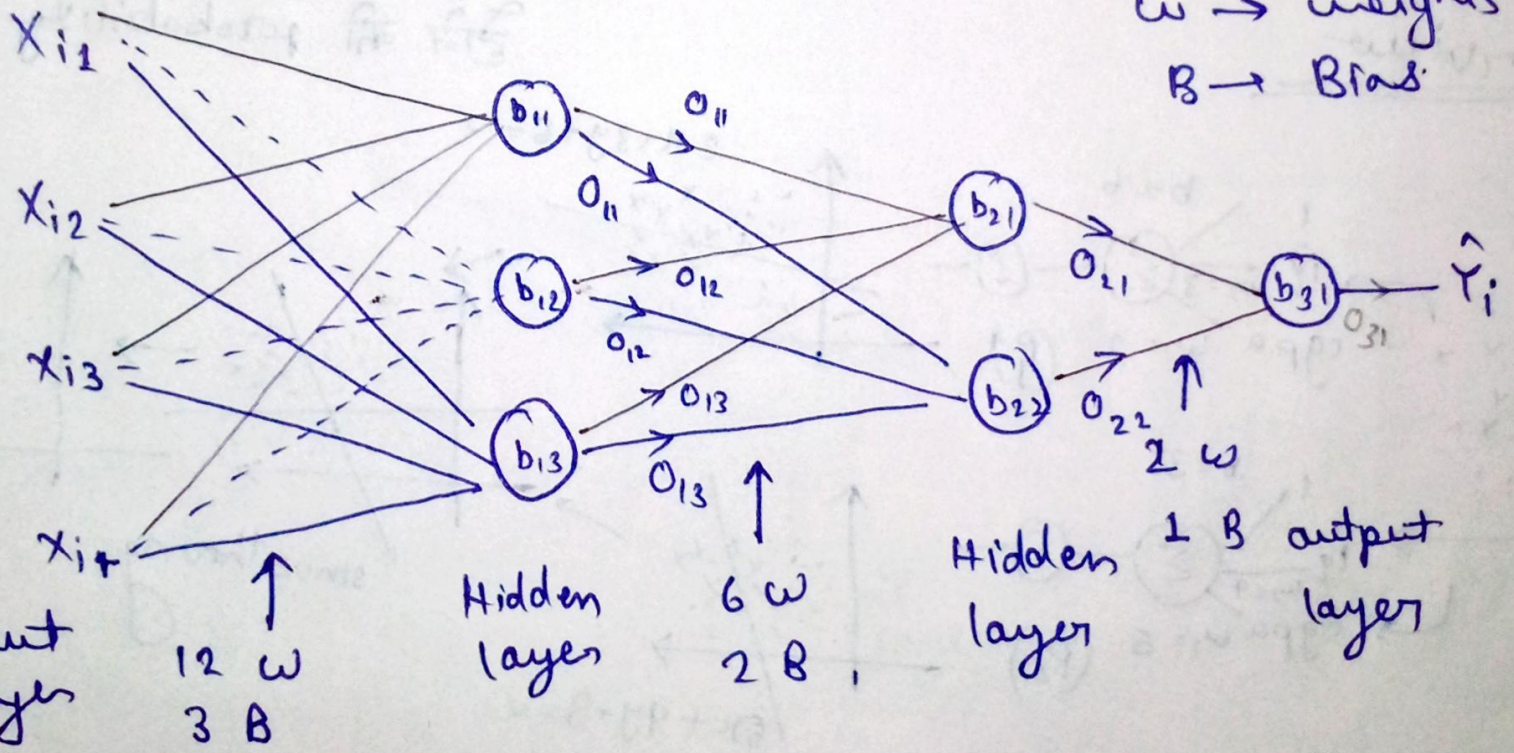
Multi layer Perceptron (MLP) Notation :

X_{i1} → i = rows 1 = column.

ω → weights

B → Bias



12 + 3 + 6 + 2 + 2 + 1 = 26 Trainable parameters

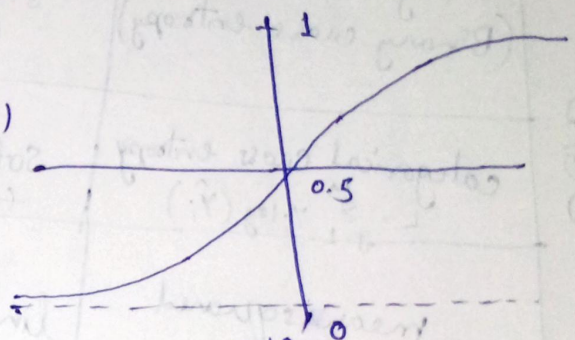
Bias	output	weight
b_{ij}	O_{ij}	w_{ij}^{k} (layer)
i = layer	i = layer	i = पिछले layer के किस node से आ रहा है।
j = node	j = node	j = आने वाले layer के किस node में जा रहा है।
		k = किस layer में जा रहा है।

$w_{PN\,CN}$
PN - previous node
CN - current node

Multilayer Perceptron (MLP):

sigmoid function

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



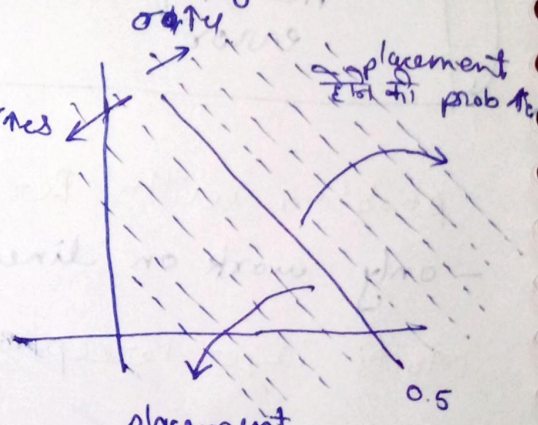
loss function — log loss

$z = \sum w_i x_i$

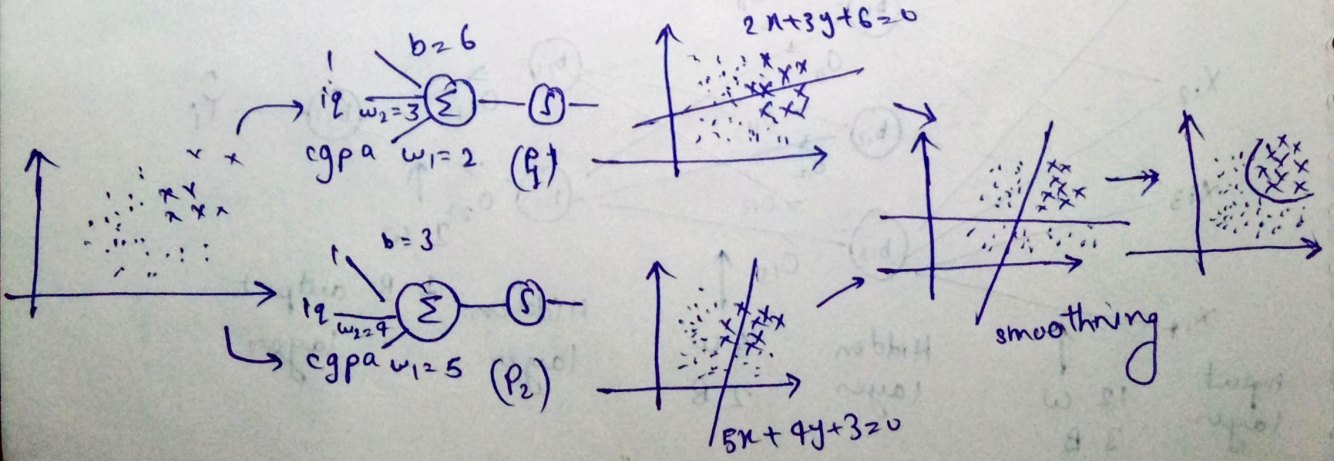
if z +ve then $\sigma(z) \geq 0.5$

if z -ve then $\sigma(z) < 0.5$

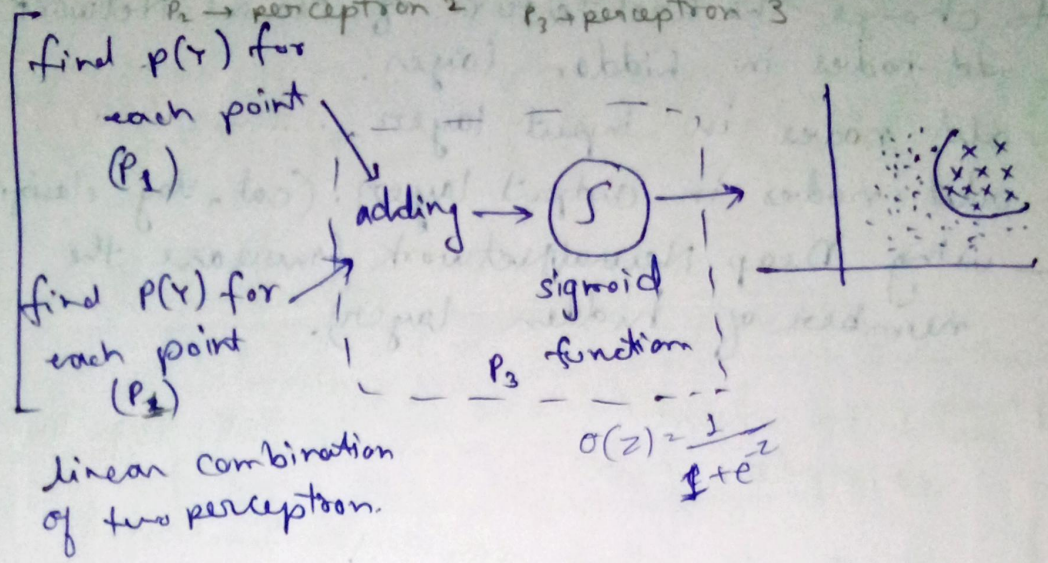
if $\sigma = 0.5$ means placement होने की probability is 0.5



overview



P_1 → perceptron 1
P_2 → perceptron 2
P_3 → perceptron 3
P_1, P_2, P_3 → perceptron.



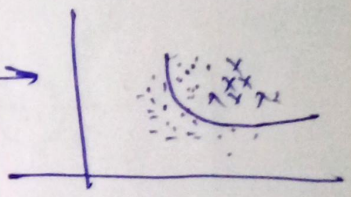
*Here we can dominate the one perceptron on other using the weighted ~~of the~~. and also can be add bias

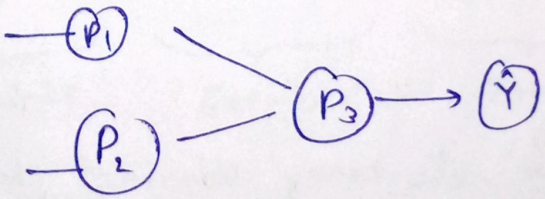
$b = 3$

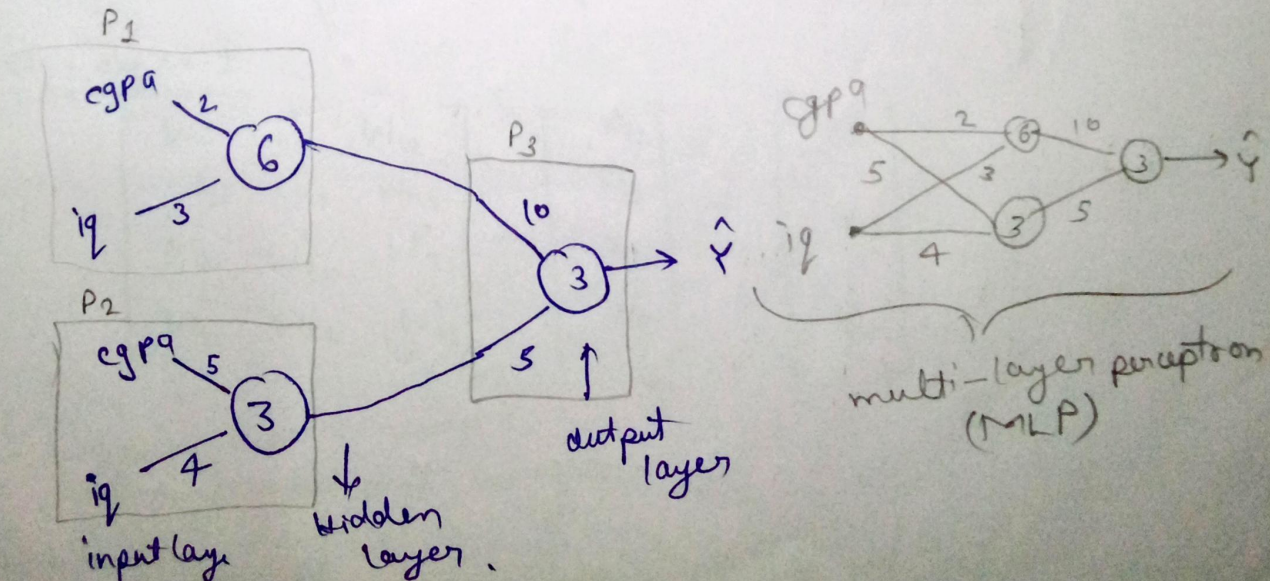
$P(Y_1) = 0.7$ $P(Y_2) = 0.6$

$z = 0.7 \times 10 + 0.8 + 5 + 3$ → S → sigmoid

↑ weight ↑ weight ↓ bias







How to change the architecture of Neural Network?

- add nodes in hidden layer.
- add nodes in input layer. (visualise in 3D)
- add nodes in output layer. (cat, dog classification)
- using Deep Neural Network (increase the number of hidden layer).

→ used for multiclass classification
example predict in image cat, dog, human

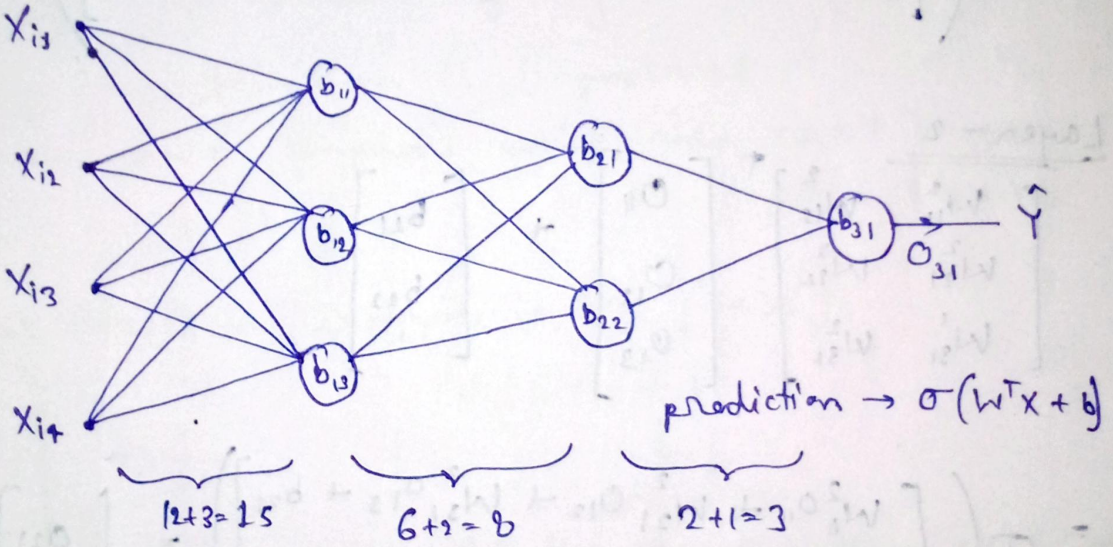
* Neural Network also called universal functional approximator.

FORWARD PROPAGATION

Forward Propagation:

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

→ # for trainable parameters



Total trainable parameters = 15+8+3 = 26

Layer-1

$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{43} \end{bmatrix}^T_{3\times 1} \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix}_{4\times 1} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}_{3\times 1}$$

rows → input weights
columns → output weights

$$\begin{bmatrix} w'_{11}x_{i1} + w'_{21}x_{i2} + w'_{31}x_{i3} + w'_{41}x_{i4} \\ w'_{12}x_{i1} + w'_{22}x_{i2} + w'_{32}x_{i3} + w'_{42}x_{i4} \\ w'_{13}x_{i1} + w'_{23}x_{i2} + w'_{33}x_{i3} + w'_{43}x_{i4} \end{bmatrix}_{3\times 1} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}_{3\times 1}$$

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

Layer - 2

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w'_{31} & w^2_{32} \end{bmatrix}^T \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

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

layer - 3

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

$$\sigma\left(\begin{bmatrix} w^3_{11}o_{21} + w^3_{21}o_{22} + b_{31} \end{bmatrix}\right) = \hat{Y}_1 = o_{31}$$

$$\sigma(a^{[0]}W^{[1]} + b^{[1]}) = a^{[1]} \quad \Big| \quad a^{[2]} = \sigma(a^{[1]}W^{[2]} + b^{[2]}) \quad \Big| \quad a^{[3]} = \sigma(a^{[2]}W^{[3]} + b^{[3]})$$

LOSS FUNCTIONS

what is loss function:
loss function is a method of evaluating how well your algorithm is modelling your dataset.

high → poor
small → great
} algo performation.

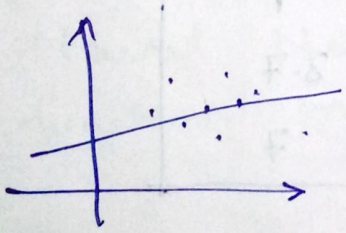
why loss function is important?
"you can't improve what you can't measure"
peter Drucker

*loss function example in ML (MSE)

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

$$\downarrow$$

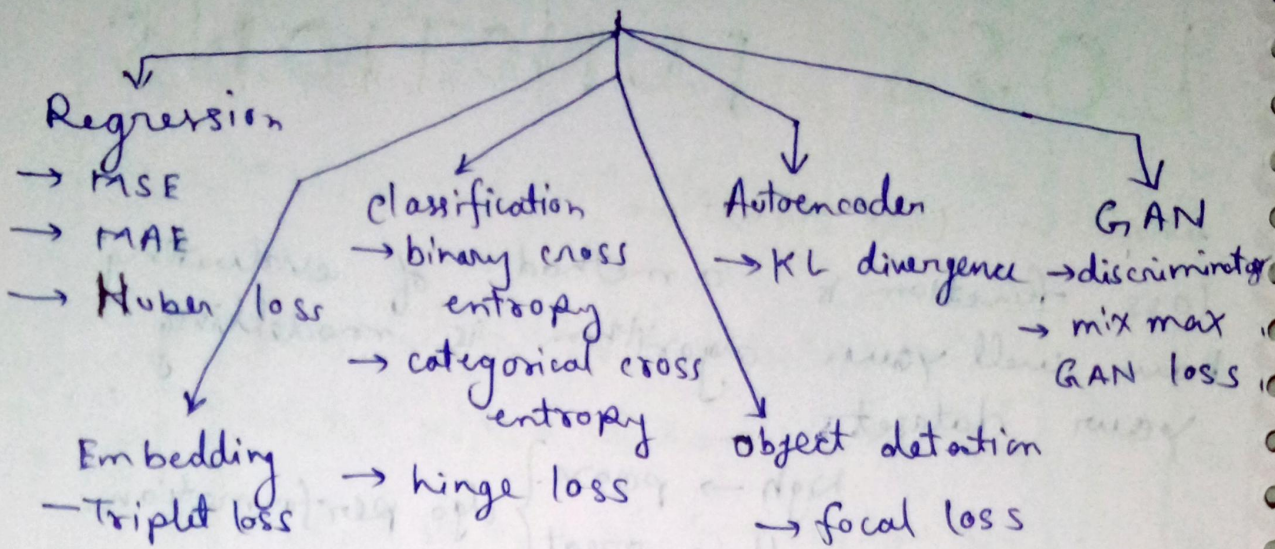
$$mx_i + b$$



calculate the combination of m and b such that $L(m, b)$ = minimum.
for minimizing we use the gradient descent.

"loss function eye of ML algorithm"

Loss functions in DL



Loss function vs Cost function:

→ loss function single training example पर calculate होता है L_i

cgpa	iq	package	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

here $Y_i = 6.3$ $\hat{Y}_i = 6.1$

$$L = (Y_i - \hat{Y}_i)^2 = (6.3 - 6.1)^2 = 0.04$$

→ cost function पूरे training data पर calculate होता है

$$\text{cost fn}(C) = \frac{1}{n} \sum (Y_i - \hat{Y}_i)^2$$

$$C = \frac{1}{4} \left[(6.1 - 6.3)^2 + (4.1 - 4)^2 + (3.5 - 3.7)^2 + (7.2 - 7)^2 \right] =$$

(1.) Mean Squared Error (MSE, squared loss, L2 loss)

$$MSE = (y_i - \hat{y}_i)^2 \qquad \text{cost fn} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\therefore MSE = (\text{true value} - \text{predicted value})^2$$

for each row to minimize MSE:

- Step-1: $(6.3 - 6.1)^2 = 0.04$
- Step-2: update the weight and bias

advantages:

- → easy to interpret
- → differentable (used in GD)
- → one local minima only.

disadvantages

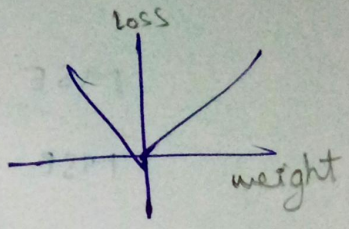
- → unit of error is squared
- → it is not robust to outliers.

* if we use this loss function in DL's Neural Network architecture then activation function must be linear. (last one node)

(2.) Mean Absolute Error (MAE or L1 loss)

Loss function (L) = $|Y_i - \hat{Y}_i|$

cost function (C) = $\frac{1}{n} \Sigma |Y_i - \hat{Y}_i|$



advantages:

→ intutive and easy to understand

→ unit is same as Y

→ Robust to outlier

disadvantages

→ not differentiable function

~~if data have outlier then use MAE~~

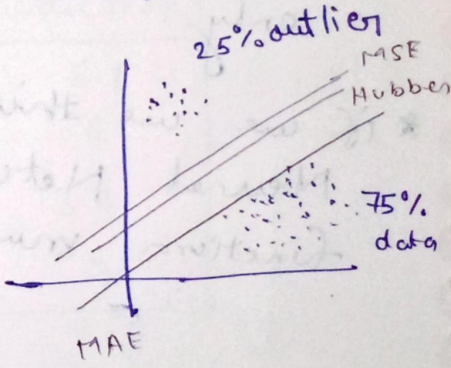
(3.) Hubber Loss:

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

if no outlier then Huber = MSE

if outlier then Huber = MAE

Here outlier depends on the (δ) hyperparameter.



(4.) Binary Cross Entropy (log loss):

- used in classification example in logistic regression
- it is used when we have only two class. prediction (0 or 1, true or False)

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

Y = actual

$\hat{Y}$ = prediction.

- if we use Binary Cross entropy then in the output layer activation function is only sigmoid, in hidden layer may be change.

$$\text{cost function} = -\frac{1}{n}\left[\sum_{i=1}^{n} Y_i \log \hat{Y}_i + (1-Y_i)\log(1-\hat{Y}_i)\right]$$

advantages:
→ differentiable
→

disadvantage:
→ multiple local minima
→ not a intetive

(5.) Categorical Cross Entropy (~~used in~~ soft) :
- it is used in softmax Regression
- it is used in classification (multi-class classification)

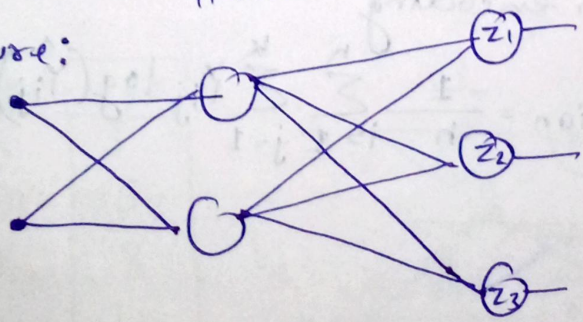
cgpa	iq	placed	Yes	No	Maybe
8	80	Yes	1	0	0
6	60	No	0	1	0
7	70	maybe	0	0	1

OHE (one hot encode)

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

Here k = number of class.

Architecture:



↑
output neurons categories (class)

- in output layer activation function is always softmax.

* Here the no. of neuron = number of class

for Yes: $f(z)_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$

for No $f(z)_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$

for may be $f(z)_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$

and the loss function for three class.

$$L = -Y_1 \log(\hat{Y}_1) - Y_2 \log(\hat{Y}_2) - Y_3 \log(\hat{Y}_3)$$

range of $f(z)_1$, $f(z)_2$ and $f(z)_3$ is 0 to 1

and $f(z)_1 + f(z)_2 + f(z)_3 = 1$

(6.) sparse categorical cross entropy

One hot encoding

number encoding

$$\text{cost function} = \frac{-1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} Y_{ij} \log(\hat{Y}_{ij})$$

~~- Here the number of~~