13

The Hopfield Model

One of the milestones for the current renaissance in the field of neural networks was the associative model proposed by Hopfield at the beginning of the 1980s. Hopfield's approach illustrates the way theoretical physicists like to think about ensembles of computing units. No synchronization is required, each unit behaving as a kind of elementary system in complex interaction with the rest of the ensemble. An energy function must be introduced to harness the theoretical complexities posed by such an approach. The next two sections deal with the structure of Hopfield networks. We then proceed to show that the model converges to a stable state and that two kinds of learning rules can be used to find appropriate network weights.

13.1 Synchronous and asynchronous networks

A relevant issue for the correct design of recurrent neural networks is the adequate synchronization of the computing elements. In the case of McCulloch-Pitts networks we solved this difficulty by assuming that the activation of each computing element consumes a unit of time. The network is built taking this delay into account and by arranging the elements and their connections in the necessary pattern. When the arrangement becomes too contrived, additional units can be included which serve as delay elements. What happens when this assumption is lifted, that is, when the synchronization of the computing elements is eliminated?

13.1.1 Recursive networks with stochastic dynamics

We discussed the design and operation of associative networks in the previous chapter. The synchronization of the output was achieved by requiring that all computing elements evaluate their inputs and compute their output simultaneously. Under this assumption the operation of the associative memory can

be described with simple linear algebraic methods. The excitation of the output units is computed using vector-matrix multiplication and evaluating the sign function at each node.

The methods we have used before to avoid dealing explicitly with the synchronization problem have the disadvantage, from the point of view of both biology and physics, that global information is needed, namely a global time. Whereas in conventional computers synchronization of the digital building blocks is achieved using a clock signal, there is no such global clock in biological systems. In a more biologically oriented simulation, global synchronization should thus be avoided. In this chapter we deal with the problem of identifying the properties of neural networks lacking global synchronization.

Networks in which the computing units are activated at different times and which provide a computation after a variable amount of time are stochastic automata. Networks built from this kind of units behave like *stochastic dynamical systems*.

13.1.2 The bidirectional associative memory

Before we start analyzing asynchronous networks we will examine another kind of synchronous associative model with bidirectional edges. We will arrive at the concept of the *energy function* in a very natural way.

We have already discussed recurrent associative networks in which the output of the network is fed back to the input units using additional feedback connections (Figure 12.3). In this way we designed recurrent dynamical systems and tried to determine their fixpoints. However, there is another way to define a recurrent associative memory made up of two layers which send information recursively between them. The input layer contains units which receive the input to the network and send the result of their computation to the output layer. The output of the first layer is transported by bidirectional edges to the second layer of units, which then return the result of their computation back to the first layer using the same edges. As in the case of associative memory models, we can ask whether the network achieves a stable state in which the information being sent back and forth does not change after a few iterations [258]. Such a network (shown in Figure 13.1) is known as a resonance network or *bidirectional associative memory* (BAM). The activation function of the units is the sign function and information is coded using bipolar values.

The network in Figure 13.1 maps an n-dimensional row vector $\mathbf{x}_0$ to a k-dimensional row vector $\mathbf{y}_0$. We denote the $n \times k$ weight matrix of the network by $\mathbf{W}$ so that the mapping computed in the first step can be written as

$$\mathbf{y}_0 = \mathrm{sgn}(\mathbf{x}_0\mathbf{W}).$$

In the feedback step $\mathbf{y}_0$ is treated as the input and the new computation is

$$\mathbf{x}_1^{\mathrm{T}} = \mathrm{sgn}(\mathbf{W}\mathbf{y}_0^{\mathrm{T}}).$$

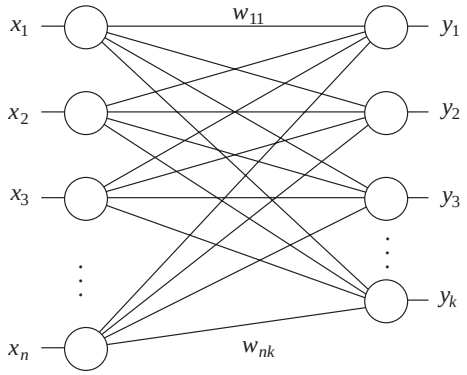

Fig. 13.1. Example of a resonance network (BAM)

A new computation from left to right produces

$$\mathbf{y}_1 = \mathrm{sgn}(\mathbf{x}_1\mathbf{W}).$$

After m iterations the system has computed a set of $m+1$ vector pairs $(\mathbf{x}_0, \mathbf{y}_0), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$ which fulfill the conditions

$$\mathbf{y}_i = \mathrm{sgn}(\mathbf{x}_i\mathbf{W}) \tag{13.1}$$

and

$$\mathbf{x}_{i+1}^{\mathrm{T}} = \mathrm{sgn}(\mathbf{W}\mathbf{y}_i^{\mathrm{T}}). \tag{13.2}$$

The question is whether after some iterations a fixpoint $(\mathbf{x}, \mathbf{y})$ is found. This is the case when both

$$\mathbf{y} = \mathrm{sgn}(\mathbf{x}\mathbf{W}) \quad \text{and} \quad \mathbf{x}^{\mathrm{T}} = \mathrm{sgn}(\mathbf{W}\mathbf{y}^{\mathrm{T}}) \tag{13.3}$$

hold. The BAM is thus a generalization of a unidirectional associative memory. An input vector, the "key", can be presented to the network from the left or from the right and, after some iterations, the BAM finds the corresponding complementary vector. As can be seen, no external feedback connections are necessary. The same edges are used for the transmission of information back and forth.

It can be immediately deduced from (13.3) that if a vector pair $(\mathbf{x}, \mathbf{y})$ is given and we want to condition a BAM to accept this pair as a fixed point, Hebbian learning can be used to compute an adequate matrix $\mathbf{W}$. If $\mathbf{W}$ is defined as $\mathbf{W} = \mathbf{x}^{\mathrm{T}}\mathbf{y}$, as prescribed by Hebbian learning, then

$$\mathbf{y} = \mathrm{sgn}(\mathbf{x}\mathbf{W}) = \mathrm{sgn}(\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{y}) = \mathrm{sgn}(\|\mathbf{x}\|^2\mathbf{y}) = \mathbf{y}$$

and also

$$\mathbf{x}^{\mathrm{T}} = \mathrm{sgn}(\mathbf{W}\mathbf{y}^{\mathrm{T}}) = \mathrm{sgn}(\mathbf{x}^{\mathrm{T}}\mathbf{y}\mathbf{y}^{\mathrm{T}}) = \mathrm{sgn}(\mathbf{x}^{\mathrm{T}}\|\mathbf{y}\|^2) = \mathbf{x}^{\mathrm{T}}.$$

If we want to store several vector pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$ in a BAM, then Hebbian learning works better if the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ and $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are pairwise orthogonal within their respective groups, because in that case the perturbation term becomes negligible (refer to Chap. 12).

For a set of m vector pairs the matrix $\mathbf{W}$ is set to

$$\mathbf{W} = \mathbf{x}_1^{\mathrm{T}}\mathbf{y}_1 + \mathbf{x}_2^{\mathrm{T}}\mathbf{y}_2 + \cdots + \mathbf{x}_m^{\mathrm{T}}\mathbf{y}_m.$$

BAMs can be used to build autoassociative networks because the matrices produced by the Hebb rule or by computing the pseudoinverse are symmetric. To see this, define $\mathbf{X}$ as the matrix, each of whose m rows is an n-dimensional vector, so that if $\mathbf{W}$ denotes the connection matrix of an autoassociative memory for those m vectors, then it is true that

$$\mathbf{X} = \mathbf{X}\mathbf{W} \quad \text{and} \quad \mathbf{X}^{\mathrm{T}} = \mathbf{W}\mathbf{X}^{\mathrm{T}},$$

because $\mathbf{W}$ is symmetric. This is just another way of writing the type of computation performed by a BAM.

13.1.3 The energy function

With the BAM we can motivate and explore the concept of an *energy function* in a simple setting. Assume that a BAM is given for which the vector pair $(\mathbf{x}, \mathbf{y})$ is a stable state. If the initial vector presented to the network from the left is $\mathbf{x}_0$, the network will converge to $(\mathbf{x}, \mathbf{y})$ after some iterations. The vector $\mathbf{y}_0$ is computed according to $\mathbf{y}_0 = \mathrm{sgn}(\mathbf{x}_0\mathbf{W})$. If $\mathbf{y}_0$ is now used for a new iteration from the right, excitation of the units in the left layer can be summarized in an excitation vector $\mathbf{e}$ computed according to

$$\mathbf{e}^{\mathrm{T}} = \mathbf{W}\mathbf{y}_0.$$

The vector pair $(\mathbf{x}_0, \mathbf{y}_0)$ is a stable state of the network if $\mathrm{sgn}(\mathbf{e}) = \mathbf{x}_0$. All vectors $\mathbf{e}$ close enough to $\mathbf{x}_0$ fulfill this condition. These vectors differ from $\mathbf{x}_0$ by a small angle and therefore the product $\mathbf{x}_0\mathbf{e}^{\mathrm{T}}$ is larger than for other vectors of the same length but further away from $\mathbf{x}_0$. The product

$$E = -\mathbf{x}_0\mathbf{e}^{\mathrm{T}} = -\mathbf{x}_0\mathbf{W}\mathbf{y}_0^{\mathrm{T}}$$

is therefore smaller (because of the minus sign) if the vector $\mathbf{W}\mathbf{y}_0^{\mathrm{T}}$ lies closer to $\mathbf{x}_0$. The scalar value E can be used as a kind of index of convergence to the stable states of an associative memory. We call E the *energy function* of the network.

Definition 16. *The energy function E of a BAM with weight matrix $\mathbf{W}$, in which the output $\mathbf{y}_i$ of the right layer of units is computed in the i-th iteration according to equation* (13.1) *and the output $\mathbf{x}_i$ of the left layer is computed according to* (13.2) *is given by*

$$E(\mathbf{x}_i, \mathbf{y}_i) = -\frac{1}{2}\mathbf{x}_i\mathbf{W}\mathbf{y}_i^{\mathrm{T}}. \tag{13.4}$$

The factor 1/2 will be useful later and is just a scaling constant for the energy function. In the following sections we show that the energy function assumes locally minimal values at stable states. The energy function can also be generalized to arbitrary vectors $\mathbf{x}$ and $\mathbf{y}$.

Up to this point we have only considered units with the sign function as activation nonlinearity in the type of associative memories we have discussed. If we now consider units with a threshold and the step function as its activation function, we must use a more general expression for the energy function. This can be done by extending the input vectors with an additional constant component. Each n-dimensional vector $\mathbf{x}$ will be transformed into the vector $(x_1, \ldots, x_n, 1)$. We proceed in a similar way with the k-dimensional vector $\mathbf{y}$. The weight matrix $\mathbf{W}$ must be extended to a new matrix $\mathbf{W}'$ with an additional row and column. The negative thresholds of the units in the right layer of the BAM are included in row $n+1$ of $\mathbf{W}'$, whereas the negative thresholds of the units in the left are used as the entries of the column $k+1$ of the weight matrix. The entry $(n+1, k+1)$ of the weight matrix can be set to zero. This transformation is equivalent to the introduction of an additional unit with constant output 1 into each layer. The weight of each edge from a constant unit to each one of the others is the negative threshold of the connected unit. It is straightforward to deduce that the energy function of the extended network can be written as

$$E(\mathbf{x}_i, \mathbf{y}_i) = -\frac{1}{2}\mathbf{x}_i \mathbf{W} \mathbf{y}_i^{\mathrm{T}} + \frac{1}{2}\theta_r \mathbf{y}_i^{\mathrm{T}} + \frac{1}{2}\mathbf{x}_i \theta_\ell^{\mathrm{T}}. \tag{13.5}$$

The row vector of thresholds of the k units in the left layer is denoted in the above expression by θ_ℓ. The row vector of thresholds of the n units in the right layer is denoted by θ_r.

13.2 Definition of Hopfield networks

So far we have considered only conventional or bidirectional associative memories working with synchronized units. Dropping the assumption of simultaneous firing of the computing elements leads to the appearance of novel network properties.

13.2.1 Asynchronous networks

In an asynchronous network each unit computes its excitation at random times and changes its state to 1 or -1 independently of the others and according to the sign of its total excitation. The probability of two units firing simultaneously is zero. Consequently, the same dynamics can be obtained by selecting one unit randomly, computing its excitation and updating its state accordingly. There will not be any delay between computation of the excitation and state update. We adopt the additional simplification that the state of a unit

is not changed if the total excitation is zero. This means that we leave the sign function undefined for the argument zero. Asynchronous networks are of course more realistic models of biological networks, although the assumption of zero delay in the computation and transmission of signals lacks any biological basis.

Using the energy function it can be shown that a BAM arrives at a stable state after a finite number of iterations. A stable state is a vector pair $(\mathbf{x}, \mathbf{y})$ which fulfills the conditions (13.3). When a BAM reaches this state pair, no component of the bipolar vectors $\mathbf{x}$ and $\mathbf{y}$ can be changed without contradicting (13.3). The vector pair $(\mathbf{x}, \mathbf{y})$ is therefore also a stable state for an asynchronous network.

Proposition 19. *A bidirectional associative memory with an arbitrary weight matrix* $\mathbf{W}$ *reaches a stable state in a finite number of iterations using either synchronous or asynchronous updates.*

Proof. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, a vector $\mathbf{y} = (y_1, y_2, \ldots, y_k)$ and an $n \times k$ weight matrix $\mathbf{W} = \{w_{ij}\}$ the energy function is the bilinear form

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(x_1, x_2, \ldots, x_n) \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1k} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \vdots & & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nk} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}.$$

The value of $E(\mathbf{x}, \mathbf{y})$ can be computed by multiplying first $\mathbf{W}$ by $\mathbf{y}^{\mathrm{T}}$ and the result with $-\mathbf{x}/2$. The product of the i-th row of $\mathbf{W}$ and $\mathbf{y}^{\mathrm{T}}$ represents the excitation of the i-th unit in the left layer. If we denote these excitations by $g_1, g_2, \ldots, g_n$ the above expression transforms to

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(x_1, x_2, \ldots, x_n) \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix}.$$

We can also compute $E(\mathbf{x}, \mathbf{y})$ multiplying first $\mathbf{x}$ by $\mathbf{W}$. The product of the i-th column of $\mathbf{W}$ with $\mathbf{x}$ corresponds to the excitation of unit i in the right layer. If we denote these excitations by $e_1, e_2, \ldots, e_k$, the expression for $E(\mathbf{x}, \mathbf{y})$ can be written as

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(e_1, e_2, \ldots, e_k) \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}.$$

Therefore, the energy function can be written in the two equivalent forms

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}\sum_{i=1}^{k} e_i y_i \quad \text{and} \quad E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}\sum_{i=1}^{n} g_i x_i.$$

In asynchronous networks at each time t we randomly select a unit from the left or right layer. The excitation is computed and its sign is the new activation of the unit. If the previous activation of the unit remains the same after this operation, then the energy of the network has not changed.

The state of unit i on the left layer will change only when the excitation g_i has a different sign than x_i, the present state. The state is updated from x_i to x_i', where x_i' now has the same sign as g_i. Since the other units do not change their state, the difference between the previous energy $E(\mathbf{x}, \mathbf{y})$ and the new energy $E(\mathbf{x}', \mathbf{y})$ is

$$E(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}', \mathbf{y}) = -\frac{1}{2} g_i (x_i - x_i').$$

Since both x_i and $-x_i$ have a different sign than g_i it follows that

$$E(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}', \mathbf{y}) > 0.$$

The new state $(\mathbf{x}', \mathbf{y})$ has a lower energy than the original state $(\mathbf{x}, \mathbf{y})$. The same argument can be made if a unit on the right layer has been selected, so that for the new state $(\mathbf{x}, \mathbf{y}')$ it holds that

$$E(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}, \mathbf{y}') > 0,$$

whenever the state of a unit in the right layer has been flipped.

Any update of the network state reduces the total energy. Since there are only a finite number of possible combinations of bipolar states, the process must stop at some point, that is, a state $(\mathbf{a}, \mathbf{b})$ is found whose energy cannot be further reduced. The network has fallen into a local minimum of the energy function and the state $(\mathbf{a}, \mathbf{b})$ is an attractor of the system. □

The above proposition also holds for synchronous networks, since these can be considered as a special case of asynchronous dynamics. Note that the proposition puts conditions on the matrix $\mathbf{W}$. This means that any given real matrix $\mathbf{W}$ possesses bidirectional stable bipolar states.

13.2.2 Examples of the model

In 1982 the American physicist John Hopfield proposed an asynchronous neural network model which made an immediate impact in the AI community. It is a special case of a bidirectional associative memory, but chronologically it was proposed before the BAM.

In the Hopfield model it is assumed that the individual units preserve their individual states until they are selected for a new update. The selection is made randomly. A Hopfield network consists of n totally coupled units, that is, each unit is connected to all other units except itself. The network is symmetric because the weight w_{ij} for the connection between unit i and

unit j is equal to the weight w_{ji} of the connection from unit j to unit i. This can be interpreted as meaning that there is a single bidirectional connection between both units. The absence of a connection from each unit to itself avoids a permanent feedback of its own state value [198].

Figure 13.2 shows an example of a network with three units. Each one of them can assume the state 1 or -1. A Hopfield network can also be interpreted as an asynchronous BAM in which the left and right layers of units have fused to a single layer. The connections in a Hopfield network with n units can be represented using an $n \times n$ weight matrix $\mathbf{W} = \{w_{ij}\}$ with a zero diagonal.

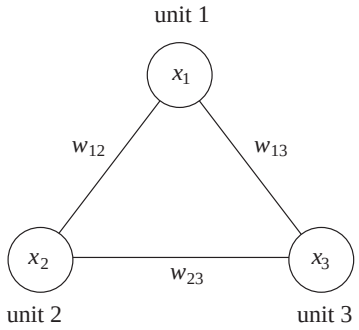

Fig. 13.2. A Hopfield network of three units

It is easy to show that if the weight matrix does not contain a zero diagonal, the network dynamics does not necessarily lead to stable states. The weight matrix

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

for example, transforms the state vector $(1, 1, 1)$ into the state vector $(-1, -1, -1)$ and conversely. In the case of asynchronous updating, the network chooses randomly among the eight possible network states.

A connection matrix with a zero diagonal can also lead to oscillations in the case where the weight matrix is not symmetric. The weight matrix

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

describes the network of Figure 13.3. It transforms the state vector $(1, -1)$ into the state vector $(1, 1)$ when the network is running asynchronously. After this transition the state $(-1, 1)$ can be updated to $(-1, -1)$ and finally to $(1, -1)$. The state vector changes cyclically and does not converge to a stable state.

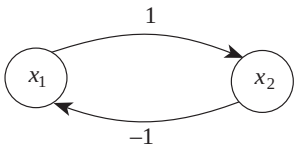

Fig. 13.3. Network with asymmetric connections

The symmetry of the weight matrix and a zero diagonal are thus *necessary conditions* for the convergence of an asynchronous totally connected network to a stable state. These conditions are also sufficient, as we show later.

The units of a Hopfield network can be assigned a threshold θ different from zero. In this case each unit selected for a state update adopts the state 1 if its total excitation is greater than θ, otherwise the state -1. This is the activation rule for perceptrons, so that we can think of Hopfield networks as asynchronous recurrent networks of perceptrons.

The energy function of a Hopfield network composed of units with thresholds different from zero can be defined in a similar way as for the BAM. In this case the vector $\mathbf{y}$ of equation (13.5) is $\mathbf{x}$ and we let $\theta = \theta_\ell = \theta_r$.

Definition 17. *Let* $\mathbf{W}$ *denote the weight matrix of a Hopfield network of* n *units and let* θ *be the* n*-dimensional row vector of units' thresholds. The energy* $E(\mathbf{x})$ *of a state* $\mathbf{x}$ *of the network is given by*

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}\mathbf{W}\mathbf{x}^{\mathrm{T}} + \theta\mathbf{x}^{\mathrm{T}}.$$

The energy function can also be written in the form

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

The factor $1/2$ is used because the identical terms $w_{ij}x_i x_j$ and $w_{ji}x_j x_i$ are present in the double sum.

The energy function of a Hopfield network is a quadratic form. A Hopfield network always finds a local minimum of the energy function. It is thus interesting to look at an example of the shape of such an energy function. Figure 13.4 shows a network of just two units with threshold zero. It is obvious that the only stable states are $(1, -1)$ and $(-1, 1)$. In any other state, one of the units forces the other to change its state to stabilize the network. Such a network is a flip-flop, a logic component with two outputs which assume complementary logic values.

The energy function of a flip-flop with weights $w_{12} = w_{21} = -1$ and two units with threshold zero is given by

$$E(x_1, x_2) = x_1 x_2,$$

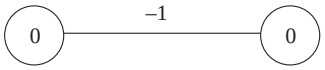

Fig. 13.4. A flip-flop

where x_1 and x_2 denote the states of the first and second units respectively. Figure 13.5 shows the energy function for the so-called continuous Hopfield model [199] in which the unit's states can assume all real values between 0 and 1. In the network of Figure 13.4 only the four discrete states $(1, 1)$, $(1, -1)$, $(-1, 1)$ and $(-1, -1)$ are allowed. The energy function has local minima at $(1, -1)$ and $(-1, 1)$. A flip-flop can therefore be interpreted as a network capable of storing one of the states $(1, -1)$ or $(-1, 1)$.

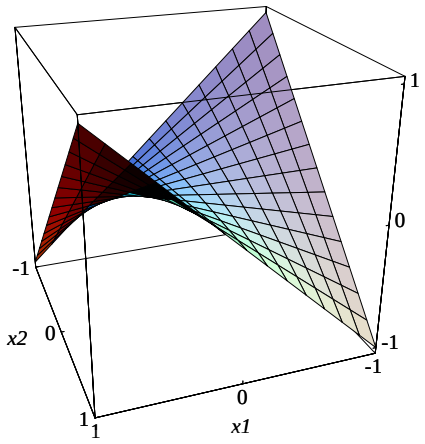

Fig. 13.5. Energy function of a flip-flop

Hopfield networks can also be used to compute logical functions. Conjunction, for example, can be implemented with a network of three units. The states of two units are set and remain fixed during the computation (clamping their states). Only the third unit can change its state. If the network weights and the unit thresholds have the appropriate values, the unconstrained unit will assume a state which corresponds to the conjunction of the two clamped states.

Figure 13.6 shows a network for the computation of the logical disjunction of two Boolean values x_1 and x_2. The input is clamped and after some time the network settles to a state which corresponds to the disjunction of x_1 and x_2. The constants "true" and "false" correspond to the numerical values 1 and -1. In this network the thresholds of the clamped units and their mutual connections play no role in the computation.

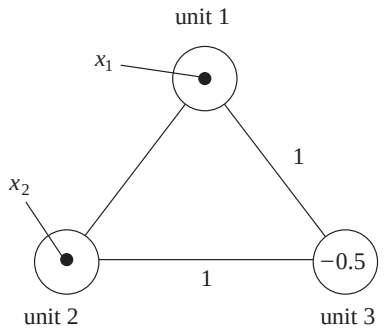

Fig. 13.6. Network for the computation of the OR function

Since the individual units of the network are perceptrons, the question of whether there are logic functions which cannot be computed by a Hopfield network of a given size arises. This is the case in our next example. Assume that a Hopfield network of three units should store the set of stable states given by the following table:

unit	1	2	3
state 1	−1	−1	−1
state 2	1	−1	1
state 3	−1	1	1
state 4	1	1	−1

From the point of view of the third unit (third column) this is the XOR function. If the four vectors shown above are to become stable states of the network, the third unit cannot change state when any of these four vectors has been loaded in the network. In this case the third unit should be capable of linearly separating the vectors $(-1,-1)$ and $(1,1)$ from the vectors $(-1,1)$ and $(1,-1)$, which we know is impossible. The same argument is valid for any of the three units, since the table given above remains unchanged after a permutation of the units' labels. This shows that no Hopfield network of three units can have these stable states. However, the XOR problem can be solved if the network is extended to four units. The network of Figure 13.7 can assume the following stable states, if adequate weights and thresholds are selected:

unit	1	2	3	4
state 1	−1	−1	−1	1
state 2	1	−1	1	1
state 3	−1	1	1	1
state 4	1	1	−1	−1

The third column represents the XOR function of the two first columns. The fourth column corresponds to an auxiliary unit, whose state can be set from

outside. The unknown weights can be found using the learning algorithms described in the next sections.

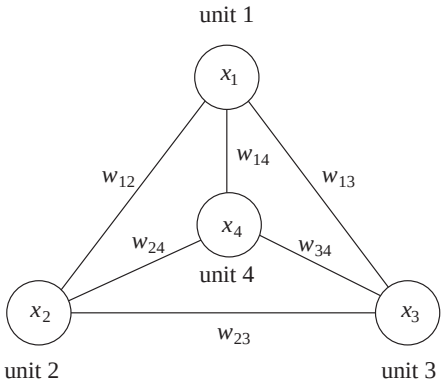

Fig. 13.7. Network for the computation of XOR

13.2.3 Isomorphism between the Hopfield and Ising models

Physicists have analyzed the Hopfield model in such exquisite detail because it is isomorphic to the *Ising model* of magnetism (at temperature zero) [25]. Ising proposed the model which now bears his name more than 70 years ago in order to describe some properties of ensembles of elementary magnets [214].

In general, the Ising model can be used to describe those systems made of particles capable of adopting one of two states. In the case of ferromagnetic materials, their atoms can be modeled as particles of spin $1/2$ (*up*) or spin $-1/2$ (*down*). The spin points in the direction of the magnetic field. All tiny magnets interact with each other. This causes some of the atoms to flip their spin until equilibrium is reached and the total magnetization of the material reaches a constant level, which is the sum of the individual spins. With these few assumptions we can show that the energy function deduced from the Ising model has the same form as the energy function of Hopfield networks.

The total magnetic field h_i sensed by the atom i in an ensemble of particles is the sum of the fields induced by each atom and the external field h^* (if present), that is

$$h_i = \sum_{j=1}^{n} w_{ij} x_j + h^*, \tag{13.6}$$

where w_{ij} represents the magnitude of the magnetic coupling between the atoms labeled i and j. The magnetic coupling changes according to the distance between atoms and the magnetic permeability of the environment. The