

where $u_i(n)$ is the convolution of the input $x_i(n)$ and the time window $\theta(n, \tau_{ji}, \sigma_{ji})$. The weight w_{ji} and time delay τ_{ji} of synapse i belonging to neuron j are all to be *learned* in a supervised manner.

This learning may be accomplished by using the standard back-propagation algorithm. Demonstrate this learning process by deriving update equations for w_{ji} , τ_{ji} , and σ_{ji} .

- 13.7 The material presented in Section 13.9 on the temporal back-propagation algorithm deals with synaptic FIR filters of equal length. How could you handle the case of synaptic FIR filters of different lengths?
- 13.8 Discuss how the temporal back-propagation algorithm may be used for the training of a distributed TLFN for single-step prediction.
- 13.9 The discrepancies between the constrained (causal) and unconstrained (noncausal) forms of the temporal back-propagation algorithm are analogous to the standard least-mean-square (LMS) versus delayed LMS algorithm; the LMS algorithm is discussed in Chapter 3. Expand on this analogy.

Computer experiment

- 13.10 In this problem we use the standard back-propagation algorithm to solve a difficult nonlinear prediction problem and compare its performance with that of the LMS algorithm. The time series to be considered is created using a discrete *Volterra model* that has the form

$$x(n) = \sum_i g_i v(n-i) + \sum_i \sum_j g_{ij} v(n-i)v(n-j) + \dots$$

where g_i , g_{ij} , ... are the Volterra coefficients, the $v(n)$ are samples of a white, independently distributed Gaussian noise sequence, and $x(n)$ is the resulting Volterra model output. The first summation term is the familiar moving average (MA) time series model, and the remaining summation terms are nonlinear components of ever-increasing order. In general, the estimation of the Volterra coefficients is considered to be difficult, primarily because of their nonlinear relationship to the data.

In this problem we consider the simple example

$$x(n) = v(n) + \beta v(n-1)v(n-2)$$

The time series has zero mean, is uncorrelated, and therefore has a white spectrum. However, the time series samples are not independent of each other, and therefore a higher-order predictor can be constructed. The variance of the model output is given by

$$\sigma_x^2 = \sigma_v^2 + \beta^2 \sigma_v^4$$

where σ_v^2 is the white noise variance.

- (a) Construct a multilayer perceptron with an input layer of 6 nodes, a hidden layer of 16 neurons, and a single output neuron. A tapped delay line memory is used to feed the input layer of the network. The hidden neurons use sigmoid activation functions limited to the interval $[0, 1]$, whereas the output neuron operates as a linear combiner. The network is trained with the standard back-propagation algorithm having the following description:

Learning-rate parameter	$\eta = 0.001$
Momentum constant	$\alpha = 0.6$
Total number of samples processed	100,000
Number of samples per epoch	1,000
Total number of epochs	100

The white noise variance σ_v^2 is set equal to unity. Hence, with $\beta = 0.5$, we find that the output variance of the predictor is $\sigma_x^2 = 1.25$.

Compute the learning curve of the nonlinear predictor, with the variance of the predictor output $x(n)$ plotted as a function of the number of epochs of training samples up to 200 epochs. For the preparation of each epoch used to perform the training, explore the following two modes:

- (i) The time ordering of the training sample is maintained from one epoch to the next in exactly the same form as it is generated.
- (ii) The ordering of the training sample is randomized from one pattern (state) to another.

Also, use cross-validation (described in Chapter 4) with a validation set of 1000 samples to monitor the learning behavior of the predictor.

- (b) Repeat the experiment using the LMS algorithm designed to perform a linear prediction on an input of six samples. The learning-rate parameter of the algorithm is set equal to $\eta = 10^{-5}$.

- (c) Repeat the entire experiment for $\beta = 1$, $\sigma_v^2 = 2$, and then for $\beta = 2$, $\sigma_v^2 = 5$.

The results of each experiment should reveal that initially the back-propagation algorithm and the LMS algorithm follow essentially a similar path, and then the back-propagation algorithm continues to improve, finally producing a prediction variance close to the prescribed value of σ_x^2 .

Neurodynamics

1 INTRODUCTION

In the previous chapter on temporal processing, we studied short-time memory structures and how to operate a static neural network (e.g., multilayer perceptron) as a dynamic mapper by stimulating it via a memory structure. Another important way in which time can be built into the operation of a neural network in an implicit manner is through the use of *feedback*. There are two basic ways of applying feedback to a neural network: *local feedback* at the level of a single neuron inside the network, and *global feedback* encompassing the whole network. Local feedback is a relatively simple matter to deal with, but global feedback has much more profound implications. In the neural network literature, neural networks with one or more feedback loops are referred to as *recurrent networks*. In this chapter and the next one, we focus attention on recurrent networks that use global feedback.

Feedback is like a two-edged sword, in that when it is applied improperly it can produce harmful effects. In particular, the application of feedback can cause a system that is originally stable to become unstable. Our primary interest in this chapter is in the stability of recurrent networks. Other aspects of recurrent networks are considered in the next chapter.

The subject of neural networks viewed as nonlinear dynamical systems, with particular emphasis on the *stability* problem, is referred to as *neurodynamics* (Hirsch, 1989). An important feature of the stability (or instability) of a nonlinear dynamical system is that it is a property of the whole system. As a corollary, *the presence of stability always implies some form of coordination between the individual parts of the system* (Ashby, 1960). It appears that the study of neurodynamics began in 1938 with the work of Nicholas Rashevsky, in whose visionary mind the application of dynamics to biology came into view for the first time.

The stability of a nonlinear dynamical system is a difficult issue to deal with. When we speak of the stability problem, those with an engineering background usually think in terms of the *bounded input–bounded output (BIBO) stability criterion*. According to this criterion, stability means that the output of a system must *not* grow without bound as a result of a bounded input, initial condition, or unwanted disturbance (Brogan, 1985). The BIBO stability criterion is well suited for a linear dynamical

system. However, it is useless to apply it to neural networks because all such nonlinear dynamical systems are BIBO stable because of the saturating nonlinearity built into the constitution of a neuron.

When we speak of stability in the context of a nonlinear dynamical system, we usually mean *stability in the sense of Lyapunov*. In a celebrated *mémoire* dated 1892, Lyapunov (a Russian mathematician and engineer) presented the fundamental concepts of the stability theory known as the *direct method of Lyapunov*.¹ This method is widely used for the stability analysis of linear and nonlinear systems, both time-invariant and time-varying. As such it is directly applicable to the stability analysis of neural networks. Indeed, much of the material presented in this chapter is concerned with the direct method of Lyapunov. However, its application is no easy task.

The study of neurodynamics may follow one of two routes, depending on the application of interest:

- *Deterministic neurodynamics*, in which the neural network model has a deterministic behavior. In mathematical terms, it is described by a set of *nonlinear differential equations* that define the exact evolution of the model as a function of time (Grossberg, 1967; Cohen and Grossberg, 1983; Hopfield, 1984).
- *Statistical neurodynamics*, in which the neural network model is perturbed by the presence of noise. In this case, we have to deal with *stochastic nonlinear differential equations*, thereby expressing the solution in probabilistic terms (Amari et al., 1972; Peretto, 1984; Amari, 1990). The combination of stochasticity and nonlinearity makes the subject more difficult to handle.

In this chapter we restrict ourselves to deterministic neurodynamics.

Organization of the Chapter

The material in this chapter is organized in three parts. In the first part of the chapter, consisting of Sections 14.2 through 14.6, we provide introductory material. Section 14.2 introduces some fundamental concepts in dynamical systems, followed by a discussion of the stability of equilibrium points in Section 14.3. In Section 14.4 we describe various types of attractors that arise in the study of dynamical systems. In Section 14.5 we revisit the additive model of a neuron that was derived in Chapter 13. In Section 14.6 we discuss the manipulation of attractors as a neural network paradigm.

The second part of the chapter, consisting of Sections 14.7 through 14.11, deals with associative memories. Section 14.7 is devoted to a detailed discussion of Hopfield models and the use of discrete Hopfield models as a content-addressable memory. Section 14.8 presents a computer experiment on this application of the Hopfield network. In Section 14.9 we present the Cohen–Grossberg theorem for nonlinear dynamical systems that includes the Hopfield network and other associative memories as special cases. In Section 14.10, we describe another neurodynamical model known as the brain state-in-a-box model that is well suited for clustering. Section 14.11 presents a computer experiment on this second model.

The last part of the chapter, consisting of Sections 14.12 through 14.14, deals with the topic of chaos. Section 14.12 discusses the invariant characteristics of a chaotic process, followed by a discussion of the closely related topic of dynamic reconstruction

of a chaotic process in Section 14.13. A computer experiment on dynamic reconstruction is presented in Section 14.14.

The chapter concludes with some final remarks in Section 14.15.

2 DYNAMICAL SYSTEMS

In order to proceed with the study of neurodynamics, we need a *mathematical model* for describing the dynamics of a nonlinear system. A model most naturally suited for this purpose is the *state-space model*. According to this model, we think in terms of a set of *state variables* whose values (at any particular instant of time) are supposed to contain sufficient information to predict the future evolution of the system. Let $x_1(t)$, $x_2(t)$, ..., $x_N(t)$ denote the state variables of a nonlinear dynamical system, where continuous time t is the *independent variable* and N is the *order* of the system. For convenience of notation, these state variables are collected into an N -by-1 vector $\mathbf{x}(t)$ called the *state vector* of the system. The dynamics of a large class of nonlinear dynamical systems may then be cast in the form of a system of first-order differential equations written as follows:

$$\frac{d}{dt}x_j(t) = F_j(x_j(t)), \quad j = 1, 2, \dots, N \quad (14.1)$$

where the function $F_j(\cdot)$ is, in general, a nonlinear function of its argument. We may put this system of equations in a compact form by using vector notation, as shown by

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}(\mathbf{x}(t)) \quad (14.2)$$

where the nonlinear function $\mathbf{F}$ is vector valued, each element of which operates on a corresponding element of the state vector:

$$\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_N(t)]^T \quad (14.3)$$

A nonlinear dynamical system for which the vector function $\mathbf{F}(\mathbf{x}(t))$ does not depend explicitly on time t , as in Eq. (14.2), is said to be *autonomous*; otherwise, it is *nonautonomous*.² We will concern ourselves with autonomous systems only.

Regardless of the exact form of the nonlinear function $\mathbf{F}(\cdot)$, the state vector $\mathbf{x}(t)$ must vary with time t ; otherwise, $\mathbf{x}(t)$ is constant and the system is no longer dynamic. We may therefore formally define a dynamical system as follows:

A dynamical system is a system whose state varies with time.

Moreover, we may think of $d\mathbf{x}/dt$ as a “velocity” vector, not in a physical but rather in an abstract sense. Then, according to Eq. (14.2), we may refer to the vector function $\mathbf{F}(\mathbf{x})$ as a velocity vector field or simply as a *vector field*.

State Space

It is informative to view the state-space equation (14.2) as describing the *motion* of a point in an N -dimensional *state space*. The state space can be a *Euclidean space* or a subset thereof. It can also be a non-Euclidean space such as a circle, a sphere, a torus, or

some other *differentiable manifold*. Our interest, however, is confined to a Euclidean space.

The state space is important because it provides us with a visual/conceptual tool for analyzing the dynamics of a nonlinear system described by Eq. (14.2). It does so by focusing our attention on the *global characteristics* of the motion rather than the detailed aspects of analytic or numeric solutions of the equation.

At a particular instant of time t , the observed state of the system (i.e., the state vector $\mathbf{x}(t)$) is represented by a single point in the N -dimensional state space. Changes in the state of the system with time t are represented as a curve in the state space, with each point on the curve carrying (explicitly or implicitly) a label that records the time of observation. This curve is called a *trajectory* or *orbit* of the system. Figure 14.1 illustrates the trajectory of a two-dimensional system. The instantaneous velocity of the trajectory, (i.e., the velocity vector $d\mathbf{x}(t)/dt$) is represented by the *tangent vector*, shown as a dashed line in Fig. 14.1 for time $t = t_0$. We may thus derive a velocity vector for each point of the trajectory.

The family of trajectories, for different initial conditions, is referred to as the *state portrait* of the system. The state portrait includes *all* those points in the state space where the vector field $\mathbf{F}(\mathbf{x})$ is defined. Note that for an autonomous system there will be only one trajectory passing through an initial state. A useful idea that emerges from the state portrait is the *flow* of a dynamical system, defined as the motion of the space of states within itself. In other words, we may imagine the space of states to flow, just like a fluid, around in itself with each point (state) following a particular trajectory (Abraham and Shaw, 1992). The idea of flow as described here is vividly illustrated in the state portrait of Fig. 14.2.

Given a state portrait of a dynamical system, we may construct a field of velocity (tangent) vectors, one for every point of the state space. The picture so obtained in turn

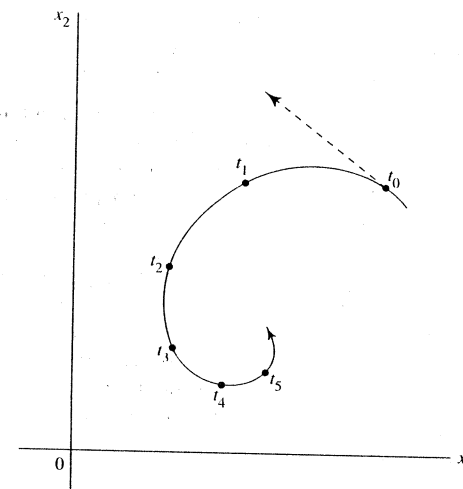


FIGURE 14.1 A two-dimensional trajectory (orbit) of a dynamical system.

FIGURE 14.2 A two-dimensional state (phase) portrait of a dynamical system.

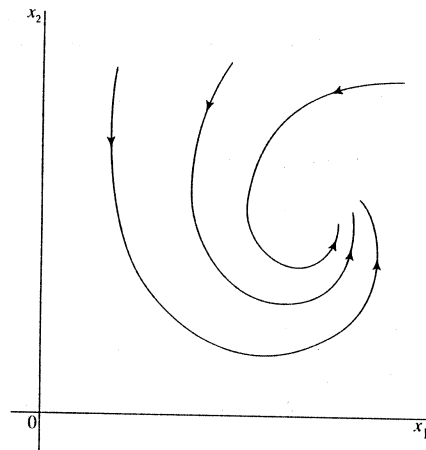
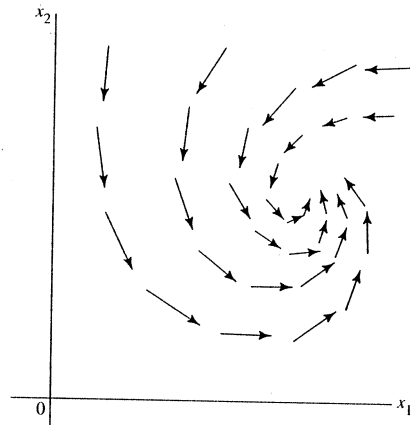


FIGURE 14.3 A two-dimensional vector field of a dynamical system.



provides a portrayal of the vector field of the system. In Fig. 14.3 we show a number of velocity vectors to develop a feeling for what a full field looks like. The usefulness of a vector field thus lies in the fact that it gives us a visual description of the inherent tendency of a dynamical system to move with a habitual velocity at each specific point of a state space.

Lipschitz Condition

For the state-space equation (14.2) to have a solution and for the solution to be unique, we must impose certain restrictions on the vector function $\mathbf{F}(\mathbf{x})$. For convenience of presentation, we have dropped dependence of the state vector $\mathbf{x}$ on time t , a practice

that we follow from time to time. For a solution to exist, it is sufficient that $\mathbf{F}(\mathbf{x})$ be continuous in all of its arguments. However, this restriction by itself does not guarantee uniqueness of the solution. To do so, we must impose a further restriction known as the *Lipschitz condition*. Let $\|\mathbf{x}\|$ denote the *norm* or *Euclidean length* of the vector $\mathbf{x}$. Let $\mathbf{x}$ and $\mathbf{u}$ be a pair of vectors in an open set $\mathcal{M}$ in a normal vector (state) space. Then, according to the Lipschitz condition, there exists a constant K such that (Hirsch and Smale, 1974; E.A. Jackson, 1989)

$$\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{u})\| \leq K \|\mathbf{x} - \mathbf{u}\| \quad (14.4)$$

for all $\mathbf{x}$ and $\mathbf{u}$ in $\mathcal{M}$. A vector function $\mathbf{F}(\mathbf{x})$ that satisfies Eq. (14.4) is said to be *Lipschitz*, and K is called the *Lipschitz constant* for $\mathbf{F}(\mathbf{x})$. Equation (14.4) also implies the continuity of the function $\mathbf{F}(\mathbf{x})$ with respect to $\mathbf{x}$. It follows, therefore, that in the case of autonomous systems, the Lipschitz condition guarantees both the existence and uniqueness of solutions for the state-space equation (14.2). In particular, if all partial derivatives $\partial F_i / \partial x_j$ are finite everywhere, then the function $\mathbf{F}(\mathbf{x})$ satisfies the Lipschitz condition.

Divergence Theorem

Consider a region of volume V and surface S in the state space of an autonomous system and assume a “flow” of points from this region. From our earlier discussion, we recognize that the velocity vector $d\mathbf{x}/dt$ is equal to the vector field $\mathbf{F}(\mathbf{x})$. Provided that the vector field $\mathbf{F}(\mathbf{x})$ within the volume V is “well behaved,” we may apply the *divergence theorem* from vector calculus (Jackson, 1975). Let $\mathbf{n}$ denote a unit vector normal to the surface at dS pointing outward from the enclosed volume. Then, according to the divergence theorem, the relation

$$\int_S (\mathbf{F}(\mathbf{x}) \cdot \mathbf{n}) dS = \int_V (\nabla \cdot \mathbf{F}(\mathbf{x})) dV \quad (14.5)$$

holds between the volume integral of the divergence of $\mathbf{F}(\mathbf{x})$ and the surface integral of the outwardly directed normal component of $\mathbf{F}(\mathbf{x})$. The quantity on the left-hand side of Eq. (14.5) is recognized as the net *flux* flowing out of the region surrounded by the closed surface S . If this quantity is zero, the system is *conservative*; if it is negative, the system is *dissipative*. In light of Eq. (14.5), we may state equivalently that if the divergence $\nabla \cdot \mathbf{F}(\mathbf{x})$ (which is a scalar) is zero the system is conservative, and if it is negative the system is dissipative.

14.3 STABILITY OF EQUILIBRIUM STATES

Consider an autonomous dynamical system described by the state-space equation (14.2). A constant vector $\bar{\mathbf{x}} \in \mathcal{M}$ is said to be an *equilibrium (stationary) state* of the system if the following condition is satisfied:

$$\mathbf{F}(\bar{\mathbf{x}}) = \mathbf{0} \quad (14.6)$$

where $\mathbf{0}$ is the null vector. The velocity vector $d\mathbf{x}/dt$ vanishes at the equilibrium state $\bar{\mathbf{x}}$, and therefore the constant function $\mathbf{x}(t) = \bar{\mathbf{x}}$ is a solution of Eq. (14.2). Furthermore,

because of the uniqueness property of solutions, no other solution curve can pass through the equilibrium state $\bar{\mathbf{x}}$. The equilibrium state is also referred to as a *singular point*, signifying the fact that in the case of an equilibrium point the trajectory will degenerate into the point itself.

In order to develop a deeper understanding of the equilibrium condition, suppose that the nonlinear function $\mathbf{F}(\mathbf{x})$ is smooth enough for the state-space equation (14.2) to be linearized in the neighborhood of $\bar{\mathbf{x}}$. Specifically, let

$$\mathbf{x}(t) = \bar{\mathbf{x}} + \Delta\mathbf{x}(t) \quad (14.7)$$

where $\Delta\mathbf{x}(t)$ is a small deviation from $\bar{\mathbf{x}}$. Then, retaining the first two terms in the Taylor series expansion of $\mathbf{F}(\mathbf{x})$, we may approximate it as follows

$$\mathbf{F}(\mathbf{x}) \approx \bar{\mathbf{x}} + \mathbf{A} \Delta\mathbf{x}(t) \quad (14.8)$$

The matrix $\mathbf{A}$ is the *Jacobian* of the nonlinear function $\mathbf{F}(\mathbf{x})$, evaluated at the point $\mathbf{x} = \bar{\mathbf{x}}$, as shown by

$$\mathbf{A} = \left. \frac{\partial \mathbf{F}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\bar{\mathbf{x}}} \quad (14.9)$$

By substituting Eqs. (14.7) and (14.8) in (14.2), and then using the definition of an equilibrium state, we get

$$\frac{d}{dt} \Delta\mathbf{x}(t) \approx \mathbf{A} \Delta\mathbf{x}(t) \quad (14.10)$$

Provided that the Jacobian $\mathbf{A}$ is nonsingular, that is, the inverse matrix $\mathbf{A}^{-1}$ exists, the approximation described in Eq. (14.10) is sufficient to determine the *local* behavior of the trajectories of the system in the neighborhood of the equilibrium state $\bar{\mathbf{x}}$. If $\mathbf{A}$ is nonsingular, the nature of the equilibrium state is essentially determined by its *eigenvalues*, and may therefore be classified in a corresponding fashion. In particular, when the Jacobian matrix $\mathbf{A}$ has m eigenvalues with positive real parts, we say that the equilibrium state $\bar{\mathbf{x}}$ is of type m .

For the special case of a *second-order system*, we may classify the equilibrium state as summarized in Table 14.1 and illustrated in Fig. 14.4 (Cook, 1986; Arrowsmith and Place, 1990). Without loss of generality, the equilibrium state is assumed to be at the origin of the state space, that is, $\mathbf{x} = \mathbf{0}$. Note also that in the case of a *saddle point*,

TABLE 14.1 Classification of the Equilibrium State of a Second-Order System

Type of Equilibrium State $\bar{\mathbf{x}}$	Eigenvalues of the Jacobian Matrix $\mathbf{A}$
Stable node	Real and negative
Stable focus	Complex conjugate with negative real parts
Unstable node	Real and positive
Unstable focus	Complex conjugate with positive real parts
Saddle point	Real with opposite signs
Center	Conjugate purely imaginary

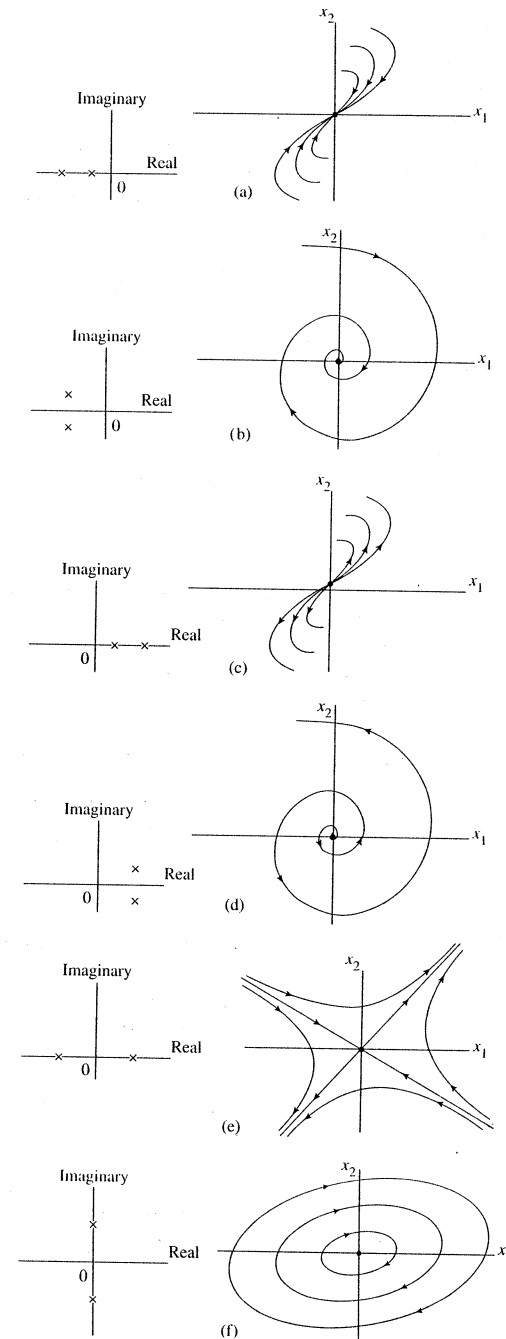


FIGURE 14.4 (a) Stable node. (b) Stable focus. (c) Unstable node. (d) Unstable focus. (e) Saddle point. (f) Center.

shown in Fig. 14.4e, the trajectories going to the saddle point are stable, whereas the trajectories coming from the saddle point are unstable.

Definitions of Stability

Linearization of the state-space equation, as outlined, provides useful information about the *local* stability properties of an equilibrium state. However, for us to be able to investigate the stability of a nonlinear dynamical system in a more detailed fashion, we need precise definitions of the stability and convergence of an equilibrium state.

In the context of an autonomous nonlinear dynamical system with equilibrium state $\bar{\mathbf{x}}$, the definitions of stability and convergence are as follows (Cook, 1986):

DEFINITION 1. The equilibrium state $\bar{\mathbf{x}}$ is said to be uniformly stable if for any given positive ϵ there exists a positive δ such that the condition

$$\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$$

implies

$$\|\mathbf{x}(t) - \bar{\mathbf{x}}\| < \epsilon$$

for all $t > 0$.

This definition states that a trajectory of the system can be made to stay within a small neighborhood of the equilibrium state $\bar{\mathbf{x}}$ if the initial state $\mathbf{x}(0)$ is close to $\bar{\mathbf{x}}$.

DEFINITION 2. The equilibrium state $\bar{\mathbf{x}}$ is said to be convergent if there exists a positive δ such that the condition

$$\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$$

implies that

$$\mathbf{x}(t) \rightarrow \bar{\mathbf{x}} \quad \text{as } t \rightarrow \infty$$

The meaning of this second definition is that if the initial state $\mathbf{x}(0)$ of a trajectory is close enough to the equilibrium state $\bar{\mathbf{x}}$, then the trajectory described by the state vector $\mathbf{x}(t)$ will approach $\bar{\mathbf{x}}$ as time t approaches infinity.

DEFINITION 3. The equilibrium state $\bar{\mathbf{x}}$ is said to be asymptotically stable if it is both stable and convergent.

Here we note that stability and convergence are independent properties. It is only when both properties are satisfied that we have asymptotic stability.

DEFINITION 4. The equilibrium state $\bar{\mathbf{x}}$ is said to be asymptotically stable or globally asymptotically stable if it is stable and all trajectories of the system converge to $\bar{\mathbf{x}}$ as time t approaches infinity.

This definition implies that the system cannot have other equilibrium states, and it requires that every trajectory of the system remain bounded for all time $t > 0$. In other words, global asymptotic stability implies that the system will ultimately settle down to a steady state for any choice of initial conditions.

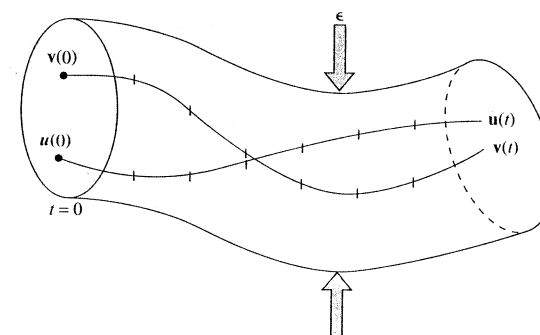


FIGURE 14.5 Illustration of the notion of uniform stability (convergence) of a state vector.

Example 14.1

Let a solution $\mathbf{u}(t)$ of the nonlinear dynamical system described by Eq. (14.2) vary with time t as indicated in Fig. 14.5. For the solution $\mathbf{u}(t)$ to be uniformly stable, we require that $\mathbf{u}(t)$ and any other solution $\mathbf{v}(t)$ remain close to each other for the same values of t (i.e., time "ticks"), as illustrated in Fig. 14.5. This kind of behavior is referred to as an *isochronous correspondence* of the two solutions $\mathbf{v}(t)$ and $\mathbf{u}(t)$ (E.A. Jackson, 1989). The solution $\mathbf{u}(t)$ is convergent provided that for every other solution $\mathbf{v}(t)$ for which $\|\mathbf{v}(0) - \mathbf{u}(0)\| \leq \delta(\epsilon)$ at time $t = 0$, the solutions $\mathbf{v}(t)$ and $\mathbf{u}(t)$ converge to an equilibrium state as t approaches infinity.

Lyapunov's Theorems

Having defined stability and asymptotic stability of an equilibrium state of a dynamical system, the next issue to be considered is that of determining stability. We may obviously do so by actually finding all possible solutions to the state-space equation of the system; however, such an approach is often difficult if not impossible. A more elegant approach is to be found in *modern stability theory*, founded by Lyapunov. Specifically, we may investigate the stability problem by applying the *direct method of Lyapunov*, which makes use of a continuous scalar function of the state vector, called a Lyapunov function.

Lyapunov's theorems on the stability and asymptotic stability of the state-space equation (14.2) describing an autonomous nonlinear dynamical system with state vector $\mathbf{x}(t)$ and equilibrium state $\bar{\mathbf{x}}$ may be stated as follows:

THEOREM 1. The equilibrium state $\bar{\mathbf{x}}$ is stable if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function $V(\mathbf{x})$ such that its derivative with respect to time is negative semidefinite in that region.

THEOREM 2. The equilibrium state $\bar{\mathbf{x}}$ is asymptotically stable if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function $V(\mathbf{x})$ such that its derivative with respect to time is negative definite in that region.

A scalar function $V(\mathbf{x})$ that satisfies these requirements is called a *Lyapunov function* for the equilibrium state $\bar{\mathbf{x}}$.

These theorems require the Lyapunov function $V(\mathbf{x})$ to be a positive definite function. Such a function is defined as: The function $V(\mathbf{x})$ is *positive definite* in the state space $\mathcal{L}$ if, for all $\mathbf{x}$ in $\mathcal{L}$, it satisfies the following requirements:

1. The function $V(\mathbf{x})$ has continuous partial derivatives with respect to the elements of the state vector $\mathbf{x}$
2. $V(\bar{\mathbf{x}}) = 0$
3. $V(\mathbf{x}) > 0$ if $\mathbf{x} \neq \bar{\mathbf{x}}$

Given that $V(\mathbf{x})$ is a Lyapunov function, according to Theorem 1 the equilibrium state $\bar{\mathbf{x}}$ is stable if

$$\frac{d}{dt} V(\mathbf{x}) \leq 0 \quad \text{for } \mathbf{x} \in \mathcal{U} - \bar{\mathbf{x}} \quad (14.11)$$

where $\mathcal{U}$ is a small neighborhood around $\bar{\mathbf{x}}$. Furthermore, according to Theorem 2, the equilibrium state $\bar{\mathbf{x}}$ is asymptotically stable if

$$\frac{d}{dt} V(\mathbf{x}) < 0 \quad \text{for } \mathbf{x} \in \mathcal{U} - \bar{\mathbf{x}} \quad (14.12)$$

The important point of this discussion is that Lyapunov's theorems can be applied without having to solve the state-space equation of the system. Unfortunately, the theorems give no indication of how to find a Lyapunov function; it is a matter of ingenuity and trial and error in each case. In many problems of interest, the energy function can serve as a Lyapunov function. The inability to find a suitable Lyapunov function does not, however, prove instability of the system. The existence of a Lyapunov function is sufficient but not necessary for stability.

The Lyapunov function $V(\mathbf{x})$ provides the mathematical basis for the *global* stability analysis of the nonlinear dynamical system described by Eq. (14.2). On the other hand, the use of Eq. (14.10) based on the Jacobian matrix $\mathbf{A}$ provides the basis for the *local* stability analysis of the system. The global stability analysis is much more powerful in its conclusions than local stability analysis; that is, every globally stable system is also locally stable, but not vice versa.

1 ATTRACTORS

Dissipative systems are generally characterized by the presence of attracting sets or manifolds of dimensionality lower than that of the state space. By a "manifold" we mean a k -dimensional surface embedded in the N -dimensional state space, which is defined by a set of equations:

$$M_j(x_1, x_2, \dots, x_N) = 0, \quad \begin{cases} j = 1, 2, \dots, k \\ k < N \end{cases} \quad (14.13)$$

where $x_1, x_2, \dots, x_N$ are elements of the N -dimensional state vector of the system, and M_j is some function of these elements. These manifolds are called *attractors*³ in that they are bounded subsets to which regions of initial conditions of nonzero state space volume converge as time t increases (Ott, 1993).

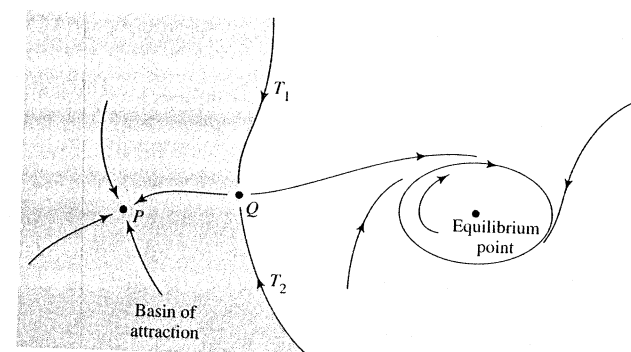


FIGURE 14.6 Illustration of the notion of a basin of attraction, and the idea of a separatrix.

The manifold may consist of a single point in the state space, in which case we speak of a *point attractor*. Alternatively, it may be in the form of a periodic orbit, in which case we speak of a *stable limit cycle*, stable in the sense that nearby trajectories approach it asymptotically. Figure 14.6 illustrates these two types of attractors. Attractors represent the only *equilibrium states* of a dynamical system that may be *observed experimentally*. Note, however, that in the context of attractors an equilibrium state does *not* imply a static equilibrium, nor a steady state. For example, a limit cycle represents a stable state of an attractor, but it varies continuously with time.

In Fig. 14.6 we note that each attractor is encompassed by a distinct region of its own. Such a region is called a *basin (domain) of attraction*. Note also that every initial state of the system is in the basin of some attractor. The boundary separating one basin of attraction from another is called a *separatrix*. In the case of Fig. 14.6, the basin boundary is represented by the union of the trajectory T_1 , the saddle point Q , and the trajectory T_2 .

A limit cycle constitutes the typical form of an oscillatory behavior that arises when an equilibrium point of a nonlinear system becomes unstable. As such, it can arise in nonlinear systems of any order. Nevertheless, limit cycles are particularly characteristic of second-order systems.

Hyperbolic Attractors

Consider a point attractor whose nonlinear dynamical equations are linearized around the equilibrium state $\bar{\mathbf{x}}$ in the manner described in Section 14.2. Let $\mathbf{A}$ denote the Jacobian matrix of the system evaluated at $\mathbf{x} = \bar{\mathbf{x}}$. The attractor is said to be a *hyperbolic attractor* if the eigenvalues of the Jacobian matrix $\mathbf{A}$ all have an absolute value less than 1 (Ott, 1993). For example, the flow of a second-order hyperbolic attractor may have the form shown in Fig. 14.4a or that of Fig. 14.4b; in both cases the eigenvalues of the Jacobian matrix $\mathbf{A}$ have negative real parts. Hyperbolic attractors are of particular interest in the study of a problem known as the vanishing gradients problem that arises in dynamically driven recurrent networks; this problem is discussed in the next chapter.

NEURODYNAMICAL MODELS

Having familiarized ourselves with the behavior of nonlinear dynamical systems, we are now ready to discuss some of the important issues involved in neurodynamics, which we do in this and the following sections. We emphasize that there is no universally agreed upon definition of what we mean by neurodynamics. Rather than try to present such a definition, we will instead define the most general properties of the neurodynamical systems considered in this chapter. In particular, the discussion is limited to neurodynamical systems whose state variables are continuous-valued, and whose equations of motion are described by differential equations or difference equations. The systems of interest possess four general characteristics (Peretto and Niez, 1986; Pineda, 1988a):

1. *A large number of degrees of freedom.* The human cortex is a *highly parallel, distributed system* that is estimated to possess about 10 billion neurons, with each neuron modeled by one or more state variables. It is generally believed that both the computational power and the fault-tolerant capability of such a neurodynamical system are the result of the collective dynamics of the system. The system is characterized by a very large number of coupling constants represented by the strengths (efficacies) of the individual synaptic junctions.
2. *Nonlinearity.* A neurodynamical system is nonlinear. In fact, nonlinearity is essential to create a universal computing machine.
3. *Dissipation.* A neurodynamical system is dissipative. It is therefore characterized by the convergence of the state-space volume onto a manifold of lower dimensionality as time goes on.
4. *Noise.* Finally, noise is an intrinsic characteristic of neurodynamical systems. In real-life neurons, membrane noise is generated at synaptic junctions (Katz, 1966).

The presence of noise necessitates the use of a probabilistic treatment of neural activity, adding another level of complexity to the analysis of neurodynamical systems. A detailed treatment of stochastic neurodynamics is beyond the scope of this book. The effect of noise is therefore ignored in the material that follows.

Additive Model

Consider the noiseless, dynamical model of a neuron shown in Fig. 14.7, the mathematical basis of which was discussed in Chapter 13. In physical terms, the synaptic weights $w_{j1}, w_{j2}, \dots, w_{jN}$ represent *conductances*, and the respective inputs $x_1(t), x_2(t), \dots, x_N(t)$ represent *potentials*; N is the number of inputs. These inputs are applied to a *current-summing junction* characterized as follows:

- Low input resistance
- Unity current gain
- High output resistance

It thus acts as a summing node for the input currents. The total current flowing *toward* the input node of the nonlinear element (activation function) in Fig. 14.7 is therefore

$$\sum_{i=1}^N w_{ji} x_i(t) + I_j$$

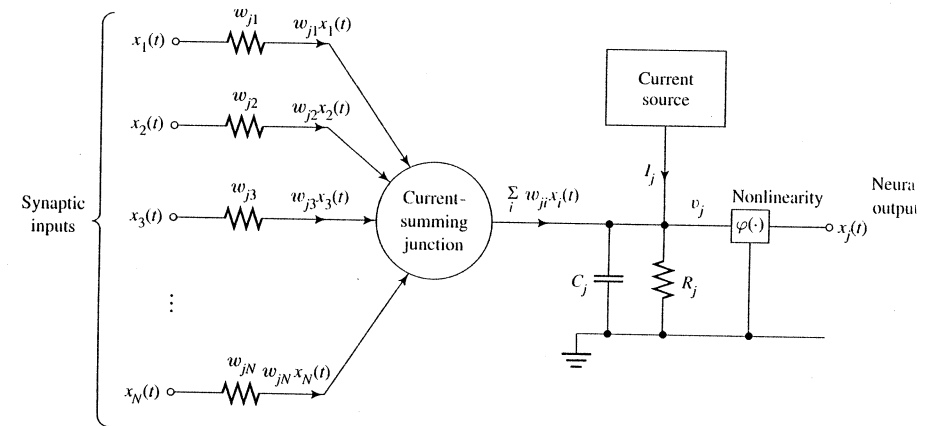


FIGURE 14.7 Additive model of a neuron.

where the first (summation) term is due to the stimuli $x_1(t), x_2(t), \dots, x_N(t)$ acting on the synaptic weights (conductances) $w_{j1}, w_{j2}, \dots, w_{jN}$, respectively, and the second term is due to the current source I_j representing an externally applied bias. Let $v_j(t)$ denote the induced local field at the input of the nonlinear activation function $\varphi(\cdot)$. We may then express the total current flowing *away* from the input node of the nonlinear element as follows:

$$\frac{v_j(t)}{R_j} + C_j \frac{dv_j(t)}{dt}$$

where the first term is due to leakage resistance R_j and the second term is due to leakage capacitance C_j . From *Kirchoff's current law*, we know that the total current flowing toward any node of an electrical circuit is zero. By applying Kirchoff's current law to the input node of the nonlinearity in Fig. 14.7, we get

$$C_j \frac{dv_j(t)}{dt} + \frac{v_j(t)}{R_j} = \sum_{i=1}^N w_{ji} x_i(t) + I_j \quad (14.14)$$

The capacitive term $C_j dv_j(t)/dt$ on the left-hand side of Eq. (14.14) is the simplest way to add dynamics (memory) to the model of a neuron. Given the induced local field $v_j(t)$, we may determine the output of neuron j by using the nonlinear relation

$$x_j(t) = \varphi(v_j(t)) \quad (14.15)$$

The *RC* model described by Eq. (14.14) is commonly referred to as the *additive model*; this terminology is used to discriminate the model from multiplicative (or shunting) models where w_{ji} is dependent on x_i (Grossberg, 1982).

A characteristic feature of the additive model described by Eq. (14.14) is that the signal $x_i(t)$ applied to neuron j by adjoining neuron i is a slowly varying function of time t . The model thus described constitutes the basis of *classical neurodynamics*.⁴

To proceed, consider a *recurrent network* consisting of an interconnection of N neurons, each one of which is assumed to have the same mathematical model

described in Eqs. (14.14) and (14.15). Then, ignoring interneuron propagation time delays, we may define the dynamics of the network by the following system of coupled first-order differential equations:

$$C_j \frac{dv_j(t)}{dt} = -\frac{v_j(t)}{R_j} + \sum_{i=1}^N w_{ji} x_i(t) + I_j, \quad j = 1, 2, \dots, N \quad (14.16)$$

which has the same mathematical form as the state equations (14.1), and which follows from a simple rearrangement of terms in Eq. (14.14). It is assumed that the activation function $\varphi(\cdot)$ relating the output $x_j(t)$ of neuron j to its induced local field $v_j(t)$ is a continuous function and therefore differentiable. A commonly used activation function is the logistic function

$$\varphi(v_j) = \frac{1}{1 + \exp(-v_j)}, \quad j = 1, 2, \dots, N \quad (14.17)$$

A necessary condition for the learning algorithms described in Sections 14.6 through 14.11 to exist is that the recurrent network described by Eqs. (14.15) and (14.16) possesses fixed points (i.e., point attractors).

Related Model

To simplify the exposition, we assume that the time constant $\tau_j = R_j C_j$ of neuron j in Eq. (14.16) is the same for all j . Then, by normalizing time t with respect to the common value of this time constant, and normalizing the w_{ji} and I_j with respect to the R_j , we may recast the model of Eq. (14.16) as follows:

$$\frac{dv_j(t)}{dt} = -v_j(t) + \sum_i w_{ji} \varphi(v_i(t)) + I_j, \quad j = 1, 2, \dots, N \quad (14.18)$$

where we have also incorporated Eq. (14.15). The attractor structure of the system of coupled first-order nonlinear differential equations (14.18) is basically the same as that of a closely related model described by (Pineda, 1987):

$$\frac{dx_j(t)}{dt} = -x_j(t) + \varphi\left(\sum_i w_{ji} x_i(t)\right) + K_j, \quad j = 1, 2, \dots, N \quad (14.19)$$

In the additive model described by Eq. (14.18), the induced local fields $v_1(t), v_2(t), \dots, v_N(t)$ of the individual neurons constitute the state vector. On the other hand, in the related model of Eq. (14.19), the outputs of the neurons $x_1(t), x_2(t), \dots, x_N(t)$ constitute the state vector.

These two neurodynamical models are in fact related to each other by a linear, invertible transformation. Specifically, by multiplying both sides of Eq. (14.19) by w_{kj} , summing with respect to j , and then substituting the transformation

$$v_k(t) = \sum_j w_{kj} x_j(t)$$

we obtain a model of the type described by Eq. (14.18), and so find that the bias terms of the two models are related by

$$I_k = \sum_j w_{kj} K_j$$

The important point to note here is that results concerning the stability of the additive model of Eq. (14.18) are applicable to the related model of Eq. (14.19).

The close relationship between the two neurodynamical models described here is also illustrated in the block diagrams shown in Fig. 14.8. Parts a and b of this figure correspond to the matrix formulations of Eqs. (14.18) and (14.19), respectively; $\mathbf{W}$ is the matrix of synaptic weights, $\mathbf{v}(t)$ is the vector of induced local fields at time t , and $\mathbf{x}(t)$ is

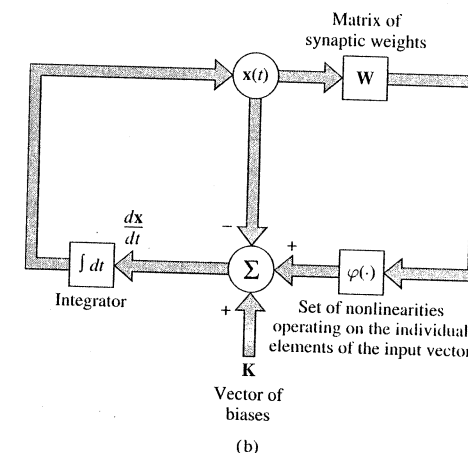
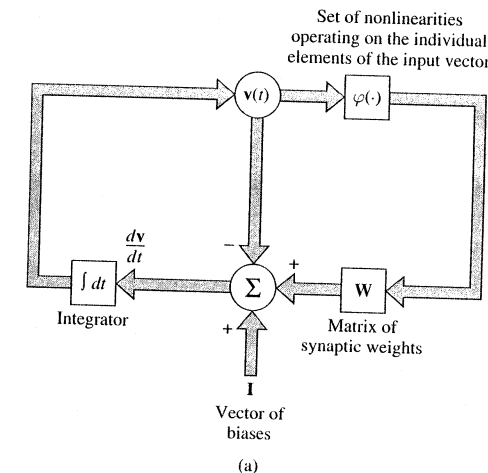


FIGURE 14.8 (a) Block diagram of a neurodynamical system represented by the coupled, first-order differential equations (14.18). (b) Block diagram of related model described by Eqs. (14.19).

the vector of neuronal outputs at time t . The presence of *feedback* in both models is clearly visible in Fig. 14.8.

MANIPULATION OF ATTRACTORS AS A RECURRENT NETWORK PARADIGM

When the number of neurons, N , is very large, the neurodynamical model described by Eq. (14.16) possesses, except for the effect of noise, the general properties outlined earlier in Section 14.5: very many degrees of freedom, nonlinearity, and dissipation. Accordingly, such a neurodynamical model can have complicated attractor structures and therefore exhibit useful computational capabilities.

The identification of attractors with computational objects (e.g., associative memories, input-output mappers) is one of the foundations of neural network paradigms. In order to implement this idea, we must exercise *control* over the locations of the attractors in the state space of the system. A learning algorithm then takes the form of a nonlinear dynamical equation that manipulates the locations of the attractors for the purpose of encoding information in a desired form, or learning temporal structures of interest. In this way, it is possible to establish an intimate relationship between the physics of the machine and the algorithms of the computation.

One way in which the collective properties of a neural network may be used to implement a computational task is by way of the concept of *energy minimization*. The Hopfield network and the brain-state-in-a-box model, to be considered in Sections 14.7 and 14.10, respectively, are well-known examples of such an approach. Both of these models are energy-minimizing networks; they differ from each other in their areas of application. The Hopfield network is useful as a content addressable memory or an analog computer for solving combinatorial-type optimization problems. The brain-state-in-a-box model, on the other hand, is useful for clustering types of applications. More will be said about these applications in subsequent sections of the chapter.

The Hopfield network and brain-state-in-a-box model are examples of an associative memory with no hidden neurons: An associative memory is an important resource for intelligent behavior. Another neurodynamical model is that of an input-output mapper, the operation of which relies on the availability of hidden neurons. In this latter case, the method of steepest descent is often used to minimize a cost function defined in terms of the network parameters, and thereby to change the attractor locations. This latter application of a neurodynamical model is exemplified by the dynamically driven recurrent networks discussed in the next chapter.

14.7 HOPFIELD MODEL

The *Hopfield network (model)* consists of a set of neurons and a corresponding set of unit delays, forming a *multiple-loop feedback system*, as illustrated in Fig. 14.9. The number of feedback loops is equal to the number of neurons. Basically, the output of each neuron is fed back, via a unit delay element, to each of the other neurons in the network. In other words, there is *no* self-feedback in the network; the reason for avoiding the use of self-feedback is explained later.

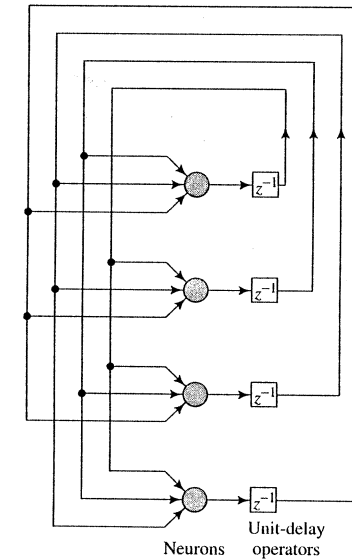


FIGURE 14.9 Architectural graph of a Hopfield network consisting of $N = 4$ neurons.

To study the dynamics of the Hopfield network, we use the neurodynamical model described in Eq. (14.16), which is based on the additive model of a neuron.

Recognizing that $x_i(t) = \varphi_i(v_i(t))$, we may rewrite Eq. (14.16) in the form

$$C_j \frac{d}{dt} v_j(t) = -\frac{v_j(t)}{R_j} + \sum_{i=1}^N w_{ji} \varphi_i(v_i(t)) + I_j, \quad j = 1, \dots, N \quad (14.20)$$

To proceed with the discussion, we make the following assumptions:

1. The matrix of synaptic weights is *symmetric*, as shown by
2. Each neuron has a *nonlinear* activation of its own—hence the use of $\varphi_i(\cdot)$ in Eq. (14.20).
3. The *inverse* of the nonlinear activation function exists, so we may write

$$v = \varphi_i^{-1}(x) \quad (14.22)$$

Let the sigmoid function $\varphi_i(v)$ be defined by the hyperbolic tangent function

$$x = \varphi_i(v) = \tanh\left(\frac{a_i v}{2}\right) = \frac{1 - \exp(-a_i v)}{1 + \exp(-a_i v)} \quad (14.23)$$

which has a slope of $a_i/2$ at the origin as shown by

$$\frac{a_i}{2} = \left. \frac{d\varphi_i}{dv} \right|_{v=0} \quad (14.24)$$

Henceforth we refer to a_i as the *gain* of neuron i .

The inverse output–input relation of Eq. (14.22) may thus be rewritten in the form

$$v = \varphi_i^{-1}(x) = -\frac{1}{a_i} \log \left(\frac{1-x}{1+x} \right) \quad (14.25)$$

The *standard* form of the inverse output–input relation for a neuron of unity gain is defined by

$$\varphi^{-1}(x) = -\log \left(\frac{1-x}{1+x} \right) \quad (14.26)$$

We may rewrite Eq. (14.25) in terms of this standard relation as

$$\varphi_i^{-1}(x) = \frac{1}{a_i} \varphi^{-1}(x) \quad (14.27)$$

Figure 14.10a shows a plot of the standard sigmoidal nonlinearity $\varphi(v)$, and Fig. 14.10b shows the corresponding plot of the inverse nonlinearity $\varphi^{-1}(x)$.

The energy (Lyapunov) function of the Hopfield network in Fig. 14.9 is defined by (Hopfield, 1984)

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} x_i x_j + \sum_{j=1}^N \frac{1}{R_j} \int_0^{x_j} \varphi_j^{-1}(x) dx - \sum_{j=1}^N I_j x_j \quad (14.28)$$

The energy function E defined by Eq. (14.28) may have a complicated *landscape* with many minima. The dynamics of the network are described by a mechanism that *seeks* out those minima.

Hence, differentiating E with respect to time, we get

$$\frac{dE}{dt} = - \sum_{j=1}^N \left(\sum_{i=1}^N w_{ij} x_i - \frac{v_j}{R_j} + I_j \right) \frac{dx_j}{dt} \quad (14.29)$$

The quantity inside the parentheses on the right-hand side of Eq. (14.29) is recognized as $C_j dv_j/dt$ by virtue of the neurodynamical equation (14.20). We may thus simplify Eq. (14.29) to

$$\frac{dE}{dt} = - \sum_{j=1}^N C_j \left(\frac{dv_j}{dt} \right) \frac{dx_j}{dt} \quad (14.30)$$

We now recognize the inverse relation that defines v_j in terms of x_j . The use of Eq. (14.22) in (14.30) yields

$$\begin{aligned} \frac{dE}{dt} &= - \sum_{j=1}^N C_j \left[\frac{d}{dt} \varphi_j^{-1}(x_j) \right] \frac{dx_j}{dt} \\ &= - \sum_{j=1}^N C_j \left(\frac{dx_j}{dt} \right)^2 \left[\frac{d}{dx_j} \varphi_j^{-1}(x_j) \right] \end{aligned} \quad (14.31)$$

From Fig. 14.10b we see that the inverse output–input relation $\varphi_j^{-1}(x_j)$ is a *monotonically increasing* function of the output x_j . It follows therefore that

$$\frac{d}{dx_j} \varphi_j^{-1}(x_j) \geq 0 \quad \text{for all } x_j \quad (14.32)$$

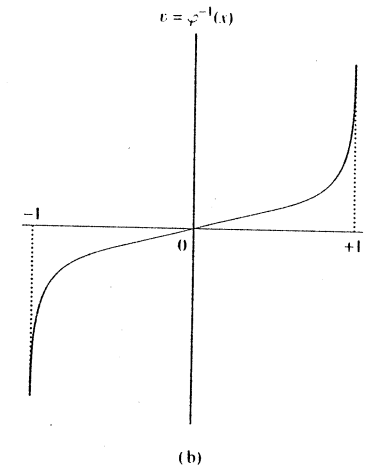
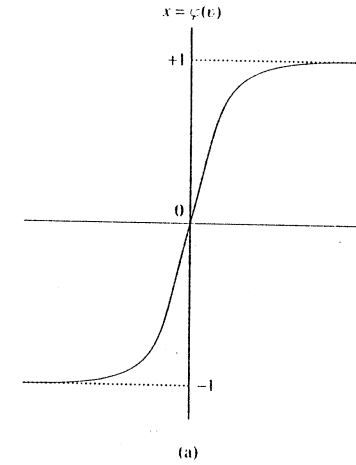


FIGURE 14.10 Plots of (a) the standard sigmoidal nonlinearity, and (b) its inverse.

We also note that

$$\left(\frac{dx_j}{dt} \right)^2 \geq 0 \quad \text{for all } x_j \quad (14.33)$$

Hence, all the factors that make up the sum on the right-hand side of Eq. (14.31) are nonnegative. In other words, for the energy function E defined in Eq. (14.28), we have

$$\frac{dE}{dt} \leq 0$$

From the definition of Eq. (14.28), we note that the function E is bounded. Accordingly, we may make the following two statements:

1. The energy function E is a Lyapunov function of the continuous Hopfield model.
2. The model is stable in accordance with Lyapunov's Theorem 1.

In other words, the time evolution of the continuous Hopfield model described by the system of nonlinear first-order differential equations (14.20) represents a trajectory in state space, which seeks out the minima of the energy (Lyapunov) function E and comes to a stop at such fixed points. From Eq. (14.31) we also note that the derivative dE/dt vanishes only if

$$\frac{d}{dt}x_j(t) = 0 \quad \text{for all } j$$

We may thus go one step further and write

$$\frac{dE}{dt} < 0 \quad \text{except at a fixed point} \quad (14.34)$$

Equation (14.34) provides the basis for the following theorem:

The (Lyapunov) energy function E of a Hopfield network is a monotonically decreasing function of time.

Accordingly, the Hopfield network is globally asymptotically stable; the attractor fixed-points are the minima of the energy function, and vice versa.

Relation between the Stable States of the Discrete and Continuous Versions of the Hopfield Model

The Hopfield network may be operated in a continuous mode or discrete mode, depending on the model adopted for describing the neurons. The continuous mode of operation is based on an additive model, as previously described. On the other hand, the discrete mode of operation is based on the McCulloch–Pitts model. We may readily establish the relationship between the stable states of the continuous Hopfield model and those of the corresponding discrete Hopfield model by redefining the input–output relation for a neuron such that we may satisfy two simplifying characteristics:

1. The output of a neuron has the asymptotic values

$$x_j = \begin{cases} +1 & \text{for } v_j = \infty \\ -1 & \text{for } v_j = -\infty \end{cases} \quad (14.35)$$

2. The midpoint of the activation function of a neuron lies at the origin, as shown by

$$\phi_j(0) = 0 \quad (14.36)$$

Correspondingly, we may set the bias I_j equal to zero for all j .

In formulating the energy function E for a continuous Hopfield model, the neurons are permitted to have self-loops. A discrete Hopfield model, on the other hand, need not have self-loops. We may therefore simplify our discussion by setting $w_{jj} = 0$ for all j in both models.

In light of these observations, we may redefine the energy function of a continuous Hopfield model given in Eq. (14.28) as follows:

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} x_i x_j + \sum_{j=1}^N \frac{1}{R_j} \int_0^{x_j} \phi_j^{-1}(x) dx \quad (14.37)$$

The inverse function $\phi_j^{-1}(x)$ is defined by Eq. (14.27). We may thus rewrite the energy function of Eq. (14.37) as follows:

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} x_i x_j + \sum_{j=1}^N \frac{1}{a_j R_j} \int_0^{x_j} \phi^{-1}(x) dx \quad (14.38)$$

The integral

$$\int_0^{x_j} \phi^{-1}(x) dx$$

has the standard form plotted in Fig. 14.11. Its value is zero for $x_j = 0$, and positive otherwise. It assumes a very large value as x_j approaches ± 1 . If, however, the gain a_j of neuron j becomes infinitely large (i.e., the sigmoidal nonlinearity approaches the idealized hard-limiting form), the second term of Eq. (14.38) becomes negligibly small. In the limiting case when $a_j = \infty$ for all j , the maxima and minima of the continuous Hopfield model become identical with those of the corresponding discrete Hopfield model. In the latter case, the energy (Lyapunov) function is defined simply by

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} x_i x_j \quad (14.39)$$

where the j th neuron state $x_j = \pm 1$. We conclude, therefore, that the only stable points of the very high-gain, continuous, deterministic Hopfield model correspond to the stable points of the discrete stochastic Hopfield model.

When, however, each neuron j has a large but finite gain a_j , we find that the second term on the right-hand side of Eq. (14.38) makes a noticeable contribution to the

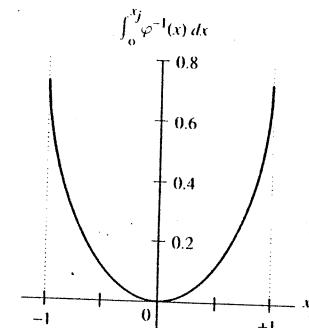


FIGURE 14.11 Plot of the integral $\int_0^{x_j} \phi^{-1}(x) dx$.

energy function of the continuous model. In particular, this contribution is large and positive near all surfaces, edges, and corners of the unit hypercube that defines the state space of the model. On the other hand, the contribution is negligibly small at points that are far removed from the surface. Accordingly, the energy function of such a model has its maxima at corners, but the minima are displaced slightly toward the interior of the hypercube (Hopfield, 1984).

Figure 14.12 depicts the *energy contour map* or *energy landscape* for a continuous Hopfield model using two neurons. The outputs of the two neurons define the two axes of the map. The lower left- and upper right-hand corners of Fig. 14.12 represent stable minima for the limiting case of infinite gain; the minima for the case of finite gain are displaced inward. The flow to the fixed points (i.e., stable minima) may be interpreted as the solution to the minimization of the energy function E defined in Eq. (14.28).

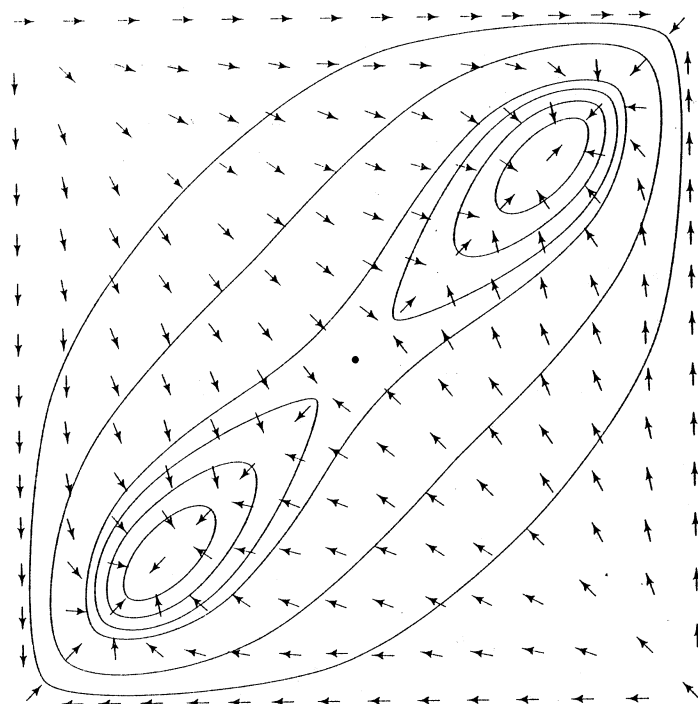


FIGURE 14.12 An energy contour map for a two-neuron, two-stable-state system. The ordinate and abscissa are the outputs of the two neurons. Stable states are located near the lower left and upper right corners, and unstable extrema at the other two corners. The arrows show the motion of the state. This motion is not generally perpendicular to the energy contours. (From J.J. Hopfield, 1984, with permission of the National Academy of Sciences of the U.S.A.)

The Discrete Hopfield Model as a Content-Addressable Memory

The Hopfield network has attracted a great deal of attention in the literature as a *content-addressable memory*. In this application, we know the fixed points of the network *a priori* in that they correspond to the patterns to be stored. However, the synaptic weights of the network that produce the desired fixed points are unknown, and the problem is how to determine them. The primary function of a content-addressable memory is to retrieve a pattern (item) stored in memory in response to the presentation of an incomplete or noisy version of that pattern. To illustrate the meaning of this statement in a succinct way, we can do no better than to quote from Hopfield's 1982 paper:

Suppose that an item stored in memory is "H.A. Kramers & G.H. Wannier *Phys. Rev.* 60, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "Kramers & Wannier (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Wannier, (1941)."

An important property of a content-addressable memory is therefore the ability to retrieve a stored pattern, given a reasonable subset of the information content of that pattern. Moreover, a content-addressable memory is *error-correcting* in the sense that it can override inconsistent information in the cues presented to it.

The essence of a content-addressable memory (CAM) is to map a fundamental memory ξ_μ onto a fixed (stable) point x_μ of a dynamic system, as illustrated in Fig. 14.13. Mathematically, we may express this mapping in the form

$$\xi_\mu \Rightarrow x_\mu$$

The arrow from left to right describes an *encoding* operation, whereas the arrow from right to left describes a *decoding* operation. The attractor fixed points of the state space of the network are the *fundamental memories* or *prototype states* of the network.

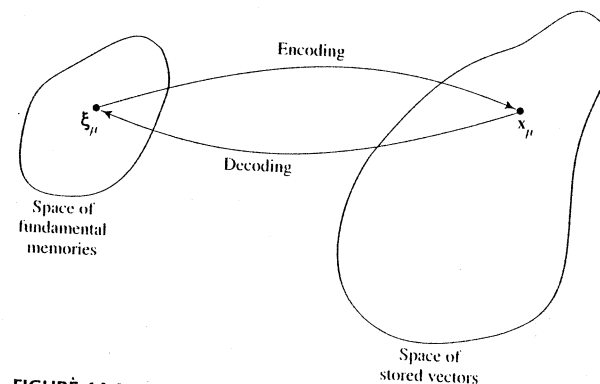


FIGURE 14.13 Illustration of the encoding-decoding performed by a recurrent network.

Suppose now that the network is presented a pattern containing partial but sufficient information about one of the fundamental memories. We may then represent that particular pattern as a starting point in the state space. In principle, provided that the starting point is close to the fixed point representing the memory being retrieved (i.e., it lies inside the basin of attraction belonging to the fixed point), the system should evolve with time and finally converge onto the memory state itself. At that point the entire memory is generated by the network. Consequently, the Hopfield network has an *emergent* property, which helps it retrieve information and cope with errors.

With the Hopfield model using the formal neuron of McCulloch and Pitts (1943) as its basic processing unit, each such neuron has two states determined by the level of the induced local field acting on it. The "on" or "firing" state of neuron i is denoted by the output $x_i = +1$, and the "off" or "quiescent" state is represented by $x_i = -1$. For a network made up of N neurons, the *state* of the network is thus defined by the vector

$$\mathbf{x} = [x_1, x_2, \dots, x_N]^T$$

With $x_i = \pm 1$, the state of neuron i represents one *bit* of information, and the N -by-1 state vector $\mathbf{x}$ represents a binary word of N bits of information.

The induced local field v_j of neuron j is defined by

$$v_j = \sum_{i=1}^N w_{ji} x_i + b_j \quad (14.40)$$

where b_j is a fixed *bias* applied externally to neuron j . Hence, neuron j modifies its state x_j according to the *deterministic rule*

$$x_j = \begin{cases} +1 & \text{if } v_j > 0 \\ -1 & \text{if } v_j < 0 \end{cases}$$

This relation may be rewritten in the compact form

$$x_j = \text{sgn}[v_j]$$

where sgn is the *signum function*. What if v_j is exactly zero? The action taken here can be quite arbitrary. For example, we may set $x_j = \pm 1$ if $v_j = 0$. However, we will use the following convention: If v_j is zero, neuron j remains in its previous state, regardless of whether it is on or off. The significance of this assumption is that the resulting flow diagram is symmetrical, as will be illustrated later.

There are two phases to the operation of the discrete Hopfield network as a content-addressable memory, namely the storage phase and the retrieval phase, as described here.

1. Storage Phase. Suppose that we wish to store a set of N -dimensional vectors (binary words), denoted by $\{\xi_\mu | \mu = 1, 2, \dots, M\}$. We call these M vectors *fundamental memories*, representing the patterns to be memorized by the network. Let $\xi_{\mu,i}$ denote the i th element of the fundamental memory ξ_μ , where the class $\mu = 1, 2, \dots, M$. According to the *outer product rule* of storage, that is, the generalization of *Hebb's postulate of learning*, the synaptic weight from neuron i to neuron j is defined by

$$w_{ji} = \frac{1}{N} \sum_{\mu=1}^M \xi_{\mu,i} \xi_{\mu,j} \quad (14.41)$$

The reason for using $1/N$ as the constant of proportionality is to simplify the mathematical description of information retrieval. Note also that the learning rule of Eq. (14.41) is a "one shot" computation. In the normal operation of the Hopfield network, we set

$$w_{ii} = 0 \quad \text{for all } i \quad (14.42)$$

which means that the neurons have *no* self-feedback. Let $\mathbf{W}$ denote the N -by- N *synaptic weight matrix* of the network, with w_{ji} as its j th element. We may then combine Eqs. (14.41) and (14.42) into a single equation written in matrix form as follows:

$$\mathbf{W} = \frac{1}{N} \sum_{\mu=1}^M \xi_\mu \xi_\mu^T - \mathbf{I} \quad (14.43)$$

where $\xi_\mu \xi_\mu^T$ represents the outer product of the vector ξ_μ with itself, and $\mathbf{I}$ denotes the identity matrix. From these defining equations of the synaptic weights/weight matrix, we may reconfirm the following:

- The output of each neuron in the network is fed back to all other neurons.
- There is no self-feedback in the network (i.e., $w_{ii} = 0$).
- The weight matrix of the network is symmetric as shown by (see Eq. (14.21))

$$\mathbf{W}^T = \mathbf{W} \quad (14.44)$$

2. Retrieval Phase. During the retrieval phase, an N -dimensional vector ξ_{probe} , called a *probe*, is imposed on the Hopfield network as its state. The probe vector has elements equal to ± 1 . It typically represents an incomplete or noisy version of a fundamental memory of the network. Information retrieval then proceeds in accordance with a *dynamical rule* in which each neuron j of the network *randomly* but at some fixed rate examines the induced local field v_j (including any nonzero bias b_j) applied to it. If, at that instant of time, v_j is greater than zero, neuron j will switch its state to $+1$ or remain in that state if it is already there. Similarly, if v_j is less than zero, neuron j will switch its state to -1 or remain in that state if it is already there. If v_j is exactly zero, neuron j is left in its previous state, regardless of whether it is on or off. The state updating from one iteration to the next is therefore deterministic, but the selection of a neuron to perform the updating is done randomly. The *asynchronous* (serial) updating procedure described here is continued until there are no further changes to report. That is, starting with the probe vector $\mathbf{x}$, the network finally produces a time invariant state vector $\mathbf{y}$ whose individual elements satisfy the *condition for stability*:

$$y_i = \text{sgn} \left(\sum_{j=1}^N w_{ji} y_j + b_i \right), \quad i = 1, 2, \dots, N \quad (14.45)$$

or, in matrix form,

$$\mathbf{y} = \text{sgn}(\mathbf{W}\mathbf{y} + \mathbf{b}) \quad (14.46)$$

where $\mathbf{W}$ is the synaptic weight matrix of the network, and $\mathbf{b}$ is the externally applied *bias vector*. The stability condition described here is also referred to as the *alignment condition*. The state vector $\mathbf{y}$ that satisfies it is called a *stable state* or *fixed point* of the state space of the system. We may therefore make the statement that the Hopfield network will always converge to a stable state when the retrieval operation is performed *asynchronously*.⁵

TABLE 14.2 Summary of the Hopfield Model

1. *Learning.* Let $\xi_1, \xi_2, \dots, \xi_M$ denote a known set of N -dimensional fundamental memories. Use the outer product rule (i.e., Hebb's postulate of learning) to compute the synaptic weights of the network:

$$w_{ji} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^M \xi_{\mu,j} \xi_{\mu,i} & j \neq i \\ 0, & j = i \end{cases}$$

where w_{ji} is the synaptic weight from neuron i to neuron j . The elements of the vector ξ_{μ} equal ± 1 . Once they are computed, the synaptic weights are kept fixed.

2. *Initialization.* Let ξ_{probe} denote an unknown N -dimensional input vector (probe) presented to the network. The algorithm is initialized by setting

$$x_j(0) = \xi_{j,\text{probe}}, \quad j = 1, \dots, N$$

where $x_j(0)$ is the state of neuron j at time $n = 0$, and $\xi_{j,\text{probe}}$ is the j th element of the probe vector ξ_{probe} .

3. *Iteration Until Convergence.* Update the elements of state vector $\mathbf{x}(n)$ asynchronously (i.e., randomly and one at a time) according to the rule

$$x_j(n+1) = \text{sgn} \left[\sum_{i=1}^N w_{ji} x_i(n) \right], \quad j = 1, 2, \dots, N$$

Repeat the iteration until the state vector $\mathbf{x}$ remains unchanged.

4. *Outputting.* Let $\mathbf{x}_{\text{fixed}}$ denote the fixed point (stable state) computed at the end of step 3. The resulting output vector $\mathbf{y}$ of the network is

$$\mathbf{y} = \mathbf{x}_{\text{fixed}}$$

Step 1 is the storage phase, and steps 2 through 4 constitute the retrieval phase.

Table 14.2 presents a summary of the steps involved in the storage phase and retrieval phase of operating a Hopfield network.

Example 14.2

To illustrate the emergent behavior of the Hopfield model, consider the network of Fig. 14.14a, which consists of three neurons. The weight matrix of the network is

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

which is legitimate since it satisfies the conditions of Eqs. (14.42) and (14.44). The bias applied to each neuron is assumed to be zero. With three neurons in the network, there are $2^3 = 8$ possible states to consider. Of these eight states, only the two states $(1, -1, 1)$ and $(-1, 1, -1)$ are stable; the remaining six states are all unstable. We say that these two particular states are *stable* because they both satisfy the alignment condition of Eq. (14.46). For the state vector $(1, -1, 1)$ we have

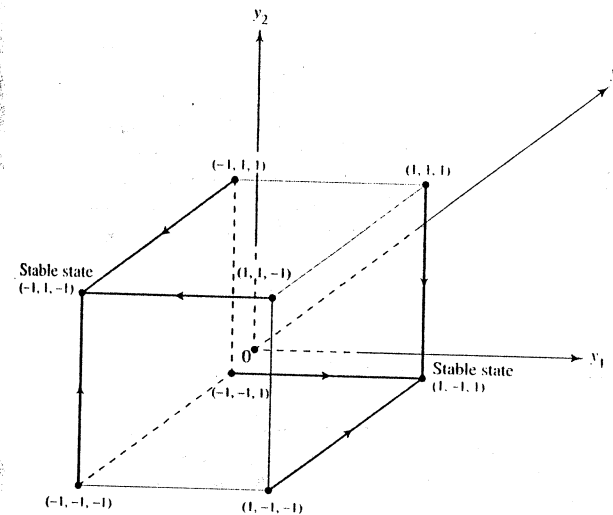
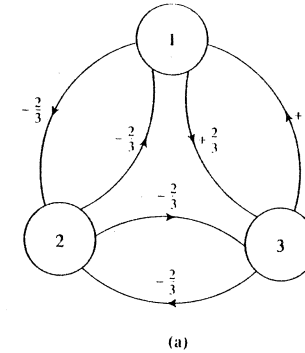


FIGURE 14.14
(a) Architectural graph of Hopfield network for $N = 3$ neurons. (b) Diagram depicting the two stable states and flow of the network.

$$\mathbf{W}\mathbf{y} = \frac{1}{3} \begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix} \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} +4 \\ -4 \\ +4 \end{bmatrix}$$

Hard limiting this result yields

$$\text{sgn}[\mathbf{W}\mathbf{y}] = \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \mathbf{y}$$

Similarly, for the state vector $(-1, 1, -1)$ we have

$$\mathbf{W}\mathbf{y} = \frac{1}{3} \begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} -4 \\ +4 \\ -4 \end{bmatrix}$$

which, after hard limiting, yields

$$\text{sgn}[Wy] = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} = y$$

Hence, both of these state vectors satisfy the alignment condition.

Moreover, following the asynchronous updating procedure summarized in Table 14.2, we get the flow described in Fig. 14.14b. This flow map exhibits symmetry with respect to the two stable states of the network, which is intuitively satisfying. This symmetry is the result of leaving a neuron in its previous state if the induced local field acting on it is exactly zero.

Figure 14.14b also shows that if the network of Fig. 14.14a is in the initial state $(1, 1, 1)$, $(-1, -1, 1)$, or $(1, -1, -1)$, it will converge onto the stable state $(1, -1, 1)$ after one iteration. If the initial state is $(-1, -1, -1)$, $(-1, 1, 1)$, or $(1, 1, -1)$, it will converge onto the second stable state $(-1, 1, -1)$.

The network therefore has two fundamental memories, $(1, -1, 1)$ and $(-1, 1, -1)$, representing the two stable states. The application of Eq. (14.43) yields the synaptic weight matrix

$$\begin{aligned} \mathbf{W} &= \frac{1}{3} \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} [+1, -1, +1] + \frac{1}{3} \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} [-1, +1, -1] - \frac{2}{3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \frac{1}{3} \begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix} \end{aligned}$$

which checks with the synaptic weights shown in Fig. 14.14a.

The error correcting capability of the Hopfield network is readily seen by examining the flow map of Fig. 14.14b:

1. If the probe vector ξ_{probe} applied to the network equals $(-1, -1, 1)$, $(1, 1, 1)$, or $(1, -1, -1)$, the resulting output is the fundamental memory $(1, -1, 1)$. Each of these values of the probe represents a single error, compared to the stored pattern.
2. If the probe vector ξ_{probe} equals $(1, 1, -1)$, $(-1, -1, -1)$, or $(-1, 1, 1)$, the resulting network output is the fundamental memory $(-1, 1, -1)$. Here again, each of these values of the probe represents a single error, compared to the stored pattern.

Spurious States

The weight matrix $\mathbf{W}$ of a discrete Hopfield network is symmetric, as indicated in Eq. (14.44). The eigenvalues of $\mathbf{W}$ are therefore all real. However, for large M , the eigenvalues are ordinarily *degenerate*, which means that there are several eigenvectors with the same eigenvalue. The eigenvectors associated with a degenerate eigenvalue form a subspace. Furthermore, the weight matrix $\mathbf{W}$ has a degenerate eigenvalue with a value of zero, in which case the subspace is called the *null space*. The null space exists by virtue of the fact that the number of fundamental memories, M , is smaller than the number of neurons, N , in the network. The presence of a null subspace is an intrinsic characteristic of the Hopfield network.

An eigenanalysis of the weight matrix $\mathbf{W}$ leads us to take the following viewpoint of the discrete Hopfield network used as a content-addressable memory (Aiyer et al., 1990):

1. The discrete Hopfield network acts as a *vector projector* in the sense that it projects a probe vector onto a subspace $\mathcal{M}$ spanned by the fundamental memory vectors.
2. The underlying dynamics of the network drive the resulting projected vector to one of the corners of a unit hypercube where the energy function is minimized.

The unit hypercube is N -dimensional. The M fundamental memory vectors, spanning the subspace $\mathcal{M}$, constitute a set of fixed points (stable states) represented by certain corners of the unit hypercube. The other corners of the unit hypercube that lie in or near subspace $\mathcal{M}$ are potential locations for *spurious states*, also referred to as *spurious attractors* (Amit, 1989). Spurious states represent stable states of the Hopfield network that are different from the fundamental memories of the network.

In the design of a Hopfield network as a content-addressable memory, we are therefore faced with a tradeoff between two conflicting requirements: (1) the need to preserve the fundamental memory vectors as fixed points in the state space, and (2) the desire to have few spurious states.

Storage Capacity of the Hopfield Network

Unfortunately, the fundamental memories of a Hopfield network are not always stable. Moreover, spurious states representing other stable states that are different from the fundamental memories can arise. These two phenomena tend to decrease the efficiency of the Hopfield network as a content-addressable memory. Here we explore the first of these two phenomena.

Let a probe equal to one of the fundamental memories, ξ_{ν} , be applied to the network. Then, permitting the use of self-feedback for generality and assuming zero bias, we find using Eq. (14.41) that the induced local field of neuron j is

$$\begin{aligned} v_j &= \sum_{i=1}^N w_{ji} \xi_{\nu,i} \\ &= \frac{1}{N} \sum_{\mu=1}^M \xi_{\mu,j} \sum_{i=1}^N \xi_{\mu,i} \xi_{\nu,i} \\ &= \xi_{\nu,j} + \frac{1}{N} \sum_{\substack{\mu=1 \\ \mu \neq \nu}}^M \xi_{\mu,j} \sum_{i=1}^N \xi_{\mu,i} \xi_{\nu,i} \end{aligned} \quad (14.47)$$

The first term on the right-hand side of Eq. (14.47) is simply the j th element of the fundamental memory ξ_{ν} ; now we can see why the scaling factor $1/N$ was introduced in the definition of the synaptic weight w_{ji} in Eq. (14.41). This term may therefore be viewed as the desired "signal" component of v_j . The second term on the right-hand side of Eq. (14.47) is the result of "crosstalk" between the elements of the fundamental memory ξ_{ν} under test and those of some other fundamental memory ξ_{μ} . This second term may therefore be viewed as the "noise" component of v_j . We therefore have a situation similar to the classical "signal-in-noise detection problem" in communication theory (Haykin, 1994b).

We assume that the fundamental memories are random, being generated as a sequence of MN Bernoulli trials. The noise term of Eq. (14.47) then consists of a sum of

$N(M-1)$ independent random variables, taking values ± 1 divided by N . This is a situation where the central limit theorem of probability theory applies. The *central limit theorem* states (Feller, 1968):

Let $\{X_i\}$ be a sequence of mutually independent random variables with a common distribution. Suppose that X_i has mean μ and variance σ^2 , and let $Y = X_1 + X_2 + \dots + X_n$. Then, as n approaches infinity, the probability distribution of the sum random variable Y approaches a Gaussian distribution.

Hence, by applying the central limit theorem to the noise term in Eq. (14.47), we find that the noise is asymptotically Gaussian distributed. Each of the $N(M-1)$ random variables constituting the noise term in this equation has a mean of zero and a variance of $1/N^2$. It follows, therefore, that the statistics of the Gaussian distribution are

- Zero mean
- Variance equal to $(M-1)/N$

The signal component $\xi_{v,j}$ has a value of $+1$ or -1 with equal probability, and therefore a mean of zero and variance of one. The *signal-to-noise ratio* is thus defined by

$$\begin{aligned} \rho &= \frac{\text{variance of signal}}{\text{variance of noise}} \\ &= \frac{1}{(M-1)/N} \\ &\approx \frac{N}{M} \quad \text{for large } M \end{aligned} \quad (14.48)$$

The components of the fundamental memory ξ_v will be *stable* if, and only if, the signal-to-noise ratio ρ is high. Now, the number M of fundamental memories provides a direct measure of the *storage capacity* of the network. Therefore, it follows from Eq. (14.48) that so long as the storage capacity of the network is not overloaded—that is, the number M of fundamental memories is small compared to the number N of neurons in the network—the fundamental memories are stable in a probabilistic sense.

The reciprocal of the signal-to-noise ratio, that is,

$$\alpha = \frac{M}{N} \quad (14.49)$$

is called the *load parameter*. Statistical physics considerations reveal that the quality of memory recall of the Hopfield network deteriorates with increasing load parameter α , and breaks down at the *critical value* $\alpha_c = 0.14$ (Åmit, 1989; Müller and Reinhardt, 1990). This critical value is in agreement with the estimate of Hopfield (1982), where it is reported that as a result of computer simulations $0.15N$ states can be recalled simultaneously before errors become severe.

With $\alpha_c = 0.14$, we find from Eq. (14.48) that the critical value of the signal-to-noise ratio is $\rho_c \approx 7$, or equivalently 8.45 dB. For a signal-to-noise ratio below this critical value, memory recall breaks down.

The critical value

$$M_c = \alpha_c N = 0.14 N \quad (14.50)$$

defines the *storage capacity with errors* on recall. To determine the storage capacity without errors we must use a more stringent criterion defined in terms of probability of error as described next.

Let the j th bit of the probe $\xi_{\text{probe}} = \xi_v$ be a symbol 1, that is, $\xi_{v,j} = 1$. Then the *conditional probability of bit error on recall* is defined by the shaded area in Fig. 14.15. The rest of the area under this curve is the *conditional probability that bit j of the probe is retrieved correctly*. Using the well-known formula for a Gaussian distribution, this latter conditional probability is given by

$$P(v_j > 0 | \xi_{v,j} = +1) = \frac{1}{\sqrt{2\pi}\sigma} \int_0^\infty \exp\left(-\frac{(v_j - \mu)^2}{2\sigma^2}\right) dv_j \quad (14.51)$$

With $\xi_{v,j}$ set to $+1$, and the mean of the noise term in Eq. (14.47) equal to zero, it follows that the mean of the random variable V is $\mu = 1$ and its variance is $\sigma^2 = (M-1)/N$. From the definition of the *error function* commonly used in calculations involving the Gaussian distribution, we have

$$\text{erf}(y) = \frac{2}{\sqrt{\pi}} \int_0^y e^{-z^2} dz \quad (14.52)$$

where y is a variable defining the upper limit of integration. We may now simplify the expression for the conditional probability of correctly retrieving the j th bit of the fundamental memory ξ_v by rewriting Eq. (14.51) in terms of the error function as:

$$P(v_j > 0 | \xi_{v,j} = +1) = \frac{1}{2} \left[1 + \text{erf}\left(\sqrt{\frac{\rho}{2}}\right) \right] \quad (14.53)$$

where ρ is the signal-to-noise ratio defined in Eq. (14.48). Each fundamental memory consists of n bits. Also, the fundamental memories are usually equiprobable. It follows therefore that the *probability of stable patterns* is defined by

$$P_{\text{stab}} = (P(v_j > 0 | \xi_{v,j} = +1))^N \quad (14.54)$$

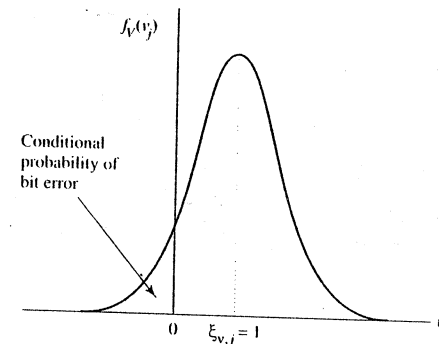


FIGURE 14.15 Conditional probability of bit error, assuming a Gaussian distribution for the induced local field v_j of neuron j ; the subscript V in the probability density function $f_v(v_j)$ denotes a random variable with v_j representing a realization of it.

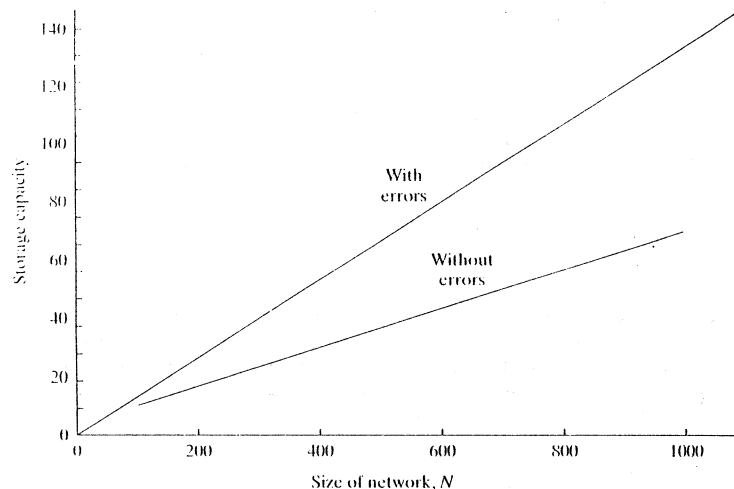


FIGURE 14.16 Plots of storage capacity of the Hopfield network versus network size for two cases: with errors and almost without errors.

We may use this probability to formulate an expression for the capacity of a Hopfield network. Specifically, we define the *storage capacity almost without errors*, $M_{\max}$, as the largest number of fundamental memories that can be stored in the network and yet insist that most of them be recalled correctly. In Problem 14.8 it is shown that this definition of storage capacity yields the formula

$$M_{\max} = \frac{N}{2 \log_e N} \quad (14.55)$$

where $\log_e$ denotes the natural logarithm.

Figure 14.16 shows graphs of the storage capacity with errors defined in Eq. (14.50) and the storage capacity almost without errors defined in Eq. (14.55), both plotted versus the network size N . From this figure we note the following points:

- Storage capacity of the Hopfield network scales essentially *linearly* with the size N of the network.
- A major limitation of the Hopfield network is that its storage capacity must be maintained small for the fundamental memories to be recoverable.⁶

14.8 COMPUTER EXPERIMENT I

In this section we use a computer experiment to illustrate the behavior of the discrete Hopfield network as a content-addressable memory. The network used in the experiment consists of $N = 120$ neurons, and therefore $N^2 - N = 12,280$ synaptic weights. It was

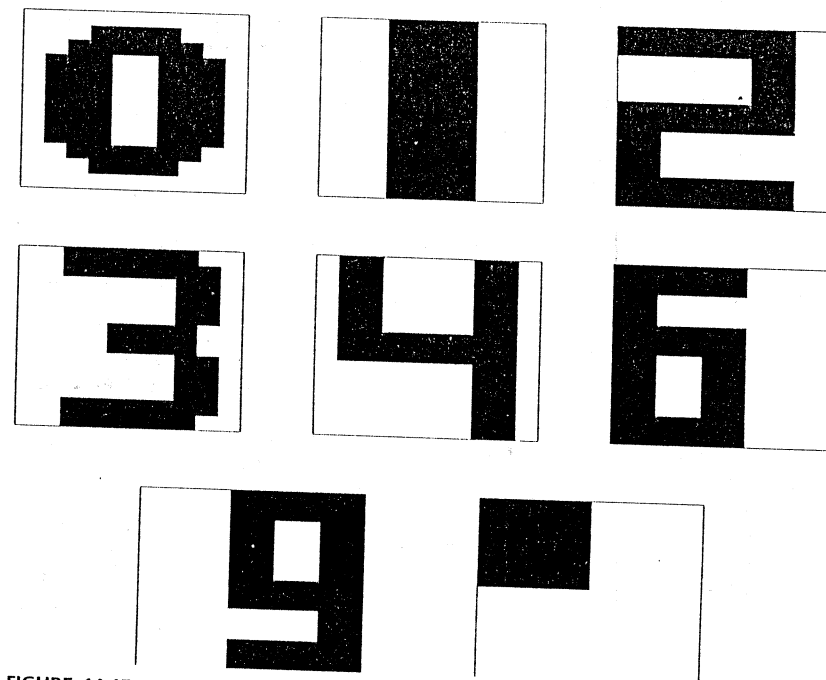


FIGURE 14.17 Set of handcrafted patterns for computer experiment on the Hopfield network.

trained to retrieve the eight digitlike black and white patterns shown in Fig. 14.17, with each pattern containing 120 pixels (picture elements) and designed specially to produce good performance (Lippmann, 1987). The inputs applied to the network assume the value $+1$ for black pixels and -1 for white pixels. The eight patterns of Fig. 14.17 were used as fundamental memories in the storage (learning) phase of the Hopfield network to create the synaptic weight matrix W , which was done using Eq. (14.43). The retrieval phase of the network's operation was performed asynchronously, as described in Table 14.2.

During the first stage of the retrieval part of the experiment, the fundamental memories were presented to the network to test its ability to recover them correctly from the information stored in the synaptic weight matrix. In each case, the desired pattern was produced by the network after one iteration.

Next, to demonstrate the error-correcting capability of the Hopfield network, a pattern of interest was distorted by randomly and independently reversing each pixel of the pattern from $+1$ to -1 and vice versa with a probability of 0.25, and then using the corrupted pattern as a probe for the network. The result of this experiment for digit 3 is presented in Fig. 14.18. The pattern in the mid-top part of this figure represents a corrupted version of digit 3, which is applied to the network at zero time. The patterns produced by the network after 5, 10, 15, 20, 25, 30, and 35 iterations are presented in

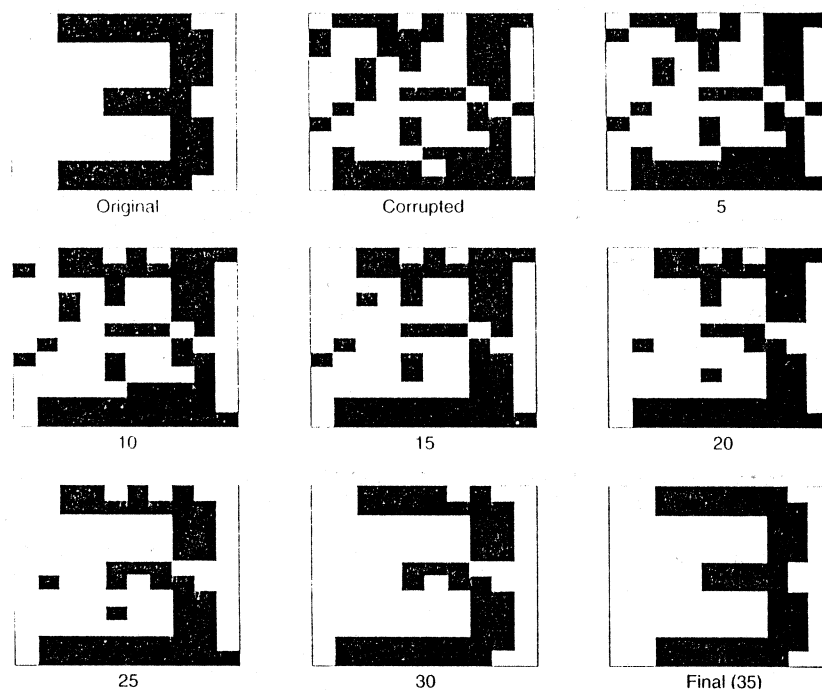


FIGURE 14.18 Correct recollection of corrupted pattern 3.

the rest of the figure. As the number of iterations is increased, we see that the resemblance of the network output to digit 3 is progressively improved. Indeed, after 35 iterations, the network converges onto the exactly correct form of digit 3.

Since, in theory, one quarter of the 120 neurons of the Hopfield network end up changing state for each corrupted pattern, the number of iterations needed for recall, on average, is 30. In our experiment, the number of iterations needed for the recall of the different patterns from their corrupted versions were as follows:

Pattern	Number of patterns needed for recall
0	34
1	32
2	26
3	35
4	25
6	37
"P ₁ "	32
9	26

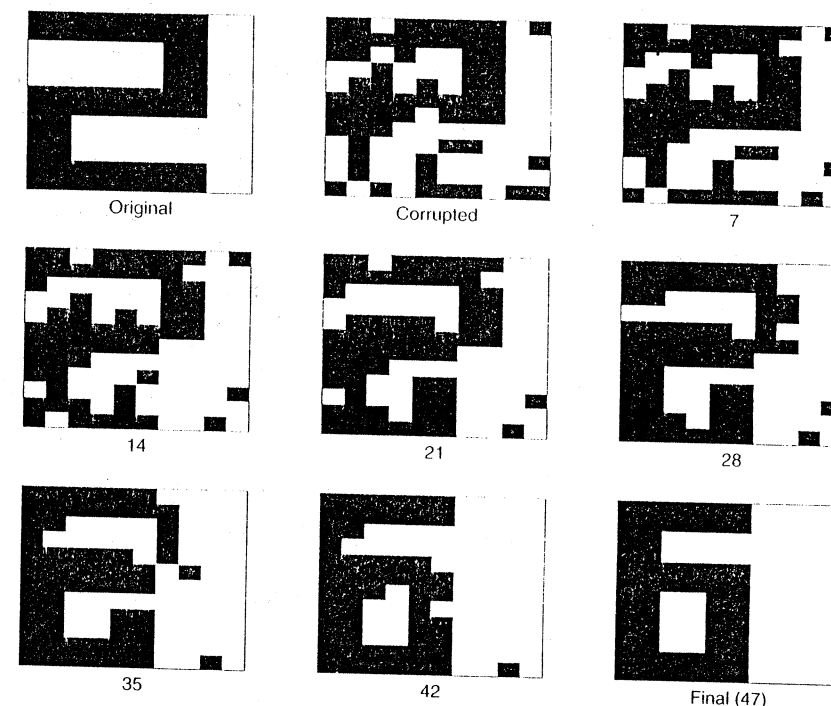


FIGURE 14.19 Incorrect recollection of corrupted pattern 2.

The average number of iterations needed for recall, averaged over the eight patterns, was about 31, which shows that the Hopfield network behaved as expected.

A problem inherent to the Hopfield network arises when the network is presented with a corrupted version of a fundamental memory, and the network then proceeds to converge onto the wrong fundamental memory. This is illustrated in Fig. 14.19, where the network is presented with a corrupted pattern "2," but after 47 iterations it converged to the fundamental memory "6."

As mentioned earlier, there is another problem that arises in the Hopfield network: the presence of spurious states. Figure 14.20 (viewed as a matrix of 14-by-8 network states) presents a listing of 108 spurious attractors found in 43,097 tests of randomly selected digits corrupted with the probability of flipping a bit set at 0.25. The spurious states may be grouped as follows (Amit, 1989):

1. *Reversed fundamental memories.* These spurious states are reversed (i.e., negative) versions of the fundamental memories of the network; see, for example, the state in location 1-by-1 in Fig. 14.20, which represents the negative of digit 6 in Fig. 14.17. To explain this kind of a spurious state, we note that the energy function E is symmetric in the sense that its value remains unchanged if the states of the neurons are reversed (i.e., the state x_i is replaced by $-x_i$ for all i).

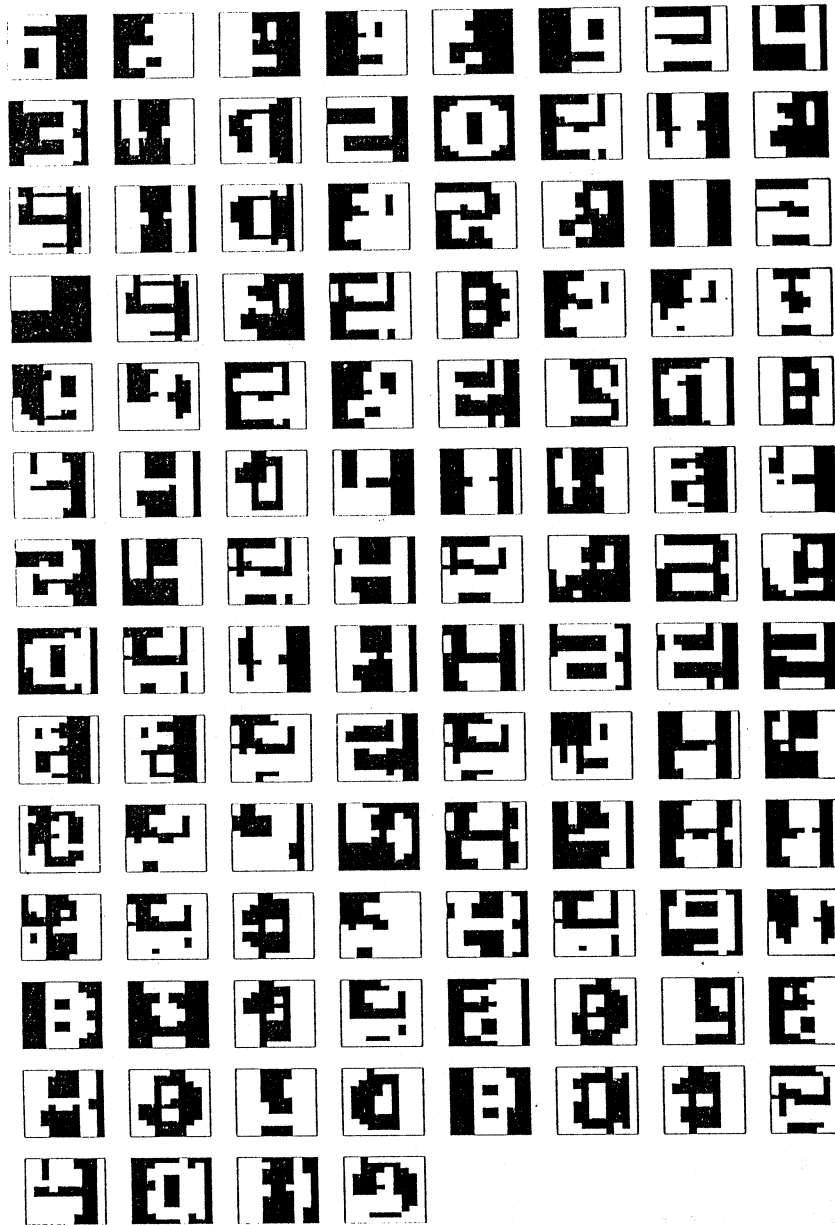


FIGURE 14.20 Compilation of the spurious states produced in the computer experiment on the Hopfield network.

Accordingly, if the fundamental memory ξ_μ corresponds to a particular local minimum of the energy landscape, that same local minimum also corresponds to $-\xi_\mu$. This sign reversal does not pose a problem in the retrieval of information if it is agreed to reverse all the information bits of a retrieved pattern if it is found that a particular bit designated as the "sign" bit is -1 instead of $+1$.

2. *Mixture states.* A mixture spurious state is a linear combination of an *odd* number of stored patterns. For example, consider the state

$$x_i = \text{sgn}(\xi_{1,i} + \xi_{2,i} + \xi_{3,i})$$

which is a three-mixture spurious state. It is a state formed out of three fundamental memories ξ_1 , ξ_2 , and ξ_3 by a majority rule. The stability condition of Eq. (14.45) is satisfied by such a state for a large network. The state in location row 6, column 4 in Fig. 14.20 represents a three-mixture spurious state formed by a combination of the fundamental memories: ξ_1 = negative of digit 1, ξ_2 = digit 4, and ξ_3 = digit 9.

3. *Spin-glass states.* This kind of a spurious state is so named by analogy with spin-glass models of statistical mechanics. Spin-glass states are defined by local minima of the energy landscape that are *not* correlated with any of the fundamental memories of the network; see, for example, the state in location row 7, column 6 in Fig. 14.20.

14.9 COHEN-GROSSBERG THEOREM

In Cohen and Grossberg (1983), a general principle for assessing the stability of a certain class of neural networks is described by the following system of coupled nonlinear differential equations:

$$\frac{d}{dt}u_j = a_j(u_j) \left[b_j(u_j) - \sum_{i=1}^N c_{ji} \phi_i(u_i) \right], \quad j = 1, \dots, N \quad (14.56)$$

According to Cohen and Grossberg, this class of neural networks admits a Lyapunov function defined as

$$E = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N c_{ij} \phi_i(u_i) \phi_j(u_j) - \sum_{j=1}^N \int_0^{u_j} b_j(\lambda) \phi_j'(\lambda) d\lambda \quad (14.57)$$

where

$$\phi_j'(\lambda) = \frac{d}{d\lambda}(\phi_j(\lambda)) \quad (14.58)$$

For the definition of Eq. (14.57) to be valid, however, we require the following conditions to hold:

1. The synaptic weights of the network are "symmetric:"

$$c_{ij} = c_{ji} \quad (14.59)$$

2. The function $a_j(u_j)$ satisfies the condition for "nonnegativity:"

$$a_j(u_j) \geq 0 \quad (14.60)$$

3. The nonlinear input-output function $\varphi_j(u_j)$ satisfies the condition for "monotonicity:"

$$\varphi_j'(u_j) = \frac{d}{du_j} \varphi_j(u_j) \geq 0 \quad (14.61)$$

We may now formally state the *Cohen-Grossberg theorem*:

Provided that the system of nonlinear differential equations (14.56) satisfies the conditions of symmetry, nonnegativity, and monotonicity, the Lyapunov function E of the system defined by Eq. (14.57) satisfies the condition

$$\frac{dE}{dt} \leq 0$$

Once this basic property of the Lyapunov function E is in place, global stability of the system follows from Lyapunov's Theorem 1.

Hopfield Model as a Special Case of the Cohen-Grossberg Theorem

By comparing the general system of Eq. (14.56) with the system of Eq. (14.20) for a continuous Hopfield model, we may make the correspondence between the Hopfield model and the Cohen-Grossberg theorem that are summarized in Table 14.3. The use of this table in Eq. (14.57) yields the following Lyapunov function for the continuous Hopfield model:

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ji} \varphi_i(v_i) \varphi_j(v_j) + \sum_{j=1}^N \int_0^{v_j} \left(\frac{v_j}{R_j} - I_j \right) \varphi_j'(v) dv \quad (14.62)$$

where the nonlinear activation function $\varphi_j(\cdot)$ is defined by Eq. (14.23).

We next make the following observations:

1. $\varphi_i(v_i) = x_i$
2. $\int_0^{v_j} \varphi_j'(v) dv = \int_0^{x_j} dx = x_j$
3. $\int_0^{v_j} v \varphi_j'(v) dv = \int_0^{x_j} x dx = \int_0^{x_j} x \varphi_j^{-1}(x) dx$

Basically, relations 2 and 3 result from the use of $x = \varphi_j(v)$. Thus the use of these observations in the Lyapunov function of Eq. (14.62) yields a result identical to that we

TABLE 14.3 Correspondence between the Cohen-Grossberg Theorem and the Hopfield Model

Cohen-Grossberg Theorem	Hopfield Model
u_j	$C_j v_j$
$a_j(u_j)$	1
$b_j(u_j)$	$-(v_j/R_j) + I_j$
c_{ji}	$-w_{ji}$
$\varphi_j(u_j)$	$\varphi_j(v_j)$

defined earlier; see Eq. (14.28). Note, however, that although $\varphi_j(v)$ must be a nondecreasing function of the input v , it does not need to have an inverse in order for the generalized Lyapunov function of Eq. (14.62) to hold.

The Cohen-Grossberg theorem is a general principle of neurodynamics with a wide range of applications (Grossberg, 1990). In the next section we consider another application of this important theorem.

14.10 BRAIN-STATE-IN-A-BOX MODEL

In this section we continue the neurodynamical analysis of an associative memory by studying the *brain-state-in-a-box (BSB) model*, which was first described by Anderson et al. (1977). The BSB model is basically a *positive feedback system with amplitude limitation*. It consists of a highly interconnected set of neurons that feed back upon themselves. This model operates by using the built-in positive feedback to *amplify* an input pattern until all the neurons in the model are driven into saturation. The BSB model may thus be viewed as a categorization device in that an analog input pattern is given a digital representation defined by a stable state of the model.

Let $\mathbf{W}$ denote a *symmetric weight matrix* whose largest eigenvalues have positive real components. Let $\mathbf{x}(0)$ denote the *initial state vector* of the model, representing an input activation pattern. Assuming that there are N neurons in the model, the state vector of the model has dimension N , and the weight matrix $\mathbf{W}$ is an N -by- N matrix. The BSB algorithm is then completely defined by the following pair of equations:

$$\mathbf{y}(n) = \mathbf{x}(n) + \beta \mathbf{W} \mathbf{x}(n) \quad (14.63)$$

$$\mathbf{x}(n+1) = \varphi(\mathbf{y}(n)) \quad (14.64)$$

where β is a small positive constant called the *feedback factor* and $\mathbf{x}(n)$ is the state vector of the model at discrete time n . Figure 14.21a shows a block diagram of the combination of Eqs. (14.63) and (14.64); the block labeled $\mathbf{W}$ represents a single-layer linear neural network, as depicted in Fig. 14.21b. The activation function φ is a *piecewise-linear function* that operates on $y_j(n)$, the j th component of the vector $\mathbf{y}(n)$, as follows (see Fig. 14.22):

$$x_j(n+1) = \varphi(y_j(n)) = \begin{cases} +1 & \text{if } y_j(n) > +1 \\ y_j(n) & \text{if } -1 \leq y_j(n) \leq +1 \\ -1 & \text{if } y_j(n) < -1 \end{cases} \quad (14.65)$$

Equation (14.65) constrains the state vector of the BSB model to lie within an N -dimensional unit cube centered on the origin.

The algorithm thus proceeds as follows. An activation pattern $\mathbf{x}(0)$ is input into the BSB model as the initial state vector, and Eq. (14.63) is used to compute the vector $\mathbf{y}(0)$. Equation (14.64) is then used to truncate $\mathbf{y}(0)$, obtaining the updated state vector $\mathbf{x}(1)$. Next, $\mathbf{x}(1)$ is cycled through Eqs. (14.63) and (14.64), thereby obtaining $\mathbf{x}(2)$. This procedure is repeated until the BSB model reaches a *stable state* represented by a particular corner of the unit hypercube. Intuitively, positive feedback in the BSB model

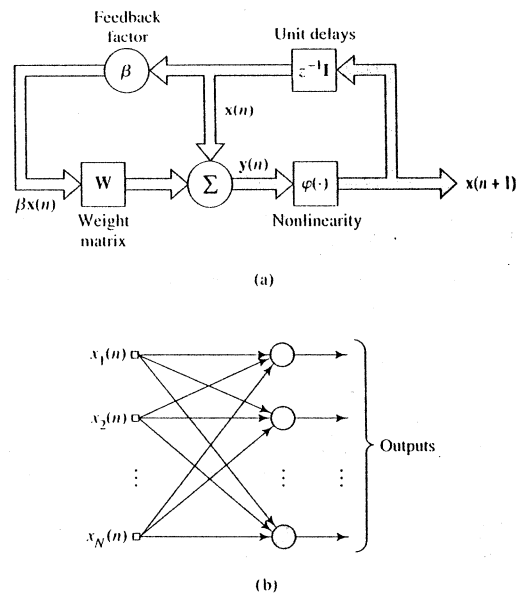


FIGURE 14.21 (a) Block diagram of the brain-state-in-a-box (BSB) model. (b) Signal-flow graph of the linear associator represented by the weight matrix W .

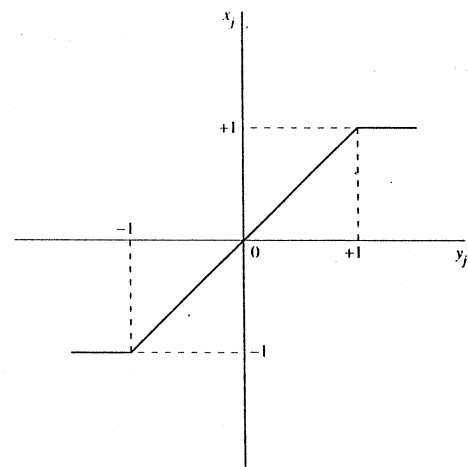


FIGURE 14.22 Piecewise-linear activation function used in the BSB model.

causes the initial state vector $\mathbf{x}(0)$ to increase in Euclidean length (norm) with an increasing number of iterations until it hits a wall of the box (unit hypercube), then slides along the wall and eventually ends up in a stable corner of the box, where it keeps on “pushing” but cannot get out of the box (Kawamoto and Anderson, 1985) hence the name of the model.

Lyapunov Function of the BSB Model

The BSB may be redefined as a special case of the neurodynamical model described in Eq. (14.16) as follows (Grossberg, 1990). To see this, we first rewrite the j th component of the BSB algorithm described by Eqs. (14.63) and (14.64) in the form

$$x_j(n+1) = \phi\left(\sum_{i=1}^N c_{ji}x_i(n)\right), \quad j = 1, 2, \dots, N \quad (14.66)$$

The coefficients c_{ji} are defined by

$$c_{ji} = \delta_{ji} + \beta w_{ji} \quad (14.67)$$

where δ_{ji} is the Kronecker delta equal to 1 if $j = i$ and 0 otherwise, and w_{ji} is the ji -th element of the weight matrix W . Equation (14.66) is written in discrete time form. To proceed further, we need to reformulate it in a continuous time form, as shown by

$$\frac{d}{dt}x_j(t) = -x_j(t) + \phi\left(\sum_{i=1}^N c_{ji}x_i(t)\right), \quad j = 1, 2, \dots, N \quad (14.68)$$

where the bias I_j is zero for all j . However, for us to apply the Cohen–Grossberg theorem, we have to go one step further and transform Eq. (14.68) into the same form as the additive model. We may do so by introducing a new set of variables,

$$v_j(t) = \sum_{i=1}^N c_{ji}x_i(t) \quad (14.69)$$

Then, by virtue of the definition of c_{ji} given in Eq. (14.67), we find that

$$x_j(t) = \sum_{i=1}^N c_{ji}v_i(t) \quad (14.70)$$

Correspondingly, we may recast the model of Eq. (14.68) in the equivalent form

$$\frac{d}{dt}v_j(t) = -v_j(t) + \sum_{i=1}^N c_{ji}\phi(v_i(t)), \quad j = 1, 2, \dots, N \quad (14.71)$$

We are now ready to apply the Cohen–Grossberg theorem to the BSB model. By comparing Eq. (14.71) with (14.56), we may deduce the correspondence listed in Table 14.4 between the BSB model and the Cohen–Grossberg theorem. Therefore, using the results of Table 14.4 in Eq. (14.57), we find that the Lyapunov function of the BSB model is given by

$$E = -\frac{1}{2} \sum_{j=1}^N \sum_{i=1}^N c_{ji}\phi(v_j)\phi(v_i) + \sum_{j=1}^N \int_0^{v_j} v\phi'(v)dv \quad (14.72)$$

TABLE 14.4 Correspondence between the Cohen-Grossberg Theorem and the BSB Model

Cohen-Grossberg Theorem	BSB Model
u_i	v_i
$a_i(u_i)$	1
$b_i(u_i)$	$-v_i$
c_{ji}	$-c_{ji}$
$\phi_j(u_i)$	$\phi_j(v_i)$

where $\phi'(v)$ is the first derivative of the sigmoid function $\phi(v)$ with respect to its argument. Finally, substituting the definitions of Eqs. (14.65), (14.67), and (14.69) in (14.72), we can define the Lyapunov (energy) function of the BSB model in terms of the original state variables as follows (Grossberg, 1990):

$$E = -\frac{\beta}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ji} x_j x_i \quad (14.73)$$

$$= -\frac{\beta}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}$$

The evaluation of the Lyapunov function for the Hopfield network presented in Section 14.7 assumes the existence of the derivative of the inverse of the model's sigmoidal nonlinearity, which is satisfied by the use of a hyperbolic tangent function. In contrast, this condition is not satisfied in the BSB model when the state variable of the j th neuron in the model is either +1 or -1. Despite this difficulty, the Lyapunov function of the BSB model can be evaluated via the Cohen-Grossberg theorem, which clearly illustrates the general applicability of this important theorem.

Dynamics of the BSB Model

In a direct analysis carried out by Golden (1986), it is demonstrated that the BSB model is in fact a gradient descent algorithm that minimizes the energy function E defined by Eq. (14.73). This important property of the BSB model, however, presumes that the weight matrix $\mathbf{W}$ satisfies the following two conditions:

- The weight matrix $\mathbf{W}$ is *symmetric*:

$$\mathbf{W} = \mathbf{W}^T$$

- The weight matrix $\mathbf{W}$ is *positive semidefinite*; that is, in terms of the eigenvalues of $\mathbf{W}$, we have

$$\lambda_{\min} \geq 0$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{W}$.

The energy function E of the BSB model thus decreases with increasing n (number of iterations) whenever the state vector $\mathbf{x}(n+1)$ at time $n+1$ is different from the state vector $\mathbf{x}(n)$ at time n . Moreover, the minimum points of the energy function E define the *equilibrium states* of the BSB model that are characterized by

$$\mathbf{x}(n+1) = \mathbf{x}(n)$$

In other words, like the Hopfield model, the BSB model is an *energy-minimizing network*.

The equilibrium states of the BSB model are defined by certain corners of the unit hypercube and its origin. In the latter case, any fluctuation in the state vector, no matter how small, is amplified by positive feedback in the model, and therefore causes the state of the model to shift away from the origin in the direction of a stable configuration; in other words, the origin is a saddle point. For every corner of the hypercube to serve as a possible equilibrium state of the BSB model, the weight matrix $\mathbf{W}$ has to satisfy a third condition (Greenberg, 1988):

- The weight matrix $\mathbf{W}$ is *diagonal dominant*, which means that

$$w_{jj} \geq \sum_{i \neq j} |w_{ij}| \quad \text{for } j = 1, 2, \dots, N \quad (14.74)$$

where w_{ij} is the ij -th element of $\mathbf{W}$.

For an equilibrium state $\mathbf{x}$ to be *stable*—that is, for a certain corner of the unit hypercube to be a fixed point *attractor*—there has to be a basin of attraction $\mathcal{N}(\mathbf{x})$ in the unit hypercube such that for all initial state vectors $\mathbf{x}(0)$ in $\mathcal{N}(\mathbf{x})$ the BSB model converges onto $\mathbf{x}$. For every corner of the unit hypercube to be a possible point attractor, the weight matrix $\mathbf{W}$ has to satisfy a fourth condition (Greenberg, 1988):

- The weight matrix $\mathbf{W}$ is *strongly diagonal-dominant*, as shown by

$$w_{jj} \geq \sum_{i \neq j} |w_{ij}| + \alpha \quad \text{for } j = 1, 2, \dots, N \quad (14.75)$$

where α is a positive constant.

The important point in this discussion is that in the case of a BSB model for which the weight matrix $\mathbf{W}$ is symmetric and positive semidefinite, as is often the case, only some (but not all) of the corners of the unit hypercube act as point attractors. For all the corners of the unit hypercube to act as potential point attractors, the weight matrix $\mathbf{W}$ has to satisfy Eq. (14.75) as well, which of course subsumes the condition of Eq. (14.74).

Clustering

A natural application for the BSB model is *clustering*. This follows from the fact that the stable corners of the unit hypercube act as point attractors with well-behaved basins of attraction, which therefore divide the state space into a corresponding set of well-defined regions. Consequently, the BSB model may be used as an *unsupervised*

clustering algorithm, with each stable corner of the unit hypercube representing a "cluster" of related data. The self-amplification provided by positive feedback (in conformity with Principle 1 of self-organization described in Chapter 8) is an important ingredient of this clustering property.

Anderson et al. (1990b) describe the use of the BSB model to cluster and therefore identify radar signals from different emitters. In this application the weight matrix $\mathbf{W}$, basic to the operation of the BSB model, is *learned* using the *linear associator* (*associative memory*) with *error correction learning* that is described in Chapter 2. To be specific, suppose that information is represented by a set of K training vectors that are associated with themselves as follows:

$$\mathbf{x}_k \rightarrow \mathbf{x}_k, \quad k = 1, 2, \dots, K \quad (14.76)$$

Let a training vector $\mathbf{x}_k$ be selected at random. Then the weight matrix $\mathbf{W}$ is incremented in accordance with the error correction algorithm (see Problem 3.9)

$$\Delta \mathbf{W} = \eta(\mathbf{x}_k - \mathbf{W}\mathbf{x}_k)\mathbf{x}_k \quad (14.77)$$

where η is the learning-rate parameter. The goal of learning the set of stimuli $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K$ is to have the linear associator behave as

$$\mathbf{W}\mathbf{x}_k = \mathbf{x}_k, \quad k = 1, 2, \dots, K \quad (14.78)$$

The error correction algorithm described by Eq. (14.77) approximates the ideal condition of Eq. (14.78) in a least-mean-square error sense. The net effect of this learning process is to force the linear associator to develop a particular set of eigenvectors (defined by the training vectors) with eigenvalues equal to unity.

To perform radar clustering, the BSB model uses the linear associator with error correction learning to construct the weight matrix $\mathbf{W}$ and performs the following computation (Anderson et al., 1990):

$$\mathbf{x}(n+1) = \varphi(\gamma\mathbf{x}(n) + \beta\mathbf{W}\mathbf{x}(n) + \delta\mathbf{x}(0)) \quad (14.79)$$

which is slightly different from the version of the BSB algorithm described in Eqs. (14.63) and (14.64). The difference is in two respects:

- The decay constant γ in the first term $\gamma\mathbf{x}(n)$ is included to cause the current state to decay slightly; provided that γ is a positive constant less than unity, the errors may eventually decay to zero.
- The third term $\delta\mathbf{x}(0)$ is included to keep the initial state vector $\mathbf{x}(0)$ constantly present; it has the effect of limiting the possible states of the BSB model.

Repeated iteration of the BSB model leads to an activity dominated by the eigenvectors of the weight matrix $\mathbf{W}$ with the largest positive eigenvalues, and therefore the vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K$ learned by the linear associator. The clustering ability of the BSB model develops largely as a result of signal-related eigenvectors being associated with large eigenvalues, becoming enhanced by positive feedback in the model, and thereby dominating the state of the model after a number of iterations. On the other hand, noise-related eigenvectors are usually associated with small eigenvalues, and therefore have a diminishing influence on the state of the BSB model, provided that the received signal-to-noise ratio is sufficiently high.

In a radar surveillance environment, detailed descriptions of emitters operating in the environment are not known *a priori*. Hundreds of thousands of radar pulses are typically received for processing in fractions of seconds. Hence there is no lack of data; the challenge is how to make sense of the data. The BSB model is able to help by learning the microwave structure of the radar environment through its inherent clustering property. Clusters are formed around the point attractors of the BSB model (i.e., stable corners of the unit hypercube), with each point attractor representing a particular emitter. The BSB model may thus identify received pulses as being produced by a particular emitter.

14.11 COMPUTER EXPERIMENT II

Figure 14.23 presents the results of an experiment performed on a BSB model containing two neurons. The two-by-two weight matrix $\mathbf{W}$ is defined by

$$\mathbf{W} = \begin{bmatrix} 0.035 & -0.005 \\ -0.005 & 0.035 \end{bmatrix}$$

which is symmetric, positive-definite, and satisfies Eq. (14.75).

The four different parts of Fig. 14.23 correspond to four different settings of the initial state $\mathbf{x}(0)$, as follows:

- (a) $\mathbf{x}(0) = [0.1, 0.2]^T$
- (b) $\mathbf{x}(0) = [-0.2, 0.3]^T$
- (c) $\mathbf{x}(0) = [-0.8, -0.4]^T$
- (d) $\mathbf{x}(0) = [0.6, 0.1]^T$

The areas shown shaded in this figure are the four basins of attraction characterizing the model. The figure clearly illustrates that when the initial state of the model lies in a particular basin of attraction, the underlying dynamics of the model drive the weight matrix $\mathbf{W}(n)$ with increasing number of iterations n , until the network state $\mathbf{x}(n)$ terminates on the fixed point attractor (i.e., a corner of the two-by-two square) belonging to that basin of attraction. A case of particular interest is the trajectory shown in Fig. 14.23d: The initial condition $\mathbf{x}(0)$ lies in the first quadrant, yet the trajectory terminates on the corner $(+1, -1)$ in the fourth quadrant because that is where the point attractor is for the pertinent basin of attraction.

14.12 STRANGE ATTRACTORS AND CHAOS

Up to this point in our discussion of neurodynamics we have focused attention on the kind of behavior exhibited by nonlinear dynamical systems characterized as fixed point attractors. In this section we consider another class of attractors called strange attractors that characterize certain nonlinear dynamical systems of order greater than 2.

A strange attractor exhibits a chaotic behavior that is highly complex. What makes the study of strange attractors and chaos particularly interesting is the fact that the system in question is *deterministic* in the sense that its operation is governed by *fixed* rules, yet such a system with only a few degrees of freedom can exhibit a behavior so complicated

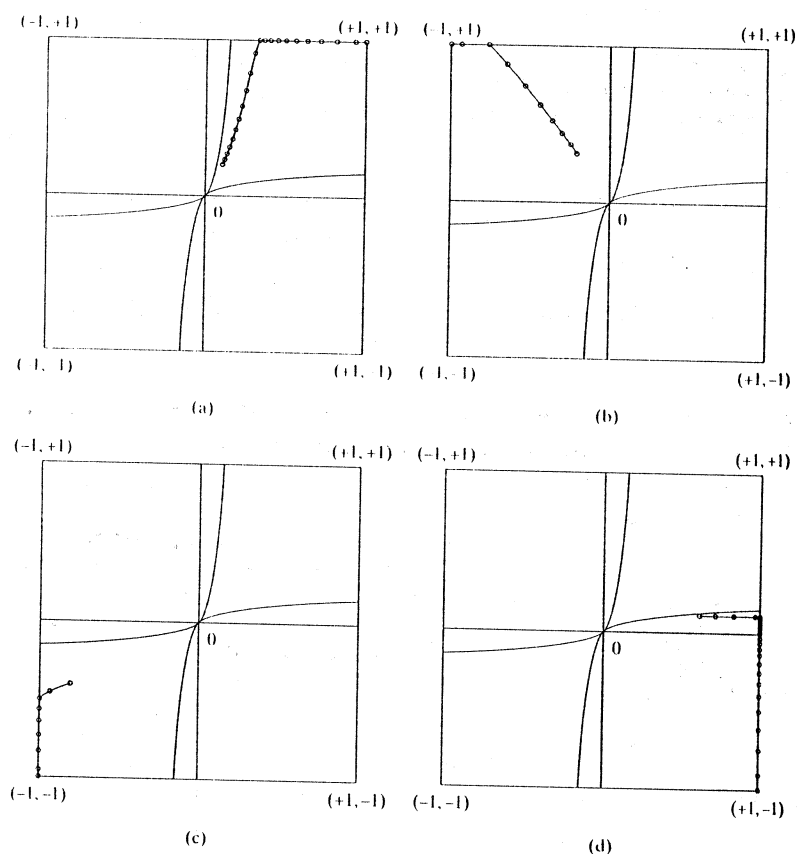


FIGURE 14.23 Trajectories for computer experiment on the BSB model; the results shown in parts (a) through (d) correspond to different initial conditions.

that it looks random. Indeed, the randomness is fundamental in the sense that the second-order statistics of a chaotic time series seem to indicate that it is random. However, unlike a true random phenomenon, the randomness exhibited by a chaotic system does not go away by gathering more information! In principle, the future behavior of a chaotic system is completely determined by the past, but in practice, any uncertainty in the choice of initial conditions, no matter how small, grows exponentially with time. Consequently, even though the dynamic behavior of a chaotic system is predictable in the short term, it is impossible to predict the long-term behavior of the system. A chaotic time series is therefore paradoxical in the sense that its generation is governed by a deterministic dynamical system, and yet it has a randomlike appearance. It is this

attribute of a chaotic phenomenon that was originally emphasized by Lorenz with the discovery of an attractor that bears his name (Lorenz, 1963).

In a nonlinear dynamical system, when the orbits in an attractor with neighboring initial conditions tend to move apart with increasing time, the system is said to possess a *strange attractor* and the system itself is said to be *chaotic*. In other words, a fundamental property that makes an attractor "strange" is the *sensitive dependence on initial conditions*. Sensitivity in this context means that if two identical nonlinear systems are started at slightly different initial conditions, namely, $\mathbf{x}$ and $\mathbf{x} + \epsilon$, where ϵ is a very small quantity, their dynamic states will diverge from each other in state space and their separation will increase exponentially on the average.

Invariant Characteristics of Chaotic Dynamics

Two major features, namely fractal dimensions and Lyapunov exponents, have emerged as the classifiers of a chaotic process. Fractal dimensions characterize the geometric structure of a strange attractor. The term "fractal" was coined by Mandelbrot (1982). Unlike integer dimensions (as in a two-dimensional surface or a three-dimensional object), fractal dimensions are *not* integers. As for Lyapunov exponents, they describe how orbits on the attractor move under the evolution of the dynamics. These two invariant characteristics of chaotic dynamics are discussed in what follows. The term "invariant" signifies the fact that both fractal dimensions and Lyapunov exponents of a chaotic process remain unchanged under smooth nonlinear changes of the coordinate system of the process (Abarbanel, 1996).

Fractal Dimensions

Consider a strange attractor whose dynamics in d -dimensional state space are described by

$$\mathbf{x}(n+1) = \mathbf{F}(\mathbf{x}(n)), \quad n = 0, 1, 2, \dots \quad (14.80)$$

which is the discrete-time version of Eq. (14.2). This is readily seen by setting $t = n\Delta t$, where Δt is the sampling period. Assuming that Δt is sufficiently small, we may correspondingly set

$$\frac{d}{dt}\mathbf{x}(t) = \frac{1}{\Delta t}[\mathbf{x}(n\Delta t + \Delta t) - \mathbf{x}(n\Delta t)]$$

We may thus formulate the discrete-time version of Eq. (14.2) as follows:

$$\frac{1}{\Delta t}[\mathbf{x}(n\Delta t + \Delta t) - \mathbf{x}(n\Delta t)] = \mathbf{F}(\mathbf{x}(n\Delta t)) \quad \text{for small } \Delta t$$

By putting $\Delta t = 1$ for convenience of presentation and rearranging terms, we get

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \mathbf{F}(\mathbf{x}(n))$$

which may be cast into the form shown in Eq. (14.80) simply by redefining the vector-valued function $\mathbf{F}(\cdot)$.

Returning to Eq. (14.80), suppose we construct a small sphere of radius r around some location y on or near an orbit of the attractor. We may then define a *natural distribution* of points for the attractor as follows:

$$\rho(y) = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N \delta(y - x(n)) \quad (14.81)$$

where $\delta(\cdot)$ is a d -dimensional delta function, and N is the number of data points. Note the change of notation concerning the use of N . The natural distribution $\rho(y)$ plays a role for a strange attractor that is analogous to that of the probability density function for a random variable. Accordingly, we may define an invariant f with respect to a function $f(y)$ under the evolution of the dynamics described as the multifold integral

$$f = \int_{-\infty}^{\infty} f(y) \rho(y) dy \quad (14.82)$$

A function $f(y)$ of interest is one that gives us a measure of how the number of points within the small sphere scales as the radius r of the sphere is reduced to zero. Recognizing that the volume occupied by the d -dimensional sphere is proportional to r^d , we may get a sense of attractor dimension by seeing how the density of points on the attractor behaves at small distances in state space.

The Euclidean distance between the center y of the sphere and the point $x(n)$ at time step n is $\|y - x(n)\|$. Hence, the point $x(n)$ lies inside the sphere of radius r provided that

$$\|y - x(n)\| < r$$

or equivalently,

$$r - \|y - x(n)\| > 0$$

Thus the function $f(x)$ for the situation described here may be written in the general form

$$f(x) = \left(\frac{1}{N-1} \sum_{\substack{k=1 \\ k \neq n}}^N \theta(r - \|y - x(k)\|) \right)^{q-1} \quad (14.83)$$

where q is an integer, and $\theta(\cdot)$ is the *Heaviside function* defined by

$$\theta(z) = \begin{cases} 1 & \text{for } z > 0 \\ 0 & \text{for } z < 0 \end{cases}$$

Substituting Eqs. (14.81) and (14.83) in (14.82), we get a new function $C(q, r)$ that depends on q and r , as shown by

$$C(q, r) = \int_{-\infty}^{\infty} \left(\frac{1}{N-1} \sum_{\substack{k=1 \\ k \neq n}}^N \theta(r - \|y - x(k)\|) \right)^{q-1} \left(\frac{1}{N} \sum_{n=1}^N \delta(y - x(n)) \right) dy$$

Hence, using the sifting property of a delta function, namely the relation

$$\int_{-\infty}^{\infty} g(y) \delta(y - x(n)) dy = g(x(n))$$

for some function $g(\cdot)$, and interchanging the order of summation, we may redefine the function $C(q, r)$ as

$$C(q, r) = \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{N-1} \sum_{\substack{k=1 \\ k \neq n}}^N \theta(r - \|x(n) - x(k)\|) \right)^{q-1} \quad (14.84)$$

The function $C(q, r)$ is called the *correlation function*; it is a measure of the probability that two points $x(n)$ and $x(k)$ on the attractor are separated by a distance r . The number of data points N in the defining equation (14.84) is assumed to be large.

The correlation function $C(q, r)$ is an invariant of the attractor in its own right. Nevertheless, the customary practice is to focus on the behavior of $C(q, r)$ for small r . This limiting behavior is described by

$$C(q, r) \approx r^{(q-1)D_q} \quad (14.85)$$

where D_q , called a *fractal dimension* of the attractor, is assumed to exist. Taking the logarithm of both sides of Eq. (14.85), we may formally define D_q as

$$D_q = \lim_{r \rightarrow 0} \frac{\log C(q, r)}{(q-1) \log r} \quad (14.86)$$

However, since we usually have a finite number of data points, the radius r must be just small enough to permit enough points to fall inside the sphere. For a prescribed q , we may then determine the fractal dimension D_q as the slope of the part of the function $C(q, r)$ that is *linear* in $\log r$.

For $q = 2$, the definition of the fractal dimension D_q assumes a simple form that lends it to reliable computation. The resulting dimension, D_2 , is called the *correlation dimension* of the attractor (Grassberger and Procaccia, 1983). The correlation dimension reflects the complexity of the underlying dynamical system and bounds the degrees of freedom required to describe the system.

Lyapunov Exponents

The Lyapunov exponents are statistical quantities that describe the uncertainty about the future state of an attractor. More specifically, they quantify the exponential rate at which nearby trajectories separate from each other while moving on the attractor. Let $x(0)$ be an initial condition and $\{x(n), n = 0, 1, 2, \dots\}$ be the corresponding orbit. Consider an infinitesimal displacement from the initial condition $x(0)$ in the direction of a vector $y(0)$ tangential to the orbit. Then, the evolution of the tangent vector determines the evolution of the infinitesimal displacement of the perturbed orbit $\{y(n), n = 0, 1, 2, \dots\}$ from the unperturbed orbit $\{x(n), n = 0, 1, 2, \dots\}$. In particular, the ratio $\|y(n)\|/\|y(0)\|$ defines the infinitesimal displacement of the orbit from $x(n)$, and the ratio $\|y(n)\|/\|y(0)\|$ is the factor by which the infinitesimal displacement *grows* if $\|y(n)\| > \|y(0)\|$ or *shrinks* if $\|y(n)\| < \|y(0)\|$. For an initial condition $x(0)$ and initial displacement $\alpha_0 = y(0)/\|y(0)\|$, the *Lyapunov exponent* is defined by

$$\lambda(x(0), \alpha) = \lim_{n \rightarrow \infty} \frac{1}{n} \log \left(\frac{\|y(n)\|}{\|y(0)\|} \right) \quad (14.87)$$

A d -dimensional chaotic process has a total of d Lyapunov exponents that can be positive, negative, or zero. Positive Lyapunov exponents account for the instability of an

orbit throughout the state space. Stated in another way, positive Lyapunov exponents are responsible for the *sensitivity of a chaotic process to initial conditions*. Negative Lyapunov exponents, on the other hand, govern the decay of transients in the orbit. A zero Lyapunov exponent signifies the fact that the underlying dynamics responsible for the generation of chaos are describable by a coupled system of nonlinear differential equations, that is, the chaotic process is a *flow*. A volume in d -dimensional state space behaves as $\exp(L(\lambda_1 + \lambda_2 + \dots + \lambda_d))$, where L is the number of time steps into the future. It follows therefore that for a *dissipative* process, the sum of all Lyapunov exponents must be negative. This is a necessary condition for a volume in state space to shrink as time progresses, which is a requirement for physical realizability.

Lyapunov Dimension

Given the Lyapunov spectrum $\lambda_1, \lambda_2, \dots, \lambda_d$, Kaplan and Yorke (1979) suggested a *Lyapunov dimension* for a strange attractor as follows:

$$D_L = K + \frac{\sum_{i=1}^K \lambda_i}{|\lambda_{K+1}|} \quad (14.88)$$

where K is an integer that satisfies the two conditions:

$$\sum_{i=1}^K \lambda_i > 0 \quad \text{and} \quad \sum_{i=1}^{K+1} \lambda_i < 0$$

Ordinarily, the Lyapunov dimension D_L is about the same size as the correlation dimension D_2 . This is an important property of a chaotic process. That is, although the Lyapunov and correlation dimensions are defined in entirely different ways, their values for a strange attractor are usually quite close to each other.

Definition of a Chaotic Process

Throughout this section we have spoken of a chaotic process without a formal definition of it. In light of what we now know about Lyapunov exponents, we can offer the following definition:

A chaotic process is generated by a nonlinear deterministic system, with at least one positive Lyapunov exponent.

The positivity of at least one Lyapunov exponent is a necessary condition for sensitivity to initial conditions, which is the hallmark of a strange attractor.

The largest Lyapunov exponent also defines the *horizon of predictability* of a chaotic process. Specifically, the short-term predictability of a chaotic process is approximately equal to the reciprocal of the largest Lyapunov exponent (Abarbanel, 1996).

14.13 DYNAMIC RECONSTRUCTION

Dynamic reconstruction may be defined as the identification of a mapping that provides a model for an unknown dynamical system of dimensionality m . Our interest here is in the dynamic modeling of a time series produced by a physical system that is known to be chaotic. In other words, given a time series $\{y(n)\}_{n=1}^N$, we wish to build a model that captures the underlying dynamics responsible for generation of the observable $y(n)$. As we pointed out earlier in the previous section, N denotes the sample size.

The primary motivation for dynamic reconstruction is to make physical sense from such a time series, thereby bypassing the need for a detailed mathematical knowledge of the underlying dynamics. The system of interest is typically much too complex to characterize in mathematical terms. The only information available to us is contained in a time series obtained from measurements on one of the observables of the system.

A fundamental result in dynamic reconstruction theory⁸ is a geometric theorem, called the delay-embedding theorem due to Takens (1981). Takens considered a noise-free situation, focusing on *delay coordinate maps* or *predictive models* that are constructed from a time series representing an observable from a dynamical system. In particular, Takens showed that if the dynamical system and the observable are generic, then the delay coordinate map from a d -dimensional smooth compact manifold to $\mathbb{R}^{2d+1}$ is a diffeomorphism on that manifold, where d is the dimension of the state space of the dynamical system. (Diffeomorphism is discussed on p. 744.)

For an interpretation of Takens' theorem in signal processing terms, first consider an unknown dynamical system whose evolution in discrete time is described by the nonlinear difference equation

$$\mathbf{x}(n+1) = \mathbf{F}(\mathbf{x}(n)) \quad (14.89)$$

where $\mathbf{x}(n)$ is the d -dimensional state vector of the system at time n , and $\mathbf{F}(\cdot)$ is a vector-valued function. It is assumed here that the sampling period is normalized to unity. Let the time series $\{y(n)\}$ observable at the output of the system be defined in terms of the state vector $\mathbf{x}(n)$ as follows:

$$y(n) = g(\mathbf{x}(n)) + v(n) \quad (14.90)$$

where $g(\cdot)$ is a scalar-valued function, and $v(n)$ denotes additive noise. The noise $v(n)$ accounts for the combined effects of imperfections and imprecisions in the observable $y(n)$. Equation (14.89) and (14.90) describe the state-space behavior of the dynamical system. According to Takens' theorem, the geometric structure of the multivariable dynamics of the system can be unfolded from the observable $y(n)$ with $v(n) = 0$ in a D -dimensional space constructed from the new vector

$$\mathbf{y}_R(n) = [y(n), y(n-\tau), \dots, y(n-(D-1)\tau)]^T, \quad (14.91)$$

where τ is a positive integer called the *normalized embedding delay*. That is, given the observable $y(n)$ for varying discrete time n , which pertains to a single observable (component) of an unknown dynamical system, dynamic reconstruction is possible using the D -dimensional vector $\mathbf{y}_R(n)$ provided that $D \geq 2d + 1$, where d is the dimension of the state space of the system. Hereafter we refer to this statement as the *delay-embedding theorem*. The condition $D \geq 2d + 1$ is a *sufficient* but not necessary condition for dynamic reconstruction. The procedure for finding a suitable D is called *embedding*, and the minimum integer D that achieves dynamic reconstruction is called the *embedding dimension*; it is denoted by D_E .

The delay-embedding theorem has a powerful implication: Evolution of the points $\mathbf{y}_R(n) \rightarrow \mathbf{y}_R(n+1)$ in the reconstruction space follows that of the unknown dynamics $\mathbf{x}(n) \rightarrow \mathbf{x}(n+1)$ in the original state space. That is, many important properties of the unobservable state vector $\mathbf{x}(n)$ are reproduced without ambiguity in the reconstruction space defined by $\mathbf{y}_R(n)$. However, for this important result to be

attainable, we need *reliable estimates* of the embedding dimension D_E and the normalized embedding delay τ , as summarized here:

- The sufficient condition $D \geq 2d + 1$ makes it possible to undo the intersections of an orbit of the attractor with itself, which arise from projection of that orbit to lower dimensions. The embedding dimension D_E can be less than $2d + 1$. The recommended procedure is to estimate D_E directly from the observable data. A reliable method for estimating D_E is the *method of false nearest neighbors* described in Abarbanel (1996). In this method we systematically survey the data points and their neighbors in dimension $d = 1$, then $d = 2$, and so on. We thereby establish the condition when apparent neighbors stop being “unprojected” by the addition of more elements to the reconstruction vector $\mathbf{y}_R(n)$, and thus obtain an estimate for the embedding dimension D_E .
- Unfortunately, the delay-embedding theorem has nothing to say on the choice of the normalized embedding delay τ . In fact, it permits the use of any τ so long as the available time series is infinitely long. In practice, however, we always have to work with observable data of finite length N . The proper prescription for choosing τ is to recognize that the normalized embedding delay τ should be large enough for $y(n)$ and $y(n - \tau)$ to be essentially independent of each other so as to serve as coordinates of the reconstruction space, but not so independent as to have no correlation with each other. This requirement is best satisfied by using the particular τ for which the *mutual information* between $y(n)$ and $y(n - \tau)$ attains its first minimum (Fraser, 1989). Mutual information is discussed in Chapter 10.

Recursive Prediction

From the discussion presented, the dynamic reconstruction problem may be interpreted as one representing the signal dynamics properly (the embedding step), as well as the construction of a predictive mapping (the identification step). Thus in practical terms we have the following network topology for dynamic modeling:

- A *short-term memory* (e.g., delay-line memory) structure to perform the embedding, whereby the reconstruction vector $\mathbf{y}_R(n)$ is defined in terms of the observable $y(n)$ and its delayed versions: see Eq. (14.91).
- A multiple input, single output (MISO) adaptive nonlinear system trained as a *one-step predictor* (e.g., neural network) to identify the unknown mapping $f: \mathbb{R}^D \rightarrow \mathbb{R}^1$, which is defined by

$$\hat{y}(n + 1) = f(\mathbf{y}_R(n)) \quad (14.92)$$

The predictive mapping described in Eq. (14.92) is the center piece of dynamic modeling: Once it is determined, the evolution $\mathbf{y}_R(n) \rightarrow \mathbf{y}_R(n + 1)$ becomes known, which in turn determines the unknown evolution $\mathbf{x}(n) \rightarrow \mathbf{x}(n + 1)$.

Presently, we do not have a rigorous theory to help us decide if the nonlinear predictor has successfully identified the unknown mapping f . In linear prediction, minimizing the mean-square value of the prediction error leads to an accurate model. However, a chaotic time series is different. Two trajectories in the same attractor are vastly different on a sample-by-sample basis, so minimizing the mean-square value of the prediction error is a necessary but not a sufficient condition of a successful mapping.

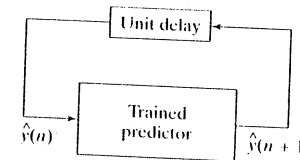


FIGURE 14.24 One-step predictor used in iterated prediction for dynamic reconstruction of a chaotic process.

The dynamic invariants, namely correlation dimension and Lyapunov exponents, measure global properties of the attractor, so they should gauge the success of dynamic modeling. Hence, a *pragmatic approach for testing the dynamic model* is to seed it with a point on the strange attractor, and to feed the output back to its input as an autonomous system as illustrated in Fig. 14.24. Such an operation is called *iterated prediction* or *recursive prediction*. Once the initialization is completed, the output of the autonomous system is a realization of the dynamic reconstruction process. This of course presumes that the predictor has been designed properly in the first place.

We say that dynamic reconstruction performed by means of the autonomous system described in Fig. 14.24 is successful if the following two conditions are satisfied (Haykin and Principe, 1998):

1. *Short-term behavior.* Once the initialization is completed, the reconstructed time series $\{\hat{y}(n)\}$ in Fig. 14.24 closely follows the original time series $\{y(n)\}$ for a period of time, on average equal to the horizon of predictability determined from the Lyapunov spectrum of the process.
2. *Long-term behavior.* The dynamic invariants computed from the reconstructed time series $\{\hat{y}(n)\}$ closely match the corresponding ones computed from the original time series $\{y(n)\}$.

To gauge the long-term behavior of the reconstructed dynamics, we need to estimate (1) the correlation dimension as a measure of attractor complexity, and (2) the Lyapunov spectrum as a framework for assessing sensitivity to initial conditions and for estimating the Lyapunov dimension; see Eq. (14.88). The Lyapunov dimension should have a value close to that of the correlation dimension.

Two Possible Formulations for Recursive Prediction

The reconstruction vector $\mathbf{y}_R(n)$ defined in Eq. (14.91) is of dimension D_E , assuming that the dimension D is set equal to the embedding dimension D_E . The size of the delay line memory required to perform the embedding is τD_E . But the delay line memory is required to provide only D_E outputs (the dimension of the reconstruction space); that is, we use τ equally spaced taps, representing sparse connections.

Alternatively, we may define the reconstruction vector $\mathbf{y}_R(n)$ as a full m -dimensional vector as follows:

$$\mathbf{y}_R(n) = [y(n), y(n - 1), \dots, y(n - m + 1)]^T \quad (14.93)$$

where m is an integer defined by

$$m \geq D_E \tau \quad (14.94)$$

This second formulation of the reconstruction vector $y_R(n)$ supplies more information to the predictive model than that provided by Eq. (14.91) and may therefore yield a more accurate dynamic reconstruction. However, both formulations share a common feature: their compositions are uniquely defined by knowledge of the embedding dimension D_E . In any event, it is wise to use the minimum permissible value of D , namely D_E , to minimize the effect of additive noise $v(n)$ on the quality of dynamic reconstruction.

Dynamic Reconstruction Is an Ill-Posed Filtering Problem

The dynamic reconstruction problem is, in reality, an *ill-posed inverse problem* for one or more of the following reasons. (The conditions for an inverse problem to be well posed are discussed in Chapter 5.) First, for some unknown reason the existence condition may be violated. Second, there may not be sufficient information in the observable time series to reconstruct the nonlinear dynamics uniquely; hence, the uniqueness criterion is violated. Third, the unavoidable presence of additive noise or some form of imprecision in the observable time series adds uncertainty to the dynamic reconstruction. In particular, if the noise level is too high, it is possible for the continuity criterion to be violated. How then do we make the dynamic reconstruction problem well posed? The answer lies in the inclusion of some form of *prior knowledge* about the input-output mapping as an essential requirement. In other words, some form of constraints (e.g., smoothness of input-output mapping) would have to be imposed on the predictive model designed for solving the dynamic reconstruction problem. One effective way in which this requirement can be satisfied is to invoke Tikhonov's *regularization theory*, which is also discussed in Chapter 5.

Another issue that needs to be considered is the ability of the predictive model to solve the inverse problem with sufficient accuracy. In this context, the use of a neural network to build the predictive model is appropriate. In particular, the universal approximation property of a multilayer perceptron or that of a radial-basis function network means that we can take care of the issue of reconstruction accuracy by using one or the other of these neural networks with an appropriate size. In addition, however, we need the solution to be regularized for the reasons explained. In theory, both multilayer perceptrons and radial-basis function networks lend themselves to the use of regularization; in practice, it is in radial-basis function networks that we find regularization theory included in a mathematically tractable manner as an integral part of their design. Accordingly, in the computer experiment described in the next section, we focus on the regularized radial-basis function (RBF) network (described in Chapter 5) as the basis for solving the dynamic reconstruction problem.

atmosphere, which stands as a workhorse set of equations for testing ideas in nonlinear dynamics. The equations for the Lorenz attractor are:

$$\begin{aligned}\frac{dx(t)}{dt} &= -\sigma x(t) + \sigma y(t) \\ \frac{dy(t)}{dt} &= -x(t)z(t) + rx(t) - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t)\end{aligned}\quad (14.95)$$

where σ , r , and b are dimensionless parameters. Typical values for these parameters are $\sigma = 10$, $b = 8/3$, and $r = 28$.

Figure 14.25 shows the results of iterated prediction performed on two RBF networks with 400 centers using a "noisy" time series based on the component $x(t)$ of the Lorenz attractor. The signal-to-noise ratio was +25 dB. In Fig. 14.25a, the design of the network is regularized. In Fig. 14.25b, the design of the network is unregularized. These two parts of Fig. 14.25 clearly demonstrate the practical importance of regularization. Without regularization, the solution to the dynamic reconstruction problem presented in Fig. 14.25b is unacceptable as it fails to approximate the true trajectory of the Lorenz attractor; the unregularized system is just a predictor. On the other hand, the solution to the dynamic reconstruction problem presented in Fig. 14.25a using a regularized form of the RBF network has *learned* the dynamics, in the sense that the output of the network under iterated prediction closely approximates the actual trajectory of the Lorenz attractor in the short term. This is borne out by the results presented in Table 14.5, where we have a summary of Lorenz data for three cases:

- (a) Noise-free Lorenz system.
- (b) Noisy Lorenz system with signal-to-noise ratio SNR = 25 dB.
- (c) Reconstructed data, using the noisy Lorenz time series described under Case b.

The invariants of the reconstructed data using noisy data are close to the corresponding ones pertaining to the noise-free Lorenz data. The deviations in absolute values are due to the residual effect of noise embedded in the reconstructed attractor and to inaccuracies in the estimation procedure. Figure 14.25 clearly shows that there is more to dynamic modeling than just prediction. This figure, and many others not included here, demonstrate the "robustness" of the regularized RBF solution with respect to the point on the attractor that is used to initialize the iterated prediction process.

The following two observations from Fig. 14.25a, pertaining to the use of regularization, are particularly noteworthy:

1. The short-term predictability of the reconstructed time series in Fig. 14.25a is about 60 samples. The theoretical horizon of predictability computed from the Lyapunov spectrum of the noiseless Lorenz attractor is approximately 100 samples. The experimental deviation from the horizon of predictability of the noise-free Lorenz attractor is merely a manifestation of the presence of noise in the actual data used to perform the dynamic reconstruction. The theoretical horizon

14.14 COMPUTER EXPERIMENT III

To illustrate the idea of dynamic reconstruction, we consider the system of three coupled ordinary differential equations, abstracted by Lorenz (1963) from the Galerkin approximation to the partial differential equations of thermal convection in the lower

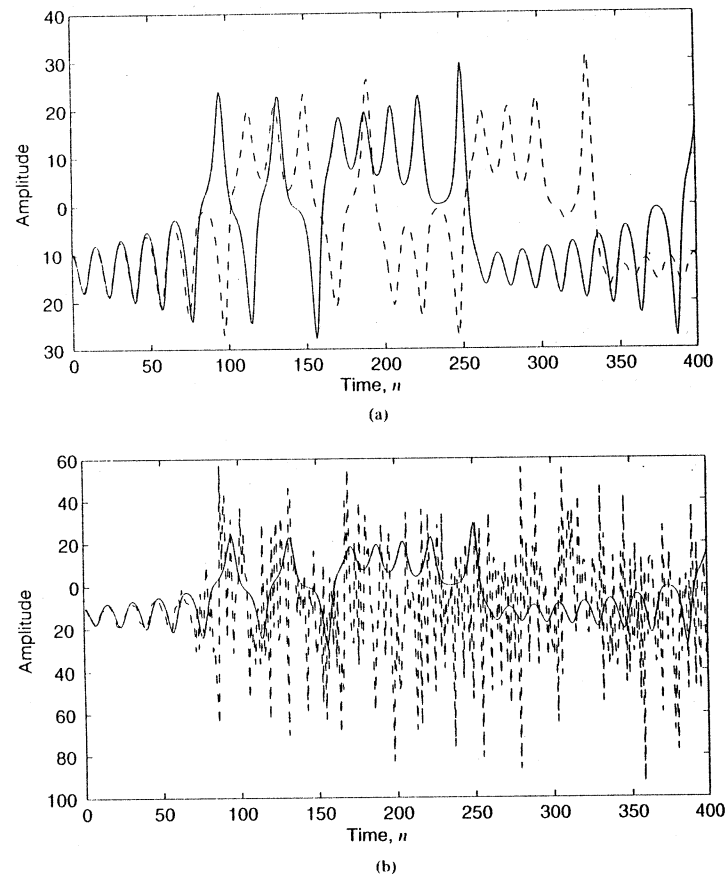


FIGURE 14.25 (a) Regularized iterated prediction ($N = 400, m = 20$) on Lorenz data at SNR = +25 dB. (b) Unregularized iterated prediction ($N = 400, m = 20$) on Lorenz data at SNR = +25 dB. In both parts (a) and (b), the solid curve is the actual chaotic signal and the dashed curve is the reconstructed signal.

of predictability computed from the reconstructed data was 61 (Table 14.5), which is quite close to the experimentally observed value of short-term predictability.

- Once the period of short-term predictability is over, the reconstructed time series in Fig. 14.25a begins to deviate from the noiseless realization of the actual Lorenz attractor. This is basically a manifestation of chaotic dynamics, namely sensitivity to initial conditions. As mentioned previously, sensitivity to initial conditions is a hallmark of chaos.

TABLE 14.5 Summary of Parameters for Experiment on Dynamic Reconstruction Using the Lorenz System

(a) <i>Noise-free Lorenz System</i>	
Number of samples used: 35,000	
1.	Normalized embedding delay, $\tau = 4$
2.	Embedding dimension, $D_E = 3$
3.	Lyapunov exponents:
	$\lambda_1 = 1.5697$
	$\lambda_2 = -0.0314$
	$\lambda_3 = -22.3054$
4.	Horizon of predictability ≈ 100 samples
(b) <i>Noisy Lorenz system: 25 dB SNR</i>	
Number of samples used: 35,000	
1.	Normalized embedding delay, $\tau = 4$
2.	Embedding dimension, $D_E = 5$
3.	Lyapunov exponents:
	$\lambda_1 = 13.2689$
	$\lambda_2 = 5.8562$
	$\lambda_3 = -3.1447$
	$\lambda_4 = -18.0082$
	$\lambda_5 = -47.0572$
4.	Horizon of predictability ≈ 12 samples
(c) <i>Reconstructed system using the noisy Lorenz data under (b)</i>	
Number of samples generated (recursively): 35,000	
1.	Normalized embedding delay, $\tau = 4$
2.	Embedding dimension, $D_E = 3$
3.	Lyapunov exponents:
	$\lambda_1 = 2.5655$
	$\lambda_2 = -0.6275$
	$\lambda_3 = -15.0342$
4.	Horizon of predictability ≈ 61 samples

Notes: All of the Lyapunov exponents are expressed in nats per second; a nat is a natural unit for measuring information as discussed in Chapter 10. Also, in case b, the effect of noise is to increase the size of the Lyapunov spectrum, and the number and magnitude of positive Lyapunov exponents.

Choice of m and λ

The size of the input layer, m , is determined in accordance with Eq. (14.94). As explained previously, the recommended method is to use the smallest permissible value of m in accordance with the equality sign so as to minimize the effect of noise on dynamic reconstruction.

The estimated value of the normalized embedding delay τ is essentially independent of the presence of noise for moderate to high signal-to-noise ratios. In contrast, the presence of noise has a profound impact on the estimated value of the embedding

dimension D_E , which is intuitively satisfying. For example, for the noise-free Lorenz attractor, the correlation dimension is 2.01. We may therefore choose the embedding dimension $D_E = 3$, which is confirmed by the method of false nearest neighbors. The normalized embedding delay is $\tau = 4$. Thus, using Eq. (14.94) with the equality sign yields $m = 12$ for dynamic reconstruction. However, for a noisy Lorenz attractor with $\text{SNR} = +25$ dB, the use of the method of false nearest neighbors yields $D_E = 5$, and the use of the method of mutual information yields $\tau = 4$. By substituting these estimated values in Eq. (14.94) with the equality sign, we get $m = 20$ for the noisy dynamic reconstruction reported in Fig. 14.25. Table 14.5 includes the normalized embedding delay τ and embedding dimension D_E for the three cases summarized therein.

As for the regularization parameter λ used in Fig. 14.25a, it was determined from the training data using the generalized cross-validation (GCV); this method is discussed in Chapter 5. The value of λ that was used in Fig. 14.25a, calculated on the basis of GCV, varied between a minimum value of 10^{-14} to a maximum value of 10^{-2} in accordance with the data.

15 SUMMARY AND DISCUSSION

Much of the material in this chapter is devoted to the Hopfield model and the brain-state-in-a-box (BSB) model as examples of an associative memory rooted in neurodynamics. These two models share some common features:

- They both employ positive feedback.
- They both have an energy (Lyapunov) function, and the underlying dynamics tend to minimize it in an iterative fashion.
- They both learn in a self-organized manner using Hebb's postulate of learning.
- They are both capable of performing computation using attractor dynamics.

Naturally, they differ in their areas of application.

The BSB model has an inherent clustering capability that may be put to good use for data representation and concept formation. The most interesting application of the BSB model is perhaps as the basic computational unit in a *network of networks* proposed as a plausible model of describing different levels of system organization within the brain (Anderson and Sutton, 1995). In this model, the computational units form local networks that are distributed in a two-dimensional array, hence the term "network of networks." Instead of communicating only average activity from column to column, these local networks are designed to communicate with other local networks by means of *activity patterns (vectors)*. In place of synaptic weights between neurons as in a conventional neural network, we now have a set of interaction matrices that describe the coupling between attractors in two local networks. The local networks form clusters and levels, based on their interconnections, with the result that the *anatomical* connectivity is sparse. That is, the local networks are more densely connected within clusters than across clusters. However, the *functional* connectivity among clusters has a rich dynamic, due in part to temporally correlated activity among local networks.

In contrast, the Hopfield model may be used to solve the following computational problems:

1. *Content-addressable memory*, which involves the recall of a stored pattern by presenting a partial or distorted version of it to the memory. For this application the usual procedure is to use the "discrete" Hopfield model that is based on the McCulloch–Pitts neuron (i.e., one using a hard limiting activation function). Viewed in a computational context, the construction of a content-addressable memory is a trivial matter. Nevertheless, the Hopfield model of a content-addressable memory is important because it elucidates the *link between dynamics and computation* in a novel way. In particular, the Hopfield model exhibits the following properties that have neurobiological relevance:
 - The dynamics of the model are dominated by a large number of point attractors in a high-dimensional state space.
 - A point attractor (i.e., fundamental memory) of interest may be located by merely initializing the model with an inexact description of that attractor's location and then allowing the dynamics to evolve the state of the model to the nearest point attractor.
 - Learning (i.e., computation of the free parameters of the model) takes place in accordance with Hebb's postulate of learning. Moreover, this learning mechanism permits the insertion of new point attractors into the model if so desired.
2. *Combinatorial optimization problems*, which rank among the most difficult known to mathematicians. This class of optimization problems includes the *traveling salesman problem* (TSP), considered to be a classic. Given the positions of a specified number of cities, assumed to lie in a plane, the problem is to find the shortest tour that starts and finishes at the same city. The TSP is simple to state but hard to solve exactly, in that there is no known method of finding the optimum tour, short of computing the length of every possible tour and then selecting the shortest one. It is said to be *NP-complete* (Hopcroft and Ullman, 1979). In a pioneering paper, Hopfield and Tank (1985) demonstrated how an analog network, based on the system of coupled first-order differential equations (14.20), can be used to represent a solution of the TSP. Specifically, the synaptic weights of the network are determined by distances between the cities visited on the tour and the optimum solution to the problem is a fixed point of the neurodynamical equations (14.20). Herein lie the difficulties encountered with "mapping" combinatorial optimization problems onto the continuous (analog) Hopfield network. The network acts to minimize a single energy (Lyapunov) function, and yet the typical combinatorial optimization problem requires the minimization of an objective function *subject to* some hard constraints (Gee et al., 1993). If any of these constraints are violated, the solution is considered to be invalid. The early mapping procedures were based on a Lyapunov function constructed in an *ad hoc* manner, usually employing one term for each constraint, as shown by

$$E = E^{\text{opt}} + c_1 E_1^{\text{cns}} + c_2 E_2^{\text{cns}} + \dots \quad (14.96)$$

The first term, E^{opt} , is the objective function to be minimized (e.g., the length of a TSP tour); it is determined by the problem at hand. The remaining terms $E_1^{\text{cns}}, E_2^{\text{cns}}, \dots$, represent penalty functions whose minimization satisfies the *constraints*. The scalars $c_1, c_2, \dots$, are constant weights assigned to the respective penalty functions $E_1^{\text{cns}}, E_2^{\text{cns}}, \dots$, usually by trial and error. Unfortunately, the

many terms in the Lyapunov function of Eq. (14.96) tend to frustrate one another, and the success of the Hopfield network is highly sensitive to the relative values of $c_1, c_2, \dots$ (Gee et al., 1993). It is therefore not surprising that the network often produces a large number of invalid solutions (Wilson and Pawley, 1988; Ansari and Hou, 1997). In Gee (1993) a number of basic questions concerning the use of the continuous Hopfield network as a tool for solving combinatorial optimization problems is addressed; the main findings reported therein may be summarized as follows:

- Given a combinatorial optimization problem expressed in terms of quadratic 0-1 programming, as in the traveling salesman problem, there is a straightforward method for programming the network for its solution, and the solution found will not violate any of the problem's constraints.
- Building on results from complexity theory and mathematical programming, it is shown that, except when the problem's constraints have special properties producing an integral polytope, it is not possible to force the network to converge to a valid, interpretable solution. In geometric terms, a polytope, that is a bounded polyhedron, is said to be an *integral polytope* if all the vertices of the polytope are 0-1 points. Even when dealing with integral polytopes, if the objective function E^{opt} is quadratic, the problem is NP-complete and there is no guarantee that the network will find the optimum solution; this class of problems includes the traveling salesman problem. However, a valid solution will be found and, given the nature of the descent process to this solution, there is a good chance that the solution will be reliable.

The Hopfield model, considered in this chapter, uses symmetric couplings between its neurons. The dynamics of such a structure are similar to gradient descent dynamics, thereby assuring convergence to a fixed point. However, the dynamics of a brain differ from that of the Hopfield model in two important respects:

- The interneuron connections in the brain are asymmetric.
- Oscillatory and complex nonperiodic behavior are observed in the brain.

Indeed, it is because of these special features of the brain that there has been a long-standing interest in the study of asymmetric networks, predating the Hopfield model.

If we abandon the restriction of symmetry, the next simplest model is the *excitatory-inhibitory network* whose neurons fall into two populations: one with only excitatory outputs, and the other with only inhibitory outputs. The synaptic connections between the two populations are antisymmetric, while the synaptic connections within each population are symmetric. In Seung et al. (1998), the dynamics of such a network are considered. The analysis presented therein exploits the similarity of the underlying dynamics of the inhibitory-excitatory network to *gradient descent–gradient ascent dynamics*, where the equations of motion are gradient descent in some state variables and gradient ascent in the others. Consequently, unlike the gradient descent dynamics characterizing the Hopfield model, the dynamics of the model considered by Seung et al. can converge to a fixed point or a limit cycle, depending on the choice of network parameters. Thus the antisymmetric model studied in Seung et al. (1998) represents an advance on the symmetric Hopfield model.

NOTES AND REFERENCES

1. The direct method of Lyapunov is also referred in the literature as the second method. For an early account of this pioneering work, see the book by LaSalle and Lefschetz (1961).
The alternative spelling, Liapunov, is frequently used in literature; the difference in spelling arose during transliteration from Russian characters (Brogan, 1985).
2. A *nonautonomous* dynamical system is defined by the state equation

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}(\mathbf{x}(t), t)$$

with the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. For a nonautonomous system the vector field $\mathbf{F}(\mathbf{x}(t), t)$ depends on time t . Therefore, unlike the case of an autonomous system, we generally cannot set the initial time equal to zero (Parker and Chua, 1989).

3. For a rigorous definition of an attractor we offer the following (Lanford, 1981; Lichtenberg and Lieberman, 1992):
A subset (manifold) M of the state space is called an attractor if:
 - M is invariant under the flow
 - There is an (open) neighborhood around M that shrinks down to M under the flow
 - No part of M is transient
 - M cannot be decomposed into two nonoverlapping invariant pieces.

4. Integrate-and-Fire Neuron

The additive model of Eq. (14.14) does not fully capture the essence of what a biological neuron does. In particular, it ignores the timing information encoded into action potentials; action potentials are described briefly in qualitative terms in Chapter 1. Hopfield (1994) describes a dynamical model that accounts for action potentials by considering an integrate-and-fire neuron. The operation of such a neuron is described by the first-order differential equation

$$C \frac{d}{dt}u(t) = -\frac{1}{R}(u(t) - u_0) + i(t) \quad (1)$$

where

- $u(t)$ = interior potential of the neuron
- C = capacitance of the membrane surrounding the neuron
- R = leakage resistance of the membrane
- $i(t)$ = electrical current injected into the neuron by another neuron
- u_0 = potential to which the neuron is reduced when $i(t)$ vanishes.

An action potential is generated each time the interior potential $u(t)$ reaches a threshold value.

The action potentials are treated as Dirac delta (impulse) functions as shown by

$$g_k(t) = \sum_n \delta(t - t_{k,n}) \quad (2)$$

where $t_{k,n}$, $n = 1, 2, 3, \dots$ denotes the times at which neuron k fires action potentials. These times are defined by Eq. (1).

The behavior of total current $i_k(t)$ flowing into neuron k is modeled as

$$\frac{d}{dt}i_k(t) = -\frac{1}{\tau}i_k(t) + \sum_j w_{kj}g_j(t) \quad (3)$$

where w_{kj} is the synaptic weight from neuron j to neuron k , τ is a characteristic time constant of neuron k , and the function $g_j(t)$ is defined in accordance with Eq. (2).

The additive model of Eq. (14.14) may be viewed as a special case of Eq. (3). Specifically, the spiky nature of $g_j(t)$ is ignored by replacing it with the convolution of $g_j(t)$ with a smoothing function. Such a move is justified if, during a reasonable time interval, there are many contributions to the sum on the right-hand side of Eq. (3) due to high connectivity, and all that we are really interested in is the short-term behavior of the firing rate of neuron k .

5. The *Little model* (Little, 1974; Little and Shaw, 1975) uses the same synaptic weights as the Hopfield model. However, they differ from each other in that the Hopfield model uses *asynchronous (serial) dynamics*, whereas the Little model uses *synchronous (parallel) dynamics*. Accordingly, they exhibit different convergence properties (Bruck, 1990; Goles and Martinez, 1990): The Hopfield network will always converge to a stable state, whereas the Little model will always converge to a stable state or a limit cycle of length at most 2. By such a "limit cycle" we mean that the cycles in the state space of the network are of a length less than or equal to 2.

6. Nonmonotonic Activation Function

Various proposals have been made in literature for overcoming the limitations of the Hopfield model as a content-addressable memory. Perhaps the most significant improvement suggested to date is that due to Morita (1993), which applies to the continuous (analog) form of the Hopfield model. The modification is confined to the activation function $\varphi(\cdot)$ of a neuron, thereby retaining the simplicity of the network as an associative memory. Specifically, the usual hard-limiting or sigmoid activation function of each neuron in the network is replaced by a *nonmonotonic function*. In mathematical terms this activation function is defined as the product of two factors, as shown by

$$\varphi(v) = \left(\frac{1 - \exp(-av)}{1 + \exp(-av)} \right) \left(\frac{1 + \kappa \exp(b(|v| - c))}{1 + \exp(b(|v| - b))} \right) \quad (1)$$

where v is the induced local field. The first factor on the right-hand side of Eq. (1) is the usual sigmoid function (hyperbolic tangent) used in the continuous version of the Hopfield network; the parameter a is the gain of the neuron. The second factor is responsible for making the activation function $\varphi(v)$ nonmonotonic. Two of the parameters characterizing this second factor, namely b and c , are positive constants; the remaining parameter κ is usually negative. In the experiments performed by Morita (1993), the following parameter values were used:

$$\begin{aligned} a &= 50; & b &= 15 \\ c &= 0.5; & \kappa &= -1 \end{aligned}$$

According to Morita, the exact form of the activation function and the parameters used to describe it are not very critical; the essential factor is the nonmonotonic property of the activation function.

The model of a content-addressable memory described by Morita exhibits two interesting properties (Yoshizawa et al., 1993):

1. For a network made up of N neurons, the storage capacity of the model is about $0.3N$, which (for large N) is much greater than the corresponding value $N/(2\log N)$ of the conventional Hopfield model.
2. The model exhibits *no* spurious states; instead, when it fails to recall a correct memorized pattern, the state of the network is driven into a chaotic behavior. The notion of chaos is discussed in Section 14.13.

7. The idea of a correlation function $C(q, r)$ as defined in Eq. (14.84) was known in statistics from the work of Rényi (1970). However, the use of it to characterize a strange attractor is due to Grassberger and Procaccia (1983). They originally discussed the use of $C(q, r)$ in the context of correlation dimension for $q = 2$.
8. The construction of dynamics using independent coordinates from a time series was first advocated by Packard et al. (1980). However, this paper does not give proof, and its "derivative" embeddings rather than time-delay embeddings. The idea of time-delay delay coordinate embeddings is attributed to Ruelle and Takens. Specifically, in 1981 Takens published a mathematically profound paper on time-delay embeddings, which applies to attractors that are surfaces, or like a torus; see also the paper by Mañé (1981) on the same subject published in the same issue. Takens' paper is difficult to read for non-mathematicians, and Mañé's paper is even more difficult to read. The idea of delay coordinate mapping was refined in 1991 by Sauer et al. The approach taken in this latter paper integrates and expands on previous results due to Whitney (1936) and Takens (1981).
9. The treatment of biological neural networks as nonlinear dynamical systems that exhibit oscillatory behavior and traveling waves has a long history (Wilson and Cowan 1972; Amari, 1977a, 1977b; Amari and Arbib, 1977); see also the discussion in Carpenter et al. (1987).

PROBLEMS

Dynamical Systems

- 14.1 Restate Lyapunov's theorems for the state vector $\mathbf{x}(0)$ as the equilibrium state of a dynamical system.
- 14.2 Verify the block diagrams of Figs. 14.8a and 14.8b for the neurodynamical equation (14.18) and (14.19), respectively.
- 14.3 Consider a general neurodynamical system with an unspecified dependence on internal dynamical parameters, external dynamical stimuli, and state variables. The system is defined by the state equations

$$\frac{dx_j}{dt} = \varphi_j(\mathbf{W}, \mathbf{u}, \mathbf{x}), \quad j = 1, 2, \dots, N$$

where the matrix $\mathbf{W}$ represents the internal dynamical parameters of the system, the vector $\mathbf{u}$ represents the external dynamical stimuli, and $\mathbf{x}$ is the state vector whose j th element is denoted by x_j . Assume that trajectories of the system converge onto point attractors for values of $\mathbf{W}$, $\mathbf{u}$, and initial states $\mathbf{x}(0)$ in some operating region of the state space (Pineda, 1988b). Discuss how the system described may be used for the following applications:

- (a) Continuous mapper, with $\mathbf{u}$ as input and $\mathbf{x}(\infty)$ as output
- (b) Autoassociative memory, with $\mathbf{x}(0)$ as input and $\mathbf{x}(\infty)$ as output

Hopfield Models

- 14.4 Consider a Hopfield network made up of five neurons, which is required to store the following three fundamental memories:

$$\xi_1 = [+1, +1, +1, +1, +1]^T$$

$$\xi_2 = [+1, -1, -1, +1, -1]^T$$

$$\xi_3 = [-1, +1, -1, +1, +1]^T$$

- (a) Evaluate the 5-by-5 synaptic weight matrix of the network.
 (b) Use asynchronous updating to demonstrate that all three fundamental memories, ξ_1 , ξ_2 , and ξ_3 , satisfy the alignment condition.
 (c) Investigate the retrieval performance of the network when it is presented with a noisy version of ξ_1 in which the second element is reversed in polarity.
- 14.5 Investigate the use of synchronous updating for the retrieval performance of the Hopfield network described in Problem 14.4.
- 14.6 (a) Show that

$$\xi_1 = [-1, -1, -1, -1, -1]^T$$

$$\xi_2 = [-1, +1, +1, -1, +1]^T$$

$$\xi_3 = [+1, -1, +1, -1, -1]^T$$

are also fundamental memories of the Hopfield network described in Problem 14.4. How are these fundamental memories related to those of Problem 14.4?

- (b) Suppose that the first element of the fundamental memory ξ_3 in Problem 14.4 is masked (i.e., reduced to zero). Determine the resulting pattern produced by the Hopfield network. Compare this result with the original form of ξ_3 .
- 14.7 Consider a simple Hopfield network made up of two neurons. The synaptic weight matrix of the network is

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

The bias applied to each neuron is zero. The four possible states of the network are

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

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

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

$$\mathbf{x}_4 = [+1, -1]^T$$

- (a) Demonstrate that states $\mathbf{x}_2$ and $\mathbf{x}_4$ are stable, whereas states $\mathbf{x}_1$ and $\mathbf{x}_3$ exhibit a limit cycle. Do this demonstration using the following tools:
1. The alignment (stability) condition
 2. The energy function
- (b) What is the length of the limit cycle characterizing states $\mathbf{x}_1$ and $\mathbf{x}_3$?
- 14.8 In this problem we derive the formula of Eq. (14.55) for the storage capacity almost without errors pertaining to the Hopfield network used as a content-addressable memory.
- (a) The asymptotic behavior of the error function is closely described by

$$\text{erf}(y) \approx 1 - \frac{e^{-y^2}}{\sqrt{\pi}y} \text{ for large } y$$

Using this approximation, show that the conditional probability of Eq. (14.53) may be approximated as

$$P(v_j > 0 | \xi_{\nu,j} = +1) \approx 1 - \frac{e^{-\rho/2}}{\sqrt{2\pi\rho}}$$

where ρ is the signal-to-noise ratio. Show that the probability of stable patterns is correspondingly approximated as

$$p_{\text{stab}} \approx 1 - \frac{Ne^{-\rho/2}}{\sqrt{\pi\rho}}$$

- (b) The second term in the formula for p_{stab} given in part (a) is the probability that a bit in a fundamental memory is unstable. For the definition of storage capacity almost without errors it is not sufficient to require that this term be small; rather it must be small compared to $1/N$, where N is the size of the Hopfield network. Show that the signal-to-noise ratio must satisfy the condition

$$\rho > 2\log_e N + \frac{1}{2}\log_e(2\pi\rho)$$

- (c) Using the result derived in part (b), show that the minimum permissible value of the signal-to-noise ratio for the perfect recall of most of the fundamental memories is

$$\rho_{\min} = 2\log_e N$$

What is the corresponding value of p_{stab} ?

- (d) Using the results of part (c), show that

$$M_{\max} \approx \frac{N}{2\log_e N}$$

as described in Eq. (14.55).

- (e) The formula derived in part (d) for the storage capacity is based on the premise that most of the fundamental memories are stable. For a more stringent definition of storage capacity without errors, we require that all of the fundamental memories be retrieved correctly. Using this latter definition, show that the maximum number of fundamental memories that can be stored in the Hopfield network is given by (Amit 1989)

$$M_{\max} \approx \frac{N}{4\log_e N}$$

- 14.9 Show that the energy function of a Hopfield network may be expressed as

$$E = -\frac{N}{2} \sum_{\nu=1}^M m_{\nu}^2$$

where m_{ν} denotes overlaps defined by

$$m_{\nu} = \frac{1}{N} \sum_{j=1}^N x_j \xi_{\nu,j}, \quad \nu = 1, 2, \dots, M$$

where x_j is the j th element of the state vector $\mathbf{x}$, $\xi_{\nu,j}$ is the j th element of the fundamental memory ξ_{ν} , and M is the number of fundamental memories.

- 14.10 A Hopfield network is designed to store the two fundamental memory patterns $(+1, +1, -1, +1, +1)$ and $(+1, -1, +1, -1, +1)$. The synaptic matrix of the network is given by

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

- (a) The sum of the eigenvalues of the matrix $\mathbf{W}$ is zero. Why?
 (b) The state space of the network is a subspace of $\mathbb{R}^5$. Specify the configuration of this subspace.
 (c) Specify the subspace $\mathcal{M}$ spanned by the fundamental memory vectors, and the null subspace $\mathcal{N}$ of the weight matrix $\mathbf{W}$. What are the fixed points (stable states) and spurious states of the network?

(The reader may wish to refer to the paper by deSilva and Attikiouzel (1992) for a more detailed description of the dynamics of the network described here.)

- 14.11** Figure P14.11 shows a piecewise-linear form of nonmonotonic activation function. The recalling dynamics of the Hopfield network using this approximation are defined by

$$\frac{d}{dt}\mathbf{v}(t) = -\mathbf{v}(t) + \mathbf{W}\mathbf{x}(t), \quad \mathbf{x}(t) = \text{sgn}(\mathbf{v}(t)) - k\mathbf{v}(t)$$

where $\mathbf{v}(t)$ is the vector of induced local fields, $\mathbf{W}$ is the synaptic weight matrix, $\mathbf{x}(t)$ is the state (output) vector, and $-k$ is a negative constant slope. Let $\mathbf{v}$ be an equilibrium state of the network that lies in the quadrant of the fundamental memory ξ_1 , and let

$$\bar{\mathbf{x}} = \text{sgn}(\mathbf{v}) - k\mathbf{v}$$

Show that $\bar{\mathbf{x}}$ is characterized by the following three conditions (Yoshizawa et al., 1993):

- (a) $\sum_{i=1}^N \bar{x}_i \xi_{\mu,i} = 0, \quad \mu = 2, 3, \dots, M$
 (b) $\sum_{i=1}^N \bar{x}_i \xi_{1,i} = M$
 (c) $\bar{x}_i < 1, \quad i = 1, 2, \dots, N$

where $\xi_1, \xi_2, \dots, \xi_M$ are the fundamental memories stored in the network, $\xi_{\mu,i}$ is the i th element of ξ_μ , $\bar{x}_i$ is the i th element of $\bar{\mathbf{x}}$, and N is the number of neurons.

- 14.12** Consider the simple neurodynamical model described by the system of equations

$$\frac{dv_j}{dt} = -v_j + \sum_i w_{ji} \varphi(v_i) + I_j, \quad j = 1, 2, \dots, N$$

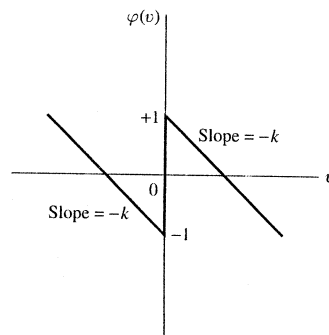


FIGURE P14.11

The system described always converges to a unique point attractor, provided that the synaptic weights w_{ji} satisfy the condition

$$\sum_j \sum_i w_{ji}^2 < \frac{1}{(\max|\varphi'|)^2}$$

where $\varphi' = d\varphi/dv_j$. Explore the validity of this condition. You may refer to the paper (Atiya, 1987) where this condition is derived.

Cohen-Grossberg Theorem

- 14.13** Consider the Lyapunov function E defined in Eq. (14.57). Show that

$$\frac{dE}{dt} \leq 0$$

provided that the conditions in Eqs. (14.59) to (14.61) are satisfied.

- 14.14** In Section 14.10 we derived the Lyapunov function of the BSB model by applying the Cohen-Grossberg theorem. In carrying out the derivation, we omitted some of the details leading to Eq. (14.73). Fill in the details.

- 14.15** Figure P14.15 shows a plot of the nonmonotonic activation function due to Morita (1993) discussed in note 6. This activation function is used in place of the hyperbolic tangent function in the construction of a Hopfield network. Is the Cohen-Grossberg theorem applicable to the associative memory so constructed? Justify your answer.

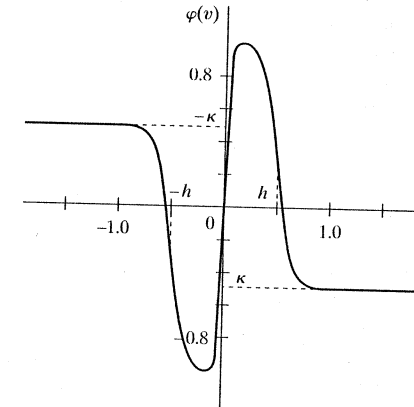


FIGURE P14.15

15

Dynamically Driven Recurrent Networks

INTRODUCTION

As mentioned in the previous chapter, *recurrent networks* are neural networks with one or more feedback loops. The feedback can be of a *local* or *global* kind. In this chapter we continue the study of recurrent networks with global feedback.

Given a multilayer perceptron as the basic building block, the application of global feedback can take a variety of forms. We may have feedback from the output neurons of the multilayer perceptron to the input layer. Yet another possible form of global feedback is from the hidden neurons of the network to the input layer. When the multilayer perceptron has two or more hidden layers, the possible forms of global feedback expand even further. The point is that recurrent networks have a rich repertoire of architectural layouts.

Basically, there are two functional uses of recurrent networks:

- *Associative memories*
- *Input-output mapping networks*

The use of recurrent networks as associative memories is considered in detail in Chapter 14. In the present chapter, we will study their use as input-output mapping networks. Whatever the use, an issue of particular concern in the study of recurrent networks is that of *stability*; that issue is also considered in Chapter 14.

By definition, the input space of a mapping network is mapped onto an output space. For this kind of an application, a recurrent network responds *temporally* to an externally applied input signal. We may therefore speak of the recurrent networks considered in this chapter as *dynamically driven recurrent networks*. Moreover, the application of feedback enables recurrent networks to acquire *state* representations, which make them suitable devices for such diverse applications as nonlinear prediction and modeling, adaptive equalization of communication channels, speech processing, plant control, and automobile engine diagnostics. As such, recurrent networks offer an alternative to the dynamically driven feedforward networks described in Chapter 13.

Because of the beneficial effects of global feedback, they may actually fare better in these applications. The use of global feedback has the potential of reducing the memory requirement significantly.

Organization of the Chapter

The chapter is organized in four parts: architectures, theory, learning algorithms, and applications. Part 1, consisting of Section 15.2, deals with recurrent network architecture:

Part 2, consisting of Sections 15.3 to 15.5, deals with theoretical aspects of recurrent networks. Section 15.3 describes the state-space model and the related issues of controllability and observability. Section 15.4 derives an equivalent to the state-space model known as the nonlinear autoregressive with exogenous inputs model. Section 15.5 discusses some theoretical issues pertaining to the computational power of recurrent networks.

Part 3, consisting of Sections 15.6 to 15.12, is devoted to learning algorithms and related issues. It starts with an overview of the subject matter in Section 15.6. Then Section 15.7 discusses back-propagation through time that builds on material presented in Chapter 4. Section 15.8 discusses another popular algorithm: real-time recurrent learning. In Section 15.9 we present a brief review of classical Kalman filter theory, followed by a description of the decoupled extended Kalman filtering algorithm in Section 15.10. A computer experiment on this latter algorithm for recurrent learning is presented in Section 15.11. Gradient-based recurrent learning suffers from the vanishing gradients problem, which is discussed in Section 15.12.

The fourth and last part of the chapter, consisting of Sections 15.13 and 15.14, deals with two important applications of recurrent networks. Section 15.13 discusses system identification. Section 15.14 discusses model-reference adaptive control.

The chapter concludes with some final remarks in Section 15.15.

15.2 RECURRENT NETWORK ARCHITECTURES

As mentioned in the introduction, the architectural layout of a recurrent network takes many different forms. In this section we describe four specific network architectures, each of which highlights a specific form of global feedback.¹ They share the following common features:

- They all incorporate a *static* multilayer perceptron or parts thereof.
- They all exploit the nonlinear mapping capability of the multilayer perceptron.

Input-Output Recurrent Model

Figure 15.1 shows the architecture of a generic recurrent network that follows naturally from a multilayer perceptron. The model has a single input that is applied to a tapped-delay-line memory of q units. It has a single output that is fed back to the input via another tapped-delay-line memory also of q units. The contents of these two tapped-delay-line memories are used to feed the input layer of the multilayer perceptron. The present value of the model input is denoted by $u(n)$, and the corresponding