6: Hopfield nets (contd.)

Kevin Gurney

Dept. Human Sciences, Brunel University
Uxbridge, Middx. UK

1 Finding the weights

So far we have described the dynamics of Hopfield nets but nothing has been said about the way the weights are established for a particular problem. In his original paper, Hopfield (1982) did not give a method for training the nets, rather he gave a *prescription* for making a weight set, given a set of patterns to be stored. Here, we shall relate the storage prescription, later on, to a biologically inspired learning rule - the Hebb rule - and show that the nodes may also be trained individually using the delta rule.

1.1 The storage prescription

The rationale behind the prescription is based on the desire to capture, in the value of the weights, local correlations between node outputs when the net is in one of the required stable states. Recall that these correlations also gave rise to the energy description and the same kind of arguments will be used again.

Consider two nodes which, on average over the required pattern set, tend to take on the same value. That is, they tend to form either the pair (0, 0) or (1, 1). The latter pairing will be reinforced by there being a positive weight between the nodes, since each one is then making a positive contribution to the others activation which will tend to foster the production of a '1' at the output. Now suppose that the two nodes, on average, tend to take on opposite values. That is they tend to form either the pair (0, 1) or (1, 0). Both pairings are reinforced by a negative weight between the two nodes, since there is a negative contribution to the activation of the node which is 'off' from the node which is 'on', supporting the former's output state of '0'. Note that, although the pairing (0, 0) is not actively supported by a positive weight *per se*, a negative weight would support the mixed output pair-type just discussed.

These observations may be encapsulated mathematically in the following way. First we introduce an alternative way of representing binary quantities. Normally these have been denoted by 0 or 1. In the *polarised* or *spin*[*] representation they are denoted by -1 and 1 respectively, so there is the correspondence $0 \leftrightarrow -1, 1 \leftrightarrow 1$. Now let v_i^p, v_j^p be components

[*]This name is derived from the fact that Hopfield nets have many similarities with so-called *spin glasses* in physics, the prototypical example of which is a collection of magnetic domains whose polarisation of ± 1 is determined by the average spin of the electrons in each domain

of the pth pattern to be stored where these are in the spin representation. Consider what happens if the weight between the nodes i and j is given by

$$w_ij = \sum_p v_i^p v_j^p \tag{1}$$

Where the sum is over all patterns p to be stored. If, on average, the two components take on the same value then the weight will be positive since we get terms like 1×1 and -1×-1 predominating. If, on the other hand, the two components, on average, take on opposite values we get terms like -1×1 and 1×-1 predominating which gives a negative weight. This is just what was required according to the arguments given above. Equation (1) is therefore the storage prescription used with Hopfield nets. Note that, the same weights would accrue if we had tried to learn the inverse of the patterns formed by taking each component of every pattern and changing it to the opposite value. The net therefore, always learns the patterns *and* their inverses.

1.2 The Hebb rule

The use of (1) which is an algorithm or recipe for fixing the weights without adapting to a training set may appear to run counter to the ideas being promoted in the connectionist cause. It is possible, however, to view the prescription in (1) as a short-cut to a process of adaptation which would take place if we were to obey the following training algorithm

1. present the components of one of the patterns to be stored at the outputs of the corresponding nodes of the net.
2. If two nodes have the same value then make a small positive increment to the internode weight. If they have opposite values then make a small negative decrement to the weight.

Steps 1) and 2) then get repeated many times with a different pattern selection in 1). Symbolically step 2) (which is the learning rule) may be written

$$\Delta w_{ij} = \alpha v_i^p v_j^p \tag{2}$$

where, as usual α is a rate constant and $0 < \alpha < 1$. It is clear that the storage prescription is just the result of adding all the weight changes that would accumulate under this scheme if enough pattern presentations were made. The rule in (2) is one of a family of rules known as *Hebb rules* after D. O. Hebb. The characteristic of such a rule is that it involves the product of a pair of node activations or outputs.

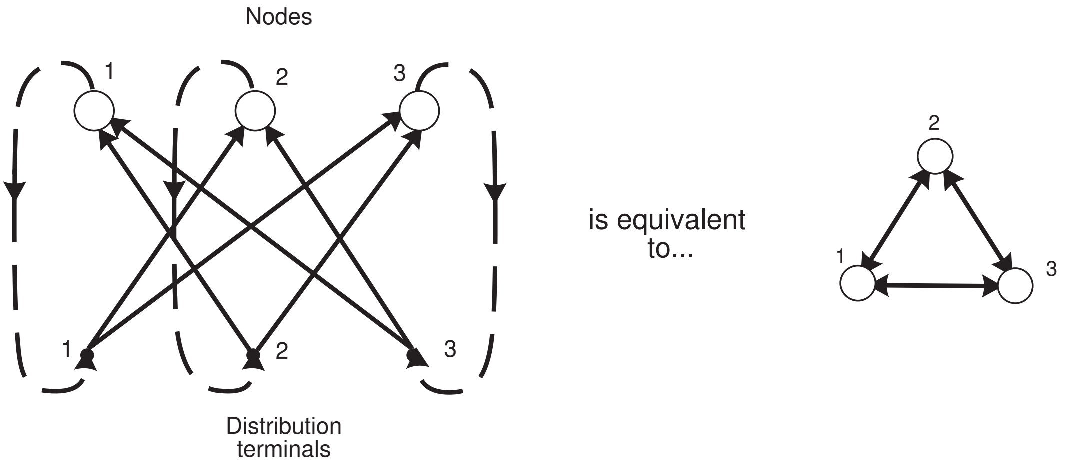
As a variation, suppose we had used the usual Boolean representation for the components x_i^p so that x_i^p is 0 or 1 The Hebb rule would now be $\Delta w_{ij} = \alpha x_i^p x_j^p$. Interpreting this, we have that the change in weight is only ever positive and only occurs if both nodes are firing (output '1'). This is, in fact closer to the original rule proposed by Hebb (1949) in a neurophysiological context. In his book *The Organization of behaviour* Hebb postulated that

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

That is, the correlation of activity between two cells is reinforced by increasing the synaptic strength (weight) between them. Simpson (course book) contains a list of Hebb rule variants.

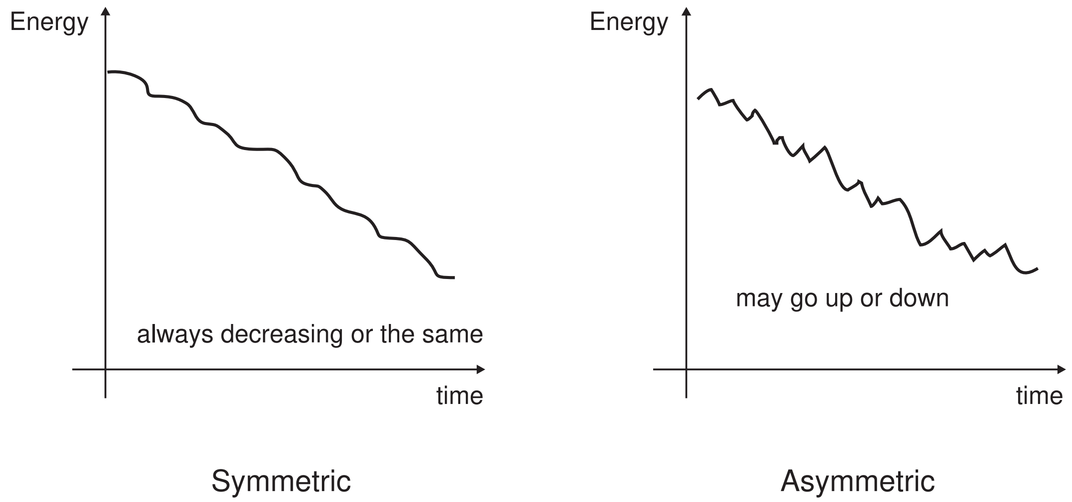
1.3 Using the delta rule

We may draw the connections in, say, a 3 node Hopfield net as follows.



3 node Hopfield net as feedforward with recurrence

Each node may now be thought of as taking part in some input- output function between the distribution terminals and the node outputs. They may therefore each be trained with the delta rule. If the corresponding training set for each one is linearly separable then the set of stable states may be learnt. However, there is no guarantee now that $w_{ij} = w_{ji}$. The change in energy at each node update is now no longer necessarily less than or equal to zero. The consequence is that, given the stable states have been trained, the system moves through state space with decreasing error towards a stable state but has, superimposed on this, some noise.



energy v time for symmetric and asymmetric nets

2 Storage capacity

How good is the storage prescription (1) at storing the patterns so that they are stable states? Clearly, as the number of patterns m increases, the chances of accurate storage must decrease. In some empirical work in his 1982 paper, Hopfield showed that about half the memories were stored accurately in a net of N nodes if $m = 0.15N$. The other patterns did not get stored as stable states. In a more rigorous piece of analysis McCliece et al. (1987) showed theoretically that, if we require almost all the required memories to be stored accurately, then the maximum number of patterns m is $N/2\log N$. For $N = 100$ this gives $m = 11$.

Suppose a pattern which was required to be stored did not, in fact produce a stable state and we start the net in this state. The net must evolve to *some* stable state and this is usually not related to any of the original patterns used in the prescription. The stable state represents a spurious energy minimum of the system - one that is not there by design.

3 The analogue Hopfield model

In a second important paper (Hopfield, 1984) Hopfield introduced a variant of the discrete time model discussed so far which uses nodes described by their rate of change of activation. This kind of node was discussed in the last part of lecture 1 but we review it here. Denote the sum of excitation from other nodes for the j node by s^j so that

$$s^j = \sum_i w_{ji} x_i \tag{3}$$

then the rate of change of activation da^j/dt is given by

$$\frac{da^j}{dt} = ks^j - ca^j \tag{4}$$

here, k and c are constant. The first term (if positive) will tend to make the activation increase while the second term is a decay term (see lecture 1). The output y^j is then just the sigmoid of the activation as usual. Hopfield also introduced the possibility of external input at this stage and a variable threshold.

In the previous TLU model, the possible states of an N node net are just the corners of the N-dimensional hypercube. In the new model, because the outputs can take any values between 0 and 1, the possible states include now, the interior of the hypercube. Hopfield defined an energy function for the new network and showed that if the inputs and thresholds were set to zero, as in the TLU discrete time model, and if the sigmoid was quite 'steep', then the energy minima were confined to regions close to the corners of the hypercube and these corresponded to the energy minima of the old model.

There, however, are two advantages of the new model. The first, is that the use of the sigmoid and time integration make more contact possible with real biological neurons. The second is that it is possible to build the new neurons out of simple, readily available hardware. In fact, Hopfield writes the equation for the dynamics - eqn (4) - as if it were built from an operational amplifier and resistor network. This kind of circuit was the basis of several implementations - see for example Graf et al. (1987).

4 Combinatorial optimisation

Another innovation made by Hopfield was to show how to solve large combinatorial optimisation problems on neural nets (Hopfield and Tank, 1985). The classical example of this is the so- called Travelling salesman Problem (TSP). Here, a travelling salesman has to visit each of a set of cities in turn in such a way as to minimise the total distance travelled. Each city must be visited once and only once. This kind of problem is computationally difficult in a technical sense (NP-complete) in that the time to solution scales with the number cities faster than the time t raised to any fixed power and therefore might scale like e^t.

The solution of the TSP consists of a sequence of cities. The problem for N cities may be coded into an N by N network as follows. Each row of the net corresponds to a city. The position of the city in the solution sequence is given by putting a '1' at the corresponding place in the row and 0's everywhere else in that row. Thus, if the city corresponding to row 5 was 7th in the sequence there would be a 1 in the 7th place of the 5th row and zeros everywhere else. Note that most states of the net do not correspond to valid tours - there must be only one '1' per row. The problem then, is to construct an energy function (and hence a set of weights) which lead to stable states (states of lowest energy) of the network that, not only express valid city tours, but which also are tours of short length. The validity criterian results in negative weights (inhibition) between nodes in the same row, and between nodes in the same column. The path-length criterion leads to inhibition between adjacent columns (cities in a path) proportional to the path length between the cities (row positions). The net is now allowed to run until an energy minimum is reached which should now correspond to a solution of the problem.

References

Graf, H., Hubbard, W., Jackel, L., and deVegvar P.G.N (1987). A cmos associative memory chip. In *1st Int. Conference Neural Nets, San Diego.*
(I have this).

Hebb, D. (1949). *The Organization of behaviour.* John Wiley.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational properties. *Proceedings of the National Academy of Sciences of the USA*, 79:2554 – 2588. Hopfield has another, related, model which uses continuous outputs. Beware when reading the literature which model is being discussed.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the USA*, 81:3088 – 3092.

Hopfield, J. and Tank, D. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141 – 152.

McEliece, R., Posner, E., Rodemich, E., and Venkatesh, S. (1987). The capacity of the hopfield associative memory. *IEEE Transactions on Information Theory*, IT-33:461 – 482.
There are many papers on this area, but this has some non- technical material near the beginning before getting into the welter of maths needed for this kind of analysis.