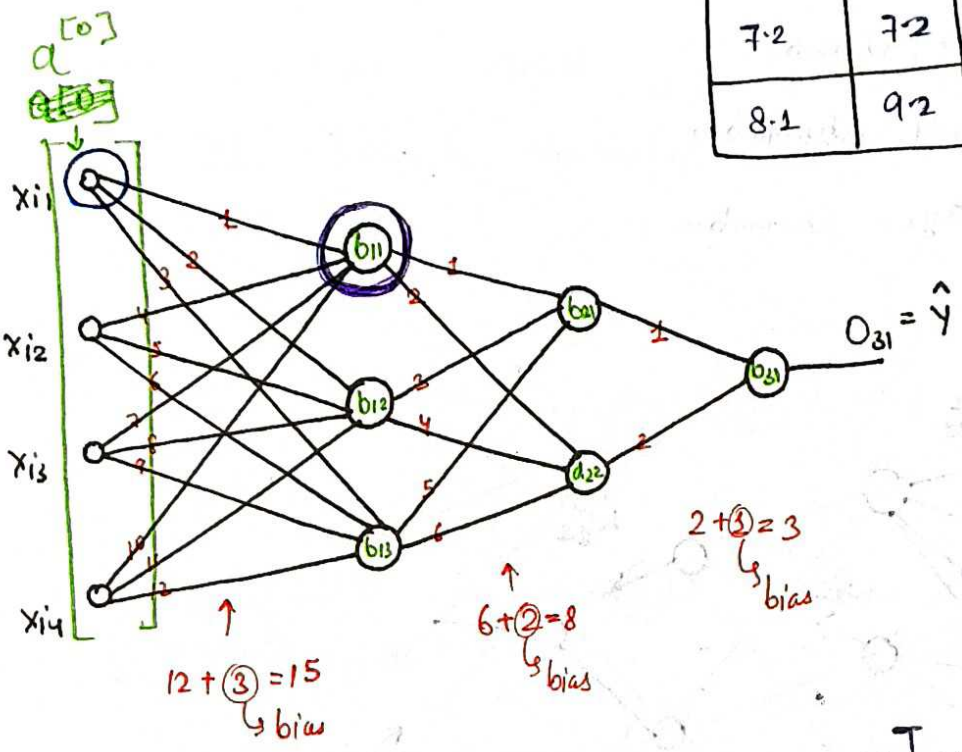
Forward Propogation

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

# of trainable parameters



prediction $\rightarrow \sigma(W^T x + b)$

Layer #1

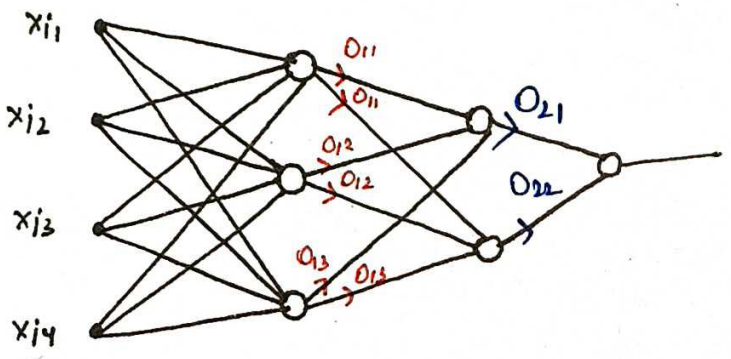
$$\begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \\ w^1_{41} & w^1_{42} & w^1_{43} \end{bmatrix}^T \cdot \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$4\times3^T \longrightarrow 3\times4$ $\quad$ 4×1 $\quad$ 3×1

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

$$= \sigma\left(\begin{bmatrix} w^1_{11}x_{i1} + w^1_{21}x_{i2} + w^1_{31}x_{i3} + w^1_{41}x_{i4} + b_{11} \\ w^1_{12}x_{i1} + w^1_{22}x_{i2} + w^1_{32}x_{i3} + w^1_{42}x_{i4} + b_{12} \\ w^1_{13}x_{i1} + w^1_{23}x_{i2} + w^1_{33}x_{i3} + w^1_{43}x_{i4} + b_{13} \end{bmatrix}\right)$$

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



Layer #2

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

$3\times2 \xrightarrow{T} 2\times3$ $\quad 3\times1$ $\quad 2\times1$

$$= \sigma\left(\begin{bmatrix} w^2_{11}o_{11} + w^2_{21}o_{12} + w^2_{31}o_{13} + b_{21} \\ w^2_{12}o_1 + w^2_{22}o_2 + w^2_{32}o_{13} + b_{22} \end{bmatrix}\right) = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} \rightarrow a^{[2]}$$

Layer #3

$$\begin{bmatrix} w_{11}^3 \\ w_{21} \end{bmatrix}^T \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} + \begin{bmatrix} b_{31} \\ \downarrow \\ 1\times1 \end{bmatrix}$$

$2\times1 \xrightarrow{T} 1\times2 \qquad 2\times1$

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

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

Simple method

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

$a^{[1]}$

$a^{[2]}$

$a^{[3]}$

What is Loss function?

Loss function is a method of evaluating how well your algo is modelling your dataset.

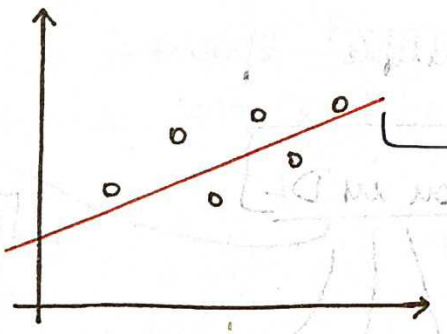
Loss function → high → Model → poor

↳ low → Model → great

Lossfunc → Mathematical func

L(parameter) → parameter change → loss function's value change

For eg:



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

$\hat{y}_i = mx_i + b$

$L(m,b) = \min$ → Find value of m, b where loss func is minimum

Why is loss function important?

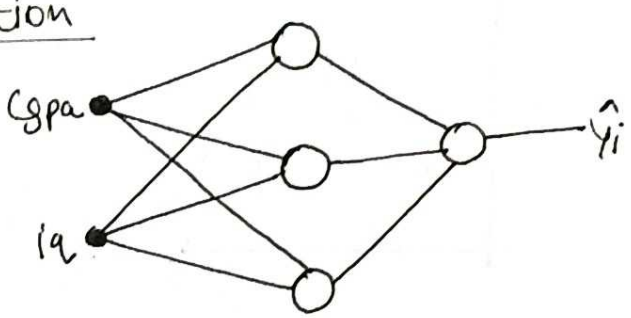
[You can't improve what you can't measure]

↳ Peter Drucker

Loss function Vs Cost function

cgpa	iq	y_i pckg	$\hat{y}_i$ Prediction
6.3	100	6.3	6.1
7.1	91	4.1	4
8.5	83	3.5	3.7
9.2	102	7.2	7

batch



Loss func^n → single row

$y_i = 6.3 \quad \hat{y}_i = 6.1$ → $(y_i - \hat{y}_i)^2$

Cost function

→ $\frac{1}{n} \sum (y_i - \hat{y}_i)$

$\frac{1}{4}[(6.1-6.3)^2 + (4.1-4)^2 + \text{---}]$ = Cost function

1. Mean Squared Error (MSE)

Squared loss or L_2 loss

→ $(y_i - \hat{y}_i)$

(true - predict)^2

$(6.3 - 6.1)^2 = \text{---}$

Why we use square?

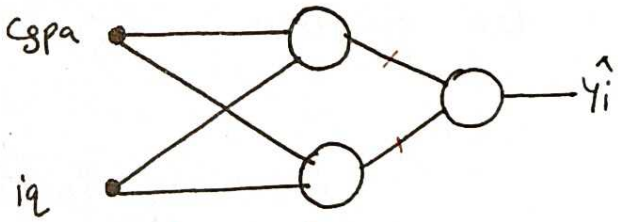
Cz if error is -ve then we cannot able add so we use square → -ve become +ve. but problem ↴

(true - predict) → 1 unit $\xrightarrow{\text{Square}}$ 1 unit

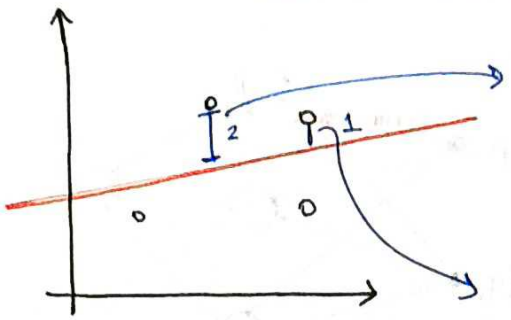
2 unit $\xrightarrow{\text{Square}}$ 4 unit

4 unit $\xrightarrow{\text{Square}}$ 16 unit

cgpa	iq	y_i pckg	$\hat{y}_i$ Pred	$y_i - \hat{y}_i$
6.3	100	6.3	6.1	0.2
7.1	91	4.1	4	0.1
8.5	83	3.5	3.7	-0.2
9.2	102	7.2	7	0.2



forward



error → heavy value drop in loss function

$2^2 = (4)$ → means double the error

$1^2 = (1)$ → means double the error 1 but small drop in loss func".

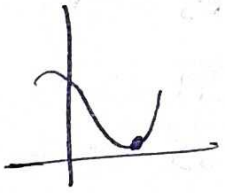
→ When large error → double large error

Advantage

1. Easy to intrepret
2. This loss fun" always differentiable (GD)
3. 1 local minima

(w,b) → one value → minimum

for low lossfun → global minima



Disadvantage

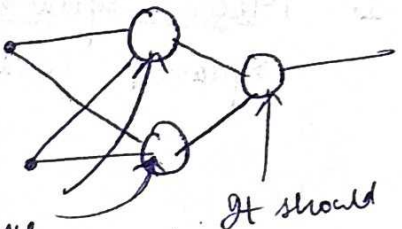
1. Error Unit → Squarred

lpa → loss fun" → $(lpa)^2$

unit → diff

2. not Robust to outlier

3. gpa



if any activation func"

It should be activation = linear

Cost function of MSE = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

{Batch of Points} → n

2. Mean Absolute Error (MAE) → L1 Loss

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

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

Advantage

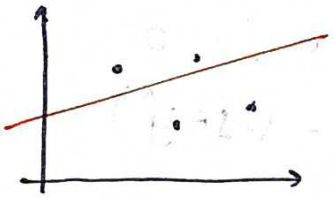
1) Intuitive and easy

2) Unit same

y_i → Lapa, $\hat{y}_i$ → LPA, $|y_i - \hat{y}_i|$ → LPA

3) Robust to outliers

⊙ → outlier

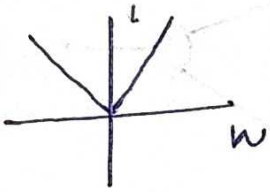


disadvantage

1) Not differentiable

↳ GD use differentiable

then use ← have to use Sub GD → Little bit tricky



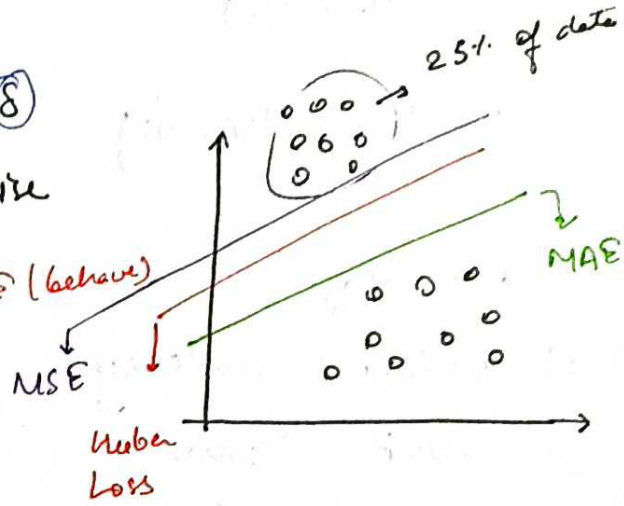
3. Huber loss

Data Normal → MSE (behave)

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

δ → hyperparameter

↳ data outlier → MAE (behave)



This loss func use when data contain many outlier.

4. Binary Cross Entropy

→ Classification

→ Two classes

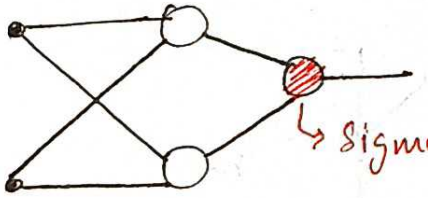
Cgpa	iq	Placement
8	80	1
7	70	0
6	60	0

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

y → actual value / target

$\hat{y}$ → NN prediction

condition →



↳ Sigmoid (Activation)

when we use binary cross entropy loss func then Output node activation func always be sigmoid.

Cost function:

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

5. Categorical Cross Entropy [used in Softmax Regression]

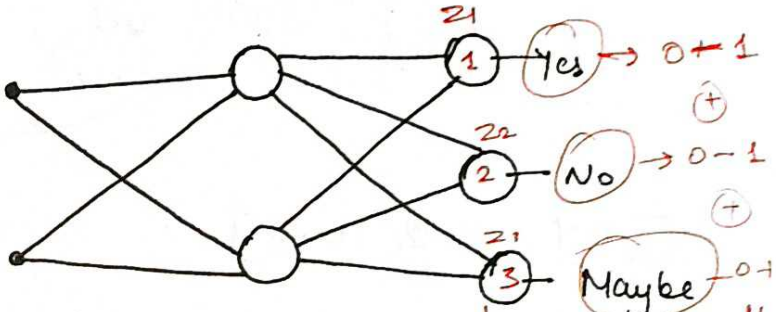
→ Multi-class classification

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

where k is # classes in the data

gpa	iq	placed
8	80	Yes
6	60	No
7	70	Maybe

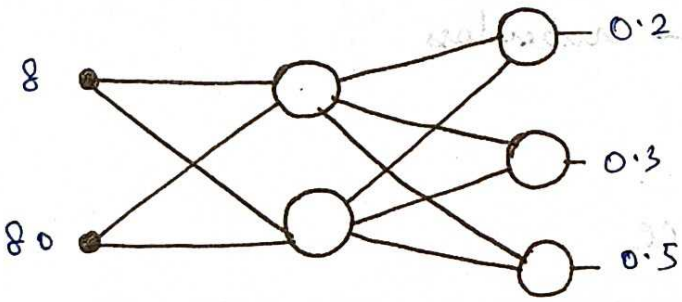
Yes	No	Maybe
1	0	0
0	1	0
0	0	1



$$\text{Maybe} \rightarrow \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$$\text{No} \rightarrow \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$$f(z_j) = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$$



$y = [1 \quad 0 \quad 0]$

$\hat{y} = [0.2 \quad 0.3 \quad 0.5]$

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

$$L = -1 \log(0.2) - 0 - 0$$

$\log(0.2)$ → loss of first point

Update weight and biased using gradient descent

Sparse categorical cross entropy →

All method same ↑

best kil diff →

$\hat{y} = [0.1 \quad 0.4 \quad 0.5]$ $y = 1$ ↑

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

gpa	iq	placed	OHE
7	70	Yes	1
8	80	No	2
6	60	Maybe	3

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

↳ Choose only first becau original y is 1

* If original y is 2 then we choose $-y_2 \log(\hat{y}_2)$

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

$\hat{y} = [0.1 \quad 0.2 \quad 0.6]$

$y = [2]$ → 1 → $L = -(1) \log(0.2)$

We take 1.

Reg → mse
↳ Outlier — mae
} Huber loss

Classifi → binary - bce
↳ multi — CCE
— SCE

** Why combination of binary cross entropy and sigmoid function work well?

1. The Sigmoid function's output fits perfectly into the BCE loss function, which expects a prob value for its calculation.
2. The gradients of the BCE loss function with respect to the model parameters are well-behaved when used with the sigmoid activation, facilitating effective training.

3. The output of the sigmoid function directly represent the probability of the positive class, making it intuitive to interpt the results.