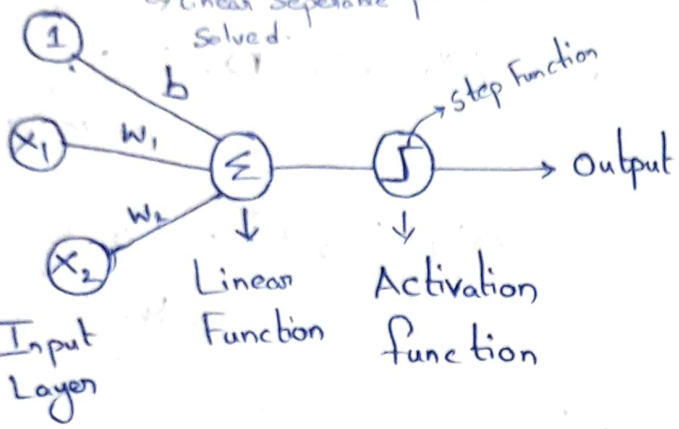
Deep Learning

Supervised Algo

↳ Feedforward Neural Network

Perceptron → Binary Classification

↳ Linear Seperable problems can be solved.



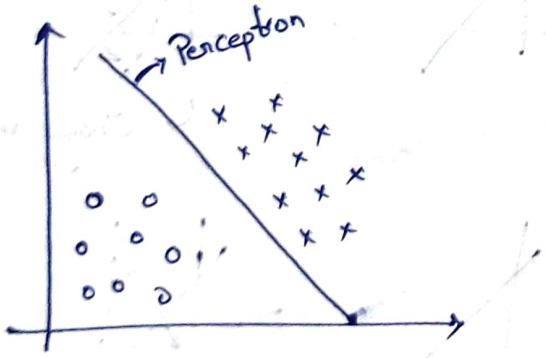
$Z = w_1x_1 + w_2x_2 + b$

$z = \sum_{i=1}^{n} w_i x_i + b$ → Generalized Fn

x_1 & x_2 → Inputs / features

b → Bias / Intercept

w → Weights



* Works best / only on Linear Equations, doesn't work on non-linear Equations

- Perceptron is nothing but a line. It creates regions to

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

↳ Step Function

Activation function

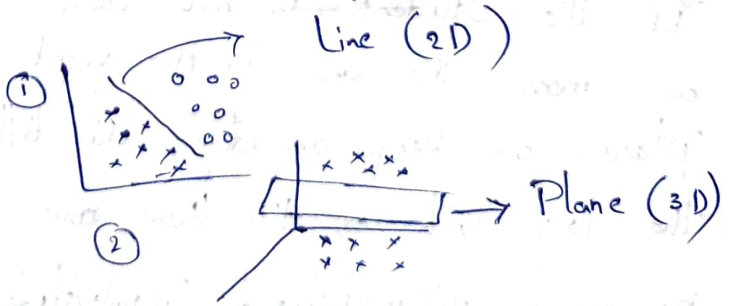
↳ It tries to normalize (or) Standardise the z value. So that the output will be either 0/1 (or) -1/1...

↳ It tries to bring the values under a range.

* What does Weight Convey?

↳ It tells us which feature is more important.

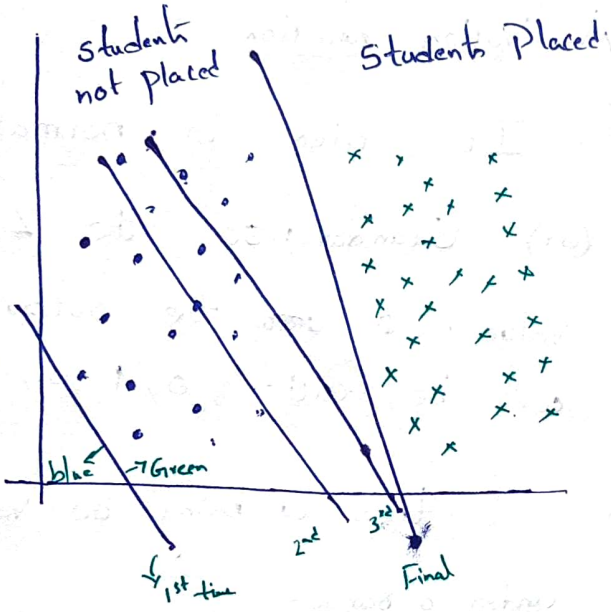
↳ On which input the Output is more dependent.



3 → Hyperplane (> 3D)

Perceptron Trick

epocs → The no. of times a loop runs



what is happening here is that we run a loop 1000 times (or) any no. of times and in each loop we select a student in random and we check if the student is in their respective regions with respect to the line.

If the student is not then we move the line to the right place. we keep on doing this till the loop exits (or) till there are no more mismatched points/students

Here we are updating W & b in every iteration ~~for a~~ till we reach right region/place.

$Wx_1 + Wx_2 + Wx_3 + b$

$\downarrow$

$Ax_1 + Bx_2 + Cx_3 + b = 0 \rightarrow$ line Eq

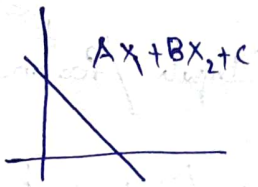
if $Ax_1 + Bx_2 + b/c > 0$

↳ positive Region

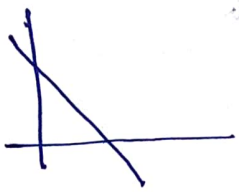
$Ax_1 + Bx_2 + b/c < 0$

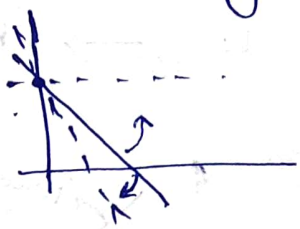
↳ Negative Region

↳ Reference → Desmos.com/calculator

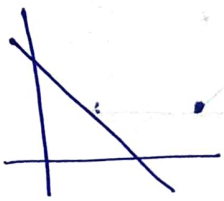


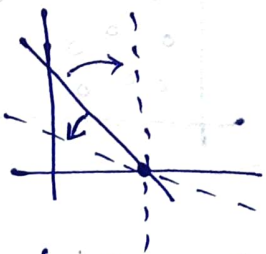
↳ If change in `C`





↳ Change in `A`



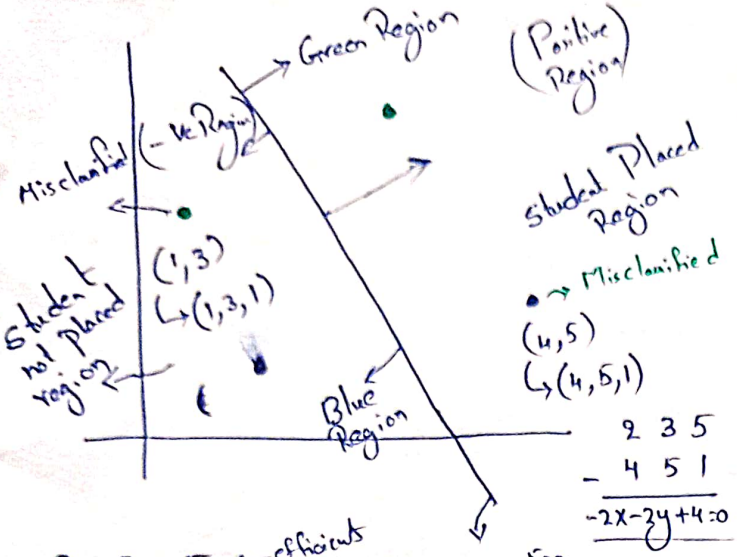


↳ Change in B

$Ax_1 + Bx_2 + C = 0 \rightarrow$ line

After substituting the points line, if the result is zero, that means the point lie on the line.

Transformations



2 3 5 → co-efficients

1 3 1 → co-ordinates → multiply points with learning rate

\+ 3x+6y+6 = 0

Formula

$$coeff_{new} = coeff_{old} - \eta \, co\text{-}ordinates$$

Algorithm (Pseudo Code)

epoch = 1000 , $\eta = 0.01$

for i in range (epoch):

  randomly select a Student

  if $x_i \in N$ (Not placed) & $\sum_{i=0}^{n} w_i x_i \geq 0$: → eq1

    $w_{new} = w_{old} - \eta x_i$

  if $x_i \in P$ (placed) & $\sum_{i=0}^{n} w_i x_i < 0$: → eq2

    $w_{new} = w_{old} + \eta x_i$

y	$\hat{y}$	$y_i - \hat{y}_i$
1	1	0
0	0	0
1	0	1
0	1	-1

* Let Simplify the Algo

Instead of having two eqns let have One eq

$$\boxed{w_{new} = w_{old} + \eta (y - \hat{y}) x_i}$$

Case(i) $y = 1$, $\hat{y} = 1$ → Matched/Proper classification

$w_{new} = w_{old} + \eta (1 - 1) x_i$

$\boxed{w_{new} = w_{old}}$

Case(ii) $y = 0$, $\hat{y} = 0$ → Proper classification

$w_{new} = w_{old} + \eta (0) x_i$

$\boxed{w_{new} = w_{old}}$

Case(iii) $y = 1$, $\hat{y} = 0$ → Mis classified

$\boxed{w_{new} = w_{old} + \eta (1) x_i}$

$\boxed{w_{new} = \text{Some new value}}$

Case(iv) $y = 0$, $\hat{y} = 1$ → Mis classified

$$\boxed{w_{new} = w_{old} - \eta x_i}$$

Loss Function

↳ It is a mathematical function that measures how well a model's predictions match the actual outcomes

In Simple Terms:-

↳ It tells us how wrong your model.

- Small Loss → Better model is performing

Perceptron Loss Function

↳ sklearn → SGD

$$L(w_1, w_2, b) = \frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

↳ $w_1 x_i + w_2 x_2 + b$

$n \rightarrow$ no. of rows

$y_i \rightarrow$ output/target

x_1	x_2	y
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2

$$f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$$

∵ x_{ij} → row, col

$$\max(0, -y_i f(x_i))$$

$$= \max(0, x) \qquad x = -y_i f(x_i)$$

$x \geq 0 \quad x = $ value

$x < 0 \quad x = 0$

$$L = \frac{1}{2}\left[\max(0, -y_1 f(x_1)) + \max(0, -y_2 f(x_2))\right]$$

↳ Since we have only two points

Ex

y	$\hat{y}$	$\max(0, -y_i f(x_i))$
1	1	$\max(0, -ve) = 0$
-1	-1	$\max(0, -ve) = 0$
1	-1	$\max(0, +ve) = y_i f(x_i)$
-1	1	$\max(0, +ve) = y_i f(x_i)$

We need to find the value of $f(x_i)$

$$f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$$

we need to find w_1, w_2, b

Gradient Descent

$$w_1 = w_1 + \eta \frac{\delta L}{\delta w_1}$$

$$w_2 = w_2 + \eta \frac{\delta L}{\delta w_2}$$

$$b = b + \eta \frac{\delta L}{\delta b}$$

$$\frac{\delta L}{\delta w_1} = \frac{\delta L}{\delta f(x_i)} \times \frac{\delta f(x_i)}{\delta w_1}$$

$$\frac{\delta L}{\delta f(x_i)} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases}$$

$$\frac{\delta f(x_i)}{\delta w_1} = x_{i1}$$

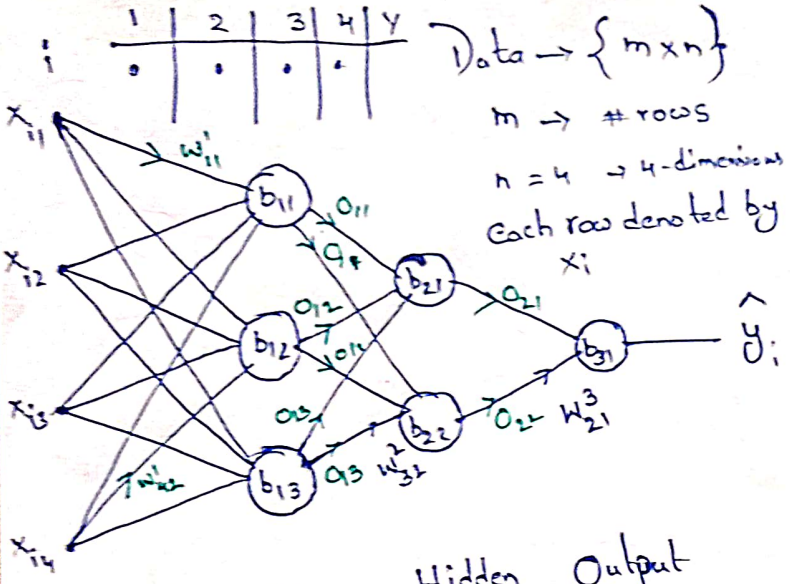
$$\frac{\delta L}{\delta w_1} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i x_{i1} & \text{if } y_i f(x_i) < 0 \end{cases}$$

$$\frac{\delta L}{\delta w_2} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i x_{i2} & \text{if } y_i f(x_i) < 0 \end{cases}$$

$$\frac{\delta L}{\delta b} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases}$$

Loss function	Activation function	Output
Hinge Loss	step	Act as Perceptron → binary Classification (-1, 1)
log loss (binary Cross Entropy)	Sigmoid	" Logistic Regression → binary classification (0, 1)
Categorical Cross Entropy	Softmax	" Softmax Regression → Multiclass classification ↳ Probability
MSE	Linear	" Linear Regression → Number

Multi Layer Perceptron



Input Layer (L_0)

Hidden Layer #1 (L_1)

Hidden Layer #2 (L_2)

Output Layer (L_3)

↑ 12 weights, 3 bias (15)

↑ 6 weights, 2 bias (8)

↑ 2 weights, 1 bias (3)

∴ There are 26 Trainable parameters.

When you train the neural network it tries to find these 26 parameters.

b_{ij} → bias , i → layer , j → node

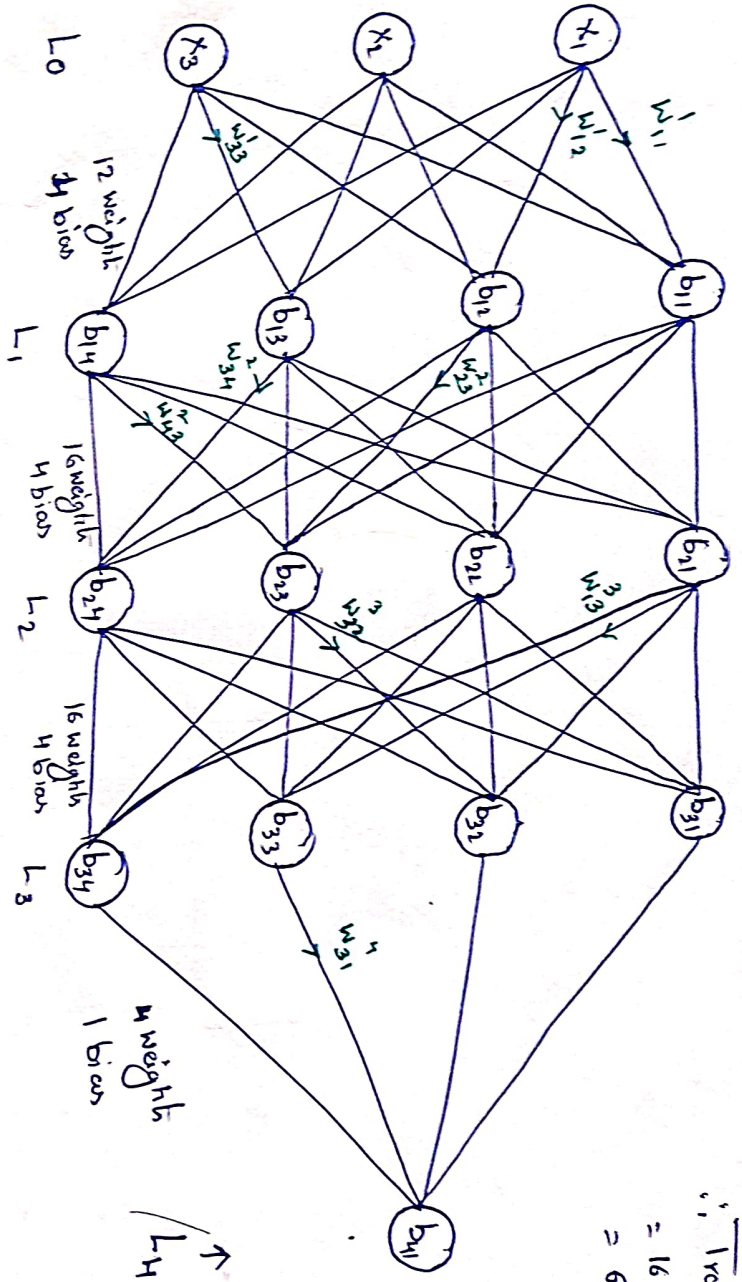
Output → O_{ij}

W^k_{ij} → weight

k → which layer (current layer)

i → Originating node

j → Destination node



∴ Trainable Parameters
= 16 + 20 + 20 + 5
= 61

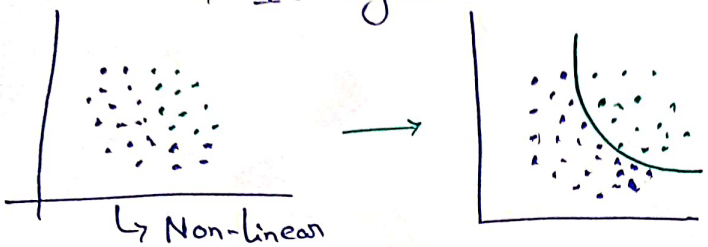
Multi Layer Perceptron Intution

↳ Tensorflow Playground

Perceptron with Sigmoid

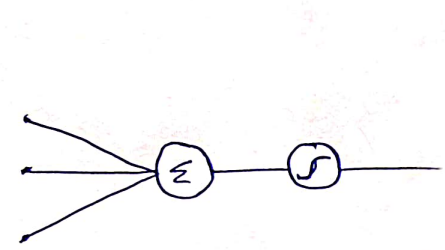
Problem with Perceptron

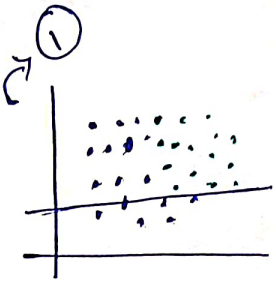
↳ It only works on linear Data



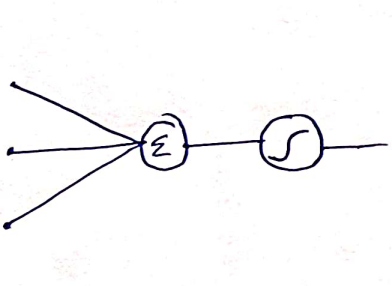
↳ Non-linear

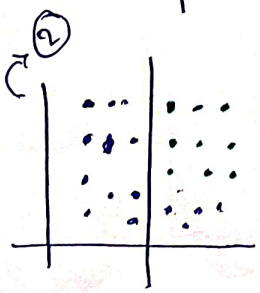
Perceptron with Sigmoid



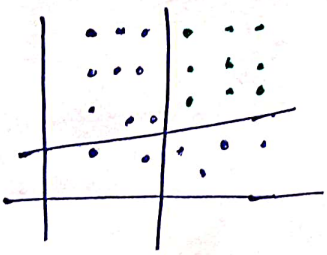


+



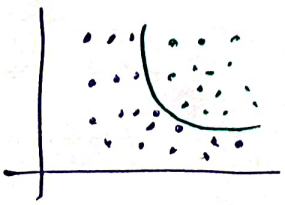


(1) + (2) =

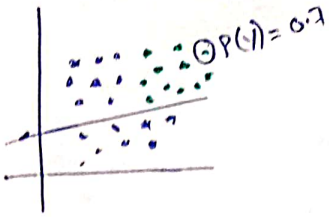


Smoothening

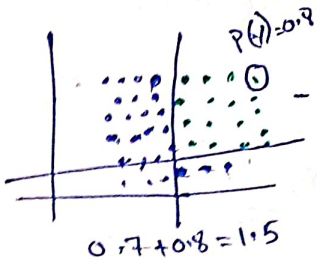
=



Mathematical Intuition



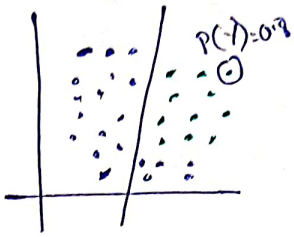
=



0.7 + 0.8 = 1.5

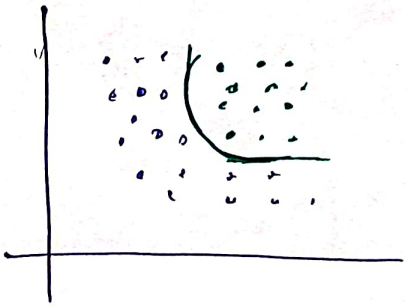
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

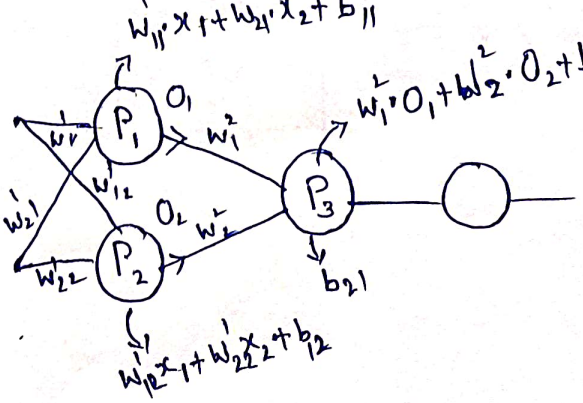
= 0.8



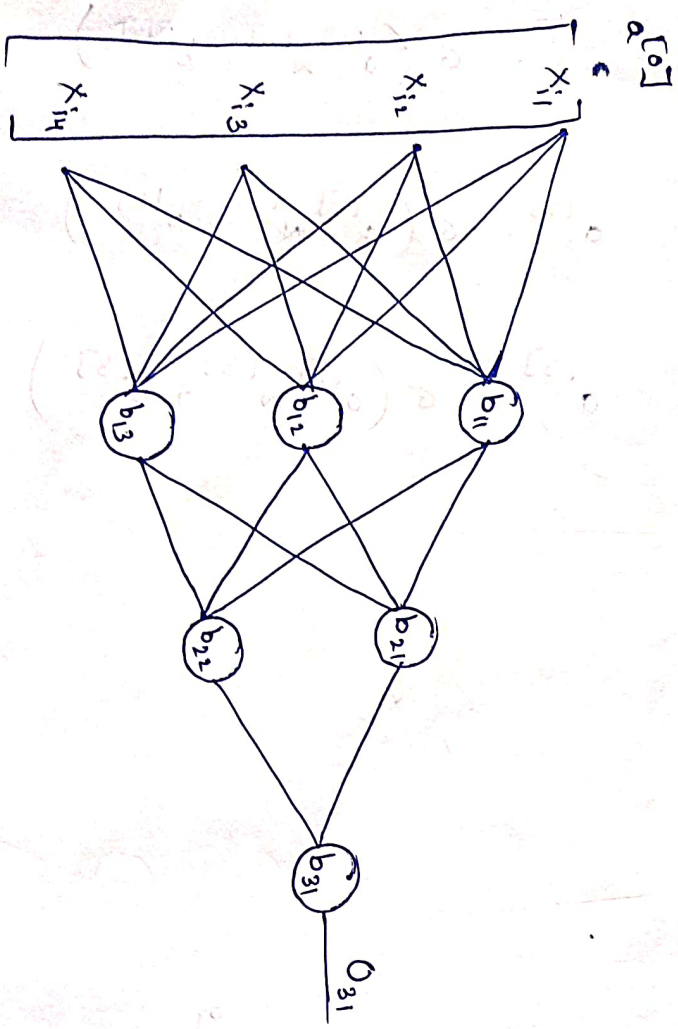
We are doing Linear Combination of two perceptrons to find out new probability.

Result





Forward Propogation



Layer 1

$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{43} \end{bmatrix}^T \quad \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

4×3 $\quad$ 4×1

Row wise → Weight Originating from Single node

Eg:- x_{i1}

Column wise ↓ Weight Covering to a Single node

Eg:- b_{11}

$$\Rightarrow \begin{bmatrix} w'_{11} x_{i1} + w'_{21} x_{i2} + w'_{31} x_{i3} + w'_{41} x_{i4} + b_{11} \\ w'_{12} x_{i1} + w'_{22} x_{i2} + w'_{32} x_{i3} + w'_{42} x_{i4} + b_{12} \\ w'_{13} x_{i1} + w'_{23} x_{i3} + w'_{33} x_{i3} + w'_{43} x_{i4} + b_{13} \end{bmatrix}$$

$\Rightarrow \sigma \left(\quad \right)$

$$\Rightarrow \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix}$$

CGPA	IQ	10th M	12th M	Placed
7.2	72	69	81	1
8.1	92	75	76	0

Prediction → $\sigma(W^T x + b)$ → Sigmoid

Layer 2

$$\begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \\ w_{31}^2 & w_{32}^2 \end{bmatrix}^T \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

$3\times 2 \xrightarrow{T} 2\times 3 \quad 3\times 1$

$$= \sigma\left(\begin{bmatrix} w_{11}^2 o_{11} + w_{21}^2 o_{12} + w_{31}^2 o_{13} + b_{21} \\ w_{12}^2 o_{11} + w_{22}^2 o_{12} + w_{32}^2 o_{13} + b_{22} \end{bmatrix} \right)$$

$$= \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}$$

$$a^{[1]} = \sigma\left(a^{[0]} w^{[1]} + b^{[1]}\right)$$

$$a^{[2]} = \sigma\left(a^{[1]} w^{[2]} + b^{[2]}\right)$$

$$a^{[3]} = \sigma\left(a^{[2]} w^{[3]} + b^{[3]}\right)$$

Layer 3

$$\begin{bmatrix} w_{11}^3 \\ w_{21}^3 \end{bmatrix}^T \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \end{bmatrix}$$

$2\times 1 \xrightarrow{T} 1\times 2 \quad 2\times 1$

$$= \sigma\left(\begin{bmatrix} w_{11}^3 o_{21} + w_{21}^3 o_{22} + b_{31} \end{bmatrix} \right)$$

$$= \hat{y}_i = o_{31}$$

$$a^{[3]} \rightarrow a^{[2]} \rightarrow a^{[1]} \rightarrow \sigma\left(\sigma\left(\sigma\left(a^{[0]} w^{[1]} + b^{[1]}\right) w^{[2]} + b^{[2]}\right) w^{[3]} + b^{[3]}\right)$$

Loss Function

↳ It is a method of evaluating how well your algorithm is modelling your dataset.

↳ Measures performance of your algorithm.

High → Poor Performence

Low → Great Performence

MSE

$$L = (y_i - \hat{y})^2$$

$\hat{y} \downarrow$

$mx_i + c$

$$L(m, c) = \min.$$

Why Loss fn is important ?

You can't improve what you can't measure.

Loss Functions in Deep Learning

Regression

(i) MSE (ii) MAE (iii) Huber Loss

Classification

(i) Binary Cross Entropy
(ii) Categorical Cross Entropy
(iii) Hinge Loss

Loss Function Vs Cost Function

Error Function

Loss Function → Single Training Example

$$\Rightarrow (y_i - \hat{y}_i)^2$$

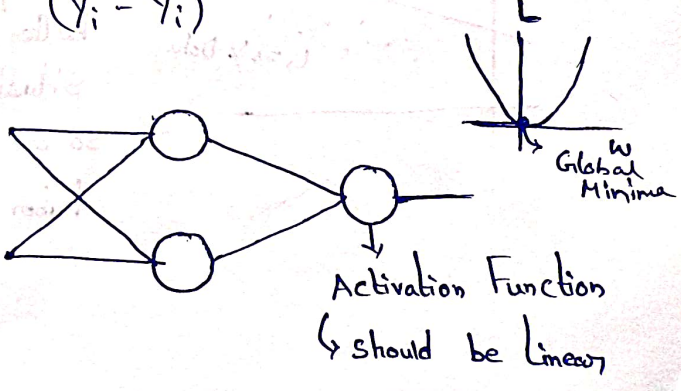
Cost Function → Complete Training Data

$$\Rightarrow \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

① MSE / Squared Loss / L2 Loss in DL

$$\Rightarrow (y_i - \hat{y}_i)^2$$

↳ Not robust for outliers



Activation Function

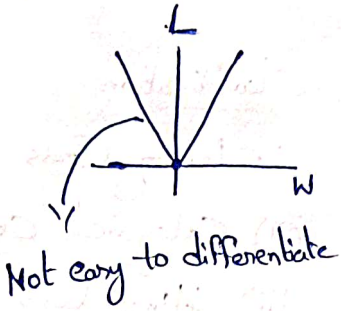
↳ Should be Linear

② Mean Absolute Error → L1 Loss

$$L = |y_i - \hat{y}_i|$$

$$CF = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

↳ Robust for Outliers

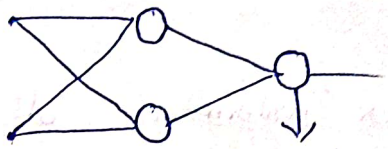


Not easy to differentiate

④ Binary Cross Entropy / Log loss

$$L = -y \log \hat{y} - (1-y)\log(1-\hat{y})$$

$$CF = \frac{-1}{n}\left[\sum_{i=1}^{n} y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$$

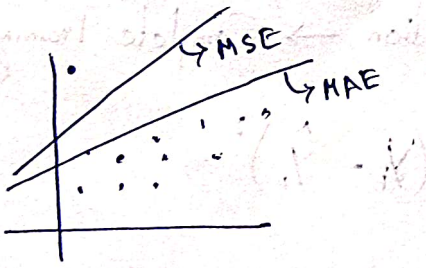


Activation Fn
↳ only Sigmoid

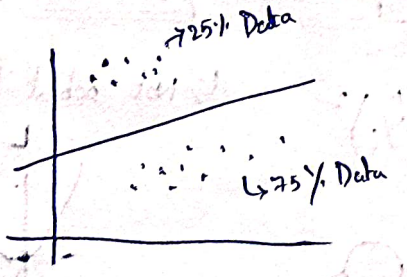
③ Huber Loss

$$L = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

↳ no outliers (MSE)
↳ Hyper parameter
↳ outliers
↳ (MAE)



when Outliers are present



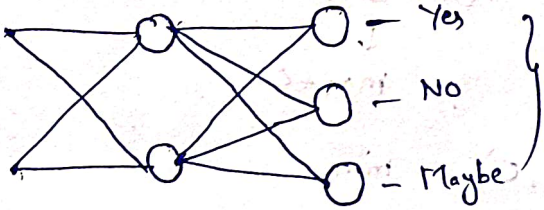
MSE & MAE won't perform better in this situation, so we need huber loss

⑤ Categorical Cross Entropy

$$L = -\sum_{j=1}^{K} y_i \log(\hat{y}_i)$$

K → Classes in data

$$CF = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij} \log(\hat{y}_{ij})$$



Activation Fn
↓
Softmax

* One Hot Encoding is used in this

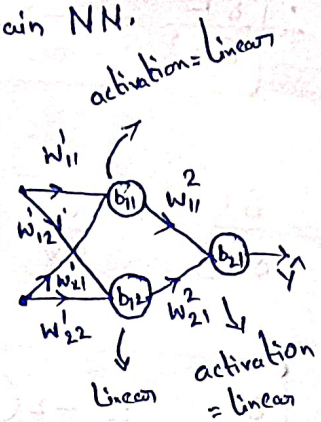
⑥ Sparse Categorical Cross Entropy

↳ Similar to Categorical Cross Entropy but One Hot Encoding is not used, instead Integer Encoding is used.

Back Propogation

↳ An algorithm to train NN.

iq	cgpa	LPa
80	8	3
60	9	5
70	5	8



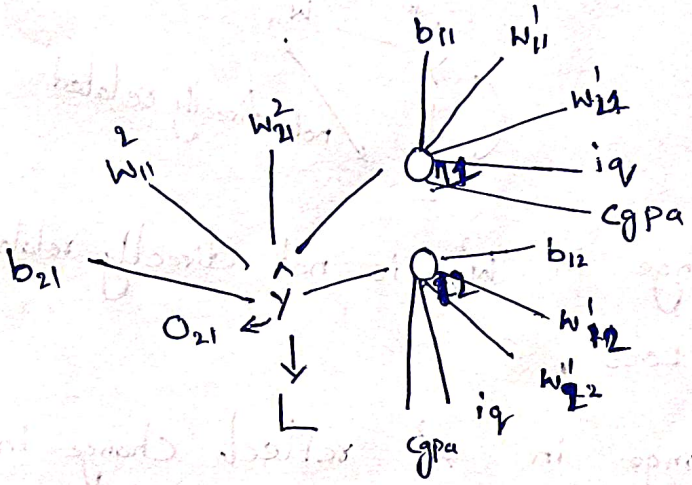
$$\hat{y} = O_{21}$$

$$O_{21} = W_{11}^2 O_{11} + W_{21}^2 O_{12} + b_{21}$$

↳ From forward Propagation

Steps

0.) Initialize W, b (Randomly) $\boxed{W=1, b=0}$



1.) You select a point (row)
↳ student

2.) Predict (lpa) → Forward Prop (DotProduct)

3.) Choose a loss Function

∵ it regression problem → MSE

$$L = (Y - \hat{Y}_i)^2$$

$$= (5 - 18)^2$$

↳ let $\hat{y} = 18$

L = High

To reduce loss, we need to decrease / increase $\hat{y}$

if $\hat{y} = 18, Y = 5$ ↓

if $\hat{y} = 1, Y = 5$ ↑

4.) Update Weight & biases

↳ Gradient Descent

$$W_{new} = W_{old} - \eta \frac{\delta L}{\delta W_{old}}$$

$$b_{new} = b_{old} - \eta \frac{\delta L}{\delta b_{old}}$$

$$W_{11\,new}^2 = W_{11\,old}^2 - \eta \frac{\delta L}{\delta W_{11\,old}^2}$$

$$W_{21\,new}^2 = W_{21\,old}^2 - \eta \frac{\delta L}{\delta W_{21\,old}^2}$$

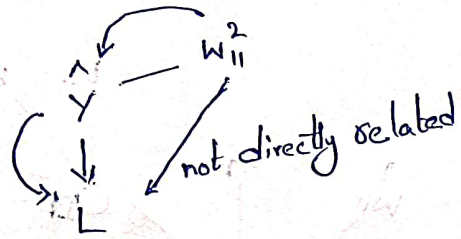
$$b_{21\,new} = b_{21\,old} - \eta \frac{\delta L}{\delta b_{21\,old}}$$

① - O_{21} ② - O_{11}

$$\frac{\delta L}{\delta W_{11}^2}, \frac{\delta L}{\delta W_{21}^2}, \frac{\delta L}{\delta b_{21}} \quad \frac{\delta L}{\delta W_{11}^1}, \frac{\delta L}{\delta W_{21}^1}, \frac{\delta L}{\delta b_{11}}$$

$$\frac{\delta L}{\delta W_{12}^1}, \frac{\delta L}{\delta W_{22}^1}, \frac{\delta L}{\delta b_{12}} \rightarrow ③ \quad \rightarrow O_{12}$$

We need to Calculate these 9 derivatives



Change in W_{11}^2 is not directly related to Loss'.

Change in W_{11}^2 reflect change in $\hat{Y}$ & Change in $\hat{y}$ results in change in L

$$\therefore \frac{\delta L}{\delta W_{11}^2} = \underbrace{\frac{\delta L}{\delta \hat{y}}}_{\text{change in Loss w.r.t } \hat{y}} \times \underbrace{\frac{\delta \hat{y}}{\delta W_{11}^2}}_{\text{change in } \hat{y} \text{ w.r.t change in } W_{11}^2}$$

↳ Chain Rule of differentiation

$$\frac{\delta L}{\delta \hat{y}} = \frac{\delta}{\delta \hat{y}} (y - \hat{y})^2 = -2(y - \hat{y})$$

$$\frac{\delta \hat{y}}{\delta W_{11}^2} = \frac{\delta}{\delta W_{11}^2} \left[O_{11} W_{11}^2 + O_{12} W_{21}^2 + b_{21} \right]$$

$$= O_{11}$$

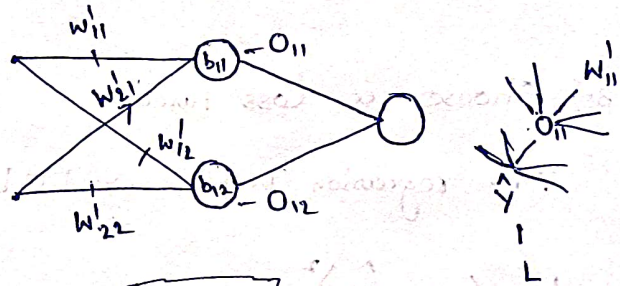
$$\therefore \boxed{\frac{\delta L}{\delta W_{11}^2} = -2(y - \hat{y}) O_{11}} \rightarrow 1$$

$$\frac{\delta L}{\delta W_{21}^2} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta W_{21}^2}$$

$$\boxed{\frac{\delta L}{\delta W_{21}^2} = -2(y - \hat{y}) O_{12}} \rightarrow 2$$

$$\frac{\delta L}{\delta b_{21}} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta b_{21}}$$

$$\boxed{\frac{\delta L}{\delta b_{21}} = -2(y - \hat{y})} \rightarrow 3$$



$$\frac{\delta L}{\delta W_{11}^1} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta O_{11}} \times \frac{\delta O_{11}}{\delta W_{11}^1}$$

$$\frac{\delta L}{\delta W_{21}^1} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta O_{11}} \times \frac{\delta O_{11}}{\delta W_{21}^1}$$

$$\frac{\delta L}{\delta b_{11}} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta O_{11}} \times \frac{\delta O_{11}}{\delta b_{11}}$$

② $\rightarrow O_{11}$

$$\frac{\delta L}{\delta W_{12}^1} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta O_{12}} \times \frac{\delta O_{12}}{\delta W_{12}^1}$$

$$\frac{\delta L}{\delta W_{22}^1} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta O_{12}} \times \frac{\delta O_{12}}{\delta W_{22}^1}$$

$$\frac{\delta L}{\delta b_{12}} = \frac{\delta L}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta O_{12}} \times \frac{\delta O_{12}}{\delta b_{12}}$$

③ $\rightarrow O_{12}$

$$\frac{\delta \hat{y}}{\delta o_{11}} = \frac{\delta}{\delta o_{11}}\left[w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}\right]$$

$$= w_{11}^2$$

$$\frac{\delta \hat{y}}{\delta o_{12}} = \frac{\delta}{\delta o_{12}}\left[w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}\right]$$

$$= w_{21}^2$$

$$\frac{\delta o_{11}}{\delta w_{11}^1} = \frac{\delta}{\delta w_{11}^1}\left[\overset{x_{i1}}{iq}\, w_{11}^1 + \overset{x_{i2}}{cgpa}\cdot w_{21}^1 + b_{11}\right]$$

$$= iq \rightarrow x_{i1}$$

$$\frac{\delta o_{11}}{\delta w_{21}^1} = x_{i2}$$

$$\frac{\delta o_{11}}{\delta b_{11}} = 1$$

$$\frac{\delta o_{12}}{\delta w_{12}^1} = \frac{\delta}{\delta w_{12}^1}\left[x_{i1} w_{12}^1 + x_{i2} w_{22}^1 + b_{12}\right]$$

$$= x_{i1}$$

$$\frac{\delta o_{12}}{\delta w_{22}^1} = \frac{\delta}{\delta w_{22}^1}\left[x_{i1} w_{12}^1 + x_{i2} w_{22}^1 + b_{12}\right]$$

$$= x_{i2}$$

$$\frac{\delta o_{12}}{\delta b_{12}} = \frac{\delta}{\delta b_{12}}\left[x_{i1} w_{12}^1 + x_{i2} w_{22}^1 + b_{12}\right]$$

$$= 1$$

$$\frac{\delta L}{\delta w_{11}^1} = -2(y - \hat{y}) w_{11}^2 x_{i1} \rightarrow 4$$

$$\frac{\delta L}{\delta w_{21}^1} = -2(y - \hat{y}) w_{11}^2 x_{i2} \rightarrow 5$$

$$\frac{\delta L}{\delta b_{11}} = -2(y - \hat{y}) w_{11}^2 \rightarrow 6$$

$$\frac{\delta L}{\delta w_{12}^1} = -2(y - \hat{y}) w_{21}^2 x_{i1} \rightarrow 7$$

$$\frac{\delta L}{\delta w_{22}^1} = -2(y - \hat{y}) w_{21}^2 x_{i2} \rightarrow 8$$

$$\frac{\delta L}{\delta b_{12}} = -2(y - \hat{y}) w_{21}^2 \rightarrow 9$$

Steps (once again)

0.) Weight / bias Initialization

for i in 100/1000 :

1.) for i in range (4): (4 → no. of student)

1a) 1 student → Forward propagation → Predict

1b) Loss Calculate (mse)

1c.) Adjust all weights & bias

→ Gradient Descent

Until loss becomes min, this keeps on repeating