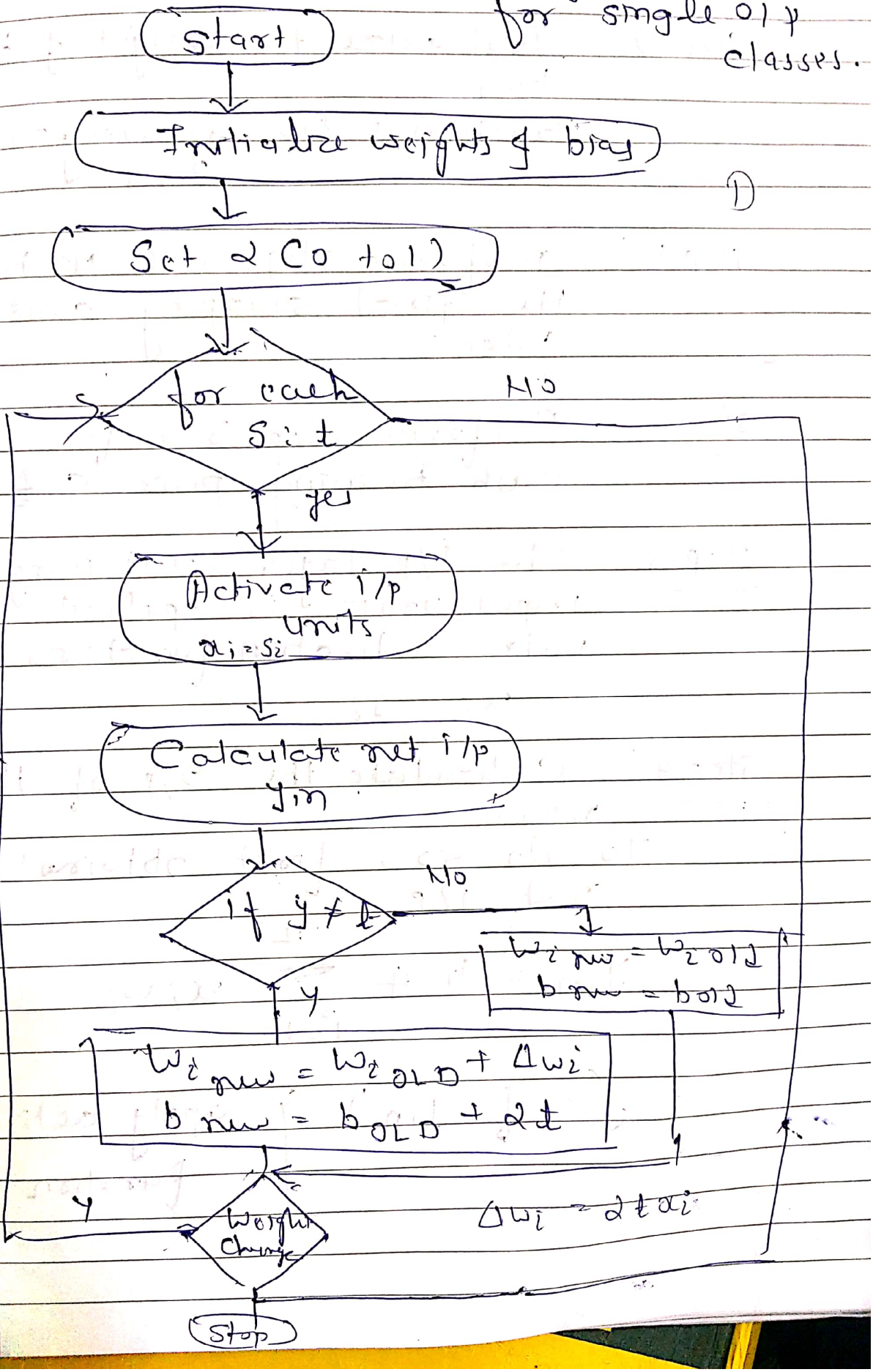
(*) Perceptron ~~learning~~ Training Algorithm for single o/p classes.



Algorithm :-

Step 0 : Initialize the weight & the bias.

Initialize the learning rate.

$\alpha = 1$

Step 1 :- perform step 2-6 until the final stopping cond^n false.

Step 2 : Perform step 3-5 for each training pair S:t

Step 3 : The i/p layer containing input units is applied with activation functions

$x_i = s_i$

Step 4 : Calculate the o/p of the n/w.

To do so, first obtain the net i/p.

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

$y = f(y_{in})$ } apply activation function

Step 5 Weight & bias adjustment

Compare the value of actual (calculated) & targeted output

if $y \neq t$ then

$$w_{i\,new} = w_{i\,old} + \alpha (t) x_i$$

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

else

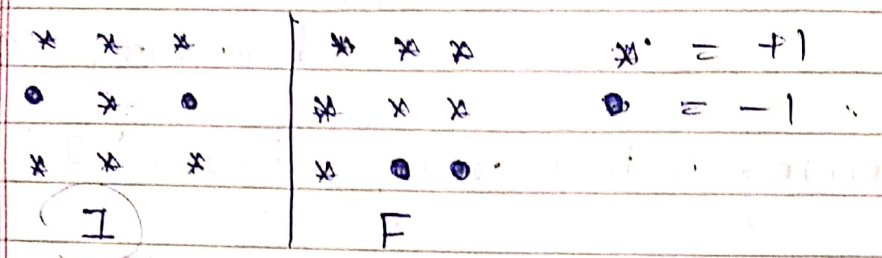
$$w_{i\,(new)} = w_{i\,old}$$

$$b_{new} = b_{old}.$$

Step 6 Train the n/w until there is no weight change. This is the stopping condn for n/w else again start from step 2

$y_{in} = 0$

Ex:-	Pattern	Input x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	Output Target
	I	1	1	1	-1	1	-1	1	1	1	1
	E	1	1	1	1	1	1	1	-1	-1	-1



$w_1 = 0$ to $w_9 = 0$ (Assume)

→ Activation function

$$y = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

→ for input sample I,

$$y_{in} = \sum_{i=1}^{9} x_i w_{ji}$$

$$= 1\times0 + 1\times0 + 1\times0 + (-1)\times0 + 1\times0 + (-1)\times0 + 1\times0 + 1\times0 + 1\times0$$
$$= 0$$

$y_{in} = 0$.

→ Calculate new weight

$$w_{ji\,new} = w_{ji\,old} + \Delta w$$

$$\Delta w = \alpha(t - y_{in})\, x_i, \quad \text{here } \alpha = 1$$
$$= \alpha(t - 0)\, x_i = t\, x_i$$

$w_{1new} = w_{1old} + 1 * 1$

$= 0 + 1 * 1 = 1$

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

$w_{3new} = 0 + 1(1) = 1$

$w_{4new} = 0 + 1(-1) = -1$

$w_{5new} = 0 + 1(1) = 1$

$w_{6new} = 0 + 1(-1) = -1$

$w_{7new} = 0 + 1(+1) = 1$

$w_{8new} = 0 + 1(1) = 1$

$w_{9new} = 0 + 1(1) = 1$

Weight Input Sample = [1 1 1 -1 1 -1 1 1 1]

→ for 2nd i/p F

$$y_{in} = \sum_{i=1}^{n} w_{ji} x_i$$

$= 1x1 + 1x1 + 1x1 + 1x(-1) + 1x1$
$+ -1x1 + 1x1 + 1x(-1) + 1x(-1)$

$= 5 - 4 = 1$

$y = f(y_{in}) = 1$ (as $y_{in} > 0$) (1 ≠ 1)

→ Since t ≠ y find new weights.

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

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

$w_{3new} = 1 + (-1)(1) = 0$

$w_{4new} = (-1) + (-1)(1) = -2$

$b = 1 + (1)(-1) = 0$

$W_{5new} = 1 + (-1) * 1 = 0$

$W_{6new} = -1 + (-1) * 1 = -2$

$W_{7new} = 1 + (-1) * 1 = 0$

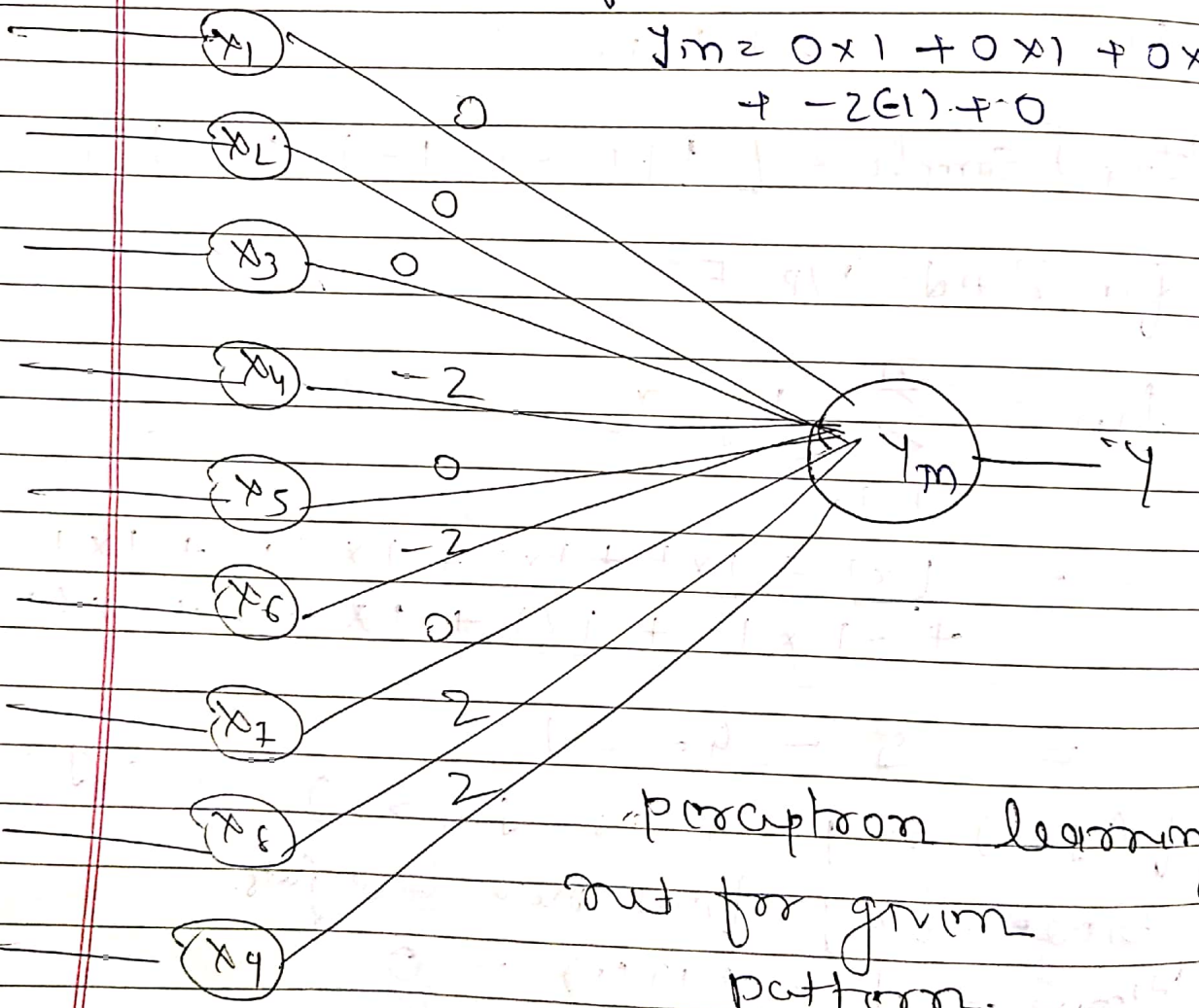
$W_{8new} = 1 + (-1) * (-1) = 2$

$W_{9new} = 1 + (-1) * (-1) = 2$

$W = [0\ 0\ 0\ -2\ 0\ -2\ 0\ 2\ 2]$

$\rightarrow$ for pattern I

$y_{in} = 0 \times 1 + 0 \times 1 + 0 \times 1 + -2(-1) + 0$



perceptron learning rule for given pattern.

ADALINE :-

Adaptive Linear neuron.

→ The units with linear activation functions are called linear units.

→ A nw with single linear unit is called an Adaline.
i.e. here i/p o/p relationship is linear

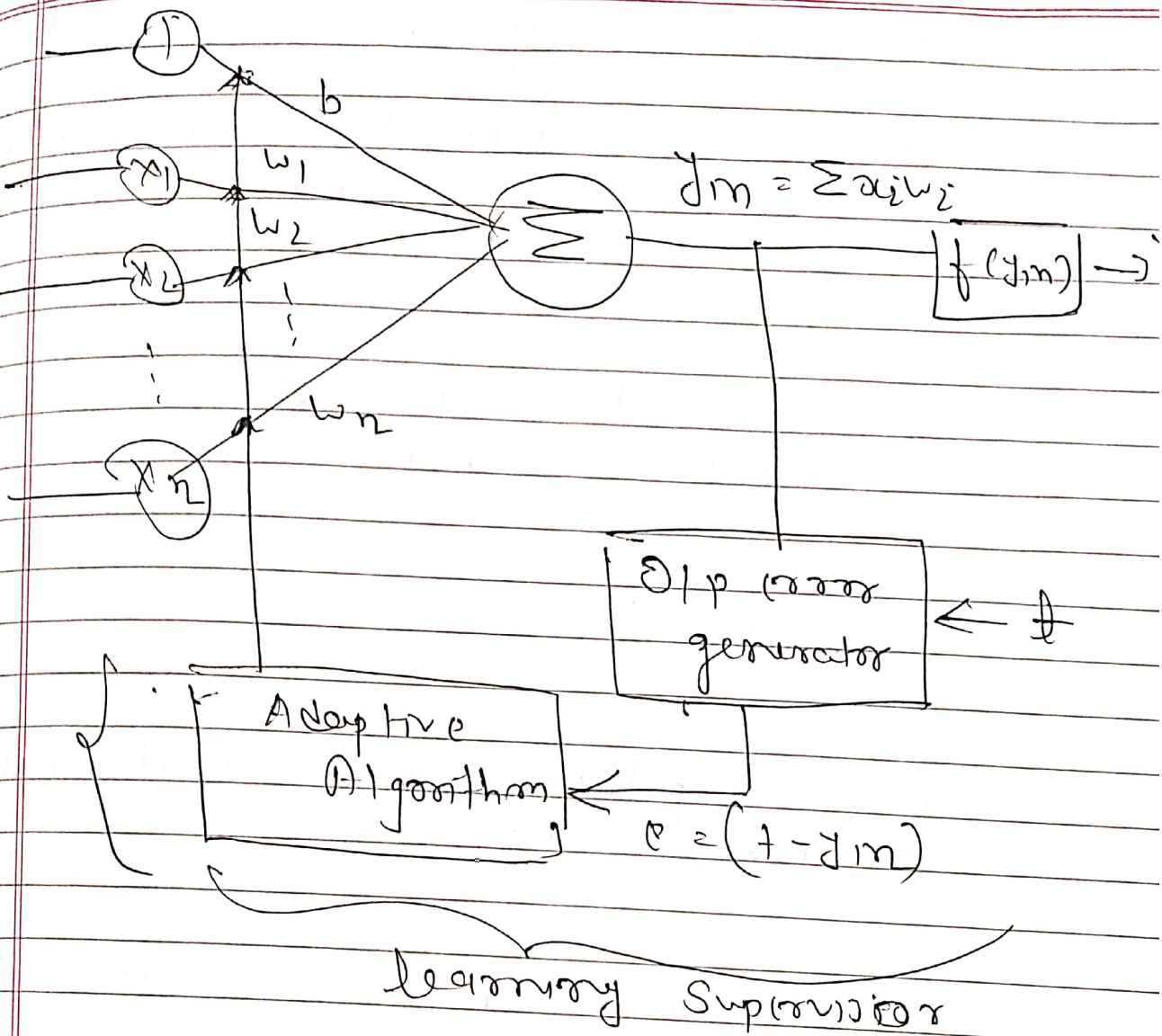
→ trained by using delta rule or Widrow-Hoff rule

i.e. $\Delta w_i = \alpha (t - y_{in}) x_i$

→ this rule is found to minimize the mean square error between the activation & the target value.

→ The delta rule in case of several outputs units

$$\Delta w_{ij} = \alpha (t - y_{in}) x_i$$



→ Algorithm (Training Algorithm)

Step-0 Weights and bias are set to some random values but non zero

→ Set the learning rate parameter α

Step-1 Perform step 2-6 when stopping condition is false.

Step-2 Perform step 3-5 for each

bipolar training pair s:t

Step 3: Set activation for input units $i = 1$ to n.

$$a_i = s_i$$

Step 4: Calculate net i/p to the o/p unit.

$$y_{in} = b + \sum_{i=1}^{n} a_i w_i$$

Step 5: Update weights and bias for $i = 1$ to n:

$$w_{i(new)} = w_{i(old)} + \alpha (t - y_{in}) a_i$$

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

Step-6: If the highest weight change that occur during training is smaller than a specified tolerance then stop the training process else continue.
This is the test for a stopping condition in network.

⇒ Adaline testing Algorithm

→ Adaline is used to classify the i/p new patterns.

→ A step function is used to test the performance of the new

→ Procedure is as follows.

Step 0: Initialize weights.

step 1: Perform step 2-4 for each bipolar input vector x.

Step 2: Set the activations of the input units to x.

step 3: Calculate net I/p to the O/p unit.

$$y_{in} = b + \sum x_i w_i$$

Step 4: Apply the activation function over the net input calculated.

$$y = \begin{cases} 1 & \text{if } y_{in} \geq 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

b = -θ

Activation function

Without bias

$y = f(y_{in})$

$$= \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } y_{in} = \theta \\ -1 & \text{if } y_{in} < \theta \end{cases}$$

$$y_{in} = \sum_{i=1}^{n} a_i w_i$$

where θ = threshold for the given input function.

⇒ if you add bias b here. it will become 0.

With bias.

$y = f(y_{in})$

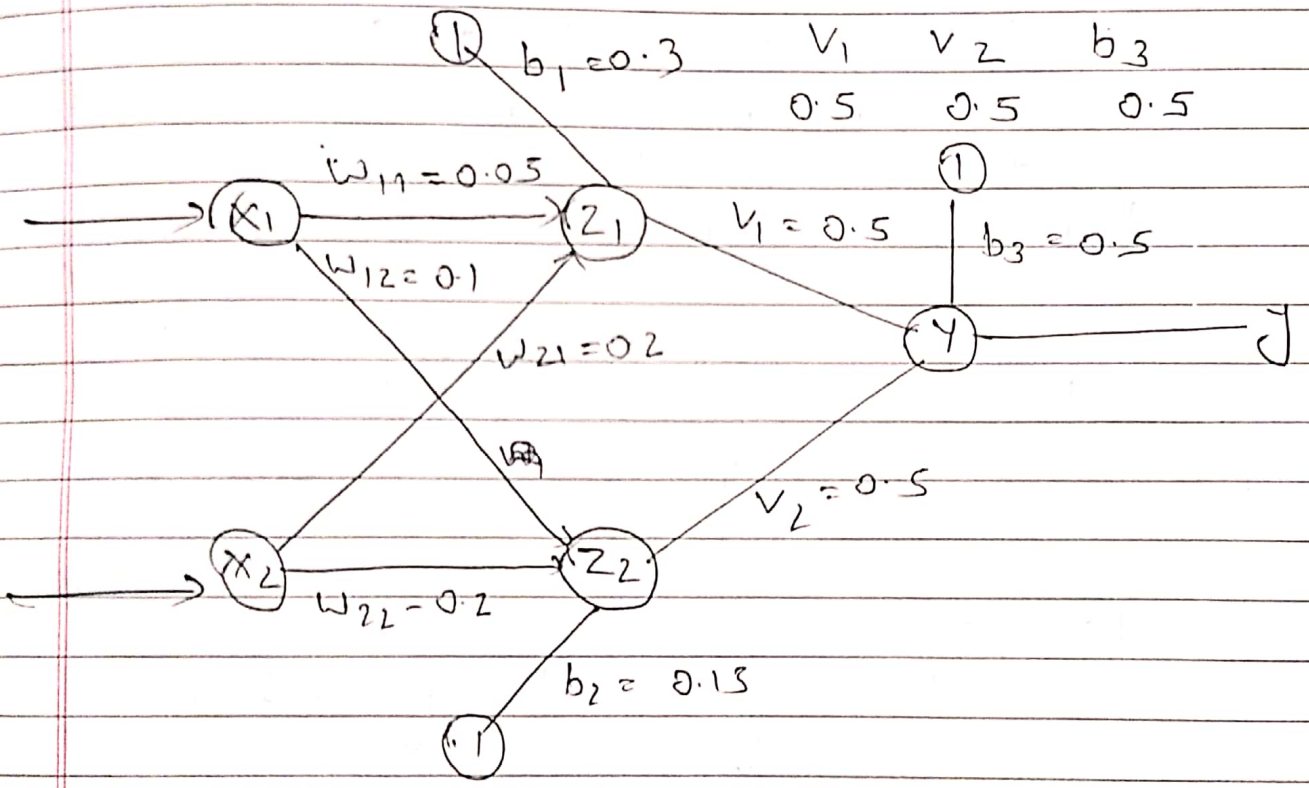
$$= \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

$$y_{in} = b + \sum_{i=1}^{n} a_i w_i$$

Here threshold $\theta = 0$ as we are adding bias to the y_{in} (net input)

Ex: Using Madaline Network, implement XOR function with bipolar inputs & targets. Assume required parameters for training nw. $\alpha = 0.5$

x_1	x_2	I	t	w_{11}	w_{21}	b_1
1	1	1	-1	0.05	0.2	0.3
1	-1	1	1			
-1	1	1	1	w_{12}	w_{22}	b_2
-1	-1	1	-1	0.1	0.2	0.15



⇒ For Input $(a_1, a_2) = (1, 1)$ & target $t = -1$ & $\alpha = 0.5$

① → Calculate net input to hidden units z_{in1} & z_{in2}.

$$Z_{in_1} = b_1 + a_1 w_{11} + a_2 w_{21}$$

$$= 0.3 + (1 \times 0.05) + (1 \times 0.2)$$

$$\boxed{Z_{in_1} = 0.55}$$

$$Z_{in_2} = b_2 + a_2 w_{12} + a_1 w_{22}$$

$$= 0.15 + (1 \times 0.1) + (1 \times 0.2)$$

$$\boxed{Z_{in_2} = 0.45}$$

(2) ⇒ Calculate output Z_1 & Z_2 by applying activation function

the activation function is given by

$$f(z_{in}) = \begin{cases} 1 & \text{if } z_{in} \geq 0 \\ -1 & \text{if } z_{in} < 0 \end{cases}$$

Hence

$$Z_1 = f(z_{in1}) = f(0.55) = 1$$

$$Z_2 = f(z_{in2}) = f(0.45) = 1$$

(3) ⇒ Calculate net input into the output unit.

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

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

⇒ Apply activation function over the net input y_{in} to calculate y.

$$y = f(y_{in}) = f(1.5) = 1$$

(4) → Since $t \neq y$ weight updation has to be performed.

→ Since $t = -1$, weights updation will be done on positive net i/p. & z_{in1} & z_{in2} both are positive, hence weights will be updated on both with i/p.

$$w_{11(new)} = w_{11(old)} + \alpha(t - z_{in1})x_1$$

$$= 0.05 + 0.5(-1 - 0.55) \times 1$$

$$\boxed{w_{11new} = -0.725}$$

$$w_{12(new)} = w_{12(old)} + \alpha(t - z_{in2})x_1$$

$$= 0.1 + 0.5(-1 - 0.45) \times 1$$

$$\boxed{w_{12new} = -0.625}$$

$$b_{1(new)} = b_{1(old)} + \alpha (t - z_{in_1})$$
$$= 0.3 + 0.5 (-1 - 0.55)$$

$$\boxed{b_{1new} = -0.475}$$

$$\Rightarrow W_{21_{new}} = W_{21(old)} + \alpha (t - z_{in_1}) a_2$$

$$= 0.2 + 0.5 (-1 - 0.55) \times 1$$

$$\boxed{W_{21new} = -0.575}$$

$$\Rightarrow W_{22new} = W_{22old} + \alpha (t - z_{in_2}) a_2$$

$$= 0.2 + 0.5 (-1 - 0.45) \times 1$$

$$\boxed{W_{22new} = -0.525}$$

$$b_{2(new)} = b_{2old} + \alpha (t - z_{in_2})$$

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

$$\boxed{b_{2new} = -0.575}$$

⑤ Back propagation algorithm :

→ Used for multilayer feed forward network consisting of processing elements with continuous differentiable activation functions.

→ The n/w associated with back propagation algorithm is called back propagation n/w.

→ Used to classify multiple patterns correctly.

→ Gradient - Descent method is used for weight updation

→ Notations.

i = input layer. (where $i = 1$ to n)

j = Hidden layer (where $j = 1$ to n_0)

k = Output layer (where $k = 1$ to N)

V_{jk} = Weights from Hidden to O/p layer.

b_k = bias applied to output layer.

w_{ij} = Weights from input to hidden layer.

b_j = bias applied to hidden layer.

z_{inj} = net input from i/p to hidden layer

y_{ink} = net input from hidden to o/p layer

Z_j = O/p of hidden layer.

Y_k = O/p of O/p layer

→ BPN done in 3 stages.

① Feedforward of the input training pattern.

② Calculation & back propogation of Error. (Back propogations)

③ Updation of weights.

⇒ Phase I Feedforward

① Each i/p unit receives i/p signal x_i & sends it to the hidden unit.

② Each hidden unit Z_j sums it weighted i/p to calculate net i/p.

$$Z_{inj} = b + \sum_{i=1}^{n} x_i w_{ij}$$

Calculate output Z_j

$$Z_j = f(Z_{inj})$$

③ for each O/p unit Y_k Calculate net i/p

$$y_{in_k} = b + \sum_{j=1}^{n} z_j w_{jk}$$

⇒ Phase II – Backpropogation phase.

① Each o/p unit y_k (1 to m) receives target pattern corresponding to i/p training pattern & compute change in weight

$$\Delta V_{jk} = \alpha (t_k - y_k) f'(y_{in_k}) z_j$$

δ_k

$$\delta_k = (t_k - y_k) f'(y_{in_k})$$

$$\Delta V_{jk} = \alpha \delta_k z_j \quad \text{(weight update)}$$

$$\Delta b_k = \alpha \delta_k \quad \text{(bias update)}$$

② Each hidden unit (z_j) sum the delta inputs from the output unit.

$$\delta_{in_j} = \sum_{k=1}^{n} \delta_k V_{jk}$$

$$\delta_j = \delta_{inj} f'(z_{inj})$$

$$\Delta w_{ij} = \alpha \delta_j o_i$$

$$\Delta b_{2j} = \alpha \delta_j$$

⟹ Phase III – Update weight & bias.

① For each output unit y_k update the bias & weight

$$v_{jk(new)} = v_{jk(old)} + \Delta v_{jk}$$

$$b_{new} = b_{old} + \Delta b_k$$

② for each hidden unit update weight & bias

$$w_{ij(new)} = w_{ij(old)} + \Delta w_{ij}$$

$$b_{new} = b_{old} + \Delta b_j$$

⇒ Algorithm : (Back propogation)

Step-1 Initialize weights & learning rate.

Step-2 Perform step 2-6 untill stopping condⁿ false

Step-3 ~~Execute~~ feed forward phase - I

Step-4 Execute Back propogation phase - II

Step-5 Execute weight & bias update phase - III

Step-6 chek for stopping condⁿ stopping condⁿ may be certain no of epoch reached or when the actual o/p equal to target o/p.

Ex:1. Using Back propogation NW find the new weights for net shown in fig.

Input Pattern		Target
x_1	x_2	t
0	1	1

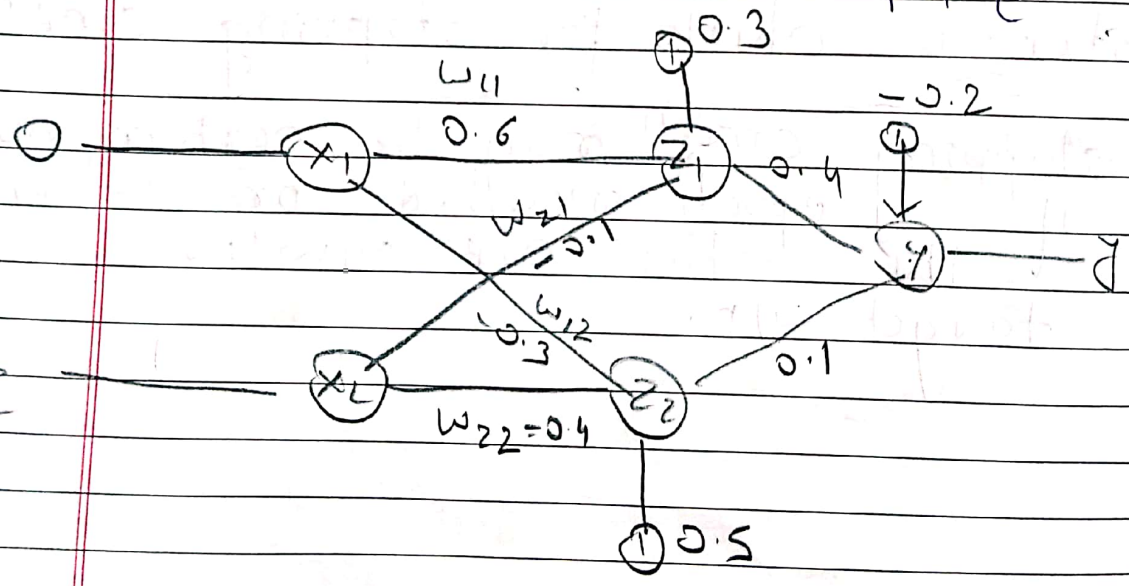
$\alpha = 0.25$, Use binary sigmoidal activation function

Initial Weights.

w_{11}	w_{21}	b_1	w_{12}	w_{22}	b_2
0.6	-0.1	0.3	-0.3	0.4	0.5

v_1	v_2	b_3
0.4	0.1	-0.2

$$f(a) = \frac{1}{1+e^{-x}}$$



⇒ (1) Calculate the i/p. for Z_1 layer.

$$z_{in_1} = b + a_1 w_{11} + a_2 w_{21}$$

$$= 0.3 + (0 \times 0.6) + (1 \times (-0.1))$$

$$\boxed{z_{in_1} = 0.2}$$

$$z_{in_2} = b + a_2 w_{22} + a_2 w_{12}$$

$$= 0.5 + 1 \times 0.4 + 0 \times (-0.3)$$

$$\boxed{z_{in_2} = 0.9}$$

⇒ (2) Apply Activation function.

$$z_1 = f(z_{in_1}) = \frac{1}{1 + e^{-z_{in_1}}}$$

$$= \frac{1}{1 + e^{-0.2}} = 0.54$$

$$\boxed{z_1 = 0.54}$$

$$z_2 = f(z_{in_2}) = \frac{1}{1 + e^{-0.9}} = 0.71$$

$$\boxed{z_2 = 0.71}$$

(3) Calculate net input to O/p layer.

$$y_{in} = b_3 + z_1 w_1 + z_2 w_2$$

$$= -0.2 + (0.54 \times 0.4) + (0.71 \times 0.1)$$

$$y_{in} = 0.091$$

$$y = f(y_{in}) = \frac{1}{1 + e^{-0.091}} = 0.5227$$

$$y = 0.52$$

(4) Compute $\delta_k = (t_k - y_k) f'(y_{in})$

$$f'(y_{in}) = f(y_{in})[1 - f(y_{in})]$$

$$= 0.52[1 - 0.52]$$

$$f'(y_{in}) = 0.2495$$

$\alpha(t-y)f'(y_{in})z_j$

$\delta_1 \quad y_1 \quad f'(y_{in})$

$\delta_1 = (1 - 0.52)(0.24)$

$\boxed{\delta_1 = 0.1191}$

(5) find Change of Weight bet^n hidden & o/p layer.

$\Delta W_{11} = \alpha \delta_1 z_1 = 0.25 \times 0.1191 \times 0.54$

$\boxed{\Delta W_{11} = 0.0164}$

$\Delta W_{21} = \alpha \delta_1 z_2 = 0.25 \times 0.1191 \times 0.71$

$\boxed{\Delta W_{21} = 0.02117}$

$\Delta b_3 = \alpha \delta_1 \times 1 = 0.25 \times 0.1191 = 0.029$

$\boxed{\Delta b_3 = 0.029}$

(6) find change in Weight input & hidden layer.

Compute error portion δ_j between input & hidden layer.

$\delta_j = \delta_{inj} f'(z_{inj})$

$\delta_{inj} = \sum_{k=1}^{m} \delta_k W_{jk}$

classmate Date Page

③ $f'(v) = 1$

Input layer

$\delta in_1 = \delta_1 \times v_{11} + \delta_1 \times v_{12}$

$= 0.11 \times 0.4 = 0.04$

$\delta in_2 = \delta_1 \times v_{21}$

$= 0.11 \times 0.1 = 0.011$

Error

$\delta_j = \delta m_1 * f'(z m_1)$

$f'(z m_1) = f(z m_1)[1 - f(z m_1)]$

$= 0.54 [1 - 0.54]$

$f'(z m_1) = 0.24$

$\delta_1 = 0.04 \times 0.24$

$\delta_1 = 0.0118$

$\delta_2 = \delta in_2 * f'(z m_2)$

$f'(z m_2) = f(z in_2)[1 - f(z m_2)]$

$= 0.71 [1 - 0.71]$

$f'(z m_2) = 0.2055$

$$\delta_2 = \delta_{in_2} * f'(z_{in_2})$$

$$= 0.011 \times 0.2055$$

$$\boxed{\delta_2 = 0.00245}$$

(7) find Change of Weights betn input & hidden layer

$$\Delta w_{11} = \alpha \delta_1 a_1 = 0.25 \times 0.0118 \times 0$$
$$= 0$$

$$\Delta w_{21} = \alpha \delta_1 a_2 = 0.25 \times 0.0118 \times 1$$
$$= 0.00295$$

$$\Delta b_1 = \alpha \delta_1 = 0.25 \times 0.0118$$
$$= 0.00295$$

$$\Delta w_{12} = \alpha \delta_2 a_1 = 0.25 \times 0.0024 \times 0 = 0$$

$$\Delta w_{22} = \alpha \delta_2 a_2 = 0.25 \times 0.0024 \times 1$$
$$= 0.000625$$

$$\Delta b_2 = \alpha \delta_2 = 0.25 \times 0.0024$$
$$= 0.000625.$$

(8) Compute final Weights-

$W_{11(new)} = W_{11(old)} + \Delta W_{11}$

$= 0.6 + 0 = 0.6$

$W_{12(new)} = W_{12(old)} + \Delta W_{12}$

$= -0.3 + 0 = -0.3$

$W_{21(new)} = W_{21(old)} + \Delta W_{21}$

$= -0.1 + 0.00295$

$= -0.09705$

$W_{22(new)} = 0.4 + 0.0006125$

$= 0.4006125$

$V_{1\,new} = V_{1(old)} + \Delta V_1$

$= 0.4 + 0.0164$

$= 0.4164$

$V_{2\,new} = 0.1 + 0.02117$

$= 0.12117$

$b_{1\,new} = b_{1(old)} + \Delta b_1$

$= 0.3 + 0.00295 = 0.30295$

$b_{2\,new} = b_{2(old)} + \Delta b_2$

$= 0.5 + 0.0006125 = 0.5006125$

$b_{3\,new} = b_{3(old)} + \Delta b_3$

$= -0.2 + 0.029 = 0.170$

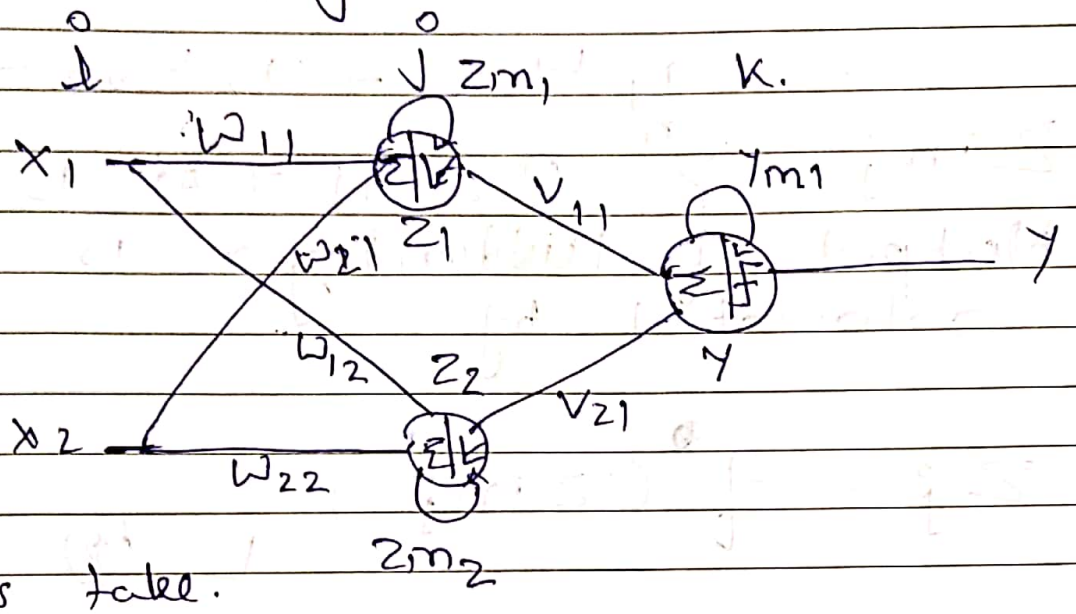
Error based Algorithms

(Gradient Descent Approch)
Delta rule

in Supervised Learning (Perceptron) Single layer

	Perceptron Learning / Training Algorithm	Adaptive Linear Neuron (Adaline)	Multiple Adaptive Linear Neuron (Madaline)	Back Propogation
Initial Weights	for simplicity 0	non zero (0.1 to 1)	non zero (0.1 to 1)	non zero (0.1 to 1)
Initial bias	for simplicity 0	non zero (0.1 to 1	non zero (0.1 to 1)	non zero (0.1 to 1)
learning rate α	0 to 1	0.1 to 1	0.1 to 1	0.1 to 1
Layer	Single layer	Single layer	Multiple layer	Multiple Layer
Weight Change Δw_i	$\alpha t x_i$	$\alpha (t-y) x_i$ Delta rule	$\alpha (t-y) x_i$	$\alpha (t-y) f'(v) x_i$
Activation fundtion	Linear	Linear $f(x) = x$	Linear	Linear Sigmoidal tansig, logsig. (any).
Hidden layer	No	No	Yes.	Yes.
O/p	Single O/p class	Single O/p class.	Single O/p class	Multiple O/p class.

Derive Error Equation for input to hidden layer in back propogation nw.



Let's take.

→ input x_1, x_2

→ Weights from input(i) to hidden layer (j) are $W_{ij} = [W_{11}, W_{12}, W_{21}, W_{22}]$

→ Weights from hiddenlayer (j) to output layer (k) are.

$$V_{jk} = [V_{11}, V_{21}]$$

→ net input to hidden layer is calculated by.

$$z_{in_1} = \sum_{i=1}^{n} x_i W_{ij} + b_j$$

$$= x_1 W_{11} + x_2 W_{12} + b_1$$

$$z_{in_2} = \sum_{i=1}^{n} x_i w_{ij} + b_2$$

$$= a_1 w_{12} + a_2 w_{22} + b_2$$

→ $z_{in_j} = \sum_{i=1}^{n} x_i w_{ij} + b_j$ ——— (1)

→ Output of Hidden layer is calculated by

$$z_j = f(z_{in_j}) \quad \text{——— (2)}$$

→ net i/p to output layer

$$y_{in_k} = \sum v_{jk} z_j \quad \text{——— (3)}$$

→ Output of Output layer y_k

$$y_k = f(y_{in_k}) \quad \text{——— (4)}$$

→ now the Error ~~for~~ by using gradient descent approach can be calculated by

$$E = \frac{1}{2}(t-J)^2 \quad \text{— (5)}$$

→ here we have to find error from input to hidden layer,

Let's say

$$\Delta W_{ij} = \frac{\partial E}{\partial W_{ij}}$$

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial y_k} \times \frac{\partial y_k}{\partial y_{in_k}} \times \frac{\partial y_{in_k}}{\partial z_j} \times \frac{\partial z_j}{\partial z_{in_j}} \times \frac{\partial z_{in_j}}{\partial W_{ij}}$$

from eqn (5)

$$\frac{\partial E}{\partial y_k} = -(t_k - y_k)$$

$$\frac{\partial y_k}{\partial y_{in_k}} = f'(y_{in_k}) \quad \text{from eqn (4)}$$

$$\frac{\partial y_{in_k}}{\partial z_j} = V_{jk} \quad \text{from eqn (3)}$$

$$\frac{\partial z_j}{\partial z_{inj}} = f'(z_{inj}) \quad \text{from eqn (2)}$$

$$\frac{\partial z_{inj}}{\partial w_{ij}} = x_i \quad \text{from eqn (1)}$$

So,

$$\frac{\partial E}{\partial w_{ij}} = -(t-1) \times f'(y_{ink}) \times v_{jk} \times f'(z_{inj}) \times x_i$$

$$\Delta w_{ij} = \alpha(t-y) f'(y_{ink}) \times v_{jk} \times f'(z_{inj}) \times x_i$$

⇒ Derive Error Equation from hidden to output layer.

$$\Delta v_{jk} = \frac{\partial E}{\partial v_{jk}}$$

$$= \frac{\partial E}{\partial y_k} \times \frac{\partial y_k}{\partial y_{ink}} \times \frac{\partial y_{ink}}{\partial v_{jk}}$$

$$\Delta v_{jk} = -2(t-1) \times f'(y_{ink}) \times z_j$$