Bi-directional Associative Memory :- Kosko developed the Bi-directional associative Memory (BAM) in the year 1988. It is a hetro associative recurrent neural N/w consisting of two layers until each neuron's activation remains constant for several steps. (till it reaches equilibrium.) The net associates a set of patterns by summing bipolar correlation matrices. This type of net consists of two layers of neurons, which are connected by means of directional weighted connection paths. The iteration is carried out by sending signal back and forth b/w two layers until all neurons reach equilibrium. The net can respond to input on either layer.

The layers in this case are referred as X-layer and Y-layer instead of I/P and O/P layer, becoz the weights are bi-directional.

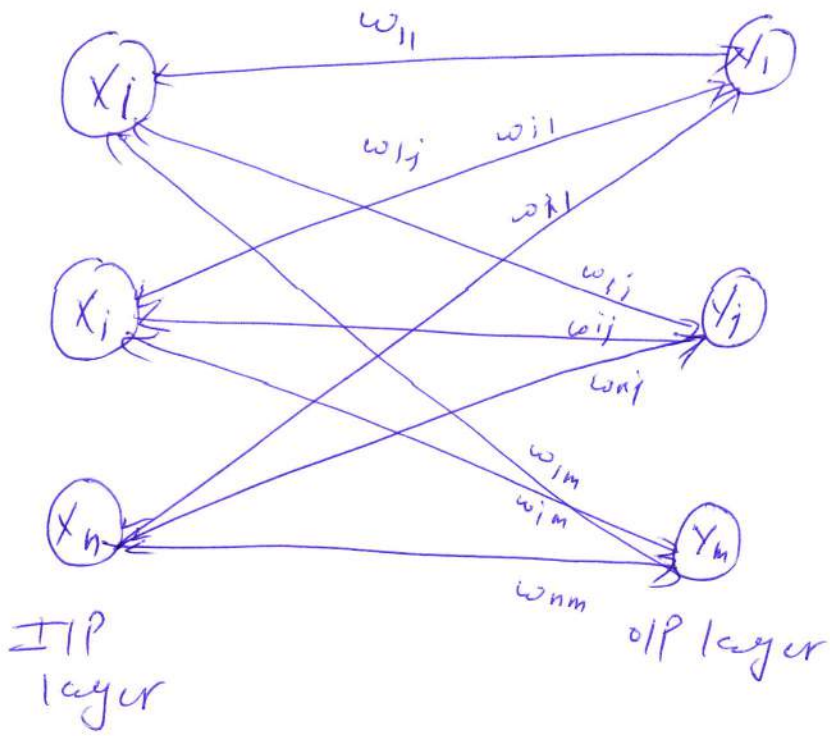
There forms of BAM are:

(i) Binary

(ii) Bipolar

(iii) Continuous

Architecture:- The hetero associative BAM n/w has 'n' units in X-layer and m-units in the Y-layer.. The connection b/w the layers are Bidirectional ie if the weight matrix for signal sent from the X-layer to Y layer is W, Then the weight matrix for signal sent from Y layer to X-layer is W^T.



Architecture of BAM

This architecture resembles a single layer feed forward n/w. It consist of one layer of weighted interconn^ns. There exist 'n' no. of I/P neurons in the input layer and 'm' no. of O/P neurons in the O/P layer. The training process is based on Hebb.

b/w, wherein the inputs and o/p's are different. The most important feature of BAM is that there exists weighted interconn^n from both X-layer to Y-layer and vice versa also. The weights are adjusted b/w X-layer to Y-layer and also from Y-layer to X-layer.

Types of Bi-directional Associative Memory.

i) Discrete BAM

ii) Continous BAM

(i) Discrete BAM:- The binary and bipolar forms of BAM are closely related. In each case the weights are found from the sum of the Hebb out product of the bipolar form of the training vector pairs. The activation f^n used is a step activation f^n with the non-zero threshold. But generally we know that Bipolar vectors improve the performance of the net.

The weight matrix to store the set of I/P and target vectors $s(P) : t(P), P = 1, \ldots P$ when

$$S(P) = (S_1(P), \ldots S_i(P), \ldots S_n(P))$$

and can be determined with the help of Hebb rule.

For binary vectors, the weight matrix can be determined by the formula.

$$w_{ij} = \sum_p (2s_i(p) - 1)(2t_j(p) - 1)$$

For bipolar input vectors, the weight matrix is given by.

$$w_{ij} = \sum_p s_i(p) t_j(p)$$

Activation Function:- The activation f^n for discrete BAM depends on whether binary or bipolar vectors are used. The activation f^n is an appropriate step f^n. For binary vectors, the activation f^n for the Y-layer is:-

$$Y_j = \begin{cases} 1 & \text{if } y\text{-in}_j > 0 \\ y_j & \text{if } y\text{-in}_j = 0 \\ 0, & \text{if } y\text{in}_j < 0 \end{cases}$$

and the activation f^n for the X-layer is

$$X_i = \begin{cases} 1 & \text{if } X\text{-in}_i > 0 \\ x_i & \text{if } X\text{-in}_i = 0 \\ 0 & \text{if } X_{in_i} < 0. \end{cases}$$

For <u>bipolar vector</u>, the activation f^n for Y-layer is:

$$y_j = \begin{cases} 1 & \text{if } y\_in_j > \theta_j \\ y_j & \text{if } y\_in_j = \theta_j \\ 0 & \text{if } y\_in_j < \theta_j \end{cases}$$

The activation f^n for the X layer is

$$X_i = \begin{cases} 1 & \text{if } x\_in_i > \theta_i \\ X_i, & \text{if } x\_in_i = \theta_i \\ 0 & \text{if } x\_in_i < \theta_i \end{cases}$$

<u>Note</u>:- That if the net input is equal to the Threshold value, the activation f^n decides to leave the activation of that unit at its previous value. In the θ_i, indicates the <u>Threshold</u> value

<u>Continuous BAM</u>:- The Continous BAM was introduced by Kosko, 1988.

A Continous BAM has the capability to transfer the input <u>smoothly</u> and <u>continuously</u> into the respective o/p in the range b/w [0,1]. The Continous BAM uses <u>logistic</u> Sigmoid <u>f^n</u> as the <u>activation</u> f^n for

For bipolar vector, The activation f^n for Y-layer is.

$$y_j = \begin{cases} 1, & \text{if } y_{inj} > \theta_j \\ y_j, & \text{if } y_{inj} = \theta_j \\ 0, & \text{if } y_{inj} = \theta_j \end{cases}$$

The the activation f^n for the X-layer is

$$X_j = \begin{cases} 1 & \text{if } X_{inj} > \theta_j \\ X_j & \text{if } X_{inj} = \theta_j \\ 0 & \text{if } X_{inj} < \theta_j \end{cases}$$

Note that if the net input is equal to the threshold value, the activation f^n decided to leave the activation of that unit at its previous value, In the θ_j, indicates the threshold value.

2. Continuous BAM:- The Continuous BAM was introduced by Kosko, 1988. A Continuous BAM has the Capability to tranfer the input smoothly and continuously into the respective O/P in the range b/w [0, 1]. The Continous BAM uses logistic sigmoid f^n as the activation. f^n for all unit

For binary input vectors $(s(P), t(P))$, The weights are determined

$$w_{ij} = \sum_p (2 s_i(p) - 1)(2 t_j(p) - 1)$$

<u>Y-layer</u>: - The <u>logistic</u> <u>sigmoid</u> f^n is given by.

$$f(y_{inj}) = \frac{1}{1 + \exp(-y_{inj})}$$

If bias is included in calculated the net input then.

$$f(y_{inj}) = b_j + \sum x_i w_{ij}.$$

<u>X-layer</u>:-

The <u>logistic</u> <u>sigmoid</u> activation f^n is given by

$$f(x_{inj}) = \frac{1}{1 + \exp(-x_{inj})}$$

if bias is included in calculating the net input then.

$$x_{inj} = b_i + \sum y_j w_{ij}.$$

Application Algorithm: From the training process, obtain the the final weights. The input pattern are presented to both X-layer and Y layer. The net I/P and the activation ~~to~~ the Y-layer are calculated. Then the signals are sent to X-layer and here its net input and activations are found. In this manner, the bi-directional associative memory is tested for its performance. The algorithm for the bi-directional memory net is given as follows.

Step1: Intialize the weight to store a set of P vectors. Initialize the weight to store a set of P vectors. Initialize all activations to 0.

Step2: For each testing input follow steps. 3-8

Step3: Set activation of X-layer to current input pattern.

Step4: Input pattern Y is presented to Y layer.

Step5: While activations are not converged do steps 6-8.

Step 6:- Activation unit in Y- layer and net input are computed as:-

Compute the net input $y_{in_j} = \sum_i w_{ij} x_i$.
Then compute the activations $x_i = f(x_{in_i})$
Send signals to the Y layer.

Step 8:- Test for convergence.
The stopping condition may be that the activation vectors X and Y have reach equilibrium.

Example:- Consider a BAM N/W (with bipolar vectors) to map two simple letters (given by 5x3 patterns) to the following bipolar target codes.

```
* * *        * . *
*   .        *   *
* * *        * * *
*   .        *   *
* * *        *   *
```

Pattern (E) (-1,1) Pattern (H) (1,1)

a) Find the weight matrix with input pattern E and H.
(b) Obtain the response of the net E and H as o/p

it is presented with a input vector Y, which is a noisy version of one of the training Y vectors and no info^n is given about the corresponding X vector, the noisy version of Y is given by [0,1].

(d) Also, the test the Performance of net when the following I/P - O/P vector is presented.

$X = [0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1]$

$Y = [0, 1]$

Sol^n:

(a) Step 0: To find the weight matrix '*' will be marked as 1 and '.' will be marked as -1. To store E, the target is [-1,1] and to store H, the target is (1,1). The individual weight matrices of the respective patterns are given as by W_1 and W_2.

W = weight matrix of 'E' + weight matrix of H

$= W_1 + W_2$.

$$\begin{bmatrix} -1 & 1 \\ -1 & -1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & +1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \end{bmatrix}$$

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

(b) To test the net with input pattern E

$$Y = XW$$

$$[1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1]W$$

$$= [-2 \quad -2 \quad -2 \quad -2 \quad 2+2+2+2+2+2+2+2+2+2+2+2]$$

$$= [-8 \quad 22]$$

Applying bipolar step activation f^n.

$f[-8 \quad 22] \Rightarrow [-1 \quad 1]$ which is the correct response for E.

To test the net with input pattern H.

$$Y = XW$$

$$[1 \quad -1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad -1 \quad 1]W$$

$$= [2+2+2+2+2+2+2+2+2+2+2+2+2+2]$$

$$= [8 \quad 22]$$

Applying bipolar step activation f^n.

$f[8 \quad 22] \Rightarrow (1, 1)$ which is the correct response for H.

Alternatively, for the input vector associated with pattern E, as BAM is bidirectional in nature, Y vectors can also be used as input.

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

$$X = YW^T$$

$$= (-1\ 1)\ W^T$$

$$= [2\ \ 2\ \ 2\ \ 2\ \ -2\ \ -2\ \ 2\ \ 2\ \ 2\ \ 2\ \ -2\ \ -2\ \ 2+2\ \ 2]$$

$$= [1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1]$$

which is pattern H, the pattern is recognised.

(c) $X = YW^T = (0\ 1)\ W^T$

$$= [2\ \ 0\ \ 2\ \ 2\ \ -2\ \ 0\ \ 2\ \ 2\ \ 2\ \ 2\ \ -2\ \ 0\ \ 2\ \ 0\ \ 2]$$

$$X \rightarrow [1\ \ 0\ \ 1\ \ 1\ \ -1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ -1\ \ 0\ \ 1\ \ 0\ \ 1]$$

The vector X is sent back to the Y-layer using the weight matrix W.

$$Y = XW.$$

$$= [1\ \ 0\ \ 1\ \ 1\ \ -1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ -1\ \ 0\ \ 1\ \ 0\ \ 1]\ W$$

$$= [0\ \ 1]$$

Thus the net does not give any info^n about E or H. The net has converged to a spurious stable state ie, the sol^n is not one of the stored pattern pairs.

(d) The vector X is similar to input vector E.

$$(0\ 1)\ W^T$$

$= [2\ 0\ 2\ 2\ -2\ 0\ 2\ 2\ 2\ 2\ -2\ 0\ 2\ 0\ 2]$

Now since the algorithm specifies that if net input to a unit is zero the activation of the units remains unchanged.

we get

$[1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ 1\ -1\ 1\ 1\ 1\ 1]$ which is pattern E, Also

$$x_i = \begin{cases} 1 & \text{if } x_{ini} > 0 \\ x_i & \text{if } x_{ini} = 0 \\ -1 & \text{if } x_{ini} < 0 \end{cases}$$

<u>Retrieving</u> a <u>stored</u> <u>Association</u>:- Long term memory are stored in the weight array W and W'. Each memory consist of two vectors: A which appears at the o/p's of layer 1 and B and <u>associated</u> memory that is o/p of layer 2. To retrieve an <u>associated</u> memory, all or part of vector A is <u>momentarilly</u> forced onto the o/ps of layer 1. A is then removed and the n/w is allowed to <u>stabilize</u>, producing the associated vector B at the o/p of layer 2. B, then operates through the transpose matrix W' to produce

at the o/p of layer 1. Each pass around the loop causes the vector o/ps of layers 1 and 2 to come closer to the stored memory, until a stable point is reached and changes cease. This point be considered to constitute a resonance, as the vector back pass and forth b/w the layers, always reinforcing the current o/ps, but no longer changing them.

The state of the neurons represents a short-term memory (STM), as it may be changed quickly by applying another input vector, The values in the weight matrix form a long term memory (LTM) and are changable only on a longer time scale.

Encoding the associations:- A n/w is usually trained to recognize a set of memories. These comprise a training set, composed of vector pairs A and B. This n/w is trained by calculation, That is, the weight matrix is computed to be the products of all of the

Symbolically

$$W = \sum_i A_i' B_i$$

we assume that the memories consist of binary vectors. This limitations seems less severe if one remembers that the entire contents of the library of Congress could be encoded into a single rather lengthy binary vector) Also Kosko, has achieved better performance by making the vector <u>component</u> greater than <u>zero</u> becomes one and a component less than or equal to zero becomes minus one.

As an eg, let us suppose that we wish to train a n/w to remember three binary - vector pairs. In this eg., the vectors A_i, have the same no. of components as the vector B_i.. However this is not necessary to the algorithm, <u>associations</u> can be formed b/w vector of diff. size.

<u>Original vector</u>	<u>Associated vector</u>	<u>Bipolar vector</u>
$A_1 = (1,0,0)$	$A_1' = (1,-1,-1)$	$B_1' = (-1,-1,1)$
$A_2 = (0,1,0)$	$A_2' = (-1,1,-1)$	$B_2' = (-1,1,-1)$
$B_1 = (0,0,1)$		
$B_2 = (0,1,0)$	$A_3' = ($	

Computing the weight matrices.

$$W = A_1' B_1' + A_2' B_2' + A_3' B_3'$$

$$\begin{matrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & -1 \end{matrix} + \begin{matrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{matrix} + \begin{matrix} -1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & -1 \end{matrix} = \begin{matrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{matrix}$$

Now, applying an input vector $A_1 = (1, 0, 0)$ we compute the o/p vector O

$$O = A_1' W = (1 \ 0 \ 0) \begin{matrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{matrix} = -1 \ -1 \ 3$$

Next, applying the threshold value

$b_i = 0$ if $O_i > 0$

$b_i = 1$ if $O_i < 0$

b_i unchanged if $O_i = 0$

$B_1' = (0 \ 0 \ 1)$ which is the desired association. Then applying B_1' around the loop to W'

$$B_1' W' = [0 \ 0 \ 1] \begin{matrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{matrix} = [3 \ -1 \ -1]$$

which becomes $(1, 0, 0)$ after thresholding, restoring the value of A_1

Storage Capacity:- Storage capacity of a neural net refers to the quantity of info^n that can be stored in a n/w in such a way that it it can be retrieved without error. The storage capacity can be measured as.

$$C = \frac{\text{no. of stored patterns}}{\text{No. of neurons in the n/w.}}$$

If a n/w is not fully connected, then another measure of capacity may be more appropriate such as:-

$$C = \frac{\text{No. of stored patterns}}{\text{No. of conn}^n \text{ wights in the n/w}}$$

The storage capacity of an associated memory neural n/w is the no. patterns or pattern pairs that can be stored before the net begins to forget.

Adaptive Resonance Theory (ART)

ART refers to a class of self organizing neural architectures that cluster the pattern space and produce appropriate weight vector templates.

Conventional artificial neural n/ws have failed to solve the stability-plasticity-dilemma. A n/w remains open to new learning (remain plastic) without washing away previously learned codes. Too often, learning a new pattern erases or modifies previous training. If there is only a fixed set of training vectors, the n/w can be cycled through there repeatedly and may eventually learn all. In a real n/w, it will exposed to a constantly changing environment: it may never see the same training vector twice. Under such conditions, BPN will learn nothing. It will continuously modify its weights to no avail, never arriving at satisfactory settings.

ART n/ws and algorithms maintain the plasticity required to learn new patterns, while preventing the modification of

ART Fundamentals:- ART requires a no. of neurons in addition to the input units, cluster units and units for the comparison of the input signal with the cluster units weights. The fundamental of ART is studied by knowing its basic architecture, operation its way of learning and basic steps involving in training.

Basic Architecture:-

The basic architecture of adaptive resonance neural net involves three groups of neurons.

1). Input processing field - F_1 layer.

2). Cluster units - F_2 layer.

3). Reset mechanism - That controls the degree of similarity of patterns placed on same cluster.

1). Input Processing (F_1 layer) It divided into two portitions.

(i) Input portion denoted as (F_1 (a))

(ii) Interface portion (denoted as F_1 (b))

The input portion just appears the input vector given, but some processing in ART2 N/w.

The interface portion combines the signal from the input portion with that of F_2 layer. That signal is used in comparing the similarity of input signal to the weight vector for the cluster unit which is selected for learning

$F_1(b)$ layer is connected to F_2 layer through bottom up weight b_{ij} and F_2 layer is connected to $F_b(b)$ layer through top down weights t_{ji}

2). Cluster units (F_2 layer). This is a competitive layer. The cluster unit with largest net input is selected to learn the input pattern. The activation of all other F_2 units are set to zero. The interface units now combine inf^n from the input & cluster units.

3). Reset mechanism:- The inf^n from the input cluster units is combined in the interface units. Depending on the similarity b/w the top-down weight & the input vector, the cluster may or may not be allowed to learn the pattern. This is done at the reset signals it receives

of F_1 layer. If, the cluster unit is not allowed to learn, it is inhibited and a new cluster unit is selected as the Candidate.

Basic operation:- The basic operation of ART is must before entering into the op^n of ART1 or ART2 n/w.

1). Learning Trail:- It consist of presentation of one input pattern to the n/w.

Before the pattern is presented.

(i) The activation of all units should be zero.

(ii) The F_2 units are made inactive.

When a pattern is given to the n/w, it continuously sends input signal, till learning trail is completed.

2). Controlling the degree of Similarity:- It is controlled by vigilance parameter. it is denoted as ρ (in the range from 0 to 1)

3) Reset mechanism states:- The reset mechanism differs for ART1 and ART2. Its f^n is to control the state of node in F_2 layer. The F_2 layer

States mentioned here.

(i) Active :- 'ON'. The unit in F_2 is on. Its its activation is given by d, where d=1 for ART1 and d=0 to 1 for ART2

(ii) Inactive - 'OFF' the unit F_2 is off. Its activation = 0, but it is available to participate in the next Competition.

(iii) Inhibit :- 'OFF'. The unit F_2 is off. Its activation = 0, and is prevented from participation in further Competition during the presentation of current input vector.

Learning in ART :- There are two types of learning in ART viz.

1). Fast learning :- It is used in ART1 N/w. Here input is binary. The weight changes during the resonance period occur more rapidly. The n/w is stablized each time the pattern has to choose the correct Cluster unit when it is presented. The weight vector from the learning is more similar to that of the input.

2). Slow Learning :- It is used in ART 2 n/w. The weight changes during resonance period, occur slowly. The weights do not reach equilibrium in each learning trail, hence many more presentation of the learning pattern are required. The weight produced are Continously Changing. So the n/w will not be stablized. It will become stablized, only after presenting a large no. of inputs. The method for finding the equilibrium is also very difficult. The weight produced by slow learning may be more appropriate that those produced by fast learning for certain types of data.

Basic Training Steps :- The steps that occur during the training of ART 1 n/w are :-

Step1 :- Initialise the parameter.

Step2 : When stopping condition is false perform steps 3 – 10.

Step3 : For each I/P vector do steps 4-9

Step4 : F_1 layer process starts.

Step5 : When condition is true do

Step 6:- Find unit to learn current input pattern. ie F_2 unit with largest $I_{1,0}$

Step 7 $F_1(b)$ units combine their inputs from $F_1(a)$ and F_2.

Step 8:- Test for reset condition. If reset is true, then current candidate unit is rejected (inhibited) Return to step 5.

If reset is false, then the current candidate unit is accepted for learning, then do step 9.

Step 9:- Learning ~~Steps~~
Weight updation occurs as per the differential equations.

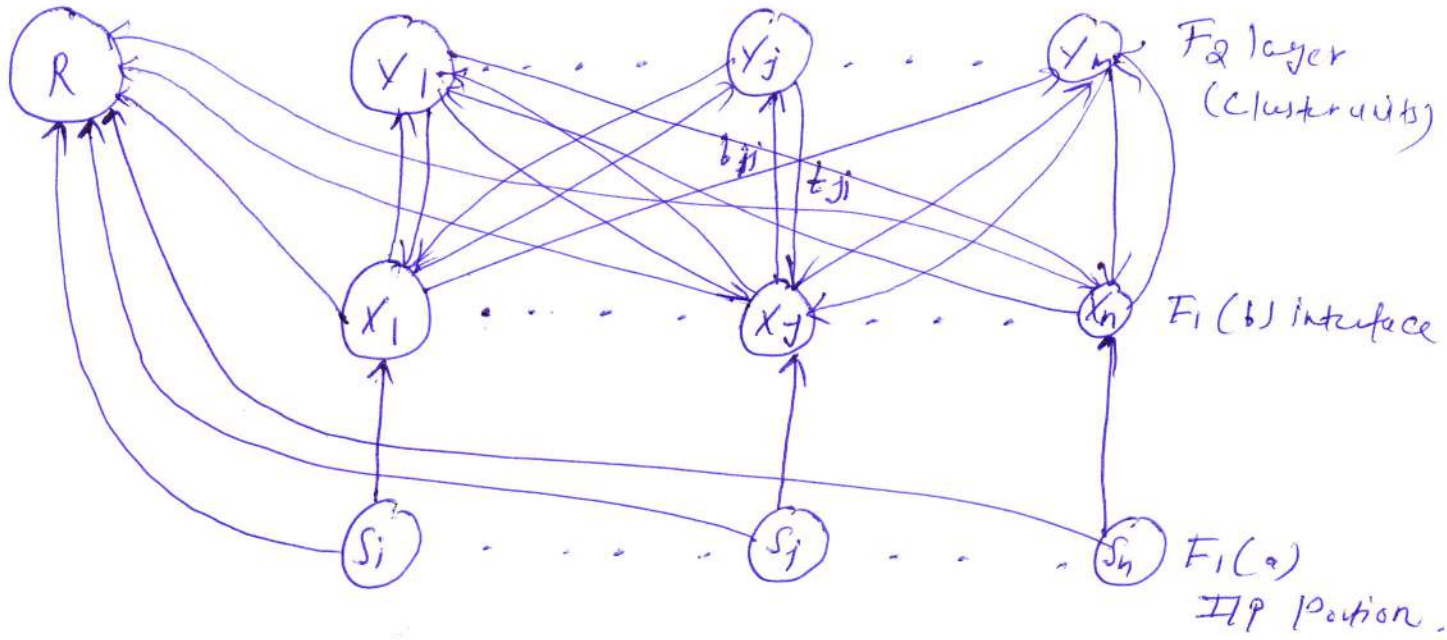
Step 10:- Test for stopping condition
In ART1 it is not required that all the patterns should be presented in the same order.

Art Classification

* ART 1
* ART 2

ART-1:- It is the first member of the ART family. It is a two-layer n/w that discovers pattern cluster templates in arbitrary boolean pattern sets.

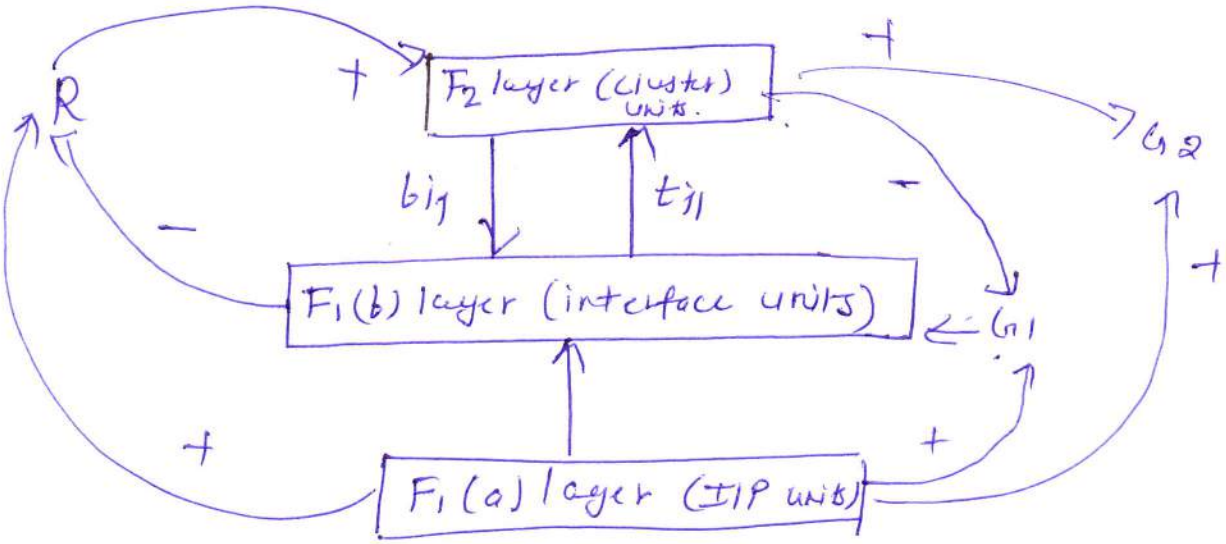
ART1 - Architecture:- ART-1 has computational units and supplemental units. Its architecture is shown in fig:-



Computational units:- The computational unit has F_1 and F_2 units and reset unit. The F_1 (a) is connected to F_1 (b) interface unit. Both input and interface units are connected to reset mechanism unit. By means of top-down and bottom-up weights, the interface layers [illegible] cluster units.

& their reciprocity is also achieved.

Supplemental Units:- The structure of supplemental unit is shown as:



Structure of Supplemental units

Difficulty posed in computational units (F_1, F_2 and reset unit) is that these units are required to respond differently at different stages of the process. Also the problem with the reset mechanism is that F_2 units must be inhibited under certain conditions, but returned to availability on subsequent learning trials.

To avoid such problems, two supplemented units (called gain control units G1 and G2) are present in addition to reset units R. ~~Must be~~ Inhibited under

Then special units receives signal from and send their signals to, all the units present in occupational structure. Excitatory signals are indicated by '+' and inhibitory signals by '-'. The signal may be sent, whenever any unit in interface or cluster layer has three sources from which it can receive a signal. Each of these units also receives two excitatory signals in order to be 'on'. Hence close to this, the requirement is called two-third rule. This rules plays a role in the choice of parameters & initial weights. The Reset unit R controls vigilance matching also.

Implementation Algorithm:- The parameters used in the algorithm are.

n — no. of components in I/P vector

m — Max. no. of clusters that can be formed.

b_{ij} — bottom up weights (from $f_1(b)$ to f_2 units)

t_{ji} — top down weights (from f_2 to $f_1(b)$ units)

ρ — vigilance parameter

S — binary input vector.

The training algorithm of ART1 n/w. is as follows.

Step 1: Initialize parameters.

$L > 1$ and $0 < \rho \leq 1$

Initialize weights.

$$0 < b_{ij}(0) < \frac{L}{L-1+n}, \quad t_{ji}(0) = 1$$

Step 2: While stopping condition is false, perform steps 3-14.

Step 3: For each training input do steps 4-13

Step 4:- Set activations of all F_2 units to 0

Set activations of $F_1(a)$ units to I/P vector S.

Step 5. Compute the norm s

$$\|S\| = \sum_i S_i$$

Step 6:- Send input signal from $F_1(a)$ to $F_1(b)$ layer

$$x_i = S_i$$

Step 7:- for each F_2 node that is not inhibited.

$$\text{if } y_j \neq -1 \text{ then } y_j = \sum_i b_{ij} x_i$$

Step 8: While reset is true, perform steps 9-12

Step 9 - Find j such that $y_j \geq y_{j'}$ for all

if $y_j = -1$ Then all the odds are inhibited and This pattern Can't be Clustered.

<u>Step 10</u>:- Recompute activation X of F_1 (b)

$$x_i = s_i \, t_{ji}$$

<u>Step 11</u>. Compute The norm vector X

$$||x|| = \sum_i x_i$$

<u>Step 12</u> Test for Reset

if $\frac{||x||}{||s||} < \rho$ Then.

$y_j = -1$ (inhibit node j) Continue step 8 again.

if $\frac{||x||}{||s||} \geq \rho$ Then proceed to Step 13.

<u>Step 13</u>. update weight for node j

$$b_{ij}(new) = \frac{L x_i}{L - 1 + ||x||}$$

$$t_{ji}(new) = x_i$$

<u>Step 14</u>:- Test for stopping Condition.

In winner section, if there is a tie, take i to be <u>smallest</u> such index. Also t_{ji} is either 0 or 1, and once it is set to 0 during learning, it can be never set back to 1 becoz of <u>stable</u> learning method.

The parameter used have typical values as shown below.

Parameter	Range	Typical value
L	$L > 1$	2
ρ	$0 < \rho \leq 1$ (vigilance parameter)	0.9
b_{ij}	$0 < b_{ij}(0) < \frac{L}{L-1+n}$ (Bottom up weights)	$\frac{1}{1+n}$
t_{ji}	$t_{ji}(0) = 1$ (top down weights)	1

<u>Example</u>. Consider an ART1 n/w with nine I/P (F_1) units and two clusters (F_2) units. After some training the bottom up weights (b_{ij}) and top down weights (t_{ji}) has the following values:

Bottom up weights b_{ij}

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \end{bmatrix}$$

Top-down weights t_{ji}

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The pattern [1 1 1 1 0 1 1 1 1] is presented to the N/w. Compute the action of N/w if vigilance parameter is 0.4

<u>Sol^n</u> The <u>vigilance parameter</u> is 0.4

<u>Step 1</u>: <u>Initialize parameters</u>.

$n = 9$, $m = 2$, $L = 2$, $\rho = 0.4$.

$$b_{ij} = \begin{bmatrix} \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{5} \end{bmatrix} \qquad t_{ji} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Step 2 :- Begin training.

Step 3: For input vector (1 1 1 1 0 1 1 1 1) do Steps 4 to 13.

Step 4. Set activations of all F_2 units 0

Set activations of $F_1(a)$ units to I/P Vector.

$S = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1)$

Step 5. Compute the norm of S.

$||S|| = \sum s_i$

$||S|| = 8$

Step 6: Input signal from $F_1(a)$ to $F_2(1)$

So

$X = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1)$

Step 7: Calculate the net I/P $y_j = \sum_i x_i b_{ij}$

$y_1 = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} = 2$

$y_2 = 8\left(\frac{1}{5}\right) = 1.6$

$y_1 > y_2$

$J = 1$

Step 8 :- when reset is true do steps 9-12

Step 9 :- y_1 unit has largest value ~~of~~

So $J = 1$

Step 10: - Recomputing activation of $F_1(b)$ layer.

$$X = S + cX$$

$$X_i = S_i = S_i t_{ji}$$

$$= (1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1)\ (1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)$$

Step 11: Calculating the norm vector X

$$\|X\| = \sum_i X_i$$

$$\|X\| = 4$$

Step 12: Test for Reset

$$\frac{\|X\|}{\|S\|} = \frac{4}{8} = 0.5 > 0.4 \quad \text{Hence } \frac{\|X\|}{\|S\|} > \rho$$

Since reset is false, go to step 13.

Step 13: updates the weights.

$$b_{ij}(new) = \frac{L X_i}{L-1+\|X\|}$$

$$b_{11}(new) = \frac{2 \times 1}{2-1+4} = \frac{2}{5} = 0.4$$ Calculating for all b_{ij}'s.

$$\begin{bmatrix} \frac{2}{5} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{5} \\ 0 & \frac{1}{5} \\ \frac{2}{5} & \frac{1}{5} \end{bmatrix}$$

The new top-down weight is

$$t_{ji}(new) = X_i$$

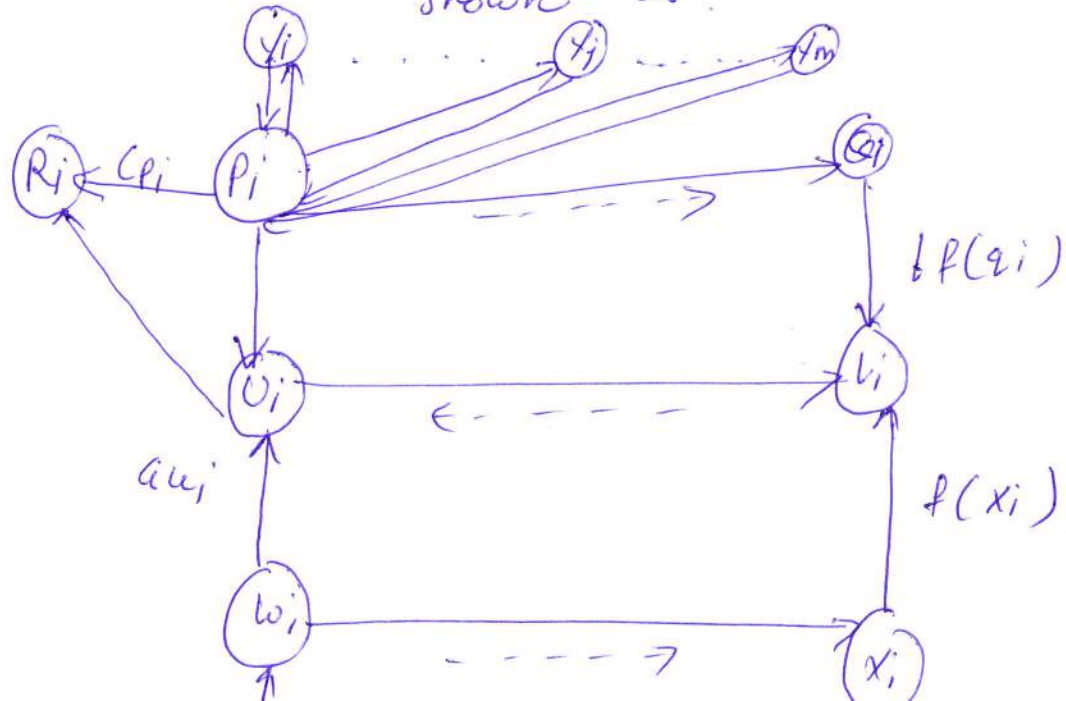
$$t_{ji}(new) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Step 14: Test for stopping condition

ART-2 :- ART accept Continuous valued vectors. The difference b/w ART2 & ART1 deflects the modification need to accomodate patterns with continuous valued components. ART 2 has a highly complex F_1 units. The F_2 unit of ART2 posses the combination of normalization and noise suppression, along with the comparison of weights needed for reset mechanism.

ART2 has 2 types of continous valued inputs. One is called noisy binary signal and the other, truly continuous. The first one can operate with the fast learning type of data. The second type of data is more suitable with the slow learning mode.

Architecture :- The architecture of ART2 N/W is shown as



From the architecture, it is seen that, the F_1 layer has six types of units (W, X, U, V, P and Q) B/w all units say, W and X, P and Q, V and U there exists supplemental unit, which receives signals from respective units, calculates its norm and sends to the other units. The receiving unit receives both inhibitory and excitatory signals from the sending units through supplemental units.

The X and Q units are connected to V units. The transformation occuring in the signal is indicated by symbols indicated in connection. The P unit path is connected to the cluster units by bottom up and top down weights. The symbol ----> indicates normalization. The units perform the operation of $F_1(a)$ layer as in ART1 and P units performs the operation of $F_1(b)$ layer interface portion as in ART1

Calculation of F_1 layer: Then calculations are necessary whenever "update F_1 activations" occur in the algorithm. The normalization and noise suppression is

winning unit of the F_2 layer based on the Competition. If no winning unit is chosen then $d=0$ for all units. The Calculation of w_i and x_i and q_i can be performed in parallel.

The Calculations involved are

$$u_i = \frac{v_i}{e + \|v\|}$$

$$w_i = s_i + a u_i, \quad p_i = u_i + d t_{ji}$$

$$x_i = \frac{w_i}{e + \|w\|}, \quad q_i = \frac{p_i}{e + \|p\|}$$

$$v_i = f(x_i) + b f(q_i)$$

The activation f^n is

$$f(x) = \begin{cases} x & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases}$$

The Training Algorithm:-

Step 1: Initialize parameter $a, b, \theta, c, d, e, \alpha, \rho$

Step 2: Perform step 3-13 up to specified no. of epochs of training.

Step 3: For each input vector 's', do steps 4-12

Step 4: Update F_1 unit activations

$u_i = 0, \quad x_i = \frac{s_i}{\|s\|}$

update F_1 unit activation again.

$$x_i = \frac{v_i}{e + \|v\|}, \quad w_i = s_i + a u_i$$

$$p_i = u_i, \quad x_i = \frac{w_i}{e + \|w\|}, \quad q_i = \frac{p_i}{e + \|p\|}$$

$$v = f(x_i) + b f(q_i)$$

<u>Step 5</u>: Compute signals to F_2 units

$$y_j = \sum_i b_{ij} p_i$$

<u>Step 6</u>: While reset is true, perform 7–8

<u>Step 7</u>: For F_2 unit choose y_j with largest signal.

<u>Step 8</u>: Check for reset

$$u_i = \frac{v_i}{e + \|v\|}, \quad p_i = u_i + d t_{ji}$$

$$r_i = \frac{u_i + c p_i}{e + \|u\| + c\|p\|}$$

if $\|r\| < \rho - e$, then

$$y_j = -1 \text{ (inhibit } j\text{)}$$

Since reset is true, go to step 6.

if $\|r\| > \rho - e$, then

$$w_i = s_i + a u_i$$

$$v = f(x_i) + b f(q_i)$$

Step 9: Perform steps 10-12 up to specified no. of learning iterations.

Step 10: Update weights for winning unit J

$$t_{ji}(new) = \alpha d u_i + \{1 + \alpha d(d-1)\} t_{ji}(old)$$

$$b_{ij}(new) = \alpha d u_i + \{1 + \alpha d(td-1)\} b_{ij}(old)$$

Step 11:- Update F_1 activations.

Step 12 Test for stopping Condition for weight updates.

Step 13: Test stopping Condition for no. of epochs.

In slow learning, no. of learning iterations is 1. In fast learning, for the first pattern learned by cluster u, will be parallel to t throughout the training cycle & equilibrium weights are

$$t_{ji} = \frac{1}{1-d} u_i, \quad b_{ij} = \frac{1}{1-d} u_i$$

The stopping Condition may be no. of epochs or the weight changes below a specified tolerance

The parameters used in the algorithm Can has typical values as follows.

Parameters	Description	Typical value
n	no. of I/P units	user defined
m	no of cluster units	User defined
a	Fixed weight in F_1 layer	10
b	Fixed weight in F_1 layer	10
c	Fixed weight used in testing for reset	0.1
d	Activation of winning F_2 unit	0.9
e	Parameter to prevent division by zero when norm of vector is zero	Negligibal value
θ	Noise suppression parameter	$1/\sqrt{n}$
α	Learning rate	Small value
ρ	vigilance parameter	Small value
$t_{ji}(0)$	initial top down weights	0
$b_{ij}(0)$	initial bottom up weights	$b_{ij}(0) \leq \frac{1}{(1-d)\sqrt{n}}$

The c and d parameters should be chosen to satisfy the eqn $\frac{cd}{1-d} \leq 1$

ART Characteristics :- ART has a no. of Characteristics as :-

1). Top down weight initialization :- The top down weights must be initialized to ones, if they were initialize to ~~one~~ 0 Then input vector c would also be zero.

→ Training may be viewed as a process of pruning Components of stored vectors that do not match input vectors. This process is irreversible ie once a top-down weight has been set to zero, The training algorithm Can never restore it to a one.

→ This characteristics has importance implications for the learning process. Suppose that a group of closely related vector should be classified into same Category, indicated by firing the same recognition layer neuron.

→ If they were presented sequentially to n/w, first will be assigned a recognition layer neuron, its weights will be trained to match the n/w input vector. Training with rest of vectors will

→ Thus, the stored vector come to represent the logical intersection of all training vectors. A new vector consisting only their essential features will be assigned to this category.

2) <u>Bottom up weights adjustments</u>: The summation of the bottom up weight formula represent the no. of ones in the o/p of comparison layer. It also represent the 'size' of vector

→ Thus, large C vectors produce smaller weight values than do small C vectors. This property makes possible to separate vector where one is subset of other

→ Suppose n/w has been trained on two input vectors with a recognition layer neuron assigned to each. Here X_1 is subset of X_2. Without this property, weights would be trained to the same values of each pattern.

→ If X_1 is applied once more, both recognition layer neurons receives the same activations hence wrong neurons 2 will win the competition.

3) Bottom up weight initialization: For the correct f^n of ART, bottom up weights must be initialized to small values.

→ If they are too high, I/P vectors that already have been learned will activate an unconditioned recognition layer neuron rather than one previously trained.

→ Setting this weights to low value ensures that uncommitted neuron will not overpower a trained recognition layer neuron.

→ For eg: For $L=2$, $m=5$, $b_{ij} < \frac{1}{3}$ & we set

$$b_{ij} = \frac{1}{6}$$

with these weights applying a vector for which n/w has been trained will cause directly trained recognition layer neuron to win over uncommitted neuron.

Image Compression using ART:

Like CPN's ARTs also used for data compression.

→ The unique ability of ART to control the tradeoff b/w compression ration &

→ The large amount of time & storage required to transmit pictorial data brings the need of image data compression.

→ In this work, ART is studied and its performance compared with CPNs. ART has CPN advantages such as fast learning & self organizing, in addition ART has vigilance parameter to directly control the tradeoff b/w compression ratio & distortion ratio.

→ Another factor is that we are concerned only with the quality of reconstructed pictorial data and length of time needed to transmit data from source to destination & not the relation among nodes.

→ Also, modification are made in the ART learning algorithm to achive error free compression.

→ In the process of compression, a picture is split into small frames. Those sub-images then fed to ART N/W for self-orgnization & O/PS represent category indices for those sub-images.

→ After learning those indices and their

according to their indices and prototypes.

→ The benefit received from this opn is measured by Comparison ratio (CO) which is computed as:-

$$CO = \frac{NF}{C \cdot N + F \log_2 C}$$

N – no. of dimension of sub images.
F – Total no. of frames.
C – Total no. of Categories formed during learning.