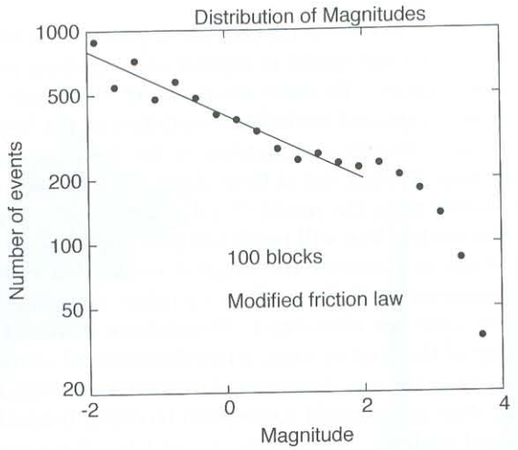

FIGURE 12.21: Earthquake distribution for a system of 100 blocks and the friction law described in Exercise 5. The straight line corresponds to the Gutenberg-Richter law (12.2) with $b \sim 0.35$.

with the frictional force (12.6). However, the slope for small $\mathcal{M}$ now corresponds to a value of b that is much smaller than 1, so the model still seems to lack an important ingredient. Study the behavior with other forms for the frictional force and try to determine one that gives a power law with a larger value of b.

12.3 NEURAL NETWORKS AND THE BRAIN

The Ising model consists of a large number of very simple units, that is, spins, which are connected together in a very simple manner. By "connected" we mean that the orientation of any given spin s_i, is influenced by the direction of other spins through the interaction energy Js_is_j. The behavior of an isolated spin, as outlined in our discussions leading up to mean-field theory, was unremarkable. Things only became really interesting when we considered the behavior of a large number of spins and allowed them to interact. In that case we found that under the appropriate conditions some remarkable things could occur, including the singular behavior associated with a phase transition. In this section we will explore a rather different system, which shares some of these features.

The human brain consists of an extremely large number ($\sim 10^{12}$) of basic units called neurons, each of which is connected to many other neurons in a relatively simple manner. A biologically complete discussion of neurons and how they function is a long story. Here we will give only a brief description of those features that seem to be most relevant to a physicist's understanding of the brain.A schematic picture of two neurons is given in Figure 12.22. Each neuron has a body (called a soma), along with dendrites and an axon.[23] The size scale depends on the type of neuron,

[23]There are other parts as well, but keep in mind that we are giving only a simplified description here.

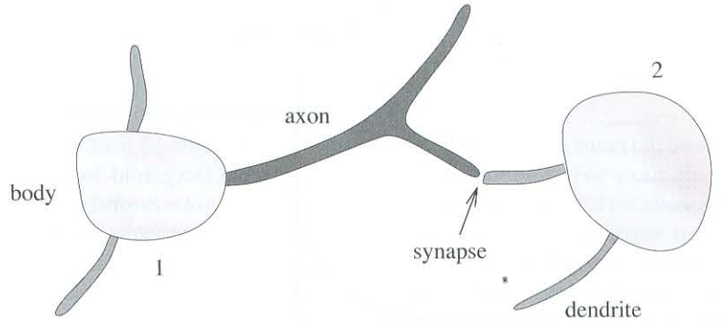

FIGURE 12.22: Schematic of two connected neurons.

but the body is typically of order 10 μm across. Neurons are electrically active and communicate with other neurons through electrical signals carried by the dendrites and axons. The dendrites serve as the input lines of a neuron, while the axons are output lines. The axon of one neuron is "connected" to the dendrite of another via a synapse, through which is transmitted the electrical output signal of one neuron to the input of the other.

The axon of the hypothetical neuron (number 1) on the left in Figure 12.22 connects to a dendrite of neuron number 2 via the synapse, as indicated. A very important feature is that an axon is generally split into many branches, and thereby connects to many other neurons. Correspondingly, each neuron generally has many dendrites and thus accepts inputs from many other neurons. When we say many, the number we have in mind is typically $\sim 10^4$. The neurons are thus highly interconnected.

Neurons communicate using electrical pulses carried by the axons and dendrites. These pulses are typically of order 10^{-3} s in duration and 5×10^{-2} V in magnitude. A neuron emits these pulses (this is called firing) in a roughly periodic manner, with the period being a function of the input signals experienced by the neuron at that moment. If the inputs are very active, that is if there are many pulses being received through the dendrites, the neuron will fire often. On the other hand, if its inputs are not active, the neuron will fire at a much lower rate. The picture is actually a bit more complicated than this. The interface where an axon and a dendrite meet (the synapse) may be either excitory or inhibitory. In the former case a high firing rate of the sending neuron will favor a high firing rate of the receiving neuron. With an inhibitory synapse, a high input firing rate will tend to cause a low firing rate of the receiving neuron.

The firing rate of a particular neuron is a function of all of its inputs. To a very rough approximation, this rate is determined by the sum of all inputs that come into a neuron through excitory synapses minus the total of the inputs that

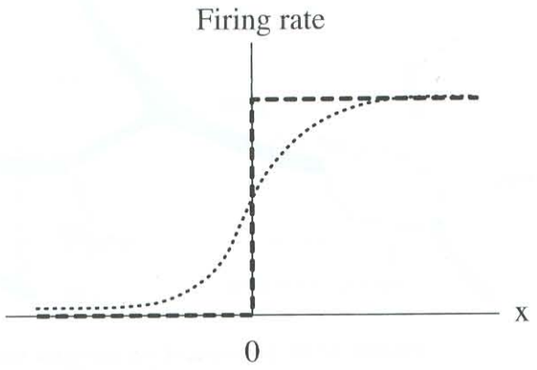

FIGURE 12.23: Firing rate function. The dotted curve is closer to that found in real neurons. The dashed curve is the step function used in our modeling.

enter through inhibitory synapses.[24] The firing rate, R, can thus be written as

$$R = f\left(\sum V_i\right) , \tag{12.13}$$

where V_i is the input signal from dendrite i (which may be either positive or negative). Experiments show that the firing function f is significantly nonlinear, as shown schematically in Figure 12.23. Because of this nonlinearity, a neuron is often modeled as a sort of on/off device.

A neural network is composed of a very large number of neurons whose basic properties we have just described. Such a network, that is, a brain, is perhaps the most remarkable object in nature, as it is responsible for the intelligence of living organisms.[25] The problem of understanding how a brain functions is of great concern to biologists, psychologists, computer scientists, and (believe it or not) physicists. It is useful to state this problem in the following way. The brain consists of a very large number of neurons that are highly interconnected. The connections are not limited to neighboring neurons, but can extend to neurons in remote areas of the brain. The state of a neuron can be described by its firing rate, which is a function of the firing rates of all of the neurons that have inputs to the neuron in question. The connectivity of the network and the strength of the connections vary from brain to brain. They vary from individual to individual (even within the same species, although there are many common features), and with time

[24] The synapses also vary in strength, so some signals contribute more strongly to this sum than others. We will ignore this slight complication here, although it will be included in the model we construct below.

[25] Some people (the authors included) believe that a brain is a neural network (essentially no more and no less), while others believe that a real brain contains some physics that is not contained in such a network. Arguments over this point have filled many books (but not this one). In any case, the majority of people on both sides of the argument would probably agree that neural networks are a major component of biological brains, and this is enough to make them worth studying.

for a given individual. On a very coarse scale, brains seem to possess a significant degree of randomness, yet we know that they are capable of very precise behavior, such as logical reasoning and memory. Understanding how such an arrangement of neurons can actually function as a brain is a key problem.

In this section we will consider how a neural network can function as a memory. Many aspects of biological memories[26] are not understood. For example, it is not yet known how information is stored (and forgotten) or how it is recalled. However, some ideas from physics have contributed greatly to recent progress in this area, and the answers to these questions may not be far away. Rather than try to give a logical or historical derivation (or justification) of the model of memory that we will study, it is simpler to just plunge ahead, and that is what we will now do. A little bit of the history of this model will be given at the end of this section.

We will model a neuron as a simple Ising spin. As we saw in Chapter 8, such a spin has two possible states, up and down. We will therefore assume that a neuron also has only two possible states, firing or not firing. This will, among other things, mean that we will approximate the firing function by a step function (the dashed curve in Figure 12.23), which has only two possible values. These two values will correspond to the two possible states of an Ising spin, $s = \pm 1$. The value of s_i corresponds to the current firing rate of neuron i. By convention we take $s_i = +1$ to correspond to a firing rate of 1, and $s_i = -1$ to 0 firing rate.[27] By associating the firing rate with the value of the spin we have glossed over the time dependence of the synaptic signals. At first sight this might seem to be a rather drastic approximation, but there are good reasons for believing that the timing of synaptic pulses does not play an important role in real neural networks. First, it is known from experiments that the firing times of different neurons are generally not correlated. That is, the brain does not appear to operate like a conventional computer in which all neuronal firing occurs in synchrony with a master clock. Second, the interneuronal signals travel at a speed of ~ 1 m/s, so the propagation delay (which is different for each pair of connected neurons) makes a significant contribution to the pulse arrival times. For these reasons it is generally argued that the *timing* of the pulses does not play an essential role.[28] Rather, the average arrival *rate* seems to be an essential feature, and this can be modeled with simple Ising spin variables.

In the Ising model studied in Chapter 8, the effect of a spin on its neighbors was through the exchange energy, and for our neural network we can model the effect of one neuron on another in a similar way. We will assume that our spins (that is neurons; we will use the two terms interchangeably) experience an exchange interaction, but now we will permit an interaction between *every* pair of spins. This

[26] As compared to computer memories.

[27] We could have chosen the pseudospin values to be $+1$ and 0 so as to correspond directly with the firing rate. For the way we have chosen to write the energy this would contribute a constant term to E and would not affect the behavior of the model.

[28] While this seems to be the current conventional wisdom, it has recently been suggested that correlated firing times are essential for understanding certain types of computations carried out by the brain. While the verdict has not yet been reached on this issue, it seems safe to conclude that much (if not all) of the brain's operation, including the memory functions we consider in this section, does not depend on precise timing of neuronal firing.

is necessary in order to mimic the highly interconnected nature of a real neural net, which is believed to be crucial for its operation. We will find it very useful to consider the effective energy of our neural network/spin system, which can be written as

$$E = -\sum_{i,j} J_{i,j}\, s_i\, s_j \ , \tag{12.14}$$

where the $J_{i,j}$ are related to the strengths of the synaptic connections, as we will describe shortly. The sum here is over all pairs of spins i and j in the network. The exchange, or more properly the synaptic, energies $J_{i,j}$ describe the influence of neuron i on the firing rate of neuron j. One way to view (12.14) is as follows. The sum of the synaptic inputs to neuron i will be $\sum_j J_{i,j} s_j$, and this will determine the firing rate, that is, the *direction* of spin i. If this input is negative, then according to our model it will be energetically preferred for neuron i to have a firing rate that is also negative, $s_i = -1$. If the synaptic input is positive, $s_i = +1$ will be favored. This is precisely analogous to an Ising spin model since as we saw in Chapter 8, each Ising spin prefers to point in the direction of the effective field established by the exchange energy due to its neighboring spins. Hence, our neural network can be described by the energy function (12.14), in close correspondence with the behavior of an Ising magnet. Individual neurons will prefer to fire at rates determined by the effective fields established by the interactions with other neurons, and this will make the network prefer states that minimize this energy.

As implied by Figure 12.22, the connections in a real neural network are *not* symmetric. The connection of the axon from neuron i to a dendrite of neuron j is completely separate from the connection between the axon of neuron j and a dendrite of neuron i. Indeed, there may be a connection in only one direction. The biological importance of this asymmetry in $J_{i,j}$ is not known. For now we will assume that the connections are symmetric, as this will allow us to make use of some ideas and results from statistical mechanics. The more realistic asymmetric case will be explored in the exercises.

The energy function (12.14) enables us to model our neural net as an Ising model, which can be simulated using the Monte Carlo method. However, we have not yet specified how the exchange energies should be chosen, or even how our lattice of spins can function as a memory. A useful memory must be able to store, recall, and display patterns, so let us consider how these operations can be implemented. The display operation is the easiest and is illustrated in Figure 12.24. On the left we show a lattice of spins with a particular spin configuration. This configuration was chosen so that it stores the letter A, which is seen more clearly on the right, where we show the same lattice, but with the spins for which $s = -1$ replaced by blanks. In this way the spins can be used as pixels to display whatever character or other type of pattern is desired.

In order for our memory to recall a pattern, we require that the spin directions change with time in such a way that the spin configuration eventually ends up in the desired state. For example, if we want to recall the letter A, we want the system to take on the configuration in Figure 12.24. The recall process thus involves first giving the spins a configuration, then allowing the spin arrangement to evolve with time until the system settles into a new, and we hope stable, configuration.

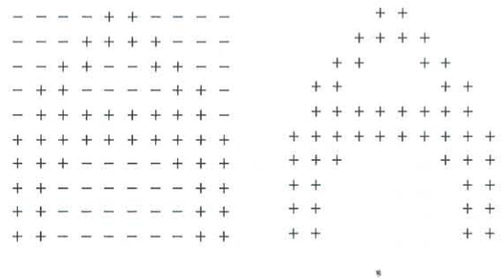

FIGURE 12.24: Left: a 10×10 lattice of spins with a particular configuration of $+$ and $-$ spins; right: the same lattice but with the spins for which $s = -1$ replaced by blanks. This network holds (i.e., displays) the letter A.

This time evolution is accomplished using an algorithm similar to the Monte Carlo method employed in Chapter 8. Starting from some particular configuration of the entire network, a spin is chosen and the energy required to flip it, ΔE_{flip}, is calculated using (12.14). If ΔE_{flip} is negative, that is, if flipping the spin would lower the energy, the spin is reversed. If $\Delta E_{\text{flip}} \geq 0$, the spin is left unchanged. This is precisely the Monte Carlo flipping rule employed in our work on the Ising model, with the assumption that the effective temperature is zero. We will have more to say about temperature and what it means in this problem a little later. For now, the important point is that the Monte Carlo rules ensure that the system will always evolve in time to states with the same or lower energy. If conditions are right, the memory will end up in a state in which the spins cease changing with time. This state corresponds to the pattern that is recalled by our memory.

The Monte Carlo rules, together with the energy function, determine how the spin system changes with time, and it is instructive to observe this directly in a simulation. Here, and with all of the other simulations below, we have used a 10×10 lattice of spins. For the moment we will not worry about how the interaction energies $J_{i,j}$ are determined. While their values are central to the behavior, it is simplest to postpone a discussion of how to calculate them until later. We are thus "given" a neural network, which in our specific case is a lattice of 100 spins interacting according to a certain collection of interactions $J_{i,j}$. We have, of course, chosen these interactions to yield some interesting behavior! On the left side of Figure 12.25 we show our lattice in a state that is close to the pattern for the letter A that we considered earlier. Even though a few of the spins are in the wrong state, that is, have been flipped with respect to Figure 12.24, it should be clear to your (human) brain that this pattern represents the letter A. We started a simulation with the lattice in this configuration, and after one pass through the lattice we obtained the state shown on the right side of Figure 12.25. Subsequent Monte Carlo sweeps through the lattice produced no further changes. The system thus found the ideal letter A, and recalled it. The memory worked as it should.

An important feature of human brains is that they are able to generalize in a reasonable manner. For example, you can usually recognize a letter even if it

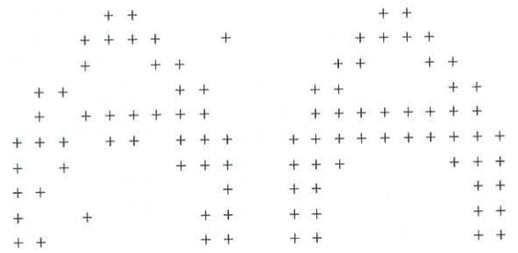

FIGURE 12.25: Left: spin arrangement close to the letter A, but with a few spins flipped to the wrong state; right: spin configuration after one Monte Carlo pass through the lattice.

is shown to *you* in *a* **different** TYPE face or $\mathcal{STYLE}$. There is some property of A-ness that you are able to recognize, and all patterns that fall into this class will cause you to recall the same fundamental letter A. Our model exhibits similar behavior. The left side of Figure 12.26 shows the letter A in outline form. When we initialized our lattice of spins with this pattern, one Monte Carlo sweep through the lattice yielded the ideal letter A shown on the right. At least with this simple test of A-ness, the model responded correctly.[29]

A similar test is shown in Figure 12.27, where the left side shows a damaged version of the ideal A. Here we have flipped 20% of the spins at random, and the resulting pattern only faintly resembled the ideal one. Nevertheless, after one Monte Carlo sweep through the lattice, the ideal pattern was recovered.

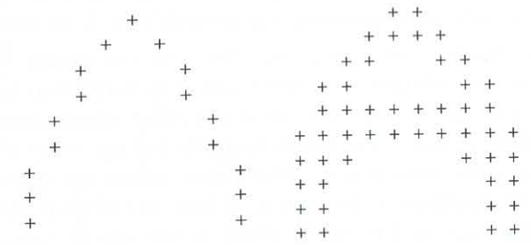

FIGURE 12.26: Left: spin arrangement that shows the letter A, but with a different pattern (that is, font) than used previously; right: spin configuration after one Monte Carlo pass through the lattice.

At this point you might suspect that we are playing a joke, as our model memory seems to recognize *every* pattern as the letter A. To demonstrate that this is not the case, we show in Figures 12.28 and 12.29 the results obtained when the lattice was initially given a configuration that resembled the letters B and C,

[29] We will leave the more challenging problems of a letter that is rotated or rescaled in size to the inquisitive reader.

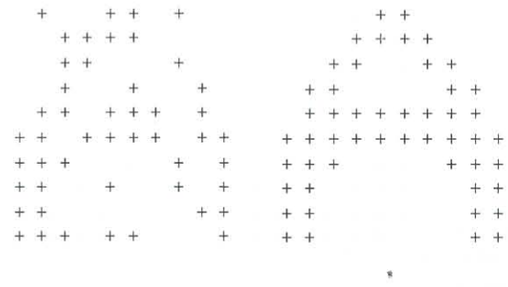

FIGURE 12.27: Left: spin arrangement that shows the letter A, but with 20% of the spins flipped at random; right: spin configuration after one Monte Carlo pass through the lattice.

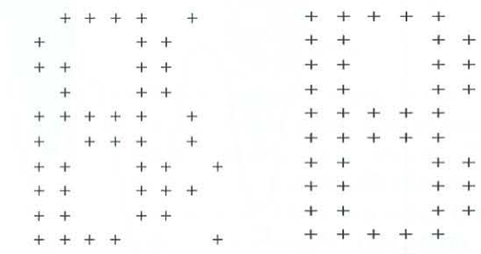

FIGURE 12.28: Left: spin arrangement that shows the letter B, but with 10% of the spins flipped at random; right: spin configuration after one Monte Carlo pass through the lattice.

respectively. In each case the initial patterns were barely recognizable, but after one sweep through the lattice the ideal patterns were recovered. Our model memory was thus able to recognize several different letters.

Observing the model in operation raises several important points and a few questions. First, the model is seen to operate as a *content addressable* memory. This is in contrast to the type of memory we are familiar with in connection with a conventional computer. In the case of a computer memory, a piece of information, such as the value of a particular variable, is given a *name* or address. In order to recall the value of the variable, the address must be presented to the memory, which is then able to retrieve the desired value. However, with a content addressable memory, information is retrieved by giving a rough description of the information itself. For example, the value of π could be retrieved by giving only the first few digits. This is a very important property of human brains. As another example, we might want to recall a friend's face given only a vague recollection of the shape of her chin and her hair style. In addition, such a memory should be able to recognize her face even if she cuts her hair or dyes it green. Human brains are able to handle such tasks, which are difficult for the more conventional computer-style memories.

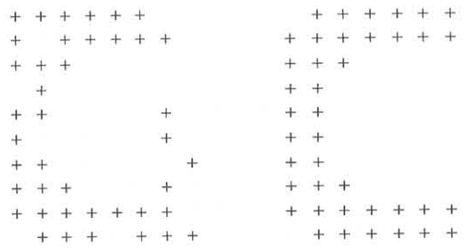

FIGURE 12.29: Left: spin arrangement that shows the letter C, but with 10% of the spins flipped at random; right: spin configuration after one Monte Carlo pass through the lattice.

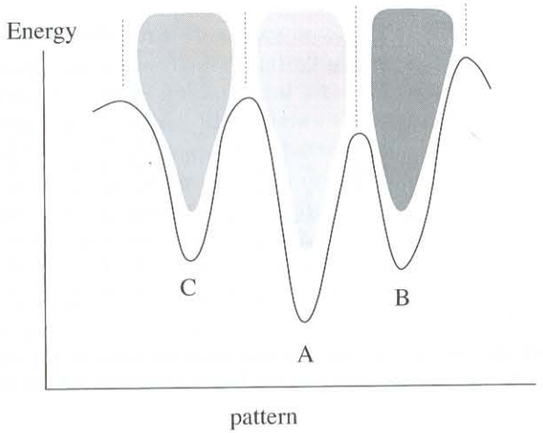

FIGURE 12.30: Schematic energy landscape. The vertical axis is energy as calculated from (12.14), while the horizontal axis corresponds to the spin configuration. The basins of attraction of several stored patterns are indicated by the shaded areas.

Second, we have seen that our memory can generalize in a reasonable manner. When presented with a pattern that is similar but not identical to one that it "knows," the memory is able to choose the correct response in a reasonable way. This behavior can be understood in terms of the energy landscape of the spin system. The energy of the network depends on the specific spin configuration, that is, memory pattern, at that time. Since the interactions $J_{i,j}$ (which we promise to explain shortly) have widely varying magnitudes and signs, this energy function will depend in a complicated way on the spin configuration. Schematically it will resemble the landscape shown in Figure 12.30. Each stored pattern, the letters A, B, and C in the example we have just considered, corresponds to minima in the energy. The Monte Carlo procedure guarantees that given an initial spin configuration, that is, some initial pattern, the system will evolve into a pattern with the same or, most likely, lower energy.

In general, the pattern presented to the network will not be one of the stored patterns, so the system will initially be situated on one of the slopes in the energy landscape. The Monte Carlo flipping rules then take it to the bottom of the nearest valley, and it thus ends up "recalling" the stored pattern associated with that minima. But what do we mean when we say that an initial pattern is close to a stored pattern? There are two ways to answer this question. In terms of the energy landscape there will be a *basin of attraction* in the neighborhood of each minima. If the network is within the basin corresponding to a particular minima, the associated stored pattern is the one that it will eventually locate. Topologically, these basins are the "valleys" surrounding each minima in Figure 12.30.

A quantitative answer to this question requires that we be a little more specific concerning the horizontal axis in Figure 12.30. A useful measure of the distance between two patterns is

$$\Delta_{m,n} = \frac{1}{N} \sum_i [s_i(m) - s_i(n)]^2 , \tag{12.15}$$

where m and n denote two different patterns, $s_i(m)$ is the configuration of spin i in pattern m, and N is the total number of spins. This is usually referred to as the *Hamming distance* between the two states of the system. Two patterns that are separated by a large Hamming distance will usually flow to different final stored patterns, while if the separation is small, there is a much better chance that they will flow to the same final state. For example, the letters C and O have many features in common, and will thus be closer in pattern space than are the letters A and V. Our memory will generally do a better job if the stored patterns are as different as possible.

While the Hamming distance is a quantitative measure of how similar two patterns are, it is difficult to be more precise in predicting how a particular network will respond to patterns that differ only slightly. This difficulty can again be traced to the energy landscape. Two patterns might be extremely similar, but if they happen to fall on opposite sides of an energy maxima they will flow to different final states.

The interaction energies $J_{i,j}$ are, as we have noted several times, key elements in our model memory. We are now ready to discuss how they should be chosen so as to yield a useful memory. We again let $s_i(m)$ denote the configuration of spin i in pattern m, and assume that we want our network to have this configuration as one of its stored patterns. That means that the $J_{i,j}$ must be chosen so as to make the energy a minima for this spin configuration. While there is no unique solution for this problem, a popular and very convenient choice is

$$J_{i,j} = s_i(m)\, s_j(m) , \tag{12.16}$$

which is often associated with the names of Hebb and Cooper, two important researchers in this area.[30] We can see that this choice for $J_{i,j}$ will make $s_i(m)$ an

[30] Their work is described by Hertz, Krogh, and Palmer (1991).

energetically stable pattern as follows. Inserting (12.16) into (12.14) yields

$$E(m) = -\sum_{i,j} J_{i,j} s_i s_j = -\sum_{i,j} s_j(m)\, s_i(m)\, s_i\, s_j \,. \tag{12.17}$$

If the network is in pattern m, we will have $s_i = s_i(m)$ and $s_j = s_j(m)$, so each term in this sum will be unity, making the energy large and negative. On the other hand, a random pattern (we will be more precise about this in a moment) will, on average, have half of its spins flipped with respect to $s_i(m)$, so roughly half of the terms in (12.17) will be positive and half negative, leading to $E \sim 0$. Thus, our desired pattern will have a much lower energy than a random pattern. Furthermore, the energy of our stored pattern will correspond to a stable minima of the landscape, since flipping any one spin from the pattern value $s_i(m)$ will increase E.

The prescription (12.16) tells us how to store a single pattern, but a useful memory must be able to store many patterns. In that case we choose the $J_{i,j}$ according to

$$J_{i,j} = \frac{1}{M} \sum_{m} s_i(m)\, s_j(m) \,, \tag{12.18}$$

where the index m refers to the stored patterns, and there are a total of M such patterns (we will see in a moment that there is a limit on the total number of patterns that can be stored). The arguments we just gave can be used to show that the energy associated with each stored pattern will be much lower than the energies of a random pattern. In addition, *if* the stored patterns are sufficiently different from one another, they will each correspond to distinct, well-separated minima in the energy landscape. This is how the values of $J_{i,j}$ were chosen in the calculations described above. We used the letters A, B, and C as our stored patterns, and calculated the energies $J_{i,j}$ using (12.18).

We have now completed the description of our neural network memory model. The key features are:

- An Ising spin is used to model the on/off behavior of a neuron. The firing rate of a neuron is assumed to have only two possible states, corresponding to the spin values $s = \pm 1$.
- The connections between the spins $J_{i,j}$ are not limited to nearest neighbors, but link all pairs of spins in the network.
- Given a collection of patterns that we want to store, the $J_{i,j}$ are calculated according to (12.18).
- The network operates as a content addressable memory. The lattice of neurons (that is, spins) is initialized with a configuration that resembles the pattern we want to recall. The Monte Carlo rules for $T = 0$ are then used, and the system evolves to a pattern that is at a minima in the energy landscape. This is the pattern that is "recalled" by the network.

The programming of this procedure is similar to the Monte Carlo routines described in Chapter 8, so we will leave the details to the exercises. However, we will give a few tips later in this section.

The modeling of neural networks is a vast industry, and we have been able to touch on only a very small piece of it here. A brief discussion of several other aspects of this field and some historical notes are given at the end of this section. However, before finishing we want to touch on a few issues associated with the choice of $J_{i,j}$ and the desired stored memories. Our network contains N spins, and since each pair is connected, there are $\sim N^2$ different values of the interaction energies $J_{i,j}$. We have described a procedure for storing a collection of patterns, and it should be clear that this information is stored *in* the $J_{i,j}$. This brings up several questions. (1) How many different patterns can be stored? (2) Can we add the concept of "learning" to the model, so as to bring it closer to the operation of a biological memory? (3) What happens if the $J_{i,j}$ are damaged in some way? We know that real memories can function even if some of the neurons die. Is our model able to function in the presence of such damage?

Each pattern involves the configuration of N spins and there are $\sim N^2$ different $J_{i,j}$, so the maximum amount of information that can be stored is of order $\sim N^2/N = N$ different patterns.[31] However, it turns out that the maximum number of stored patterns is much less than this for several reasons. First, if two desired stored patterns happen to be close to each other (in terms of their Hamming distance), they can interfere with one another. This may make one of the patterns unstable, or cause it to be recalled in a distorted (imperfect) form. This problem can be minimized by choosing the stored patterns to be orthogonal to one another to the fullest extent possible. Here orthogonal means that $\sum_i s_i(m) s_i(n) \sim 0$ if $m \neq n$. In our simple example involving letters such orthogonality was not under our control, as it was determined by the shapes of the letters. While we can often preprocess the patterns to make them orthogonal before encoding them in the network, this may not always be convenient in cases such as, for example, the storage of arbitrary optical images.

The use of orthogonal stored patterns can increase the storage capacity somewhat, but it turns out that there is a more fundamental limit. As we use more and more patterns in the calculation of $J_{i,j}$, the nature of the energy landscape changes. Increasing the number of stored patterns means that more and more local minima must be crowded together, and this also affects the depth of each minima. It has been found that if the number of stored patterns exceeds $\sim 0.13N$, the landscape changes dramatically. In this case all of the stored patterns become unstable, and the system ceases to function as a memory. This abrupt change is actually a type of phase transition and has much in common with a system known as a *spin glass*. This is a type of spin system in which the interactions are randomly distributed and which have been studied extensively by physicists over the past 30 years.[32] We do not have time here to say anything about spin glasses or how they are related to neural networks, other than to note that this is a very nice example of how

[31] This simple argument ignores the fact that the spins are binary objects, while the $J_{i,j}$ are real numbers, so this may be an underestimate of the capacity of the network.

[32] In fact, spin glasses were studied long before it was appreciated that they had anything in common with neural networks. It turns out that spin glasses are typical of random systems with energy landscapes given by functions such as (12.14). The protein-folding problem falls into this class as well.

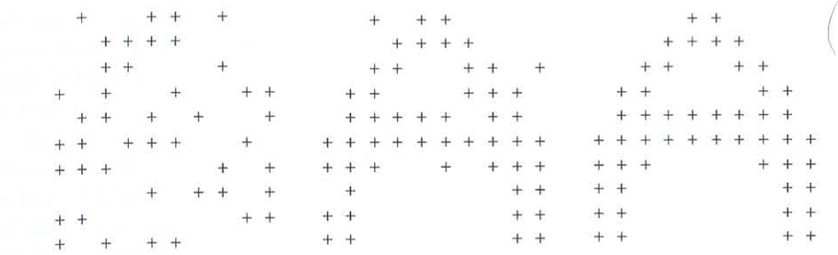

FIGURE 12.31: Operation with 80% of the $J_{i,j}$ set to zero. Left: initial pattern; middle: spin configuration after one Monte Carlo sweep through the lattice; right: configuration after two sweeps.

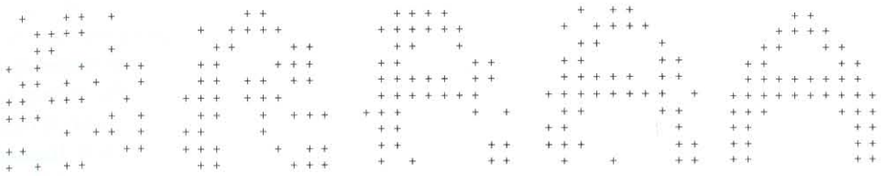

FIGURE 12.32: Operation with 90% of the $J_{i,j}$ set to zero. Left to right: initial pattern and spin configurations after one, two, five, and nine Monte Carlo sweeps through the lattice.

ideas developed for one problem (spin glasses) can be profitably applied to a rather different area.

We next consider the effects of damage on the operation of our model memory. Since the information is stored in the interactions $J_{i,j}$, it is these connection values that will be damaged. We start with the $J_{i,j}$ calculated using (12.18) and the three stored patterns, A, B, and C. We then "damage" these values by randomly setting the elements of the $J_{i,j}$ matrix to zero with probability p_{damage}. Some results for $p_{\text{damage}} = 0.8$ are shown in Figure 12.31, where the initial pattern was an A, which itself has been altered from the stored pattern with probability 0.3. We recall that with the ideal $J_{i,j}$, our memory immediately found the correct stored pattern; that is, it took only one Monte Carlo pass through the lattice to obtain the stored pattern, which was then stable for all future times. With $p_{\text{damage}} = 0.8$ and after one Monte Carlo time step (one complete pass through the lattice), the system was closer to the stored pattern, but there were still some spins pointing in the wrong direction. However, after two time steps the system reached the correct, and perfect, final pattern.

The corresponding result with $p_{\text{damage}} = 0.9$ is shown in Figure 12.32. In this case it took nine Monte Carlo time steps to reach a stable pattern, but the system did eventually come to the correct final state. However, if the $J_{i,j}$ are damaged much further, the network fails, as illustrated in Figure 12.33. Here, with $p_{\text{damage}} = 0.95$, the system was unable to find the correct pattern. Instead, it ended up in a state unlike any of the stored patterns.

These simulations show that our model memory continues to function, even when severely damaged. It thus has an important feature in common with human

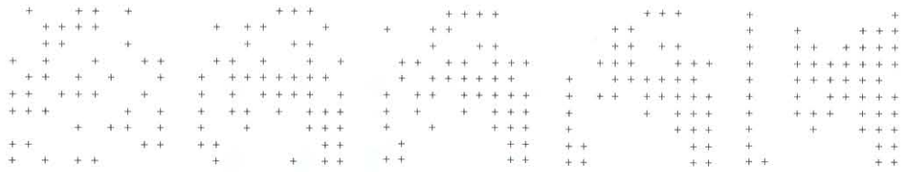

FIGURE 12.33: Operation with 95% of the $J_{i,j}$ set to zero. Left to right: initial pattern and spin configurations after three, four, five, and ten Monte Carlo sweeps through the lattice. In this case our memory did not recall the letter A.

brains. Indeed, it is amazing that the memory works so well, even with 90% of the connections removed. The level of damage that can be tolerated depends on the number of stored patterns. In this example, with only three stored patterns in a system of 100 neurons, we were well below the ideal theoretical limit of ~ 13 patterns. If the number of stored patterns had been closer to this limit, the level of damage that could have been be tolerated would have been smaller. Nevertheless, neural network memories can continue to function well even when significantly damaged. Intuitively, this can be understood in terms of their redundancy. With a network that contains the theoretical limit of $\sim 0.13N$ patterns, $\sim 87\%$ of its information content is effectively redundant. A network's operation may not be adversely affected if some of this redundant information is lost.

One aspect of neural networks that we haven't really touched on is learning. The problem is extremely important for two reasons. First, if our goal is to model a real (biological) brain, learning must be accounted for. Second, neural networks of the type described here are being considered for a number of applications, such as image processing and handwriting recognition. These applications require that the network be able to incorporate new patterns, and in some cases forget old ones. The development of learning algorithms is an active area of research, and a number of alternative ways for calculating the $J_{i,j}$ have been proposed. One of the simplest learning procedures is based on (12.18). Consider a working network that contains several stored patterns. A new pattern can be learned by adding a small contribution to the interactions

$$J_{i,j}(\text{new}) \;=\; \beta\, J_{i,j}(\text{old}) \;+\; \alpha\, s_i(p) s_j(p)\;, \tag{12.19}$$

where $s_i(p)$ is the new pattern, α is a parameter that controls how fast the learning should occur, and the value of β can be adjusted to allow for the fading of old memories. This learning algorithm has also been proposed on biological grounds. If a real network is presented with a pattern $s_i(p)$, it is believed that the synapses can adjust their strengths with time so as to make the pattern a stable configuration of the network. That is, the strengths and signs of a synapse adjust so that they are "consistent" with the states of the neurons that are involved. We will explore the behavior of this and several related learning procedures in the exercises.

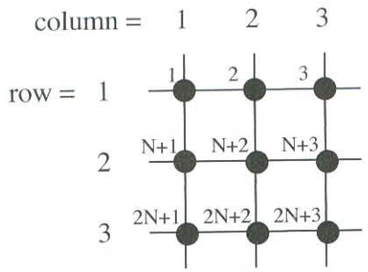

FIGURE 12.34: Numbering scheme. N is the number of spins in one row.

PROGRAMMING NOTES

The simulation of a neural network memory involves the following steps.

- The desired stored patterns must be chosen. For the example calculations described above these were just letters laid out on a 10×10 grid, but you could use any other patterns including other characters or pictures.[33] You could also use a larger array of spins. This would allow more detail in the patterns and enable the system to store more patterns, but at the cost of additional memory requirements (memory in your program).

- A natural way to store the spin values is to use a two-dimensional array `spin(m,n)` where the indices `m` and `n` specify the row and column where the spin is located. This storage scheme is convenient for displaying the stored pattern. However, we also have to store the interaction energies $J_{i,j}$. Here the indices i and j do *not* refer to rows and columns. You will recall that our model allows an interaction between each pair of spins. Hence, $J_{i,j}$ is the strength of the interaction between the ith and jth spins in the lattice, where i and j refer to a particular numbering scheme. One possible scheme is illustrated in Figure 12.34. The spins in the first row (`m=1`) are numbered $i = 1, 2, \ldots, N$, those in the second row (`m=2`) have the numbers $i = N+1, N+2, \ldots, 2N$, and so on up to the final spin, which is number $i = N^2$. Given the row and column numbers, `m` and `n`, the spin number is

$$i = N(\mathtt{m} - 1) + \mathtt{n} \,. \tag{12.20}$$

Note that this mapping can be inverted, that is, given i we can uniquely determine `m` and `n`, although we will not need to do this in our simulations.

The stored patterns can be described using the labeling scheme in terms of spin number i, with $s_i(p)$ denoting the value of the i-th spin for the p-th stored pattern. These in turn are used to calculate the interaction energies $J_{i,j}$ according to (12.18). Storage of the interaction energies requires a two dimensional array of size $N^2 \times N^2$. Thus, a 10×10 array of spins requires $\sim 10^4$ interaction energies.[34]

[33] Black-and-white pictures would be simplest. We will leave gray scale or color images to the industrious reader.

[34] An alternative scheme for storing the $J_{i,j}$ would be to use a four-dimensional array `J(m1,n1,m2,n2)` to store the interaction energy for `spin(m1,n1)` and `spin(m2,n2)`. This makes the bookkeeping simpler than the scheme involving (12.20), but some computer languages do not permit four-dimensional arrays. In addition, the method (12.20) for encoding two numbers into one in an invertible manner can be useful in other situations.

- After the interaction energies have been calculated, the memory is ready for operation. The spins are given values corresponding to a particular pattern. In our examples this was a character that was similar, though not identical, to a letter in the alphabet, but it could be a picture or other type of image. The Monte Carlo method is then used to calculate the spin directions at future times. The method is very similar to the one used to simulate the Ising model in Chapter 8. A spin is selected and the energy required to make it flip, E_{flip} is calculated from (12.18). If E_{flip} is negative, that is, if the energy would be reduced by flipping the spin, then the spin is reversed. If $E_{\text{flip}} \geq 0$, the spin is left unchanged. These flipping rules correspond to a system in thermal equilibrium with a heat bath at zero temperature. In this case the Monte Carlo procedure takes the system to the nearest energy minima, in the spirit of the energy landscape in Figure 12.30.

- The Monte Carlo procedure is used repeatedly, giving every spin a chance to flip. The spins can be chosen at random, or they can be selected by systematically moving through each row and column of the lattice. The two choices both work well. The most important thing is that *each* spin be given at least one chance to flip, so the systematic approach is often preferred. The Monte Carlo procedure is used until the state of the spin system becomes stable, indicating the pattern that is recalled by the memory. A network will usually find the desired pattern after only one or two Monte Carlo passes through the lattice, as the system will quickly locate the nearest energy minima. However, the behavior can sometimes be more complicated. If the number of stored patterns is close to the theoretical limit, or if two of the stored patterns are similar (have a small Hamming separation), the memory recall may not be ideal. The system may then recall a pattern that is different from any of the stored patterns (often an approximate mixture), or the network might never locate a time independent state. In the latter case it may switch back and forth in time between two or more patterns; such behavior is known as a limit cycle. In human terms we might think of this as corresponding to confusion!

Historical Notes

The field of neural networks is much too vast for us to be able to give a complete description in this section. A good historical discussion of the field is given in Hertz, Krogh, and Palmer (1991) which is listed in the references. In 1943 McCulloch and Pitts recognized that a network of simple neurons was capable of universal computation. This means that such a network could, in principle, perform any calculation that could be carried out with the most general computer imaginable.[35] This attracted a good deal of interest from researchers interested in modeling the brain.

[35] More precisely, such a network can calculate any computable function in the sense of a general purpose Turing machine.