1/8/23

"Soft Computing Techniques"

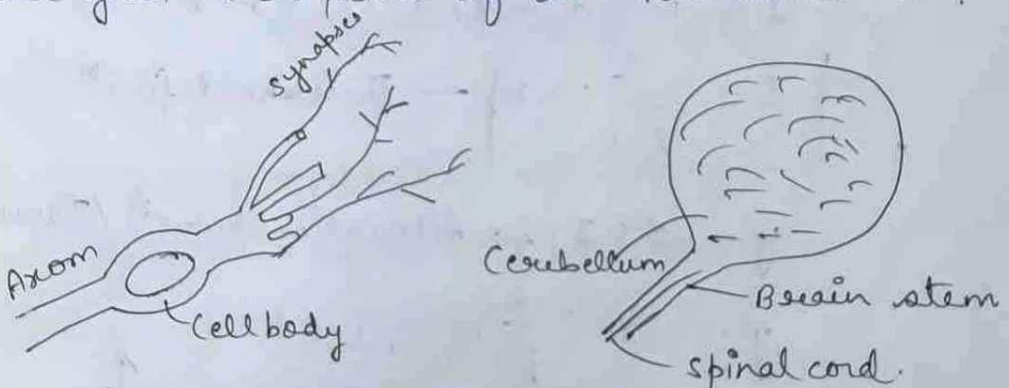
Neural Networks :-

These networks are related to human brain. They work in a similar way, human brain works.

Neural networks which are simplified models of the biological neuron system is a massively parallel distributed processing system made up of highly interconnected neural computing elements that have ability to learn and thereby acquire knowledge and make it available for the use.

Artificial neural networks :-

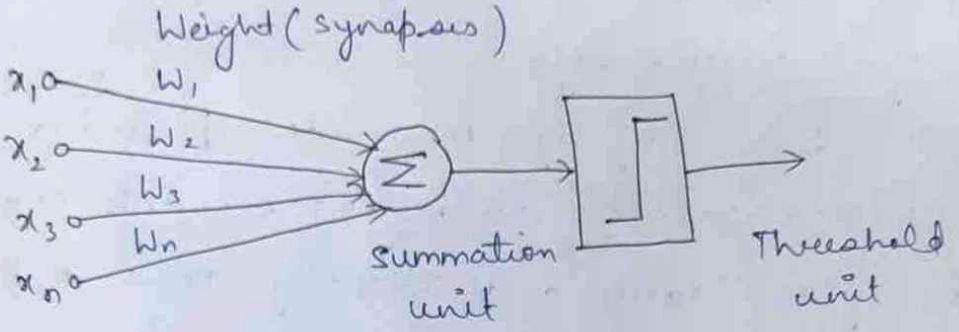
These are just a replica of our human brain.



10^{10} basic units are there in our brain i.e neurons. and 10^{4} neurons are connected to them. Dendrites are also connected to the other neurons. Axom will give the output. Synapses are used to accelerate or deaccelerate the quantity.

In artificial neural networks, we do the same thing. We provide inputs, take sum and get the output.

Model of artificial neural networks :



Higher the weight, higher will be the acceleration. otherwise it will be deacceleration.

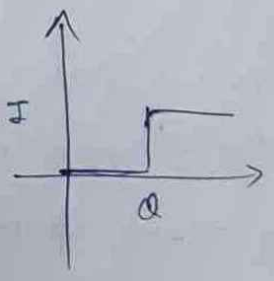
$$I = w_1 x_1 + w_2 x_2 + \cdots w_n x_n$$

$$\boxed{I = \sum_{i=1}^{n} w_i x_i} \text{ — Threshold func}^n.$$

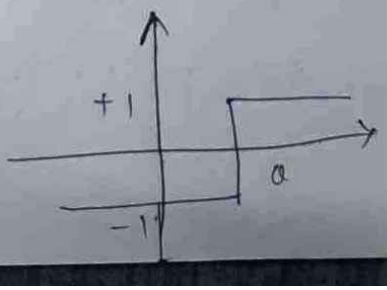
$y = \phi(I) \rightarrow$ Activation func^n / Transfer func^n

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

$$\phi(I) = 1 \quad I \geq 0$$
$$= 0 \quad I \leq 0$$



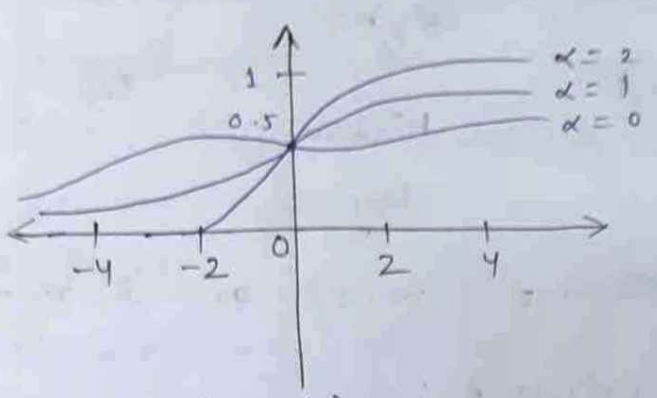
Signum function :-



$$\phi(I) = +1 \quad I > 0$$
$$\phi(I) = -1 \quad I \leq 0$$

Sigmoidal function :-

$$\phi(I) = \frac{1}{1 + e^{-\alpha I}}$$



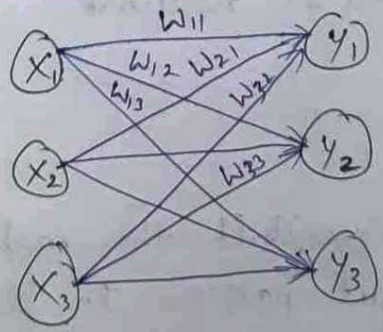
Hyperbolic tangent function :-

$$\phi(I) = \tanh(Z)$$

Architecture :- The way neurons are interconnected to each other is called architecture.

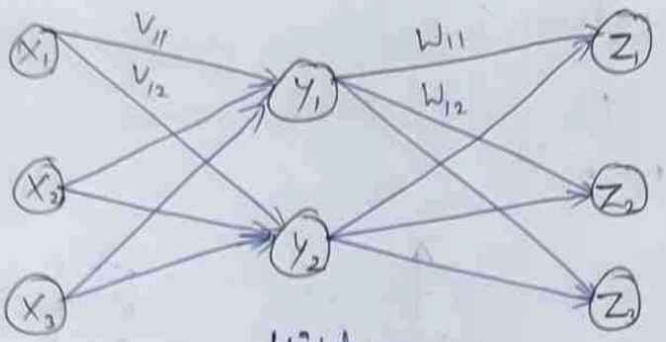
Neural Network Architecture :-

(i) Single layer feed forward network
(ii) Multilayer feed forward network.
(iii) Recurrent network.



Input layer

Output layer



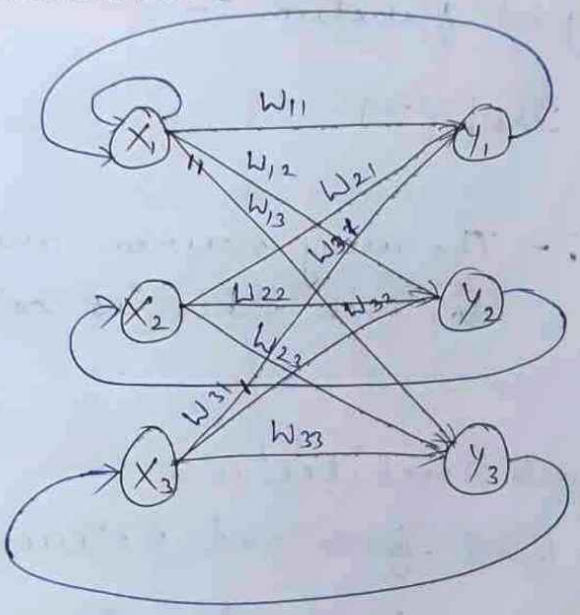
Input layer

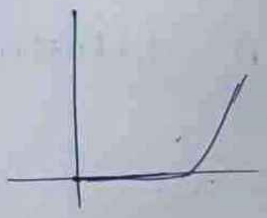
Output layer

Hidden layer

architecture = $l-m-n$ or $l-m_1-m_2-n_1-n_2$

Recurrent network :-





~~Learning methods:~~

Characterization of neural network :-

(i) Mapping

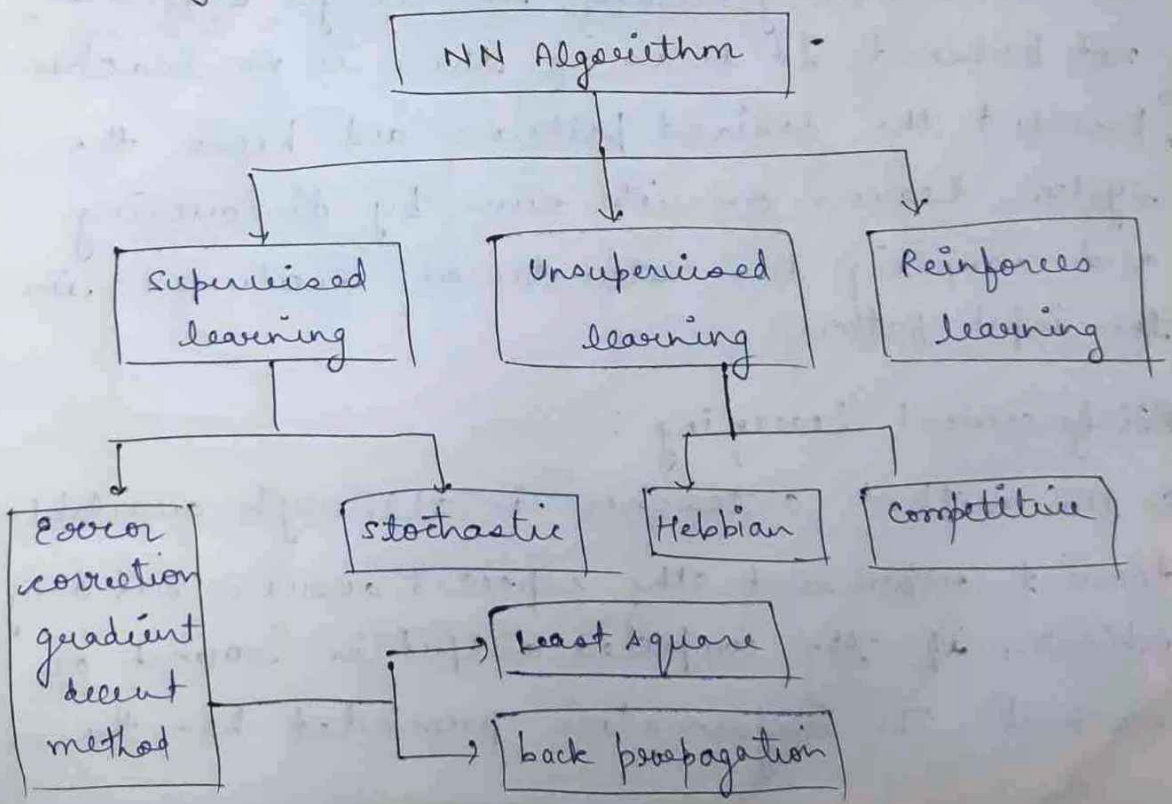
(ii) Robustness

(iii) • Neural networks exhibit the mapping capability. They can map input pattern to their associated

output patterns.

- Neural networks learnt by examples. Neural network architectures can be trained with non example of problems before they tested for their unknown instances of the problem.
- Neural network have the capability to generalize. They can predict new outcomes from the past traits trend.
- Neural networks are ~~gle~~ robust systems and are fault called tolerant. They can recall full pattern from incomplete or partial or noisy patterns.

Learning Methods :-



Supervised learning :-

In this, every input pattern that is used to train the network is associated with an output pattern which is the target or the desired pattern. A teacher is assumed to be present during the learning process. When a comparison is made between the networks computed output and the correct expected output to determine the error, the error can be used to change the network parameter which results in an improved performance.

Unsupervised learning :-

In this learning method, the target output is not present. It is as if there is no Teacher to present the desired patterns and hence the system learns on its own by discovering and adapting two structural features in the input pattern.

Reinforcement learning :

In this method, a teacher is although available doesn't represent the expected answer but only indicates if the computed output is correct or incorrect. The information provided has the

the network in its learning process. A reward is given for a correct answer computed and a penalty for a wrong answer.

Hebbian rule :- (Hebb-1949)

$$W = \sum_{i=1}^{n} x_i y_i^T$$

W: Correlation weight matrix

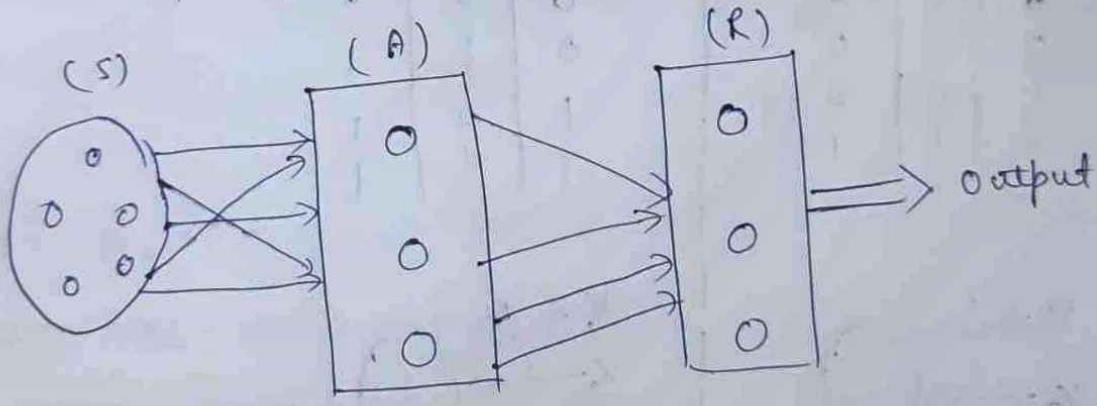
Gradient decent learning :-

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}}$$

error gradient wrt weight

η = learning rate parameter

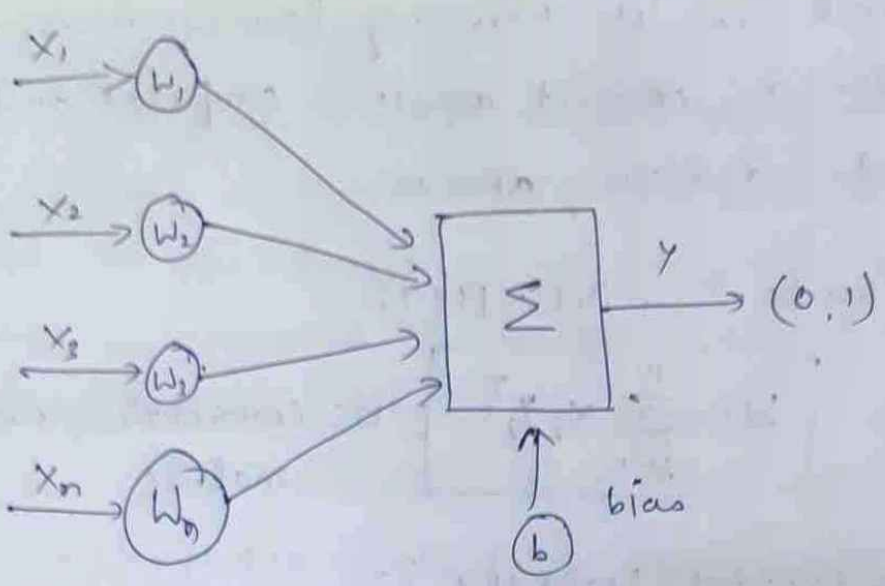
Rosenblat's perceptron model :-



S - Sensory unit

A - Association unit

R - Response unit



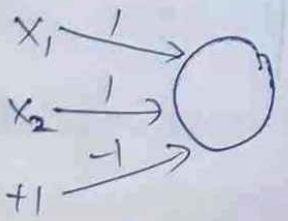
$$f\phi(x) = \begin{cases} 1 & \text{if bitwise} > 0 \\ 0 & \text{otherwise} \end{cases}$$

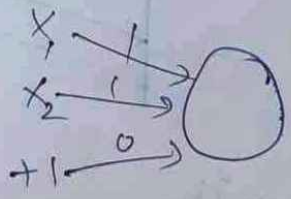
AND

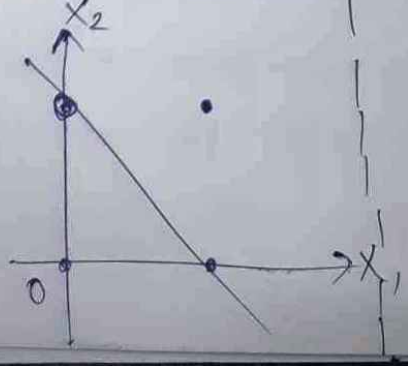
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

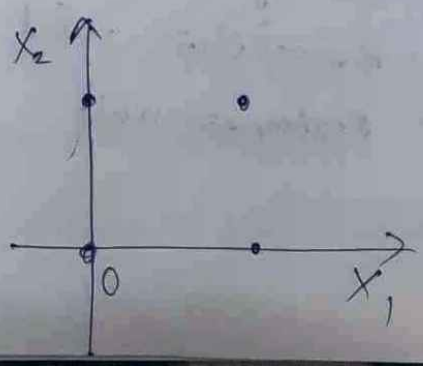
OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1







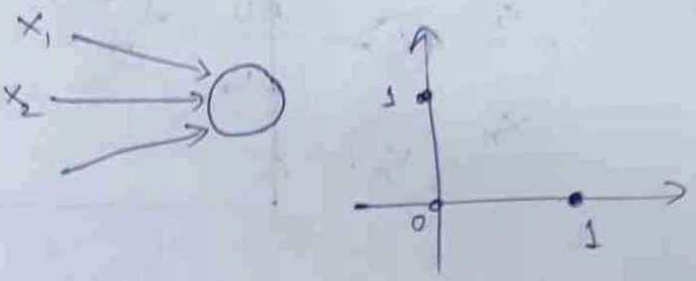


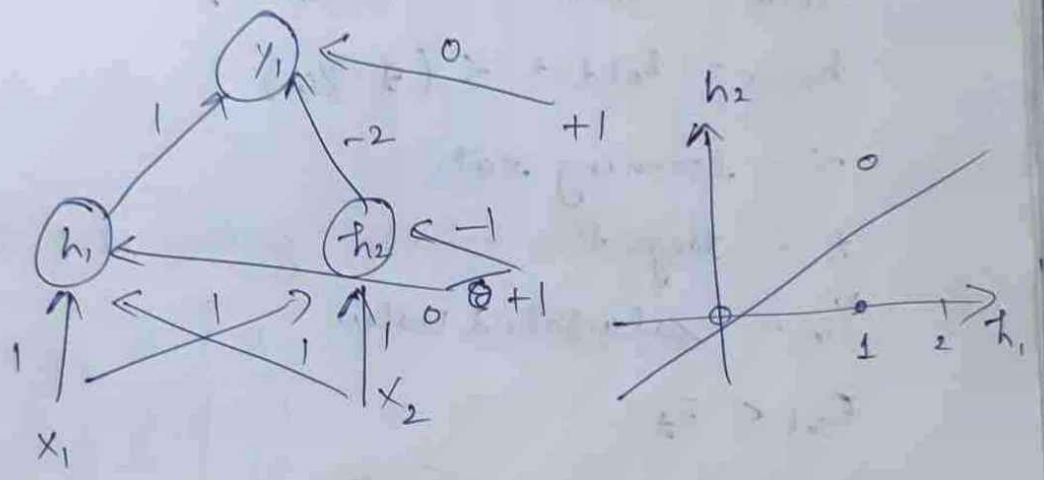
$w_1 x_1 + w_2 x_2 + b = 0$

$x_2 = -\frac{b}{w_2} - \frac{w_1}{w_2} x_1$

XOR:-

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0





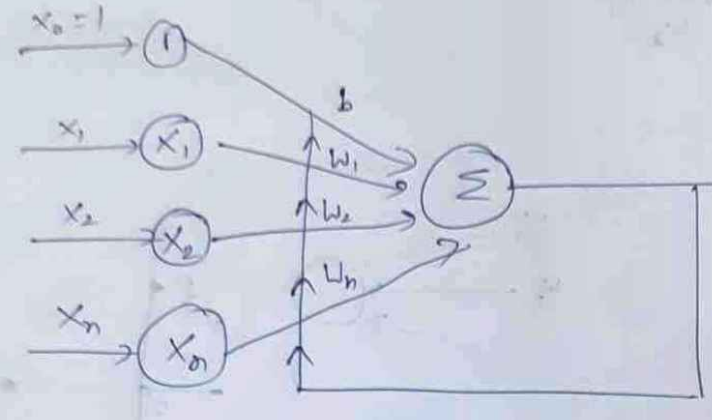
$y_1 = h_1 - 2h_2 + 0 =$

$y_1 = x_1 + x_2 - 2(x_1 + x_2 - 1) + 0$

$y_1 = -x_1 - x_2 + 0 + 2$

$y_1 = -x_1 - x_2 + 2$

ADALINE (Adaptive linear neural element network) :-



$W_{inew} = W_{iold} + \alpha (t - y_{in}) X_i$

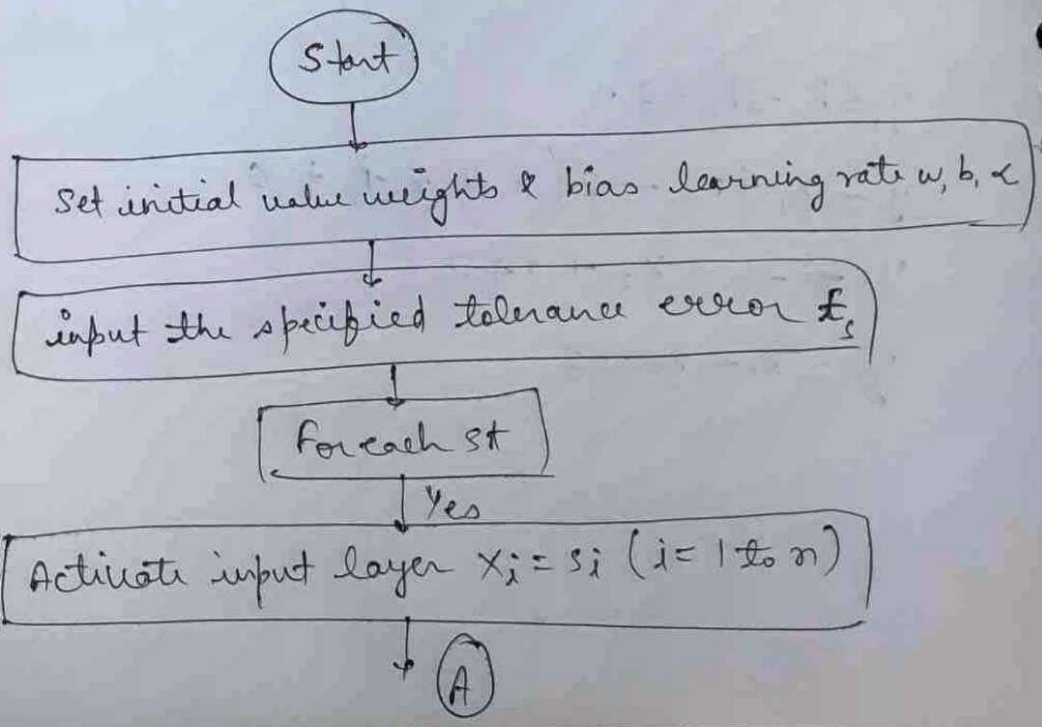
$b_{new} = b_{old} + \alpha (t - y_{in})$

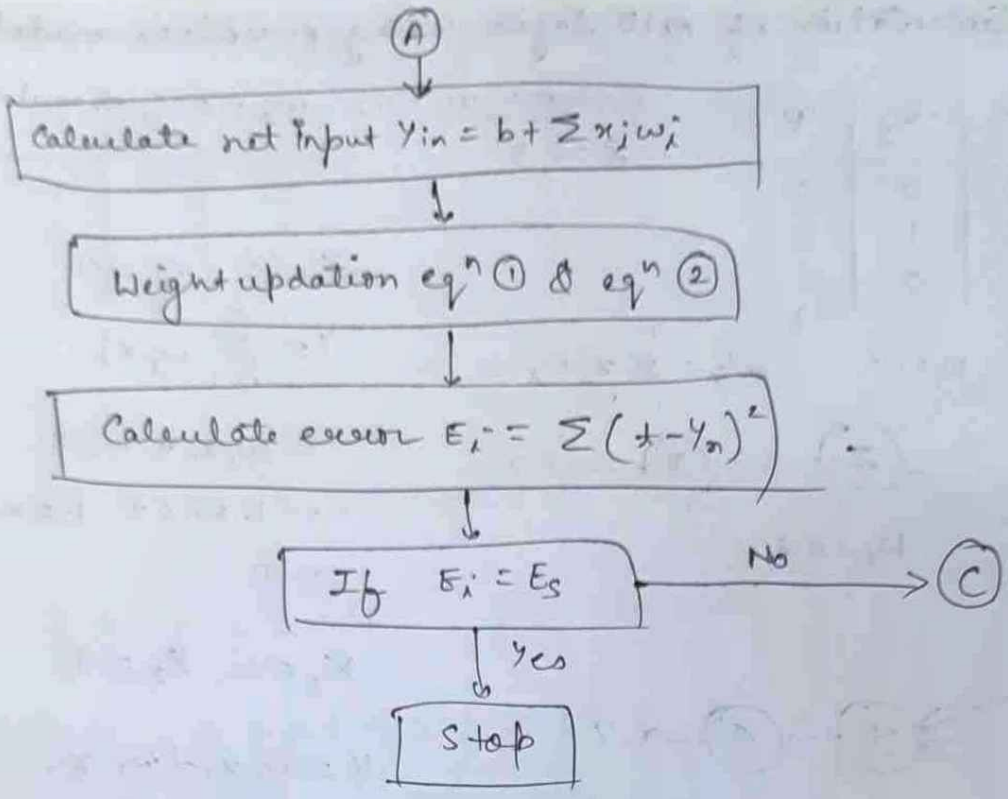
α - learning rate

t - tangent

y_{in} - calculated value

$E_{cal} < E_t$





Design OR gate using Adaline network :-

x_1	x_2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	1

$E_s = 2$

$\alpha = 0.1$

$W_1 = W_2 = 0.1$

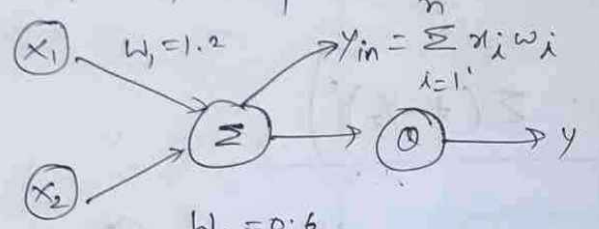
Assign 1 : Design OR & AND gate using perceptron model. Design OR gate using Adaline network

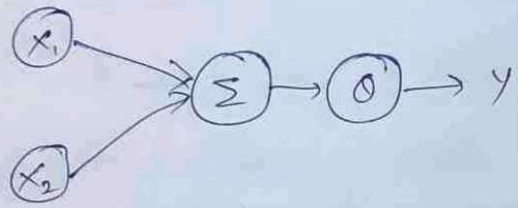
Implementation of AND logic using perceptron model :-

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Assume $w_1 = 1.2$, $w_2 = 0.6$, threshold = 1

$\alpha = 0.5$





$x_1 = 0, x_2 = 0$

$y = \sum_{i=1}^{2} w_i x_i$

$= w_1 x_1 + w_2 x_2$

$= 1.2 \times 0 + 1.2 \times 0$

$= 0$

$x_1 = 0, x_2 = 1$

$y = w_1 x_1 + w_2 x_2$

$= 1.2 \times 0 + 0.6 \times 1$

$y = 0.6$

$x_1 = 1, x_2 = 0$

$y = 1 \times 1.2 + 0 \times 0.6$

$= 1.2 > 1$

∵ it is greater than 1 so the ans will be 1

$x_1 = 1, x_2 = 1$

$y = 1 \times 1.2 + 1 \times 0.6$

$= 1.2 + 0.6$

$y = 1.8 > 1$

$W_{i\,new} = W_{i\,old} + \alpha(t - y)x_i$

$= 1.2 + 0.5(0 - 1)1 = 0.7$

$W_{2\,new} = 0.6 + 0.5(1 - 1)0$

$= 0.6$

$= 0.7 + 0.6$

$= 1.3$

Implementation of OR gate using perceptron model :-

x_1	x_2	$y(*)$
0	0	0
0	1	1
1	0	1
1	1	1

Assume $w_1 = 0.6$, $w_2 = 0.6$, thr $= 1$

$\alpha = 0.5$

$y = w_1 x_1 + w_2 x_2$

$y = 0.6 \times 0 + 0.6 \times 0 = 0$. ✓

$y = 0.6 \times 0 + 0.6 \times 1 = 0.6 < 0$

$y = 0.6 \times 1 + 0 = 0.6 < 0$.

$y = 0.6 + 0.6 = 1.2 > 1$ ✓

$w_{1new} = 0.6 + 0.5 (1-0) 0 = 0.6$ updated

$w_{2new} = 0.6 + 0.5 (1-0) 1 = 1.1$

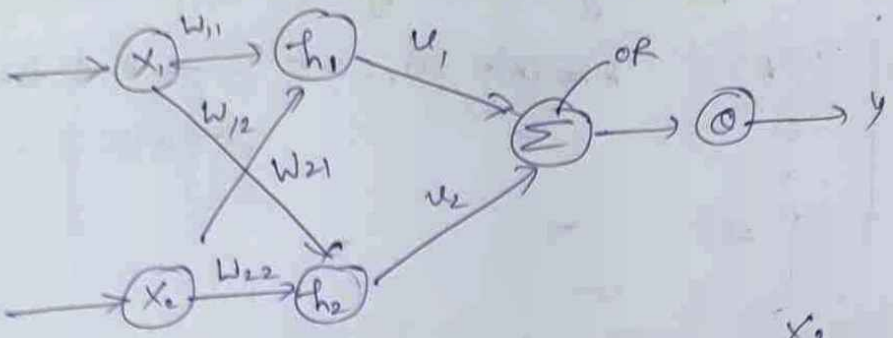
$y = 0.6 \times 0 + 1.1 \times 1 = 1.1$ ✓.

$y = 0.6 \times 1 + 0 \times 1.1 = 0.6 <$ threshold

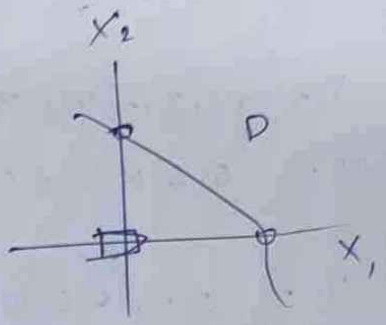
again do updation of weight

$w_1 = 0.6 + 0.5 (1-0) 1 = 1.1$

$w_2 = 1.1 + 0.5 (1-0) 0 = 1.1$

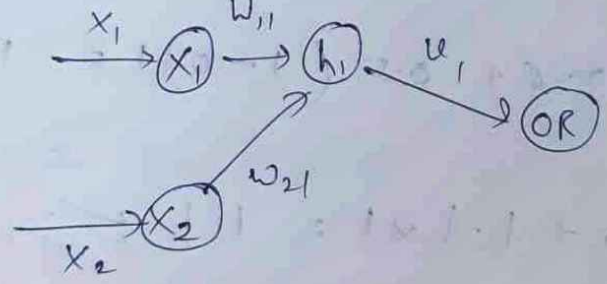


$$y = \underbrace{X_1\overline{X_2}}_{h_1} + \underbrace{\overline{X_1}X_2}_{h_2}$$



X_1	X_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$w_{11} = 1$, $w_{21} = 1$ $\alpha = 1.5$



$y = w_1x_1 + w_2x_2$

$= 1 \times 0 + 1 \times 0 = 0$

$y = 1 \times 0 + 0 \times 1 = 0$

$y = 1 \times 1 + 0 \times 1 = 1$

$y = 1 \times 1 + 1 \times 1 = 2 > 0.$

$w_{1new} = 1 + 1.5(1 - 0) 1$

$= 1 + 1.5(-1) = -0.5$

$w_{2new} = 1 + 1.5(0 - 1) 1$

$= 1 - 1.5 =$

$w_{1new} = 1 + 1.5\ (1 - 0) = 2.5$

$w_{2new} = 1 + 1.5\ ($

$w_{11} = 1,\ w_{21} = -0.5$ (updated)

$h_2 = \bar{x}_1 x_2\ ,\ w_{22} = 1,\ w_{12} = 1$

x_1	x_2	h_2
0	0	0
0	1	1
1	0	0
1	1	0

$y = w_1 x_1 + w_2 x_2 = 1 \times 0 + 1 \times 0 = 0.$

$= 1 \times 0 + 1 \times 1 = 1$

$= 1 \times 1 + 1 \times 0 = 1 > 0.$ X

$W_{1new} = 1 + 1.5(0-1)1$

$W_{1new} = 1 + 1.5 = -0.5$

$W_{2new} = 1 + 1.5(0-1)0$

$= 1$

$y = h_1 v_1 + h_2 v_2 \quad , v_1 = 1, v_2 = 1$

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	1	1	0

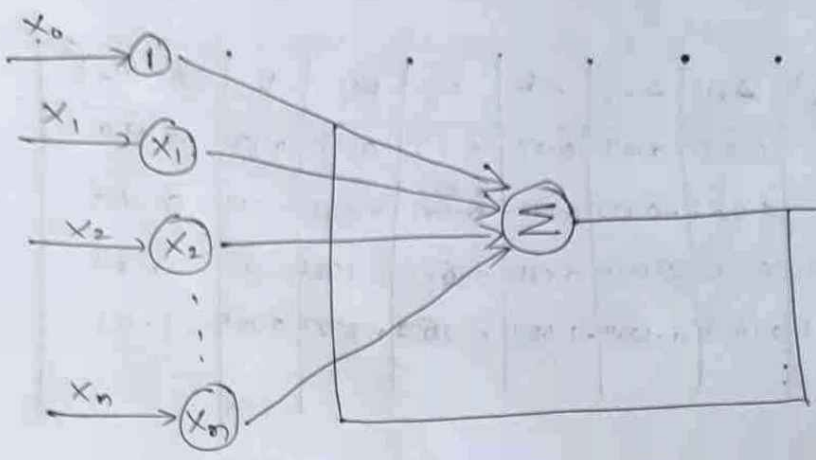
$y_1 = 0$ ✓

$y_2 = 1$ ✓

$y_3 = 1$ ✓

$y_4 = 0$ ✓

OR, AND & XOR using ADALINE :-



$$W_{inew} = W_{iold} + \alpha (t - y_{in}) x_i$$

$$b = b_{old} + \alpha (t - y_n)$$

(i) OR gate :

x_1	x_2	t
1	1	1
1	−1	1
−1	1	1
−1	−1	−1

all weight value = 0.1, $\alpha = 0.1$, $b = 0.1$

↓ learning rate

$E_{set} = 2$

↓ least square error

first input

$x_1 = 1$, $x_2 = 1$

$y_{in} = b + x_1 w_1 + x_2 w_2 = 0.1 + 0.1 \times 1 + 0.1 \times 1 = 0.3$

$(t - y_{in}) = 1 - 0.3 = 0.7$

updation of weight & bias :-

$W_{inew} = W_{iold} + \Delta w_i$

$W_{1new} = 0.1 + 0.1 (1 - 0.3) 1 = 0.17$

$W_{2new} = 0.1 + 0.1 (1 - 0.3) 1 = 0.17$

$b_{new} = 0.1 + 0.1 (1 - 0.3) = 0.17$

Least sq. error $= (0.7)^2 = 0.49$

x_1	x_2	t	y_{in}	$(t - y_{in})$	Δw_1	Δw_2	Δb	w_1	w_2	b	$(t - y_{in})^2$
1	1	1	0.3	0.7	0.07	0.07	0.07	0.17	0.17	0.17	0.49
1	-1	1	0.17	0.83	0.083	-0.083	0.083	0.253	0.087	0.253	0.69
-1	1	1	0.087	0.913	-0.0913	0.0913	0.0913	0.1683	0.1783	0.3443	0.83
1	1	1	0.0443	1.0093	0.1009	0.1004	1.009	0.2692	0.2727	0.2937	1.01

Second input

$x_1 = 1$, $x_2 = -1$

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

$= 0.17 + 1 \times 0.17 + (-1) 0.17$

$= 0.2 + 0.1$

$y_{in} = 0.1$

$t - y_{in} = 1 - 0.1 = 0.9$

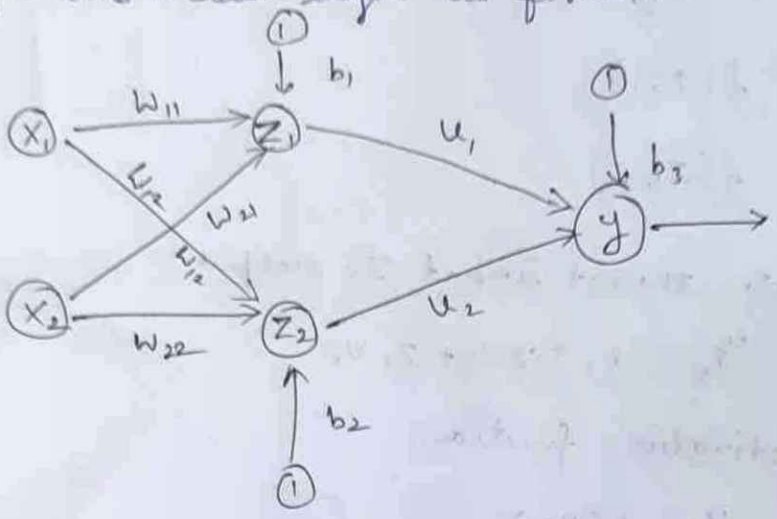
updation of weight & bias

$w_{1new} = 0.17 +$

Madaline (Multiple adaptive linear network) :-

- A madaline model consists of many adaline in parallel with a single output unit.
- The adaline layer is present between the input layer & the madaline layer. Here as adaline layer is a hidden layer.

- The weights between the input layer & the hidden layer are adjustable and the weight between the output layer and hidden layer is fixed.



Algorithm :-

Step 1: Initialize weight and set learning rate α :

for eg. $V_1 = V_2 = 0.5$, $b = 0.5$, $W_{11} = W_{12} = W_{21} = W_{22} = 0.1$

Step 2: While the stopping condition is false. do step 3 to 9.

Step 3: For each training set perform step 4 to 8.

Step 4: set activation of input unit $X_i = s_i$ for $i = 1$ to n.

Step 5: Compute net input to Adaline unit.

$$Z_{in1} = b_1 + X_1 W_{11} + X_2 W_{21}$$

$$Z_{in2} = b_2 + X_1 W_{12} + X_2 W_{22}$$

Step 6: For output of Adaline unit using activation function.

$$f(z) = 1 \quad \text{if } z \geq 0$$
$$= -1 \quad \text{if } z < 0$$

$$z_1 = f(z_{in1})$$

$$z_2 = f(z_{in2})$$

Step 7: Calculate the net input to output.

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$

Apply activation function

$$y = f(y_{in})$$

Step 8: Find the error & weight updation.

if $t \neq y$ then weight updation

if $t = y$ then no weight updation.

$$w_{inew} = w_{iold} + \alpha (t - z_{in1}) x_1$$

$$b_{inew} = b_{iold} + \alpha (t - z_{in1})$$

Step 9: Test the stopping condition for all approaches.

Q: Using Madaline network, implement XOR function with bipolar activation function.

$$[w_{11} \quad w_{21} \quad b_1] = [0.05 \quad 0.2 \quad 0.3] \qquad \alpha = 0.5$$

$$[w_{12} \quad w_{22} \quad b_2] = [0.1 \quad 0.2 \quad 0.15]$$

$$[v_1 \quad v_2 \quad b_3] = [0.5 \quad 0.5 \quad 0.5]$$

$$f(z) = 1 \quad \text{if } z \geq 0$$
$$= -1 \quad \text{if } z < 0$$

$$z_1 = f(z_{in_1})$$

$$z_2 = f(z_{in_2})$$

Step 7: Calculate the net input to output.

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$

Apply activation function

$$y = f(y_{in})$$

Step 8: Find the error & weight updation.

if $t \neq y$ then weight updation

if $t = y$ then no weight updation

$$w_{inew} = w_{iold} + \alpha (t - z_{in_1}) x_1$$

$$b_{inew} = b_{iold} + \alpha (t - z_{in_1})$$

Step 9: Test the stopping condition for all approaches.

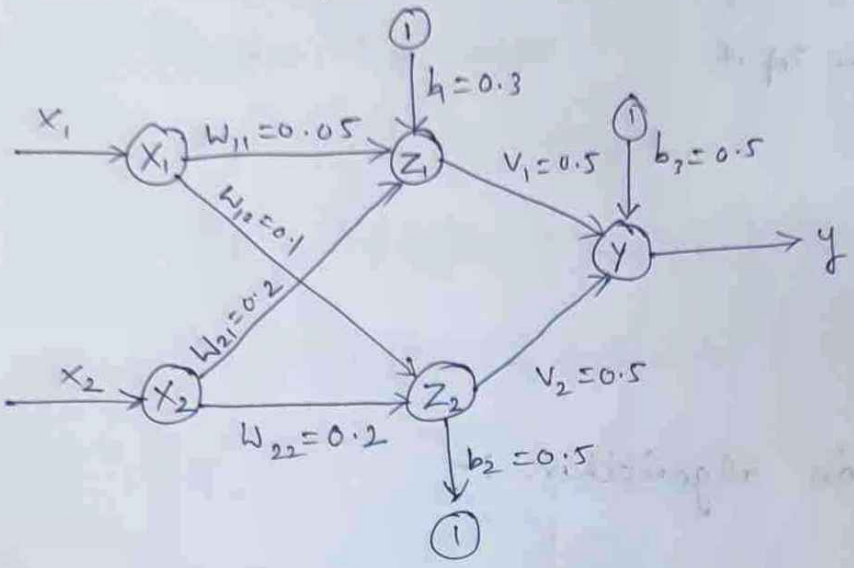
Q: Using Madaline network, implement XOR function with bipolar activation function.

$$[w_{11} \quad w_{21} \quad b_1] = [0.05 \quad 0.2 \quad 0.3] \qquad \alpha = 0.5$$

$$[w_{12} \quad w_{22} \quad b_2] = [0.1 \quad 0.2 \quad 0.15]$$

$$[v_1 \quad v_2 \quad b_3] = [0.5 \quad 0.5 \quad 0.5]$$

x_1	x_2	t
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1



first input: $x_1 = 1$, $x_2 = 1$, $t = -1$

$$Z_{in1} = x_1 w_{11} + x_2 w_{21} + b_1 = 1 \times 0.05 + 1 \times 0.2 + 0.3 = 0.55$$

$$z_{in2} = x_1 w_{12} + x_2 w_{22} + b_2 = 1 \times 0.1 + 1 \times 0.2 + 0.15 = 0.45$$

$$f(Z_{in}) = \begin{cases} +1 & \text{if } Z_{in} \geq 0 \\ -1 & \text{if } Z_{in} < 0 \end{cases} ; \quad Z_1 = f(Z_{in1}) = 1$$

$$Z_2 = f(Z_{in2}) = 1$$

$$Y_{in} = Z_1 * V_1 + Z_2 * V_2 + b_3$$

$$= 1 \times 0.5 + 1 \times 0.5 + 0.5 = 1.5$$

Activation function $= f(Y_{in}) = 1$

$W_{inew} = W_{iold} + \alpha (t - Z_{in})$

$b_{jnew} = b_{jold} + \alpha (t - Z_{in})$

$W_{11} = -0.725$, $W_{12} = -0.625$, ~~$b_{1new} = -0.475$~~ $b_{1NEW} = -0.475$

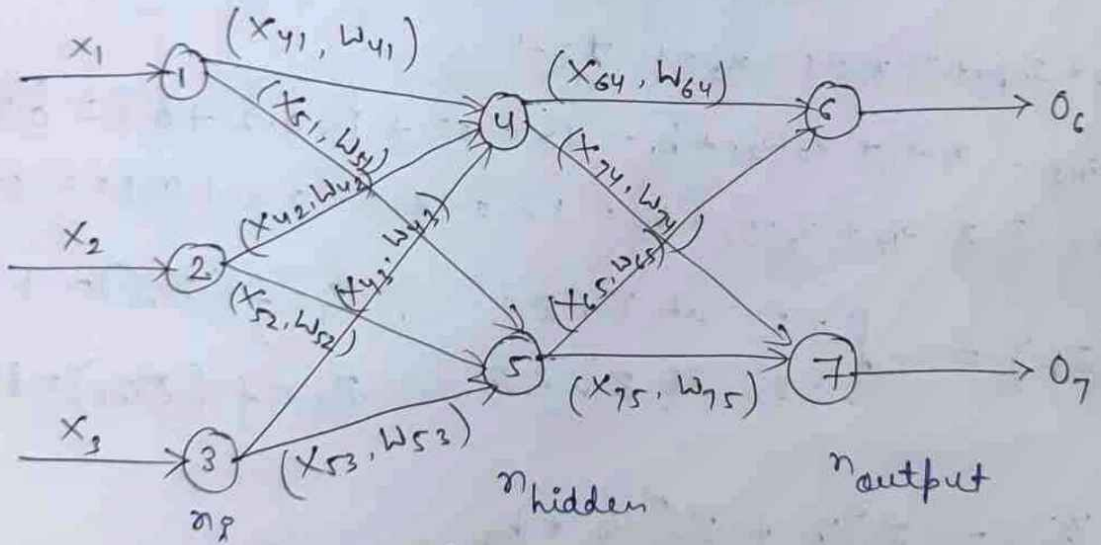
$W_{21} = 0.2 + 0.5(-1 - 0.55) \times 1 = 0.575$

$W_{22} = 0.2 + 0.5(1 - 0.45) \times 1 = -0.525$

$b_2 = 0.15 + 0.5(-1 - 0.45) = -0.575$

similar for other inputs

Back propagation algorithm :-



1, 2, 3, 4, 5, 6, 7 → units

- Each training example is a pair of the form (x, t)
- x is the learning rate (eg. 0.5)
- x_i is the no. of network input.
- n_{hidden} the no. of units in hidden layer

- n_{out} : the no. of output units.

Algorithm:

Step 1: Create a feed forward network with n_i input layer, n_{hidden} & n_{output}

$$O_4 = \delta(x_{41}w_{41} + x_{42}w_{42} + x_{43}w_{43})$$

$$O_5 = \delta(w_{51}x_{51} + x_{52}w_{52} + x_{53}w_{53})$$

$$O_6 = \delta(x_{64}w_{64} + x_{65}w_{65})$$

$$O_7 = \delta(x_{74}w_{74} + x_{75}w_{75})$$

$$\delta_x = \frac{1}{1+e^{-x}}$$

Step 2: Initialize the network weights with some small random value.

Step 3: Until the termination condition is met ... Do:

- For each (x, t) in training example. Do

(i) Input the instance x to the network and compute the output O_v for every unit v in the network.

- Propagate the error backward through the network.

(ii) For each network unit k, calculate its error term δ_k.

$$s_k \leftarrow o_k(1-o_k)(t_k - o_k)$$

3. For each network unit calculate its error term s_h.

$$s_h \leftarrow o_h(1-o_h)\sum_{k \in \text{output}} w_{h,k}\, s_k$$

Update each network weight.

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\Delta w_{ji} = \eta\, s_j\, x_{ji}$$

Derivation of back propagation algorithm :-

(i) Stochastic Gradient Descent Rule:

- To derive the eqn for updating the weight in back propagation algorithm we use stochastic gradient descent rule.
- Stochastic gradient descent rule involves iterating through the training example one at a time. descending the gradient of the error E_d wrt the single example. In other words for each training example, every weight w_{ji} is updated by Δw_{ji} where $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$

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

where E_d is the error on the training example i.e. half the sq. difference b/w the target output and the actual output over all output units in the network.

$$E_d(\vec{w}) = \frac{1}{2} \sum_{\text{output}} (t_k - o_k)^2$$

x_{ji} — the ith input to unit j

w_{ji} — the weight associated with input to unit j

$net_j = \sum w_{ji} x_{ji}$

o_j = the output computed for unit j

t_j = the target output for unit j

σ = sigmoid function.

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_{ji}} \frac{\partial net_j}{\partial w_{ji}}$$

$$net_{ji} = \sum w_{ji} x_{ji}$$

$$\frac{\partial E_d}{\partial net_{ji}} x_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_{ji}} x_{ji}$$

Case 1: where j is an output unit for the network.

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial net_j}$$

Case 2: where j is an internal unit of the network.

Case 1: Training rule for output unit weights just as w_{ji} can influence the rest of the network only through net_j. net_j can influence the network only through o_j. Therefore Hence we can invoke the chain rule again to write

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\left[\frac{1}{2}\sum_{k \in \text{output}} (t_k - o_k)^2\right]$$

$$= \frac{1}{2} \cdot 2 (t_j - o_j) \frac{\partial (t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial (net_j)} \rightarrow \frac{\partial \sigma(x)}{\partial x} = \sigma(x) \cdot (1 - \sigma(x))$$

$$= \sigma(net_j)(1 - \sigma(net_j))$$

$$= o_j (1 - o_j)$$

$$\boxed{\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\Delta w_{ji} = \eta \underbrace{(t_j - o_j)\, o_j\, (1 - o_j)}_{\delta_j} x_{ji}$$

$$\boxed{\Delta w_{ji} = \eta\, \delta_j\, x_{ji}}$$

Case 2: Training rule for hidden layer

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in downstream(j)} -\delta_k \underbrace{\frac{\partial net_k}{\partial o_j}}_{①} \underbrace{\frac{\partial o_j}{\partial net_j}}_{②}$$

$$\frac{\partial net_k}{\partial o_j} = \frac{\partial x_{kj} w_{kj}}{\partial o_j}$$

$$= \frac{\partial o_j w_{kj}}{\partial o_j} = w_{kj}$$

$$\frac{\partial O_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j}$$

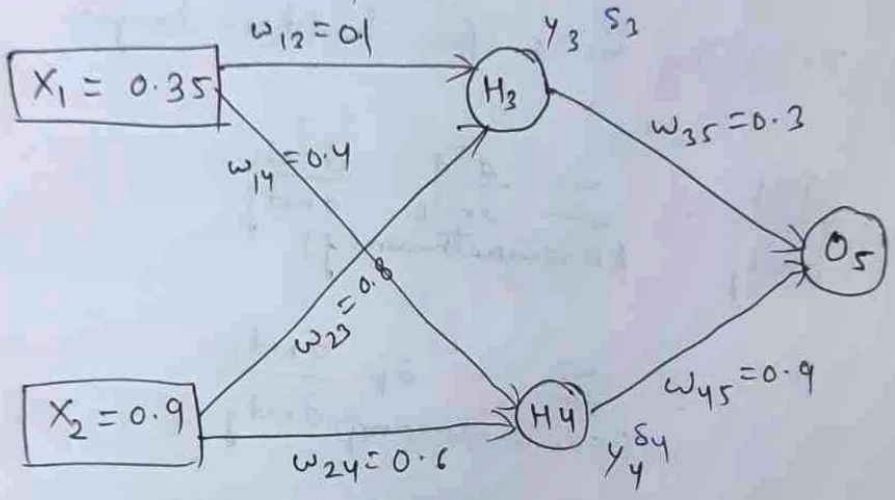
$$= \sigma(net_j)(1 - \sigma(net_j))$$

$$= O_j(1 - O_j)$$

$$\Delta w_{ji} = \eta O_j(1 - O_j) \sum_{k \in Kdownstream(j)} \delta_k w_{kj} x_{ji}$$

$$\boxed{\Delta w_{ji} = \eta \delta_j x_{ji}}$$

Eg:



Activation function sigmoid $\delta = \frac{1}{1 + e^{-x}}$

Actual output $y_5 = 0.5$, learning rate = 1

y_3 H_3 = $\sigma(X_1 w_{13} + X_2 w_{23})$

y_3 H_3 = $\sigma(0.35 \times 0.1 + 0.9 \times 0.8) = 0.755$

y_4 = $\sigma(0.35 \times 0.8 + 0.9 \times 0.6) =$

$y_3 = \frac{1}{1+e^{-0.755}} = 0.68$

$O_4 = W_{23} \times 1 + W_{24} \times 2$

$= 0.8 \times 0.35 + 0.6 \times 0.9 = 0.68$

$y_4 = \frac{1}{1+e^{-0.68}} = 0.6637$

$O_5 = W_{35} y_3 + W_{45} y_4$

$= 0.3 \times 0.68 + 0.9 \times 0.6637$

$= 0.801$

$y_5 = \frac{1}{1+e^{-0.801}} = 0.69$

Error $= y_{target} - y_5$ ~~or $y_{target} - y_5$~~

$= 0.5 - 0.69 = -0.19$

Each weight updation :-

$\Delta w_{ji} = x \delta_y O_i$

$\delta_j = O_j(1-O_j)(t_j - O_j)$

if j is an output unit

$\delta_j = O_j(1-O_j) \sum_k \delta_k w_{kj}$ if j is the hidden unit

Compute $\delta_3, \delta_4, \delta_5$ for output unit.

$$\delta_5 = y_5(1-y_5)(y_{target}-y_5)$$

$$= 0.69(1-0.69)(0.5-0.69)$$

$$= -0.0406$$

for hidden layer

$$\delta_3 = y_3(1-y_3)w_{35} \times \delta_5$$

$$= 0.68(1-0.68)(0.3 \times (-0.0406))$$

$$= -0.00265$$

$$\delta_4 = y_4(1-y_4)w_{45}\delta_5$$

$$= 0.6637 \times (1-0.6637)(0.9)(-0.0406)$$

$$= -0.0082$$

Compute new weights:

$$\Delta w_{ji} = \eta \delta_j o_{ji}$$

$$w_{45} = \eta \delta_5 - y_4$$

$$= 1 \times (-0.0406)(0.6637)$$

$$= -0.0267$$

$w_{35\,new} = w_{35\,old} + \Delta w_{35}$

$w_{45\,new} = w_{45\,old} + \Delta w_{45}$

$w_{13\,new} = w_{13\,old} + \Delta w_{13}$

$w_{23\,new} = w_{23\,old} + \Delta w_{23}$

i	j	w_{ij}	δ_j	x_i	x	Updated weight
1	3	0.1	-0.00255	0.35	1	0.0091
2	3	0.8	-0.00255	0.9	1	0.7976
1	4	0.4	-0.0082	0.35	1	0.3971
2	4	0.6	-0.0082	0.9	1	0.5926
3	5	0.3	-0.0406	0.68	1	0.2724
4	5	0.9	-0.0406	0.6637	1	0.8731

Associative Memories

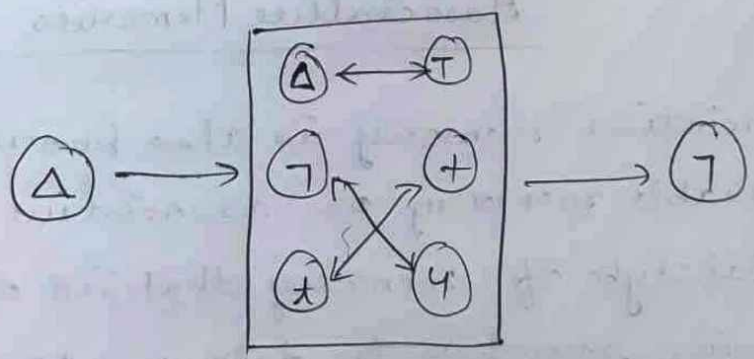
The associative memory is also known as contained addressable memory or associative storage. It is a special type of memory that is optimized for performing searches to data as oppose to providing a simple direct access to the data based on the address. It can store the set of patten as memories when the associative memory is being represented with a key pattern, it responded by producing one of

the stored pattern which closely resembles or related to the key pattern.

Types:

(i) Auto Associative memory network:

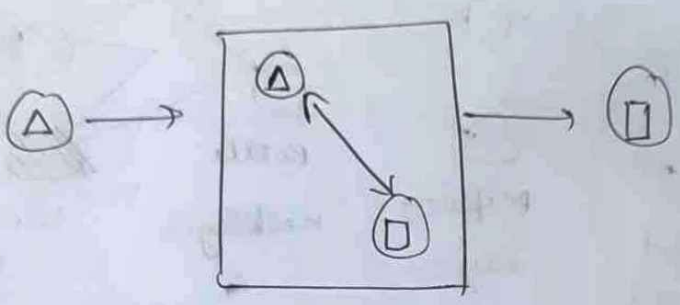
An auto associative memory network, also known as a recurrent neural network is a type of associative memory that is used to recall a pattern from partial or degraded input allowing the network to learn and remember the pattern it has been trained on. This type of memory network is commonly used in applications such as speech, image recognition where the input data may be incomplete or noisy.



(ii) Heteroassociative memory network:

A hetero associative memory network is a type of associative memory that is used to associate one set of patterns with another pattern. In a hetero

associative network. the input pattern associated with a different output allowing the network to remember the association between the two set of pattern. This type of memory network is commonly used in application such as data compression and data retrieval.



Advantages:

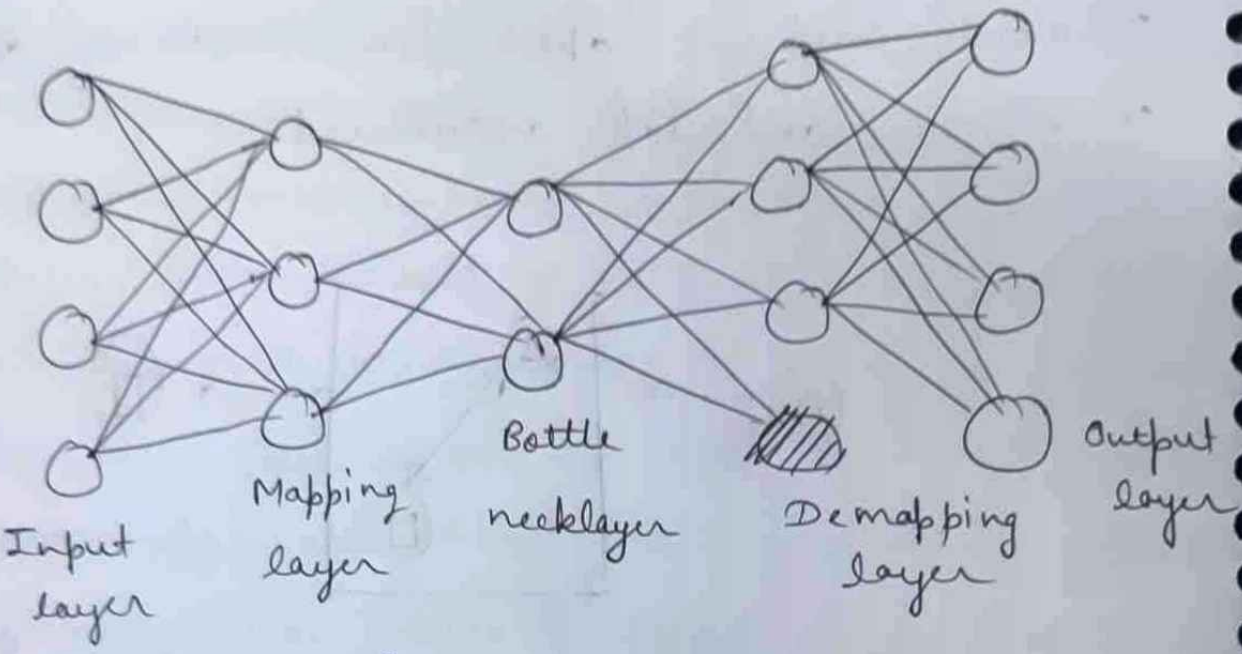
(i) It is used where search time needs to be less or short.
(ii) It is suitable for parallel searches.
(iii) It is often used to speed up data base.
(iv) It is used in page tables used by virtual memory and used in neural network.

Disadvantages:

(i) It is more expensive than RAM.
(ii) Each cell must have storage capability and logical

circuits for matching its contents with external arguments

Auto Associative Memory Neural Network :-



$$W = \sum_{p=1}^{P} s^T(p)\, s(p)$$

where W = weight matrix

T = learning ratio

$s(P)$ = P distinct n dimensional Prototype pattern

Training the algorithm :-

Step 1 : Initialize all weights for $i = 1, 2, 3 \ldots n$

$j = 1, 2, 3 \ldots n$

such that $w_{ij} = 0$

Step2: For each vector to be stored, repeat the following step

(a) set activation for each input unit $i = 1, 2 - n$

$$x_i = s_i$$

(b) set activation for each output unit $j = 1, 2 - n$

$$y_j = s_j$$

(c) Update the weight from the following eqn:

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

Testing the algorithm :-

For testing whether the input is known or unknown to the model, we need to perform the following steps

Step1: Take the weights that were generated during the training phase using Hebb's rule.

Step2: For each input vector perform the following step

(a) set activation in the input units equal to input vectors.

(b) set activation in the input units equal to input vectors

(b) Set activation for the output units $j = 1, 2 \ldots n$

$$y_{inj} = \sum_i x_i w_i$$

$$y_j = f(y_{inj}) = \begin{cases} 1 & y_{inj} > 0 \\ -1 & y_{inj} \leq 0 \end{cases}$$

eg: vector $A = [1 \ 1 \ 1 \ 1]$, vector $B = [-1 \ 1 \ 1 \ -1]$

$$W = \sum_{p=1}^{P} s^T(p)\, s(p)$$

$$= W_1 + W_2$$

$$W_1 = s^T(1)\, s(1)$$

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

$$W_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

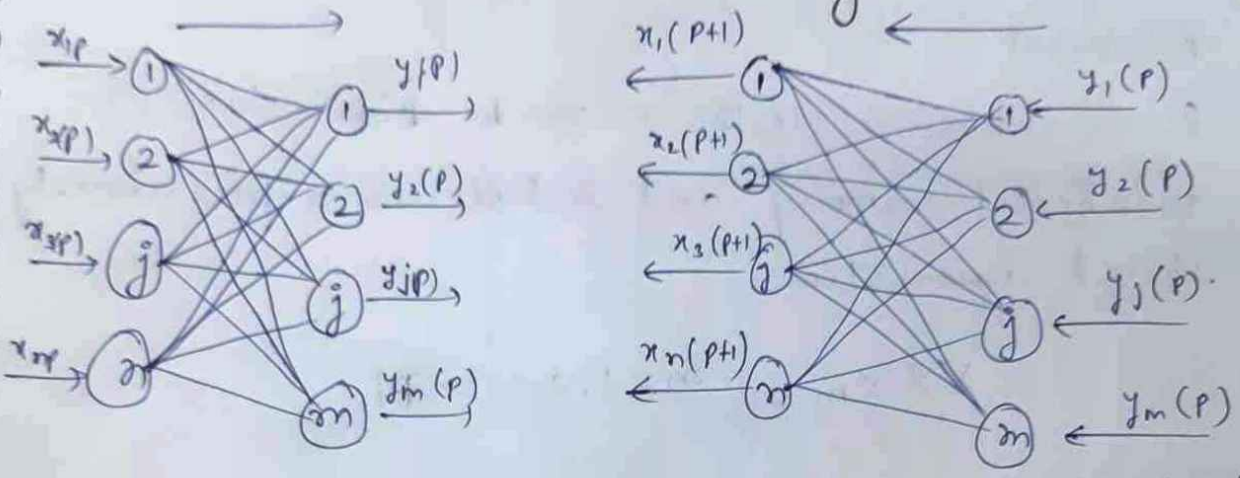
$$W_2 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$W = W_1 + W_2$$

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

Bidirectional Associative Memory :-



Bidirectional associated memory without supervised learning model in artificial neural network. This is the Hetero associated memory for input pattern. It returns another pattern which is potentially of different size. When bidirectional associated memory accepts an input of n-dimen-sional vector $\vec{x}$ from set-A then the model recalls m-dimensional vector $\vec{y}$ from set-B. Similarly $\vec{y}$ is treated as input. The bidirectional associated memory recalls $\vec{x}$

storage (learning)

$$W = \sum_{m=1}^{M} X_m Y_m^T$$

Testing $\quad Y_m = \text{sign}\,(W^T X_m), \quad m = 1, 2, \cdots n$

$$X_m = \text{sign}\,(W Y_m) \qquad m = 1, 2 \cdots n$$

Retrieval

for an unknown vector x to be bidirectional associated memory and retrieves a previously stored association.

$$X \neq X_m \qquad m = 1, 2 \cdots M$$

- Initialize the BAM

$$x(0) = X, \quad P = 0$$

- Calculate the BAM output at iteration P

$$y(P) = \text{sign}\,(W^T X(P))$$

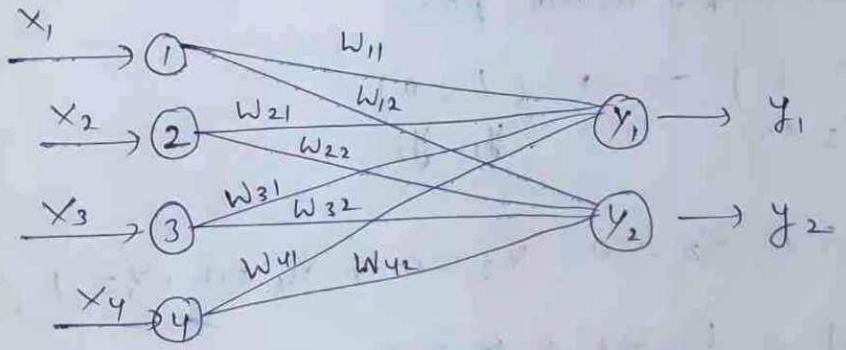
- Update the input vector.

$$X(P+1) = \text{sign}\,(W Y(P))$$

- Repeat the iteration until convergence when output and input remain unchanged.

Q: I/P / Target	S_1	S_2	S_3	S_4	T_1	T_2
1st	1	0	1	0	1	0
2nd	1	0	0	1	1	0
3rd	1	1	0	0	0	1
4th	0	0	1	1	0	1

Train a hetero associated memory using Hebb's rule to store input row vector to output row vector.



Ans: 1st input/output vector pair

$$(1\ 0\ 1\ 0) \longleftrightarrow (1\ 0)$$
$$x_1\ x_2\ x_3\ x_4 \qquad y_1\ y_2$$

Step 0: Initialize the ~~units~~ weights $W_{ij} = 0$

Step 1: $X_1 = 1 \quad X_2 = 0 \quad X_3 = 1 \quad X_4 = 0$

$y_1 = 1 \quad y_2 = 0$

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

$w_{11}(new) = w_{11}old + x_1 y_1 = 0 + 1 \times 1 = 1$

$w_{12}(new) = 0 + 1 \times 0 = 0$

$w_{21}(new) = 0 + 0 \times 0 = 0$

$w_{22}(new) = 0 + 1 \times 0 = 0$

$w_{31}(new) = 0 + 1 \times 1 = 1$

$w_{32}(new) = 0 + 1 \times 0 = 0$

$w_{41}(new) = 0 + 0 \times 1 = 0$

$w_{42}(new) = 0 + 0 \times 0 = 0$

2nd input/output vector pair

$(1\ 0\ 0\ 1) \longrightarrow (1, 0)$

$x_1\ x_2\ x_3\ x_4 \qquad y_1\ y_2$

Step 1: $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0, y_1 = 1, y_2 = 0$

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

$w_{11}(new) = 1 + 1 \times 1 = 2$

$w_{12}(new) = 0 + 1 \times 0 = 0$

$w_{21}(new) = 0 + 0 \times 1 = 0$

$w_{22}(new) = 0 + 0 \times 0 = 0$

$w_{31}(new) = 1 + 0 \times 1 = 1$

$w_{32}(new) = 0 + 0 \times 0 = 0$

$w_{41}(new) = 0 + 0 \times 0 = 1$

$w_{42}(new) = 0 + 1 \times 0 = 1$

Q:-

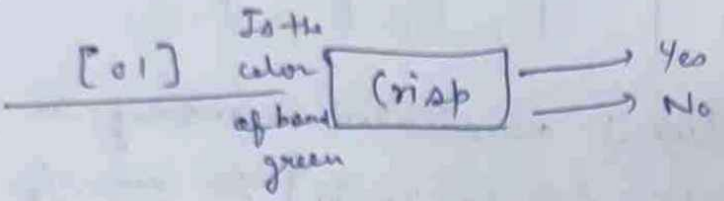
I/P/O/P	S_1	S_2	S_3	S_4	T_1	T_2
1st	1	0	0	0	0	1
2nd	1	1	0	0	0	1
3rd	0	0	0	1	1	0
4th	0	0	1	1	1	0

Train a bidirectional associated memory to store the i/p vector $S = (S_1, S_2, S_3, S_4)$ to an o/p vector $t = (T_1, T_2)$.

Fuzzy System

(i) Classical set / crisp set

(ii) Fuzzy set



eg: $X = \{1, 2, 3, 4\}$

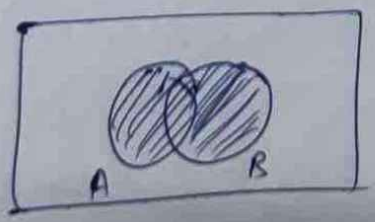
(i) Power set $= 2^4 = 16$

(ii) Cardinality $= 4$

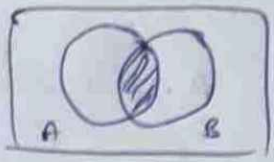
Power set = $\{1,2,3,4\}\{(1,2)\}\{2,3\}\{3,4\}\{\beta,4\}$
$\{1,2,3\}\{2,3,4\}\{1,2,4\}\{1,3,4\}$
$\{1\}\{2\}\{3\}\{4\}\{\phi\}\{2,4\}$
$\{3,2\}$

Operation of classical set :-

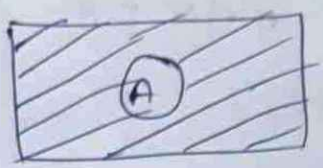
Union $A \cup B = \{x / x \in A \text{ or } x \in B\}$



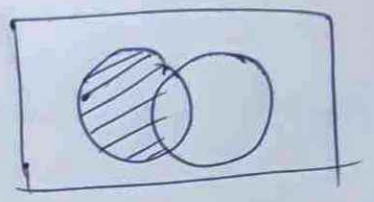
Intersection $A \cap B = \{x / x \in A \text{ and } x \in B\}$



Complement $\overline{A} = \{x / x \notin A, x \in X\}$



Difference: $A \setminus B = \{x / x \in A \text{ and } x \notin B\}$



Properties of classical set :-

(i) Commutativity

$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$

(ii) Associativity :

$$A \cup (B \cup C) = (A \cup B) \cup C$$
$$A \cap (B \cap C) = (A \cap B) \cap C$$

(iii) Distributivity

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

(iv) Idempotency :

$A \cup A = A, \quad A \cap A = A$

(v) Identity :

$A \cup X = X, \quad A \cup \phi = A, \quad A \cap X = A, \quad A \cap \phi = \phi$

(vi) Transitivity

$A \subseteq B \subseteq C$ then $A \subseteq C$

(vii) De Morgan's law

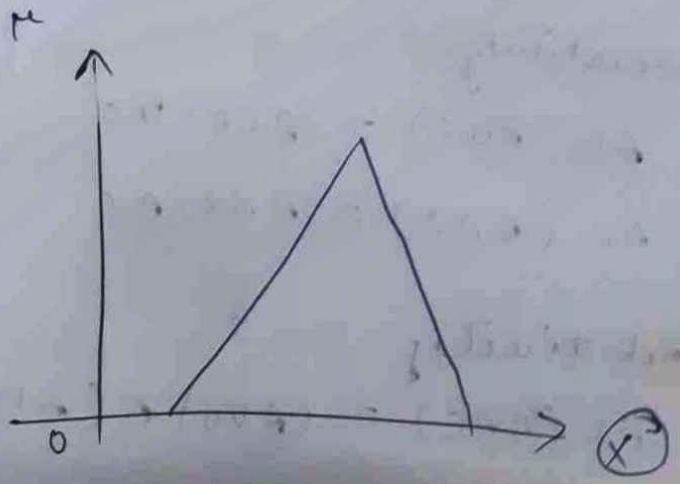
$\overline{A \cap B} = \overline{A} \cup \overline{B}$

$\overline{A \cup B} = \overline{A} \cap \overline{B}$

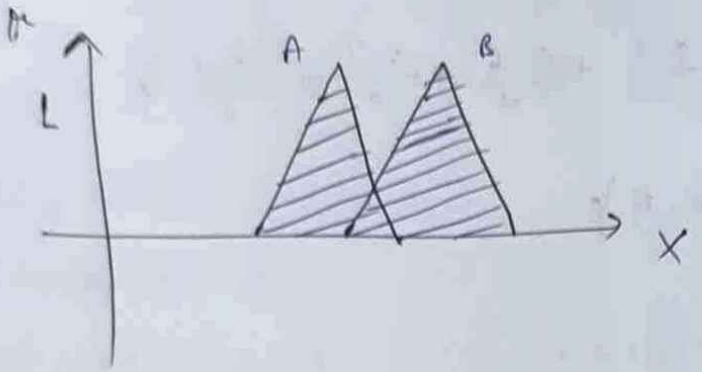
Mapping of classical set :-

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

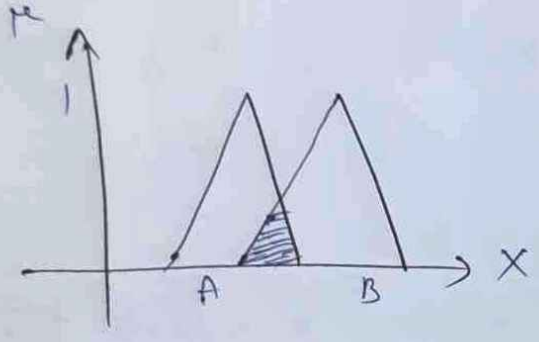
Fuzzy set :

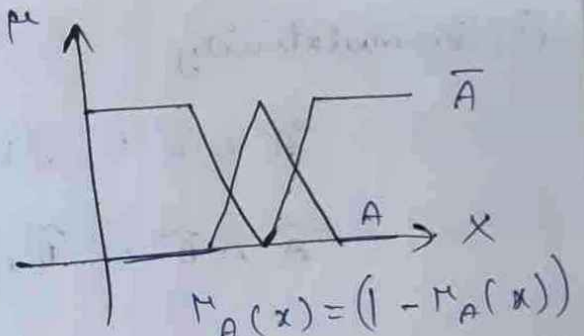


Union $\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x)$



Intersection : $\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x)$





$\mu_{\bar{A}}(x) = (1 - \mu_A(x))$

Eg: $A = \{ \frac{1}{2}^{0.5} + \frac{0.5}{3}^{0.16} + \frac{0.6}{4}^{0.15} + \frac{0.2}{5}^{0.04} + \frac{0.6}{6}^{0.1} \} = \{0.95\}$

$B = \{ \frac{0.5}{2}^{0.25} + \frac{0.8}{3}^{0.26} + \frac{0.4}{4}^{0.1} + \frac{0.7}{5}^{0.14} + \frac{0.3}{6}^{0.05} \} = 0.8$

~~$A \wedge B = \{ 0.1 \}$~~

~~$A \vee B = \{$~~ $\bar{A} = \{ \frac{0}{2} + \frac{0.5}{3} + \frac{0.4}{4} + \frac{0.8}{5} + \frac{0.4}{6} \}$

$\bar{B} = \{$ $\}$

$A \cup B = \max\{A, B\}$

$A \cup B = \left\{\frac{1}{2} + \frac{0.8}{3} + \frac{0.6}{4} + \frac{0.7}{5} + \frac{0.6}{6}\right\}$

$A \cap B = \min\{A, B\}$

Properties of Fuzzy set:

(i) Commutativity:

$$\tilde{A} \cup \tilde{B} = \tilde{B} \cup \tilde{A}$$

$$\tilde{A} \cap \tilde{B} = \tilde{B} \cap \tilde{A}$$

(ii) Associativity:-

$$\tilde{A} \cup (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cup \tilde{C}$$

$$\tilde{A} \cap (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cap \tilde{B}) \cap \tilde{C}$$

(iii) Distributivity:

$$\tilde{A} \cup (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cup \tilde{B}) \cap (\tilde{A} \cup \tilde{C})$$

$$\tilde{A} \cap (\tilde{B} \cap \tilde{C}) = (\tilde{A} \cap \tilde{B}) \cup (\tilde{A} \cap \tilde{C})$$

(iv) Idempotency:

$$\tilde{A} \cup \tilde{A} = \tilde{A}$$

$$\tilde{A} \cap \tilde{A} = \tilde{A}$$

Identity :-

$\tilde{A} \cup \phi = A$ and $\tilde{A} \cap X = A$

$\tilde{A} \cap \phi = \phi$ and $\tilde{A} \cup X = X$

Fuzzy relation :-

Note : (i) $A \times B \neq B \times A$

(ii) $|A \times B| = |A| \times |B|$

(i) Crisp relation :

ordered pair $A \times B = \{(a,b) \mid a \in A \text{ & } b \in B\}$

Note (iii) $A \times B$ provides the mapping of $a \in A$ to $b \in B$.

eg: $A = \{1, 2, 3, 4\}$ $B = \{3, 5, 7\}$

$A \times B = \{(1,3)\ (1,5)\ (1,7)\ (2,3)\ (2,5)\ (2,7)$

$(3,3)\ (3,5)\ (3,7)\ (4,3)\ (4,5)\ (4,7)\}$

let define a relation

$R = \{(a,b) \mid_b = a+1,\ (a,b) \in A \times B\}$

$= \{(2,3), (4,5)\}$

$$R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 3 & 5 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

→ Operations on Crisp relationship :-

$R(x,y)$ and $S(x,y)$ are two relations defined over two crisp relation

$x \in A, \quad y \in B$

Union: $R(x,y) \cup S(x,y) = \max\left(R(x,y), S(x,y)\right)$

Intersection: $R(x,y) \cap S(x,y) = \min\left(R(x,y), S(x,y)\right)$

Complement: $\bar{R}(x,y) = 1 - R(x,y)$

eg: $R = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$R(x,y) \cap S(x,y) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

$R(x,y) \cup S(x,y) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Composition of two crisp set:-

$R \circ S = \{(x,z) \mid (x,y) \in R \text{ and } (y,z) \in S \text{ and } \forall y \in Y\}$

$T = R \circ S$

~~$T(x,y) =$~~

$T(x,z) = \max\{\min\{R(x,y), S(y,z) \text{ and } \forall y \in Y\}\}$

eg: $X = \{1,3,5\}$ $Y = \{1,3,5\}$

$R = \{(x,y) | y = x+2\} = \{(1,3)(3,5)\}$

$S R = \{(x,y) | x < y\} = \{(1,3)(1,5)(3,5)\}$

$$R = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 3 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \qquad S = \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 3 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

$\min\{R(x,y), S(y,z)\} = \begin{bmatrix} (0,0,0) & (0,0,0) & (0,0,1) \\ (0,0,0) & (0,0,0) & (0,0,0) \\ (0,0,0) & (0,0,0) & (0,0,0) \end{bmatrix}$

$\max\{\min \text{ ——— }\} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ nine ordered pair

Fuzzy relation :-

eg: $X = \{ \text{typhoid}, \text{viral}, \text{cold} \}$

$Y = \{ \text{running nose}, \text{high temp}, \text{shivering} \}$

$$R = \begin{array}{c} \\ T \\ V \\ C \end{array} \begin{array}{c} \begin{array}{ccc} RN & HT & S \end{array} \\ \begin{bmatrix} 0.1 & 0.9 & 0.8 \\ 0.2 & 0.9 & 0.7 \\ 0.9 & 0.4 & 0.6 \end{bmatrix} \end{array}$$

Fuzzy Cartesian product :-

Set A $\quad \mu_A(x) \mid x \in y$

Set B $\quad \mu_B(y) \mid y \in Y$

then $R = A \times B \subset X \times Y$

$$\mu_R(x, y) = \mu_{A \times B}(x, y) = \min \{ \mu_A(x), \mu_B(y) \}$$

$A = \{ (a_1, 0.2) \; (a_2, 0.7) \; (a_3, 0.4) \}$

$B = \{ (b_1, 0.5) \; (b_2, 0.6) \}$

$R = A \times B$

$$A \times B = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{cc} b_1 & b_2 \end{array} \\ \begin{bmatrix} 0.2 & 0.2 \\ 0.5 & 0.6 \\ 0.4 & 0.4 \end{bmatrix} \end{array}$$

Operations on Fuzzy relation :-

Union : $\mu_{R \cup S}(a,b) = \max\{\mu_R(a,b), \mu_S(a,b)\}$

Intersection : $\mu_{R \cap S}(a,b) = \min\{\mu_R(a,b), \mu_S(a,b)\}$

Complement : $\mu_{\bar{R}}(a,b) = 1 - \mu_R(a,b)$

Composition : $T = R \circ S \Rightarrow \mu_{R \circ S} = \max\{\min(\mu_R(x,y), \mu_S(y,z)\}_{y \in Y}$

$$S = \begin{array}{c} \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 0.6 & 0.4 & 0.7 \\ 0.5 & 0.8 & 0.9 \end{bmatrix} \end{array} \quad 2 \times 3$$

$X = (x_1, x_2, x_3)$ $Y = (y_1, y_2)$ $Z = (z_1, z_2, z_3)$

$$R = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cc} y_1 & y_2 \end{array} \\ \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.9 \\ 0.8 & 0.6 \end{bmatrix} \end{array}$$

3×2

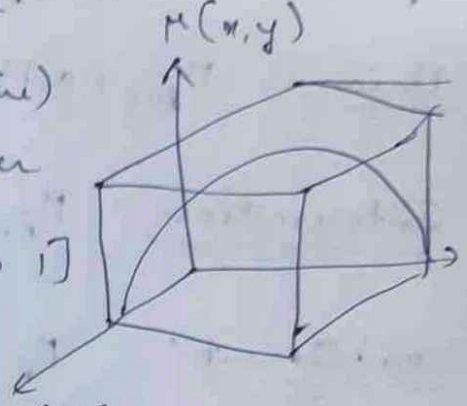
$R \times S =$

2D Membership function – Binary fuzzy system :-

Binary fuzzy relations are fuzzy sets $A \times B$ which map each element in $A \times B$. or to a membership rate 0 & 1, both are inclusive.

Let $X = R^+ = Y$ (the +ve real line)

$R = X \times Y =$ "y is much greater than x.



$$\mu_r(x,y) = \begin{cases} \frac{y-x}{4} & \text{if } y \geq x \\ 0 & \text{if } y \leq x \end{cases}$$

$X = \{3, 4, 5)\}$ & $Y = \{3, 4, 5, 7, 6\}$

$$R = \begin{array}{c} \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0.25 & 0.5 & 0.75 & 1 \\ 0 & 0 & 0.25 & 0.5 & 0.75 \\ 0 & 0 & 0 & 0.25 & 0.5 \end{bmatrix} \end{array}$$

eg: If x is A or y is B then Z is C. Given that R_1: if x is A then Z is C.

R_2: if y is B then Z is C.

Fuzzy proposition :-

- Two value logic → Boolean logic crisp func^{n}.
- Multi level logic

a	b	$\wedge$ (AND)	$\vee$ (OR)	$\neg a$ (NOT)	$\Rightarrow$ (Implication)	$=$ → equal
0	0	0	0	1	1	1
0	1/2	0	1/2	1	1	1/2
0	1	0	1	1	1	0
1/2	0	0	1/2	1/2	1/2	1/2
1/2	1/2	1/2	1/2	1/2	1/2	1
1/2	1	1/2	1	1/2	1	1/2

$a \Rightarrow b$ $\quad \bar{a} \vee b$

Symbol	Connection	Usage	Definition
$\neg$	NOT	$\neg P$	$1 - T(P)$
$\vee$	OR	$P \vee Q$	$\max\{T(P), T(Q)\}$
$\wedge$	AND	$P \wedge Q$	$\min\{T(P), T(Q)\}$
$\Rightarrow$	Implication	$P \Rightarrow Q$	$\max\{1 - T(P), T(Q)\}$
$=$	Equality	$(P-Q)$ or $P \Rightarrow Q \wedge Q \Rightarrow P$	$1 - \lvert T(P) - T(Q) \rvert$

eg: P: Mohan is efficient $T(P) = 0.8$

Q: Ram is efficient $T(Q) = 0.6$

→ Mohan is not efficient

$T(\neg P) = 1 - T(P) = 0.2$

Mohan is efficient and so is Ram. (AND → max)

Either Mohan or Ram is efficient.

If Mohan is efficient then so is Ram.

The difference b/w Fuzzy proposition Vs Crisp proposition :-

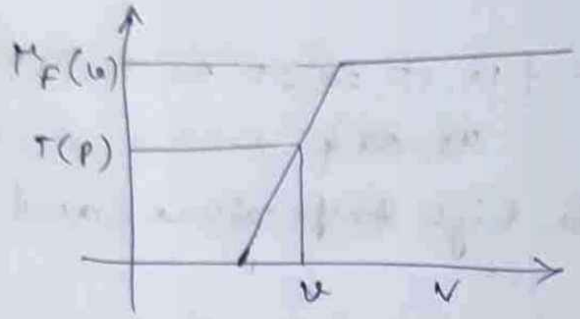
(i) The fundamental difference between the crisp proposition and fuzzy proposition is range of truth value.

(ii) While each classical proposition is required to be either true or false. The truth & falsity of fuzzy proposition is a matter of degree.

(iii) The degree of truth of each fuzzy proposition is expressed by the value in the interval b/w 0 & 1 and both are inclusive.

Canonical Representation of fuzzy logic :-

P.V is F.



Fuzzy Implications (Rules) :-

(i) Fuzzy rule
(ii) Example of fuzzy implication
(iii) Interpretation of fuzzy rule.
(iv) Product operation
(v) Zadeh's Max-min rule.

(i) Fuzzy Rule :

If $\underbrace{x \text{ is } A}_{\text{Antecent}}$ then $\underbrace{y \text{ is } B}_{\text{consequence}}$.

A fuzzy implication also known as the fuzzy. If then fuzzy rule, fuzzy conditional statement having the form if x is A then y is B. where A & B are two linguistic defined by fuzzy set A & B on the universe of the discourse x & y respectively.

Eg: If pressure is high then temp. is low.

eg: If road is good then ~~driving~~ riding is smooth as traffic is high.

$P = \{1, 2, 3, 4\}$ and $T = \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$

let the linguistic variable high temperature and low pressure.

$T_{high} = \{(20, 0.2) (25, 0.4) (30, 0.6) (35, 0.6) (40, 0.7) (45, 0.8) (50, 0.8)\}$

$P_{low} = \{(1, 0.8) (2, 0.8) (3, 0.6) (4, 0.4)\}$

$R: T_{high} \longrightarrow P_{low}$ → $\max \{(1 - T_{high}), T_{Plow}\}$

$R: P_{low} \longrightarrow T_{high}$

	1	2	3	4
20	0.2	0.2	0.2	0.2
25	0.4	0.4	0.4	0.4
30	0.6	0.6	0.6	0.4
35	0.6	0.6	0.6	0.4
40	0.7	0.7	0.6	0.4
45	0.8	0.8	0.6	0.4
50	0.8	0.8	0.6	0.4

Interpretation of fuzzy rule :-

$A \rightarrow B$

(i) A coupled with B.

(ii) A entails B.

(i) A coupled with B.

$$R : A \rightarrow B = A \times B = \int_{X \times Y} \mu_A(x) * \mu_B(y)$$

where $*$ → T-norm operator.

T-norm operator :-

- Minimum : $T_{min}(a, b) = min(a, b) = a \wedge b$
- Algebraic product. $T_{ap}(a, b) = ab$
- Bounded product $T_{bp}(a, b) = 0 \vee (a + b - 1)$

Drastic product :

$$T_{dp} = \begin{cases} a & \text{if } b = 1 \\ b & \text{if } a = 1 \\ 0 & \text{if } a, b < 1 \end{cases}$$

$$a = \mu_A(x)$$

$$b = \mu_B(y)$$

Based on T-norm operator as defined we can automatically define the fuzzy rule $R : A \rightarrow B$ as a fuzzy set with a 2 dimension membership function.

$$\mu_R(x, y) = f(\mu_A(x), \mu_B(y)) = f(a, b)$$

Minimum operator :

$$R_m = A \times B = \int_{x \times y} \mu_A(x) \wedge \mu_B(y) \,/\, (x, y)$$

$$f_{min}(a, b) = a \wedge b$$

(Mamdani rule).