14

Stochastic Networks

14.1 Variations of the Hopfield model

In the previous chapter we showed that Hopfield networks can be used to provide solutions to combinatorial problems that can be expressed as the minimization of an energy function, although without guaranteeing global optimality. Once the weights of the edges have been defined, the network shows spontaneous computational properties. Harnessing this spontaneous dynamics for useful computations requires some way of avoiding falling into local minima of the energy function in such a way that the global minimum is reached. In the case of the eight queens problem, the number of local minima is much higher than the number of global minima and very often the Hopfield network does not stabilize at a correct solution. The issue to be investigated is therefore whether a certain variation of the Hopfield model could achieve better results in combinatorial optimization problems.

One possible strategy to improve the global convergence properties of a network consists in increasing the number of paths in search space, in such a way that not only binary but also real-valued states become possible. Continuous updates of the network state become possible and the search for minima of the energy function is no longer limited to the corners of a hypercube in state space. All interior points are now legal network states. A continuous activation function also makes it possible to describe more precisely the electrical circuits used to implement Hopfield networks.

A second strategy to avoid local minima of the energy function consists in introducing *noise* into the network dynamics. As usual, the network will reach states of lower energy but will also occasionally jump to states higher up in the energy ladder. In a statistical sense we expect that this extended dynamics can help the network to skip out of local minima of the energy function. The best-known models of such stochastic dynamics are *Boltzmann machines*.

First we will discuss real-valued states and continuous dynamics. The main difficulty in handling this extended model is providing a precise definition of the energy function in the continuous case.

14.1.1 The continuous model

The Hopfield model with binary or bipolar states can be generalized by accepting, just as in backpropagation networks, all real values in the interval $[0, 1]$ as possible unit states. Consequently, the *discrete* Hopfield model becomes the *continuous* model. The activation function selected is the sigmoid and the dynamics of the network remains asynchronous.

The difference between the new and the discrete model is given by the activation assumed by unit i once it has been asynchronously selected for updating. The new state is given by

$$x_i = s(u_i) = \frac{1}{1 + \mathrm{e}^{-u_i}}$$

where u_i denotes the net excitation of the unit. Additionally we assume that a unit's excitation changes slowly in time. Changes in the state of other units will not be registered instantaneously, but with a certain delay. The net excitation of unit i changes in time according to

$$\frac{du_i}{dt} = \gamma \left(-u_i + \sum_{j=1}^{n} w_{ij} x_j \right) = \gamma \left(-u_i + \sum_{j=1}^{n} w_{ij} s(u_j) \right), \tag{14.1}$$

where γ denotes a positive learning constant and w_{ij} the weight between unit i and unit j. For a simulation we compute a discrete approximation of du_i, which is added to the current value of u_i. The result leads to the new state $x_i = s(u_i)$.

We have to show that the defined dynamics of the continuous model leads to equilibrium. With this objective in mind we define an energy function with only slight differences from the one for the discrete model. Hopfield [199] proposes the following energy function:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \int_0^{x_i} s^{-1}(x) dx.$$

We just have to show that the energy of the network becomes lower after each state update. The change in time of the energy is given by

$$\frac{dE}{dt} = -\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \frac{dx_i}{dt} x_j + \sum_{i=1}^{n} s^{-1}(x_i) \frac{dx_i}{dt}.$$

This expression can be simplified by remembering that the network is symmetric (i.e., $w_{ij} = w_{ji}$). Since $u_i = s^{-1}(x_i)$ it holds that

$$\frac{dE}{dt} = -\sum_{i=1}^{n} \frac{dx_i}{dt} \left(\sum_{j=1}^{n} w_{ij} x_j - u_i \right)$$

From equation (14.1) we now obtain

$$\frac{dE}{dt} = -\frac{1}{\gamma}\sum_{i=1}^{n}\frac{dx_i}{dt}\frac{du_i}{dt}.$$

Since $x_i = s(u_i)$, the above expression can be written as

$$\frac{dE}{dt} = -\frac{1}{\gamma}\sum_{i=1}^{n}s'(u_i)\left(\frac{du_i}{dt}\right)^2.$$

We know that $s'(u_i) > 0$ because the sigmoid is a strict monotone function, and since the constant γ is also positive we finally conclude that

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

The defined dynamics of the network implies that the energy is reduced or remains unchanged after each update. A stable state is reached when dE/dt vanishes. This happens when du_i/dt reaches the saturation region of the sigmoid where $du_i/dt \approx 0$. The state of all units can no longer change appreciably.

The time needed for convergence can increase exponentially, since du_i/dt gets smaller and smaller. We must therefore require convergence to a neighborhood of a local minimum and this can happen in finite time. Some authors have proposed variations of (14.1) that accelerate the convergence process.

Experimental results show that the continuous Hopfield model can find better solutions for combinatorial problems than the discrete model. However, in the case of really hard problems (for example the TSP), the continuous model has no definite advantage over the discrete model. Still, even in this case the continuous model remains interesting, because it can be more easily implemented with analog hardware.

14.2 Stochastic systems

The optimization community has been using a method known as *simulated annealing* for many years. [304]. Annealing is a metallurgical process in which a material is heated and then slowly brought to a lower temperature. The crystalline structure of the material can reach a global minimum in this way. The high temperature excites all atoms or molecules, but later, during the cooling phase, they have enough time to assume their optimal position in the crystalline lattice and the result is fewer fractures and fewer irregularities in the crystal.

Annealing can avoid local minima of the lattice energy because the dynamics of the particles contains a temperature-dependent component. The particles not only lose energy during the cooling phase, but sometimes borrow some energy from the background and assume a higher-energy state. Shallow

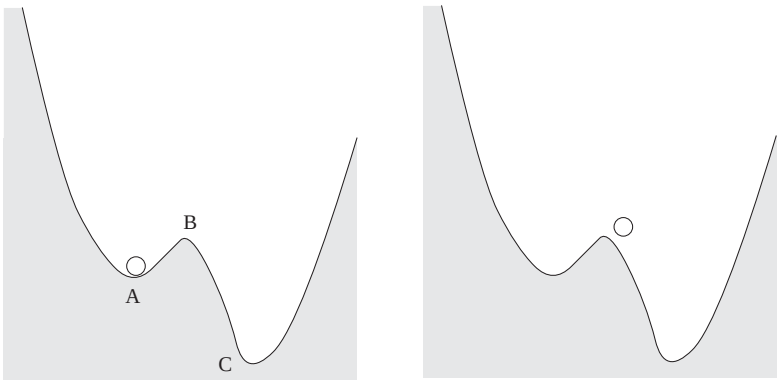

Fig. 14.1. The effect of thermal noise

minima of the energy function can be avoided in this way. This situation is shown in Figure 14.1. If the dynamics of the system is such that the system converges to local energy minima, the system state can be trapped at position A, although C is a more favorable state in terms of the global energy. Some thermal noise, that is, local energy fluctuations, can give the system the necessary impulse to skip over energy barrier B and reach C.

14.2.1 Simulated annealing

To minimize a function E we simulate this phenomenon in the following way: the value of the free variable x is changed always if the update Δx can reduce the value of the function E. However, if the increment actually increases the value of E by ΔE, the new value for x, that is $x + \Delta x$ is accepted with probability

$$p_{\Delta E} = \frac{1}{1 + \mathrm{e}^{\Delta E/T}},$$

where T is a temperature constant. For high values of T, $p_{\Delta E} \approx 1/2$ and the update is accepted half the time. If $T = 0$ only those updates which reduce the value of E are accepted with probability 1. Varying the value of T, from large values all the way down to zero, corresponds to the heating and cooling phases of the annealing process. This explains why this computational technique has been called simulated annealing.

We expect that simulated annealing will allow us to simulate the dynamics of Hopfield networks in such a way that deeper regions of the energy function are favored. It can even be shown that with this simulation strategy the global minimum of the energy function is asymptotically reached [1]. Many other kinds of optimization algorithms can be combined with an annealing phase in order to improve their results. The sigmoid is used for the computation of the probability because it corresponds to those functions used in thermodynamics for the analysis of thermal equilibrium. This simplifies the statistical analysis

of the method, because it is possible to use some results known from statistical mechanics.

Hopfield networks can be considered as optimization systems and thermal noise can be introduced in different ways in the network dynamics. We can distinguish between *Gauss* or *Boltzmann machines*, depending on the stage at which noise is introduced into the computation.

14.2.2 Stochastic neural networks

The dynamics of a Hopfield network can be generalized according to the strategy described in our next definition.

Definition 18. *A Boltzmann machine is a Hopfield network composed of n units with states $x_1, x_2, \ldots, x_n$. The state of unit i is updated asynchronously according to the rule*

$$x_i = \begin{cases} 1 & \text{with probability } p_i, \\ 0 & \text{with probability } 1 - p_i, \end{cases}$$

where

$$p_i = \frac{1}{1 + \exp(-(\sum_{j=1}^{n} w_{ij} x_j - \theta_i)/T)}$$

and T is a positive temperature constant. The constants w_{ij} denote the network weights and the constants θ_i the bias of the units.

The Boltzmann machine can be defined using bipolar or binary states, but in this chapter we will stick to binary states. The energy function for a Boltzmann machine has the same form as for Hopfield networks:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

The difference between a Boltzmann machine and a Hopfield network is the stochastic activation of the units. If T is very small then $p_i \approx 1$ when $\sum_{i=1}^{n} w_{ij} - \theta_i$ is positive. If the net excitation is negative, then $p_i \approx 0$. In this case the dynamics of the Boltzmann machine approximates the dynamics of the discrete Hopfield network and the Boltzmann machine settles on a local minimum of the energy function. If $T > 0$, however, the probability of a transition, or a sequence of transitions, from a network state $x_1, x_2, \ldots, x_n$ to another state is never zero. The state space of the Boltzmann machine contains the 2^n vertices of the n-dimensional hypercube. When a network state is selected, we are choosing a vertex of the hypercube. The probability of reaching any other vertex that is different only at the i-th single component is proportional to p_i when $x_i = 0$ or to $1 - p_i$ when $x_i = 1$. In both cases the probability of a transition does not vanish. A transition from any given

vertex to another is always possible. The transition probability along a predefined path is given by the product of all the transition probabilities from vertex to vertex along this path. The product is positive whenever $T > 0$. Therefore, Boltzmann machines with $T > 0$ cannot settle on a single state; they reduce the energy of the system but they also let it become larger. During the cooling phase we expect those transitions from higher to lower energy levels to be more probable, as it happens in simulated annealing experiments. When T is large, the network state visits practically the complete state space. During the cooling phase, the network begins to stay longer in the basins of attraction of the local minima. If the temperature is reduced according to the correct schedule, we can expect the system to reach a global minimum with the integrated transition probability 1.

14.2.3 Markov chains

Let us illustrate the dynamics defined in the last section with a small example. Take one of the smallest Hopfield networks, the flip-flop. The weight between the two units is equal to -1 and the threshold is -0.5.

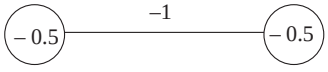

Fig. 14.2. A flip-flop network

The network works using binary states. There are four of them with the following energies:

$$\begin{array}{llll} E_{00} = & 0.0 & E_{10} = & -0.5 \\ E_{01} = & -0.5 & E_{11} = & 0.0 \end{array}$$

Figure 14.3 shows the distribution of the four states in energy levels. We have included the transition probabilities from one state to the other, under the assumption $T = 1$. For example, for a transition from state 00 to state 01, the second unit must be selected and its state must change from 0 to 1. The probability of this happening is

$$p_{00\rightarrow 01} = \frac{1}{2}\left(\frac{1}{1+\exp(-0.5)}\right) = 0.31$$

The factor $1/2$ is used in the computation because any one of two units can be selected in the update step and the probability that a unit is selected for an update is $1/2$. The argument 0.5 corresponds to the net excitation of the second unit in the network state 00. All other transition probabilities were computed in a similar way.

In order to analyze the dynamics of a Boltzmann machine we need to refer to its matrix of transition probabilities.

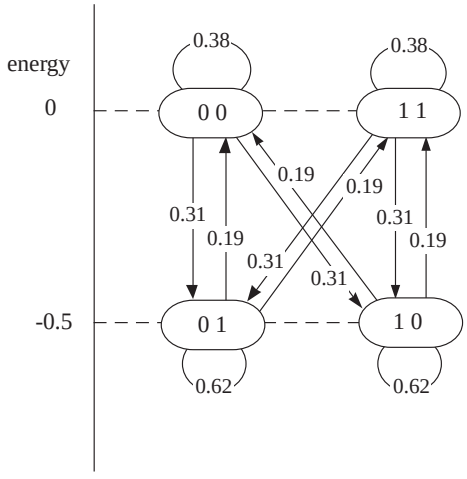

Fig. 14.3. Transition probabilities of a flip-flop network ($T = 1$)

Definition 19. *The state transition matrix* $\mathbf{P}_T = \{p_{ij}\}$ *of a Boltzmann machine with* n *units is the* $2^n \times 2^n$ *matrix, whose elements* p_{ij} *represent the probability of a transition from state* i *to the state* j *in a single step at the temperature* T*.*

$\mathbf{P}_T$ is a $2^n \times 2^n$ matrix because there are 2^n network states. The transition matrix is, according to the definition, a sparse matrix. The matrix for the flip-flop network, for example, is of dimension 4×4. The diagonal elements p_{ii} of the matrix $\mathbf{P}_T$ represent the probabilities that state i does not change. If the states 00, 01, 10, and 11 are taken in this order, the matrix $\mathbf{P} \equiv \mathbf{P}_1$ for the flip-flop network is given by:

$$\mathbf{P} = \begin{pmatrix} \frac{1}{1+e^{1/2}} & \frac{1}{2}\frac{1}{1+e^{-1/2}} & \frac{1}{2}\frac{1}{1+e^{-1/2}} & 0 \\ \frac{1}{2}\frac{1}{1+e^{1/2}} & \frac{1}{1+e^{-1/2}} & 0 & \frac{1}{2}\frac{1}{1+e^{1/2}} \\ \frac{1}{2}\frac{1}{1+e^{1/2}} & 0 & \frac{1}{1+e^{-1/2}} & \frac{1}{2}\frac{1}{1+e^{1/2}} \\ 0 & \frac{1}{2}\frac{1}{1+e^{-1/2}} & \frac{1}{2}\frac{1}{1+e^{-1/2}} & \frac{1}{1+e^{1/2}} \end{pmatrix}.$$

The numerical value of the matrix in our example is:

$$\mathbf{P} = \begin{pmatrix} 0.38 & 0.31 & 0.31 & 0 \\ 0.19 & 0.62 & 0 & 0.19 \\ 0.19 & 0 & 0.62 & 0.19 \\ 0 & 0.31 & 0.31 & 0.38 \end{pmatrix}.$$

It follows from the definition of the transition matrix, that $0 \leq p_{ij} \leq 1$. As long as $T > 0$, it holds that $0 \leq p_{ij} < 1$, since no network state is stable. It should be clear that

$$\sum_{j=1}^{n} p_{ij} = 1,$$

because at each step the state i is changed with probability $\sum_{j \neq i}^{n} p_{ij}$ or remains the same with probability p_{ii}. A matrix with elements $p_{ij} \geq 0$ and whose rows add up to 1 is called a *stochastic matrix*. Boltzmann machines are examples of a *first-order Markov process*, because the transition probabilities only depend on the current state, not on the history of the system. In this case the matrix is called a *Markov matrix*. Using the transition matrix it is possible to compute the probability distribution of the network states at any time t. Assume, for example, that each of the four states of the flip-flop network is equiprobable at the beginning. The original probability distribution $\mathbf{v}_0$ is therefore

$$\mathbf{v}_0 = (0.25,\, 0.25,\, 0.25,\, 0.25)$$

The probability distribution $\mathbf{v}_1$ after an update step of the network is given by

$$\begin{aligned}
\mathbf{v}_1 &= \mathbf{v}_0 P \\
&= (0.25,\, 0.25,\, 0.25,\, 0.25) \begin{pmatrix} 0.38 & 0.31 & 0.31 & 0 \\ 0.19 & 0.62 & 0 & 0.19 \\ 0.19 & 0 & 0.62 & 0.19 \\ 0 & 0.31 & 0.31 & 0.38 \end{pmatrix} \\
&= (0.19,\, 0.31,\, 0.31,\, 0.19)
\end{aligned}$$

As can be seen, for $T = 1$ the flip-flop network assumes the states 01 and 10 62% of the time, and 38% of the time the states 00 and 11. The development of the probability distribution of the network states is computed in general according to

$$\mathbf{v}_t = \mathbf{v}_{t-1}\mathbf{P}.$$

A dynamical system obeying such an equation in which a Markov matrix $\mathbf{P}$ is involved, is called a *Markov chain*. The main issue is finding the final probability distribution given by a vector $\mathbf{v}$ for which

$$\mathbf{v} = \mathbf{v}\mathbf{P}\,.$$

The vector $\mathbf{v}$ is therefore an eigenvector of $\mathbf{P}$ with eigenvalue 1. A fundamental result of Markov chains theory states that such a stable distribution $\mathbf{v}$ always exists, if all states can be reached from any other state in one or more steps, and with non-zero probability [363]. Since all Boltzmann machines

with $T > 0$ fulfill these conditions, such a stable probability distribution, representing *thermal equilibrium* of the network dynamics, exists. It is important to underline that the network arrives to the stable distribution independently of the initial one. If the initial distribution is denoted by $\mathbf{v}_0$, then

$$\mathbf{v}_t = \mathbf{v}_{t-1}\mathbf{P} = \mathbf{v}_{t-2}\mathbf{P}^2 = \cdots = \mathbf{v}_0\mathbf{P}^t$$

When the stable state $\mathbf{v}$ has been reached, the matrix $\mathbf{P}^T$ has stabilized. In the case of the flip-flop network, the matrix $\mathbf{P}^t$ converges to

$$\lim_{t\to\infty} \mathbf{P}^t = \begin{pmatrix} 0.19 & 0.31 & 0.31 & 0.19 \\ 0.19 & 0.31 & 0.31 & 0.19 \\ 0.19 & 0.31 & 0.31 & 0.19 \\ 0.19 & 0.31 & 0.31 & 0.19 \end{pmatrix}.$$

The stable matrix consists of four identical rows. Starting from an arbitrary distribution (a_1, a_2, a_3, a_4) we can compute the stable distribution:

$$\begin{aligned} \mathbf{v} &= (a_1, a_2, a_3, a_4) \begin{pmatrix} 0.19 & 0.31 & 0.31 & 0.19 \\ 0.19 & 0.31 & 0.31 & 0.19 \\ 0.19 & 0.31 & 0.31 & 0.19 \\ 0.19 & 0.31 & 0.31 & 0.19 \end{pmatrix} \\ &= (0.19,\ 0.31,\ 0.31,\ 0.19). \end{aligned}$$

In the computation we made use of the fact that $a_1 + a_2 + a_3 + a_4 = 1$, since this must hold for all probability distributions. The same distribution was found in the vector-matrix multiplication after a single step (because of the simplicity of the problem). As this example has shown, any Boltzmann machine can be analyzed by constructing the transition matrix in order to find its eigenvectors using any of the common iterative methods.

14.2.4 The Boltzmann distribution

It is theoretically possible to use probability functions other than the sigmoid in order to define the network dynamics. Its advantage, nevertheless, is that we can find this kind of transition function in many interesting physical processes.

Consider a system in thermal equilibrium with m different states and associated energies $E_1, E_2, \ldots, E_m$. The probability p_{ij} of a transition from a state of energy E_i to another state of energy E_j is given by

$$p_{ij} = \frac{1}{1 + \mathrm{e}^{(E_j - E_i)/T}} \tag{14.2}$$

Physical systems obeying such a probability function for energy transitions, and at thermal equilibrium, reach a stable probabilistic state known as the *Boltzmann distribution*.

According to the Boltzmann distribution the probability p_i that a system assumes the energy level E_i during thermal equilibrium is

$$p_i = \frac{\mathrm{e}^{-E_i/T}}{Z} \tag{14.3}$$

In this formula $Z = \sum_{i=1}^{m} \mathrm{e}^{-E_i/T}$ is a normalizing factor known as the state sum.

It is important to deal with the Boltzmann distribution analytically because it provides us with the solution of the eigenvector problem for a Markov matrix without having to perform the iterative computations described before. The precise derivation of the Boltzmann distribution is done in ergodic theory, which deals with the dynamics of infinite Markov chains, among other problems.

We can nevertheless verify the validity of the Boltzmann distribution with some straightforward calculations. For a system in thermal equilibrium, the transition matrix for m states with associated energies $E_1, E_2, \ldots, E_m$ is the following:

$$P = \begin{pmatrix} 1 - \sum_{i=2}^{m} \frac{1}{1+\mathrm{e}^{(E_i-E_1)/T}} & \cdots & \frac{1}{1+\mathrm{e}^{(E_m-E_1)/T}} \\ \frac{1}{1+\mathrm{e}^{(E_1-E_2)/T}} & \ddots & \vdots \\ \vdots & & \vdots \\ \frac{1}{1+\mathrm{e}^{(E_1-E_m)/T}} & \cdots 1 - \sum_{i=1}^{m-1} \frac{1}{1+\mathrm{e}^{(E_i-E_m)/T}} & \end{pmatrix}$$

The product of the Boltzmann distribution

$$\frac{1}{Z}\left(\mathrm{e}^{-E_1/T}, \mathrm{e}^{-E_2/T}, ..., \mathrm{e}^{-E_m/T}\right).$$

with $\mathbf{P}$ is a vector of dimension m. Its first component is

$$\begin{aligned} v_1 &= \frac{1}{Z}\left(\mathrm{e}^{-E_1/T}\left(1 - \sum_{i=2}^{m} \frac{1}{1+\mathrm{e}^{(E_i-E_1)/T}}\right) + \sum_{i=2}^{m} \frac{\mathrm{e}^{-E_i/T}}{1+\mathrm{e}^{(E_1-E_i)/T}}\right) \\ &= \frac{1}{Z}\left(\mathrm{e}^{-E_1/T} - \sum_{i=2}^{m} \frac{\mathrm{e}^{-E_1/T}}{1+\mathrm{e}^{(E_i-E_1)/T}} + \sum_{i=2}^{m} \frac{\mathrm{e}^{-E_i/T}}{1+\mathrm{e}^{(E_1-E_i)/T}}\right) \\ &= \frac{1}{Z}\left(\mathrm{e}^{-E_1/T} - \sum_{i=2}^{m} \frac{1}{\mathrm{e}^{E_1/T}+\mathrm{e}^{E_i/T}} + \sum_{i=2}^{m} \frac{1}{\mathrm{e}^{E_i/T}+\mathrm{e}^{E_1/T}}\right) \\ &= \frac{1}{Z}\mathrm{e}^{-E_1/T}. \end{aligned}$$

We again get the first component of the Boltzmann distribution. The same can be shown for the other components, proving that the distribution is stable with respect to $\mathbf{P}$.

It easy to show that a Boltzmann machine is governed by equation (14.2). If a transition from state α to state β occurs, a unit k must have changed its state from x_k to x'_k. The energy of the network in state α is given by

$$E_\alpha = -\frac{1}{2}\sum_{i\neq k}^{n}\sum_{j\neq k}^{n} w_{ij}x_i x_j + \sum_{i\neq k}^{n}\theta_i x_i - \sum_{i=1}^{n} w_{ki}x_i x_k + \theta_k x_k,$$

in state β by

$$E_\beta = -\frac{1}{2}\sum_{i\neq k}^{n}\sum_{j\neq k}^{n} w_{ij}x_i x_j + \sum_{i\neq k}^{n}\theta_i x_i - \sum_{i=1}^{n} w_{ki}x_i x'_k + \theta_k x'_k.$$

The contribution of unit k was expressed in both cases separately. The energy difference is therefore

$$E_\beta - E_\alpha = -\sum_{i=1}^{n} w_{ki}x_i(x'_k - x_k) + \theta_k(x'_k - x_k). \tag{14.4}$$

Since the states are different $x_k \neq x'_k$. Assume without loss of generality that $x_k = 0$ and the unit state changed to $x'_k = 1$. This happened following the defined rules, that is, with probability

$$p_{\alpha\to\beta} = \frac{1}{1+\exp\left(-\left(\sum_{i=1}^{n} w_{ki}x_i - \theta_k\right)\right)}. \tag{14.5}$$

Since $x'_k = 1$ it holds that $x'_k - x_k = 1$ and (14.4) transforms to

$$E_\beta - E_\alpha = -\sum_{i=1}^{n} w_{ki}x_i + \theta_k. \tag{14.6}$$

The transition is ruled by equation (14.5), which is obviously equivalent to (14.2). The only limitation is that in Boltzmann machines state transitions involving two units are not allowed, but this does not affect the substance of our argument. The Boltzmann distribution is therefore the stable probability distribution of the network. In our flip-flop example, the stable distribution is given by

$$\begin{aligned}
\mathbf{v} &= \frac{1}{\mathrm{e}^{-E_{00}} + \mathrm{e}^{-E_{01}} + \mathrm{e}^{-E_{10}} + \mathrm{e}^{-E_{11}}}\left(\mathrm{e}^{-E_{00}}, \mathrm{e}^{-E_{01}}, \mathrm{e}^{-E_{10}}, \mathrm{e}^{-E_{11}}\right) \\
&= \frac{1}{2}\left(\frac{1}{1+\mathrm{e}^{0.5}}\right)\left(1, \mathrm{e}^{0.5}, \mathrm{e}^{0.5}, 1\right) \\
&= (0.19, 0.31, 0.31, 0.19)\,.
\end{aligned}$$

The result is the same as that computed previously using the direct method.

14.2.5 Physical meaning of the Boltzmann distribution

It is interesting to give the Boltzmann distribution a physical interpretation. This can be done using an example discussed by Feynman. The atmosphere is a system in which the air molecules occupy many different energy levels in the gravitational field. We can investigate the distribution of the air molecules in an air column in thermal equilibrium. Figure 14.4 shows a slice of thickness dh of a vertical air cylinder. The slice is cut at height h. The weight G of a differential air cylinder with unitary base area is

$$G = -mgn\,dh$$

where m represents the mass of an air particle (all of them are equal to simplify the derivation), g the gravitational field strength, and n the number of air particles in a unitary volume. In that case, $n\,dh$ is the number of particles in a differential cylinder.

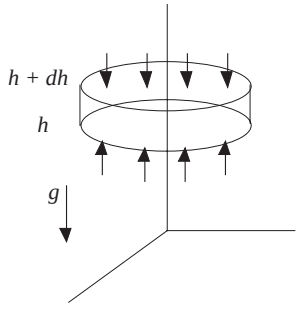

Fig. 14.4. Horizontal slice of an air column in thermal equilibrium

The weight G is equal to the difference between the pressure P_{h+dh} on the upper side of the cylinder and the pressure P_h on the lower side, that is,

$$dP = P_{h+dh} - P_h = -mgn\,dh$$

According to the theory of ideal gases,

$$P = nkT,$$

where k represents the Boltzmann constant and T the temperature. Consequently, $dP = kTdn$. Combining the last two equations we get

$$kT\,dn = -mgn\,dh$$
$$\frac{dn}{n} = -\frac{mg}{kT}dh$$

From this we conclude that

$$n = n_0 \mathrm{e}^{-mgh/kT}.$$

The equation tells us that the number of air particles decreases exponentially with altitude (and correspondingly with the energy level). The equation is similar to (14.3). The above analysis shows therefore that the Boltzmann distribution determines the mean density of the particles which reach the energy level $-mgh/kT$. We can deduce similar conclusions for other physical systems governed by the Boltzmann distribution.

The above result gives an intuitive idea of one of the essential properties of the Boltzmann distribution: a system in thermal equilibrium stays in states of low energy with greater probability. Fluctuations can occur, but the number of transitions to higher energy levels decreases exponentially. Those transitions can be increased by raising the temperature T. The system is "heated" and the number of state transitions grows accordingly.

14.3 Learning algorithms and applications

As we have shown, when the temperature T is positive there are no absolutely stable states. Nevertheless we can ask if a Boltzmann machine can be trained to reproduce a given probability distribution of network states. If this can be done the network will learn to behave in a statistically reproducible way, once some of the inputs have been fixed.

14.3.1 Boltzmann learning

Ackley, Hinton, and Sejnowski developed the algorithm called *Boltzmann learning* [7]. We follow Hertz et al. in the derivation [189]. An alternative proof can be found in [108].

The states of the input units of the network will be indexed with α and the states of the hidden units with β (hidden units are those which receive no external input). If there are m input and k hidden units, the state of the network is specified by the state of $n = m+k$ units. Without loss of generality we consider only units with threshold zero.

The probability P_α that the input units assume state α (without considering the state β of the hidden units) is given by

$$P_\alpha = \sum_\beta P_{\alpha\beta} \tag{14.7}$$

where $P_{\alpha\beta}$ denotes the probability of the combined network state $\alpha\beta$ and the sum is done considering all possible states of the hidden units. Equation (14.7) can be written as

$$P_\alpha = \frac{1}{Z} \sum_\beta \exp(-\gamma E_{\alpha\beta}) \tag{14.8}$$

where

$$Z = \sum_{\alpha,\beta} \exp(-\gamma E_{\alpha\beta}) \tag{14.9}$$

and $\gamma = 1/T$. Equations (14.8) and (14.9) are derived from the Boltzmann distribution. The energy of the network in state $\alpha\beta$ is

$$E_{\alpha\beta} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i^{\alpha\beta} x_j^{\alpha\beta}. \tag{14.10}$$

Equation (14.7) provides us with the probability distribution of the states α in a *free running* network, that is, a network without external input. We want the network to learn to take the states α with the probability distribution R_α. During the training phase, an input vector is clamped to the input units. The distance D between the desired and the actual behavior of the network is measured using the expression

$$D = \sum_{\alpha} R_\alpha \log \frac{R_\alpha}{P_\alpha}. \tag{14.11}$$

The distance D is zero only if the distribution R_α is equal to the distribution P_α. The learning algorithm must minimize D. Gradient descent on this error function leads to the weight update

$$\Delta w_{ij} = -\eta \frac{\partial D}{\partial w_{ij}} = \eta \sum_{\alpha} \frac{R_\alpha}{P_\alpha} \frac{\partial P_\alpha}{\partial w_{ij}}. \tag{14.12}$$

Combining equations (14.8), (14.9), and (14.10) we obtain the expression

$$\begin{aligned}\frac{\partial P_\alpha}{\partial w_{ij}} &= \frac{\gamma \sum_\beta \exp(-\gamma E_{\alpha\beta}) x_i^{\alpha\beta} x_j^{\alpha\beta}}{Z} \\ &\quad - \frac{\gamma \left(\sum_\beta \exp(-\gamma E_{\alpha\beta})\right) \sum_{\lambda\mu} \exp(-\gamma E_{\lambda\mu}) x_i^{\lambda\mu} x_j^{\lambda\mu}}{Z^2} \\ &= \gamma \left(\sum_\beta x_i^{\alpha\beta} x_j^{\alpha\beta} P_{\alpha\beta} - P_\alpha \left\langle x_i x_j \right\rangle_{free} \right).\end{aligned}$$

The expression $\langle x_i x_j \rangle_{free}$ denotes the expected value of the product of the unit states x_i and x_j in the free running network. Combining this result with (14.12) we get

$$\Delta w_{ij} = \eta\gamma \left(\sum_\alpha \frac{R_\alpha}{P_\alpha} \sum_\beta x_i^{\alpha\beta} x_j^{\alpha\beta} P_{\alpha\beta} - \sum_\alpha R_\alpha \left\langle x_i x_j \right\rangle_{free} \right). \tag{14.13}$$

We introduce the conditional probability $P_{\beta|\alpha}$ which is the probability that the hidden units assume state β when the input units are in state α, that is,

$$P_{\alpha\beta} = P_{\beta|\alpha} P_\alpha .$$

The expected value of the product of x_i and x_j during training (i.e., with clamped input) is given by

$$\langle x_i x_j \rangle_{fixed} = \sum_\alpha R_\alpha P_{\beta|\alpha} x_i^{\alpha\beta} x_j^{\alpha\beta}.$$

Using both definitions of the expected products and substituting in (14.13) leads to the expression

$$\begin{aligned} \Delta w_{ij} &= \eta\gamma \left(\sum_{\alpha,\beta} R_\alpha P_{\beta|\alpha} x_i^{\alpha\beta} x_j^{\alpha\beta} - \sum_\alpha R_\alpha \langle x_i x_j \rangle_{free} \right) \\ &= \eta\gamma \left(\langle x_i x_j \rangle_{fixed} - \langle x_i x_j \rangle_{free} \right) \end{aligned}$$

The above equation describes the learning method for a Boltzmann machine: the network is let run with clamped inputs and the average $\langle x_i x_j \rangle_{fixed}$ is computed. In a second pass, the network is let run without inputs. The average $\langle x_i x_j \rangle_{free}$ is computed and subtracted from the previously computed average. This provides us with an estimate of Δw_{ij}. The network weights are corrected until gradient descent has found a local minimum of the error measure D.

Note that Boltzmann learning resembles Hebbian learning. The values of $\langle x_i x_j \rangle_{fixed}$ and $\langle x_i x_j \rangle_{free}$ correspond to the entries of the stochastic correlation matrix of network states. The second term is subtracted and is interpreted as Hebbian "forgetting". This controlled loss of memory should prevent the network from learning false, spontaneously generated states. Obviously, Boltzmann learning in this form is computationally expensive for serial computers, since several passes over the whole data set and many update steps are needed.

14.3.2 Combinatorial optimization

Boltzmann machines can be used in those cases in which Hopfield networks become trapped in shallow local minima of the energy function. The energy function should have basins of attraction with clearly defined borders. Only in that case can we give a statistical guarantee that the global minimum will be reached. For problems such as the TSP this is not guaranteed. In these problems there are few global minima surrounded by many relatively deep local minima. The basins of attraction of the global minima are not much larger than the basins of attraction of the local minima. A Boltzmann machine can hardly improve on the results of the Hopfield network when the error function has an unfavorable shape. Boltzmann machines are therefore more important from a theoretical than from an engineering point of view. They can be used to model biological phenomena, such as conditioning and

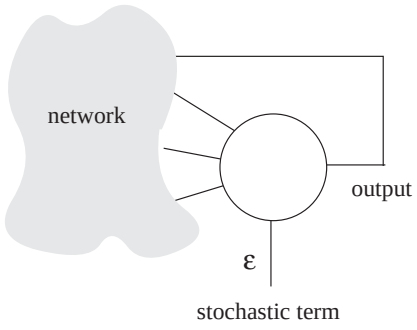

Fig. 14.5. Working principle of a Gauss machine

the emergence of memory. Biological neurons are stochastic systems and our models of artificial neural networks should reflect this fact.

Another possible realization of a stochastic Hopfield network consists in providing each unit in the network with a stochastic external input. Each unit i computes its net excitation $\sum_{j=1}^{N} w_{ij}x_j - \theta_i$ plus a stochastic term ε, as shown in Figure 14.5. The network dynamics is the standard Hopfield dynamics. Due to the stochastic term, some units update their states and can occasionally bring the network to states with a higher energy. Such stochastic systems have been called *Gauss machines*, because the stochastic term obeys a Gauss distribution. Some authors have compared Boltzmann and Gauss machines and find the latter more suitable for some tasks [13].

14.4 Historical and bibliographical remarks

Simulated annealing has been used for many years in the field of numerical optimization. The technique is a special case of the Monte Carlo method. Simulated annealing was brought into the mainstream of computing by the article published by Kirkpatrick, Gelatt, and Vecchi in 1983 [246]. The necessary conditions for convergence to a global minimum of a given function were studied in the classical paper by Geman and Geman [158]. An extensive analysis of the problem can be found in [1].

The introduction of stochastic computing units was proposed by several authors in different contexts. The Boltzmann machine concept was developed by Hinton and Sejnowski in the 1980s [191]. Their learning algorithm was the first proposed for stochastic networks and allowed dealing with hidden units in networks of the Hopfield type. In the following years other types of stochastic behavior were introduced. This led to models such as the *Cauchy machine* [423] and the Gauss machines mentioned already.

The role of noise in Hopfield networks has been studied by many physicists [25]. It can be shown, for example, that the number of false local minima that

arise in Hopfield networks can be compensated with a stochastic dynamic, so that the statistical capacity of the network is improved.

Exercises

1. Solve the eight queens problem using a Boltzmann machine. Define the network's weights by hand.
2. Find with the computer the equilibrium distribution for a 10-node Boltzmann machine with randomly generated weights. Compute the Boltzmann distribution of the network and compare with the previous result.
3. Train a Boltzmann machine to generate solutions of the eight queens problem. Compare with your manually defined network of Exercise 1.