Solution to exercise 42.4 (p.508). Take a binary Hopfield network with 2 neurons and let $w_{12} = w_{21} = 1$, and let the initial condition be $x_1 = 1$, $x_2 = -1$. Then if the dynamics are synchronous, on every iteration both neurons will flip their state. The dynamics do not converge to a fixed point.

Solution to exercise 42.12 (p.520). The key to this problem is to notice its similarity to the construction of a binary symbol code. Starting from the empty string, we can build a binary tree by repeatedly splitting a codeword into two. Every codeword has an implicit probability 2^{-l}, where l is the depth of the codeword in the binary tree. Whenever we split a codeword in two and create two new codewords whose length is increased by one, the two new codewords each have implicit probability equal to half that of the old codeword. For a complete binary code, the Kraft equality affirms that the sum of these implicit probabilities is 1.

Similarly, in `southeast`, we can associate a 'weight' with each piece on the board. If we assign a weight of 1 to any piece sitting on the top left square; a weight of 1/2 to any piece on a square whose distance from the top left is one; a weight of 1/4 to any piece whose distance from the top left is two; and so forth, with 'distance' being the city-block distance; then every legal move in `southeast` leaves unchanged the total weight of all pieces on the board. Lyapunov functions come in two flavours: the function may be a function of state whose value is known to stay constant; or it may be a function of state that is bounded below, and whose value always decreases or stays constant. The total weight is a Lyapunov function of the second type.

The starting weight is 1, so now we have a powerful tool: a conserved function of the state. Is it possible to find a position in which the ten highest-weight squares are vacant, and the total weight is 1? What is the total weight if *all* the other squares on the board are occupied (figure 42.14)? The total weight would be $\sum_{l=4}^{\infty}(l+1)2^{-l}$, which is equal to 3/4. So it is impossible to empty all ten of those squares.

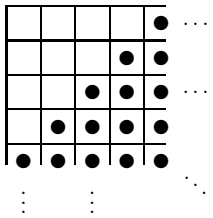

Figure 42.14. A possible position for the `southeast` puzzle?

43

Boltzmann Machines

▶ 43.1 From Hopfield networks to Boltzmann machines

We have noticed that the binary Hopfield network minimizes an energy function

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x} \tag{43.1}$$

and that the continuous Hopfield network with activation function $x_n = \tanh(a_n)$ can be viewed as *approximating* the probability distribution associated with that energy function,

$$P(\mathbf{x}\,|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})}\exp[-E(\mathbf{x})] = \frac{1}{Z(\mathbf{W})}\exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]. \tag{43.2}$$

These observations motivate the idea of working with a neural network model that actually *implements* the above probability distribution.

The *stochastic Hopfield network* or *Boltzmann machine* (Hinton and Sejnowski, 1986) has the following activity rule:

Activity rule of Boltzmann machine: after computing the activation a_i (42.3),

$$\text{set } x_i = +1 \text{ with probability } \frac{1}{1+e^{-2a_i}} \qquad \text{else set } x_i = -1. \tag{43.3}$$

This rule implements Gibbs sampling for the probability distribution (43.2).

Boltzmann machine learning

Given a set of examples $\{\mathbf{x}^{(n)}\}_1^N$ from the real world, we might be interested in adjusting the weights $\mathbf{W}$ such that the generative model

$$P(\mathbf{x}\,|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})}\exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right] \tag{43.4}$$

is well matched to those examples. We can derive a learning algorithm by writing down Bayes' theorem to obtain the posterior probability of the weights given the data:

$$P(\mathbf{W}\,|\,\{\mathbf{x}^{(n)}\}_1^N\}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] P(\mathbf{W})}{P(\{\mathbf{x}^{(n)}\}_1^N\})}. \tag{43.5}$$

We concentrate on the first term in the numerator, the likelihood, and derive a maximum likelihood algorithm (though there might be advantages in pursuing a full Bayesian approach as we did in the case of the single neuron). We differentiate the logarithm of the likelihood,

$$\ln\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)} \,|\, \mathbf{W})\right] = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{x}^{(n)\mathsf{T}}\mathbf{W}\mathbf{x}^{(n)} - \ln Z(\mathbf{W})\right], \tag{43.6}$$

with respect to w_{ij}, bearing in mind that $\mathbf{W}$ is defined to be symmetric with $w_{ji} = w_{ij}$.

Exercise 43.1.[2] Show that the derivative of $\ln Z(\mathbf{W})$ with respect to w_{ij} is

$$\frac{\partial}{\partial w_{ij}} \ln Z(\mathbf{W}) = \sum_{\mathbf{x}} x_i x_j P(\mathbf{x} \,|\, \mathbf{W}) = \langle x_i x_j \rangle_{P(\mathbf{x}\,|\,\mathbf{W})}. \tag{43.7}$$

[This exercise is similar to exercise 22.12 (p.307).]

The derivative of the log likelihood is therefore:

$$\frac{\partial}{\partial w_{ij}} \ln P(\{\mathbf{x}^{(n)}\}_1^N \,|\, \mathbf{W}) = \sum_{n=1}^{N}\left[x_i^{(n)} x_j^{(n)} - \langle x_i x_j \rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right] \tag{43.8}$$

$$= N\left[\langle x_i x_j \rangle_{\text{Data}} - \langle x_i x_j \rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right]. \tag{43.9}$$

This gradient is proportional to the difference of two terms. The first term is the *empirical* correlation between x_i and x_j,

$$\langle x_i x_j \rangle_{\text{Data}} \equiv \frac{1}{N}\sum_{n=1}^{N}\left[x_i^{(n)} x_j^{(n)}\right], \tag{43.10}$$

and the second term is the correlation between x_i and x_j under the current model,

$$\langle x_i x_j \rangle_{P(\mathbf{x}\,|\,\mathbf{W})} \equiv \sum_{\mathbf{x}} x_i x_j P(\mathbf{x} \,|\, \mathbf{W}). \tag{43.11}$$

The first correlation $\langle x_i x_j \rangle_{\text{Data}}$ is readily evaluated – it is just the empirical correlation between the activities in the real world. The second correlation, $\langle x_i x_j \rangle_{P(\mathbf{x}\,|\,\mathbf{W})}$, is not so easy to evaluate, but it can be estimated by Monte Carlo methods, that is, by observing the average value of $x_i x_j$ while the activity rule of the Boltzmann machine, equation (43.3), is iterated.

In the special case $\mathbf{W} = 0$, we can evaluate the gradient exactly because, by symmetry, the correlation $\langle x_i x_j \rangle_{P(\mathbf{x}\,|\,\mathbf{W})}$ must be zero. If the weights are adjusted by gradient descent with learning rate η, then, after one iteration, the weights will be

$$w_{ij} = \eta \sum_{n=1}^{N}\left[x_i^{(n)} x_j^{(n)}\right], \tag{43.12}$$

precisely the value of the weights given by the Hebb rule, equation (16.5), with which we trained the Hopfield network.

Interpretation of Boltzmann machine learning

One way of viewing the two terms in the gradient (43.9) is as 'waking' and 'sleeping' rules. While the network is 'awake', it measures the correlation between x_i and x_j in the real world, and weights are *increased* in proportion.

While the network is 'asleep', it 'dreams' about the world using the generative model (43.4), and measures the correlations between x_i and x_j in the model world; these correlations determine a proportional *decrease* in the weights. If the second-order correlations in the dream world match the correlations in the real world, then the two terms balance and the weights do not change.

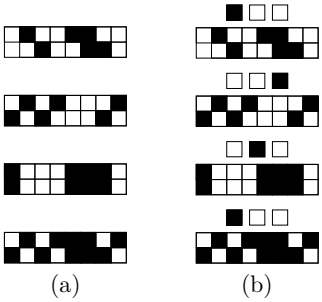

Figure 43.1. The 'shifter' ensembles. (a) Four samples from the plain shifter ensemble. (b) Four corresponding samples from the labelled shifter ensemble.

Criticism of Hopfield networks and simple Boltzmann machines

Up to this point we have discussed Hopfield networks and Boltzmann machines in which all of the neurons correspond to *visible* variables x_i. The result is a probabilistic model that, when optimized, can capture the second-order statistics of the environment. [The second-order statistics of an ensemble $P(\mathbf{x})$ are the expected values $\langle x_i x_j \rangle$ of all the pairwise products $x_i x_j$.] The real world, however, often has higher-order correlations that must be included if our description of it is to be effective. Often the second-order correlations in themselves may carry little or no useful information.

Consider, for example, the ensemble of binary images of chairs. We can imagine images of chairs with various designs – four-legged chairs, comfy chairs, chairs with five legs and wheels, wooden chairs, cushioned chairs, chairs with rockers instead of legs. A child can easily learn to distinguish these images from images of carrots and parrots. But I expect the second-order statistics of the raw data are useless for describing the ensemble. Second-order statistics only capture whether two pixels are likely to be in the same state as each other. Higher-order concepts are needed to make a good generative model of images of chairs.

A simpler ensemble of images in which high-order statistics are important is the 'shifter ensemble', which comes in two flavours. Figure 43.1a shows a few samples from the 'plain shifter ensemble'. In each image, the bottom eight pixels are a copy of the top eight pixels, either shifted one pixel to the left, or unshifted, or shifted one pixel to the right. (The top eight pixels are set at random.) This ensemble is a simple model of the visual signals from the two eyes arriving at early levels of the brain. The signals from the two eyes are similar to each other but may differ by small translations because of the varying depth of the visual world. This ensemble is simple to describe, but its second-order statistics convey no useful information. The correlation between one pixel and any of the three pixels above it is 1/3. The correlation between any other two pixels is zero.

Figure 43.1b shows a few samples from the 'labelled shifter ensemble'. Here, the problem has been made easier by including an extra three neurons that label the visual image as being an instance of either the 'shift left', 'no shift', or 'shift right' sub-ensemble. But with this extra information, the ensemble is still not learnable using second-order statistics alone. The second-order correlation between any label neuron and any image neuron is zero. We need models that can capture higher-order statistics of an environment.

So, how can we develop such models? One idea might be to create models that directly capture higher-order correlations, such as:

$$P'(\mathbf{x} \,|\, \mathbf{W}, \mathbf{V}, \ldots) = \frac{1}{Z'} \exp\left(\frac{1}{2} \sum_{ij} w_{ij} x_i x_j + \frac{1}{6} \sum_{ij} v_{ijk} x_i x_j x_k + \cdots \right). \tag{43.13}$$

Such *higher-order Boltzmann machines* are equally easy to simulate using stochastic updates, and the learning rule for the higher-order parameters v_{ijk} is equivalent to the learning rule for w_{ij}.

▷ Exercise 43.2.[2] Derive the gradient of the log likelihood with respect to v_{ijk}.

It is possible that the spines found on biological neurons are responsible for detecting correlations between small numbers of incoming signals. However, to capture statistics of high enough order to describe the ensemble of images of chairs well would require an unimaginable number of terms. To capture merely the fourth-order statistics in a 128×128 pixel image, we need more than 10^7 parameters.

So measuring moments of images is *not* a good way to describe their underlying structure. Perhaps what we need instead or in addition are *hidden variables*, also known to statisticians as *latent variables*. This is the important innovation introduced by Hinton and Sejnowski (1986). The idea is that the high-order correlations among the visible variables are described by including extra hidden variables and sticking to a model that has only second-order interactions between its variables; the hidden variables induce higher-order correlations between the visible variables.

▶ 43.2 Boltzmann machine with hidden units

We now add *hidden neurons* to our stochastic model. These are neurons that do not correspond to observed variables; they are free to play any role in the probabilistic model defined by equation (43.4). They might actually take on interpretable roles, effectively performing 'feature extraction'.

Learning in Boltzmann machines with hidden units

The activity rule of a Boltzmann machine with hidden units is identical to that of the original Boltzmann machine. The learning rule can again be derived by maximum likelihood, but now we need to take into account the fact that the states of the hidden units are unknown. We will denote the states of the visible units by $\mathbf{x}$, the states of the hidden units by $\mathbf{h}$, and the generic state of a neuron (either visible or hidden) by y_i, with $\mathbf{y} \equiv (\mathbf{x}, \mathbf{h})$. The state of the network when the visible neurons are clamped in state $\mathbf{x}^{(n)}$ is $\mathbf{y}^{(n)} \equiv (\mathbf{x}^{(n)}, \mathbf{h})$. The likelihood of $\mathbf{W}$ given a single data example $\mathbf{x}^{(n)}$ is

$$P(\mathbf{x}^{(n)} \,|\, \mathbf{W}) = \sum_{\mathbf{h}} P(\mathbf{x}^{(n)}, \mathbf{h} \,|\, \mathbf{W}) = \sum_{\mathbf{h}} \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}[\mathbf{y}^{(n)}]^{\mathsf{T}} \mathbf{W} \mathbf{y}^{(n)}\right], \tag{43.14}$$

where

$$Z(\mathbf{W}) = \sum_{\mathbf{x},\mathbf{h}} \exp\left[\frac{1}{2}\mathbf{y}^{\mathsf{T}} \mathbf{W} \mathbf{y}\right]. \tag{43.15}$$

Equation (43.14) may also be written

$$P(\mathbf{x}^{(n)} \,|\, \mathbf{W}) = \frac{Z_{\mathbf{x}^{(n)}}(\mathbf{W})}{Z(\mathbf{W})} \tag{43.16}$$

where

$$Z_{\mathbf{x}^{(n)}}(\mathbf{W}) = \sum_{\mathbf{h}} \exp\left[\frac{1}{2}[\mathbf{y}^{(n)}]^{\mathsf{T}} \mathbf{W} \mathbf{y}^{(n)}\right]. \tag{43.17}$$

Differentiating the likelihood as before, we find that the derivative with respect to any weight w_{ij} is again the difference between a 'waking' term and a 'sleeping' term,

$$\frac{\partial}{\partial w_{ij}} \ln P(\{\mathbf{x}^{(n)}\}_1^N \,|\, \mathbf{W}) = \sum_n \left\{ \langle y_i y_j \rangle_{P(\mathbf{h} \,|\, \mathbf{x}^{(n)}, \mathbf{W})} - \langle y_i y_j \rangle_{P(\mathbf{x},\mathbf{h} \,|\, \mathbf{W})} \right\}. \tag{43.18}$$

The first term $\langle y_i y_j \rangle_{P(\mathbf{h}|\mathbf{x}^{(n)},\mathbf{W})}$ is the correlation between y_i and y_j if the Boltzmann machine is simulated with the visible variables clamped to $\mathbf{x}^{(n)}$ and the hidden variables freely sampling from their conditional distribution.

The second term $\langle y_i y_j \rangle_{P(\mathbf{x},\mathbf{h}|\mathbf{W})}$ is the correlation between y_i and y_j when the Boltzmann machine generates samples from its model distribution.

Hinton and Sejnowski demonstrated that non-trivial ensembles such as the labelled shifter ensemble can be learned using a Boltzmann machine with hidden units. The hidden units take on the role of feature detectors that spot patterns likely to be associated with one of the three shifts.

The Boltzmann machine is time-consuming to simulate because the computation of the gradient of the log likelihood depends on taking the difference of two gradients, both found by Monte Carlo methods. So Boltzmann machines are not in widespread use. It is an area of active research to create models that embody the same capabilities using more efficient computations (Hinton *et al.*, 1995; Dayan *et al.*, 1995; Hinton and Ghahramani, 1997; Hinton, 2001; Hinton and Teh, 2001).

▶ 43.3 Exercise

▷ Exercise 43.3.[3] Can the 'bars and stripes' ensemble (figure 43.2) be learned by a Boltzmann machine with no hidden units? [You may be surprised!]

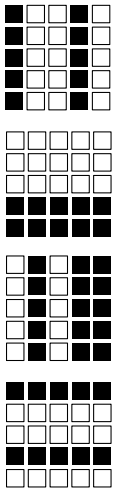

Figure 43.2. Four samples from the 'bars and stripes' ensemble. Each sample is generated by first picking an orientation, horizontal or vertical; then, for each row of spins in that orientation (each bar or stripe respectively), switching all spins on with probability $1/2$.