuron j in the network of Fig. 2.4. You may use the following:

$x_i = i$th input signal
w_{ji} = synaptic weight from input i to neuron j
c_{kj} = weight of lateral connection from neuron k to neuron j
v_j = induced local field of neuron j
$y_j = \varphi(v_j)$

What is the condition that would have to be satisfied for neuron j to be the winning neuron?

2.11 Repeat Problem 2.10, assuming that each output neuron includes self-feedback.

2.12 The connection pattern for lateral inhibition, namely "excitation close and inhibition further away," may be modeled as the difference between two Gaussian curves. The two curves have the same area, but the positive curve for excitation has a higher and narrower peak than the negative curve for inhibition. That is, we may express the connection pattern as

$$W(x) = \frac{1}{\sqrt{2\pi}\,\sigma_e} e^{-x^2/2\sigma_e^2} - \frac{1}{\sqrt{2\pi}\,\sigma_i} e^{-x^2/2\sigma_i^2}$$

where x is the distance from the neuron responsible for the lateral inhibition. The pattern $W(x)$ is used to scan a page, one half of which is white and the other half is black; the boundary between the two halves is perpendicular to the x-axis.

Plot the output that results from this scanning process with $\sigma_e = 1$ and $\sigma_i = 2$.

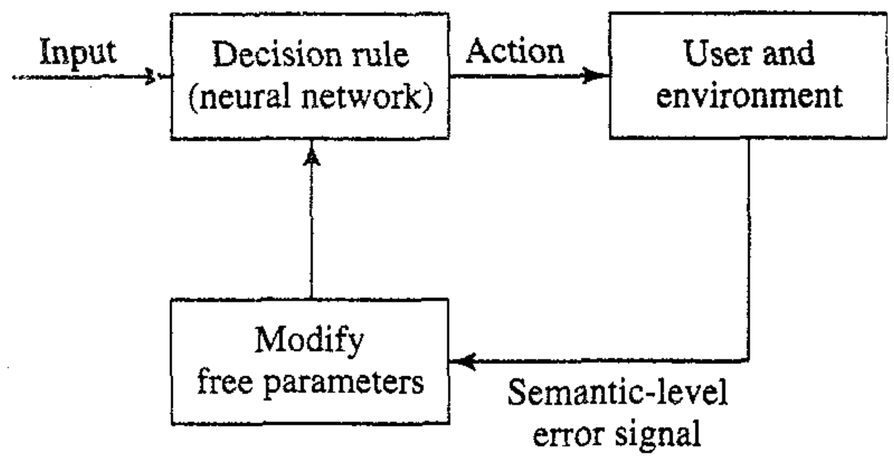


FIGURE P2.13

Learning Paradigms

2.13 Figure P2.13 shows the block diagram of an *adaptive language-acquisition system* (Gorin, 1992). The synaptic connections in the neural network part of the system are strengthened or weakened, depending on feedback as to the appropriateness of the machine's response to input stimuli. This system may be viewed as an example of reinforcement learning. Rationalize the validity of this statement.

2.14 To which of the two paradigms, learning with a teacher and learning without a teacher, do the following algorithms

(a) nearest neighbor rule

(b) k-nearest neighbor rule

(c) Hebbian learning

(d) Boltzmann learning rule

belong? Justify your answers.

2.15 Unsupervised learning can be implemented in an off-line or on-line fashion. Discuss the physical implications of these two possibilities.

2.16 Consider the difficulties that a learning machine faces in assigning credit for the outcome (win, loss, or draw) of a game of chess. Discuss the notions of temporal credit assignment and structural credit assignment in the context of this game.

2.17 A supervised learning task may be viewed as a reinforcement learning task by using as the reinforcement signal some measure of the closeness of the actual response of the system to the desired response. Discuss this relationship between supervised learning and reinforcement learning.

Memory

2.18 Consider the following orthonormal sets of key patterns, applied to a correlation matrix memory:

$$\mathbf{x}_1 = [1, 0, 0, 0]^T$$

$$\mathbf{x}_2 = [0, 1, 0, 0]^T$$

$$\mathbf{x}_3 = [0, 0, 1, 0]^T$$

The respective stored patterns are

$$\mathbf{y}_1 = [5, 1, 0]^T$$

$$\mathbf{y}_2 = [-2, 1, 6]^T$$

$$\mathbf{y}_3 = [-2, 4, 3]^T$$

(a) Calculate the memory matrix $\mathbf{M}$.

(b) Show that the memory associates perfectly.

2.19 Consider again the correlation matrix memory of Problem 2.18. The stimulus applied to the memory is a noisy version of the key pattern $\mathbf{x}_1$, as shown by

$$\mathbf{x} = [0.8, -0.15, 0.15, -0.20]^T$$

(a) Calculate the memory response $\mathbf{y}$.
(b) Show that the response $\mathbf{y}$ is closest to the stored pattern $\mathbf{y}_1$ in a Euclidean sense.

2.20 An autoassociative memory is trained on the following key vectors:

$$\mathbf{x}_1 = \tfrac{1}{4}[-2, -3, \sqrt{3}]^T$$
$$\mathbf{x}_2 = \tfrac{1}{4}[2, -2, -\sqrt{8}]^T$$
$$\mathbf{x}_3 = \tfrac{1}{4}[3, -1, \sqrt{6}]^T$$

(a) Calculate the angles between these vectors. How close are they to orthogonality with respect to each other?
(b) Using the generalization of Hebb's rule (i.e., the outer product rule), calculate the memory matrix of the network. Investigate how close to perfect the memory autoassociates.
(c) A masked version of the key vector $\mathbf{x}_1$, namely,

$$\mathbf{x} = [0, -3, \sqrt{3}]^T$$

is applied to the memory. Calculate the response of the memory, and compare your result with the desired response $\mathbf{x}_1$.

Adaptation

2.21 Figure P2.21 shows the block diagram of an adaptive system. The input signal to the *predictive model* is defined by past values of a process, as shown by

$$\mathbf{x}(n-1) = [x(n-1), x(n-2), \ldots, x(n-m)]$$

The model output, $\hat{x}(n)$, represents an *estimate* of the present value, $x(n)$, of the process. The *comparator* computes the error signal

$$e(n) = x(n) - \hat{x}(n)$$

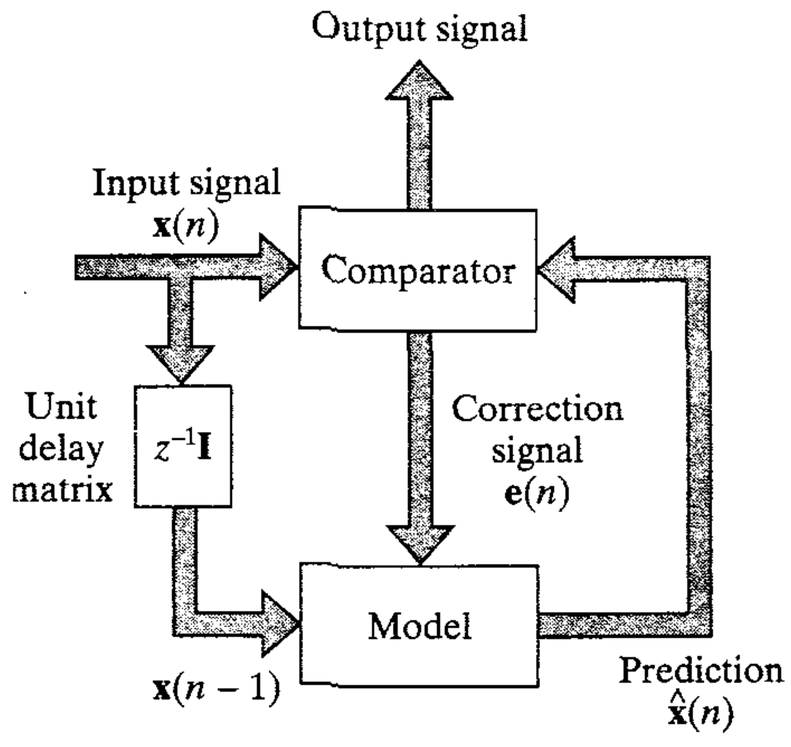


FIGURE P2.21

which in turn applies a correction to the adjustable parameters of the model. It also supplies an output signal for transfer to the next level of neural processing for interpretation. By repeating this operation on a level-by-level basis, the information processed by the system tends to be of progressively higher quality (Mead, 1990).

Fill in the details of the level of signal processing next to that described in Fig. P2.21.

Statistical learning theory

2.22 Following a procedure similar to that described for deriving Eq. (2.62) from (2.61), derive the formula for the ensemble-averaged function $L_{av}(f(\mathbf{x}), F(\mathbf{x}, \mathcal{T}))$ defined in Eq. (2.66).

2.23 In this problem we wish to calculate the VC dimension of a rectangular region aligned with one of the axes in a plane. Show that the VC dimension of this concept is four. You may do this by considering the following:

(a) Four points in a plane and a dichotomy realized by an axis-aligned rectangle.

(b) Four points in a plane, for which there is no realizable dichotomy by an axis-aligned rectangle.

(c) Five points in a plane, for which there is also no realizable dichotomy by an axis-aligned rectangle.

2.24 Consider a linear binary pattern classifier whose input vector $\mathbf{x}$ has dimension m. The first element of the vector $\mathbf{x}$ is constant and set to unity so that the corresponding weight of the classifier introduces a bias. What is the VC dimension of the classifier with respect to the input space?

2.25 The inequality (2.97) defines a bound on the rate of uniform convergence, which is basic to the principle of empirical risk minimization.

(a) Justify the validity of Eq. (2.98), assuming that the inequality (2.97) holds.

(b) Derive Eq. (2.99) that defines the confidence interval ϵ_1.

2.26 Continuing with Example 2.3, show that the four uniformly spaced points of Fig. P2.26 cannot be shattered by the one parameter family of indicator functions $f(x,a)$, $a \in \mathbb{R}$.

2.27 Discuss the relationship between the bias-variance dilemma and structural risk minimization in the context of nonlinear regression.

2.28 **(a)** An algorithm used to train a multilayer feedforward network whose neurons use a sigmoid function is PAC learnable. Justify the validity of this statement.

(b) Can you make a similar statement for an arbitrary neural network whose neurons use a threshold activation function? Justify your answer.

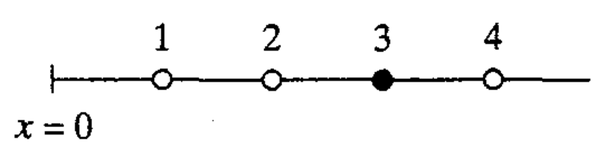


FIGURE P2.26