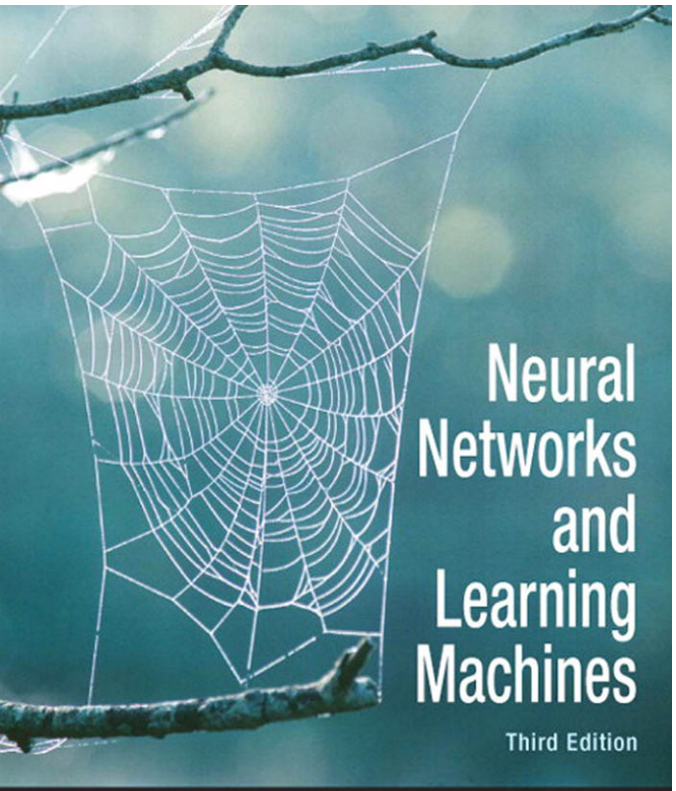

Neural Networks and Learning Machines

Third Edition

Simon Haykin

C H A P T E R 1 1

Stochastic Methods Rooted in Statistical Mechanics

ORGANIZATION OF THE CHAPTER

The theme of this chapter is the study of stochastic algorithms for simulation, optimization, and learning by building on ideas rooted in statistical mechanics.

The chapter is organized as follows:

1. Section 11.1 is an introductory section that provides motivation for studying the subject matter just described.
2. Section 11.2 gives an introductory treatment of statistical mechanics, with an emphasis on the concepts of free energy and entropy, as viewed in thermodynamics.
3. Section 11.3 discusses a special kind of stochastic processes known as Markov chains, the use of which is commonly encountered in the study of statistical mechanics.
4. Sections 11.4, 11.5, and 11.6 discuss the following three stochastic simulation/optimization methods:
   - The Metropolis algorithm
   - Simulated annealing
   - Gibbs sampling

   The Metropolis algorithm and Gibbs sampling provide tools for the simulation of stationary and nonstationary processes, respectively, and simulated annealing is oriented toward optimization.
5. The next three sections, Sections 11.7, 11.8, and 11.9, address the following stochastic machines rooted in statistical mechanics:
   - The Boltzmann machine
   - Logistic belief nets
   - Deep belief nets

   The last machine has some unique properties that overcome practical limitations of the classical Boltzmann machine and logistic belief nets.
6. Section 11.10 describes deterministic annealing, which is an approximation to simulated annealing; despite its name, deterministic annealing is a stochastic algorithm. Section 11.11 introduces the expectation-maximization algorithm and discusses an analogy of deterministic annealing with this algorithm.

The chapter concludes with a summary and discussion in Section 11.12.

11.1 INTRODUCTION

For our last class of unsupervised (self-organized) learning systems, we turn to statistical mechanics as the source of ideas. The subject of *statistical mechanics* encompasses the formal study of macroscopic equilibrium properties of large systems of elements that are subject to the microscopic laws of mechanics. The main aim of statistical mechanics is to derive the thermodynamic properties of macroscopic bodies starting from the motion of microscopic elements such as atoms and electrons (Landau and Lifshitz, 1980; Parisi, 1988). The number of degrees of freedom encountered here is enormous, making the use of probabilistic methods mandatory. As with Shannon's information theory, the concept of entropy plays a vital role in the study of statistical mechanics. In this context, we may say the following:

The more ordered the system, or the more concentrated the underlying probability distribution, the smaller the entropy will be.

By the same token, we can say that the more disordered the system, or the more uniform the underlying probability distribution, the larger the entropy will be. In 1957, Jaynes showed that entropy can be used not only as the starting point of formulating statistical inference as described in the previous chapter, but also for generating the Gibbs distribution that is basic to the study of statistical mechanics.

Interest in the use of statistical mechanics as a basis for the study of neural networks goes back to the early works of Cragg and Tamperley (1954) and Cowan (1968). The *Boltzmann machine* (Hinton and Sejnowski, 1983, 1986; Ackley et al., 1985) is perhaps the first multilayer learning machine inspired by statistical mechanics. The machine is named in recognition of the formal equivalence between Boltzmann's original work on statistical thermodynamics and the network's own dynamic behavior. Basically, the Boltzmann machine is a stochastic system for modeling the underlying probability distribution of a given data set, from which conditional distributions for use in tasks such as pattern completion and pattern classification can be derived. Unfortunately, the learning process in the early version of the Boltzmann machine is painfully slow. This shortcoming has motivated new developments in the Boltzmann machine and inspired the formulation of new stochastic machines. These issues constitute the material presented in this chapter.

11.2 STATISTICAL MECHANICS

Consider a physical system with many degrees of freedom, which can reside in any one of a large number of possible states. Let p_i denote the probability of occurrence of state i of a stochastic system with the following properties:

$$p_i \geq 0 \qquad \text{for all } i \tag{11.1}$$

and

$$\sum_i p_i = 1 \tag{11.2}$$

Let E_i denote the *energy* of the system when it is in state i. A fundamental result from statistical mechanics tells us that when the system is in thermal equilibrium with its surrounding environment, state i occurs with a probability defined by

$$p_i = \frac{1}{Z}\exp\left(-\frac{E_i}{k_B T}\right) \tag{11.3}$$

where T is the *absolute temperature* in kelvins, k_B is *Boltzmann's constant,* and Z is a constant that is independent of all states. One degree kelvin corresponds to $-273°$ on the Celsius scale, and $k_B = 1.38 \times 10^{-23}$ joules/kelvin.

Equation (11.2) defines the condition for the normalization of probabilities. Imposing this condition on Eq. (11.3), we get

$$Z = \sum_i \exp\left(-\frac{E_i}{k_B T}\right) \tag{11.4}$$

The normalizing quantity Z is called the *sum over states*, or the *partition function.* (The symbol Z is commonly used because the German name for this term is *Zustadsumme.*) The probability distribution of Eq. (11.3) is called the *canonical distribution*, or *Gibbs distribution*[1]; the exponential factor $\exp(-E_i/k_B T)$ is called the *Boltzmann factor.*

The following two points are noteworthy from the Gibbs distribution:

1. States of low energy have a higher probability of occurrence than states of high energy.
2. As the temperature T is reduced, the probability is concentrated on a smaller subset of low-energy states.

The parameter T may be viewed as a *pseudotemperature* that controls thermal fluctuations representing the effect of "synaptic noise" in a neuron. Its precise scale is therefore irrelevant. Accordingly, we may choose to measure it by setting the constant k_B equal to unity and thereby redefine the probability p_i and partition function Z as follows, respectively:

$$p_i = \frac{1}{Z}\exp\left(-\frac{E_i}{T}\right) \tag{11.5}$$

and

$$Z = \sum_i \exp\left(-\frac{E_i}{T}\right) \tag{11.6}$$

Henceforth our treatment of statistical mechanics is based on these two definitions, where T is referred to simply as the *temperature of the system.* From Eq. (11.5), we find that $-\log p_i$ may be viewed as a form of "energy" measured at unit temperature.

Free Energy and Entropy

The Helmholtz *free energy* of a physical system, denoted by F, is defined in terms of the partition function Z as

$$F = -T \log Z \tag{11.7}$$

The *average energy* of the system is defined by

$$<E> = \sum_i p_i E_i \tag{11.8}$$

where $<\cdot>$ denotes the ensemble-averaging operation. Thus, using Eqs. (11.5) to (11.8), we see that the difference between the average energy and free energy is given by

$$<E> - F = -T \sum_i p_i \log p_i \tag{11.9}$$

The quantity on the right-hand side of Eq. (11.9), except for the temperature T, is recognized as the *entropy* of the system, as shown by

$$H = -\sum_i p_i \log p_i \tag{11.10}$$

(This definition is consistent with that introduced in Chapter 10 on information-theoretic models.)

We may therefore rewrite Eq. (11.9) in the form

$$<E> - F = TH$$

or, equivalently,

$$F = <E> - TH \tag{11.11}$$

Consider two systems A and A' placed in thermal contact with each other. Suppose that system A is small compared with system A', so that A' acts as a *heat reservoir* at some constant temperature T. The total entropy of the two systems tends to increase in accordance with the relation

$$\Delta H + \Delta H' \geq 0$$

where ΔH and $\Delta H'$ denote the entropy changes of systems A and A', respectively (Reif, 1965). The implication of this relation, in light of Eq. (11.11), is that the free energy of the system, F, tends to decrease and become a minimum in an equilibrium situation. From statistical mechanics, we find that the resulting probability distribution is defined by the Gibbs distribution. We thus have an important principle called the *principle of minimal free energy*, which may be stated as follows (Landau and Lifshitz, 1980; Parisi, 1988):

> *The minimum of the free energy of a stochastic system with respect to variables of the system is achieved at thermal equilibrium, at which point the system is governed by the Gibbs distribution.*

Nature likes to find a physical system with minimum free energy.

11.3 MARKOV CHAINS

Consider a system whose evolution is described by a stochastic process $\{X_n, n = 1, 2, ...\}$, consisting of a family of random variables. The value x_n, assumed by the random variable X_n at discrete time n, is called the *state* of the system at that time instant. The space of all possible values that the random variables can assume is called the *state space* of the system. If the structure of the stochastic process $\{X_n, n = 1, 2, ...\}$ is such that the dependence of x_{n+1} on the entire past is completely captured by the dependence on the last sample x_n, we say that the process is a *Markov chain* (Feller, 1950; Ash, 1965). More precisely, we have

$$P(X_{n+1} = x_{n+1}|X_n = x_n, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n) \qquad (11.12)$$

which is called the *Markov property.* In words, we say the following:

A sequence of random variables $X_1, X_2, \ldots, X_n, X_{n+1}$ forms a Markov chain, if the probability that the system is in state x_{n+1}, given the sequence of past states it has gone through, is exclusively determined by the state of the system at time n.

We may therefore think of the Markov chain as a *generative model*, consisting of a number of states linked together (on a pairwise basis) by possible transitions. Each time a particular state is visited, the model outputs the symbol associated with that state.

Transition Probabilities

In a Markov chain, the transition from one state to another is *probabilistic*, but the production of an output symbol is deterministic. Let

$$p_{ij} = P(X_{n+1} = j|X_n = i) \qquad (11.13)$$

denote the *transition probability* from state i at time n to state j at time $n + 1$. Since the p_{ij} are conditional probabilities, all transition probabilities must satisfy two conditions:

$$p_{ij} \geq 0 \qquad \text{for all } i, j \qquad (11.14)$$

$$\sum_j p_{ij} = 1 \qquad \text{for all } i \qquad (11.15)$$

We will assume that the transition probabilities are fixed and do not change with time; that is, Eq. (11.13) is satisfied for all time n. In such a case, the Markov chain is said to be *homogeneous* in time.

In the case of a system with a finite number of possible states K, for example, the transition probabilities constitute the K-by-K matrix

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \vdots & \vdots & & \vdots \\ p_{K1} & p_{K2} & \cdots & p_{KK} \end{bmatrix} \qquad (11.16)$$

whose individual elements satisfy the conditions described in Eqs. (11.14) and (11.15); the latter condition says that each row of **P** must add to unity. A matrix of this type is called a *stochastic matrix.* Any stochastic matrix can serve as a matrix of transition probabilities.

The definition of one-step transition probability given in Eq. (11.13) may be generalized to cases where the transition from one state to another takes place in some fixed number of steps. Let $p_{ij}^{(m)}$ denote the *m-step transition probability* from state i to state j:

$$p_{ij}^{(m)} = P(X_{n+m} = x_j|X_n = x_i), \qquad m = 1, 2, \ldots \qquad (11.17)$$

We may view $p_{ij}^{(m)}$ as the sum over all intermediate states, k, through which the system passes in its transition from state i to state j. Specifically, $p_{ij}^{(m+1)}$ is related to $p_{ij}^{(m)}$ by the recursive relation

$$p_{ij}^{(m+1)} = \sum_k p_{ik}^{(m)} p_{kj}, \qquad m = 1, 2, \ldots \tag{11.18}$$

with

$$p_{ik}^{(1)} = p_{ik}$$

Equation (11.18) may be generalized as

$$p_{ij}^{(m+n)} = \sum_k p_{ik}^{(m)} p_{kj}^{(n)}, \qquad m, n = 1, 2, \ldots \tag{11.19}$$

which is a special case of the *Chapman–Kolmogorov identity* (Feller, 1950).

Specification of a Markov Chain

With the notions of state and transition probability at hand, we may now summarize how a Markov chain is specified:

(i) *A stochastic model is identified in terms of the following:*

- *a finite set of K possible states denoted by $S = \{1, 2, \ldots, K\}$;*
- *a corresponding set of probabilities $\{p_{ij}\}$, where p_{ij} is the transition probability from state i to state j, subject to two conditions:*

$$p_{ij} \geq 0$$

and

$$\sum_j p_{ij} = 1 \qquad \text{for all } i$$

(ii) *Given the stochastic model just described, a Markov chain is specified in terms of a sequence of random variables $X_0, X_1, X_2, \ldots$ whose values are taken from the set S in accordance with the Markov property*

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j | X_n = i)$$

which holds for all times n, all states $i, j \in S$, and all possible sequences $i_0, \ldots, i_{n-1}$ pertaining to earlier states.

Recurrent Properties

Suppose a Markov chain starts in state i. State i is said to be a *recurrent* state if the Markov chain returns to state i with probability 1; that is,

$$p_i = P(\text{ever returning to state } i) = 1$$

If the probability p_i is less than 1, state i is said to be a *transient* state (Leon-Garcia, 1994).

If the Markov chain starts in a recurrent state, that state reoccurs an infinite number of times. If it starts in a transient state, that state reoccurs only a finite number of times, which may be explained as follows: We may view the reoccurrence of state i as a *Bernoulli trial*[2] with a probability of success equal to p_i. The number of returns is thus a geometric random variable with a mean of $(1 - p_i^{-1})$. If $p_i < 1$, it follows that the number of an infinite number of successes is zero. Therefore, a transient state does not reoccur after some finite number of returns.

If a Markov chain has some transient states and some recurrent states, then the process will eventually move only among the recurrent states.

Periodicity

Figure 11.1 shows a Markov chain of the recurrent type. This Markov chain moves through a sequence of substates, ending up in the same substate after three time-steps. The figure illustrates a recurrent Markov chain that is periodic.

In light of Fig. 11.1, a recurrent Markov chain is said to be *periodic* if all of its states can be grouped into d disjoint subsets $S_1, S_2, \ldots, S_d$, where $d > 1$, and in such a way that all the transitions from one subset lead to the next subset; in the figure, $d = 3$. More precisely, a periodic recurrent Markov chain satisfies the following condition (Bertsekas and Tsitsiklis, 2002):

$$\text{If } i \in S_k \text{ and } p_{ij} > 0, \text{ then } \begin{cases} j \in S_{k+1}, & \text{for } k = 1, \ldots, d-1 \\ j \in S_1, & \text{for } k = d \end{cases}$$

A recurrent Markov chain is said to be *aperiodic* if it is not periodic.

Irreducible Markov Chains

The state j of a Markov chain is said to be *accessible* from state i if there is a finite sequence of transitions from i to j with positive probability. If the states i and j are accessible to each

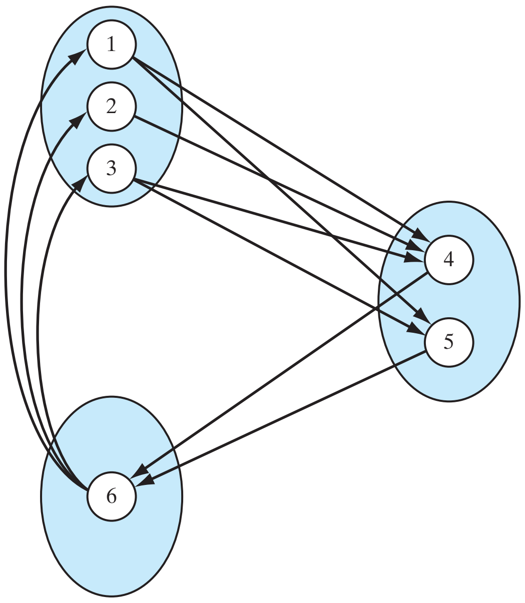

FIGURE 11.1 A periodic recurrent Markov chain with $d = 3$.

other, they are said to *communicate* with each other. This communication is described by writing $i \leftrightarrow j$. Clearly, if state i communicates with state j and state j communicates with state k (that is, $i \leftrightarrow j$ and $j \leftrightarrow k$), then state i communicates with state k (that is, $i \leftrightarrow k$).

If two states of a Markov chain communicate with each other, they are said to belong to the same *class.* In general, the states of a Markov chain consist of one or more disjoint classes. If, however, all the states consist of a single class, the Markov chain is said to be *indecomposible*, or *irreducible.* In other words, by starting at any state of an irreducible Markov chain, we can reach any other state with positive probability. Reducible chains are of little practical interest in most areas of application. Accordingly, we restrict our attention to irreducible chains.

Consider an irreducible Markov chain that starts in a recurrent state i at time $n = 0$. Let $T_i(k)$ denote the time that elapses between the $(k - 1)$th and kth returns to state i. The *mean recurrence time* of state i is defined as the expectation of $T_i(k)$ over the returns k. The *steady-state probability* of state i, denoted by π_i, is equal to the reciprocal of the mean recurrence time $\mathbb{E}[T_i(k)]$, as shown by

$$\pi_i = \frac{1}{\mathbb{E}[T_i(k)]}$$

If $\mathbb{E}[T_i(k)] < \infty$—that is, $\pi_i > 0$—the state i is said to be a *positive recurrent* (*persistent*) *state.* If $\mathbb{E}[T_i(k)] = \infty$—that is, $\pi_i = 0$—the state i is said to be a *null recurrent* (*persistent*) *state.* The implication of $\pi_i = 0$ is that the Markov chain will eventually reach a point where a return to state i is impossible. Positive recurrence and null recurrence are *different* class properties, which means that a Markov chain with positive recurrent and null recurrent states is reducible.

Ergodic Markov Chains

In principle, *ergodicity* means that we may substitute time averages for ensemble averages. In the context of a Markov chain, ergodicity means that the long-term proportion of time spent by the chain in state i corresponds to the steady-state probability π_i, which may be justified as follows: The *proportion of time spent in state i after k returns*, denoted by $v_i(k)$, is defined by

$$v_i(k) = \frac{k}{\sum_{l=1}^{k} T_i(l)}$$

The return times $T_i(l)$ form a sequence of statistically *independent and identically distributed* (iid) random variables, since, by definition, each return time is statistically independent of all previous return times. Moreover, in the case of a recurrent state i, the chain returns to state i an infinite number of times. Hence, as the number of returns, k, approaches infinity, the *law of large numbers* states that the proportion of time spent in state i approaches the steady-state probability, as shown by

$$\lim_{k \to \infty} v_i(k) = \pi_i \qquad \text{for } i = 1, 2, ..., K \tag{11.20}$$

where K is the total number of states.

A sufficient, but not necessary, condition for a Markov chain to be *ergodic* is for it to be both irreducible and aperiodic.

Convergence to Stationary Distributions

Consider an ergodic Markov chain characterized by a stochastic matrix $\mathbf{P}$. Let the row vector $\boldsymbol{\pi}^{(n-1)}$ denote the *state distribution vector* of the chain at time $n-1$; the jth element of $\boldsymbol{\pi}^{(n-1)}$ is the probability that the chain is in state x_j at time $n-1$. The state distribution vector at time n is defined by

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(n-1)}\mathbf{P} \tag{11.21}$$

By iteration of Eq. (11.21), we obtain

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(n-1)}\mathbf{P} = \boldsymbol{\pi}^{(n-2)}\mathbf{P}^2 = \boldsymbol{\pi}^{(n-3)}\mathbf{P}^3 = \cdots$$

and finally we may write

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(0)}\mathbf{P}^n \tag{11.22}$$

where $\boldsymbol{\pi}^{(0)}$ is the *initial value* of the state distribution vector. In words, we say the following:

The state distribution vector of the Markov chain at time n is the product of the initial state distribution vector $\boldsymbol{\pi}^{(0)}$ and the nth power of the stochastic matrix $\mathbf{P}$.

Let $p_{ij}^{(n)}$ denote the ij-th element of $\mathbf{P}^n$. Suppose that as time n approaches infinity, $p_{ij}^{(n)}$ tends to π_j independent of i, where π_j is the steady-state probability of state j. Correspondingly, for large n, the matrix $\mathbf{P}^n$ approaches the limiting form of a square matrix with identical rows as shown by

$$\lim_{n\to\infty} \mathbf{P}^n = \begin{bmatrix} \pi_1 & \pi_2 & \cdots & \pi_K \\ \pi_1 & \pi_2 & \cdots & \pi_K \\ \vdots & \vdots & & \vdots \\ \pi_1 & \pi_2 & \cdots & \pi_K \end{bmatrix} \tag{11.23}$$

$$= \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\pi} \\ \vdots \\ \boldsymbol{\pi} \end{bmatrix}$$

where $\boldsymbol{\pi}$ is a row vector consisting of $\pi_1, \pi_2, \ldots, \pi_K$. We then find from Eq. (11.22) that, after rearranging terms,

$$\left[\sum_{j=1}^{K} \pi_j^{(0)} - 1\right]\boldsymbol{\pi} = \mathbf{0}$$

Since, by definition, $\sum_{j=1}^{K} \pi_j^{(0)} = 1$, this condition is satisfied by the vector $\boldsymbol{\pi}$ independent of the initial distribution.

We may now state the *ergodicity theorem* for Markov chains as follows (Feller, 1950; Ash, 1965):

Let an ergodic Markov chain with states $x_1, x_2, \ldots, x_K$ and stochastic matrix $\mathbf{P} = \{p_{ij}\}$ be irreducible. The chain then has a unique stationary distribution to which it converges from any *initial state; that is, there is a unique set of numbers* $\{\pi_j\}_{j=1}^{K}$ *such that*

1. $\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j$ for all i (11.24)

2. $\pi_j > 0$ for all j (11.25)

3. $\sum_{j=1}^{K} \pi_j = 1$ (11.26)

4. $\pi_j = \sum_{i=1}^{K} \pi_i p_{ij}$ for $j = 1, 2, ..., K$ (11.27)

Conversely, suppose that the Markov chain is irreducible and aperiodic and that there exist numbers $\{\pi_j\}_{j=1}^{K}$ *satisfying Eqs. (11.25) through (11.27). Then the chain is ergodic, the* π_j *are given by Eq. (11.24), and the mean recurrence time of state j is* $1/\pi_j$.

The probability distribution $\{\pi_j\}_{j=1}^{K}$ is called an *invariant*, or *stationary*, *distribution*. It is so called because it persists forever once it is established. In light of the ergodicity theorem, we may thus make the following two-part statement:

1. Starting from an arbitrary initial distribution, the transition probabilities of a Markov chain will converge to a stationary distribution provided that such a distribution exists.
2. The stationary distribution of the Markov chain is completely independent of the initial distribution if the chain is ergodic.

EXAMPLE 1 An Ergodic Markov Chain

Consider a Markov chain whose *state-transition diagram* is depicted in Fig. 11.2. The chain has two states x_1 and x_2. The stochastic matrix of the chain is

$$\mathbf{P} = \begin{bmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

which satisfies the conditions of Eqs. (11.14) and (11.15).

Suppose the initial condition is

$$\boldsymbol{\pi}^{(0)} = \begin{bmatrix} \frac{1}{6} & \frac{5}{6} \end{bmatrix}$$

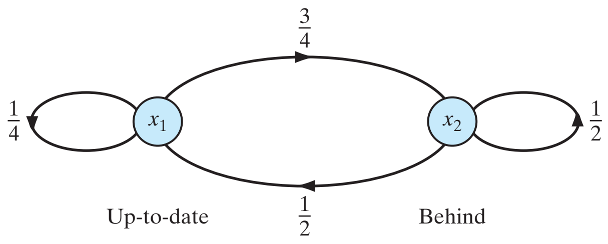

FIGURE 11.2 State-transition diagram of Markov chain for Example 1: The states x_1 and x_2 may be identified as up-to-date and behind, respectively.

From Eq. (11.21), we find that the state distribution vector at time $n = 1$ is

$$\begin{aligned}\boldsymbol{\pi}^{(1)} &= \boldsymbol{\pi}^{(0)}\mathbf{P} \\ &= \begin{bmatrix} \frac{1}{6} & \frac{5}{6} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \\ &= \begin{bmatrix} \frac{11}{24} & \frac{13}{24} \end{bmatrix}\end{aligned}$$

Raising the stochastic matrix $\mathbf{P}$ to power $n = 2, 3,$ and 4, we have

$$\mathbf{P}^2 = \begin{bmatrix} 0.4375 & 0.5625 \\ 0.3750 & 0.6250 \end{bmatrix}$$

$$\mathbf{P}^3 = \begin{bmatrix} 0.4001 & 0.5999 \\ 0.3999 & 0.6001 \end{bmatrix}$$

$$\mathbf{P}^4 = \begin{bmatrix} 0.4000 & 0.6000 \\ 0.4000 & 0.6000 \end{bmatrix}$$

Thus, $\pi_1 = 0.4000$ and $\pi_2 = 0.6000$. In this example, convergence to the stationary distribution is accomplished essentially in $n = 4$ iterations. With both π_1 and π_2 being greater than zero, both states are positive recurrent, and the chain is therefore irreducible. Note also that the chain is aperiodic, since the greatest common divisor of all integers $n \geq 1$ such that $(\mathbf{P}^n)_{jj} > 0$ is equal to 1. We therefore conclude that the Markov chain of Fig. 11.2 is ergodic. ■

EXAMPLE 2 An Ergodic Markov Chain with Stationary Distribution

Consider a Markov chain with a stochastic matrix, some of whose elements are zero, as shown by

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}$$

The state transition diagram of the chain is depicted in Fig. 11.3.

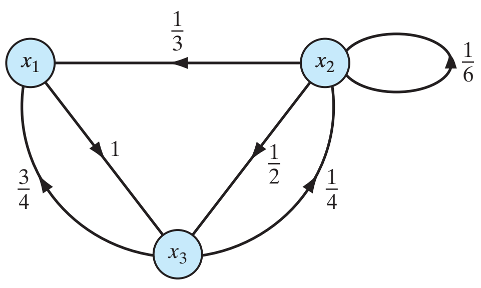

FIGURE 11.3 State-transition diagram of Markov chain for Example 2.

By applying Eq. (11.27), we obtain the following set of simultaneous equations:

$$\pi_1 = \frac{1}{3}\pi_2 + \frac{3}{4}\pi_3$$

$$\pi_2 = \frac{1}{6}\pi_2 + \frac{1}{4}\pi_3$$

$$\pi_3 = \pi_1 + \frac{1}{2}\pi_2$$

By solving these equations for π_1, π_2, and π_3, we get

$$\pi_1 = 0.3953$$
$$\pi_2 = 0.1395$$
$$\pi_3 = 0.4652$$

The given Markov chain is ergodic with its stationary distribution defined by π_1, π_2, and π_3. ■

Classification of States

On the basis of the material presented in this section, we may develop a summary of the classes to which a state can belong as shown in Fig. 11.4 (Feller, 1950; Leon-Garcia, 1994). This figure also includes the associated long-term behavior of the state.

Principle of Detailed Balance

This principle is commonly used in statistical mechanics. In words, the *principle of detailed balance* asserts the following:

At thermal equilibrium, the rate of occurrence of any transition equals the corresponding rate of occurrence of the inverse transition, as shown by

$$\pi_i p_{ij} = \pi_j p_{ji} \tag{11.28}$$

A Markov chain that satisfies the principle of detailed balance is said to be *reversible*.

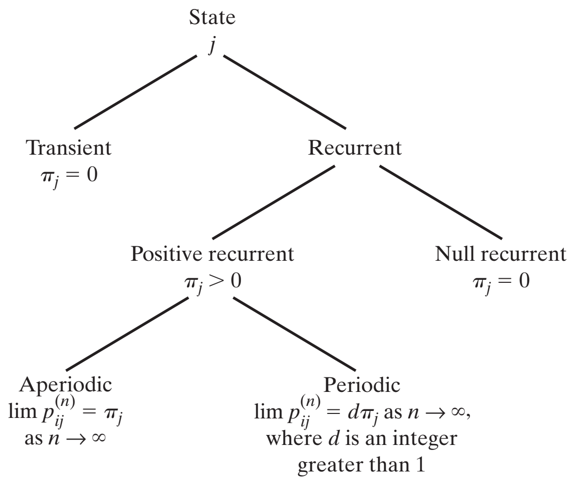

FIGURE 11.4 Classification of the states of a Markov chain and their associated long-term *behavior*.

For an illustrative application of this principle, we will use it to derive the relation of Eq. (11.27), which is the definition of a *stationary distribution*. To this end, we may manipulate the summation on the right-hand side of this equation as follows:

$$\begin{aligned}\sum_{i=1}^{K} \pi_i p_{ij} &= \sum_{i=1}^{K} \left(\frac{\pi_i}{\pi_j} p_{ij}\right)\pi_j \\ &= \sum_{i=1}^{K} (p_{ji})\pi_j \\ &= \pi_j\end{aligned}$$

In the second line of this expression, we made use of the principle of detailed balance, and in the last line we made use of the fact that the transition probabilities of a Markov chain satisfy the following condition (see Eq. (11.15), with the roles of i and j interchanged):

$$\sum_{i=1}^{K} p_{ji} = 1 \qquad \text{for all } j$$

From this discussion, it follows that the principle of detailed balance implies that the distribution $\{\pi_j\}$ is a stationary distribution. Insofar as a stationary distribution is concerned, the principle of detailed balance is therefore much stronger than Eq. (11.27), in the sense that it is sufficient, but not necessary, for the existence of a stationary distribution.

11.4 METROPOLIS ALGORITHM

Now that we understand the composition of a Markov chain, we will use it to formulate a stochastic algorithm for simulating the evolution of a physical system to thermal equilibrium. The algorithm is called the *Metropolis algorithm* (Metropolis et al., 1953). It is a modified Monte Carlo method, introduced in the early days of scientific computation for the stochastic simulation of a collection of atoms in equilibrium at a given temperature.

Because it is a modified Monte Carlo method, the Metropolis algorithm is commonly referred to as a *Markov chain Monte Carlo (MCMC) method*. In this context, we may formally state the following definition (Robert and Casella, 1999):

> *A Markov Chain Monte Carlo method for the simulation of an unknown probability distribution is any method that produces an ergodic Markov chain whose own stationary distribution is the unknown distribution.*

The Metropolis algorithm fits this definition perfectly, and so does a generalization of it: the Metropolis–Hastings algorithm.[3]

Statistical Analysis of the Metropolis Algorithm

Suppose that the random variable X_n, representing an arbitrary Markov chain, is in state x_i at time n. We randomly generate a new state x_j, representing a realization of another random variable Y_n. It is assumed that the generation of this new state satisfies the symmetry condition

$$P(Y_n = x_j | X_n = x_i) = P(Y_n = x_i | X_n = x_j)$$

Let ΔE denote the energy difference resulting from the transition of the system from state $X_n = x_i$ to state $Y_n = x_j$. Given ΔE, we proceed as follows:

1. If the energy difference ΔE is negative, the transition leads to a state with lower energy, and the transition is accepted. The new state is then accepted as the starting point for the next step of the algorithm; that is, we put $X_{n+1} = Y_n$.
2. If, on the other hand, the energy difference ΔE is positive, the algorithm proceeds in a probabilistic manner at that point. First, we select a random number ξ uniformly distributed in the range [0, 1]. If $\xi < \exp(-\Delta E/T)$, where T is the operating temperature, the transition is accepted and we put $X_{n+1} = Y_n$. Otherwise, the transition is rejected and we put $X_{n+1} = X_n$; that is, the old configuration is reused for the next step of the algorithm.

Choice of Transition Probabilities

Let an arbitrary Markov chain have a *proposed set of transition probabilities* denoted by τ_{ij}, which satisfy three conditions:

1. *Nonnegativity:*
$$\tau_{ij} \geq 0 \qquad \text{for all } i, j$$
2. *Normalization:*
$$\sum_j \tau_{ij} = 1 \qquad \text{for all } i$$
3. *Symmetry:*
$$\tau_{ji} = \tau_{ji} \qquad \text{for all } i, j$$

Let π_i denote the steady-state probability that the Markov chain is in state x_i, $i = 1, 2, \ldots, K$. We may then use the symmetric τ_{ij} and the probability distribution ratio π_j/π_i, to be defined, to formulate the *desired set of transition probabilities* as follows (Beckerman, 1997):

$$p_{ij} = \begin{cases} \tau_{ij}\left(\dfrac{\pi_j}{\pi_i}\right) & \text{for } \dfrac{\pi_j}{\pi_i} < 1 \\ \tau_{ij} & \text{for } \dfrac{\pi_j}{\pi_i} \geq 1 \end{cases} \tag{11.29}$$

To ensure that the transition probabilities are normalized to unity, we introduce an additional definition for the probability of no transition, given as

$$\begin{aligned} p_{ii} &= \tau_{ii} + \sum_{j \neq 1} \tau_{ij}\left(1 - \frac{\pi_j}{\pi_i}\right) \\ &= 1 - \sum_{j \neq i} \alpha_{ij}\tau_{ij} \end{aligned} \tag{11.30}$$

where α_{ij} is the *moving probability* defined by

$$\alpha_{ij} = \min\left(1, \frac{\pi_j}{\pi_i}\right) \tag{11.31}$$

The only outstanding requirement is determining how to choose the ratio π_j/π_i. To cater to this requirement, we choose the probability distribution to which we want the Markov chain to converge to be a Gibbs distribution, as shown by

$$\pi_j = \frac{1}{Z}\exp\left(-\frac{E_j}{T}\right)$$

in which case the probability distribution ratio π_j/π_i takes the simple form

$$\frac{\pi_j}{\pi_i} = \exp\left(-\frac{\Delta E}{T}\right) \tag{11.32}$$

where

$$\Delta E = E_j - E_i \tag{11.33}$$

By using the ratio of probability distributions, we have eliminated dependence on the partition function Z.

By construction, the transition probabilities are all nonnegative and normalized to unity, as required by Eqs. (11.14) and (11.15). Moreover, they satisfy the principle of detailed balance defined by Eq. (11.28). This principle is a sufficient condition for thermal equilibrium. To demonstrate that the principle of detailed balance is indeed satisfied, we offer the following considerations:

Case 1: $\Delta E < 0$. Suppose that in going from state x_i to state x_j, the energy change ΔE is negative. From Eq. (11.32), we find that $(\pi_j/\pi_i) > 1$, so the use of Eq. (11.29) yields

$$\pi_i p_{ij} = \pi_i \tau_{ij} = \pi_i \tau_{ji}$$

and

$$\pi_j p_{ji} = \pi_j\left(\frac{\pi_i}{\pi_j}\tau_{ji}\right) = \pi_i \tau_{ji}$$

Hence, the principle of detailed balance is satisfied for $\Delta E < 0$.

Case 2: $\Delta E > 0$. Suppose next that the energy change ΔE in going from state x_i to state x_j is positive. In this case, we find that $(\pi_j/\pi_i) < 1$, and the use of Eq. (11.29) yields

$$\pi_i p_{ij} = \pi_i\left(\frac{\pi_j}{\pi_i}\tau_{ij}\right) = \pi_j \tau_{ij} = \pi_j \tau_{ji}$$

and

$$\pi_j p_{ji} = \pi_i p_{ij}$$

Here again, we see that the principle of detailed balance is satisfied.

To complete the picture, we need to clarify the use of the proposed set of transition probabilities denoted by τ_{ij}. These transition probabilities are in fact the probabilistic model of the random step in the Metropolis algorithm. From the description of the algorithm presented earlier, we recall that the random step is followed by a random decision. We may therefore conclude that the transition probabilities p_{ij} defined in Eqs. (11.29) and (11.30) in terms of the proposed transition probabilities τ_{ij} and the steady-state probabilities π_j are indeed the correct choice for the Metropolis algorithm.

We therefore conclude that the Metropolis algorithm generates a Markov chain,[4] the transition probabilities of which do indeed converge to a unique and stable Gibbs distribution (Beckerman, 1997).

11.5 SIMULATED ANNEALING

Consider next the problem of finding a low-energy system whose states are ordered in a Markov chain. From Eq. (11.11), we observe that as the temperature T approaches zero, the free energy F of the system approaches the average energy $<E>$. With $F \rightarrow <E>$, we observe from the principle of minimal free energy that the Gibbs distribution, which is the stationary distribution of the Markov chain, collapses on the global minima of the average energy $<E>$ as $T \rightarrow 0$. In other words, low-energy ordered states are strongly favored at low temperatures. These observations prompt us to raise the question, Why not simply apply the Metropolis algorithm for generating a population of configurations representative of the stochastic system at very low temperatures? We do not advocate the use of such a strategy because the rate of convergence of the Markov chain to thermal equilibrium is extremely slow at very low temperatures. Rather, the preferred method for improved computational efficiency is to operate the stochastic system at a high temperature where convergence to equilibrium is fast, and then maintain the system at equilibrium as the temperature is carefully lowered. That is, we use a combination of two related ingredients:

1. a schedule that determines the rate at which the temperature is lowered;
2. an algorithm—exemplified by the Metropolis algorithm—that iteratively finds the equilibrium distribution at each new temperature in the schedule by using the final state of the system at the previous temperature as the starting point for the new temperature.

The twofold scheme that we have just described is the essence of a widely used stochastic relaxation technique known as *simulated annealing*[5] (Kirkpatrick et al., 1983). The technique derives its name from analogy with an annealing process in physics and chemistry in which we start the process at high temperature and then lower the temperature slowly while maintaining thermal equilibrium.

The primary objective of simulated annealing is to find the global minimum of a cost function that characterizes large and complex systems. As such, this technique provides a powerful tool for solving nonconvex optimization problems, motivated by the following simple idea:

When optimizing a very large and complex system (i.e., a system with many degrees of freedom), instead of always going downhill, try to go downhill most of the time.

Simulated annealing differs from conventional iterative optimization algorithms in two important respects:

1. The algorithm need not get stuck, since transition out of a local minimum is always possible when the system operates at a nonzero temperature.
2. Simulated annealing is *adaptive* in that gross features of the final state of the system are seen at higher temperatures, while fine details of the state appear at lower temperatures.

Annealing Schedule

As already mentioned, the Metropolis algorithm is the basis for the simulated-annealing process, in the course of which the temperature T is decreased slowly. That is, the temperature T plays the role of a *control parameter*. The simulated-annealing process will converge to a configuration of minimal energy provided that the temperature is decreased no faster than logarithmically. Unfortunately, such an annealing schedule is extremely slow—too slow to be of practical use. In practice, we must resort to a *finite-time approximation* of the asymptotic convergence of the algorithm. The price paid for the approximation is that the algorithm is no longer guaranteed to find a global minimum with a probability of one. Nevertheless, the resulting approximate form of the algorithm is capable of producing near-optimal solutions for many practical applications.

To implement a finite-time approximation of the simulated-annealing algorithm, we must specify a set of parameters governing the convergence of the algorithm. These parameters are combined in a so-called *annealing schedule*, or *cooling schedule*. The annealing schedule specifies a finite sequence of values of the temperature and a finite number of transitions attempted at each value of the temperature. The annealing schedule due to Kirkpatrick et al. (1983) specifies the parameters of interest as follows[6]:

1. *Initial Value of the Temperature.* The initial value T_0 of the temperature is chosen high enough to ensure that virtually all proposed transitions are accepted by the simulated-annealing algorithm
2. *Decrement of the Temperature.* Ordinarily, the cooling is performed *exponentially*, and the changes made in the value of the temperature are small. In particular, the *decrement function* is defined by

$$T_k = \alpha T_{k-1}, \qquad k = 1, 2, \ldots \tag{11.34}$$

   where α is a constant smaller than, but close to, unity. Typical values of α lie between 0.8 and 0.99. At each temperature, enough transitions are attempted so that there are 10 *accepted* transitions per experiment, on average.
3. *Final Value of the Temperature.* The system is fixed and annealing stops if the desired number of acceptances is not achieved at three successive temperatures.

The latter criterion may be refined by requiring that the *acceptance ratio*, defined as the number of accepted transitions divided by the number of proposed transitions, is smaller than a prescribed value (Johnson et al., 1989).

TABLE 11.1 Correspondence between Statistical Physics and Combinatorial Optimization

Statistical physics	Combinatorial optimization
Sample	Problem instance
State (configuration)	Configuration
Energy	Cost function
Temperature	Control parameter
Ground-state energy	Minimal cost
Ground-state configuration	Optimal configuration

Simulated Annealing for Combinatorial Optimization

Simulated annealing is particularly well suited for solving combinatorial-optimization problems. The objective of *combinatorial optimization* is to minimize the cost function of a finite, discrete system characterized by a large number of possible solutions. Essentially, simulated annealing uses the Metropolis algorithm to generate a sequence of solutions by invoking an analogy between a physical many-particle system and a combinatorial-optimization problem.

In simulated annealing, we interpret the energy E_i in the Gibbs distribution of Eq. (11.5) as a numerical cost and the temperature T as a control parameter. The numerical cost assigns to each configuration in the combinatorial-optimization problem a scalar value that describes how desirable that particular configuration is to the solution. The next issue to be considered in the simulated-annealing procedure is how to identify configurations and generate new configurations from previous ones in a local manner. This is where the Metropolis algorithm performs its role. We may thus summarize the correspondence between the terminology of statistical physics and that of combinatorial optimization as shown in Table 11.1 (Beckerman, 1997).

11.6 GIBBS SAMPLING

Like the Metropolis algorithm, the *Gibbs sampler*[7] generates a Markov chain with the Gibbs distribution as the equilibrium distribution. However, the transition probabilities associated with the Gibbs sampler are nonstationary (Geman and Geman, 1984). In the final analysis, the choice between the Gibbs sampler and the Metropolis algorithm is based on technical details of the problem at hand.

To proceed with a description of this sampling scheme, consider a K-dimensional random vector $\mathbf{X}$ made up of the components $X_1, X_2, ..., X_K$. Suppose that we have knowledge of the conditional distribution of X_k, given values of all the other components of $\mathbf{X}$ for $k = 1, 2, ..., K$. The problem we wish to address is how to obtain a numerical estimate of the marginal density of the random variable X_k for each k. The Gibbs sampler proceeds by generating a value for the conditional distribution for each component of the random vector $\mathbf{X}$, given the values of all other components of $\mathbf{X}$. Specifically, starting from an arbitrary configuration $\{x_1(0), x_2(0), ..., x_K(0)\}$, we make the following drawings on the first iteration of Gibbs sampling:

$x_1(1)$ is drawn from the distribution of X_1, given $x_2(0), x_3(0), ..., x_K(0)$.

$x_2(1)$ is drawn from the distribution of X_2, given $x_1(1), x_3(0), ..., x_K(0)$.

$\vdots$

$x_k(1)$ is drawn from the distribution of X_k, given $x_1(1), ..., x_{k-1}(1), x_{k+1}(0), ..., x_K(0)$.

$\vdots$

$x_K(1)$ is drawn from the distribution of X_K, given $x_1(1), x_2(1), ..., x_{K-1}(1)$.

We proceed in this same manner on the second iteration and every other iteration of the sampling scheme. The following two points should be carefully noted:

1. Each component of the random vector $\mathbf{X}$ is "visited" in the natural order, with the result that a total of K new variates are generated on each iteration.
2. The new value of component X_{k-1} is used immediately when a new value of X_k is drawn for $k = 2, 3, \ldots, K$.

From this discussion, we see that the Gibbs sampler is an *iterative adaptive* scheme. After n iterations of its use, we arrive at the K variates $X_1(n), X_2(n), \ldots, X_K(n)$. Under mild conditions, the following three theorems hold for Gibbs sampling (Geman and Geman, 1984; Gelfand and Smith, 1990):

1. ***Convergence theorem.*** *The random variable $X_k(n)$ converges in distribution to the true probability distributions of X_k* for $k = 1, 2, ..., K$ *as n approaches infinity*; that is,

$$\lim_{n\to\infty} P(X_k^{(n)} \le x | x_k(0)) = P_{X_k}(x) \qquad \text{for } k = 1, 2, ..., K \tag{11.35}$$

where $P_{X_k}(x)$ is marginal cumulative distribution function of X_k.

In fact, a stronger result is proven in Geman and Geman (1984). Specifically, rather than requiring that each component of the random vector $\mathbf{X}$ be visited in repetitions of the natural order, convergence of Gibbs sampling still holds under an arbitrary visiting scheme provided that this scheme does not depend on the values of the variables and that each component of $\mathbf{X}$ is visited on an "infinitely often" basis.

2. ***Rate-of-convergence theorem.*** *The joint cumulative distribution of the random variables $X_1(n), X_2(n), \ldots, X_K(n)$ converges to the true joint cumulative distribution of $X_1, X_2, ..., X_K$ at a geometric rate in n.*

This theorem assumes that the components of $\mathbf{X}$ are visited in the natural order. When, however, an arbitrary, but "infinitely often," visiting approach is used, then a minor adjustment to the rate of convergence is required.

3. ***Ergodic theorem.*** *For any measurable function g of the random variables $X_1, X_2, ..., X_K$ whose expectation exists, we have*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} g(X_1(i), X_2(i), ..., X_K(i)) \to \mathbb{E}[g(X_1, X_2, ..., X_K)] \tag{11.36}$$

with probability 1 *(i.e., almost surely).*

The ergodic theorem tells us how to use the output of the Gibbs sampler to obtain numerical estimations of the desired marginal densities.

Gibbs sampling is used in the Boltzmann machine to sample from distributions over hidden neurons; this stochastic machine is discussed in the next section. In the context of a stochastic machine using binary units (e.g., the Boltzmann machine), it is noteworthy that the Gibbs sampler is exactly the same as a variant of the Metropolis algorithm. In the standard form of the Metropolis algorithm, we go downhill with a probability of unity. In contrast, in the alternative form of the Metropolis algorithm, we go downhill with a probability equal to unity minus the exponential of the energy gap (i.e., the complement of the uphill rule). In other words, if a change lowers the energy E or leaves it unchanged, that change is accepted; if the change increases the energy, it is accepted with probability $\exp(-\Delta E)$ and is rejected otherwise, with the old state then being repeated (Neal, 1993).

11.7 BOLTZMANN MACHINE

The *Boltzmann machine* is a stochastic binary machine whose composition consists of stochastic neurons. A *stochastic neuron* resides in one of two possible states in a probabilistic manner. These two states may be designated as $+1$ for the "on" state and -1 for the "off" state or as 1 and 0, respectively. We will adopt the former designation. Another distinguishing feature of the Boltzmann machine is the use of *symmetric synaptic connections* between its neurons. The use of this form of synaptic connections is also motivated by statistical physics considerations.

The stochastic neurons of the Boltzmann machine are partitioned into two functional groups, *visible* and *hidden*, as depicted in Fig. 11.5. The visible neurons[8] provide an interface between the network and the environment in which it operates. During the training phase of the network, the visible neurons are all *clamped* onto specific states determined by the environment. The hidden neurons, on the other hand, always operate freely; they are used to explain underlying constraints contained in the environmental input vectors. The hidden neurons accomplish this task by capturing higher-order

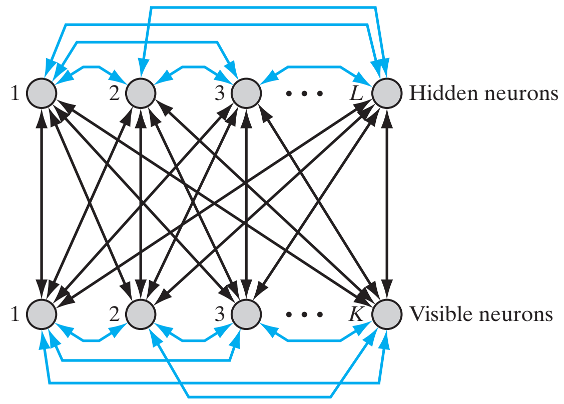

FIGURE 11.5 Architectural graph of Botzmann machine; K is the number of visible neurons, and L is the number of hidden neurons. The distinguishing features of the machine are:
1. The connections between the visible and hidden neurons are symmetric.
2. The symmetric connections are extended to the visible and hidden neurons.

statistical correlations in the clamping vectors. The network described here represents a special case of the Boltzmann machine. It may be viewed as an unsupervised-learning procedure for modeling a probability distribution that is specified by clamping patterns onto the visible neurons with appropriate probabilities. By so doing, the network can perform *pattern completion.* Specifically, when a partial information-bearing vector is clamped onto a subset of the visible neurons, the network performs completion on the remaining visible neurons, provided that it has learned the training distribution properly.

The primary goal of Boltzmann learning is to produce a neural network that correctly models input patterns according to the Boltzmann distribution. In applying this form of learning, two assumptions are made:

1. Each environmental input vector (pattern) persists long enough to permit the network to reach *thermal equilibrium.*
2. There is *no* structure in the sequential order in which the environmental vectors are clamped onto the visible units of the network.

A particular set of synaptic weights is said to constitute a perfect model of the environmental structure if it leads to exactly the same probability distribution of the states of the visible units (when the network is running freely) as when these units are clamped by the environmental input vectors. In general, unless the number of hidden units is exponentially large compared with the number of visible units, it is impossible to achieve such a perfect model. If, however, the environment has a regular structure, and the network uses its hidden units to capture these regularities, it may achieve a good match to the environment with a manageable number of hidden units.

Gibbs Sampling and Simulated Annealing for the Boltzmann Machine

Let $\mathbf{x}$ denote the state vector of the Boltzmann machine, with its component x_i denoting the state of neuron i. The state $\mathbf{x}$ represents a realization of a random vector $\mathbf{X}$. The synaptic connection from neuron i to neuron j is denoted by w_{ji}, with

$$w_{ji} = w_{ij} \qquad \text{for all } i, j \tag{11.37}$$

and

$$w_{ii} = 0 \qquad \text{for all } i \tag{11.38}$$

Equation (11.37) describes symmetry, and Eq. (11.38) emphasizes the absence of self-feedback. The use of a bias is permitted by using the weight w_{j0} from a fictitious node maintained at $+1$ and by connecting it to neuron j for all j.

From an analogy with thermodynamics, the energy of the Boltzmann machine is defined as follows[9]:

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{i}\sum_{\substack{j \\ i \neq j}} w_{ji}x_i x_j \tag{11.39}$$

Invoking the Gibbs distribution of Eq. (11.5), we may define the probability that the network (assumed to be in equilibrium at temperature T) is in state $\mathbf{x}$ as

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z}\exp\left(-\frac{E(\mathbf{x})}{T}\right) \tag{11.40}$$

where Z is the partition function.

To simplify the presentation, define the single event A and joint events B and C as follows:

A: $X_j = x_j$

B: $\{X_i = x_i\}_{i=1}^{K}$ with $i \neq j$

C: $\{X_i = x_i\}_{i=1}^{K}$

In effect, the joint event B excludes A, and the joint event C includes both A and B. The probability of B is the marginal probability of C with respect to A. Hence, using Eqs. (11.39) and (11.40), we may write

$$\begin{aligned} P(C) &= P(A, B) \\ &= \frac{1}{Z}\exp\left(\frac{1}{2T}\sum_{i}\sum_{\substack{j \\ i\neq j}} w_{ji}x_i x_j\right) \end{aligned} \tag{11.41}$$

and

$$\begin{aligned} P(B) &= \sum_{A} P(A, B) \\ &= \frac{1}{Z}\sum_{x_j}\exp\left(\frac{1}{2T}\sum_{i}\sum_{\substack{j \\ i\neq j}} w_{ji}x_i x_j\right) \end{aligned} \tag{11.42}$$

The exponent in Eqs. (11.41) and (11.42) may be expressed as the sum of two components, one involving x_j and the other being independent of x_j. The component involving x_j is given by

$$\frac{x_j}{2T}\sum_{\substack{i \\ i\neq j}} w_{ji}x_i$$

Accordingly, by setting $x_j = x_i = \pm 1$, we may express the conditional probability of A given B, as follows:

$$\begin{aligned} P(A|B) &= \frac{P(A, B)}{P(B)} \\ &= \frac{1}{1 + \exp\left(-\frac{x_j}{T}\sum_{\substack{i \\ i\neq j}} w_{ji}x_i\right)} \end{aligned}$$

That is, we may write

$$P(X_j = x|\{X_i = x_i\}_{i=1, i \neq j}^{K}) = \varphi\left(\frac{x}{T}\sum_{\substack{i \\ i \neq j}}^{K} w_{ji}x_i\right) \tag{11.43}$$

where $\varphi(\cdot)$ is the logistic function

$$\varphi(v) = \frac{1}{1 + \exp(-v)} \tag{11.44}$$

Note that although x varies between -1 and $+1$, the whole argument $v = \frac{x}{T}\sum_{i \neq j} w_{ji}x_i$ for large K may vary between $-\infty$ and $+\infty$, as depicted in Fig. 11.6. Note also that in deriving Eq. (11.43), the need for the partition function Z has been eliminated. This condition is highly desirable, since a direct computation of Z is infeasible for a network of large complexity.

The use of Gibbs sampling exhibits the joint distribution $P(A, B)$. Basically, as explained in Section 11.6, this stochastic simulation starts with the network assigned an arbitrary state, and the neurons are all repeatedly visited in their natural order. On each visit, a new value for the state of each neuron is chosen in accordance with the probability distribution for that neuron, conditional on the values for the states of all other neurons in the network. Provided that the stochastic simulation is performed long enough, the network will reach thermal equilibrium at temperature T.

Unfortunately, the time taken to reach thermal equilibrium can be much too long. To overcome this difficulty, simulated annealing for a finite sequence of temperatures $T_0, T_1, ..., T_{\text{final}}$ is used, as explained in Section 11.5. Specifically, the temperature is initially set to the high value T_0, thereby permitting thermal equilibrium to be reached fast. Thereafter, the temperature T is gradually reduced to the final value T_{final}, at which point the neural states will have (hopefully) reached their desired marginal distributions.

Boltzmann Learning Rule

Since the Boltzmann machine is a stochastic machine, it is natural to look to probability theory for an appropriate index of performance. One such criterion is the *likelihood*

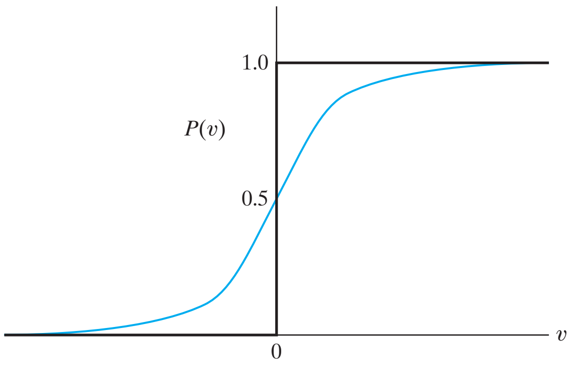

FIGURE 11.6 Sigmoid-shaped function $P(v)$.

function.[10] On this basis, the goal of Boltzmann learning is to maximize the likelihood function—or, equivalently, the log-likelihood function—in accordance with the *maximum-likelihood principle*; this principle was described in Chapter 10.

Let $\mathcal{T}$ denote a training sample drawn from the probability distribution of interest. It is assumed that the examples are all two-valued. Repetition of training examples is permitted in proportion to how common certain cases are known to occur. Let a subset of the state vector $\mathbf{x}$—say, $\mathbf{x}_\alpha$—denote the state of the visible neurons. The remaining part of the state vector $\mathbf{x}$—say, $\mathbf{x}_\beta$—represents the state of the hidden neurons. The state vectors $\mathbf{x}$, $\mathbf{x}_\alpha$, and $\mathbf{x}_\beta$ are realizations of the random vectors $\mathbf{X}$, $\mathbf{X}_\alpha$, and $\mathbf{X}_\beta$, respectively. There are two phases to the operation of the Boltzmann machine:

1. *Positive phase.* In this phase, the network operates in its clamped condition (i.e., under the direct influence of the training sample $\mathcal{T}$).
2. *Negative phase.* In this second phase, the network is allowed to run freely, and therefore with no environmental input.

Given the synaptic-weight vector $\mathbf{w}$ for the whole network, the probability that the visible neurons are in state $\mathbf{x}_\alpha$ is $P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$. With the many possible values of $\mathbf{x}_\alpha$ contained in the training sample $\mathcal{T}$ assumed to be statistically independent, the overall probability distribution is the factorial distribution $\prod_{\mathbf{x}_\alpha \in \mathcal{T}} P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$. To formulate the log-likelihood function $L(\mathbf{w})$, we take the logarithm of this factorial distribution and treat $\mathbf{w}$ as the unknown parameter vector. We may thus write

$$\begin{aligned} L(\mathbf{w}) &= \log \prod_{\mathbf{x}_\alpha \in \mathcal{T}} P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) \\ &= \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) \end{aligned} \tag{11.45}$$

To formulate the expression for the marginal probability $P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$ in terms of the energy function $E(\mathbf{x})$, we follow two points:

1. The probability $P(\mathbf{X} = \mathbf{x})$ is equal to $\frac{1}{Z}\exp(-E(\mathbf{x})/T)$, in accordance with Eq. (11.40).
2. By definition, the state vector $\mathbf{x}$ is the joint combination of $\mathbf{x}_\alpha$ pertaining to the visible neurons and $\mathbf{x}_\beta$ pertaining to the hidden neurons. Hence, the probability of finding the visible neurons in state $\mathbf{x}_\alpha$, for any $\mathbf{x}_\beta$, is given by

$$P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \frac{1}{Z}\sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \tag{11.46}$$

where the random vector $\mathbf{X}_\alpha$ is a subset of $\mathbf{X}$. From Eq. (11.6), we find that the partition function Z is itself defined by

$$Z = \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \tag{11.47}$$

Thus, substituting Eqs. (11.46) and (11.47) into Eqs. (11.45), we obtain the desired expression for the log-likelihood function:

$$L(\mathbf{w}) = \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \left(\log \sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) - \log \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right)\right) \tag{11.48}$$

The dependence on **w** is contained in the energy function $E(\mathbf{x})$, as shown in Eq. (11.39).

Differentiating $L(\mathbf{w})$ with respect to w_{ji} and using Eq. (11.39), we obtain the following result after some manipulation of terms (see Problem 11.9):

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{1}{T} \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \left(\sum_{\mathbf{x}_\beta} P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) x_j x_i - \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) x_j x_i \right) \tag{11.49}$$

To simplify matters, we introduce two definitions:

1.
$$\begin{aligned} \rho_{ji}^{+} &= <x_j x_i>^{+} \\ &= \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \sum_{\mathbf{x}_\beta} P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) x_j x_i \end{aligned} \tag{11.50}$$

2.
$$\begin{aligned} \rho_{ji}^{-} &= <x_j x_i>^{-} \\ &= \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) x_j x_i \end{aligned} \tag{11.51}$$

In a loose sense, we may view the first average, ρ_{ji}^{+}, as the mean firing rate, or *correlation*, between the states of neurons i and j when the network is operating in its clamped, or positive, phase. Similarly, we may view the second average, ρ_{ji}^{-}, as the *correlation* between the states of neurons i and j when the network is operating in its free-running, or negative, phase. With these definitions, we may simplify Eq. (11.49) to

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{1}{T}(\rho_{ji}^{+} - \rho_{ji}^{-}) \tag{11.52}$$

The goal of Boltzmann learning is to maximize the log-likelihood function $L(\mathbf{w})$. We may use *gradient ascent* to achieve that goal by writing

$$\begin{aligned} \Delta w_{ji} &= \epsilon \frac{\partial L(\mathbf{w})}{\partial w_{ji}} \\ &= \eta(\rho_{ji}^{+} - \rho_{ji}^{-}) \end{aligned} \tag{11.53}$$

where η is a *learning-rate parameter*; it is defined in terms of ϵ, the constant introduced in the first line of Eq. (11.53), and the operating temperature T as

$$\eta = \frac{\epsilon}{T} \tag{11.54}$$

The gradient ascent rule of Eq. (11.53) is called the *Boltzmann learning rule.* The learning described here is performed in batch; that is, changes to the synaptic weights are made on the presentation of the entire training sample.

Summarizing Remarks

The simplicity of the Boltzmann learning rule described in Eq. (11.53) is attributed to the fact that only *locally available observations* are used under two different operating conditions of neurons, one being clamped and the other free running. Another interesting feature of the rule, which may come as a surprise, is that adjustment of the synaptic weight w_{ji} from neuron i to neuron j is independent of whether the two neurons are both visible, both hidden, whether they are both free running, or whether they are one of each. These desirable features of the Boltzmann machine stem from a key insight by Hinton and Sejnowski (1983, 1986) that ties the machine's abstract mathematical model to neural networks by combining two concepts:

- the Gibbs distribution for describing the stochasticity of neurons, and
- the statistical physics-based energy function of Eq. (11.39) for defining the Gibbs distribution.

However, from a practical perspective, we typically find that the learning process in the Boltzmann machine is very slow, particularly when the number of hidden neurons used in the machine is large. The reason for this undesirable behavior is that the machine takes a long time to reach an equilibrium distribution, which usually happens when the visible units are unclamped.

Nevertheless, over the years, interest has continued in the search for a stochastic machine that would share with the classical Boltzmann machine the capacity to learn probability distributions over binary vectors, but would also be capable of performing the following two functions:

1. Ignore the negative phase of the Boltzmann machine that is responsible for increased computation time, and find some other means for exercising control over the learning process.
2. Operate efficiently in densely connected networks.

In the next two sections, we describe a pair of stochastic machines that have tried to address these two practical issues in their own individual ways.

11.8 LOGISTIC BELIEF NETS

In the first generation of *logistic belief networks*, devised by Neal (1992), the symmetric connections of the Boltzmann machine are replaced with *direct* connections, forming an *acyclic graph*. It is for this reason that Neal's logistic belief net is also referred to as a *directed belief net*; hereafter, both terminologies will be used interchangeably. To be more specific, the network is of a multilayer kind, as illustrated in Fig. 11.7. The acyclic property of the network makes it easy to perform probabilistic calculations. In analogy with the classical Boltzmann machine, the network uses the logistic function of Eq. (11.43) to calculate the conditional probability of a neuron being active in response to its induced local field.

Let the random vector **X**, consisting of the random variables $X_1, X_2, ..., X_n$, denote the behavior of a logistic belief network comprising N stochastic neurons. The *parents* of element X_j (i.e., the parents of node j in Fig. 11.7) in the vector **X** are defined by

$$pa(X_j) \subseteq \{X_1, X_2, ..., X_{j-1}\} \tag{11.55}$$

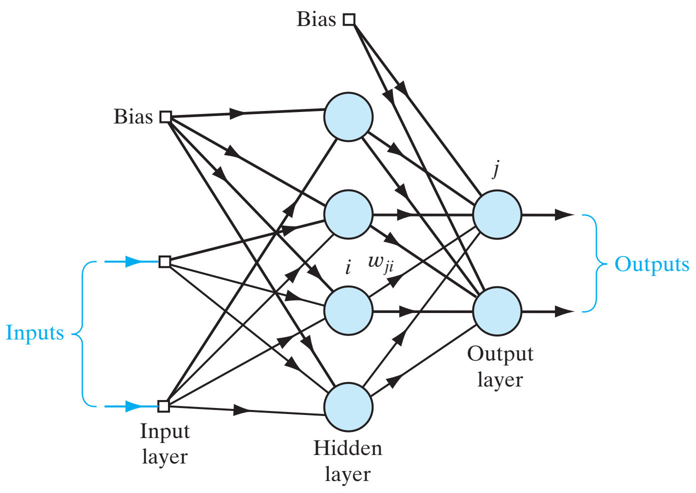

FIGURE 11.7 Directed (logistic) belief network.

where $\{x_1, x_2, \ldots, x_j\}$ is the smallest subset of the random vector $\mathbf{X}$ that excites nodes; and for which the conditional probability

$$P(X_j = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) = P(X_j = x_j | pa(X_j)) \qquad (11.56)$$

Referring to Fig. 11.7, for example, node i is a parent of node j as there is a direct link from node i to node j. An important virtue of the logistic belief network is its ability to clearly exhibit the conditional dependencies of the underlying probabilistic model of the input data, with the probability that the jth neuron is active being defined by the logistic function where w_{ji} is the synaptic weight from neuron i to neuron j. This conditional probability depends on $pa(X_j)$ solely through a sum of weighted inputs. Thus, Eq. (11.56) provides the basis for the propagation of beliefs through the network.

Calculations of conditional probabilities are performed under two different *null* conditions:

1. $w_{ji} = 0$ for all X_i not belonging to $pa(X_j)$, which follows from the definition of a parent.
2. $w_{ji} = 0$ for $i \geq j$, which follows from the fact that the network is acyclic.

As with the Boltzmann machine, the learning rule of the logistic belief net is derived by maximizing the log-likelihood function $L(\mathbf{w})$ of Eq. (11.45), computed for the training sample $\mathcal{T}$. It turns out that this maximization is accomplished through the use of gradient ascent in probability space by defining the change in the synaptic weight w_{ji} as

$$\Delta w_{ji} = \eta \frac{\partial}{\partial w_{ji}} L(\mathbf{w})$$

where η is the learning-rate parameter, and the weight vector $\mathbf{w}$ is for the whole net.

However, a serious limitation of the logistic belief learning procedure is that when it is applied to densely connected networks, computation of the posterior distribution over the hidden neurons becomes computationally intractable, except in some simple applications such as linear models with additive Gaussian noise. As with the Boltzmann machine, Gibbs sampling can be used to approximate the posterior distribution, but the use of Gibbs sampling is considerably more complicated in a directed belief net.

11.9 DEEP BELIEF NETS

To overcome the difficulty in performing inference in logistic (directed) belief nets, Hinton et al. (2006) have devised a new way of learning logistic belief nets so that inference is easy to accomplish. The models that are learned in this new way are identical to those in logistic belief nets, except for the fact that they differ in their top layers, which (in the new way) form an undirected *associative memory*. Indeed, it is in virtue of this difference that the new belief networks are called "deep belief nets."

Deep belief nets build on a neural network structure that was first described in Smolensky (1986); at that time, this structure was referred to as a "harmonium." A distinctive feature of the harmonium is that there are no connections among the visible or hidden neurons; otherwise, it is similar to the classical Botzmann machine in that it uses symmetric connections between the visible neurons and hidden ones. On account of the difference cited, the harmonium has been renamed a *restricted Boltzmann machine* (RBM) in Hinton et al. (2006). At first sight, it may appear surprising to find that a symmetrically connected module such as a restricted Boltzmann machine could learn a directed generative model like a logistic belief net.

Since there are no connections between the hidden neurons in the RBM, and since the connections between the visible layer and the hidden layer are undirected, as depicted in Fig. 11.8, it follows that the states of the hidden neurons are *conditionally independent* of each other, given the visible states. Hence, the RBM is capable of extracting an unbiased sample from the posterior distribution, given a data vector clamped onto the visible neurons. This property of the RBM gives it a big advantage over the corresponding directed belief net (Hinton, 2007).

Another point of interest is that learning with an infinitely deep directed belief net, all of whose weight vectors are tied in the manner described in Fig. 11.9, is equivalent to learning with the single restricted Boltzmann machine shown in Fig. 11.8.

Maximum-Likelihood Learning in a Restricted Boltzmann Machine

The probability that a hidden neuron in the RBM is active is defined by the logistic function of Eq. (11.44). Let $\mathbf{x}_\alpha^{(0)}$ denote a data vector clamped onto the visible layer of the RBM at time 0. Then the learning process alternates back and forth between the following two operations:

- Updating the hidden states all in parallel, given the visible states
- Doing the same, but in reverse: updating the visible states all in parallel, given the hidden states

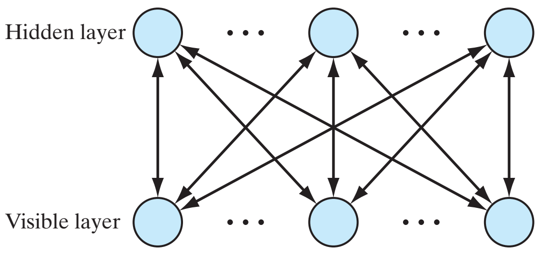

FIGURE 11.8 Neural structure of restricted Boltzmann machine (RBM). Contrasting this with that of Fig. 11.6, we see that unlike the Boltzmann machine, there are no connections among the visible neurons and the hidden neurons in the RBM.

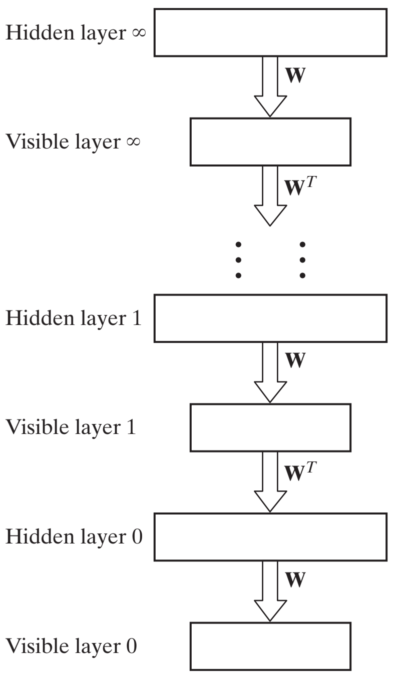

FIGURE 11.9 Top-down learning, using logistic belief network of infinite depth.

Let **w** denote the weight vector for the whole net. Accordingly, we find that the gradient of the log-likelihood function $L(\mathbf{w})$ with respect to the weight w_{ji}, symmetrically connecting the visible neuron i and the hidden neuron j, is given by

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \rho_{ji}^{(0)} - \rho_{ji}^{(\infty)} \tag{11.57}$$

where $\rho_{ji}^{(0)}$ and $\rho_{ji}^{(\infty)}$ are the *average correlations* between the states of neurons i and j at zero time and an infinitely long time, respectively (Hinton et al., 2006; Hinton, 2007). Except for insignificant changes in terminology, Eq. (11.57) is of the same mathematical form as that of Eq. (11.52) in classical Boltzmann learning. However, since we do not anneal in an RBM, Eq. (11.57) does not involve the use of temperature as a parameter.

Training of a Deep Belief Net

The training of a deep belief net progresses on a *layer-by-layer* basis, as follows (Hinton, et al., 2006; Hinton, 2007):

1. A restricted Boltzmann machine (RBM) is trained directly on the input data, thereby making it possible for the stochastic neurons in the hidden layer of the RBM to capture the important features that characterize the input data. Hereafter, we refer to this hidden layer as the *first hidden layer* of the deep belief net.
2. The activations of the trained features are then treated as "input data," which, in turn, are used to train a second RBM. In effect, the learning process just described

may be viewed as one of *learning features of features*. The origin of this idea may be traced to an early paper by Selfridge (1958), who proposed a pattern-recognition system called the "pandemonium."

3. The process of learning features of features is continued until a prescribed number of hidden layers in the deep belief net have been trained.

An important point to note here is the fact that every time a new layer of features is added to the deep belief net, a variational lower bound on the log-probability of the original training data is improved (Hinton et al., 2006).

Generative Model

Figure 11.10 pictures a deep belief net after three hidden layers of the net have been trained. The upward arrows indicate the weights computed as a result of learning the features of features. The function of these weights is to *infer* the binary feature values in each hidden layer of the belief net when a data vector is clamped into the neurons of the visible layer.

The *generative model* is identified by the unshaded arrows in Fig. 11.10. Note that the generative model does not include the bottom-up connections represented by the red shaded upward arrows; but most importantly, it does include the up-and-down connections in the top-level RBM (i.e., layers 2 and 3), which plays the dual role of a *bipartite associative memory*. When bottom-up training is performed, the top-level RBM learns from the hidden layer below. When top-down generation is performed, the top-level RBM is the initiator of generative modeling.

With the picture depicted in Fig. 11.10 in mind, data generation proceeds along the following lines:

1. An equilibrium sample is taken from the top-level RBM by the performance of alternating Gibbs sampling for many time-steps in the manner described in Fig. 11.11; this operation is permitted to go on for a sufficiently long time until equilibrium is reached.
2. A single top-down pass starting with the "visible" units of the top-level RBM is then used to stochastically pick the states of all the other hidden layers of the net.

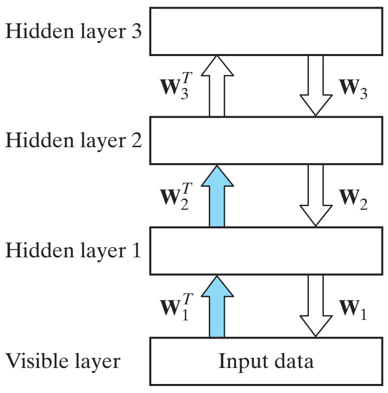

FIGURE 11.10 A hybrid generative model in which the two top layers form a restricted Boltzmann machine and the lower two layers form a directed model. The weights shown with red shaded arrows are *not* part of the generative model; they are used to infer the feature values given to the data, but they are not used for generating data.

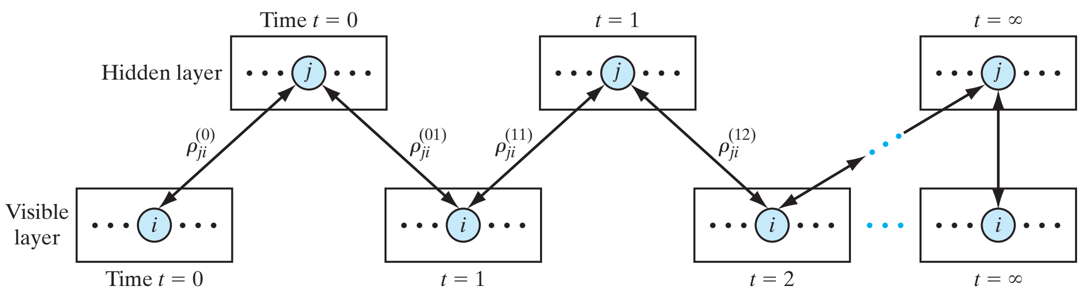

FIGURE 11.11 Illustrating the progression of alternating Gibbs sampling in an RBM. After sufficiently many steps, the visible and hidden vectors are sampled from the stationary distribution defined by the current parameters of the model.

Data generation is slow because, first of all, the top-level RBM must reach its equilibrium distribution. Fortunately, generation is not required for perceptual inference or learning.

Hybrid Learning Process

Each RBM in the deep belief net divides the task of modeling its own "visible" data into two subtasks, as portrayed in Fig. 11.12 (Hinton, 2007):

Subtask 1. The machine learns generative weights **w** that convert the aggregated posterior distribution over the hidden neurons into a close approximation to the data distribution over the visible neurons.

Subtask 2. The same set of weights, denoted by the vector **w**, also define a prior distribution over the hidden state vectors. Sampling for this prior distribution requires extensive alternating Gibbs sampling, as illustrated in Fig. 11.11. However, it is the very existence of this complicated prior that is responsible for making inference so simple in an RBM. When the next RBM is learned under subtask 2, that particular RBM replaces the complicated prior (defined by **w**) with a new prior that

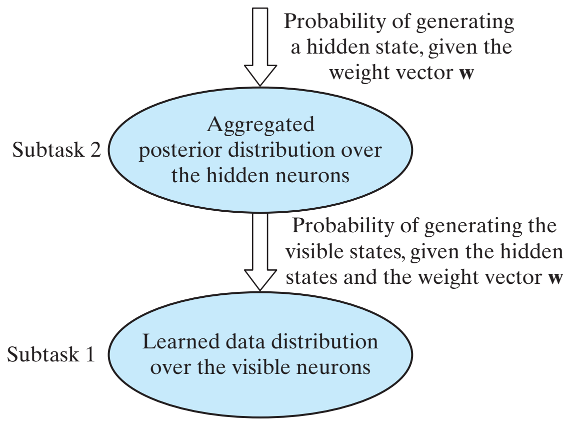
Probability of generating a hidden state, given the weight vector **w**
Subtask 2
Aggregated posterior distribution over the hidden neurons
Probability of generating the visible states, given the hidden states and the weight vector **w**
Subtask 1
Learned data distribution over the visible neurons

FIGURE 11.12 The task of modeling the sensory data is divided into two subtasks.

better approximates the aggregated posterior distribution over the hidden units of the lower-level RBM.

Concluding Remarks

1. Except for the top two layers, a deep belief net is a multilayer logistic belief network that has directed connections from one layer of the network to the next one down.
2. Learning proceeds bottom up on a layer-by-layer basis without any supervision. Because of the way the learning process is performed, perceptual inference is very simple in a deep belief net: Simply put, the inference process consists of a single bottom-up pass.
3. Deep belief nets provide the designer a great deal of freedom. The challenge for the designer is how to exploit this freedom in a creative way.

11.10 DETERMINISTIC ANNEALING

We now come to the final topic of the chapter, deterministic annealing. In Section 11.5, we discussed simulated annealing, which is a stochastic relaxation technique that provides a powerful method for solving nonconvex optimization problems. However, in the application of simulated annealing, care must be exercised in choosing the annealing schedule. In particular, a global minimum is achieved only if the temperature is decreased at a rate no faster than logarithmically. This requirement makes the use of simulated annealing impractical in many applications. Simulated annealing operates by making random moves on the energy surface (landscape). By contrast, in *deterministic annealing*, some form of randomness is incorporated into the energy or cost function itself, which is then deterministically optimized at a sequence of decreasing temperatures (Rose et al., 1990; Rose, 1998).

In what follows, we describe the idea of deterministic annealing in the context of an unsupervised-learning task: clustering.[11]

Clustering via Deterministic Annealing

The idea of clustering was discussed in Chapter 5. In light of the discussion presented therein, we say that clustering is the partitioning of a given set of data points into subgroups, each one of which is as similar or homogeneous as possible. Clustering is typically a nonconvex-optimization problem, since most distortion functions used in clustering are nonconvex functions of the input data. (The optimal manifold representation of data described in Chapter 10 is an exceptional case.) Moreover, a plot of the distortion function versus the input is riddled with local minima, making the task of finding the global minimum even more difficult.

In Rose (1991, 1998), a probabilistic framework is described for clustering by *randomization of the partition*, or equivalently, *randomization of the encoding rule.* The main principle used here is that each data point is *associated in probability* with a particular cluster (subgroup). To be specific, let the random vector $\mathbf{X}$ denote a *source (input) vector*, and let the random vector $\mathbf{Y}$ denote the best *reconstruction (output) vector* from a codebook of interest. Individual realizations of these two vectors are denoted by vectors $\mathbf{x}$ and $\mathbf{y}$, respectively.

To perform clustering, we need a distortion measure, denoted by $d(\mathbf{x}, \mathbf{y})$. It is assumed that $d(\mathbf{x}, \mathbf{y})$ satisfies two desirable properties:

1. It is a convex function of $\mathbf{y}$ for all $\mathbf{x}$.
2. It is finite whenever its two arguments $\mathbf{x}$ and $\mathbf{y}$ are both finite.

These two mild properties are satisfied, for example, by the squared Euclidean distortion measure:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 \tag{11.58}$$

which was also used in Chapters 5 and 10. The *expected distortion* for the randomized pattern is defined by

$$\begin{aligned} D &= \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) d(\mathbf{x}, \mathbf{y}) \\ &= \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) \sum_{\mathbf{y}} P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) d(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{11.59}$$

where $P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y})$ is the probability of the joint event $\mathbf{X} = \mathbf{x}$ and $\mathbf{Y} = \mathbf{y}$. In the second line of Eq. (11.59), we have used the formula for the probability of a joint event:

$$P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) P(\mathbf{X} = \mathbf{x}) \tag{11.60}$$

The conditional probability $P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x})$ is referred to as the *association probability*—that is, the probability of associating the code vector $\mathbf{y}$ with the source vector $\mathbf{x}$.

The expected distortion D is traditionally minimized with respect to the free parameters of the clustering model: the reconstruction vector $\mathbf{y}$ and the association probability $P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x})$. This form of minimization produces a "hard" clustering solution—hard in the sense that a source vector $\mathbf{x}$ is assigned to the nearest code vector $\mathbf{y}$. In deterministic annealing, on the other hand, the optimization problem is reformulated as that of seeking the probability distribution that minimizes the expected distortion, *subject to a specified level of randomness.* For a principled measure of the level of randomness, we use the joint entropy, defined in the Shannon sense, by the following (see Section 10.2):

$$H(\mathbf{X}, \mathbf{Y}) = -\sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) \log P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) \tag{11.61}$$

The constrained optimization of the expected distortion is then expressed as the minimization of the *Lagrangian*, defined by

$$F = D - TH \tag{11.62}$$

where T is treated as the Lagrange multiplier. From Eq. (11.62), we observe the following points:

- For large values of T, the joint entropy H is maximized.
- For small values of T, the expected distortion D is minimized, resulting in a hard (nonrandom) clustering solution.
- For intermediate values of T, the minimization of F provides a tradeoff between an increase in the entropy H and a reduction in the expected distortion D.

Most importantly, by comparing Eq. (11.62) with Eq. (11.11), we may identify the correspondence between the terms of the constrained-clustering optimization problem and the terms of statistical mechanics, as shown in Table 11.2. In light of this analogy, we henceforth refer to T as the temperature.

To develop further insight into the Lagrangian F, following the formula of Eq. (10.26), we decompose the joint entropy $H(\mathbf{X}, \mathbf{Y})$ into two terms:

$$H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y}|\mathbf{X})$$

$H(\mathbf{X})$ is the *source entropy*, and $H(\mathbf{Y}|\mathbf{X})$ is the *conditional entropy* of the reconstruction vector $\mathbf{Y}$ given the source vector $\mathbf{X}$. The source entropy $H(\mathbf{X})$ is independent of clustering. Accordingly, we may drop the source entropy $H(\mathbf{X})$ from the definition of the Lagrangian F and thereby focus on the conditional entropy

$$H(\mathbf{Y}|\mathbf{X}) = -\sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) \sum_{\mathbf{y}} P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x}) \log P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x}) \tag{11.63}$$

This expression highlights the role of the association probability $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$. Hence, keeping in mind the correspondence between the constrained-clustering optimization problem and statistical physics, and invoking the principle of minimal free energy described in Section 11.2, we find that minimizing the Lagrangian F with respect to the association probabilities results in the Gibbs distribution

$$P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(-\frac{d(\mathbf{x}, \mathbf{y})}{T}\right) \tag{11.64}$$

where $Z_{\mathbf{x}}$ is the partition function for the problem at hand. It is defined by

$$Z_{\mathbf{x}} = \sum_{\mathbf{y}} \exp\left(-\frac{d(\mathbf{x}, \mathbf{y})}{T}\right) \tag{11.65}$$

As the temperature T approaches infinity, we find from Eq. (11.64) that the association probability approaches a uniform distribution. The implication of this statement is that at very high temperatures, each input vector is equally associated with all clusters. Such associations may be viewed as "extremely fuzzy." At the other extreme, as the temperature T approaches zero, the association probability approaches a delta function. Accordingly, at very low temperatures, the classification is hard, with each input sample being assigned to the nearest code vector with probability 1.

TABLE 11.2 Correspondence between Constrained-Clustering Optimization and Statistical Physics

Constrained-clustering optimization	Statistical physics
Lagrangian, F	Free energy, F
Expected distortion, D	Average energy, $<E>$
Joint entropy in the Shannon sense, H	Entropy, H
Lagrange multiplier, T	Temperature, T

To find the minimum value of the Lagrangian F, we substitute the Gibbs distribution of Eq. (11.64) into Eqs. (11.59) and (11.63) and then use the resulting expressions in the formula for the Lagrangian F in Eq. (11.62). The result obtained is as follows (see Problem 11.16):

$$\begin{aligned} F^* &= \min_{P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})} F \\ &= -T \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) \log Z_{\mathbf{x}} \end{aligned} \tag{11.66}$$

To minimize the Lagrangian with respect to the remaining free parameters—namely, the code vectors $\mathbf{y}$—we set the gradients of F^* with respect to $\mathbf{y}$ to zero. Hence, we obtain the condition

$$\sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) \frac{\partial}{\partial \mathbf{y}} d(\mathbf{x}, \mathbf{y}) = 0 \qquad \text{for all } \mathbf{y} \in \mathcal{Y} \tag{11.67}$$

where $\mathcal{Y}$ is the set of all code vectors. Using the formula of Eq. (11.60) and normalizing with respect to $P(\mathbf{X} = \mathbf{x})$, we may redefine this minimizing condition as

$$\frac{1}{N} \sum_{\mathbf{x}} P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x}) \frac{\partial}{\partial \mathbf{y}} d(\mathbf{x}, \mathbf{y}) = 0 \qquad \text{for all } \mathbf{y} \in \mathcal{Y} \tag{11.68}$$

where the association probability $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ is itself defined by the Gibbs distribution of Eq. (11.64). In Eq. (11.68), we have included the scaling factor $1/N$ merely for completeness, where N is the number of available examples.

We may now describe the deterministic-annealing algorithm for clustering (Rose, 1998):

The deterministic-annealing algorithm consists of minimizing the Lagrangian F^* with respect to the code vectors at a high value of temperature T and then tracking the minimum while the temperature T is lowered.

In other words, deterministic annealing operates with a specific annealing schedule whereby the temperature is lowered in an orderly fashion. At each value of the temperature T, a two-step iteration central to the algorithm is performed:

1. The code vectors are fixed, and the Gibbs distribution of Eq. (11.64) for a specific distortion measure $d(\mathbf{x}, \mathbf{y})$ is used to calculate the association probabilities.
2. The associations are fixed, and Eq. (11.68) is used to optimize the distortion measure $d(\mathbf{x}, \mathbf{y})$ with respect to the code vector $\mathbf{y}$.

This two-step iterative procedure is monotonically nonincreasing in F^* and is therefore assured of converging to a minimum. At high values of temperature T, the Lagrangian F^* is fairly smooth and is a convex function of $\mathbf{y}$ under the mild assumptions previously made on the distortion measure $d(\mathbf{x}, \mathbf{y})$. A global minimum of F^* can be found at high temperatures. As the temperature T is lowered, the association probabilities become hard, resulting in a hard-clustering solution.

As the temperature T is lowered in the course of going through the annealing schedule, the system undergoes a sequence of phase transitions, which consists of natural cluster splits through which the clustering model grows in size (i.e., number of

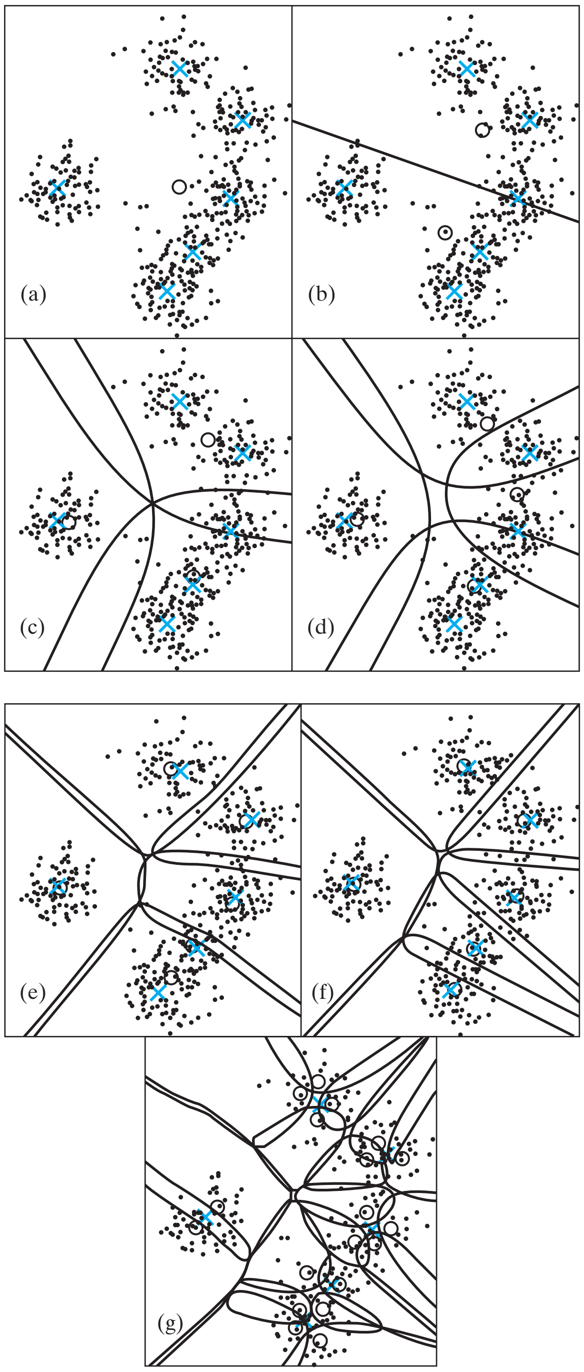

FIGURE 11.13 Clustering at various phases. The lines are equiprobability contours, $p = \frac{1}{2}$ in (b), and $p = \frac{1}{3}$ elsewhere:
(a) 1 cluster ($B = 0$),
(b) 2 clusters ($B = 0.0049$),
(c) 3 clusters ($B = 0.0056$),
(d) 4 clusters ($B = 0.0100$),
(e) 5 clusters ($B = 0.0156$),
(f) 6 clusters ($B = 0.0347$), and
(g) 19 clusters ($B = 0.0605$).

clusters) (Rose et al., 1990; Rose, 1991). This phenomenon is significant for the following reasons:

1. The sequence of phase transitions provides a useful tool for *controlling* the size of the clustering model.
2. As in ordinary physical annealing, the phase transitions are the *critical points* of the deterministic-annealing process during which care has to be exercised with the annealing.
3. The critical points are *computable*, thereby providing information that can be used to accelerate the algorithm in between phase transitions.
4. An *optimum model size* may be identified by coupling a validation procedure with the sequence of solutions produced at various phases, which represent solutions of increasing model size.

Case Study: Mixture of Gaussian Distributions

Figures 11.13 and 11.14 illustrate the evolution of the clustering solution via deterministic annealing at various phases as the temperature T is decreased or the reciprocal of temperature, $B = 1/T$, is increased (Rose, 1991). The data set used to generate these figures is a mixture of six Gaussian distributions whose centers are marked with an "X" in Fig. 11.13. The centers of the computed clusters are marked with an "o." Since the clustering solutions at nonzero temperatures are not hard, this random partition is depicted by contours of equal probability—for example, probability $\frac{1}{3}$ of belonging to a particular cluster. This process starts with one natural cluster containing the training sample (Fig. 11.13a). At the first phase transition, it splits into two clusters (Fig. 11.13b) and then passes

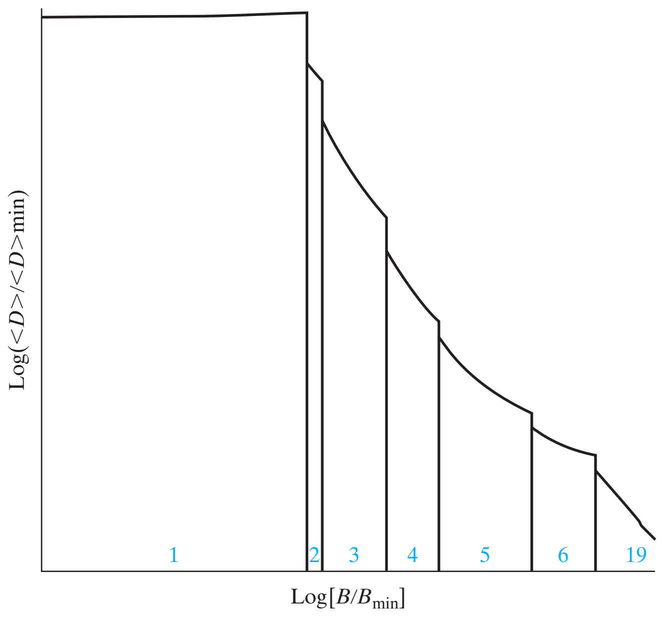

FIGURE 11.14 Phase diagram for the Case Study in deterministic annealing. The number of effective clusters is shown for each phase.

through a sequence of phase transitions until it reaches the "natural" set of six clusters. The next phase transition results in an "explosion" when all clusters split. Figure 11.14 shows the phase diagram, displaying the behavior of the average distortion throughout the annealing process and the number of natural clusters at each phase. In this figure, the average distortion (normalized with respect to its minimum value) is plotted versus the reciprocal of temperature, B, normalized with respect to its minimum value B_{min}. Both axes are labeled in their relative logarithmic forms.

11.11 ANALOGY OF DETERMINISTIC ANNEALING WITH THE EXPECTATION-MAXIMIZATION ALGORITHM

For another important aspect of the deterministic-annealing algorithm, suppose we view the association probability $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ as the expected value of a random binary variable $V_{\mathbf{xy}}$ defined as

$$V_{\mathbf{xy}} = \begin{cases} 1 & \text{if the source vector } \mathbf{x} \text{ is assigned to code vector } \mathbf{y} \\ 0 & \text{otherwise} \end{cases} \tag{11.69}$$

Then, from such a perspective, we recognize the two-step iteration of the deterministic-annealing algorithm to be a form of the *expectation-maximization (EM) algorithm.* In order to appreciate the relevance of this point, we will digress briefly to describe the underlying theory of the EM algorithm.

The EM Algorithm[12]

Let the vector $\mathbf{z}$ denote some missing or unobservable data. Let $\mathbf{r}$ denote the complete-data vector, made up of some observable data point d and the missing data vector $\mathbf{z}$. There are therefore two data spaces, $\mathcal{R}$ and $\mathcal{D}$, to be considered, with the mapping from $\mathcal{R}$ to $\mathcal{D}$ being many-to-one. However, instead of observing the complete data vector $\mathbf{r}$, we are actually able to observe only the incomplete data $d = d(\mathbf{r})$ in $\mathcal{D}$.

Let $p_c(\mathbf{r}|\boldsymbol{\theta})$ denote the conditional probability density function (pdf) of $\mathbf{r}$, given a parameter vector $\boldsymbol{\theta}$. It follows that the conditional pdf of random variable D, given $\boldsymbol{\theta}$, is defined by

$$p_D(d|\boldsymbol{\theta}) = \int_{\mathcal{R}(d)} p_c(\mathbf{r}|\boldsymbol{\theta})d\mathbf{r} \tag{11.70}$$

where $\mathcal{R}(d)$ is the subspace of $\mathcal{R}$ that is determined by $d = d(\mathbf{r})$. The EM algorithm for parameter estimation is directed at finding a value of $\boldsymbol{\theta}$ that maximizes the *incomplete-data log-likelihood function*

$$L(\boldsymbol{\theta}) = \log p_D(d|\boldsymbol{\theta})$$

This problem, however, is solved indirectly by working *iteratively* with the *complete-data log-likelihood function*

$$L_c(\boldsymbol{\theta}) = \log p_c(\mathbf{r}|\boldsymbol{\theta}) \tag{11.71}$$

which is a random variable because the missing data vector $\mathbf{z}$ is unknown.