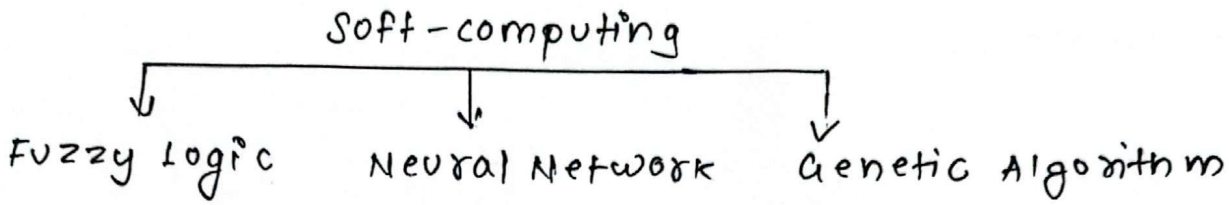
# (PE-2): soft computing paradigms
## Notes For Endsems

# soft-computing

Definition: It is the use of Approximate calculations to provide Approximate but usable solutions to complex computational problems.



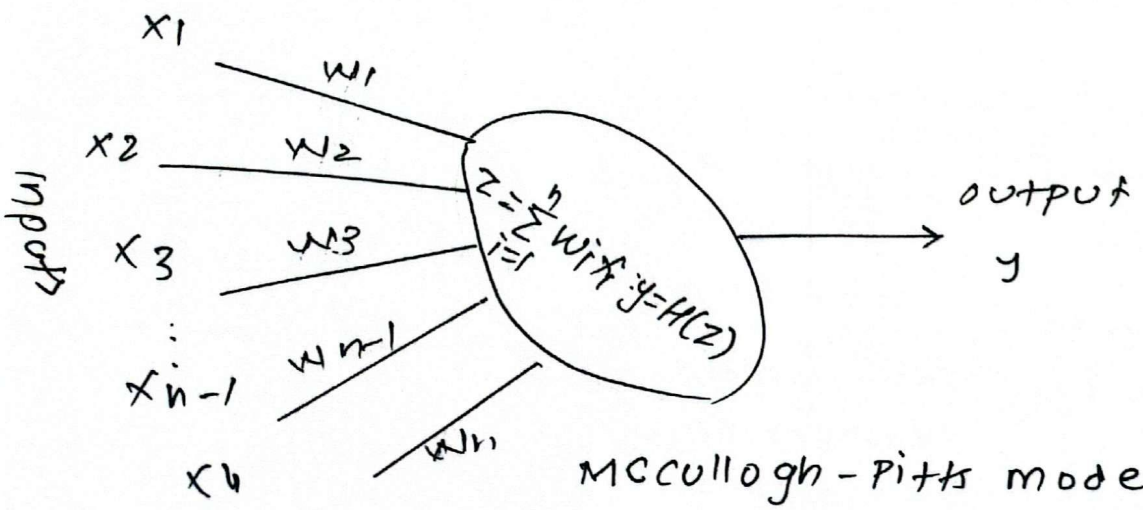

Characteristics:

1. Does not require any mathematical model to solve a problem
2. It may not yield to precise solutions
3. Algorithms are adaptive in nature
4. Low cost solution
5. uncertainity in output.

## Neural Networks

Artificial neurons

Neurons work by processing information. They receive and provide information in form of spikes




MCCullogh-Pitts model

Non-linear generalization of MC Cullogh - Pitts neuron

$$y = f(x, w)$$

y : neuron's output

Example

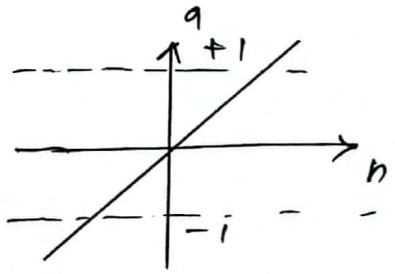
$$y = \frac{1}{1 + e^{-w^T x - a}}$$ sigmoidal neuron

$$y = e^{-\frac{\|x - w\|^2}{2a^2}}$$ Gaussian neuron

## Activation functions

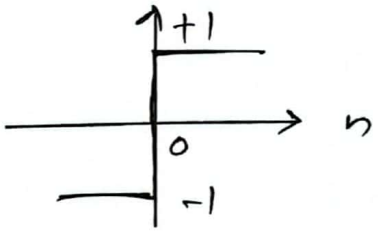
Activation function is generally non-linear

Linear fn are limited because the output is simply proportional to input



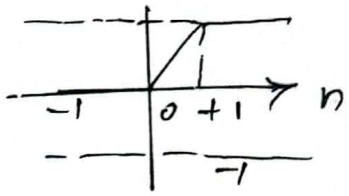

Linear Transfer function

a = purelin(n)



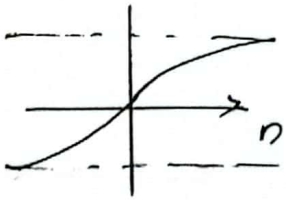

a = hardlims(n)

Symmetric Hard Limit Trans fn




a = satlin(n)

Satlin Transfer function

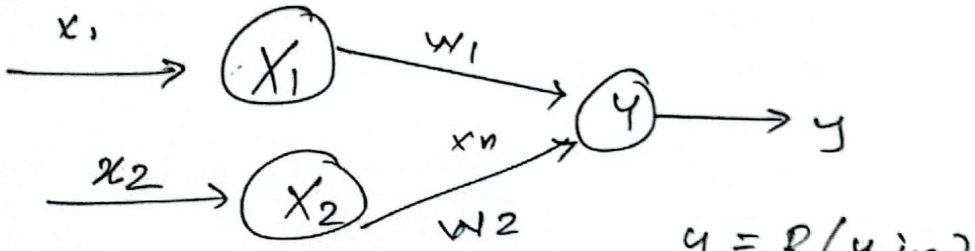



a = tansig)

Tan-sigmoid transfer fn

## ANN : Artificial Neural Networks

- ANN posses a large number of processing elements called nodes/neurons which operate in parallel
- Neurons are connected with others by connection link
- Each link is associated with weights which contain information about input signal
- Each neuron has internal state of its own which is a Fn of inputs that neuron receives activation levels

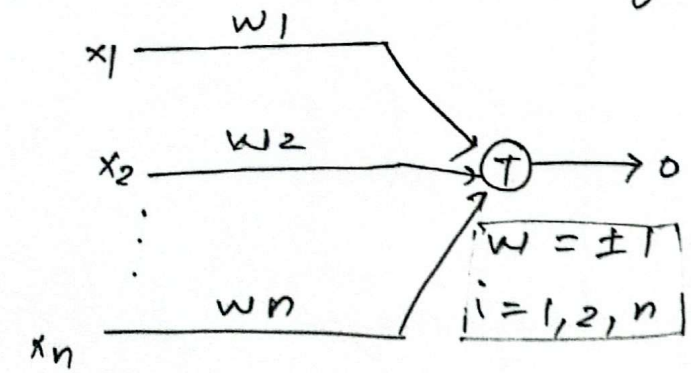


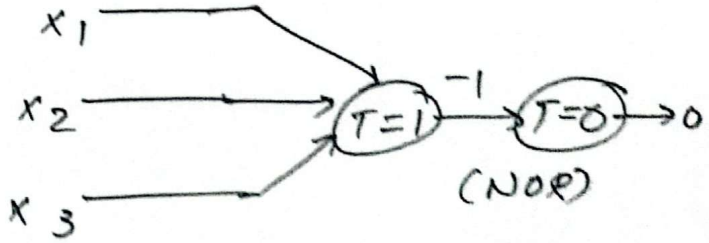

$y = f(y_{in})$

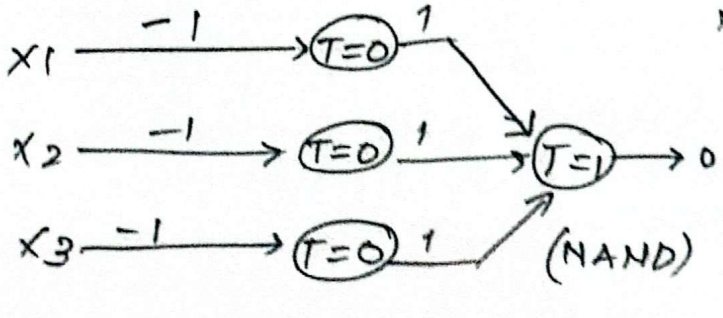
$$y_{in} = x_1 w_1 + x_2 w_2$$

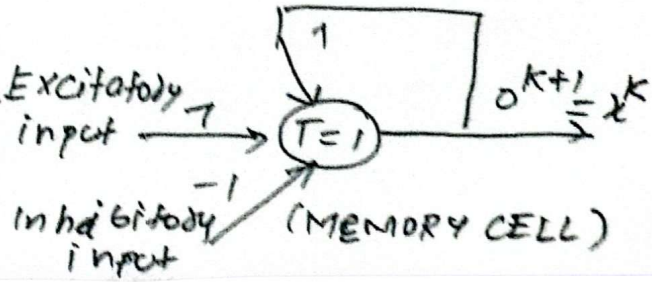
## McCulloch - Pitts Neuron Model

$$o^{k+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$



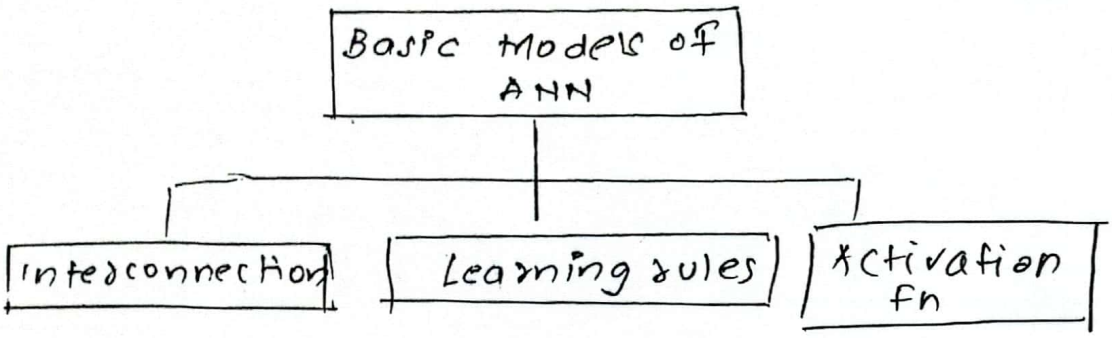



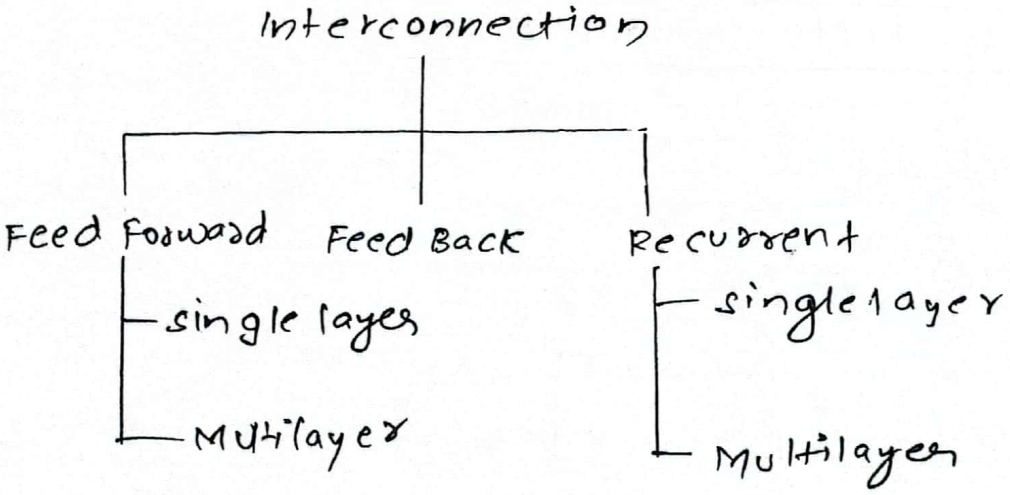
## Features of McCulloch-Pitts model

* Allow binary 0,1 states
* operates under discrete time assumption
* Weights and the neurons' thresholds are fixed in the model and no interaction among network neurons.
* Just a primitive model

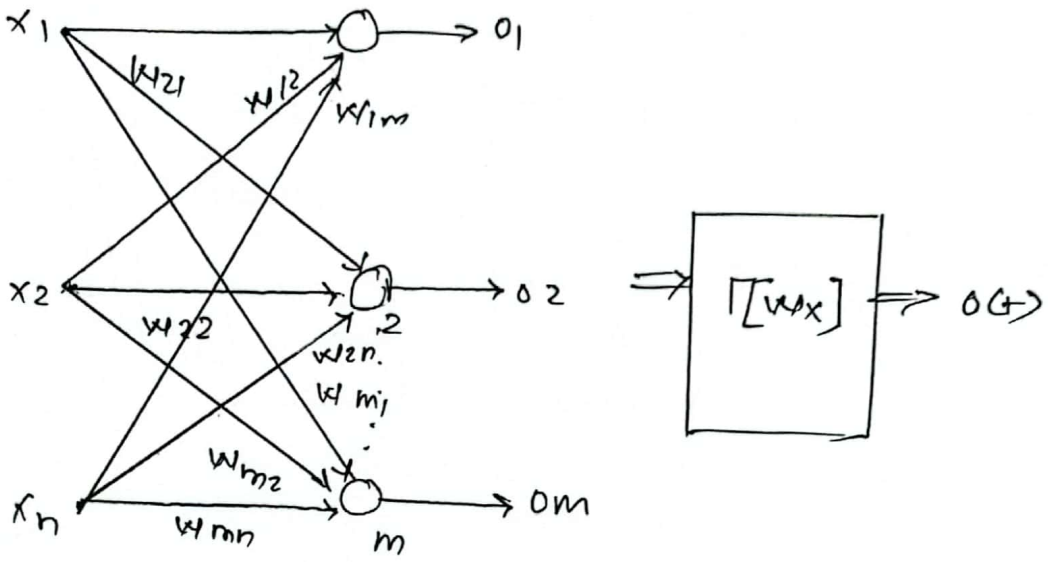
Basic model of ANN



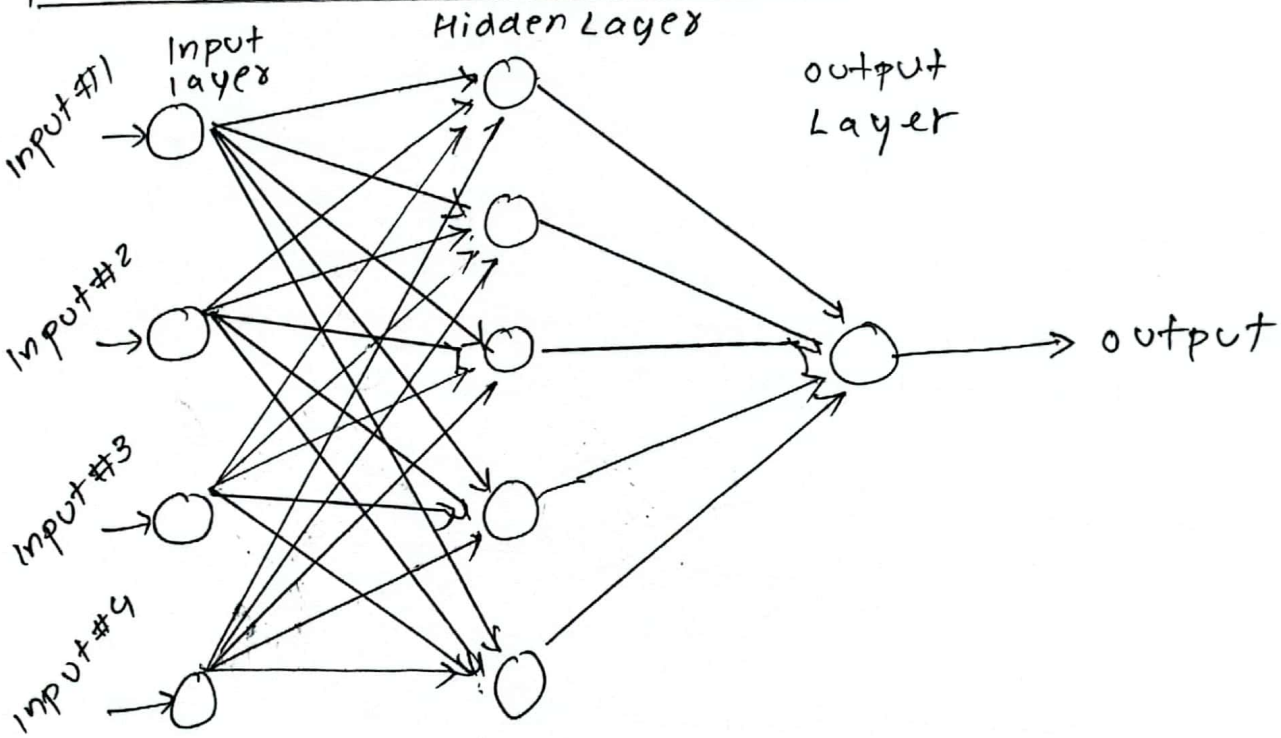

classification based on interconnections



## single layer Feedforward Network

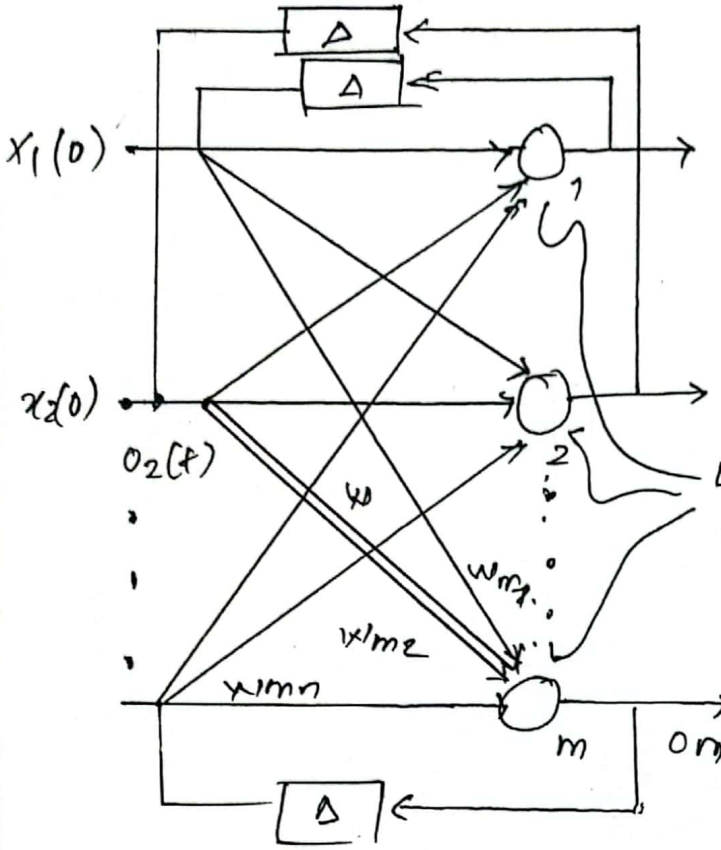


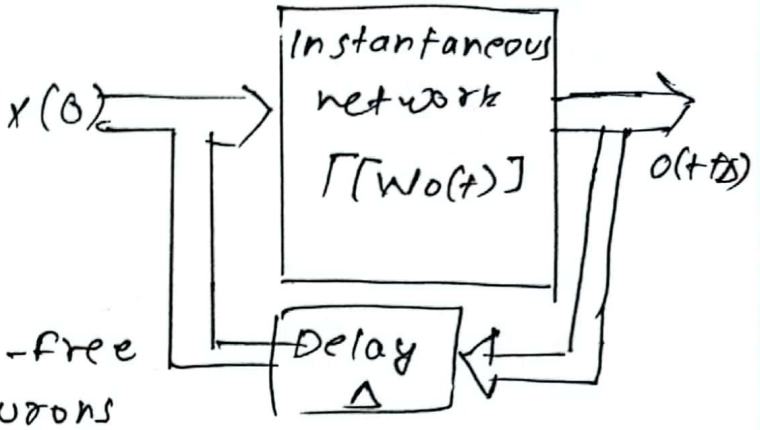

## Multiplayer Feed Forwaad network




## Feedback network

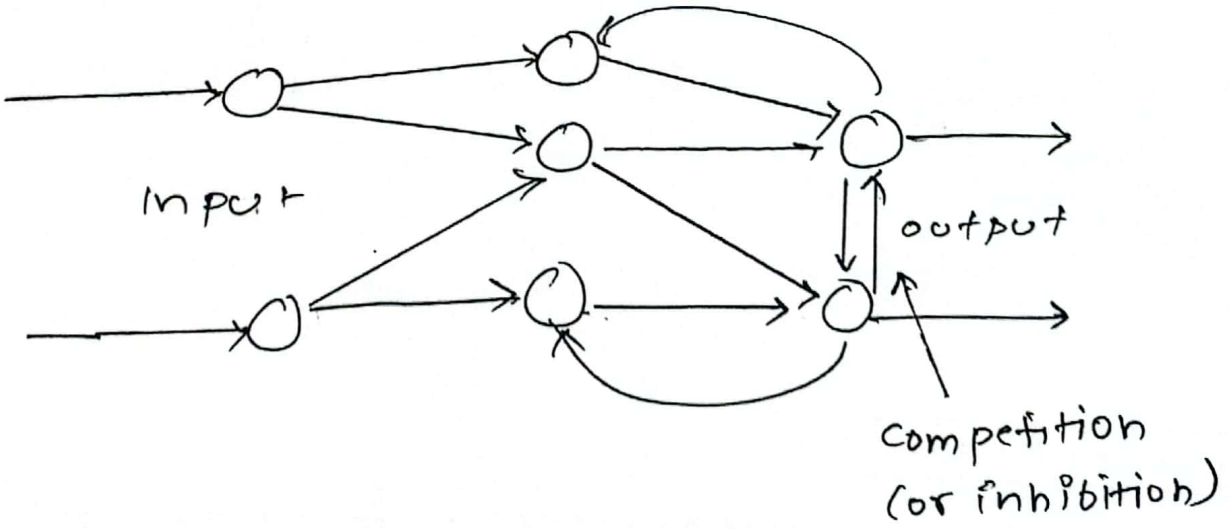
When outputs are directed backs as input to same or preceding layers nodes it results in the formation of feedback networks.



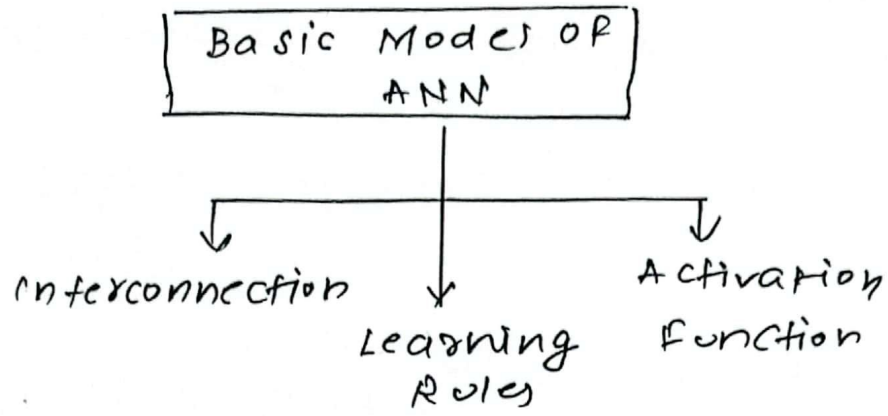




## Lateral Feedback

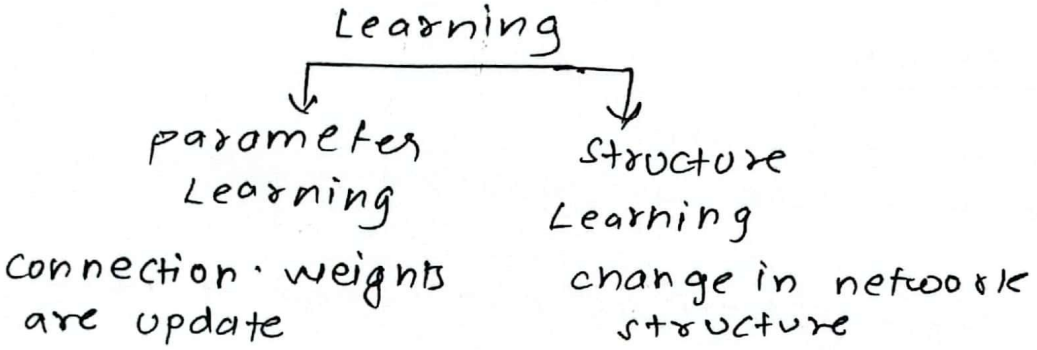
If the feedback of the output of the processing elements is directed back as input to the processing elements in the same layer then it is called lateral feedback



## Basic models of ANN




learning:- is a process by which a NN adapts itself to a stimulus by making proper parameter adjustments, resulting in the production of desired response.



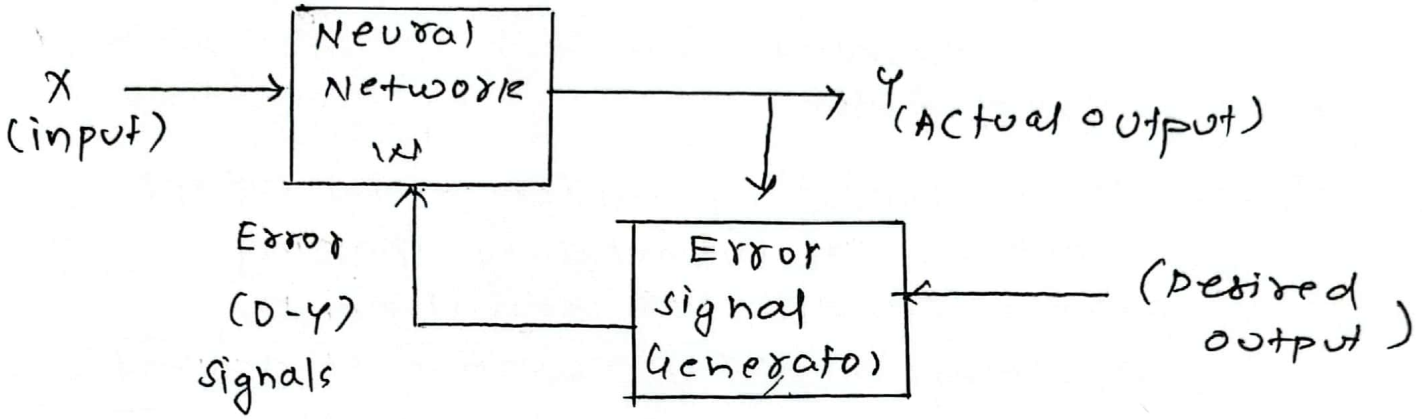

### Training:-

The process of modifying the weights in the connections between network layers with the objective of achieving the expected output is called training a network

achived through → Supervised learning
→ unsupervised learning
→ Reinforcement learning

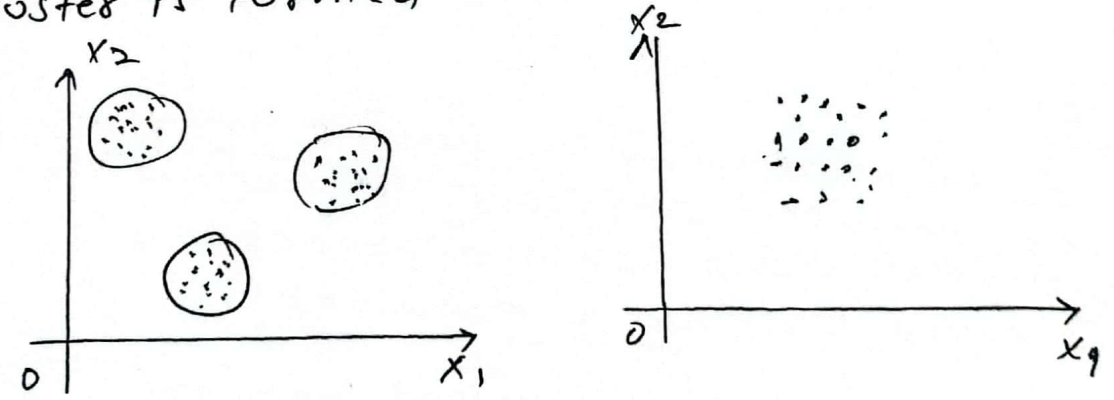
## Supervised Learning

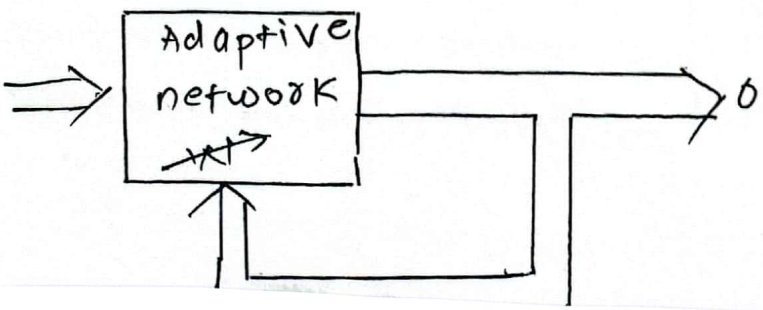
* child learns from a teacher
* Each input vector requires a corresponding target vector.
* training pair = [input vector, target vector]




## Unsupervised Learning

* How a fish or tadpole learns
* All similar input patterns are grouped together as clusters
* If a matching input pattern is not found a new cluster is formed

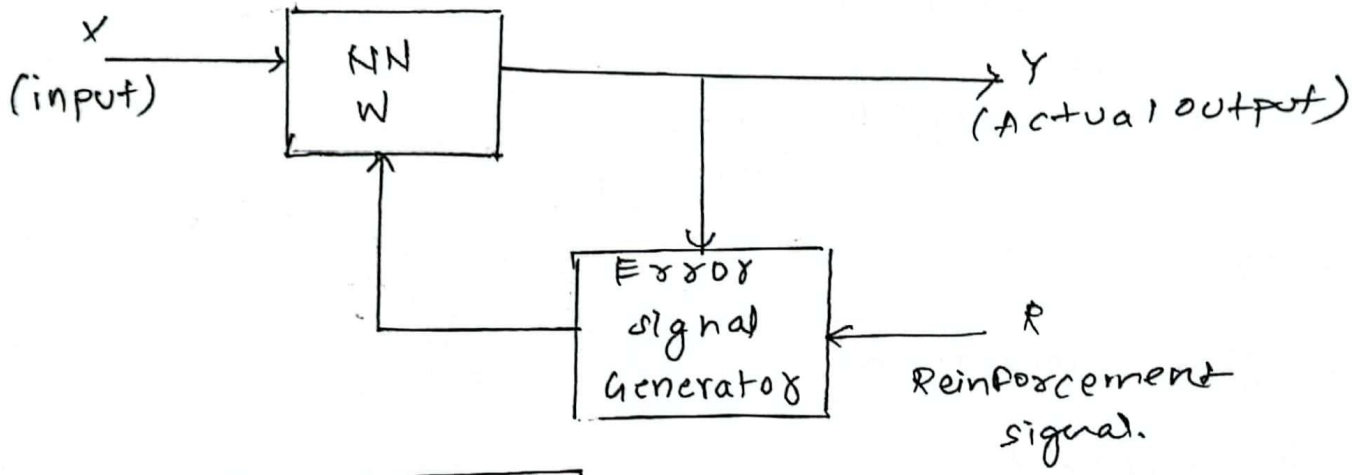


## Self-organizing

- In unsupervised learning there is no feedback.
- Network must discover patterns, regularities, features for input data over the output.
- While doing so the network might change in ~~patterns~~ parameters
- This process is called self-organizing

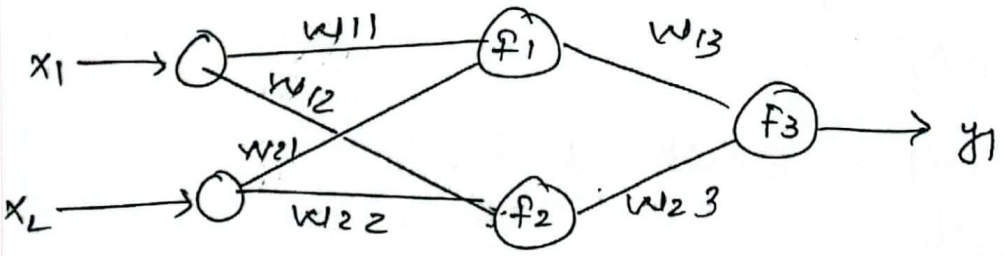
## Reinforcement Learning




## Basic Models of ANN

- Interconnections
- Learning Rules
- Activation Functions

## perceptron Network structure



## perceptron training

i) setup network i/p nodes, layer(1) & connections

ii) Randomly assign weight $w_{ij}$ & bias $\theta_j$

iii) input training sets $x_i$ (preparing $T_j$ for verification)

iv) training computation

$$net_j = \sum w_{ij} x_i - \theta_j$$

$$Y_i = \begin{cases} 1 & \text{if } net_j > 0 \\ 0 & net_j \leq 0 \end{cases}$$

Training process

Trainining computation

if $(T_j - Y_j) \neq 0$

then

$$\Delta w_{ij} = \eta (T_j - Y_j) x_i$$

$$\Delta \theta_j = -\eta (T_j - Y_j)$$

update weights and bias

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\theta_j = \theta_j + \Delta \theta_j$$

repeat 3 & 5 until ∀ input pattern is satisfied

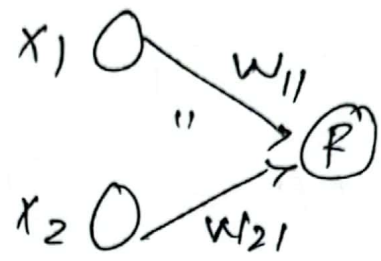
$$(T_j - Y_j) == 0$$

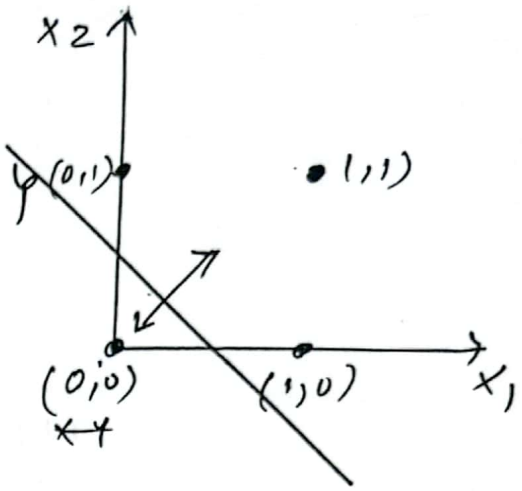
## Exercise 1: Solving the OR problem

Q1)

Let training pattern are as follows

| $x_1$ | $x_2$ | T | y |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | |
| 1 | 1 | 1 | |




Let $w_{11} = 1$ $w_{21} = 0.5$ $\theta = 0.5$

Let learning rate $\eta = 0.1$

initial net Function

$$net = w_{11} \cdot x_1 + w_{21} \cdot x_2 - \theta$$

$$i.e = 1 \cdot x_1 + 0.5 w_{21} - 0.5$$

Feed the input pattern into network one by one.

(0,0) : $net = (0 \times 1) + (0.5 \times 0) - 0.5 = 0$ $net = -0.5$

$(T - y) = (0 - 0) = 0$ O.K

(0,1) : $net = (1 \times 0) + (0.5 \times 1) - 0.5 = 0.5 - 0.5 = 0$

$(T - y) = (1 - 0) = 1$ need to update weight

(1,0) $net = 0.5$ $y = 1$ $(T - y) = (1 - 1) = 0$ O.K

(1,1) $net = 1$ $y = 1$ $(T - y) = (1 - 1) = 0$ O.K

$T_j - y_j \neq 0$

then now

$$\Delta w_{ij} = \eta (T_j - y_j) x_i$$

$$\Delta \theta_j = -\eta (T_j - y_j)$$

$\Delta W_{11} = (0.1)(1)(0) = 0$ $\quad W_{11} = 0+1 = 1$

$\Delta W_{12} = (0.1)(1)(1) = 0.1$

$W_{21} = 0.5 + 0.1 = 0.6$

$\Delta\theta = -(0.1)(1) = -0.1$

$\theta = \Delta\theta + \theta = -0.1 + 0.5 = 0.4$

$$\boxed{net = (1 \cdot x_1) + (0.6 \times x_2) - 0.4}$$

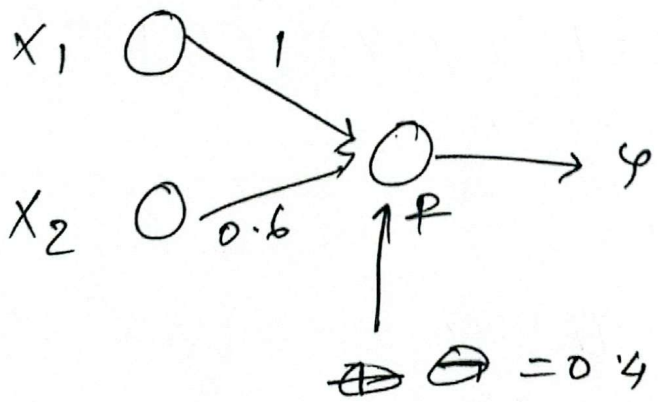
Test Again for patterns
(0,1), (1,0) (1,1) & (0,0)

if all $(T - Y) = 0$ is O.K

4. Then all patterns are satisfied
Hence network is successfully trained for OR patterns

so we get final network is

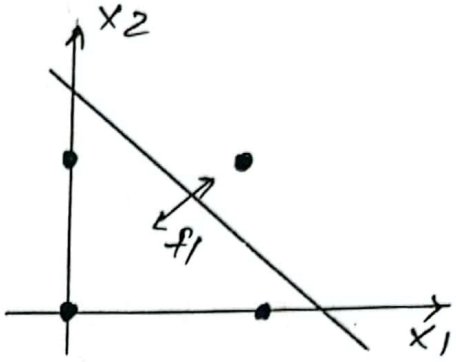
$$\boxed{net = 1 \cdot x_1 + 0.6 x_2 - 0.4}$$



## Exercise 2: solving the AND problem

Let the training patterns are used as follows

| $X_1$ | $X_2$ | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




Let $W_{11} = 0.5$ $W_{21} = 0.5$ $\underline{\theta = 1}$ (bias) $\underline{\eta = 0.1}$ (learning rate)

$net = X_{11}W_{11} + X_{21}W_{21} - \theta$

$= 0.5X_{11} + 0.5X_{21} - 1$

Feed input pattern in network

$(0,0)$ : $net = (0.5 \times 0) + (0.5 \times 0) - 1 = -1 : Y = 0$

$(T - Y) = (0 - 0) = 0$ ok.

$(0,1)$ : $net = (0.5 \times 0) + (0.5 \times 1) - 1 = 0 + 1 - 1 = 0$ $Y = 0$

$(T - Y) = (0 - 0) = 0$ Ok.

$(1,0)$ : $net = (0.5 \times 1) + (0.5 \times 0) - 1 = 1 - 1 = 0$ $Y = 0$

$(T - Y) = 0 - 0 = 0$ OK

$(1,1)$ : $net = (0.5 \times 1) + (0.5 \times 1) - 1 = ++$

$0.5 + 0.5 - 1 = 1 - 1 = 0$ $Y = 0$

$(T - Y) = 0 \Rightarrow (1 - 0) = 1$

update weight.

update weights for pattern (1,1) which does not satisfy exp o/p

$$\Delta W_{11} = \eta (T_j - Y_j) x_i$$

$$= (0.1)(1)(1)$$

$$\Delta W_{11} = 0.1$$

$$W_{11} = 0.5 + 0.1 = 0.6$$

$$\Delta W_{21} = \eta (T_j - Y_i) x_1$$

$$= (0.1)(1)(1)$$

$$= 0.1$$

$$W_{21} = W_{21} + \Delta W_{21} =$$

$$= 0.1 + 0.5 = 0.6$$

$$\Delta \theta = -(0.1)(1) = -0.1$$

$$\theta = \Delta\theta + \theta = -0.1 + 1 = 0.9$$

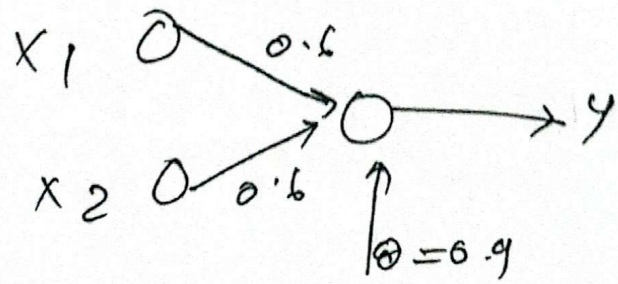
$$\boxed{\theta = 0.9}$$

Apply new weights

$$\boxed{net = 0.6 x_1 + 0.6 x_2 - 0.9}$$

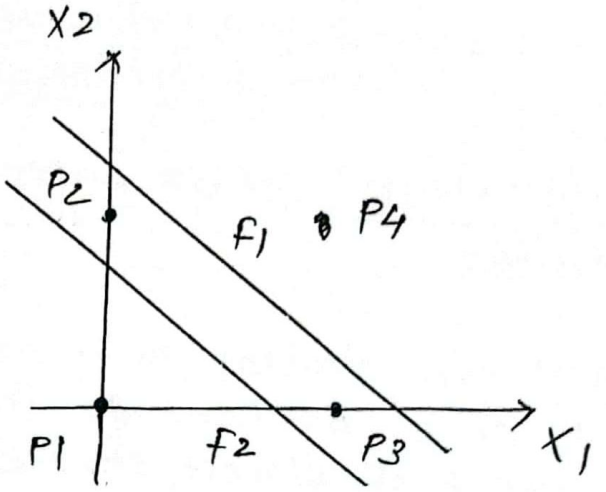
if all pattern $(T - Y) = 0$ then

pattern Recognition function



Q3) Solving the XOR problem

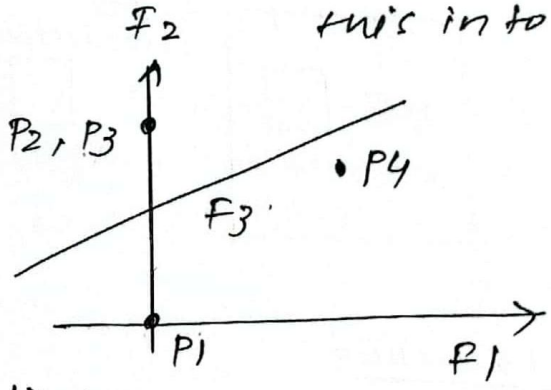
| $x_1$ | $x_2$ | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




Let $W_{11} = 1.0$ $W_{21} = -1.0$

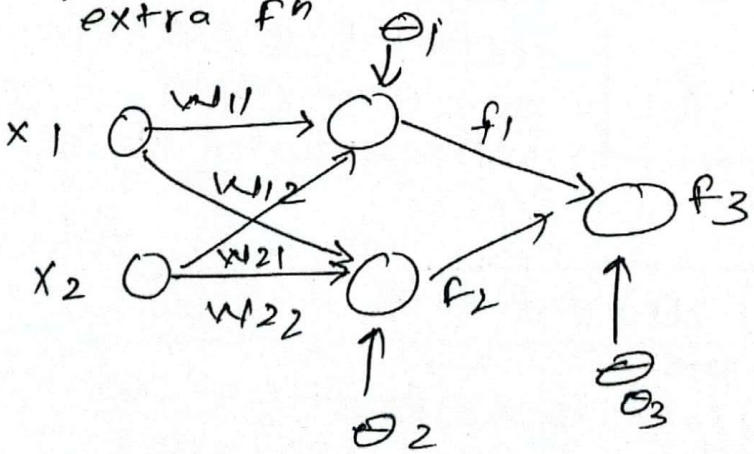
$\theta = 0$

we can converge this into




if we choose one layer network it will be proved that the network cannot be converged. because XOR problem is non-linear problem. one single -Linear fn is not enough to recognize pattern

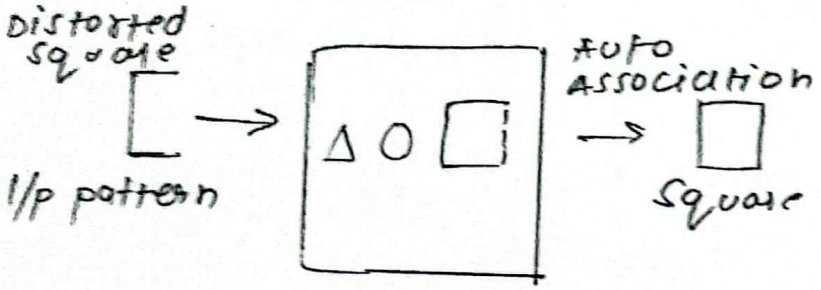
Therefore solution is to add one hidden Layer for extra fn



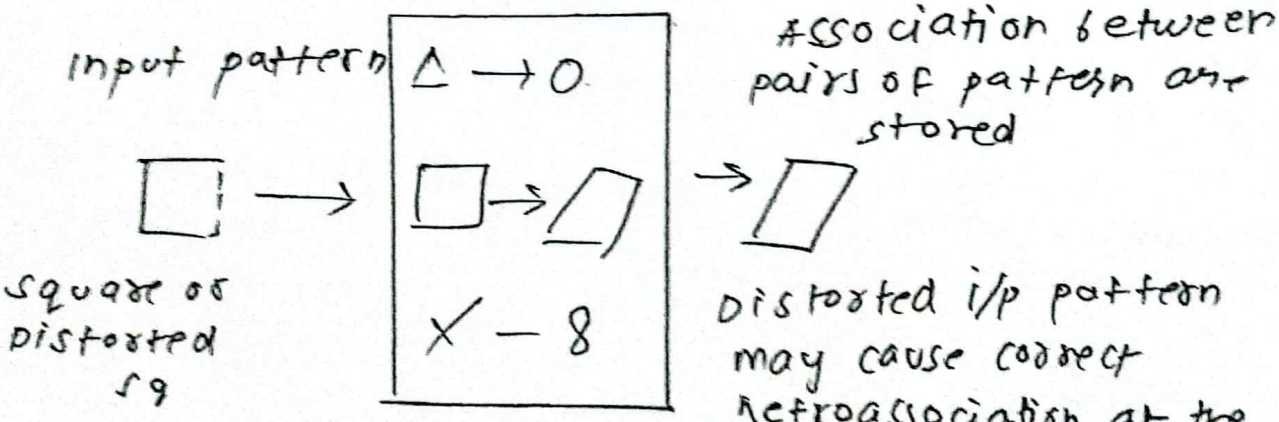
Neural processing → Auto-association
→ Hetro-association
→ classification

**Autoassociation:** Set of patterns can be stored in network

If a pattern similar to a member of the stored set is presented, an association with input of closest stored patten

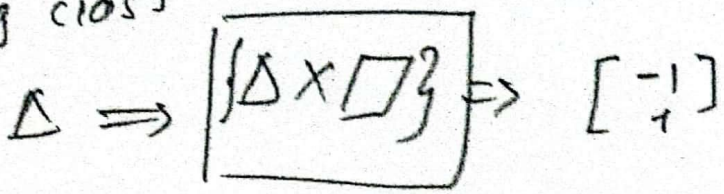



**Hetro-association**



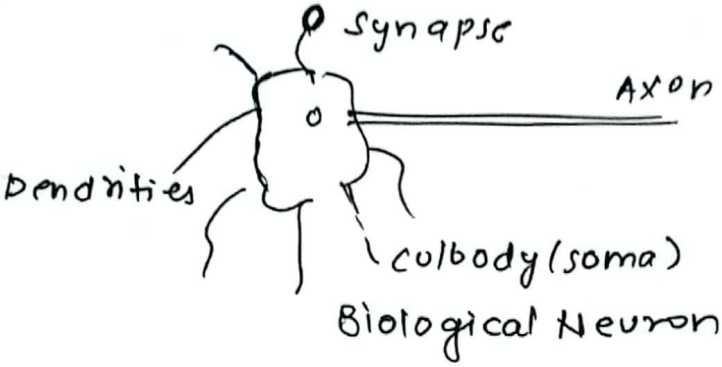

**Classification**

Set of I/p patterns is divided into a no. of classes or categories in response to an i/p pattern from the set classifier is suppose to recall information regarding class



## 4 | Hebb / Hebbian Rule

Biological Definition: Learning occurs in humans due to to the change in synaptic gaps



Biological Neuron

### Hebb rule:

→ When an axon of cell A is near enough to excite cell B repeatedly / permanently firing it, then growth / metabolic change place in both of the cells

Unsupervised algorithm

### Hebb Law :-

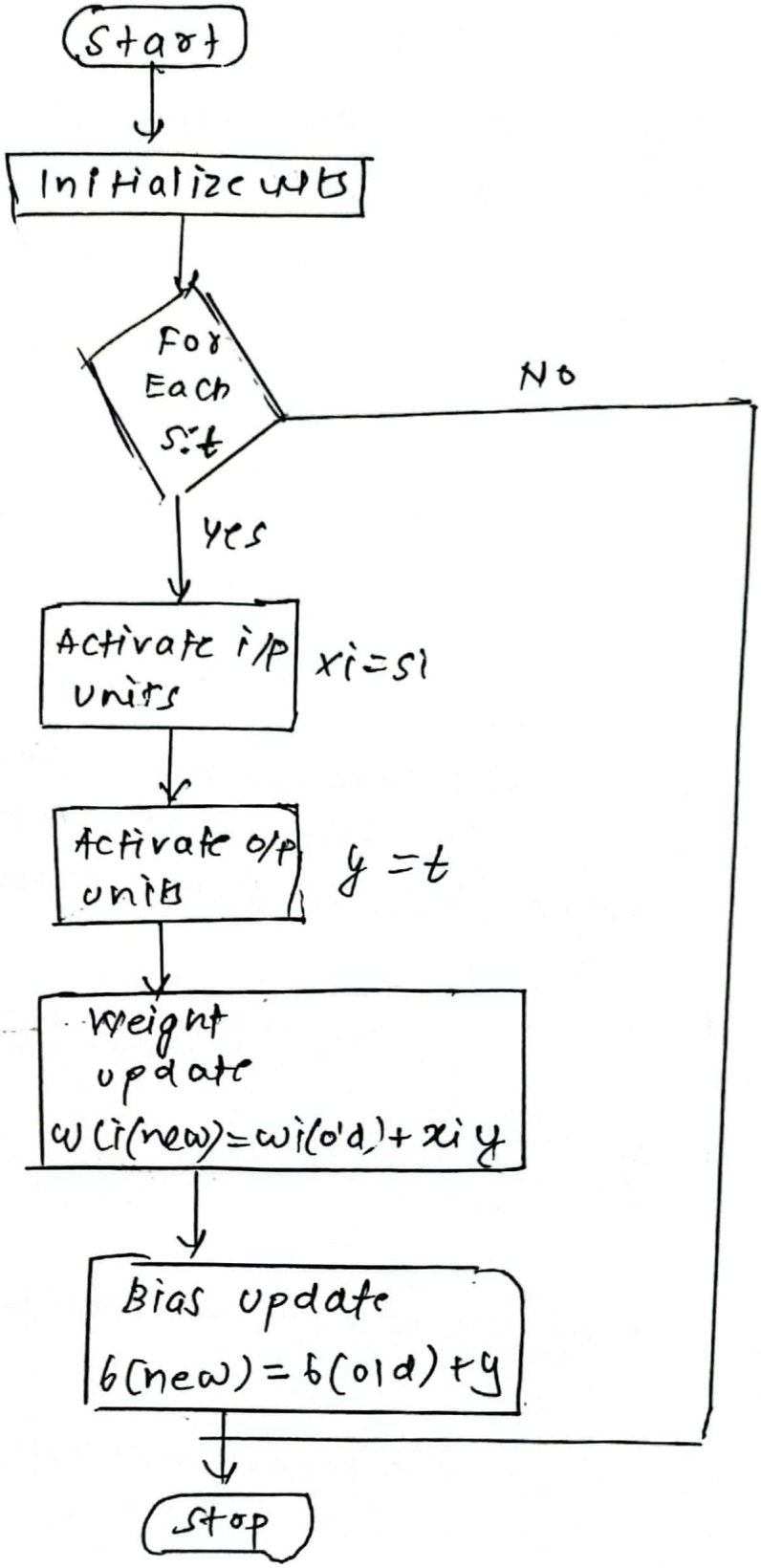
i) If 2 neurons on either side ⇒ synchronously ⇒ weight increased

ii) " " ⇒ 1 ⇒ wt Asynchronously ⇒ wt decreased

$$w_i(new) = w_i(old) + x_i y$$ → change in synaptic gap

$x_i$ → i/p vector

$y$ → o/p vector


Start
Initialize Wts
For Each S:t
No
Yes
Activate i/p units
xi=si
Activate o/p units
y=t
Weight update
w(i(new)=wi(old)+xi y
Bias update
b(new)=b(old)+y
Stop

Training Algorithm

step 0: Initialize the weights & bias

step 1: For each training i/p & op pair perform step 2 – 4

step 2: $x_i = s_i$ (activation)

step 3: $y = t$ (activation)

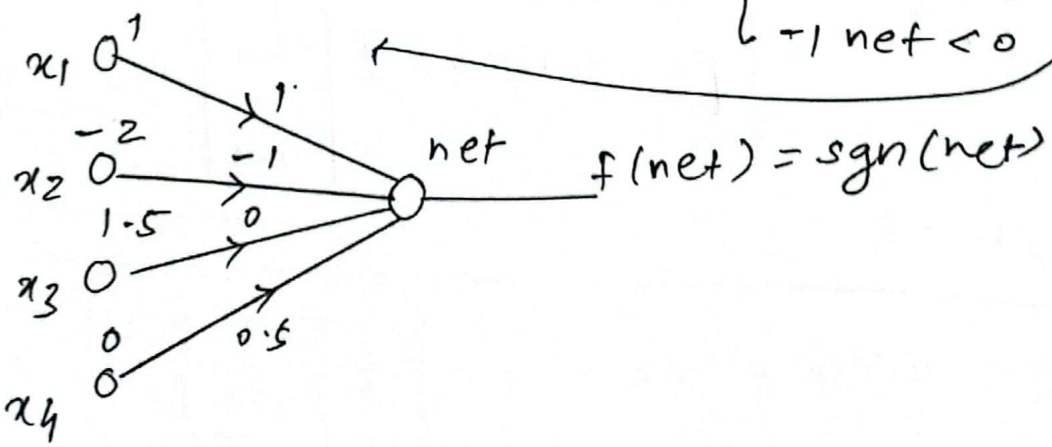
step 4 wt updation

$$w_i(new) = w_i(old) + x_i y$$
$$b(new) = b(old) + y$$

**Question:-** Implement the Hebbian training rule for a network using $f(net) = sgn(net)$ the three input vectors for training and the initial weight vector is given as follows

$$X_1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} \quad X_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} \quad X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix} \quad W' = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

$$f(net) = \begin{cases} 1 & net \geq 0 \\ -1 & net < 0 \end{cases}$$



## Iteration 1:

step 1: present i/p vector $x_1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}$

$net_1 = w_1^T x_1 = \begin{bmatrix} 1 & -1 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = 3$

$f(net_1) = sgn(net_1) = 1$ $\quad c = 1$

$\Delta w = c f(net_1) x_1 =$

$\Delta w = 1 \times \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}$

$w_2 = \Delta w_1 + w_1$

$w_2 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix}$

## Iteration 1 step 2:

present i/p vector $x_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix}$

$net_2 = w^{2T} x_2 = \begin{bmatrix} 2 & -3 & 1.5 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} = -0.25$

$f(net_2) = -1$

$\Delta w_2 = -1 \times \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \\ 2 \\ 1.5 \end{bmatrix}$

$w_3 = \Delta w_2 + w_2 = \begin{bmatrix} -1 \\ 0.5 \\ 2 \\ 1.5 \end{bmatrix} + \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -2.5 \\ 3.5 \\ 2 \end{bmatrix}$

Step 3 :

present i/p vector $x_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$

$net_3 = W_3^T x_3 = [1 \quad -2 \quad 3.5 \quad 2] \times \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$

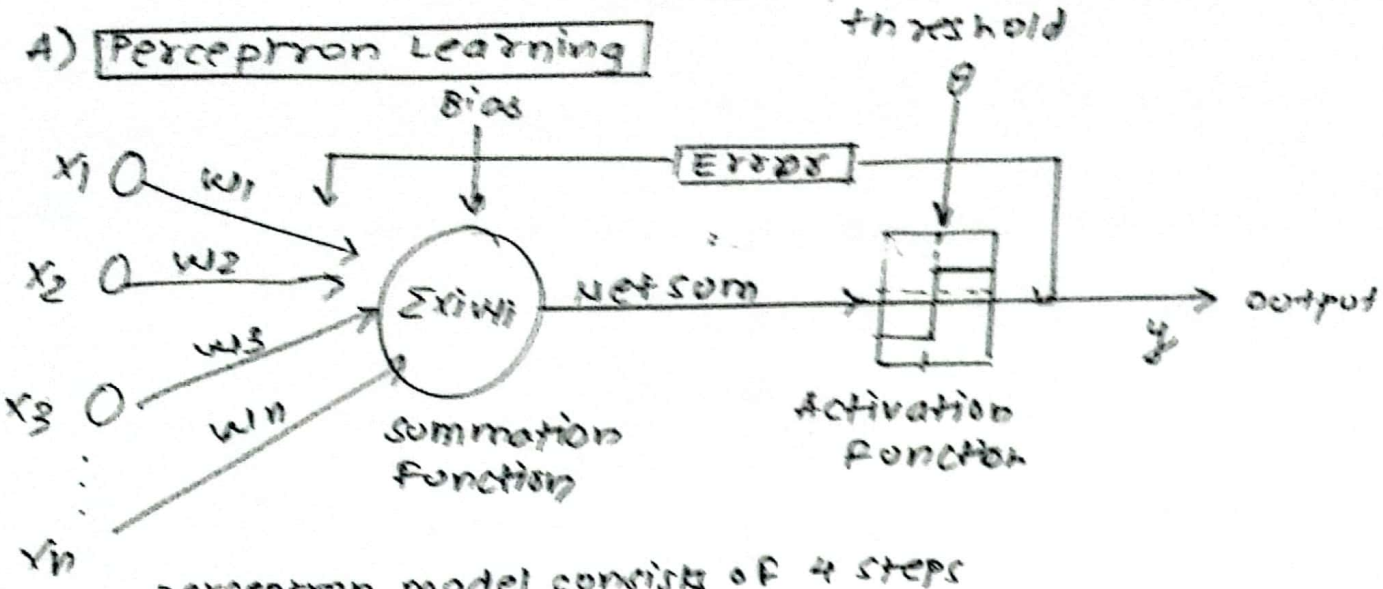
$f(net_3) =$

$\Delta W_3 = c f(net_3) x_3$

$W_4 = \Delta W_3 + W_3 =$

Neural Network Learning Rules:-

A) Perceptron Learning




perceptron model consists of 4 steps

1. Inputs from other neurons
2. Weights and bias
3. Net sum
4. Activation function

$$Y = \begin{cases} 1 & \text{if } f(x) \geq \theta \\ 0 & \text{if } f(x) < \theta \end{cases}$$

Algorithm – Perceptron Learning

Set Initial weights $w_1, w_2 \ldots w_n$ and Bias $\theta$ to a random value range in $[-0.5, 0.5]$.

For Each Epoch

1. compute the weighted sum by multiplying the inputs with weights and add the products
2. Apply Activation Function on weighted sum

$$Y = \text{step}((x_1 w_1 + x_2 w_2) - \theta)$$

3. If sum is above the threshold value, output the value as positive else output the value as negative

4. Calculate the error by subtracting the estimated output $Y_{estimated}$ from desired output

$$\text{error } e(t) = Y_{desired} - Y_{estimated}$$

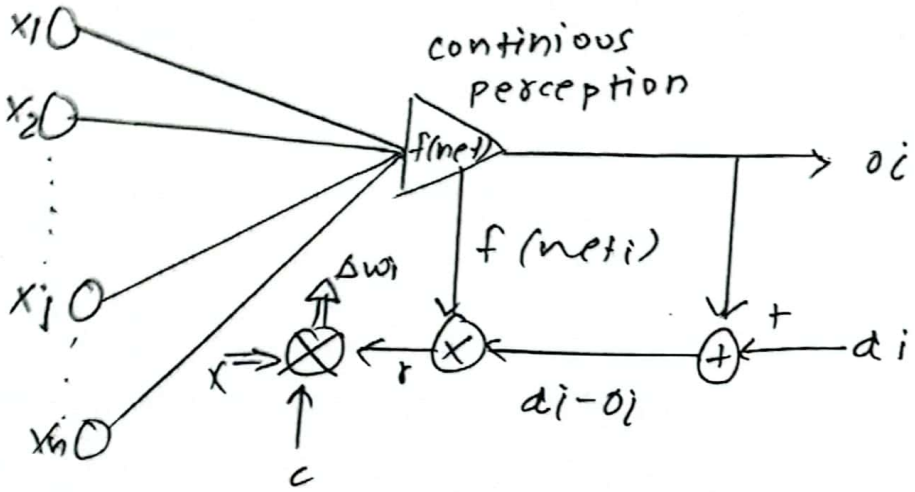
5. Update the weights if there is error

$$\Delta w_i = \alpha \times e(t) \times x_i$$

$$w_i = w_i + \Delta w_i$$

## Delta Learning Rule

(1) only valid for continious activation functions
(2) Learning signal for this rule is called delta
(3) Aim of delta role is to minimize error over all training patterns

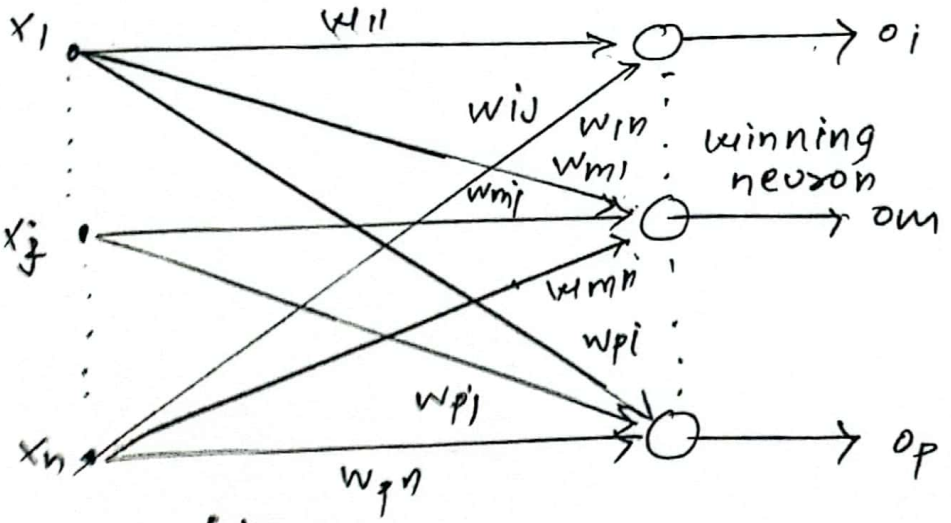



~~$\Delta w_{ij} = \eta \Delta$~~

$$\Delta w_j = -\eta \nabla E$$

$$\Delta w_i = \eta (d - o_i) f'(net_i) x$$

## Winner - Take All Learning Rules




Adjusted weights his

ron be explained for a layer of neurons

This neuron is declared the winner with a weight

$$W_m = [W_{m1} \; W_{m2} \cdots W_{mn}]^T$$

$$\Delta W_m = \alpha (x - W_m)$$

perceptron convergence Theorem

Rules: i) training data is linearly separable
ii) and $\eta$ is sufficiently small.

If two classes of vectors $x_1$ and $x_2$ are linearly seperable, the application of perceptron training algorithm will eventually result in a weight vector $w_0$, such that $w_0$ defines a TLU whose decision hyper-plane seperates $x_1$ and $x_2$

## Convergence of Learning algorithm

"suppose datasets $C_1, C_2$ are linearly seperable. The perceptron convergence algorithm converges no iterations with $n_0 \leq n_{max}$ on training set $C_1 \cup C_2$

# Module 2: Fuzzy Logic and Fuzzy interference system.

**Characteristic Function (C.F)** is also called Indicator function Let U be a universal set and be a subset of U.

$$\psi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

Example: $U = \{5, 7, 9, 10, 15\}$ and $A = \{5, 9, 15\}$

$\psi_A(5) = 1$ $\psi_B(7) = 0$ $\psi_A(9) = 1$, $\psi_A(10) = 0$ $\psi_A(15) = 1$

$\psi_x$ as 10101 characteristic functions

Example2

$U = \{1, 2, 3, 4, 5, 6\}$ $A = \{1, 2\}$ $B = \{2, 4, 6\}$

$C = \{1, 2, 3, 4, 5, 6\}$

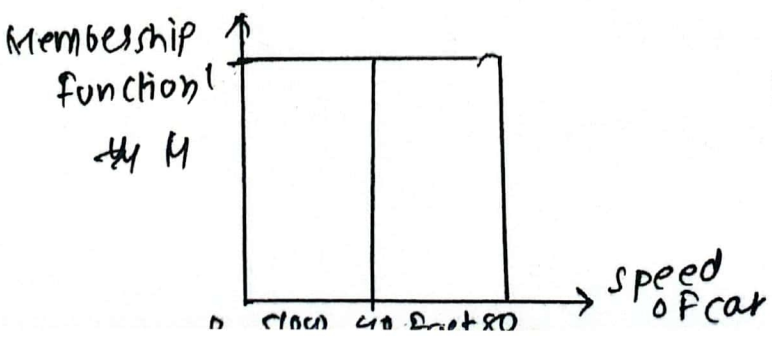
1. $\psi_A(x)$ : 110000
2. $\psi_B(x)$ : 010101
3. $\psi_C(x)$ : 111111

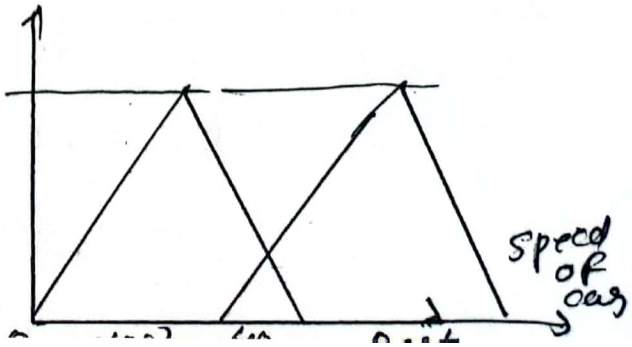
FUZZY LOGIC (Lotfi Zadeh)

→ Represent uncertainty [0,1]

→ Represent with degree [0,1]

→ Represent the belongings of a member of a crisp set to Fuzzy set



| Sl.no | Member | Age | Gender |
|---|---|---|---|
| 1 | Grand Paa | 72 | M |
| 2 | Grand Maa | 63 | F |
| 3 | Dad | 41 | M |
| 4 | Mom | 38 | F |
| 5 | Daughter | 15 | F |
| 6 | Son | 13 | M |
| 7 | Aunt | 52 | F |

$U = \{GP, GM, Dad, Mom, Dtr, Son, Aunt\}$

$M = \{GP, Dad, Son\}$

$F = \{GM, Mom, Dtr, Aunt\}$

$$\mu(x) = \begin{cases} 0 & x < 30 \\ (x-30)/40, & 30 \le x \le 70 \\ 1, & x \ge 70 \end{cases}$$

Fuzzy set(A) $= \{(GP, 1), (\frac{63-30}{40}; GM), (Mom, \frac{38-30}{40}), (Aunt, \frac{52-30}{40})\}$

Fuzzy set(A) $= \{(GP, 1), (GM, 0.825), (Mom, 0.275), (Aunt, 0.55)\}$

## Equal Fuzzy sets

Two fuzzy sets A(x) and B(x) are said to be equal, if $\mu_A(x) = \mu_B(x)$ for all $x \in X$. It is expressed as follows

$$A(x) = B(x) \quad \text{if } \mu_A(x) = \mu_B(x)$$

if $\mu A(x) = \mu_B(x)$

## complement of fuzzy set A(x)

Complement : The complement is the opposite of the set. The complement of fuzzy set is denoted $\bar{A}(x)$ or $A^c$ and is defined with respect to the universal set x as follows:

$$\bar{A}(x) = 1 - A(x) \text{ For all } x \in X$$

$$A^c = 1 - \mu_A(x)$$

Exp... $A = \{(1,2), (0.3,4), (0.5,6), (0.2,8)\}$

$B = \{(0.5,2), (0.4,4), (0.1,6), (1,8)\}$

$A^c = \{(0,2), (0.7,4), (0.5,6), (0.8,8)\}$

$B^c = \{(0.5,2), (0.6,4), (0.9,6), (0,8)\}$

## union of fuzzy sets

- Union of fuzzy sets consists of every elements that falls into either set. union is analogous to logical OR operation. The value of the membership value is will be the largest membership value of element in either set

$(A \cup B)(x)$

$$\mu(A \cup B)(x) = \max\{\mu A(x), \mu B(x)\}$$

$A = \{(1,2), (0.3,4), (0.5,6), (0.2,8)\}$

$B = \{(0.5,2), (0.4,4), (0.1,6), (1,8)\}$

$$A \cup B = \{\text{Max}(\mu_A(x), \mu_B(x)\} = \{(1,2), (0.4,4), (0.5,6), (1,8)\}$$

## Intersection of fuzzy set

Intersection of a fuzzy sets defines how much of the elements belongs to both sets.

$$\mu(A \cap B)(x) = \min\{\mu A(x), \mu B(x)\}$$

Let $A = \{(1,2), (0.3,4), (0.5,6), (0.2,8)\}$

$B = \{(0.5,2), (0.4,4), (0.1,6), (1,8)\}$

$A \cap B = \{\text{Min}(\mu_A(x), \mu B(x)\} = \{(0.5,2), (0.3,4), (0.1,6), (0.2,8)\}$

Algebraic sum:- can be calculated as:-

$$\boxed{\mu_{A+B}(x) = [(\mu_A(x) + \mu_A(x)] - [\mu_A(x) \times \mu_A(x)]}$$

$A = \{(x_1, 1), (x_2, 0.5), (x_3, 0.3), (x_4, 0.4)\}$

$B = \{(x_1, 0.2), (x_2, 0.2), (x_3, 0.7), (x_4, 0.1)\}$

$\mu_{A+B}(x) = [(\mu_A(x) + \mu_A(x)] - [\mu_A(x) \cdot \mu_A(x)]$

$\mu_{A+B}(x) = [1.2 - 0.2 = \underline{1.0}; 0.7 - 0.1 = \underline{0.6}; 1.0 - 0.21 = \underline{0.79};$
$0.5 - 0.04 = 0.46]$

$\mu_{A+B}(x) = \{(x_1, 1), (x_2, 0.6), (x_3, 0.79), (x_4, 0.46)\}$

difference

Exp : $A = \{(1,2), (0.3,4), (0.5,6), (0.2,8)\}$

$B = \{(0.5,2), (0.4,4), (0.1,6), (1,8)\}$

$A - B = A \cap B^c = \{(0.5,2), (0.3,4),$

## Algebraic Product (vector product)

The 'Algebraic product' of two fuzzy sets denoted by "$A \cap B$" or "$A \cdot B$", is a fuzzy set whose the membership function at each point is calculated by simply multiplying the membership values of corresp. points

$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x)$ is called dot product

$A = \{(x_1, 0.2), (x_2, 0.4), (x_3, 0.3)\}$

$B = \{(x_1, 0.3), (x_2, 0.2), (x_3, 0.5)\}$

Find $A \cdot B$ and $0.2A$

$A \cdot B = \mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x) = \{(x_1, 0.06), (x_2, 0.08), (x_3, 0.15)\}$

Scalar product $\alpha A(x) = \alpha \mu_A(x)$

$0.2A = \{(x_1, 0.04), (x_2, 0.08, (x_3, 0.06)$

## Fuzzy set operations: cartesian product

Cartesian Product (x product) can be represented as A-B and can be calculated as

$$\mu_{A-B}(x, y) = \text{Min}(\mu_A(x), \mu_B(y))$$

$A = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5)\}$ and

$B = \{(y_1, 0.3), (y_2, 0.2), (y_3, 0.5)\}$

$A \times B =$

| | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|
| $x_1$ | 0.2 | 0.2 | 0.2 |
| $x_2$ | 0.3 | 0.2 | 0.3 |
| $x_3$ | 0.5 | 0.2 | 0.5 |

## Bounded sum

The Bounded sum of two Fuzzy sets $A(x)$ and $B(x)$ for all $x \in X$ is denoted by $A(x) \oplus B(x)$ are defined as

$$A(x) \oplus B(x) = \{(x, \mu_{A \oplus B}(x)), x \in X\}$$

where $\mu_{A \oplus B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$

$$A(x) = \{(x_1, 0.1), (x_2, 0.2), (x_3, 0.3), (x_4, 0.4)\}$$
$$B(x) = \{(x_1, 0.5), (x_2, 0.7), (x_3, 0.8), (x_4, 0.9)\}$$

$$A(x) \oplus B(x) = \{(x_1, 0.6), (x_2, 0.9), (x_3, 1.0), (x_4, 1.0)\}$$

## Bounded difference

$$A(x) \ominus B(x) = \{(x, \mu_{A \ominus B}(x)), x \in X\}$$

where $\mu_{A \ominus B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$

Example :-

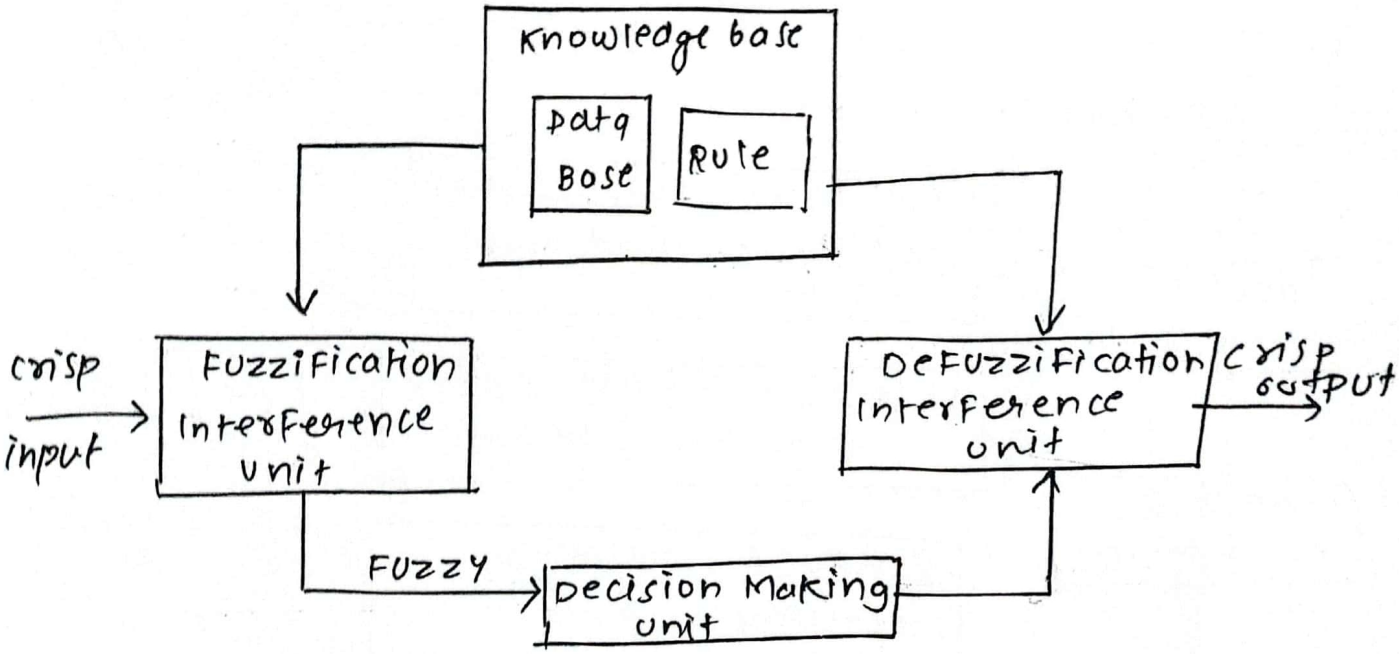
$$A(x) = \{(x_1, 0.1), (x_2, 0.2), (x_3, 0.3), (x_4, 0.4)\}$$
$$B(x) = \{(x_1, 0.5), (x_2, 0.7), (x_3, 0.8), (x_4, 0.9)\}$$

$$A(x) \ominus B(x) = \{(x_1, 0), (x_2, 0), (x_3, 0.1), (x_4, 0.3)\}$$

# Fuzzy Logic and Fuzzy Interference System

Fuzzy interference (FIS) is the key unit of Fuzzy logic system having decision making as its primary work. It uses "IF THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

BLOCK DIAGRAM OF FUZZY INTERFERENCE SYSTEM




Rule base :- It contains Fuzzy IF-THEN rules

Database : It defines the membership function of Fuzzy sets used in Fuzzy rules

Decision-making unit : It performs operation on rules

Fuzzification Interface unit : It converts the crisp quantities into Fuzzy quantities

Defuzzification Interface unit : It converts the Fuzzy quantities into Crisp quantities

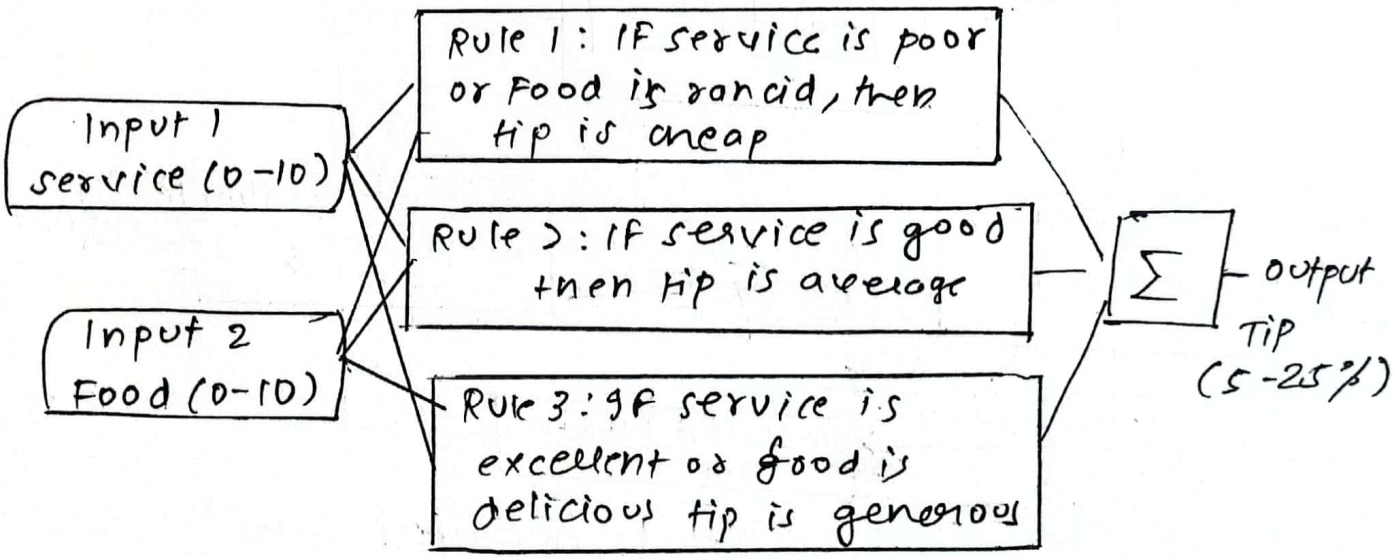
The working of FIS :-

1) A fuzzification unit supports the application of numerous fuzzification methods and converts into crisp input into fuzzy inputs.

2) A knowledge base :- collection of rule base and database is formed upon the conversion of crisp input into fuzzy input

3) Defuzzification unit fuzzy input is finally converted into crisp o/p.

For Example :-




Mamdani systems.

if $x_1$ is $A_1$ and $x_2$ is $A_2$ THEN $y$ is $B$

if $x_1$ is $A^k$ and $x_2$ is $A_2^k$, then $y^k$ is $B^k$

we will be considering a 2 input mamdani systems :-

Two cases for 2-input Mamdani systems

1. Max-min inference Method
2. Max-product Inference Method

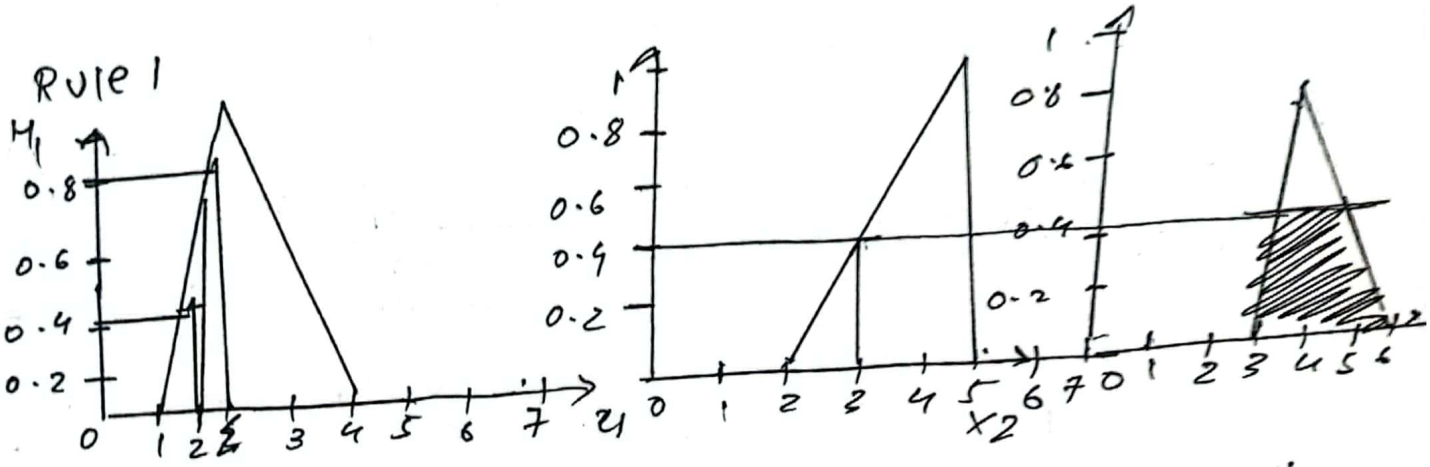
## Max-min inference Method

Rule 1: If $x_1$ is $A_1'$ and $x_2$ is $A_2'$ THEN y' is B'

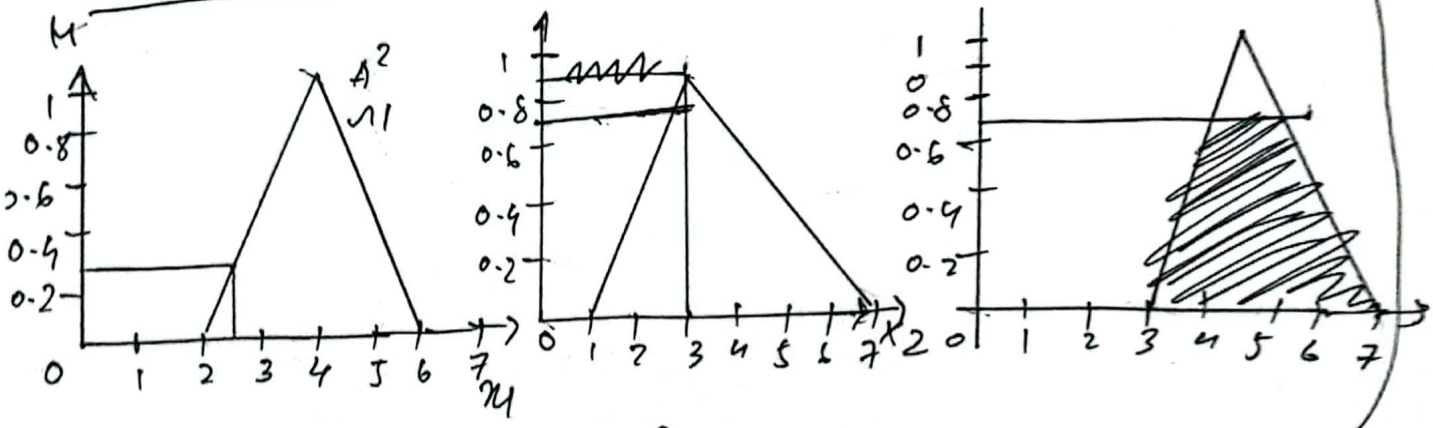
Rule 2: If $x_1$ is $A_1^2$ or $x_2$ is $A^2$ then $y^2$ is $B^2$

$x_1 = 2.5 \quad x_2 = 3$

Rule 1




Rule 2




For product inference we