Using perceptron net for AND function with binary inputs and targets
Plot the convergence curves and the decision boundary lines once obtained.

Perceptron:

- Type of ANN
- Consist of a single layer of weights connected to the input
- It learns a linear decision boundary by adjusting it's weights.

Bipolar signal for activation function

$y = f(y_{in}) = \begin{cases} -1 & y_{in} < 0 \\ 0 & y_{in} = 0 \\ 1 & y_{in} > 0 \end{cases}$

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

x1	x2	AND
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

→ Target

Hence, bipolar input, [[-1 -1]
[-1 1]
[1 -1]
[1 1]]

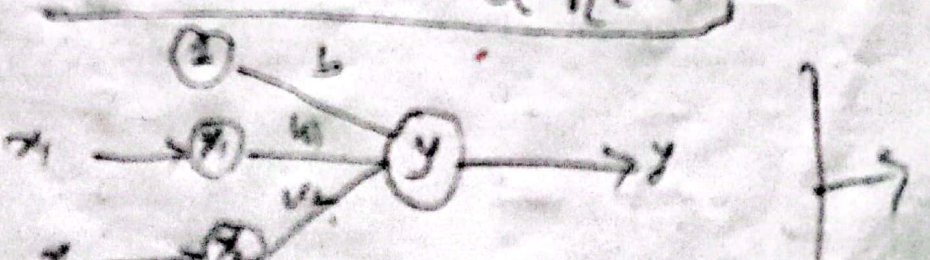
and Target [-1
-1
-1
1]

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

update if $y \neq t$ (y: calculated, t: target)

$$w(new) = w(old) + \eta (y - \hat{y}) x$$

$$w(new) = w(old) + \eta t x$$



$$b(new) = b(old) + \eta (y - \hat{y})$$

bias = $b_k(new) = b_k(old) + \eta t$

Epoch-1 $\eta = 1$

x_1	x_2	t	y_{in}	y_{cal}	Δw_1 ($\eta t x_1$)	Δw_2 ($\eta t x_2$)	Δb (ηt)	w_1	w_2	b
					Weight changes			Weight		
								0	0	0
+1	+1	-1	0	0	-1	-1	-1	-1	-1	-1
1	-1	-1	1	1	1	-1	-1	0	-2	-2
-1	1	-1	2	1	1	-1	-1	1	-1	-3
-1	-1	-1	-3	-1	0	0	0	1	-1	-3

Bipolar sigmoid function; $y = f(y_{in}) = \begin{cases} -1 & y_{in} < 0 \\ 0 & y_i = 0 \\ 1 & y_i > 0 \end{cases}$

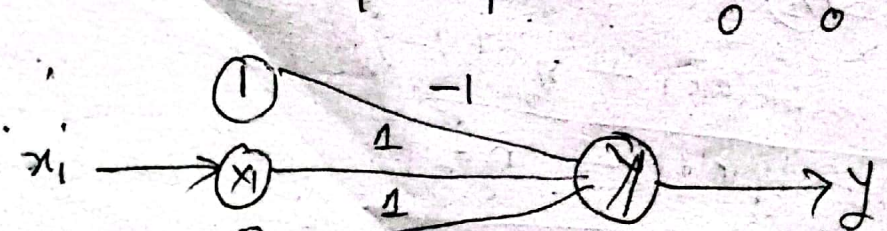
$\eta = 1$

Epoch-1

x_1	x_2	t	y_{in}	y	Δw_1 ($\eta t x_1$)	Δw_2 ($\eta t x_2$)	Δb (ηt)	w_1	w_2	b
					Weight change			Weight		
								0	0	0
-1	-1	-1	0	0	1	1	-1	1	1	-1
-1	1	-1	-1	-1	0	0	0	1	1	-1
1	-1	-1	-1	-1	0	0	0	1	1	-1
1	1	1	1	1	0	0	0	1	1	-1

Epoch-2

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
-1	-1	-1	-3	-1	0	0	0	1	1	-1
-1	1	-1	-1	-1	0	0	0	1	1	-1
1	-1	-1	-1	-1	0	0	0	1	1	-1
1	1	1	1	1	0	0	0	1	1	-1



Q.2 To generate the XOR function using the Mc Culloch-Pitts neuron by writing on .py file. The Convergence curve and the decision bo

Mc Culloch-Pitts Neuron:

- model of a neuron
- Either active (1) or not active (0) based on a threshold function
- linear classifier, to solve AND, OR →
- XOR is non-linearly separable, → (Need) multi-layer perceptron

XOR → Two Mc Culloch-Pitts in the hidden layer.

$$y = x_1\bar{x}_2 + \bar{x}_1 x_2 \quad -(1)$$

X-OR Truth table:

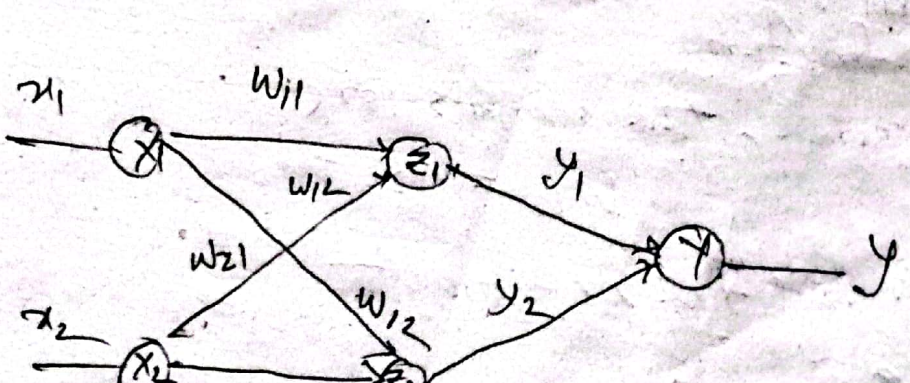
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

X-OR $y = z_1 + z_2$

where, $z_1 = x_1\bar{x}_2$

$z_2 = \bar{x}_1 x_2$

$\therefore y = z_1$ OR z_2



First condition,

$$Z_1 = x_1 \bar{x}_2$$

x_1	x_2	Z_1
0	0	0
0	1	0
1	0	1
1	1	0

Assume, the weights are

$$w_{11} = w_{21} = 1$$

Here Activation function

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \rightarrow \text{Threshold} \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

Calculate the net inputs,

$(0,0)$, $Z_{in} = 0\times1 + 0\times1 = 0$

$(0,1)$, $Z_{in} = 0\times1 + 1\times1 = 1$

$(1,0)$, $Z_{in} = 1\times1 + 0\times1 = 1$

$(1,1)$, $Z_{in} = 1\times1 + 1\times1 = 2$

Here,

$\theta = 1, 2 \Rightarrow$ X

Again, $w_{11} = 1$; $w_{21} = -1$

Calculate the net input,

$(0,0) = 0$

$(0,1) = -1$

$(1,0) = 1$ ✓ Accepted

$(1,1) = 0$

Here

$\theta = 1$ ✓

Similarly, $Z_2 = \bar{x}_1 x_2$

x_1	x_2	Z_2
0	0	0
0	1	1
1	0	0
1	1	0

Second function,

Assume, the weights are

$$w_{12} = w_{22} = 1$$

Calculate the net input

$(0,0) = 0$

$(0,1) = 1$

$(1,0) = 1$

$(1,1) = 2$

for $\theta = 1, 2 \rightarrow$ X not possible

Again, $w_{12} = -1$, $w_{22} = 1$

Calculate the net inputs,

(0,0) = 0
(0,1) = 1 ✓ for $\theta = 1$
(1,0) = −1
(1,1) = 0

Third function $y = z_1$ (OR) z_2

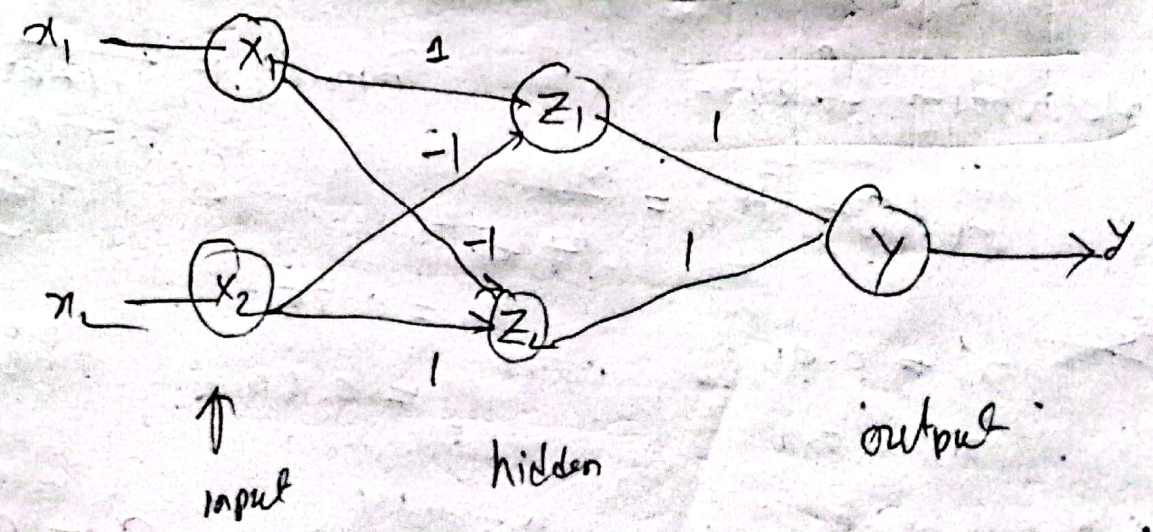
Now, $y_{in} = z_1 v_1 + z_2 v_2$

Assume, $v_1 = v_2 = 1$

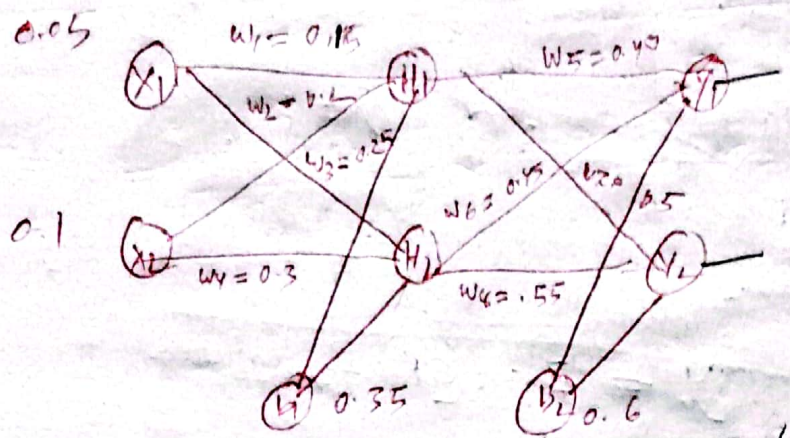
Calculate the net inputs,

(0,0) = 0
(0,1) = 1 ✓ for $\theta = 1$
(1,0) = 1 ✓
(0,0) = 0

x_1	x_2	z_1	z_2	y
		0	0	0
		0	1	1 →
		1	0	1 →
		0	0	0



Q.7 To learn this network using Back propagation Network



Target value
$T_1 = 0.01$
$T_2 = 0.99$

ANN → Computational model. ~~Thus~~ follow. BNN
→ Consist of input, hidden, output layer.
→ Two phases — Forward
— Back propagation

Back propagation work in
Two phases — Forward pass
— Backward pass
↳ For error ~~calc~~ update

Ans:

$$H_1 = x_1 w_1 + x_2 w_2 + b_1$$

Activation function is Sigmoid $= \frac{1}{1+e^{-x}}$

output $H_1 = \frac{1}{1+e^{-H_1}}$

Ex

$x_1 = 0.05$
$x_2 = 0.1$

$b_1 = 0.35$
$b_2 = 0.60$

Target value
T_1 = 0.01 T_2 = 0.99

Initial weights,
$w_1 = 0.15$
$w_2 = 0.2$
$w_3 = 0.25$
$w_4 = 0.3$

$w_5 = 14$
$w_6 = 0.45$
$w_7 = 0.50$
$w_8 = 0.55$

$\eta = 0.5$

Forward pass:

$H_1 = x_1 w_1 + x_2 w_2 + b_1$

$= 0.3775$

and $\text{out } H_1 = \frac{1}{1+e^{-H_1}}$

$= \frac{1}{1+e^{-.3775}}$

$= 0.593269992$

Also,

$H_2 = 0.596884378$

Now for calculating y_1

$y_1 = \text{out } H_1 w_5 + \text{out } H_2 w_6 + b_2$

$= 1.105905967$

$\text{out } y_1 = \frac{1}{(1+e^{-y_1})} = 0.751365070$

In the same way,

$\text{out } y_2 = 0.77292$

Calculate Total Error, $= \sum \frac{1}{2} (\text{target} - \text{output})^2$

$E_{total} = \underbrace{\frac{1}{2}(T_1 - \text{out } y_1)^2}_{E_1} + \underbrace{\frac{1}{2}(T_2 - \text{out } y_2)^2}_{E_2}$

$= \frac{1}{2}(0.01 - 0.7516)^2 + \frac{1}{2}(0.99 - 0.772)^2$

$E_{Total} = 0.298371109$

To update weight,

Consider w_5 → Partial differentiation

Error of $w_5 = \frac{\delta E_{total}}{\delta w_5}$

$$\therefore \frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta out\, y_1} \times \frac{\delta out\, y_1}{\delta y_1} * \frac{\delta y_1}{\delta w_5}$$

$$\therefore \frac{\delta E_{total}}{\delta out\, y_1} = 2 * \frac{1}{2} (T_1 - out\, y_1)^{2-1} * -1 + 0$$

$$= -(T_1 - out\, y_1)$$

$$= 0.74136$$

$$\therefore \frac{\delta out\, y_1}{\delta y_1} = out\, y_1 (1 - out\, y_1)$$

$$= 0.75 (1 - 0.75)$$

$$= 0.186\,215\,602$$

$$\therefore \frac{\delta y_1}{\delta w_5} = 1 \times out\, H_1 \times w_5^{(1-1)} + 0 + 0$$

$$= out\, H_1$$

$$= 0.593\,269\,992$$

$\therefore$ Finally, $\frac{\delta E_{total}}{\delta w_5} = 0.74 \times 0.186 \times 0.593$

$= 0.08216$ → Change in w_5

Updating w_5,

$$w_5 = w_5 - \eta * \frac{\delta E_{total}}{\delta w_5}$$

$= 0.4 - 0.5 \times 0.0821 = 0.3589164$ → w_5

In the same way,

$w_6 = 0.408$
$w_7 = 0.61130$
$w_8 = 0.0613$

} New updated weights

Now at Hidden layer, updating $w_1\ w_2\ w_3\ w_4$

$$\frac{\delta E_{total}}{\delta w_1} = \frac{\delta E_{total}}{\delta out_{H_1}} * \frac{\delta out_{H_1}}{\delta H_1} * \frac{\delta H_1}{\delta w_1} \quad \text{--- (iv)}$$

$$\therefore \frac{\delta E_{total}}{\delta out_{H_1}} = \frac{\delta E_1}{\delta out_{H_1}} + \frac{\delta E_2}{\delta out_{H_2}}$$

$$= \left[\frac{\delta E_1}{\delta y_1} * \frac{\delta y_1}{\delta out_{H_1}}\right] + \left[\frac{\delta E_1}{\delta out_{y_1}} \times \frac{\delta out_{y_1}}{\delta y_1} \times w_5\right] +$$

$$= (0.7413 \times 0.1868 \times 0.40) +$$

$$= (0.13849 \times 0.4) +$$

$$= 0.05539$$

Similarly, $\frac{\delta E_2}{\delta out_{H_1}} = -0.019049$

$$\therefore \frac{\delta E_{total}}{\delta out_{H_1}} = 0.05539 + (-0.0190)$$

$$\frac{\delta E_{total}}{\delta out_{H_1}} = 0.03635$$

$$\therefore \frac{\delta out_{H_1}}{\delta H_1} = out_{H_1}(1 - out_{H_1}) = 0.5932(1 - 0.5932)$$

$$= 0.2413$$

$$\therefore \frac{\delta H_1}{\delta w_1} = x_1 = 0.05$$

Finally: equation (iv) $\frac{\delta E_{total}}{\delta w_1} = 0.036 \times 0.2413 \times 0.05$

$$= 0.000436$$

Updating w_1

$$w_1 = w_1 - \eta * \frac{\delta E_{total}}{\delta w_1}$$

$$= 0.16 - 0.5 \times 0.000438$$

$$= 0.1497$$

In the same way,

$$w_2 = 0.19956$$
$$w_3 = 0.2497$$
$$w_4 = 0.2995$$

In process ই- চলতে থাকবে যতক্ষণ না y_1 এবং y_2 Target value কাছাকাছি চলে না আসে।

Verify.

$$H_1 = x_1 w_1 + x_2 w_2 + b_1$$

$$= 0.05 \times 0.143 + 0.1 \times 0.266 + 1.017$$

$$= 1.05275$$

$$H_2 = 0.05 \times 0.2826 + 0.1 \times 0.365 + 1.025$$

$$= 1.053$$

$$\text{out } H_1 = \frac{1}{1+e^{-H_1}}$$

$$= 0.74$$

$$\text{out } H_2 = \frac{1}{1+e^{-H_2}}$$

$$= .74$$

$$y_1 = \text{out } H_1 \times (-1.47) + \text{out } H_2 \times (-1.426)$$

$$= -1.08746$$

$$= 2.14$$

Q.9 To purchase Classification Prediction using SVM

SVM → linear / Non-linear → using a kernel

→ Text, image, spam classification

→ To create best line

→ best decision boundary is called hyperplane.

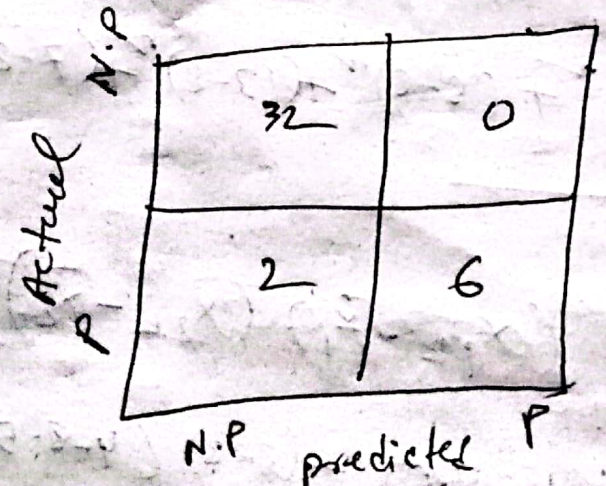
→ Two types: linear : → straight line

Non-linear : → $z = x^2 + y^2$

→ maximum margin, max $\frac{2}{\|w\|}$

100 — 73

1 — $\frac{73}{100}$ × 100



Q.10 To reduce dimensions of a dataset using PCA

PCA → identifies a set of orthogonal axes.

Ex.

$\bar{x}_1 = \frac{1}{8}(4+6+13+7)$

$= 8$

$\bar{x}_2 = \frac{1}{4}(11+4+5+14)$

$= 8.5$

F	Ex1	Ex2	Ex3	Ex4
x_1	4	6	13	7
x_2	11	4	5	14

Step-2: Calculation of the ~~Covariance~~ Covariance matrix

$$S = \begin{bmatrix} Cov(x_1,x_1) & Cov(x_1,x_2) \\ Cov(x_2,x_1) & Cov(x_2,x_2) \end{bmatrix}$$

$$\therefore Cov(x_1x_1) = \frac{1}{N-1}\sum_{k=1}^{N}(x_{1k}-\bar{x}_1)(x_{1k}-\bar{x}_1)$$

$$= \frac{1}{3}\left[(4-8)^2+(8-8)^2+(13-8)^2+(7-8)^2\right]$$

$$= 14$$

$$\therefore Cov(x_1,x_2) = \frac{1}{N-1}\sum_{k=1}(x_{1k}-\bar{x}_1)(x_{2k}-\bar{x}_2)$$

$$= \frac{1}{3}\left[(4-8)(11-8.5)+(8-8)(4-8.5)\right.$$
$$\left.+(13-8)(5-8.5)+(7-8)(14-8.5)\right]$$

$$= -11$$

Similarly, $Cov(x_2,x_1) = Cov(x_1,x_2)$

$$= -11$$

$$Cov(x_2,x_2) = \frac{1}{N-1}\sum_{k=1}^{N}(x_{2k}-\bar{x}_2)(x_{2k}-\bar{x}_2)$$

$$= \frac{1}{3}\left[(11-8.5)^2+(4-8.5)^2+(5-8.5)^2\right.$$
$$\left.+(14-8.5)^2\right]$$

$$= 23$$

$\therefore S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$

Step-3: Eigenvalues of the covariance matrix

$0 = \det(S - \lambda I)$

$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\therefore \lambda I = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$

$= \begin{vmatrix} 14-\lambda & -11 \\ -1 & 23-\lambda \end{vmatrix}$

$= (14-\lambda)(23-\lambda) - (-11) \times (-11)$

$0 = \lambda^2 - 37\lambda + 201$

$\therefore \lambda = 30.3849, \ 6.6151$

$= \lambda_1 \quad \lambda_2$ → Largest

Step-4: Computation of the eigenvector.

$\therefore U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = (S - \lambda I) U$

$= \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} (14-\lambda)u_1 - 11u_2 \\ -11u_1 + (23-\lambda)u_2 \end{bmatrix}$

$(14-\lambda)u_1 - 11u_2 = 0 \Rightarrow \frac{u_1}{11} = \frac{u_2}{14-\lambda}$

and $-11u_1 + (23-\lambda)u_2 = 0$

$u_1 = 11t, \quad u_2 = (14-\lambda)t$

If $t = 1$

$\therefore U = \begin{bmatrix} 11 \\ 14-\lambda \end{bmatrix}$

To find a unit eigenvector,

$$U_1 = \begin{bmatrix} 11 \\ 14 - \lambda_1 \end{bmatrix}$$

$$\therefore \|U_1\| = \sqrt{11^2 + (14 - \lambda_1)^2}$$

$$= \sqrt{11^2 + (14 - 30.38)^2}$$

$$= 19.7348$$

$$\therefore e_1 = \begin{bmatrix} 11/\|U_1\| \\ (14 - \lambda_1)/\|U_1\| \end{bmatrix}$$

$$= \begin{bmatrix} 0.5574 \\ -0.8308 \end{bmatrix}$$

Similarly,

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

Step-5: First principal Components.

$$e_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix} = \begin{bmatrix} 0.5574 - 0.83 \end{bmatrix} \begin{bmatrix} X_{11} - \bar{X}_1 \\ X_{21} - \bar{X}_2 \end{bmatrix}$$

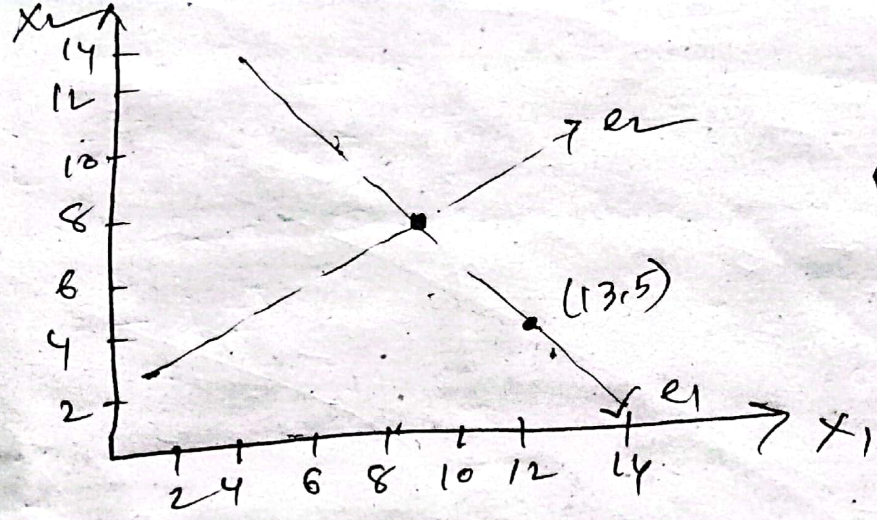
$$= -4.3053$$

$$= 0.55(4-8) - .830(11-8.5)$$

$$= -4.3053$$

Similarly

Feature	Ex1	Ex2	Ex3	Ex4
X_1	4	8	13	7
X_2	11	4	5	14
First Principal Components	-4.305	3.736	5.6978	-5.1238



point গুলোর
লাইন
যুক্ত দেখাতে
হবে

Q.3 SGD method using Delta learning →

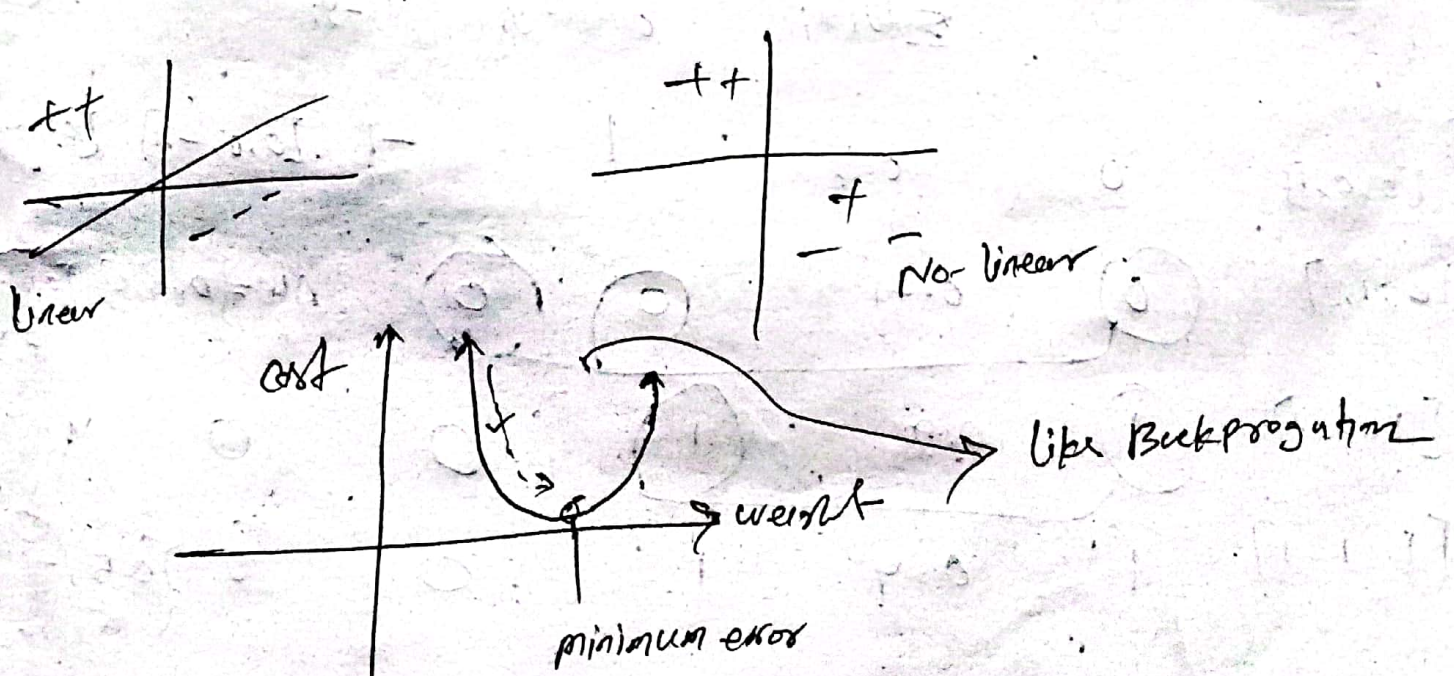
Xinput = [0 0 1; 0 1 1; 1 0 1; 1 1 1];

Dtarget = [0; 0; 1; 1]

SGD
- optimization algorithm
- → minimize the loss function.
- → update model parameters using single / few data point

Delta learning Rule:

- → Widrow – Hoff rule
- → Used for non-linearly separable data



How to modify weights, $w_i \leftarrow w_i + \Delta w_i$; $\Delta w_i = -\eta \nabla E(w)$

$\therefore \nabla E(w)$ → derivative of error w.r.t. weights
also called as gradient

$$w(new) = w(old) + \eta (y - \hat{y}) x$$

$$\therefore \nabla E(w) = \left[\frac{\delta E}{\delta w_0}, \frac{\delta E}{\delta w_1}, \ldots \frac{\delta E}{\delta w_n}\right]$$

Ex.

x_1	x_2	b	t
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

Randomly

Assume $W = [0.5, -0.5, 0.5]$

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

Epoch-1

$\eta = 0.1$

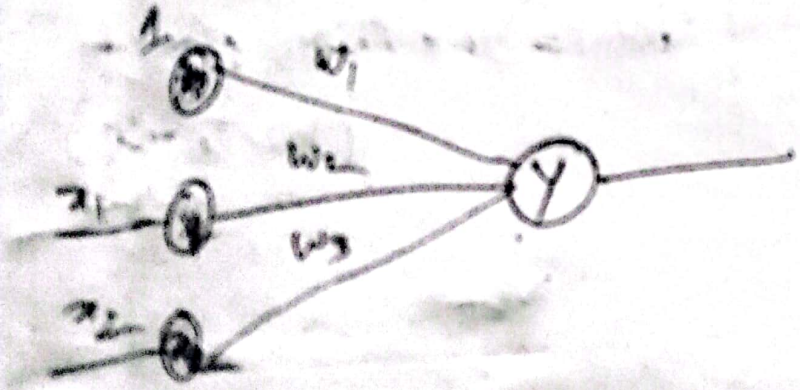
Input	Target d_i	Weighted sum $\sum w_i x_i$	predicted output y_i	Error $d_i - y_i$	Δw	new w
[0,0,1]	0	0.5	1	-1	[0,0,-1]	[0.5-
[0,1,1]	0	0.-1	0	0	Δw=[0,0,0]	[0.5,-5 0.4]
[1,0,1]	1	0.9	1	0	0	v
[1,1,1]	1	0.4	1	0	0	[0.5,-0.5,0.

3 4 To compute [illegible] gradient method [illegible] with [illegible] rate

Batch method:

Batch gradient descent

$\eta = 0.1$



Q.5 Five by Five image

$W_2 = W_2 + dW_2$

$= W_2 + \eta\,(\text{delta}, y_1 \cdot T)$

$= W_2 + \eta\,((d-y), y_1 \cdot T)$

$\therefore W_2 = W_2 + \eta\,(y-y)$

$W_1 = W_1 + dW_1$

$= W_1 + \eta * (\text{delta}1, x \cdot T)$

delta1 = y1 * (1−y1) * hp.dot (W2.T, d−y)

Q 6 CNN

- extended version of ANN
- used to extract the feature from the matrix data

Q.8 Speech Recognize

- MFCC → mel frequency cepstral coefficients
- ANN → Combine with MFCC

Jupyter install

ANN BNN

To get optimal solution need cost function

ANN
- input
- weight
- output
- Hidden layer

BNN
- Dendrites
- synapse
- axon
- cell body