MACHINE LEARNING

AYUSH KUMAR, SA

Data mining:- Applying ML techniques to dig deep into large amounts of data to help discover patterns that were not immediately apparent.

Why Machine learning is important?

Machine learning is great for

* Problems for which existing soln require a lot of hand tuning or long lists of rules
  ex → Spam Mail Detection

* Complex problems for which no good soln is present using traditional approach
  ex → Speech Recognization

* Fluctuating ~~techniques~~ environments
  ex → Change in approach of spammers who send spam mail

* Getting insights about complex problems and large amounts of data, eg:- clustering

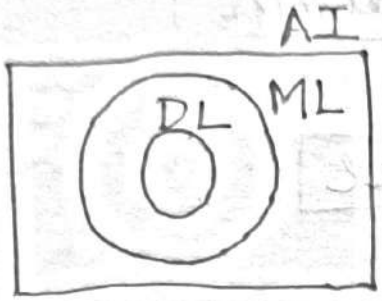
Labels → soln given with the feathers. Only soln value are known as labels.

Attribute Vs feature (used interchangeably)

↓ eg: mileage ↓ eg: mileage is 5,000

Unsupervised learning also help in learning association rule learning (to ~~make~~ identify relation between attributes.

Deep Learning and Machine Learning



Machine Learning

- Supervised
  - Regression
  - Classification
- Unsupervised
  - Clustering
    - Grouping
      - ex → Customer segmentation
  - Dimensional Reduction
    - eg → 1000 dimension to 100 dimension
- Reinforcement

Features → Independent
→ Dependent

Supervised learning Algorithms

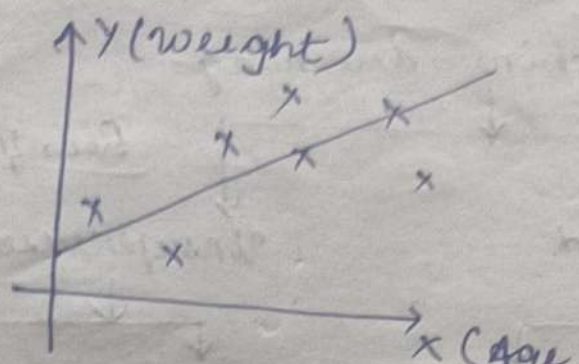
1) Linear Regression
2) Ridge and Lasso
3) Logistic Regression
4) Decision Tree
5) Adaboost
6) Random Forest
7) Gradient Boosting
8) Xgboost
9) Naive Bayes
10) SVM
11) KNN

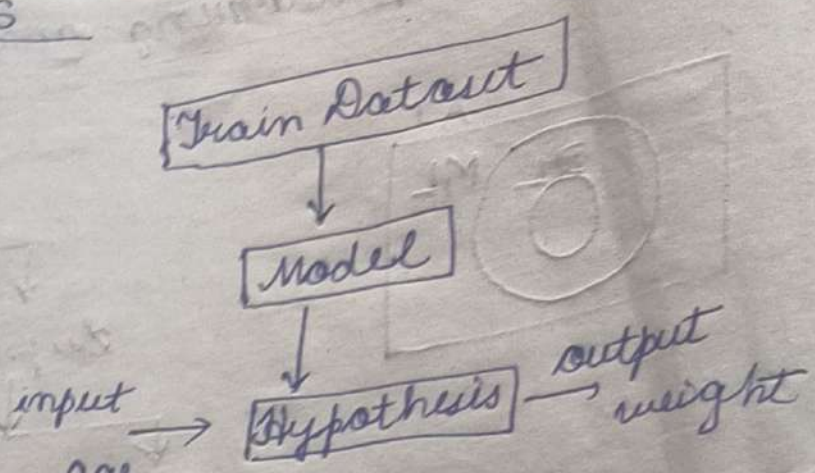
Unsupervised learning algorithms

1. K Means
2. DB Scan
3. Hierarchical
4. K Means Nearest Neighbour Clustering
5. PCA
6. LDA

ALGORITHMS

① Linear Regression

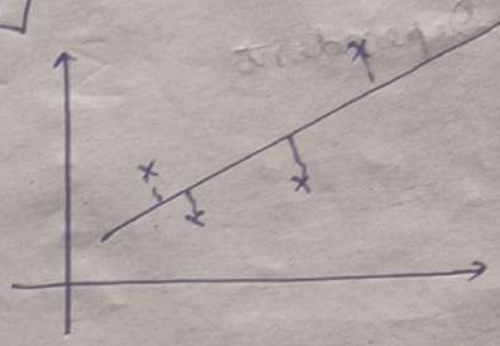




Eqⁿ of straight line

$h_\theta(x) = \theta_0 + \theta_1 x$ [Hypothesis]

θ_0 ↓ intercept, θ_1 ↓ slope



Cost function

$$= \sum_{i=1}^{m} \left(h_\theta(x)^{(i)} - y^{(i)}\right)^2$$

$$J(\theta_0,\theta_1) \frac{}{2m} \rightarrow$$ (2 is divided so that it's derivative will be simpler.

$$\boxed{J(\theta_0,\theta_1) = \frac{\sum \left[h_\theta(x)^{(i)} - y^{(i)}\right]^2}{2m}}$$

↓ Squared Error Function

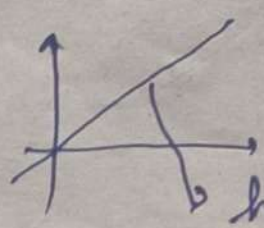
What we need to solve

$$\underset{(\theta_0,\theta_1)}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta x^{(i)} - y^{(i)}\right]^2$$

or

$$\underset{(\theta_0,\theta_1)}{\text{minimize}} \; J_{(\theta_0,\theta_1)}$$

eg → $h_{(\theta(x))} = \theta_0 + \theta_1 (x)$ if $\theta_0 = 0$



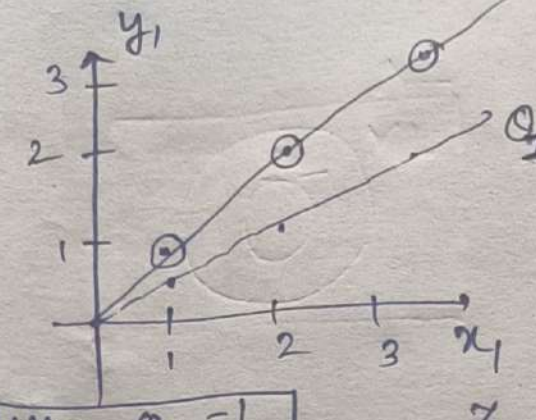
If $\theta_0 = 0$

$h_\theta(x) = \theta_1 x$

$h_\theta(x) = \theta_1 x$

When $\theta_1 = 1$

Datapoints
(1, 1)
(2, 2)
(3, 3)

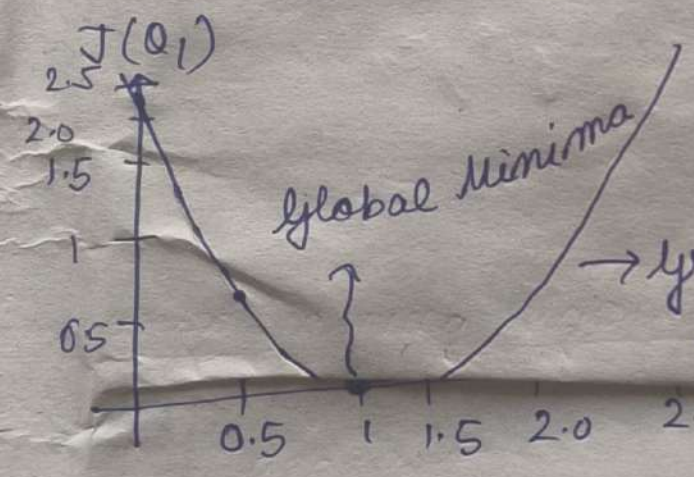


When $\theta_1 = 1$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{3} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \left[(1-1)^2 + (2-2)^2 + (3-3)^2 \right]$$

$$= 0$$



When $\theta_1 = 0.5$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{3} \left[(0.5-1)^2 + (1-2)^2 + (3-1.5)^2 \right]$$

($\theta_1 \downarrow 0.5$)

$$= \frac{1}{2 \times 3} \left[0.25 + 1 + 2.25 \right]$$

$$\approx 0.58$$

When $\theta_1 = 0$

$$J(\theta_1) = \frac{1}{2m} \left[1^2 + 9^2 + 4^2 \right] = \frac{1}{6} \left[14 \right] = 2.3$$

Convergence Algorithm

Repeat until convergence

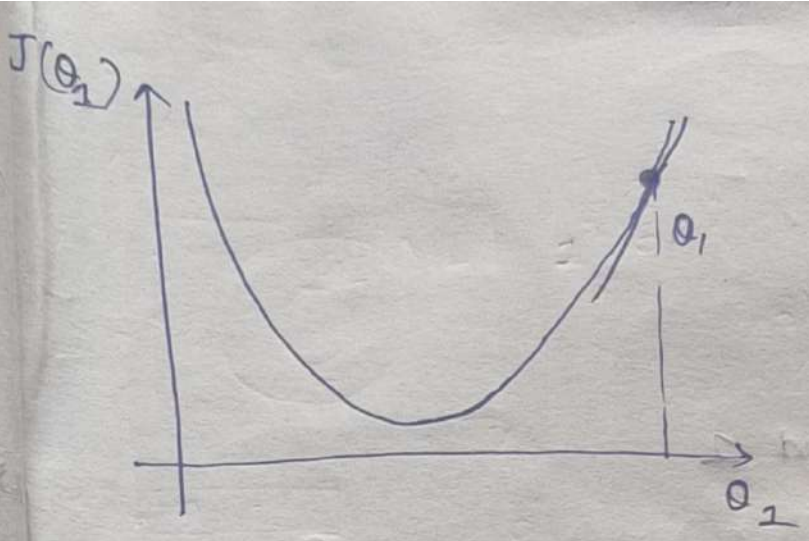
{

$$\theta_j := \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \right]$$

(α: learning rate; bracketed term: Derivative (slope))

}

$\alpha = 0.01$

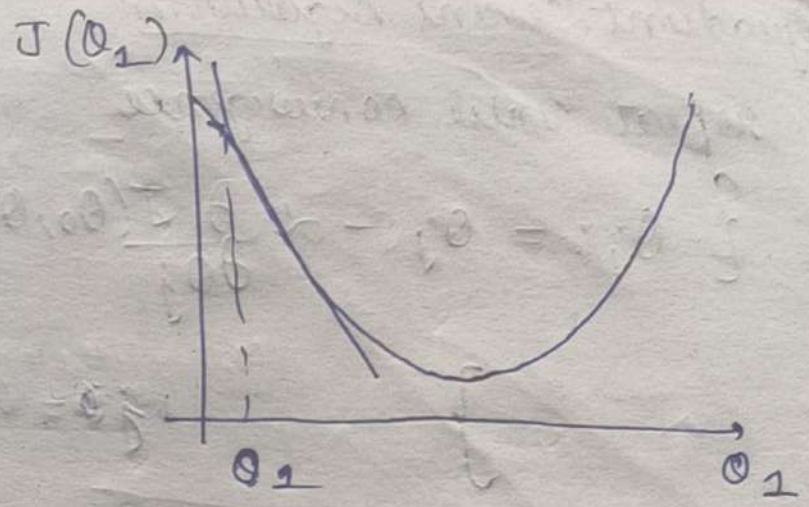


$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta, \theta_i)}{\partial \theta_j}$$

In this case $\frac{\partial J(\theta, \theta_i)}{\partial \theta_j}$ = +ve

$\therefore \theta_j = \theta_j - \alpha(+ve)$

∴ so, we should reach global minima



$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta, \theta_i)}{\partial \theta_j}$$

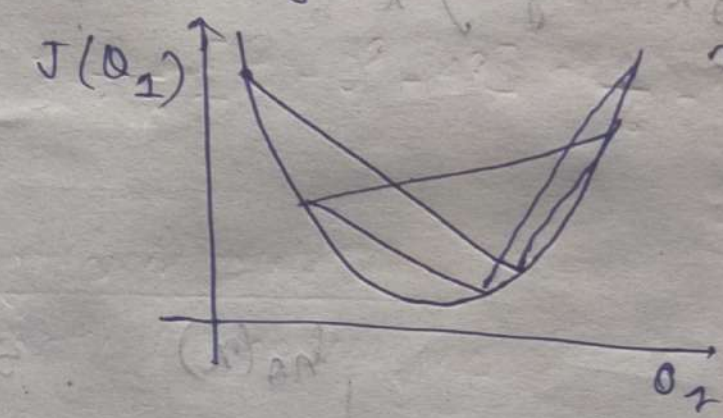
In this case $\frac{\partial J(\theta, \theta_i)}{\partial \theta_j}$ = -ve

$\therefore \theta_j = \theta_j - \alpha(-ve)$

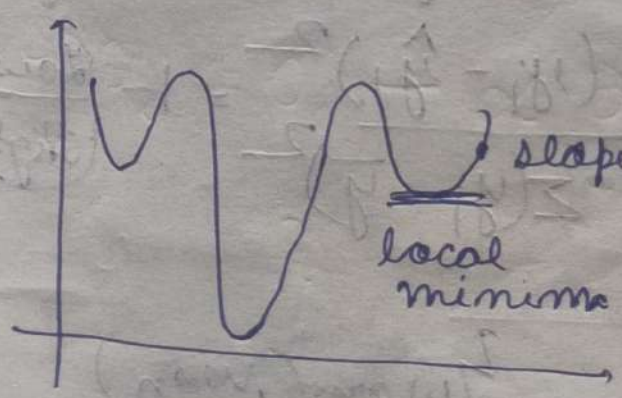
∴ so, we should reach global minima.

α (learning rate) should be less; so updation of θ_j should be in steps, (like @ $\alpha \approx 0.01$)

If $\alpha = 100$ eg →



We will not get any local minima



In deep learning techniques we may get such gradient Descent graph which can solved by various optimiser, but in linear regression we will not get such graph

Gradient Descent Algorithm

repeat until convergence

$$\{ \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\} \qquad j = 0 \text{ and } 1$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})^2$$

$h_\theta(x) = \theta_0 + \theta_1 x$

$$j = 0 \Rightarrow \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta x^{i} - y^{(i)})$$

$$j = 1 \Rightarrow \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)}) x^{(i)}$$

repeat until convergence $\alpha = 0.01$ (Learning rate)

$$\{ \quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)}) x^{(i)}$$

$$\}$$

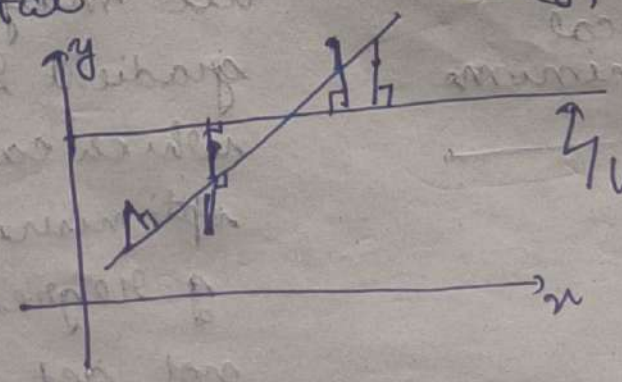
Performance metrics

R^2 and Adjusted R^2

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\text{Low}}{\text{High}}$$

($h_\theta(x)$ ↓ ; Small number ↓)

* R^2 can also be negative



↳ Worse (Mean)

* R^2 also increases when attributes nowhere related to the output are added. So we use adjusted R^2.

Adjusted R^2.

$$\left(R^2_{\text{adjusted}} = 1 - \frac{(1-R^2)(N-1)}{N-P-1}\right)$$

Adjusted $R^2 \leq R^2$

* N = no. of datapoint
* P = no. of predictors

Ridge and Lasso Regression

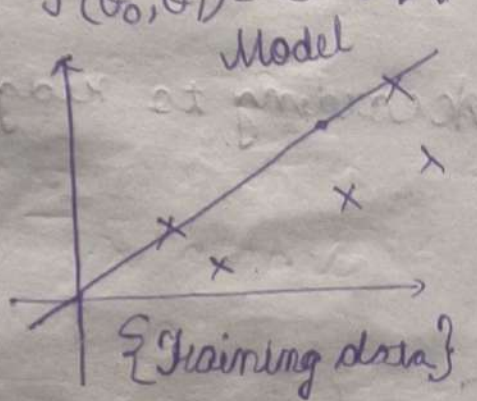
Overfitting (Low Bias)

① Model performs well → Training data

② Fail to perform well → Test Data (High variance)

$\theta_0 = 0$

$J(\theta_0, \theta_1) = 0$ ↓↓↓



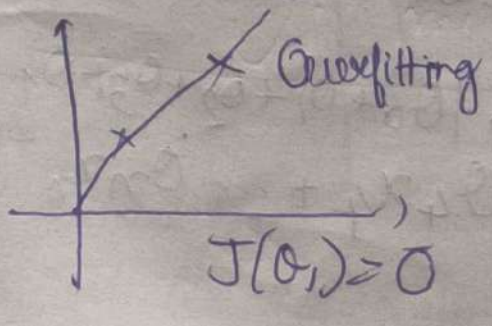
Underfitting {High bias, high variance}

① Model accuracy is bad with training data.

② Model accuracy is also bad with test data.

ex →

Model 1	Model 2	Model 3
Training Acc = 90%	Training acc = 92%	Training acc = 70%
Test Acc = 80%	Test acc = 91%	Test Acc = 65%
Overfitting	Generalised	Underfitting
Low bias, high variance	Low bias, Low variance	High bias, high variance



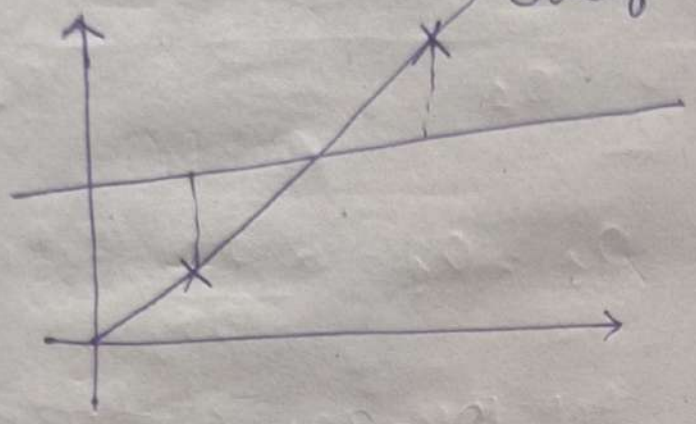
$$J(\theta_1) = 0$$

$$= \frac{1}{2m} \sum (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$= (\hat{y}^i - y^{(i)})^2 = 0$$

Ridge (L2 Regularization)

$$J(\theta_1) = (\hat{y}^i - y^{(i)})^2 + \lambda(\text{slope})^2$$



$$J(\theta_1) = 0$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})^2$$

$$= (\hat{y}^{(i)} - y^{(i)})^2 + \lambda(\text{slope})^2$$

According to ridge we add $[\lambda(\text{slope})^2]$

$$= 0 + 1(2)^2 \leftarrow = 4$$

$$= (\hat{y}^{(i)} - y^{(i)})^2 + \lambda(\text{slope})^2$$

{Prevents Overfitting} = small value + $1(1.5)^2$

= small value + 2.25

≈ 3

iterations {hyperparameter}

R^2, adjusted R^2

$\lambda \rightarrow$ hyperparameter

convergence algorithm

Lasso (L1 Regularization)

$$J(\theta_1) = (\hat{y}^{(i)} - y^{(i)})^2 + \lambda |\text{slope}| \longrightarrow |\theta_0 + \theta_1 + \theta_2 + \theta_3 + \theta_4 \cdots + \theta_n$$

→ feature selection

$$h_\theta(x) = \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \cdots \theta_n x_n$$

Using Lasso

1) Preventing Overfitting } → L_1 regularization
2) Feature selection } L_2 regularization

{3 → cross validation}

Summary:-

Ridge regression (L2 norm)

$$\text{cost function} = \frac{\sum\left((h_\theta^{(i)}(x) - y^{(i)})^2 + \lambda(\text{slope})^2\right)}{2m}$$

Purpose: Preventing overfitting

Lasso regression (L1 norm)

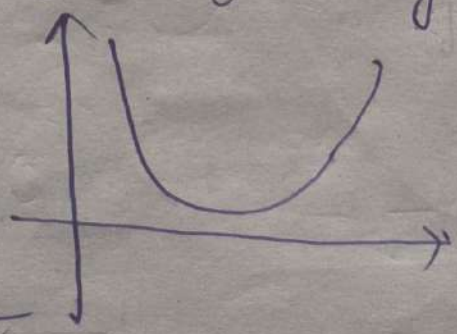
$$\text{cost function} = \frac{1}{2m}\left(\sum\left((h_\theta^{(i)}(x) - y^{(i)})^2 + \lambda|\text{slope}|\right)\right)$$

Purpose: 1) Prevent overfitting
2) Feature selection

Assumption of Linear Regression

1) Normal (Gaussian Distribution → model will get trained well)

2) Standardization {Scaling data} → Z score $\mu = 0, \sigma = 1$



3) Linearity

4) Multi collinearity

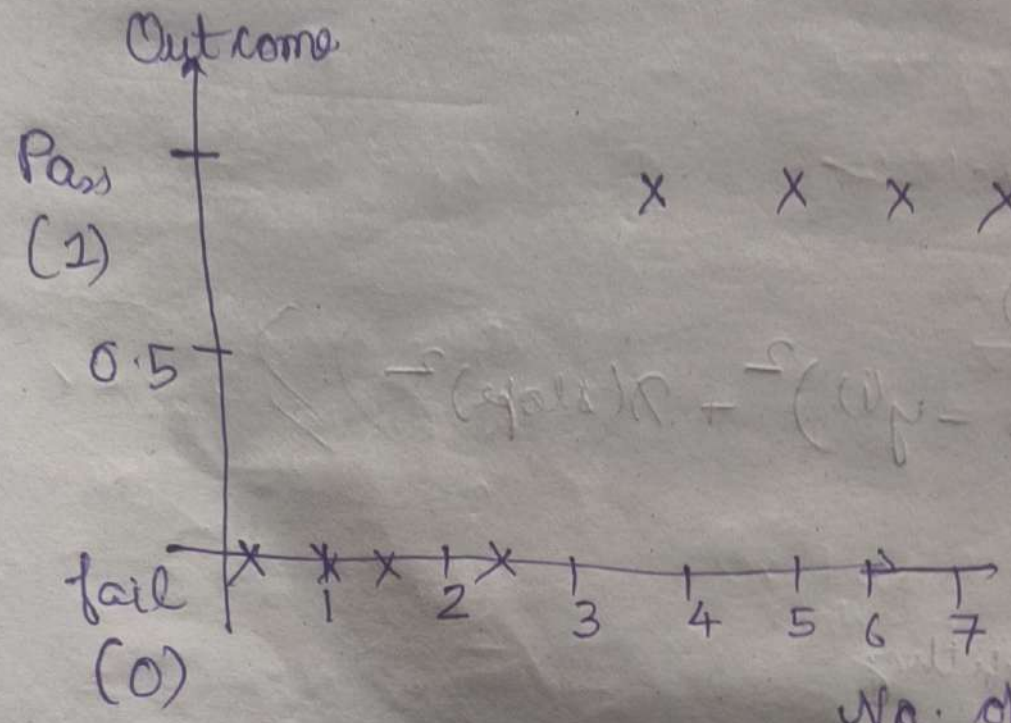
X_3 [X_1 X_2] 95% → variation inflation factor

* Homoscedacitity

Logistics Regression (Classification algorithm) → Binary classification (also we can solve multiclass classification)

eg →

No. of study	No. of play	P/F
—	—	P
—	—	F



Linear Regression??

$h_\theta(x) < 0.5 \Rightarrow 0$

↓

Fail

$h_\theta(x) \geq 0.5 \Rightarrow 1$

↓

Pass

{Sigmoid function}

0 to 1

* Why we dont use linear regression?

→ because, in linear regression, values less then 0 and more than 1 also occur due to which it does not become a sigmoid f^n (values should be between 0 to 1)

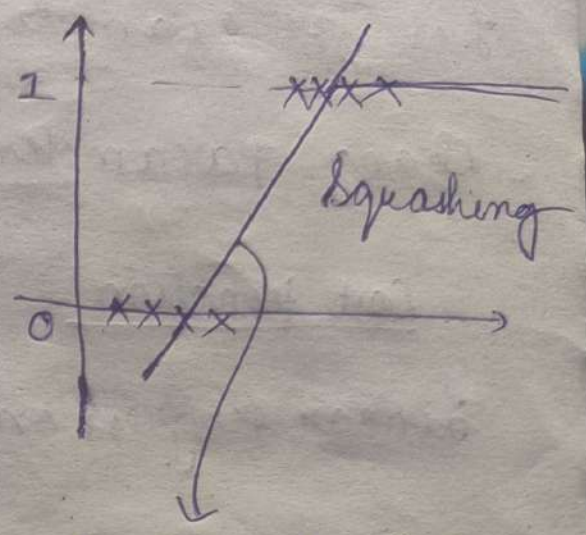
* ~~Why don't~~

→ Because of outliers, the predicted values are far off from actual values.

Decision boundary logistic regression

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$$

$$\boxed{h_\theta(x) = \theta^T x}$$



Let $z = \theta_0 + \theta_1 x$

$h_\theta(x) = g(z)$

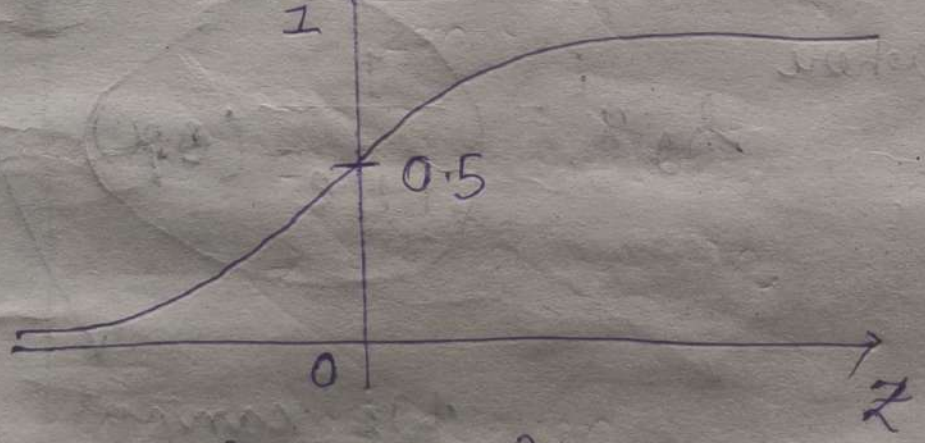
$$h_\theta(x) = \frac{1}{1+e^{-z}}$$ → Sigmoid or Logistic function

$h_\theta(x) = \theta_0 + \theta_1 x_1$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1)$

$$\boxed{h_\theta(x) = \frac{1}{1+e^{-(\theta_0 + \theta_1 x)}}}$$

$g(z)$



$g(z) \geq 0.5$ when $z \geq 0$

Training Set

$\{(x^1, y^1), (x^2, y^2), (x^3, y^3), \ldots\ldots (x^n, y^n)\}$

$y \in \{0, 1\}$ → 2 o/p

$h_\theta(z) = \frac{1}{1+e^{-z}}$ $\quad z = \theta_0 + \theta_1 x$

Let for example $\theta_0 = 0$, $z = \theta_1 x$

Change parameter θ_1?

Cost function

Linear regression $J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^i - y^i)^2$

Logistic regression, $h_\theta x = \frac{1}{1+e^{-(\theta_1 x)}}$

Cost function,

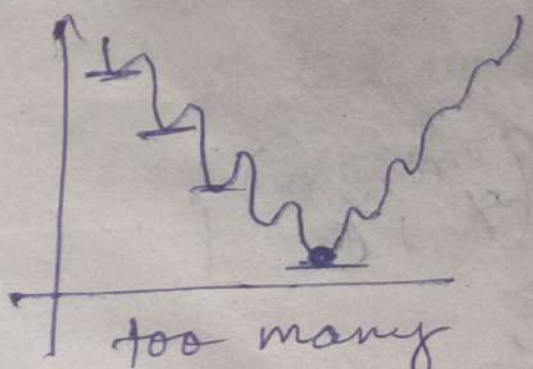
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^i - y^i)^2$$

where $h_\theta x^i = \frac{1}{1+e^{-(\theta x)}}$

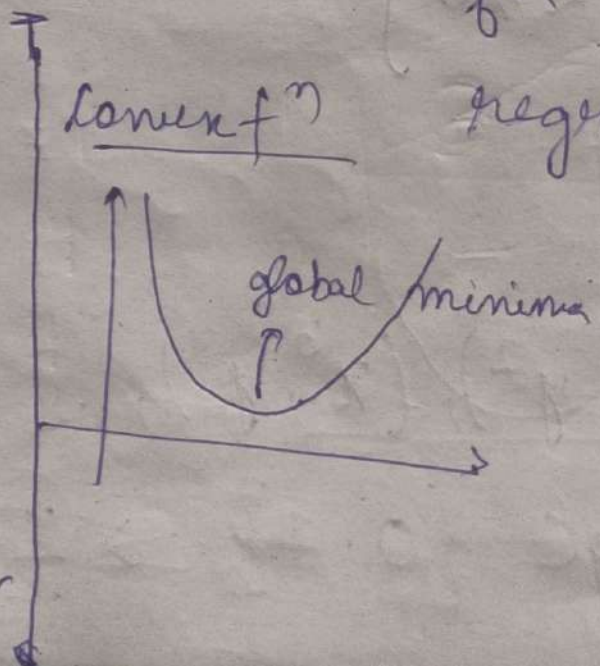
non-convex f^n.

We cannot use this cost f^n for logistic regression.

Non convex f^n

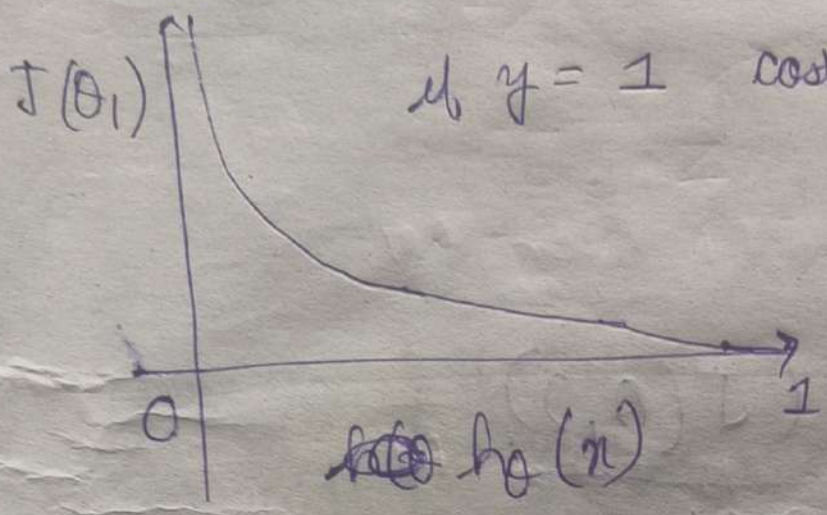


Convex f^n



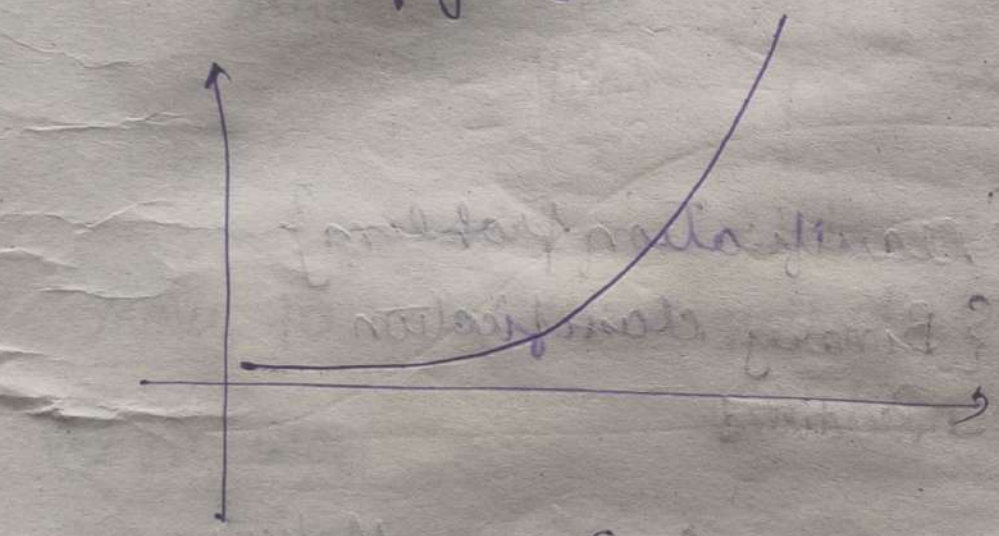
Logistic Regression cost function → $h_\theta(x) = \frac{1}{1+e^{-(\theta_1 x)}}$

$$J(\theta_1) = \begin{cases} -\log(h_\theta(x^{(i)})) & y=1 \\ -\log(1-h_\theta(x^{(i)})) & y=0 \end{cases}$$



if $y=1$ cost $=0$ if $y=1$, $h_\theta x = 1$

if $y=0$



$$\text{cost}(h_\theta(x^i), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

$$\boxed{\text{cost}(h_\theta x^i, y) = -y \log h_\theta(x^i) - (1-y)\log(1-h_\theta(x))}$$

if $y=1$

$$\text{cost}(h_\theta x^i, y) = -\log(h_\theta x^{(i)})$$

If $y = 0$

$$\text{cost}(h_\theta x^i, y) = -\log(1 - h_\theta(x^i))$$

Cost function

$$J(\theta_1) = \frac{-1}{2m} \sum_{i=1}^{m} y^i \log(h_\theta x^i) + (1 - y^i) \log(1 - h_\theta(x^i))$$

$$h_\theta(x^i) = \frac{1}{1 + e^{-\theta_1 x}}$$

Repeat until convergence

$$\left\{ \theta_j = \theta_j - \alpha \frac{\partial (J(\theta_1))}{\partial \theta_j} \right\}$$

Performance metrics {Classification problem}

{Binary classification

y → actual

$\hat{y}$ → Predicted

x_1	x_2	y	$\hat{y}$
—	—	0	1
		1	1
—	—	0	0
—	—	1	1
—	—	1	1
—	—	0	1
—	—	1	0

Confusion Matrix

Predicted \ Actual	1	0
1	3	2
0	1	1

Predicted \ Actual	1	0
1	TP	FP
0	FN	TN

← Confusion Matrix

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$= \frac{3+1}{3+2+1+1} = 4/7 = 0.57 = 57\%$$

0 → 900
1 → 100
} Unbalanced Dataset

0 → 600
1 → 400
} Balanced Data

$$\text{Accuracy} = \frac{900}{1000} = 90\%$$

↓

Not biased data

More parameters required for unbalanced data

① Precision ② Recall ③ F-Score

$$\frac{TP}{TP + FP} \qquad \frac{TP}{TP + FN}$$

Predict \ Actual	1	0
1	TP	FP
0	FN	TN

{Spam classification}
↳ Precision

{has CANCER OR NOT}
→ Recall

{Tomarrow stock market will fall} → f → Beta

$$\underline{\underline{F-\text{Beta}}} = (1+\beta^2)\frac{\text{Precis} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

$\beta = 1$ — $F_1 = 2\left(\frac{\text{Pre} \times \text{Recall}}{\text{Pre} + \text{Recall}}\right)$ [HM]

↙ both fP, fN imp

β decrease → more imp to fP than fN
↘ 0.5 $f_{0.5}$

β increase → more imp to fN than fP
↘ 2 f_2

NAIVE BAYES INTUITION

{Classification}
⇓
{Baye's Theorem}

Independent event → rolling a dice
Dependent event → taking out marble without replacement

For dependent event:

$$P(A \text{ and } B) = P(A) * P(B/A)$$

$$P(B \text{ and } A) = P(A \text{ and } B)$$

$$P(A) * P(B/A) = P(B) * P(A/B)$$

$$\boxed{P(B/A) = \frac{P(B) * P(A/B)}{P(A)}}$$ → Baye's Theorem

Crux behind Naive Bayes

ex on Dataset:-

X_1	X_2	X_3	X_4	.....	X_n	Y

→ Independent feature

Y → output

$$P(y/x_1 x_2 x_3 \dots x_n) = \frac{P(y) * P(x_1 x_2 \dots x_n / y)}{P(x_1 x_2 \dots x_n)}$$

$$= \frac{P(y) * P(x_1/y_1) * P(x_2/y_2) * \dots P(x_n/y_n)}{P(x_1) * P(x_2) * P(x_3) \dots P(x_n)}$$

ex →

x_1	x_2	x_3	x_4	y
				Yes
				No

$P(y = yes/x_i) = P(yes) * P(x_1/yes) * P(x_2/yes) * P(x_3/yes) * P(x_4/yes)$

#fixed $P(x_1) * P(x_2) * P(x_3) * P(x_4)$

Similarly for $P(y = No/x_i)$

$$P(y=No/x_i) = \frac{P(no) * P(x_1/no) * P(x_2/no) * P(x_3/no) * P(x_4/no)}{P(x_1) * P(x_2) * P(x_3) * P(x_4)}$$

↙ #fixed

Let $P(Yes/x_i) = 0.13$ $\quad P(No/x_1) = 0.05$

$\geq 0.5 \Rightarrow 1$

$< 0.5 \Rightarrow 0$

$$P(yes/x_1) = \frac{0.13}{0.13+0.05} = 0.72 = 72\%$$

$$P(No/x_1) = 1 - 0.72 = 0.28 = 28\%$$

ex→ DATASET — (1)

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	yes
D4	Rain	Mild	High	Weak	yes
D5	Rain	Cool	Normal	Weak	yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	yes
D10	Rain	Mild	Normal	Weak	yes
D11	Sunny	Mild	Normal	Strong	yes
D12	Overcast	Mild	High	Strong	yes
D13	Overcast	Hot	Normal	Weak	yes
D14	Rain	Mild	High	Strong	No

Outlook (x_1)

	ye	No	P(Y) P(Y/S)	P(N)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9 P(Y/O)	0/5
Rain	3	2	3/9 P(Y/R)	2/5
Total	9	5		

Temp (x_2)

	ye	No	P(y)	P(N)
~~Hot~~	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cold	3	1	3/9	1/5
	9	5		

Play ~~(x_3)~~

		P(yes)	P(No)
yes	9	9/14	5/14
No	5		
Total	14		

Then, for a test dataset, (Sunny, Hot) → Output?

$$\Rightarrow P(\text{yes}/\text{Sunny Hot}) = \frac{P(\text{yes}) * P(\text{Sunny}/\text{ye}) * P(\text{Hot}/\text{ye})}{P(\text{Sunny}) * P(\text{Hot})}$$

$$= 9/14 * 2/9 * 2/9$$

$$= 2/63 = 0.031$$

P(No/Sunny Hot)

$$= 5/14 * 3/5 * 2/5$$

$$= 0.085$$

$P(\text{Yes} / \text{Sunny}, \text{Hot}) = 0.031 \quad = 1 - 0.73 = 27\%$

$P(\text{No} / \text{Sunny}, \text{Hot}) = \dfrac{0.085}{0.031 + 0.085} = 0.73 = 73\%$

Normalized.

→ (Sunny, hot) → yes or NO

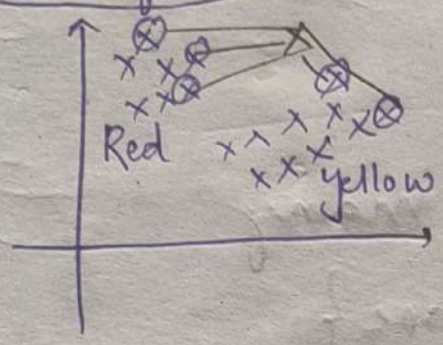
Assignment :-

(Overcast, Mild) → Naive Bayes?

KNN Algorithm {K - Nearest Member}

- Classification
- Regression

① Classification



K-Nearest Neighbour

K = 5

Maximum No

Red = 3

Yellow/~~white~~ = 2

Distances

Euclidean distance

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

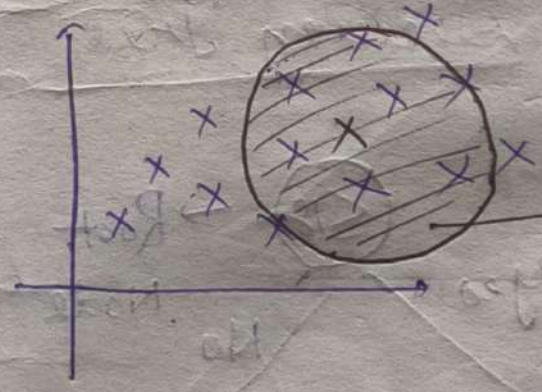
(x_2, y_2)

(x_1, y_1)

Manhattan Distance

$$|(x_2-x_1)|+|(y_2-y_1)|$$

② Regression

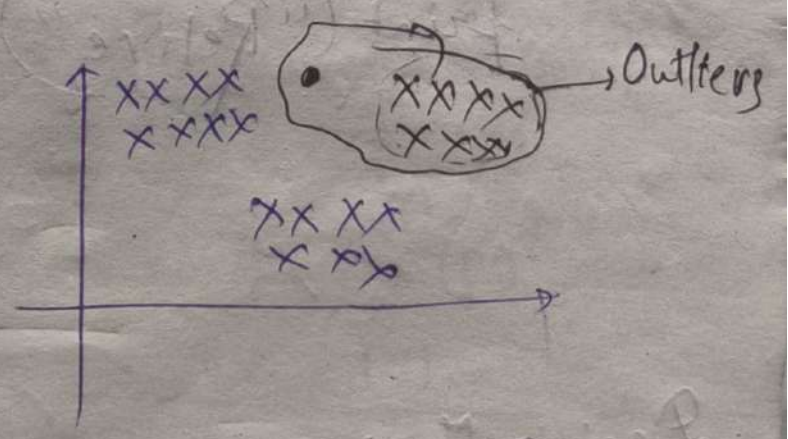


K = 5 ← Hyperparameter

K Nearest works bad with

① Outlier

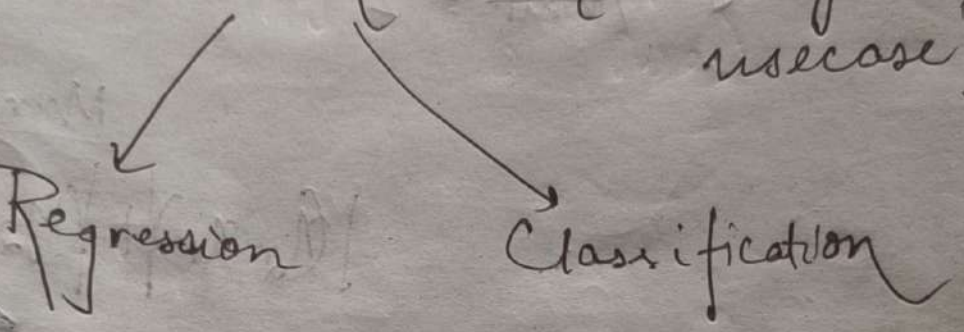
② Imbalanced DataSet



Agenda

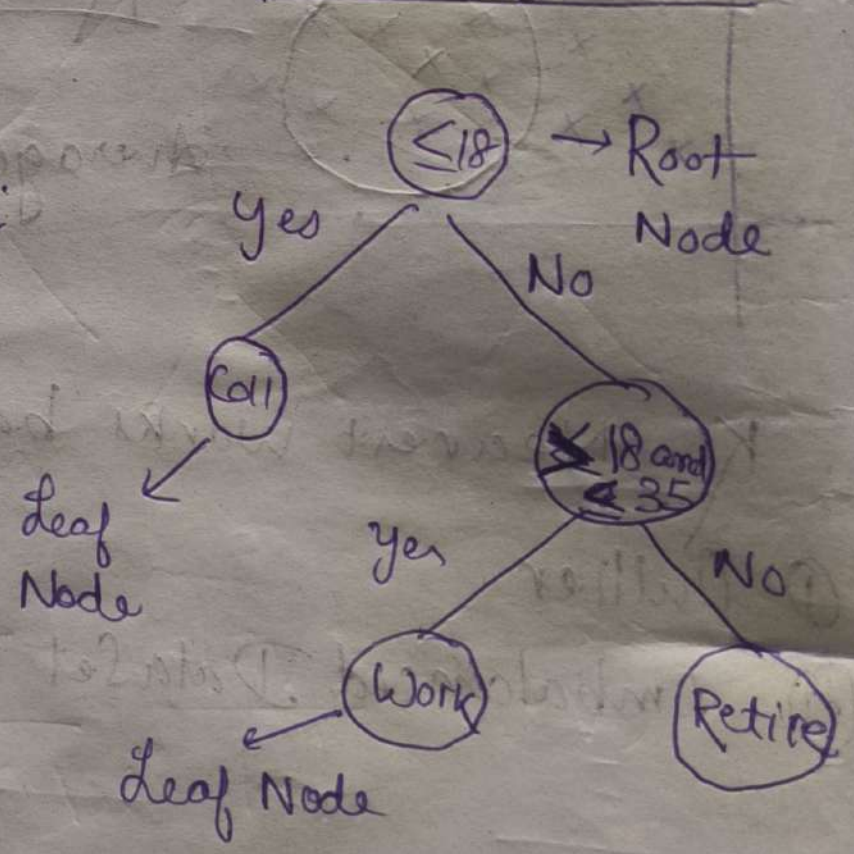
1. Decision Tree Classification
2. Decision Tree Regression
3. Practical Implementation
4. Consemble Techniques

Decision Tree { Solving many usecase }




```
if (age ≤ 18):
    print ("college")
dif (age > 18 and age ≤ 35):
    print ("Work)
else:
    print ("Retire")
```

Decision Tree

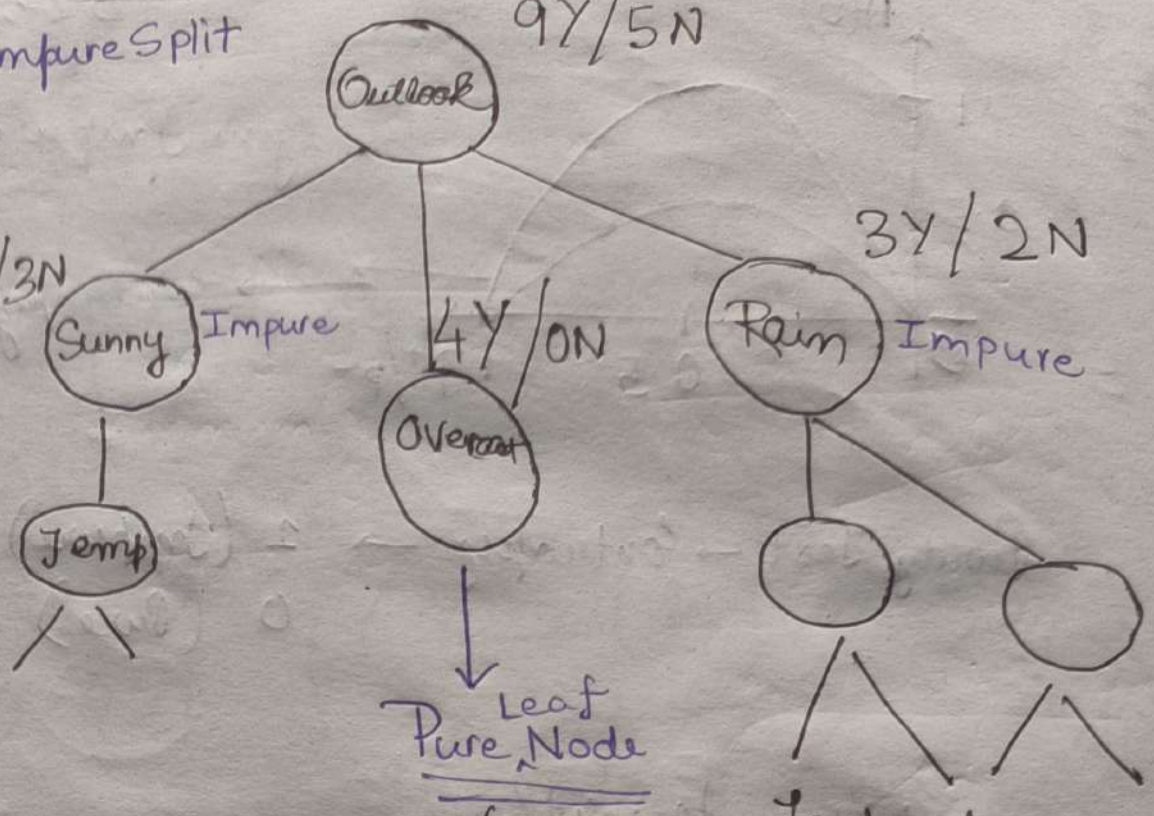


Decision Tree

→ nested if else ⇒ Decision Tree

Refer to Dataset - 1

Pure split / Impure Split



1) How to know purity of a given node? (Mathematically)

- Entropy
- Gini Coefficient / Impurity

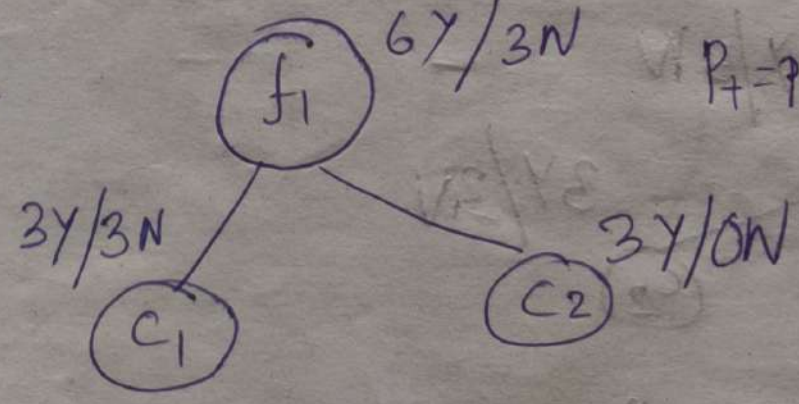
2) How the features are selected?

- Information gain

① ENTROPY

$$H(S) = -(P_+)\log_2 P_+ - (P_-)\log_2 P_-$$

ex



P_+ = probability of yes.

Entropy $H(S_2) = -\frac{3}{3}\log_2 \frac{3}{3} - \frac{0}{3}\log_2 \frac{0}{3}$

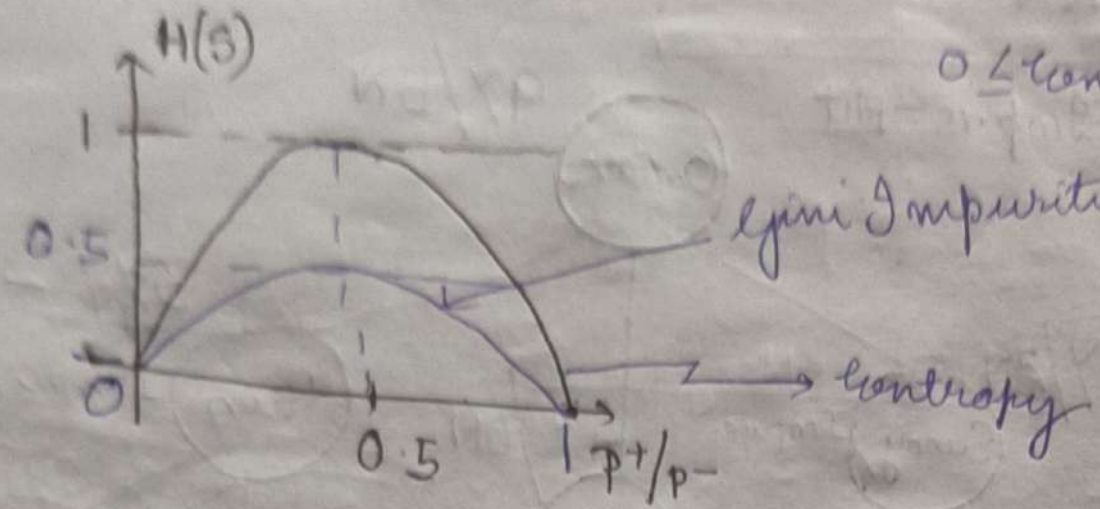
$H(S_{C1}) = 1$

$= -1 \log_2 1 = 0$

② Decision tree classifier by default uses Gini Impurity

② Gini Impurity

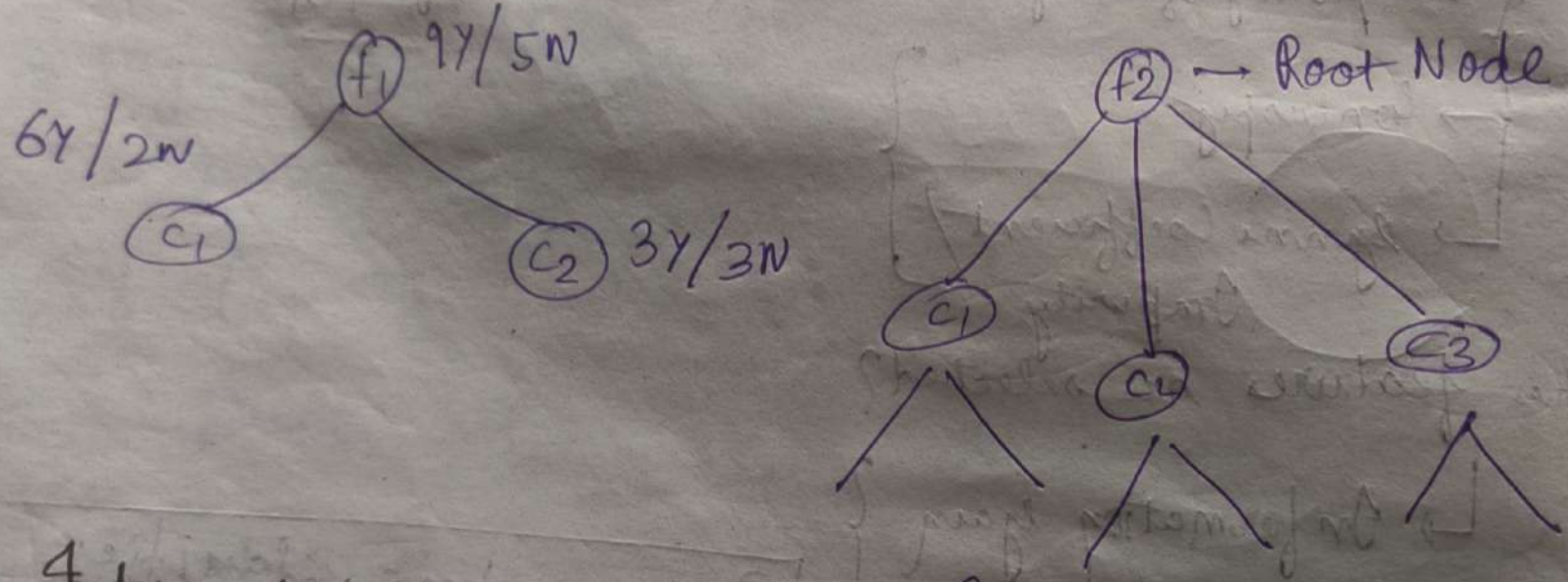
$$GI = 1 - \sum_{i=1}^{n} (p)^2$$



$0 \leq \text{Entropy} \leq 1$

Purity Test → Entropy → 1 (Impure)
0 (Pure)

② Which feature to split?

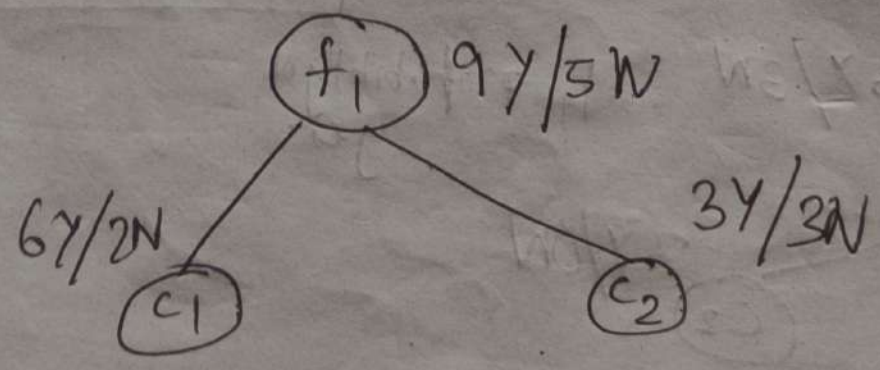


Information gain

$$gain(S, f_1) = H(S) - \sum_{v \in val} \frac{|S_v|}{|S|} H(S_v)$$

↓ Entropy of Root Node

ex→



$$H(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$
$$= -\frac{9}{14} \log_2 (9/14) - \frac{5}{14} \log_2 (5/14) \approx \underline{0.94}$$

$H(S_{vc_1}) = -\frac{6}{8}\log_2 6/8 - 2/8 \log_2 2/8 = 0.81$

$H(S_{vc_2}) = 1$

$\text{gain}(S, f_1) = 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$

$\text{gain}(S, f_1) = 0.049$

$\text{gain}(S, f_2) \text{ if } = 0.051$

$\text{gain}(S, f_2) > \text{gain}(S, f_1)$

$\therefore f_2$ should be selected.

* Gini impurity

$GI = 1 - \sum_{i=1}^{n} p^2$ $\quad n = 2$ output $\{$ yes / no

$= 1 - ((p_+)^2 + (p_-)^2)$

$= 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$

$= 1 - \left(\frac{1}{2}\right)$

$= \underline{0.5}$

2Y/2N ⇒ Complete Impure Split

* Decision tree has one of the worst time complexities.

→ More execution time → Entropy

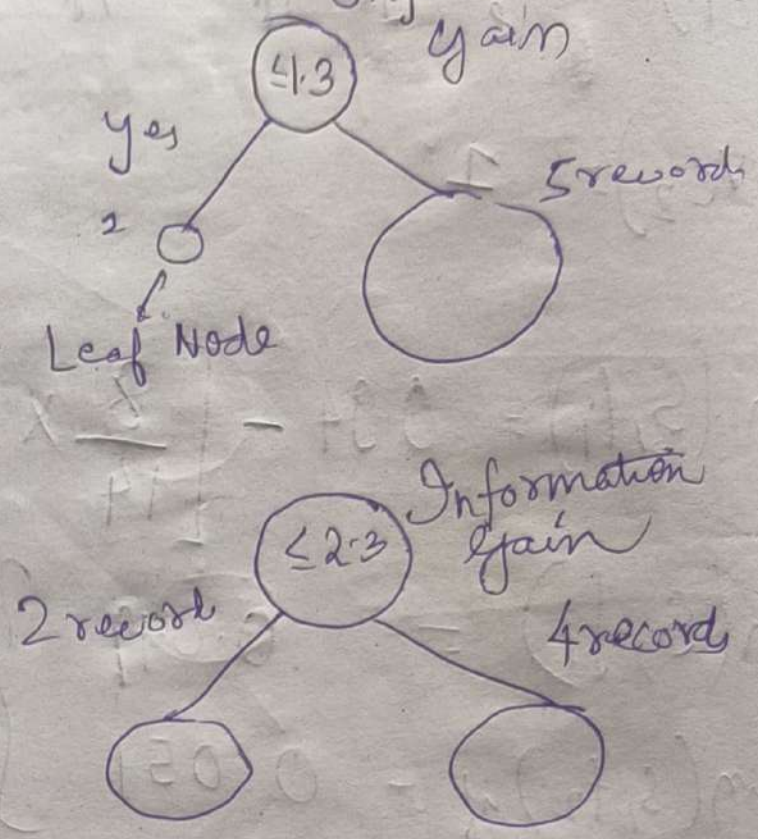
→ Gini Impurity is faster

huge no. of feature apply GI

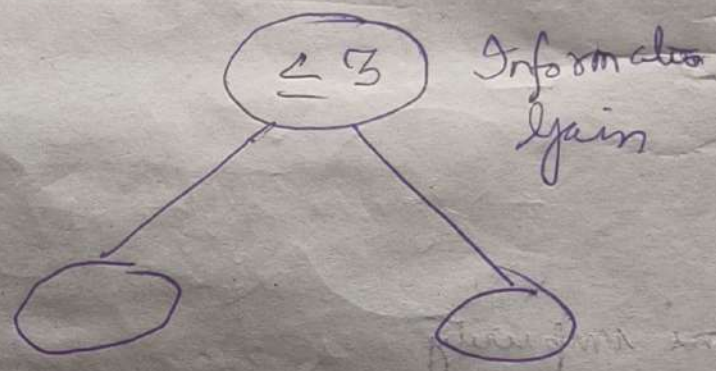
less no. of feature — Interest

f_1 → Continuous

f_1	O/P
2.3	1.3
1.3	2.3
4	3
5	4
7	5
3	7



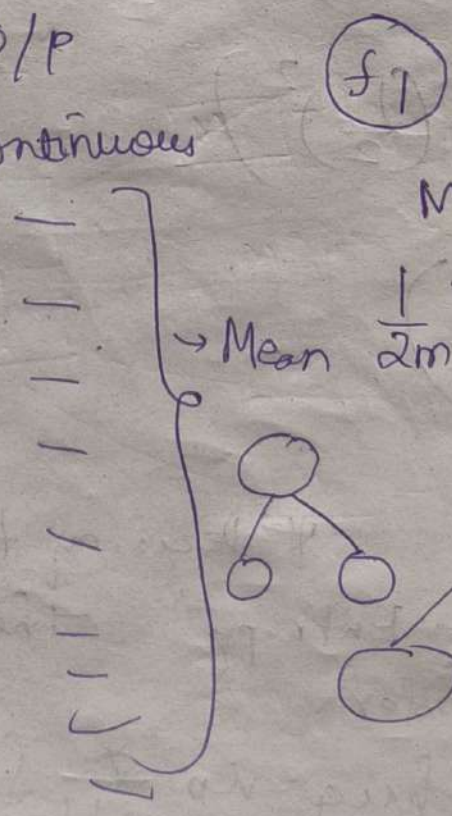
Best information gain will get selected, and from there splitting will happen.



Decision Tree Regressor

f_1 f_2 O/P

Continuous



MSE or MAE

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

Variance Reduction

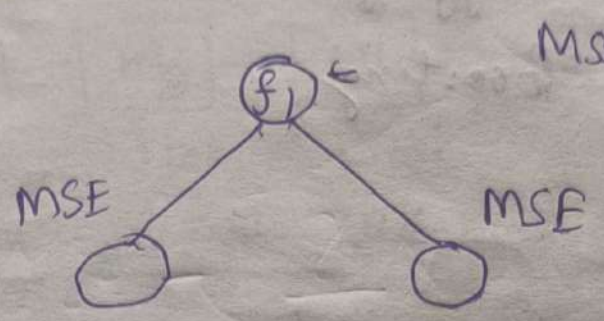
$= Var(Root) - \sum W_i Var(Child)$

Whose variation reduction is more, we will do

We take mean if two values present at leaf node.

Decision tree regressor

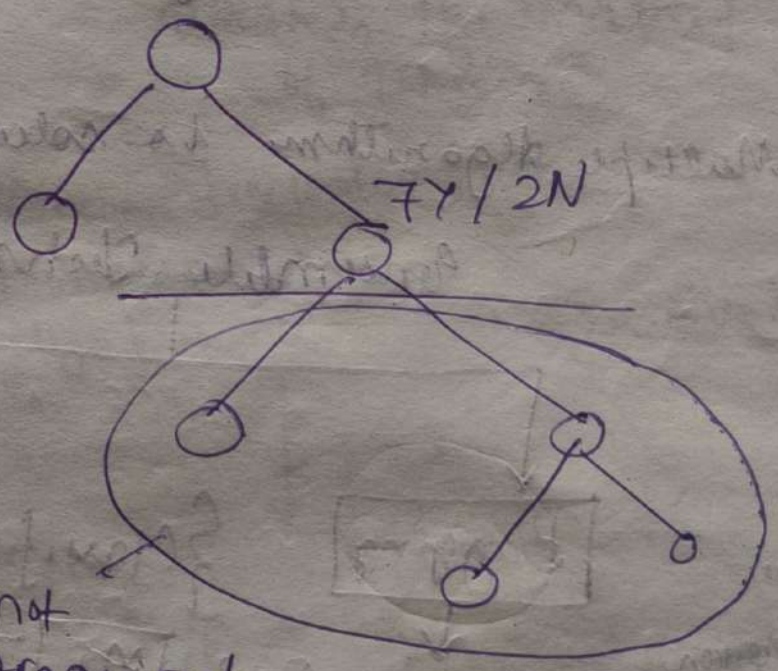
f_1	O/P
C_1	20
C_1	24
–	26
–	28
–	30
:	:



Hyperparameter

Decision → Overfitting

① Post Pruning
② Pre Pruning
} → To avoid overfitting.



Post Pruning

Pre pruning hyperparameters

max_depth, max_leaf → GridSearchCV

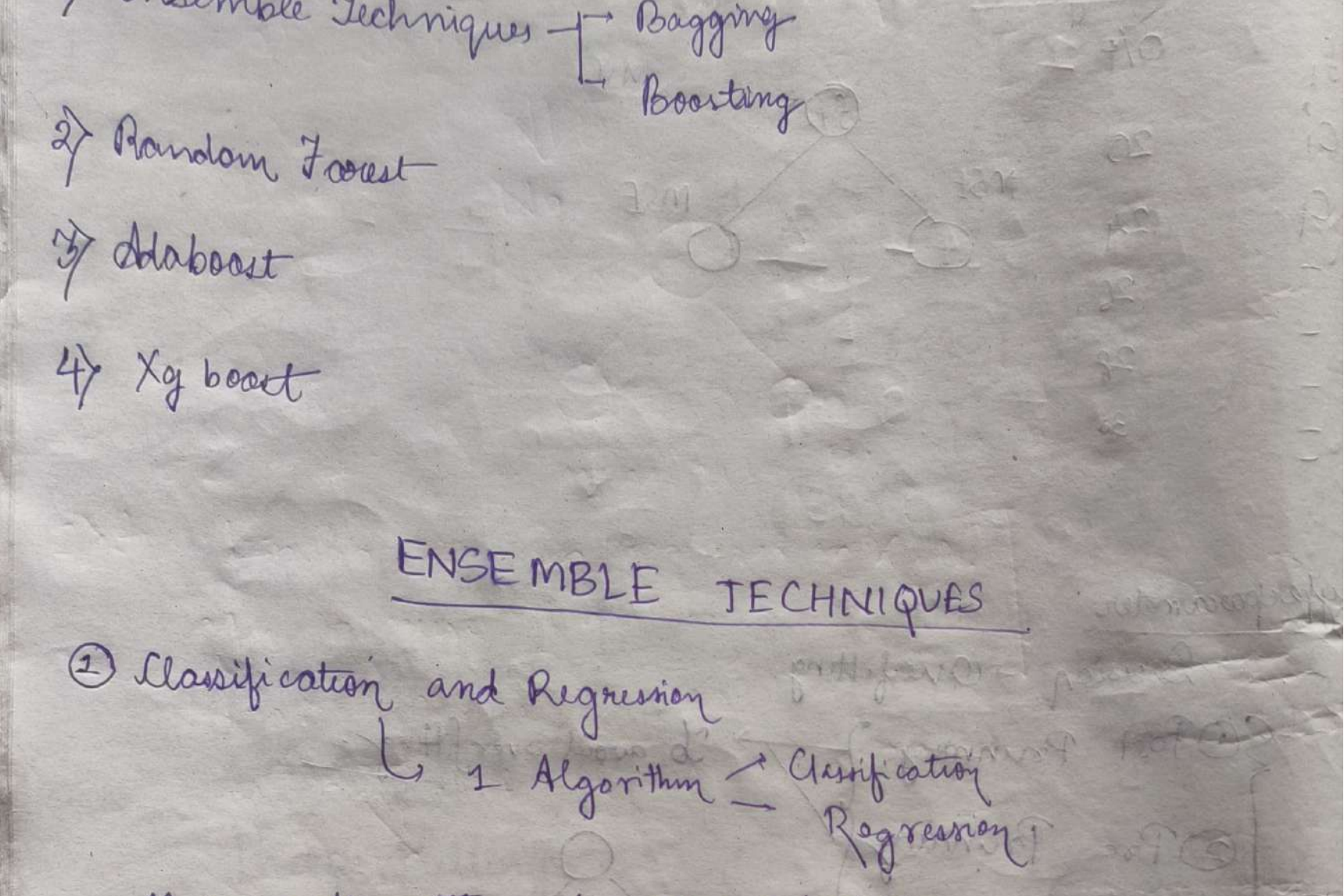
Agenda

1) Ensemble Techniques → Bagging, Boosting
2) Random Forest
3) Adaboost
4) Xg boost

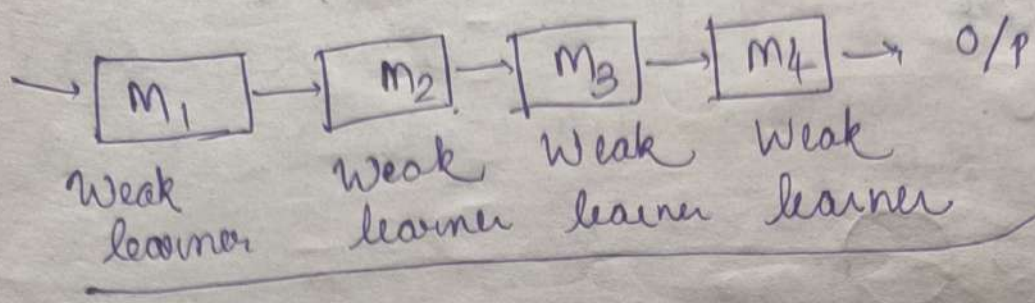
ENSEMBLE TECHNIQUES

① Classification and Regression
↳ 1 Algorithm → Classification, Regression

Multiple Algorithms to solve a problem.

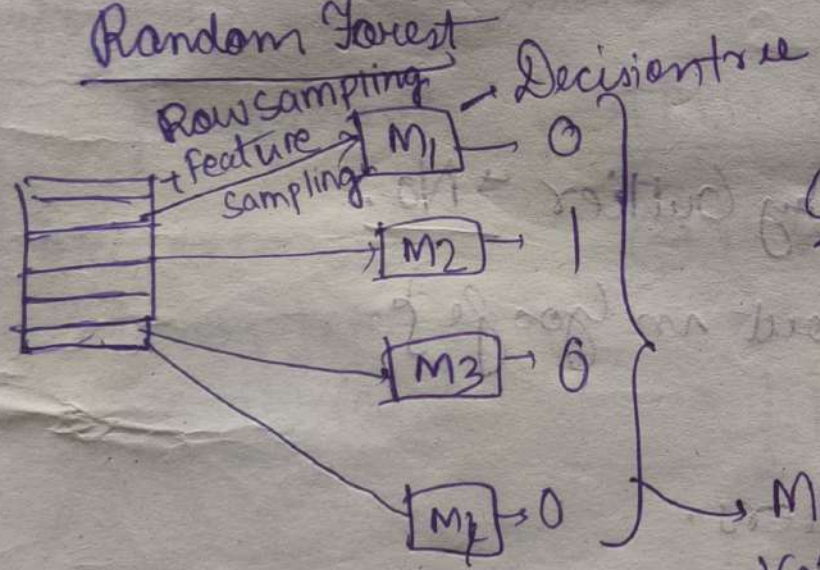


Boosting (models will be present sequentially)

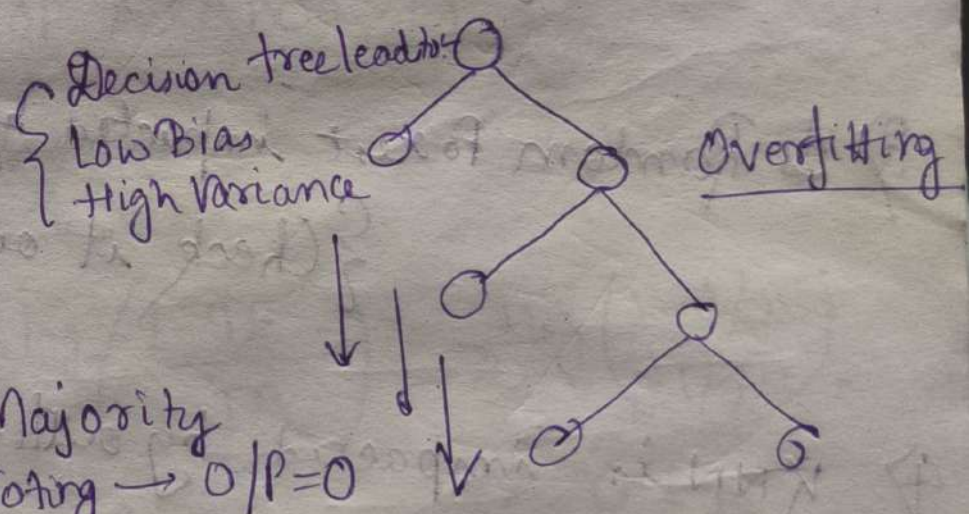


Bagging	Boosting
① Random Forest classifier	① Adaboost
② Random Forest Regressor	② Gradient Boost
	③ XG Boost (Extreme Gradient Boost)

1) Random Forest classifier and Regressor



what is the main problem of DT?

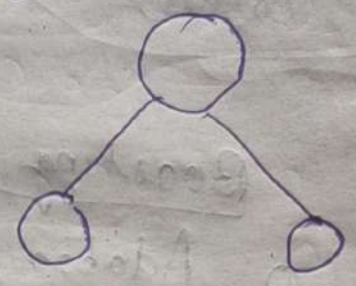


* In random forest, only decision tree is used.
* Because of the overfitting condition of the decision tree, we are combining many decision tree so that we can get a generalised model which has low bias and low variance.
* In random forest regressor, the mean is taken out of the values given by various decision trees.

1) ~~Also~~ Is normalization required in Random Forest? or Decision Tree

⇓

No



Decision tree does splits, so even if we minimize the data also the splits will not be of that much importance.

2) Is standardization required in KNN?

Yes, because it has Euclidean and Manhattan distance, so that the computational distance becomes easy.

3) Is Random Forest impacted by Outlier ⇒ No.

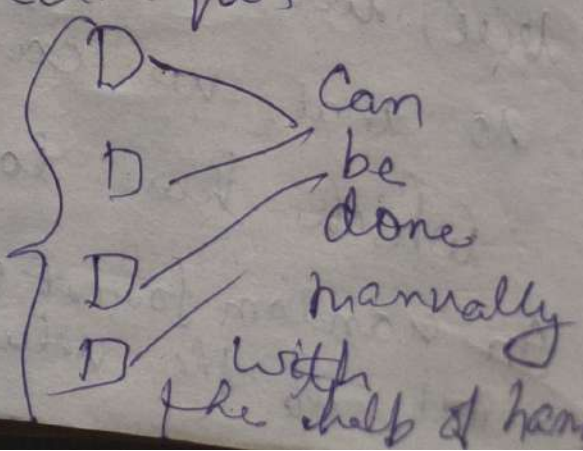
{Check it out in Google}

4) KNN is impacted by outliers.

Bagging → Random Forest is used mostly

or

Custom Bagging Techniques can be used:-



BOOSTING

i) **Adaboost** → Decision Trees specifically

$0 < \text{weight} < 1$

$\sum \text{Weight} = 1$

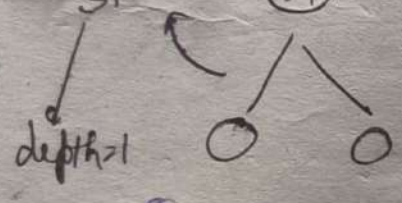
f_1	f_2	f_3	f_4	O/P	Weight	Updation
—	—	=	—	yes	1/7	→ 0.05
—	—	—	—	No	1/7	→ 0.05
—	—	—	—	—	1/7	→ 0.05
—	—	—	—	—	1/7	→ 0.349
—	—	—	—	—	1/7	→ 0.05
—	—	—	—	—	1/7	→ 0.05
—	—	—	—	—	1/7	→ 0.05

$\frac{1}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7} \geq 1$

* Initially equal weight is given to every record.

1-level decision tree ↑ Stumps

info gain / entropy / gini → is used to select which feature to branch out



{Weak learner}

STEPS OF ADABOOST

Step ① Total error from 1st decision tree

= 1/7 (assume)

S-② Performance of Stump

$$= \frac{1}{2}\log_e\left(\frac{1-TE}{TE}\right) = \frac{1}{2}\log\left(\frac{1-1/7}{1/7}\right)$$

$$= 0.895$$

S-3 Updation of Weight

$$\text{New Sample Weight} = \frac{\text{Correct Record}}{\text{Weight} * e^{-(\text{Perf. of Stump})}}$$

$$= \frac{1}{7} \times e^{-0.895} = 0.05$$

$$\text{Incorrect Weight} = \text{Weight} * e^{(\text{Perf of Stump})}$$

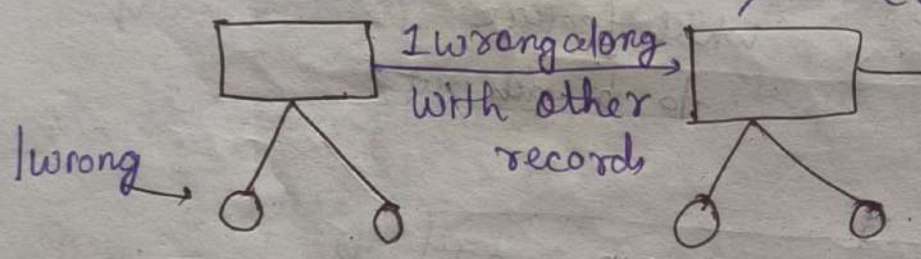
$$= 1/7 \times e^{0.895} = 0.349$$

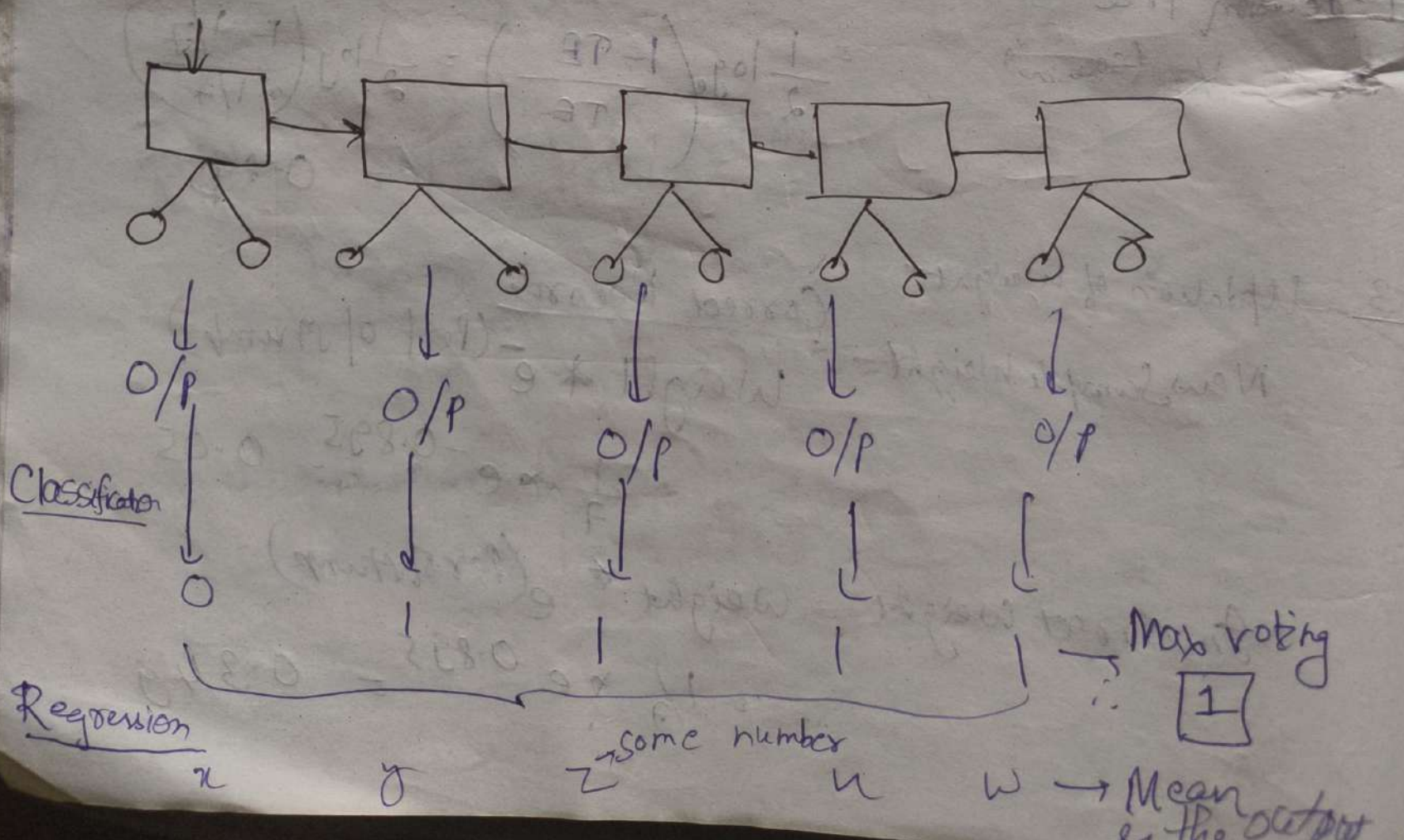
→ New Weight

New Weight	Normalized Weight		Buckets
0.05	0.05 / 0.649	→ 0.07 →	[0 - 0.07]
0.05	0.05 / 0.649	→ 0.07 →	[0.07 - 0.14]
0.05	0.649	→ 0.07 →	[0.14 - 0.21]
0.349	0.349 / 0.649	→ 0.537 →	[0.21 - 0.747]
0.05	0.649	→ 0.07 →	[0.747 - 0.75]
0.05	0.05 / 0.649	→ 0.07 →	[-]
+ 0.05	0.05 / 0.649	→ 0.07 →	[-]
Σ ≠ 1			

has the biggest bucket size.

= 0.649





Black box model VS White Box Model

→ Linear Regression → White box
→ Random Forest → Black box
→ Decision Tree → White box
→ ANN → Black box

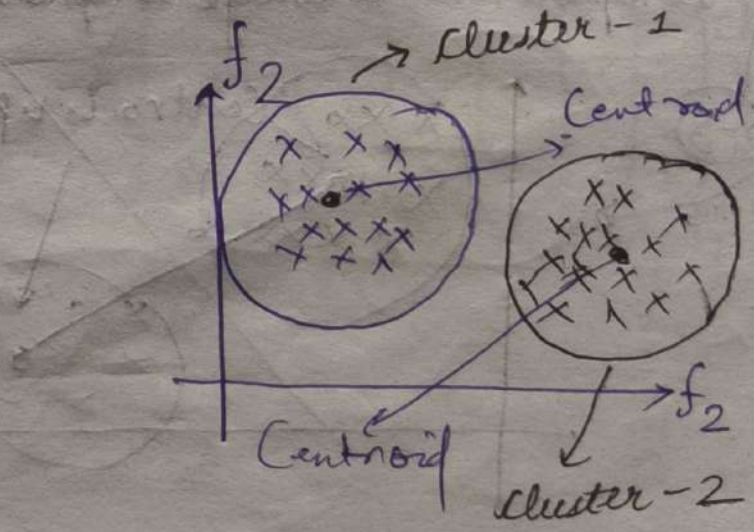
UNSUPERVISED ML
MACHINE LEARNING

① K-Means Clustering
② Hierarchial Clustering
③ Silhoutte Score
④ DB-Scan Clustering

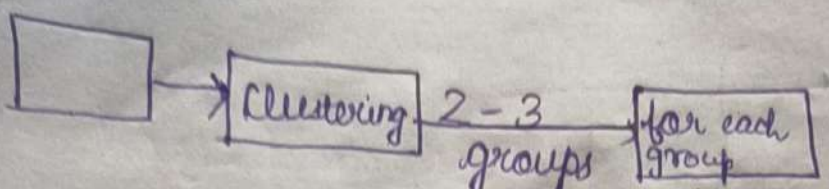
} Agenda

K Means Clustering

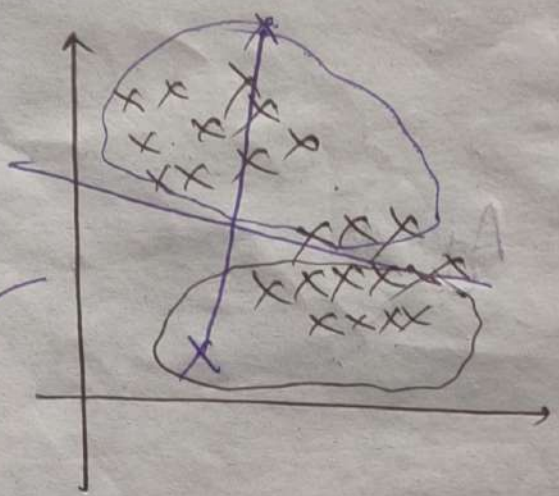
f_1	f_2	output
—	—	clustering ⇓
—	—	{ Similar kind of data }
—	—	
—	—	
—	—	



Custom Ensemble Technique



K - Means → K = Centroids



K = 2

Euclidean distance

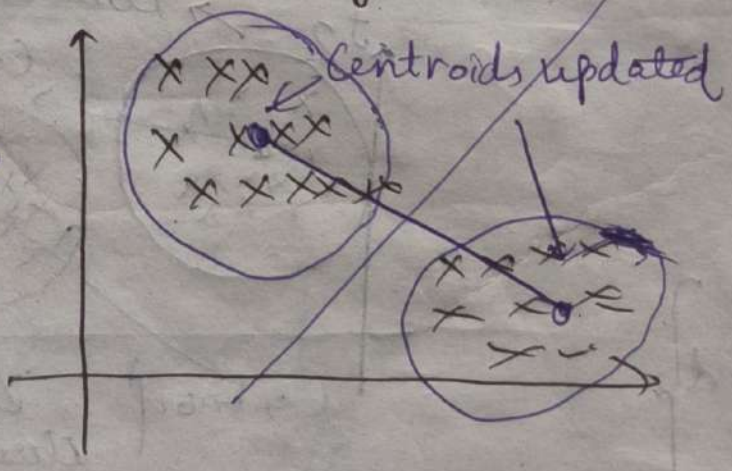
Steps :- → centroids

① We try K-values ⇒ Suitable ex K = 2

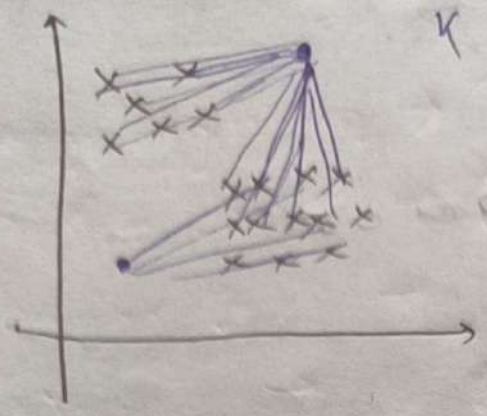
② Initialize K - number of centroids

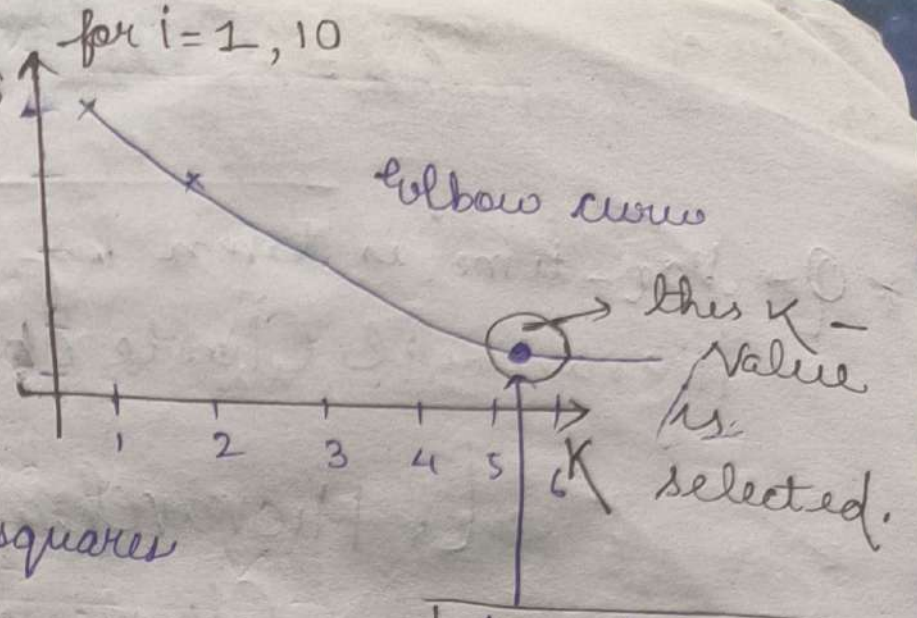
③ Compute the average to update centroids.

5-3



Elbow method (helps to initialize K-value
or optimise K-value
or what K-value
to choose)

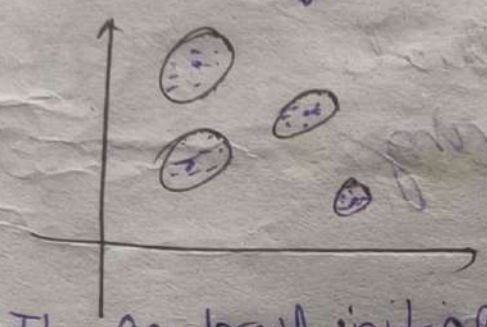




WCSS → within cluster sum of squares

for finding K-value : we use elbow curve
for validating our KNN model : we use Silhoutte score

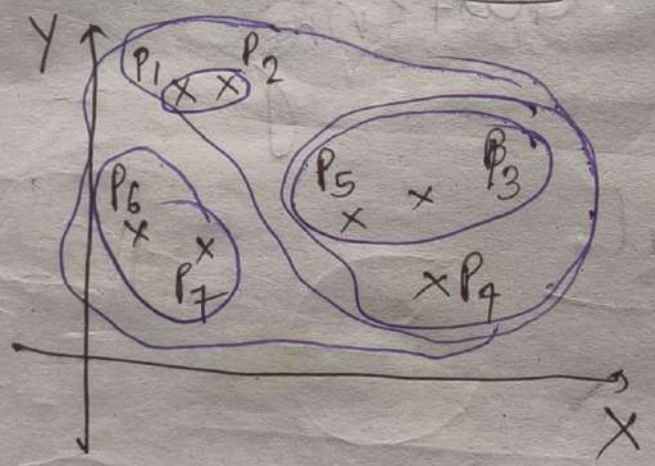
abrupt change from here and we take this K=Value



for K=4 ⇒ 4 groups
K=n ⇒ n groups

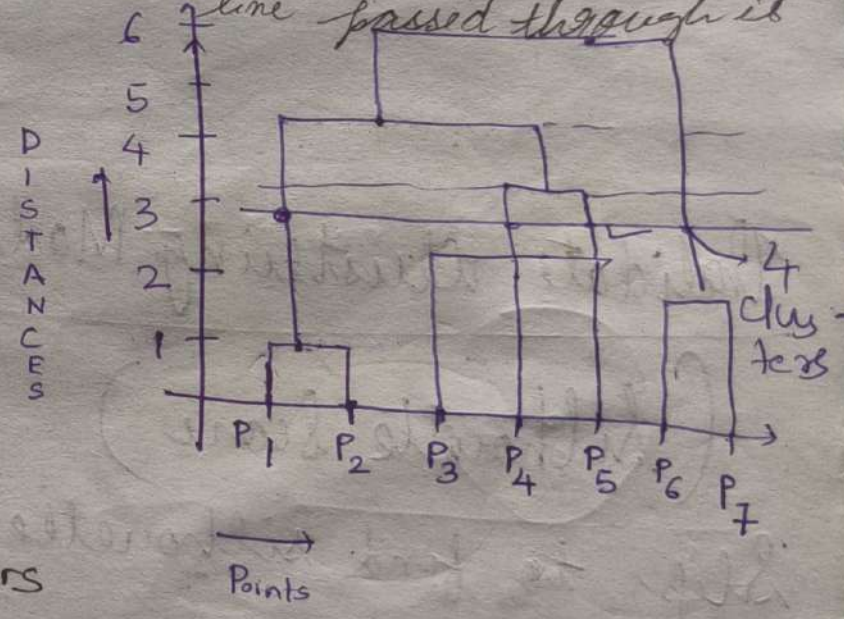
⑥ * The centroid initially taken should be far away from the cluster. {K means ++}

② Hierarchial clustering



you need to find the longest vertical line that has no horizontal line passed through it

Dendogram



S-1
* Smallest distance between pairs are grouped.

S-2
* Then, consequently other points are grouped

S-3
* All the points have been grouped into 1-group.

S-4
* Find the longest vertical line, which does not have any horizontal line passing through it

S. Draw a horizontal l. - passing through it and find the

Number of intersecting points passing through the horizontal line, the number of intersection equals the number of groups.

* Q → Max-time is taken by K Means or Hierarchial Clustering?

↳ Hierarchial Clustering (due to Dendogram)

* Dataset is small → Hierarchial Clustering

* Dataset is large → K-Means Clustering

Validate Clustering Model (Silhouette Score)

Steps to find silhouette score :-

(1) a_i ↓ find out average of all the distances from the centroid in one of the clusters.

(2) b_i is the average of all the distances from all the points of one cluster to the points of another cluster

③ for a good model
$b(i) >> a(i)$

for a worse model
$a(i) >> b(i)$

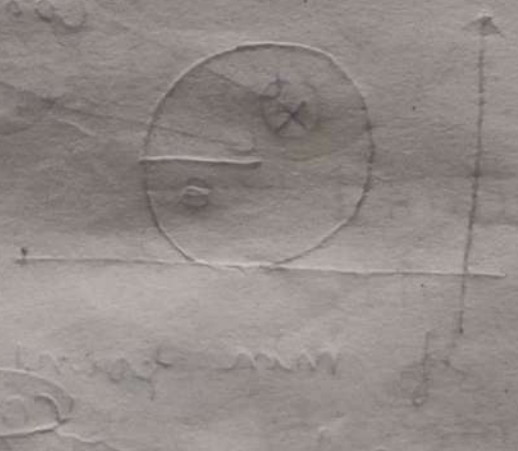
④ Silhouette score varies from −1 to 1.
−1 being the worst
to 1 being the
best

⑤ Formula of Silhouette score (S)

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

i^{th} cluster

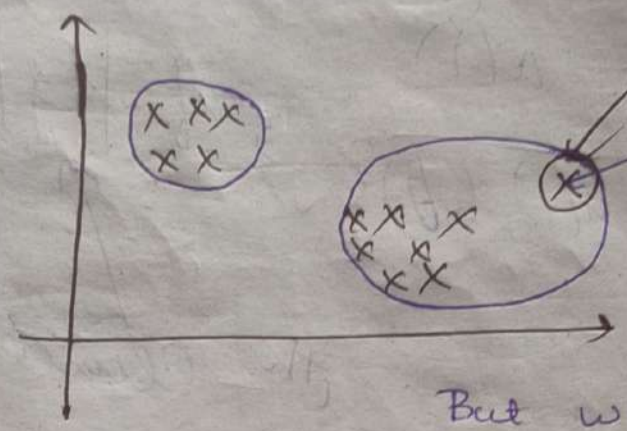
$S(i) = 0$, if $|C_i| = 1$



DB-Scan Clustering

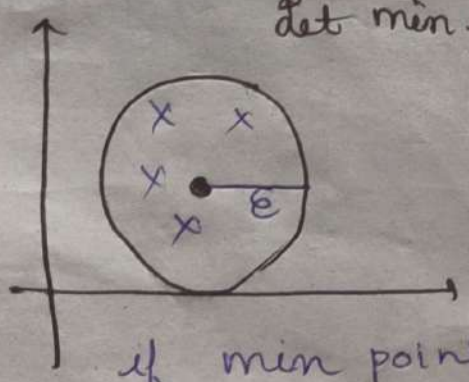
(Density Based Spatial Clustering of Applications with Noise)
(DBSCAN)

1. Min pts (one hyperparameter) (eps → min = 4, 5 etc)
2. Core points
3. Border points
4. Noise points
5. Epsilon (Radius of Circle)

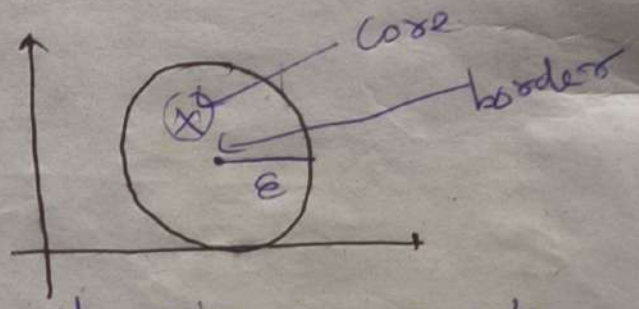


But we want to leave out the outliers, in some cases therefore we use DB SCAN.

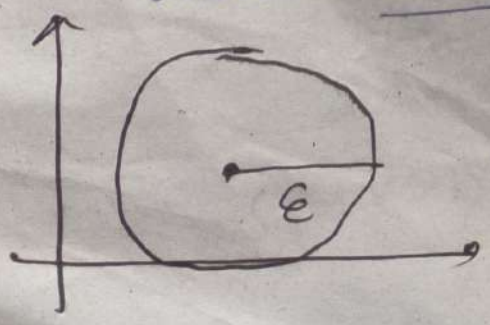
Let min pt = 4



if min point = 4, and no. of points inside circle = 4, then the centre becomes core point



if min point = 4, and no. of point (core) ~~=4~~, ~~the~~ is greater than 0 but less than 4, (here 1), the center become border point



if no point inside the circle, the center becomes noisy point.

*We neglect all the noise points, and combine border and core point.

DB Scan is much better than K-Means

*Practical implementation

↳ we use Silhouette score for validation & clustering if my cluster ~~is high~~ we get choices as for n = 2 or n = 4, choose the larger model as it helps in generalisation.

Defⁿ of BIAS AND VARIANCE

Training Dataset = 80%
Test Dataset = 20%
⇒ Overfitting

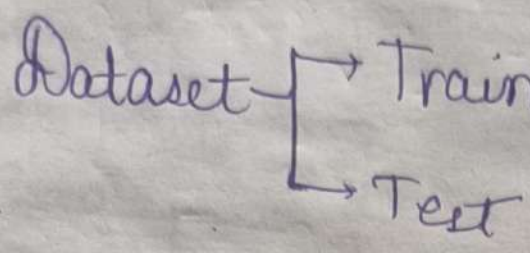
Low Bias ✓
High Variance ✓

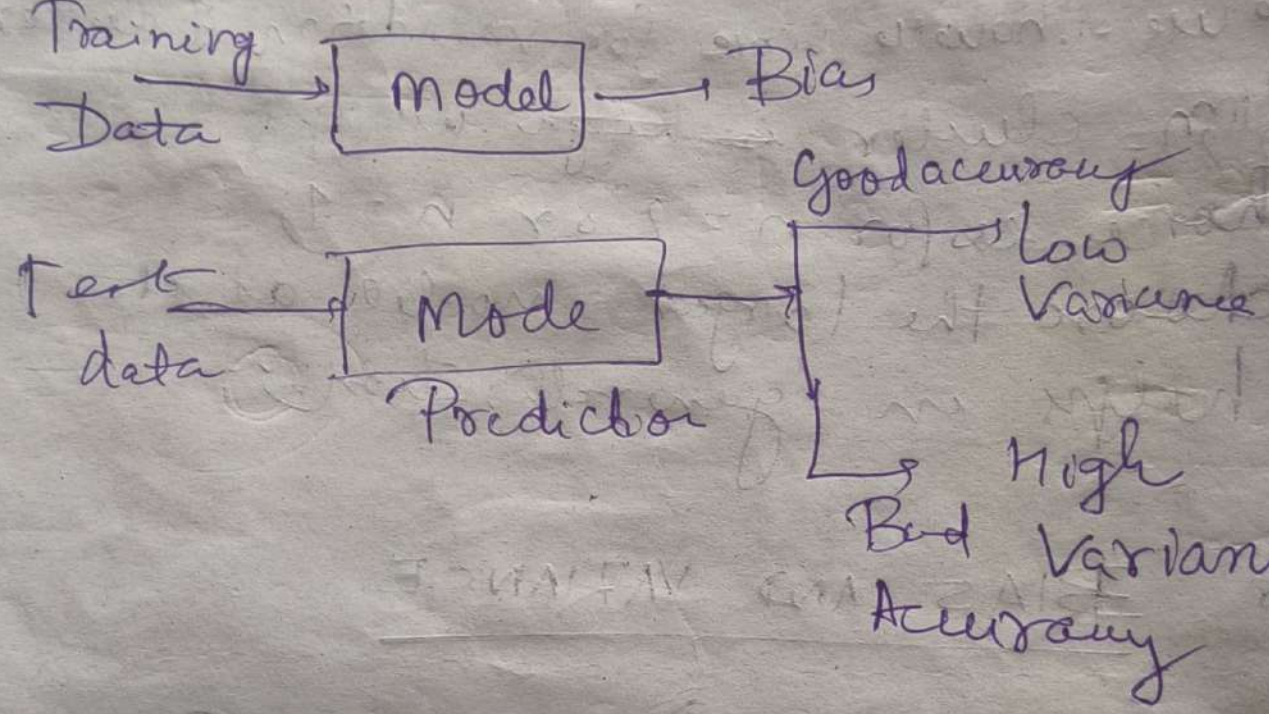
① Bias :- It is a phenomenon that skews the result of an algorithm in favour or against an idea.

↳ Training Dataset

Training Dataset → Model → High Bias (if it does not perform well with training dataset)
Low Bias (if it ~~does not~~ perform well with training dataset)

Variance: Variance refers to the changes in the model when using different portion of training or test data.





Model 1	Model 2	Model 3
Train Acc: 90%	60%	90%
Test Acc: 75%	55%	92%
Low Bias	High Bias	Low Bias
High Variance (Overfitting)	High Variance (Underfitting)	Low Variance (Generalized)

DAY 7: XGBOOST CLASSIFIER AND REGRESSOR

Agenda:

1. Xgboost Classifier
2. Xgboost Regressor
3. SVM
4. SVR

① XGBOOST CLASSIFIER

Extreme Gradient Boosting

(internally uses ~~dat~~ decision tree)

{Dataset}

Salary	Credit	Approval	Residual
<=50	B	0	-0.5
<=50	G	1	0.5
<=50	G	1	0.5
>50	B	0	-0.05
>80	G	1	0.95
>50	N	1	0.5
<=50K	N	0	-0.5

Dummy model

→ Base model (Weak learner)

(pr=0.5 in case of classification)

pr = 0.5

Default

Step① Create a Binary decision tree using features

Step② Calculate Similarity Weight

$$= \frac{(\Sigma \text{Residual})^2}{\sum Pr(1-Pr) + \lambda}$$

Step③ Calculate Information gain

Use Binary decision tree in Xg boost always

Similarity = 0.14

Salary [-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]

<=50 [-0.5, 0.5, 0.5, -0.5]

>50 [-0.05, 0.5, 0.5]

Considering $\lambda = 0$ (hyperparamater)

= Similarity = 0

$$= \frac{(-0.5 + 0.5 + 0.5 - 0.5)^2}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)}$$

$$= 0$$

Similarity

$$= \frac{(-0.5 + 0.5 + 0.5)^2}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)}$$

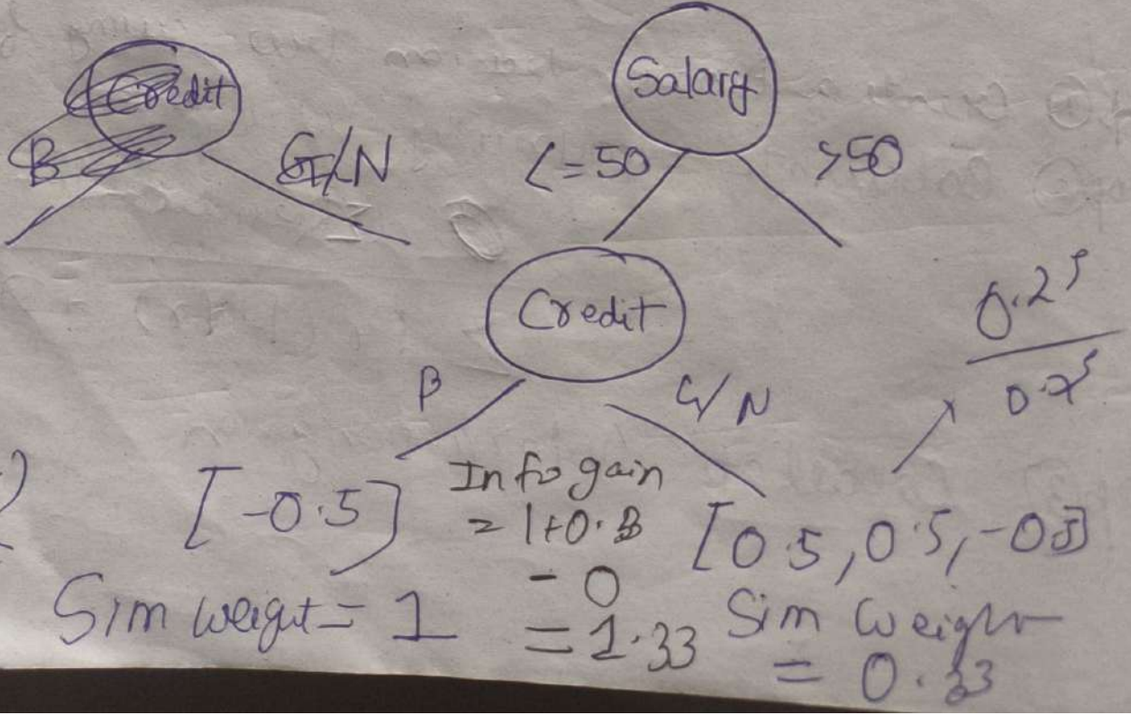
$$= \frac{0.25}{0.75}$$

$$= 0.33$$

(3) Information gain

$$= 0 + 0.33 - 0.14$$

$$= 0.19$$



for base model only / in dataset
ex → for first record

Real probability $= \log\left(\frac{p}{1-p}\right)$

$= \log\left(\frac{0.5}{0.5}\right) = 0$

Binary ... from ... a specific ... in Dataset

$\sigma[0 + \alpha(1)]$ ← learning rate ($\alpha = 0.01$)

σ → Sigmoid (Activation f^n) (Since, it is a classification problem)

(1) → Similarity weight

→ Output from 0 to 1

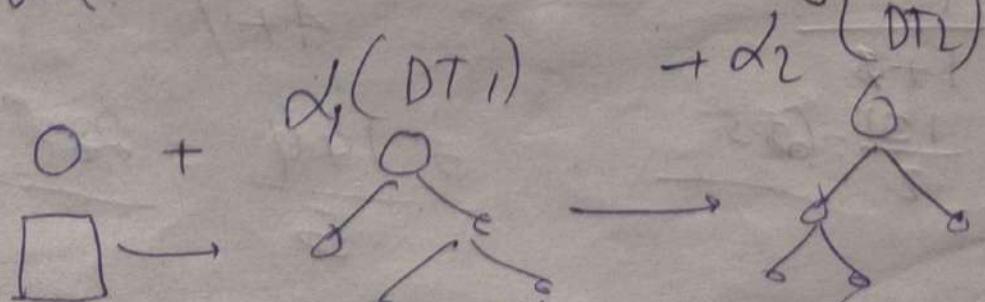
DT - made parallely

$$\sigma\left[0 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \ldots \alpha_4(DT_4) + \ldots - \alpha_n(DT_n)\right]$$

output for any new inference from Record output

* Xg boost is a black box model
* Pre Pruning done (hyperparameter tuning)
* Combining many decision tree

$0 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \ldots \alpha_n(DT_n)$



based on independent features

*) is a hyper para meter

*) is setup by cross-validation

X-G Boost Regressor

Similarity Weight = $\frac{(\Sigma \text{Residual})^2}{\text{No. of Residual} + \lambda}$

Exp	Gap	Salary → O/P	Residual
2	yes	40K	-11
2.5	y	42K	-9
3	NO	52K	1
4	NO	60K	9
4.5	yes	62K	11

Any input → Base model → 51K (avg of all values)

Avg = 51K

[-11, -9, 1, 9, 11]

Exp

≤ 2 (i=1 (let)) | > 2

[-11]

Similarity Weight = $\frac{121}{1+0} = \frac{121}{2} = 60.5$

[-9, 1, 9, 11]

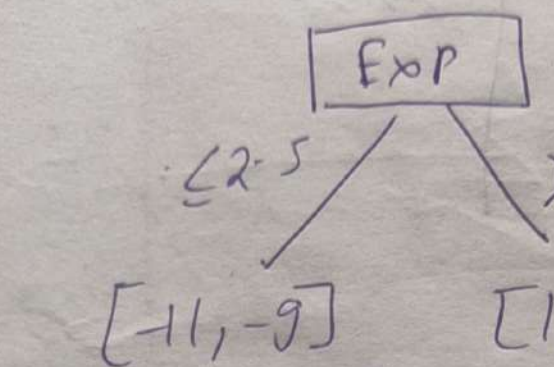
$\frac{(-9+1+9+11)^2}{4+1} = \frac{144}{5} = 28.8$

SW = $\frac{(-11-9+1+9+11)^2}{6} = 0$

$= 1/6$

Information gain = 60.5 + 28.8 − 1/6

= 98.34 − 89.13



$SW = \frac{(-11-9)^2}{2+1}$

$= \frac{400}{3} = 133.33$

$SW = \frac{(1+9+11)^2}{4}$

$= \frac{(21)^2}{4}$

$= 110.25$

$IG = [133.3 + 110.25 - 1/6]$

better than previous split, we will use this split.

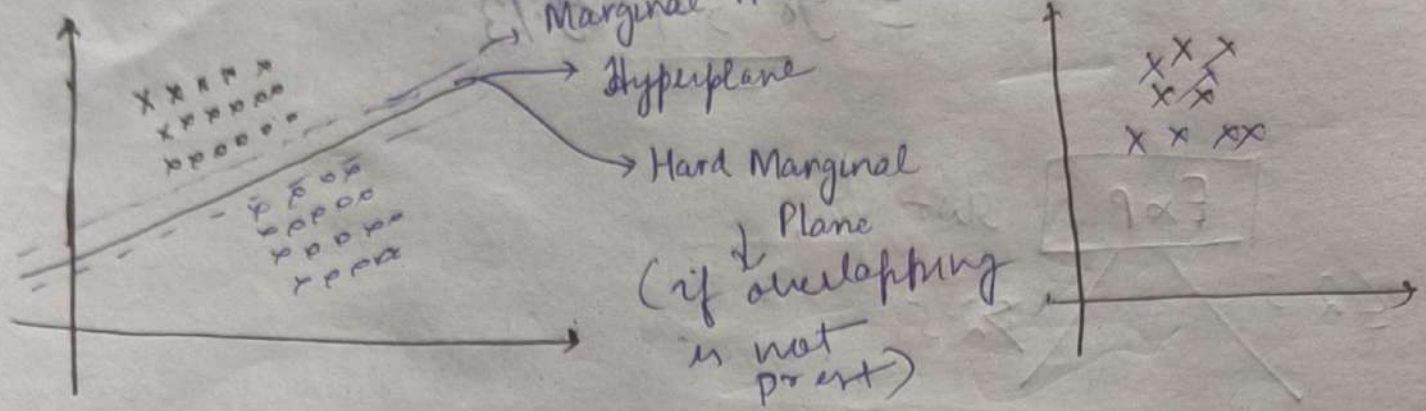
avg at the end of DT_2

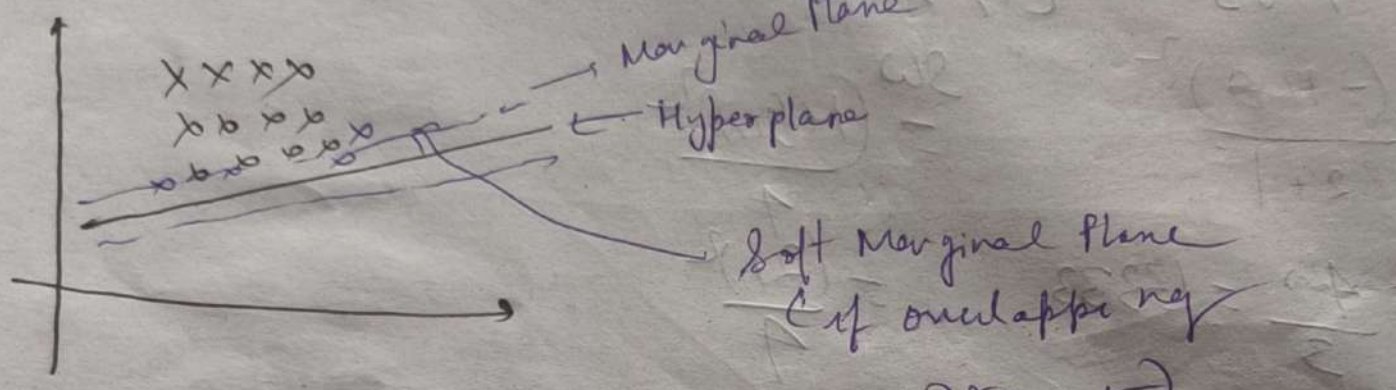
$O/P = 51 + \alpha_1\left(\frac{-11-9}{2}\right) + \alpha_2(DT_2) + \alpha_3(DT_3)$

or

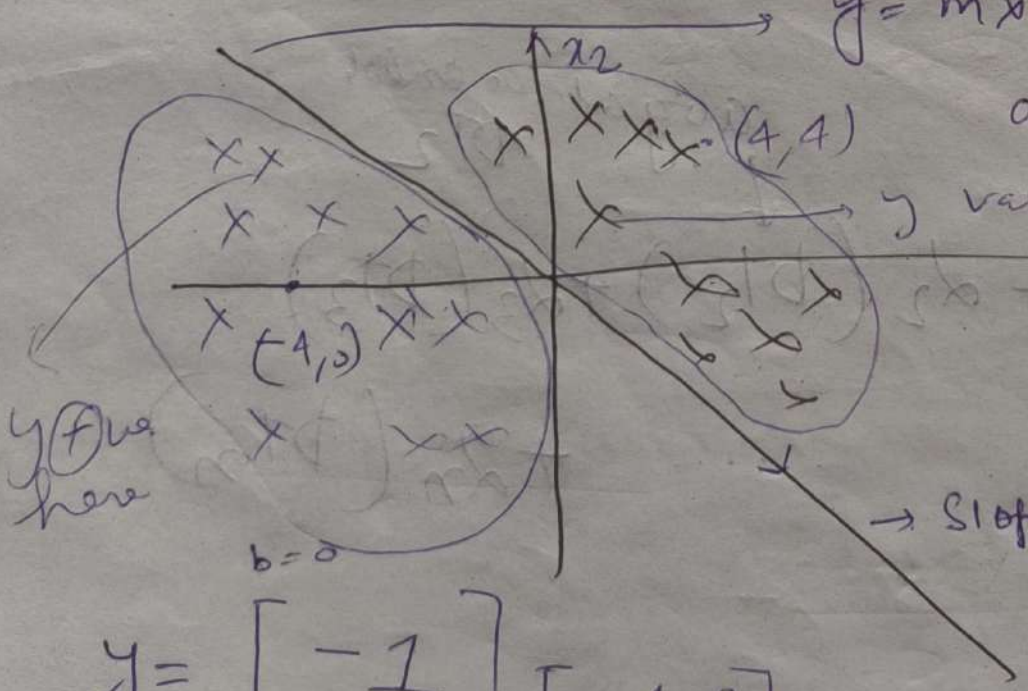
$7 \quad \cdots + \alpha_n(DT_n)$

SVM





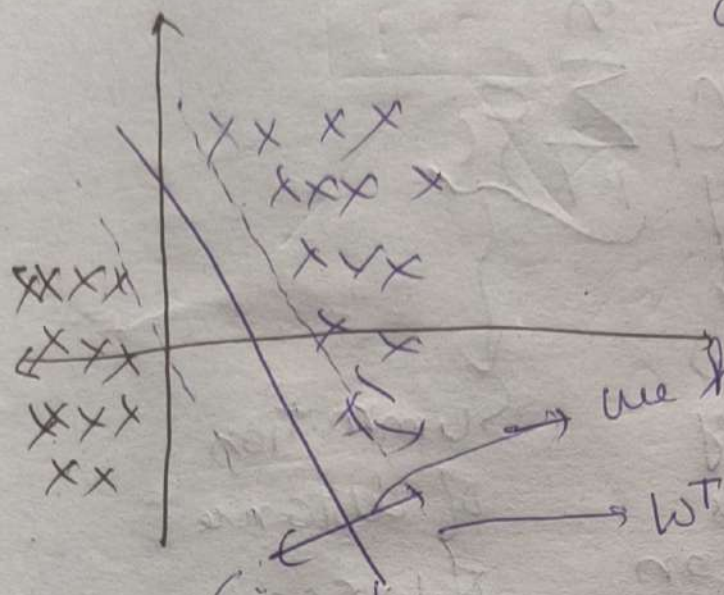
* We focus on creating marginal plane, with larger distance, even though there are errors we use hyperparameter to solve the problem.



$$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [-4, 6]$$

$$= 4 \Rightarrow +ve$$

$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4, 4] = -4$



$ax + by + c = 0$

$\Downarrow$

$w^T x + b = 0$

we have to increase the distance

$w^T x + b = (-1)$ → or $-k$

just to treat as Θw

$w^T x + b = 0$

$w^T x + b = (1)$ → or $+k$ (in many research paper they use $k = 1$)

$w^T x_1 + b = 1$

$w^T x_2 + b = -1$

$w^T (x_1 - x_2) = 2$

Any vector has magnitude

$$\frac{w^T (x_1 - x_2)}{|w|} = \frac{2}{|w|}$$

Maximise $\underset{(w,b)}{2/\|w\|}$ Such that $y_i = \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$

$\Leftarrow$

Major aim $y_i * (w^T x_i + b) \geq 1$

for correct points

Maximize $\frac{2}{\|w\|} \Leftrightarrow$ Minimize $\frac{\|w\|}{2}$
(w,b)

$\Downarrow$ but

Minimize $\frac{\|w\|}{2} + C_i \sum_{i=1}^{n} \xi_i$
(w,b)

C_i: How many errors we can have

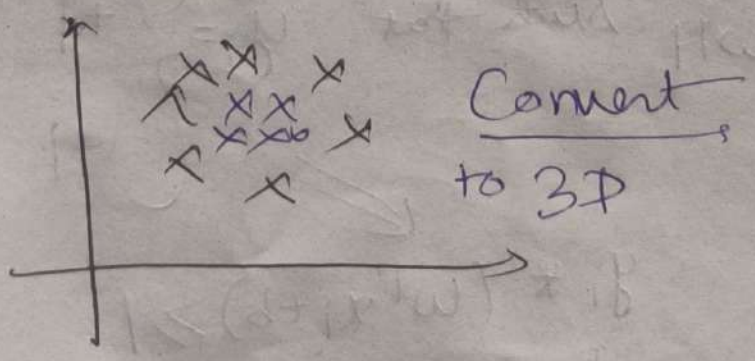
$\sum_{i=1}^{n} \xi_i$: Summation of distance of wrong point from their respective marginal line of their group

SVR

Minimize $\frac{\|w\|}{2}$ + Homework

Rest all are same as SVM.

SVM Kernel



$$f(x_1, x_2) \cdot f^T(x_1, x_2) = K(x_1, x_2)$$

↑

Kernel function

DEEP LEARNING

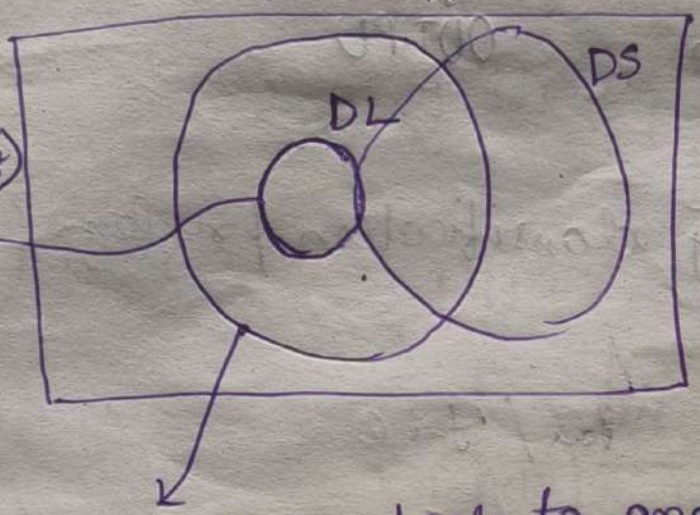
Agenda

① Deep Learning → Perceptron {AI vs ML vs DL vs DS}
② Forward propagation
③ Backward propagation
④ Loss function
⑤ Activation functions
⑥ Optimizers

AI VS ML VS DL VS DS

→ Deep learning

→ AI {application which can do its own task without any human intervention}



Researches (1958)
↓
Multi layred neural network {Perceptron}
{Mimic the human brain}

1. Netflix app
2. Self Driving Cars
3. Amazon Application
4. Sofia

ML → Stat tool to analyze the data, visualize the data, predictions, forecasting, clustering
(ML is subset of AI)

Why deep learning is becoming popular?

① 2005 → ORRUT, FACEBOOK, INSTA, WHATSAA, LINKEDIN, TWITTER
DATA → exponentially ↑

2008 → {Big Data} → Efficiency

2013 → Company had huge amount of data
↓↓
Seamless products {AI became popular}

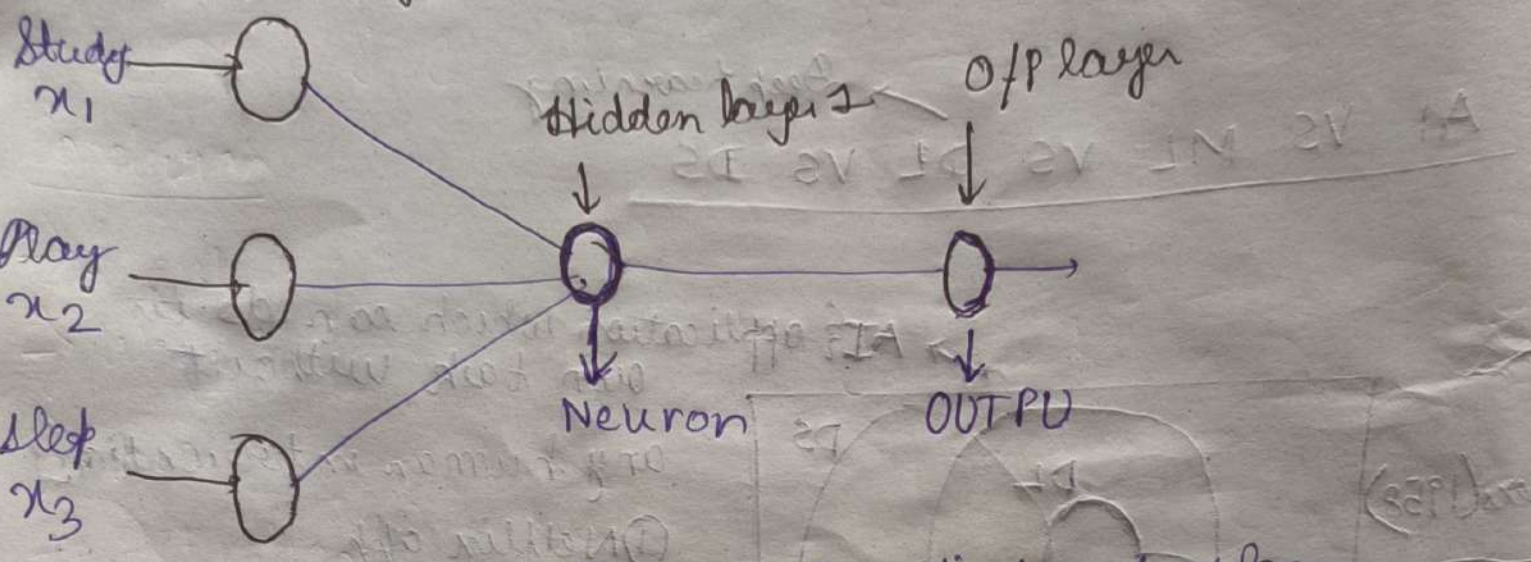
(2) Hardware advancements (NVIDIA) → GPU's

↓

Training the model

GPU's → cost ↓↓↓

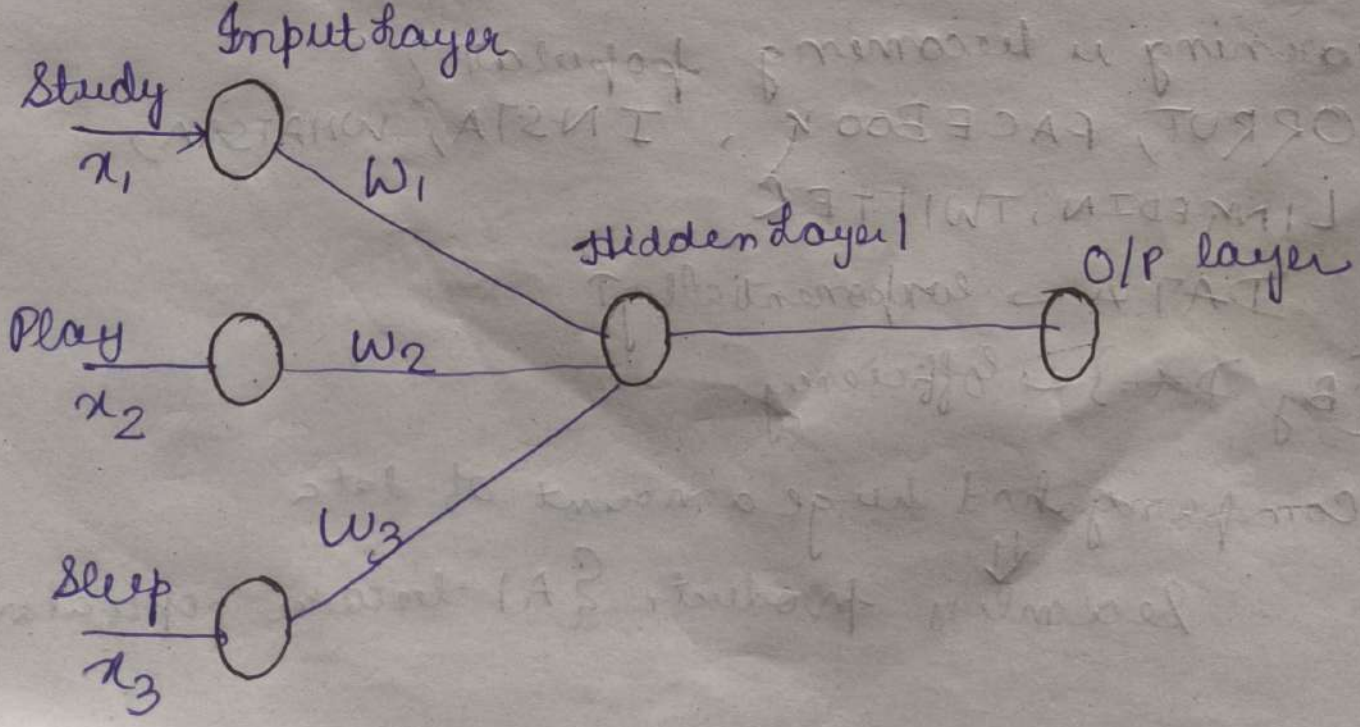
Perceptron { Singled layer Neural Network }

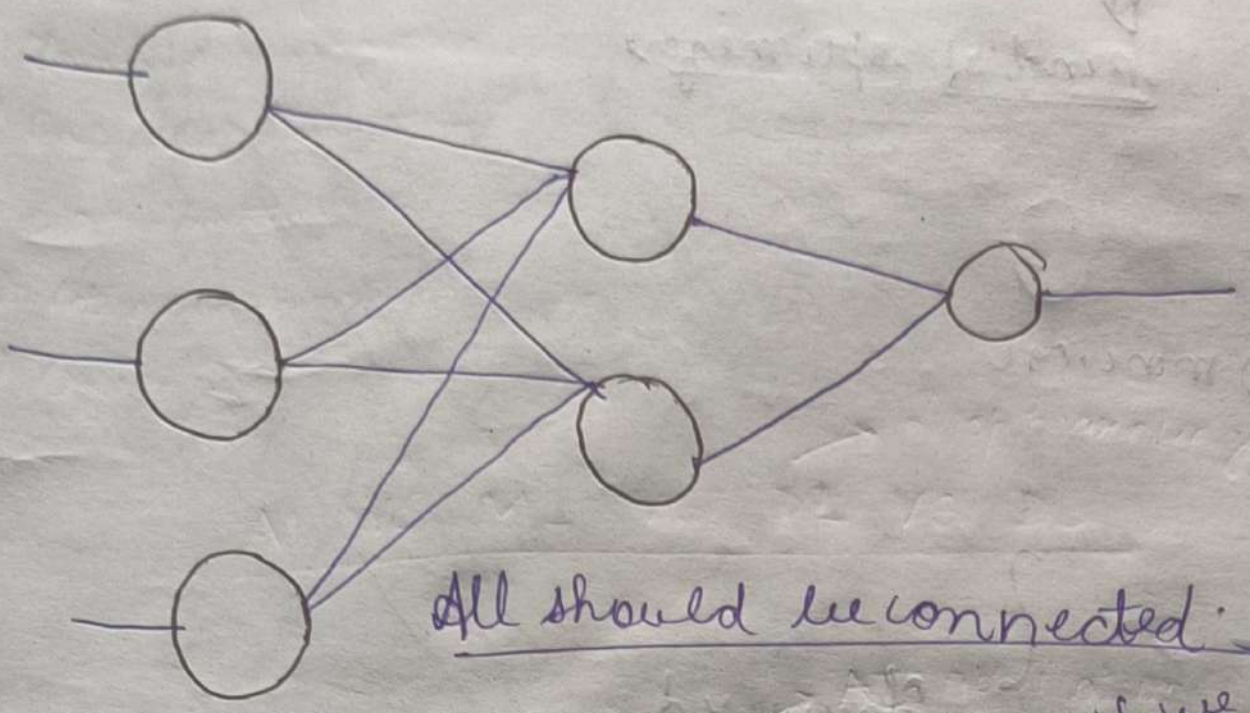


Binary classification problem

Dataset

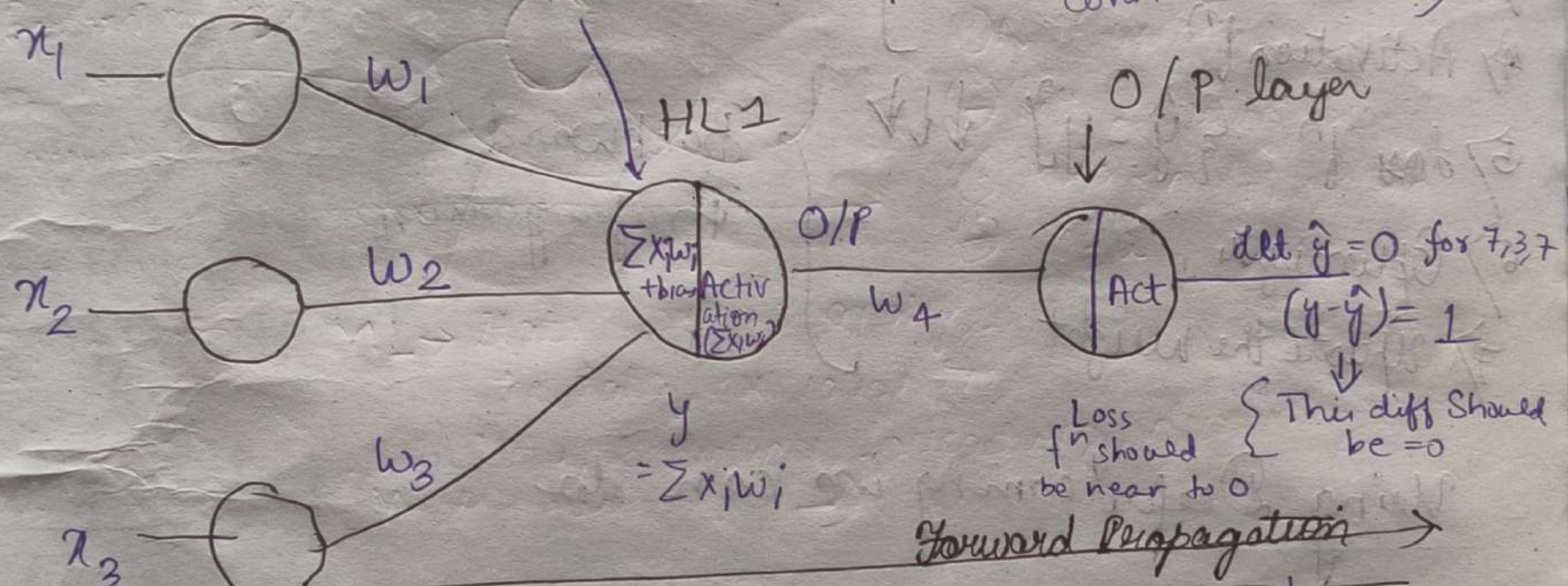
Study	play	sleep	Pass / Fail
7	3	7	1
2	5	8	0
4	3	7	1





All should be connected.

Also bias will be added (So that if we can continue even if we have all weights = 0)
($y = mx + c$, c ↓ constant)



Forward Propagation →

← (using Optimizers) Back Propagation {to update weights}

Just like updation in gradient descent in linear Regression

$$\sum x_i w_i = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$= w^T x + b$$

$y = \beta_0 + \beta x$

$y = \beta^T x$

$y = mx + C$

for Binary Classification

Sigmoid Activation $= \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-(\sum w_i x_i + b)}}$ = 0 or 1

↓

whether the neuron is activated or not

$\geq 0.5 = 1$

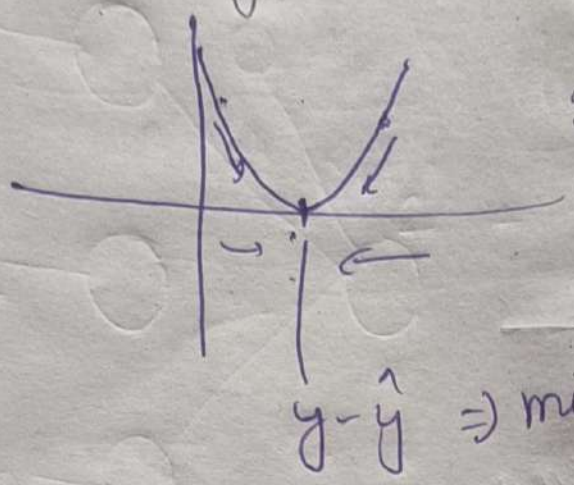
$< 0.5 = 0$

In Simple Linear Regression,

Gradient Descent

⇓

kind of optimizers



$y - \hat{y}$ ⇒ minimal

Conclusion

1) Input layers
2) Weights
3) Bias
4) Activation f^n
5) Loss f^n $\{\hat{y} - y\}$ ↓↓↓
6) Optimizers
7) Update the weight

} Forward Propagation

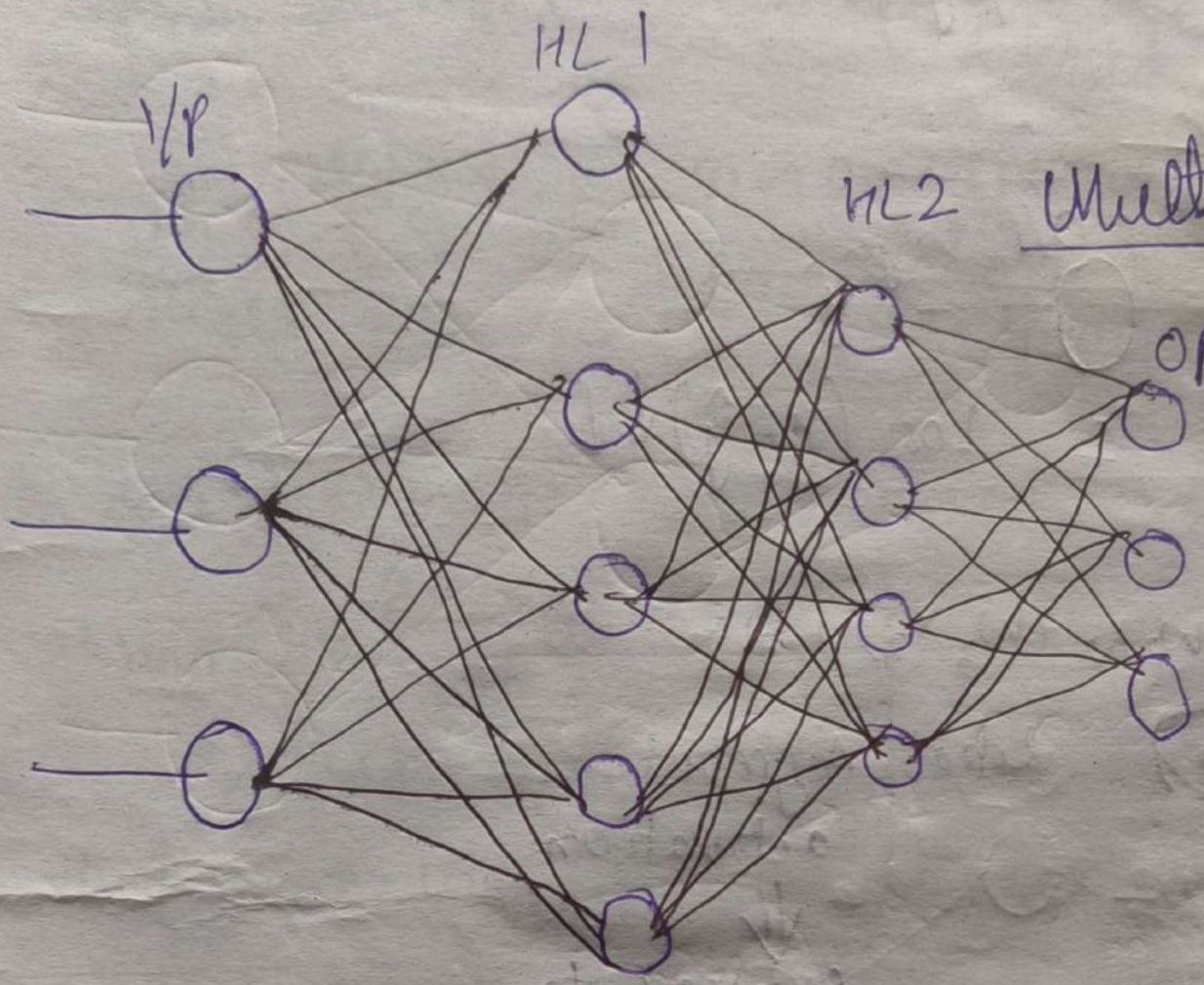
} Backward Propagation

Using DeepLearning we can do

① ANN
② CNN
③ RNN
④ Object Detection

Multi Layered Network

In DL, we can also have more than 1 neuron in output layer (multiclass classification)



Multi Layered Network

1) ANN
2) CNN
3) RNN

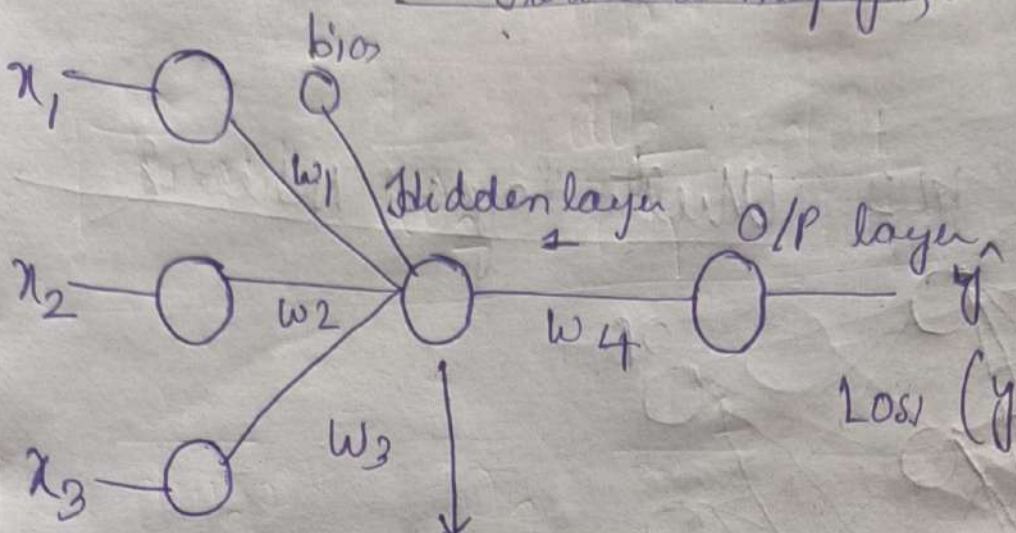
} NLP (7 days)

Day-2-Deep Learning

Agenda

1. Forward Propagation
2. Chain rule of derivative
3. Vanishing gradient Problem
4. Loss function

Forward Propagation



$y = (w_1x_1 + w_2x_2 + w_3x_3) + bias$

$z = \sigma(y)$ like sigmoid activation

$y = w^T x + b$

help in solving non linear properties / problem

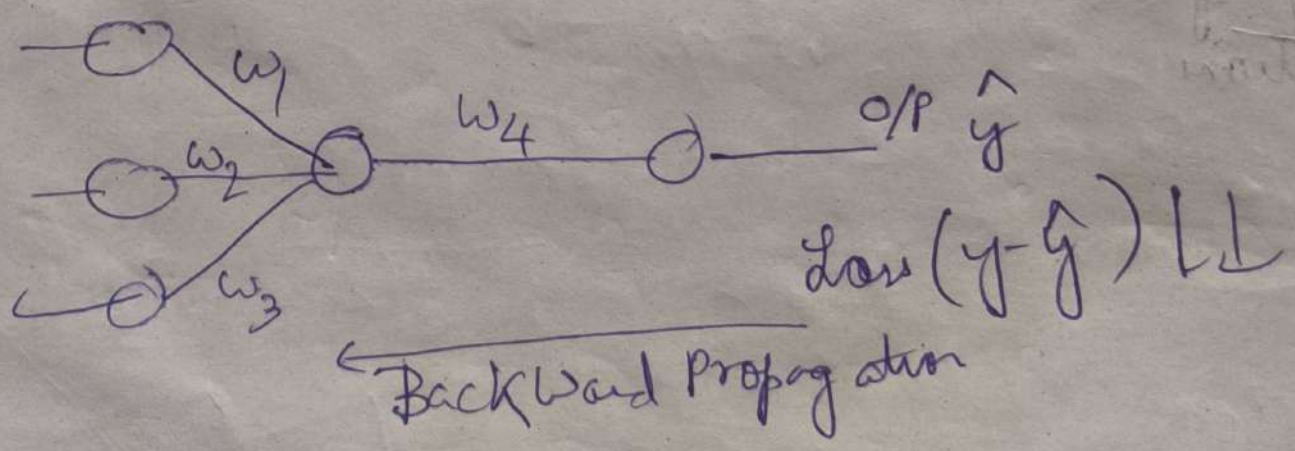
convert to 0 and 1 ($\geq 0.5 \rightarrow 1$, $< 0.5 \rightarrow 0$)

happens in forward propagation

② Back Propagation

① Weight Updation formula

② Chain Rule of Differentiation



Weight updation formula

$$W_{new} = W_{old} - \eta \frac{dh}{dW_{old}}$$

η → learning rate

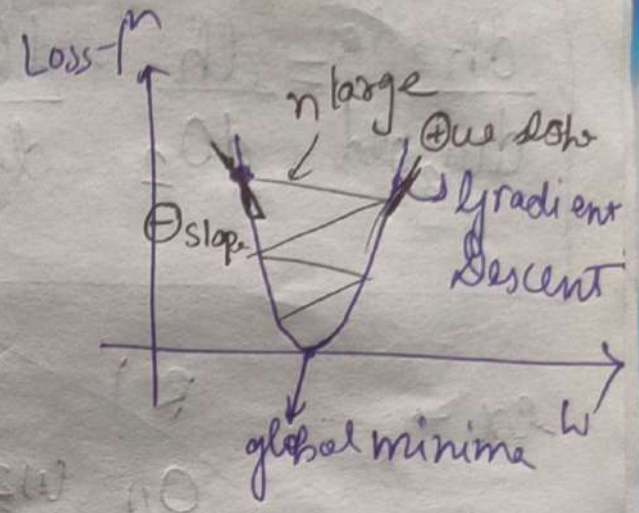
$\frac{dh}{dW_{old}}$ → slope

h → loss

η ≈ small number ≈ 0.01 or 0.001

if η is large, there will be jumps

In simple linear regression



In case of ⊖ve slope,

$W_{new} = W_{old} - \eta(\ominus ve)$

$= W_{old} + \eta(some \oplus ve)$

the $W_{new} > W_{old}$

In case of ⊕ve slope,

$W_{new} = W_{old} - \eta(\oplus ve)$

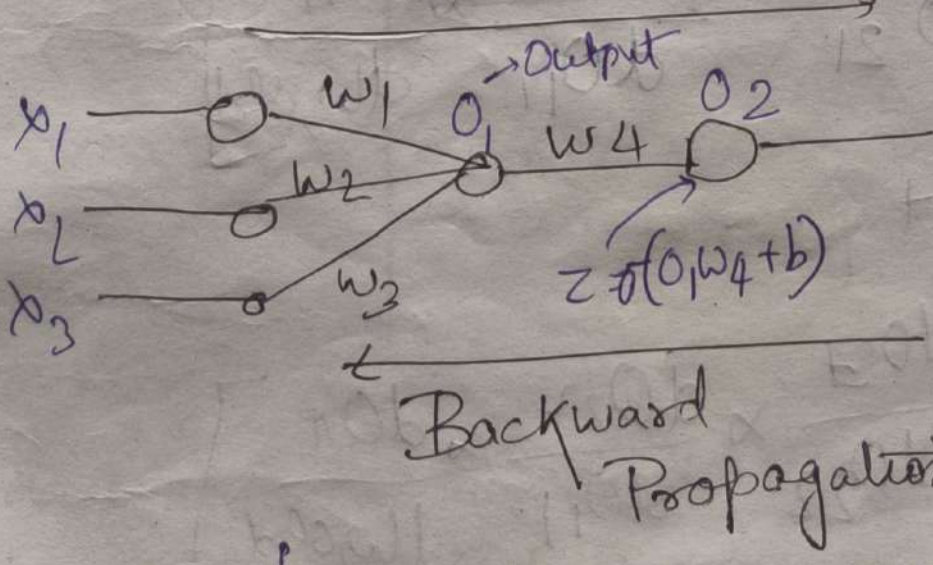
$= W_{old} - \eta(some \oplus ve)$

the $W_{new} < W_{old}$

$$b_{new} = b_{old} - \eta \frac{dh}{d b_{old}}$$

bias updation formula

② Chain rule of Differentiation



$W_{new} = W_{old} - \eta \frac{dh}{dW_{old}}$

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

$\frac{dh}{dW_{old}}$

$W_{4new} = W_{4old} - \eta \frac{dh}{dW_{4old}}$

$$\frac{dh}{dW_{old}} = \frac{dh}{dO_2} \cdot \frac{dO_2}{dW_{4old}}$$

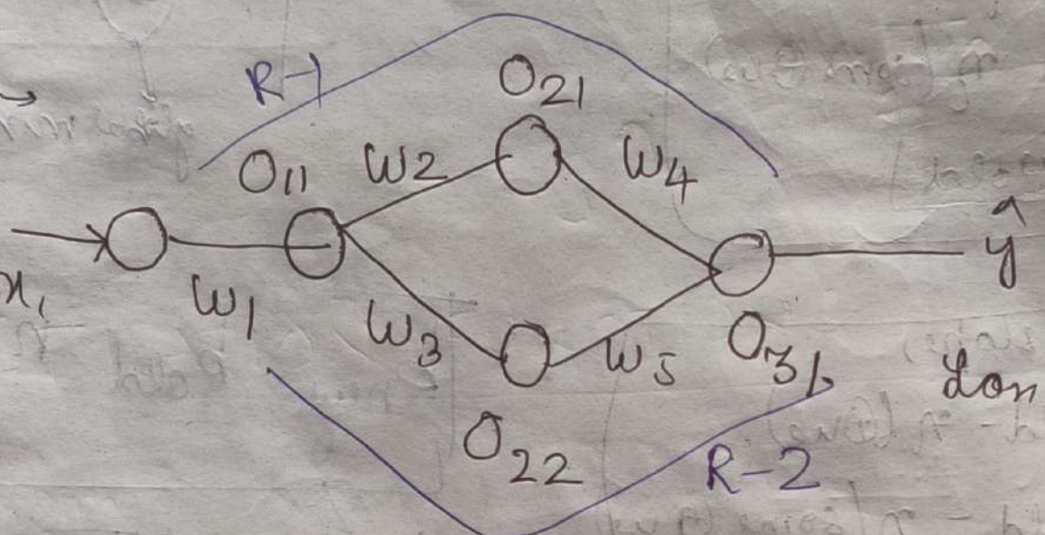
$$w_{1new} = w_{1old} - \eta \frac{dh}{dw_{1old}}$$

$$\frac{dh}{dw_{1,old}} = \frac{dh}{dO_2} \cdot \frac{dO_2}{dO_1} \cdot \frac{dO_1}{dw_{1old}}$$

$$\frac{dO_2}{dO_1} = \frac{dO_2}{dw_4} \times \frac{dw_4}{dO_1}$$

$$\frac{dh \to loss}{dw_{2old}} = \frac{dh}{dO_2} \cdot \frac{dO_2}{dO_1} \cdot \frac{dO_1}{dw_{2old}}$$

ex →



$$w_{inew} = w_{iold} - \eta \frac{dh}{dw_{i,old}}$$

$$\frac{dh}{dw_{1old}} = \left[\frac{dh}{dO_{31}} \times \frac{dO_{31}}{dO_{21}} \times \frac{dO_{21}}{dO_{11}} \times \frac{dO_{11}}{dw_{1,old}}\right]$$

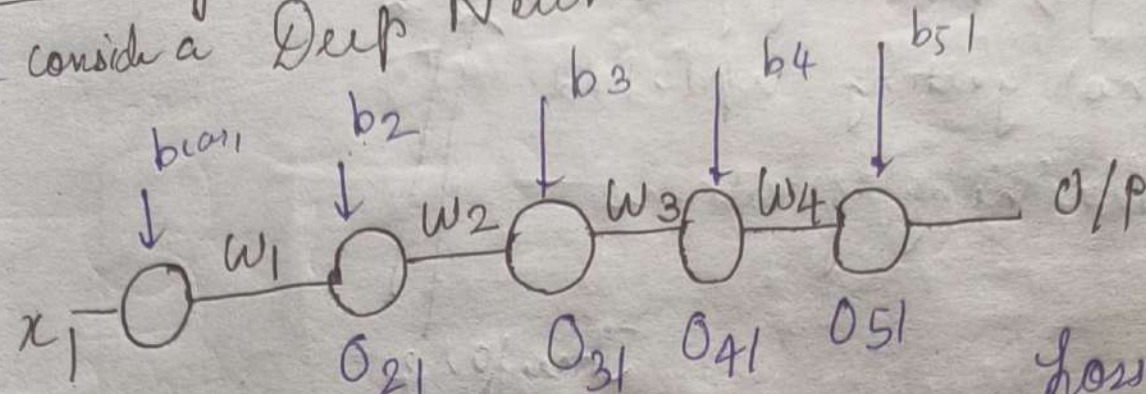
$$+$$

$$\left[\frac{dh}{dO_{31}} \times \frac{dO_{31}}{dO_{22}} \times \frac{dO_{22}}{dO_{11}} \times \frac{dO_{11}}{dw_{1,old}}\right]$$

Chain rule of derivatives

Vanishing Gradient Problem

We consider a Deep Neural Network,



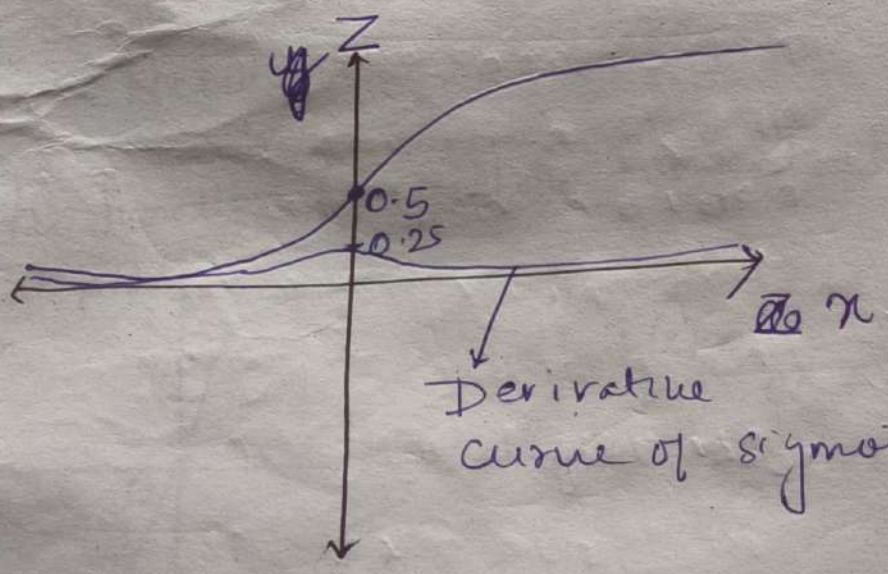
$$\text{Loss fn (mse)} = \frac{1}{2}(y-\hat{y})^2$$

Sigmoid Activation

$$w_{1\,new} = w_{1\,old} - \eta \frac{dh}{dw_{1\,old}}$$

$$\frac{dh}{dw_{1\,old}} = \frac{dh}{dO_{51}} \times \frac{dO_{51}}{dO_{41}} \times \frac{dO_{41}}{dO_{31}} \times \frac{dO_{31}}{dO_{21}} \times \frac{dO_{21}}{dw_{1\,old}}$$

Sigmoid Activation $\left(Z = \frac{1}{1+e^{-x}}\right)$ → 0 or 1; ≥ 0.5 → 1; < 0.5 → 0



Derivative Condn

$$0 \le \sigma(y) \le 0.25$$

So,

$$O_{51} = \sigma((O_4 \times w_4) + b$$

So, it is for all cases O_{51}, O_3, etc 0.25

So their derivative ranges from 0 to 0.25

for ex-

$$\frac{dh}{dw_{1\,old}} = \frac{dh}{dO_{51}} \times \frac{dO_{51}}{dO_{41}} \times \frac{dO_{41}}{dO_{31}} \times \frac{dO_{31}}{d21} \times \frac{dO_{21}}{dw_{1\,old}}$$

Let $= (0.25) \times 0.15 \times 0.10 \times 0.05 \times 0.12$ → it will decrease

$$= \text{Small value}$$

$$W_{new} = W_{old} - \eta(\text{small number})$$

So, after some time W_{new} will hardly change

No change in weights $\Leftarrow$ $W_{new} \approx W_{old}$ $\Rightarrow$ Vanishing Gradient Problem

So, weights are not getting updated.

So, this is the Vanishing Gradient

Problem:

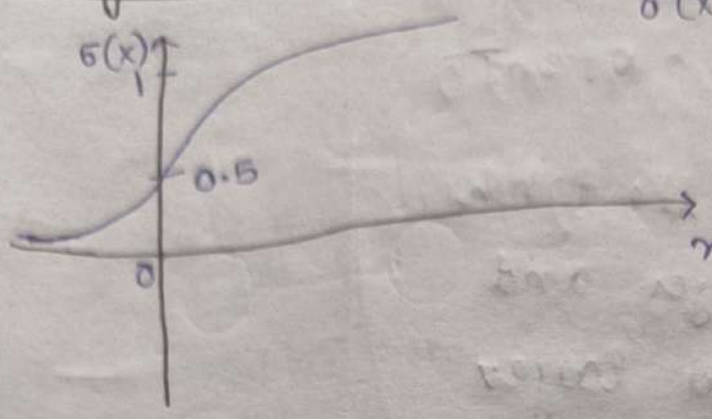
Therefore to solve Vanishing Gradient Problem, we use another Activation fn.

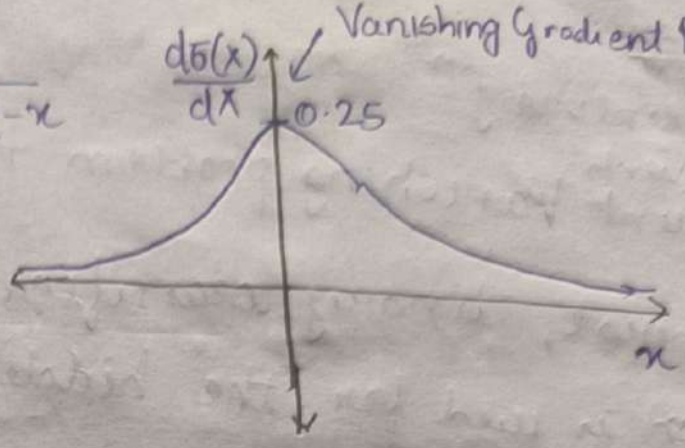
ACTIVATION FUNCTIONS

1. Sigmoid
2. Tanh
3. Relu
4. Leaky Relu
5. Pre Relu

① Sigmoid f^n

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

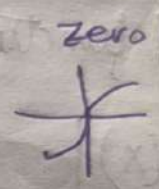




Advantages:

1) Smooth Gradient, preventing jumps in output values (& Easy Convergence of Weights)
(But the f^n is not centered 0, so weight updation efficiency will get reduce.)

zero centered curve



2) Output values bound between 0 and 1, normalizing the output of each neuron

3) Clear Predictions i.e. very close to 1 or 0

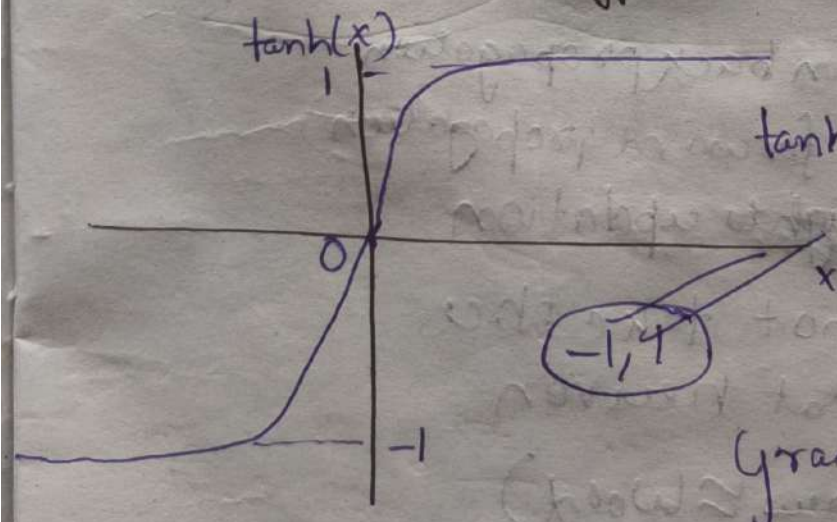
Disadvantages:

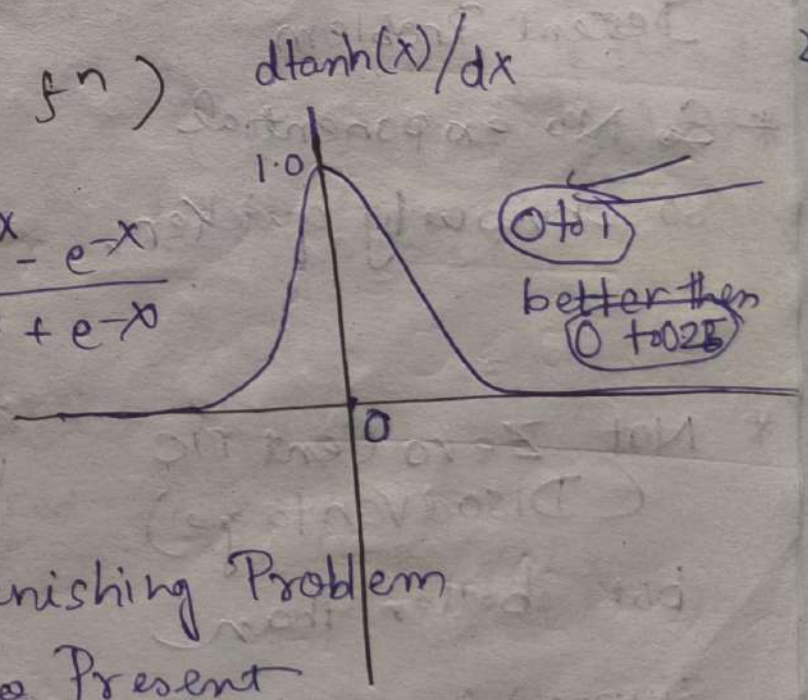
1) Prone to Gradient Vanishing
2) Function Output is not zero-centered
3) Power ops are relatively time consuming

So, due to Gradient Vanishing Problem, we cannot create Deep Neural Network.

② tanh function (hyperbolic tangent f^n)

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$





Grading Vanishing Problem is also Present if we create Deep Deep Neural Network.

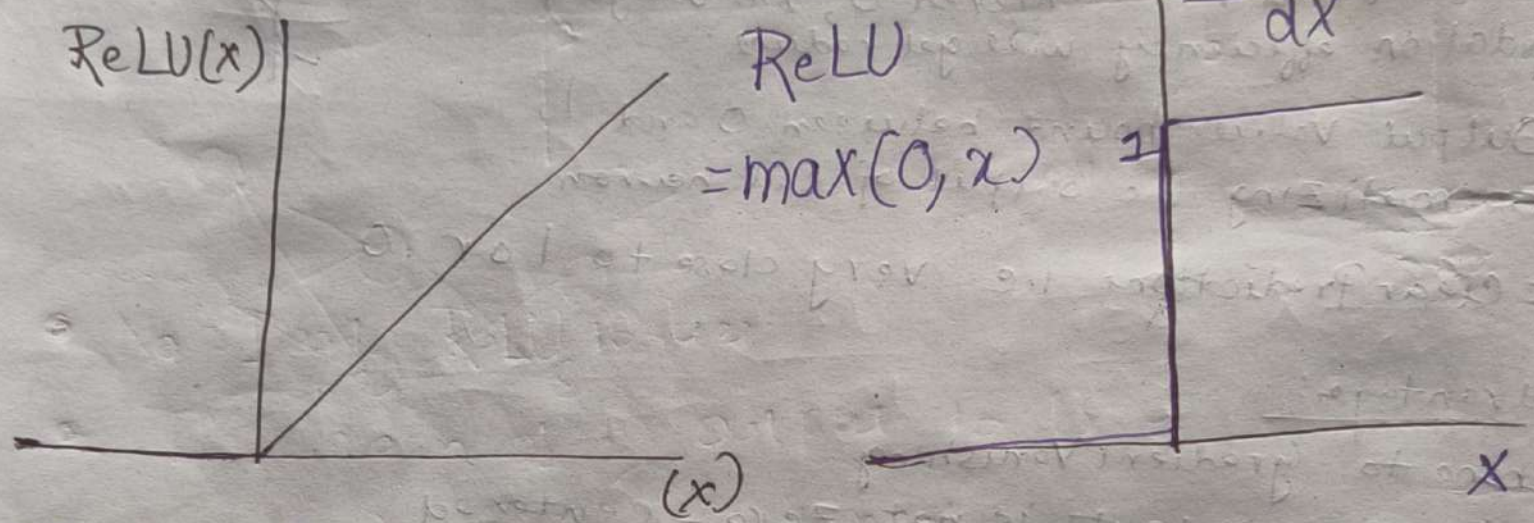
Advantage:

① Zero centered f^n

② Prevents Gradient Vanishing problem to an extent

In general, binary classification, the tanh function is used for the hidden layer and the sigmoid f^n is used for the output layer

3. ReLU function (Rectified linear unit)



Advantages:

* Exceptional Performance
* Solving the Gradient Descent Problem
* So No exponential so obviously quicker,
* Not Zero Centric (Disadvantage) but better than Sigmoid and tanh function.

Disadvantage:

× If one of the derivative becomes 0, as derivative in ReLU can be either 0 or 1 then the whole neuron becomes dead in back propagation (not in forward propagation) as weights updation will not take place for that neuron

$(W_{new} \leq W_{old})$

$\left(\frac{dh}{dw_{old}} = 0\right)$

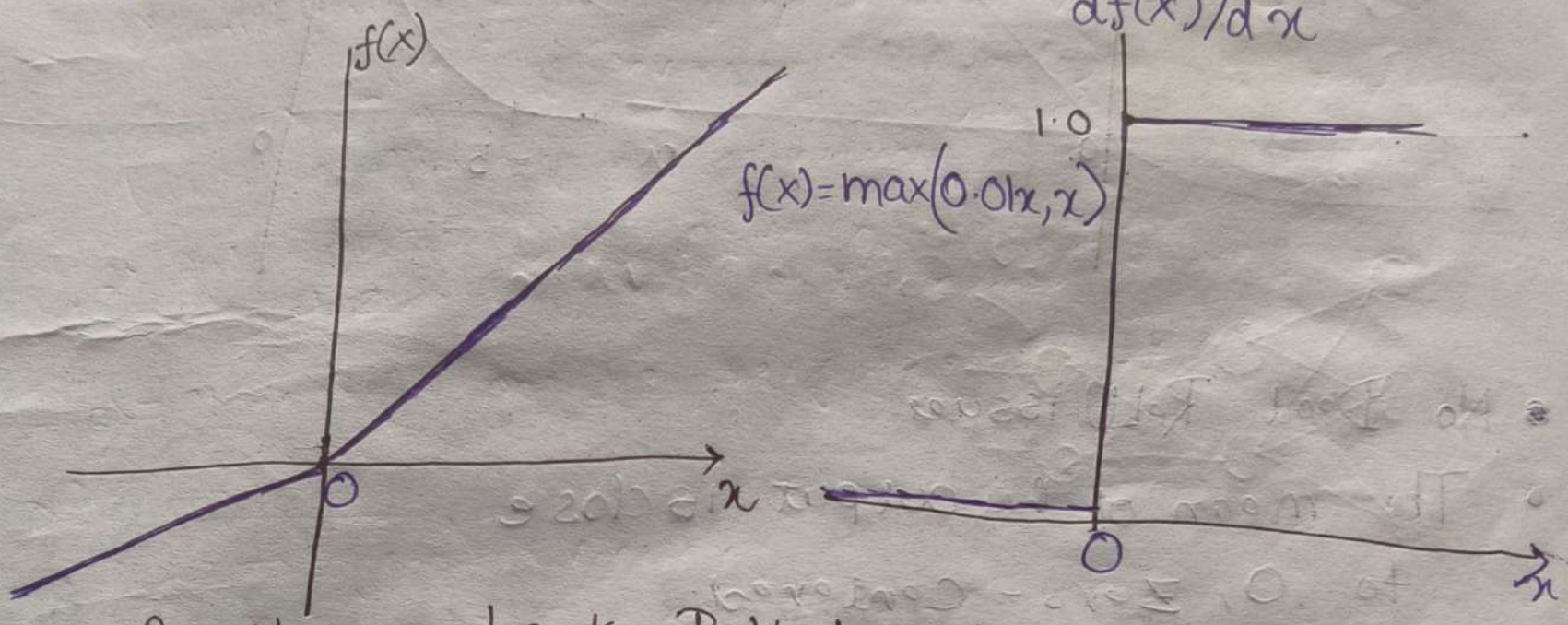
Why?

0 → Dead

* Once a negative number is entered ReLU is inactive and ReLU will die (Disadvantage).

4. Leaky ReLU ~~Fn~~ function

→ To solve the problem of dead Neuron / Relu Problem.



- In theory, Leaky ReLU has all the advantages of ReLU plus no
- But in actual ops, it has not been fully proved that leaky ReLU is always better than ReLU.

case of Dead ReLU

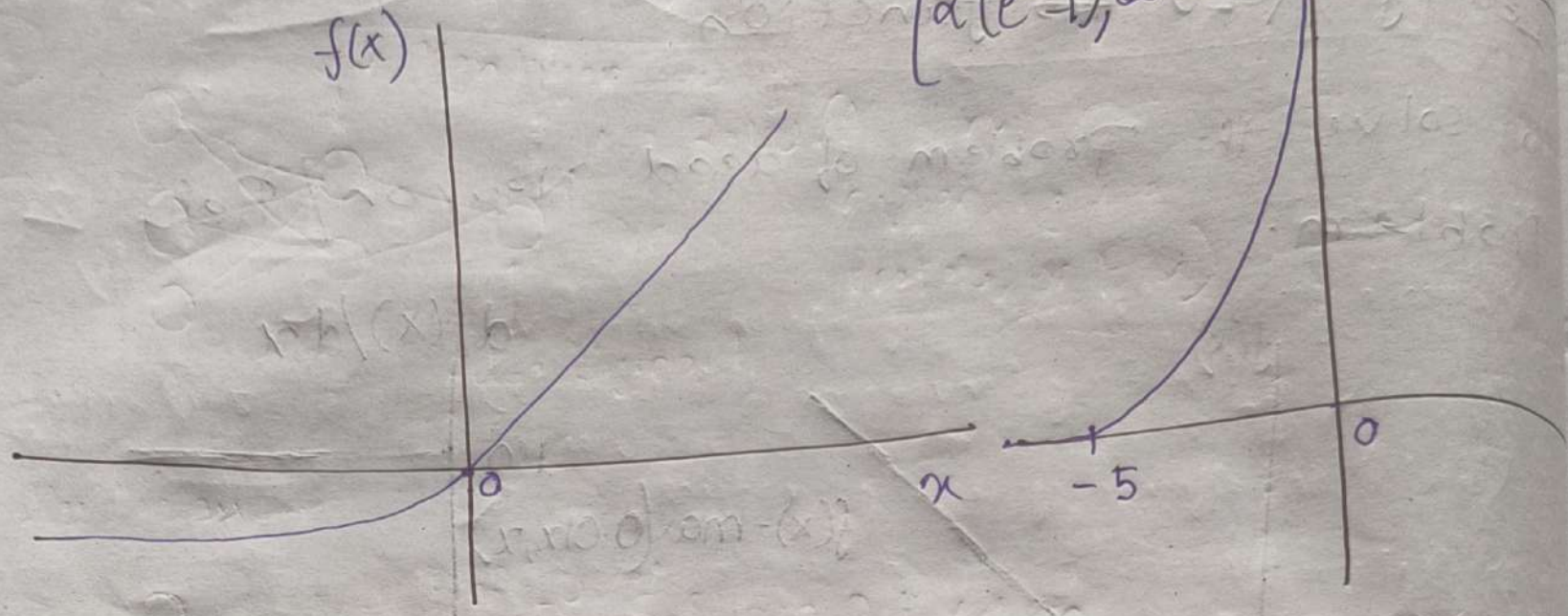
- Another intuitive idea is a parameter based method.

Parametric Relu: $f(x) = \max(\alpha x, x)$

which α can be learned from back propagation.

5. Exponential Leaky Units function (ELU)

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$



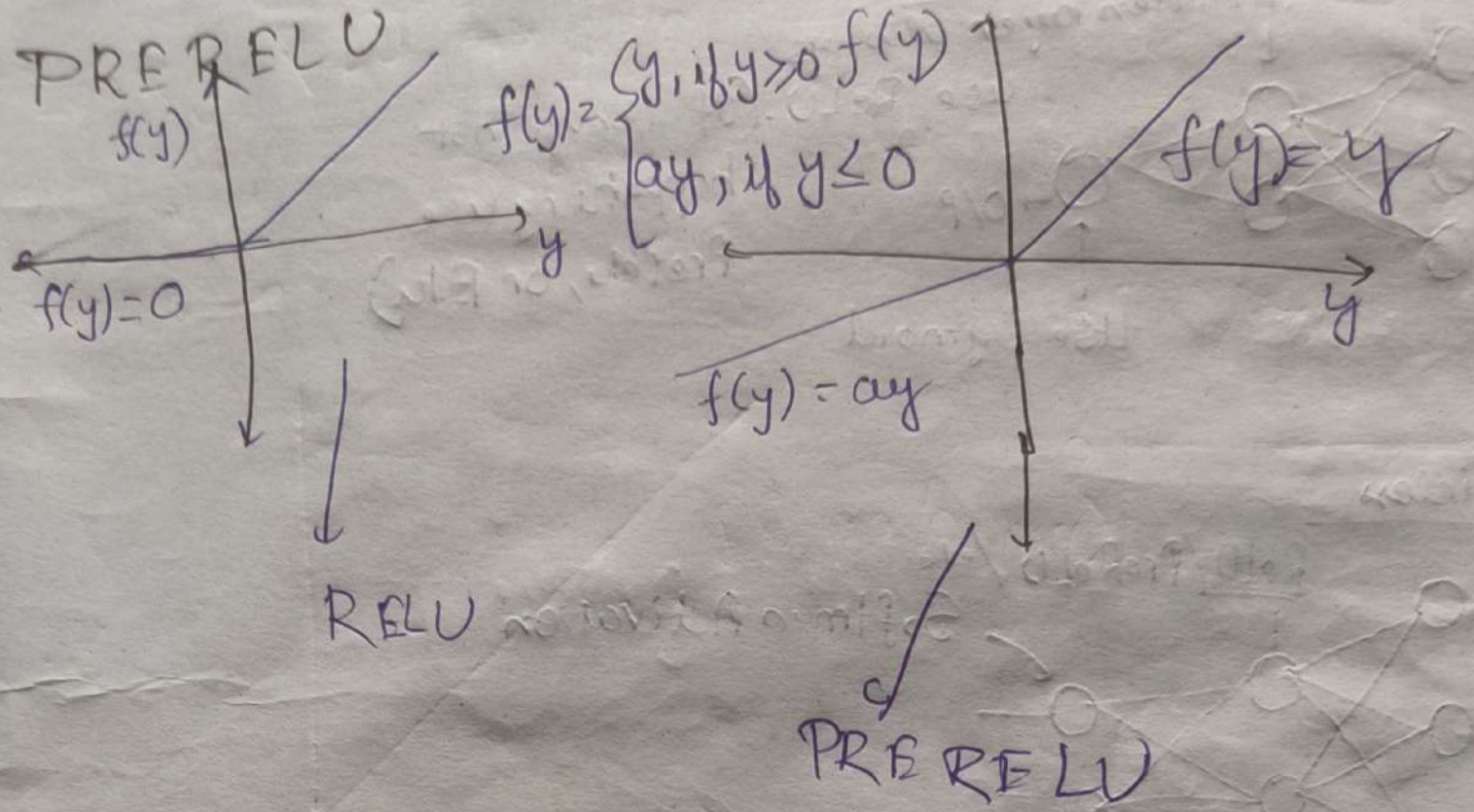
- No Dead ReLU issues
- The mean of the output is close to 0, zero-centered.

One small problem is that it is slightly more computationally intensive. Similar to Leaky ReLU, although theoretically better than than ReLU, there is currently no good evidence in practice that ELU is always better than ReLU.

6. Softmax

Will Discuss in Activation fn

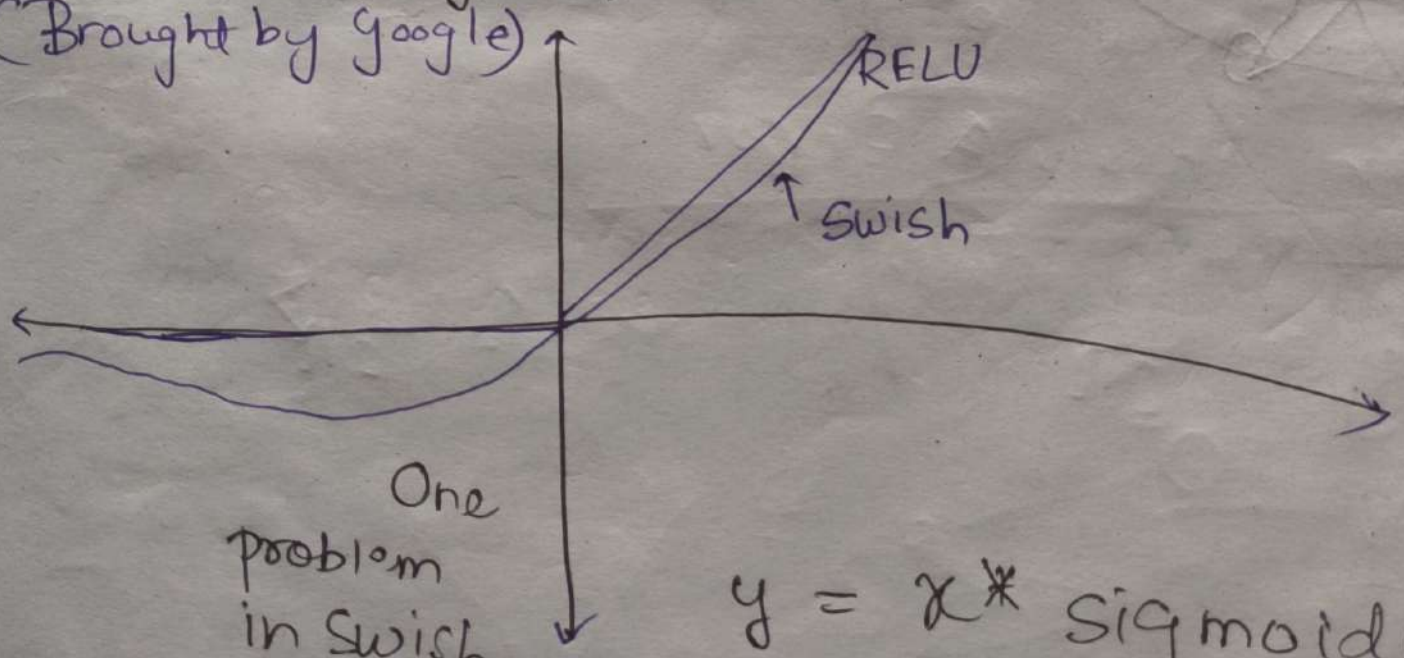
7. PRELU



- if $a=0$ it becomes ReLU
- if $a>0$, it become leaky Relu
- if a is a lernable parameter, it becomes

PRelu

8. Swish (A Self-Gated) Function.
(Brought by Google)

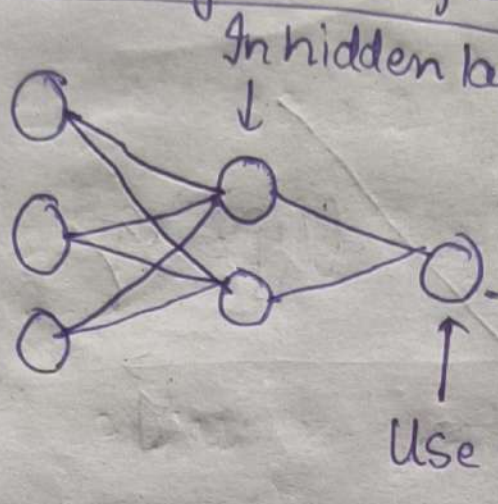


$y = x * \text{sigmoid}(x)$

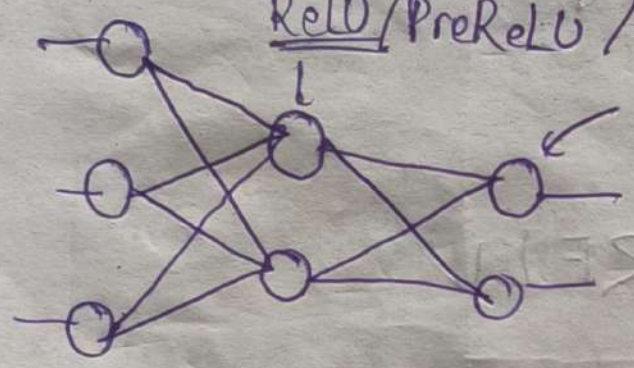
57

Technique which activation f^n should we use???

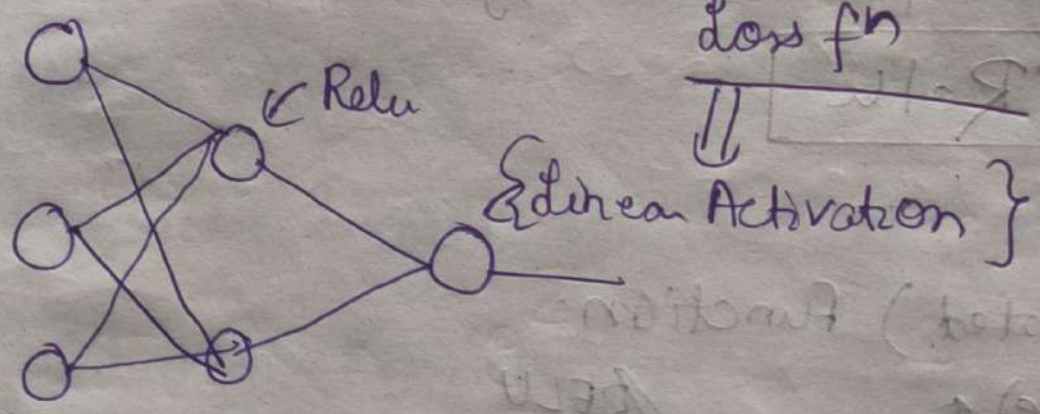
For Binary Classificcation



Multiclass



Regression



Loss functions

Deep learning (ANN) (Artificial Neural Network)

- Regression
- classification

Exp	Deg	Salary
10	Phd	–
–	–	–
–	–	–

Play	Study	Pass / Fail
10	2	F
4	3	F
5	5	maybe
2	7	Pass

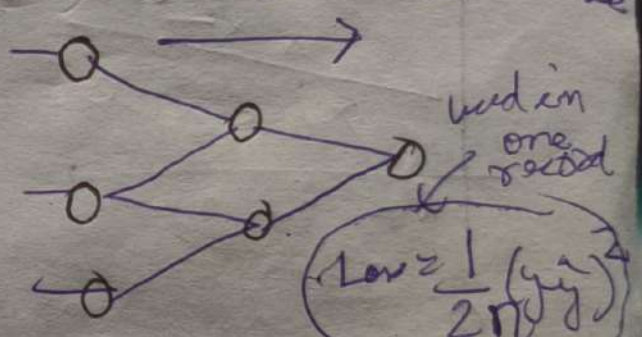
Regression Problem

① Loss fn in Regression

1) MSE (Mean Squared Error)
2) MAE (Mean Absolute Error)
3) Huber Loss

Loss fn and cost fn

Dataset = 100 rec



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

① Mean Squared Error (MSE)

Loss fn = $\frac{1}{2n}(y-\hat{y})^2$ cost fn = $\frac{1}{2n}\sum_{i=1}^{n}(y-\hat{y})^2$

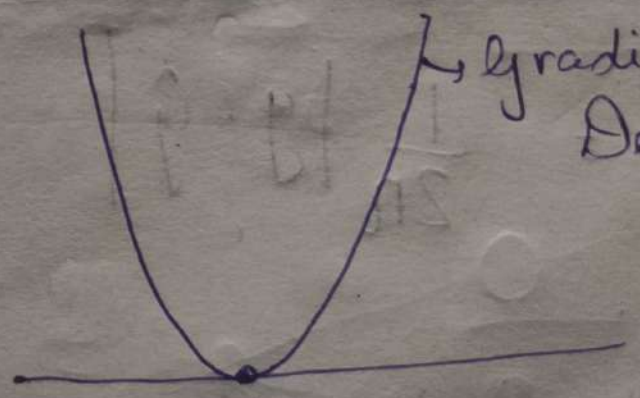
⇓

Quadratic Eqn

$(a-b)^2 = a^2 + b^2 + 2ab$

⇓

$ax^2 + bx + c$

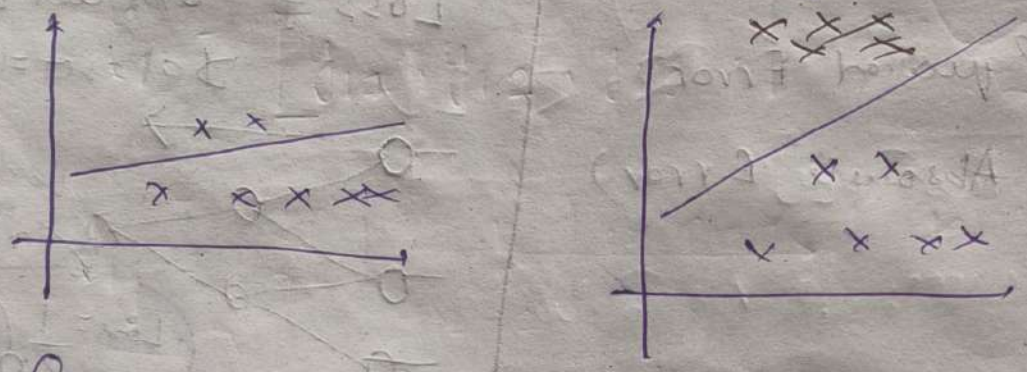


Advantages of using a Quadratic eqn Curve

1. Differentiable
2. It has only one local or global minima
3. It converges faster.

Disadvantages:

1) Not Robust to outliers.



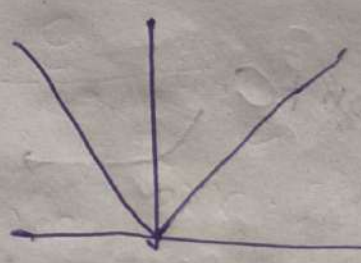
Since we are penalizing and squaring the error, the line has a major shift on introduction of outliers.

Mean Absolute Error

$$Loss = \frac{1}{2n}|y-\hat{y}| \qquad Cost\ fn = \frac{1}{2n}\sum_{i=1}^{n}|y-\hat{y}|$$

Adv :—

1. Robust to outliers. (Minor shift)



⇒ To update weight / find derivate we have to find subgradient.

Dis :- Time Consuming ↑↑

③ Huber Loss :-

Combination of MSE or MAE.

① MSE ② MAE

$$\text{Loss} = \begin{cases} \frac{1}{2n}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$$

← Hyperparameter

i.e. when outliers are not there

(i.e when outliers are present)

Classification loss functions

Cross entropy
- Binary CROSS entropy
  - ↳ Binary classification
- Categorical ~~clas~~ CROSS entropy
  - ↳ Multiclass classification

Binary CROSS ENTROPY

$$\text{loss} = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$ ⇒ Logistic Regression

$$\text{Loss} = \begin{cases} -\log(1-\hat{y}) & \text{if } y=0 \\ -\log(\hat{y}) & \text{if } y=1 \end{cases}$$ Binary classification

$$\hat{y} = \frac{1}{1+e^{-x}}$$

i → row j = column

② Categorical class entropy { Multiclass Classification Problem }

One hot Encoding

f_1	f_2	f_3	O/P → Good	Bad	Neutral
2	3	4	Good	1	0	0
5	6	7	Bad	0	1	0
8	9	10	Neutral	0	0	1

C ← no. of categories

$$Loss(X_i, y_i) = -\sum_{j=1}^{C} y_{ij} * \ln(\hat{y}_{ij})$$

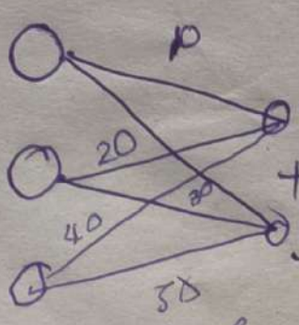
$$y_i = [y_{i1}, y_{i2}, y_{i3} \dots y_{ic}]$$

$$y_{ij} = \begin{cases} 1 & \text{if element is in class} \\ 0 & \text{otherwise} \end{cases}$$

$\hat{y}_{ij}$ = Softmax Activation layer applied for the output of multiclass classification

$$\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \Bigg\} \rightarrow \text{Softmax activation}$$

ex



probability 0.4, 0.5

$$\text{Softmax} = \frac{e^{10}}{(e^{10}+e^{20}+e^{40})} + \frac{e^{20}}{(e^{10}+e^{20}+e^{40})} + \dots$$

$= 1$

Both prob. summation = 1

and the class with higher probability that will come up.

Conclusion:

ReLU, Softmax ⇒ Multi-Class classification → Categorical Cross Entropy

ReLU, Sigmoid ⇒ Binary → Binary cross entropy

Linear Regression

ReLU, Linear Activation → MSE,
MAE,
Huber Loss

OPTIMIZERS AND ANN IMPLEMENTATION

Agenda :-

1) Optimizizers
2) Gradient Descent
3) SGD (Stochastic Gradient Descent)
4) MiniBatch SGD
5) SGD with momentum
6) Adagrad
7) RMSPROP
8) Adam optimizers

9) Batch, Epochs, Iterations

↓

ANN

OPTIMISERS

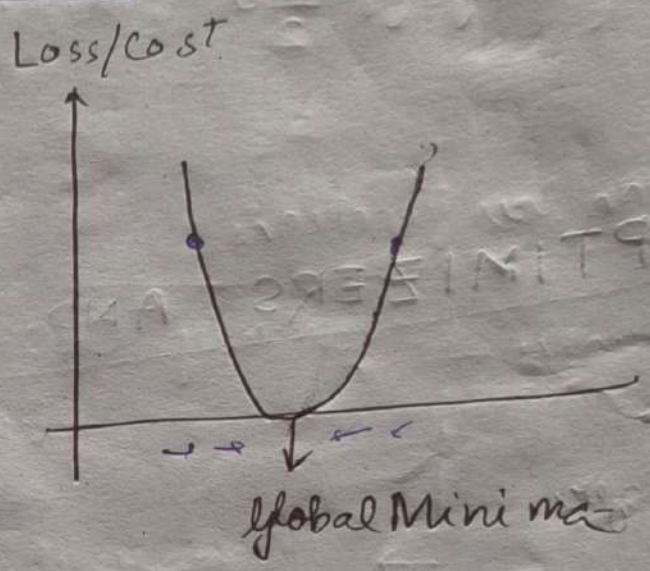
(1) Gradient Descent

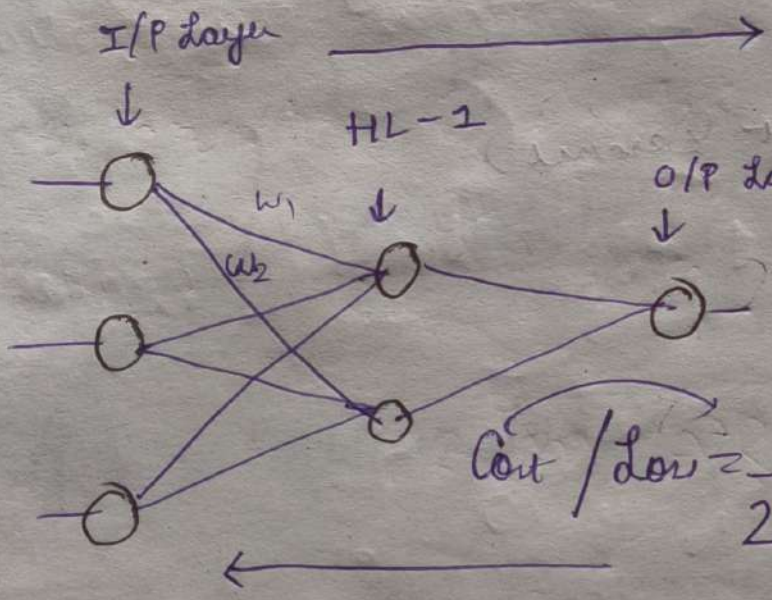
Weight updation formula

$$W_{new} = W_{old} - \eta \frac{d(L)}{dW_{old}}$$

(old)

Learning rate





MSE

$$\text{Cost/Loss} = \frac{1}{2n}\sum_{i=1}^{n}(y-\hat{y})^2$$

Epoch ⟶ ⟵ } 1 Epoch (1 forward propagation, 1 backward propagation)

Disadvantages

1) Resource contensive {Huge RAM}
{as million record in 1 epoch}

② Stochastic Gradient Descent

{We pass 1 record in every epoch}

① RAM ↓↓
but
② Disadvantage

① Convergence will be very slow

② Time Complexity will also be high.

Epoch-1

1st record → ŷ ← Loss ← Update weights } → Iteration

2nd record → ← } Iteration 2
← Update weight

③ Mini Batch SGD

① Resource intensive

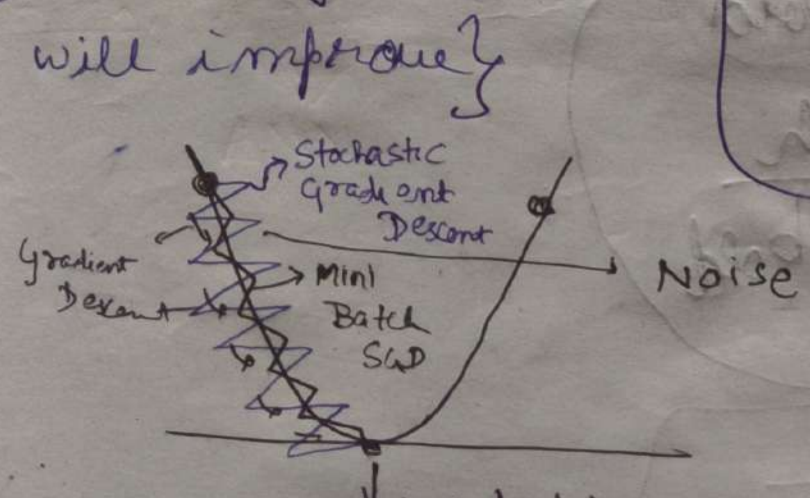
② Convergence will be better

③ Time {complexity will improve}

iteration reduced to 1000

Let 100000 data are there, batch size = 1000

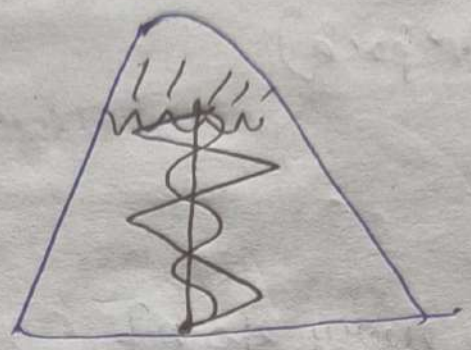
Epoch = 1
1000 → ← } Iteration 1
⋮
} Iteration 1000



Noise

Gradient Descent < Mini Batch SGD < SGD

if we have noise, it will definitely take more time'



How to remove the noise?

We use momentum to remove the noise

④ SGD with momentum

(helps to smoothen the energy)

{Exponential Weighted Average}

⇓

Time Series

⇓

ARIMA, AR MA, ...

$$w_{new} = w_{old} - \eta \frac{dh}{dw_{old}}$$

$$b_{new} = b_{old} - \eta \frac{dh}{db_{old}}$$

$$w_t = w_{t-1} - \eta \frac{dh}{dw_{t-1}}$$

Exponential Weighted Average

t_1 t_2 t_3 t_4 ... t_n ← time

a_1 a_2 a_3 a_4 ... a_n ← value

{Forecasting}

$$V_{t_1} = a_1$$

$$V_{t_2} = \beta * V_{t_1} + (1-\beta) * a_2$$

β — hyperparameter

if $\beta = 0$ to 1

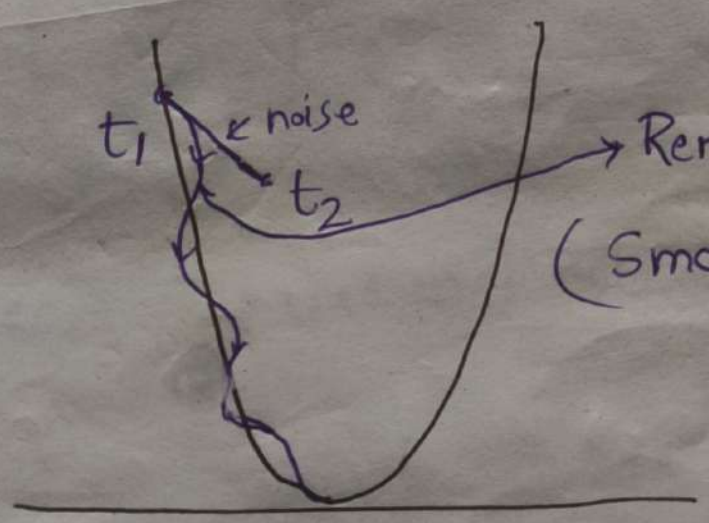
if $\beta = 0.95$

$$V_{t_2} = (0.95)\, V_{t_1} + (0.05)\, a_2$$

a_2 ↓ a_1 smoothening will happen

more control to a_1 (previous point) than current point

Les imp. to a_2



$$V_{t_3} = \beta \times V_{t_2} + (1-\beta) a_3$$

Where to apply this?

We apply this in the derivative of loss to derivative of weight

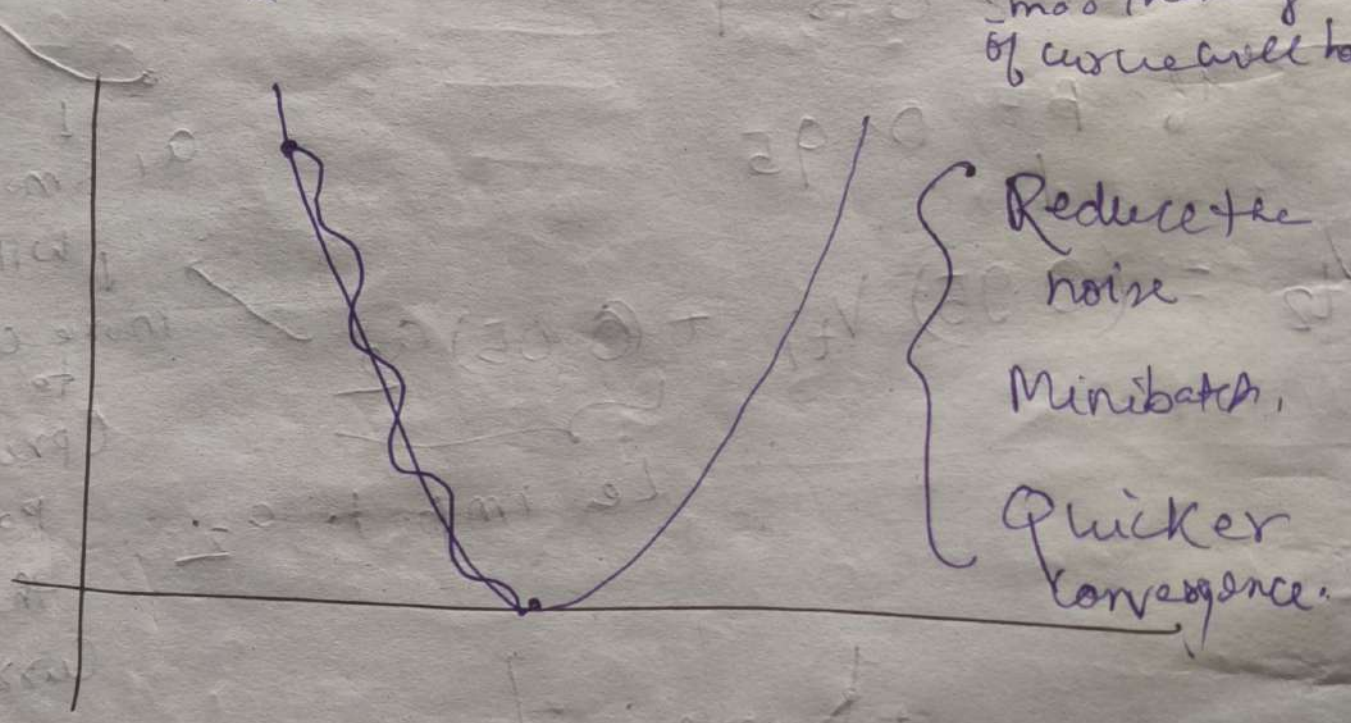
Exponential Weighted average

$$W_t = W_{t-1} - \eta V_{dW_t}$$

$$V_{dW_t} = \beta \times V_{dW_{t-1}} + (1-\beta) \times \frac{dL}{dW_t}$$

$V_d \rightarrow$ value of derivative

Smoothening of curve will happen



Recap

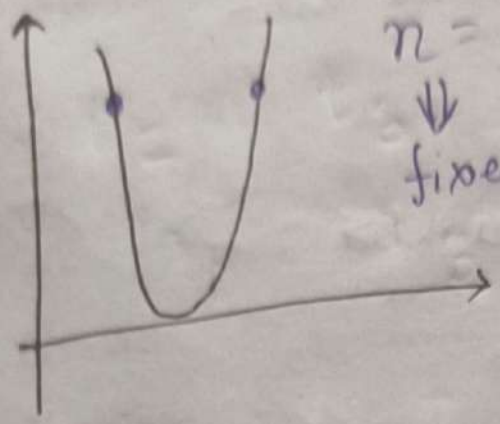
① Gradient Descent

② SGD

③ Minibatch SGD

④ SGD with momentum.

⑤ Adagrad → Adaptive Gradient Descent



η = learning rate

⇓

fixed

$$w_t = w_{t-1} - \frac{\eta \, dL}{dw_{t-1}}$$

η = fixed ⟶ adaptive

$$\boxed{w_t = w_{t-1} - \eta' \frac{dh}{dw_{t-1}}}$$

η' = $\frac{\eta \leftarrow \text{Learing rate}}{\sqrt{\alpha_t + \epsilon}}$ small number

so denominator $\neq 0$

$t \leftarrow$ current time stamp

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial h}{\partial w_t} \right)^2$$

η should decrease as we are reaching global minima.

$t=1$	$t=2$	$t=3$
$\eta=0.01$	$\eta=0.005$	$\eta=0.002$

∴ Learning rate decreased.

Since, $\alpha_t + \epsilon$ is large therefore ~~$\eta' \approx 0$~~ $\eta' \approx 0$, therefore very less change in updation. So, to avoid this we use:-

Adadelta and RMS Prop

$$\eta' = \frac{\eta}{\sqrt{S_{dw} + \epsilon}}$$

We will apply exponential weighted average

$$S_{dw} = 0$$

$$S_{dw_t} = \beta S_{dw_{t-1}} + (1-\beta)\left(\frac{dh}{dw_t}\right)^2$$

Slow decreasing of η'.

if $\beta = 0.95$

$$S_{dw_t} = (0.95)\, S_{dw_{t-1}} + (0.05)\left(\frac{dh}{dw_{t-1}}\right)^2$$

Adam Optimiser (Best Optimizer)

Momentum + RMSProp

(Adaptive Learning Rate)

Initially

$V_{dw} = 0 \quad V_{db} = 0 \quad S_{dw} = 0 \quad S_b = 0$

$$w_t = w_{t-1} - \eta' V_{dw}$$

$$b_t = w_{bt-1} - \eta' V_{db}$$

$$\eta' = \frac{\eta}{\sqrt{S_{dw} + \epsilon}}$$

$$V_{db_t} = \beta * V_{db_{t-1}} + (1-\beta)\frac{dh}{dB_{t-1}}$$

$$V_{dw} = \beta * V_{dw_{t-1}} + (1-\beta)\frac{dh}{dw_{t-1}}$$

Helps is :-

1. Smoothening
2. Learning Rate becomes adaptive
3. ~~Adap~~

Practical ANN implementation

Day-4- Deep Learning

1. ANN Practical Implementation
2. Early Stopping
3. Black Box Model Vs White Box Model
4. CNN Introduction

GOOGLE COLLAB ANN IMPLEMENTATION

1) !pip install tensorflow-gpu
import tensorflow as tf
print(tf.__version__)
↳ 2.8.0 (should be >2.0 for Keras integration)

import some basic libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Churn_Modelling.csv')
dataset.head()

divide the dataset into independent and dependent features
all ROW, Column
X = dataset.iloc[:, 3:13]
Y = dataset.iloc[:, 13]

Feature Enginering
geography = pd.get_dummies(X['Geography'], drop_first = True)
↳ One hot encoding

gender = pd.get_dummies(X['Gender'], drop_first = True)

concantenate these variables with dataframe
X = X.drop(['Geography', 'Gender'], axis = 1)
X = pd.concat([X, geography, gender], axis = 1)

Splitting the dataset into Training set and Test Set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split
                                   (X, y, test_size
                                    = 0.2)
                                   random_state = 0)
```

Interview:- ① For which all algorithms feature scaling is required?

~~ANN~~ ✓ KMeans ✓

ANN? ✓

LR? ✓

LogisticRegression? ✓

Decision? ✗

RF? ✗

XG? ✗

KNN? ✓

Feature scaling is required for:-

① ~~Decision~~ Distance based algo ✓

② Gradient Descent algo ✓

{do to make values smaller, do to make calculation faster or convg faster}

Why fit_transform is used for only train data and not test data?

↳ to avoid data leakage

feature Scaling

rotate around mean value

↓ ↙ Z-score

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Part 2 : Now let's create the ANN

Tensorflow ⇒ Google Deepmind

Pytorch ⇒ facebook

Tensorflow < 2.0

Tensorflow — wrapper — Keras

Tensorflow ⩾ 2.0

Tensorflow + Keras (wrapper)

~~Tensorflow~~ Keras is inside Tensorflow version itself

from tensorflow.~~model~~ keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LeakyReLU, PReLU, ELU, ReLU

from tensorflow.keras.layers import Dropout

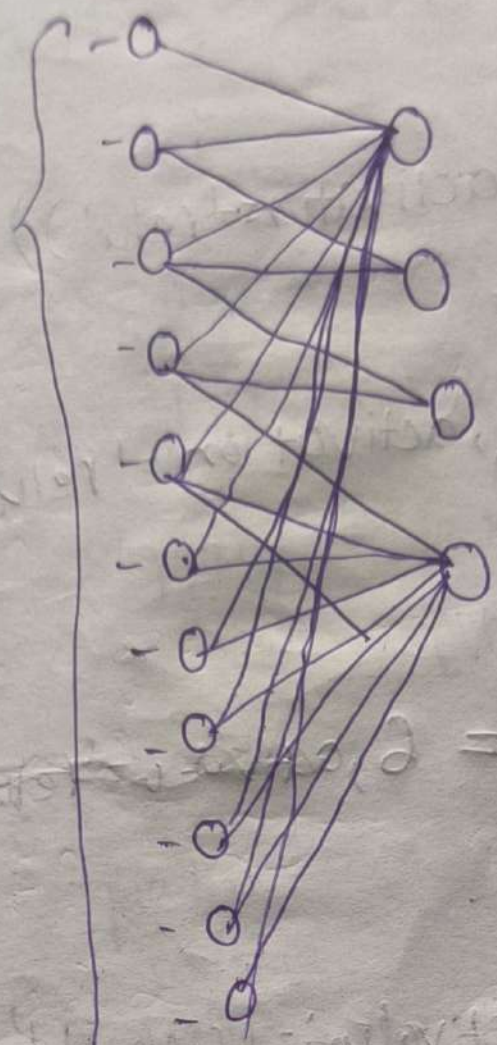
① Sequential

② Dense
(we need to create circle/neurons)

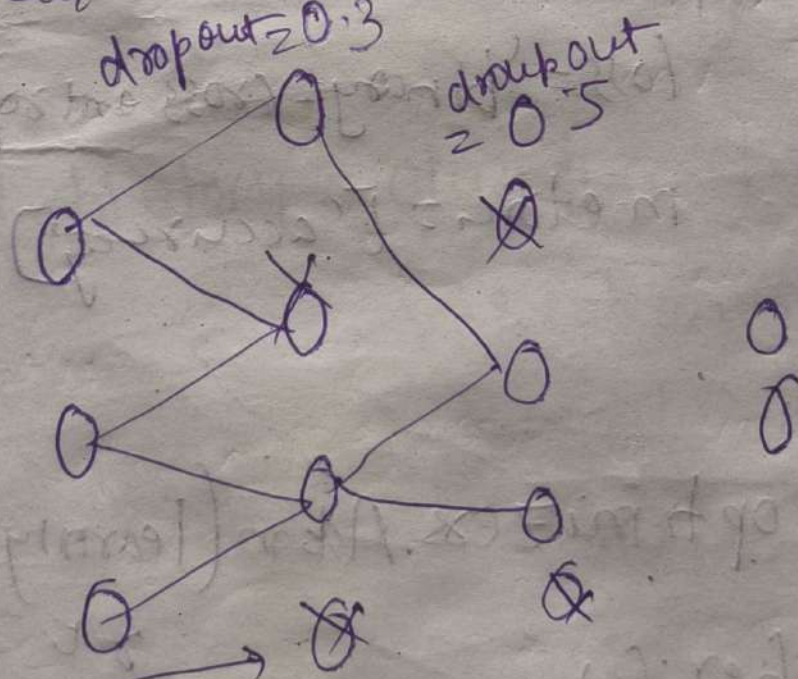
③ Activation

all these neuron network in a block, we sequential

Sequential
↓
can do forward and backward propagation



④ Drop out



This entire neural network sometimes lead to overfitting

To remove overfitting we use,
Drop out layer, → Regularization

if drop out layer is 0.3, then 30% of neuron will get deactivated.

{ L1 Norm
L2 Norm }

Let's initialize the ANN

classifier = Sequential()

Adding the input layer

classifier.add(Dense(units=11, activation='relu'))

adding the first hidden layer

classifier.add(Dense(units=11, activation='relu'))
classifier.add(Dropout(0.2))

adding the second hidden layer

classifier.add(Dense(units = 6, activation='relu'))
clas.add(Dropout(0.3))

adding the out put layer

classifier.add(Dense(1, activation='sigmoid'))

Classifier.compile(optimizer = opt/'adam',
loss = 'binary_crossentropy',
metrics = ['accuracy'])

import tensorflow

opt = tensorflow.keras.optimizers.Adam(learning_rate = 0.01)

model_history = classifier.fit
(X_train, y_train,
validation_split = 0.33,
batch_size = 10, epochs = 100)

model.

42/1000

Early Stopping

```
import tensorflow as tf
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    min_delta=0.0001,
    patience=20,
    verbose=1,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
)
model_history = classifier.fit(X_train, y_train,
    validation_split=0.33, batch_size=10, epoch=1000,
    callbacks=early_stopping)
```

play with values →

```
model_history.history.keys()
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

# Summarise history for accuracy and for val_loss also we can plot
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
ypred = classifier.predict(X_test)
y_pred = (y_pred >= 0.5)

## Make confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

## Calculate the accuracy
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
```

get the weights
classifier.get_weights()

Black Box Model Vs White Box BoModel

Random Forest :- Black box
Decision Tree → White Box Model
Linear Regression → White Box Model
ANN → Black Box Model
xgboost → Black Box Model

Explainable AI { }

Understanding CNN and Implementation

ANN → Theoretical, Practical

CNN → Convolutation Neural Network

V/
Image, Video Frame

Agenda

1. CNN Vs Human Brain
2. Convolution operation
   - Convolution
   - Padding
   - Stride
   - Filters (Kernels)
3. Max Pooling
4. Flattening
5. Practical Implementation CNN

CNN vs Human Brain

CAT img
DOG CAR
BICYCLE
HUMAN

CERERAL CORTEX (Backside of Brain)
↓
VISUAL CORTEX
(responsible for seeing obj in img/vid
↓
Different Task

Layers {
- V_1 → Moving objects
- V_2 → Animals (CAT)
- V_3 → Map the env
- V_4, V_5, V_6
- V_7
}

↓
Full O/P

CNN

Convolution

Images =

Black and white images

0 – 255

Black and white OR RGB
↓
1 channel

0 → Black
255 → White color

5×5 pixel image

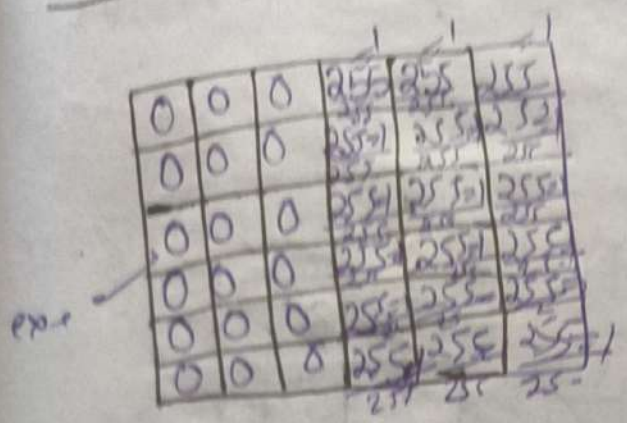
RGB image

Blue channel
Green channel
Red Channel

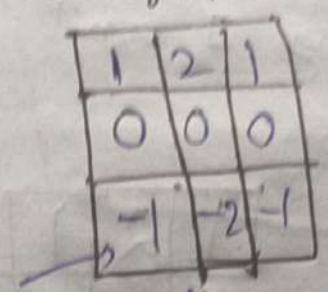
5×5×3

0 – 255
↓
wrt each and every channel

1st Step of CNN:
Convolution

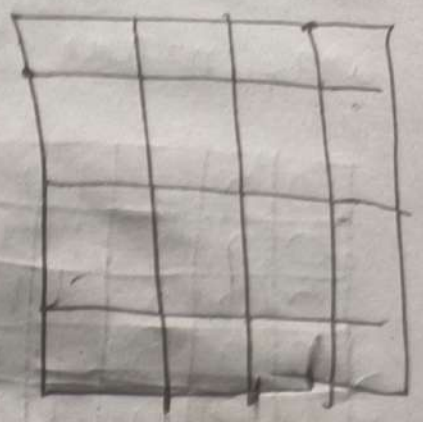


3x3 filter or Kernel



{Horizontal Edge filter}

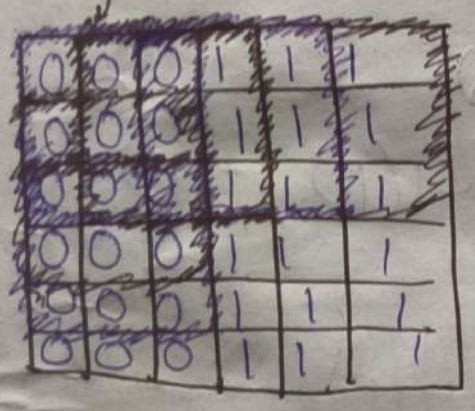
⇒



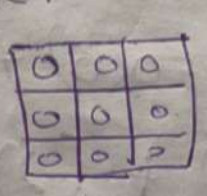
4x4 output

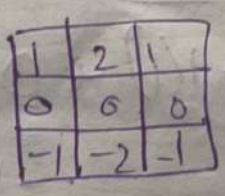
Step-1
We bring the values ranging from 0-255 to 0 to 1.
This process is known as min-max scaling.

Step-2
We place the filter on top of input layer.
Stride → How many posn to skip
If Stride = 1



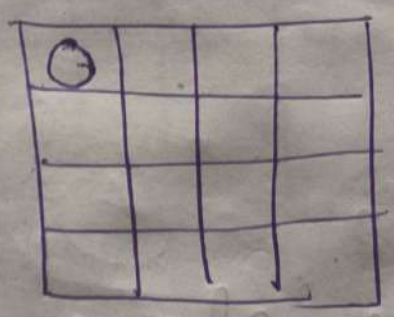
let's consider for first opn



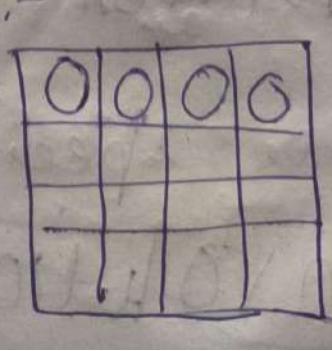


$1(0)+2(0)+1(0)+0(0)+0(0)+(0)$
$+(0)(-1)+(0)(-2)(-1)(0)$
$= 0$

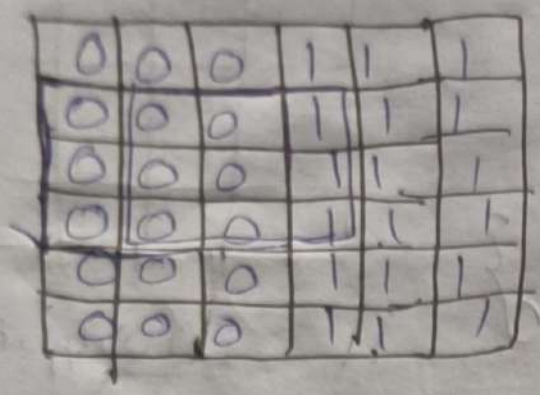
So output's first ele = 0

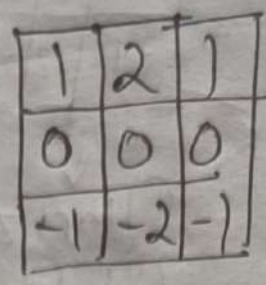


Similarly, we continue, and the first row of output becomes:-

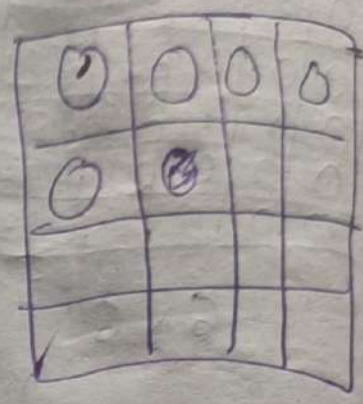


Since, stride = 1 we move to right by 1.
And after completion of 1st row in the output layer, we move down by 1 unit.





=



$$1(0)+2(0)+(1)(0)+0(0)+0(0)+0(0)$$
$$+(-1)(0)+(-2)(0)+(-1)(0)$$
$$= 0$$

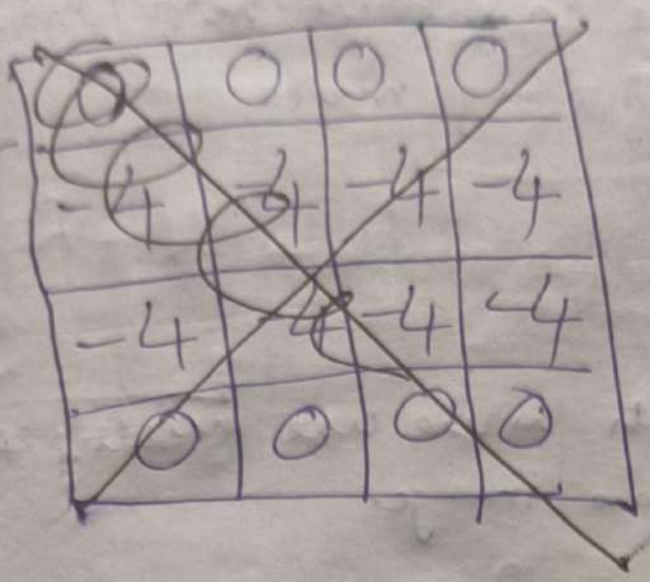
In next step

$$1(0)+2(0)+1(1)+0(0)+0(0)+1(0)$$
$$-1(0)+(-2)(0)-1(1) = 0$$

H.W → Calculate the above by doing convolution process.

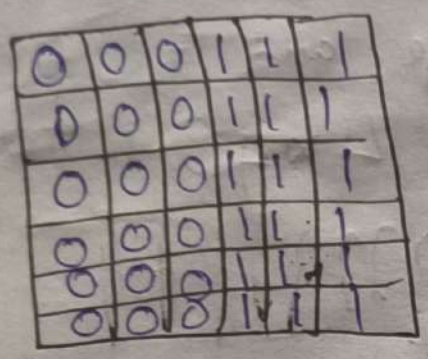
These operations are specifically known as "convolution".

~~Let's say we get some output as~~



Image

Q→



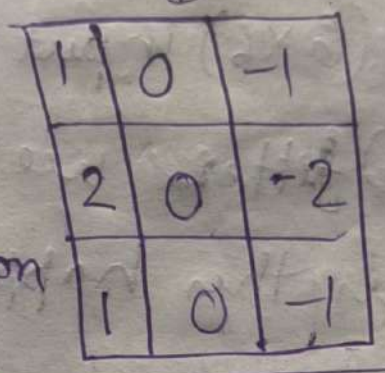
We had done min max scaling.

6x6

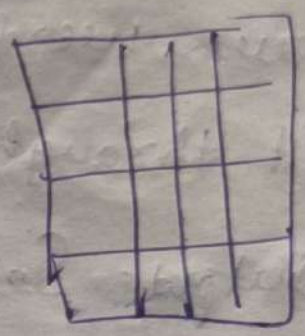
*

Convolution

Already defined



=?



helps to identify the vertical edge.

Verticaledge filter ~~detect~~ or Kernel.

3x3

4x4

Output

4x4

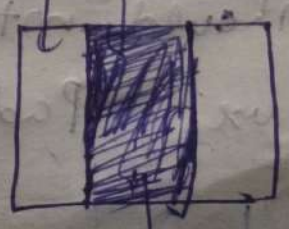
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0
0	-4	-4	0

Reverting back from feature scaling

255	0	0	255
255	0	0	255
255	0	0	255
255	0	0	255

255 → white color

0 → black color



* These filters like horizontal edge filter, vertical edge filter, round edge filter ~~for~~ helps to identify or extract various info. So, we can have horizontal, vertical or round edge and many more filters.

* We will get multiple outputs as we use multiple different filters for the same image.

Whenever we pass a (6×6) input image and pass it through (3×3) filter we get 4×4 output. So, how is this happening??

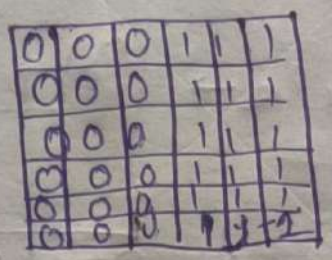
if 'n' is the size of the image.
and
if 'f' is the size of the filter
then we get
'n-f+1' as output size.

∴ if n=6 f=3
∴ n-f+1 = 6-3+1 = (4)

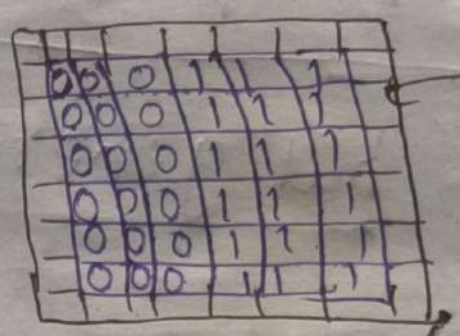
Here the image size decrease from 6×6 to 4×4, which should not happen, (we are losing some info) therefore we use 'Padding'.

Padding is nothing but a layer on top of image which protects ~~for~~ the image.

6 × 6



We apply padding,



Here padding = 1 (as padding layer = 1)

Padding layer

~~Image~~ Image is 6 × 6 only but (8 × 8) is total.

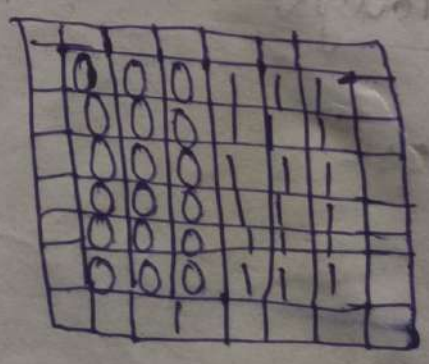
Padding are of different types according to the values ~~f~~ filled in the cells of padding.

① Zero Padding
↳ fillit with 0.

② Fillit with the nearest values.

So after padding

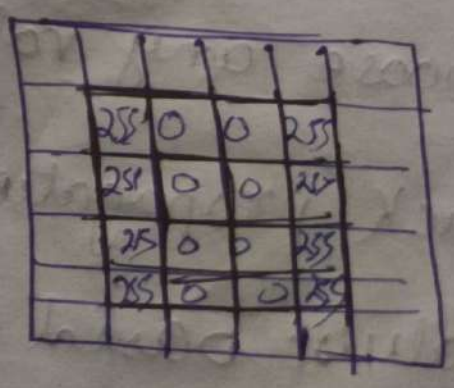
$$8 - 3 + 1 = 6$$



*

1	0	-1
2	0	-2
1	0	-1

=



So, after padding:

formula for output image size is ← stride = 1

$$n + 2p - f + 1 \quad \text{for } s = 1$$

but general formula →

$$\frac{n + 2p - f + 1}{s}$$

s → stride

So $n = 6$
$p = 1$
$f = 3$

$\therefore 6 + 2(1) - 3 + 1 = 8 - 3 + 1 = 6$

What is the importance of padding??

→ To prevent the info. loss of image we apply different types of padding technique

* We don't need to hard code the values in filter/kernel. First we randomly choose any values, and then through back propagation, we update these values. based on input image.

So, ~~where~~ how to do backpropagation will happen we be still have not used any ~~an~~ activation function. So where do we specifically use activation function??

Ans:- After applying convolution ~~of~~ operation, and getting the output we apply ReLU activation fn on each and ~~an~~ every cell.

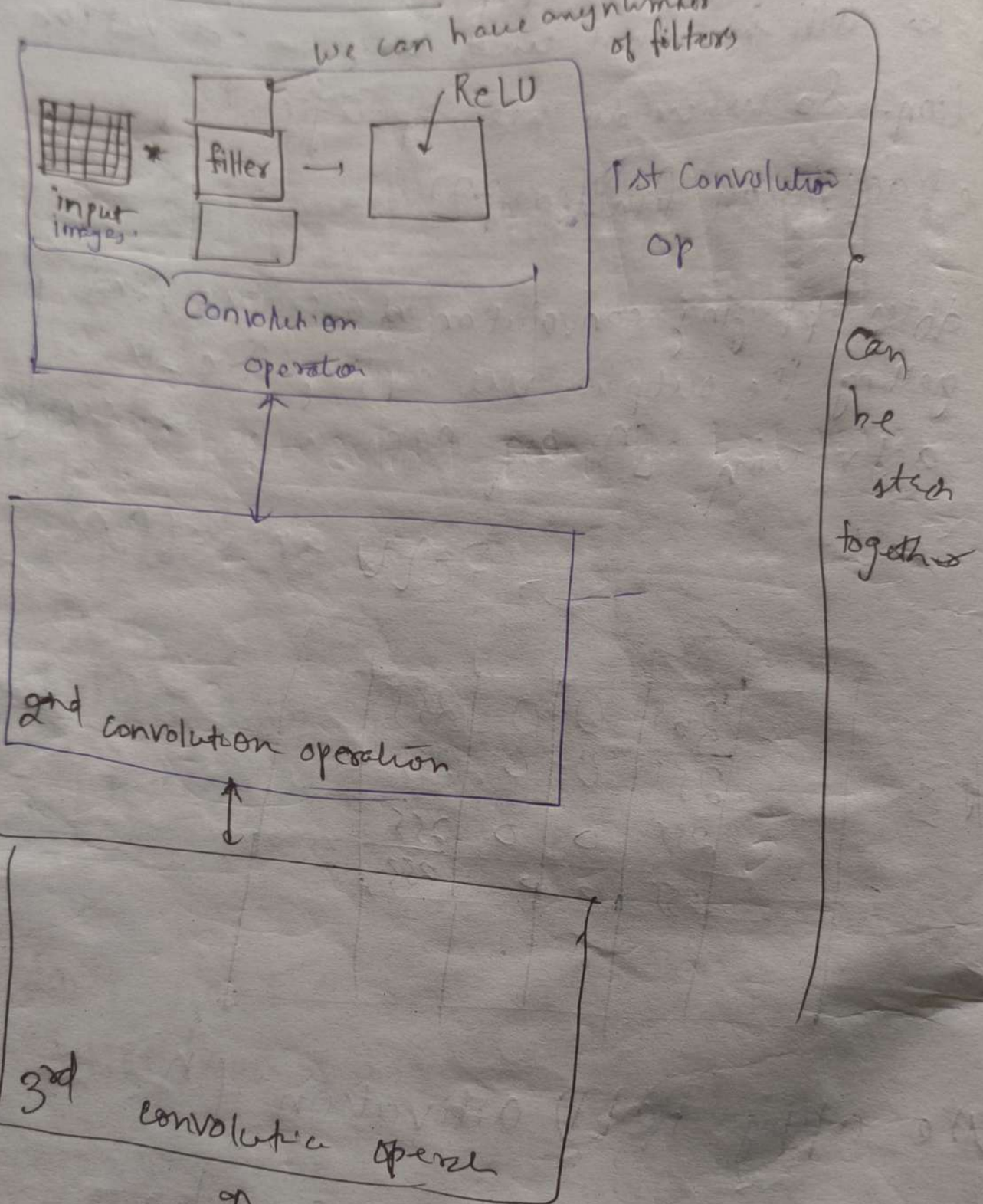
ReLU → max(0, n)

O/P =

255	0	0	255	
255	0	0	255	
255	0	0	255	
255	0	0	255	

We apply ReLU activation fn so that derivative can be found out during back propagation which can help us to update filter values.

So, steps of CNN

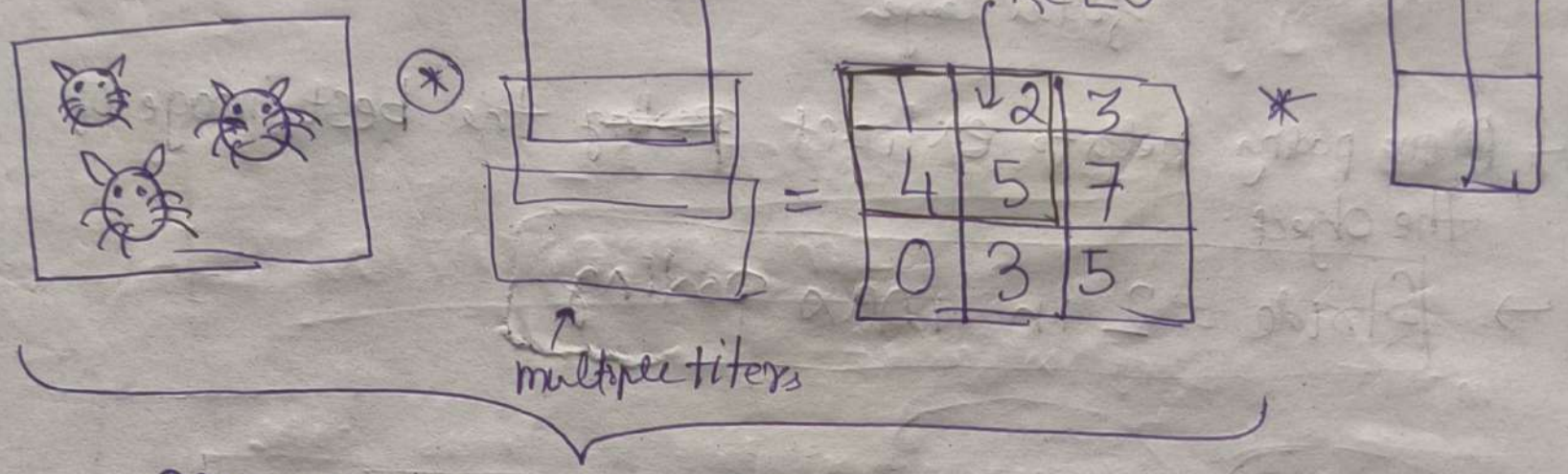


6

* We need to learn from these filters
* We need to learn and update ~~from~~ based on the input images.

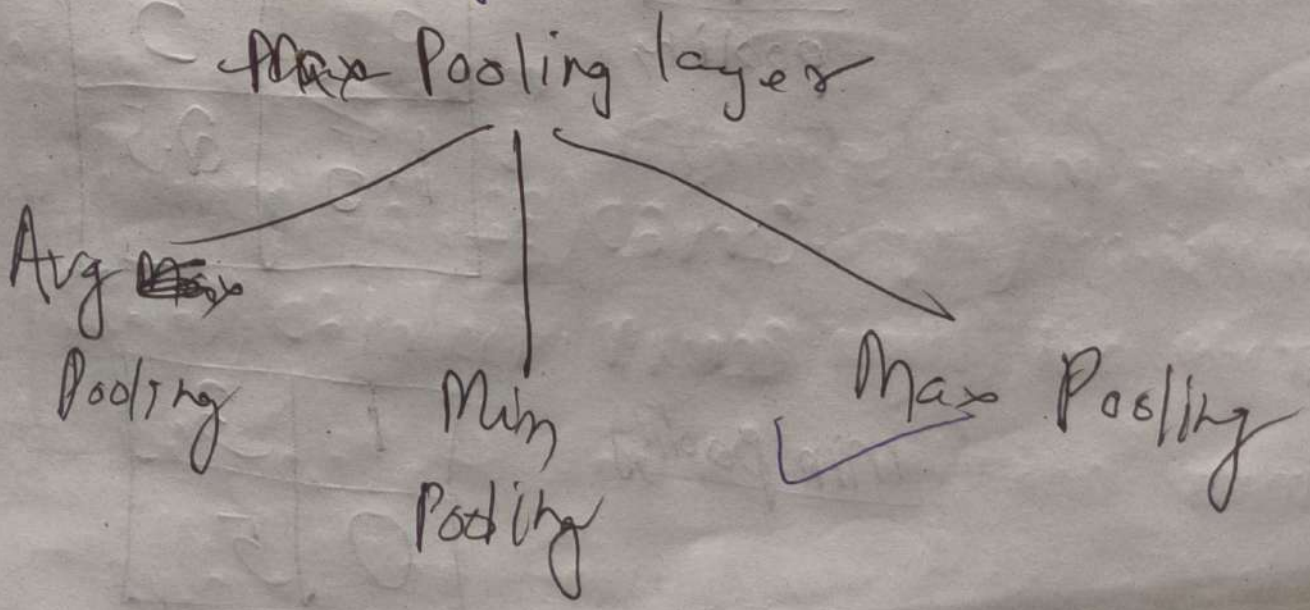
Main Aim

MAX-POOLING (Max Pooling layer) {Location invariant}

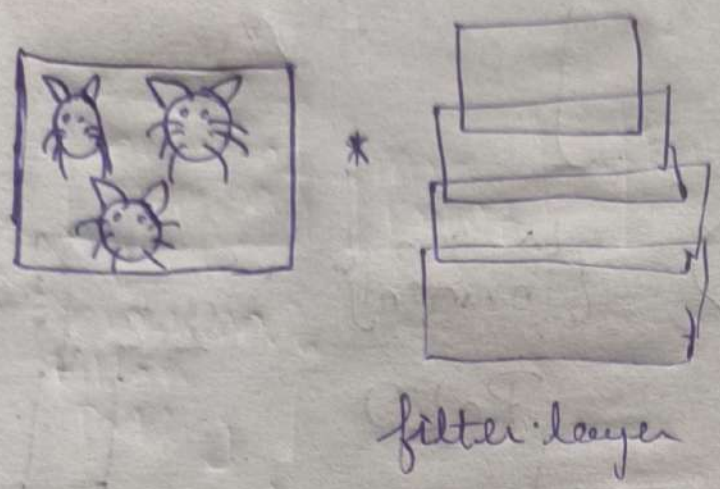


Location invariant says that my CNN till it goes to the next layer should be able to extract more clear and clearer information; (location of img can be anywhere)

Therefore, to extract much clearer information, we use max pooling.



* In pooling stride=2 ☆☆☆

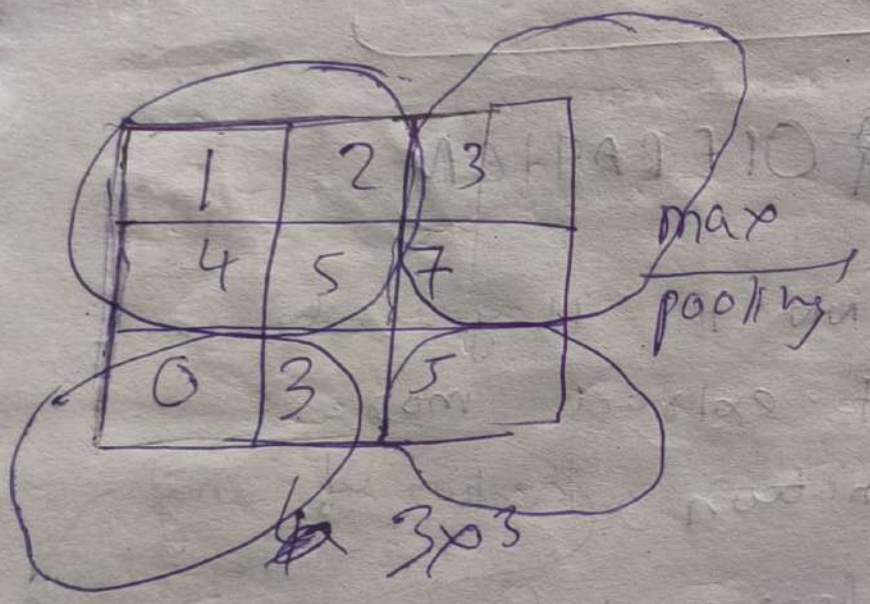


filter layer

1	2	3
4	5	7
0	3	5

→ Max pooling helps to extract ~~pooling~~ the best image of the object.

→ Stride = 2 in max pooling.



5	7
3	5

2x2

avg pooling →

6	5
1.5	2.5

min pooling →

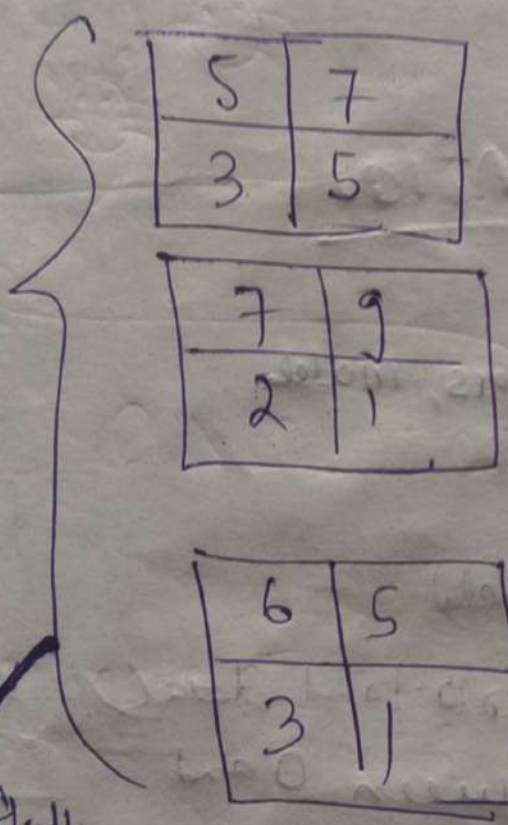
1	3
0	5

* According to different different problem statement we can use max pooling, min pooling and average pooling.
* Max pooling solves the problem of local invariant.
* With respect to different different filter operation we get different max pooling value.
We combinedly use CNN and maxpooling.

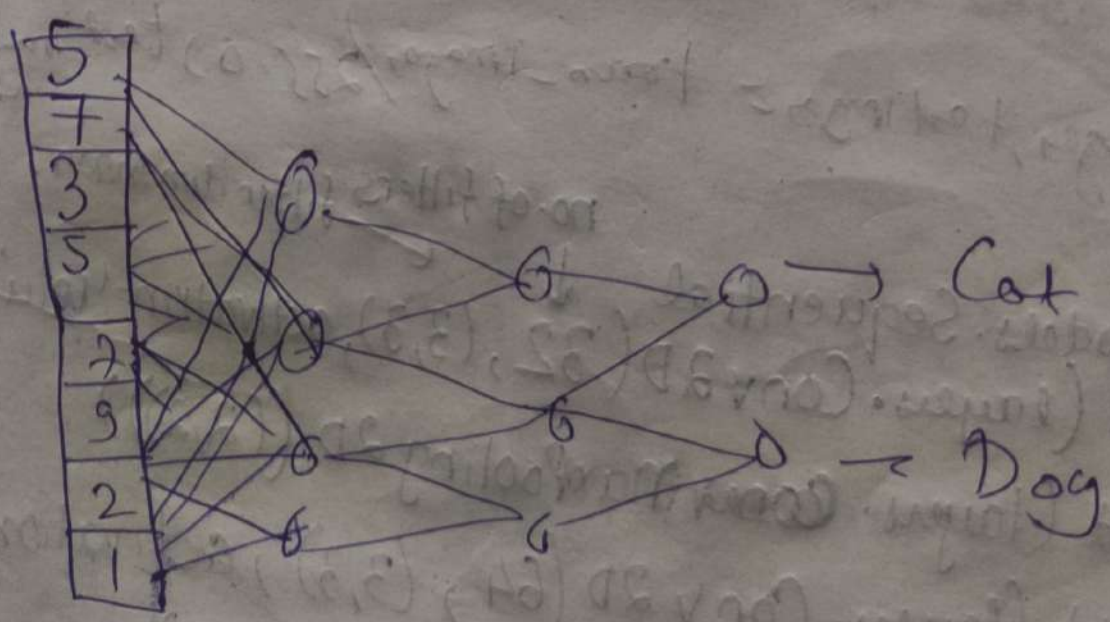
(4) Flattening layer (looks like an ANN Dense Layer)

Whatever outputs we get from max-pooling we will straightforwardly just flatten it.

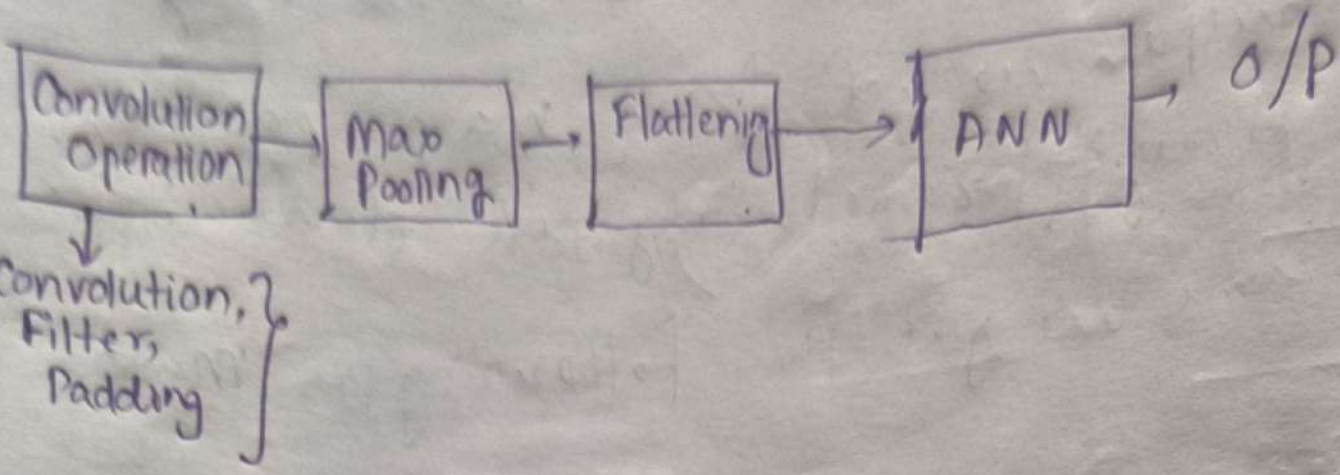
Let us assume we get the following as max pooling :-



elongate/flatten it likewise we give an input to ANN



Summary:



CNN PRACTICAL IMPLEMENTATION

```
import tensorflow as tf
import
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
(train_images, train_labels), (test_images, test_labels)
                    = datasets.cifar10.load_data()
# Normalize pixel values to be between 0 and
  1

train_images, test_images = train_images/255.0, test_images/255.0

#                          no. of filters   filter dimension
model = models.Sequential
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape
                                                    = (32,32,3
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3), activation='relu'))
```

```
model.summary()

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10)) ← output layer

model.summary()


model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategorical
                   Crossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels,
                    epochs=10, validation_
                    data=(test_images,
                    test_labels))

# Visualizing
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label
                                    ='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

```
test_loss, test_acc = model.evaluate(test_images,
                                     test_labels,
                                     verbose=2)

print('\nTest accuracy:', test_acc)
```