5: Associative memories - the Hopfield net

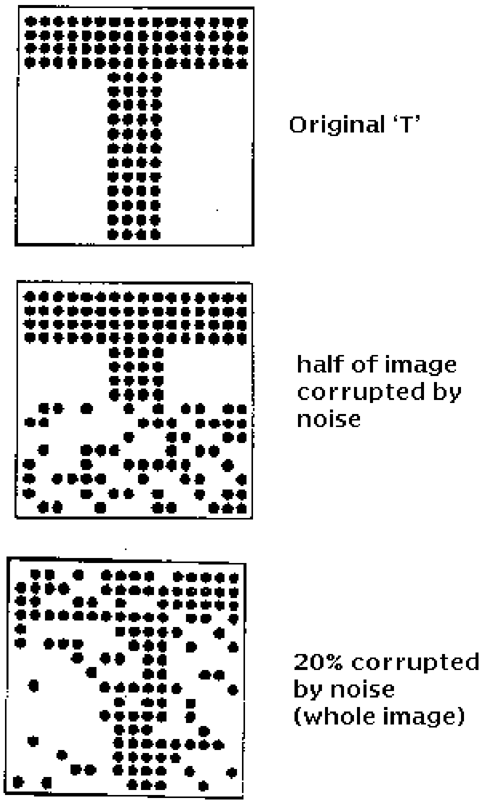
Kevin Gurney

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

1 The nature of associative memory

'Remembering' something in common parlance usually consists of associating something with a sensory cue. For example, someone may say something, like the name of a celebrity, and we immediately recall a chain of events or some experience related to the celebrity - we may have seen them on TV recently for example. Or, we may see a picture of a place visited in our childhood and the image recalls memories of the time. The sense of smell (olfaction) is known to be especially evocative in this way.

On a more mundane level, but still in the same category, we may be presented with a partially obliterated letter, or one seen through a window when it is raining (letter + noise) and go on to recognise the letter.



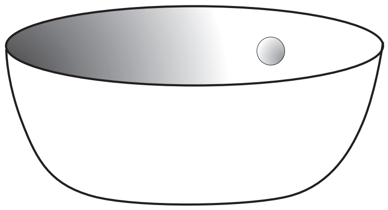
slide of 'T's

The common paradigm here may be described as follows. There is some underlying collection of data which is ordered and interrelated in some way and which is stored in memory. The data may be thought of, therefore, as forming a stored pattern. In the recollection examples above, it is the cluster of memories associated with the celebrity or the phase in childhood. In the case of character recognition, it is the parts of the letter (pixels) whose arrangement is determined by an archetypal version of the letter. When part of the pattern of data is presented in the form of a sensory cue, the rest of the pattern (memory) is recalled or *associated* with it. Notice that it often doesn't matter which part of the pattern is used as the cue, the whole pattern is always restored.

Conventional computers (von Neumann machines) can perform this function in a very limited way. The typical software for this is usually referred to as a database. Here, the 'sensory cue' is called the *key* which is to be searched on. For example, the library catalogue is a database which stores the authors, titles, classmarks and data of publication of books and journals. We may search on any one of these discrete items for a catalogue entry by typing the complete item after selecting the correct option from a menu. Suppose now we have only the fragment 'ion, Mar' from the full title 'Vision, Marr D.'. There is no way that the database can use this fragment of information to even start searching. We don't know if it pertains to the author or the title, and even if we did, we might get titles or authors that start with 'ion'. The kind of input to the database has to be very specific and complete.

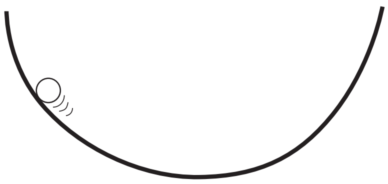
2 A physical analogy with memory

The networks that are used to perform associative recall are specific examples of a wider class of physical systems which may be thought of as doing the same thing. This allows the net operation to be viewed as a the dynamics of a physical system and its behaviour to be described in terms of the network's 'energy'. Consider a bowl in which a ball bearing is allowed to roll freely



bowl and ball bearing in 3D

This is more easily drawn using a 2D cross section



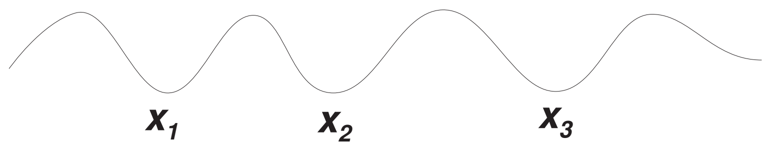
2d X-section of bowl

Suppose we let the ball go from a point somewhere up the side of the bowl with, possibly, a push to one side as well. The ball will roll back and forth and around the bowl until it comes to rest at the bottom.

The physical description of what has happened may be couched in terms of the energy of the system. The ball initially has some *potential* energy. That is work was done to push it up the side of the bowl to get it there and it now has the potential to gain speed and acquire energy. When the ball is released, the potential energy is released and the ball rolls around the bowl (it gains *kinetic* energy). Eventually the ball comes to rest where its energy (potential and kinetic) is zero. (The kinetic energy gets converted to heat via friction with the bowl and the air). The main point is that the ball comes to rest in the same place every time and this place is determined by the energy minimum of the system (ball + bowl). The resting state is said to be *stable* because the system remains there after it has been reached

There is another way of thinking of this process which ties in with our ideas about memory. We suppose that the ball comes to rest in the same place each time because it 'remembers' where the bottom of the bowl is. We may push the analogy further by giving the ball a coordinate description. Thus, its position or *state* at any time is given by the three coordinates (x, y, z) or the position vector $\mathbf{x}$. The location of the bottom of the bowl, $\mathbf{x}_0$ represents the pattern which is stored. By writing the ball's vector as the sum of $\mathbf{x}_0$ and a displacement $\Delta\mathbf{x}$ thus, $\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x}$, we may think of the ball's initial position as representing the partial knowledge or cue for recall, since it approximates to the memory $\mathbf{x}_0$.

If we now use a corrugated surface instead of a single depression (the bowl) we may store many 'memories'.



$\{ x_1 , x_2 , x_3 \ldots \; x_n \}$ are the stored memories

X-section through corrugated surface

If now the ball is started somewhere on this surface, it will eventually come to rest at the local depression which is closest to its initial starting point. That is it evokes the stored pattern which is closest to its initial partial pattern or cue. Once again, this is an energy minimum for the system.

There are therefore two complementary ways of looking at what is happening. One is to say that the system falls into an energy minimum; the other is that it stores a set of patterns and recalls that which is closest to its initial state. If we are to build a network which behaves like this we must include the following key elements

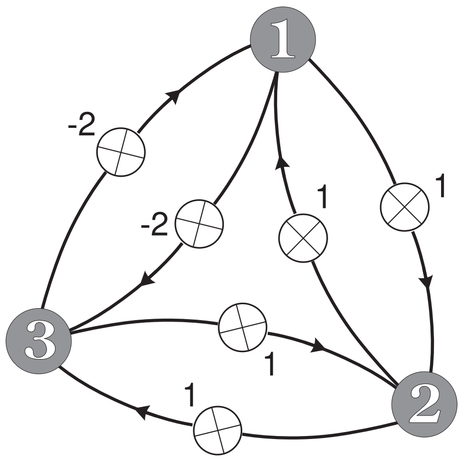
1. It is completely described by a *state vector* $\mathbf{v} = (v_1, v_2, \ldots, v_n)$

2. There are a set of *stable states* $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_1, \ldots, \mathbf{v}_n$ These will correspond to the stored patterns and, in, the corrugated surface example, were the bottoms of the depressions in the surface.

3. The system evolves in time from any arbitrary starting state $\mathbf{v}$ to one of the stable states, and this may be described as the system decreasing its energy E. This corresponds to the process of memory recall.

3 The Hopfield net

Consider the network consisting of three TLU nodes shown below



3 node Hopfield net

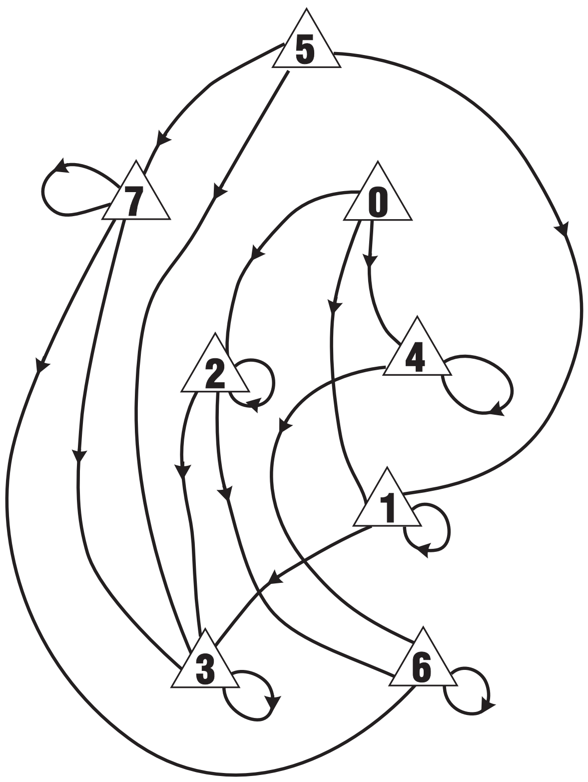
Every node is connected to every other node (but *not* to itself) and the connection strengths or weights are symmetric in that the weight from node i to node j is the same as that from node j to node i. That is, $w_{ij} = w_{ji}$, and $w_{ii} = 0$ for all i, j.* Notice that the flow of information in this net is not in a single direction as it has been in the nets dealt with so far. It is possible for information to flow from a node back to itself via other nodes. That is, there is feedback in the network and so they are known as feedback or *recurrent* nets as opposed to *feedforward* nets which were the subject of the Backpropagation algorithm.

The state of the net at any time is given by the vector of the node outputs (x_1, x_2, x_3). Suppose we now start this net in some initial state and choose a node at random and let it update its output or 'fire'. That is, it evaluates its activation in the normal way and outputs a '1' if this is greater than or equal to zero and a '0' otherwise. The net now finds itself either in the same state as it started in, or in a new state which is at Hamming distance one from the old. We now choose a new node at random to fire and continue in this way over many steps. What will the behaviour of the net be? For each state, we may evaluate the next state given each of the three nodes fires. This gives the following table.

State				**New state** (after node has fired)		
Number	vector					
	x_1	x_2	x_3	Node 1	Node 2	Node 3
0	0	0	0	4	2	1
1	0	0	1	1	3	1
2	0	1	0	6	2	3
3	0	1	1	3	3	3
4	1	0	0	4	6	4
5	1	0	1	1	7	3
6	1	1	0	6	6	6
7	1	1	1	3	7	6

*The weight from node i to node j is sometimes also denoted by w_i^j

This information may be represented in graphical form as a *state transition diagram*.



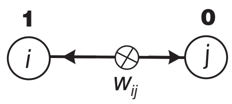
state transition diagram for 3 node net

States are represented by the circles with their associated state number. Directed arcs represent possible transitions between states and the number alongside each arc is the probability that each transition will take place. The states have been arranged in such a way that transitions tend to take place down the diagram; this will be shown to reflect the way the system decreases its energy. The important thing to notice at this stage is that, no matter where we start in the diagram, the net will eventually find itself in one of the states '3' or '6'. These reenter themselves with probability 1. That is they are stable states - once the net finds itself in one of these it stays there. The state vectors for '3' and '6' are (0,1,1) and (1,1,0) respectively and so these are the 'memories' stored by the net.

3.1 Defining an energy for the net

The dynamics of the net are described perfectly by the state transition table or diagram. However, greater insight may be derived if we can express this in terms of an energy function and, using this formulation, it is possible to show that stable states will always be reached in such a net.

Consider two nodes i, j in the net which are connected by a positive weight and where j is currently outputting a '0' while i is outputting a '1'.



two nodes in conflict

If j were given the chance to update or fire, the contribution to its activation from i is positive and this may well serve to bring j's activation above threshold and make it output a '1'. A similar situation would prevail if the initial output states of the two nodes had been reversed since the connection is symmetric. If, on the other hand, both units are 'on' they are reinforcing each other's current output. The weight may therefore be thought of as fixing a constraint between i and j that tends to make them both take on the value '1'. A negative weight would tend to enforce opposite outputs. One way of viewing these networks is therefore as *constraint satisfaction* nets.

This idea may be captured quantitatively in the form of a suitable energy function. Define

$$e_{ij} = -w_{ij}x_i x_j \tag{1}$$

The values that e_{ij} take are given in the table below

x_i	x_j	e_{ij}
0	0	0
0	1	0
1	0	0
1	1	$-w_{ij}$

If the weight is positive then the last entry is negative and is the lowest value in the table. If e_{ij} is regarded as the 'energy' of the pair ij then the lowest energy occurs when both units are on which is consistent with the arguments above. If the weight is negative, the '11' state is the highest energy state and is not favoured. The energy of the net is found by summing over all pairs of nodes

$$E = \sum_{\text{pairs}} e_{ij} = -\sum_{\text{pairs}} w_{ij}x_i x_j \tag{2}$$

This may be written

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j \tag{3}$$

Since the sum includes each pair twice (as $w_{ij}x_i x_j$ and $w_{ji}x_j x_i$) and $w_{ij} = w_{ji}$, $w_{ii} = 0$.

It is now instructive to see what the change in energy is when a node fires. Suppose node k is chosen to be updated. Write the energy E by singling out the terms involving this node.

$$E = -\frac{1}{2}\sum_{\substack{i\neq k\\ j\neq k}} w_{ij}x_i x_j - \frac{1}{2}\sum_i w_{ki}x_k x_i - \frac{1}{2}\sum_i w_{ik}x_i x_k \tag{4}$$

Now, because $w_{ik} = w_{ki}$, the last two sums may be combined

$$E = -\frac{1}{2}\sum_{\substack{i\neq k\\ j\neq k}} w_{ij}x_i x_j - \sum_i w_{ki}x_k x_i \tag{5}$$

For ease of notation, denote the first sum by S and take the x_k outside the sum since it is constant throughout, then

$$E = S - x_k \sum_i w_{ki} x_i \tag{6}$$

but the sum here is just the activation of the kth node so that

$$E = S - x_k a^k \tag{7}$$

Let the energy after k has updated be E' and the new output be x'_k. Then

$$E' = S - x'_k a^k \tag{8}$$

Denote the change in energy $E' - E$ by ΔE and the change in output $x'_k - x_k$ by Δx_k, then subtracting (7) from (8)

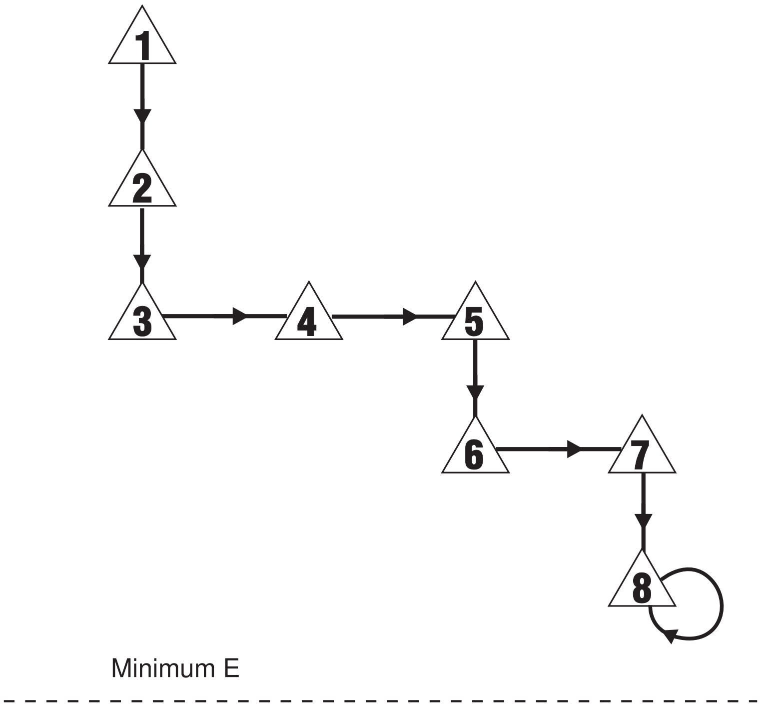
$$\Delta E = -\Delta x_k a^k \tag{9}$$

There are now two cases to consider

1. $a^k \geq 0$. Then the output goes from '0' to '1' or stays at '1'. In either case $\Delta x_k \geq 0$. Therefore $\Delta x_k a^k \geq 0$ and so, $\Delta E \leq 0$

2. $a^k < 0$. Then the output goes from '1' to '0' or stays at '0'. In either case $\Delta x_k \leq 0$. Therefore, once again $\Delta x_k a^k \geq 0$ and $\Delta E \leq 0$

Thus, for any node being updated we always have $\Delta E \leq 0$ and so the energy of the net decreases or stays the same. But the energy is bounded below by a value obtained by putting all the $x_i, x_j = 1$ in (3). Thus E must reach some fixed value and then stay the same. Once this has occurred, it is possible for further changes in the network's state to take place since $\Delta E = 0$ is still allowed. However, for this to be the case ($\Delta x_k \neq 0$ and $\Delta E = 0$) we must have $a^k = 0$. This implies the change must be of the form $0 \to 1$. There can be at most N (of course there may be none) of these changes, where N is the number of nodes in the net. After this there can be no more change to the net's state and a stable state has been reached.

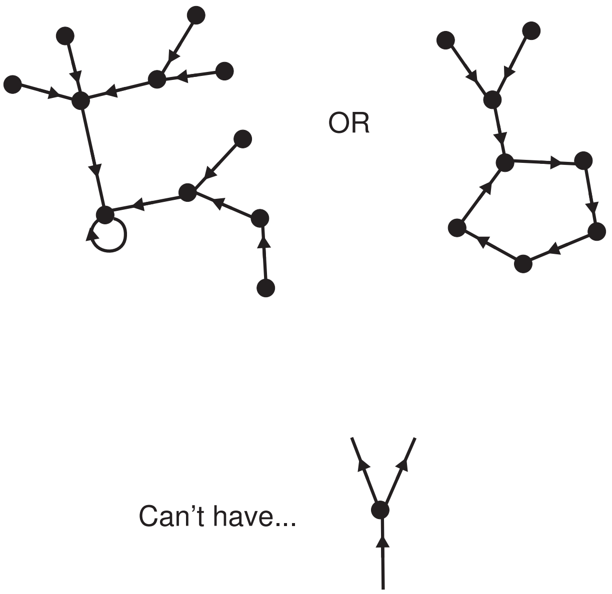


state transitions to stable state

In the example given above, all state have zero energy except for state 5 which has energy 2, and the stable states 3 and 6 which have energy -1.

3.2 Asynchronous vs. synchronous update

So far we have allowed only a single node to update or fire at any time step. All nodes are possible candidates for update and so they operate *asynchronously*; that is, there is no coordination between them in time. The other extreme case occurs if we make all nodes fire at the same time, in which case we say there is *synchronous* update. To do this, we must ensure that each nodes previous output is available to the rest of the net until all nodes have evaluated their activation and been updated. It is therefore necessary to store both the current state vector *and* the next state vector. The behaviour is now deterministic; given any state, a state transition occurs to a well defined next state, there being no probabilistic behaviour. The analysis, in terms of energy changes at each update, given above no longer applies but there is now a simplified type of state diagram in which only a single arc emerges from any state. This allows us to predict, once again, the general type of behaviour. In particular, state-cycles occur again but now there is the possibility for *multiple-state cycles*.



state diagrams for synchronous update

These may be useful in storing sequences of events or patterns. A little thought will show that the (single) state cycles remain the same under synchronous dynamics so the single stored patterns remain the same under both dynamics.

3.3 Ways of inputting information

So far it has been assumed that the net is started in some initial state (the memory cue) and the whole net allowed to run freely until a state cycle is encountered (recall slide of noisy 'T'). There is another possibility in which some part of the net has its outputs fixed while the remainder is allowed to update. The part that is fixed is said to be *clamped* and if the clamp forms part of a state cycle, the remainder (unclamped) part of the net will complete the pattern stored at that cycle (recall slide of partially correct 'T'). Which mode is used will depend on any prior knowledge about parts of the cue being uncorrupted or noise free.

The problem of how to fix the weights in Hopfield nets will be dealt with next time.

References

Aleksander, I. and Morton, H. (1990). *Neural Computing.* Chapman hall.
Quite good on Hopfield nets and contains a similar example to the one given in these notes.

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.

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.