Hopfield Training Rules

To memorise a single pattern

Suppose we set the weights thus -

$$w_{ij} = \frac{1}{N} p_i p_j$$

where,

w_{ij} is the weight between nodes i & j
N is the number of nodes in the network
p_i is the value required for the i-th node

What will the network do when the memorised pattern is presented?

For each node, i, its activation, s_i, will be given by -

$$s_i = sign\left(\sum_{j=1}^{N} w_{ij} p_j\right) = sign\left(\sum_{j=1}^{N} \frac{1}{N} p_i p_j p_j\right)$$

$$= sign\left(\sum_{j=1}^{N} \frac{1}{N} p_i \times 1\right) \qquad \left[\left(p_j\right)^2 = 1 \quad \forall j\right]$$

$$= sign\left(p_i\right)$$

$$= p_i \quad \forall i$$

In other words, the activation of each node will remain unchanged. The memorised pattern is a **stable state** of the network.

Note that **any pattern** presented to the network which is similar to the memorised pattern will migrate towards the memorised pattern as the activation rule is repeatedly applied.

In fact, if more than half the bits of a presented pattern are the same as the memorised pattern then the memorised pattern will eventually be recreated in its entirety. If less than half are the same then the inverse of the memorised pattern (*+1*s instead of *–1*s and vice versa) will be generated. The memorised pattern and its inverse are **attractors** and the network will eventually end up at one of them.

To memorise more than one pattern

Now suppose we have *n* patterns which we wish to memorise. Extending the equation we used to set the weights for a single memory, we could try -

$$w_{ij} = \frac{1}{N} \sum_{k=1}^{n} p_i^k p_j^k$$

where,

w_{ij} is the weight between nodes *i* & *j*
N is the number of nodes in the network
n is the number of patterns to be learnt
p_i^k is the value required for the *i*-th node in pattern *k*

This equation will increase the weight between two nodes, *i* & *j*, whenever they are both active together. Note however that it is not uncommon to set w_{ii} to 0 for all *i*.This should remind you of the Hebb Rule. In fact the equation does more than this, it also reduces the weight between any pair of nodes where one node is active and the other is inactive. For this reason it is sometimes called the *Generalised Hebb Rule*.

All of the above properties are sound, both computationally and biologically. However, there is one further property of the above equation which is not biologically feasible; the weight between any two nodes which are simultaneously inactive is also increased by the equation. This is why Hopfield networks always create pairs of memories (the desired ones and their inverses). It does not (indeed cannot) distinguish between these two situations when the weights are being set.

Summary of Hopfield Network Equations

Weight setting (training) for *n* memories in an *N* node Hopfield Network -

$$w_{ij} = \frac{1}{N} \sum_{k=1}^{n} p_i^k p_j^k$$

Execution (iteration until convergence) -

$$s_i = sign\left(\sum_{j=1}^{N} w_{ij} s_j \right)$$

Hopfield's Energy Analysis

How can we be sure that repeated applications of the activation equation will result in a stable state being achieved?

Hopfield's most important contribution to the study of ANNs was his idea of calculating an energy level for his network. He defined the energy in such a way that states of the network (activations of the nodes) which represented learned memories had the lowest levels of energy. Any other states had a higher energy level and he showed that applying the activation equation reduced the energy of the network, thus moving the network closer to a learnt memory.

Hopfield defined the energy as follows -

$$H \propto -\left(\sum_{i=1}^{N} s_i p_i \right)^2$$

where,

s_i is the activation of node i
p_i is the i-th bit of a memory

Clearly, H will be at a minimum when $s_i = p_i$ AND when $s_i = -p_i$ so both the memory and its inverse will be energy minima of the network, as required. Whilst the precise value of the constant of proportionality is not crucial, it is convenient to set it to *1/(2*N)*.

Thus -

$$H = -\frac{1}{2N} \left(\sum_{i=1}^{N} s_i p_i \right)^2$$

When considering more than one memory we can attempt to make them all local minima of H by summing the above over all of the memories -

$$H = -\frac{1}{2N} \sum_{k=1}^{n} \left(\sum_{i=1}^{N} s_i p_i^k \right)^2$$

where,

n is the number of memories

Multiplying this out we get -

$$H = -\frac{1}{2N}\sum_{k=1}^{n}\left(\sum_{i=1}^{N} s_i p_i^k\right)\left(\sum_{j=1}^{N} s_j p_j^k\right)$$

$$= -\frac{1}{2}\sum_{i\&j}\left(\frac{1}{N}\sum_{k=1}^{n} p_i^k p_j^k\right)s_i s_j$$

and if we substitute using the Generalised Hebb Rule which is used to set the weights

$$w_{ij} = \frac{1}{N}\sum_{k=1}^{n} p_i^k p_j^k$$

we get -

$$H = -\frac{1}{2}\sum_{i\&j} w_{ij} s_i s_j$$

****

Now, given symmetric connections, $w_{ij} = w_{ji}$, we can rewrite the above equation as -

$$H = C - \sum_{(ij)} w_{ij} s_i s_j$$

where *(ij)* means all distinct pairs, *ij*, (I.e. counting *12* as the same pair as *21*) and where the *ii* terms are included in the constant *C*.

Following application of the activation equation to a node, *i*, that node's activation will change from s_i to s_i'.

$$s_i' = sign\left(\sum_{j=1}^{N} w_{ij} s_j\right)$$

We note that an update to the activation of node *i* will only have occurred if $s_i' = -s_i$ and that there will be a consequent change in the energy of the network which we can describe as *H'*-*H* and which will be given by –

$$H' - H = C - C + \sum_{j \neq i} w_{ij} s_i s_j - \sum_{j \neq i} w_{ij} s_i' s_j$$

since only node *i* has changed and the rest of the terms in the summation of distinct pairs will cancel out leaving just those terms involving node *i*. Given that $s_i' = -s_i$ the two summations above are actually the same once the difference in their signs is taken into account -

$$H' - H = 2 s_i \sum_{j \neq i} w_{ij} s_j$$

$$= 2 s_i \sum_{j=1}^{N} w_{ij} s_j - 2 w_{ii} s_i s_i$$

Now we know that s_i has a different sign to s_i' so

$$2 s_i \sum_{j=1}^{N} w_{ij} s_j = 2 s_i s_i' < 0$$

and

$$w_{ii} = \frac{1}{N} \sum_{k=1}^{n} p_i^k p_i^k = \frac{n}{N} > 0$$

unless we've chosen to set w_{ii} to 0 for all *i* of course.

So H'-H is something less than 0 minus something greater than or equal to 0.

I.e.

$$H' - H < 0$$

****

Therefore any application of the activation rule will result in either no change (convergence on an energy minimum has been achieved) or a decrease in the energy of the system (convergence towards a minimum continues).

Stability of Memories in Hopfield Networks

Let net_i^k be the total input to a node i when a pattern k is to be memorised. For this pattern k to be a stable state of the network we need -

$$sign\left(net_i^k\right) = p_i^k \quad \forall i$$

Now,

$$net_i^k = \sum_{j=1}^{N} w_{ij} p_j^k = \frac{1}{N} \sum_{j=1}^{N} \sum_{l=1}^{n} p_i^l p_j^l p_j^k$$

Within the summation over l we have the special case when $l = k$. The summation over l yields a term $p_j^k p_j^k p_i^k = p_i^k$ which is then summed N times in the summation over j and then divided by N giving a "separated-out" p_i^k term-

$$net_i^k = p_i^k + \frac{1}{N} \sum_{j=1}^{N} \sum_{l \neq k} p_i^l p_j^l p_j^k$$

From this we can see that if the second term is zero we have stability – the activation of node i will be the same as p_i^k . Furthermore, we can see that if the second term (the "crosstalk" term) is small enough then it will not alter the stability of the node. If the magnitude of the crosstalk term is less than *1* then the sign of net_i^k cannot be changed and stability will be preserved.

Storage Capacity of the Hopfield Network

We cannot specify the storage capacity of a Hopfield network in a deterministic way. Some memories interfere with each other more than others. Any attempt to calculate the capacity of a Hopfield network has to be statistical in nature.

We shall now use our earlier stability analysis to create a mathematical artefact which will enable us to derive some statistical results about the memory capacity of a Hopfield network.

Consider the quantity –

$$C_i^k \equiv -p_i^k \times \frac{1}{N} \sum_{j=1}^{N} \sum_{l \neq k} p_i^l p_j^l p_j^k$$

I.e. $-p_i^k$ multiplied by the crosstalk term

If C_i^k is negative then the crosstalk term has the same sign as p_i^k so node *i* is stable. If C_i^k is positive and greater than *1* then the sign of net_i^k will be changed and bit *i* of pattern *k* has become unstable. In fact, if we placed pattern *k* onto the network and applied the activation equation (executed the net) then a different pattern would emerge with node *i* having changed value, from *+1* to *–1* or vice versa.

Note that C_i^k depends only on the patterns we are trying to store.

We shall now consider what happens to a pattern chosen at random in which each bit has an equal probability of being *+1* or *–1* and derive an estimate of the probability that any bit is unstable –

$$P_{error} = P(C_i^k > 1)$$

P_{error} will obviously increase as we increase the number of patterns to be memorised. We need a criterion for acceptable performance. For example, if we wish the probability of a bit being unstable to be less than *0.01* ($P_{error} < 0.01$) then we want to be able to calculate the maximum number of patterns, n_{max}, we can store.

It turns out that C_i^k is binomially distributed with mean *0* and variance *n/N*. If *N*n* is large we can use the Central Limit Theorem to approximate the distribution of C_i^k with a Normal (Gaussian) distribution with the same mean and variance.

Now we can draw up a table by selecting different values of P_{error} and determining what implications they have for n_{max}/N –

P_{error}	n_{max}/N
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61

So, if we choose $P_{error} < 0.01$, for instance, then n_{max} must not exceed $0.185*N$. If we tried to memorise $0.185*N$ patterns then less than *1%* of the bits would be unstable – initially!

Unfortunately even one unstable bit can have dramatic knock-on effects as we iteratively apply the activation equation and it is quite possible for an initial *1%* of unstable bits to evolve into a situation where the majority have become unstable. It can, in fact, be shown that this kind of avalanche occurs when the number of patterns to be memorised exceeds $0.138*N$ (or when $P_{error} > 0.0036$).

There is not one single correct way in which to approach the question of the capacity of a Hopfield network and alternatives to the above analysis have been investigated. One of the more useful approaches is to consider the probability of perfect recall of bits and patterns. The outcomes of this alternative approach are merely stated here. See the Hertz, Krogh and Palmer book on the reading list for further details.

In order to recall all N bits of a pattern correctly with a *99%* probability –

$$n_{\max} \approx \frac{N}{2 \ln N}$$

In order to recall all N bits of all of the patterns perfectly with a *99%* probability –

$$n_{\max} \approx \frac{N}{4 \ln N}$$