Perceptron Loss function

Perceptron: $L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$

Hinge loss: $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$

SGD → optimization

$$E(w, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

where L is a loss function that measures model (mis) fit & R is regularization that penalizes model complexity; $\alpha > 0$ is a non-negative hyperparameter that controls the regularization strength.

$$L(w_1, w_2, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w_1, w_2) \quad \text{← Regularization}$$

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

$f(x_i) = w_1 x_1 + w_2 x_2 + b$

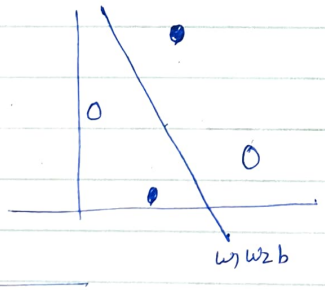
$$L = \underset{w_1, w_2, b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i))$$

minimum

Gradient descent

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

Eg



cgpa	iq	placed
7	8	1
6	8	-1
4	2	1
1	1	-1

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

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

x_{ij}: row, col

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

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

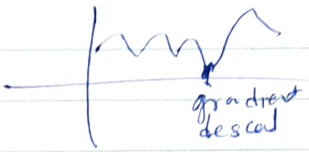
$f(x_2) = w_1 x_{21} + w_2 x_{22} + b$

$$\max(0, -y_i f(x)) = \begin{cases} -y_i f(x_i) & -y_i f(x_i) \geq 0 \\ 0 & -y_i f(x_i) < 0 \end{cases}$$

y_i	$\hat{y}_i$	$\max(0, -y_i f(x_i))$			
1	1	0		$w_1x_1 + w_2x_2 + b \geq 0$	i.e. $\max(0, -ve) = 0$
-1	-1	0	y_i = -ve, $f(x_i)$ = -ve	$w_1x_1 + w_2x_2 + b < 0$	i.e. $\max(0, -ve) = 0$
1	-1		y_i = +ve	$f(x_i)$ = -ve	$-y_i f(x_i)$ = +ve, i.e. $\max(0, +ve) = +ve$
-1	1		y_i = -ve	$f(x_i)$ = +ve	$-y_i f(x_i)$ = +ve, i.e. $\max(0, +ve) = +ve$

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

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



for i in epochs: $\eta = 0.01$

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

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

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

$$\left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial b}\right]$$

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

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

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

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

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

Loss Function : Binary Cross Entropy

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

⇒ Perceptron $\int$ Step function use as a perceptron

⇒ Perceptron == Logistic Regression if Activation Fun^n = Sigmoid
Loss Fun^n = Binary cross entropy

⇒ Activation function = Softmax
Loss function = Categorical Cross entropy

For multiclass
Softmax regression

$$L = \sum_{j=1}^{m} y_j \cdot \log(\hat{y}_j)$$

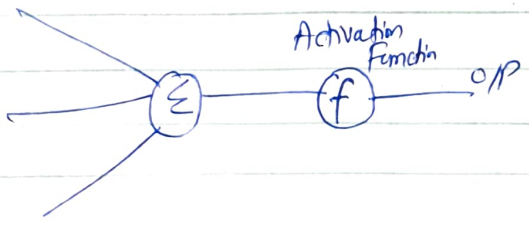
$$f = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

⇒ For Regression Activation Function → Linear function (z → z)
Loss function → MSE $(y_i - \hat{y}_i)^2$ Linear regression

⇒ Perceptron is a mathematical model which can be used in multiple ways because it is flexible by design

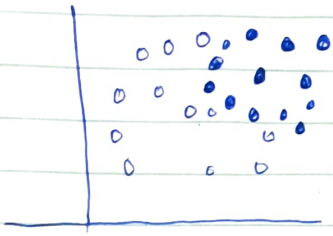
ie

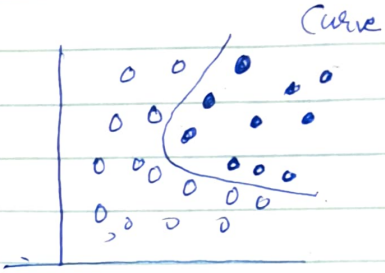
Loss function	Activation	Output
Hinge loss	Step	perceptron → binary classifier (o/p −1 or 1)
Log loss (Binary cross entropy)	Sigmoid	Logistic regression → binary classifn (o/p 0 → 1)
Categorical Cross entropy	Softmax	Softmax regression → multiclass classifn (o/p probability)
MSE	linear	Linear regression (o/p = number)

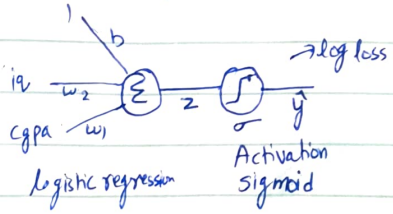


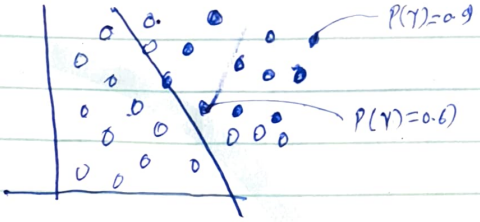
Multi Layer Perceptron (MLP)

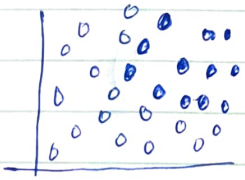
Date
Page

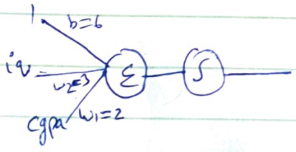


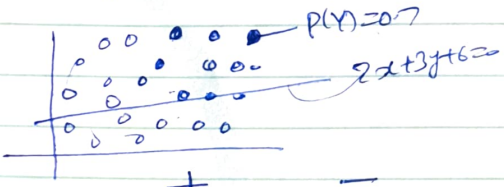


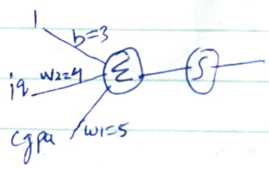




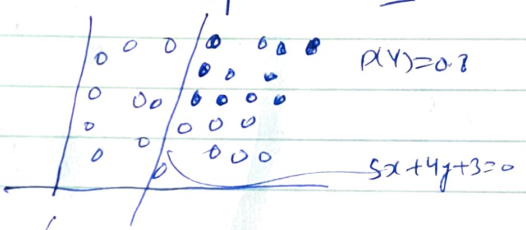


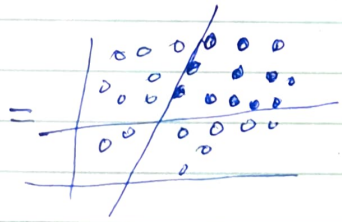




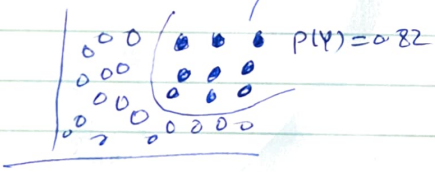


+ =





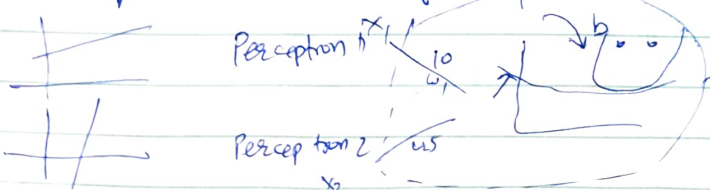
= smoothy =



$0.7+0.8=1.5$ $\sigma(z)=\frac{1}{1+e^{-z}}=0.82$

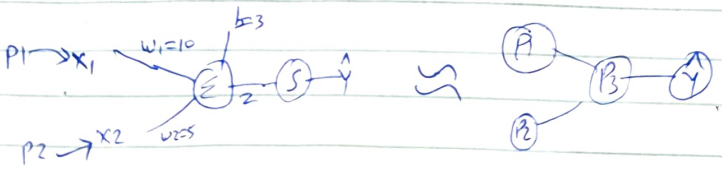
$\sigma(1.5)=\frac{1}{1+e^{-2}}$

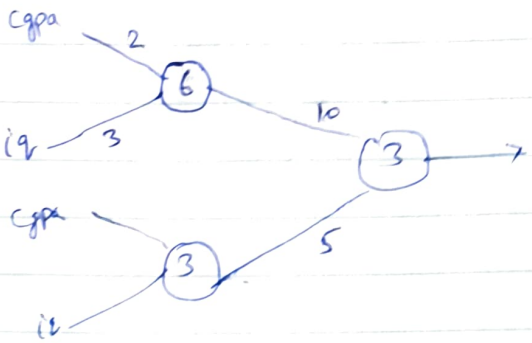
➡ linear Combination of two perceptron

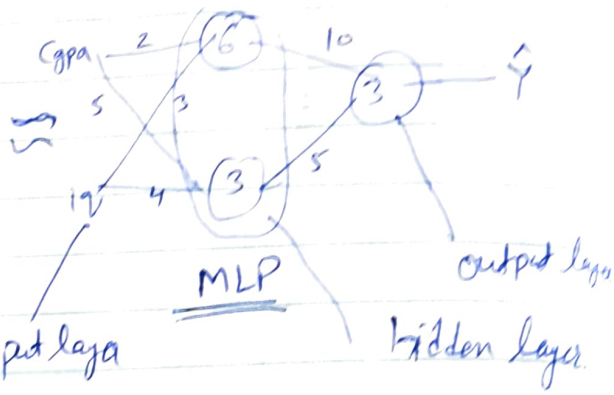


weighted $z=0.7\times10+0.8\times5+b$ weightage of Perceptron is more

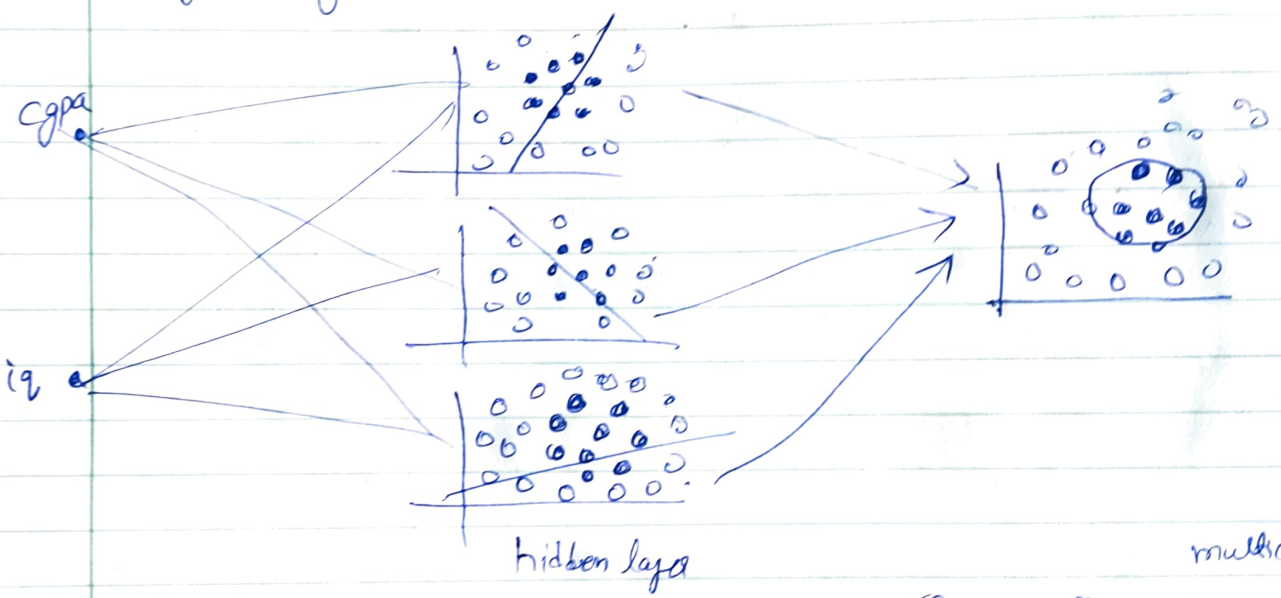
$\sigma(z)=\frac{1}{1+e^{-z}}$

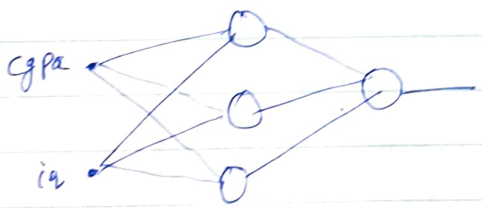


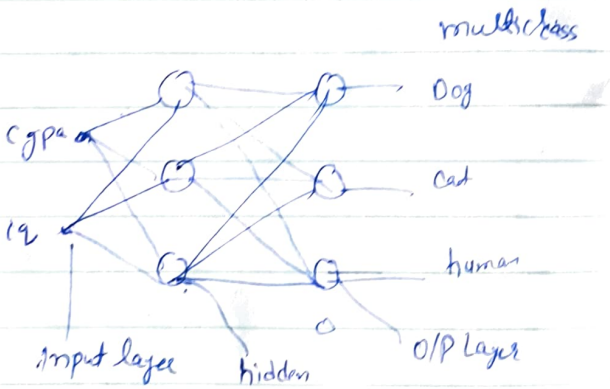


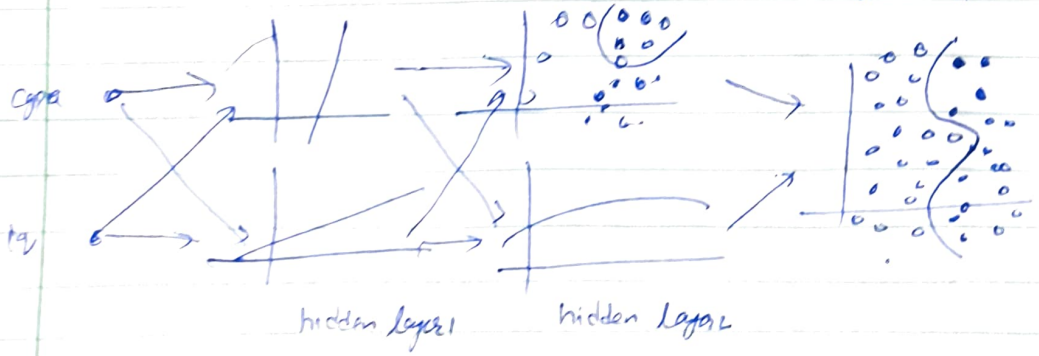


⇒ By using multiple perceptron we can capture non-linearity







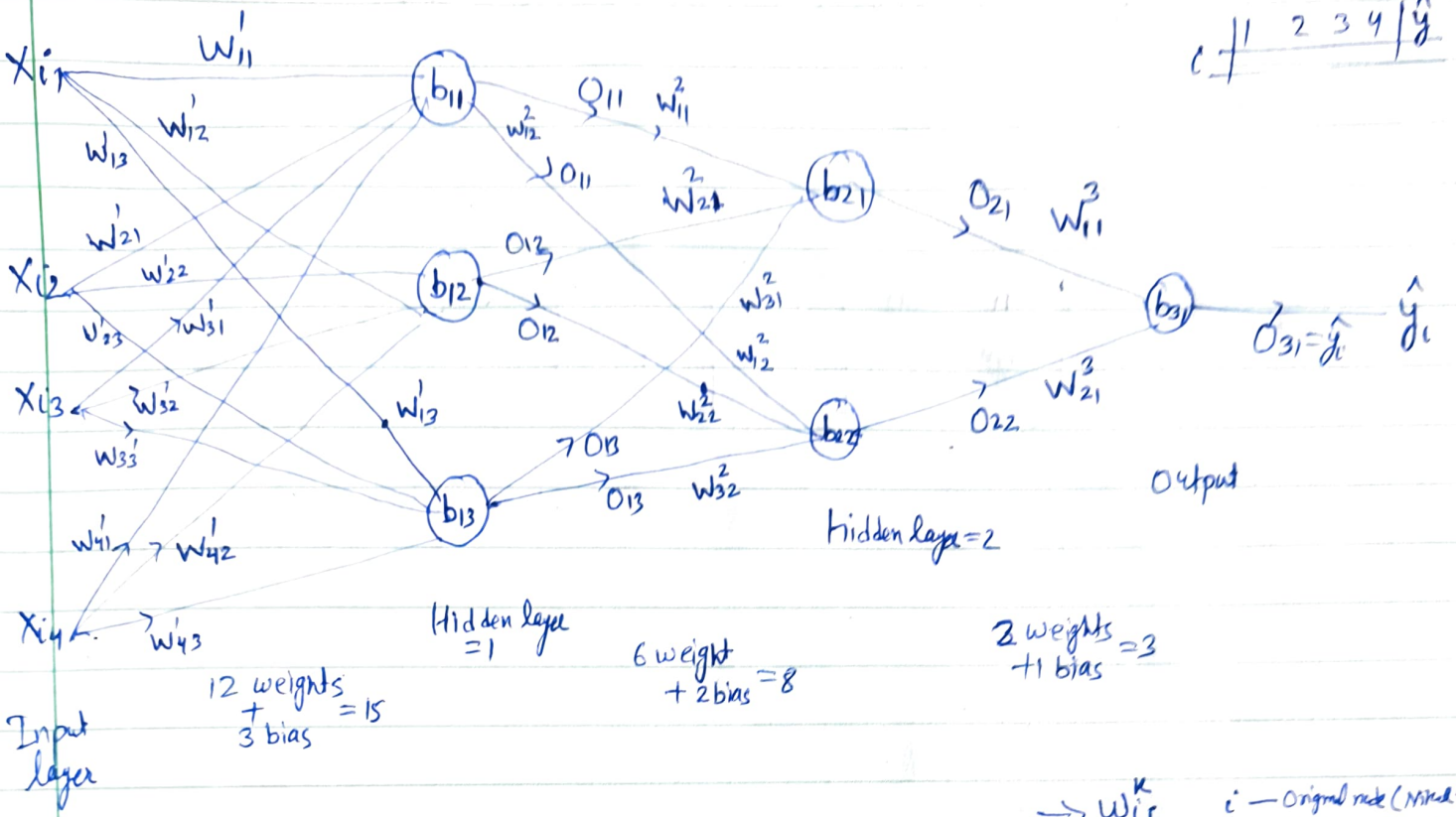


Demo at https://playground.tensorflow.org/

Multilayer Perceptron Notation

Data → {m x n}
m → rows
n = 4
each row denoted by X_i

i	1	2	3	4	$\hat{y}$



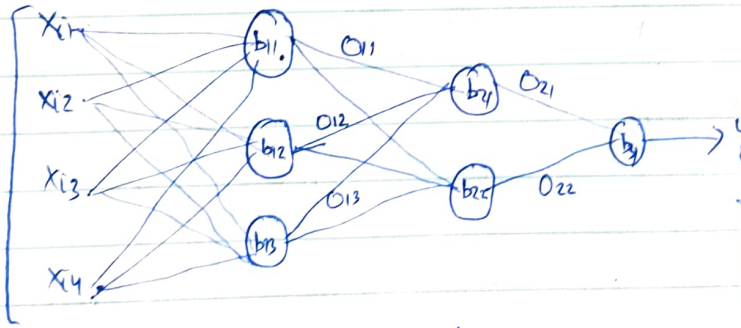
Trainable parameter = 15 + 8 + 3 = 26

$\Rightarrow w_{ij}^{k}$ — i → original node (where edge starts), j → destination node (where edge ends)

$\Rightarrow b_{ij}$ — i → layer, j → node

$\Rightarrow$ Output O_{ij}

$a^{[0]}$ =



Forward Propagation

How a NN Predicts O/P?

# of trainable parameters

cgpa	iq	10th	12th	Placed
7.2	72	69	81	1
8.1	92	75	76	0

Prediction $\Rightarrow \sigma(W^T X + b)$

Layer #1

$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{43} \end{bmatrix}^{T} \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} w'_{11}x_{i1} + w'_{21}x_{i2} + w'_{31}x_{i3} + w'_{41}x_{i4} \\ w'_{12}x_{i1} + w'_{22}x_{i2} + w'_{32}x_{i3} + w'_{42}x_{i4} \\ w'_{13}x_{i1} + w'_{23}x_{i2} + w'_{33}x_{i3} + w'_{43}x_{i4} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$(4\times3)^T \rightarrow 3\times4$; 4×1 ; 3×1

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

$$= \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} \leftarrow a^{[1]}$$

Layer#2

$$\underset{3\times 2 \leftrightarrow 2\times 3}{\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix}^T} \underset{3\times 1}{\begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix}} + \underset{2\times 1}{\begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}}$$

$$= \sigma\left(\begin{bmatrix} w^2_{11}o_{11} + w^2_{21}o_{12} + w^2_{31}o_{13} + b_{21} \\ w^2_{12}o_{11} + w^2_{22}o_{12} + w^2_{32}o_{13} + b_{22} \end{bmatrix}\right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} \leftarrow a^{[2]}$$

Layer#3

$$\underset{2\times 1 \leftrightarrow 1\times 2}{\begin{bmatrix} w^3_{11} \\ w^3_{21} \end{bmatrix}^T} \underset{2\times 1}{\begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix}} + b_{31}$$

$$= \sigma\left(\left[w^3_{11}o_{21} + w^3_{21}o_{22} + b_{31}\right]\right) = \hat{y}_i \leftarrow a^{[3]}$$

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

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

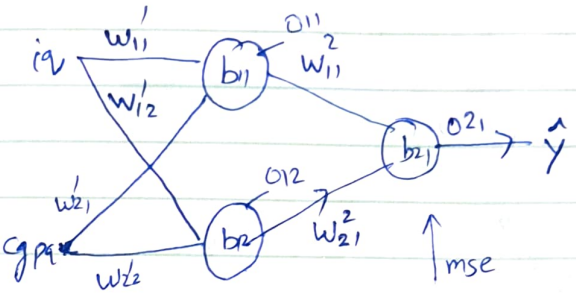
($a^{[1]}$, $a^{[2]}$, $a^{[3]} = \hat{y}$)

Backpropagation

Date
Page

Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of ANN using gradient descent. Given an ANN and an error function, the method calcules the gradient of the error function w.r.t. the neural network's weights.

iq	cgpa	lpa
80	8	3
60	9	5
70	5	8
120	7	11



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

9 trainable parameter

0) Initialize w, b — (w, b) Random Value

$w \to 1, \quad b \to 0$

Step 1: You select a point (row) ↳ student

2. Predict (lpa) → forward propagation [Dot product]

e.g $\hat{y}_1 = 18$ lpa

$(y - \hat{y})$

$(3-18)^2 = 225$ error

$\hat{y} \leftarrow 18$

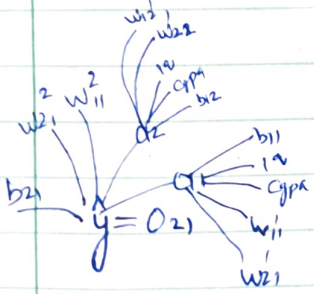
$\hat{y} \to 19$

3. Choose a loss function

4. Weight and bias update using ↳ Gradient Descent

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

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



derivative of Loss wrt weight & bias

$$w_{11}^2{}_{new} = w_{11}^2{}_{old} - \eta \frac{\partial L}{\partial w_{11}^2}$$

$$w_{21}^2{}_{new} = w_{21}^2{}_{old} - \eta \frac{\partial L}{\partial w_{21}^2}$$

$$b_{21}{}_{new} = b_{21}{}_{old} - \eta \frac{\partial L}{\partial b_{21}}$$

9 derivative

$$\frac{\partial L}{\partial w_{11}^2}, \frac{\partial L}{\partial w_{21}^2}, \frac{\partial L}{\partial b_{21}} \quad \frac{\partial L}{\partial w_{11}^1}, \frac{\partial L}{\partial w_{21}^1}, \frac{\partial L}{\partial b_{11}} \quad \frac{\partial L}{\partial w_{12}^1}, \frac{\partial L}{\partial w_{22}^1}, \frac{\partial L}{\partial b_{12}}$$

$$\min(i) \quad \frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial [O_{11} w_{11}^2 + O_{12} w_{21}^2 + b_{21}]}{\partial w_{11}^2}$$

$$= -2(y - \hat{y}) \times O_{11}$$

(ii) $$\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{21}^2} = -2(y - \hat{y}) \times O_{12}$$

(iii) $$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial b_{21}} = -2(y - \hat{y}) \times 1 = -2(y - \hat{y})$$

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

$$\hat{y} = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$$

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

(iv) $$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{11}^1}$$

$$= -2(y - \hat{y}) \cdot w_{11}^2 \cdot x_{i1}$$

(v) $$\frac{\partial L}{\partial w_{21}^1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w_{21}^1}$$

$$= -2(y - \hat{y}) w_{11}^2 \cdot x_{i2}$$

(vi) $$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{11}} \frac{\partial o_{11}}{\partial b_{11}}$$

$$= -2(y - \hat{y}) \cdot w_{11}^2 \cdot 1$$

(vii) $$\frac{\partial L}{\partial w_{12}^1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_{12}} \frac{\partial o_{12}}{\partial w_{12}^1}$$

$$= -2(y - \hat{y}) \cdot w_{21}^2 \cdot x_{i1}$$

(viii) $$\frac{\partial L}{\partial w_{22}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial w_{22}^1}$$

$$= -2(y - \hat{y}) w_{21}^2 \cdot x_{i2}$$

(ix) $$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial b_{12}}$$

$$= -2(y - \hat{y}) w_{21}^2 \cdot 1$$

$$\therefore \frac{\partial o_{11}}{\partial w_{11}^1} = \frac{\partial}{\partial w_{11}^1}[iq w_{11}^1 + cgpa\, w_{21}^1 + b_{11}] = iq = x_{i1}$$

$$\frac{\partial o_{11}}{\partial w_{21}^1} = \frac{\partial [\quad]}{\partial w_{21}^1} = x_{i2}$$

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

$$\frac{\partial o_{12}}{\partial w_{12}^1} = \frac{\partial}{\partial w_{12}^1}[iq w_{12}^1 + cgpa\, w_{22}^1 + b_{12}] = iq(x_{i1})$$

$$\frac{\partial o_{12}}{\partial w_{22}^1} = cgpa = x_{i2}$$

$$\frac{\partial o_{12}}{\partial b_{12}} = 1$$

Algorithm Steps

0) weights / bias → initialization — random: w=1, b=0

loop for epochs until converge

1) for i in range(4):

1(a) 1 Student → forward prop → predict (lpa)

1(b) Loss calculate (mse)

1(c) Adjust all weights & bias

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$ 9 times

epoch=5

for i in range(epochs):

for j in range(X.shape[0]):

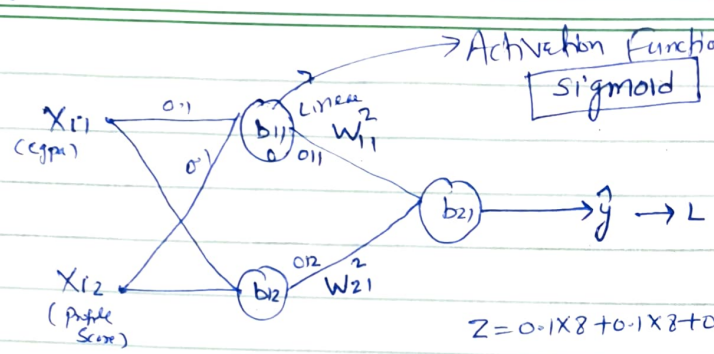
1(a), 1(b), 1(c)

calc. avg loss for the epoch

$\frac{(L_1 + L_2 + L_3 + L_4)}{4}$ = avg loss, epoch

Classification Example

Eggs	Profile Score	Placement
8	8	1
7	9	1
6	10	0
5	5	0



$Z = 0.1 \times 8 + 0.1 \times 8 + 0$

$\sigma(z) = o_{11}$

$$\text{Loss} = -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$$ Binary cross entropy

$\hat{y} = \sigma(z)$

(i) $\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{11}^2} = \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})} \times \hat{y}(1-\hat{y}) \times \frac{\partial z}{\partial w_{11}^2} = -(y-\hat{y}) \times \frac{\partial z}{\partial w_{11}^2}$ $\quad L \to \hat{y} \to z_{final}$

$= -(y-\hat{y}) o_{11}$ $\quad \frac{\partial z}{\partial w_{11}^2}$

$= -(y-\hat{y}) o_{11} (w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21})$

(ii) $\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{21}^2} = -(y-\hat{y}) o_{12}$

Common

(iii) $\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial b_{21}} = -(y-\hat{y}) \cdot 1$

$\therefore \frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} [-y \log \hat{y} - (1-y)\log(1-\hat{y})] = \frac{-y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})} = \frac{-y(1-\hat{y}) + \hat{y}(1-y)}{\hat{y}(1-\hat{y})} = \frac{-y + \hat{y}y + \hat{y} - \hat{y}y}{\hat{y}(1-\hat{y})}$

$= \frac{-(y-\hat{y})}{\hat{y}(1-\hat{y})}$

$\therefore \frac{\partial \hat{y}}{\partial z} = \frac{\partial(\sigma(z))}{\partial z} = \sigma(z)[1-\sigma(z)] = \hat{y}(1-\hat{y})$

$$z_f = w_{11}^2 o_{11} + w_{21}^2 o_{12} + b_{21}$$ $\quad \frac{\partial z}{\partial w_{11}^2} = o_{11}$

(iv) $\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{prev}} \times \frac{\partial z_{prev}}{\partial w_{11}^1} = -(y-\hat{y}) \times w_{11}^2 \times o_{11}(1-o_{11}) x_{i1}$ $\quad L \to \hat{y} \to z_f \to o_{11} \to z_{prev} \to w_{11}^1$

$\frac{\partial z_f}{\partial o_{11}} = w_{11}^2$ $\quad \frac{\partial o_{11}}{\partial z_{prev}} = o_{11}(1-o_{11})$ $\quad \times x_{i1}$

(v) $\frac{\partial L}{\partial w_{21}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f} \cdot \frac{\partial z_f}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial z_{prev}} \cdot \frac{\partial z_{prev}}{\partial w_{21}^1} = -(y-\hat{y}) w_{11}^2 o_{11}(1-o_{11}) x_{i2}$

(vi) $\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f} \cdot \frac{\partial z_f}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial z_{prev}} \cdot \frac{\partial z_{prev}}{\partial b_{11}} = -(y-\hat{y}) w_{11}^2 o_{11}(1-o_{11}) \times 1$

$o_{11} = \sigma(z_{prev})$

$\frac{\partial o_{11}}{\partial z_{prev}} = \sigma(z_{prev})[1-\sigma(z_{prev})]$

$= o_{11}[1-o_{11}]$

$$z_p = w_{11}^1 x_{i1} + w_{21}^1 x_{i2} + b_{11}$$

$Z_f = w^2_{11} O_{11} + w^2_{21} O_{12} + b_{21}$ $\quad Z_{prev} = w'_{12} x_{i1} + w'_{22} x_{i2} + b_{12}$

$L \to \hat{y} \to Z_f \to O_{12} \to Z_p \to w'_{12}$

Date
Page

(vii) $\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial Z_f} \cdot \frac{\partial Z_f}{\partial O_{12}} \frac{\partial O_{12}}{\partial Z_{prev}} \cdot \frac{\partial Z_{prev}}{\partial w'_{12}} = -(y-\hat{y})\, w^2_{21}\, O_{12}(1-O_{12})\, x_{i1}$

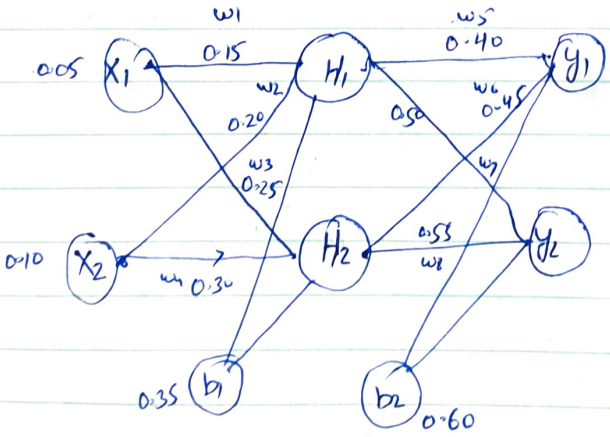
(viii) $\frac{\partial L}{\partial w'_{22}} = -(y-\hat{y})\, w^2_{21}\, O_{12}(1-O_{12})\, x_{i2}$

(ix) $\frac{\partial L}{\partial b_{12}} = -(y-\hat{y})\, w^2_{21}\, O_{12}(1-O_{12})$

$\frac{\partial L}{\partial b}$ = rate of change of L w.r.t b

$\frac{\partial L}{\partial b}$ = -ve $\quad b \uparrow \; L \downarrow$ $\qquad$ $\frac{\partial L}{\partial b}$ = +ve $\quad b \uparrow \; L \uparrow$

$w_{new} = w_{old} - \eta \frac{\partial L}{\partial b}$



e.g

Target Values
T1 = 0.01
T2 = 0.99

First Calculate the values of H1 & H2 by a forward pass

$H1 = X1 \times 0.15 + X2 \times 0.20 + b1$
$= 0.05 \times 0.15 + 0.10 \times 0.20 + 0.35 = 0.3775$

$H1_{final} = \sigma(H1) = \frac{1}{1+e^{-0.3775}} = 0.593269992$

$H2 = 0.05 \times 0.25 + 0.30 \times 0.10 + 0.35 = 0.3925$

$H2_{final} = \sigma(H2) = \frac{1}{1+e^{-0.3925}} = 0.596884378$

$y_1 = H_1 \times 0.40 + H_2 \times 0.45 + 0.60$

$= 0.59326992 \times 0.40 + 0.59688 \times 0.45 + 0.60 = 1.10590597$

$\sigma(y_1) = y_{1final} = \frac{1}{1+e^{-1.105905}} = 0.75136507$

$y_2 = 0.59326992 \times 0.50 + 0.59688 \times 0.55 + 0.60 = 1.2249214$

$\sigma(y_2) = y_{2final} = \frac{1}{1+e^{-1.2249217}} = 0.772928465$

$E_{total} = \frac{1}{2}\sum(target - output)^2 = \frac{1}{2}[0.01 - 0.751365]^2 + \frac{1}{2}[0.99 - 0.772928]^2$

$= 0.29837111$

Now we will backpropagate this error to update the weights using a backward pass

$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial y_{1final}} \times \frac{\partial y_{1final}}{\partial y_1} \times \frac{\partial y_1}{\partial w_5}$

$= 0.74136507 \times 0.186815602 \times 0.59326992$

$= 0.0821670407$

$y_1 = w_5 \times H_{1final} + w_6 H_{2final} + b_2$

$\frac{\partial y_1}{\partial w_5} = H_{1final} = 0.59326992$

$\frac{\partial y_{1final}}{\partial y_1} = \frac{\partial \sigma(y_1)}{\partial y_1} = \sigma(y_1)[1 - \sigma(y_1)]$

$= y_{1final}[1 - y_{1final}]$

$= 0.751365[1 - 0.751365]$

$= 0.186815602$

$\frac{\partial E}{\partial y_{1final}} = \frac{\partial \frac{1}{2}\sum(y_1 - \hat{y}_{1final})^2}{\partial y_{1final}}$

$= -(y_1 - \hat{y}_{1final})$

$= -(0.01 - 0.751365)$

$= 0.74136507$

Similarly Cal. $\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_3}, \frac{\partial E}{\partial w_4}, \frac{\partial E}{\partial w_5}, \frac{\partial E}{\partial w_6}, \frac{\partial E}{\partial w_7}$

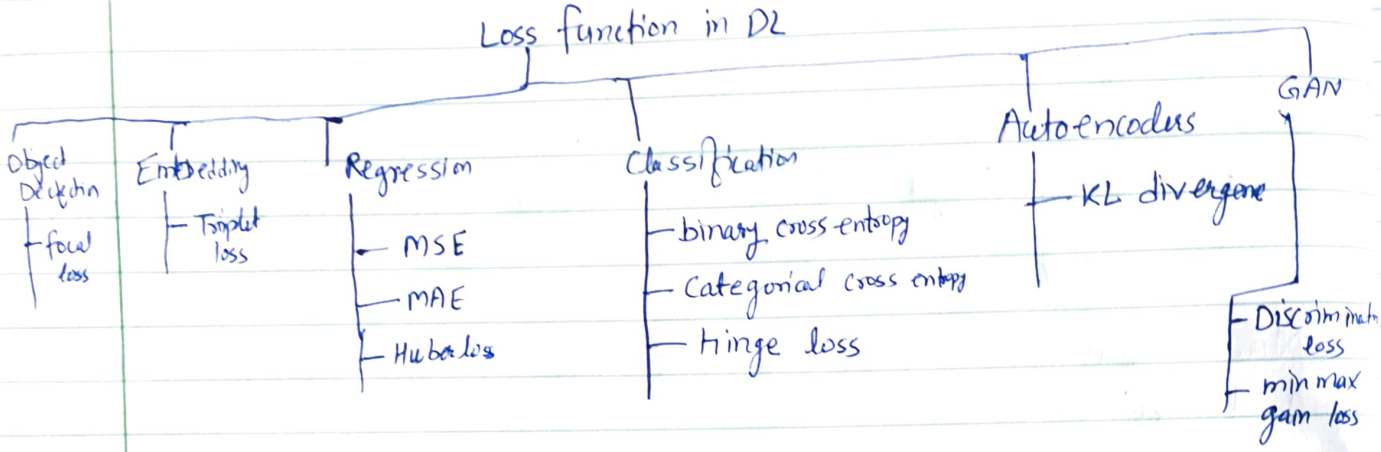
$w_{5new} = w_{5old} - \eta \frac{\partial E}{\partial w_5}$ Let $\eta = 0.5$

$= 0.4 - 0.5 \times 0.0821670$

$= 0.35891648$

Loss Function in Deep Learning

Loss Function: Loss function is a method of evaluating how well your algorithm is modelling your dataset.



Loss function → single training example

Cost function → Whole training example

Loss function $= (y_i - \hat{y}_i)^2$

cost function $= \frac{1}{n} \Sigma (y_i - \hat{y}_i)^2$

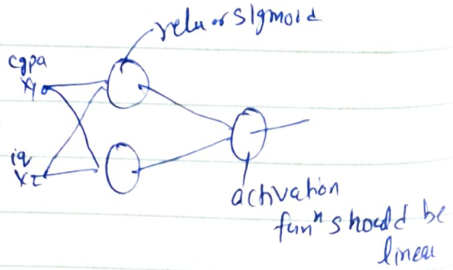
1. Mean Squared Error (MSE)

squared loss L2 loss $= (y_i - \hat{y}_i)^2 = (\text{true} - \text{predicted})^2$

quadratic

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

Advantages:	Disadvantage
1. Easy to interpret	1. Error unit (Squared) — different
2. Differentiable (GD)	2. Not Robust to outliers
3. 1 local minima	



2. Mean Absolute Error (MAE) → L1 loss

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

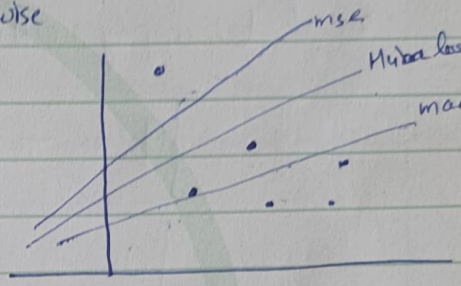
$C = \frac{1}{n} \Sigma |y_i - \hat{y}_i|$

Advantage	Disadvantage
1. Intuitive and easy	1. Not differentiable. We cannot apply gradient descent
2. Unit is same	
3. Robust to outliers	

3. **Huber Loss**

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

mse — outlier input
mae — normal points



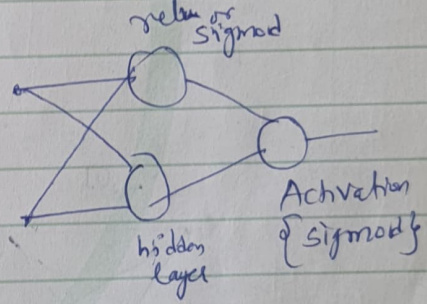
δ = hyperparameter

It lies b/w MSE & MAE

4. **Binary Cross Entropy** or Log loss — used for classification

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

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



Advantage	Disadvantage
1. Differentiable	1. Multiple local minimum
	2. intuitive (not)

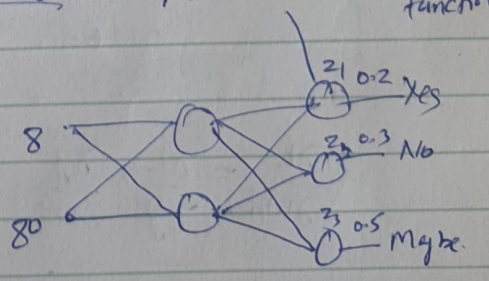
5. **Categorical Cross Entropy** [used in Softmax Regression]

[Multi class] {Classification}

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

where K is # classes in the data

Activation funtin = Softmax function



$$L = -y_1\log\hat{y}_1 - y_2\log(\hat{y}_2) - y_3\log\hat{y}_3$$

[0.2 0.3 0.5] [1 0 0] → $L = -1\times\log(0.2)$

[0.3 0.6 0.1] [0 1 0] → $L = -1\log(0.6)$

cgpa	iq	placed
8	80	Yes 1
6	60	No 2
7	70	maybe 3

OHE
Yes No May

6. **Sparse Categorical Cross Entropy**

assigning integer value to classes

[0.1 0.4 0.5]

$L = -1 \times \log(0.1)$

It will be fast. Because we don't need to apply OHE.

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

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

cgpa | placed (OHE)

7 | Yes | 1

8 | No | 2

6 | Maybe | 3

[0.1 0.4 0.6] if second label

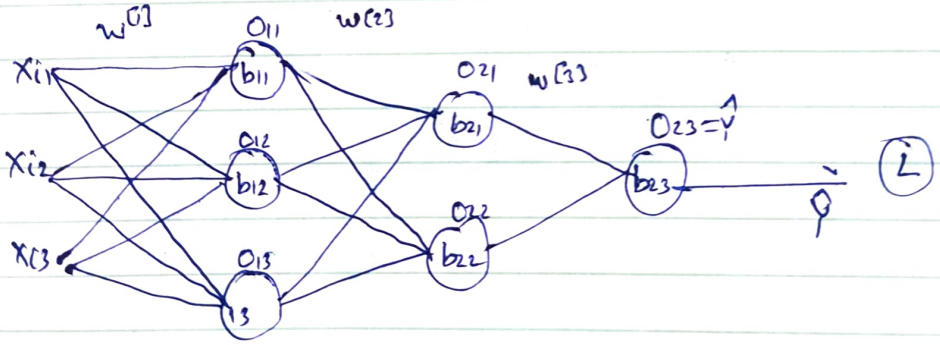
$L = -1 \times \log(0.4)$

3rd

[0.6 0.2 0.2]

$L = -1 \times \log(0.2)$

MLP Memoization



3×3 | 3×2 | 2×1

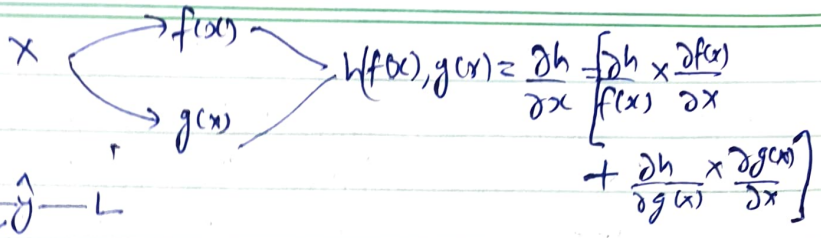
9+3=12 | 6+2=8 | 2+1=3 | 23

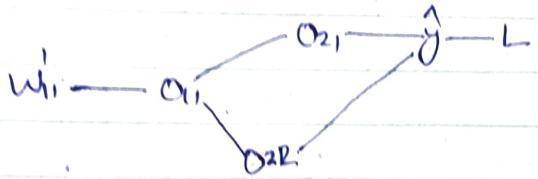
$$\frac{\partial L}{\partial w_{11}^3} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{11}^3}$$

$L \rightarrow \hat{Y} \rightarrow W_{11}^3$

$$\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial w_{11}^2}$$

$L \rightarrow \hat{Y} \rightarrow O_{21} \rightarrow W_{21}^2$





$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \left[\frac{\partial \hat{y}}{\partial O_{21}} \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w'_{11}} + \frac{\partial \hat{y}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w'_{11}} \right]$$

for larger hidden layer it will be more complex.

∴ We will save the intermediate derivative in memory for further processing.

Keras, TF → Backpropagation → Chain diff(rule) + memoization

Gradient Descent in Neural Network

Batch GD (Vanilla GD) — entire dataset → update

Stochastic GD → For each row → update
→ help the algo to move out of local minima

Mini batch GD

⇒ Which is faster (given same no. of epochs
↳ Batch GD

⇒ which is the faster to converge (given same # epochs)
↳ stochastic

Vectorization:- epoch=10 ← Batch GD
for in range(10):
y = np.dot(X,W)+b
solution

dot → smart replacement → loops
↓
Vectorization → faster than loop

Fast
BGD > MBGD > SGD

Convergence BGD < MBGD < SGD

Vanishing Gradient Problem in ANN

Date
Page

As more layers using certain activation function are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train

When there are more layers in the network, the value of the product of derivative decreases until at some point the partial derivative of the loss function approaches a value close to zero, and the partial derivative vanishes. We call this the vanishing gradient Problem.

Exploding Gradient Problem:- If the gradients are large, the multiplication of these gradients will become huge over time. This results in the model being unable to learn & its behaviour becomes unstable. This problem is called the exploding gradient problem.

Solution of Vanishing Gradient Problem
- Change Activation function (use ReLU)
- Reduce model complexity (i.e reduce hidden layer)
- Proper weight initialization
- Batch Normalization
- Use Residual N/w

Solution of Exploding Gradient
- use Batch Normalization
- Use less no. of layers
- Carefully initialize weights
- Use gradient clipping

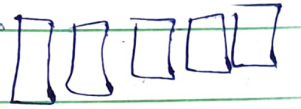
Conclusion:

During the backpropagation in the deep neural networks, the vanishing gradient problem occurs due to the sigmoid & tan activation function & the exploding gradient problem occurs due to large weights.

Vanishing Gradient Problem: In ML, the VGP is encountered when training ANN with Gradient based learning methods & backpropagation. In such methods during each iteration of training each of the NN weights receives an update proportional to the partial derivatives of the error function with rot the current weight. The problem is that in some cases, the gradient will be vanishingly

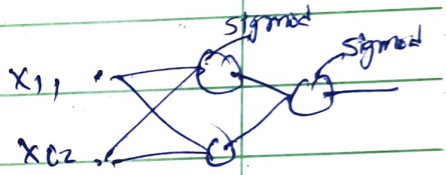
small, effectively preventing the weight from changing its value In the worst case, this may completely stop the NN from further training.

(i) $0.1 \times 0.1 \times 0.1 \times 0.1 = 0.0001$ ← VGP

(ii) Deep NN → 

(iii) Sigmoid / tanh → Activation function → VGP occur

$$w_n = w_o - \eta \frac{\partial L}{\partial w}$$



$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial z} \times \frac{\partial z}{\partial a_1} \times \frac{\partial a_1}{\partial w_{11}}$$

→ Sigmoid derivative

This will be very small

0.2×2.1

0.2×1

if $w_o = 1$

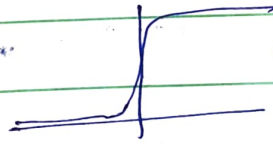
$w_n = 1 - 0.01 \times 0.0001$

$w_n = 0.9999$

Change is not occurs (1 ≈ 0.9999)

$\frac{\partial L}{\partial w_{11}} = 0.00001$

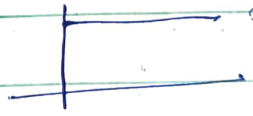
Deep learning failed



→ How to recognize the Vanishing Gradient Problem Occur

(i) Loss function change not occuring then VGP

(ii) Draw Graph btw weights Vs epoch.



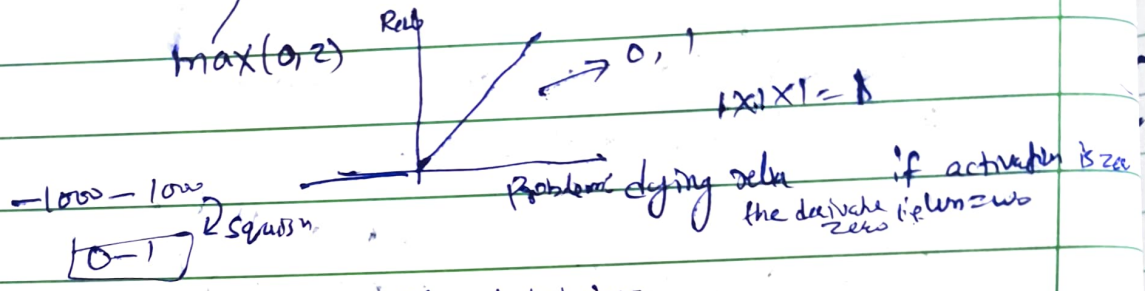
NAME :

S. No. | Date | Topic/Chapter

Use ReLU to avoid VGP

ReLU Activation function

max(0,z)



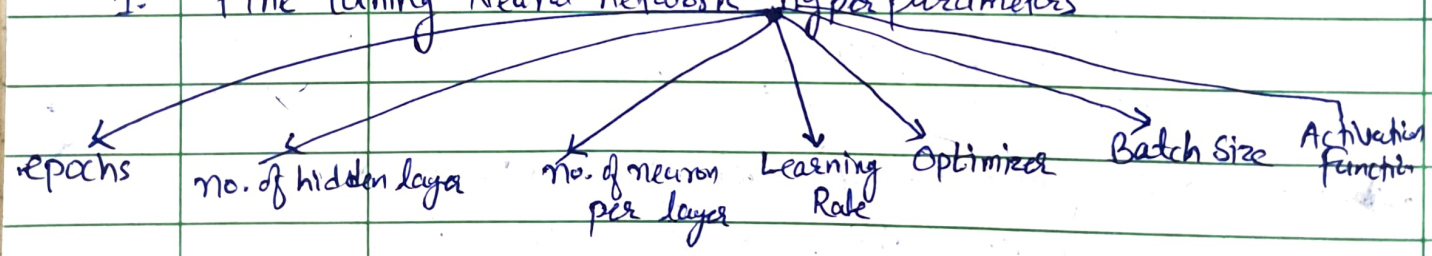
(iii) Proper weight Initialization → Gloarot technique
→ Xavier technique.

(iv) Batch Normalization → layer →

(v) Residual Network
└ building block → CNN Resnet
→ ANN

HOW TO IMPROVE THE PERFORMANCE OF A NEURAL NETWORK

1. Fine tuning Neural network hyper parameters



(2) By Solving problems
- Vanishing/Exploding gradient Problem
- Not enough data
- slow training
- Overfitting.