**POLYTECHNIC UNIVERSITY**
**Department of Computer and Information Science**

# Iterative Autoassociative Net: Bidirectional Associative Memory

**K. Ming Leung**

**Abstract:** Iterative associative neural networks are introduced. In particular the Bidirectional Associative Memory net is discussed.

## Directory




Last Revision Date: March 21, 2006

# Table of Contents

# 1. Motivation

After storing a given pattern, an autoassociative net may still not be able to respond immediately to an input signal with a stored pattern, but the response may be close to a stored pattern to suggest its use as input to the net again. In a sense, we are connecting the output units back to the input units. This results in a recurrent autoassociative net.

To illustrate the basic idea behind such a recurrent net, we consider the following simple example.

Suppose we want to store a single pattern: $\begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$. Hebb rule gives the weight matrix:

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

if the NN has no self connections.

We first consider the case where the NN has no self connections. The original pattern is of course stored successfully by the rule. Now

suppose we input a pattern with 3 missing values, like $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$, we find

$$\mathbf{y}_{\text{in}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & -1 \end{bmatrix}$$

and so $\mathbf{y} = \begin{bmatrix} 0 & 1 & 1 & -1 \end{bmatrix}$, which is close to but not the same as the stored pattern $\begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$. Then we feed this output pattern into the NN as input. This gives

$$\mathbf{y}_{\text{in}} = \begin{bmatrix} 0 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 2 & -2 \end{bmatrix}$$

and so $\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$, which is precisely the stored pattern.

On the other hand, if the NN is allowed to have self connections, it is easy to see that inputs $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$, and $\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$ all produce the same correct output immediately.

However, the input $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ converges to the negative of the stored pattern $\begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}$ in one step. So the NN behaves better if self connections are eliminated.

## 2. Bidirectional Associative Memory (BAM)

Bidirectional associative memory (BAM), developed by Kosko in 1988 and 1992, has the following features:

1. Its architecture consists of two layers: an $\mathbf{X}$ layer containing $N$ neurons and a $\mathbf{Y}$ layer containing $M$ neurons.
2. The transfer function is taken to be

$$f_{\text{BAM}}(y_{\text{in,j}}) = \begin{cases} 1 & \text{if } y_{\text{in,j}} > 0, \\ \text{previous } y_j & \text{if } y_{\text{in,j}} = 0, \\ -1 & \text{if } y_{\text{in,j}} < 0. \end{cases}$$

Notice that if the net input into a neuron is exactly 0, then its activation is kept the same as its previous value.

3. A set of input and target patterns $\mathbf{s}^{(q)} : \mathbf{t}^{(q)}, q = 1, 2, \ldots, Q$ is given, where the input vectors $\mathbf{s}^{(q)}$ have $N$ components,

$$\mathbf{s}^{(q)} = \begin{bmatrix} s_1^{(q)} & s_2^{(q)} & \ldots & s_N^{(q)} \end{bmatrix},$$

and their targets $\mathbf{t}^{(q)}$ have $M$ components,

$$\mathbf{t}^{(q)} = \begin{bmatrix} t_1^{(q)} & t_2^{(q)} & \ldots & t_M^{(q)} \end{bmatrix}.$$

4. These patterns are stored using Hebb rule to give an $N \times M$ weight matrix, $\mathbf{W}$, so that

$$w_{ij} = \sum_{q=1}^{Q} s_i^{(q)} t_j^{(q)}.$$

Biases are not used at all.

5. Inputs can be applied to either layers.

6. Signals are sent back and forth between the 2 layers until all neurons reach equilibrium (all activations remain the same for 2 successive steps).

7. Hebb rule gives a weight matrix $\mathbf{W}$ whose elements are $w_{ij} = \sum_{q=1}^{Q} s_i^{(q)} t_j^{(q)}$. $\mathbf{W}$ is used as the weight matrix for signals sent from the $\mathbf{X}$ to the $\mathbf{Y}$ layer, and $\mathbf{W}^T$ is used as the weight matrix for signals sent from the $\mathbf{Y}$ to the $\mathbf{X}$ layer.

The BAM algorithm is:

1 Use Hebb rule to store a set of $Q$ associated vector pairs.

2 For a given input test vector, $\mathbf{x}$ do steps a - c.

a Set activations of the $X$ layer to the input test pattern.

b Set activations of the $Y$ layer to 0.

c While activations have not converged do steps i - iii

i Update Y-layer activations

$$y_{\text{in},j} = \sum_{i=1}^{N} x_i w_{ij} \qquad y_j = f_{\text{BAM}}(y_{\text{in},j}).$$

ii Update X-layer activations

$$x_{\text{in},i} = \sum_{j=1}^{M} w_{ij} y_j \qquad x_i = f_{\text{BAM}}(x_{\text{in},i}).$$

iii Test for convergence of activations in both layers.

The above algorithm assumes that the test pattern is input into the X-layer. We can also input a test pattern into the Y-layer. In that case we need to exchange the roles of the X- and the Y-layers.

If we adopt matrix notation and represent all input vectors as a row vector ( a $1 \times N$ matrix), then the activation of the Y-layer is written as $\mathbf{y}_{\text{in}} = \mathbf{x}\mathbf{W}$, since $\mathbf{W}$ is an $N \times M$ matrix. On the other hand, the activation of the X-layer is written as $\mathbf{x}_{\text{in}} = \mathbf{y}\mathbf{W}^T$.

## 2.1. BAM to Associate Letters with Simple Bipolar codes

We want to use BAM to associate the letters, A and C, with simple bipolar codes:

```
.#.
#.#
###
#.#
#.#
```

with $\mathbf{t}^{(1)} = \begin{bmatrix} -1 & 1 \end{bmatrix}$ and

```
.##
#..
#..
#..
.##
```

with $\mathbf{t}^{(2)} = \begin{bmatrix} 1 & 1 \end{bmatrix}$. For each of the 2 letters, we replace "." with $-1$ and "#" with 1, and concatenate the bits row-wise to form a bipolar row vector. Thus we have

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

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

Notice that these 2 vectors have a square magnitude of $N = 15$ and are as uncorrelated as can be since $\mathbf{s}^{(1)} \cdot \mathbf{s}^{(2)} = 1$ and $N$ is odd.

Hebb rule gives the weight matrix

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

In general, given 2 bipolar vectors $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$, $\mathbf{s}^{(2)} + \mathbf{s}^{(1)}$ is non-bipolar vector whose elements are given by 0 if the corresponding

elements in $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ are different, and by twice the value in $\mathbf{s}^{(2)}$ if the corresponding elements are the same. Also $\mathbf{s}^{(2)} - \mathbf{s}^{(1)}$ is non-bipolar vector whose elements are given by 0 if the corresponding elements in $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ are the same, and by twice the value in $\mathbf{s}^{(2)}$ if the corresponding elements are different.

Explicitly we have

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

First we check to see if the NN gives the correct output at the Y-layer if pattern A or C is input to the X-layer. We find that with

$\mathbf{x} = \mathbf{s}^{(1)}$

$$\begin{aligned} \mathbf{y}_{\text{in}} &= \mathbf{s}^{(1)}\mathbf{W} = \left[\, \mathbf{s}^{(1)}(\mathbf{s}^{(2)T} - \mathbf{s}^{(1)T}) \quad \mathbf{s}^{(1)}(\mathbf{s}^{(2)T} + \mathbf{s}^{(1)T}) \,\right] \\ &= \left[\, 1-15 \quad 15+1 \,\right] = \left[\, -14 \quad 16 \,\right] \quad \Rightarrow \quad \mathbf{y} = \left[\, -1 \quad 1 \,\right], \end{aligned}$$

and with $\mathbf{x} = \mathbf{s}^{(2)}$

$$\begin{aligned} \mathbf{x}_{\text{in}} &= \mathbf{s}^{(2)}\mathbf{W} = \left[\, \mathbf{s}^{(2)}(\mathbf{s}^{(2)T} - \mathbf{s}^{(1)T}) \quad \mathbf{s}^{(2)}(\mathbf{s}^{(2)T} + \mathbf{s}^{(1)T}) \,\right] \\ &= \left[\, 15-1 \quad 1+15 \,\right] = \left[\, 14 \quad 16 \,\right] \quad \Rightarrow \quad \mathbf{y} = \left[\, 1 \quad 1 \,\right]. \end{aligned}$$

These are the correct results at the Y-layer. Notice that the above results are independent of the actual initial activations in the Y-layer, since none of the net input signal happens to be 0.

In the opposite direction, we want to input the targets at the Y-layer and see if the correct responses are obtained at the X-layer. We find that with $\mathbf{y} = \mathbf{t}^{(1)}$

$$\begin{aligned} \mathbf{x}_{\text{in}} &= \mathbf{t}^{(1)}\mathbf{W}^T = \left[\, -1 \quad 1 \,\right] \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} \\ &= -\mathbf{s}^{(2)} + \mathbf{s}^{(1)} + \mathbf{s}^{(1)} + \mathbf{s}^{(2)} = 2\mathbf{s}^{(1)} \qquad \Rightarrow \qquad \mathbf{x} = \mathbf{s}^{(1)}, \end{aligned}$$

and with $\mathbf{y} = \mathbf{t}^{(2)}$

$$\begin{aligned} \mathbf{x}_{\text{in}} &= \mathbf{t}^{(2)}\mathbf{W}^T = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} \\ &= \mathbf{s}^{(2)} - \mathbf{s}^{(1)} + \mathbf{s}^{(1)} + \mathbf{s}^{(2)} = 2\mathbf{s}^{(2)} \qquad \Rightarrow \qquad \mathbf{x} = \mathbf{s}^{(2)}. \end{aligned}$$

These are the correct activations at the X-layer. Notice that the above results are independent of the actual initial activations in the X-layer since none of the net input signal happens to be 0.

In either of the cases above, iteration converges after just one back and forth passage of signals.

## 2.2. BAM with Noisy Input

Next let us test the BAM with noisy input. Suppose the activation on Y is $\begin{bmatrix} 0 & 1 \end{bmatrix}$, and nothing is known about the activation on X so we put

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We find that

$$\mathbf{x}_{\text{in}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{W}^T = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} = \mathbf{s}^{(2)} + \mathbf{s}^{(1)}.$$

From what we said about the sum of any 2 bipolar vectors, we see that the activation on X is

$$\mathbf{x} = f_{\text{BAM}}(\mathbf{x}_{\text{in}}) = \frac{1}{2}\left(\mathbf{s}^{(2)} + \mathbf{s}^{(1)}\right).$$

This signal has to be sent back to the Y-layer to get

$$\begin{aligned} \mathbf{y}_{\text{in}} &= \frac{1}{2}\left(\mathbf{s}^{(2)} + \mathbf{s}^{(1)}\right) \begin{bmatrix} \mathbf{s}^{(2)T} - \mathbf{s}^{(1)T} & \mathbf{s}^{(2)T} + \mathbf{s}^{(1)T} \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} 0 & \mathbf{s}^{(2)}\mathbf{s}^{(2)T} + \mathbf{s}^{(1)}\mathbf{s}^{(1)T} + 2\mathbf{s}^{(2)}\mathbf{s}^{(1)T} \end{bmatrix} = \begin{bmatrix} 0 & N+1 \end{bmatrix}. \end{aligned}$$

This gives

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

which is exactly the input on Y that we started with. So the BAM has converged, but sadly to a spurious stable state. This is certainly not a surprising result at all since with knowledge of only the second

element of $\mathbf{y}$, there is no way to expect the NN (or humans) to tell which of the 2 characters we are referring to.

So far we have followed closely the textbook. However there is a very troubling result of this NN that even with knowledge of the first element of $\mathbf{y}$, this NN still cannot tell which of the 2 characters we are referring to. We humans should have no problem with that at all.

To see that we repeat the above calculation, however this time with the activation on Y given by $\begin{bmatrix} 1 & 0 \end{bmatrix}$ We find that

$$\mathbf{x}_{\text{in}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{W}^T = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} = \mathbf{s}^{(2)} - \mathbf{s}^{(1)}.$$

From what we said about the difference of any 2 bipolar vectors, we see that the activation on X is

$$\mathbf{x} = f_{\text{BAM}}(\mathbf{x}_{\text{in}}) = \frac{1}{2}\left(\mathbf{s}^{(2)} - \mathbf{s}^{(1)}\right).$$

This signal has to be sent back to the Y-layer to get

$$\begin{aligned}\mathbf{y}_{\text{in}} &= \frac{1}{2}\left(\mathbf{s}^{(2)}-\mathbf{s}^{(1)}\right)\left[\begin{array}{cc}\mathbf{s}^{(2)T}-\mathbf{s}^{(1)T} & \mathbf{s}^{(2)T}+\mathbf{s}^{(1)T}\end{array}\right]\\ &= \frac{1}{2}\left[\begin{array}{cc}\mathbf{s}^{(2)}\mathbf{s}^{(2)T}+\mathbf{s}^{(1)}\mathbf{s}^{(1)T}-2\mathbf{s}^{(2)}\mathbf{s}^{(1)T} & 0\end{array}\right]=\left[\begin{array}{cc}N-1 & 0\end{array}\right].\end{aligned}$$

This gives

$$\mathbf{y}=\left[\begin{array}{cc}1 & 0\end{array}\right],$$

which is exactly the input on Y that we started with. So with this input, the BAM has again converged, but sadly to a spurious stable state.

However given that the first element of $\mathbf{y}$ is 1 without any information on the second element, we should know that we are referring to character 'C' and not 'A'. There is something about this NN that is not satisfactory at all. The main problem is that biases were not considered. It turns out that they are important for its proper operation.

## 2.3. BAM with Biases

As before Hebb rule gives a weight matrix $\mathbf{W}$ whose elements are $w_{ij} = \sum_{q=1}^{Q} s_i^{(q)} t_j^{(q)}$ with $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. $\mathbf{W}$ is used as the weight matrix for signals sent from the $\mathbf{X}$ to the $\mathbf{Y}$ layer, and $\mathbf{W}^T$ is used as the weight matrix for signals sent from the $\mathbf{Y}$ to the $\mathbf{X}$ layer. The biases for the Y-layer are given by $b_j^{(Y)} = \sum_{q=1}^{Q} t_j^{(q)}$, with $j = 1, 2, \ldots, M$. Since signals are sent back to the X-layer, there are biases for the X-layers. These are given by $b_i^{(X)} = \sum_{q=1}^{Q} s_i^{(q)}$, with $i = 1, 2, \ldots, N$.

The algorithm for the BAM with biases is:

1 Use Hebb rule to store a set of $Q$ associated vector pairs.

2 For a given input test vector, $\mathbf{x}$ do steps a - c.

a Set activations of the $X$ layer to the input test pattern.

b Set activations of the $Y$ layer to 0.

c While activations have not converged do steps i - iii

i Update Y-layer activations

$$y_{\text{in},j} = \sum_{i=1}^{N} x_i w_{ij} + b_j^{(Y)} \qquad y_j = f_{\text{BAM}}(y_{\text{in},j}).$$

ii Update X-layer activations

$$x_{\text{in},i} = \sum_{j=1}^{M} w_{ij} y_j + b_i^{(X)} \qquad x_i = f_{\text{BAM}}(x_{\text{in},i}).$$

iii Test for convergence of activations in both layers.

Again the above algorithm assumes that the test pattern is input

into the X-layer. We can also input a test pattern into the Y-layer. In that case we need to exchange the roles of the X- and the Y-layers.

We now re-visit the character association example but this time with biases. We find that

$$\mathbf{b}^{(Y)} = \begin{bmatrix} 0 & 2 \end{bmatrix}, \qquad \mathbf{b}^{(X)} = \mathbf{s}^{(2)} + \mathbf{s}^{(1)}.$$

Again we first check to see if the NN gives the correct output at the Y-layer if pattern A or C is input to the X-layer. We find that with $\mathbf{x} = \mathbf{s}^{(1)}$

$$\mathbf{y}_{\text{in}} = \mathbf{s}^{(1)}\mathbf{W} + \mathbf{b}^{(Y)} = \begin{bmatrix} -14 & 16 \end{bmatrix} + \begin{bmatrix} 0 & 2 \end{bmatrix} = \begin{bmatrix} -14 & 18 \end{bmatrix}$$

and therefore $\mathbf{y} = \begin{bmatrix} -1 & 1 \end{bmatrix}$, and with $\mathbf{x} = \mathbf{s}^{(2)}$

$$\mathbf{y}_{\text{in}} = \mathbf{s}^{(2)}\mathbf{W} + \mathbf{b}^{(Y)} = \begin{bmatrix} 14 & 16 \end{bmatrix} + \begin{bmatrix} 0 & 2 \end{bmatrix} = \begin{bmatrix} 14 & 18 \end{bmatrix}$$

and therefore $\mathbf{y} = \begin{bmatrix} 1 & 1 \end{bmatrix}$. These are the intended associations. Notice that the above results are independent of the actual initial activations in the Y-layer, since none of the net input signal happens to be 0.

In the opposite direction, we want to input the targets at the Y-layer and see if the correct responses are obtained at the X-layer. We find that with $\mathbf{y} = \mathbf{t}^{(1)}$

$$\begin{aligned}\mathbf{x}_{\text{in}} &= \mathbf{t}^{(1)}\mathbf{W}^T + \mathbf{b}^{(X)} = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} + \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \\ &= 3\mathbf{s}^{(1)} + \mathbf{s}^{(2)}\end{aligned}$$

and so $\mathbf{x} = \mathbf{s}^{(1)}$.

Similarly with $\mathbf{y} = \mathbf{t}^{(2)}$

$$\begin{aligned}\mathbf{x}_{\text{in}} &= \mathbf{t}^{(2)}\mathbf{W}^T + \mathbf{b}^{(X)} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} + \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \\ &= 3\mathbf{s}^{(2)} + \mathbf{s}^{(1)}\end{aligned}$$

and so $\mathbf{x} = \mathbf{s}^{(2)}$. These are the correct activations at the X-layer. Notice that the above results are independent of the actual initial activations in the X-layer since none of the net input signal happens to be 0.

In either of the cases above, iteration converges after just one back and forth passage of signals.

Next we test the BAM with noisy input at the Y-layer with activation $\begin{bmatrix} 0 & 1 \end{bmatrix}$, and no knowledge about the activations on X and so they are set to 0. We find that

$$\begin{aligned} \mathbf{x}_{\text{in}} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{W}^T + \mathbf{b}^{(X)} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} + \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \\ &= 2\left(\mathbf{s}^{(2)} + \mathbf{s}^{(1)}\right). \end{aligned}$$

The activation on X is therefore given by

$$\mathbf{x} = f_{\text{BAM}}(\mathbf{x}_{\text{in}}) = \frac{1}{2}\left(\mathbf{s}^{(2)} + \mathbf{s}^{(1)}\right).$$

This signal has to be sent back to the Y-layer to get

$$\begin{aligned} \mathbf{y}_{\text{in}} &= \frac{1}{2}\left(\mathbf{s}^{(2)} + \mathbf{s}^{(1)}\right) \begin{bmatrix} \mathbf{s}^{(2)T} - \mathbf{s}^{(1)T} & \mathbf{s}^{(2)T} + \mathbf{s}^{(1)T} \end{bmatrix} + \mathbf{b}^{(Y)} \\ &= \frac{1}{2}\begin{bmatrix} 0 & \mathbf{s}^{(2)}\mathbf{s}^{(2)T} + \mathbf{s}^{(1)}\mathbf{s}^{(1)T} + 2\mathbf{s}^{(2)}\mathbf{s}^{(1)T} \end{bmatrix} + \begin{bmatrix} 0 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 0 & N+3 \end{bmatrix}. \end{aligned}$$

This gives

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

which is exactly the input on Y that we started with. So the BAM has converged, but sadly to a spurious stable state. This is certainly not a surprising result at all since with knowledge of only the second element of $\mathbf{y}$, there is no way to expect the NN (or humans) to tell which of the 2 characters we are referring to.

However, if we start off with appropriate partial knowledge of the activation at the X-layer, the intended association can be made. For example with noisy input at the Y-layer with activation $\begin{bmatrix} 0 & 1 \end{bmatrix}$, and an initial activation at the X-layer

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

the output at the X-layer is then given by

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

which is neither $\mathbf{s}^{(1)}$ or $\mathbf{s}^{(2)}$. However with this as the input to the X-layer, the output at the Y-layer is $\begin{bmatrix} -1 & 1 \end{bmatrix}$. With this as the input to the Y-layer, the output at the X-layer is then $\mathbf{s}^{(1)}$. This in turn produces at the Y-layer the output $\begin{bmatrix} -1 & 1 \end{bmatrix}$. The iteration clearly converges to $\begin{bmatrix} -1 & 1 \end{bmatrix}$ at the X-layer and to $\mathbf{t}^{(1)}$ at the Y-layer.

Are the results reasonable? From the initial input at the Y-layer we cannot tell if one is referring to an 'A' or a 'C'. However, the input vectors $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ differ in the 3rd, 6th, 8th, 9th, 12th, 13th and 14th positions. Given that the only other given information is that the 6th neuron at the X-layer is initial "on", strongly suggests that it is the $\mathbf{s}^{(1)} : \mathbf{t}^{(1)}$ association that we want, and this is indeed what we obtain.

Next we repeat the above calculation with the activation on Y given by $\begin{bmatrix} 1 & 0 \end{bmatrix}$. This is the one where the previous BAM without biases failed to produce a reasonable output. Now we find that

$$\mathbf{x}_{\text{in}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{W}^T + \mathbf{b}^{(X)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s}^{(2)} - \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} + \mathbf{s}^{(1)} \end{bmatrix} + \mathbf{s}^{(2)} + \mathbf{s}^{(1)} = 2\mathbf{s}^{(2)}.$$

We see that the activation on X is

$$\mathbf{x} = f_{\text{BAM}}(\mathbf{x}_{\text{in}}) = \mathbf{s}^{(2)}.$$

This signal has to be sent back to the Y-layer. However for this input at the X-layer, we already know that the activation at the Y-layer is

$$\mathbf{y} = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

which is exactly $\mathbf{t}^{(2)}$. So the BAM has converged, and this time to the correct result, since given that the first element of $\mathbf{y}$ is 1, but without any information on the second element, we should know that we are referring to character 'C' and not 'A'. Our new BAM with biases perform well, thus showing the importance of including biases in the net.

# 3. Remarks

## 3.1. Hamming Distance and Correlation Encoding

For any 2 binary or bipolar vectors, $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, each of length $N$, the Hamming distance between the vectors is defined to be the number of corresponding bits that are different between the vectors, and it is denoted by $H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$. It is related to the dot-product between these vectors. Let us define $A(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$ to be the number of corresponding bits in $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ that agrees with each other. Then it is clear that for bipolar vectors

$$\mathbf{x}^{(1)} \cdot \mathbf{x}^{(2)} = A(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) - H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}),$$

because the corresponding bits are either the same, and so they contribute 1 to the dot-product, or they are different and so contribute a $-1$. A corresponding pair of bits must either agree or disagree, and so

$$N = A(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) + H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}).$$

Eliminating $A(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$ from these 2 expressions, and solving for $H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$ gives the normalized Hamming distance (our textbook calls it the averaged Hamming distance)

$$\frac{1}{N} H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \frac{1}{2}\left(1 - \frac{1}{N}\mathbf{x}^{(1)} \cdot \mathbf{x}^{(2)}\right).$$

The dot-product of 2 bipolar vectors ranges from $-N$ for 2 anti-parallel vectors to $N$ for 2 identical vectors. The dot-product is 0 if the 2 vector are orthogonal to each other. The normalized Hamming distance between 2 bipolar vectors ranges from 0 for 2 identical vectors to 1 for 2 anti-parallel vectors. It is equal to one half if the 2 vectors are orthogonal to each other.

In our example above, since $\mathbf{s}^{(1)} \cdot \mathbf{s}^{(2)} = 1$, and $\mathbf{t}^{(1)} \cdot \mathbf{t}^{(2)} = 0$, we

have

$$\frac{1}{N}H(\mathbf{s}^{(1)},\mathbf{s}^{(2)}) = \frac{1}{2}\left(1-\frac{1}{15}\right) = \frac{7}{15}.$$

$$\frac{1}{N}H(\mathbf{t}^{(1)},\mathbf{t}^{(2)}) = \frac{1}{2}.$$

These 2 normalized Hamming distances do not differ much from each other.

There is the idea of "correlation encoding" where a pair of input patterns that are separated by a small Hamming distance are mapped to a pair of output pattern that are also so separated, while a pair of input patterns that are separated by a large Hamming distance are mapped to a pair of output pattern that are also largely separated (dissimilar).

## 3.2. Erasing a Stored Association

The complement of a bipolar vector, $\mathbf{x}$, denoted by $\mathbf{x}^c$, is a vector formed from $\mathbf{x}$ by flipping all the bits. It is clear that storing pattern pair $\mathbf{s}^c : \mathbf{t}^c$ is the same as storing the pair $\mathbf{s} : \mathbf{t}$ since Hebb rule gives

exactly the same weight matrix. To erase a given association $\mathbf{s} : \mathbf{t}$, we need to store the pair $\mathbf{s}^c : \mathbf{t}$ or $\mathbf{s} : \mathbf{t}^c$.

## 3.3. Updating Strategies

Several strategies may be used to update the activations in a NN. The algorithm used here for the BAM uses a synchronous updating procedure, where all neurons within a layer update their activations simultaneously. Updating may also be asynchronous (only one neuron updates its activation at each step of the iteration) or subset asynchronous (a group of neurons updates all its member's activations at each stage).

## 3.4. Convergence of a BAM and a Lyapunov Function

The convergence of a BAM can be proved by introducing an energy or a Lyapunov function. A Lyapunov function must be a scalar function that is bounded below, and it value cannot increase with every step of the BAM algorithm. An appropriate choice for a Lyapunov function

is the average signal energy for a forward and a backward pass:

$$L = -\frac{1}{2}\left(\mathbf{x}\mathbf{W}\mathbf{y}^T + \mathbf{y}\mathbf{W}^T\mathbf{x}^T\right)$$

Since $\mathbf{x}\mathbf{W}\mathbf{y}^T$ and $\mathbf{y}\mathbf{W}^T\mathbf{x}^T$ are scalars that are the of each other, they must be identical. Thus we have

$$L = -\mathbf{x}\mathbf{W}\mathbf{y}^T = -\sum_{i=1}^{N}\sum_{j=1}^{M} x_i w_{ij} y_j.$$

For binary or bipolar neurons, $L$ is clearly bounded below by

$$L < -\sum_{i=1}^{N}\sum_{j=1}^{M} |w_{ij}|.$$

It can be proved that this Lyapunov function decreases as the net iterates, and therefore the iteration converges, for either synchronous or subgroup updates.

## 3.5. Storage Capacity

It can be shown that the upper bound on the memory of the present BAM is $\min(N, M)$. However, there was good evidence based on a combination of heuristics and exhaustive search that the storage capacity can be extended to $\min(2^N, 2^M)$ if an appropriate nonzero threshold values is chosen for each neuron.

# References

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