POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Iterative Autoassociative Net: Discrete Hopfield

K. Ming Leung

Abstract: A fully-connected iterative autoassociative neural network known as the discrete Hopfield net is studied.

Directory

Last Revision Date: March 13, 2007

Table of Contents

1. Discrete Hopfield Net

There are a few different variations and extensions of the discrete Hopfield net. We will discuss the following version of it for pattern autoassociation.

1. A set of patterns $\mathbf{s}^{(q)}, q = 1, \ldots, Q$ is assumed to be given for storage.
2. The net is fully connected in the sense that each neuron is connected to every other neuron except itself. It no longer makes any sense separating the neurons into layers. The neurons are simply labeled by $Y_i, i = 1, \ldots, N$.
3. The net has weights given by Hebb rule:

$$w_{ij} = \begin{cases} \sum_{q=1}^{Q} s_i^{(q)} s_j^{(q)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

for bipolar vectors. For binary vectors, each $\mathbf{s}^{(q)}$ must be replaced by its bipolar equivalents $2\mathbf{s}^{(q)} - 1$ in the above expression. The weight matrix is always symmetric and has zero diagonal elements.

4. Signal from an input test pattern, $\mathbf{x}$, is treated as an external signal that is applied to every neuron at each time step in addition to the signal from all the other neurons in the net. This feature can be omitted from the network with minor consequences.

5. Only one neuron updates its activation at a time (asynchronous updating) based on the external signal and the signal that it receives from all the other neurons.

6. Neurons are chosen at random to update their activations, and neurons should be updated at the same average rate.

7. The transfer function is defined so that

$$y_i = f_{\text{Hopfield}}(y_{\text{in},i}) = \begin{cases} 1, & \text{if } y_{\text{in},i} > \theta_i \\ \text{previous } y_i & \text{if } y_{\text{in},i} = \theta_i \\ -1, & \text{if } y_{\text{in},i} < \theta_i \end{cases}$$

for bipolar neurons, and

$$y_i = f_{\text{Hopfield}}(y_{\text{in},i}) = \begin{cases} 1, & \text{if } y_{\text{in},i} > \theta_i \\ \text{previous } y_i & \text{if } y_{\text{in},i} = \theta_i \\ 0, & \text{if } y_{\text{in},i} < \theta_i \end{cases}$$

for bipolar neurons. In either case, the thresholds, θ_i, are typically fixed at zero. The conventional Heaviside step function can also be used instead of the more complicated form of transfer function used above. The differences are minimal.

Suppose we are given an input test vector, $\mathbf{x}$. For bipolar neurons, the algorithm to find its associated pattern for the discrete Hopfield net is:

1 Compute the weight matrix using Hebb rule. Set $w_{ii} = 0$.

2 Set initial activation equal to the given input test vector: $y_i(0) = x_i,\ i = 1, \ldots, N$.

3 Repeat steps a - c for $t = 1, 2, \ldots$ until no change in any of the activations is possible.

a Choose a neuron, i, at random and with equal probability.

b Compute the net input

$$y_{\text{in},i}(t-1) = x_i + \sum_{j \neq i}^{N} y_j(t-1) w_{ji}$$

c Update the activations

$$y_i(t) = f_{\text{Hopfield}}(y_{\text{in},i}(t-1)).$$

For binary neurons, the training vectors, $\mathbf{s}^{(q)}, q = 1, \ldots, Q$, must first be converted to bipolar form before applying Hebb rule to com-

pute the weight matrix. We must also use the binary form for the transfer function to compute the activations for the neurons.

2. Interpretation and Analysis of the Algorithm

With the initial activations given by the test vector ($\mathbf{y}(0) = \mathbf{x}$), the updating rule for the discrete Hopfield net

$$y_i(t) = f_{\text{Hopfield}}(x_i + \sum_{j \neq i}^{N} y_j(t-1)w_{ji})$$

for $t = 1, 2, \ldots$, describe the dynamics (time evolution) of the activations of a collection of N neurons

$$y_i(t), \quad \text{for } i = 1, 2, \ldots, N \text{ and } t = 1, 2, \ldots.$$

The set $\{y_i(t), i = 1, \ldots, N\}$ specifies the state of the network at time t.

The space of all possible states of the network is called the configuration space. Each point in the configuration space specifies a

state of the network. In the case of binary or bipolar neurons, the configuration space consists of discrete points on a hypercube.

From the updating rule, it is clear that the activation of each neuron directly influences the activations of other neurons one time step later. The situation is like having a system of N interacting particles whose positions at time t are specified by $y_i(t), i = 1, \ldots, N$. Each particle exerts a force on all the other particles and thus directly influences their subsequent positions. The given set of stored patterns $\{\mathbf{s}^{(q)}, q = 1, \ldots, Q\}$, determines how one particle can influence the other particles (that is the interactions).

A given test patterns specifies the initial ($t = 0$) state of the network. The update rule governs the dynamics of the system (time evolution of the state). It turns out that after some elapsed time the state ceases to change and reaches a state of equilibrium. This equilibrium state is then the network output for that particular test pattern.

It can be shown that each stored pattern corresponds to a state of equilibrium of the network. To see that, let us assume that our test

pattern is $\mathbf{s}^{(p)}$. Therefore we set $\mathbf{x} = \mathbf{s}^{(p)}$ and $\mathbf{y}(0) = \mathbf{s}^{(p)}$, and so

$$\begin{aligned} y_{\text{in},i}(0) &= s_i^{(p)} + \sum_{j\neq i}^{N} s_j^{(p)} w_{ji} = s_i^{(p)} + \sum_{j\neq i}^{N} s_j^{(p)} \sum_{q=1}^{Q} s_j^{(q)} s_i^{(q)} \\ &= s_i^{(p)} + \sum_{j\neq i}^{N} s_j^{(p)} s_j^{(p)} s_i^{(p)} + \sum_{j\neq i}^{N} s_j^{(p)} \sum_{q\neq p}^{Q} s_j^{(q)} s_i^{(q)} \\ &= N s_i^{(p)} + \sum_{q\neq p}^{Q} s_i^{(q)} \sum_{j\neq i}^{N} s_j^{(p)} s_j^{(q)}. \end{aligned}$$

This last term on the right-hand side is referred to as the crosstalk.

Assume for the moment that there is only one stored pattern, then this crosstalk is zero. Then for $i = 1, \ldots, N$, we have

$$y_i(1) = f_{\text{Hopfield}}(y_{\text{in},i}(0)) = f_{\text{Hopfield}}(N s_i^{(p)}) = f_{\text{Hopfield}}(s_i^{(p)}) = s_i^{(p)}.$$

But this is exactly $y_i(0)$ and so none of the activation will change. Thus the stored pattern is an equilibrium state of the network. Moreover, as long as the input test vector $\mathbf{x}$ is close enough to the store vector, $\mathbf{s}^{(p)}$, (in fact in this case, as long as the majority of the bits

in $\mathbf{x}$ agree with those in the stored vector), the network output after reaching equilibrium will always be the stored vector.

When there is more than one stored pattern, the crosstalk is not expected to be zero. Nevertheless, if N is large but Q is not too large, and the stored vectors are not too correlated with each other (that is the dot-products of each pair of the stored vectors are small compared with N and tend to have different signs), then the crosstalk is not large enough to change the overall signs for $y_{\text{in},i}(0)$. As a result, $y_i(1)$ will still be given by $s_i^{(p)}$.

Thus the stored patterns are equilibrium states of the network, and they are referred to as attractors for the dynamic system. A test vector (initial state) near any one of the stored pattern will converge to that pattern. The network will correct errors as desired.

Clearly from the symmetry of the problem, for every attractor, $\mathbf{s}^{(p)}$, there is a corresponding attractor given by the reversed state, $-\mathbf{s}^{(p)}$. If a test vector has most of its bits opposite to those of $\mathbf{s}^{(p)}$, then the network will converge to $-\mathbf{s}^{(p)}$.

For any given stored pattern (attractor), $\mathbf{s}^{(p)}$, the region in config-

uration space where any test vector will eventually converge to $\mathbf{s}^{(p)}$, is called the basin of attraction for that attractor.

It turns out that other than for the reversed states, there are spurious attractors that have little direct relations to the stored patterns. The entire configuration space can be decomposed into basins of attraction for the intended attractors and the spurious ones. Of course we want to make sure that the basins of attraction for the spurious attractors are relative small, so that most of the input test patterns will converge to one of the stored patterns.

3. Lyapunov Function

Hopfield proved that the discrete Hopfield net converges to a stable limit point corresponding to a set of activities by considering the following Lyapunov function (or referred to as the energy function)

$$L = -\frac{1}{2}\sum_{i\neq j}^{N}\sum_{j=1}^{N} y_i y_j w_{ij} - \sum_{i=1}^{N} x_i y_i + \sum_{i=1}^{N} \theta_i y_i.$$

Suppose at a given iteration step, neuron Y_k is chosen to update its activation so that y_k becomes $y_k + \Delta y_k$. The change in y_k, Δy_k, is determined from the discrete Hopfield algorithm. We also want to find the change in the Lyapunov function, ΔL as a result of this change in y_k. To accomplish that, we first isolate from L all the terms that involve y_k.

First let us consider the first term on the right-hand side of the expression for L. In the sum over j, there are term proportional to y_k when $j = k$. These terms are

$$-\frac{1}{2}\sum_{i\neq k}^{N} y_i y_k w_{ik}.$$

The remaining terms also have terms proportional to y_k, and these terms

$$-\frac{1}{2}\sum_{j\neq k}^{N} y_k y_j w_{kj}$$

come from setting $i = k$ in the summation over i.

The other two terms on the right-hand side of the expression for L also contain y_k. These terms

$$-(x_k - \theta_k)y_k$$

come from setting $i = k$ in the sums.

Combining all the terms that involve y_k, we have

$$-\left[\frac{1}{2}\sum_{i\neq k}^{N} y_i w_{ik} + \frac{1}{2}\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] y_k.$$

Notice that because of the no self-connection requirement, there is no term which is proportional to the square of y_k. Also notice that the first term actually becomes the second term if the dummy index i is replaced by j, and if use is made of the fact that $w_{jk} = w_{kj}$. The resulting terms can be written as

$$-\left[\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] y_k.$$

These are the only term in L that contains y_k.

So when y_k is changed by an amount Δy_k, then L is changed by an amount

$$\Delta L = -\left[\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] \Delta y_k.$$

We assume binary neurons here. The following argument applies to bipolar neurons with only slight modifications (written in red)

First suppose that y_k is 1 (1). If $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k < 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} < \theta_k$, then y_k will change to 0 (-1), and therefore $\Delta y_k < 0$. Thus

$$\Delta L = -\left[\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] \Delta y_k < 0.$$

On the other hand, if $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k \geq 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} \geq \theta_k$, then y_k will remain unchange, and therefore $\Delta y_k = 0$, and so $\Delta L = 0$.

Next, suppose that y_k is 0 (−1). If $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k > 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} > \theta_k$, then y_k will change to 1 (1), and therefore $\Delta y_k > 0$. Thus

$$\Delta L = -\left[\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] \Delta y_k < 0.$$

On the other hand, if $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k \leq 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} \leq \theta_k$, then y_k will remain unchange, and therefore $\Delta y_k = 0$, and so $\Delta L = 0$.

For bipolar neurons, there is actually one more case that has to be considered. This is the case where $y_k = 0$, meaning that its value is uncertain.

If $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k > 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} > \theta_k$, then y_k will change to 1, and so $\Delta y_k > 0$. Thus

$$\Delta L = -\left[\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] \Delta y_k < 0.$$

If $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k < 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} < \theta_k$, then y_k will change to -1, and so $\Delta y_k < 0$. Thus

$$\Delta L = -\left[\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k\right] \Delta y_k < 0.$$

If $\sum_{j\neq k}^{N} y_j w_{kj} + x_k - \theta_k = 0$, and so $x_k + \sum_{j\neq k}^{N} y_j w_{kj} = \theta_k$, then y_k will remain at 0. Thus we have $\Delta y_k = 0$, and so $\Delta L = 0$.

Consequently we see that in all cases the Lyapunov function, L, cannot increase, it either decreases or remains unchange for that iteration.

Moreover, L is clearly bounded below:

$$L > -\frac{1}{2}\sum_{i\neq j}^{N}\sum_{j}^{N} |w_{ij}| - N - \sum_{i=1}^{N} |\theta_i|.$$

Therefore the activations of the net must eventually reach a set of stable equilibrium values such that the Lyapunov function does not change with further iteration.

The important features of the discrete Hopfield net that are necessary for the proof are:

1. asynchronous updating
2. no self-connection

On the other hand, the following features are not important at all:

1. binary or bipolar neurons
2. external signal maintained during iteration

4. Storage Capacity

Because of the existence of spurious attractors, some errors in recalling any of the stored vectors are expected in the discrete Hopfield net. In the case of binary vectors, Hopfield found experimentally that the number of patterns that can be recalled with reasonable accuracy is

$$C \approx 0.15N.$$

For a similar net with bipolar vectors, some others have found using more detailed theoretical analysis that

$$C \approx \frac{N}{2 \log_2 N}.$$

5. Example 1: Discrete Hopfield Net

We consider the binary version of the example that we have considered before for an iterative autoassociative net, in which the single binary vector $\mathbf{s} = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$ is stored. Clearly we have $Q = 1$ and $N = 4$. Hebb rule gives the weight matrix:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

assuming that the NN has no self connections. Note that the training vector has to be converted to bipolar form before applying Hebb rule to obtain the weight matrix.

We now test the network with an input vector $\mathbf{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$ that has mistakes in the first and second components. We assume that the neurons update their activations in the following random order: 1, 4, 3, 2.

So we set $\mathbf{y} = \mathbf{x}$. We first update the activation for Y_1:

$$y_{\text{in},\,1} = x_1 + \sum_{j=1}^{4} y_j w_{j1} = 0 + 1 = 1 \quad \Rightarrow y_1 = 1.$$

Thus $\mathbf{y} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$.

Next we update the activation for Y_4:

$$y_{\text{in},\,4} = x_4 + \sum_{j=1}^{4} y_j w_{j4} = 0 + (-2) = -2 \quad \Rightarrow y_4 = 0.$$

Thus y_4 and therefore $\mathbf{y}$ remain unchanged.

Then we update the activation for Y_3:

$$y_{\text{in},\,3} = x_3 + \sum_{j=1}^{4} y_j w_{j3} = 1 + 1 = 2 \quad \Rightarrow y_3 = 1.$$

Thus y_3 and therefore $\mathbf{y}$ remain unchanged.

After that we update the activation for Y_2:

$$y_{\text{in},\,2} = x_2 + \sum_{j=1}^{4} y_j w_{j2} = 0 + 2 = 2 \quad \Rightarrow y_2 = 1.$$

Thus $\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$.

Since some activations have changed, in general we need to iterate at least $N = 4$ more times to check to see if the net has converged. However, since we have only one stored vector, and the current activation vector is exactly the stored vector, we can actually terminate the iteration since we know that the net has reached an equilibrium.

6. Example 2: Discrete Hopfield Net

Next we consider storing two binary vectors

$$\mathbf{s}^{(1)} = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix} \qquad \mathbf{s}^{(2)} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix}.$$

Clearly we now have $Q = 2$ and $N = 4$. Hebb rule gives the weight matrix:

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -2 \\ 2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \end{bmatrix}$$

assuming that the NN has no self connections.

We now test the network with an input vector $\mathbf{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$ that has a Hamming distance of 2 from either of the two stored vectors. We again assume that the neurons update their activations in the following random order: 1, 4, 3, 2.

So we set $\mathbf{y} = \mathbf{x}$. We first update the activation for Y_1:

$$y_{\text{in},1} = x_1 + \sum_{j=1}^{4} y_j w_{j1} = 0 + 2 = 2 \quad \Rightarrow y_1 = 1.$$

Thus $\mathbf{y} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$.

Next we update the activation for Y_4:

$$y_{\text{in},\,4} = x_4 + \sum_{j=1}^{4} y_j w_{j4} = 0 + 0 = 0 \quad \Rightarrow y_4 = 0.$$

Thus y_4 and therefore $\mathbf{y}$ remain unchanged.

Then we update the activation for Y_3:

$$y_{\text{in},\,3} = x_3 + \sum_{j=1}^{4} y_j w_{j3} = 1 + 2 = 3 \quad \Rightarrow y_3 = 1.$$

Thus y_3 and therefore $\mathbf{y}$ remain unchanged.

After that we update the activation for Y_2:

$$y_{\text{in},\,2} = x_2 + \sum_{j=1}^{4} y_j w_{j2} = 0 + 0 = 0 \quad \Rightarrow y_2 = 0.$$

Thus y_2 and therefore $\mathbf{y}$ remain unchanged.

The only possible update that can change the activations is with

Y_1:

$$y_{\text{in},\,1} = x_1 + \sum_{j=1}^{4} y_j w_{j1} = 1 + 2 = 3 \quad \Rightarrow y_1 = 1.$$

Thus y_1 and therefore $\mathbf{y}$ remain unchanged. It is clear that the net has converged to $\mathbf{y} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$. Unfortunately this is not one of the stored vectors. It is a spurious vector.

7. Example 3: Discrete Hopfield Net

Next we consider storing the same two binary vectors, but now work with their bipolar form

$$\mathbf{s}^{(1)} = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix} \qquad \mathbf{s}^{(2)} = \begin{bmatrix} 1 & -1 & 1 & 1 \end{bmatrix}.$$

The weight matrix is the same as before:

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -2 \\ 2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \end{bmatrix}.$$

We now test the network with the bipolar version of the same input vector as in example 2: $\mathbf{x} = \begin{bmatrix} -1 & -1 & 1 & -1 \end{bmatrix}$ that has a Hamming distance of 2 from either of the two stored vectors. Note that this input vector and the two stored vector are all mutually orthogonal. We again assume that the neurons update their activations in the following random order: 1, 4, 3, 2.

So we set $\mathbf{y} = \mathbf{x}$. We first update the activation for Y_1:

$$y_{\text{in},\,1} = x_1 + \sum_{j=1}^{4} y_j w_{j1} = -1 + 2 = 1 \quad \Rightarrow y_1 = 1.$$

Thus $\mathbf{y} = \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}$.

Next we update the activation for Y_4:

$$y_{\text{in},\,4} = x_4 + \sum_{j=1}^{4} y_j w_{j4} = -1 + 2 = 1 \quad \Rightarrow y_4 = 1.$$

Thus $\mathbf{y} = \begin{bmatrix} 1 & -1 & 1 & 1 \end{bmatrix}$, which is exactly $\mathbf{s}^{(2)}$.

Then we update the activation for Y_3:

$$y_{\text{in},3} = x_3 + \sum_{j=1}^{4} y_j w_{j3} = 1 + 2 = 3 \quad \Rightarrow y_3 = 1.$$

Thus y_3 and therefore $\mathbf{y}$ remain unchanged.

After that we update the activation for Y_2:

$$y_{\text{in},2} = x_2 + \sum_{j=1}^{4} y_j w_{j2} = -1 - 2 = -3 \quad \Rightarrow y_2 = -1.$$

Thus y_2 and therefore $\mathbf{y}$ remain unchanged.

The only possible update that can change the activations is with Y_1 or Y_4. However we find that there is no further changes in the activations. Thus the net has converged to the stored vector $\mathbf{s}^{(2)}$. Actually from our earlier discussions, we could have stopped the iteration as soon as the activation vector becomes one of the stored vectors (since the stored vectors are orthogonal to each other and so there is no cross-talk).

However, if we had updated the neurons in the order: 1, 2, 3, 4,

the first step will be the same, but the second step will give

$$y_{\text{in},\,2} = x_2 + \sum_{j=1}^{4} y_j w_{j2} = -1 + 2 = 1 \quad \Rightarrow y_2 = 1.$$

Thus $\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$, which is exactly $\mathbf{s}^{(1)}$. Thus the net converges to $\mathbf{s}^{(1)}$ instead of $\mathbf{s}^{(2)}$.

We see that although the discrete Hopfield net always converges to one of the attractors, exactly which attractor it converges to actually depends on the updating sequence.

References

[1] See Chapter 3 in Laurene Fausett, "Fundamentals of Neural Networks - Architectures, Algorithms, and Applications", Prentice Hall, 1994.