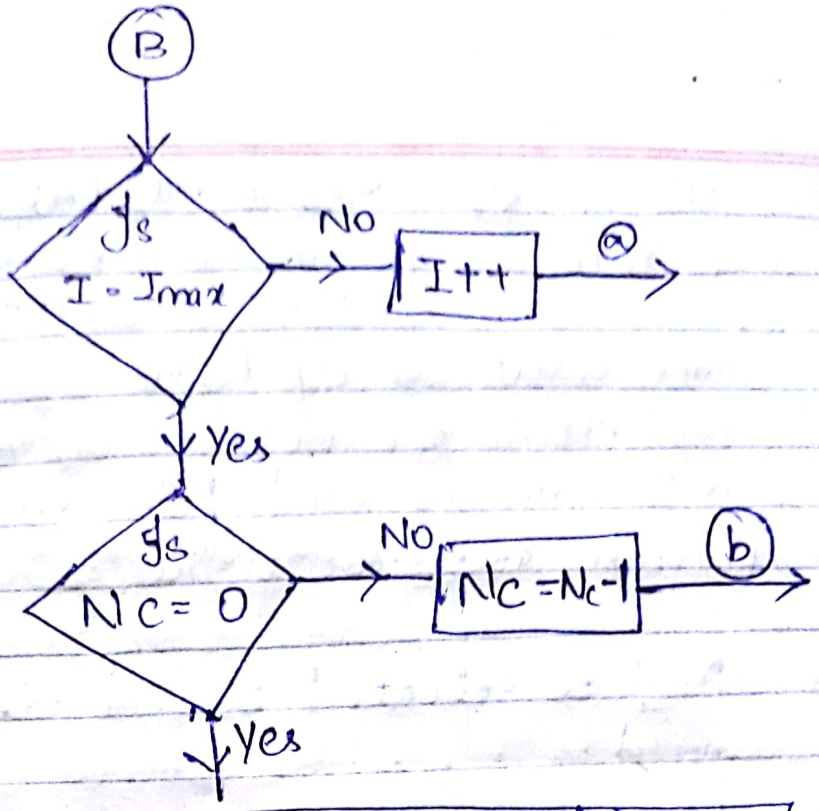
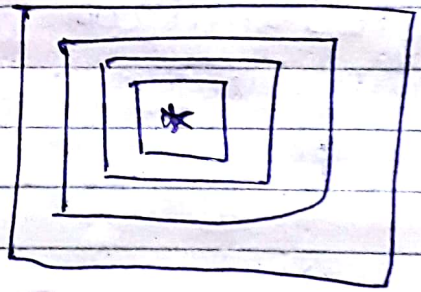


23/8/18

→ We are saving the weights, so that we can use it in the testing phase and there won't be any wt updation. Updation will only be done during the training phase.

→ When $N_c = 5$; will take the winner node and the neighbouring node upto order = 5 and their wts will be update.
Finally on every iteration N_c will change and finally $N_c = 0$ which represent the winner node

→ N_c is selected based on the 2-D grid.



Initially, $N_c = 4$.

WN
→ will update the wt. vector most similar to the I/P pattern.

→ As we go radially outwards, the wt vector of the neighbour nodes influenced less by the winner node.

→ $\eta(t) \propto \frac{1}{N_c + 1}$

$\eta \propto \frac{1}{N_c}$

So, as Nc become more, η becomes less. (so learning becomes less).

→ The nodes that are immediate neighbour of the winner node have learning constant more so they are influenced more by the winner node.

loop 1 Nc = 4 (Outermost loop)

loop 2 I = 1 to Imax (2nd loop)

loop 3 p = 1 to P (innermost loop)

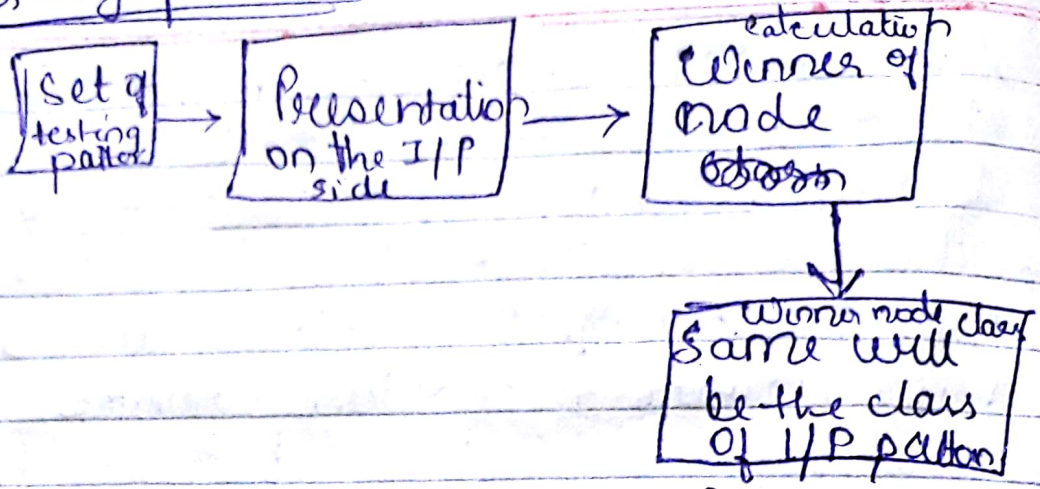
→ Imax depends on us, what we set.

→ P → well defined previously.

→ The value of Nc signifies, which ~~value of~~ wt. vector value is to be modified.

→ On testing, the input pattern is classified to that class in which the winner node lies.

Testing phase :-



Input features or attributes :

→ They identify the state of system.

→ How will we select the attributes ?

A B C D (State of system)

↑ ↑

parameter have same value — same as this

→ If one attributes doesn't change with state of the system then they are not discriminatory and should not be included.

→ If the attributes are discriminatory (change with state change) and should be included.

→ If 10 features vary in same manner as we move from one state to another, so why should we include all these 10 features, instead include only 1 of the feature.

→ At the same time we are concern of identifying the I/P pattern and reduce the computational burden.

→ If the operating domain is very large, then the state of the system that are close may fall in one class.

Multilayer Perceptron Model

→ It is used for function approximation, regression, etc.

$$y = f(x_1 x_2 \cdots x_n)$$

Non-linear relation b/w y and $(x_1 x_2 \cdots x_n)$.

→ For this we need some set of eqs:—

and these set of examples come in the form of:-

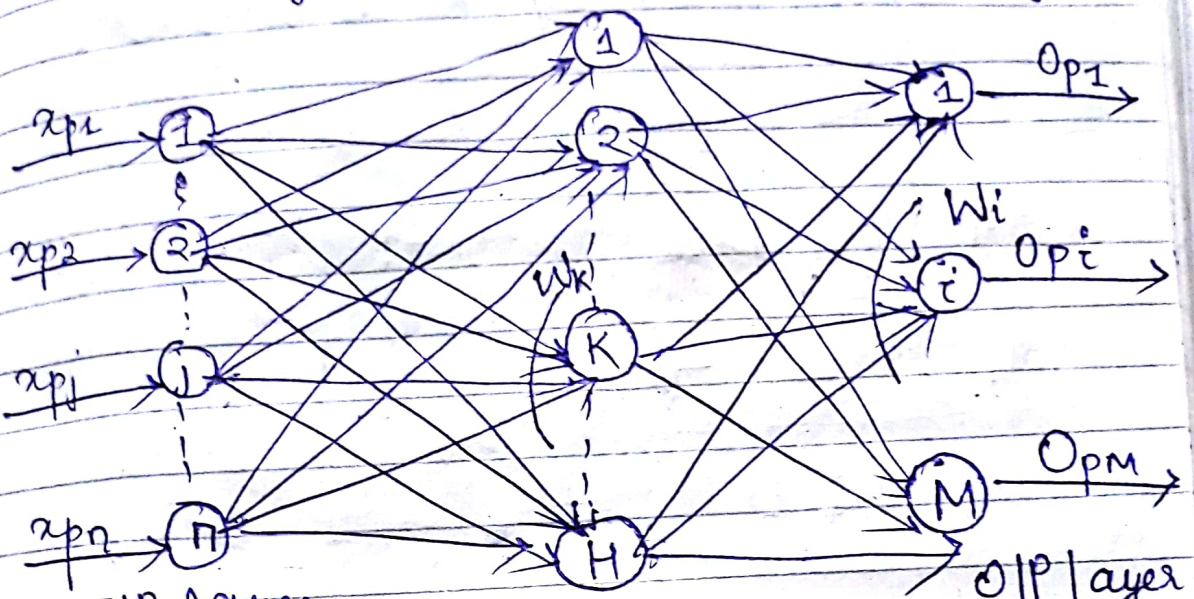
$X[1] \longrightarrow Y[1]$
$X[2] \longrightarrow Y[2]$
$\vdots$
$X[p] \longrightarrow Y[p]$
$X[p] \longrightarrow Y[p]$

→ It represent at any instant of time we have $X[1]$ set of inputs will give O/P $Y[1]$

→ We are taking these I/P O/P pattern in such a manner that entire O/P rating domain is being covered.

→ Care should be taken that I/P - O/P pattern should be taken less, so that our computation burden is less.
But with this our accuracy level in testing phase may decrease. So there is a tradeoff b/w accuracy level and training time and computational burden.
So we will take optimum I/P O/P

→ Architecture → feed-forward N/W
→ learning → supervised learning.



$$X[p] = \begin{bmatrix} x_{p1} \\ x_{p2} \\ \vdots \\ x_{pj} \\ \vdots \\ x_{pn} \end{bmatrix}$$

W_i = dimension of this wt vector will be no:- of nodes in the previous layer

→ O/P layer :- No:- of nodes in the O/P layer is problem dependent. The dimension of the O/P layer tells us the no:- of nodes in the O/P layer.

→ W_i - dimension of W_i will be the no:- of links terminating at the i^{th} node.

pattern.

TUT-1

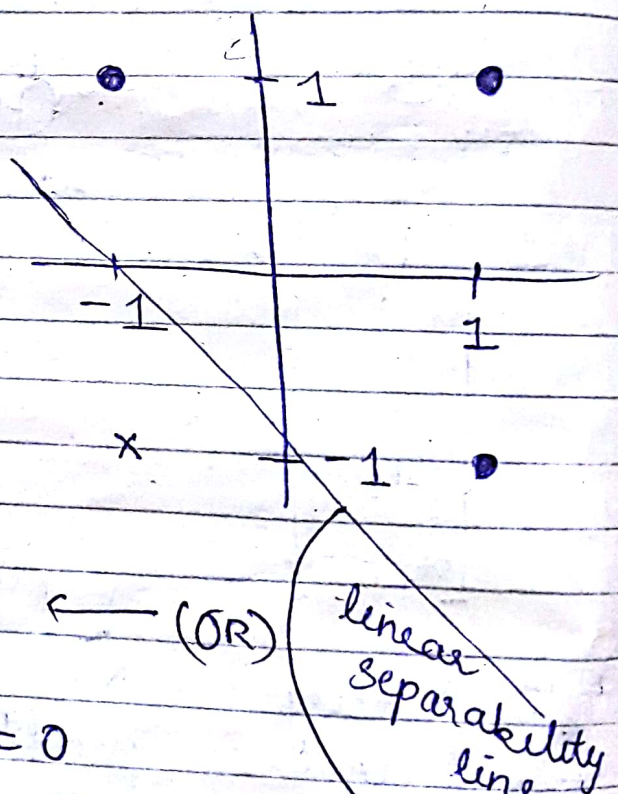
Q4 Obtain linear separability line for OR gate using bipolar binary data

OR x_1	x_2	o/p	w_1	w_2
1	1	1	1	1
-1	1	1	-1	1
1	-1	1	1	-1
-1	-1	-1	-1	-1

w_1 & w_2 = 1 & 1

1	1	2	1
-1	1	0	1
1	-1	0	1
-1	-1	-2	0

So for threshold ≥ 1
this is correct.



$\Rightarrow b + w_1 x_1 + w_2 x_2 = 0$

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

By McCulloch method

Eqn of line
$\Rightarrow x_2 = -x_1$

Ans

Here bias = 1

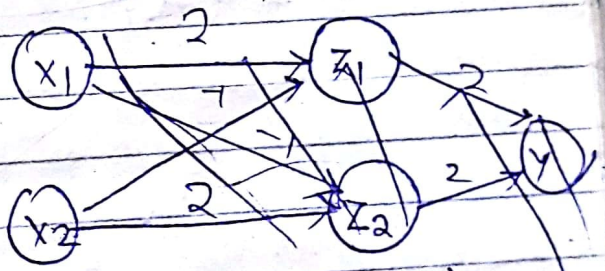
$$\Rightarrow x_2 = \frac{-1}{1} - x_1 \frac{1}{1}$$

$$\boxed{\therefore x_2 = -x_1 - 1} \longrightarrow \text{Ans}$$

for $x_2 > -x-1$ (there is one class)
for $x_2 < -x-1$ (there is another class)

Q3 Implementing Ex-OR using McCulloch Pitts model

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



$w_{31} = 2, w_{32} = -1, w_3 = 2$ (X)

x_1	x_2	neti		Target
0	0	0	0	0
0	1	-2	0	1
1	0	4	4	1
1	1	2		0

-1, -1, 2 (X)

X_1	X_2	neti	
0	0	0	0
0	1	-2	1
1	0	-2	1
1	1	-4	0

2	2	2	Target
0	0	0	0
0	1	4	1
1	0	4	1
1	1	8	0

w_{11}

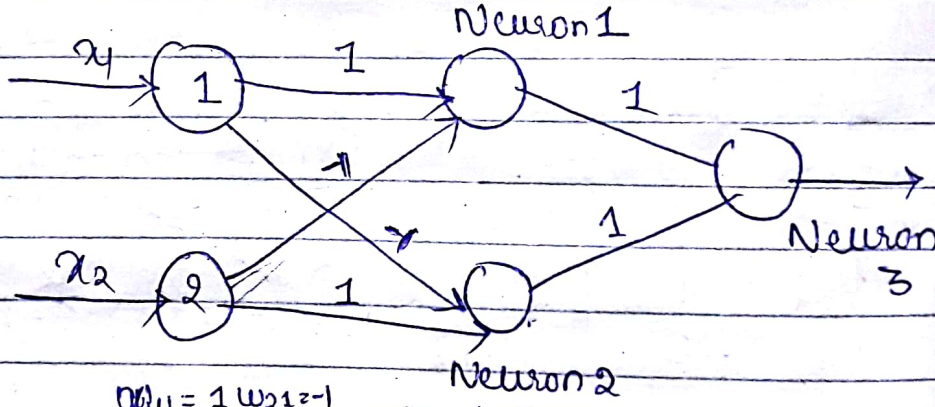
Q3 Implementing X-OR

$$x_1 \oplus x_2 = x_1 \cdot \overline{x_2} + \overline{x_1} \cdot x_2$$

We have to take wts in such a manner that $w_1 = 1$ and $w_2 = -1$ for 1st node

$w_1 = -1$ and $w_2 = 1$ for 2nd node

And then we have to perform OR operation so final w_{31} & $w_{32} = 1$ and 1



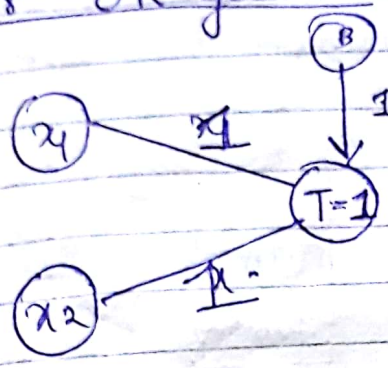
$w_{11} = 1$ $w_{21} = -1$ → $w_{12} = -1$; $w_{22} = 1$

X	Y	Net 1	Net 2	O/P1 T=1	O/P2 T=1	Net	O/P T=1
0	0	0	0	0	0	0	0
0	1	-1	1	0	1	1	1
1	0	1	-1	1	0	1	1
1	1	0	0	0	0	0	0

So thus this is our O/P

24/3/18

83 For OR gate



(Just same as previously done drawing the separa-bility line) then

$net_i = x_1 w_1 + x_2 w_2 + b = 0$

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

So we got the eq:-

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

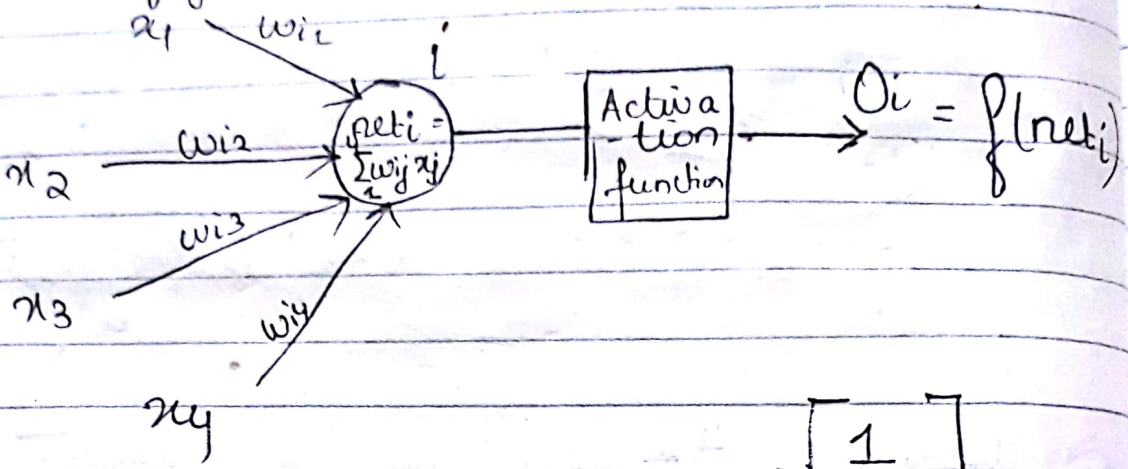
$w_1 = 1 \quad w_2 = 1 \quad b = 1$

x_1	x_2	y	net_i
1	1	1	3
1	-1	1	1
-1	1	1	1
-1	-1	-1	-1

activation $\theta = 1$

Tut-2:

Q No 1. Assume the network shown in figure.



Initial weight vector, $W_i = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$

Input vectors are:- $X[1] = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}$ $X[2] = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ 1.5 \end{bmatrix}$

$X[3] = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$

∵ Take learning constant as 1 0.5

Perform ONE iteration of learning using Hebbian LR when activation functions are:-

(a) Sgn function, $sgn(net_i)$

(b) Continuous bipolar activation function. (bipolar sigmoidal fn)

$= \frac{2}{1+e^{-\lambda net_i}} - 1$ $\lambda = 1$

(i) for sign(neti)

So:

for X_1 · neti = 1 + 2 + 0 + 0
= 3.

Oi = 1 ✓

as sgn(3) = 1

Now we have to update the wt vector

$\Delta w_{i1} = 0.5 \times 1 \times 1$
$= 0.5$

$\Delta w_{i2} = 0.5 \times (-2) \times 1$
$= -1$

$\Delta w_{i3} = 0.5 \times (1.5) \times 1$
$= 0.75$

$\Delta w_{i4} = 0 \times 0.5 \times 1$
$= 0$

Updated wt vector = $\begin{bmatrix} 1+0.5 \\ -1-1 \\ 0+0.75 \\ 0.5+0 \end{bmatrix} = \begin{bmatrix} 1.5 \\ -2 \\ 0.75 \\ 0.5 \end{bmatrix}$

Q2 Perform ONE ITERATION of Perceptron learning rule.

When initial weight vector, W_i:

$$W_i = [1 \quad -1 \quad 0 \quad 0.5]^T$$

Whereas input vectors are :-

$$X_1 = [1 \quad -2 \quad 0 \quad -1]^T$$

$$X_2 = [0 \quad 1.5 \quad -0.5 \quad -1]^T$$

$$X_3 = [-1 \quad 1 \quad 0.5 \quad -1]^T.$$

and desired responses corresponding to

$X_1 \Rightarrow d_1 = +1$

$X_2 \rightarrow d_2 = -1$

$X_3 \rightarrow d_3 = 1$.

Take learning constant $\eta = 0.1$.

Q3/ Perform ONE Iteration of training using delta learning rule when initial wt vector is.

$$W_i = \begin{bmatrix} +1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

Input / Output training vectors are :-

$X_1 = [1 \quad -2 \quad 0 \quad -1] \rightarrow d_1 = -1$.

$X_2 = [0 \quad 1.5 \quad -0.5 \quad -1] \rightarrow d_2 = 1$

$X_3 = [-1 \quad 1 \quad 0.5 \quad -1] \rightarrow d_3 = 1$.

Take constant of learning $\eta = 0.1$.
and bipolar sigmoidal f^n as activation function with gain (λ) of function as 1

$$\boxed{\Delta W_i = \eta (d_i - O_i) \cdot f'(net_i) \cdot X}$$

$$f'(net_i) = \frac{1}{2}\lambda(1 - O_i^2)$$
$$= \frac{1}{2}(1 - O_i^2)$$

i) Take unipolar sigmoidal f^n.

ional) $f'(net_i) = \lambda O_i (1 - O_i)$.

Assignment :-

① Design a Hebbian N/W to implement logical AND function (use bipolar INPUTS and targets)

(ii) OR function. " (Take a bias also)

Take all the connection weights and bias = 0.

Tutorial-2 (Ans)

① (i) For sign(neti)

For X_1 neti = 1+2+0+0 = 3.

sign(neti) = sign(3) = 1.

Now update the wt. vectors.

$\Delta w_{i1} = \eta \cdot O_i \cdot x_{ij}$
$= 0.5 \times 1 \times 1$
$= 0.5$

$\Delta w_{i2} = 0.5 \times 1 \times -2$
$= -1$

$\Delta w_{i3} = 0.5 \times 1 \times (1.5)$
$= 0.75$

$\Delta w_{i4} = 0.5 \times 1 \times 0$
$= 0$

Our updated wt. vector. $\begin{bmatrix} 1+0.5 \\ -1+-1 \\ 0+0.75 \\ 0.5+0 \end{bmatrix} = \begin{bmatrix} 1.5 \\ -2 \\ 0.75 \\ 0.5 \end{bmatrix}$

Now $X_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ 1.5 \end{bmatrix}$ $W_i = \begin{bmatrix} 1.5 \\ -2 \\ 0.75 \\ 0.5 \end{bmatrix}$

$net_i = 1.5 + 1 - 1.5 + 0.75$

$= 1.75.$

$sig(net_i) = O_i = \boxed{1}$

$\Delta w_{i1} = 0.5 \times 1 \times 1$ $\quad$ $\Delta w_{i2} = 0.5 \times 1 \times (0.5)$

$= 0.5$ $\quad$ $= -0.25$

$\Delta w_3 = 0.5 \times 1 \times -2$ $\quad$ $\Delta w_{i4} = 0.5 \times 1 \times 1.5$

$= -1$ $\quad$ $= 0.75.$

Update wt vector $\begin{bmatrix} 1.5 + 0.5 \\ -2 - 0.25 \\ 0.75 - 1 \\ 0.5 + 0.75 \end{bmatrix} = \begin{bmatrix} 2 \\ -2.25 \\ -0.25 \\ 1.25 \end{bmatrix}$

Now $X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$ $W_i = \begin{bmatrix} 2 \\ -2.25 \\ -0.25 \\ 1.25 \end{bmatrix}$

$3 \quad 1.25$

1.5

1.875

$net_i = 0 + (-2.25) + 0.25 + 1.875.$

$= -2 + 1.875$

$= -0.125$

$sig(net_i) = O_i = sign(-0.125)$

$= \boxed{-1}$

$\Delta w_{i1} = 0.5 \times (-1) \times 0 = 0$ $\Delta w_{i3} = 0.5(-1) \times -1 =$

$\Delta w_{i2} = 0.5 \times (-1) \times 1 = -0.5$ $\Delta w_{i4} = 0.5(-1) \times 1.5 = -0.7$

$$W_i = \begin{bmatrix} 0 + (2) \\ -0.5 + (-2.25) \\ 0.5 + (-0.25) \\ -0.75 + (1.25) \end{bmatrix} = \begin{bmatrix} 2 \\ -2.75 \\ 0.25 \\ 0.5 \end{bmatrix}$$

So, after the first iteration, our final wt. vector comes out to be :-

$$W_i = [2 \quad -2.75 \quad 0.25 \quad 0.5]^T$$

→ Ans

(ii) $$O_i = \frac{2}{1 + e^{-\lambda net_i}} - 1$$

Initially, $W_i = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$; $X_1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}$

$X_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ 1.5 \end{bmatrix}$; $X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$

for x_1

$net_i = 1 + 2 + 0 + 0 = 3$

$$O_i = \frac{2}{1 + e^{-3}} - 1$$

$$= 0.905 \approx 0.91$$

$\Delta w_{i1} = 0.91 \times 0.5 \times 1$

$= 0.455$

$\Delta w_{i2} = 0.91 \times 0.5 \times (-2)$

$= -0.910$

$\Delta w_{i3} = 0.91 \times 0.5 \times 1.5$

$= 0.683$

$\Delta w_{i4} = 0.91 \times 0.5 \times 0$

$= 0$

Updated weight vector

$$= \begin{bmatrix} 1+0.455 \\ -1-0.910 \\ 0+0.683 \\ 0+0.5 \end{bmatrix} = \begin{bmatrix} 1.455 \\ -1.910 \\ 0.683 \\ 0.5 \end{bmatrix}$$

for x_2

$$w_i = \begin{bmatrix} 1.455 \\ -1.910 \\ 0.683 \\ 0.5 \end{bmatrix} \quad x_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ 1.5 \end{bmatrix}$$

$net_i = 1.455 + 0.955 - 1.366 + 0.75$

$= 1.794 \approx 1.8$

$$O_i = \frac{2}{1+e^{-\lambda net_i}} - 1$$

$$= \frac{2}{1+e^{-1.794}} - 1$$

$$= 0.71$$

$\Delta w_{ij} = 0.71 \times 0.5 \times 1.455$
$= 0.52$

$\Delta w_{i2} = 0.71 \times 0.5 \times (-1.91)$
$= -0.68$

$\Delta w_{i3} = 0.71 \times 0.5 \times 0.683$
$= 0.355 \times 0.683$
$= 0.24$

$\Delta w_{i4} = 0.71 \times 0.5 \times 0.5$
$= 0.18$

Updated wt. vector $= \begin{bmatrix} 1.455+0.52 \\ -1.91-0.68 \\ 0.683+0.24 \\ 0.5+0.18 \end{bmatrix} = \begin{bmatrix} 1.97 \\ -2.59 \\ 0.92 \\ 0.68 \end{bmatrix}$

For x_3

$x_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$ $w_i = \begin{bmatrix} 1.97 \\ -2.59 \\ 0.92 \\ 0.68 \end{bmatrix}$

$net_i = 0 - 2.59 - 0.92 + 1.02$

$= -2.49$

$O_i = \frac{2}{1 + e^{-\lambda net_i}} - 1$

$= \frac{2}{1 + e^{2.49}} - 1 = -0.85$

$\Delta w_{i1} = 0.5 \times (-0.85) \times 0$

$= 0$

$\Delta w_{i2} = 0.5 \times (-0.85) \times 1$

$= -0.425$

$\Delta w_{i3} = 0.5 (-0.85) (-1)$

$= 0.425$

$\Delta w_{i4} = 0.5 \times (-0.85) (1.5)$

$= -0.64$

Updated wt vector: $\begin{bmatrix} 0 + 1.97 \\ -0.425 - 2.59 \\ 0.425 + 0.92 \\ -0.64 + 0.68 \end{bmatrix} = \begin{bmatrix} 1.97 \\ -3.02 \\ 1.34 \\ 0.04 \end{bmatrix}$ → Ans

So after the first iteration our final wt. vector comes out to be

$$W_i = [1.97 \quad -3.02 \quad 1.34 \quad 0.04]^T$$

↳ Ans

Q2 Initial weight vector,

$$W_i = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

Input weight vectors are :-

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

$d_1 = -1$; $d_2 = -1$; $d_3 = 1$

$\eta = 0.1$

For x_1

$net_i = 1 + 2 + 0 - 0.5 = 2.5$

$f\ sig(2.5) = 1$

and $d_1 = -1$

$$\Delta w_i = c \cdot (d_i - o_i) \cdot X$$

$$= 0.1 \cdot (-1 - 1) \cdot \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= -0.2 \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.2 \\ +0.4 \\ 0 \\ +0.2 \end{bmatrix}$$

$$W_i = \begin{bmatrix} 1 - 0.2 \\ -1 + 0.4 \\ 0 + 0 \\ 0.5 + 0.2 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$$

For x_2

$net_i = 0 + 1.5 \times (-0.6) - 0 - 0.7$

$= -0.9 - 0.7$

$= -1.6$

$sign(-1.6) = -1$

$d_2 = -1$ No learning

$$W_i = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$$

For X_3:

$$net_i = -0.8 - 0.6 + 0 - 0.7$$
$$= -2.1$$

$$sign(net_i) = sign(-2.1) = -1$$

$$\Delta w_i = 0.1 \times (1 - (-1)) \times \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

$$= 0.2 \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.2 \\ 0.2 \\ 0.1 \\ -0.2 \end{bmatrix}$$

$$W_i = \begin{bmatrix} 0.8 - 0.2 \\ -0.6 + 0.2 \\ 0 + 0.1 \\ 0.7 - 0.2 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$$

→ Ans

So, after the first iteration our final weight vector comes out to be
$[0.6 \quad -0.4 \quad 0.1 \quad 0.5]^T$

③ $W_i = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} \quad x_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} \quad d_1 = -1$

$x_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix} \quad d_2 = 1 \qquad x_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} \quad d_3 = 1$

$\eta = 0.1 \quad ; \lambda = 1$

$f'(neti) = \frac{1}{2}\lambda(1 - Oi^2) = \frac{1}{2}(1 - Oi^2)$

$\Delta w_i = \eta (di - Oi) \cdot f'(neti) \cdot x$

$Oi = \frac{2}{1 + e^{-neti}} - 1 = \frac{2}{1 + e^{-neti}} - 1$

For x_1

$$net_i = 1 + 2 + 0 - 0.5$$
$$= 2.5$$

$$O_i = \frac{2}{1 + e^{-2.5}} - 1$$

$$= 0.85$$

$$f'(net_i) = \frac{1}{2}(1 - (0.85)^2)$$
$$= \frac{1}{2}(0.28)$$
$$= 0.14$$

$$\Delta w_i = 0.1(-1 - 0.85) \cdot 0.14 \cdot \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= -0.026 \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.026 \\ -0.052 \\ 0 \\ 0.026 \end{bmatrix}$$

$$= \begin{bmatrix} -0.03 \\ -0.05 \\ 0 \\ 0.03 \end{bmatrix}$$

$$w_i = \begin{bmatrix} 1 & -0.03 \\ -1 & -0.05 \\ 0 & +0 \\ 0.5 & +0.03 \end{bmatrix}$$

$$= \begin{bmatrix} 0.97 \\ -1.05 \\ 0 \\ 0.53 \end{bmatrix}$$

For x_2:

$$net_i = 0 + 1.5 \times (-1.05) - 0 - 0.53$$

$$= -2.11.$$

$$O_i = \frac{2}{1 + e^{+2.11}} - 1$$

$$= -0.78.$$

$$f'(net_i) = \frac{1}{2}\left(1 - (-0.78)^2\right)$$

$$= 0.19.$$

$$\Delta w_i = 0.1 (1 + 0.78) \cdot 0.19 \cdot \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}$$

$$= 0.03 \cdot \begin{bmatrix} 0 & -0.5 \\ 1.5 & -1 \end{bmatrix}$$

$$= 0.03 \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0.045 \\ -0.015 \\ -0.03 \end{bmatrix}$$

$$Wi = \begin{bmatrix} 0.97 + 0 \\ -1.05 + 0.045 \\ 0 - 0.015 \\ 0.53 - 0.03 \end{bmatrix} = \begin{bmatrix} 0.97 \\ -1.00 \\ -0.02 \\ 0.50 \end{bmatrix}$$

For x_3.

$$neti = -0.97 - 1 - 0.01 - 0.5$$

$$= -2.48.$$

$$Oi = \frac{2}{1 + e^{+2.48}} - 1$$

$$= -0.85.$$

$f'(net_i) = \frac{1}{2}(1 - (0.85)^2)$

$= 0.14$

$\Delta w_i = 0.1(1 + 0.85) \cdot 0.14 \cdot \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$

$= 0.03 \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.03 \\ 0.03 \\ 0.015 \\ -0.03 \end{bmatrix}$

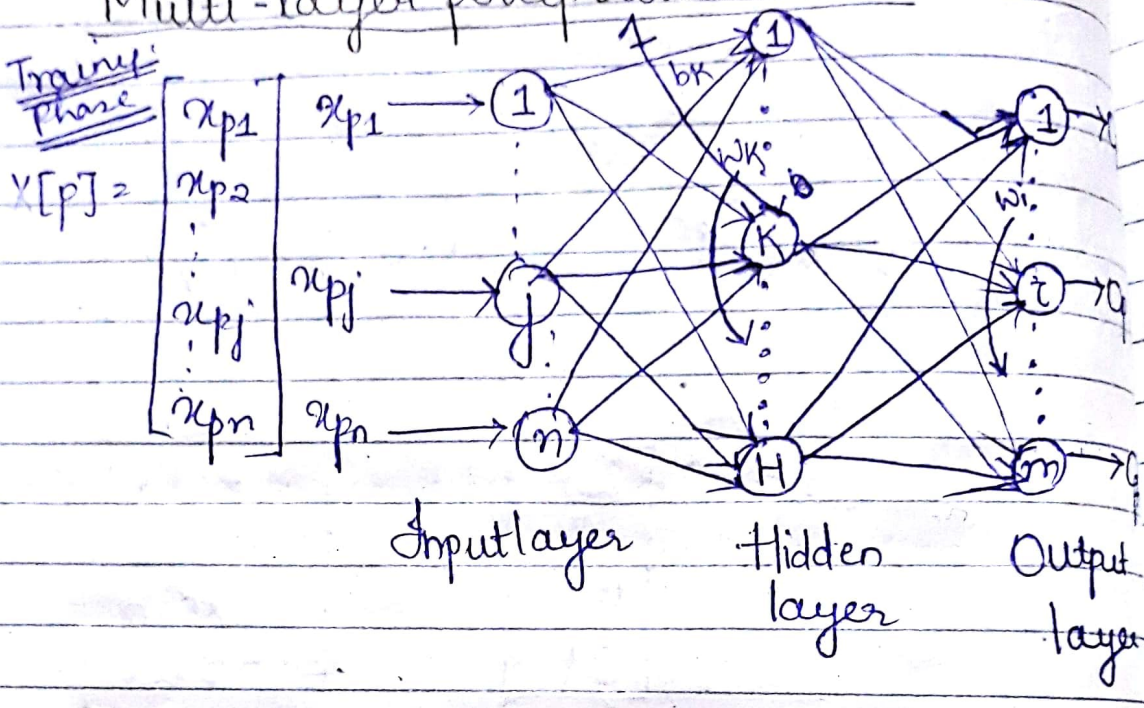
$w_i = \begin{bmatrix} 0.97 - 0.03 \\ -1 + 0.03 \\ -0.02 + 0.015 \\ 0.50 - 0.03 \end{bmatrix} = \begin{bmatrix} 0.94 \\ 0.97 \\ -0.005 \\ 0.47 \end{bmatrix}$ → Ans

So, after the first iteration our final wt. vector comes out to be

$[0.94 \quad 0.97 \quad -0.005 \quad 0.47]^T$ → Ans

27/8/18

Multi-layer perceptron model :-



X[1] → Y[1]

X[2] → Y[2]

X[p] → Y[p]

X[P] → Y[P]

→ Depending on the dimension on the input patter, we have to decide the nodes in the input layer and same applies for output pattern & output layer nodes

→ No:- of neurons & hidden layer depends on the degree of linearity of the problem

Input layer

→ Nodes on the input layer serve as distributors just carrying the signal to the next layer.

$$O_{pj} = x_{pj} \longrightarrow (1)$$

for $j = 1$ to n.

Hidden layer.

→ $$O_{pk} = \frac{1}{1 + e^{-\lambda net_k}} \longrightarrow (2)$$ (Taking unipolar as sigmoidal function).

where $$net_k = \sum_{j=1}^{n} w_{kj} O_{pj} + b_k$$ (b_k - bias for the kth node).

$k = 1$ to H.

Take $\lambda = 1$.

Output layer.

→ $$O_{pi} = \frac{1}{1 + e^{-\lambda net_i}} \longrightarrow (3)$$

where $$net_i = \sum_{k=1}^{H} w_{ik} O_{pk} + b_i$$ (b_i - bias for ith node)

$i = 1$ to m.

→ Weights should lie in the input space. (pick uniformly)

w_{ik} = −1 to 1 | −0.5 to 0.5

w_{hj} = −1 to 1 | −0.5 to 0.5

→ For a given presentation, we have calculated the response at each of the layer.

→ The N/w should be adapt itself such that the computed O/P is close to the desired output.

→ After first input pattern is presented, we have to calculate the error and then update our wts and then go for another input pattern presentation and then we go for multiple iterations.

$$E_p = \frac{1}{2} \sum_{i=1}^{m} (t_{pi} - O_{pi})^2 \quad \rightarrow (4)$$

↳ Error.

This t_{pi} we get from the input-output pattern of a O/P pattern $Y[p]$.

→ Error should be propagated in the backward direction to make adjustment to the weights.
So now when we make again the presentation the computed o/p will be more close to the desired O/P.

$$\rightarrow \frac{\partial E_p}{\partial w_{ik}} = \frac{-1}{2} \cdot 2 (t_{pi} - O_{pi}) \cdot \frac{\partial O_{pi}}{\partial w_{ik}}$$

$$\boxed{\frac{\partial E_p}{\partial w_{ik}} = -(t_{pi} - O_{pi}) \cdot \frac{\partial O_{pi}}{\partial w_{ik}}}$$

$$\boxed{\text{where } \frac{\partial O_{pi}}{\partial w_{ik}} = f'(net_i) \cdot \frac{\partial net_i}{\partial w_{ik}}}$$

$$O_{pi} = f\left[\underbrace{\sum_{k=1}^{H} w_{ik} O_{pk} + b_i}_{net_i}\right]$$

$$\frac{\partial net_i}{\partial w_{ik}} = O_{pk}.$$

$$\frac{\partial O_{pi}}{\partial w_{ik}} = f'(net_i) \cdot O_{pk}$$

$$\frac{\partial E_p}{\partial w_{ik}} = -(t_{pi} - O_{pi}) \cdot f'(net_i) \cdot O_{pk}$$

↳ Error gradient. ↳ (5)

$$\Delta w_{ik}(p) = -\eta \frac{\partial E_p}{\partial w_{ik}} \quad \text{—— (6)}$$

$$\Delta w_{ik}(p+1) = \eta (t_{pi} - O_{pi}) \cdot f'(net_i) \cdot O_{pk}.$$

↳ (7) (We are doing this for ~~ith~~ kth node).

$$f'(net_i) = O_{pi}(1 - O_{pi}) \quad \text{for unipolar sigmoidal.}$$

↳ (8)

$$\Delta w_{ik}(p) = \eta (t_{pi} - O_{pi})\, O_{pi}(1 - O_{pi}) \cdot O_{pk}$$

$k = 1$ to H ; $i = 1$ to m. ↳ (9)

→ If the signal is strong; it is influencing the error to a larger extent

Δwik depends on both
→ Strength of the signal and error.

→ It is directly proportional to strength of signal and error; when error is more or strength of signal is more; we have to make the change.

→ Opk → strength of the signal.

→ If there is a load on the output side and if it has to be referred to the input side (it depends on transformer ratio).

~~impedance~~ / impedance

$$Z_p = \left(\frac{N_1}{N_2}\right)^2 Z_s$$

→ load on the o/p side

→ refered to the input side.

So here, $\eta\,(t_{pi} - O_{pi})\,O_{pi}(1 - O_{pi})$

→ error referred to the input side.

pk

(9)

$$w_{ik}(p+1) = w_{ik}(p) + \Delta w_{ik}(p)$$

$$w_{ik}(p+1) = w_{ik}(p) + \eta (t_{pi} - O_{pi}) O_{pi} (1 - O_{pi}) O_{pk} \quad \rightarrow (10)$$

for $k = 1$ to H

$i = 1$ to m.

$$w_{ik}(p+1) = w_{ik}(p) + \eta \cdot S_{pi} \cdot O_{pk} \quad \rightarrow (11)$$

So now our wts between hidden layer and output layer is being updated ; we have to update the wt b/w input layer and hidden layer.

* Weight b/w input and hidden layer

$$\Delta w_{kj}(p) = \eta \cdot S_{pk} \cdot O_{pj} \quad \rightarrow (12)$$

where $S_{pk} = \left(\sum_{i=1}^{m} S_{pi} \cdot w_{ik} \right) \cdot O_{pk} (1 - O_{pk})$

error at the O/P layer multiplied by the synaptic strength for all the O/P nodes

output when refered as input

28/8/18

$$\Delta w_{kj}(p) = \eta \cdot \delta_{pk} \cdot O_{pj}$$

for $j = 1$ to n

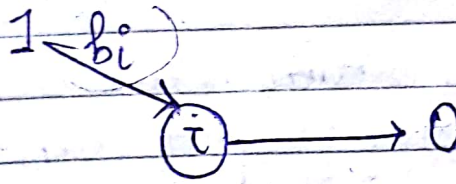
$k = 1$ to H.

$$\boxed{w_{kj}(p+1) = w_{kj}(p) + \eta \delta_{pk} \cdot O_{pj}} \rightarrow (13)$$

$j = 1$ to n

$K = 1$ to H.

2/8/18



$$\Delta b_i = \eta \cdot \delta_{pi} \cdot 1 \rightarrow (14)$$

$$\boxed{\Delta b_i = \eta \cdot (t_{pi} - O_{pi}) O_{pi} (1 - O_{pi}) \cdot 1}$$

$$b_i(p+1) = b_i(p) + \eta (t_{pi} - O_{pi}) O_{pi} (1 - O_{pi})$$

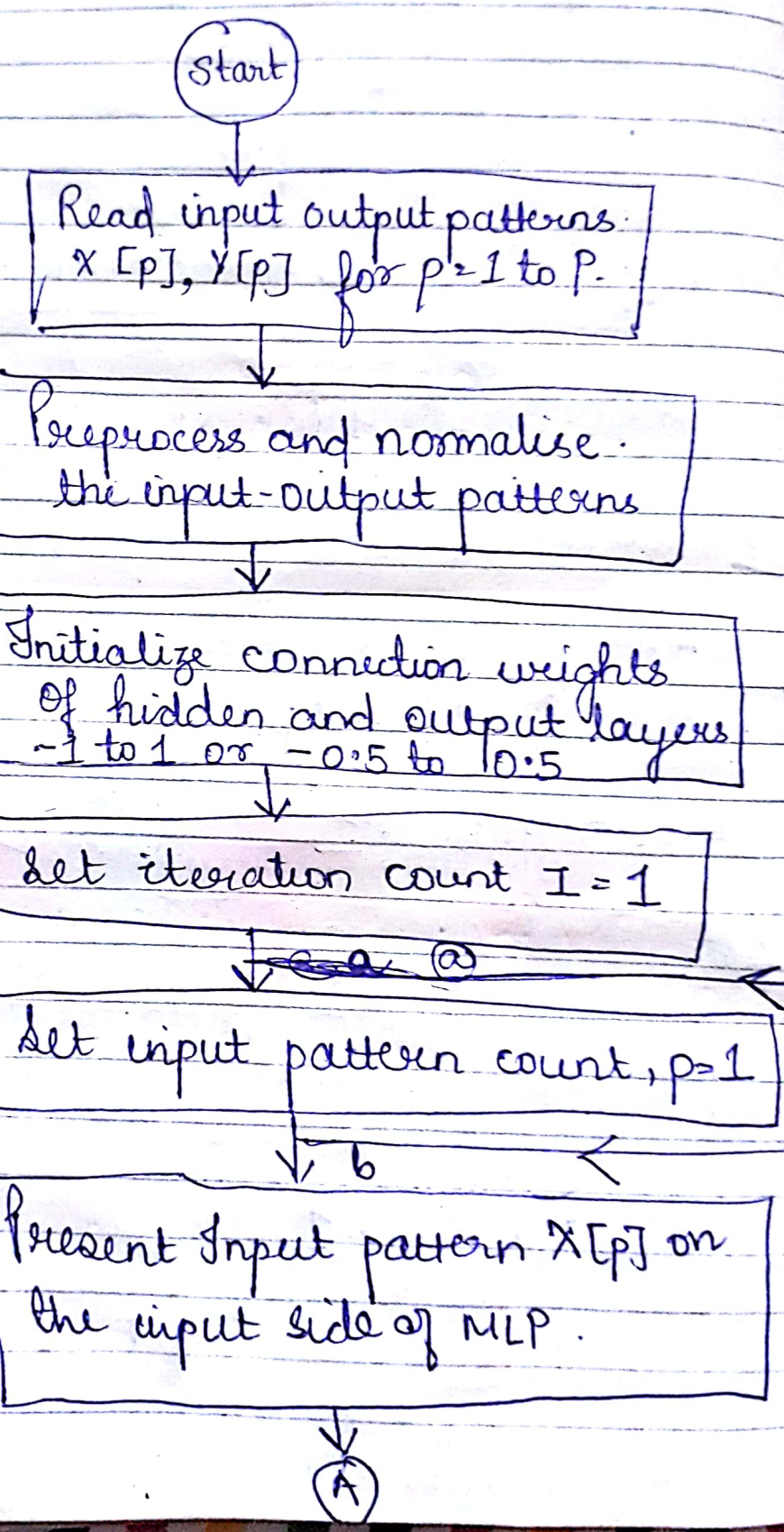
$i = 1$ to m.

$$\Delta b_k = \eta \cdot \delta_{pk} \cdot 1$$

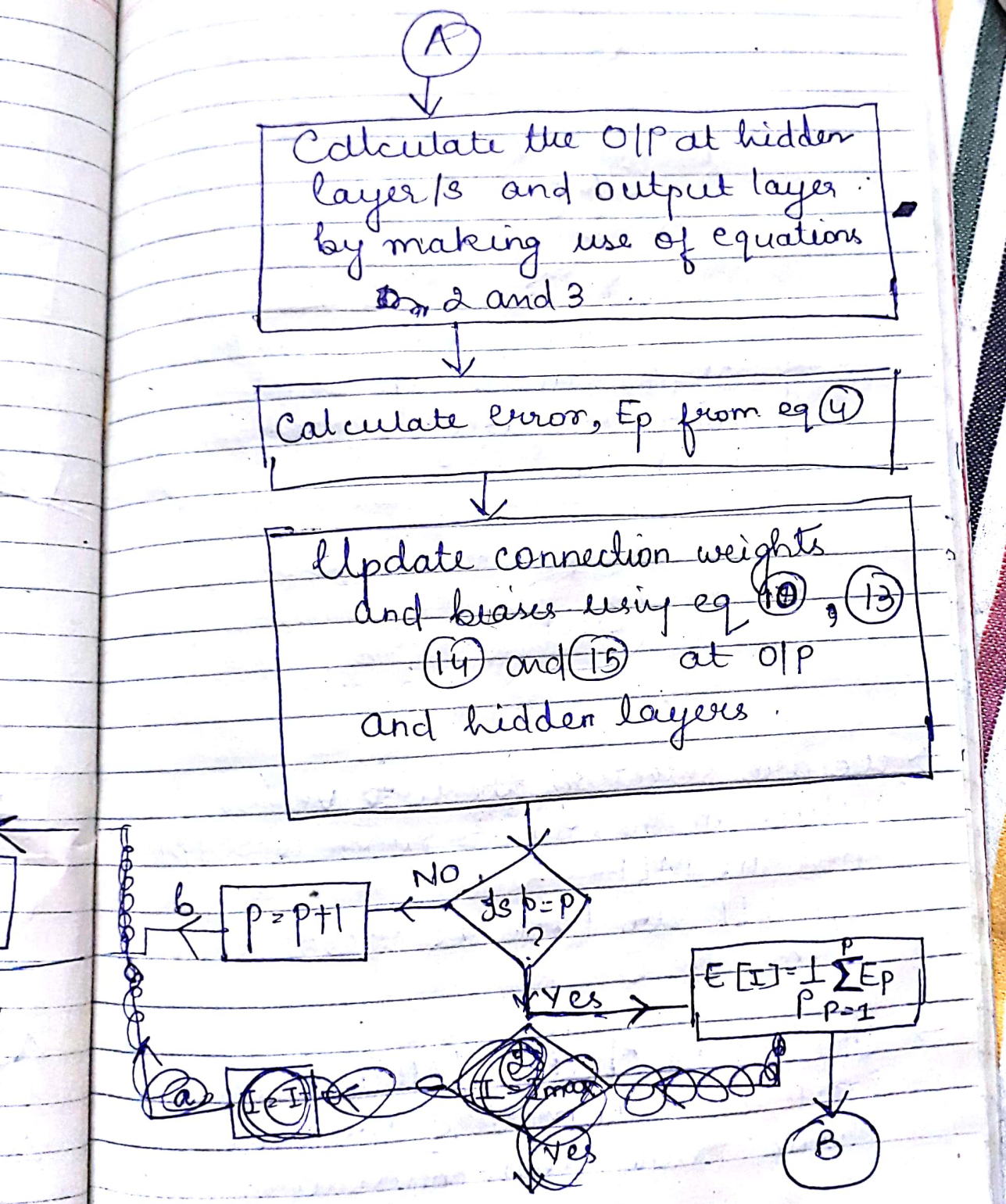
$$\Delta b_k = \eta \left(\sum_{i=1}^{m} w_{ik} \cdot \delta_{pi} \right) \cdot O_{pk} (1 - O_{pk}) \rightarrow (15)$$

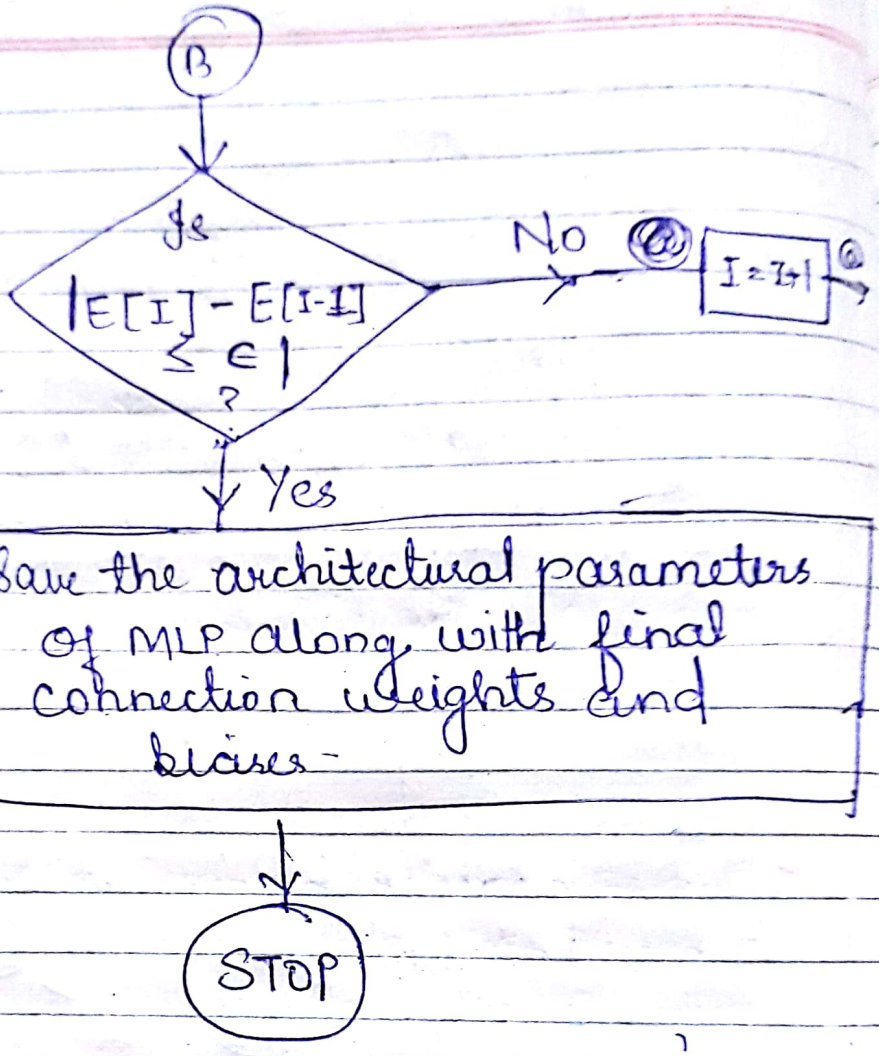
$k = 1$ to H.

Training of MLP. (Steps).



If we take wts very small ; then we need to more steps to come closer to the o/p If we take very large wts then flexibility of training will be very less.





→ We are selecting the wt vector values in the input range is SOFM not in MLP.
In MLP we take wt vectores close to 0.

→ The error checking / saturation criteria is concerned we can have many such conditions.

One is which we have used, the other can be taking some minimum value of error (E) (but the drawback is this is we don't know error min previously, so it is not a good method).

The another method can be taking average of five consecutive E[I] and then compare it with the the stored error. (so there can be many methods for saturation criteria).

$\Delta w_{ik} = \eta \cdot \delta_{pi} \cdot O_{pk}$.

Where $\delta_{pi} = (t_{pi} - O_{pi}) O_{pi}(1 - O_{pi})$

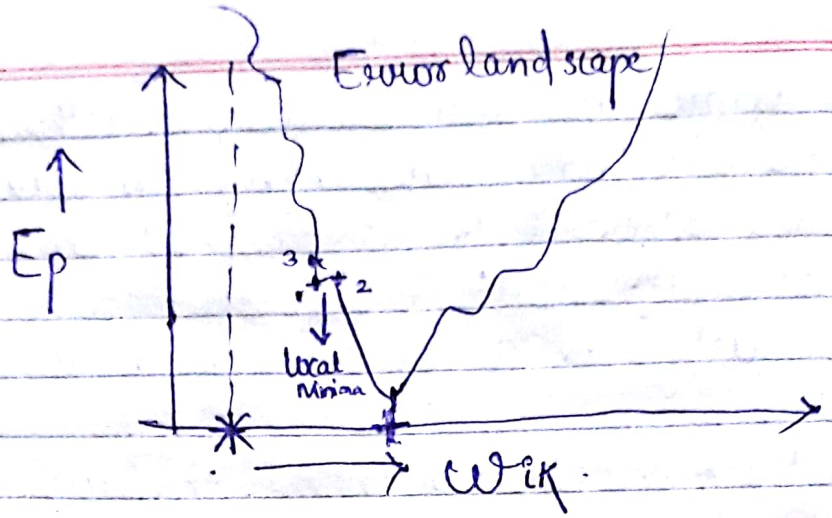
change

$\Delta w_{ik}(p) = \eta \cdot \delta_{pi} \cdot O_{pk} + \alpha \Delta w_{ik}(p-1)$.

α = momentum constant. (0 to 1) (0.1 to 0.9).

This change in formula helps us to keep a track of change in wt with previous presentation wt.

30/8/18



* → start of w.
+ → error is minimum (so we get Optimum wt).

→ Here 1 is trapped between 3 and 2. As we move a little step from 1 and 2 the error will increase, so we will decrease the weight.
→ Now when we decrease the wt and moves from 1 to 3, error again increase.
So 1 is trapped and will oscillate between 3 and 2.
So here there is no concept of local minima.

10/3/18

→ If the gradient in the same direction, steps are larger

→ As by adding the momentum constant, as we are going in the same gradient / slope, α becomes more & more and when we reach the trapping state it will get add up to the previous correct & the step will be larger & that point (2) will be skipped.

→ By adding this term, we are able to overcome the trapping in local minima and we will land somewhere in global minima.

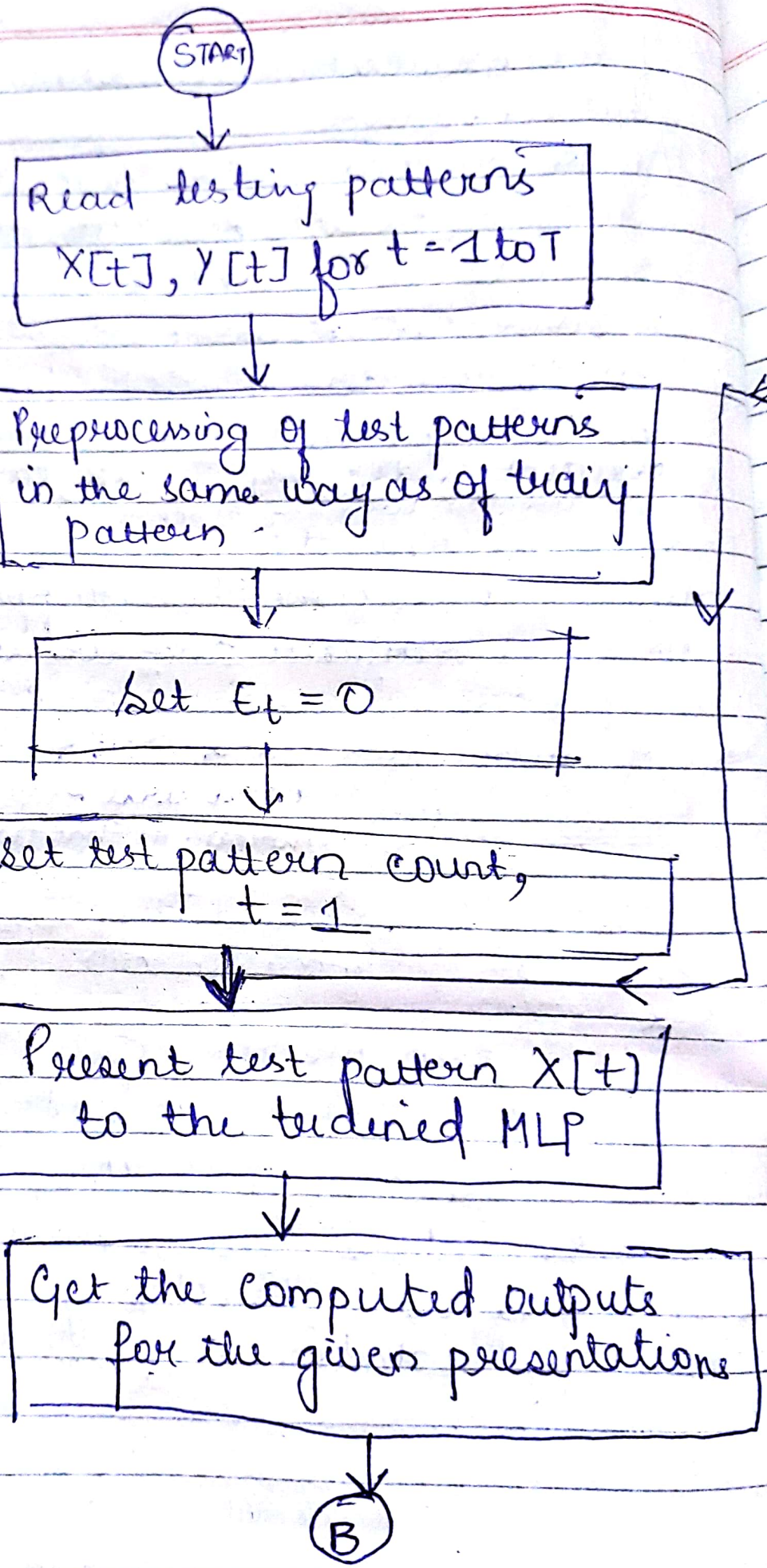
But if global minima is close to local minima, then it will trap in local minima. At that time we will have to change the I/P pattern which will change our Error lattice.

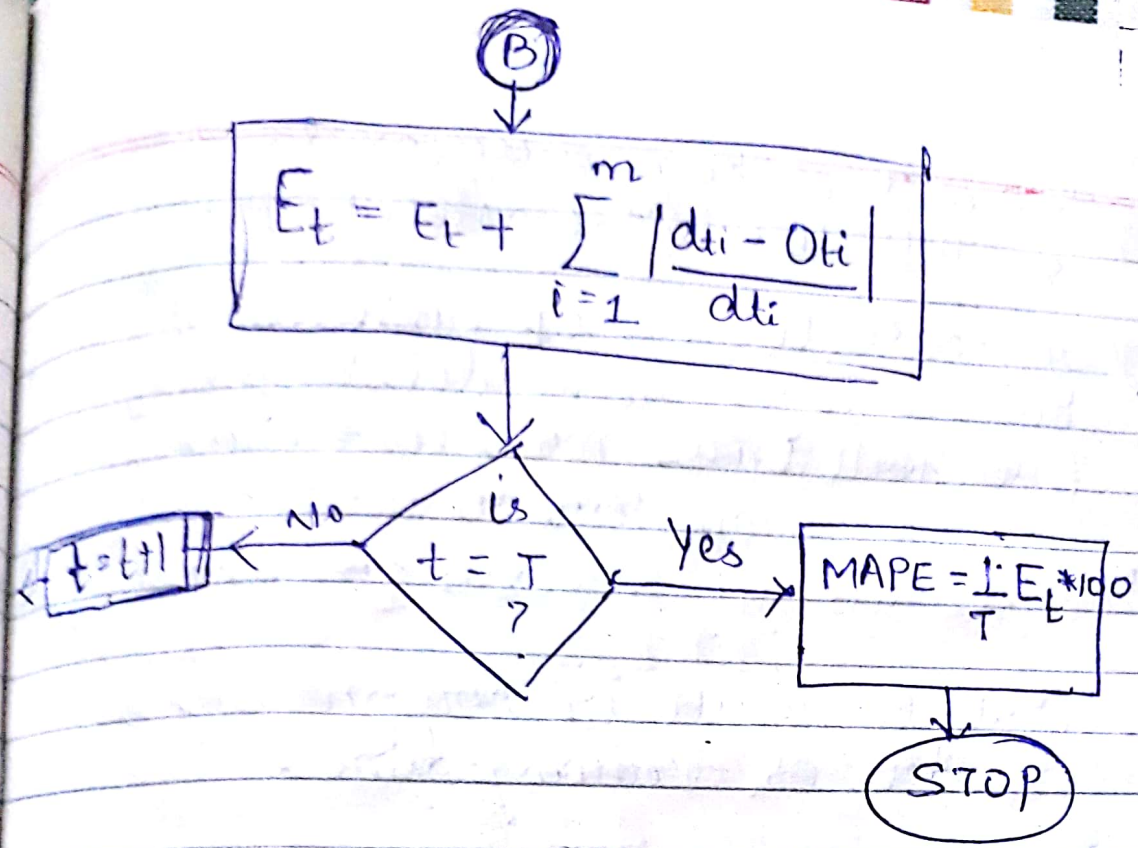
Testing phase

→ Present the presentations to the traing architecture. We got and then we will get O/P

O/P will be compared with the desired output (Here no correction will be made.

Then we will directly go for the next presentation.





Mean absolute percentage error (MAPE) when it is higher side we will again have to do some tuning. For tuning we can do the following:-

1. Training pattern will increase (It will be in a better position which may cover the entire operating pattern). So that accuracy will increase.

→ A stage will come when adding more I/P pattern; there is no significant change in accuracy so we will stop there.

2. → ~~Taking~~ all important features will be incorporated in I/P feature

This will make our error less or our O/P close to the desired O/P.

→ Increase the no. of nodes in the hidden layer or hidden layer (we will have more controllable parameters so that our computational O/P is close to desired O/P)

But it result in computational burden or training time.

→ The size of the model should fit the non-linearity problem.

If underfit then accuracy less.

If over fit or over size then generalisation will be poorer.

⇒ So optimum size of the ANN model should be picked up so that it should commensurate to the non-linear problem.

So the entire Input Output pattern is divided in 3 parts

Training (60%) Testing (20%) Validation (20%)

In validation process

→ On the completion of the particular iteration / every iteration, whatever the wts and biases are there, we will feed this validation pattern and check the error.

→ Then we will go the next iteration, same I/P O/P pattern is presentation, and same validation pattern is presented and error is checked.

→ As we go on further error will keep on decreasing.
A point will come when validation error will start increasing, that is the optimum time to terminate the training process.

→ One accept of termination criteria is take fixed no. of epochs.

→ Other accept of termination criteria is to check the error, and when there is no change in error or very less than error change.

→ 3rd accept is this validation process.

Tut-2

① Hebbian N/w for logical AND.

AND

x_1	x_2	b	T
-1	-1	1	-1
-1	1	1	-1
1	-1	1	-1
1	1	1	1

$Wi = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ (Initial wt vector)

$X[1] = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$ $X[2] = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$ $X3 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$

$X4 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ $\alpha = 1$

for x1

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

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

$W_2 = W_1 + \Delta w$

$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$

For x2

$w_3 = w_2 + \Delta w$

$\Delta w = -1 \times \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$

$w_3 = \begin{bmatrix} 1+1 \\ 1-1 \\ -1-1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix}$

For x3

$w_4 = w_3 + \Delta w$

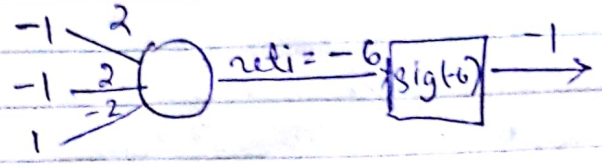
$= \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ -3 \end{bmatrix}$

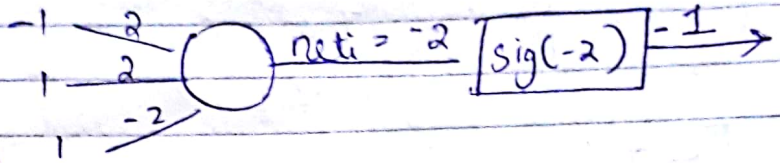
For x4

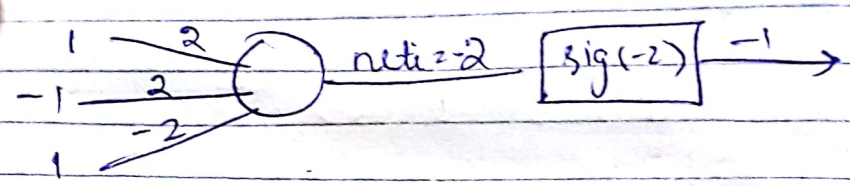
$w_5 = w_4 + \Delta w$

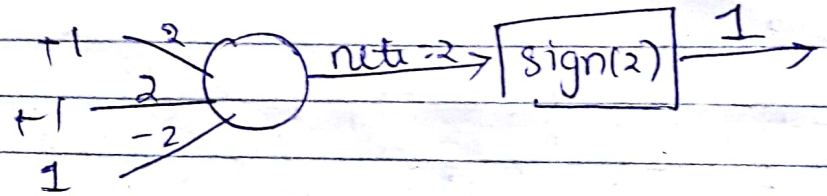
$= \begin{bmatrix} 1 \\ 1 \\ -3 \end{bmatrix} + \begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ -2 \end{bmatrix}$

Now for this wts lets see the calculated o/p.

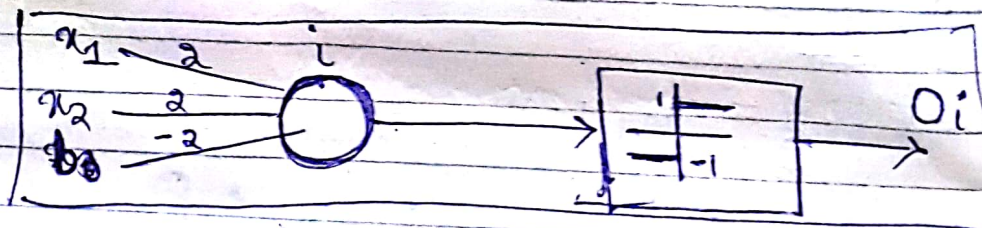








So, Hebbian network for logical AND is :-



(ii) OR function :

X_1	X_2	B	T
-1	-1	1	-1
-1	1	1	1
1	-1	1	1
1	1	1	1

$$W_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

For X_1.

$$w_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + -1 \times \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

For X_2.

$$w_3 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 1 \times \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

For X_3

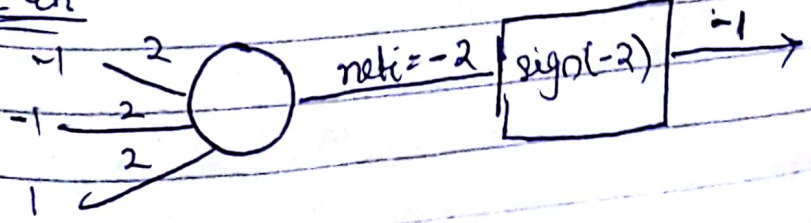
$$w_4 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} + 1 \times \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

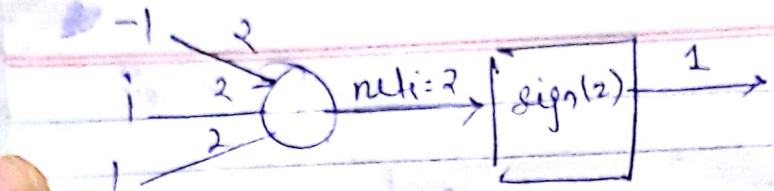
For X_4

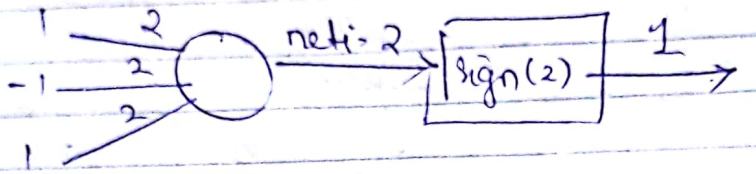
$$w_5 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + 1 \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

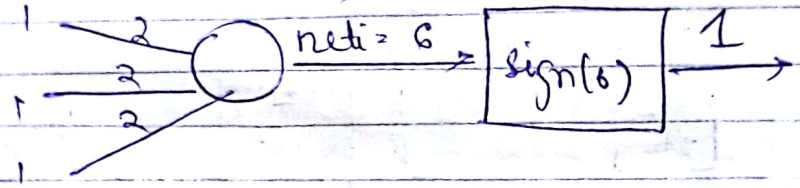
The final weights are = $[2 \quad 2 \quad 2]^T$.

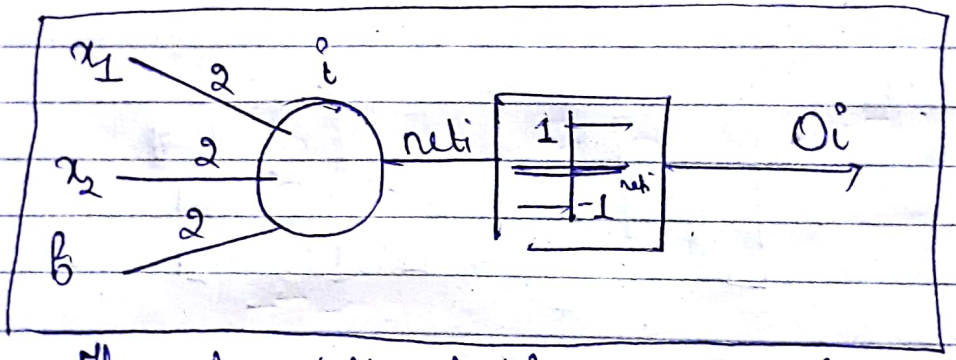
Check











The above is the hebbian network of logical OR.

— x —

31/8/18

Tutorial - 3

Q Write a code for implementing BPN (Back propagation N/W) for training a single hidden layer back propagation network with bipolar sigmbidal function ($\lambda = 1$) to achieve the following two-in-one mappings.

(a) $y = 6\sin(\pi x_1) + \cos(\pi x_2)$

(b) $y = \sin(\pi x_1) \cdot \cos(0.2\pi x_2)$

Set up two sets of data, each consisting of 10 input-output pairs, one for training and other for testing. The input-output data are obtained by varying input variables (x_1, x_2) within $[-1, +1]$ randomly. Also the output data are normalized within $[-1, +1]$.

(Start with 2 nodes in HL).

① Generate and calculate input/output patterns for training and testing

② Initialize connection wts. b/w (−0.5 to 0.5) given.

③ Make 5 patterns to use for validat...

④ Do normalization of O/P data. b/w [-1 1]

$$\left(\frac{y - y_{min}}{y_{max} - y_{min}}\right) * 2 - 1$$

Q2. Construct a kohonen self-organizing map to cluster the four given vectors.

$X_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$, $X_2 = [1\ 0\ 0\ 0]^T$,

$X_3 = [0\ 1\ 1\ 0]^T$

and $X_4 = [0\ 0\ 0\ 1]^T$.

The number of clusters to be formed is two.

Assume and initial learning rate of 0.5.

Initial wt vectors are :-

$W_1 = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.6 \\ 0.8 \end{bmatrix}$ $W_2 = \begin{bmatrix} 0.9 \\ 0.7 \\ 0.5 \\ 0.3 \end{bmatrix}$

Find final connection weights.

Q3.

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

(The o/p dimen = 5)

For the given KSOFM, use the square of Euclidian distance to find the cluster unit, j closest to input vector $[0.2 \ 0.4]^T$. Given learning coast, $\eta = 0.2$. Find the new weight for winner unit, j.

Also for the input vector $[0.6 \ 0.6]^T$ with $\eta = 0.1$. Find the winner cluster unit and its new weights.

* Another way around is calc. the unit length vector. (see back).