Page No.
Date

Associative Memory:-

i) Associative memory network can be seen as simplified model of a human brain that associate similar patterns.

ii) It exhibits hebbian learning.

iii) An associative memory is a storehouse of associated patterns which are encoded in some form.

iv) When the storehouse is triggered with a pattern, the associated pair is recalled.

"0" ⟶ $\boxed{\div \times \div 0 -}$ ⟶ "1"

(input pattern) (storehouse) (output pattern)

v) AM can be classified as Static and Dynamic network.

vi) Static: recalls output in one single feed-forward pass.

Dynamic: recalls through recurrent pass

vii) AM is of two types:-

a) Auto associative memory

b) Hetero associative memory

Autoassociative Memory:-

It is a capable of retrieving a piece of data upon presentation of only partial information from that piece of data.

Neural network using auto-associative memory are called Auto-associative network

i) Training input vector and target output vector are same.
ii) Uses Hebb's rule / Outer product rule to find the weights of an associative memory neural network.
iii) The input and output are connected through weighted connection.

(iv) 'n' number of inputs training vectors and similar for output vectors.

Training Algorithm: For training the network, we are using Hebb or Delta learning rule.

Step 1: Initialize all the weights to zero as $w_{ij} = 0$, $i = 1$ to n, $j = 1$ to n

Step 2: Perform steps 3-4 for each input vector

Step 3: Activate each input unit as follows-

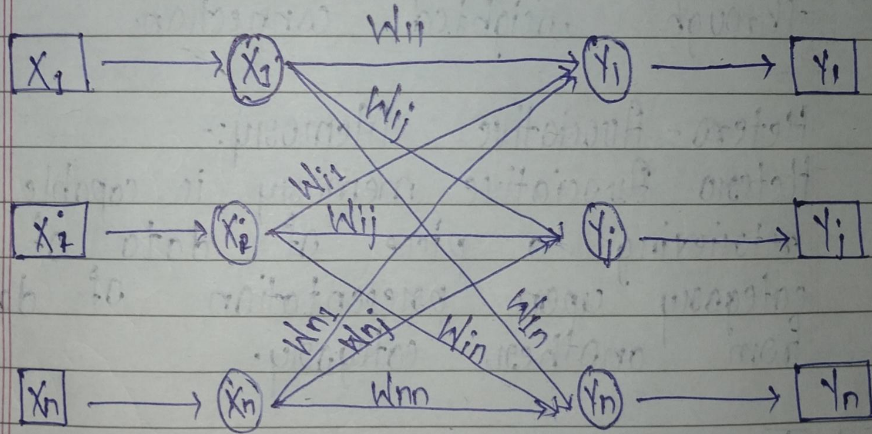
$$x_i = s_i \,(i = 1 \text{ to } n)$$

Step 4: Activate each output unit as follows-

$$y_j = s_j \,(j = 1 \text{ to } n)$$

Step 5: Adjust the weights as follows-

$$w_{ij}(new) = w_{ij}(old) + x_i y_j$$



Testing Algorithm S1: Set the weights obtained during training for ~~Hb~~ Hebb's rule.

Step 2: Perform steps 3-5 for each input vector.

Step 3: Set the activation of the input unit's equal to that of input vector

Page No. Date

Step 4: Calculate the net input to each output unit $j = 1$ to n

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5: Apply the following activation function to calculate output

$$y_j = f(y_{inj}) = \begin{cases} +1 & \text{if } y_{inj} > 0 \\ -1 & \text{if } y_{inj} \le 0 \end{cases}$$

Hetero-Associative Memory:-

Hetero Associative memory is capable of retrieving a piece of data of one category upon presentation of data from another category.

i) Training input vector and target output vector are different.
ii) Uses Hebb's rule / Delta rule / Outer product rule to find the weight matrix.
iii) The input vector has 'n' inputs similarly for output vector.

Training Algorithm: For training the network, we are using Hebb or Delta learning rule.

Step 1: Initialize all the weights to zero as $w_{ij} = 0$, $i = 1$ to n, $j = 1$ to n

Step 2: Perform steps 3-4 for each input vector

Step 3: Activate each input unit as follows-

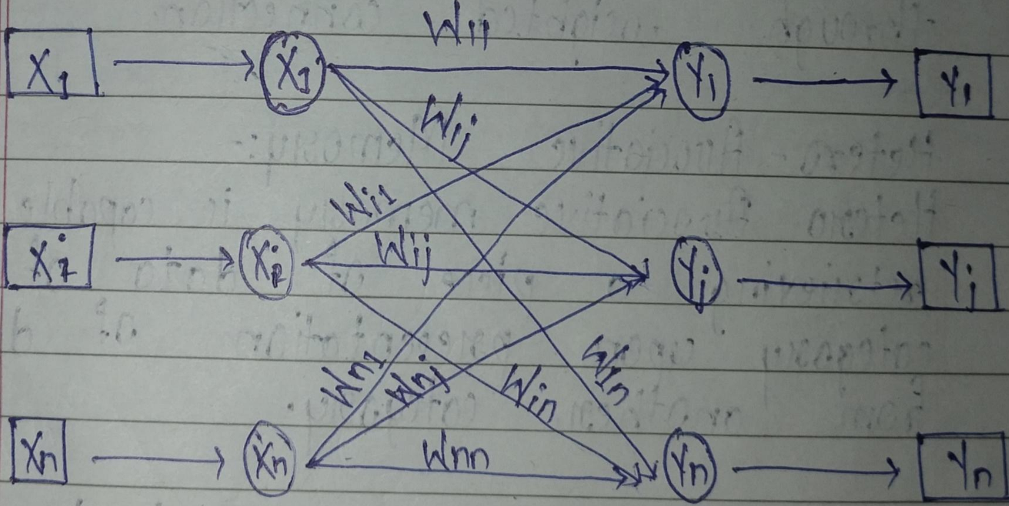
$$X_i = S_i \ (i = 1 \text{ to } n)$$

Step 4: Activate each output unit as follows-

$$Y_j = S_j \ (j = 1 \text{ to } n)$$

Step 5: Adjust the weights as follows-

$$w_{ij}(new) = w_{ij}(old) + X_i Y_j$$



Testing Algorithm: **S1:** Set the weights obtained during training for ~~Hb~~ Hebb's rule.

Step 2: Perform steps 3-5 for each input vector.

Step 3: Set the activation of the input unit's equal to that of input vector

Page No.

Date

Step 4: Calculate the net input to each output unit $j = 1$ to n

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5: Apply the following activation function to calculate output

$$y_j = f(y_{inj}) = \begin{cases} +1 & \text{if } y_{inj} > 0 \\ -1 & \text{if } y_{inj} < 0 \\ 0 & \text{if } y_{inj} = 0 \end{cases}$$

Page No.
Date

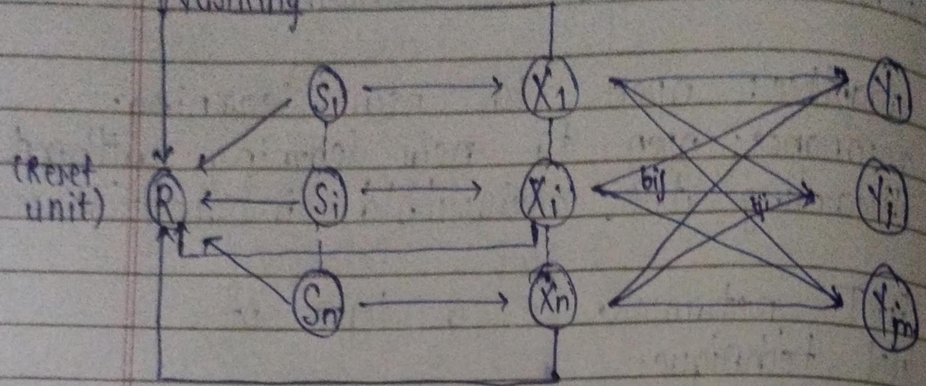
Adaptive Resonance Theory:-

adaptive: open to new learning.
resonance: open to new learning without discarding the previous/old information.

i) ART networks is a type of NN technique.
ii) ART networks are known to solve stability-plasticity dilemma i.e stability: nature of memorizing the learning, plasticity: refers to they are flexible to gain new information.
iii) Input is presented to the network and algorithm checks whether it fits into one of the ~~all~~ already stored clusters.
iv) If it fits input is added to the cluster that matches the most else new cluster is formed.

Types of ART:-
i) ART1: Capable of clustering binary input values.
ii) ART2: extension of ART1, capable of clustering continuous valued input data.
iii) Fuzzy ART: augmentation of fuzzy logic and ART.
iv) ARTMAP: supervised learning form of ART learning where one ART learns based on previous ART module. Also known as predictive ART.

Working:-



1. Computational Unit:-

a) Input unit

F_1 (a) layer input portion :- There would be no of processing in this portion rather than having input vector only. It is connected to F_1(b).

F_1 (b) layer Interface portion :- These portion combine signal from i/p portion with that F_2 layer.

b) Cluster unit :-

This is a competitive layer. The input is selected to learn the i/p pattern. Activation of all clusters unit are set to 0.

c) Reset Mechanism:-

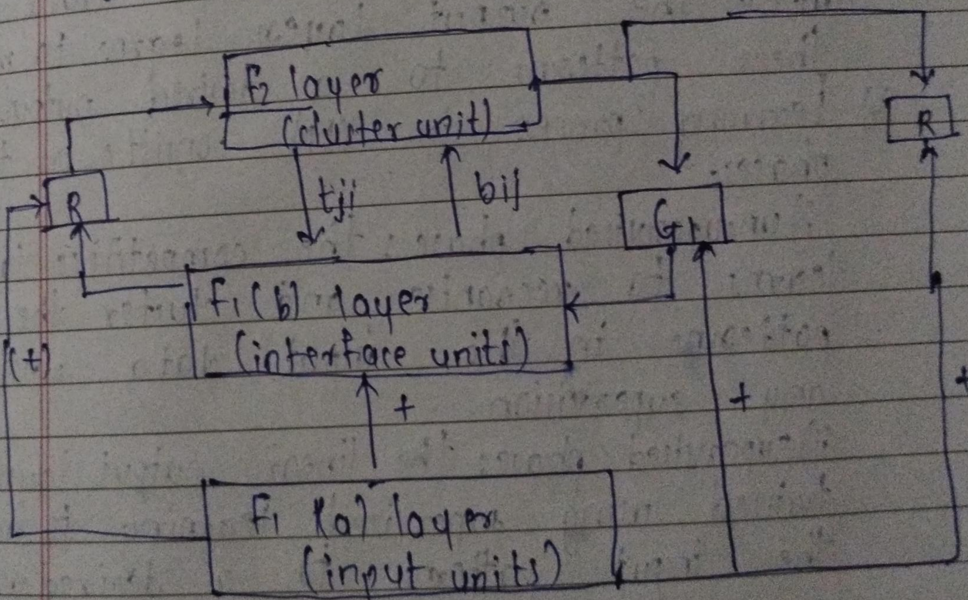
It is based on similarity vector how if degree of this similarity is less

less than vigilance, then the cluster is not allowed to learn pattern and a reset would happen.

2) Supplement unit :- Due to issue with reset mechanism (supplement unit) are added along with reset unit R. They are called gain control units.

+ → indicate excitatory signal

- → indicate inhibitory signal



bij = weight from F1b to F2 layer.
bottom-up weights

tji = top-down weights.

Applications:-

i) Target Recognition

ii) Signature Verification

iii) Medical Diagnosis.

Counter Propagation Network:-

i) CPN is a Artifical Neural Network architecture that combines both supervised and unsupervised learning.

ii) CPN consist of two layers of neurons: a competitive layer and a linear output layer.

iii) During training phase, the competitive layer learns to recognise and classify the patterns in the input data while the output layer learns to map these patterns to a desired output.

iv) Learning process in CPN consists of two phases.

i) unsupervised phase: the competitive layer learns to recognise and cluster the patterns in the input data without any supervision.

ii) supervised phase: the linear output layer is trained using supervised learning to map the input patterns to a desired output

Applications:-

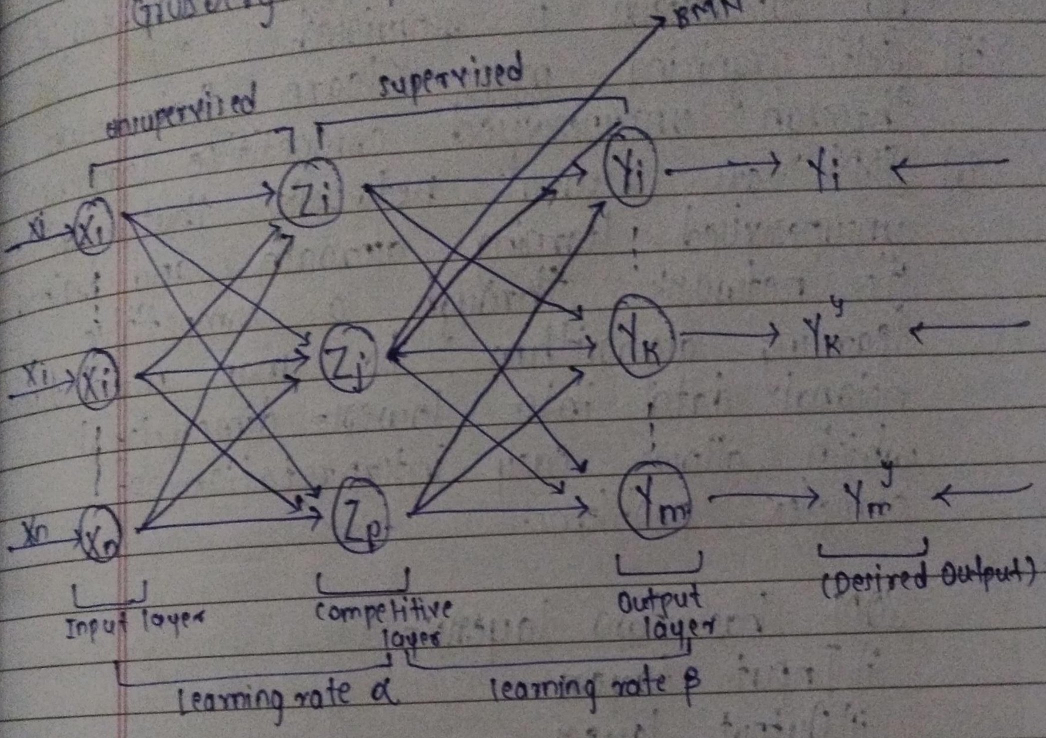
i) Pattern Recognition.

ii) Signal Processing

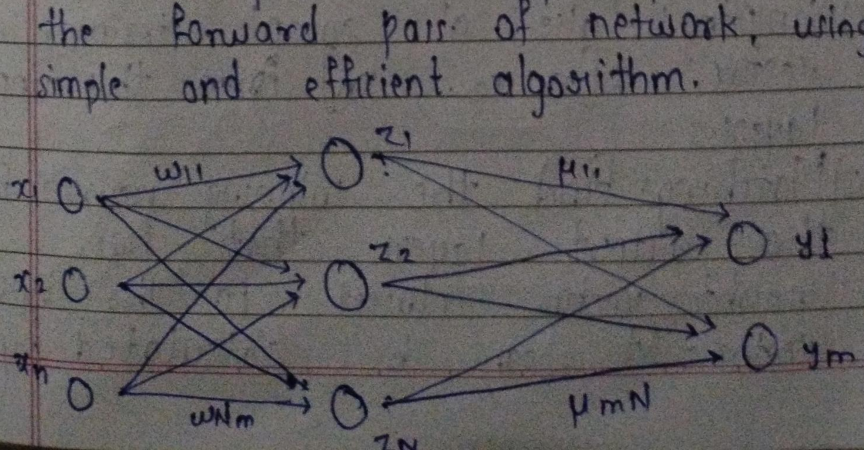
iii) Data Compression

Types of Counter propagation network:

i) Full counterpropagation network: It is composed of two layers: a Kohonen layer (self-organizing layer) and a Grosberg layer.



ii) Foward only counter propogation: It consist of two layers: a Kohonen layer and a Grossberg layer, similar to full CPN. In this, the weights in both layers are only updated during the forward pass of network, using a simple and efficient algorithm.



Self-Organizing Maps

Self-Organizing Maps:-

i) It was developed by T. Kohonen in 1982.
ii) SOM named as "self-organizing" because no supervision is required.
iii) Self-orgnizing maps learn their own through unsupervised competitive learning.
iv) SOM are neural networks that uses unsupervised learning approach and trained it's network through a competitive learning algorithm to map multi-dimensional data into lower-dimensional which allows easy interpretation of complex problems.

SOM has two layers:-
i) Input layer
ii) Output layer

SOM training algorithm:-
i) Initilaization: Choose random values for the initial weights.
ii) Sampling: Take a sample training input vector $x[x_1, x_2, \cdots x_n]$ from the input layer.
iii) Matching: Find the winning neuron from the output layer that has weight vector closest to the input vector. It can be calculated by

taking the square of Euclidean distance for each output layer and and find the output unit that has minimum euclidean distance. from input vector.

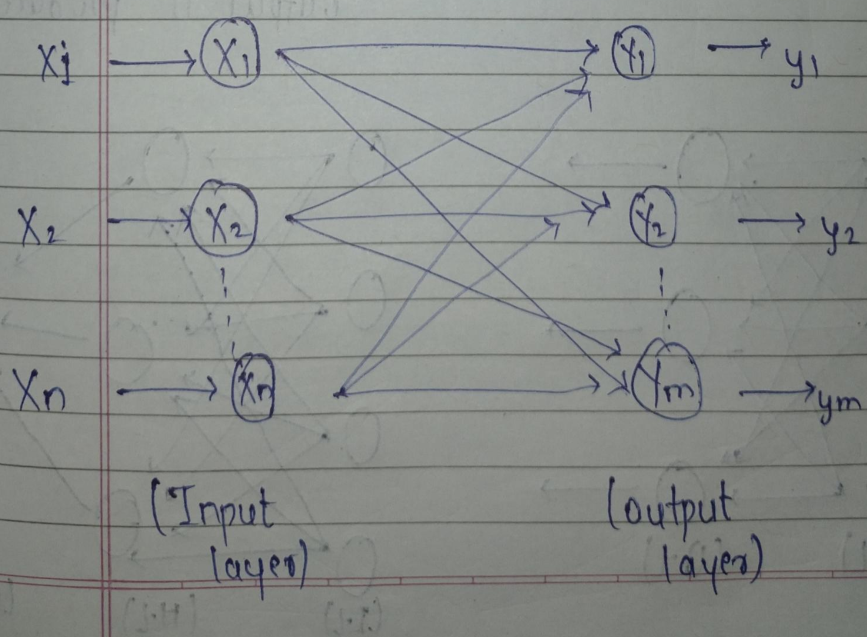
j=1 to n

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

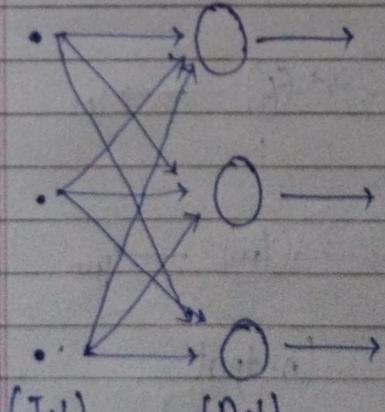
iv) New weight calculation: find the new weights between input vector sample and winning output unit (neuron)

$$W_{ij}(new) = W_{ij}(old) + \alpha \{x_i - W_{ij}(old)\}$$

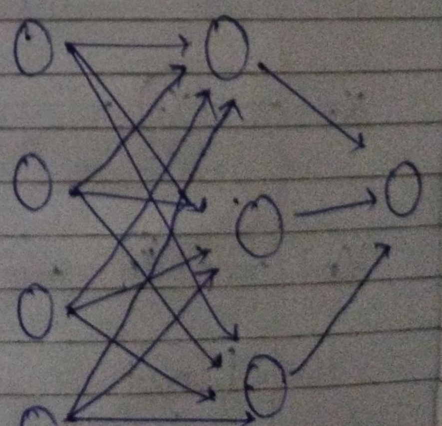
v Continuation: Repeat step 2 to 4 until weight updation is negligible (i.e new weights are similar to old weights) or feature map stop changing



Single layer Perceptron	Multilayer Perceptron
i) Layer is formed by taking processing elements & combining it with other processing elements.	It is formed by interconnection of several layers.
ii) Input and output are linked with each other.	There are multiple layer's between input & output layer which are known as hidden layer.
iii) Zero hidden layer are present.	Zero to several hidden layers are there in a network.
iv) Not efficient in certain areas.	More the hidden layers more the complexity of networks, but efficient output is produced.



(I·L) (O·L)



(I·L) (H·L) (O·L)