SOFT COMPUTING

Module 3: UNSUPERVISED NETWORKS

SYLLABUS

- Self-organizing maps
- LVQ network
- ART network

UNSUPERVISED LEARNING NETWORKS

- Unsupervised learning is the **second major learning paradigm**
- The system(environment) does not provide any feedback to indicate the desired output
- The **network itself has to discover any relationships of interest** (**features, patterns, correlations, classifications**) in the input data
- Translate the discovered relationships into outputs
- These type of ANNs are also called **self-organizing networks**

UNSUPERVISED LEARNING NETWORKS CONTD…

- **It can determine how similar a new input pattern is to typical patterns already seen**
- **The network gradually learns what similarity is**
- **It can set up several axes along which to measure similarity to previous patterns**
- The axes can be any one of the following:
- **Principal Component Analysis**
- **Clustering**
- **Adaptive Vector Quantization**
- **Feature Mapping**

UNSUPERVISED LEARNING NETWORKS CONTD…

- The net has **added structure** by means of which it is **forced to make a decision**
- **If there are several neurons for firing then only one of them is selected to fire (respond)**
- The process for achieving this is called a **competition**

- **An Example:**
- Suppose we have a **set of students**
- Let us **classify them on the basis of their performance**
- The **scores** will be calculated

UNSUPERVISED LEARNING NETWORKS CONTD…

- The one whose score is higher than all others will be the **winner**
- The **same principle is followed for pattern classification** in neural networks
- Here, **there may be a tie**
- **Some principle is followed** even when there is a tie
- These nets are called **competitive nets**
- The extreme form of these nets are called **winner-take-all**
- In such a case, **only one neuron in the competing group will possess a non-zero output signal** at the end of the competition

UNSUPERVISED LEARNING NETWORKS CONTD…

- **Several ANNs exist under this category**

  1.Maxnet

  2.Mexican hat

  3.Hamming net

  4.Kohonen self-organizing feature map

  5.Counter propagation net

  6.Learning vector quantization

  7.Adaptive Resonance Theory (ART)

UNSUPERVISED LEARNING NETWORKS CONTD…

- In case of these ANNs the net seeks to **find patterns or regularity** in the input data by **forming clusters**
- ARTs are called **clustering nets**
- **In such nets there are as many input units as an input vector possessing components**
- Since each output unit represents a cluster, **the number of output units** will limit **the number of clusters that can be formed**
- **The learning algorithm used in most of these nets is known as Kohonen learning**

KOHONEN LEARNING

- The **units update their weights** by forming a **new weight vector**, which is **a linear combination of the old weight vector and the new input vector**
- The **learning continues for the unit whose weight vector is closest to the input vector**
- The **weight updation formula** for output cluster unit 'j' is given by

$$w_j(new) = w_j(old) + \alpha[x - w_j(old)]$$

- x: Input vector
- w_j : weight vector for unit j
- α: learning rate, **its value decreases monotonically** as training continues

DECISION ON WINNERS

- There are **two methods** to determine the winner of the network during competition
- **Method 1:**
- The **square of the Euclidean distance** between the **input vector** and the **weight vector** is computed
- The **unit whose weight vector** is at the **smallest distance** from the **input vector** is chosen as the **winner**
- **Method 2:**
- The **dot product** of the **input vector** and the **weight vector** is computed
- This **dot product** is nothing but the **net inputs** calculated for the **corresponding cluster units**

DECISION ON WINNERS

- The weight updation is performed over it because the one with the **largest dot product corresponds to the smallest angle between the input and the weight vector**, if both are of unit length
- We know that for two vectors a and b, the dot product is given by the formula

$$a \cdot b = |a||b| \cos\theta$$

- If $|a| = |b| = 1$ then a . b = $\cos\theta$
- Also, as θ increases a . b decreases
- **Both the methods can be used if the vectors are of unit length**
- **The first method is normally preferred**

KOHONEN SELF-ORGANISING FEATURE MAPS

- Feature mapping is a process which **converts the patterns of arbitrary dimensionality** into a **response of one or two dimensional** array of neurons
- It **converts a wide pattern space into a typical feature space**
- **Feature map** is a network performing such a map
- It **maintains the neighbourhood relations of input patterns**
- It has to obtain a topology (**structure**) preserving map
- For such feature maps **it is required to find a self-organizing neural array** which consists of neurons **arranged in a one-dimensional array** or **two-dimensional array**

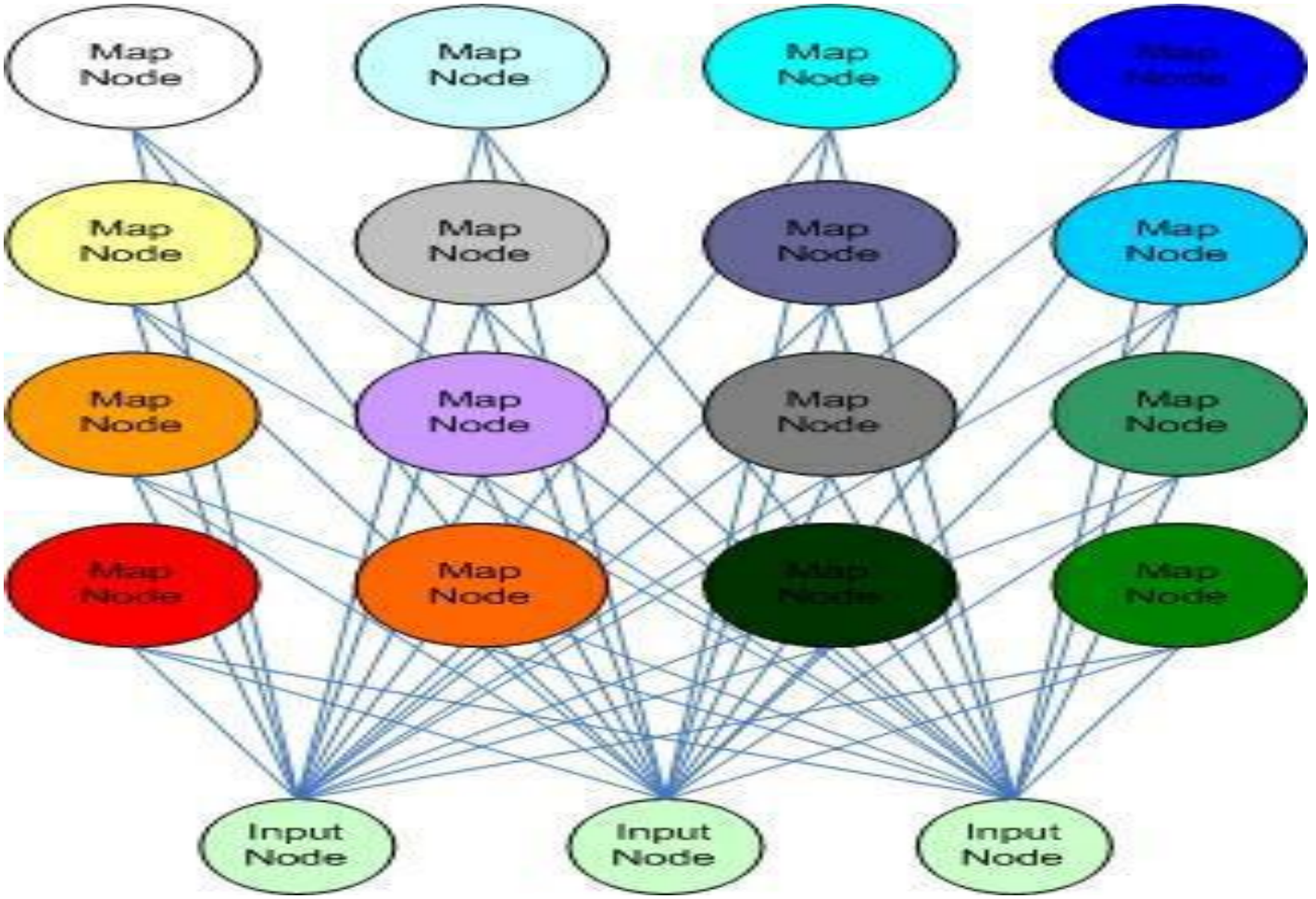
KOHONEN SELF-ORGANISING FEATURE MAPS CONTD…

- Developed in **1982**
- By **Tuevo Kohonen**, a professor emeritus of the **Academy of Finland**
- SOMs **learn on their own through unsupervised competitive learning**
- **Called "Maps"** because they **attempt to map their weights to conform to the given input data**
- The nodes in a SOM network attempt to become like the inputs presented to them
- **Retaining principle 'features' of the input data** is a fundamental principle of SOMs

KOHONEN SELF-ORGANISING FEATURE MAPS CONTD…

- **SOMs provide a way of representing multidimensional data in a much lower dimensional space** – typically one or two dimensions
- This aides in their visualization benefit, as **humans are more proficient at comprehending data in lower dimensions than higher dimensions**
- SOMs are a valuable tool in **dealing with complex or vast amounts of data**
- These are **extremely useful for the visualization and representation of complex or large quantities of data in a manner that is most easily understood by the human brain**

KOHONEN SELF-ORGANISING FEATURE MAPS CONTD…



KOHONEN SELF-ORGANISING FEATURE MAPS CONTD…

- Few **key things** to notice:
- **Each map node is connected to each input node**
- For this small 4x4 node network, that is 4x4x3=**48 connections**
- Map nodes are **not connected to each other**
- Nodes are organized in this manner, as a 2-D grid makes it easy to visualize the results
- **Each map node has a unique (i, j) coordinate**
- This makes it **easy to reference a node in the network**
- Also, it is **easy to calculate the distances between nodes**
- Because of the **connections only to the input nodes**, the map nodes are oblivious (**unaware**) as to what values their neighbours have

KSO FEATURE MAPS: TRAINING ALGORITHM

- The steps involved in the training algorithm are:
- **STEP 0:** Initialize the weights w_{ij} (**Random values are assumed**)
- These **can be chosen as** the same as the components of the input vector
- Set the **topological neighbourhood parameters** (radius of the neighbourhood etc.)
- Initialize the learning rate
- **STEP 1:** Perform Steps 2 – 8 when stopping condition is false
- **STEP 2:** Perform Steps 3 – 5 for each input vector x
- **STEP 3:** Compute the square of the Euclidean distance for j = 1,2,…m.

KSO FEATURE MAPS: TRAINING ALGORITHM

- We have $D(j) = \sum_{i=1}^{m} (x_i - w_{ij})^2$

- We can use the dot product method also
- **STEP 4:** Find the **winning unit** j, so that **D(j) is minimum**.
- **STEP 5:** For all units j within a specific neighbourhood of j and for all i, calculate the new weights as:

$$w_{ij}(new) = w_{ij}(\text{old}) + \alpha.[\text{x}_i - w_{ij}(\text{old})]$$

$$\Updownarrow$$

$$w_{ij}(new) = (1-\alpha)w_{ij}(\text{old}) + \alpha.x_i$$

KSO FEATURE MAPS: TRAINING ALGORITHM

- **STEP 6:** Update the learning rate α using the formula

$$\alpha(t+1) = (0.5).\alpha(t)$$

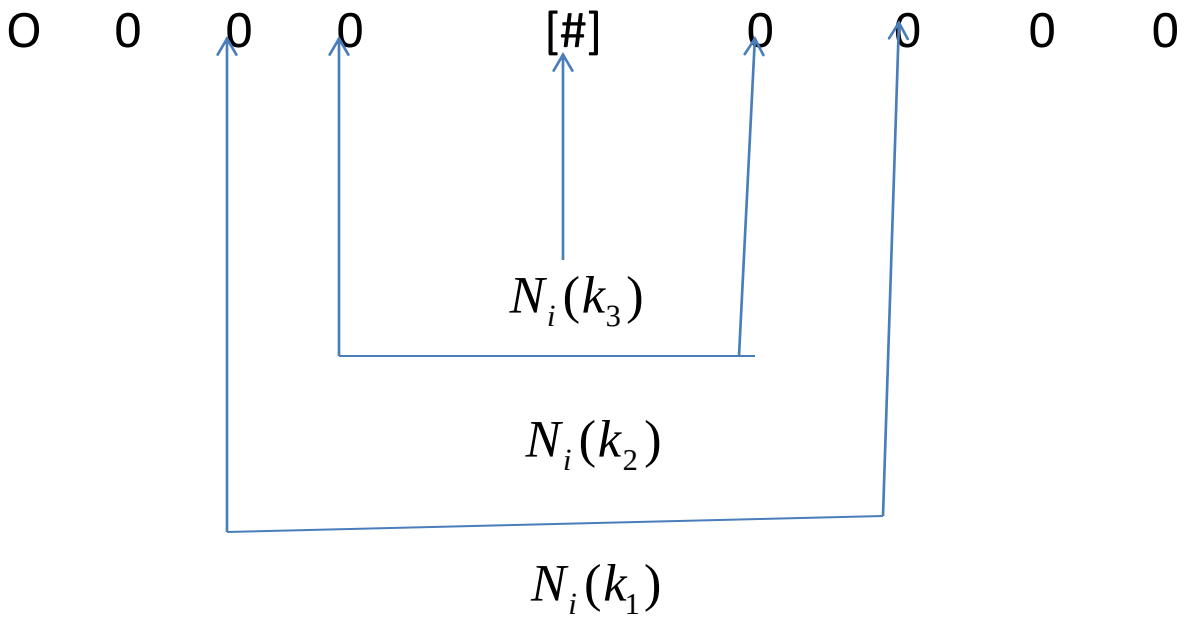
The value of alpha decreases as we proceed

- **STEP 7:** Reduce radius of topological neighbourhood at specified times
- **STEP 8:** Test for stopping condition of the network
-

KSO FEATURE MAPS: ARCHITECTURE

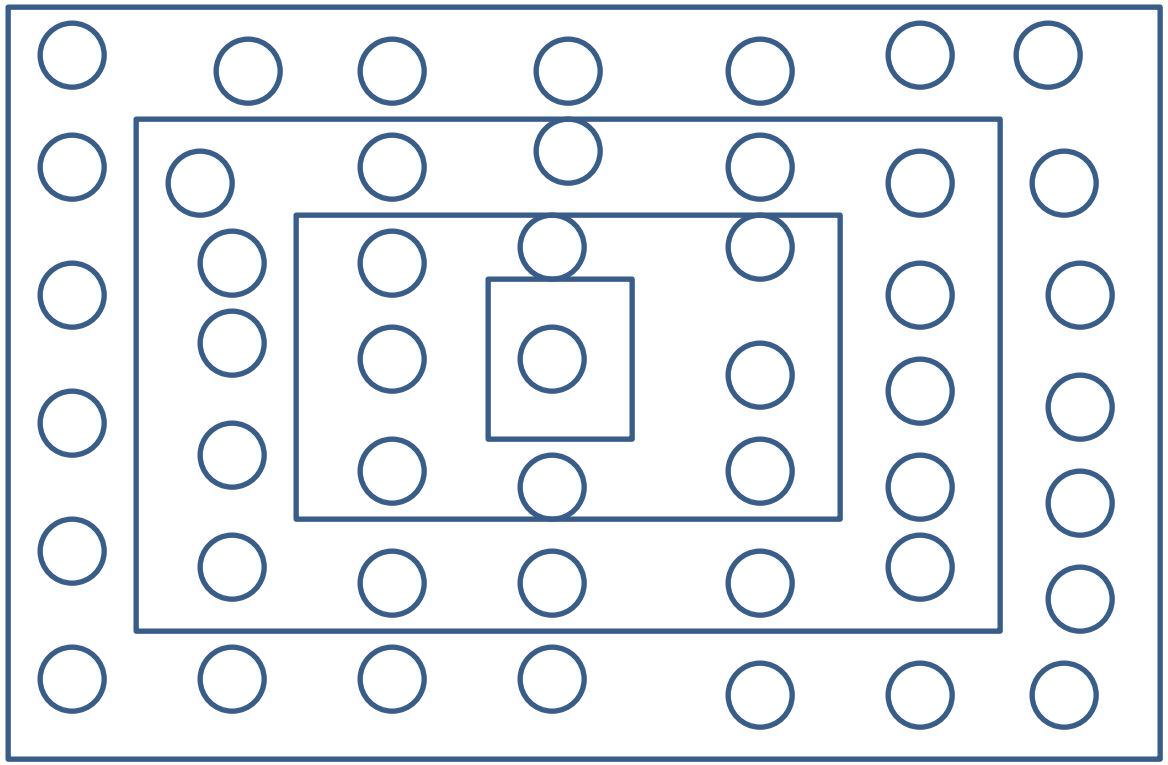
- For a linear array of cluster units, the neighbourhood units designated by ‘o’ of radii $N_i(k_1), N_i(k_2), N_i(k_3)$;

$$k_1 > k_2 > k_3; k_3 = 0, k_2 = 1, k_1 = 3$$



KSO FEATURE MAPS: ARCHITECTURE

- For a rectangular grid it looks like



KSO FEATURE MAPS: STEPS

- The self-organization process involves **four major components:**

- **Initialization:** All the connection weights are initialized with **small random values**

- **Competition:** For each input pattern, the **neurons compute their respective values of a discriminant function (D)** which provides the basis for competition. The **particular neuron with the smallest value of the discriminant function is declared the winner.**

KSO FEATURE MAPS: STEPS CONTD…

- **Cooperation:** The **winning neuron determines the spatial location of a topological neighbourhood of excited neurons**, thereby providing the basis for cooperation among neighbouring neurons.

- **Adaptation:** The **excited neurons decrease their individual values of the discriminant function** in relation to the input pattern through **suitable adjustment of the associated connection weights**, such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced

EXAMPLE

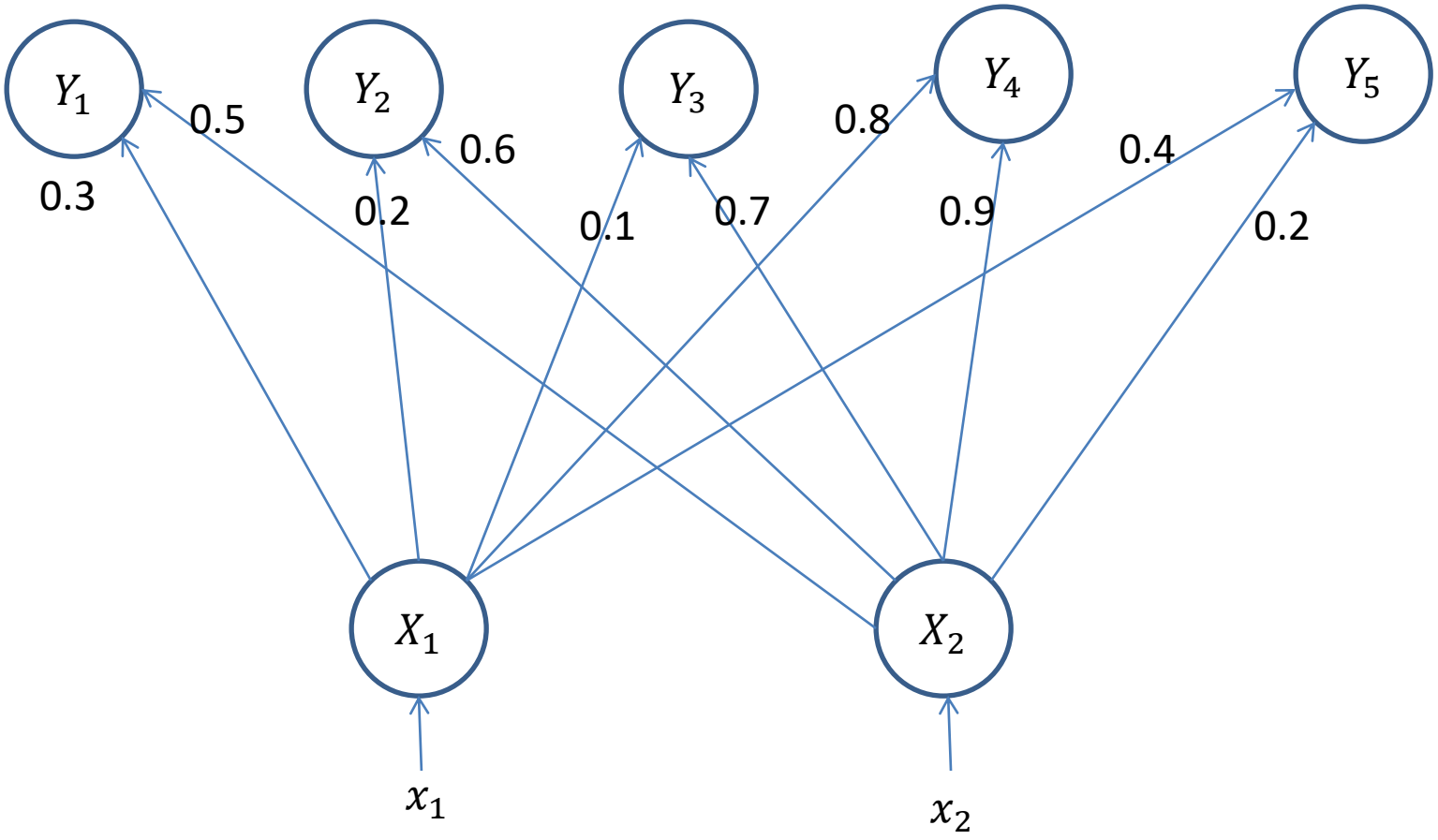
- For a **given Kohonen self-organising feature map** with weights shown in the next slide:

- (a) use the **square of the Euclidean distance** to find the cluster unit Y_j closest to the input vector (0.2, 0.4). Using a learning rate 0.2 find the new weights for unit Y_j.

- (b)For the input vector (0.6, 0.6) with learning rate 0.1, find the **winning cluster unit** and its **new weights**.

EXAMPLE CONTD…

a



EXAMPLE CONTD…

(a) For the input vector (0.2, 0.4) = (x_1, x_2) and learning rate $\alpha = 0.2$, the weight vector W is given by (**chosen arbitrarily**)

$$W = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

Next we find out the winner unit by using square of unit distance.

Here the formula is

$$D(j) = \sum_{i=1}^{2} \left(w_{ij} - x_i\right)^2 = \left(w_{1j} - x_1\right)^2 + \left(w_{2j} - x_2\right)^2$$

j= 1, 2, 3 , 4 ,5

EXAMPLE CONTD…

$$D(1) = (0.3 - 0.2)^2 + (0.5 - 0.4)^2 = 0.01 + 0.01 = 0.02$$

$$D(2) = (0.2 - 0.2)^2 + (0.6 - 0.4)^2 = 0.0 + 0.04 = 0.04$$

- $D(3) = (0.1 - 0.2)^2 + (0.7 - 0.4)^2 = 0.01 + 0.09 = 0.1$

$$D(4) = (0.8 - 0.2)^2 + (0.9 - 0.4)^2 = 0.36 + 0.25 = 0.61$$

$$D(5) = (0.4 - 0.2)^2 + (0.2 - 0.4)^2 = 0.04 + 0.04 = 0.08$$

- Since D(1)=0.02 is the minimum value, the winner unit is j =1.

- We now update the weights on the winner unit j =1. The weight updation formula used is

- $w_{iJ}(new) = w_{iJ}(old) + \alpha[x_i - w_{iJ}(old)]$

EXAMPLE CONTD…

- Since j =1, the formula becomes
- $w_{i1}(new) = w_{i1}(old) + \alpha[x_i - w_{i1}(old)]$
- Putting i = 1, 2 we get
- $w_{11}(n) = w_{11}(\mathrm{O}) + \alpha[x_1 - w_{11}(\mathrm{O})] = 0.3 + 0.2[0.2 - 0.3] = 0.28$
  $w_{21}(n) = w_{21}(\mathrm{O}) + \alpha[x_2 - w_{21}(\mathrm{O})] = 0.5 + 0.2[0.4 - 0.5] = 0.48$
- The updated matrix is now
- $W = \begin{bmatrix} 0.28 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.48 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$
- (b) For the input vector $(x_1, x_2) = (0.6,0.6)$ and $\alpha = 0.1$, the initialised weight matrix we take as same for part (a)

EXAMPLE CONTD…

- The square of the Euclidean distance formula is same as in part (a)
- So, taking j = 1,…5, we get

$$D(1) = (0.3 - 0.6)^2 + (0.5 - 0.6)^2 = 0.09 + 0.01 = 0.1$$
$$D(2) = (0.2 - 0.6)^2 + (0.6 - 0.6)^2 = 0.08 + 0.0 = 0.08$$

- $D(3) = (0.1 - 0.6)^2 + (0.7 - 0.6)^2 = 0.25 + 0.01 = 0.26$
  $D(4) = (0.8 - 0.6)^2 + (0.9 - 0.6)^2 = 0.04 + 0.09 = 0.13$
  $D(5) = (0.4 - 0.6)^2 + (0.2 - 0.6)^2 = 0.04 + 0.16 = 0.2$
- Since D(2)=0.08 is the minimum, the winner is unit j = 2

EXAMPLE CONTD…

- We update the weight for j = 2 with the same formula
- $w_{i2}(new) = w_{i2}(old) + \alpha[x_i - w_{i2}(old)]$
- Taking i = 1, 2
- $w_{12}(n) = w_{12}(O) + \alpha[x_1 - w_{12}(O)] = 0.2 + 0.1[0.6 - 0.2] = 0.24$
  $w_{22}(n) = w_{22}(O) + \alpha[x_2 - w_{22}(O)] = 0.6 + 0.1[0.6 - 0.6] = 0.6$

- The new weight matrix is given by
- $W = \begin{bmatrix} 0.3 & 0.24 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$

LEARNING VECTOR QUANTIZATION

-LVQ

LVQ

- It is a **process of classifying the patterns**, wherein each output unit represents a particular class
- For each class several units should be used
- **Reference vector**/ **code book vector**:
- It is the output unit weight vector for the class which the unit represents
- This is a **special case of competitive net**
- It follows supervised learning methodology
- During training the output units are found to be positioned to approximate the decision surfaces of the existing Bayesian classifier

LVQ CONTD…

- At the end of the training process
- LVQ net is found to classify an input vector by assigning it to the same class as that of the output unit, which has its weight vector very close to the input vector
- It is a classifier paradigm that adjusts the boundaries between categories to minimize existing misclassification
- It is used for:
- Optical character recognition
- Converting speech into phonemes

LVQ ARCHITECTURE

- It has n input and m output units The weights from the ith input unit to the jth output unit is given by wij
- Each output unit is associated with a class/cluster/category
- x: Training vector (x1, x2, …xn)
- T: category or class of the training vector x
- wj = weight vector for the jth output unit (w1j, …, wij, …wnj)
- cj = cluster or class or category associated with jth output unit
- The Euclidean distance of jth output unit is
- $D(j) = \sum_{i=1}^{n} \left(x_i - w_{ij}\right)^2$

TRAINING ALGORITHM

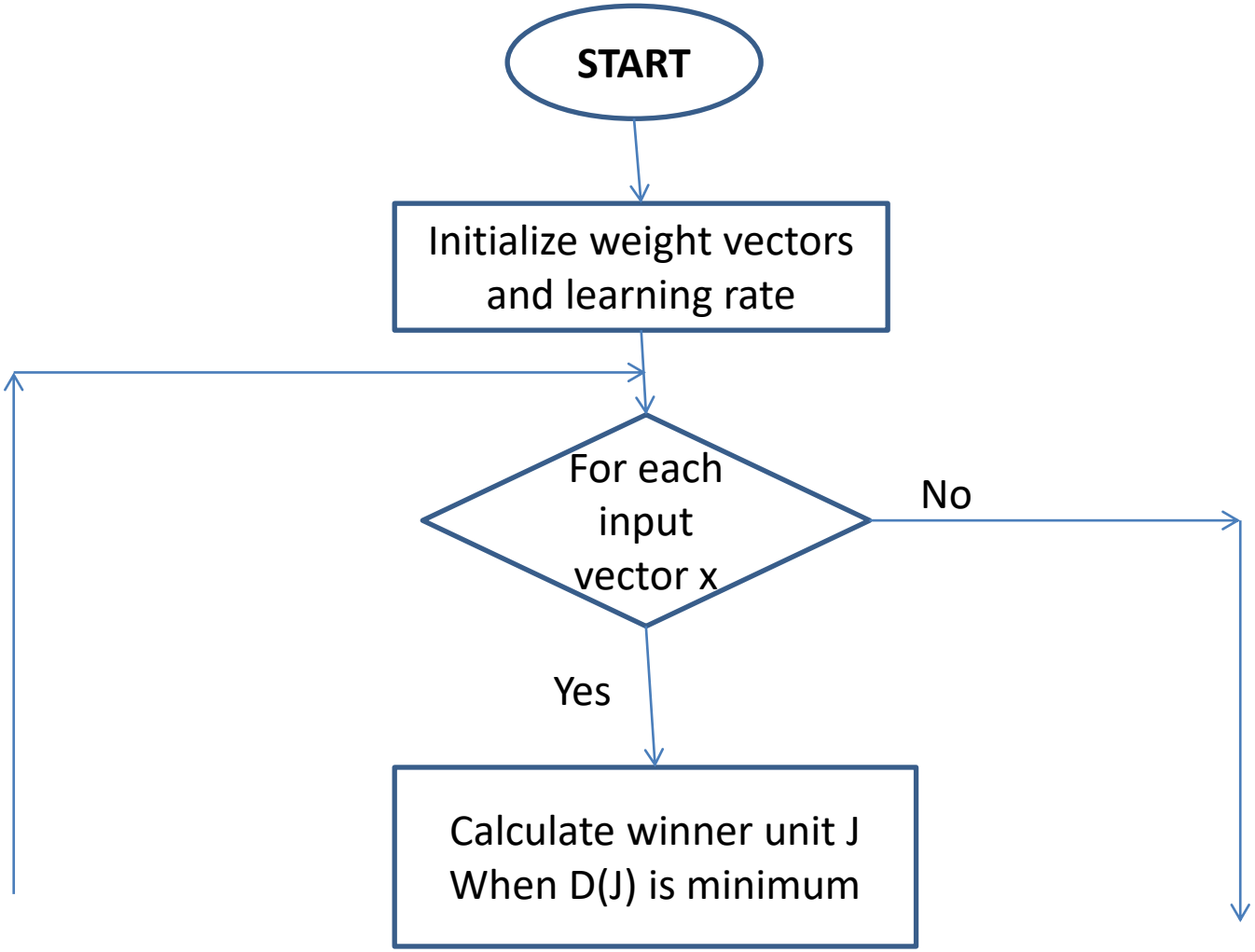
- **STEP 0:** Initiate the reference vectors. That is,
- i. From the given set of training vectors, take the first "m" (number of clusters) training vectors and use them as weight vectors, the remaining vectors can be used for training
- ii. Assign the initial weights and classifications randomly
- ii. K-means clustering method
- Set initial learning rate α
- **STEP 1:** Perform steps 2-6 if the stopping condition is false
- **STEP 2:** Perform steps 3-4 for each training input vector x
- **STEP 3:** Calculate the Euclidean distance, for i = 1 to n, j =1 to m $D(j) = \sum_{i=1}^{n} \left(x_i - w_{ij}\right)^2$

TRAINING ALGORITHM CONTD..

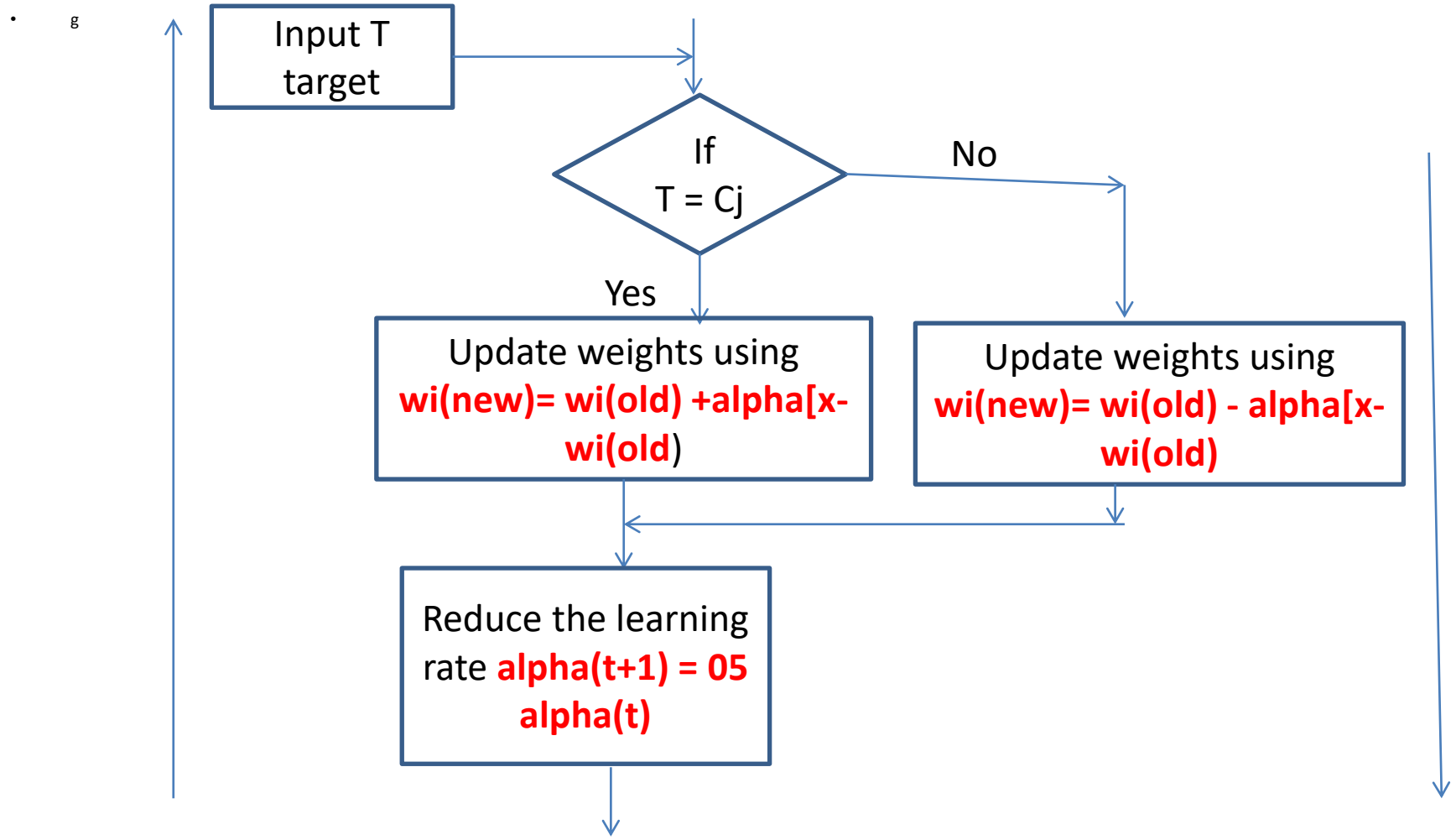
- Find the winning unit index J, when D(J) is minimum
- **STEP 4:** Update the weights on the winning unit wj using the following conditions
- If T = cj, then $w_j(new) = w_j(old) + \alpha[x - w_j(old)]$
- If $T \neq c_j$ then $w_j(new) = w_j(old) - \alpha[x - w_j(old)]$
- **STEP 5:** reduce the learning rate α
- **STEP 6:** Test for the stopping condition of the training process. (the stopping conditions may be fixed number of epochs or if learning rate has reduced to a negligible value)

FLOW CHART

- G

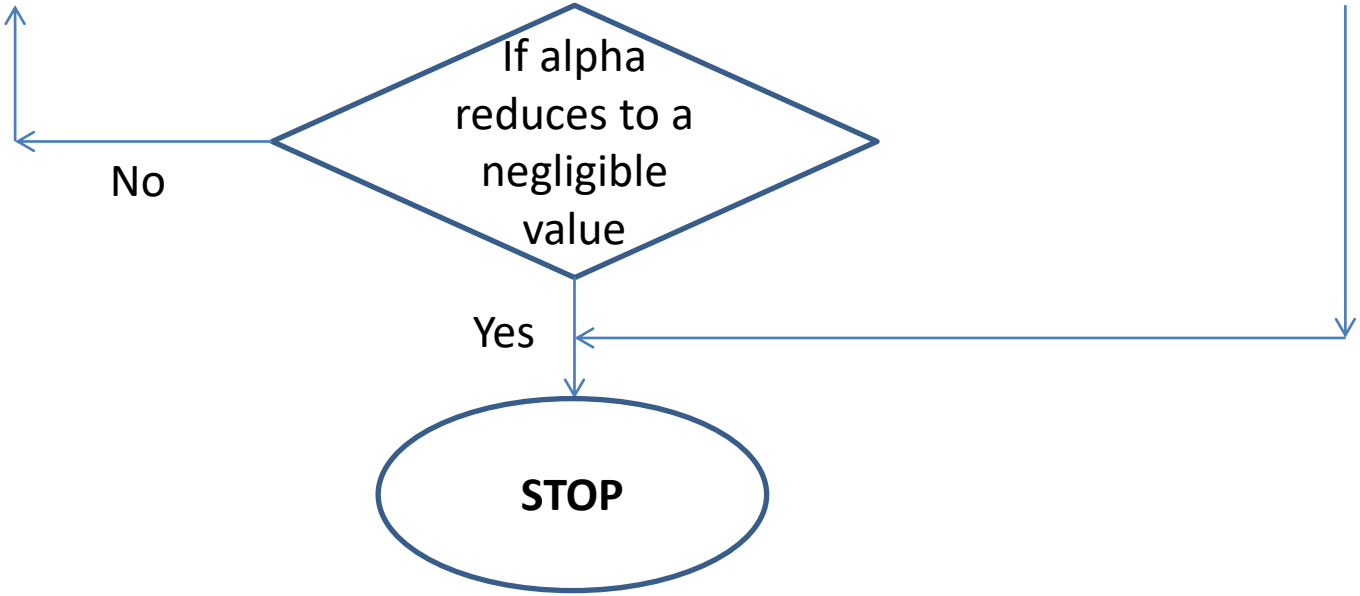

START
Initialize weight vectors and learning rate
For each input vector x
No
Yes
Calculate winner unit J When D(J) is minimum

FLOW CHART


Input T
target
If
T = Cj
No
Yes
Update weights using
wi(new)= wi(old) +alpha[x-
wi(old)
Update weights using
wi(new)= wi(old) - alpha[x-
wi(old)
Reduce the learning
rate alpha(t+1) = 05
alpha(t)

FLOW CHART CONTD…

- g



AN EXAMPLE

- Consider an LVQ net with 2 input units x1, x2 and 4 output classes c1, c2, c3 and c4
- There exists 16 classification units, with weight vectors as indicated in the table below

x2/x1	0.0	0.2	0.4	0.6	0.8	1
0.0						
0.2		**c1**	**c2**	**c3**	**c4**	
0.4		**c3**	**c4**	**c1**	**c2**	
0.6		**c1**	**c2**	**c3**	**c4**	
0.8		**c3**	**c4**	**c1**	**c2**	
1						

EXAMPLE CONTD…

- Use square of Euclidean distance to measure the changes occurring
- (a) Given an input vector of (0.25, 0.25) representing class 1 and using a learning rate of 0.25, show which classification unit moves where?
- (b) Given the input vector of (0.4, 0.35) representing class 1, using initial weight vector and learning rate of 0.25, note what happens?

SOLUTION

- Let the input vectors be u1 and u2. Output classes be c1, c2, c3 and c4
- The initial weights for the different classes are as follows:
- Class 1:
- $w_1 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.6 \\ 0.2 & 0.6 & 0.8 & 0.4 \end{bmatrix}$ c_1 t =1
- Class 2:
- $w_2 = \begin{bmatrix} 0.4 & 0.4 & 0.8 & 0.8 \\ 0.2 & 0.6 & 0.8 & 0.4 \end{bmatrix}$ c_2 t = 2
- Class 3:
- $w_3 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.6 \\ 0.4 & 0.8 & 0.6 & 0.2 \end{bmatrix}$ c_3 t = 3

SOLUTION CONTD…

- Class 4:
- $w_4 = \begin{bmatrix} 0.4 & 0.4 & 0.8 & 0.8 \\ 0.4 & 0.8 & 0.6 & 0.2 \end{bmatrix}$ c_4 t = 4

- Part (a):
- For the given input vector (u1, u2) = (0.25, 0.25) with $\alpha = 0.25$ and t = 1, we compute the square of the Euclidean distance as follows:
- D(j) = $\left(w_{1j} - x_1\right)^2 + \left(w_{2j} - x_2\right)^2$

SOLUTION CONTD…

- For j = 1 to 4
- D(1) = 0.005, D(2) = 0.125, D(3) = 0.145 and D(4) = 0.425

- As D(1) is minimum, the winner index J = 1. Also t =1
- So, we use the formula

$$w_j(new) = w_j(old) + \alpha[x - w_j(old)]$$

The updated weights on the winner unit are

$$w_{11}(new) = w_{11}(old) + \alpha[x_1 - w_{11}(old)]$$
$$= 0.2 + 0.25(0.25 - 0.2) = 0.2125$$
$$w_{21}(new) = w_{21}(old) + \alpha[x_2 - w_{21}(old)]$$
$$= 0.2 + 0.25(0.25 - 0.2) = 0.2125$$

SOLUTION CONTD…

- So, the new weight vector is
- $W_1 = \begin{bmatrix} 0.2125 & 0.2 & 0.6 & 0.6 \\ 0.2125 & 0.6 & 0.8 & 0.4 \end{bmatrix}$

(b) For the given input vector $(w_1, w_2) = (0.4, 0.35)$, $\alpha = 0.25$ and t = 1 we calculate the Euclidean distance using

$$D(j) = \sum_{i=1}^{2} \left(w_{ij} - x_i\right)^2 = \left(w_{1j} - x_1\right)^2 + \left(w_{2j} - x_2\right)^2$$

Then we have D(1) = 0.0625, D(2)= 0.1025, D(3) = 0.2425 and D(4) = 0.0425

So, D(4) is minimum and hence the winner unit index is J = 4

SOLUTION CONTD…

- The fourth unit is the winner unit that is closest to the input vector
- Since $t \neq J$, the weight updation formula to be used is:
- $w_J(new) = w_J(old) - \alpha[x - w_J(old)]$
- Updating the weights on the winner unit, we obtain
- $$w_{14}(new) = w_{14}(old) - \alpha[x_1 - w_{14}(old)]$$
$$= 0.6 - 0.25(0.4 - 0.6) = 0.65$$
$$w_{24}(new) = w_{24}(old) - \alpha[x_2 - w_{24}(old)]$$
$$= 0.4 - 0.25(0.35 - 0.4) = 0.4125$$

SOLUTION CONTD…

- Therefore the new weight vector is
- $W_1 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.65 \\ 0.2 & 0.6 & 0.8 & 0.4125 \end{bmatrix}$

ART NETWORK

ART NETWORK

- It is Adaptive Resonance Theory (ART) Network
- Designed by Carpenter and Grossberg in 1987
- It is designed for both binary inputs and analog valued inputs
- The input patterns can be presented in any order
- This is unsupervised learning model based on competition
- It finds the categories automatically and learns new categories if needed
- This was proposed to solve the problem of instability occurring in feed forward networks
- There are two versions of it: ART1 and ART2

ART NETWORK

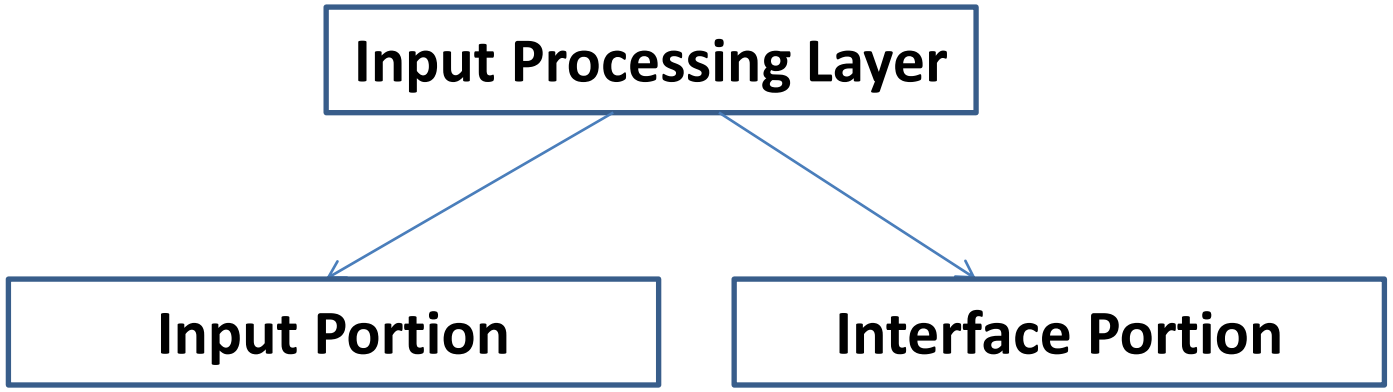
- **ART1** was developed for **clustering binary vectors**
- **ART2** was developed to accept **continuous valued vectors**
- For each pattern presented to the network, an appropriate cluster unit is chosen and the weights of the cluster unit are adjusted to let the cluster unit to learn the pattern
- The network controls the degree of similarity of the patterns placed on the same cluster units
- During training:
- Each training pattern may be presented several times
- The input patterns should not be presented on the same cluster unit when it is presented each time

THE ART NETWORK

- The stability may be achieved by reducing the learning rates
- The ability of the network to respond to a new pattern equally at any stage of learning is called a **plasticity**
- The ART networks are designed to possess the properties of stability and plasticity
- But, **it is difficult to handle both stability and plasticity**
- The ART networks are designed particularly to **resolve the stability-plasticity dilemma**
- This means they are **stable to preserve significant past learning** but nevertheless **remain adaptable to incorporate new information whenever it appears**

FUNDAMENTAL ARCHITECTURE OF ART

- **Three groups** of neurons are used
- **Input processing layer** (F1 layer)
- **Clustering units** (F2 layer)
- **Control mechanism** (It controls degree of similarity of patterns placed on the same cluster)



FUNDAMENTAL ARCHITECTURE CONTD…

- The input portion may perform some processing based upon the inputs it receives
- This is mostly performed in case of ART2 compared to ART1
- The interface portion of the F1 layer combines the input portion from F1 and F2 layers for comparing the similarity of the input signal with the weight vector for the cluster unit that has been selected as a unit for learning F1 layer input portion may be denoted as F1(a) and interface as F1(b)
- There exists two sets of weighted interconnections for controlling the degree of similarity between the units in the interface portion and the cluster layer

ARCHITECTURE CONTD…

- The bottom-up weights are used for the connection from F1(b) layer to F2 layer and are represented by b_{ij} (ith F1 unit to the jth F2 unit)
- The top-down weights are used for the connections from F2 layer to F1(b) layer and are represented by t_{ji} (jth F2 unit to ith F1 unit)
- The competitive layer in this case is the cluster layer and the cluster unit with largest net input is the victim to learn input pattern
- The activations of all other F2 units are made zero

ARCHITECTURE CONTD…

- The interface units combine the data from input and cluster layer units
- On the basis of similarity between the top-down weight vector and input vector, the cluster unit may be allowed to learn the input pattern
- The decision is done by reset mechanism unit on the basis of the signals it receives from interface portion and input portion of the F1 layer
- When cluster unit is not allowed to learn, it is inhibited an a new cluster unit is selected as the victim

FUNDAMENTAL OPERATING PRINCIPLE

- Presentation of one input pattern forms a **learning trial**
- The activations of all the units in the net are set to zero before an input pattern is presented
- All the units n the F2 layer are inactive
- On presentation of a pattern, the input signals are sent continuously until the learning trial is completed
- There exists a user-defined parameter called vigilance parameter
- This parameter controls the degree of similarity of the patterns assigned to the patterns assigned to the same cluster unit

FUNDAMENTAL OPERATING PRINCIPLE

- The function of the reset mechanism is to control the state of each node on F2 layer
- The units in F2 layer can be in any one of the three states at any instant of time

1. Active: Unit is ON. The activation in this case is equal to 1.

For ART1, d = 1 and for ART2, $0 < d < 1$

2. Inactive: Unit is OFF. The activation here is zero and the unit may be available to participate in competition.

3. Inhibited: Unit is OFF. The activation here is also zero but the unit here is prevented from participating in any further competition during the presentation of current input vector

FUNDAMENTAL OPERATING PRINCIPLE

- ART net can work in **two ways**:
- **Fast Learning**
- **Slow Learning**
- **FAST LEARNING:**
- Weight updation takes place rapidly relative to the length of time a pattern is being presented on any particular learning trial
- The weights reach equilibrium in each trial
- **SLOW LEARNING:**
- The weight changes occur slowly
- The weights do not reach equilibrium in each trial

FUNDAMENTAL OPERATING PRINCIPLE

- The patterns are binary in ART1
- Weights are associated with each cluster unit stabilize in the fast learning mode
- In ART2 network, the weights produced by fast learning continue to change each time a pattern is presented
- Net is found to stabilize only afer few presentations of each training pattern
- It is not easy to find equilibrium weights immediately for ART2
- In slow learning is not adapted in ART1
- IN ART2 the weights produced by slow learning are far better than those produced by fast learning for particular type of data

FUNDAMENTAL ALGORITHM

- This algorithm discovers clusters of a set of pattern vectors
- **STEP 0:** Initialize the necessary parameters
- **STEP 1:** Perform Steps 2-9 when stopping condition is false
- **STEP 2:** Perform Steps 3-8 for each input vector
- **STEP 3:** F1 layer processing is done
- **STEP 4:** Perform Steps 5-7 when reset condition is true
- **STEP 5:** Find the victim unit to learn the current input pattern. The victim unit is going to be the F2 unit with the largest input
- **STEP 6:** F1(b) units combine their inputs from F1(a) and F2
- **STEP 7:** Test for reset condition

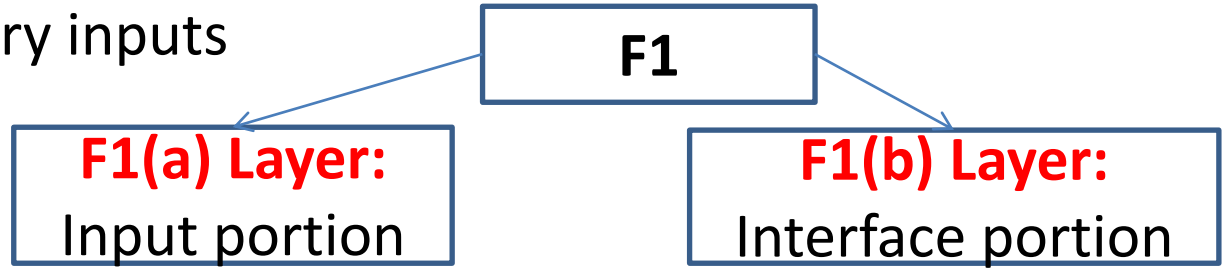
FUNDAMENTAL ALGORITHM

- If reset is true then the current victim unit is rejected (inhibited)
- Go to step 4 if reset is false and the current victim unit is accepted for learning
- Go to next step (Step 8)
- **STEP 8:** Weight updation is performed
- **STEP 9:** Test for stopping condition
- **Note:** The ART network does not require all training patterns to be presented in the same order or even if all patterns are presented in the same order , we refer to this as an epoch

ART1

ADAPTIVE RESONANCE THEORY (ART-1)

- Designed for binary inputs
- Input unit: F1
- Output unit: F2



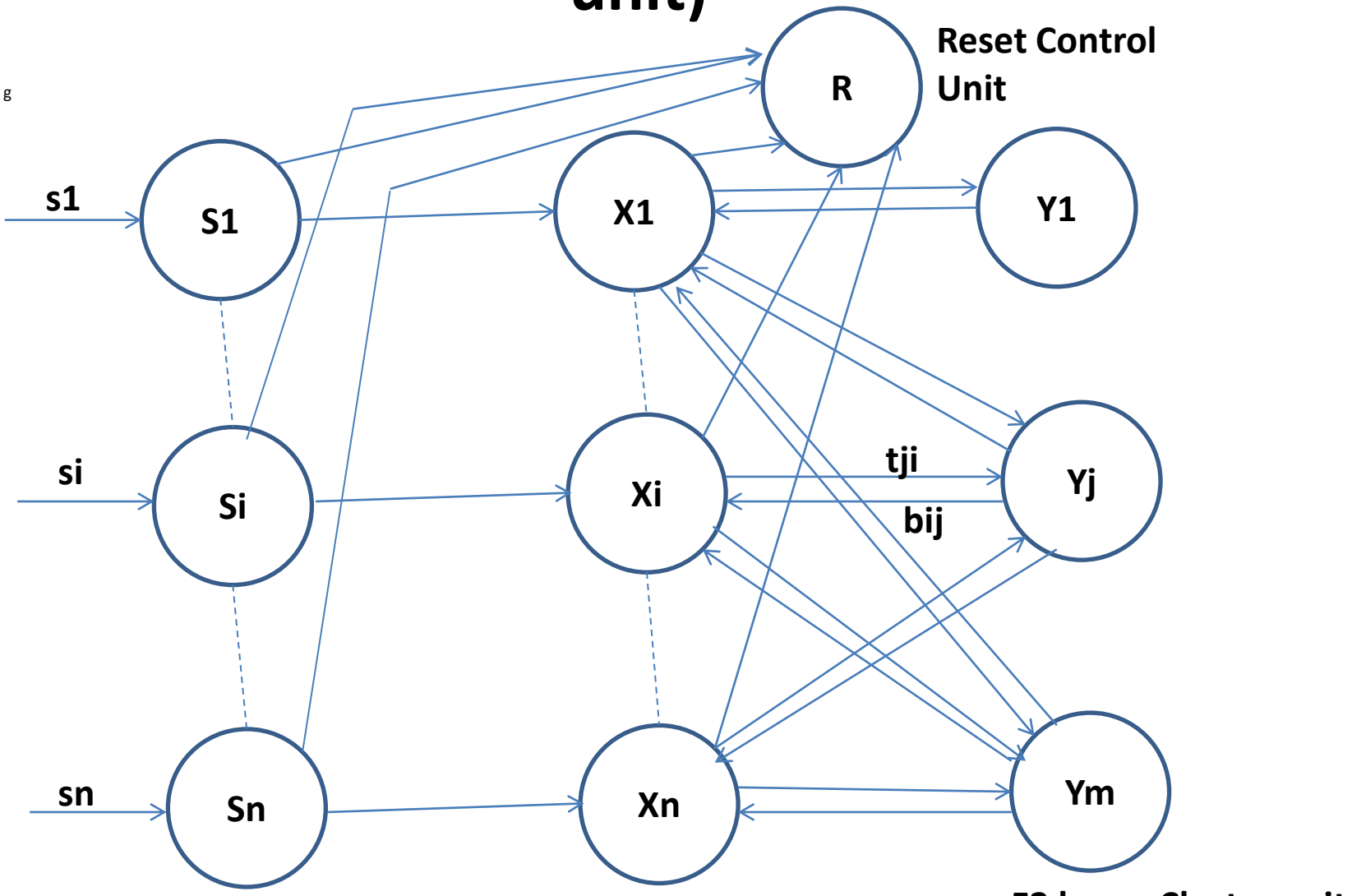
- Architecture of ART1 network: It is made up of two units
- 1. Computational unit
- 2. Supplemental unit
- **Computational Unit:**

❖ Input units (F1 unit) (input portion and interface portion)

❖ Cluster units (F2 unit) (output unit)

❖ Reset control unit (controls degree of similarity of patterns placed in the same cluster)

BASIC ARCHITECTURE OF ART 1 (Computational unit)

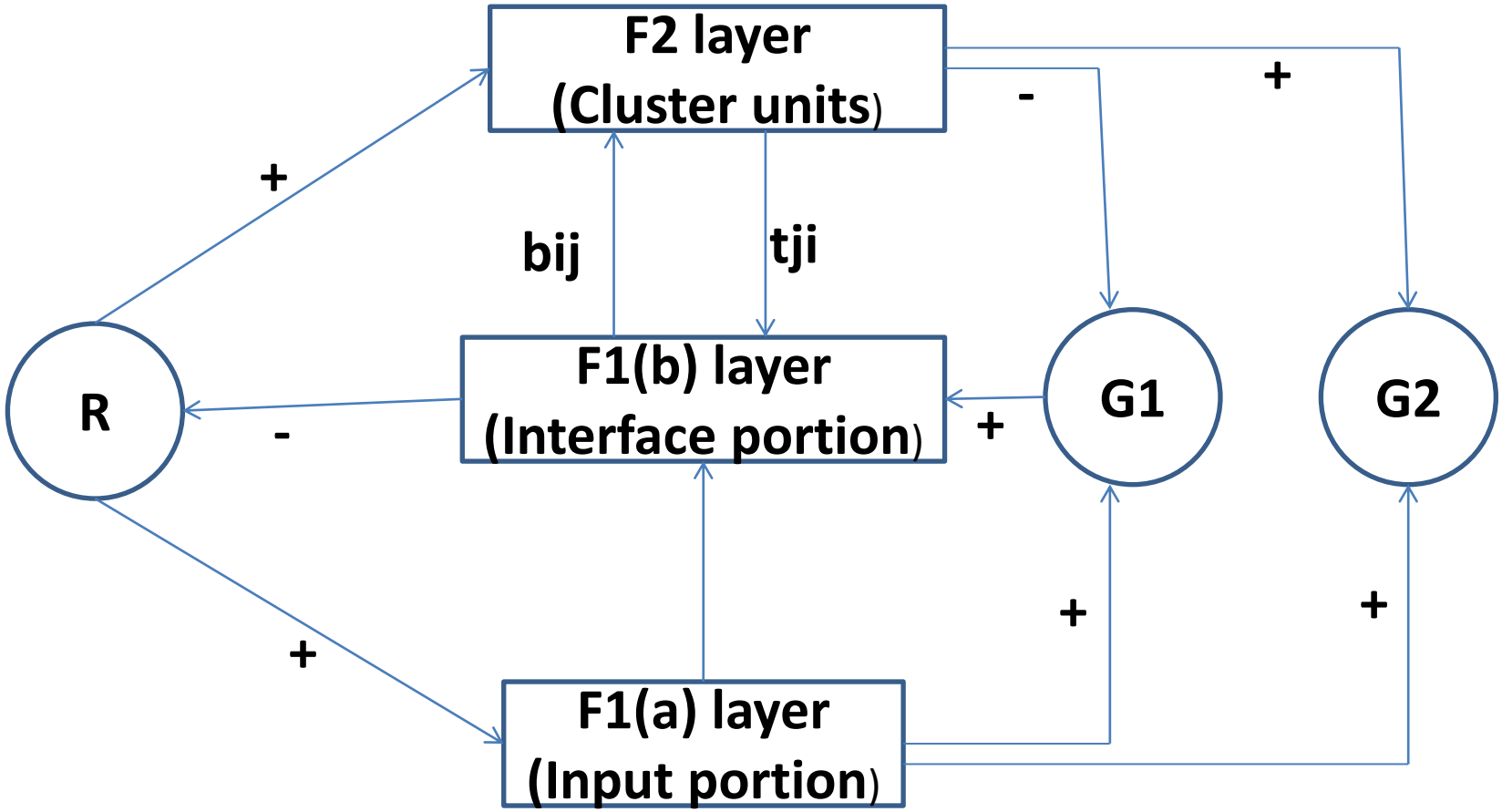


ARCHITECTURE EXPLAINED

- Each unit in the input portion of the F1 layer is connected to the respective unit n the interface portion of F1 layer
- Reset control unit has connections from the F1(a) and F1(b) units
- Each unit in F1(b) layer is connected through two weighted interconnection paths to each of the F2 unit
- The weight of the connection from Xi unit of F1(b) layer to Yj unit of F2 layer is bij
- The weight of the connection from Yj unit of F2 layer to the Xi unit of the F1(b) layer tji

ARCHITECTURE (SUPPLEMENTAL UNIT)

- G



SUPPLEMENTAL UNIT

- The supplemental unit provides the efficient neural control of the neural process
- In ART1 can be practically implemented by analog circuits governing differential equations
- That is, the bottom-up and top-down approach are controlled by differential equations
- It works throughout autonomously
- It does not require any external control signals and can run stably with infinite patterns of input data
- The computational units respond differently at different stages of the process and these are not supported by any of the biological neuron to decide what to then

SUPPLEMENTAL UNIT CONTD…

- G1 and G2 are called as gain control units
- There are three control units G1, G2 and reset control unit F
- These three units receive signals from and send signals to all of the units in the input layer and cluster layer
- The excitatory weighted signals are denoted by “+” and inhibitory signals are indicated by “-”
- Whenever any unit in a designated layer is “on”, a signal is sent
- F1(b) unit and F2 unit receive signal from three sources
- F1(b) unit can receive signal from either F1(a) unit or F2 units or G1 unit

SUPPLEMENTAL UNIT CONTD…

- F2 unit receives signal from F1(b) unit or reset control unit R or gain control unit G2
- An F1(b) unit or F2 unit should receive two excitatory signals for them to be on
- Both F1(b) unit or F2 unit should receive two excitatory signals for them to be on
- Both F1(b) unit or F2 unit can receive signals through three possible ways (called two-third rule)
- F1(b) unit should send a signal whenever it receives input from F1(a) and no F2 unit is active

TRAINING ALGORITHM

- **STEP 0:** Initialize the parameters: $\alpha > 1$ and $0 < \rho \leq 1$
- Initialize the weights: $0 < b_{ij}(0) < \frac{\alpha}{\alpha - 1 + n}$, $t_{ji}(0) = 1$
- **STEP 1:** Perform Steps 2-13 when stopping condition is false
- **STEP 2**: Perform Steps 3-12 for each of the training input
- **STEP 3**: Set activations of all F2 units to zero
- Set the activations of F1(a) units to input vectors
- **STEP 4**: Calculate the norm of s:
- $\|s\| = \sum_i s_i$
- **STEP 5:** Send input signal from F1(a) layer to F1(b) layer
- $x_i = s_i$

TRAINING ALGORITHM CONTD…

- **STEP 6:** For each F2 node that is not inhibited, the following rule should hold:
- If $y_i \neq -1$ then $y_i = \sum_i b_{ij} x_i$
- **STEP 7:** Perform steps 8 – 11 when rest is true
- **STEP 8:** Find J for $y_J \geq y_j$ for all nodes j. If $y_J = -1$, then all the nodes are inhibited and note that this pattern cannot be clustered
- **STEP 9:** Recalculate activation X of $F_1(b)$:
- $x_i = s_i t_{ji}$

TRAINING ALGORITHM CONTD…

- **STEP 10:** Recalculate the norm of vector x:
- $\|x\| = \sum_i x_i$
- **STEP 11:** Test for reset condition
- If $\|x\|/\|s\| < \rho$ then inhibit node J, $y_J = -1$. Go back to step 7 again. Else proceed to step 12 (next step)
- **STEP 12:** Perform weight updation for the node J (fast learning):
- $b_{iJ}(new) = \frac{\alpha x_i}{\alpha - 1 + \|x\|}$ and $t_{Ji}(new) = x_i$
- **STEP 13:** test for stopping condition:
- (a) No change in weights (b) No reset of units
- (c) Maximum number of epochs reached

TRAINING ALGORITHM CONTD…

- When calculating the winner unit, if there occurs a tie, the unit with smallest index is chosen as winner
- In Step 3, all the inhibitions obtained from the previous learning trial are removed
- When $y_J = -1$, the node is inhibited and it will be prevented from becoming the winner
- The unit t_{ji} is either 0 or 1 and once it is set to 0, during training, it can never be set back to 1 (provides stable learning method)
- The optimal values of the initial parameters are $\alpha = 2, \rho = 0.9, b_{ij} = 1/(1+n), t_{ji} = 1$
- The algorithm uses fast learning, which uses the fact that the input pattern is presented for a longer period of time for weights to reach equilibrium

EXAMPLE ART1

- Consider an ART1 neural net with four F1 units and three F2 units. After some training the weights are as follows:
- $b_{ij} = \begin{bmatrix} 0.67 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 0.67 & 0.2 \end{bmatrix}$ $t_{ji} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$
- Determine the new weight matrices after the vector [0, 0, 1, 1] is presented if
- (a) The vigilance parameter is 0.3
- (b) The vigilance parameter is 0.7

SOLUTION

- Here, n = 4, m = 3 and $\alpha = 2$. Vigilance parameter $\rho = 0.3$
- Bottom-up weight,
- $$b_{ij}(0) = \frac{1}{1+n} = \frac{1}{1+4} = 0.2$$
- Top-down weight, $t_{ji}(0) = 1$
- Now norm of s = [0 0 1 1], $\|s\| = 0 + 0 + 1 + 1 = 2$
- Then we compute the activations of F1 layer, x = [0 0 1 1]
- We evaluate the net input $y_j = \sum_{i=1}^{4} x_i b_{ij}$
- $y_1 = \sum_{i=1}^{4} x_i b_{i1} = 0.67 \times 0 + 0 \times 0 + 1 \times 0 + 1 \times 0 = 0$
- $y_2 = \sum_{i=1}^{4} x_i b_{i2} = 0 \times 0 + 0 \times 0 + 1 \times 0 + 1 \times 0.67 = 0.67$
-

SOLUTION CONTD…

- $y_3 = \sum_{i=1}^{4} x_i b_{i3} = 0 \times 0.2 + 0 \times 0 + 1 \times 0.2 + 1 \times 0.2 = 0.4$
- Since y2 is the largest, the winner unit is J = 2
- Let us compute the F1 activations function again
- $x_i = s_i t_{ji} = [0011][0001]$T (The second row of the tji matrix)
- = [0 0 0 1] = x
- So, $\|x\| = 0 + 0 + 0 + 1 = 1$
- We now test for reset. Actually, $\frac{\|x\|}{\|s\|} = \frac{1}{2} = 0.5 \geq 0.3(\rho)$. So,
- We update the weights (b_{iJ})
- $b_{iJ}(new) = \frac{dx_i}{\alpha - 1\|x\|}(\alpha = 2)$

SOLUTION CONTD…

- $b_{12} = \frac{2\times0}{2-1+1} = 0, b_{22} = \frac{2\times0}{2-1+1} = 0$
  $b_{32} = \frac{2\times0}{2-1+1} = 0, b_{42} = \frac{2\times1}{2-1+1} = 1$
- Update the top-down weights, $t_{Ji}(new) = x_i$
- The new top-down weights are $t_{Ji} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$
- The new bottom-up weights are $b_{iJ} = \begin{bmatrix} 0.67 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 1 & 0.2 \end{bmatrix}$

SOLUTION CONTD…

- (b) Vigilance parameter $\rho = 0.7$.
- The input vector is s = [0 0 1 1]
- He norm of s is, $\|s\|$= 0 + 0 + 1 + 1 = 2
- Activations of F1 layer as x [0 0 1 1]
- Calculating the net input we obtain
- $y_j = \sum_{i=1}^{4} x_i b_{ij}$

$y_1 = \sum_{i=1}^{4} x_i b_{i1} = 0.67 \times 0 + 0 \times 0 + 1 \times 0 + 1 \times 0 = 0$

- $y_2 = \sum_{i=1}^{4} x_i b_{i2} = 0 \times 0 + 0 \times 0 + 1 \times 0 + 1 \times 0.67 = 0.67$

$y_3 = \sum_{i=1}^{4} x_i b_{i3} = 0 \times 0.2 + 0 \times 0 + 1 \times 0.2 + 1 \times 0.2 = 0.4$

SOLUTION CONTD…

- As y2 is the largest, the winner unit index is J =2
- Recomputing the activations of F1 layer we get
- $x_i = s_i t_{Ji} = [0\ \ 0\ \ 1\ \ 1][0\ 0\ 0\ 1]^T = [0\ 0\ 0\ 1]$
- $\|x\| = 0 + 0 + 0 + 1 = 1$
- Test for reset condition:
- We have $\frac{\|x\|}{\|s\|} = \frac{1}{2} = 0.5 < 0.7(\rho)$
- y2 = -1 (inhibit node 2). So, y1 = 0, y2 = -1, y3 = 0.4
- As the largest one is y3, we have J = 3
- WE now recompute F1 layer activations

SOLUTION CONTD…

- $x_i = s_i t_{Ji} = [0\ 0\ 1\ 1][1\ 1\ 1\ 1]^T = [0\ 0\ 1\ 1]$
- So, $\|x\| = 2$
- Test for the reset condition is:
- $\frac{\|x\|}{\|s\|} = \frac{2}{2} = 1 > 0.7(\rho)$
- Hence we update the weights
- The bottom-up weights are (xi = [0 0 1 1], J = 3)
- $b_{iJ}(new) = \frac{\alpha x_i}{\alpha - 1 + \|x\|}$
- $$b_{13} = \frac{2\times 0}{2-1+2} = 0,\ b_{23} = \frac{2\times 0}{2-1+2} = 0$$
  $$b_{33} = \frac{2\times 1}{2-1+2} = 0.67,\ b_{43} = \frac{2\times 1}{2-1+2} = 0.67$$

SOLUTION CONTD…

- The updated bottom-up weights are
- $b_{iJ} = \begin{bmatrix} 0.67 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.67 \\ 0 & 0.67 & 0.67 \end{bmatrix}$
- The top-down weights are given by $t_{Ji}(new) = x_i$. So,
- $t_{Ji} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$