4- BIDIRECTIONAL ASSOCIATIVE MEMORY (BAM)

Several versions of the heteroassociative recurrent neural network, or bidirectional associative memory (BAM), developed by Kosko (1988, 1992a).

- A bidirectional associative memory [Kosko, 1988] stores a set of pattern associations by summing bipolar correlation matrices (an n by m outer product matrix for each pattern to be stored).
- The architecture of the net consists of two layers of neurons, connected by directional weighted connection paths.
- The net iterates, sending signals back and forth between the two layers until all neurons reach equilibrium (i.e., until each neuron's activation remains constant for several steps).
- Bidirectional associative memory neural nets can respond to input to either layer.
- Because the weights are bidirectional and the algorithm alternates between updating the activations for each layer, we shall refer to the layers as the X-layer and the Y-layer (rather than the input and output layers).

Figure : Bidirectional Associative Memory (BAM)
Architecture
- The single-layer nonlinear feedback BAM network (with heteroassociative content-addressable memory) has \( n \) units in its X-layer and \( m \) units in its Y-layer.
- The connections between the layers are bidirectional; i.e., if the weight matrix for signals sent from the X-layer to the Y-layer is \( W \), the weight matrix for signals sent from the Y-layer to the X-layer is \( W^T \).

4.1 Discrete BAM
The two bivalent (binary or bipolar) forms of BAM are closely related. In each, the weights are found from the sum of the outer products of the bipolar form of the training vector pairs. Also, the activation function is a step function, with the possibility of a nonzero threshold. The bipolar vectors improve the performance of the net.

- The weight matrix to store a set of input and target vectors \( s(p) : t(p) \), \( p = 1, \ldots, P \), where
  \[
  s(p) = (s_1(p), \ldots, s_i(p), \ldots, s_n(p)) ; \\
  t(p) = (t_1(p), \ldots, t_j(p), \ldots, t_m(p))
  \]
can be determined by the Hebb rule.

- The formulas for the entries depend on whether the training vectors are binary or bipolar. For binary input vectors, the weight matrix \( W = \{w_{ij}\} \) is given by
  \[
  w_{ij} = \sum_{p=1}^{P} (2s_i(p) - 1)(2t_j(p) - 1)
  \]
- For bipolar input vectors, the weight matrix \( W = \{w_{ij}\} \) is given by
  \[
  w_{ij} = \sum_{p=1}^{P} s_i(p)t_j(p)
  \]
**Activation Function:** The activation function for the discrete BAM is the appropriate step function, depending on whether binary or bipolar vectors are used.

- For binary input vectors, the activation function for the Y-layer is

  \[
  y_j = \begin{cases} 
  1 & \text{if } y_{inj} > 0 \\
  y_j & \text{if } y_{inj} = 0 \\
  0 & \text{if } y_{inj} < 0, 
  \end{cases}
  \]

  and the activation function for the X-layer is

  \[
  x_i = \begin{cases} 
  1 & \text{if } x_{inl} > 0 \\
  x_i & \text{if } x_{inl} = 0 \\
  0 & \text{if } x_{inl} < 0.
  \end{cases}
  \]

- For bipolar input vectors, the activation function for the Y-layer is

  \[
  y_j = \begin{cases} 
  1 & \text{if } y_{inj} > \theta_j \\
  y_j & \text{if } y_{inj} = \theta_j \\
  -1 & \text{if } y_{inj} < \theta_j,
  \end{cases}
  \]

  and the activation function for the X-layer is

  \[
  x_i = \begin{cases} 
  1 & \text{if } x_{inl} > \theta_i \\
  x_i & \text{if } x_{inl} = \theta_i \\
  -1 & \text{if } x_{inl} < \theta_i.
  \end{cases}
  \]

- Note that if the *net input is exactly equal to the threshold value*, the activation function "decides" to leave the activation of that unit at its previous value.

- The activations of all units are initialized to zero.

- The first signal is to be sent from the X-layer to the Y-layer. However, if the input signal for the X-layer is the zero vector, the input signal to the Y-layer will be unchanged by the activation function, and the process will be the same as if the first piece of information had been sent from the Y-layer to the X-layer.
Signals are sent only from one layer to the other at any step of the process, not simultaneously in both directions.

**Algorithm**

1- Initialize the weights to store a set of P vectors;
   initialize all activations to 0
2- For each testing input, do Steps 3-7.
3a- Present input pattern x to the X-layer, (i.e., set activations of X-layer to current input pattern).
3b- Present input pattern y to the Y-layer, (Either of the input patterns may be the zero vector.)
4- While activations are not converged, do Steps 5-7.
5- Update activations of units in Y-layer:
   Compute net inputs:
   \[ y.in_j = \sum_i w_{ij} x_i \]
   Compute activations:
   \[ y_j = f(y.in_j) \]
   Send signal to X-layer.
6- Update activations of units in X-layer:
   Compute net inputs:
   \[ x.in_i = \sum_j w_{ij} y_j \]
   Compute activations:
   \[ x_i = f(x.in_i) \]
   Send signal to Y-layer.
7- Test for convergence:
   If the activation vectors x and y have reached equilibrium, then stop; otherwise, continue.
4-2 Continuous BAM

A continuous bidirectional associative memory [Kosko, 1988] transforms input smoothly and continuously into output in the range [0, 1] using the logistic sigmoid function as the activation function for all units.

- For binary input vectors \((s(p), t(p))\), \(p = 1, 2, \ldots, P\), the weights are determined by the formula

\[
\sum_{p=1}^{P} (2s_i(p) - 1)(2t_j(p) - 1)
\]

- The activation function is the logistic sigmoid

\[
f(y_{in_j}) = \frac{1}{1 + \exp(-y_{in_j})}
\]

- where a bias is included in calculating the net input to any unit and corresponding formulas apply for the units in the X-layer.

\[
y_{in_j} = b_j + \sum_i w_{ij} x_i
\]

A number of other forms of BAMs have been developed. In some, the activations change based on a differential equation known as Cohen-Grossberg activation dynamics [Cohen & Grossberg, 1983]

Application
Example-11: A BAM net to associate letters with simple bipolar codes

Consider the possibility of using a (discrete) BAM network (with bipolar vectors) to map two simple letters (given by 5 x 3 patterns) to the following bipolar codes:

\[
\begin{align*}
\text{A:} & \quad \begin{bmatrix} \# \# \# \\ \# \# \# \\ \# \# \# \\ \# \# \# \\ \# \# \# \\ \end{bmatrix} \\
\text{C:} & \quad \begin{bmatrix} \# \# \# \\ \# \# \# \\ \# \# \# \\ \# \# \# \\ \# \# \# \\ \end{bmatrix}
\end{align*}
\]

The target output vector \(t\) for letter A is \([-1, 1]\) and for the letter C is \([1, 1]\).
The weight matrices are:

\[
\begin{pmatrix}
1 & -1 \\
-1 & 1 \\
1 & -1 \\
-1 & 1 \\
-1 & 1 \\
1 & -1 \\
-1 & 1 \\
-1 & 1
\end{pmatrix}
\quad \begin{pmatrix}
-1 & -1 \\
1 & 1 \\
1 & 1 \\
-1 & -1 \\
-1 & 1 \\
1 & 1 \\
-1 & 1 \\
1 & 1
\end{pmatrix}
\quad \begin{pmatrix}
0 & -2 \\
0 & 2 \\
2 & 0 \\
0 & 2 \\
0 & -2 \\
2 & 0 \\
0 & 2 \\
0 & 2
\end{pmatrix}
\]

To illustrate the use of a \textbf{BAM}, we first demonstrate that the net gives the correct \(y\) vector when presented with the \(x\) vector for either the pattern \(A\) or the pattern \(C\):

**INPUT PATTERN A**

\[-1\ 1 \ -1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -1 \ 1 \ -1 \ 1 \ 1\]

\(w = [-14\ 16] \Rightarrow [-1\ 1]\)

**INPUT PATTERN C**

\[-1\ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1\]

\(w = [14\ 16] \Rightarrow [1\ 1]\)

To see the bidirectional nature of the net, observe that the \(Y\) vectors can also be used as input. For signals sent from the \(Y\)-layer to the \(X\)-layer, the weight matrix is the transpose of the matrix \(W\), i.e., \(w^T\).

For the input vector associated with pattern \(A\), namely, \((-1, 1)\), we have

\[\begin{pmatrix}
-1 & 1
\end{pmatrix}^T = [-2 \ -2 \ -2 \ 2 \ -2 \ 2 \ 2 \ 2 \ -2 \ 2 \ 2 \ -2 \ 2\]

\[\Rightarrow [-1\ 1\ -1\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ 1\ -1\ 1\ -1\ 1\ 1\ 1\ -1\ 1\ 1\ -1\ 1\ -1\ 1\ -1\ 1\ 1]\]

This is pattern \(A\).

Similarly, if we input the vector associated with pattern \(C\), namely, \((1, 1)\), we obtain

\[\begin{pmatrix}
1 & 1
\end{pmatrix}^T = [-1 \ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ -1 \ 1 \ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ 1\]

which is pattern \(C\).