

Simple Linear Regression

Supervised ML \rightarrow Regression.

Dataset -
HLP feature
weight

74

80

75

-

HLP fit

height

170cm

180cm

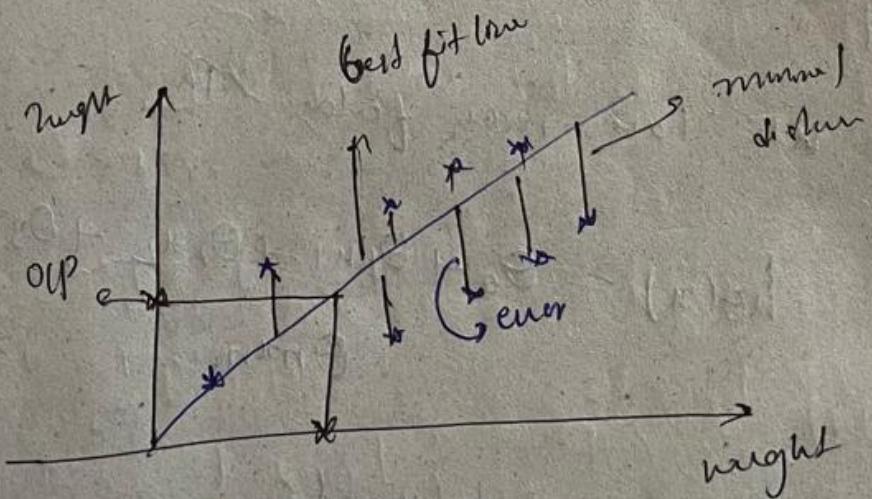
175cm

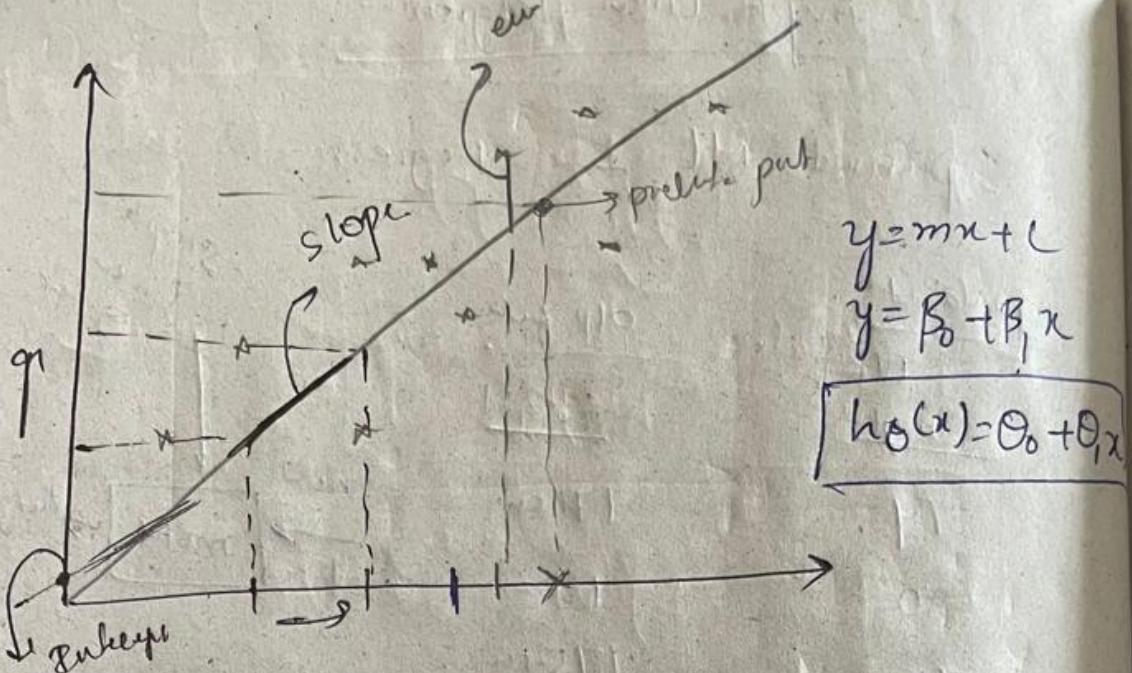
simple
linear
regression

new
wt \rightarrow model \rightarrow height

\Rightarrow Whenever we have one HP feature only
Call it as Simple linear regression

\Rightarrow if we have multiple feature only
Call it as multiple linear regression





$$\boxed{h_0(x) = \theta_0 + \theta_1 x}$$

θ_0 = Intercept \rightarrow when $x=0$, where the straight line meets y -axis

θ_1 = slope or coeff.

if $x=0$
 $h_0(x) = \theta_0$

with the unit moment with
 x -axis, what is the moment with respect to y -axis

if we have more feature then

$$h_0(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

$h_0(x) \rightarrow$ predicted points

on
 y

$$\text{error} = y - \hat{y}$$

Our main aim is to come up with the best fit line when in summation of all error it should be minimal.



Best fit line

cost function in Regression.

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

↑
predict
↓
true val

• m = total data points.
mean squared error

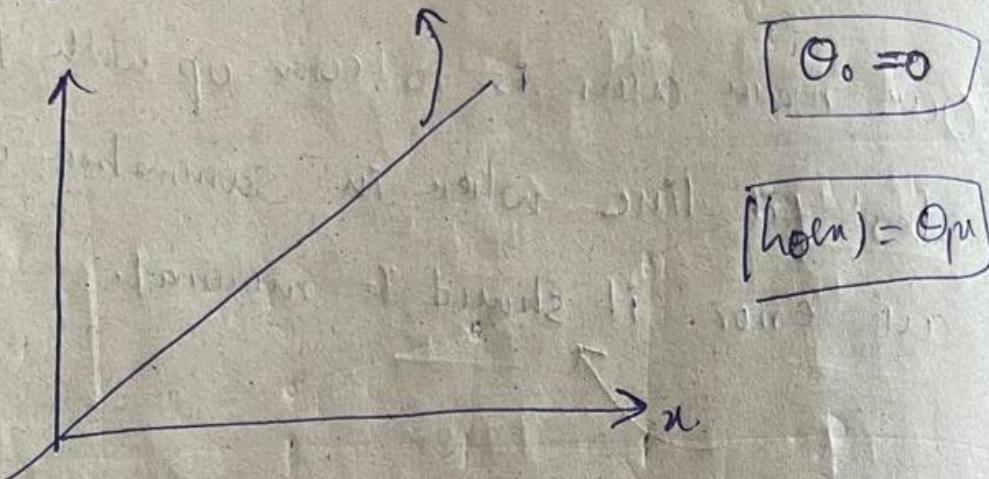
final aim what we need to solve

$$\begin{aligned} & \underset{\theta_0, \theta_1}{\text{minimize}} \quad J(\theta_0, \theta_1) \\ &= \frac{1}{2m} \sum_{i=1}^m (h_\theta(x)^i - y^{(i)})^2 \end{aligned}$$

θ_0, θ_1

$$\textcircled{1} \quad h_0(x) = \theta_0 + \theta_1 x$$

line passing the origin

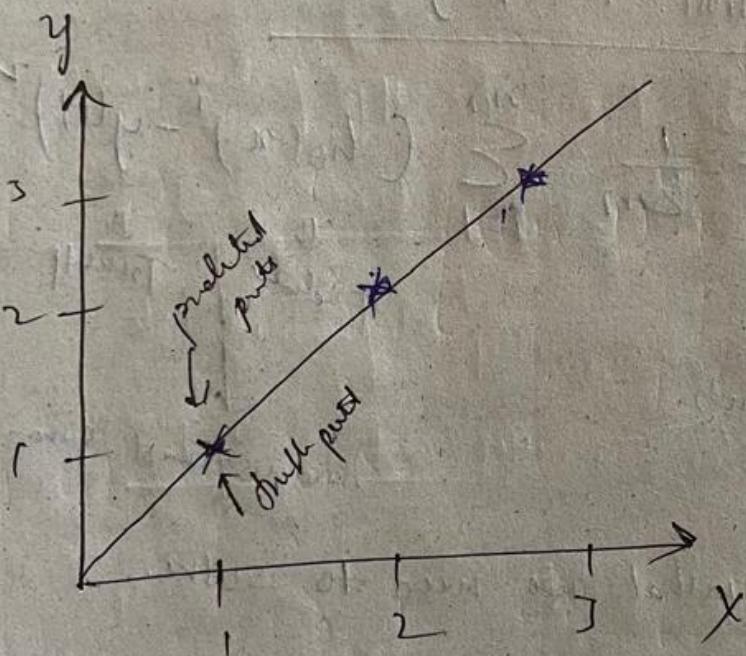


$$\theta_0 = 0$$

$$h_0(x) = \theta_1 x$$

Data set

x	y
1	1
2	2
3	3



$$h_0(x) = \theta_1 x$$

$$\text{det } \theta_1 = 1 \text{ (simplifying)}$$

$$h_0(x) = 1, x=1$$

$$h_0(x) = 2, x=2$$

$$h_0(x) = 3, x=3$$

Calculating cost function

θ₀ = 0 passing through origin

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

predicted points = true points

\downarrow
m
no. of
datapoints

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

$$J(\theta_1) = 0 \quad \text{my cost funct is zero.}$$

\downarrow

~~This~~ This is absolutely correct, b/c we have no error, & exactly the best fit line passes through all the points.

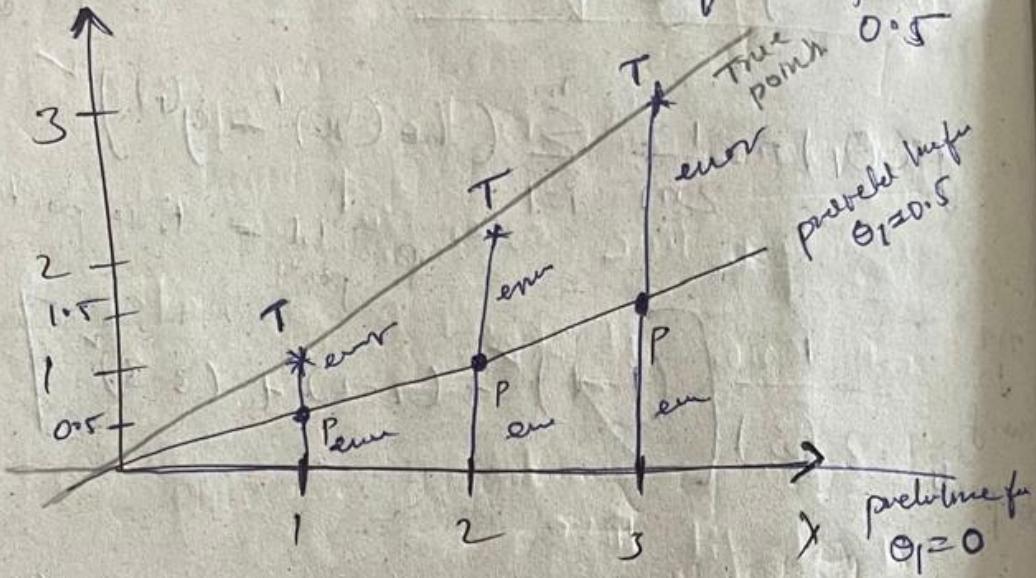
Now lets change the slope little bit

$$\text{let } \theta_1 = 0.5 \quad h_\theta(x) = \theta_1 x$$

$$h_\theta(x) = 0.5 \quad \text{if } x = 1$$

$$h_\theta(x) = 1 \quad \text{if } x = 2$$

$$h_\theta(x) = 1.5 \quad \text{if } x = 3$$



Now calculating the cost function.

$$\theta_1 = 0.5$$

after changing

$$\theta_1 = 0.5$$

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

$$J(\theta_1) \approx 0.58$$

Now calculating the cost function by changing

$$\theta_1 = 0$$

$$J(\theta_1) = \frac{1}{2 \times 3} \left[(0 - 1)^2 + (0 - 2)^2 + (0 - 3)^2 \right]$$

$$h(x) = \theta_1 x \quad \theta_1 = 0$$

$$J(\theta_1) \approx 2.5$$

$$h(x) = 0, \quad \text{if } x = 1$$

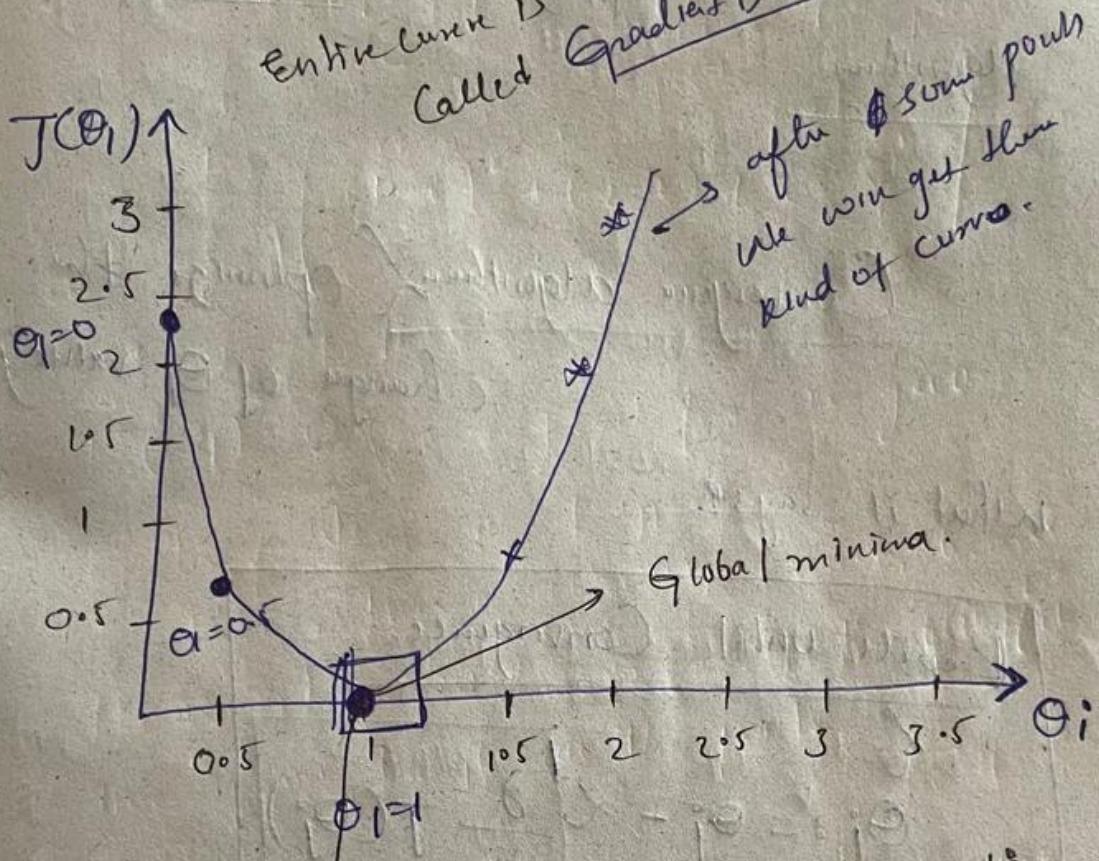
$$h(x) > 0, \quad \text{if } x \neq 1$$

$$h(x) = 0, \quad \text{if } x \neq 1$$

Beautiful assumption

plot between predicted values and slopes

entire curve is
called Gradient Descent



at this point we are getting best fit line
we need to achieve that point, by changing
rate need to achieve that point, by changing

$\theta_1 \theta_0$ values.
↑
slope
rate

→ We cannot keep on changing θ_0 & θ_1 values, that is not at all possible
so we should apply some convergence
algorithms.

Convergence algorithms Optimize the
change of θ_1 value

What it says?

Repet until Convergence

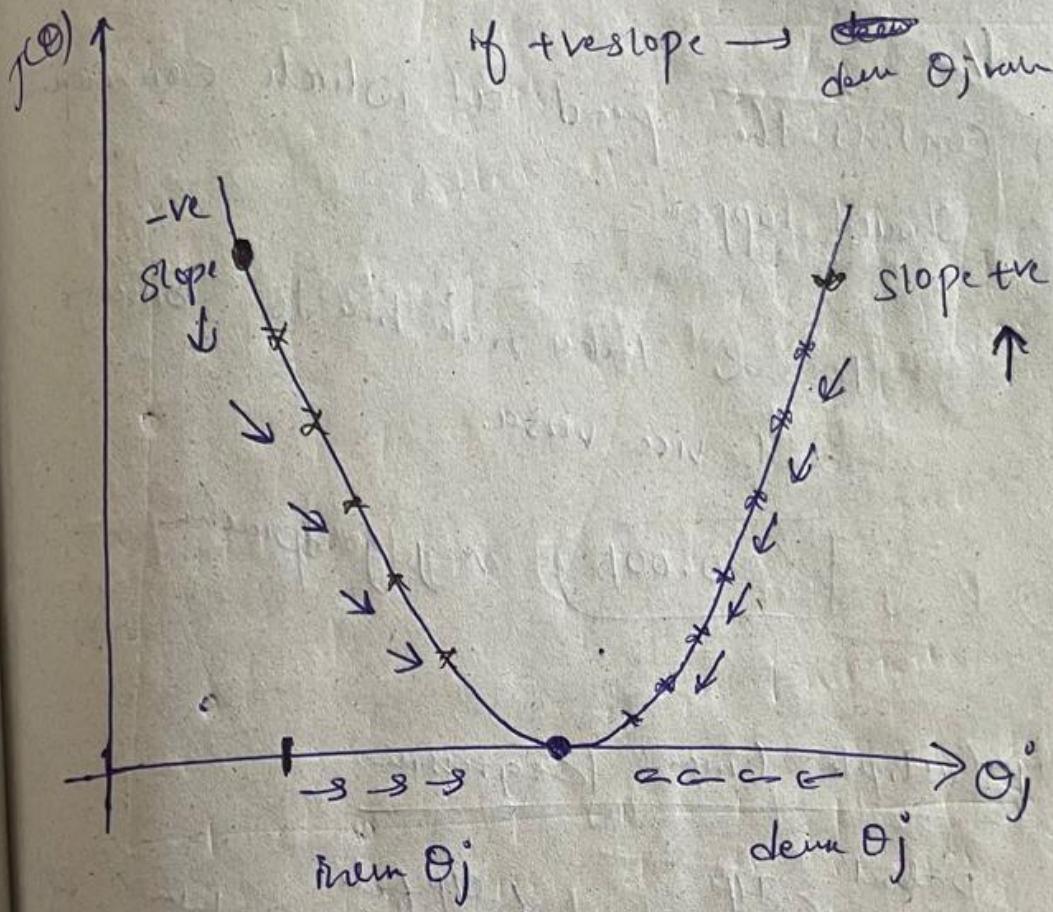
$$\theta_j := \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta) \right]$$

θ_1 value is going to be change
much more slowly.

Derivative \rightarrow Slope at that particular point

If -ve slope \rightarrow add θ_j^* value.

If +ve slope \rightarrow ~~add~~ θ_j^* value.



for -ve slope:

$$\theta_j = \theta_j^* - \alpha(-ve)$$

$$\theta_j = \theta_j^* + (+ve)$$

for +ve slope

$$\theta_j = \theta_j^* - \alpha(+ve)$$

$$\theta_j = \theta_j^* - (-ve)$$

We are calculating the slope in such a way that
if it is a -ve slope θ_j^* value is going to be.

mean and +ve slope θ_j^* value is
going to be decrease

Σ is nothing but learning rate. $\alpha = 0.001$

\$ Controls the speed at which convergence should happen.

↑ less α takes more time to change
and vice versa.

$\alpha = 0.001$ very good problem

Multiple Linear Regression

Patient	IP	IP fact	Intercept	Slope
<u>weight</u>	<u>height</u>			
-	-			
-	-			
-	-			

There will be only one intercept in any regression (single or multiple)

Hour pricing factors depend
 f
 No. of rows Size of hour locah O/P f
 x_1 x_2 x_3 pri.

$$h(n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

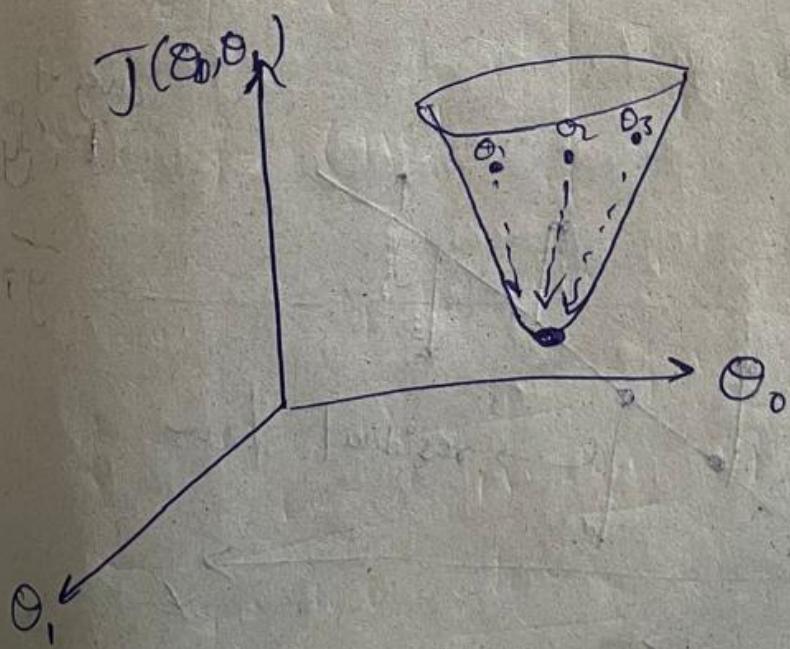
\uparrow \uparrow \uparrow
 no. of size of location
 rows hour

(multiple linear regions)

$\theta_1, \theta_2, \theta_3$ = slope coeff.

θ_0 → intercept

how gradient descent looks in multiple linear regions



Performance Metrics Used In Linear

Regression:

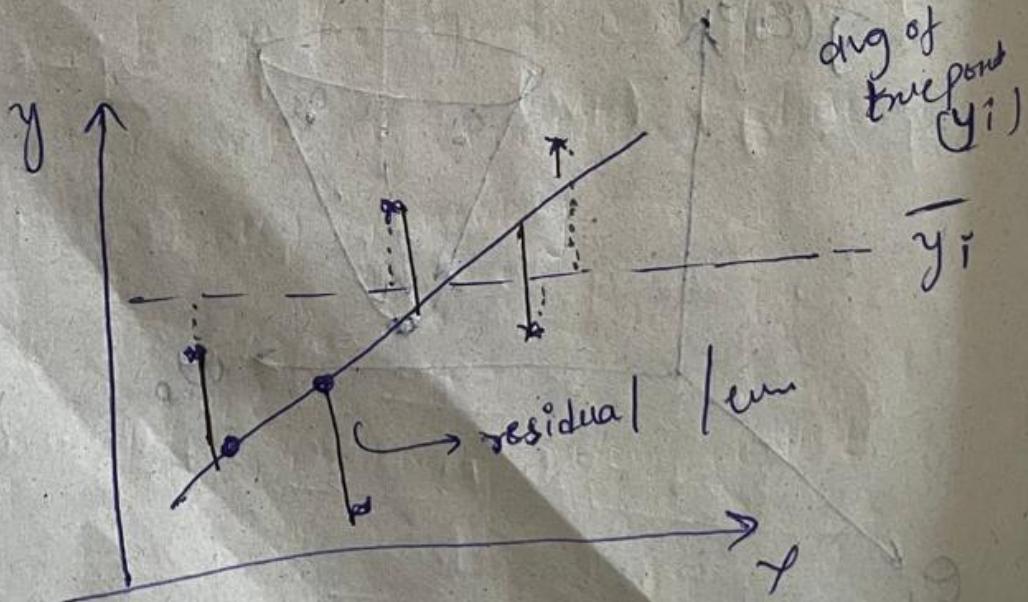
Used to determine whether the model is good or not for a specific problem statement.

① R squared

② Adjusted R squared

$$R^2 = \frac{SS_{Reg}}{SS_{Total}}$$

$$R^2 = 1 - \frac{\text{Sum of Squared Residual}}{\text{Sum of Squared Total}}$$



$$= 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2}$$

$$= 1 - \frac{\text{Small number}}{\text{Bigg number}}$$

$$= 1 - \text{Small number}$$

$\approx 1 \Rightarrow 0.70 \Rightarrow 70\% \text{ accuracy}$

$0.85 \Rightarrow 85\% \text{ accuracy}$

$0.90 \Rightarrow 90\% \text{ accuracy}$

report now our model is working

more the value towards 1 \Rightarrow higher the accuracy.

Overfitting
Underfitting

Adjusted R Squared

Dateset \rightarrow price $\frac{Sol}{m} + P^{MP}$

size of house

price $+ve$ correlation

Rsq = 77%

$\Rightarrow \text{Sqr} = 75\% - 0.75$

No of bedrooms P⁹ Price⁹
Area

$$R_{\text{squn}} = 80\% = 0.80$$

Location LT P⁹

$$R_{\text{squn}} \rightarrow 85 = 0.85$$

Gender: if we add this feature which
is not at all correlated with price
and calculate R_{\text{squn}} = 87%.
↑
increases but not big
value.

Even though the gender and price are
not at all correlated, still it is
basically meaningful.

g

This is the problem with R_{\text{squn}}

In order to prevent that we are going to use Adjusted R squared

$$\text{Adjusted R squared} = \frac{1 - (1-R^2)(N-1)}{N-P-1}$$

N = No. of data points (excluding data)

P = No. of independent factors (excluding shape(1)).

$P=2 \quad R^2=90\% \quad R^2_{\text{adjusted}} = 86\%$

$P=3 \quad R^2=92\% \quad R^2_{\text{adjusted}} = 82\%$

When the fish is not always
complete with OPL for

if it is correlated with
radius is
going to be
present than
prior

Overfitting and Underfitting (Bias and variance)

① Training dataset

② Test dataset

③ Validation dataset

from date part	size	bedroom	age	price
	1100	2	20	30k
second part	2000	3	5	40k
third part	1100	2	10	25k

Dataset

1000 data points

Now we have
three data parts

size of house	no. of bedrooms	price
—	—	—
—	10	—
—	—	—
—	—	—
—	—	—
—	—	—

Split the data into two parts

Training dataset
(700)

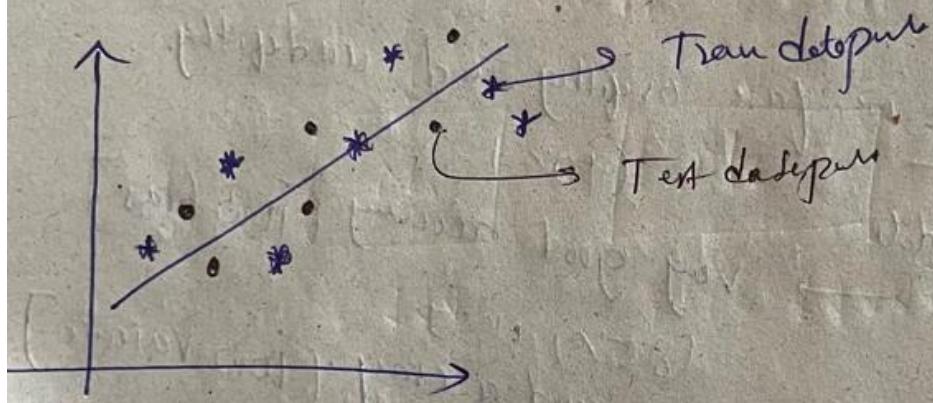
Test dataset
(300)

↓
use to model to train

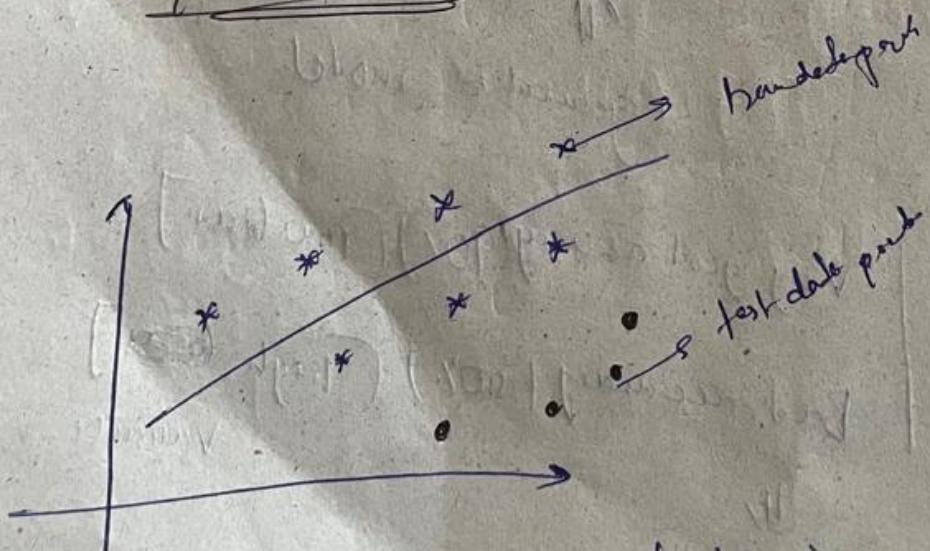
6 Test my model.
→ while testing my model
is never going to know about this

TRAIN	accuracy is low [high bias]
TEST	accuracy is low [high variance]

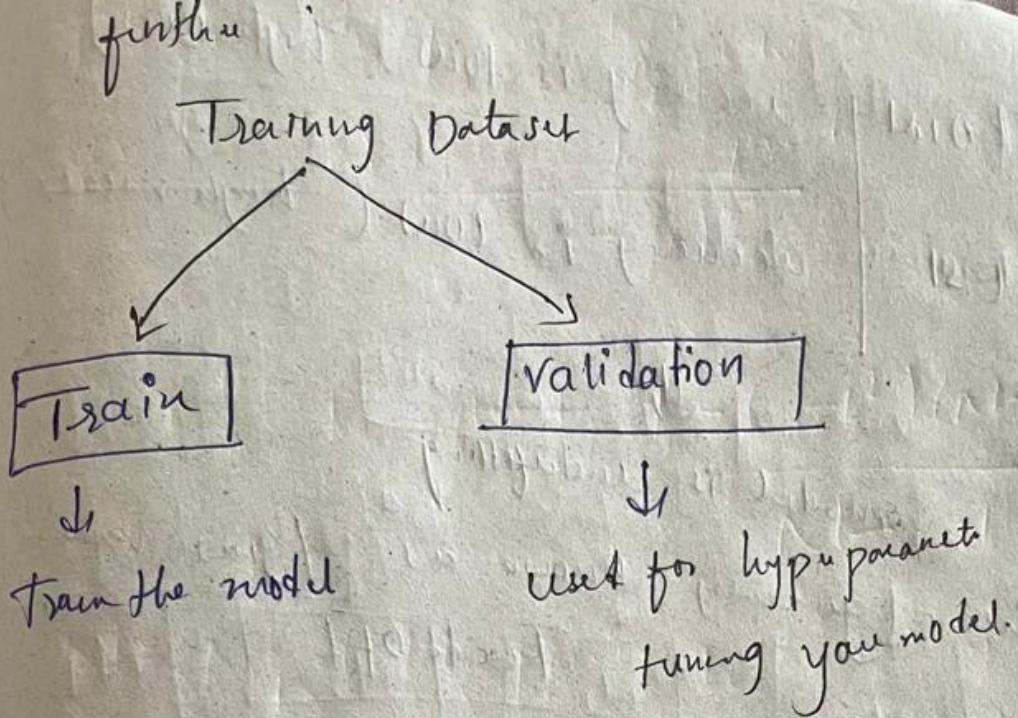
y
 model is underfitting.
 This is generally known as Bias-variance Trade Off



Generalization model: low bias, low variance



low bias, high variance.



When do we face overfitting and underfitting.

TRAIN data → very good accuracy [low bias]
 (90%)

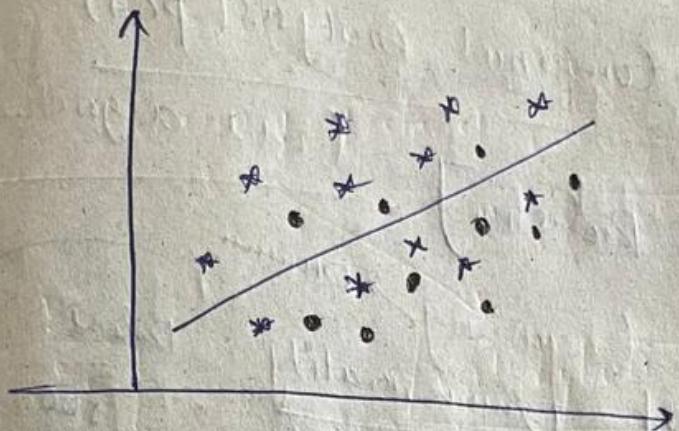
TEST data → very good accuracy [low variance].
 (85%)

↑
generalized model.

TRAIN	Very good accuracy (90%) [low bias]
TEST	Bad accuracy (50%) [high variance] variance

↑

model is overfitting.



Underfitting (high bias, high variance.)

→ Accuracy of a model is high with trained dataset
and bad/below with test dataset.

↳ Overtfitting (low bias,
high variance)

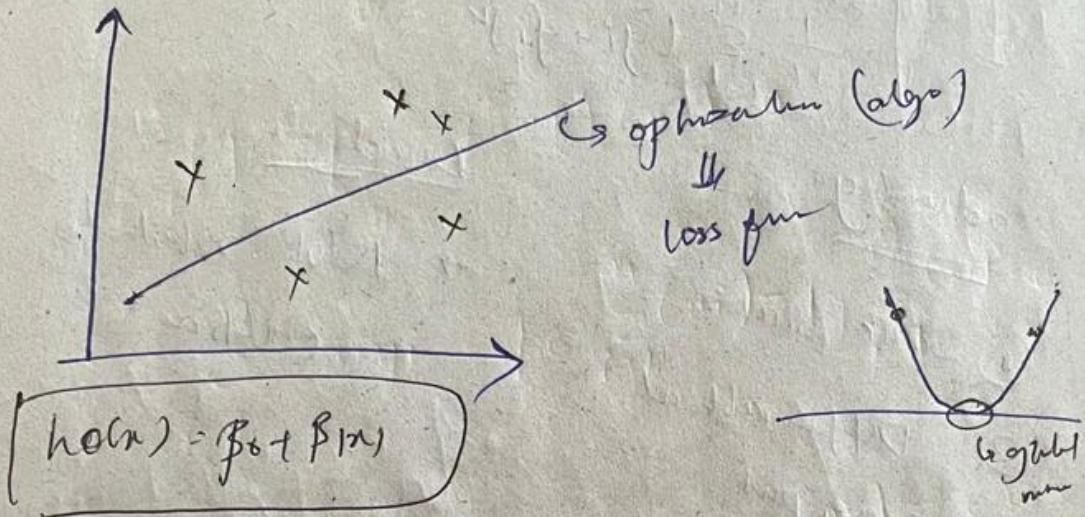
→ Similarly, accuracy of a model is low with
trained dataset and bad/below with test
dataset

Underfitting (high bias,
high variance)

Bias, Variance trade-off

Linear Regression Using OLS (Ordinary)

least square



OLS technique basically says:

Can apply some formula and calculate

β_0 and β_1

Ordinary least square

$$S(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$$

and β_0 and β_1

$$\frac{\partial S}{\partial \beta_0} (\beta_0, \beta_1) = \frac{2}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) (0-1)$$

$$= \frac{-2}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) = 0 \rightarrow \textcircled{1}$$

$$\frac{\partial S}{\partial \beta_1} (\beta_0, \beta_1) = \frac{2}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)(0 - 0 - x_i) = 0$$

$\hookrightarrow \textcircled{2}$

$$= \frac{-2}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)(x_i) = 0$$

$\hookrightarrow \textcircled{2}$

eq-s $\textcircled{1}$

$$- \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) = 0$$

$$- \sum_{i=1}^n y_i + n \times \beta_0 + \beta_1 \sum_{i=1}^n x_i = 0$$

$$= n\beta_0 + \beta_1 \sum_{i=1}^n x_i = \sum_{i=1}^n y_i$$

$$\beta_0 = \frac{\sum_{i=1}^n y_i}{n} - \beta_1 \frac{\sum_{i=1}^n x_i}{n}$$

$$\boxed{\beta_0 = \bar{y} - \beta_1 \bar{x}}$$

Eq 2:

$$\frac{-2}{n} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) (u_i) = 0$$

↓

$$\sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) (u_i) = 0$$

↓

$$\sum_{i=1}^n n_i y_i - \beta_0 \sum_{i=1}^n u_i - \beta_1 \sum_{i=1}^n (u_i)^2 = 0$$

$$\sum_{i=1}^n (n_i y_i - \beta_0 n_i - \beta_1 u_i^2) = 0$$

Solve β_0 ↗

$$\sum_{i=1}^n (n_i y_i - (\bar{y} - \bar{u}_i) n_i - \beta_1 u_i^2) = 0$$

$$\sum_{i=1}^n (n_i y_i - n_i \bar{y} + \beta_1 \bar{n}_i n_i - \beta_1 u_i^2) = 0$$

from

$$\beta_1 = - \frac{\sum_{i=1}^n (y_i - \bar{y})}{\sum_{i=1}^n (n_i - \bar{n}_i)}$$

$$\beta_1 = \frac{-\sum_{i=1}^n (y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})}$$

$$\boxed{\beta_1 = \frac{\sum_{i=1}^n (y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})}}$$

and

finding by any OLS if we need to calculate intercept (β_0) and β_1 (slope) number as

$$\boxed{\beta_0 = \bar{y} - \beta_1 \bar{x}} \rightarrow \text{Intercept}$$

$$\boxed{\beta_1 = \frac{\sum_{i=1}^n (y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})}} \rightarrow \cancel{\text{slope}}$$

~~slope~~

$$X \quad Y \quad (y_i - \bar{y}) \div (x_i - \bar{x}) \quad \beta_1 \quad B = (\bar{Y} - \bar{\beta}_1 \bar{x})$$

cofficient value

✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-
✓	-	-	-

~~OLS & the reg.~~

Overall Result of

$$\begin{matrix} \text{Lower regres} \\ \text{Gradual decay} \end{matrix} \cong \begin{matrix} \text{Upper regres} \\ \text{OLS} \end{matrix}$$

Correlation

$r_{xc} \rightarrow$ close to +1 (input 9, out 9)

$r_{vc} \rightarrow$ close to -1 (input 11, out 11)

$r_o \rightarrow$ number for update on

measures the strength and direction of
their linear relationship.

Simple Linear Regression: Practical Implementation.

```
import pandas as pd  
import numpy as np  
import matplotlib.matplotlib.pyplot as plt  
%matplotlib inline.
```

Reading the dataset

```
df = pd.read_csv('height_weight.csv')
```

```
df.head()
```

use scatter plot to check whether the data is linearly distributed or not.

```
plt.scatter(df['weight'], df['height'])
```

```
plt.xlabel('WEIGHT')
```

```
plt.ylabel('HEIGHT')
```

check the correlation

whether it is positive or not

```
df.corr()
```

corr > 0.7 → strong

or corr < 0.7 → moderate strength
< 0.5 weak, LR might not be the best model

use Seaborn to run

import Seaborn as sns

sns.pairplot(df)

First Step in Linear Regression

Divide features into independent and dependent
features.

- major thing.
- make sure that your independent features
should be in 2D array or df
- and dependent / output feature in 1D array

Sure.

$x = df[["weight"]]$ → independent / input feature

$y = df["height"]$ → dependent / output

Next Train test split

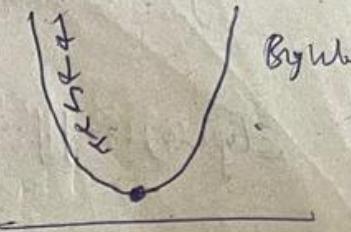
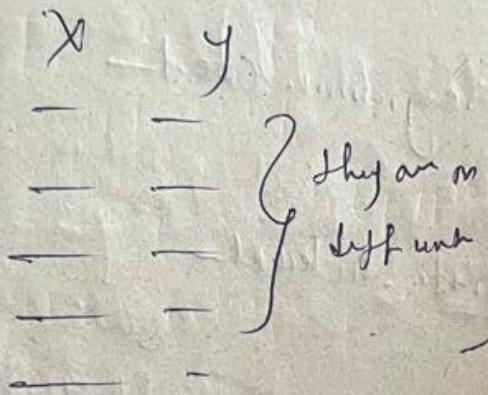
75% train
25% test

for sklearn model select weight train-test split

$x_{train}, x_{test}, y_{train}, y_{test} = train_test_split(x, y,$
 $\text{test size} = 0.25,$
 $\text{random state} = 42)$

Next important step is Saveli's graph.

(kg/l. (cm))



By W.

$$n=0, \pm 0.2f$$

$$z_{\text{ave}} = \frac{w - u}{6}$$

Standardized

→ only for right feather

from steken. preparing imp. 1 Standard Scale.

$$\text{Scale} = \text{Shd. Scale} (.)$$

$$z_{\text{ave}} = \text{Scale} \cdot f + \text{bias} (\text{in body})$$

n-hm

$$n_{\text{test}} = \text{Scale} \cdot \text{bias} (\text{in test})$$

not

applying linear Regression

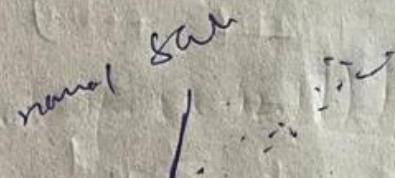
from steken. I was used input Curve Regress.

$$\text{regress} = \text{Curve Regress} (n_{\text{job}} = 1)$$

regress-fit (x-hem, y-hem)

slopeCoeff = regress. coef

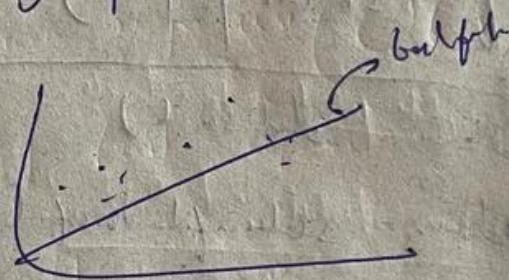
intercept = regress. intercept



Best fit line

plt.scatter (x-hem, y-hem)

plt.plot (x-hem, regress.predict (x-hem))



Now we need to predict the test data

y-pred = regress.predict (x-test)

y-pred

are

Performance Metrics

from Sklearn.metrics import mean_absolute_error,
mean_squared_error

$$mse = \text{mean}(\text{y-test}, \text{y-pred})$$

$$mae = \text{mean}(\text{y-test}, \text{y-pred})$$

$$rmse = np.sqrt(mse)$$

Calculating R₂, S_{resid} and adjusted R₂ from

from Sklearn.metrics import r2_score.

$$S_{\text{resid}} = \text{r2_score}(\text{y-test}, \text{y-pred}) \quad 0.73$$

$$\text{adjusted } R^2 = 1 - (1-R^2) * (n-1) / n-p-$$

n - no. of stop words

p - no. of input features

4) predicting the new data

regress. predict([C72])

↳ it is not in standard form
4) 1401 ht

regress. predict(solver.bayes([C72]))

↳
155 @ht ✓

Implementation linear regress. w/ OLS

ordinary least squares

target stations tel. agt as sm.

model = sm.OLS(y-hm, x-h).fit()

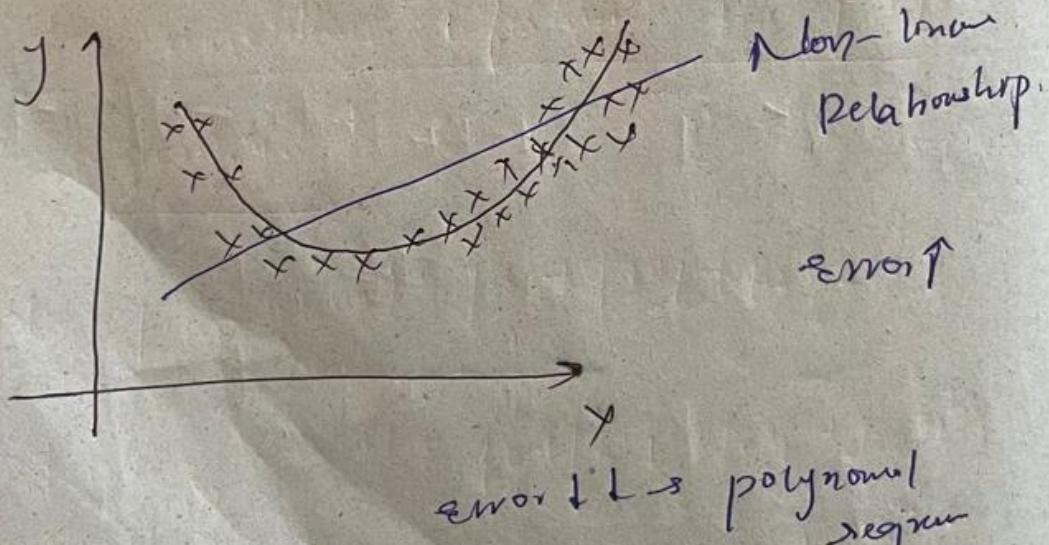
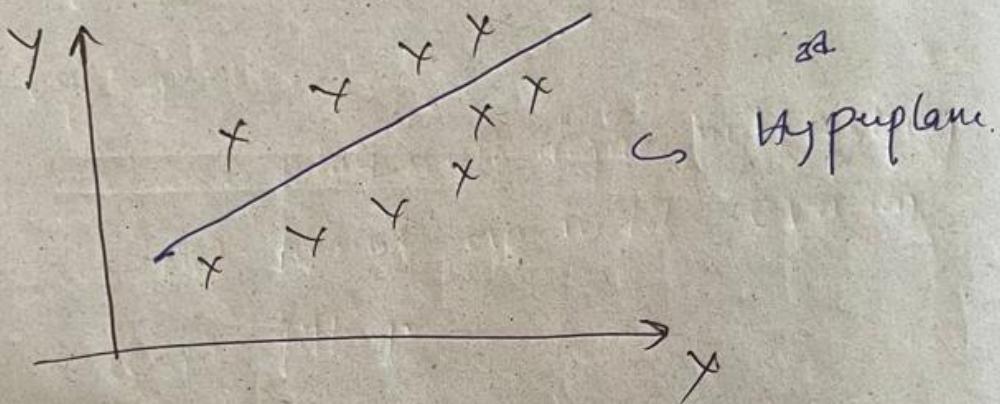
Multiple Linear Regression

Polynomial regression

$h_{\theta}(x) = \beta_0 + \beta_1 x \rightarrow \underline{\text{Single Linear Regression}}$

$$h_{\theta}(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

\hookrightarrow multiple Linear Regression



error $\uparrow \rightarrow$ polynomial regression

Polynomial Degree (1 IP and 1 OP feature)

$$\text{deg} = 0 \quad h_0(x) = \beta_0 + \beta_1 x^n$$

$\beta_0 x^n + \boxed{\beta_1 x^n}$

Simple
polys
regress

Poly deg=0, $h_0(x) = \beta_0 \Rightarrow$ constant value

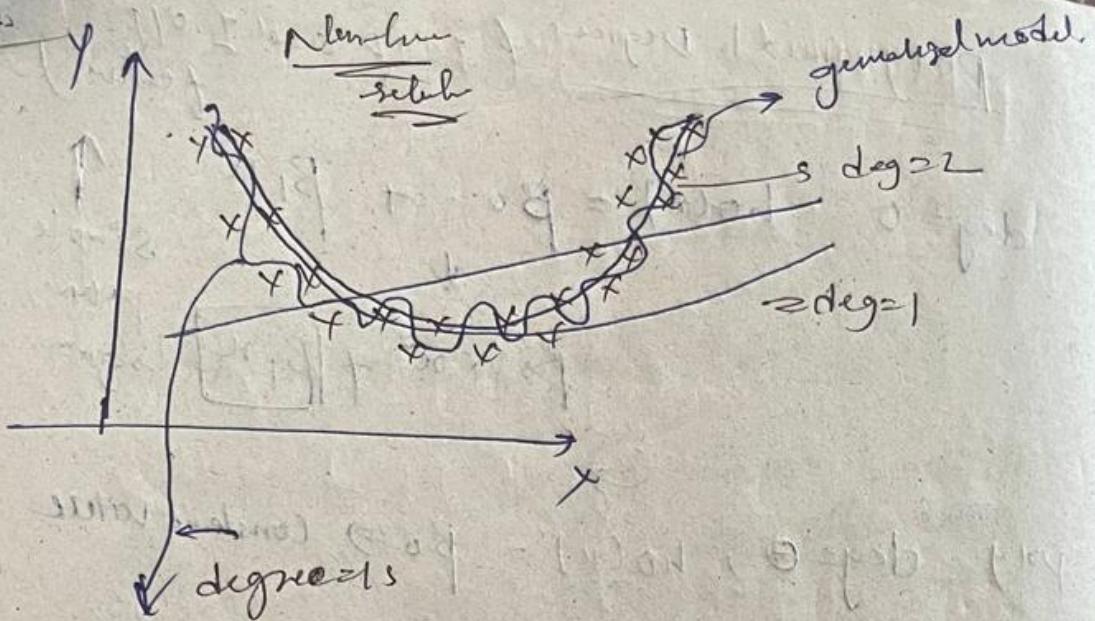
Poly deg=1, $h_0(x) = \beta_0 x^0 + \beta_1 x^1$
End for with different
Poly

simple linear
regression

Poly deg=2, $h_0(x) = \beta_0 x^0 + \beta_1 x^1 + \beta_2 x^2$

Poly deg=n, $h_0(x) = \beta_0 x^0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3 + \dots + \beta_n x^n$

for simple polynomial regression.



Overfitting,
one should play with degree, not making
sure not to make our model over fit
under fitting.

for more than 1 feature the polynomial

regression.

$$\text{deg} = 1 \quad h_0(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\text{deg} = 2 \quad h_0(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2$$

Polynomial regression Implementation

- Import necessary libraries
 - Create non linear dataset points
 - See the visualization b/w x, y
 - Train test split
 - you can also
shuffle the data
 - Implement linear regression
 - Performance metrics
 - Check R² score
 - Do some visualization
 - Best fit line untrained unhandled
- Now we will see good amount of error.
- Let's apply polynomial regression.
- Applying polynomial transformation from sklearn-prepr import polyfitter

$\text{poly} = \text{polyfit}(\text{degrees}, \text{weights}, \text{bias}=\text{True})$

$$h_0(x) = \beta_0 + \beta_1 x^1 + \beta_2 x^2$$

Pipeline in polynomial.

Pipelining Concepts:

Create a generic function, in which, whatever degree we assign. we, would do. See, what kind curve, it will be fitting for a new data set.

from sklearn.pipeline import Pipeline.

def poly_regression(degree):

x_new = np.linspace(-3, 3, 200).reshape(200, 1)

poly_features = PolynomialFeatures(degree=degree, include_bias=True)

lin_reg = LinearRegression()

poly_regression = Pipeline([

(*poly_features*, poly_features),

(*lin_reg*, lin_reg)

])

poly_regression.fit(x_train, y_train)

y_pred_new = poly_regression.predict(x_new)

~~#~~ Plotting prediction line

plt.plot(x-new, ypred-new, 'r', label='Degree
str (degree), linewidth=3)

~~plt.plot(x-harm, ypred-harm)~~

plt.plot(x-harm, y-harm, 'b.', linewidth=3)

plt.plot(~~x~~-test, y-test, 'g:',
linewidth=3)

plt.legend(loc='upper left')

plt.xticks('x')

plt.ylabel('y')

plt.axis([-4, 4, 0, 10])

plt.show()

poly-reg (1)

(2)

(3) By using degree we are going to
get the best fit line

(4)

(15) Using more may leads to
(20) overfitting

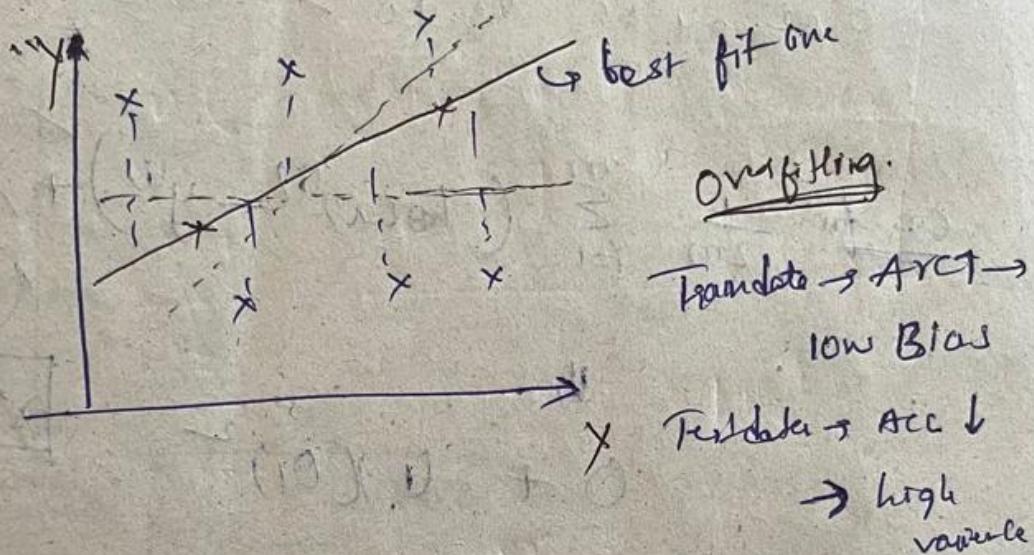
Ridge Regression, Lasso Regression, ElasticNet

Regression → used to reduce overfitting.

reduce overfitting.

alone

Ridge Regression. (L2 regularization)



$$\text{Cost fn: } \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n (\theta_j)^2$$

$$\downarrow \\ 0$$

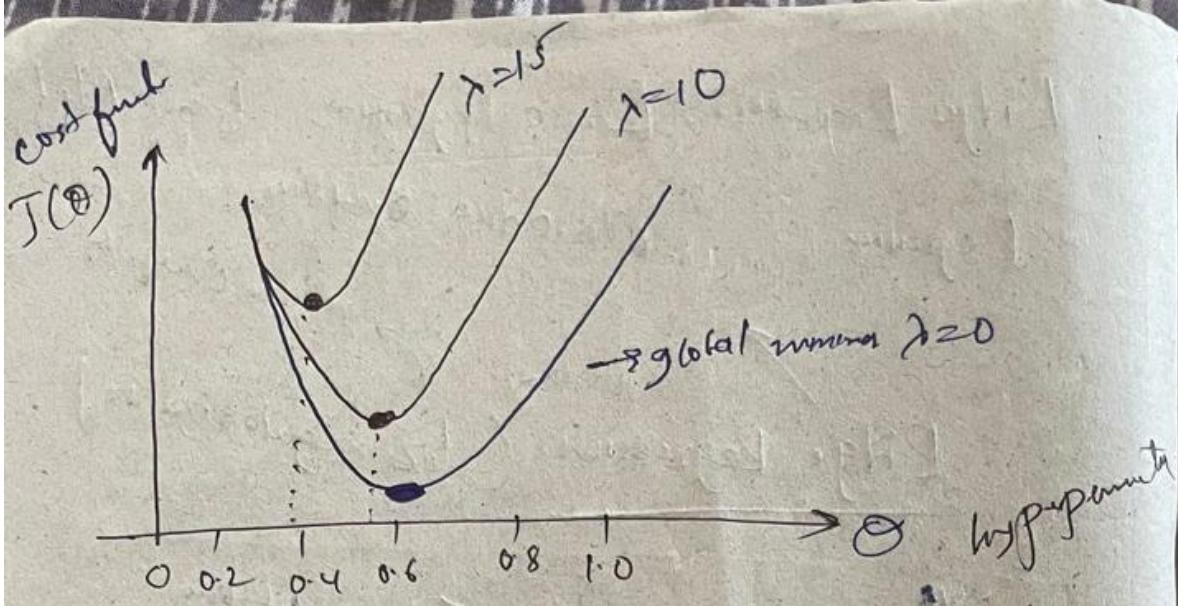
This means
model is Overfitting

$$0 + (1)[(\theta_1)^2]$$

$$\lambda = 1$$

$$> 0$$

This is how ridge regression is making sure
that Overfitting will not happen



cost-fn: $\frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \epsilon_j^2$ (stop)

$\lambda \uparrow$ slope ↘

$$h_\theta(u) = \theta_0 + \sum_{j=1}^n \theta_j u_j$$

$$h_\theta(u) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

~~θ₀~~

~~θ₁~~

~~θ₂~~

$$\rightarrow 1 = 0.34 + 0.52x_1 + 0.48x_2 + 0.24x_3$$

~~θ₃~~

now. after applying ridge regression.

$$= 0.34 + 0.40u_1 + 0.38u_2 + 0.4u_3$$

it just reduces the coefficient but not

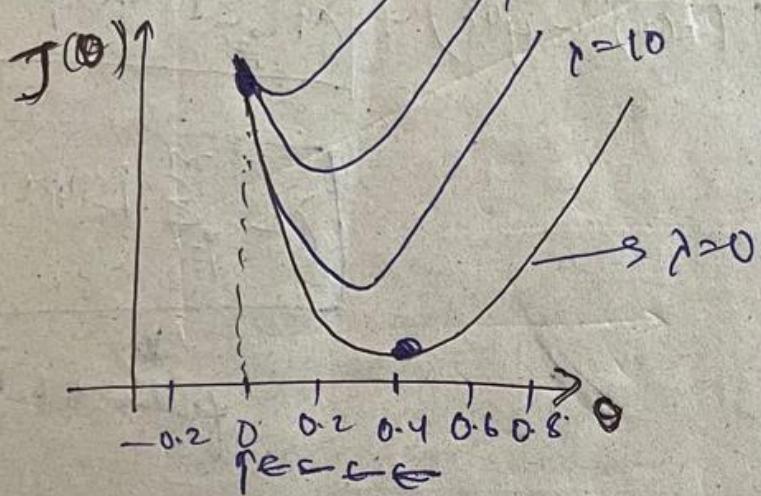
make's geno. that does not make the
geno.

Lasso Regression (L₁ Regularization)

used for feature selection

$$\text{cost fn} = \frac{1}{2m} \sum_{i=1}^m (\hat{y}(x_i) - y^{(i)})^2 + \lambda \sum_{i=1}^n |\text{slope}|$$

hyperplane



unlike ridge
regression
can observe
that after
point slope
is becoming 0

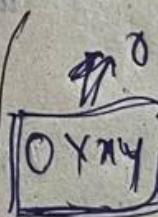
that new talk
are removing
feature

$$h_0(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$= 0.52 + 0.6x_1 + 0.72x_2 + 0.34x_3 + 0.12x_4$$

Lasso regression

$$= 0.52 + 6.51x_1 + 0.60x_2 + 0.14x_3 +$$



which one feature is
not that consider this is
is going to be removed
this feature is not the 3rd
with OIP feature.

Elastic Net Regression

(Ridge + Lasso)

① Feature Selection
② Reduce Overfitting

Reduce overfitting

Cost fn =

$$\frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 + \boxed{\lambda_1 \sum_{i=1}^m (\text{slope})^2}$$

$$\boxed{\lambda_2 \sum_{i=1}^m |\text{slope}|}$$

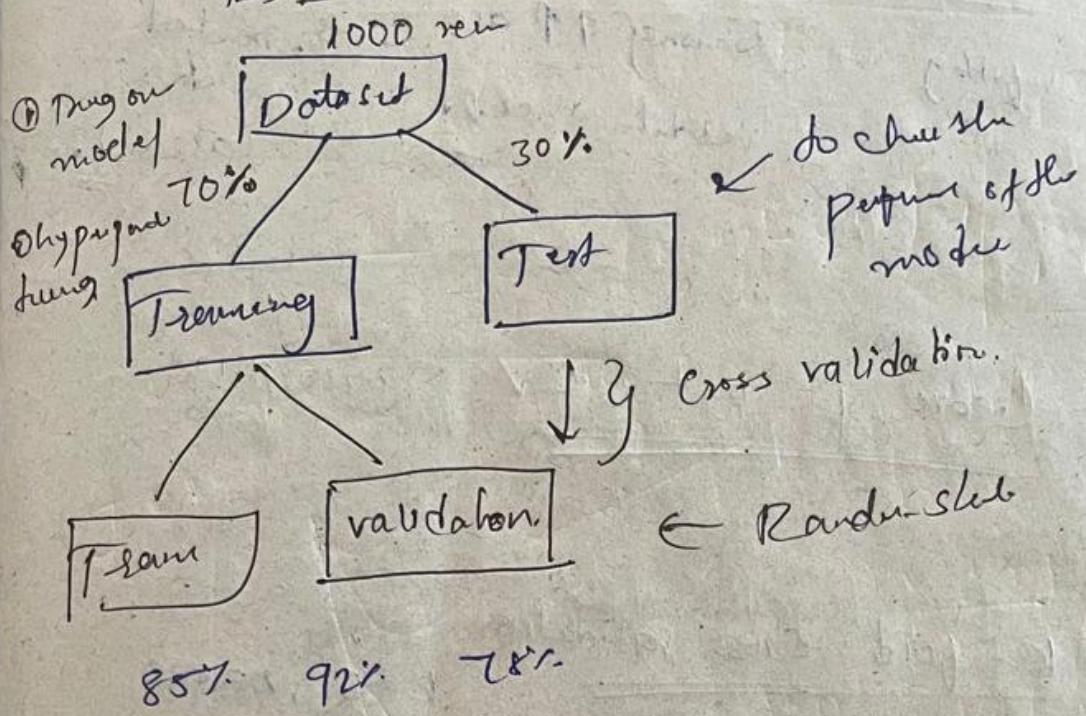
feature selection

Suppose our real life model that is overfitted and has lots of features then we can use Elastic Net regression

Ridge, Lasso, elastic Net

hyperparameters tuning the Under Regression

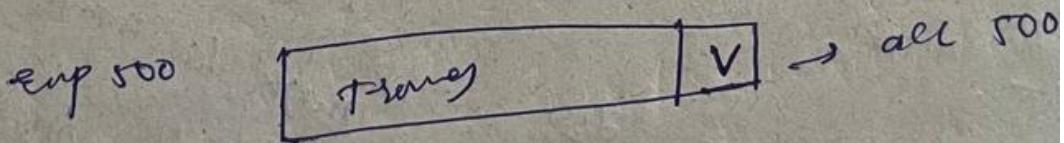
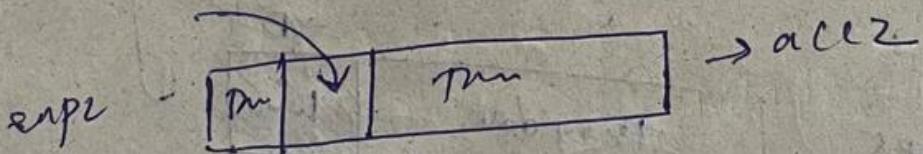
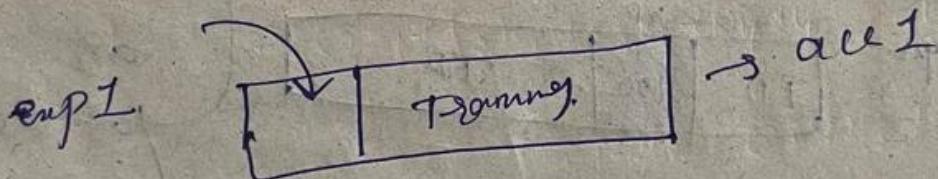
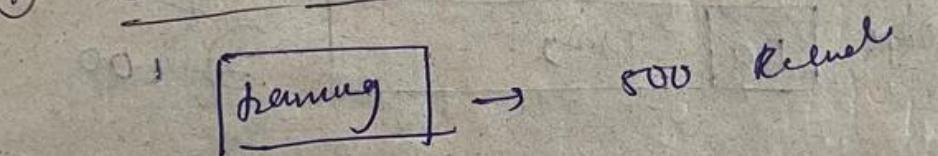
Type of cross validation



85% 92% 78%

Type of CV

① Leave One Out CV (LOOCV)



desidev Records T1 complexity of having mod 1

② Desadus

Overfitting \rightarrow training $\uparrow\downarrow$ acc \rightarrow newest date \rightarrow acc $\downarrow\downarrow$
 validation acc $\downarrow\downarrow$

③ Leave P out CV

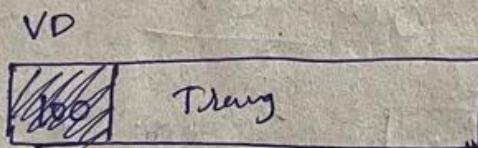
$$P=10, P=20, \\ P=30 \dots$$

④ K-fold Cross Validation

$$k=5, n=500$$

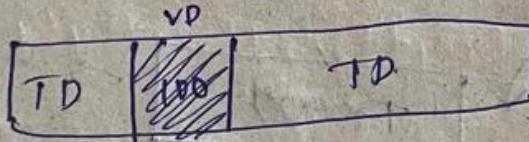
$$\text{Test size} = \frac{n}{k}$$

$$CV=1 \text{ exp.}$$



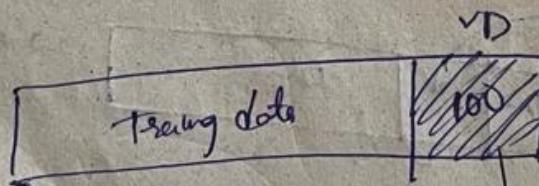
$$= \frac{500}{5} \\ = 100$$

$$\text{exp. 2}$$



any learning,

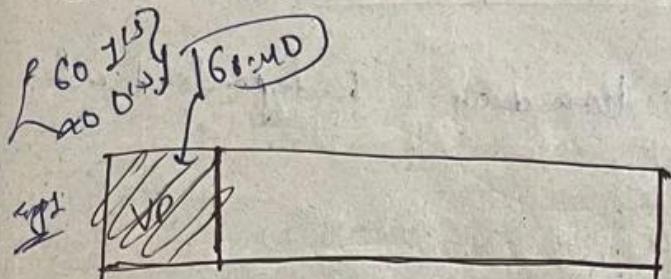
$$\text{exp. 5}$$



dis.

might fallison with
classif. type
like having all ones' (0)
all ones in
validation

④ Shuffled & Fold CV



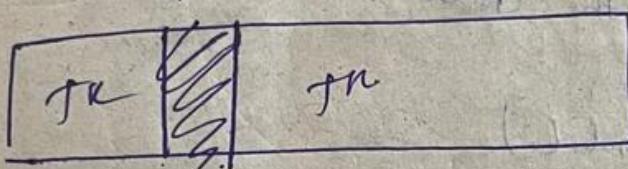
$k=5$

$$\text{fold sys} = \frac{100}{5} = 100$$

'll
val the
bal

→ tries to provide almost
equal proportion validation data

v.D.



⑤ Time Series Cross Validation

produced ~~series~~ seasonal analysis

at steady

Bed 2

↓
good man

time

JAN → DEC

Day 1 Day 2 Day 3 Day 4 ... Day 4-N

val date

Day 4-N

Time series application

for spr Tr 1-3 day | test on day 4

or spr Tr 1-4 | for day 5

then split in 1 ns , in day 6

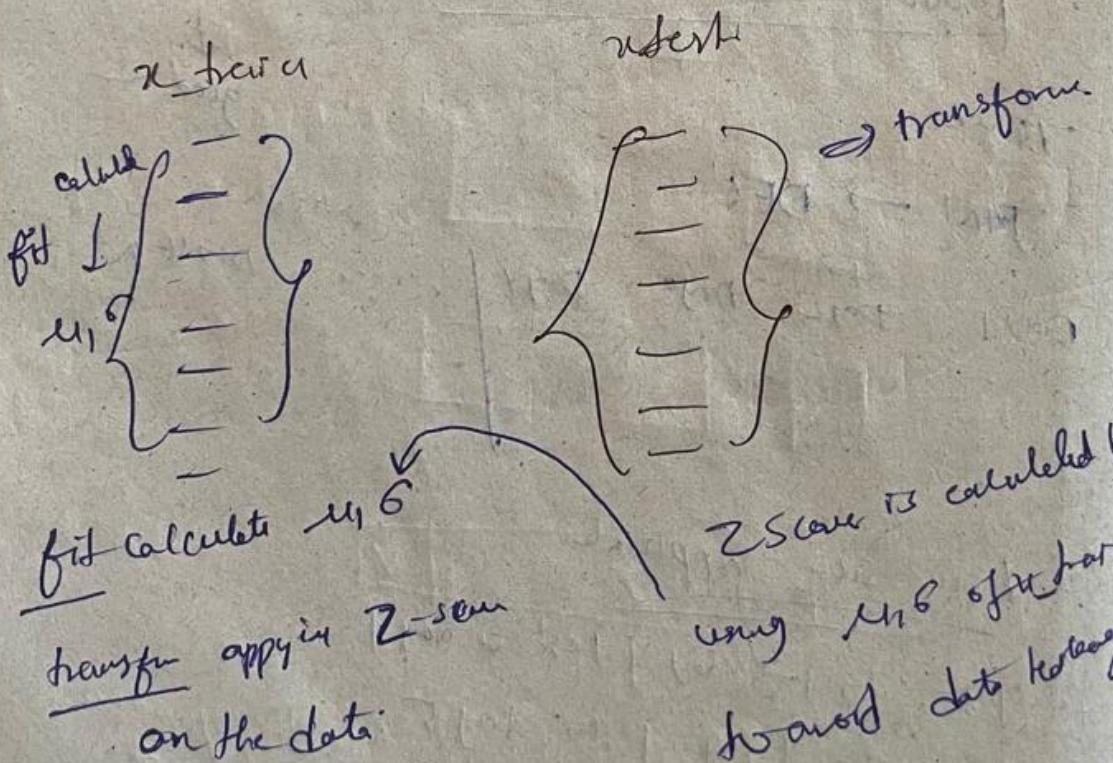
Simple Linear Regression project

- ① Divide the features based on independent and dependent
- ② Train set split of the dataset
- ③ Standardize the data.

scaler = StandardScaler()

scaler.fit_transform(x_train)

scaler.transform(x_test) to avoid data leakage



① Train the model using Cross-Region

⑤ produce on test data

⑥ performance metrics

MAE, MSE, ~~R²~~, adjusted R²

Assumptions

new data point weight 80 kg is given

selected wt = 50 kg : target ([C80])

selected wt

regress predicted (selected wt)

assumptions that our model is performing good

assumption: plot obs vs ytest, ypred
Should be linear

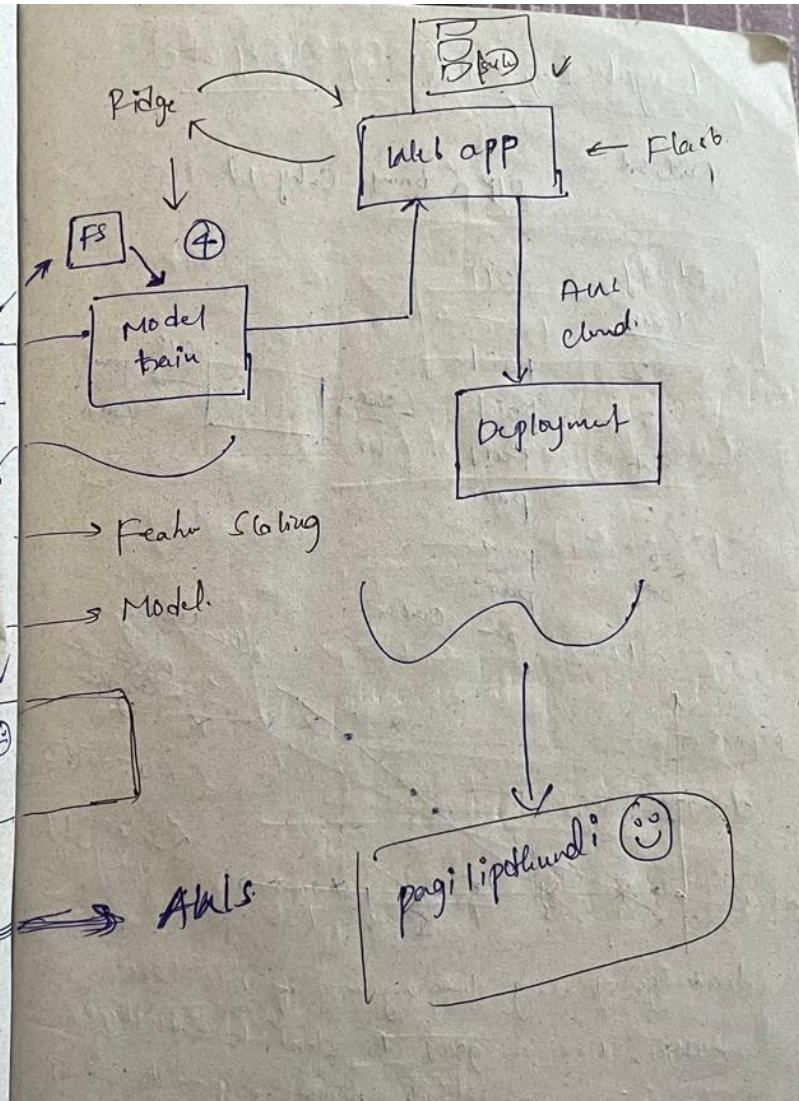
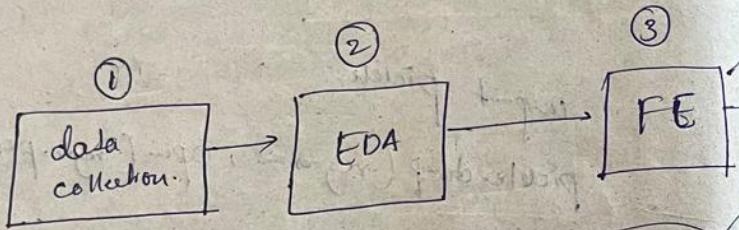
assumption: residuals = ytest - ypred

sns. distplot(residuals) should form
normal distribution

assumption: plot obs vs ypred and residuals

End-to-end ML project implementation

ML project Lifecycle



Model pickling

Pickle in python is a module used to save (Serialize) and load (de-serialize) objects, such as trained machine learning models, into a file so they can be reused later without retraining.

```
import  
with open('model.pkl', 'wb') as file:  
    pickle.dump(model, file)
```

Saving a model:

```
with open('model.pkl', 'rb') as file:  
    loaded_model = pickle.load(file)
```

Loading the saved model

import pickle

pickle.dump (regressor, open('regressor.pkl', 'wb'))

model = pickle.load(open('regressor.pkl', 'rb'))

model.predict (X_test)

if i want to pickle StandardScaler module

import pickle

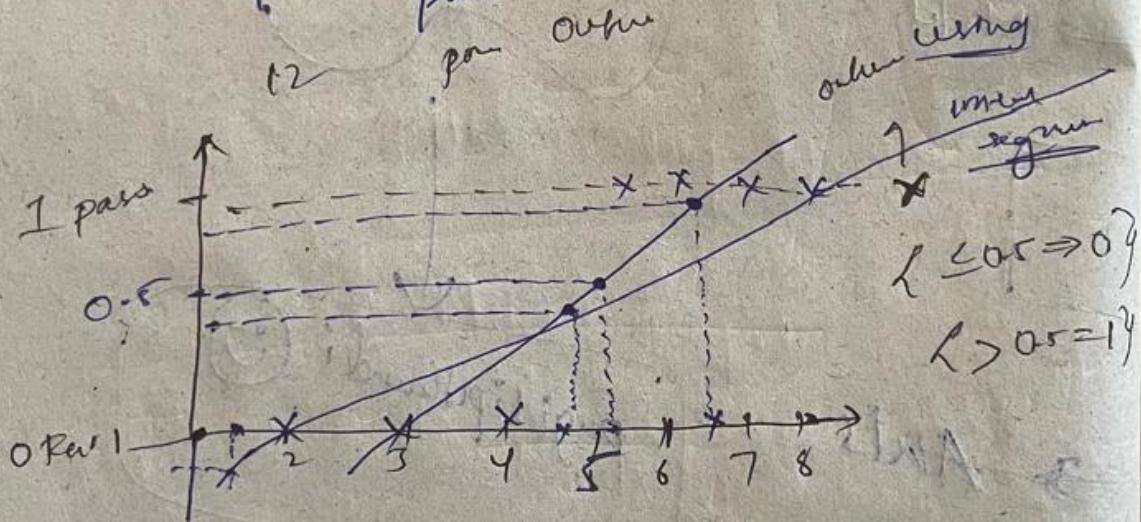
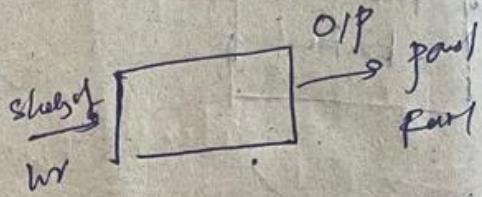
pickle.dump (scaler, open('scaler.pkl', 'wb'))

Logistic Regression (Binary Classification)

Dataset

OIP (Binary Category)

Study hours	OIP
2	Fail
3	Fail
4	Fail
5	Pass
6	Pass
7	Pass
8	Pass
12	Outlier



Disadvantage of using linear regression if we have outlier, line is going to bend towards the outlier, even though person is going to sleep 5 hr he is going to fail.

$$L > 1 \text{ and } y \leftarrow 0$$

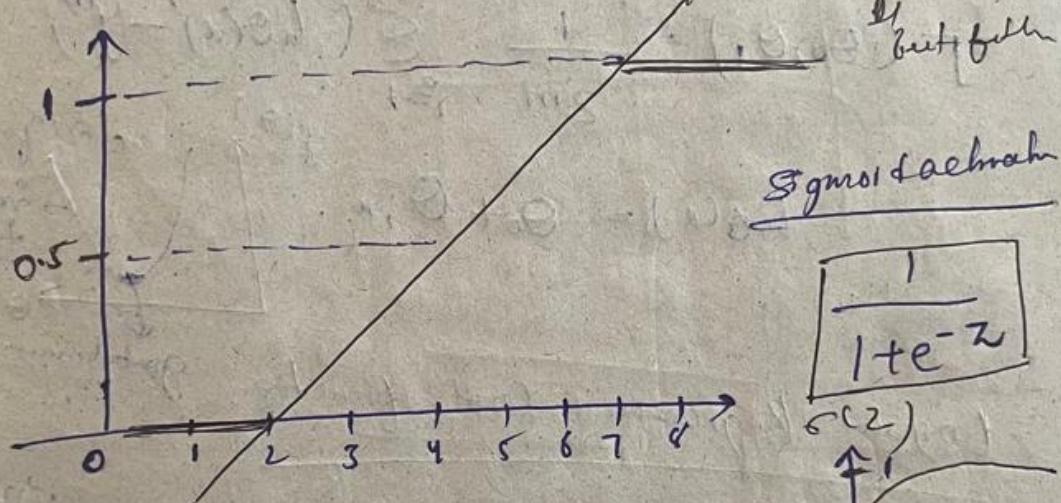
Why we cannot use Linear Regression for classification?

- ① Outlier \hat{y} Best fit line change
- ② y_1 and y_0 Squash line

How Logistic Regression solves Classification

problem

$$\ln(\hat{y}) = \theta_0 + \theta_1 x_1$$

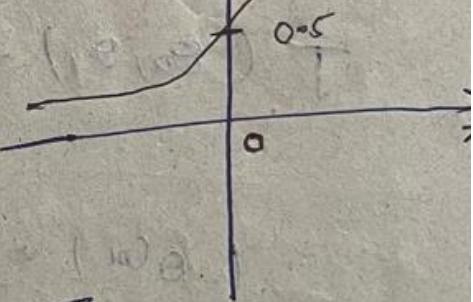


Sigmoid function

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

irrespective of any z values
values are going to ranging b/w

0 to 1



$$z \geq 0$$

$$\sigma(z) \geq 0.5$$

Since we are squashing the line.
if we are going to get any
outlier it is fixed as 1

$$h_{\theta}(x) = \sigma(\theta_0 + \theta_1 x)$$

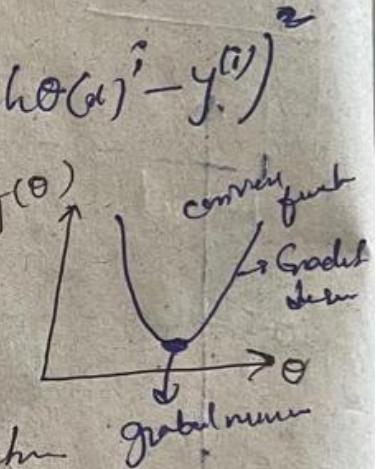
$\sigma = \frac{1}{1+e^{-z}}$

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

Linear Regression Cost function

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

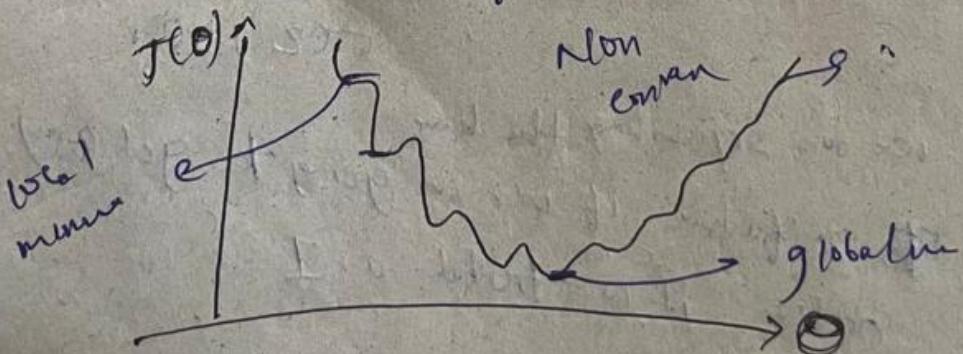
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



Logistic Regression Cost function

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

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



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

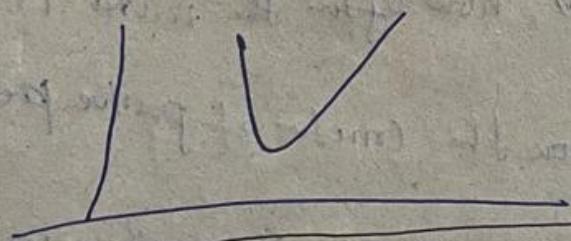
$$h(x)^i = \frac{1}{1+e^{-z}} \quad z = \theta_0 + \theta_1 x$$

lets denote $\text{cost}(h(x)^i, y^{(i)})$

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

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

By using this we are going to get
cool convex function



$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left(y^{(i)} \log(h(x)^i) - (1-y^{(i)}) \log(1-h(x)^i) \right)$$

minimize cost function $J(\theta_0, \theta_1)$ by changing

$$\theta_0 + \theta_1$$

Convergence algorithm

Repete

h

$$J = \theta_0 + \theta_1$$

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

Q P Y

Regression model

R-squared

adjusted R-squared

accuracy: just tells how often the model is right

precision: focus on the concept of positive prediction

recall:

focus on find all the actual posites

also

precision

Balanced precision and recall to give a score

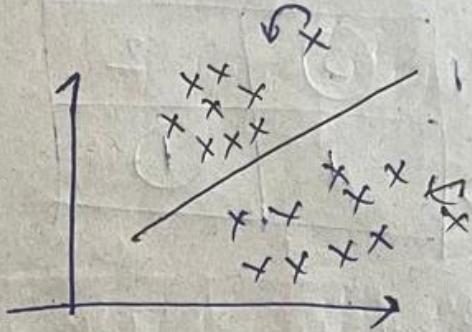
now

Performance Metrics

which are specifically used in
binary and multiclass classification

Topics to be covered:

- ① Confusion matrix
- ② Accuracy
- ③ precision
- ④ Recall
- ⑤ F Beta score



Dataset		0	1	predic. model
x_1	x_2	y	\hat{y}	
-	-	0	1	
-	-	1	1	
-	-	0	0	
-	-	1	1	
-	-	1	1	
-	-	0	1	
-	-	1	0	

Confusion matrix: In ~~the~~ binary class classifier
~~it is~~ is 2×2 mat.

	1	0	→ actual values
1	(3)	2	diagonal are correct ones
0	1	(1)	

	1	0	actual values
1	True positive	False positive	NPV error
0	False negative	True negative	

predicted value → Type 2 error

actual value → Type 1 error

② accuracy: $\frac{TP+TN}{TP+FP+FN+TN}$

$$= \frac{3+1}{7} = \frac{4}{7} =$$

Data set Binary classifier
 $\hookrightarrow 1000$ datapoint

$\rightarrow 900 - 1$	}
$\rightarrow 100 - 0$	}

Unlabelled Data

90% accuracy thus is not true

to avoid that we are going to use
precision, recall (not to use accuracy)

$$\textcircled{1} \text{ Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

} out of all the actual values how many are correctly predicted

	1	0	→ actual
1	TP	FP	
0	FN	TN	
↓			

probs

$$\textcircled{2} \text{ Recall: } \frac{\text{TP}}{\text{TP} + \text{FN}}$$

	1	0	actual val
1	TP	FP	
0	FN	TN	

} out of all the predict values how many are correctly predicted.

useless

spam classification

Mail → spam } good

Not mail → spam

mail → not spam } chance

Not mail → spam

in this case rather we should use
false positive or False negative

↳ This case for spam
review

		spam	notspam	<u>sachet</u>
spam	1	TP	FP	man → no spm model → spm
	0	FN	TN	

↓
predict

in this case we are going to use

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

FP↓

like case:

To predict whether person has diabetes or not

Blunder { Truth → diabetic

not blunder { Model → does not have diabetes

Passenger to be checked again {
not blunder { Truth → Not diabetic
Model is diabetic

Labels No doubt Certain

	TP	FP
	(FN)	TN

↓
we care of disease.
We should reduce false negative

In this case we are going to use.

$$\text{relat} = \frac{TP}{TP + FN}$$

$FN \downarrow \uparrow$

assign: to make the stock market won't crash

on model

reducing $FP \downarrow$ on $FN \downarrow$

crash	not crash	gather
0	TP	FP
0	FN	TN

predict

With \rightarrow crash } FN
model \rightarrow not crash }
With \rightarrow Not crash } FP
model \rightarrow crash }

$$F - \text{Beta score} = \frac{(1+\beta)}{\beta}$$

true + false
true + false

① If FPF FN fall are equal

$$\beta = 1$$

$$F_1 \text{ score} = 2 \times \frac{\text{true} * \text{false}}{\text{true} + \text{false}}$$

true
false

② If FP is more culprit PN

$$\beta = 0.5$$

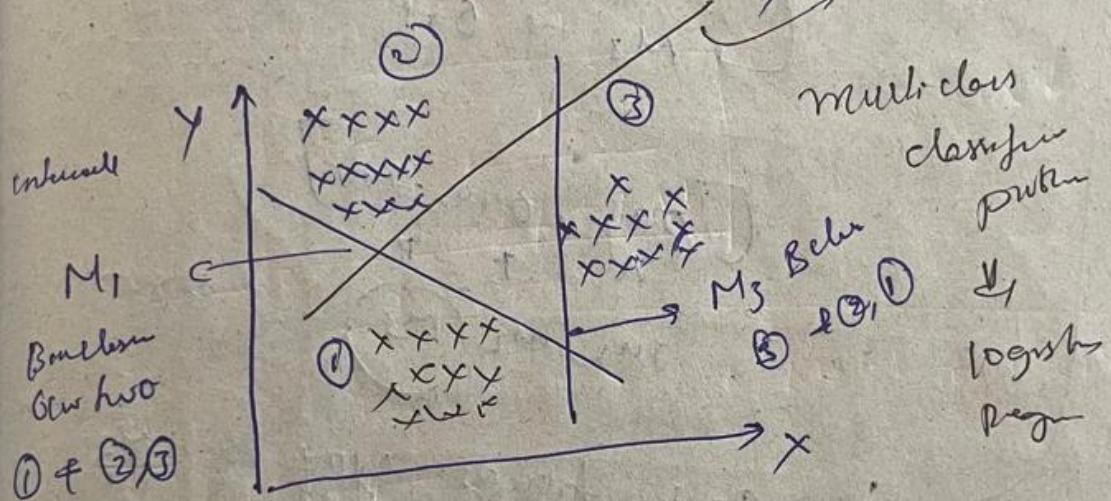
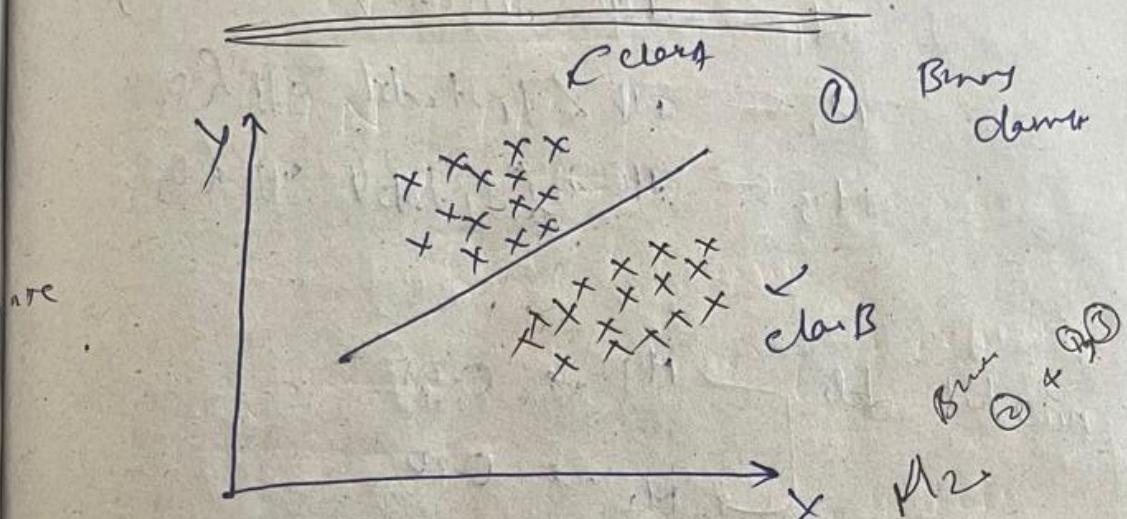
$$F_{0.5} \text{ score} = (1+0.25) \frac{P * R}{P + R}$$

③ If FN > PP

$$\beta = 2$$

$$F_2 \text{ score} = (1+4) \frac{P * R}{P + R}$$

Logistic Regress OVR (One Versus Rest)



f_1	f_2	f_3	01P	O_1, O_2, O_3
-	-	-	O_1	1 0 0
-	-	-	O_2	0 1 0
-	-	-	O_3	0 0 1
-	-	-	O_1	1 0 0
-	-	-	O_3	0 0 1
-	-	-	O_2	0 1 0

Model

$M_1 \leftarrow \text{exp} L_{f_1, \text{initial}} \circ \text{IP}(L_1)$

$M_2 \leftarrow \text{exp} L_{f_2, \text{initial}} \circ \text{IP}(L_2)$

$M_3 \leftarrow \text{exp} L_{f_3, \text{initial}} \circ \text{IP}(L_3)$

new last obs

$$\begin{cases} M_1 \rightarrow 0.25 \\ M_2 \rightarrow 0.20 \\ M_3 \rightarrow 0.05 \end{cases}$$

$[0.25, 0.20, 0.05]$

m_1, m_2, m_3 high probability

O_3 is failure of IP for M_3

Logistic Regression Implementation

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

$X, Y = \text{make_classification}(\text{n_samples}=1000, \text{n_features}=10,$
 $\text{n_classes}=2, \text{random_state}=42)$

from sklearn.model_selection import train_test_split

~~def~~ $\text{logit}(x)$
 $x_{\text{train}}, y_{\text{train}}, y_{\text{test}}, y_{\text{true}}, y_{\text{pred}} = \text{train_test_split}(X, Y, \text{test_size}=0.30,$
 $\text{random_state}=42)$

from sklearn.linear_model import LogisticRegression

~~def~~
 $\text{logit} = \text{LogisticRegression}()$

$\text{logit}.fit(X, y_{\text{train}})$

$y_{\text{pred}} =$

~~def~~
predict_proba

from sklearn.metrics import accuracy_score, classification_report

hyperparameter tuning of logreg model
using [Grid Search hyperparameter]

model = LogisticRegression()

penalty = ['L1', 'L2', 'elasticnet']

Calpha = [100, 10, 1.0, 0.1, 0.01]

Solver = ['liblinear', 'newton-cg', 'sag', 'saga']

for sklm-model input Sklearn Models

cv = StratifiedKFold(5)

param = dict (penalty=penalty, C=Calpha,
solver=solver)

for skbm-model like imp GridSearchCV

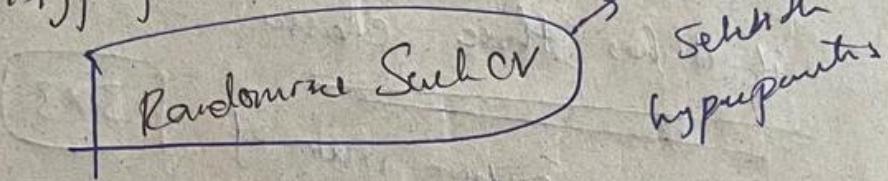
grid = gridSearchCV (estimator, param_grid=
param, scoring='accuracy'; cv=cv,
n_jobs=-1)

grad. fct (x_1 , x_2)

performance

from stochastic input averyn, cofinants
classification

① Hypopanel fnc using



from stochastic model inputs Random Sub CV.

model = Logistic Regul)

random = Rodriguez-Suever (estim = well,

param = param, cov, sing, diag)

Logistic Regression for multiclass classification

every thing is similar to binary class
classification

just

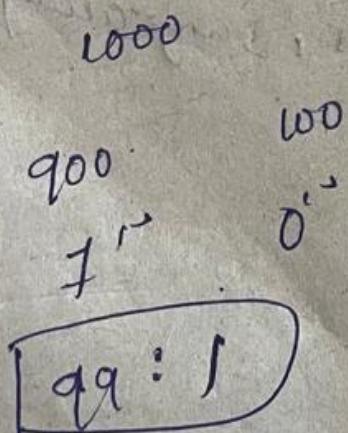
$$\text{Logistic} = \text{Logit}_k \text{ (multi-class = "OVR")}$$

One vs Rest

if it has three classes

we get confusion matrix of 3×3

Logistic Regression For imbalanced Datasets



There is a very important part
in logistic regression that is
class weights
 \downarrow
helps in assigning more
importance to less
no. of feature available
in the dataset

By using class weight we are going to handle the imbalance details

By adding

① penalty

② c-values

③ solver

④ classwt

$\text{classwt} = [L_0: w, 1:y) \text{ for } w \in [1, 10, 50, 100] \text{ for } y \in [1, 10, 50, 100]]$

$[L_0: 1, 1: 1]$

$L_0: 1, 1: 10$

$L_0: 1, 1: 50$

$L_0: 1, 1: 100$

$L_0: 100, 1: 10$

$L_0: 100, 1: 50$

Now just perform the same process as

grid Search CV.

Logistic Regression with ROC Curve and ROC AUC score

In logistic regression by default threshold value is set to be 0.5 if $\pi_{i0} > 0.5 \rightarrow 1$
 $< 0.5 \rightarrow 0$
but this threshold value can ~~be~~ change to problem statement to proper submit.
Some times $> 0.5 \rightarrow 1$, $> 0.7 \rightarrow 1$ ~~is~~ find by domain expert.

for soft. metrics most ROC curve
from soft. metrics import ROC-AUC score

Important points in Logistic Regression

penalty: Control the type of regularization

Options:

- 'l1' → Lasso (Lasso makes some coefficients zero)
- 'l2' → Ridge (Ridge makes coefficients non-zero)
- 'elasticnet' → min of both l1 and l2 right

Class weights

useful for handling imbalanced data (when one class has more data than the other)

Tot. It helps @ detect when to stop training, especially for large dataset

② C : Controls the strength of regularization

small value of C apply L1 regularization

large value of C apply L2 regularization

③ solver:

This defines the algo the model will use
to find the best parameters

options

'liblinear' → good for small datasets, α_{reg} , L_1 or
 L_2 penalty

'lbgf': good for large datasets, only supports L_2
penalty

'saga': supports both L_1 and L_2 penaltys
handles miss-class. datasets

④ max_iter:

scoring:

sets the maximum no of iterations the solver will
go through to find the best model

default 100

⑤ random_state: controls the randomization of
reproducibility

Support Vector Machines ML algorithm

① SVC

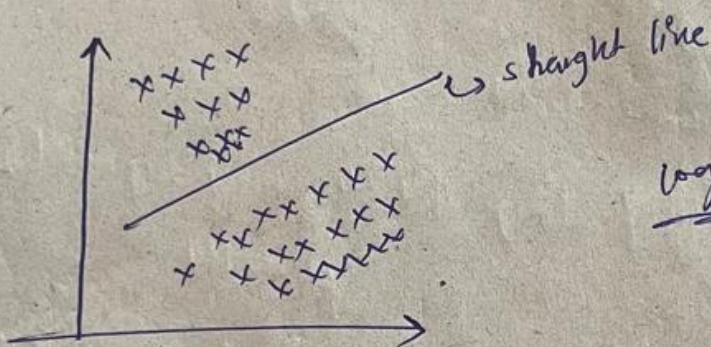
↳ (Support vector classifier)

↓
used to solve both classifier
and regression.

② SVR

↳ (Support Vector Regressor)

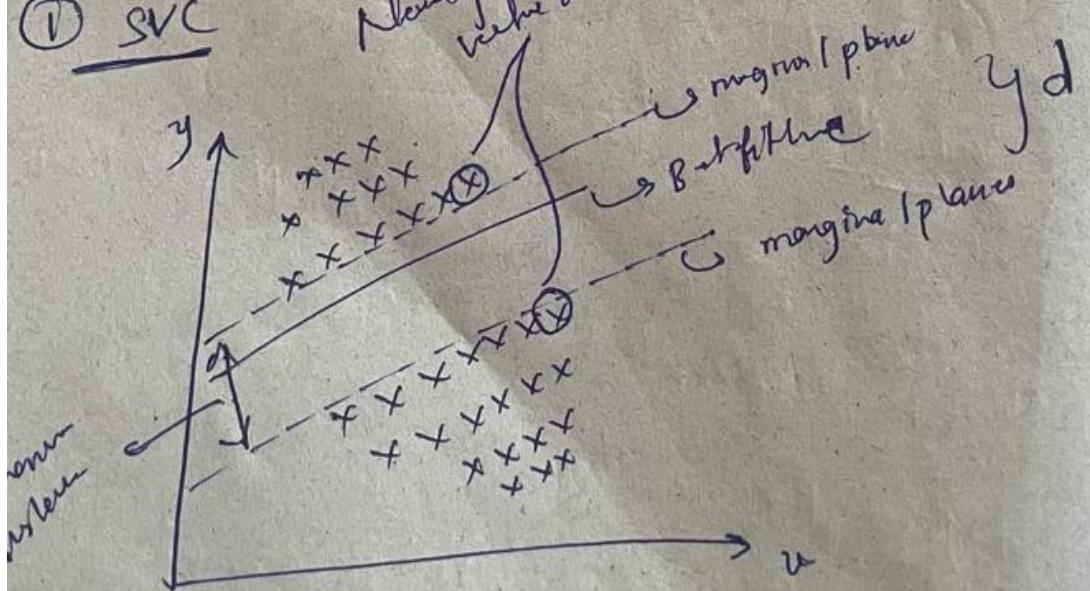
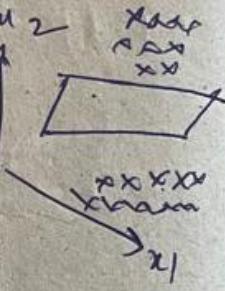
logistic regm

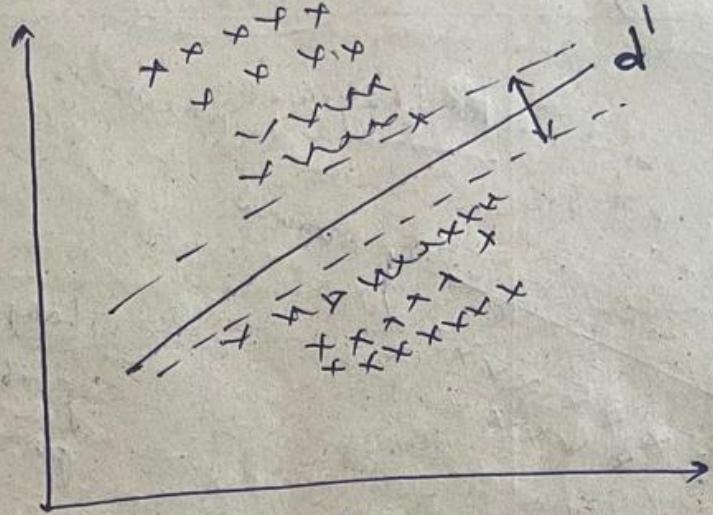


① SVC

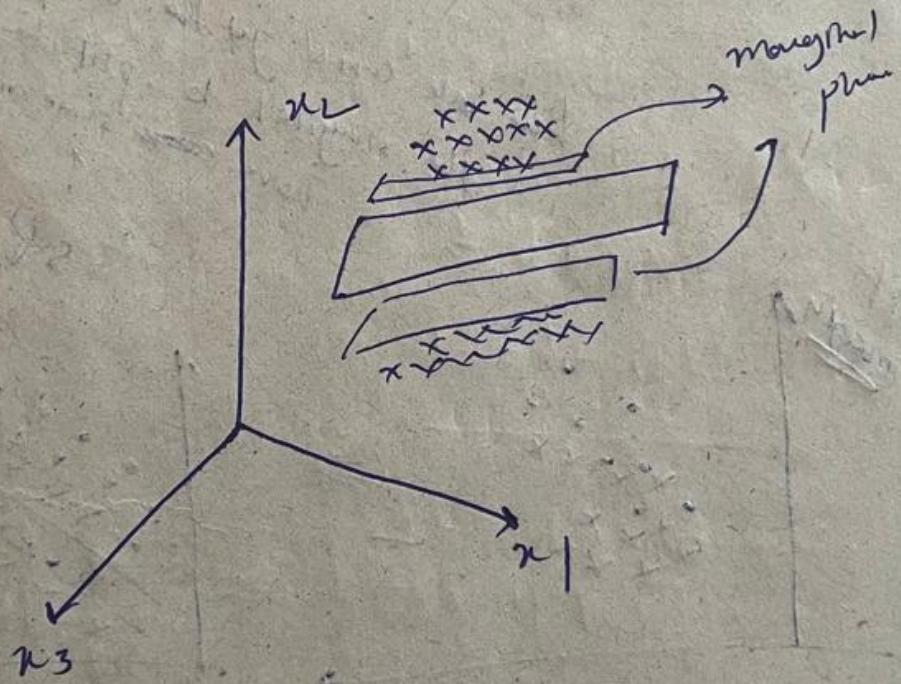
Non-linear to linear
value are support vector

logm

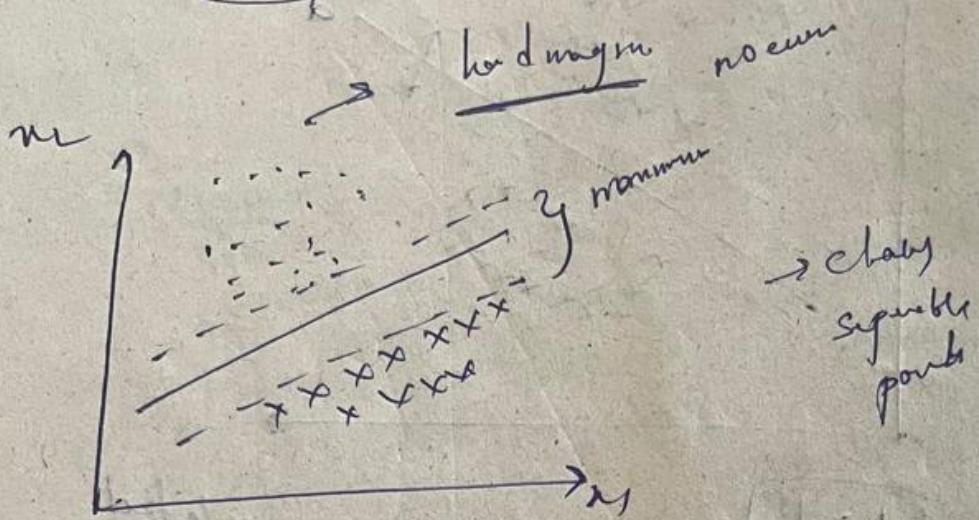




$d > d'$
so the marginal plane with d is selected

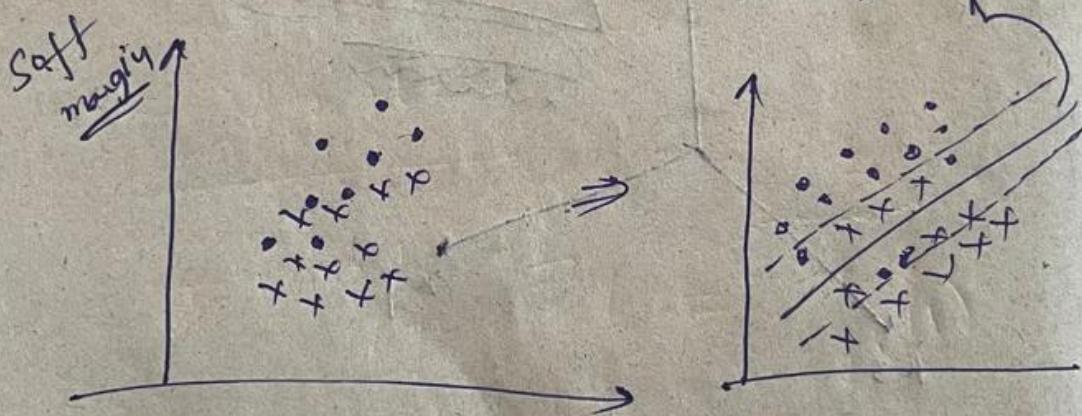


Soft Margin and Hard Margin in SVM



here we can't get the best fit line along the margin to split the point, we have to some amount error

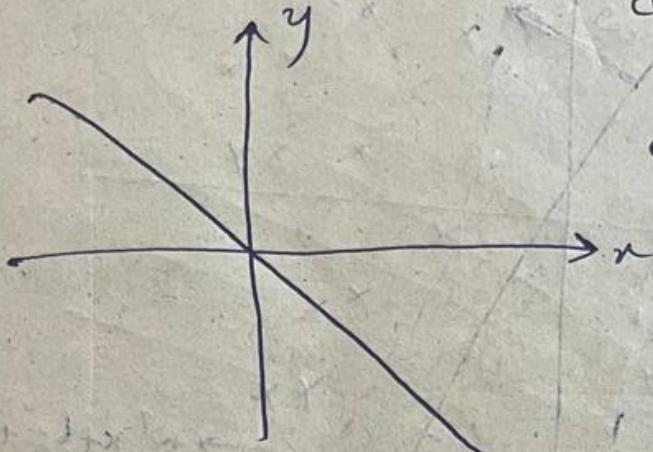
1
soft margin



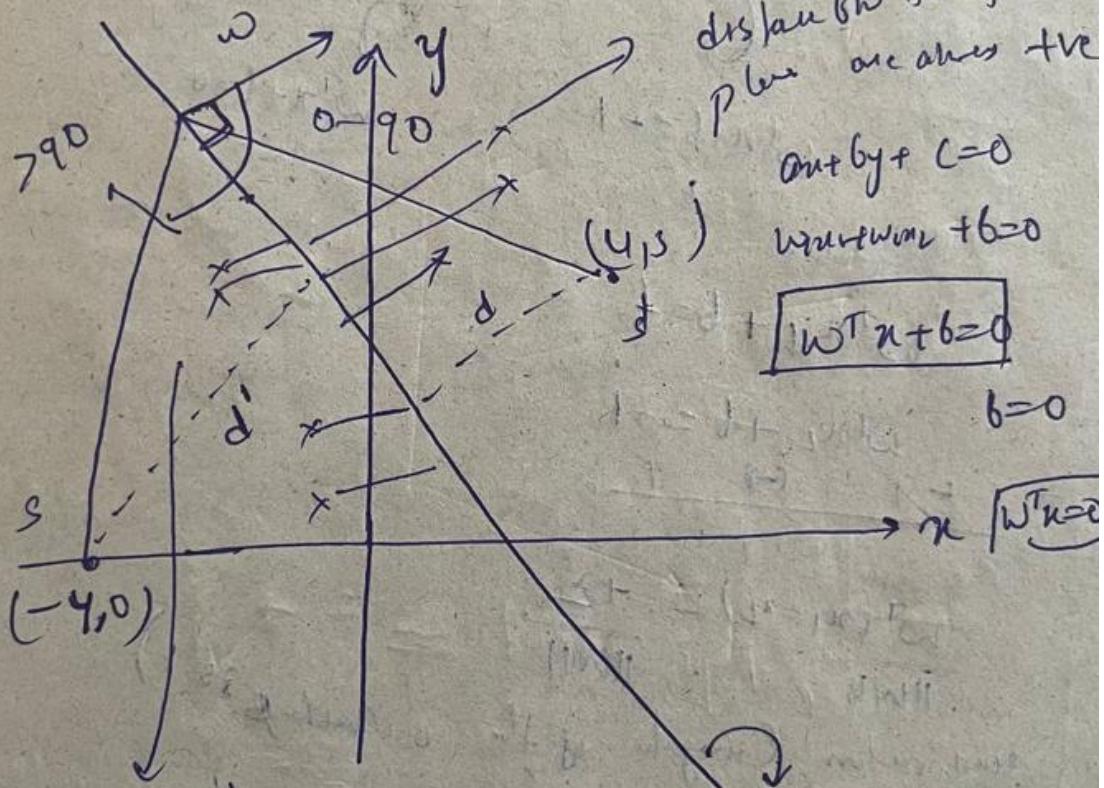
hard margin \rightarrow no error clear separn

soft margin \rightarrow error, some arror error will be there

SVM Marginal



$d \rightarrow -ve$
below the plane
distance
above the plane



$$w^T x + b = 0$$

distance from the plane
across the
margin $= d$

$$w^T x + b = 0$$

$$w^T x + b = 0$$

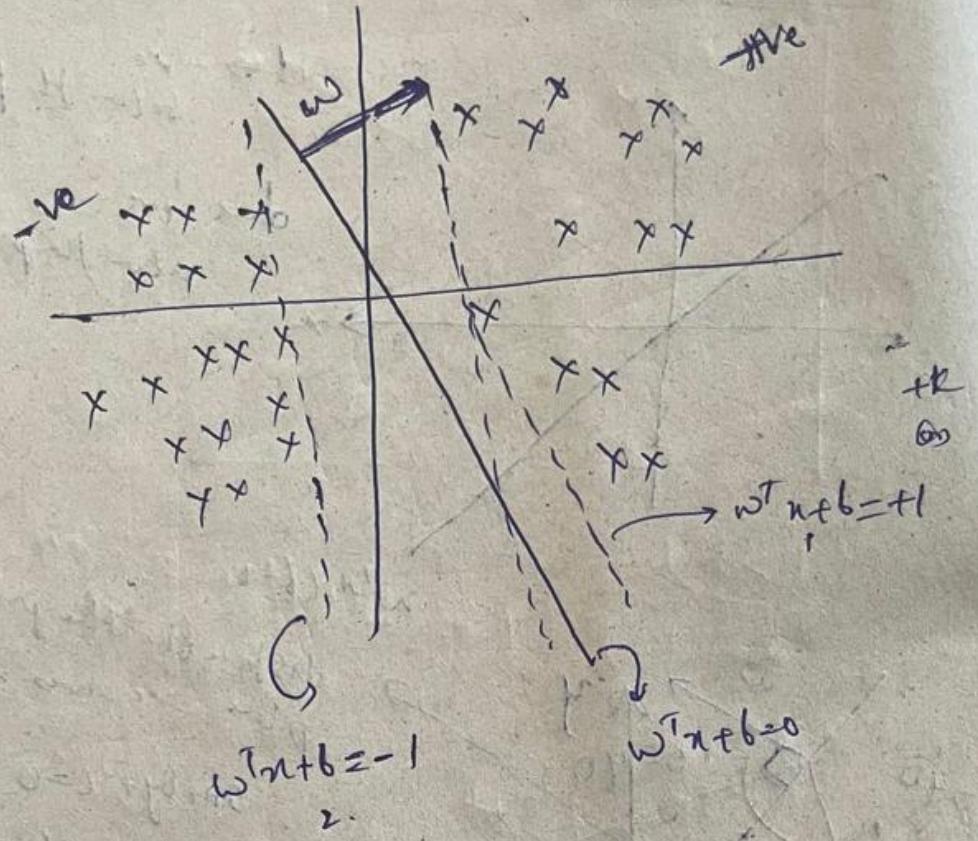
$$w^T x + b = 0$$

$$b = 0$$

Support vectors
separating plane
between classes
margin width $= 2d$

$$w^T x + b = 0$$

margin width $= 2d$



$$w^T x_1 + b = 1$$

$$w^T x_2 + b = -1$$

$\leftarrow 1 \rightarrow -1$

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

unit vector (length of the unit vector β is 1)

Cost function

minimize

$$w^T b$$

$$\frac{2}{\|w\|}$$

\Rightarrow Distance (the margin)
plane

constant such that $y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$
correctly classified point

for all the comes part

contd

$$y_i \times (w^T x + b) \geq 1$$

max margin
 w, b

$$\frac{2}{\|w\|} \Rightarrow \min_{(w,b)} \frac{\|w\|}{2}$$

cost funct of SVM (src) etc

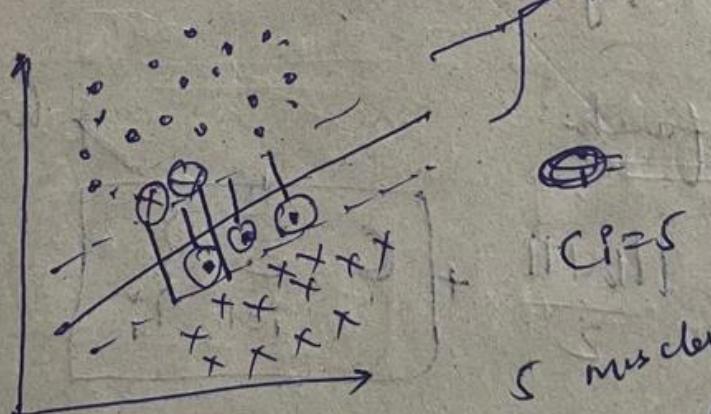
Min $\frac{\|w\|}{2}$ + $\sum_{i=1}^n C_i \xi_i$ hinge loss

soft margin

Margin = distance from the decision boundary to the nearest data points on either side.

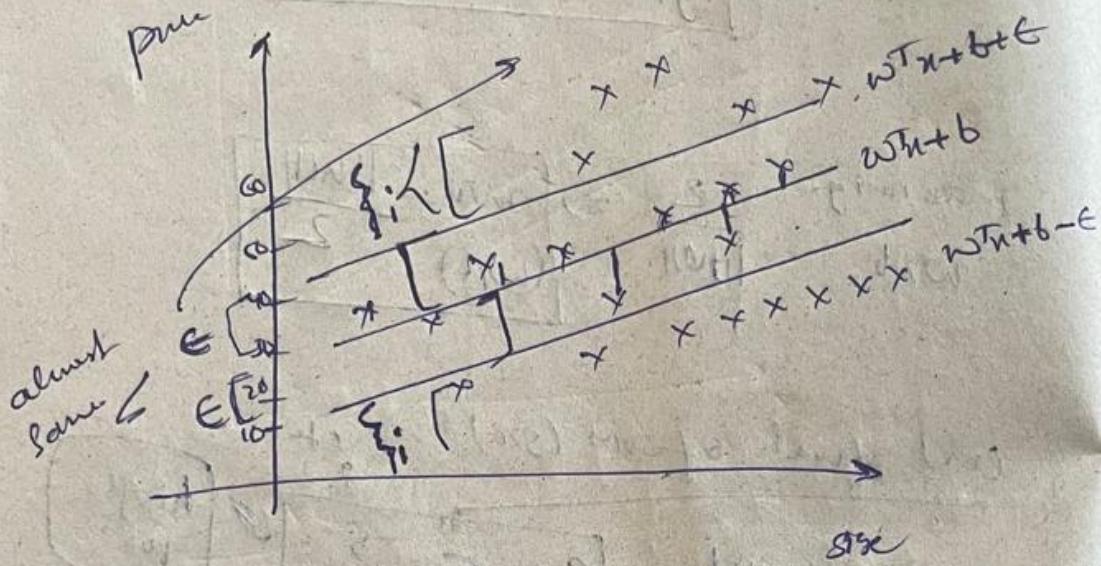
C_i is a weight vector for the i^{th} data point.

Sum of the distances of the misclassified points from the margin plane.



This is related to soft margin

Support Vector Regression



cost function

$\text{Min}_{w,b} \|w\|$ → minimizing $\frac{\|w\|^2}{2}$, by choosing

$$\text{w}, b$$

value

Constant: b hull Predict price
 r

$$|y_i - w^T x_i| \leq \epsilon + \xi_i$$

cost function:

$$\text{Min}_{w,b} \frac{\|w\|^2}{2}$$

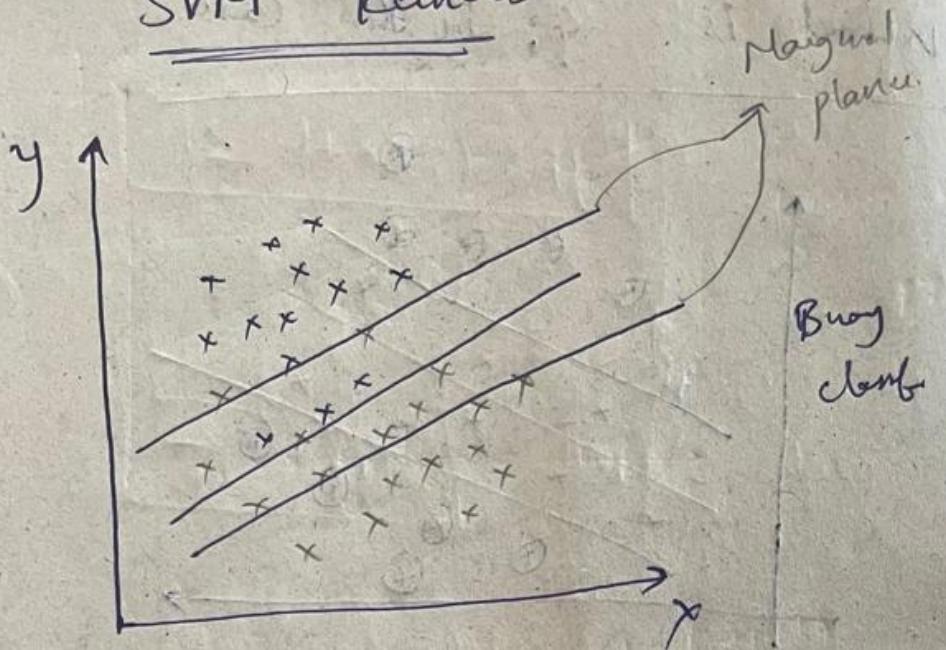
$$+ C \sum_{i=1}^n \xi_i$$

hinge loss

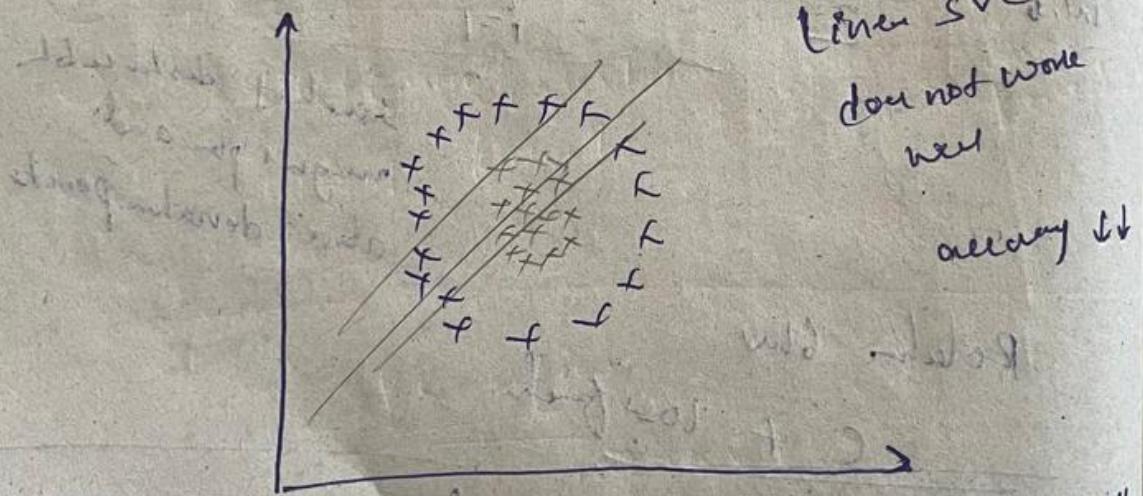
$\epsilon \rightarrow$ margin error

$\xi_i \rightarrow$ error above the margin

SVM Kernels



Linear SVC

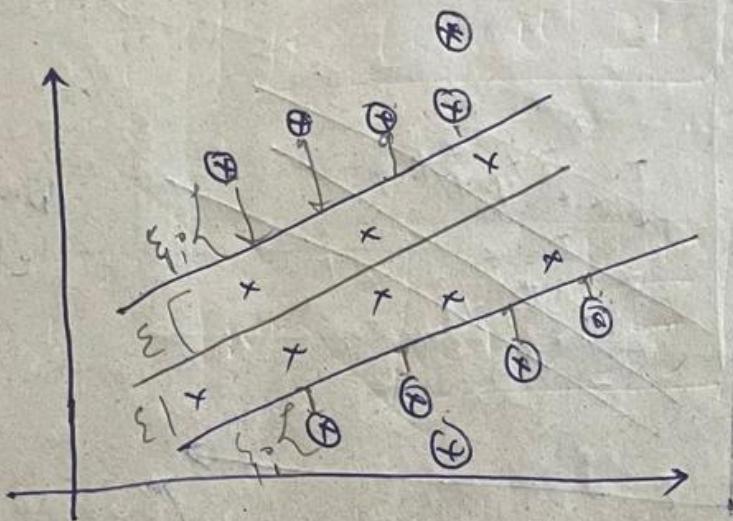


When the data points are not linearly separable
then we can use SVM kernels

Linear SVC
does not work
well

accuracy ↓

What is ξ_i in this case



$$\text{cost funct} \quad \text{hyp-param.}$$

$$\min_{w, b} \frac{\|w\|}{2} + C \sum_{i=1}^n \xi_i$$

Since ξ_i of distance between margin plane and above decision points

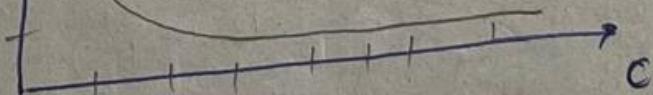
Relaxation func

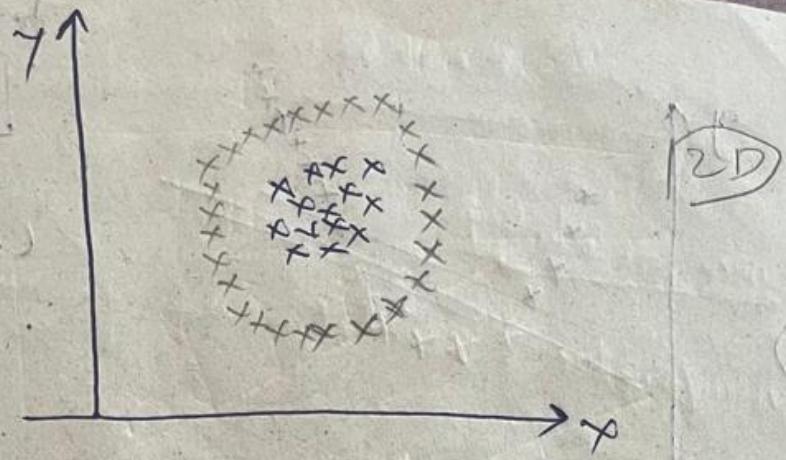
$C + \text{loss funct}$

loss
funct

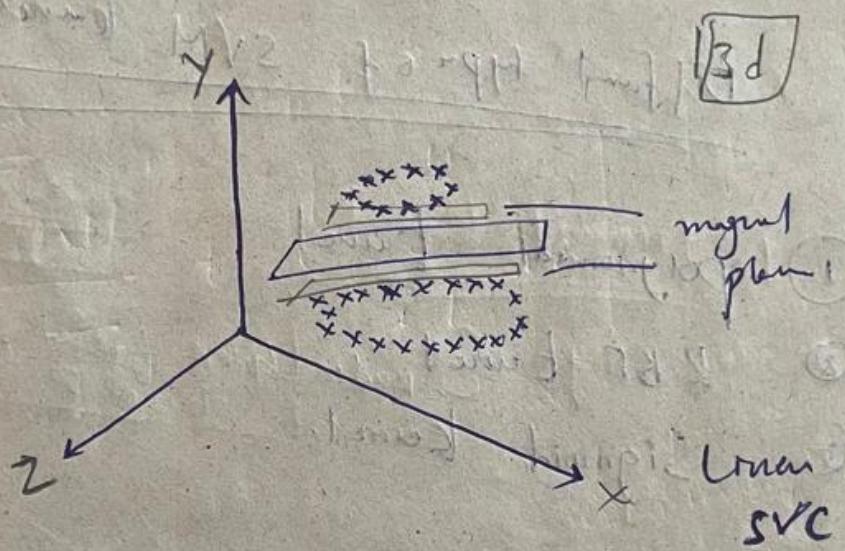
$$C \alpha \eta \quad \text{loss } \downarrow \downarrow$$

Inversely
proportional



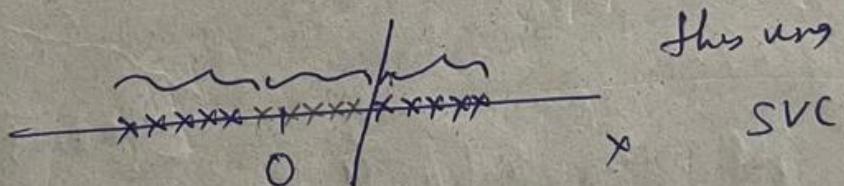


\Downarrow applying transform
(math not fun)



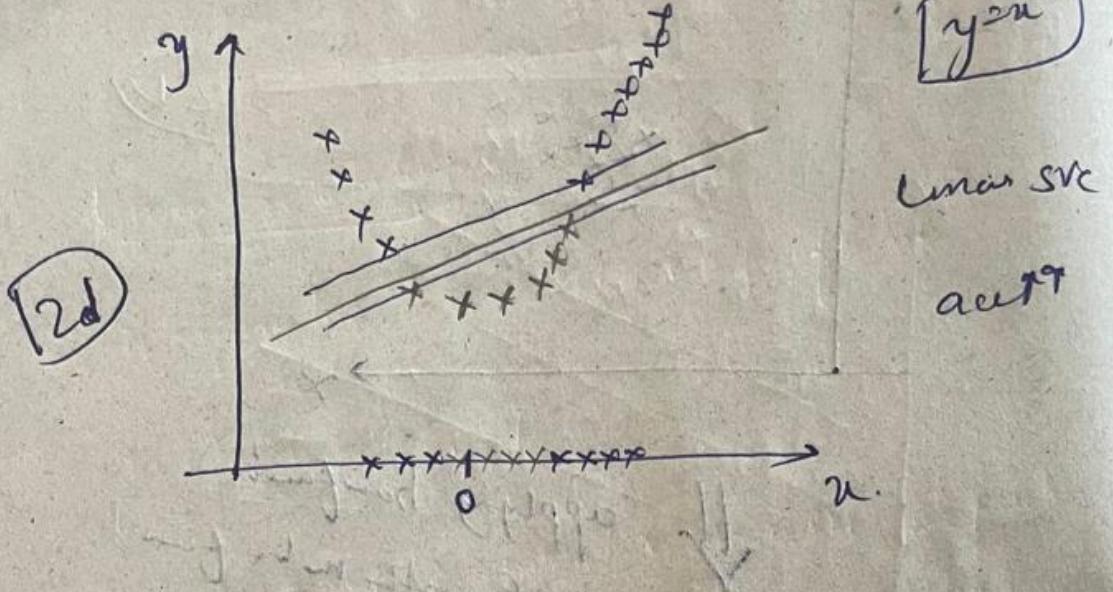
Data set 1D

we cannot see



transform

$$y = x^2$$



Different types of SVM kernels

- ① polynomial kernel
- ② RBF kernel radial basis function or Gaussian kernel
- ③ Sigmoid kernel.

SV Classifiers

Implementation

When to use what kernels

polynomial kernel for non linear data with polynomial relationships

RBF kernel: for complex, non-linear data without clear patterns

Sigmoid kernel: used when the data has smooth behavior to neural network models

there is when kernels helps to map them complex patterns into higher dimensional space where such patterns can be easily separated

$$y = u^v \text{ parabolic}$$

$$y = u^3 \text{ cubic}$$

8. linear make classifier

Linear relationship

- no. of features = total no. of feature
- no. of classes - parameter = 1
- class - sep = weight value make Separation more distinct

polynomial kernel → Paraffin

intensity Create

$$k(u_i y_i) = (u_i^T y_i + c)^d$$

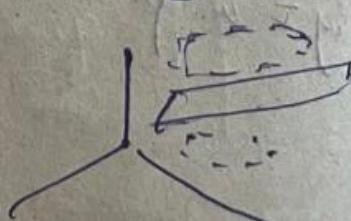
u_i square

y_i square

$u_i \neq u_j$

$$rbf \rightarrow k(u_i, T_i) = e^{-\frac{\|u_i - T_i\|^2}{2\sigma^2}}$$

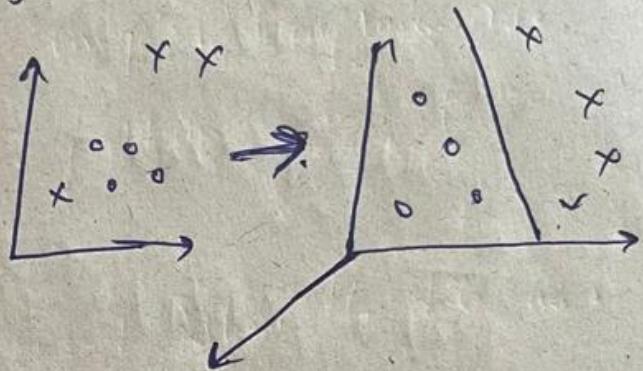
Initial
Paraffin



Mathematical transformation used in the kernel to change to higher dimensions.

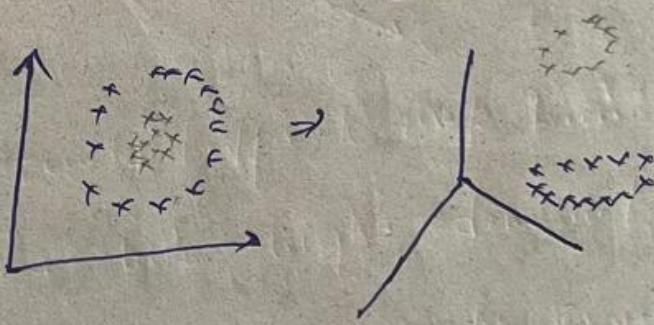
polynomial kernel

$$K(u_i, u_j) = (u_i^T u_j + c)^d$$



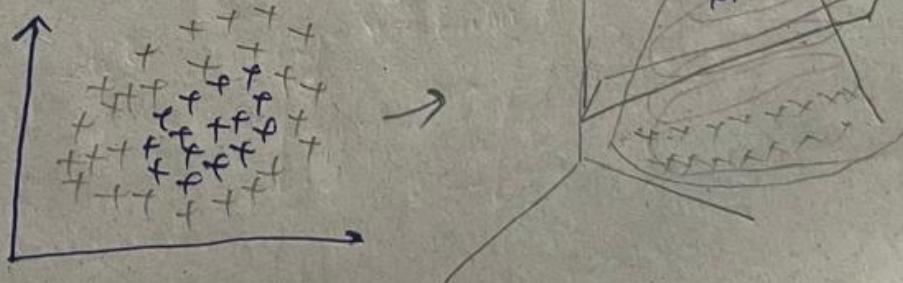
RBF kernel

$$k(u_i, u_j) = \exp\left(-\frac{|u_i - u_j|^2}{2\sigma^2}\right)$$

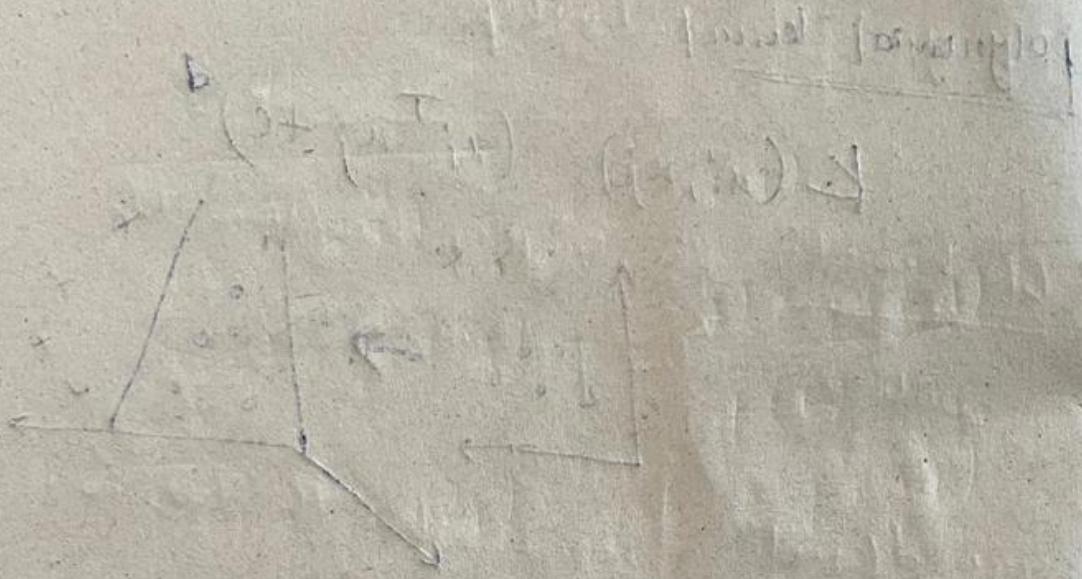


Sigmoid kernel

$$k(u_i, u_j) = \tanh(\alpha u_i^T u_j + c)$$



Support Vector Regression Implement



Linear Function



(x_1, y_1) and (x_2, y_2)

Genue way.

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

Bayes theorem

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

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

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

Bayes
theorem.

$$\boxed{P(A|B) = \frac{P(A) * P(B/A)}{P(B)}}$$

$P(A|B) \rightarrow$ probability of event A given B has occurred.

$P(A) \rightarrow$ probability of event A

$P(B) \rightarrow$ probability of event B

$P(B/A) \rightarrow$ probability of event B given A has occurred

Naive Baye's Algoithmn (classification)

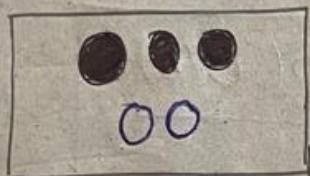
- ① probability ↳ Binary class and multi-class classifier
- ② Bayes theorem

Independent Events:

rolling a dice $\{1, 2, 3, 4, 5, 6\}$

$$pr(1) = \frac{1}{6}, \quad pr(2) = \frac{1}{6}, \quad pr(3) = \frac{1}{6}$$

Dependent Events:



What is the probability of removing a orange marble and then a yellow marble?

$$p(o) = \frac{3}{5} \rightarrow 1^{\text{st}} \text{ event} \rightarrow \text{Orange marble has been removed}$$

$$p(y) = \frac{2}{4} \rightarrow 2^{\text{nd}} \text{ event} \rightarrow \text{removed the yellow marble}$$

$$p(o \text{ and } y) = p(o) \times [p(y/o)] = \frac{3}{5} \times \frac{1}{2} = \boxed{\frac{3}{10}}$$

↓
Conditional probability

probability of an event considering a specific event has already occurred

how bayes' theorem is going to be used in
ml.

The algo that specifically uses bayes' theorem
is called Naive Bayes'

flip face	+ Depressed
x_1	y
x_2	y_1
x_3	y_2
-	y_3
-	y_4
-	y_5
-	y_6

thus we can write as

$$P(y | (x_1, x_2, x_3)) = \frac{P(y) * p(x_1, x_2, x_3) / y}{p(x_1, x_2, x_3)}$$

$$\frac{p(y) * p(x_1 | y) * p(x_2 | y) * p(x_3 | y)}{p(x_1) * p(x_2) * p(x_3)}$$

for new test data

$$pr(\text{Yes} | (x_1, x_2, x_3)) = pr(\text{Yes}) * p(x_1 | \text{Yes}) * p(x_2 | \text{Yes}) * p(x_3 | \text{Yes})$$

$$= 0.60 \quad \text{Yes} \checkmark \frac{p(x_1) * p(x_2) * p(x_3)}{p(x_1) * p(x_2) * p(x_3)}$$

$$pr(\text{No} | (x_1, x_2, x_3)) = \frac{p(\text{No}) * pr(x_1 | \text{No}) * pr(x_2 | \text{No}) * pr(x_3 | \text{No})}{p(x_1) * pr(x_2) * pr(x_3)}$$

$$= 0.40 \quad \text{No}$$

lets solve this doubt

Day	outlook	temp	humidity	wind	play
1	Sunny	hot	high	weak	No
2	overcast	mild	Normal	change	Yes
:					
14	Rain	cold			

outlook		yes	No	$p(E/Y_1)$	$p(E/N_0)$
Sunny	2	3	2/9	3/5	
Overcast	4	0	4/9	0/5	
Rainy.	3	2	3/9	2/5	

temperature		yes	No	$p(E/Y_1)$	$p(E/N_0)$
Hot	2	2	2/9	2/5	
mild	4	2	4/9	2/5	
cold	3	1	3/9	1/5	

play

$p(\text{Yes}) \quad p(\text{No})$

Yes 9 9/14 5/14

No. 5

for new test data:

Test (sunny, hot) \rightarrow O(P)?

$$pr(\text{Yes} / \text{sunny, hot}) \Rightarrow p(\text{Yes}) * pr(\text{sunny} / \text{Yes}) * \\ pr(\text{hot} / \text{Yes})$$

non need to consider it is constant

$$\frac{pr(\text{sunny}) * pr(\text{hot})}{= \frac{9/14 * 2/9 * 4/9}{= \frac{2}{63}} = \boxed{0.031} \text{ Yes}}$$

$$pr(\text{No} / \text{sunny, hot}) \Rightarrow \frac{pr(\text{No}) * pr(\text{sunny} / \text{No}) * pr(\text{hot} / \text{No})}{pr(\text{sunny}) * pr(\text{hot})} \\ = \frac{5/14 * 3/8 * 2/5}{= \frac{3}{35} = \boxed{0.085} \text{ No}}$$

To make it 100% proportion

$$pr(Yes | (\text{Sunny}, \text{hot})) = \frac{0.031}{(0.031 + 0.085)} = 0.27 \\ - 27\%$$

$$pr(No | (\text{Sunny}, \text{hot})) = \frac{0.085}{(0.031 + 0.085)} = 0.73 \\ = 73\%$$

for

outlook

Sunny

temp

hot

O/P

73% \rightarrow they will
not play
tennis

27% they will
play
tennis.

Y

O/No \Rightarrow person is not going to
play tennis.

② Multinomial Naive Bayes With good with direct data
When ever you input data is in text form

dataset) Spame classifier O/P

Spam / Not spam
Spam body.
Not spam

Good job
Thank you have done
HAM

Y
Numerical values (using NLP)
Mr
vector. Natural Language proc

- ① BOW
- ② Tf-IDF
- ③ Word2Vec.

③ Gaussian Naive Bayes Bell curve

If the feature are following Gaussian distribution, then we use Gaussian Naive Bayes.

Variants of Naive Bayes

- ① Bernoulli Naive Bayes
- ② Multinomial Naive Bayes
- ③ Gaussian Naive Bayes

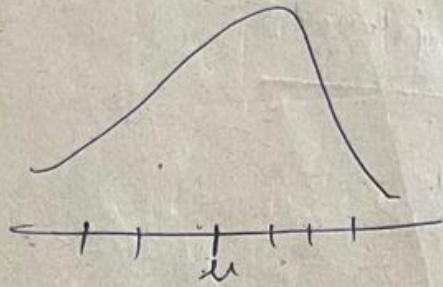
① Bernoulli Naive Bayes!

When our features are following a binomial distribution, then we need to use Bernoulli.

Naive Bayes algo.

Bern \rightarrow 0, 1
 \rightarrow sum / sum

ens	daht	f ₁	f ₂	f ₃	OIP	OIP
yes	pass	M			Y ₄	
no	pass	F			No	Con also
yes	fail	M			Y ₄	be
yes	fail	F			No	multiclass
No	pass	M			Y ₄	clustering



dataset

feature \rightarrow continuous

age	height	weight	Yes/No
25	170	78	
38	160	75	
22	150	60	
24	140	55	

Suppose if we have the dataset that is following Gaussian and Bernoulli then, we need to see which feature is following the main distribution.



Naive Bayes Implementation

- import load_m
- imp train test split
- $X, y = \text{load_m}(\text{scale}=\text{true})$
- $x_h, y_h, y_t = \text{train}()$
- import GaussianNB
- gnb = GaussianNB()
- gnb.fit(x_h, y_h)
- gnb.predict(x_t)
- import metrics, accuracy_score, confusion_matrix

practicing Naive Bayes' on tips data

input tips		input					open
totalbill	tip	Sex	smoke	day	size		time
carb	cut	female	binary	binary	multiple categories	order	
			label	everyday		one	

- ① split the data into
dependent and indept for

$$x = [\dots]$$

$$y = [\text{"time"}]$$

- ② Train the split
xh, yh, ytr, h = ()

- ③ apply label everyday on Sex smoke
be. felc)

- ④ Smoke apply on first dataset

⑤ applying one hand evenly on clay fish

steh.-cupose imp. Cobalt blue.

steh. pr. impd. Orelltet French

Ct. cal. (herb - (C. orchid); orange (dips fruit))
renard - pastel (1.)

⑥ when = cl. for her (when)

when = cl. transfer (when)

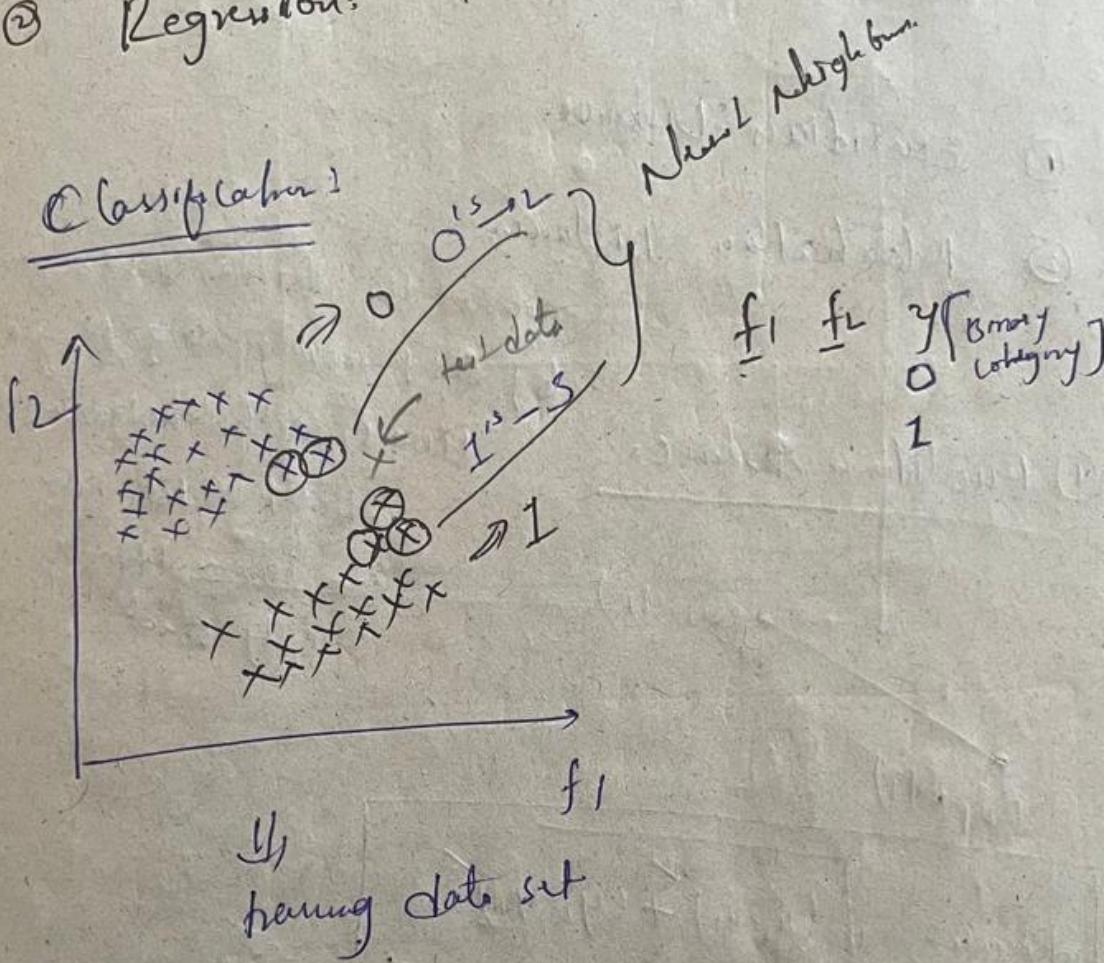
⑦ apply different colors bay and check
accuracy.

K-Nearest Neighbor (KNN)

used for both

① Classification.

② Regression:



steps:

① We have to initialize the K value.

Suppose $K=5$ selected. $K > 0 \dots \infty$
 $K=1, 2, 3, 4, \dots$ \Rightarrow hyperparameters
 $n_{\text{neigh}} \dots$

② Find the K nearest Neighbors for the test data.

③ From those $K=5$ how many neighbors belongs to 0 Category and 1 Category.
predicted as 1 (1 is more)

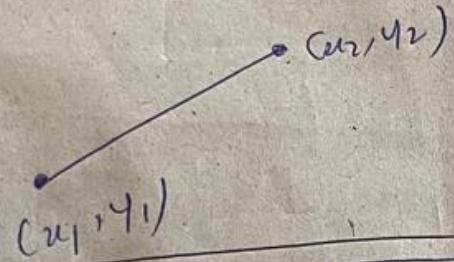
for calculating distance b/w them. (first
two point and nearest points) we
use two different distance formulae

① Euclidean Distance

② Manhattan Distance

↳ means the distance b/w two points
by only moving along the grid

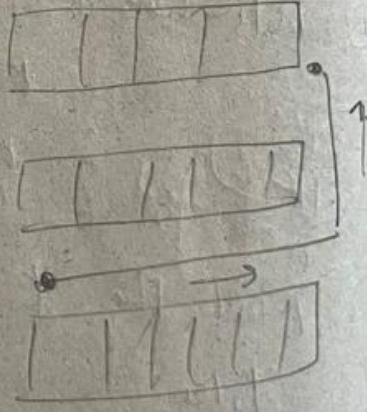
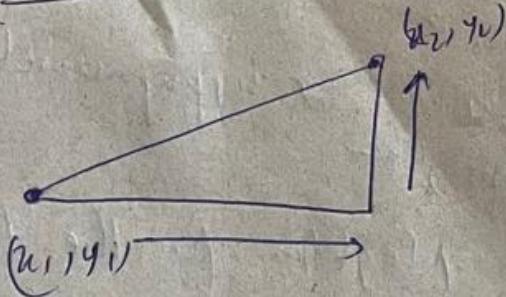
① Euclidean distance.



Value of like how
can move through
city street

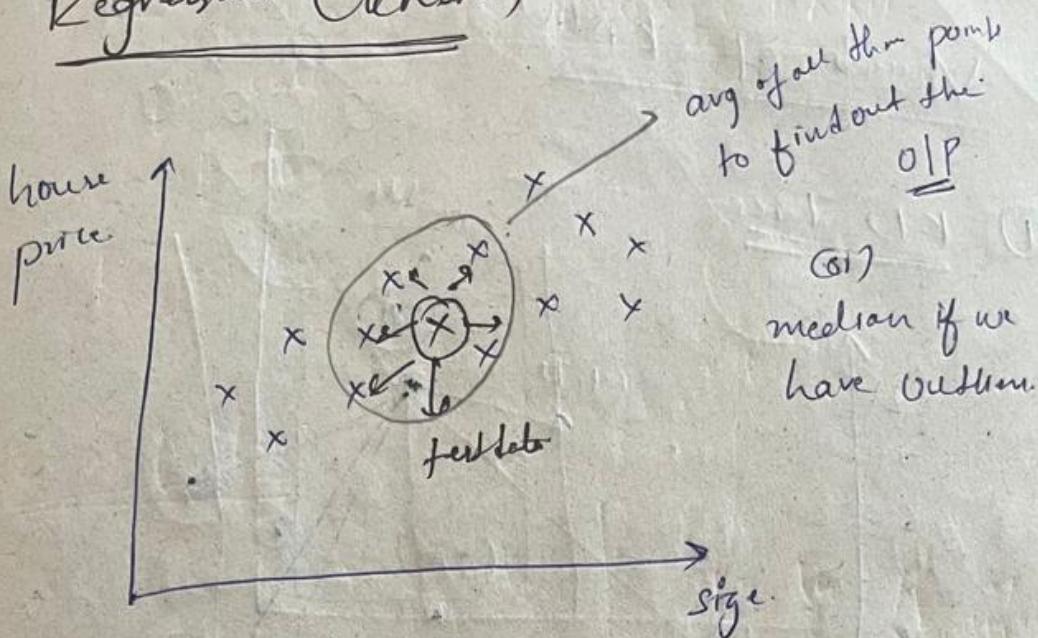
$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

② Manhattan distance (horizontal & vertical)



$$\text{dist} = |x_2 - x_1| + |y_2 - y_1|$$

Regression (KNN)



Why to learn various of KNN?



to calculate the nearest neighbor first we need to calculate the distance to all the other point and select the k nearest one

Time complexity

$O(n)$ \Rightarrow number of data point

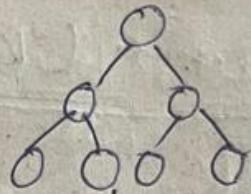
K D Tree \setminus optimize

Ball Tree.

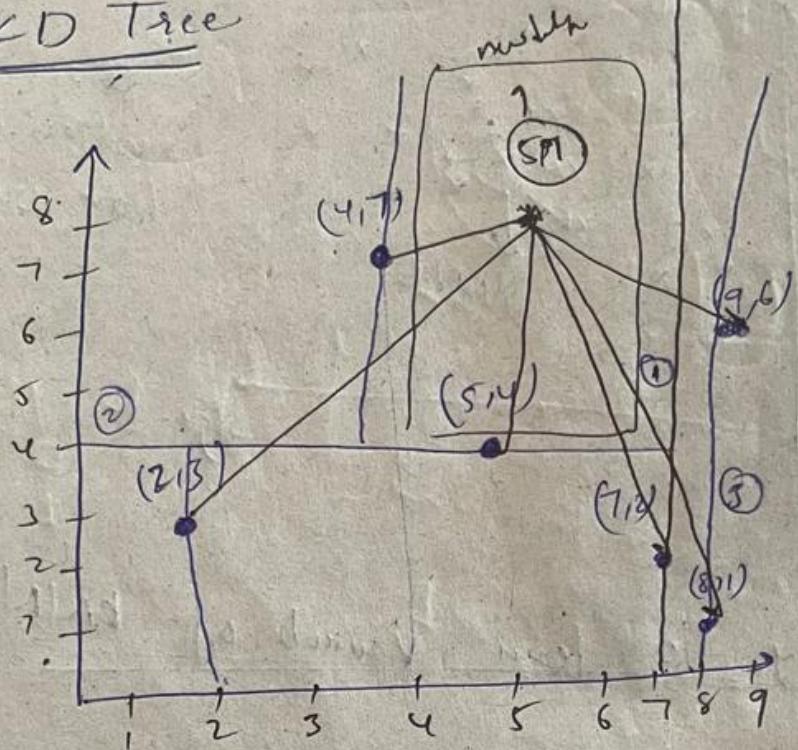


Binary tree

Value of KNN



① KD Tree



f_1	f_2
7	2
5	4
9	6
2	5
8	7

— median —

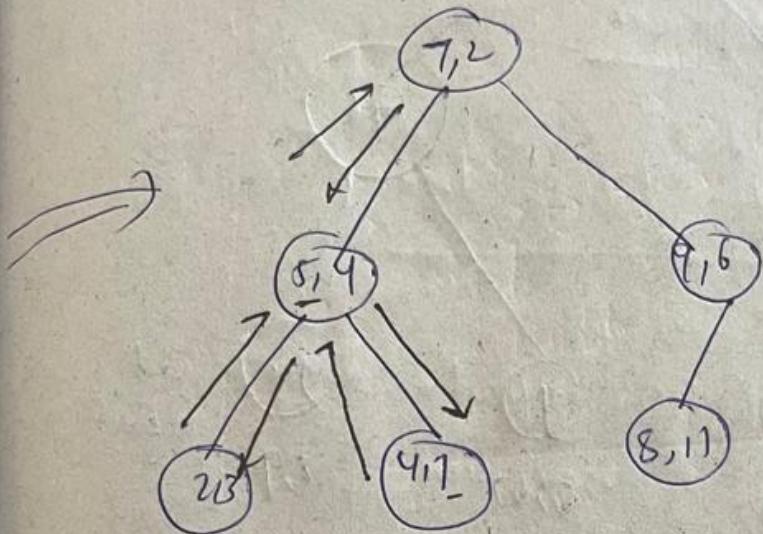
$\underline{f_1} : 2, 4, \boxed{5, 7}, 8, 9$ full table
any

$$\frac{5+7}{2} = \frac{12}{2} = 6.5$$

$\underline{f_2} : 1, 2, \boxed{3, 4}, 6, 7$

3.5

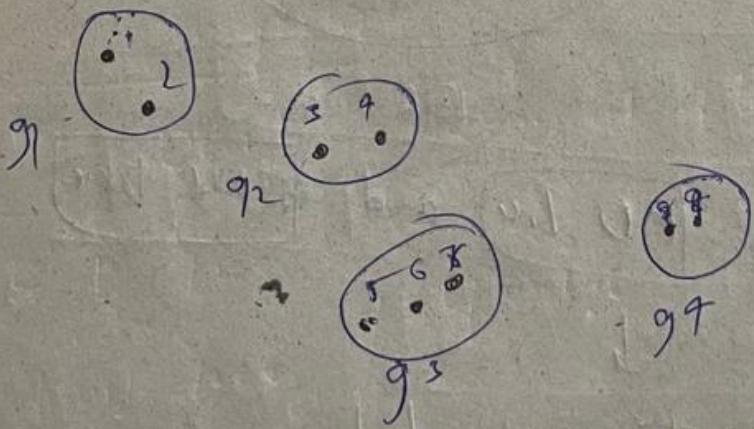
new del p (5,7)



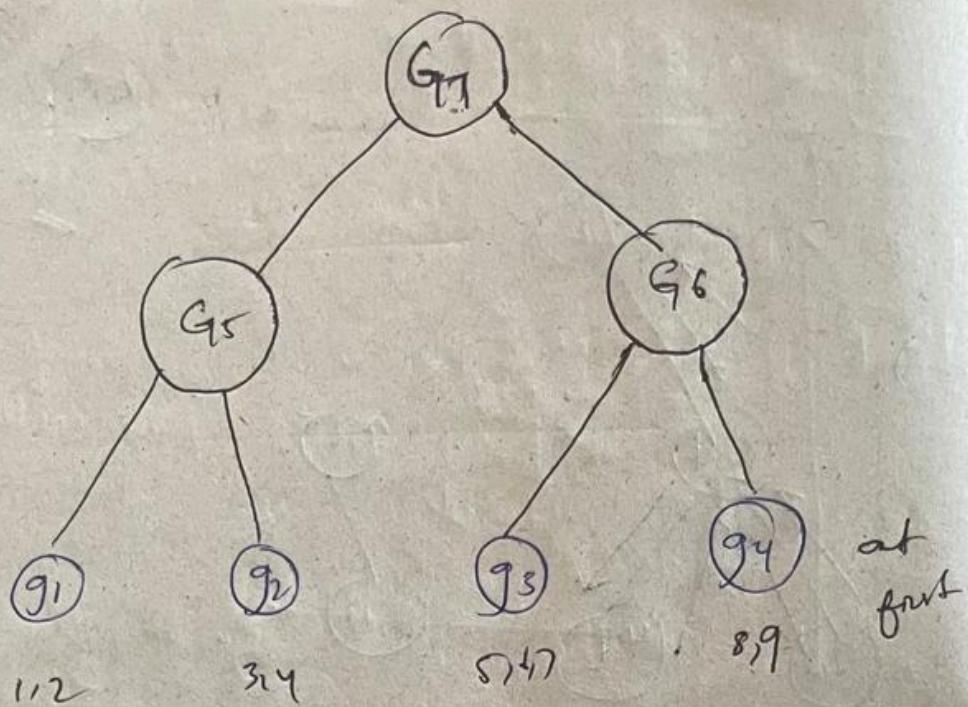
Balancing is used initially.

② Ball tree

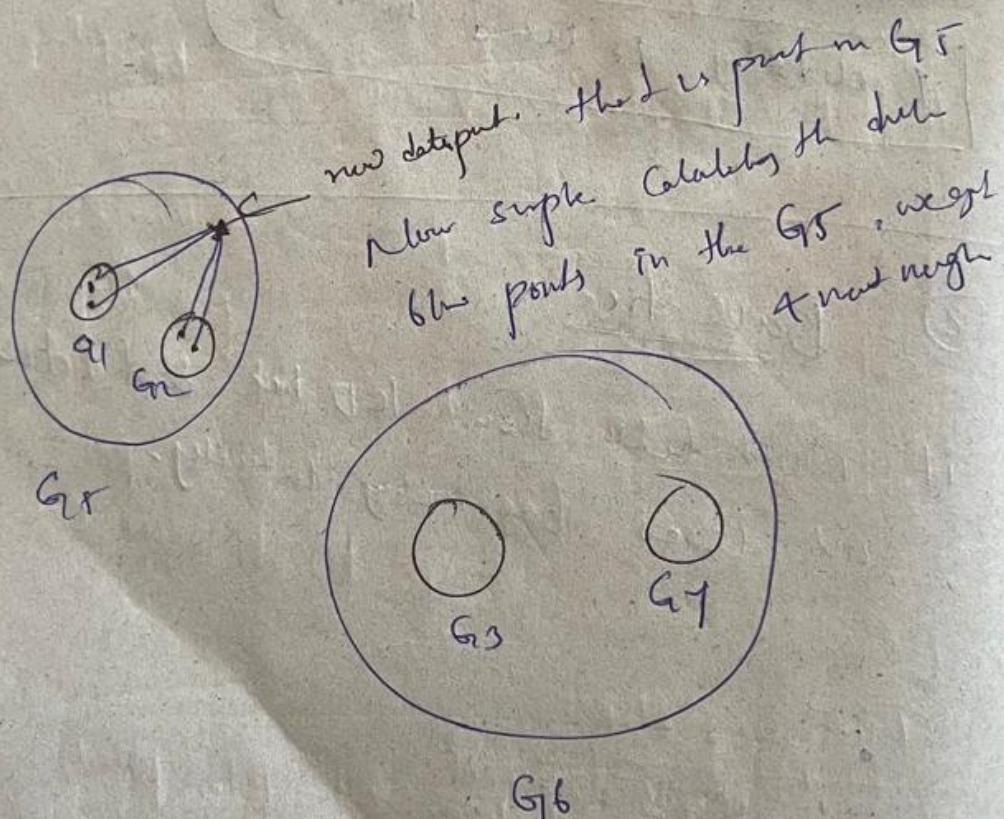
if works better than led tree and does not uses balancing initially.



→ group next point, new group next group's



at
first



By using kd Tree and Ball tree
 ↓
 Time complexity ↓ ↓

kNN Imput

→ import librae

→ make classifier using sklearn.

→ do train test split

→ apply kNearestClassifier.

from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=5, algorithm='auto')

p = 1 → euclidean

or
2 → manhattan

→ predict

→ perform metrics

'ball_knn'

'kd-tree'

'brute'

'auto'

it chooses the most appropriate algo based on the value passed to fit method

KNN Regressor

from sklearn.neighbors import KNeighborsRegressor

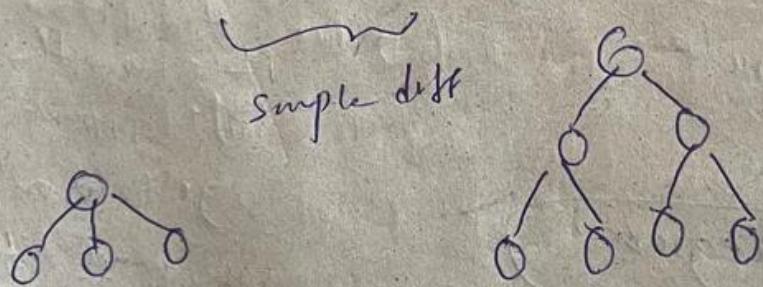
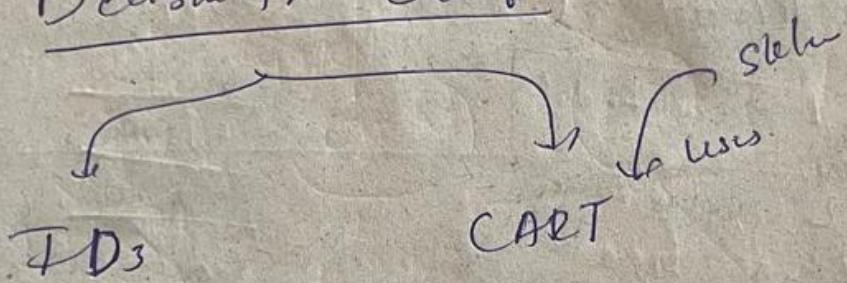
Decision Tree Classifier and Regressor

Introduction to Decision Tree

used to solve

both classifier as well as regression.

Decision Tree Classifier



② [Entropy and Gini Index]
↳ purity split

⑥ [Information Gain]
↳ feature selection for DT construct

$age = 14$

if ($age \leq 15$)

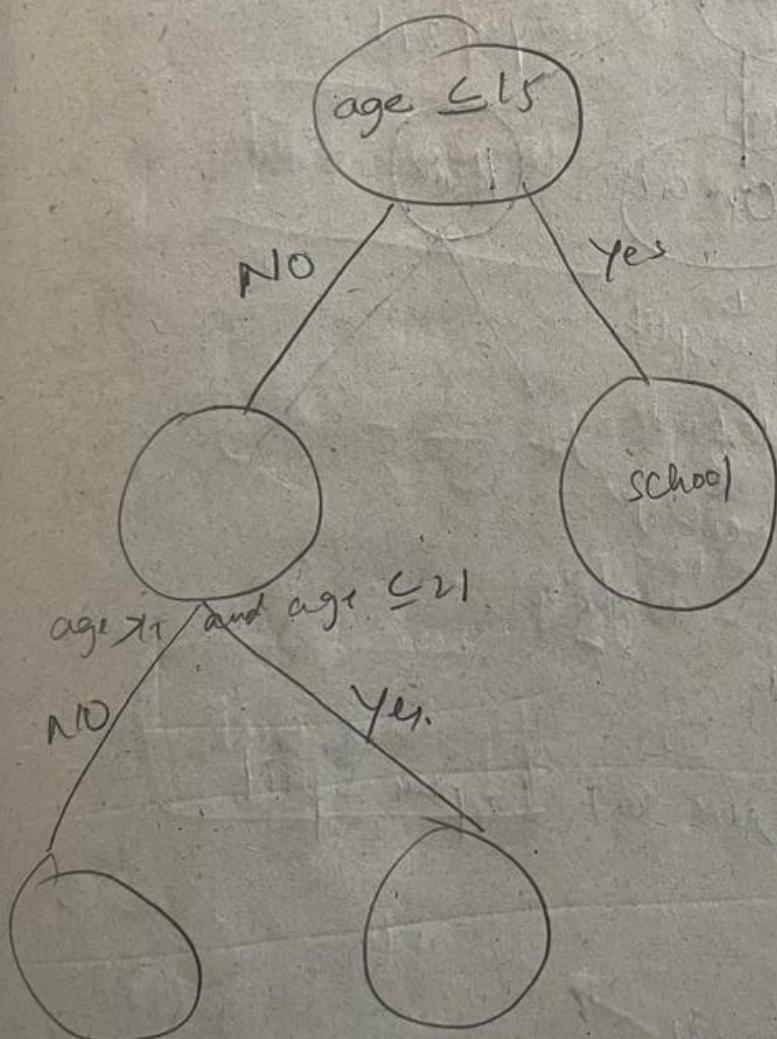
print ("school")

else if ($age > 15$ and $age \leq 21$)

print ("course")

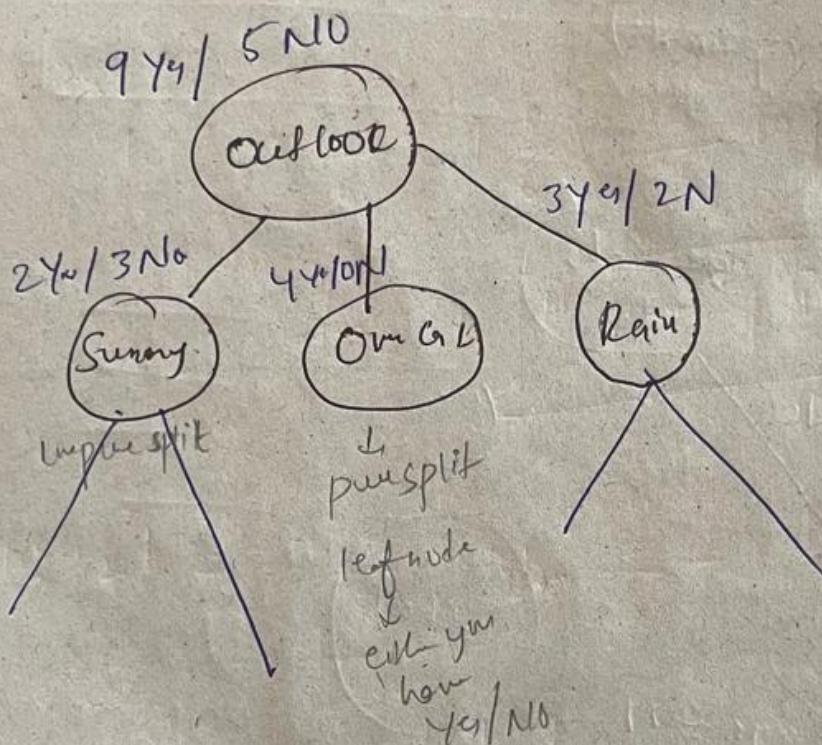
else:

print ("purple")



Data setBinary class

Day	Outlook	temp	humidity	wind	play
1	Sunny	Hot	High	Weak	Yes
2	Overcast	Mild	Normal	Strong	No
3	Rain	Cool			
4					

two output ports① Purity \rightarrow pure or Impure Split

① Entropy

② Gini Impurity

② Which feature you need to select for splitting. \rightarrow Information Gain

Entropy and Gini Impurity.

① Entropy

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

P_+ → probability of being Yes/1
in data set

P_- → probability of being No/0
in data
for binary classifier

for multiclass classifier (3 class)

$$H(P_1, P_2, P_3) = -(P_1 \log_2(P_1) + P_2 \log_2(P_2) + P_3 \log_2(P_3))$$

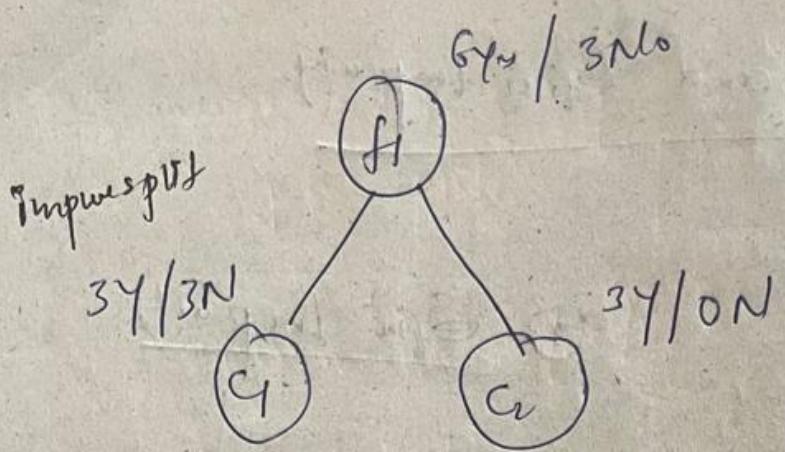
P_1 → probability of Category 1

P_2 → " 2

P_3 → " 3 ..

② Gini Impurity

$$G.I = 1 - \sum_{i=1}^n (P_i)^2$$



$$H(C_1) = -P_t \log_2 P_t - P_f \log_2 P_f$$

entropy of

$$= -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6}$$

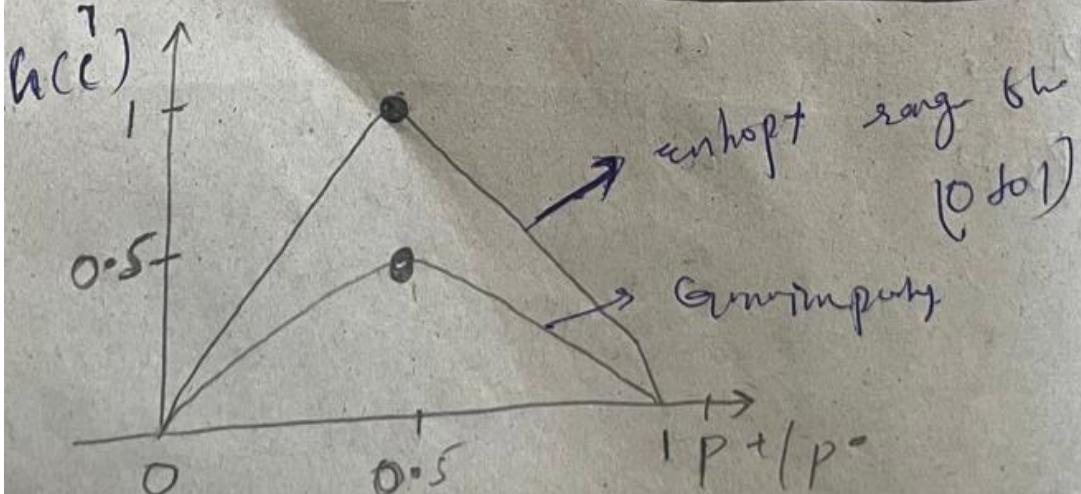
$$= 1 \Rightarrow \boxed{\text{impure split}}$$

$$H(C_2) = -\frac{3}{3} \log_2 \frac{3}{3} - 0 \log_0$$

$$= -\log_2 1$$

pure entropy

$$= 0 \Rightarrow \boxed{\text{pure split}}$$



② Gravitational entropy sign in $(0, 1)$

$$G_{\text{grav}} = 1 - \sum_{i=1}^n (P_i)^2$$

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

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

$$= 0.5 \quad \boxed{\text{impurity split}} \quad \leftarrow 3Y/3N$$

G.I. 3Y/0N

$$1 - \left(\left(\frac{3}{3}\right)^2\right)$$

$$= 1 - 1$$

$$= 0 \Rightarrow \boxed{\text{pure split}}$$

★ ★ ★
entropy gives max value as ① for impurity
split, similarly

and ① for pure

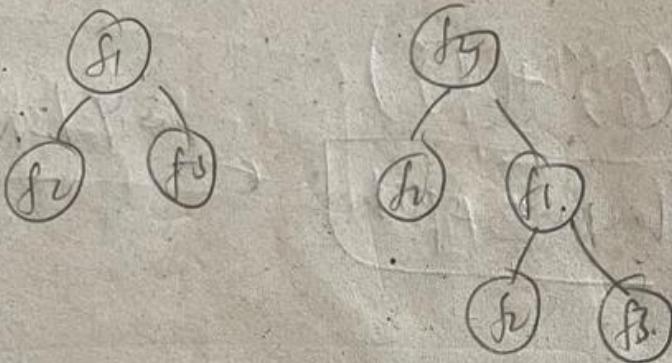
Gravitational entropy giving 0.5 for
impurity split and ⑥ for pure split

We should expand the feature if it
is impure split.

→ how to select feature

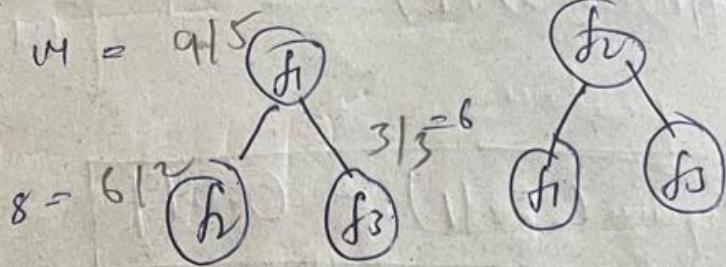
f_1, f_2, f_3

labelled file to
select



for this we are going to calculate
about information gain.

Information Gain: f_1, f_2, f_3 O/P



entropy of the root node.

$$\text{Gain}(S, f_1) = H(S) - \sum_{V \in \text{val}} \frac{|S_V|}{|S|} H(S_V)$$

$$H(S) = -P_1 \log_2 P_1 - P_2 \log_2 P_2$$

entropy of root node.

$$= -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14}$$

$$\approx 0.94$$

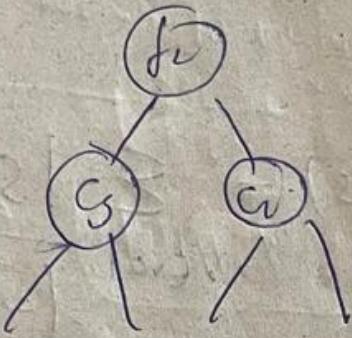
$$H(C_1) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8}$$

$$\boxed{H(C_1) = 0.81}$$

$$\boxed{H(C_2) = 1}$$

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

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



$$\boxed{\text{Gain}(S, f_2) = 0.051} \Rightarrow \text{Gain}(S, f_1) = 0.049$$

This tells about lake should start
with f_2

C This is how information gain is
calculated.

When do we use Entropy and
Gini Impurity

Entropy vs Gini Impurity.

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

$$\boxed{O/P = 3 \text{ Categ}}$$

$$H(S) = -P_C_1 \log_2 C_1 - P_C_2 \log_2 P_C_2 - P_C_3 \log_2 P_C_3$$

$$G.I. = 1 - \sum_{i=1}^n (P_i)^2$$

< 100.00

When dataset is small \rightarrow Entropy

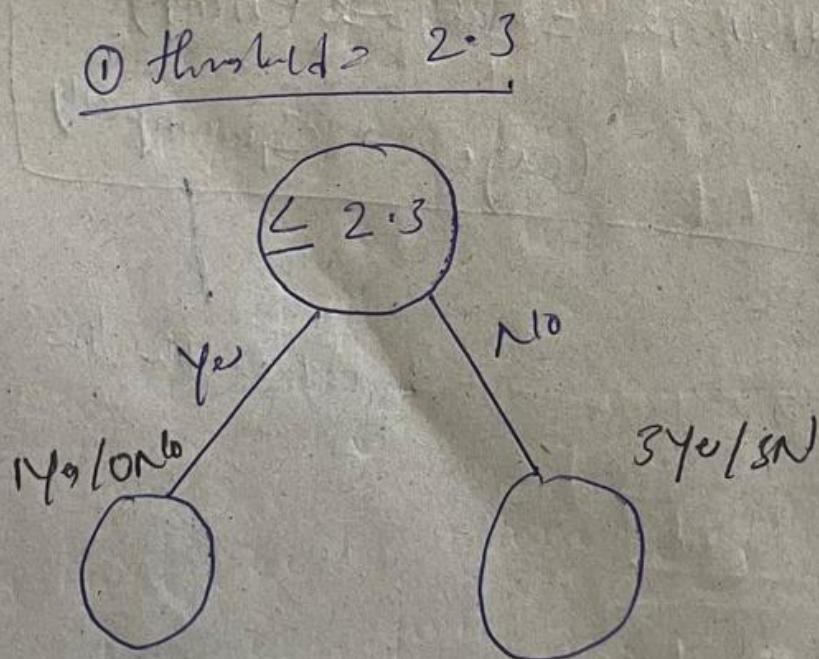
Large \rightarrow Gini impurity

most of the time

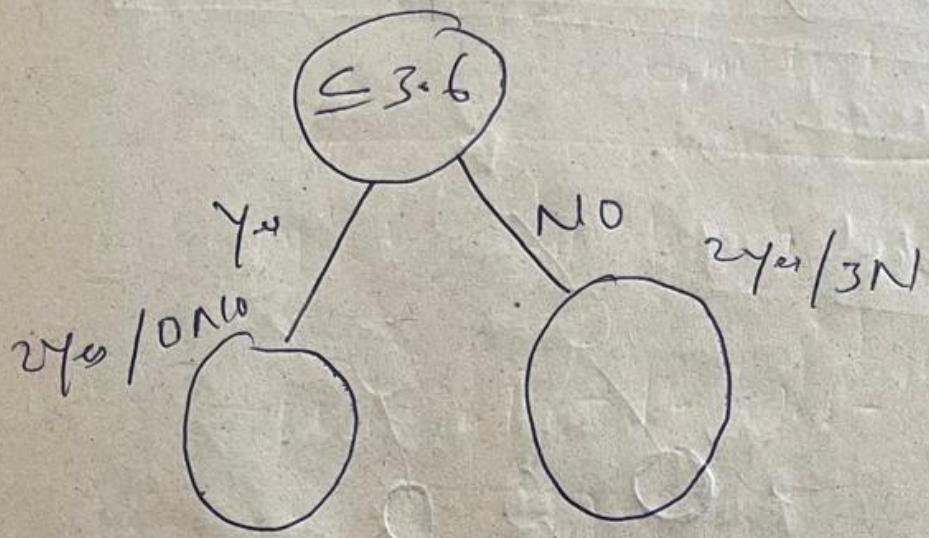
Decision Tree split for Numerical features

<u>f_1</u>	<u>O/P</u>
$\rightarrow 2 \cdot 3$	Yes
$\rightarrow 3 \cdot 6$	Yes
$\rightarrow 4$	No
$\rightarrow 5 \cdot 2$	No
$\rightarrow 6 \cdot 7$	Yes
$\rightarrow 8 \cdot 9$	No
$\rightarrow 10 \cdot 5$	Yes

- ① Send the feature value (Casually ask)



② threshold = 3.6

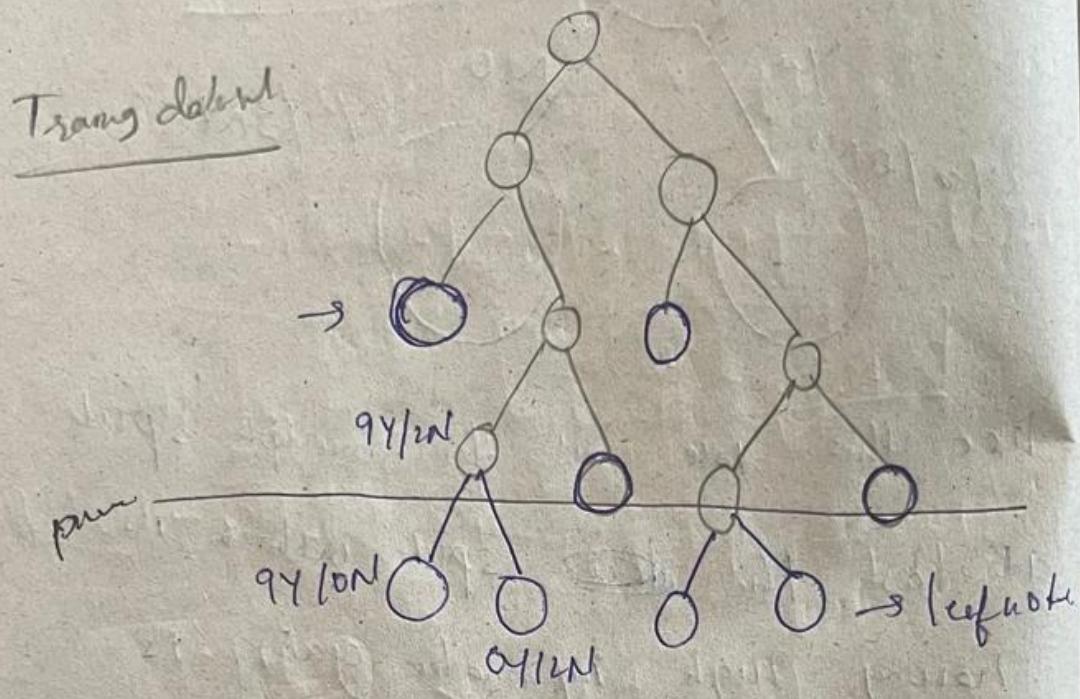


thus we will do multiple split.
at last the ~~leaf~~ split which is
having high information gain is
selected.

If more of records

Time complexity ↑↑

Post pruning and pre pruning (decision tree)



if i split my decision tree completely to 1b

depth it may leads to

Overfitting → low Bias \nwarrow heavy

high var \nwarrow testing

hang acc^p

feeling acc^s

\rightarrow In order to reduce Overfitting we use

two techniques:

- ① post pruning , ② pre pruning

Post pruning:

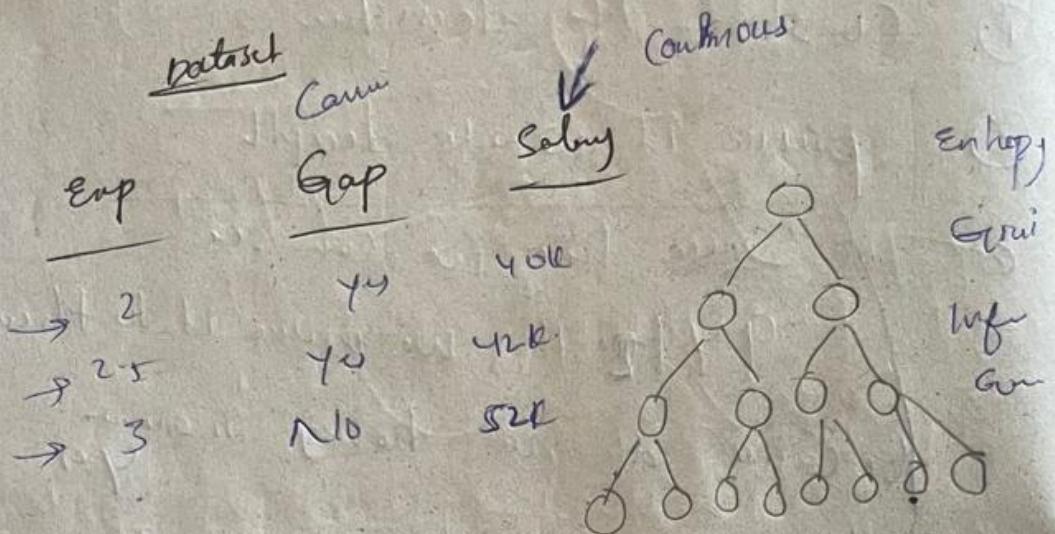
- ① Construct the decision tree
- ② prune it w.r.t depth.
Since here we can see 94% , we prune it to then because we are having majority.
- ③ Used for small datasets

Pruning:

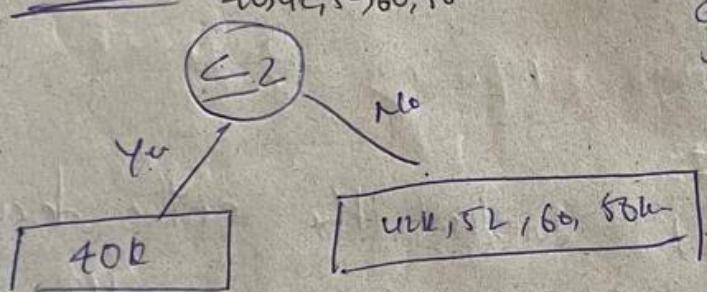
→ late don't construct the entire decision tree, instead late play with hyperparameters (max features, max depth, splits)

- ① ~~Train with~~ hyper parameters tuning while constructing DT

Decision Tree Regression



threshold = 2

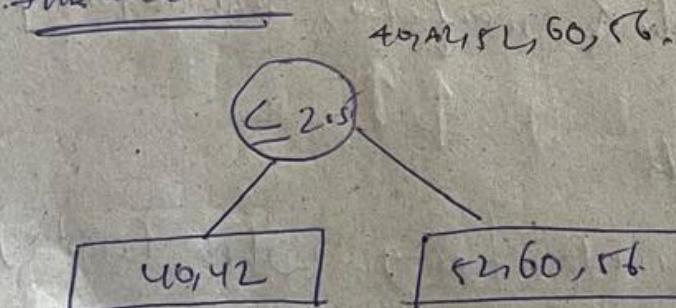


→ Here we are not going to use it.

Unlike classifier

how we are going to calculate the variance Reduction for each feature and Select the one with highest

threshold = 2.5



In normal decision tree classification we are information gain to select the feature

but here we use Variance Reduction
(Regres problem)

$$\text{Variance} = \frac{1}{n} \sum_{i=1}^n (y - \bar{y})^2$$

Mean
 Square
 average

① Variance of (root)

$$= \frac{1}{5} \left[(40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2 \right]$$

$$= \underline{60.8}$$

$$\text{Variance}(c_1) = \frac{1}{n} \sum_{i=1}^n (y - \bar{y})^2$$

$$= \frac{1}{1} (40-50)^2$$

$$= 100$$

$$\text{Variance}(c_2) = \frac{1}{n} \sum_{i=1}^n (y - \bar{y})^2$$

$$\frac{1}{4} \left[(42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2 \right]$$

$$= \frac{1}{4} [64 + 4 + 106 + 36]$$

$$= 51$$

Variance Reduction

$$= \text{Var}(\text{Root}) - \sum w_i \text{Var}(\text{child})$$

$$= 60.8 - \left[\frac{1}{3} \times 100 + \frac{4}{3} \times 51 \right]$$

$$= 60.8 - 20 - 40.8$$

$$\boxed{\text{Variance Reduction} = 0}$$

~~100% Var~~

$$\underline{\text{Var}(C_1)} =$$

$$= \frac{1}{2} \left[(40-50)^2 + (42-50)^2 \right]$$

$$= \frac{1}{2} [100+64]$$

$$= \frac{164}{2} = 82$$

$$\underline{\text{Var}(C_2)} = \frac{1}{3} [4 + 100 + 51]$$

$$= \frac{116}{3}$$

$$= \underline{38.66}$$

Variance Reducer

$$60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.66 \right]$$

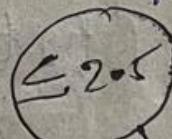
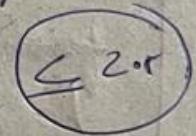
$$60.8 - [32.8 + 27.996]$$

$$\boxed{-0.004}$$

Variance Reducer (left split) < VR (right split)

↓
left tree

this is
selected for
splitting.



Let consider
left node

$$\boxed{40, 42} \quad \boxed{52, 60, 56}$$

$$OPI = \left[\frac{40+42}{2} \right]$$

$$\underline{\underline{= 41}}$$

$$\left[\frac{s_2 + 60 - 56}{5} \right]$$

$$OPI = \underline{\underline{56}}$$

Decision Tree Implementation and

done post
Pruning

- import dataset
- Split independent and dependent features.
- Trained split
- apply dec tree algo.

from sklearn import DecisionTreeClassifier

(criterion='gini')

Visualize the decision tree

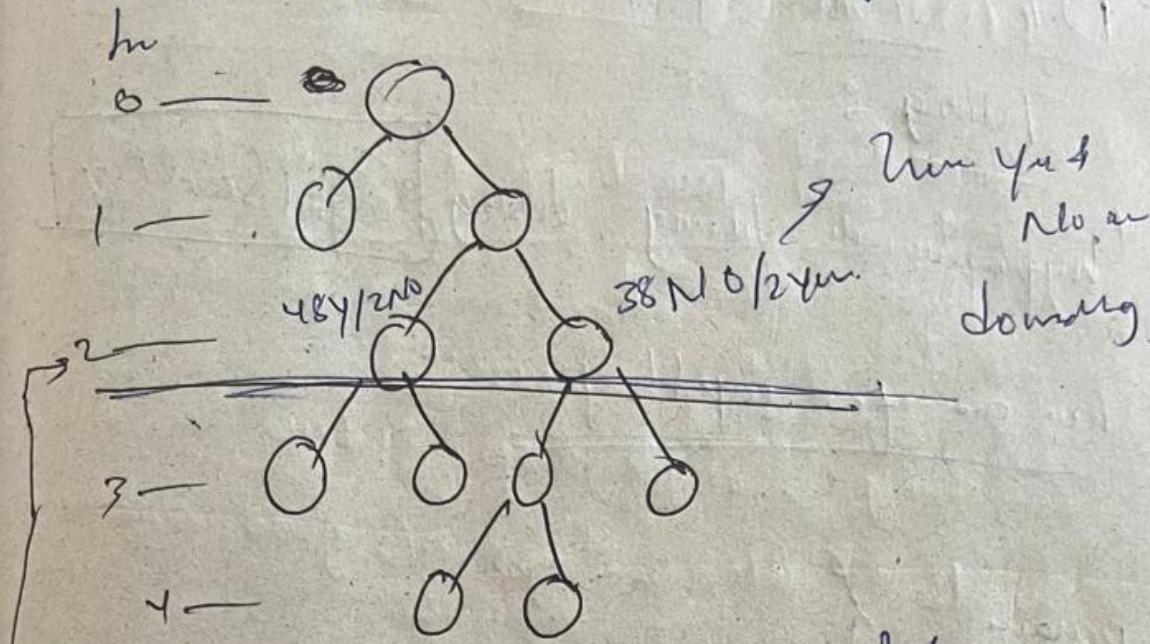
from sklearn import tree

plt.figure(figsize=(15, 10))

tree.plot_tree(tree_clf, filled=True)

Performance metrics, Confusion matrix, classification report

here we are going to discuss computer vision



but it may leads to overfitting condition
so we can do post pruning by cutting
the tree

postpruning is applied only for smaller
dataset (≤ 10000)

is made by growing ~~depth~~ max depth
parameter while creating objects
to Decision Classif (max depth = 2)

DT classifier pruning and Hyperparameters

Tuning :-

Hyperparameters tuning using Grid Search CV.

here are few inputs parameters

param = {

'criterion': ['gini', 'entropy', 'logloss'],

'splitter': ['best', 'random'],

'max_depth': [1, 2, 3, 4, 5],

'max_features': ['auto', 'sqrt', 'log2']

3

Now just apply GridSearch CV.

Diabetes prediction using Decision tree

Regression

from sklearn datasets import load_diabetes

RANDOM FOREST Machine Learning.

Bagging and Boosting (Ensemble technique)

ensemble techniques
Bag + Boo. used for both classification

Combining multiple algo to train the model.
So that we can get the good accuracy.

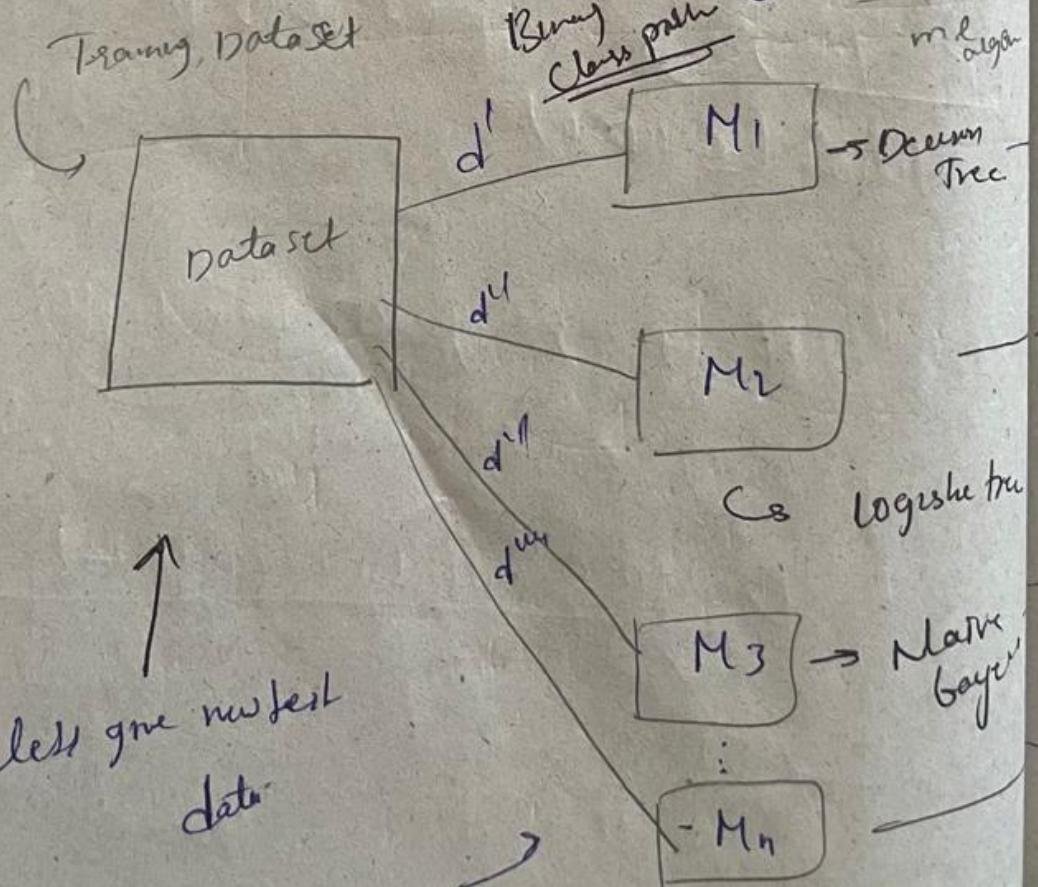
① Bagging: ① Random Forest algorithm

6 Idea behind Bagging.

we have
multiple
base learners

2

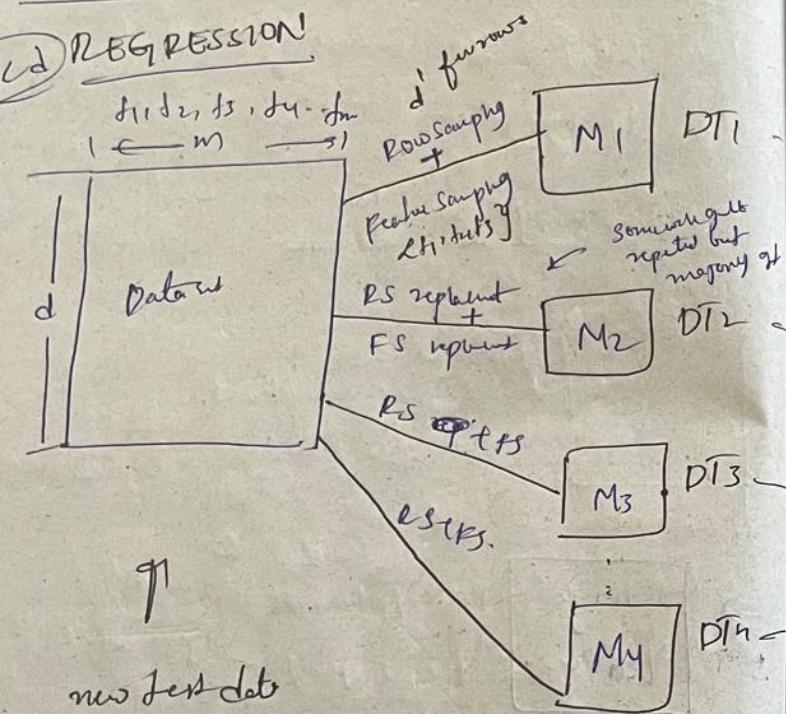
ml algo



*** all the base learners are learned parallelly

RANDOM FOREST CLASSIFICATION AND

(a) REGRESSION



Classification

1st

majority voting classifier (O)

Regression: average output of the models.

Inference

Why Random forest instead of DT?

Decision

Overfitting

Train acc \rightarrow low bias

Test acc \rightarrow high var \rightarrow low bias
using rand forest

Generalized model

\rightarrow low bias

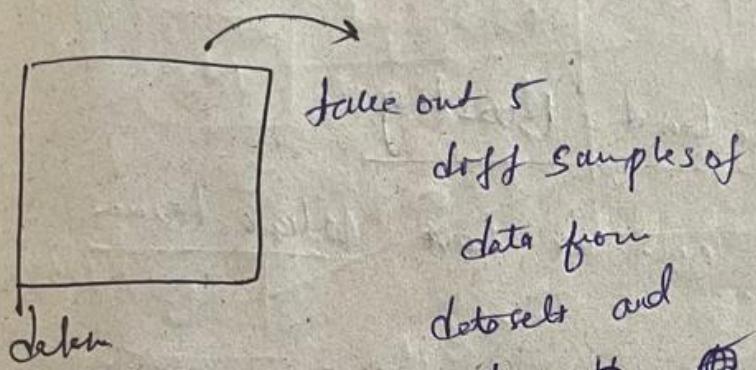
\rightarrow low bias

if we add 100 new records to dataset and try to train the model it does not impact much. because this splits among all the models

~~***~~

In Bagging we should give different samples of training dataset to each model.

Suppose if we have 5 models.



and give ~~from the~~ to 5 diff model

and check for the majority voting class

→ 0 ✓ Bagging misclassification problem Bagging for regression problem

⇒ majority voting classifier

→ 1 ✓ ⇒ 0 // average of all the outputs

→ 0 ✓ this is the output

→ 0 ✓ at

② Boosting:

diff algo

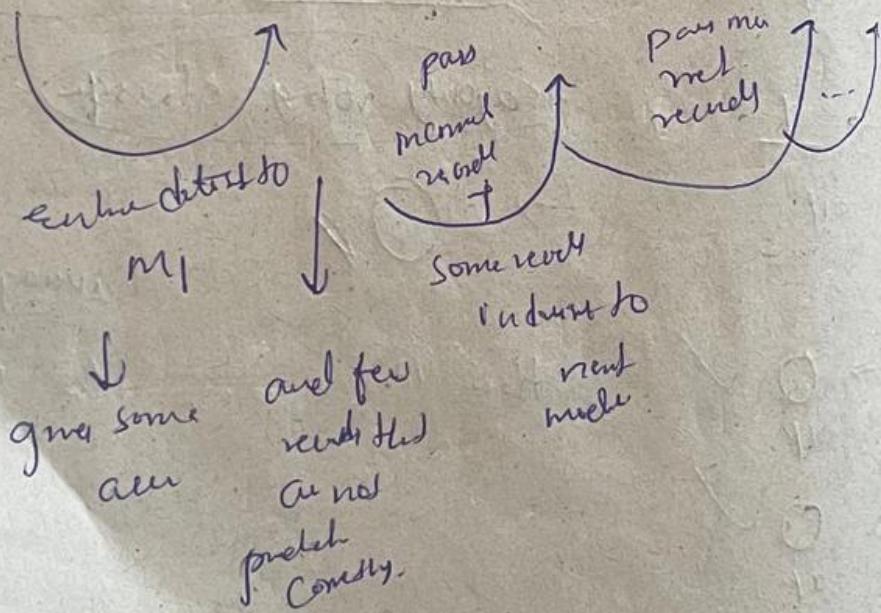
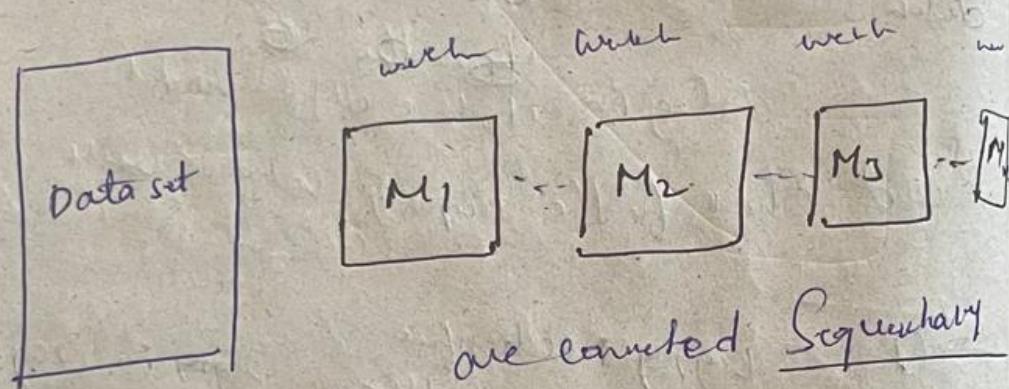
① Ada boost

② Gradient Boosting

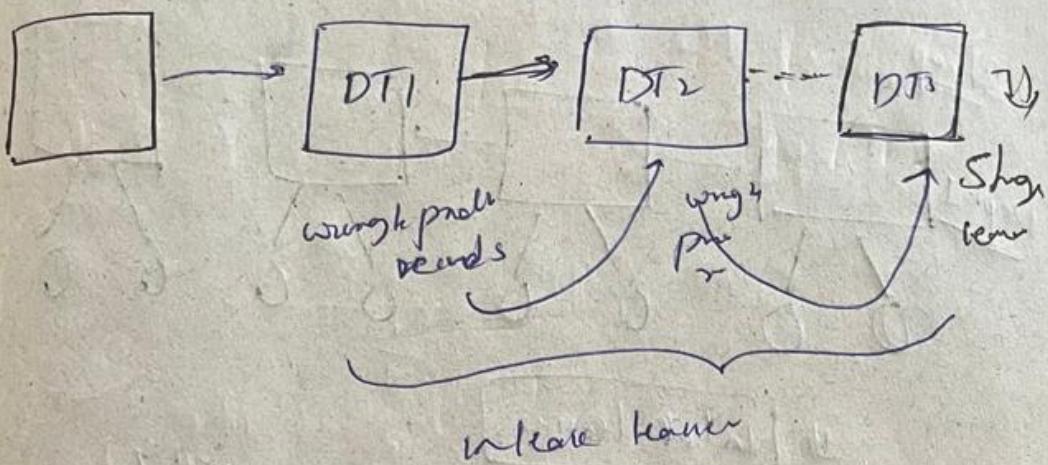
③ Xg boost (Extreme Gradient Boost)

Idea behind Boosting:

like base learner we have weak learner.



Boosting: $\{S_i\}$ sequentially connected



weak learners: haven't learnt much from the learning dataset

Random Forests: \rightarrow Majority voting classifier
average of b/p all of P.
 \approx Regn

AdaBoost: assignment weights to the weak learners

$$f = \alpha_1(m_1) + \alpha_2(m_2) + \alpha_3(m_3) + \dots + \alpha_n(m_n)$$

$m_1, m_2, \dots, m_n \rightarrow$ Decision Tree Stumps
 $\{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n\}$ weights

Ensemble technique

Decision Tree
(Base line)
Bagging

① Random Forest
Classifier

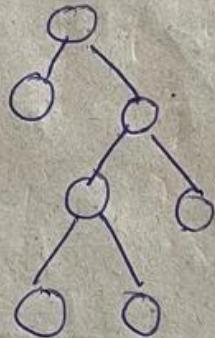
② Random Forest
Regressor

Decision
Weak learner

Boosting

- ① AdaBoost
- ② Gradient Boosting
- ③ Xgboost

Decision Tree



Ovifiting:

Train acc⁹
less acc¹

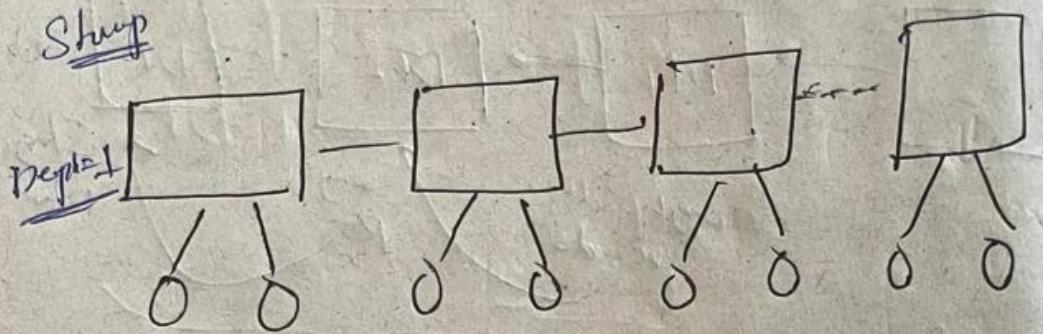
low bias
high variance

integrate sample
of data

Random Forest

low bias
low variance
(Bagging)

Decision Tree stump (with Depth is just 1)



$y_i \mid \text{Node-learn}$

(Underfitting (Because depth is just 1))

Training acc 40%

Testing acc 0%

$$\left\{ \begin{array}{l} \text{high bias} \\ \text{low variance} \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} \text{low bias} \\ \text{high variance} \end{array} \right\}$$

AdaBoost Classifer Model Indepths Induhn

Salary Credit Approve

$C=50k$ B No

$C=50k$ G Yes

$C=50k$ G Yes

$>50k$ B No

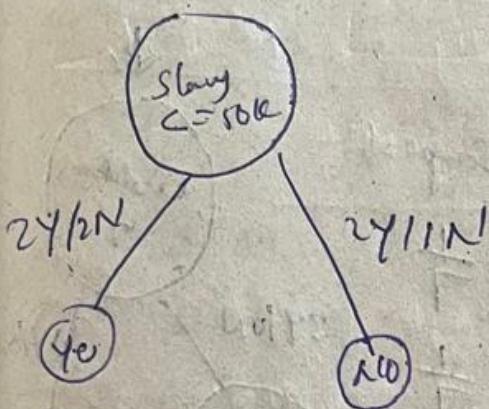
$>50k$ G Yes

$>50k$ N Yes

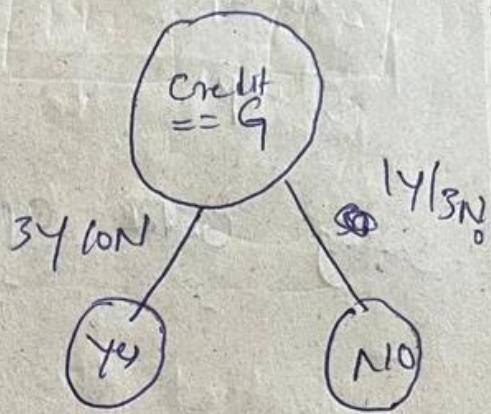
$\leq 50k$ N No

⑦ We Create Decision Tree Shap.

Decum for Shap 1



DT Shap 2



Now which one to select as the one

(Entropy or Gini Impurity)

We can see that DT Shap 2 has less impurity compare to Shap 1

∴ we selected the best DT Shap using
Entropy or Gini Impurity

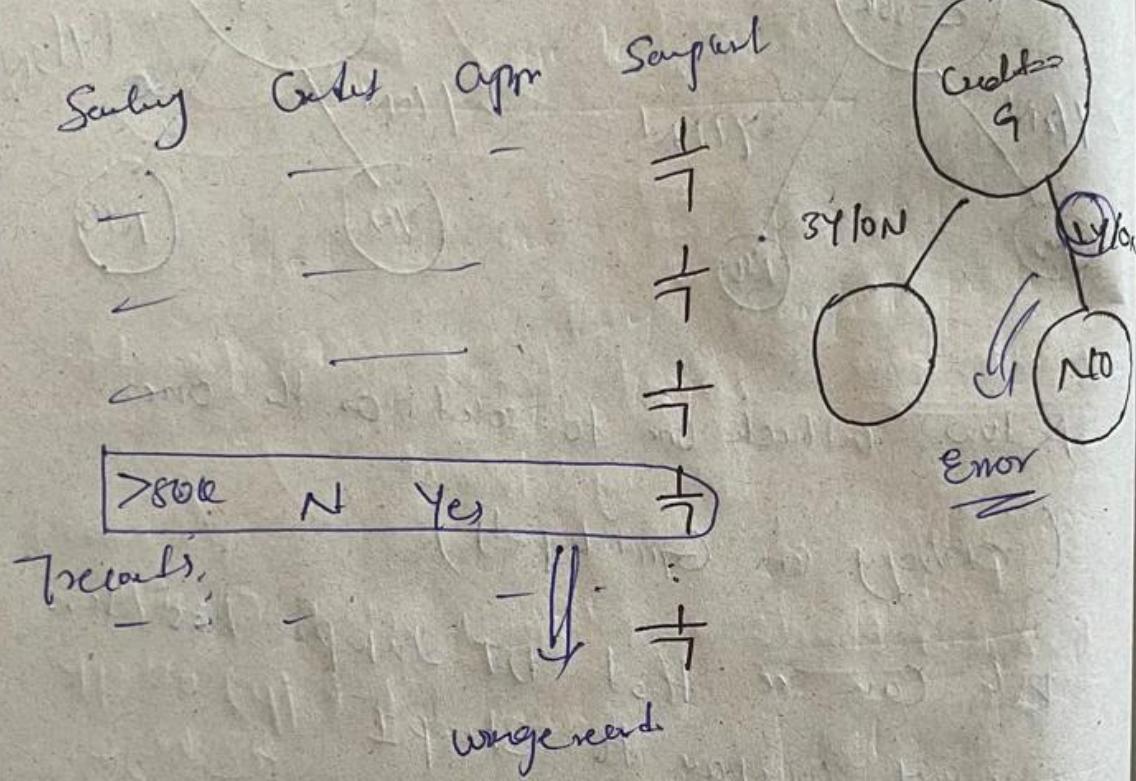
Best Entropy says that

④ You will get the leaf node
very quickly.

Step 3: Performance of Decision Tree Shump.

~~Sum of the total errors and performance of shumps~~

Sum of the total errors and performance of shumps



$$\text{Sum of all the total error} = \frac{1}{7}$$

$$\textcircled{2} \quad \text{Performance of shump} = \frac{1}{2} \ln \left(\frac{1 - TE}{TE} \right)$$

$$= \frac{1}{2} \ln \left[\frac{1 - \frac{1}{7}}{\frac{1}{7}} \right]$$

$$= \frac{1}{2} \ln(6) = \approx 0.896$$

0.896 performance of ship

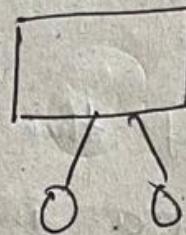
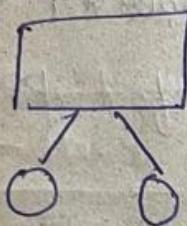
$$f = \alpha_1(m_1) + \alpha_2(m_2) + \dots + \alpha_n(m_n)$$



$$\boxed{\alpha_1 = 0.896}$$

weight

Credit DT Ship 1



next DT Ship

predicted some wrong records

0.896 → helps to assign weight to full
decision tree ship

Step 3: update the weight for correctly
and incorrectly classified points

<u>Salary</u>	<u>Credit</u>	<u>Age</u>	<u>Spouse</u>
—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

wrong
rec

Now we need to update this weight and make sure that which are the clamped pounds we have to reduce the weights and which are the incorrectly clamped pounds we have to increase the weight

→ so that there is a high probability of selecting a correctly clamped part by the next DT stamp.

$$\begin{array}{c}
 \text{---} \quad \text{---} \quad \text{---} \\
 \text{Updated} \quad \text{for correctly clamped} \\
 \text{wts} \quad \text{parts} \\
 \hline
 0.058 \downarrow \\
 \boxed{\text{incorrectly clamped part}} \quad 0.349 \quad \text{for unclamped part} \\
 \end{array}$$

first we \rightarrow $e^{-pf \cdot off}$
 $= \frac{1}{7} * e$
 $= 0.058$

$$\begin{array}{c}
 0.058 \downarrow \\
 \boxed{\text{incorrectly clamped part}} \quad 0.349 \quad \text{for unclamped part} \\
 \end{array}$$

first we \rightarrow $e^{pf \cdot off}$
 $= \frac{1}{7} * e$
 $= 0.058$

GRADIENT BOOSTING

Regression

Classification

Regression Dataset			\hat{y}		$(y - \hat{y})$		OIP of DT1	update
exp	degree	Salary	y	\hat{y}_1	\hat{y}_2	\hat{y}_3	\hat{y}	
2	B.E	50k	75k	75k	-25k	-25	72.7	
3	Maste	70k	75k	75k	-5k	-3	72.9	
5	M	80k	75k	75k	5k	3	88.0	
6	PhD	100k	75k	75k	25k	20	105	
		<u>100k</u>	<u>75k</u>	<u>75k</u>	<u>25k</u>	<u>20</u>	<u>105</u>	
							$R_s(y - \hat{y})$	

Steps:

- ① Create a base model.



OIP every time 75
average $\frac{50+70+80+100}{4} = 75$

- ② Compute Residuals, Error

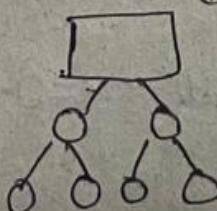
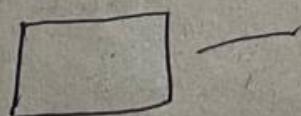
- ③ Construct a decision tree consider inputs

x_i and OIP R_i

DT takes all input features and R_i as OIP

L_{ui}, R_i

base node



Step 1 Normalized Weights Computing and
assigning Bins

predicted

0.658

0.658

0.656

0.349

0.058

0.697

target not giving 1

Normalized weights 1

0.08 bins assigned by density

0.08 0.08-0.66 for any

0.08 0.16-0.24 range

0.24-0.32

0.32-0.40

0.40 0.40-0.90

0.08 0.90-0.98

≈ 1

To send wrongly predicted points to next DT
 DT stump we need to do bins assignment



$$\lambda_1 = 0.0896$$



prepared
data points to
send

⑤ Selected the data points to send to next step

Step

Salary	Credit	Approv	Bus assign
$\leq 50k$	B	NO	0-0-08
≤ 80			0.08-0.16

750	N	Yes	0.40-0.90
-----	---	-----	-----------

We will generate random numbers between 0 to 1 equal to the no of records and we are going to get more or less predicted records since it has high bias.

size (0.40-0.90)

S	C	appr	Rand
750	N	Y ₁	0.50
≤ 50	G	Y ₂	0.10
$> 50k$	N	Y ₃	0.60
$> 80k$	N	Y ₄	0.75
$\leq 50k$	G	Y ₅	0.24
$> 10k$	B	NO	0.32
$> 50k$	N	Y ₆	0.87

⑥ These records will be sent to next DT step.

S Credit Approv Step

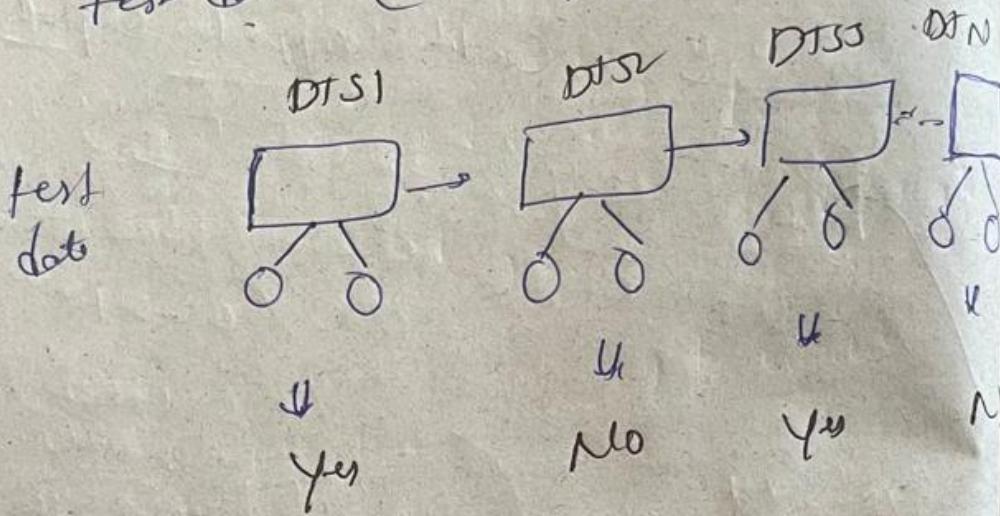
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

Now all the steps that we have done from step 1 to 6 will keep on repeating.

Final prediction of adaboost

(Classify)

test data ($\leq 800, G_1$)



Value may only wrong classify

$$x_1 \quad x_2 \quad \dots \quad x_3 \dots x_4$$

Const $x_1 = 0.896 \quad x_2 = 0.600 \quad x_3 = 0.24 \quad x_4 = -0.30$

$$f = x_1(M_1) + x_2(M_2) + x_3(M_3) + x_4(M_4)$$

$$= 0.896(Y_1) + 0.600(N_2) + 0.24(Y_3)$$

Prefers of guy with ^{Say} $(Y_1) > N_2 - 0.30(N_2)$

add up Y_1 , also

$$= 1.136(Y_1) + 0.30(N_2)$$

OP: Y_1

after having model 2 we may get value in

$$-23, -3, 3, 20 \text{ (R}_2)$$

→ again compute the residuals.

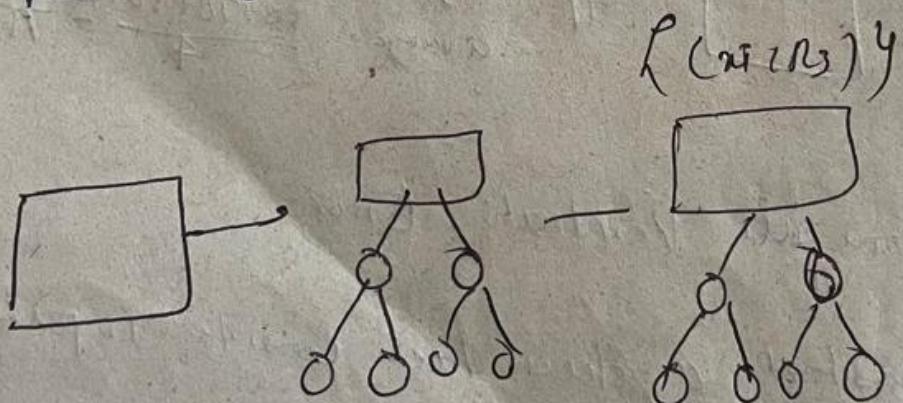
$$\begin{array}{r} \text{predicted o/p} = 75 + (-23) = 75 - 23 \\ \hline \text{real (S0). } \end{array} \quad \begin{array}{l} \text{R}_2 \\ = 53 \text{ (Overflow)} \end{array}$$

∴ so we are not going to calculate like this

$$\text{predicted o/p} = 75 + \alpha (\text{DT}_1) = 75 + (0.1)(-23)$$

$$\begin{aligned} \alpha &= \text{leaving rate (0-1)} \\ \alpha &= 0.1 \end{aligned} \quad \begin{aligned} &= 75 - 2.3 \\ &= \underline{\underline{72.7}} \end{aligned}$$

Now again repeat the steps ($R_3 = y - \hat{y}$)



$$P(x) = \alpha h_0(n) + \lambda_1(h_1(h_0(n))) + \lambda_2(h_2(h_0(n))) + \dots + \lambda_n(h_n(h_0(n)))$$

leaving rate $\alpha = 0.1$

$$F(x) = \sum_{i=0}^n \alpha_i h_i(x)$$

final function
of Gradient
Boosting

Implementing Gradient Boost classifier

get the best parenthesis and tune the
model with those parenthesis
to reduce error

C

Implementation Gradient Boost Regressor

- Input: feature set algo (regressor)
- RFR, ABR, GBR, LR, Ridge, Lasso, Venetian Regressor, DTR, pruned, msc, max
- Evaluation function: to evaluate model
diff error metric (loss, predict):
- Begging model: choose algo

models > d

- LR : LR(1);
- Lass : Lasso(1)
- Ridge : Ridge(1).
- NNReg : KNN(1);

y

we will get have a list of all models

- Based on performance choosing algo to hyper parameters tuning.
- goal for $y = L$

XG Boost (class and Regn) (Extreme Gradient Boosting)

<u>Classifcat</u>	<u>Date</u>	$\beta = 0.5$	bias	$y - \hat{y}$
<u>Salary</u>	<u>Credit</u>	<u>approval</u>	R^2	R
$< 50k$	B	0	-0.5 0.5	-0.4
$< 50k$	G	1	0.5 or 0.58	0.4
$< 50k$	G	1	0.5	-
$> 50k$	B	0	-0.5	-
$> 50k$	G	1	0.5	-
$> 50k$	N	1	0.5	-
$< 50k$	N	0	-0.5	-

Sentiment w.r.t

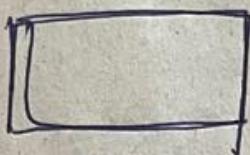
$E(\text{Credit})^2$

$E(p_i(1-p_i))$ \rightarrow hyper parabola

\downarrow
con. value if sentiments \in cover.
we are going to see it on other

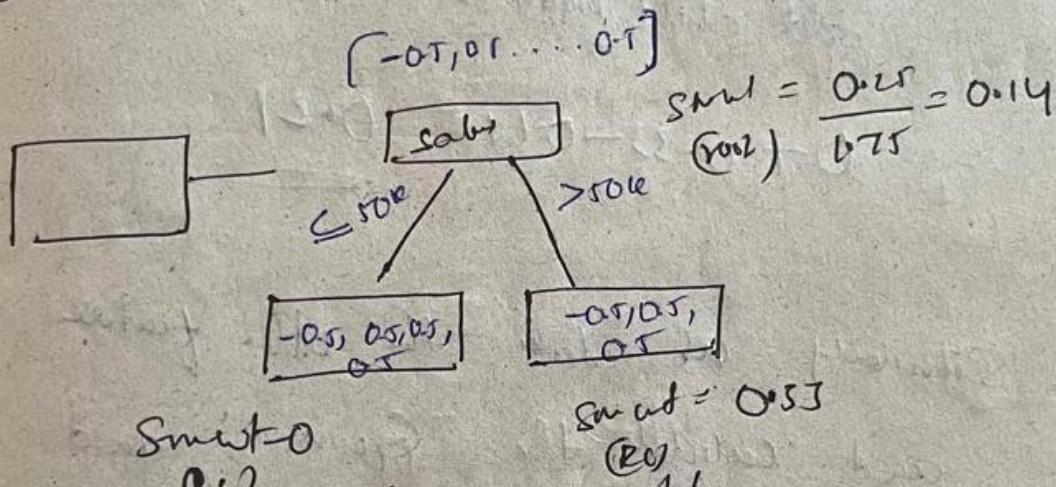
Steps to create XG Boost model

- ① Construct a base model
Since it's a binary classifier.



gives prob 0.5

- ② Construct a decision tree with root



- ③ Calculate similarity weight

$$\text{Similarity weight } S_{\text{sim}}(a) = \frac{\sum_{l \in C} (\text{child}_l)^2}{\sum p_l (1-p_l)}$$

$$= \frac{[-0.5 + 0.5 + 0.5 - 0.5]^2}{1}$$

$$[0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)]$$

$$= 0.4$$

Snapping & g food clausur.

PIP until & g food.
eg busypm

$$\text{Similarly weight (RC)} = \frac{(-0.5 + 0.5 + 0.8)}{0.75}$$

$$= \frac{0.25}{0.75} = \frac{1}{3}$$

(7)

Calculate Gain

add child nodes and subtract root node

$$= 0 + 0.33 - 0.14 = \underline{\underline{0.21}}$$

Similarly we take the other feature
and calculate the Gain, and
which ever feature we are getting
high gain take one going to take
that feature and start split.

highway xg board Rogue

χG Boost class = Base loss + $\lambda_1 (DT_1) + \lambda_2 (DT_2) + \dots + \lambda_n (DT_n)$

λ = length scale

$\alpha > 0.1$ hyper

χG Best Regn = Base loss + $\lambda (DT)$

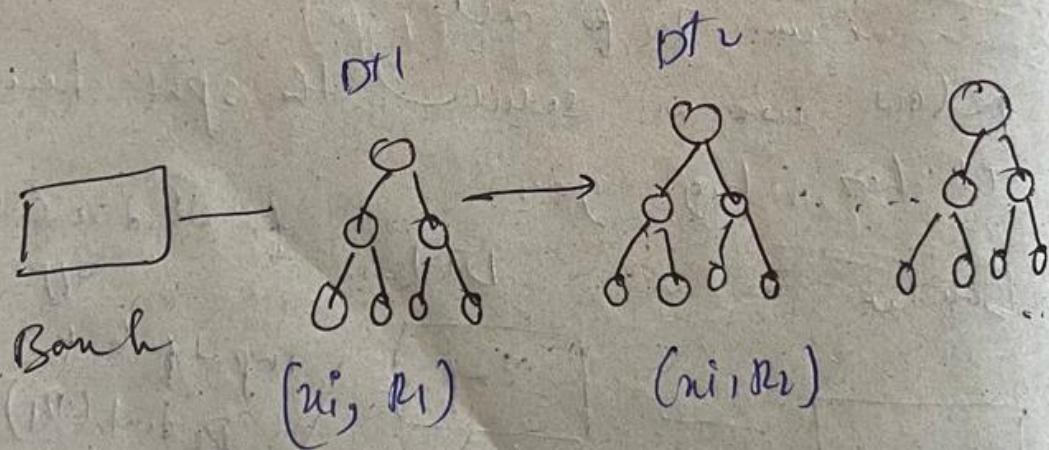
$$= 51k + 0.1(5)$$

$$= 51 + 0.5$$

$$= 51.5$$

=====

Smoothing need more $\lambda (DT_2)$ vs. Create very
rapid fluctuation and R_2 (overfit)



Snapping & g food clausur.

PIP until & g food.
eg busypm

Highway ag board Rogers

Unsupervised ML

- ① K-Means algorithm
 - ② Hierarchical Clustering
 - ③ DBScan Clustering
- Silhouette score
- validate

Unsupervised Learning

Supervised ML

Regime of
classifer

SIP Database f
f₁ f₂ f₃ f₄ OIP

unsupervised ML (no output)

↳ Clustering & group your data into
Similar Cluster

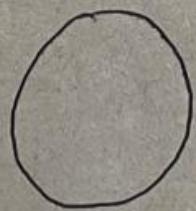
age expmt. salary.

—	—	—
—	—	—
—	—	—
—	—	—

cluster 1

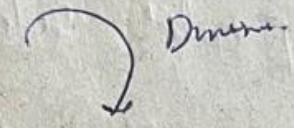
cluster

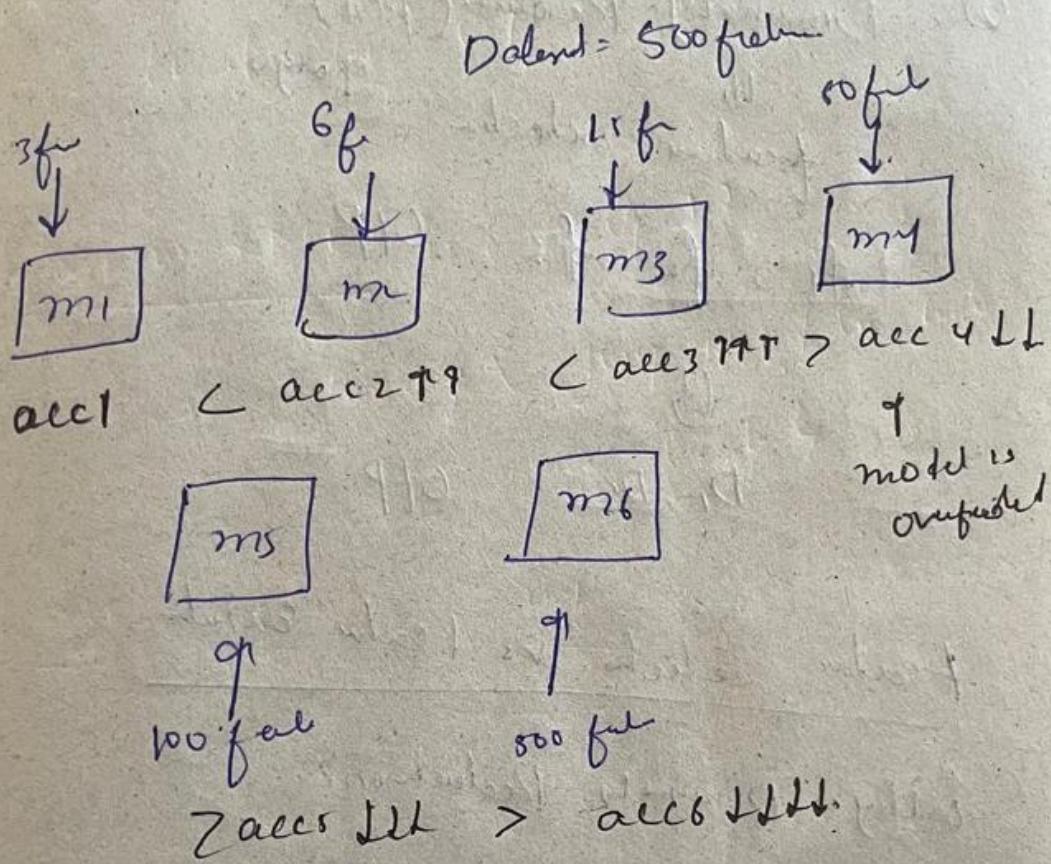
cluster 3



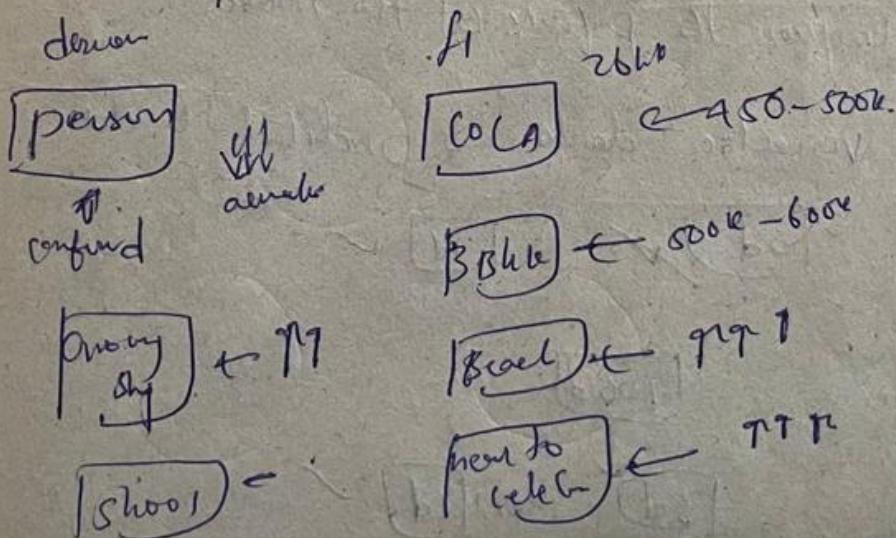
Principle Component Analysis (PCA)

[Dimensionality Reduction]

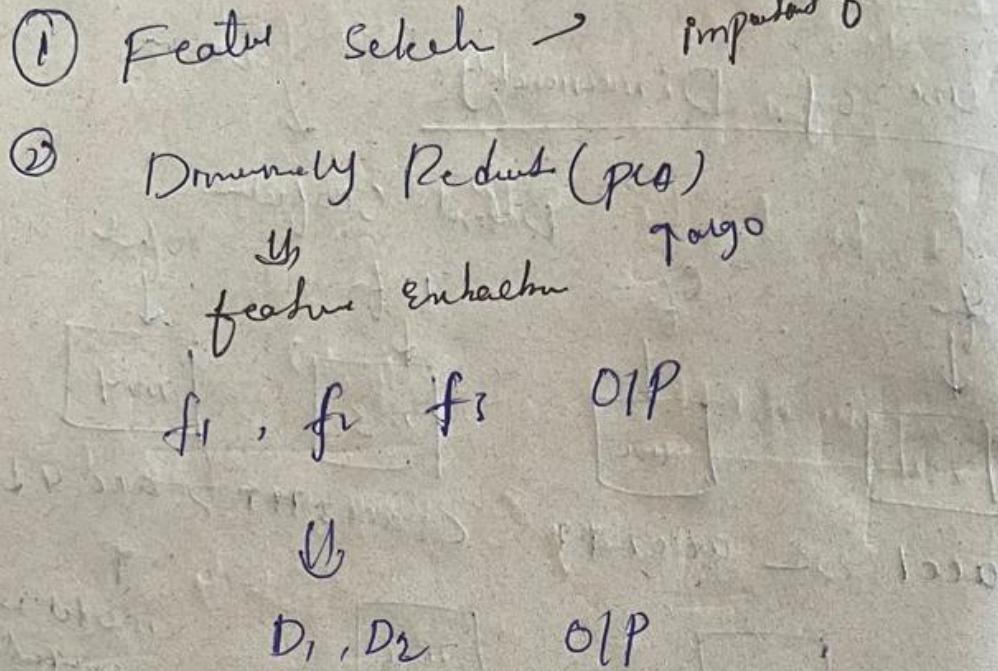
Curse of Dimensionality: 



→ model performance degrades.



Two different ways to remove curse of dimensionality:



Feature Selection vs Feature Extraction

Why Dimensionality Reduction?

- ④ Prevent Curse of Dimensionality.
- ⑤ Improve the performance of the model
- ⑥ Visualize data → understand the data

3d ✓ 2d ✓

100d
↓
 3d 2d ✓

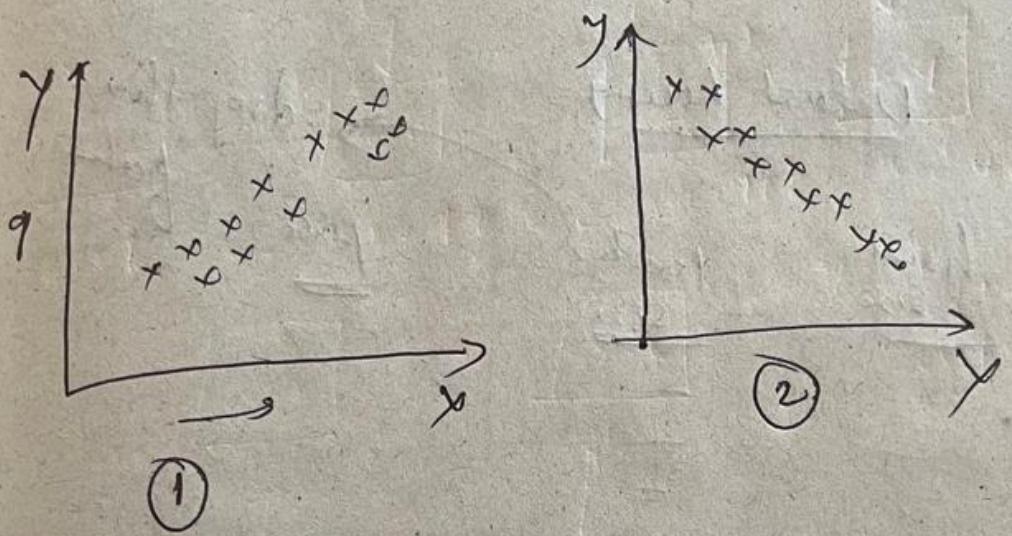
Franz Sebeke

①

x↑ y↑
x↓ y↓

②

x↓ y↑
x↑ y↓



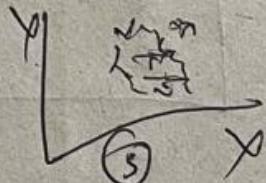
$$\text{cov}(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

$$= +ve \rightarrow \textcircled{1}$$

$$\approx -re \rightarrow 0$$

$$\approx -re \rightarrow 0$$

≈ 0 \rightarrow no reln \emptyset



range range if can be any positive value and any negative value

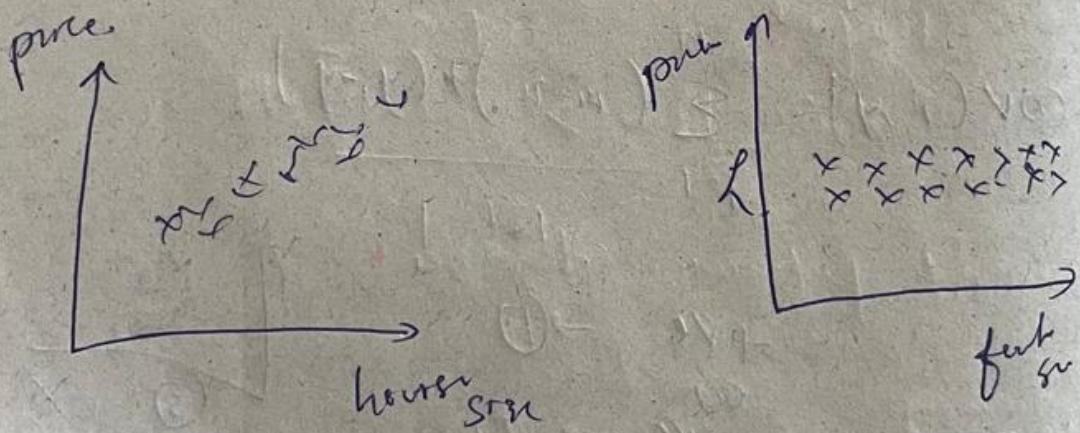
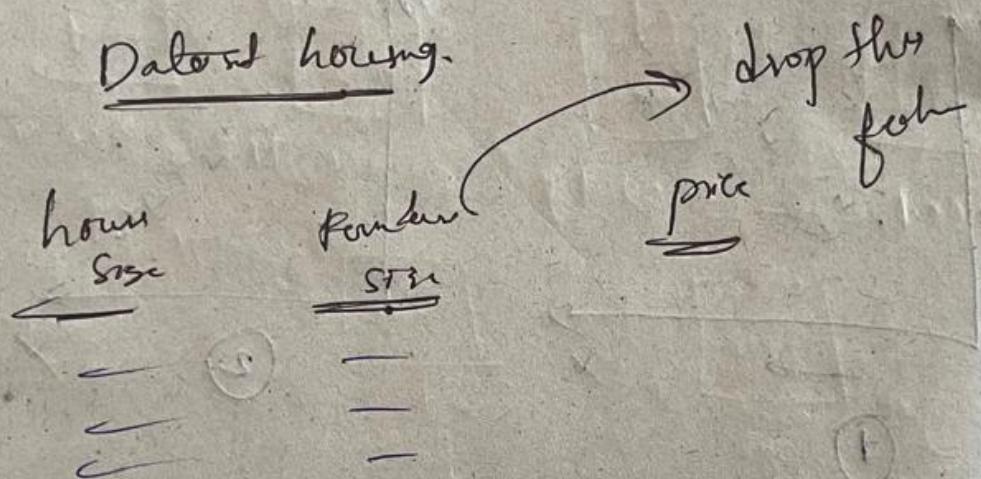
$$\text{Pearson Correlation} = \frac{\text{Cov}(x,y)}{\sigma_x \sigma_y} = -1 \text{ to } 1$$

more the value towards +1 \rightarrow pos correlation
 $x \& y$

more the value towards -1 \rightarrow -ve correlation
 $x \& y$

lower 0 \rightarrow non-linear

Data and housing.

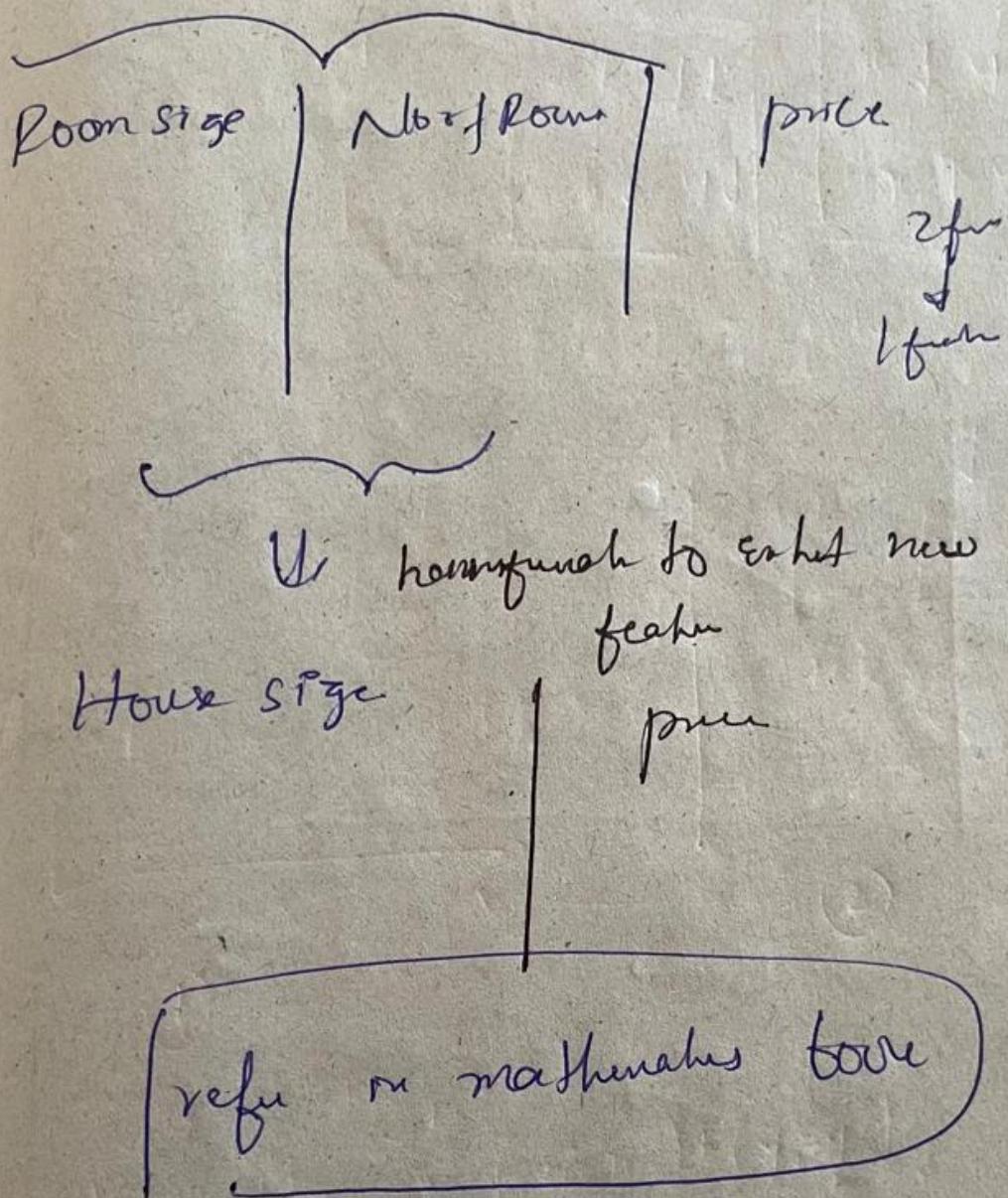


Closer relationship



≈ 0 no relation

Feature Extraction

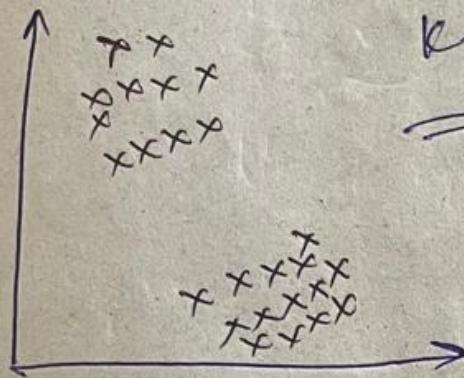


PCA Implementation

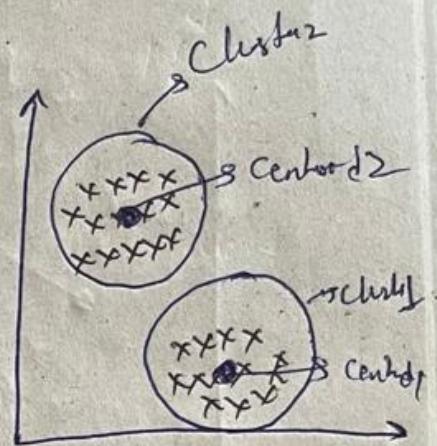
K-Means Clustering algorithm

Geometric Intuition

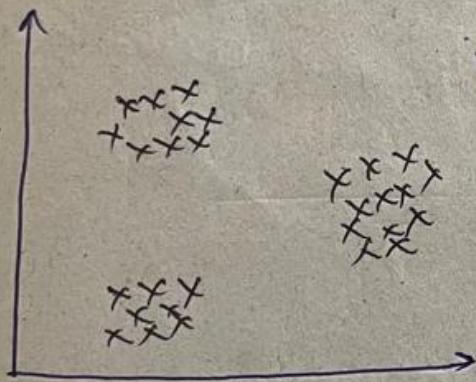
Data points.



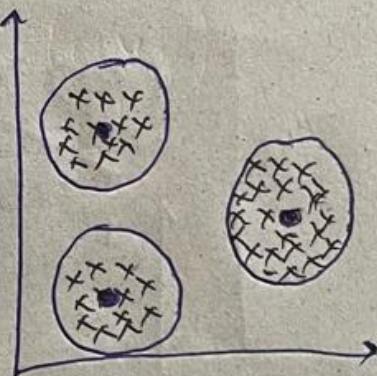
K-Means
→



means.



→



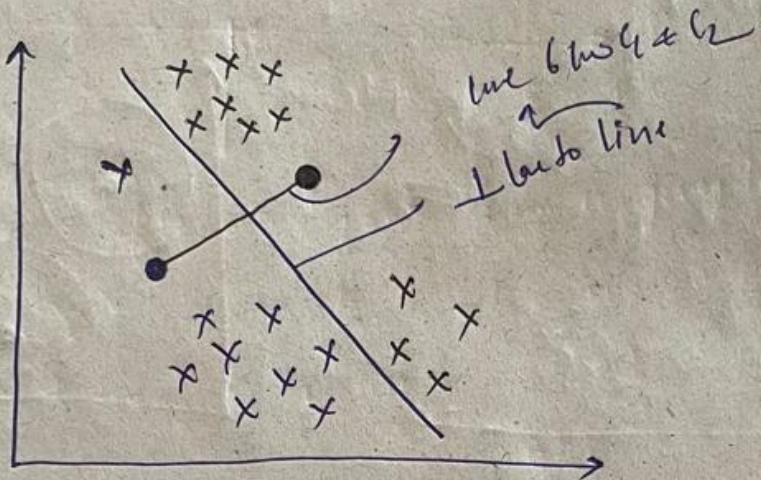
above:

we are going to group similar

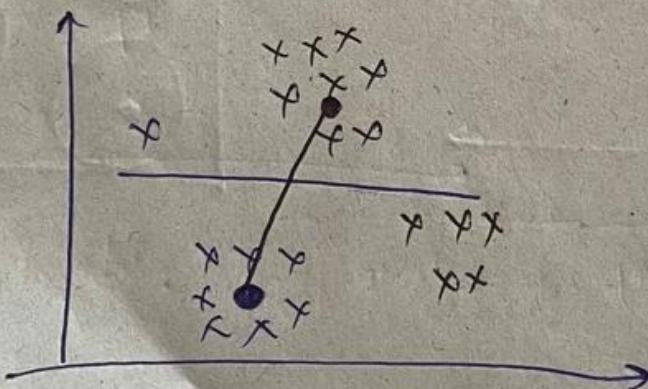
data points as clusters

K Means

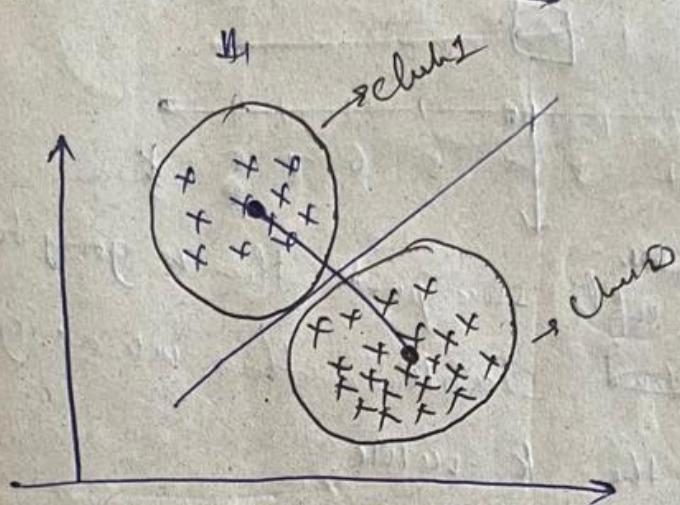
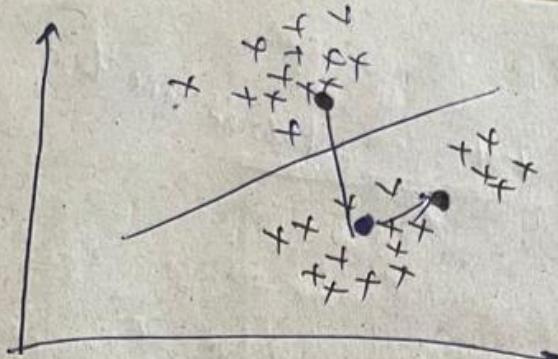
- ① Initialize some k Centroids
- ② Points that are nearest to the Centroids \rightarrow Group



- ③ move the Centroids \rightarrow Average
(Avg of points is new centroid location)



Now repeat the steps from step 2

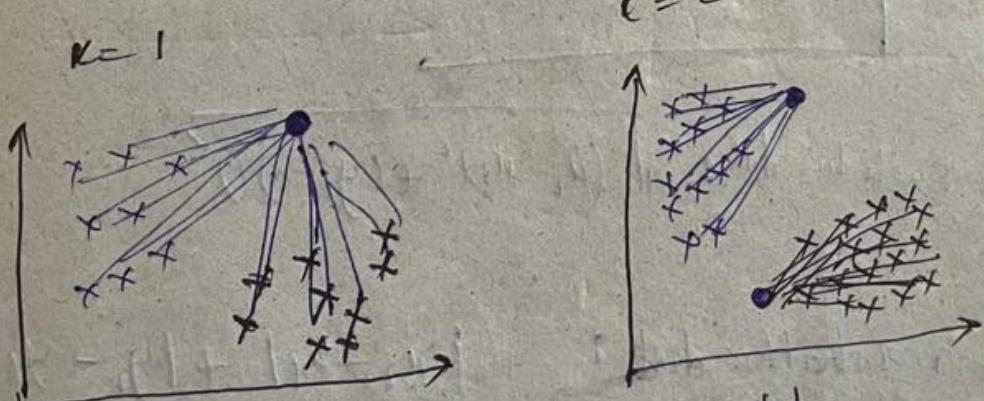


how do we Select k value?

WCSS \rightarrow [within cluster sum of square]

lets

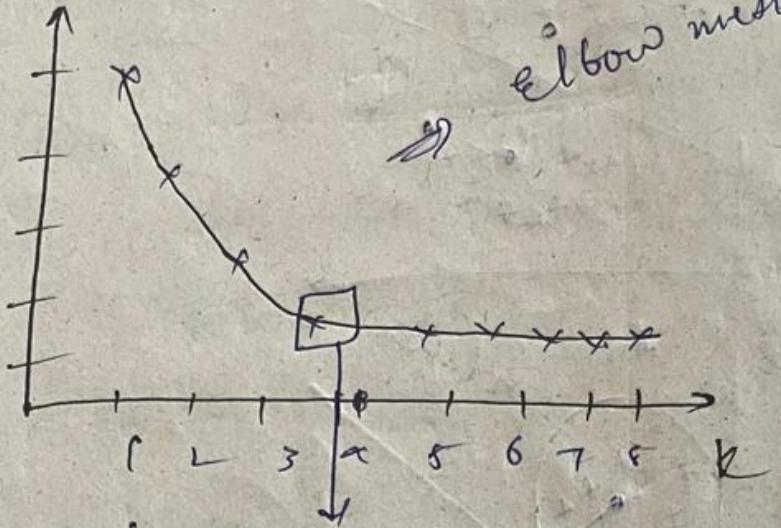
initial $k=1$ to 20



WCSS

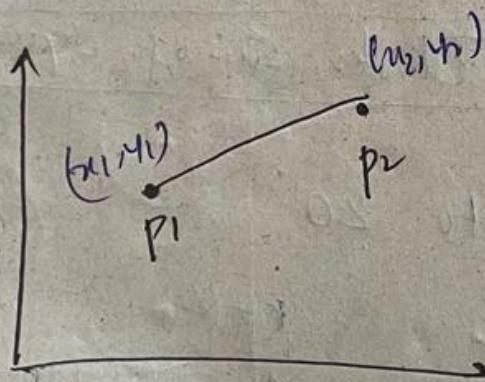
$$\text{WCSS} = \sum_{i=1}^S \left(\text{distance from point to nearest centroid} \right)^2$$

unless



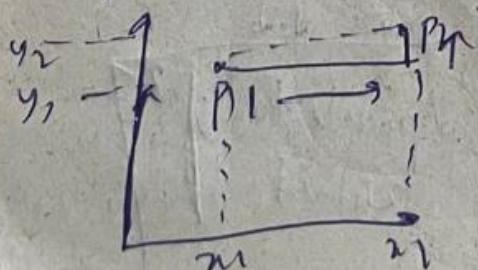
abruptly decreases
after some part this is going to be
a ~~converge~~ stable, then we are going
select the k value

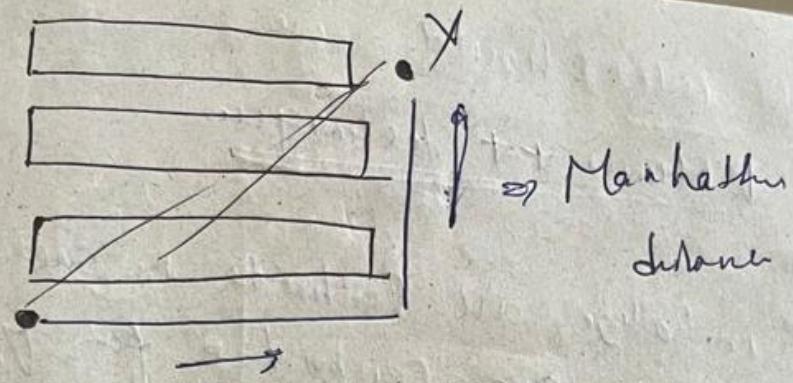
Euclidean distance



$$\text{Euclidean dist: } \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

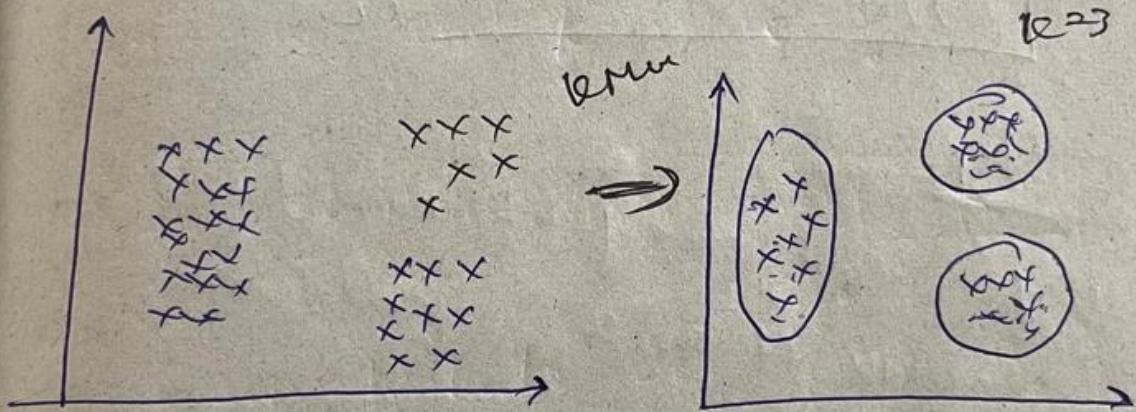
manhattan dist: $|x_2 - x_1| + |y_2 - y_1|$





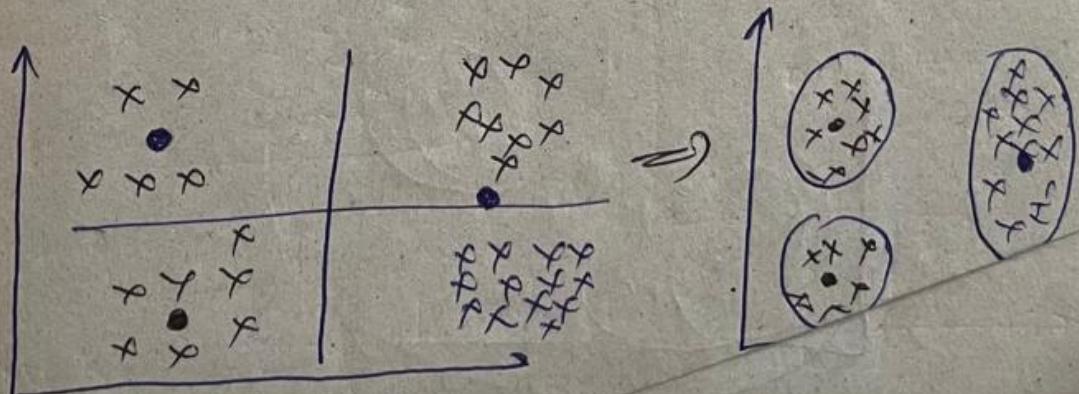
euclidean distance

Random Sampling Trap (k-means++)



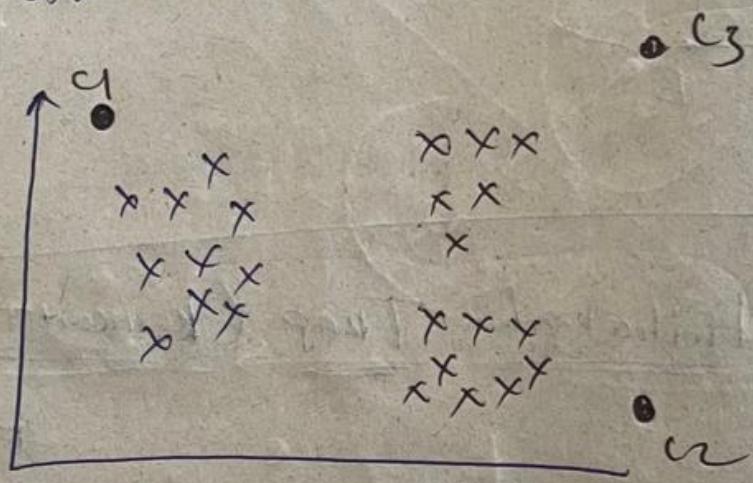
method of random initialization of centroid goes
without change ~~stars~~

lets



Then we should use
KMeans ++ technique

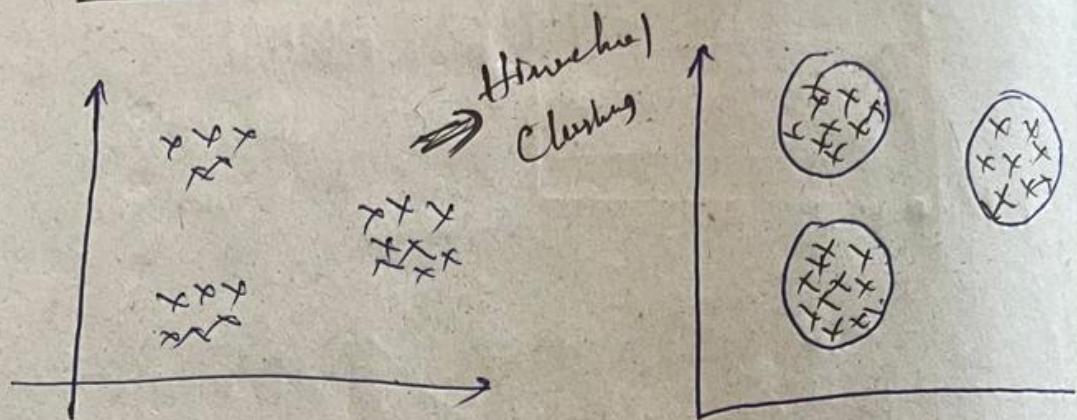
↓
we initialize all centroids in such a way that one centroid is far from the other centroid.



K-Means Clustering Implementation

Hierarchical Clustering

No
Centroid



Hierarchical Cluster

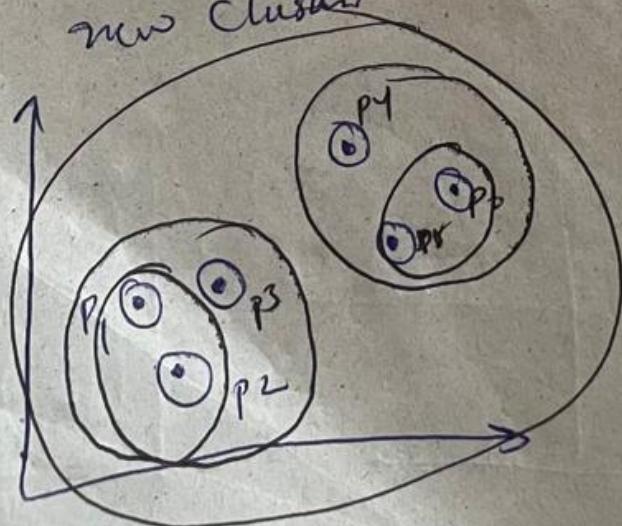
agglomerative

Divisive

Steps:

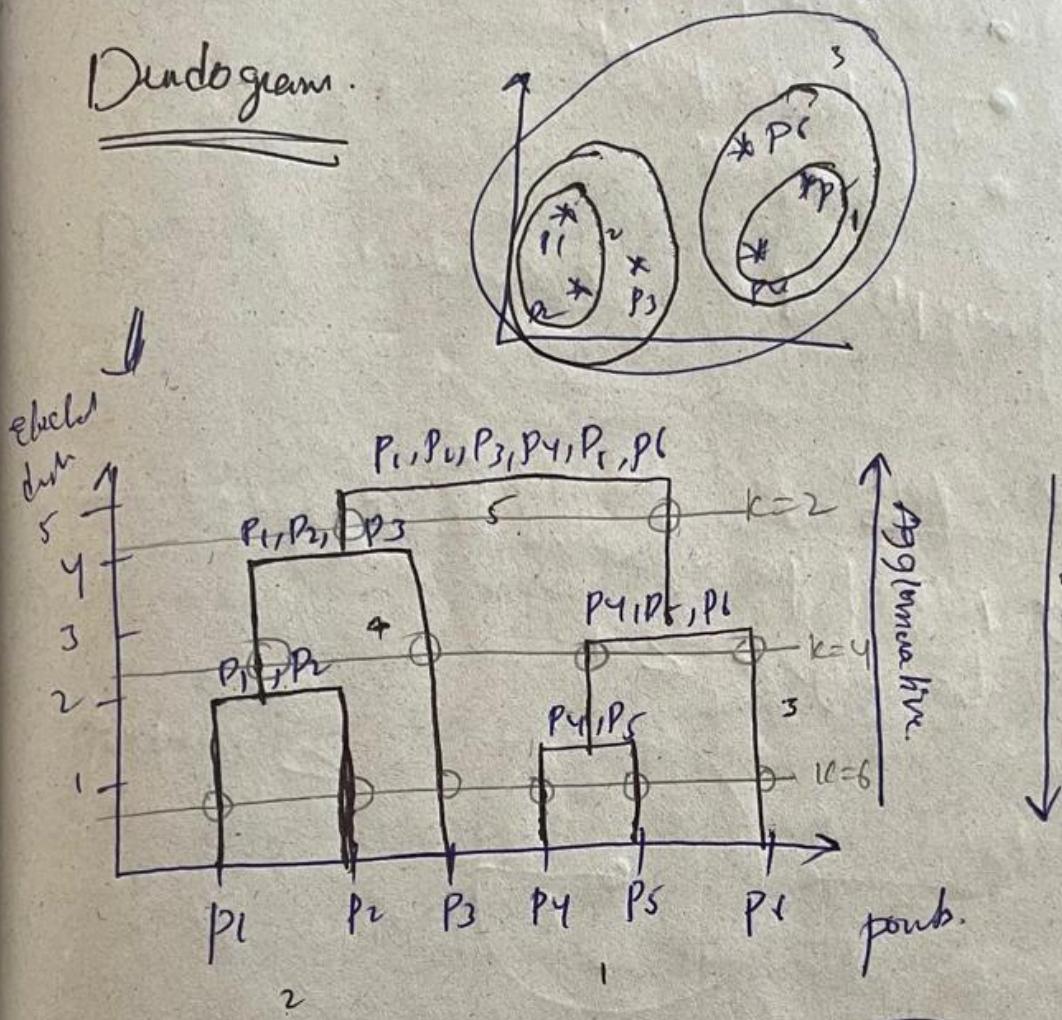
- ① for each data point initially with
consider it as a separate class

- ② find the mean point and create a
new cluster



③ Keep on doing the same process until we get single cluster

Dendrogram



how to decide no of clusters.

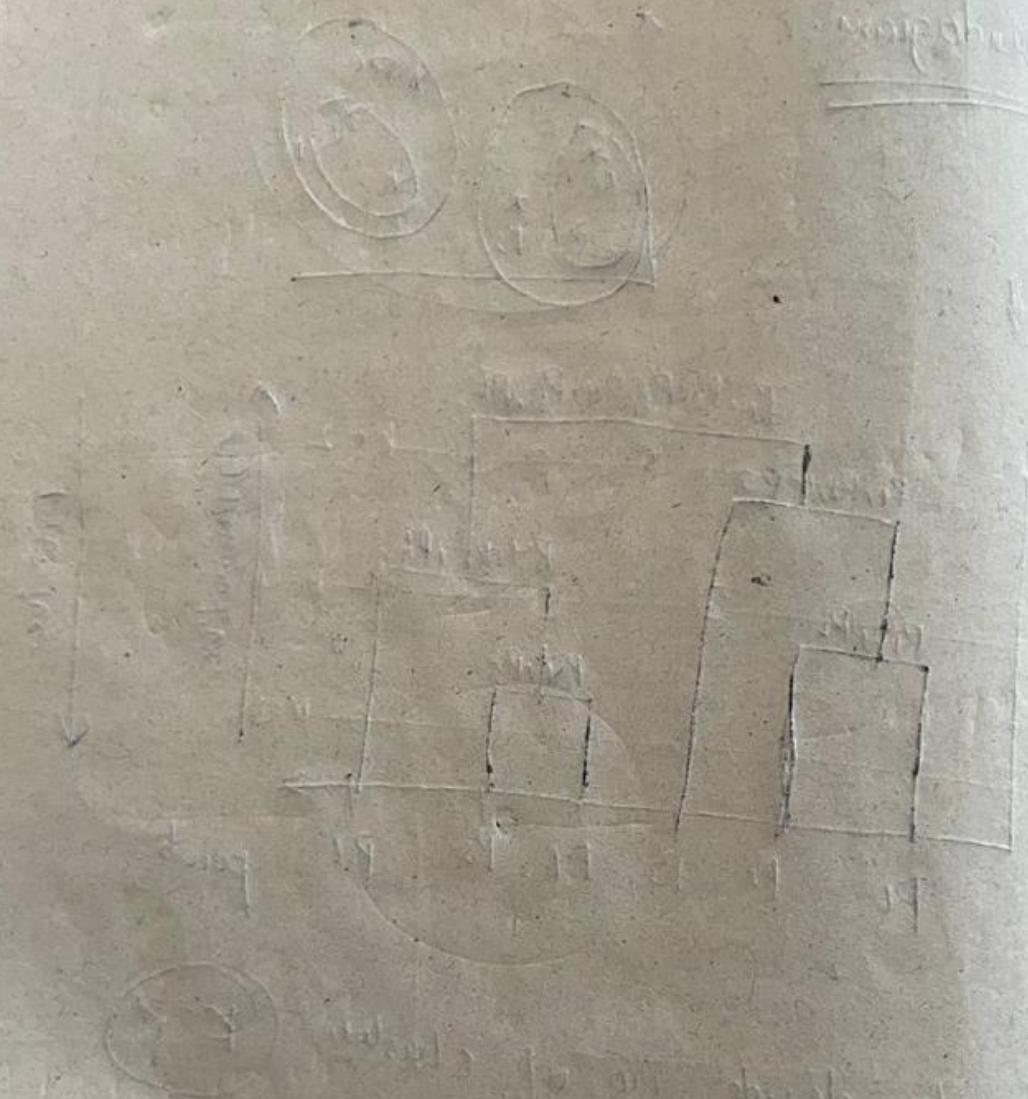
$$K=?$$

by setting up threshold value on eucleod dist

here to select the threshold.

Select the longest vertical line such that no horizontal line passes through it.

Agglomerative Clustering Implementation



K Mean Vs Hierarchical Clustering

Scalability and Flexibility.

① Dataset size → huge → K means

small → hierarchical

② Kmean → Numerical datasets

Hierarchical → not only numerical data ~~but~~ but

also when we use cosine similarity.

(Want of data) \Rightarrow

③ for visualization

K mean Centers → Elbow method

→ diff to find no of centers.

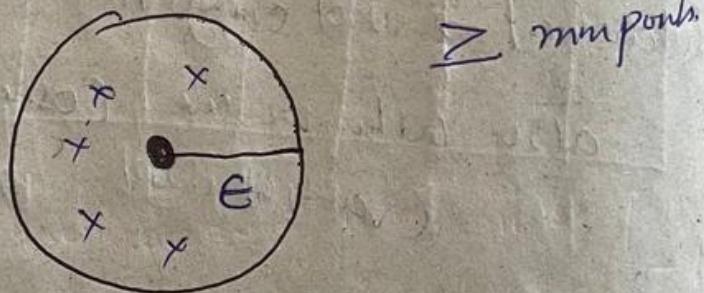
DBSCAN Clustering

- \rightarrow core point
- \rightarrow Border point
- \rightarrow outlier

$\boxed{\text{minpts} = 4, \epsilon = \text{radius}}$

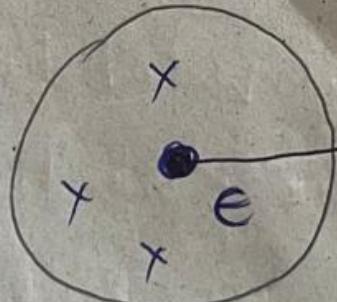
Core points

- ① No of points within the ϵ (radius) should
be greater ≥ 4 .



Border points

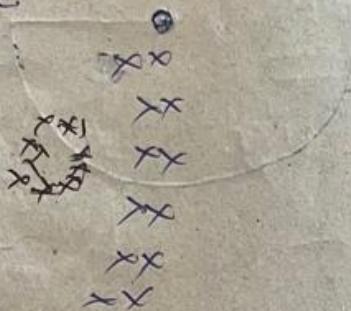
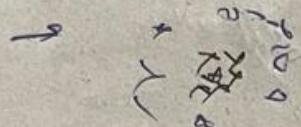
- No of data points within this radius will
be less than minpts



Adv & Dis of DBSCAN

→ no need to specify no of clusters

→ robust to outliers



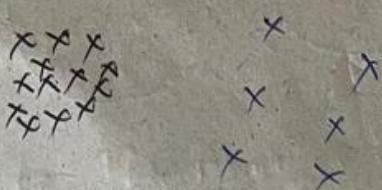
→ requires just 2 parameters

→ minpt and ϵ can be set by the domain expert

Disadvantage:

→ Some noise border points can be part of both the clusters

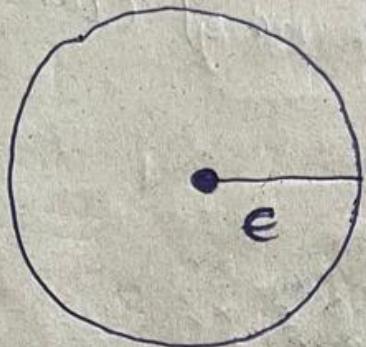
→ Cannot work well with clusters with large diff in density



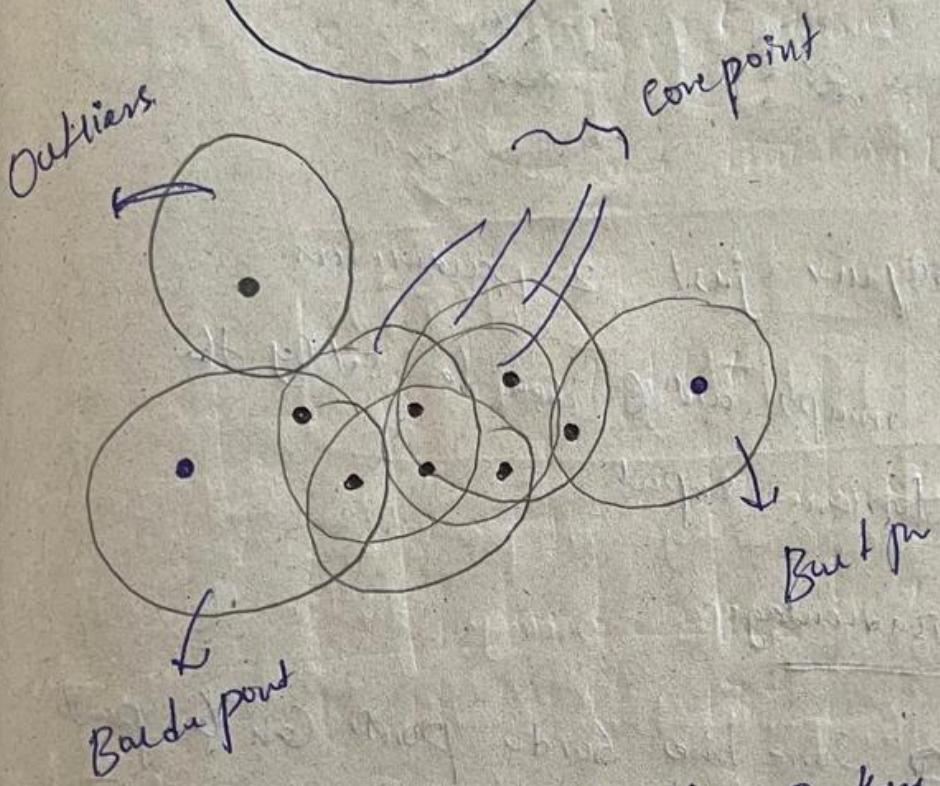
→ If the db and scab are not well ordered, choosing a reasonable cluster can be difficult.

③ Asterix (Noru)

if no other pond is existing within the radius



mmph29



→ Even though your data has outliers it's going to handle in amazing way.

→ DBSCAN also handle noisy clustering.

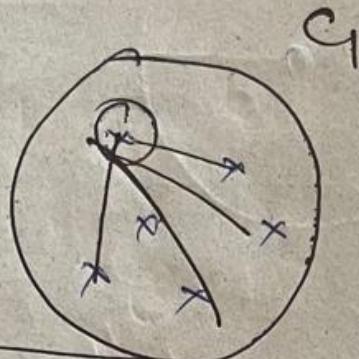
Impression of Boston.

[100b 3 1 - (100)
100c 1 - (100)]

SP/Harber Clustering.

To validate the clustering.

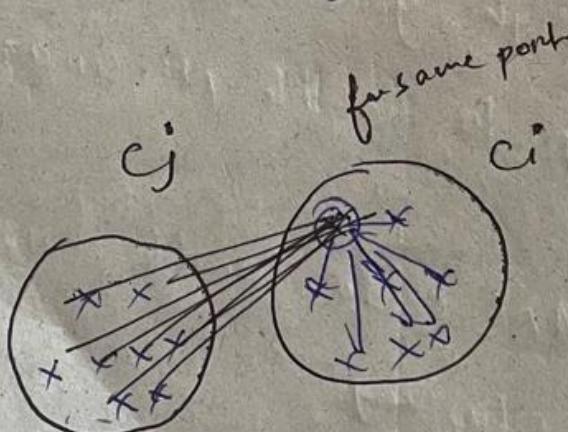
①



$$dci = \frac{1}{|C_1|-1} \sum_{j \in C_1, j \neq i} d(i,j)$$

Take one point from that we can calculate the average distance to all the other points, then sum of all the dist and divide it by no of points - 1

②



Now to find cluster for c_i to calculate the avg distance to all other points in C_j

$$b(i) = \min_{j \neq i} \frac{1}{|C(j)|} \sum_{j \in C(j)} d(i, j)$$

$$a(i) < b(i)$$

③ Calculate the silhouette score

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

$$[-1 \text{ to } 1]$$

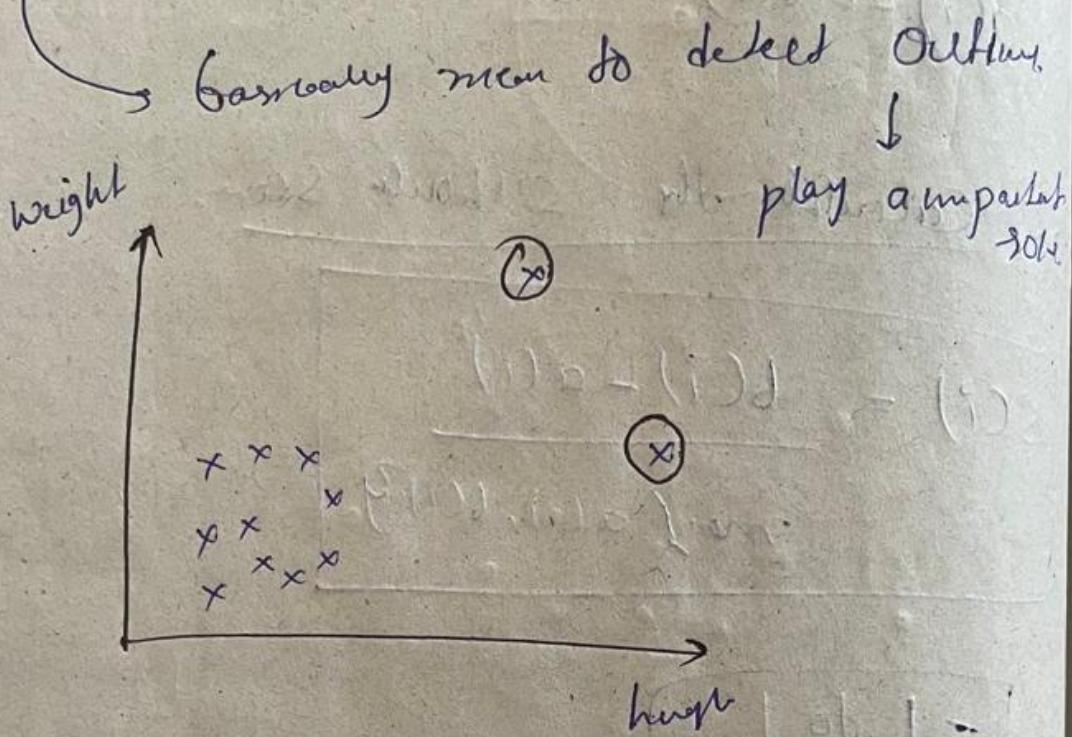
more new to 1 both clustering model we have created.

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/(a(i)-1) & \text{if } a(i) > b(i) \end{cases}$$

$$-1 \leq s(i) \leq 1$$

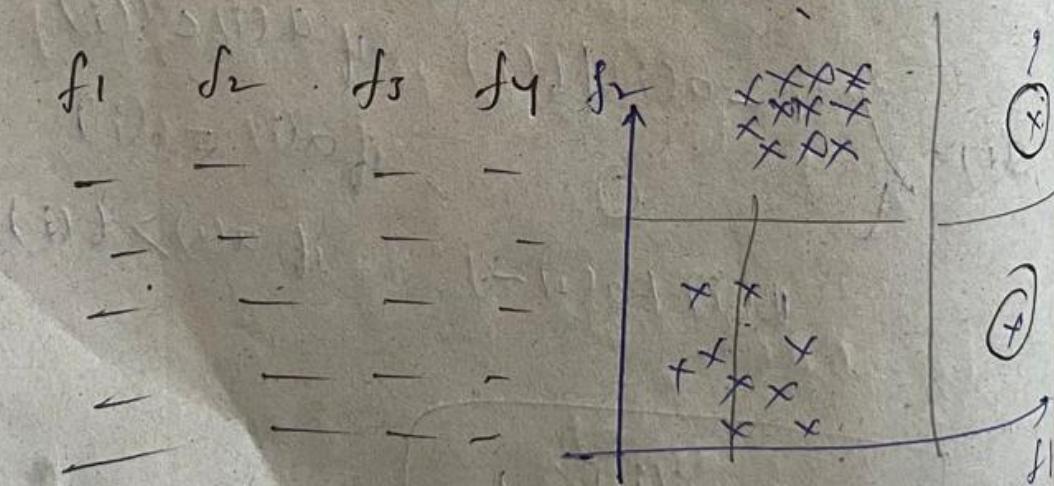
Anomaly Detection Using Isolation Forest

Indeepth Intuition



Isolation Forest (AD Technique)

↳ intially uses [Decision Trees]

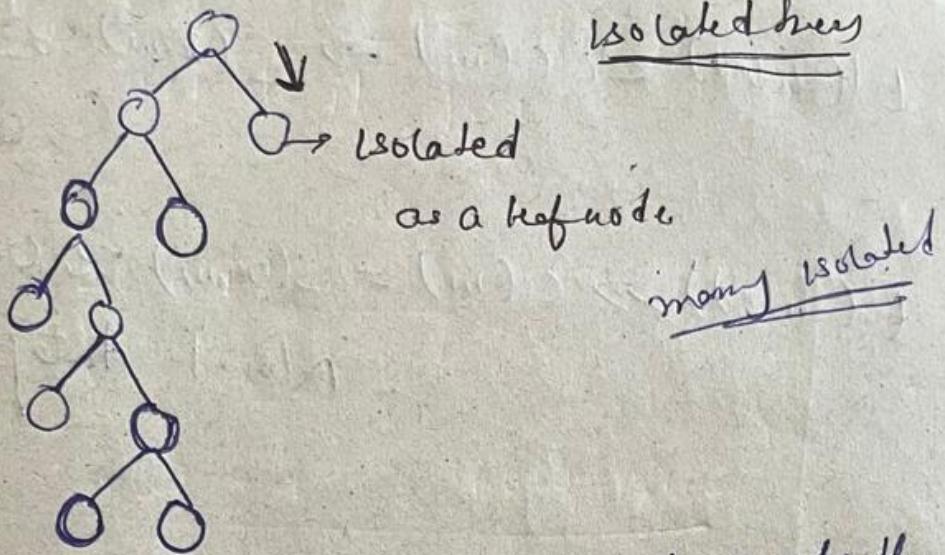


We are going to construct decision tree for all the data points.

(Hence ≈ 0.5)

$E(h_m) \ll C_m \Rightarrow S(n_m) \approx 1$
↳ outline

$E(h_m) \gg C_m \Rightarrow S(n_m) \approx 0.5$
None! datap on



- how quickly we are going to isolate them
- (also we will calculate the Anomaly Score)
- for each node, if it crosses the specific threshold, we consider it as a Outlier

Mathematical Formula (Anomaly score)

how to compute anomaly score for new point

$m = \text{no of data points}$

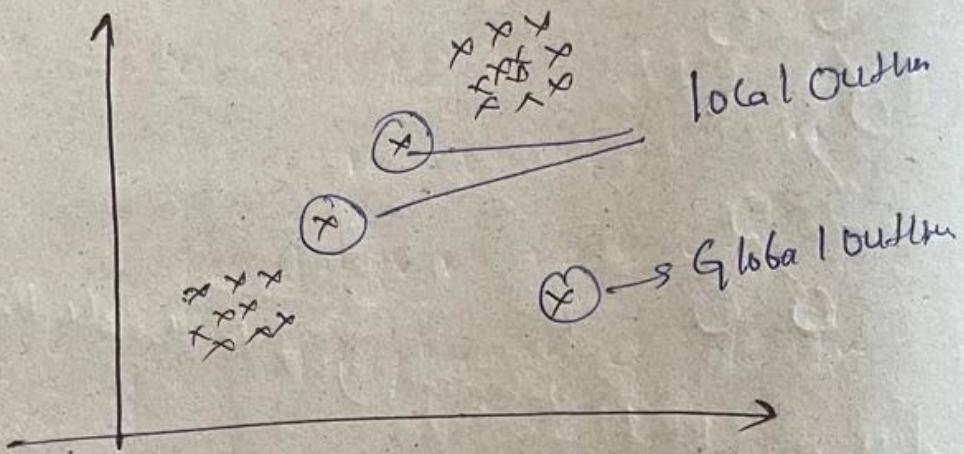
$$S(n, m) = 2 \frac{-E(h_m)}{C(m)}$$

$h_m \Rightarrow$ average search depth for $n^{(i)}$
from the isolated tree.

$C(m) \Rightarrow$ Average depth of h_m

DBSCAN Clustering Anomaly detection

Local Outlier Factor Anomaly Detection

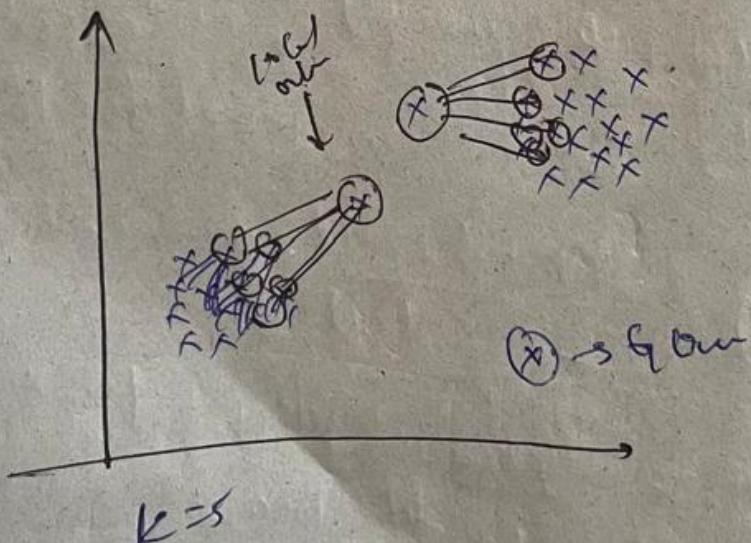


① Local Outlier \ominus

$[LOF] \rightarrow LOF \text{ score}$

② Global Outlier.

↳ Intuitively via k -Nearest Neighbor
Local density.



Comparing the local density of a sample to
the local density of k neighbor

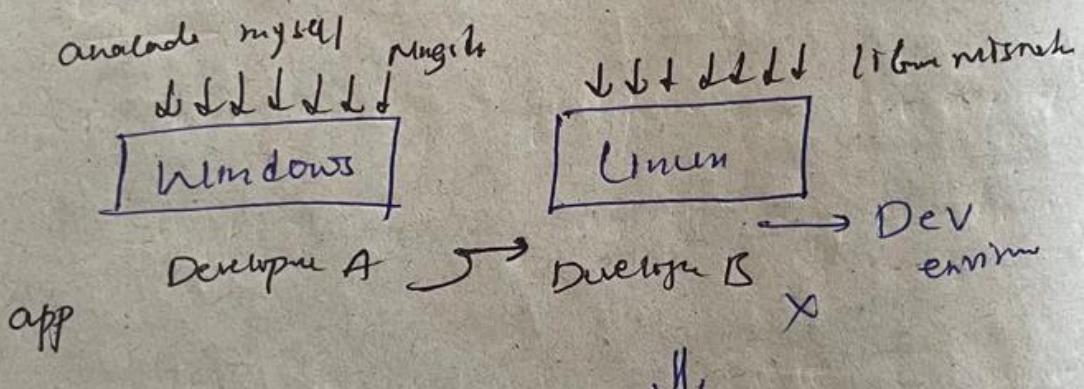
Implementing local culture factors

Docker Series

- ① What is Docker?
- ② What are Containers?
- ③ Containers vs Virtual Machines
- ④ Docker Images vs Container
- ⑤ Practical implementation of Docker

Docker and Containers

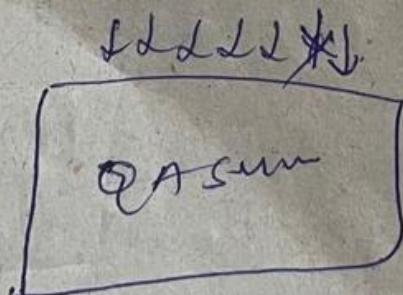
(Data science application)



Dev environment

II,

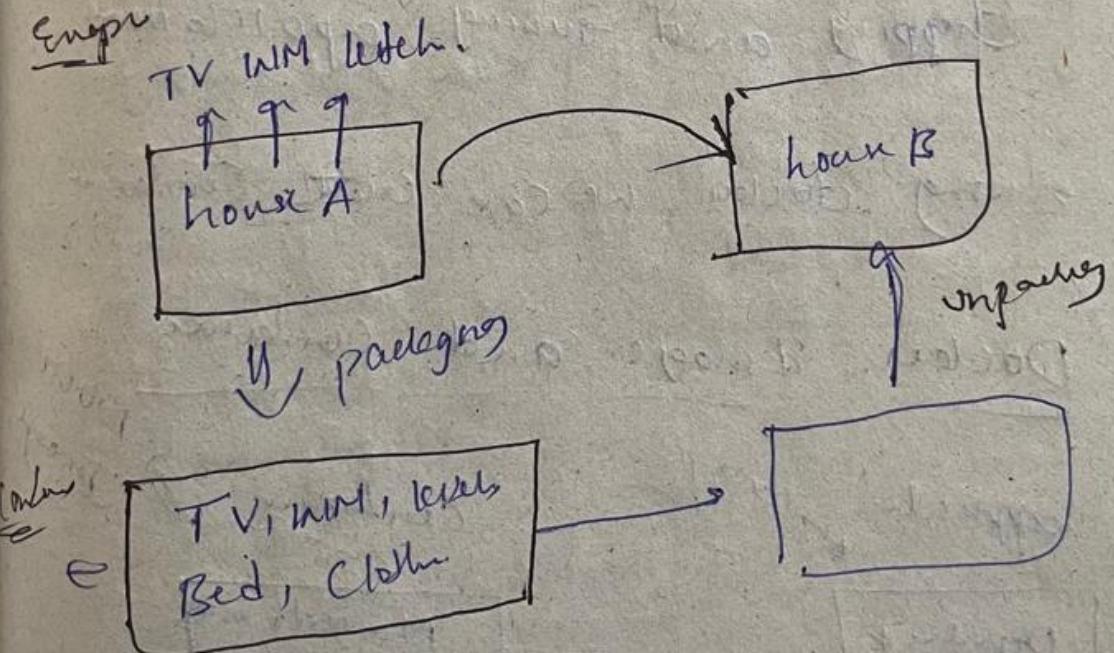
QA

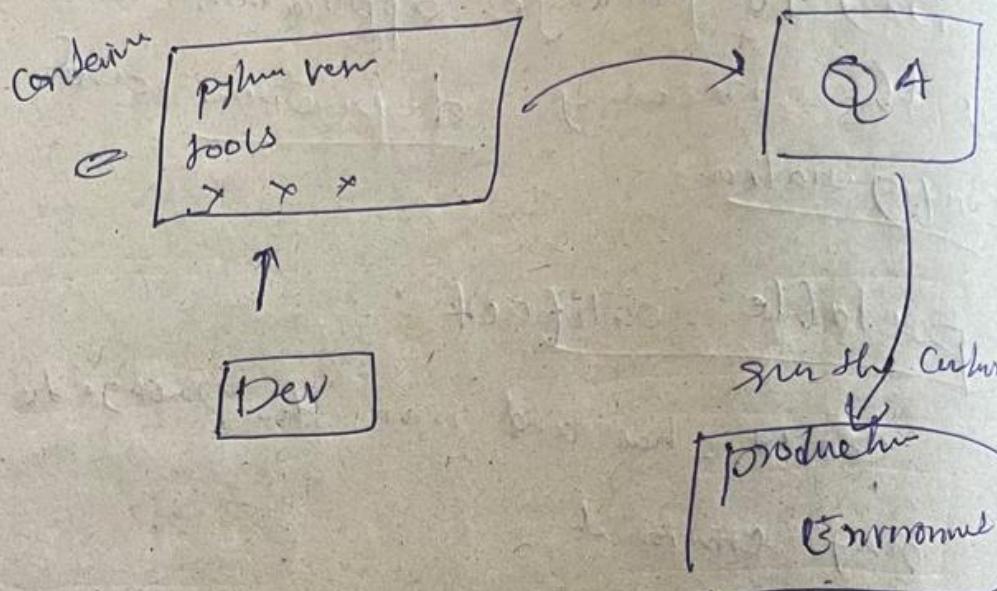


few modules may not work

Contains:

- A way to package application with all the necessary dependencies and configuration.
- Packable artifact
 - ↳ easier, share and move the package to any environment.
- This make development and deployment and more easy and efficient.



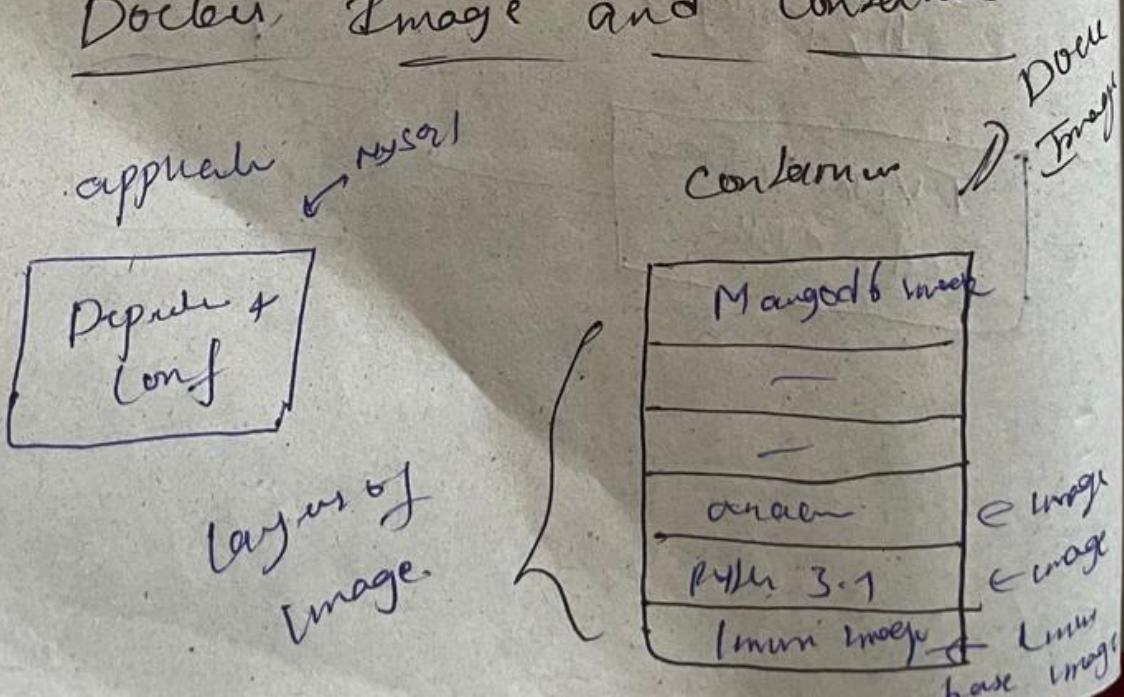


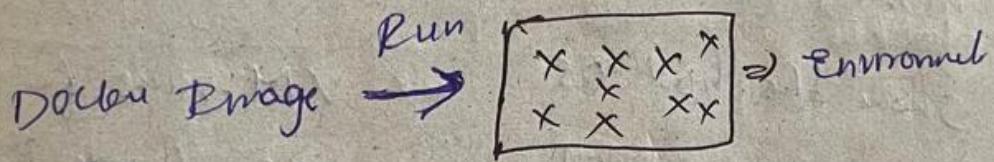
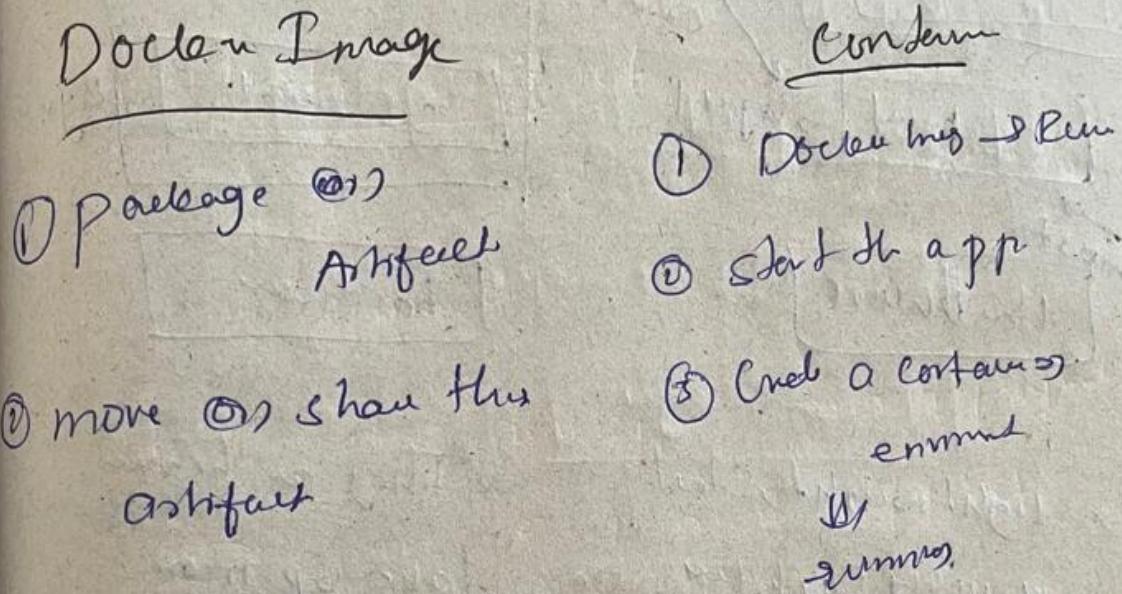
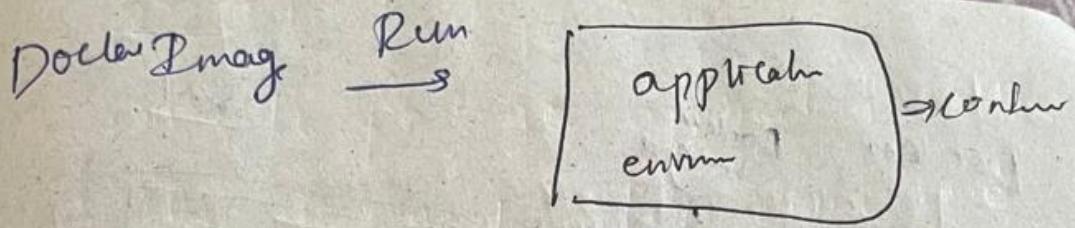
Docker

Docker is an open platform for developing, shipping and running applications

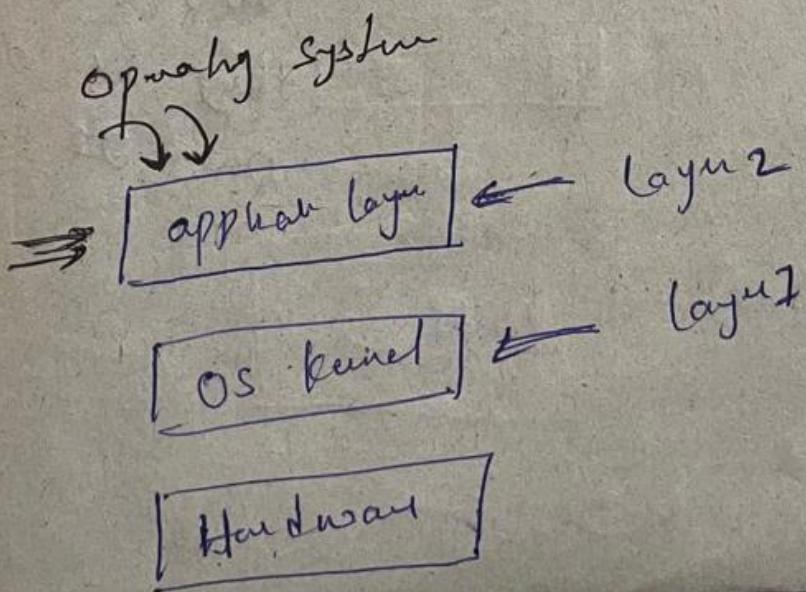
→ using Docker we can create Container

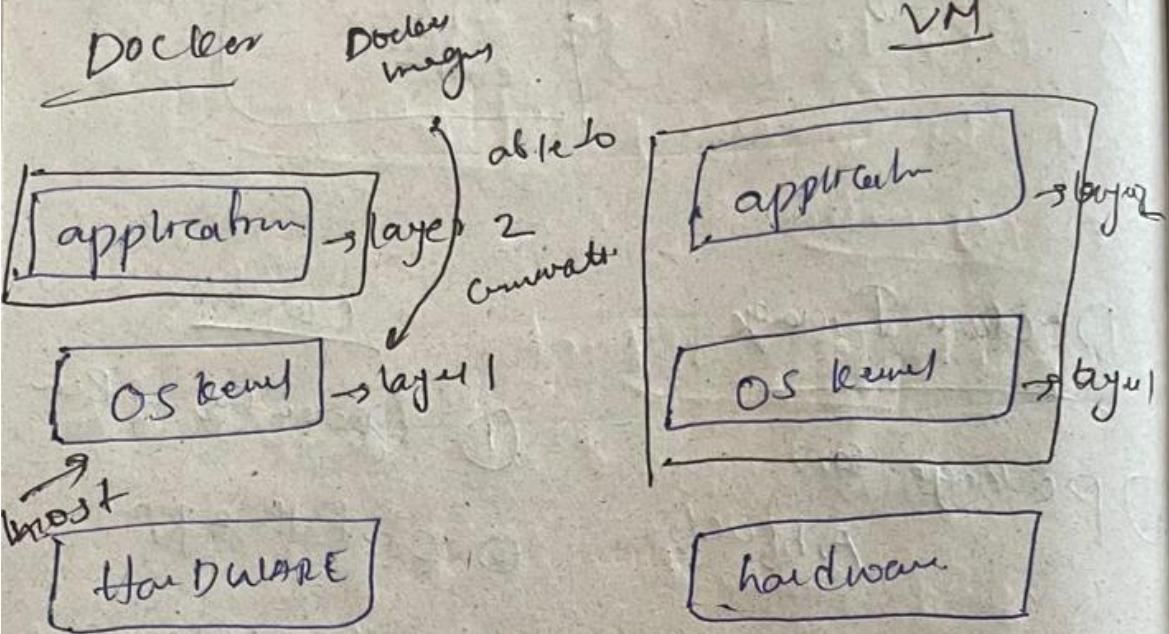
Docker Image and Container





Docker vs Virtual Machin.





- lightweight, start quickly, uses less memory
- share the host OS and use only memory reserved to run an app
- They are heavier, take up more resources (CPU, RAM)
- VM runs full OS inside them
- take longer time to run because they run on their own OS.

→ Docker customization and setup

→ Docker Basic Commands.

Docker pull command

docker pull docker/getting-started

G to check images

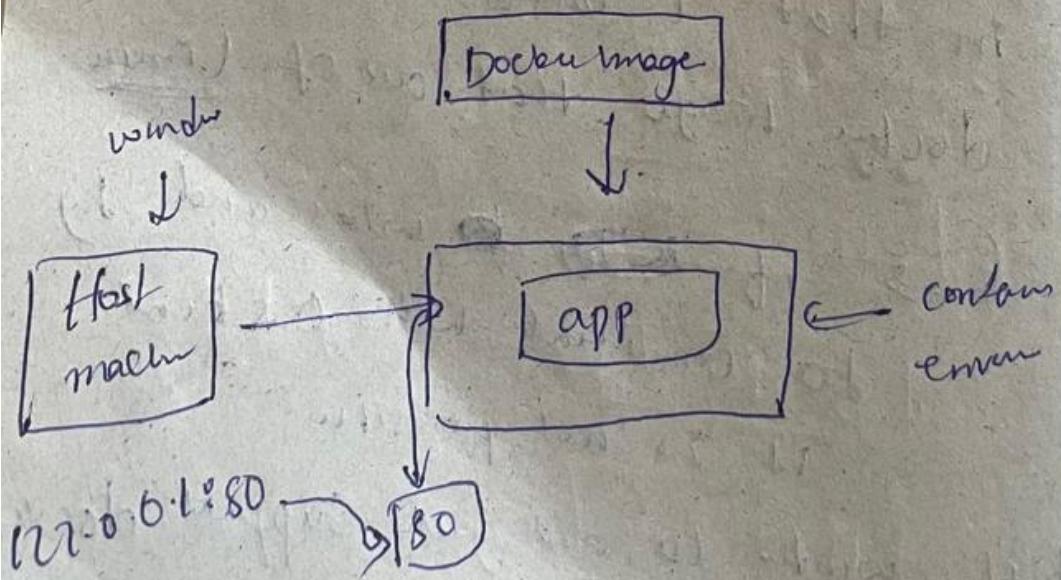
docker images

To run that image.

docker run -d -p 80:80 docker/getting-started

& detached mode

Container port and host port.



advantage and disadvantages

Docker

Storage

Docker image size is usually small (MB) → size will be huge (GB)

speed

→ Docker start and run much faster → VM is definitely slower

Compatibility?

→ We can install VM on any operating system

→ Docker images have compatibility issue

i.e.

i specifically have windows machine
in that I cannot install docker image that are of Linux

→ if ~~OS~~ is windows and try to communicate with OS kernel

it is not possible

but windows 10 and gate support docker Linux image

To check whether appn is rung or not

docker ps

local host can be access by

127.0.0.1:80

To stop the container

docker stop [container id]

to remove docker images.

docker image rm [image-id].

→ if wont work then

then remove it forcefully.

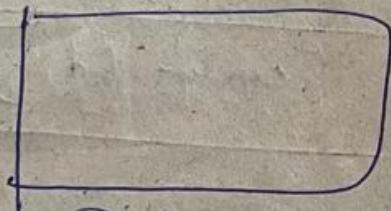
docker image rm -f [image-id]

another image (hello world)

docker pull hello world

docker run hello-world

We can enter shell container



hello
world

12.0.0.1:80 → (80)



Getting
started

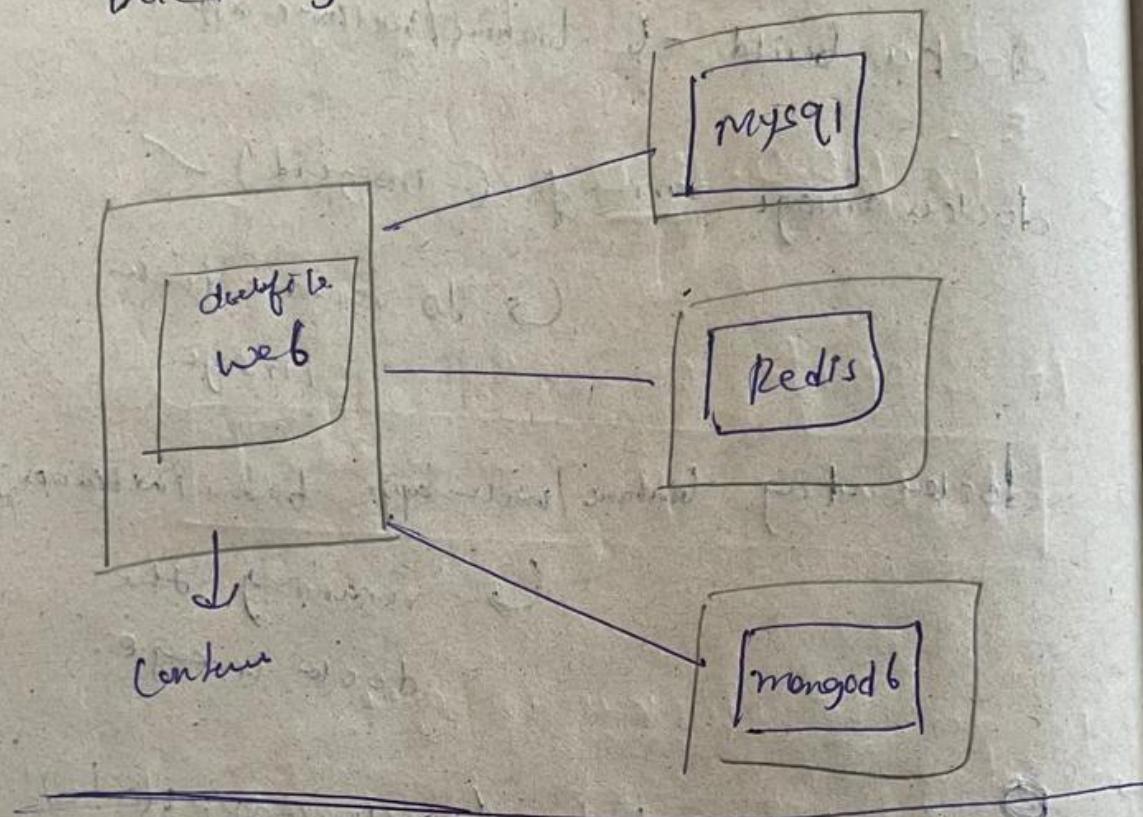
12.0.0.1:800 → (80)

after login to vs

- docker login ✓
- docker image ✓
- docker build -t brahme/welcome-app . ✓
 - ↳ Create docker image
- docker image rm -f (-rmegid) ✓
 - ↳ to remove docker image
- docker tag brahme/welcome-app brahme/welcomeapp1
 - ↳ renaming the docker image
-  docker push brahme/welcome-app:latest
 - ↳ to push docker image to repository

Docker compose

- To run multiple ~~different~~ containers we are going to use docker compose.



To remove ^{remote} Origin and add new origin

git remote remove Origin

to check → git remote -v

to add new orgn:

git remote add orgn "

Github

- go to the file in cmd
- git init
- git config --global user.name "Bhuvan"
 ↳ "Bhuvan"
- git config --global user.email "rayabhuvan@gmail.com"
 ↳ "rayabhuvan"
- git check
 ↳ "git check"
- git status
 ↳ "no tracking"
- git add README.md
 ↳ "git starts tracking"
- git commit -m "this is the first commit"

Sending to staging environment

↓
is a type of environment where we keep all files
which we need to send to repository

→ git status

↳ on branch master

adding Github repository
or Deploy

→ git branch ✓

→ git remote add origin

→ if there are multiple file to add

git add .

→ git remote -v

origin
me

To clone online git repository into
your folder

→ Create empty folder

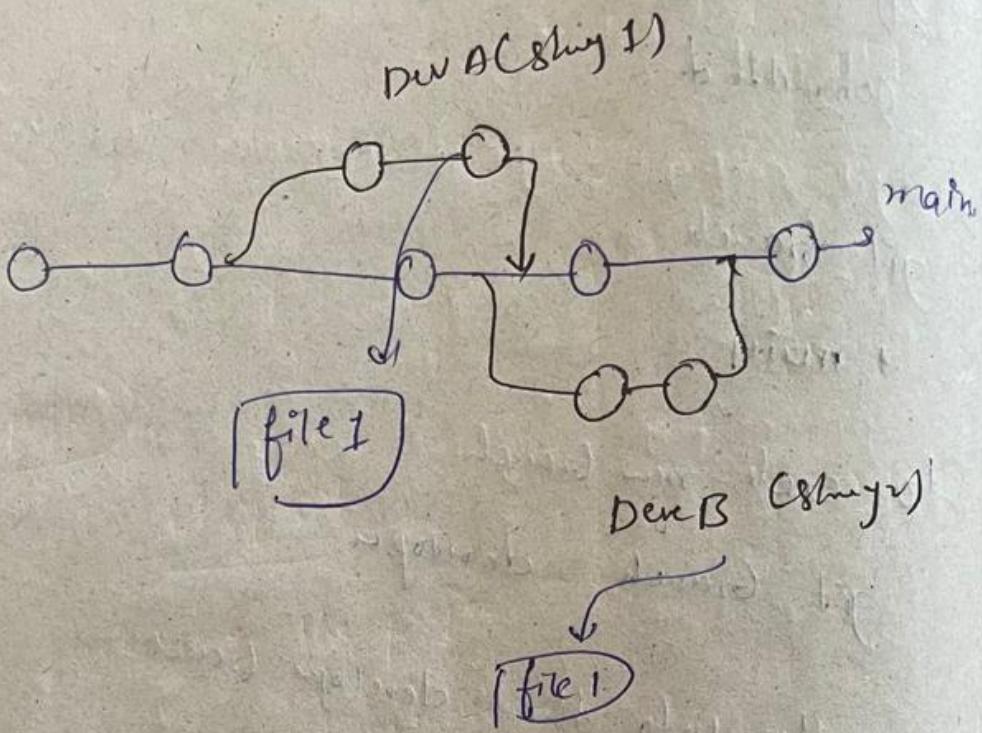
git clone http:// -- gitfun.git

if some one are updated the original code.

then pull it and push it again. Copy from
git code

git pull origin main --rebase

Resolving Git ~~merge~~ branch merge conflict



It is a good practice to create a new
branching when we make a new
story. It is a better practice together.

git remote -v
to check the same

if you are doing for the first time
git config --global user.name "Brahm27"
git config --global user.email "zaynabahmed@gmail.com"

git diff --staged

↳ files which are just added but not committed.

git branch

* main

to get new branch.

git branch develop

to enter into that develop branch

git checkout develop

do some commits on develop

switch to main

then

git merge develop

everything is in the staging environment

now push to repository

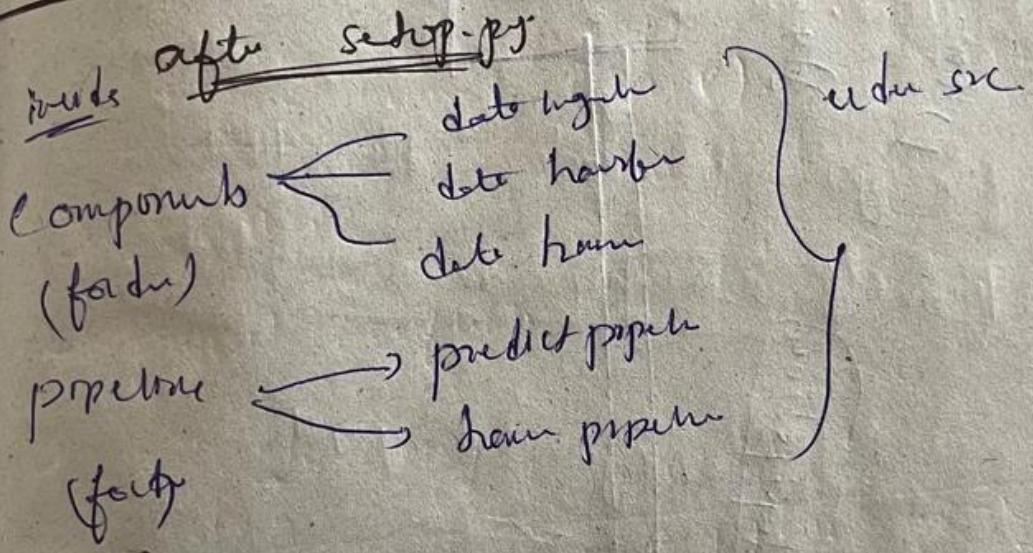
git log Shows Commits on the new branch
but has

② Logging and exception

logging.py → for log files.
exception.py

↳ for custom Exceptions

③ Discussing project problem Statement, EDA
and Model Training.



under src

exception.py

logger.py

utils.py

Getting started with MLOps with MLflow and Dagshee

MLflow is an open source platform to manage the ML lifecycle, including experiments, reproducibility, deployment and a central model registry. MLflow currently offers four components:

- ① MLFlow Tracking
- ② MLflow projects
- ③ MLflow Models
- ④ Model Registry

conda create -p min python=3.9 -y

requirements.txt > mlflow=22.1.0

Why mlflow is used

easier to manage the entire process of building,

tracking and deploying the models.

- ① track experiments (like parents and results of each run).
- ② organize code & models
- ③ Simplify Deployment

What is version control?

- ① Tracking changes
- ② Reverting to previous versions.
- ③ Collaborating with others.
- ④ Reproducing work

popular tools → Git for code
→ DVC for large data +
ML models

C DVC (Data Version Control)

even though we have github why DVC
to handle huge amount of data

Steps to work with DVC

- cmd: Conda activate venv python=3.9 -y
- 3 gitignore = venv/
- 3 Conda deactivate venv/
- 3 git init

practical work with BentoML ↗ pyh
3.8

- new folder
 - conda create -p venv Python==3.9 -y
 - requirements.txt
 - ↳ bentoML == 1.0.15
 - selekt-learn
 - Conda activate venv
 - pip install -r requirements.txt
 - train.py
-
- ↓
- bentoML simple model → clf

Save model

Save mod = bentoML save - Save-model
("Bns-clf")

→ python train.py

test.py to run the model

↓
for predictions.

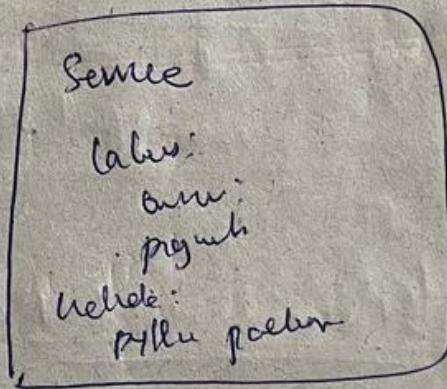
Service.py

Contains code for creating APIs and
Swagger UI

Run this you will get to check
your model

~~for creating Bento (as a full package)~~

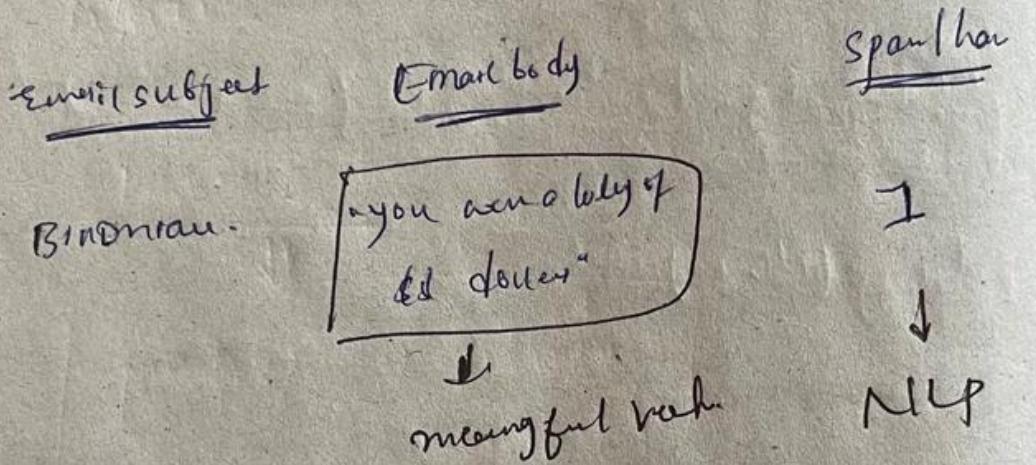
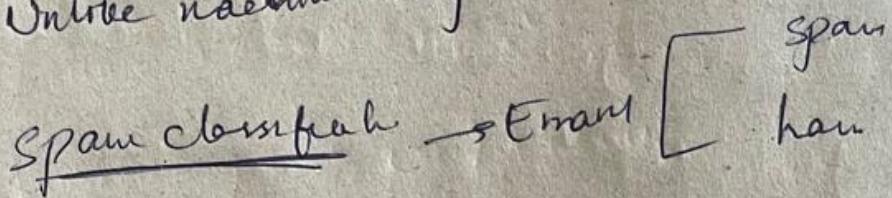
bentofile.yaml



→ bentoml build

Roadmap to learn NLP for machine learning.

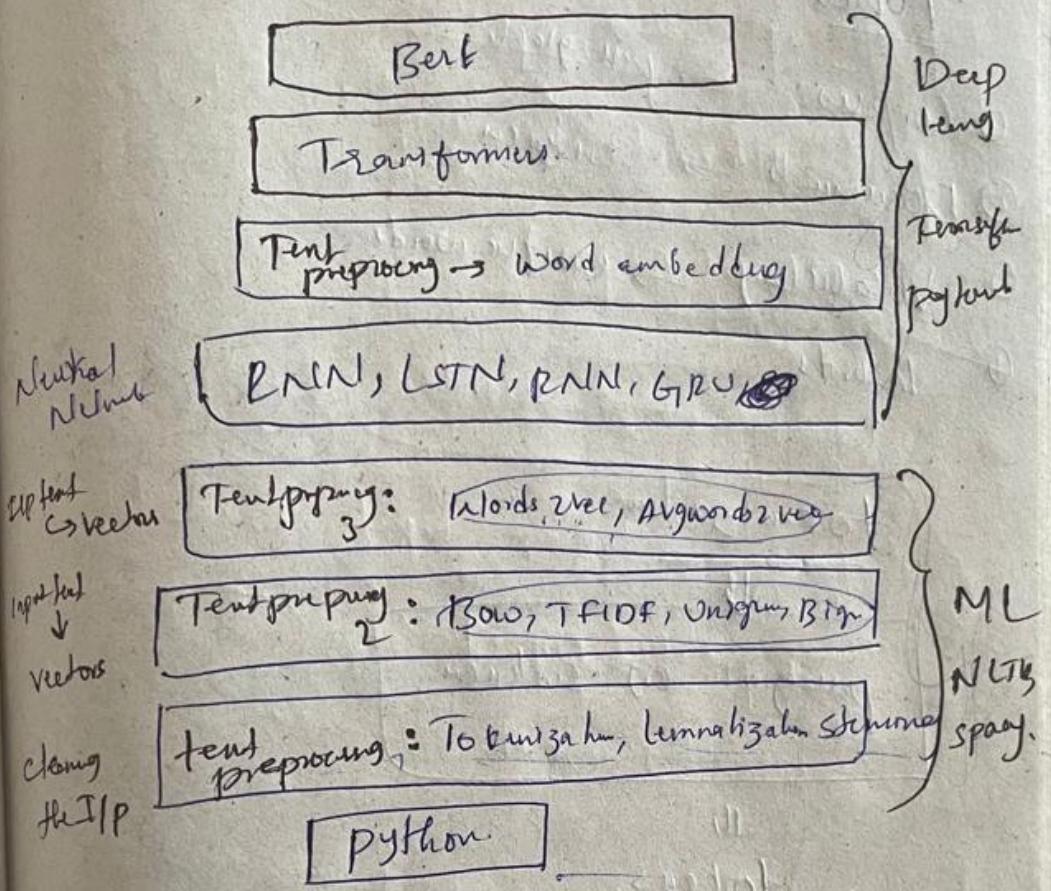
Unlabeled machine learning.



Practical use case of NLP:

- auto complete while typing (Gmail, chats etc.)
- automate replies in LinkedIn messages.
- translation of language.

Roadmap for NLP



Tokenization in NLP

Topics

- ① Corpus → paragraph
- ② Document → sentence
- ③ Vocabulary → unique words
- ④ Words

1:0 0 0
S

tokenization



paragraph or sentence

tokens

My name is Graham. I am an ai engineer.

I am also a youtuber



- ① My name is Graham tokenization → words.
- ② I am an ai engineer → my, name,
- ③ I am also a youtuber. → is, brother, I am

∴ In short words and sentence can be a token.

I like to drink apple juice. My friend likes mango juice.

Y to leenazah

tokens (sentences)

- ① I like to drink apple juice } 11 words
- ② my friend likes mango juice } words

Y Unique words (vocabulary)

10 words

(NLUke) and Spacy are used
for NLP

NLUke: broad, beginn. friendly, research
Slower

Spacy: faster, production-ready, practical
application

few important functions in Tokenization

ntle
dumb
("punkt")

Paragraph → sentence
→ from ntlc. tokenize input sent → tokenise

Senttokenize (Corpus)

• Sentence → words
→ from ntlc. tokenize input word → tokenise

wordtoken (Corpus)

mostly used

runey
run
news

wordpure tokenize → punctuation can also
break as diff word

Tree bankword tokenize →

C only last full stop is broken

separate word; all other full
stops are not broken

Stemming (Another Text processing Technique)

stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in NLP understanding (NLU) and NLP.

→ Why stemming:

→ to simplify words to their root so that they're easier to analyze

→ e.g. running, runner → run
connected, connectivity, connects → connect

→ helps computers to understand that these words have similar meanings, making it easier to search, analyse and categorise the text.

major Stemming Techniques (NLTR)

① Porter Stemmer

from ntlk stem import PorterStem

② Regexp Stemmer class

from ntlk stem import RegexpStem

use regular expression

③ Snowball Stemmer

from ntlk stem import SnowballStem

∴ simply stemming is nothing but reducing
the word to its stem

Word Stem: Standard form for language processing,

may not be a real word
history → histori

Root word: Original word with more
history → history

Stemming $\xrightarrow{O/P}$ Word stem.

Lemmatization:

It is a technique like stemming. The output we will get after lemmatization is called 'Lemma' which is root word rather than word stem / root stem, after lemmatization we will be getting a valid word that means the same thing.

lemmatization $\xrightarrow{\text{OIP}}$ root word.

from nltk.stem import WordNetLemmatizer

obj = WordNetLemmatizer()

obj.lemmatize (~~words~~)

5
82

6
63 83

Tend preposing - Stop words

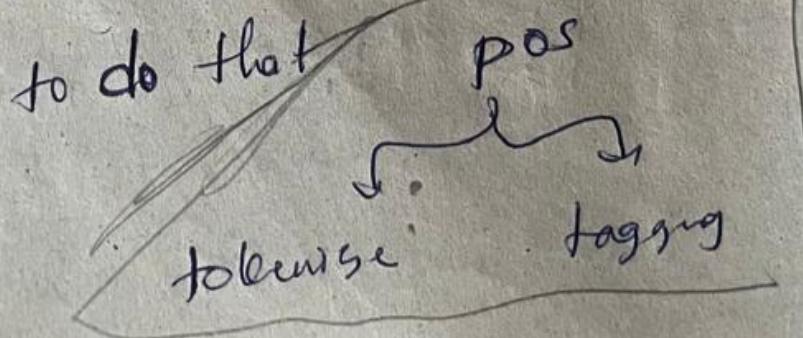
Stop words are the words that don't add much meaning to the sentence and are often removed to simplify text analysis.

→ a, the, and, is, an, or etc...

Parts of Speech Tagging using NLTK

→ pos in NLTK is about labeling each word in a sentence with its role, like noun, verb, adjective, etc..

Imagine you have a sentence, and you want to know the job of each word - pos tagging helps us



Named Entity Recognition

(NER) in NLTK is about tokenizing words

- ① phrase in a sentence that represent specific entities like names of people, locations, organizations, dates, etc.

e.g. Brahmin was born in Mangalore)

person → Brahmin

location → Mangalore

steps

→ tokenize the sentence

→ pos tagging

→ NER

result = nltk.ne_chunk(pos tags) or .chart

pos(tags)

↓
to get
tree

One hot encoding

Tent

OP

D₁ The food is good 1

D₂ The food is bad 0

D₃ Pizza is amazing 1

V=7

Vocabulary (unique words)

no of unique words

The food is good bad pizza amazing

D ₁	1	0	0	0	0	0	0
	0	1	0	0	0	0	0
	0	0	1	0	0	0	0
	0	0	0	1	0	0	0

$$D_1 = \begin{bmatrix} [1\ 0\ 0\ 0\ 0\ 0\ 0] \\ [0\ 1\ 0\ 0\ 0\ 0\ 0] \end{bmatrix},$$

$$\begin{bmatrix} [0\ 0\ 1\ 0\ 0\ 0\ 0] \end{bmatrix}$$

$$\begin{bmatrix} [0\ 0\ 0\ 1\ 0\ 0\ 0] \end{bmatrix}$$

4×7

↓

vocabulary.

no of
words

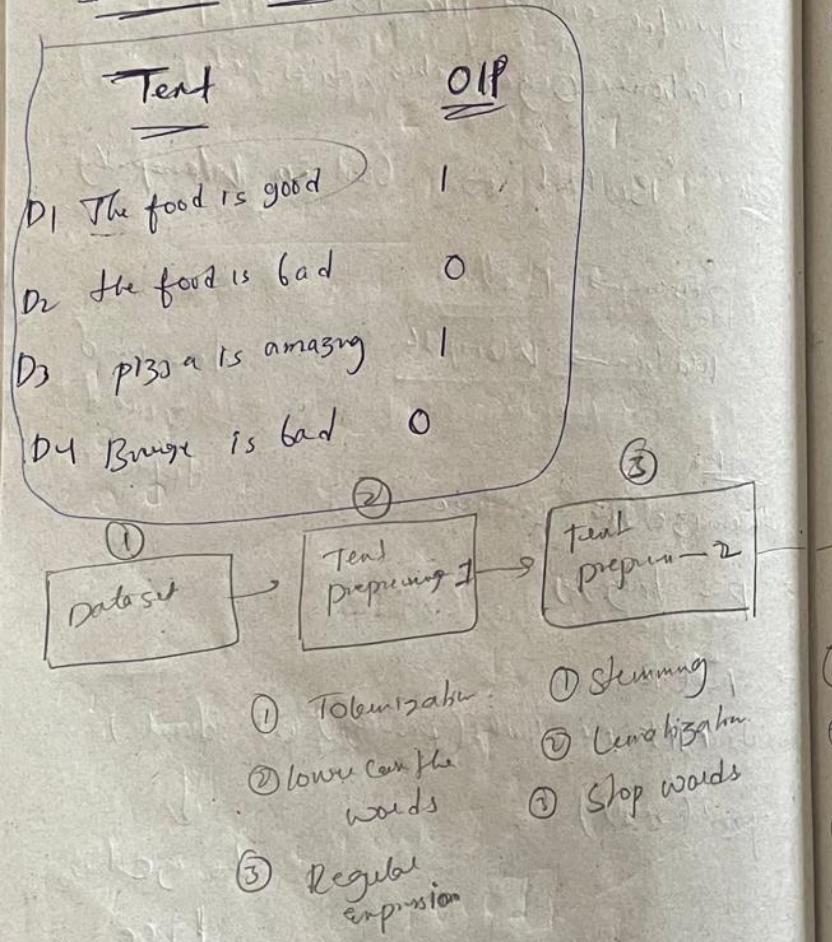
in

D₁

What's Next

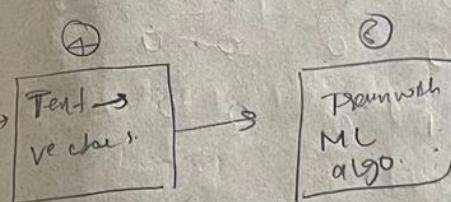
Text processing What we have learnt?

Surface analysis:



Text → vectors

- ① One hot encoded
- ② Bag of words (BOW)
- ③ TF-IDF
- ④ Word2Vec
- ⑤ Avg word2Vec



① One hot encoded

② Bag of words (BOW)

③ TF-IDF

④ word2Vec

↑ use can harm
custom w2w

⑤ Avg word2Vec.

Bag of words

Dataset

Text

OP

he is a good boy

1

she is a good girl

1

Boy and girl are good

1

1) lower all the words

(boy, Boy) X

stop words

→ 1

(is, and, or) X

after this

s₁ → good boy

s₂ → good girl

s₃ → boy girl good

Vocabulary (unique words)

frequency

good

3

boy

2

girl

2

SIMPLY for D2

4x7

for D3

3x7

advantages and disadvantages.

① adv

easy to implement with python.

[Stream Onehot encoder, pd.get_dummies()]

② disadv some vocab size

① Sparse matrix \rightarrow Outfitting.

② ML algorithms \rightarrow fixed size $W \in \mathbb{R}^{(7n, 3x7)}$

③ No semantic meaning is captured

④ Out of vocabulary (OOV)

pizza is intense

↓
who don't have west in
vocabulary.

We can't find a way to ~~convert~~
convert to vector

In ~~Boo~~ BOW we don't need to consider vocabulary with frequency 1. By some times we can select top ^{best} ~~1~~ vocabulary with more frequency.

 based on top most frequent.

good boy girl O/P

S₁ 1 1 0 1

S₂ 1 0 1 1

S₃ 1 1 1 1

Binary BOW and BOW

L1 and OY L count will be updated
based on frequency
2, 3, 4, ...

advantages and disadvantages of BOG.

adv:

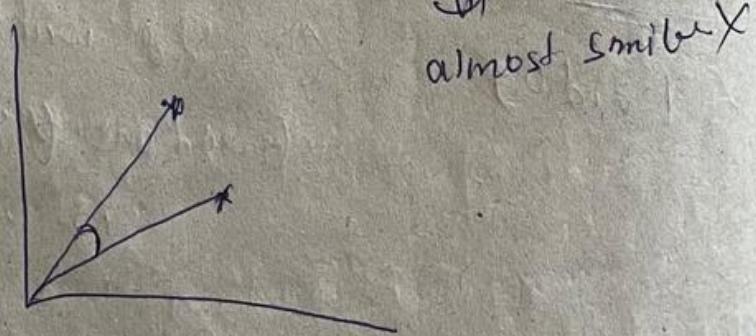
- ① Simple and intuitive
- ② Fixed size
BFS NL algo

Disadv:

- ① Space required \rightarrow O(N^2)
- ② Ordering of the word is getting change
- ③ Out of vocabulary
- ④ Semantic meaning is still not captured

The food is good $\rightarrow [11101] \rightarrow v_1$

The food is not good $[11111] \rightarrow v_2$



Cosine similarity

Smaller angle b/w them \Rightarrow very low

we can say they both are similar

BOW Implementing Using NLTK

- Take the dataset (CSV)
- Convert into df.
- remove unnecessary columns (RE)
- remove stop words
- make a cleaned sentence
- apply stemming and get list of ~~sentences~~ words.
- Convert them to sentences and add each sentence to the corpus []

apply bag of words using sklearn

from sklearn.feature_extraction import
CountVectorizer

We can play with parameters

$x = \text{CV}.fit(\text{df})\cdot\text{transform}()$

$x = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \rightarrow \text{numerical values}$

N-Grams (used in BOW,
TF-IDF)

e.g. Bigram, trigram

The food is good

The food is not good

food not good

→ |

0

1

→ |

|

|

[food not good food good food not not good]

s1

| 0 1 | | 0 0

s2

| | | 0 | |

↓

Should be
continuous.

Q Now we applied bi-gram (2 word)

Similarly we can apply tri-gram

Dr. Stein

↪ n-gram $\rightarrow (1,1) \rightarrow$ Unigram

$(1,n) \rightarrow$ unigram, bigram

$(1,3) \rightarrow$ Uni, bi, trigram

$(2,3) \rightarrow$ bigram, trigram

}

Combination of
Unigram and
bigram

N-grams \rightarrow helps to capture the context or

meaning in phrases by looking at the
words in groups rather than individually

1-gram \rightarrow Brahmin

2-gram \rightarrow Best Jokes

3-gram \rightarrow Best Joke

N-Gram BOW Implem by MLT6

Final TF-IDF TF × IDF



good boy girl

Sent 1

0	$\frac{1}{2} \log_e(2)$	0
0	0	$\frac{1}{2} \log_e(3)$
0	$\frac{1}{3} \log_e(3)$	$\frac{1}{3} \log_e(3)$

Sent 2

Sent 3

TF-IDF

[Term Frequency - Inverse Document Frequency]

$s_1 \rightarrow$ good boy

$s_2 \rightarrow$ good girl

$s_3 \rightarrow$ boy girl good

$$TF = \frac{\text{No of repetition of words in sentence}}{\text{No of words in sentence}}$$

$$IDF = \log_e \left(\frac{\text{No of sentences}}{\text{No of sentences containing the word}} \right)$$

	<u>Term frequency</u>			<u>IDF</u>
	s_1	s_2	s_3	
good	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	good $\log_e(\frac{3}{3}) = 0$
boy	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	boy $\log_e(\frac{3}{2})$
girl	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	girl $\log_e(\frac{3}{2})$

Advantages & Disadvantages of TF-IDF

- ① intrusive
- ② fixed sized inputs
- ③ word importance is getting captured.

- ① Sparsity still exists
- ② OOV
(out of vocabulary)

if a word is present in all the sentences we should give the less priority.

Because that does not add any value to that

TF-IDF performs better than BOW

RTTF IDF implementation using python

→ Python library → also based on
→ C++ API → will implement
→ Python objects with corresponding
→ C++ code and will start writing
→ Python interface like API

Word Embeddings

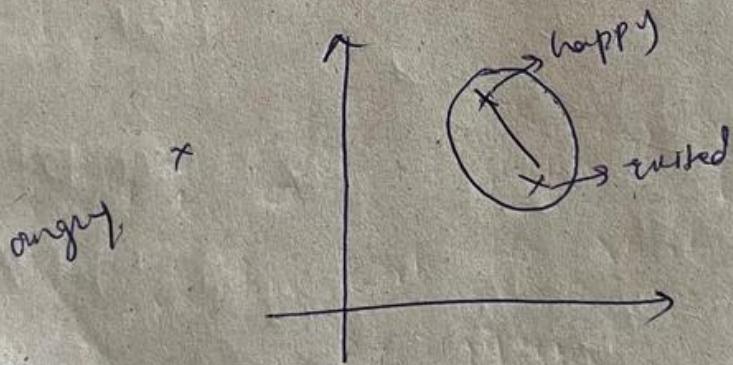
are the way to represent words as numbers
(vector)

so that words with similar meaning have similar representations. This helps Computer understand relationships b/w words, making it easier to perform tasks like translation, sentiment analysis and search.

$$\text{angry} = [2.5 \quad 3.5] \quad \text{using PCA}$$

$$\text{happy} = [7.2 \quad 8.2] \quad \text{almost similar}$$

$$\text{excited} = [7.3 \quad 8.0]$$



word embedding

Count on frequency

- OHE
- BOW
- TF-IDF

based on
Deep learning based
models

Word2Vec
(Very good and)

ANN

Word2Vec

CBOW

Skipgram

[continuous BAG of words]

Word2Vec

→ google

is a technique for NLP published in 2013.
The Word2Vec algo uses a neural network
model to learn word associations from a large
corpus of text. Once trained, such a model
can detect synonymous words or suggest
additional words for a partial sentence.

as the name implies, word2vec, word2vec
represents each distinct word with a particular
list of numbers called vector.

Value: feature representation

①

② Vocabulary → unique words → Corpus

feature
representat
→ can be in both
sw

	Boy	girl	king	Queen	apple	Mango
gender	-1	1	-0.92	+0.93	0.01	0.03
Royal	0.01	0.02	0.95	0.96	-0.02	0.02
age	0.03	0.02	0.75	0.68	0.95	0.96
food	-	-	-	-	0.91	0.92
.	-	-	-	-	-	-
n th dim	-	-	-	-	-	-
300 dimension	-	-	-	-	-	-

Based on the relationship with feature representation wrt the vocabulary a numeric value is given.

$$[\text{King} - \text{Boy} + \text{Queen} = \text{Girl}]$$

Google Word2Vec is trained with 3 billion words

king [0.95, 0.95]

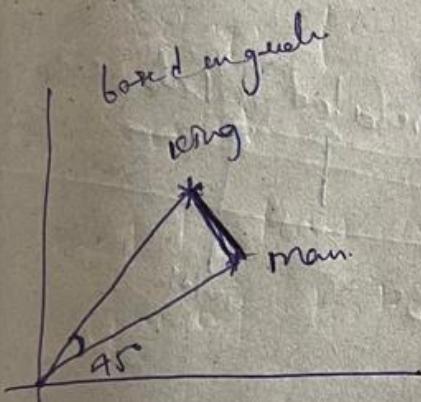
man [0.95 0.98]

Queen [-0.95, 0.95] woman [-0.94, -0.95]

Rong - MAN + Queen = Woman

Cosine Similarity:

Distance = 1 - Cosine Similarity.

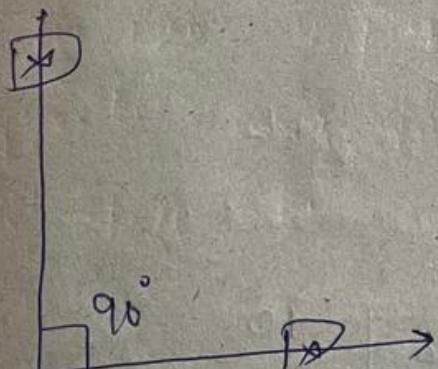


$$CS = \cos 45^\circ = \frac{1}{\sqrt{2}} = 0.7071$$

$$\text{distance} = 1 - 0.70$$

$$= 0.30$$

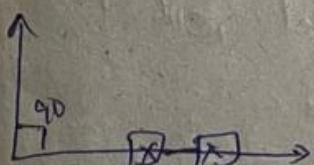
almost same words.



$$\text{DWS} = 1 - \cos 90^\circ$$

$$= 1 - 0$$

$$= 1 \quad \boxed{\text{different words}}$$



$$\text{DWS} = 1 - \cos 0^\circ$$

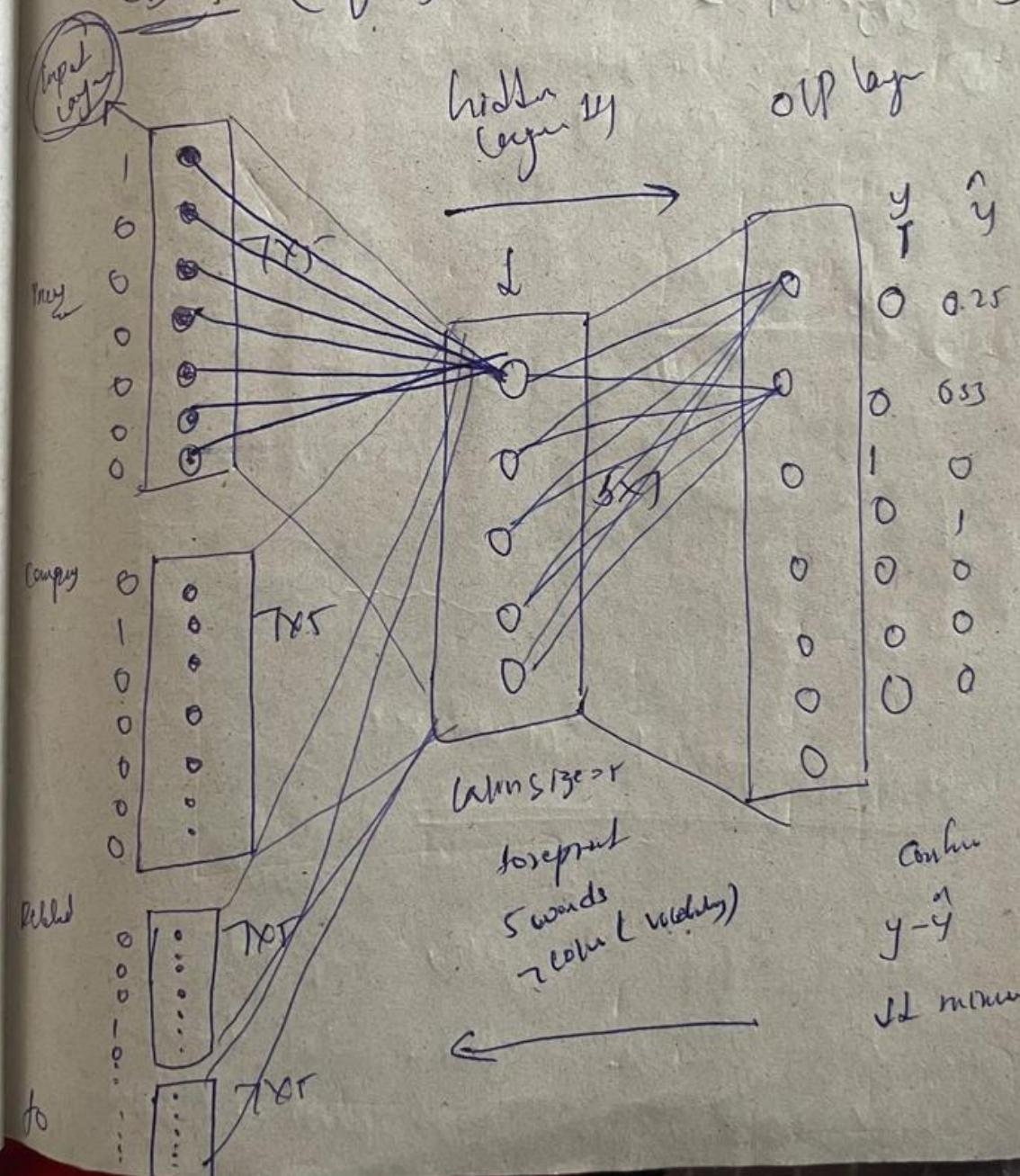
$$= 1 - 1$$

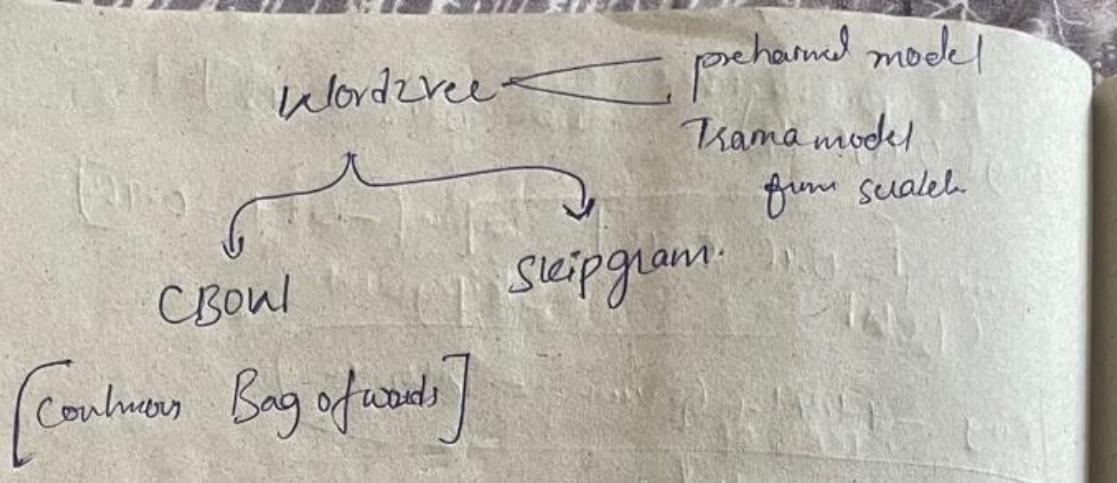
Fewly similar

with OHE

new	1	0	0	0	0	0	0
company	0	1	0	0	0	0	0
related	0	0	0	1	0	0	0
to	0	0	0	0	1	0	0

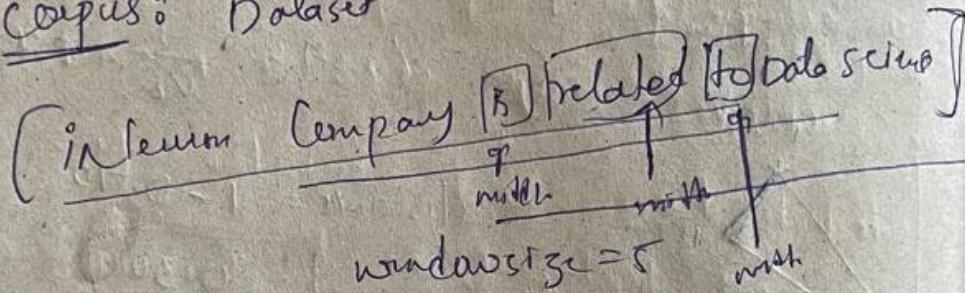
Cbow L fuzzy Connected Neural Network





① CBOW [Continuous Bag of Words]

Corpus: Dataset



zip

OPP

is

→ [intern, Company, Related, to]

move window by 1 pos

→ [Company, is, Related, Data]

Related

(pos →)

→ [is, Related, Data, science.] TO

I'm going to train my model with this

Whole word \rightarrow Vector

OIP [size of 5 vectors]

Due to window size (+)

window size = 5

because I want to probably provide a
feature representation with a vector
size of '5'

SkipGram (word2vec)

another architecture of word2vec

OIP

SIP

IS

→ [Innum, Company, Related, TO]

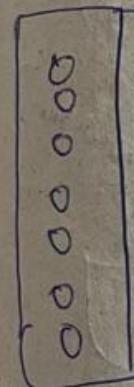
→ [Company, IS, TO, Date]

Related

→ [IS, Related, Date, Source]

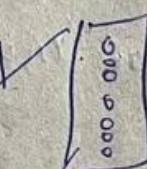
TO

Input layer



window
size

5x7



OP
layer

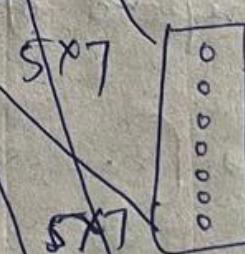
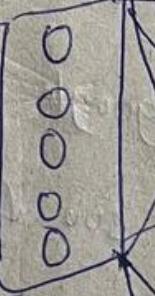
software
for
y,
...

y,
...

Y
-
-

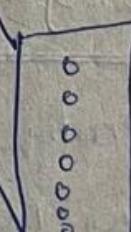
using ↓↓

for
backpr
↑



5x7

5x7



When should we apply CBOW or skipgram

Small dataset \rightarrow CBOW

Huge dataset \rightarrow Skipgram

to increase accuracy (CBOW or skipgram)

- ① Increasing the training data
- ② Increase the window size - vector dimension is also increasing.

google word2vec

3 billion words \rightarrow google news (Vocabulary)

further report of 300 dim vec (vector)

Curve \rightarrow [. 300dim]

advantages of word2vec

- \rightarrow Sparse matrix \rightarrow Dense matrix
- \rightarrow Semantic info is getting captured (good, better)
- \rightarrow Vocabulary size \rightarrow fixed set of dimension vector
- \rightarrow OOV is also stored google word2vec (300 dim)

*** Arrange wordtree

Tent

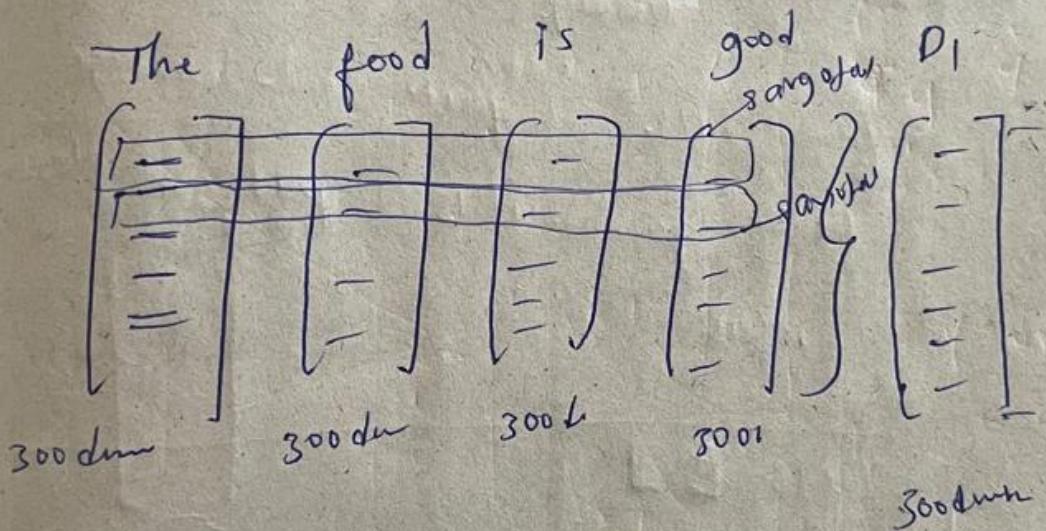
OLP

D1 The food is good .1

D2 The food is bad 0

D3 pizza is amazing 1

google purchased wordtree model

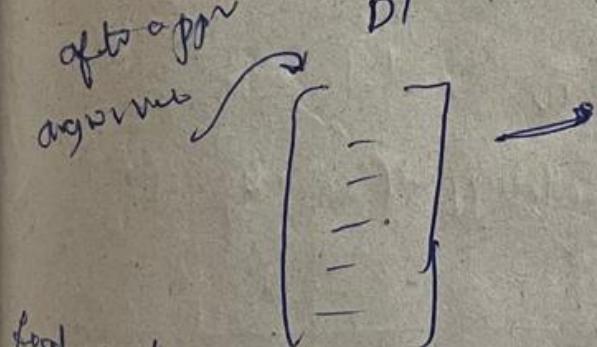


after we get vecn of 300¹

after appr
answers

D1

OLP



OLP

food regard

food bad

pizza amazing

OLP

1 0 1

With the help of Gutenberg Library

Gensim,

Glove.

↳ purchased Google word2vec

↳ have a wrote from scratch

Word2vec practical implementation

Amazon Kindle Review Sentimental analysis

using Nap

using apply()

df =

Name	age	Salary
—	—	50000
—	—	51000
—	—	50000

using apply()

$df['salary'] = df['salary'].apply(\lambda b: n * b)$

↓

Name	age	salary
—	—	55000
—	—	59400
—	—	55000

↑
change

Best practices for solving ML problems

① Preprocessing and Cleaning



② BOW and TF-IDF



③ Train Test Split may leads
to data leakage



④ Trained our models



Best practices

① preprocessing and Cleaning



② Train test split (Corpus, Y)



③ BOW and TF-IDF



④ Training the model



no data leakage

Span ham using word2vec and Argmax

Creating word2vec from scratch.

Without using google learned.

+ another way of cleaning the data in NLP

using simple-preprocess from gensim
library.

from nltk import sent_tokenize

from gensim.utils import simple_prepr

-cess

words = []

for sentence in corpus:

sent_tokens = sent_tokenize(sentence)

for token in sent_tokens:

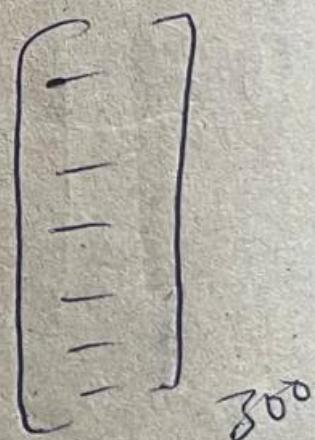
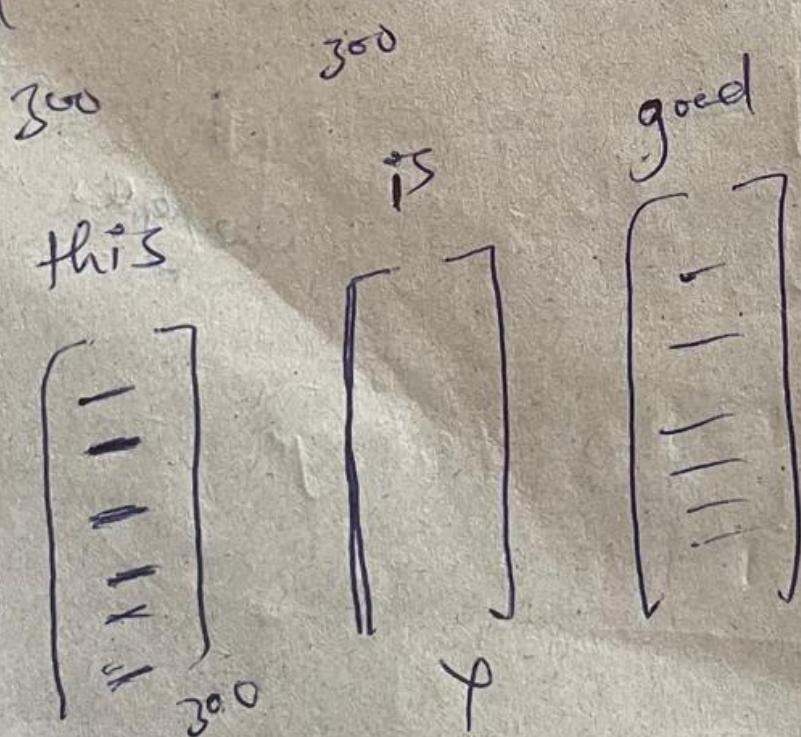
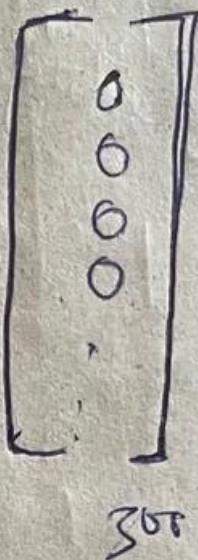
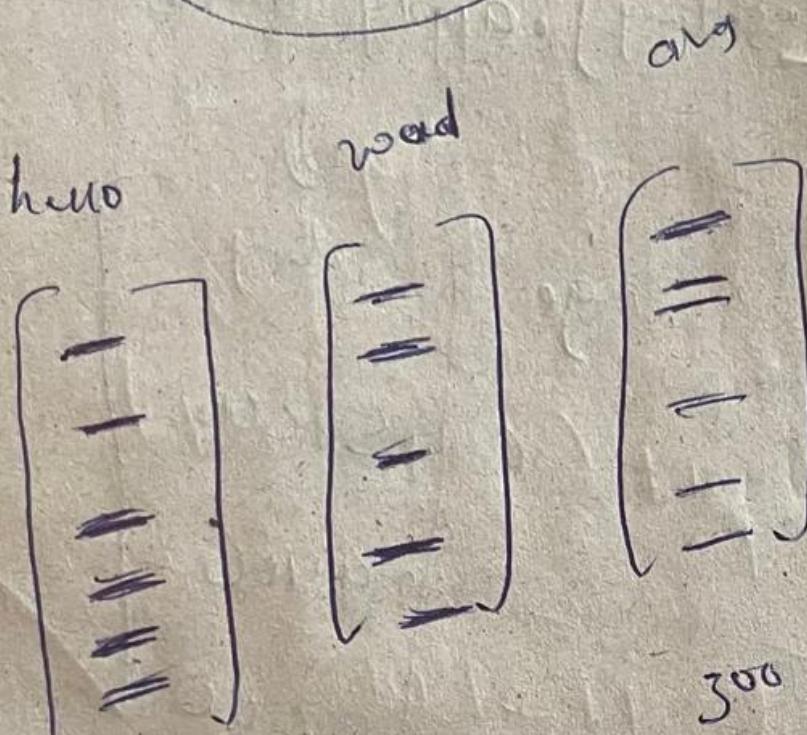
tokens.append(simple_preprocess(token))

~~text = "hello word"~~

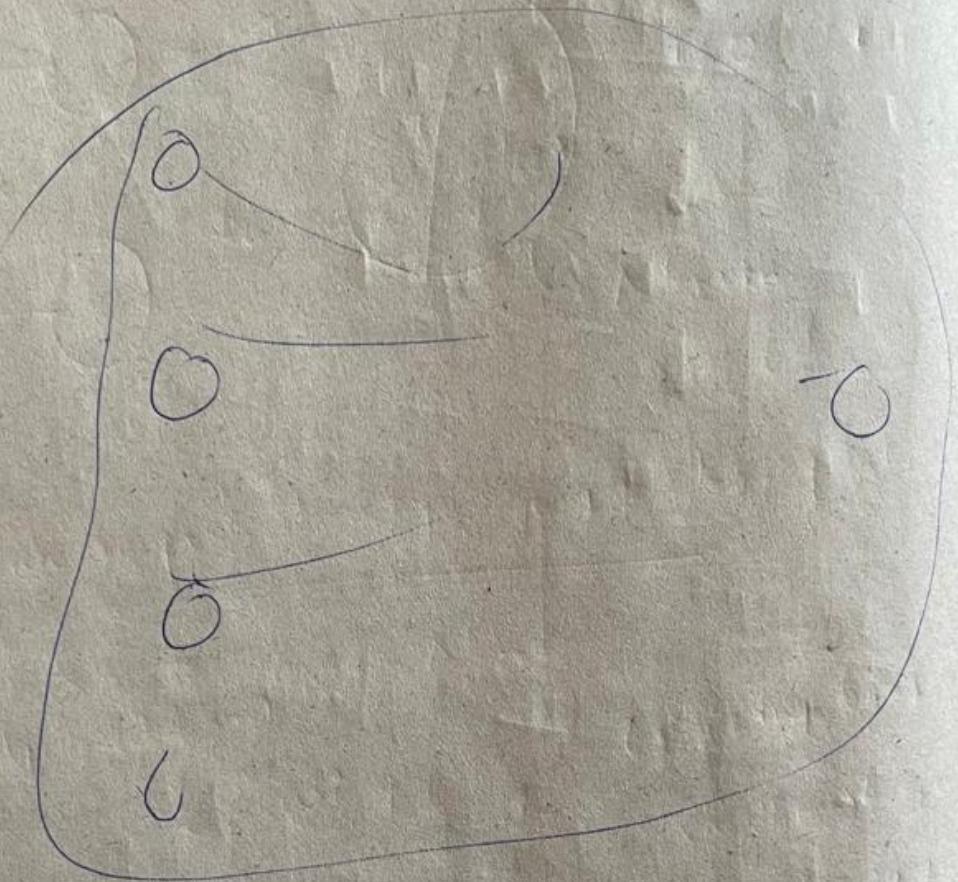
~~This is good~~

~~2xyb si zzt+i~~

2xyb sr 224



Deep Learning



Perceptron

[artificial Neuron, Neural Network Unit]



Single Layered Neural Network

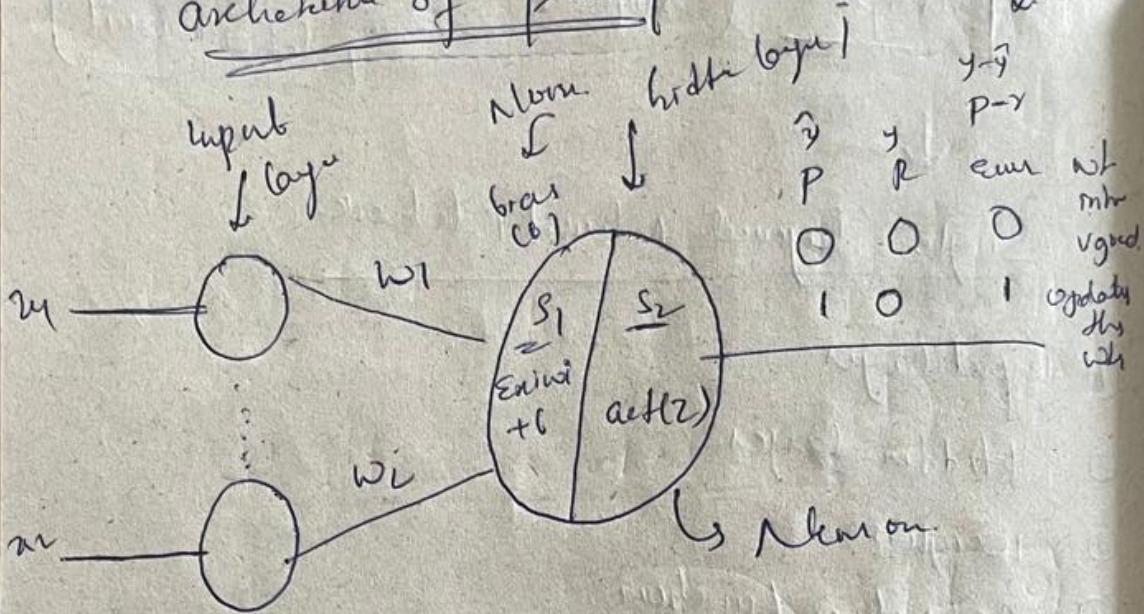
- ① Input Layer
- ② Hidden Layer.
- ③ weights
- ④ Activation function.

SINN is used for binary classification

Dataset

IQ	No. of study hrs	OIP pass (%)
~	3	0
95	1	1
110	4	1
100	5	
x_1	x_2	

Architecture of Perceptrons



Two important steps going to take place.

~~in~~ ~~addition~~ make Neuron
(Weighted sum)

Step 1: $Z = \sum w_i x_i + b$

$$Z = w_1 x_1 + w_2 x_2 + b$$

$$Z = \sum_{i=1}^n w_i x_i + b$$

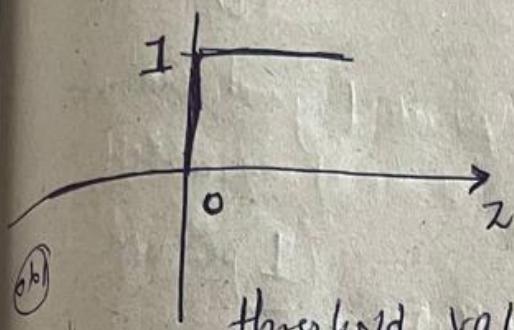
by adding bias
even though given
are randomly given
it is going to add
something and
not make dead
neuron

Step 2: activation function

transform the output

between 0 to 1
-1 to 1

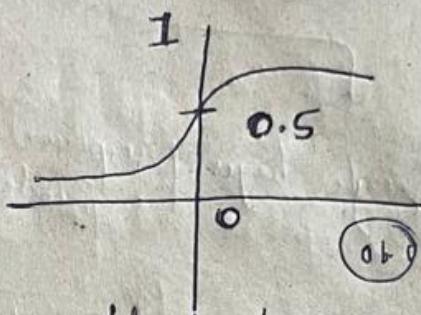
Step function



threshold val = 0

$$\begin{cases} z \leq 0 & 0 \\ z > 0 & 1 \end{cases}$$

Sigmoid function



threshold = 0.5

$$\begin{cases} z > 0.5 & 1 \\ z < 0.5 & 0 \end{cases}$$

step 1:

$$z = \sum_{i=1}^n w_i n_i + b$$

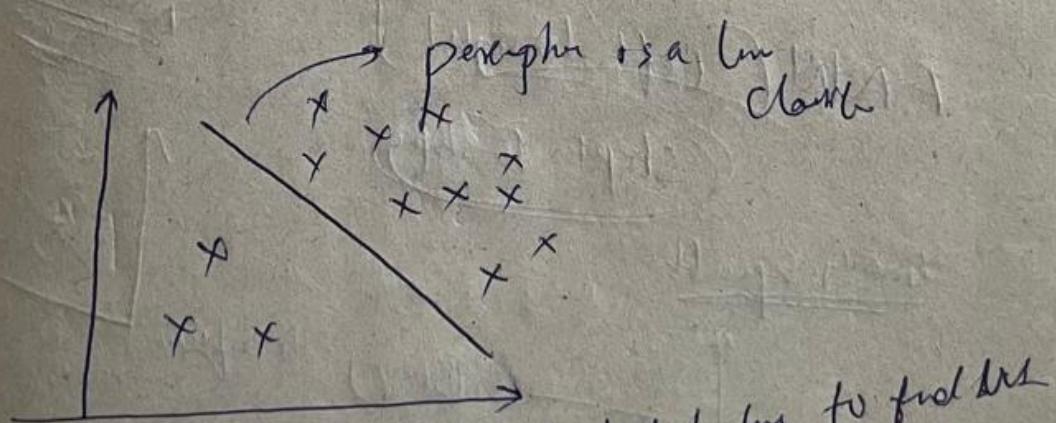
$$z = b + w_1 n_1 + w_2 n_2 + w_3 n_3 + \dots + w_n n_n$$

||

$$y = m n + c$$

$$y = \beta_0 + \beta_1 n_1 + \beta_2 n_2 + \beta_3 n_3 + \dots + \beta_n n_n$$

Linear
problem
statement



the class to find the best fit line to find the

Purposive model

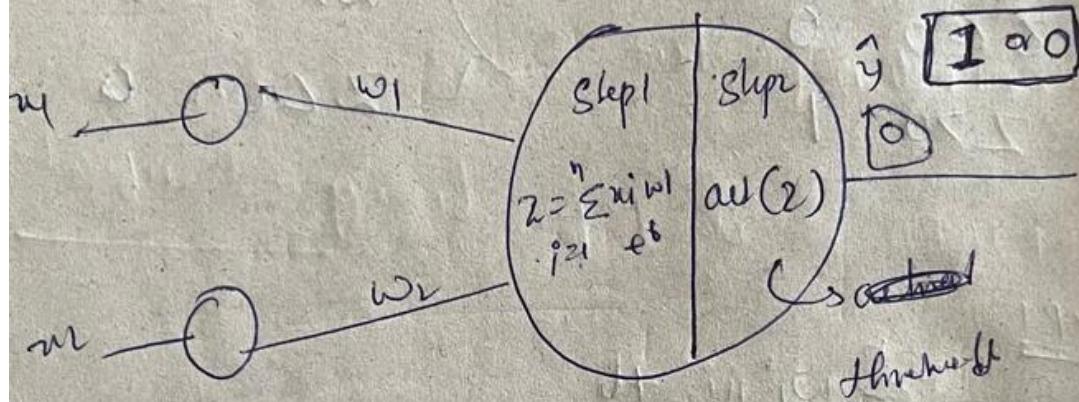
[ANN]

Single layer purposive model

Multi layered perceptron model

Feed Forward Neural Netw

1) feed off



$$\hat{y} = 0, y = 1$$

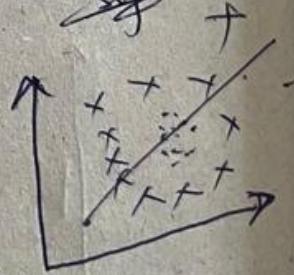
here it give us error so we go back

change weight and do again

BS.

FNN

Non linear
sigm



FNN (left to right)
Step1 to step2

Linearly Separable

Binary classifier

If we get an error it's not efficient
to train FNN every time by updating
the weights.

In multilayer neural networks we have
following like :

- loss functions
- Backward propagation \rightarrow gradient methods.
- Optimizers

Steps to train in MNN

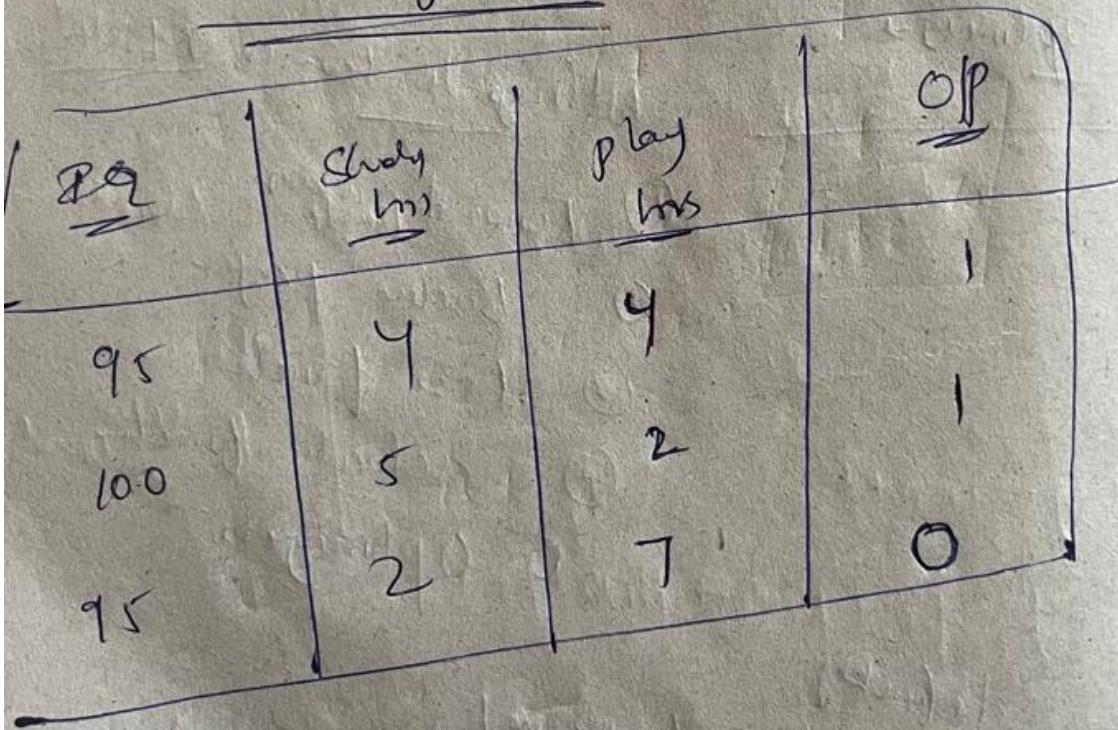
- ① Forward pass
- ② Backward pass
- ③ Loss function
- ④ Gradient function
- ⑤ Optimizers.

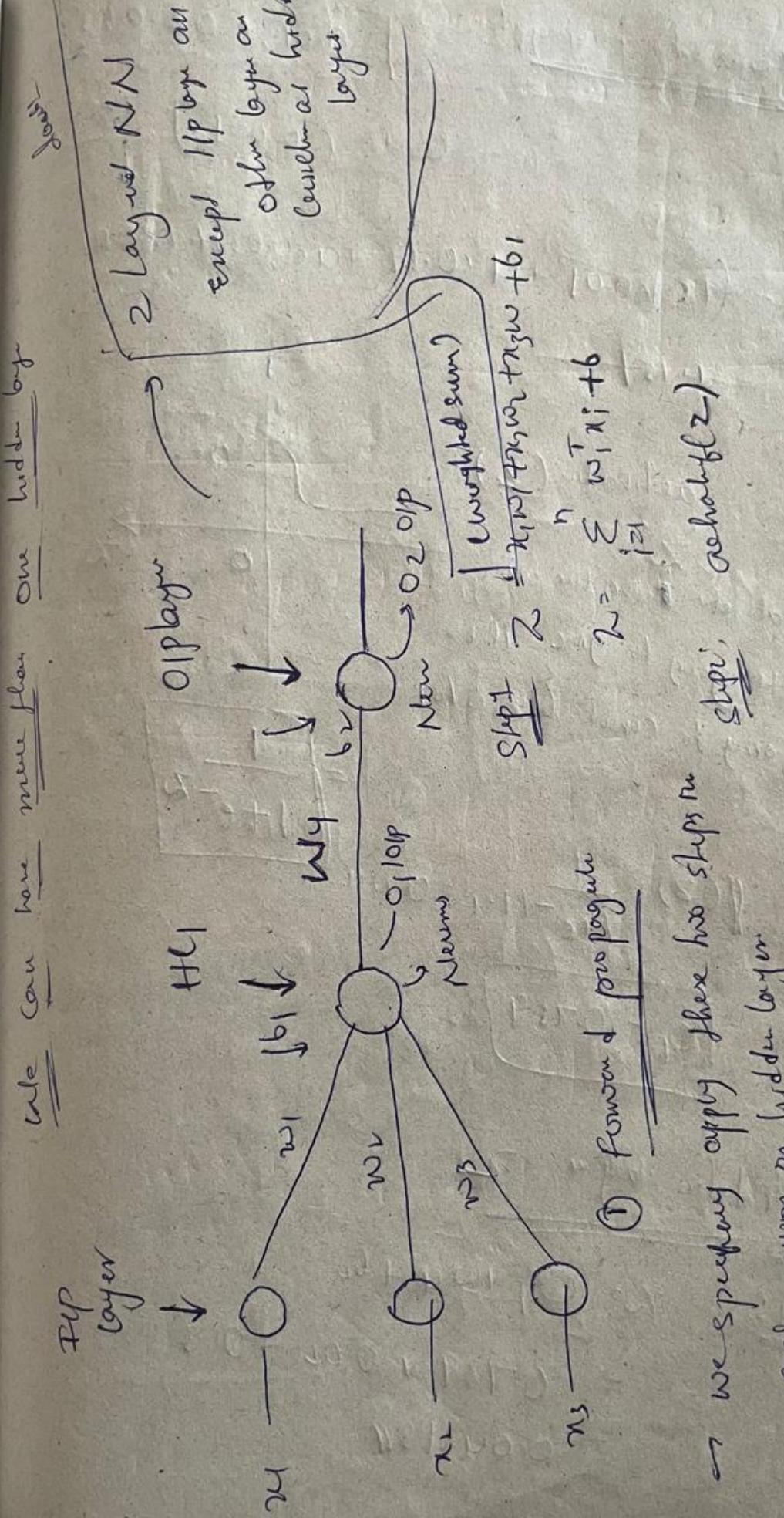
definitely
multilayer neural
networks should be used

Multi-layered Perceptron Model [artificial neural net]

- ① Forward propagation
- ② Backward propagation - [invented by Geoffrey Hinton]
- ③ Loss function
- ④ Optimizer
- ⑤ Activation function

Multi Layered Net





① forward propagation

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ 0.01 & 0.02 & 0.03 \end{bmatrix}$$

$$bias = [0.001] \quad \downarrow \text{bias}$$

Step 1:

$$z = (9 \times 0.01 + 4 \times 0.02 + 0 \times 0.03) + 1 \times 0.01 \\ = 1.151$$

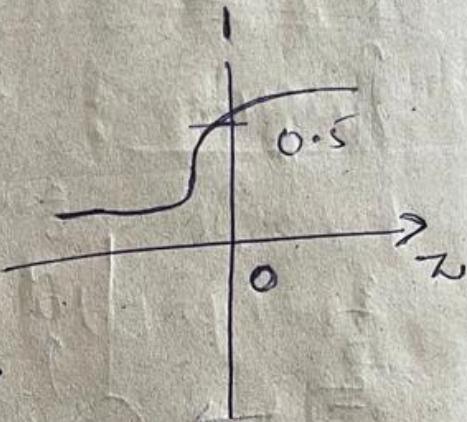
Step 2:

activation (z)

\tilde{z}

What ever value we

are going to give is, is
going to come below 0 to 1



$$= \boxed{\frac{1}{1+e^{-z}}}$$

$$f(z) = \frac{1}{1+e^{-1.151}} = 0.759$$

$$\boxed{O_1 = 0.759}$$

Hidden layer 2

Step 1:

$$z = O_1 * w_4 + b_2$$

$$= 0.759 * 0.02 + 0.3 \\ = 0.04518$$

$$\text{step 2} \quad \text{actual (2)} \quad \frac{1}{1+e^{-(0.0045 \cdot 8)}} = 0.51129$$

$$T_{O_2} = 0.51129$$

whatever or right that is the predicted OIP

$$\text{here } O_2 = 0.51129$$

Loss function

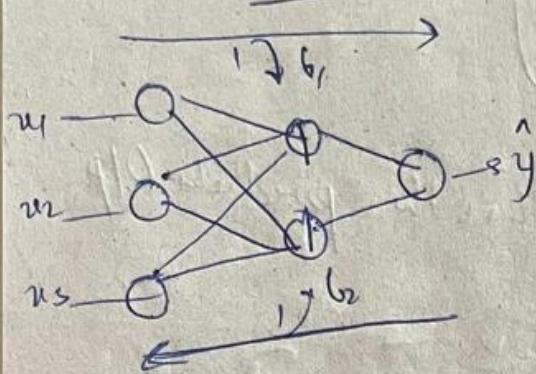
Real OIP - Predicted OIP

$$1 - 0.51129$$

\Downarrow
Basically takes about error ≈ 0.49

my main aim is to train the model by
~~decreasing~~ minimizing the error $0.49 \downarrow$
 by updating the weights w_0, w_1, w_2, w_3
 it is going to done by Backward propagation

Loss function vs Cost function



$$\text{Costfn} = \text{Loss}(y - \hat{y})$$

using optimization

Gradient Descent

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

Loss fn

Cost fn

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

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

Loss fn takes one data point once and updates weights

→ Cost fn takes all the data points at a time and update weight only once.

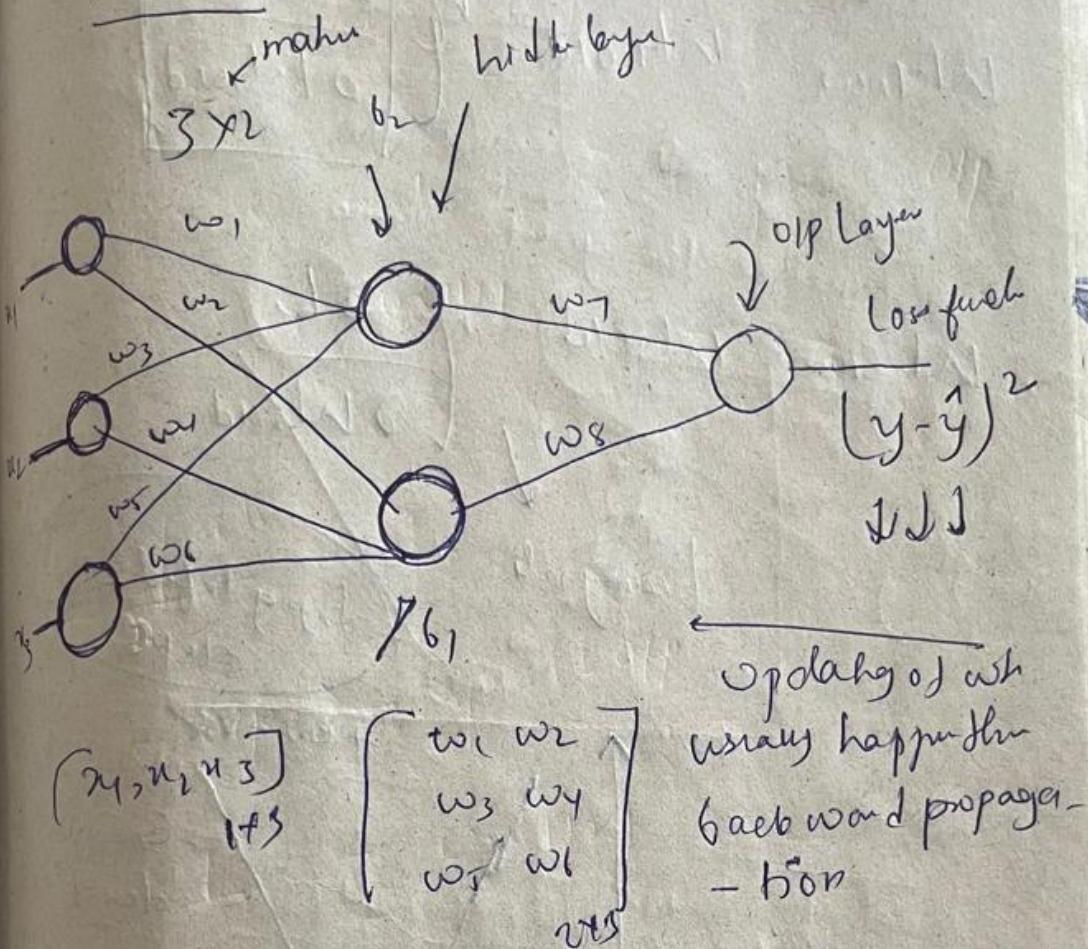
PIP feature

	u1	u2	u3	0/P
Study ms	95	4	4	1
play house	100	5	2	1
	95	2	1	0

We have loss function and cost function with
to go with regression and classification

Backward propagation and weight update

Formula:



Different types of loss function

Regression

- ① MSE
- ② MAE
- ③ Huber loss
- ④ RMSE

Classification

- ① Binary cross entropy
- ② Categorical cross entropy

weight update formula.

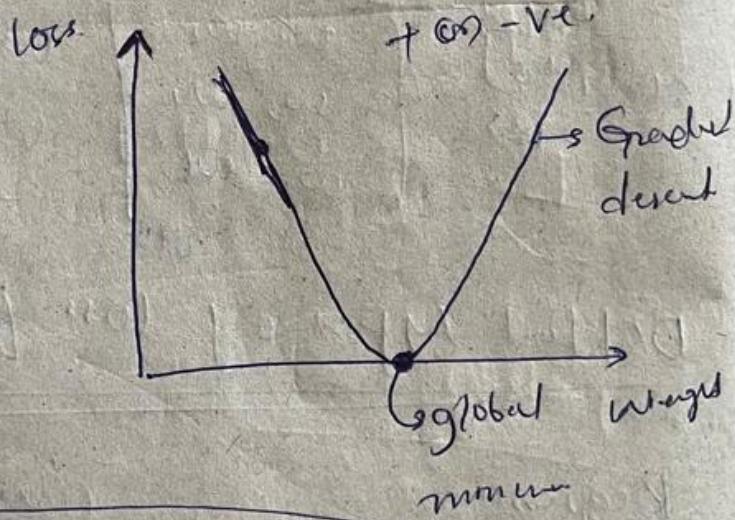
$$w_{\text{new}} = w_{\text{old}} - \eta$$

$$\boxed{\frac{\partial L}{\partial w_{\text{old}}}}$$

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial h}{\partial w_{\text{old}}}$$

Learnrate

(trying to find the
super)



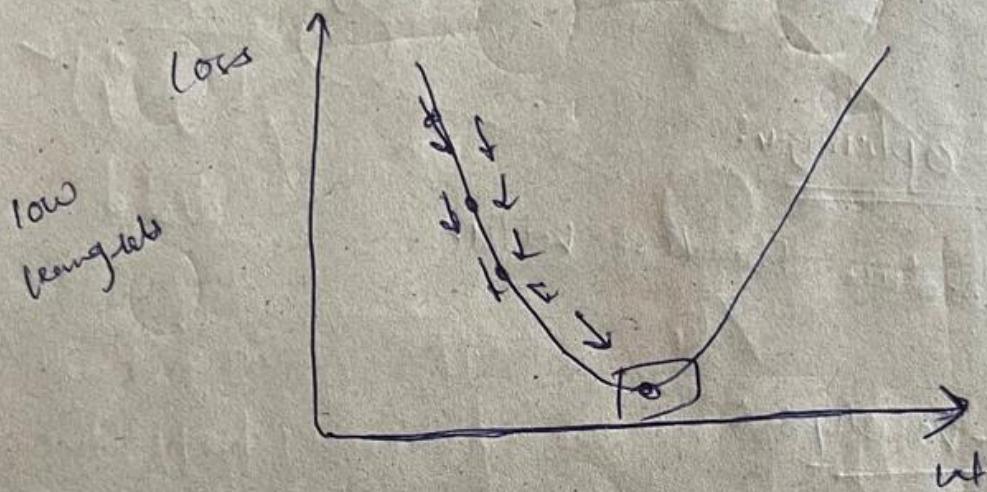
$$\boxed{w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial h}{\partial w_{\text{old}}}}$$

weight update formula

$$\eta \rightarrow \text{Learning rate}$$

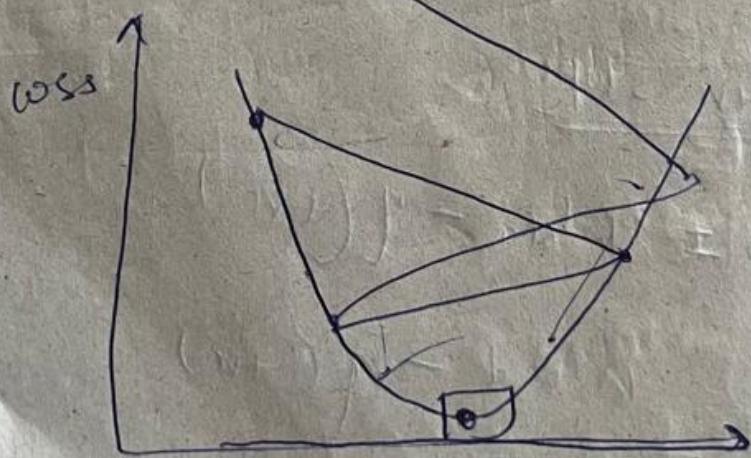
$\eta = 0.001$

learning rate decides the convergence of
towards the global minima

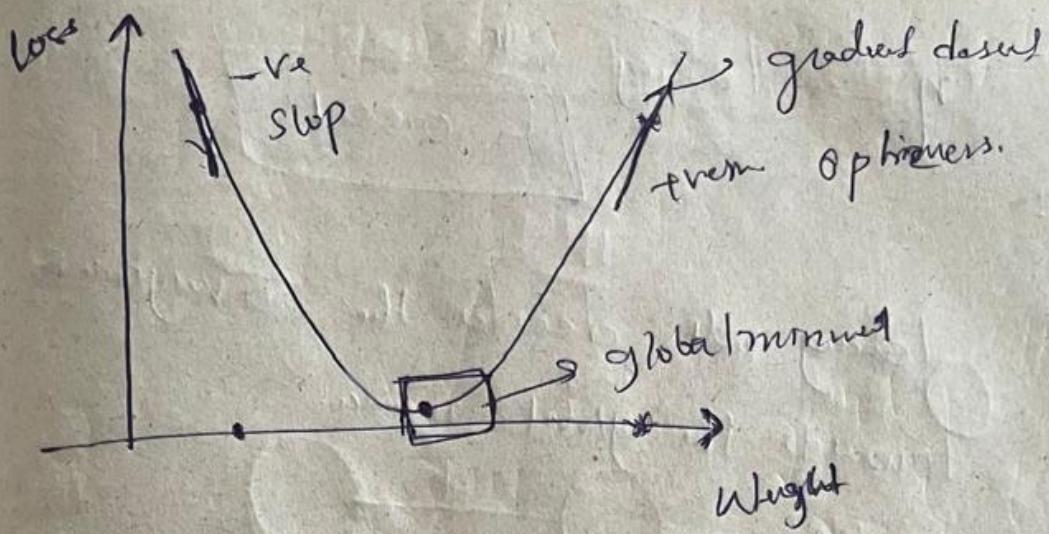


Inlet of large learning rate:

$$\eta = 20.$$



Sometime it may never converge



role of Optimizer?

to reduce the loss value.

for -ve slope :

$$w_{\text{new}} = w_{\text{old}} - \eta (-\text{ve})$$

$$= w_{\text{old}} + \eta (\text{+ve})$$

$$\boxed{w_{\text{new}} > w_{\text{old}}}$$

for +ve slope :

$$w_{\text{new}} = w_{\text{old}} - \eta (\text{+ve})$$

$$w_{\text{old}} - \eta (-\text{ve})$$

$$\boxed{w_{\text{new}} < w_{\text{old}}}$$

~~A~~
When

W reaches global minima

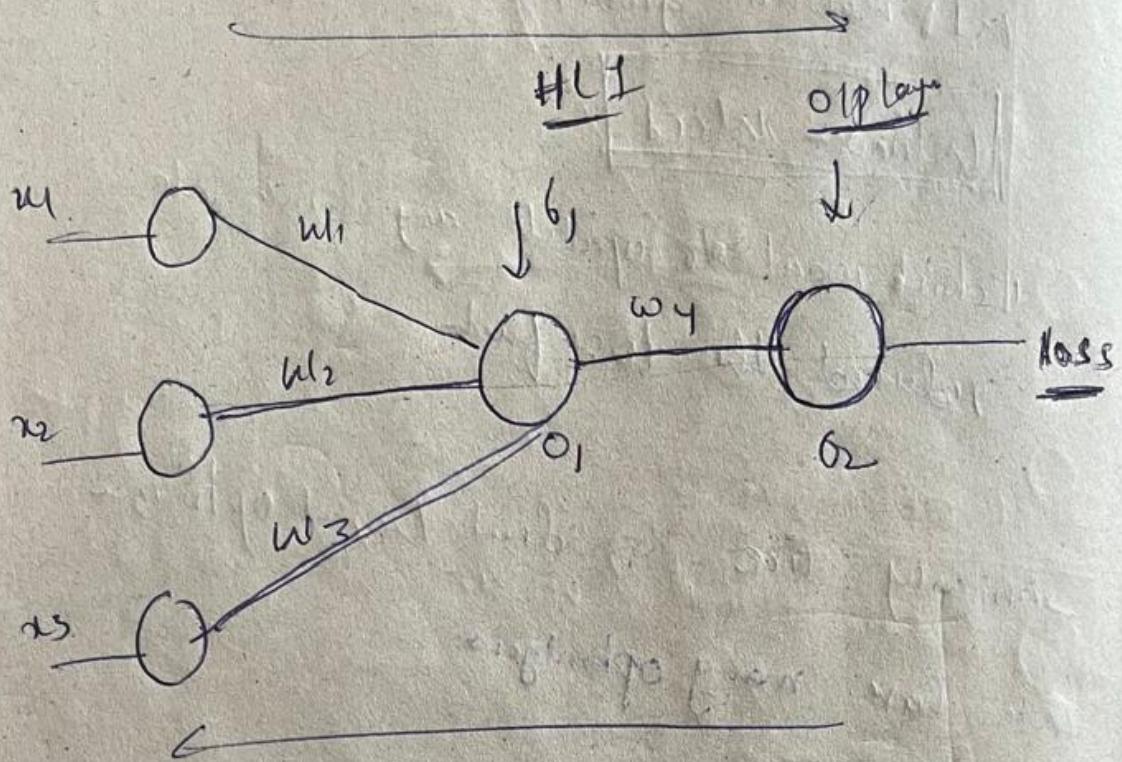
$$\boxed{W_{new} = W_{old}}$$

I don't want to update any single like
reduced the loss function

Similarly like Gradient Descent optimization
we have many optimizers

- Momentum
- adaptive learning rate Optimizer
- less Common Optimizer
- Second Order Optimization methods

Chain Rule of Derivatives



update of $w_{4\text{new}}$

$$w_{4\text{new}} = w_{4\text{old}} - \eta \left[\frac{\partial L}{\partial w_{4\text{old}}} \right]$$

$$\boxed{\frac{\partial L}{\partial w_{4\text{old}}} = \frac{\partial L}{\partial o_2} * \frac{\partial o_2}{\partial w_{4\text{old}}}}$$

↳ chain rule of
derivation

ML END-TO-END PROJECT

AGENDA

Why
↑
Store code Online
Collaborate easily
Showcase work

① Set up the github {repository}

② New environment

③ Set up py

c) requirements.txt

project
folder
open
anaconda
prompt
Cd & D: __

Why new envirn (venv).

① Isolation

② prevent conflicts

③ easier to manage

Conda create -p venv python=3.8 -Y

Conda activate venv /

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin [copy from git hub](https://github.com/username/repo_name)

git push -u origin main

Creating gitignore file (in English)

- ① avoids unnecessary file (sysfile, tempfile, logfile)
- ② prevent sensitive data (API keys, password)
git pull
- ③ Reduces repository size
main to copy
- ④ Setup.py, requirements.txt
helps keep your
git repository clean
secure.

(pypi)

Setup.py → simply building our project
as a application

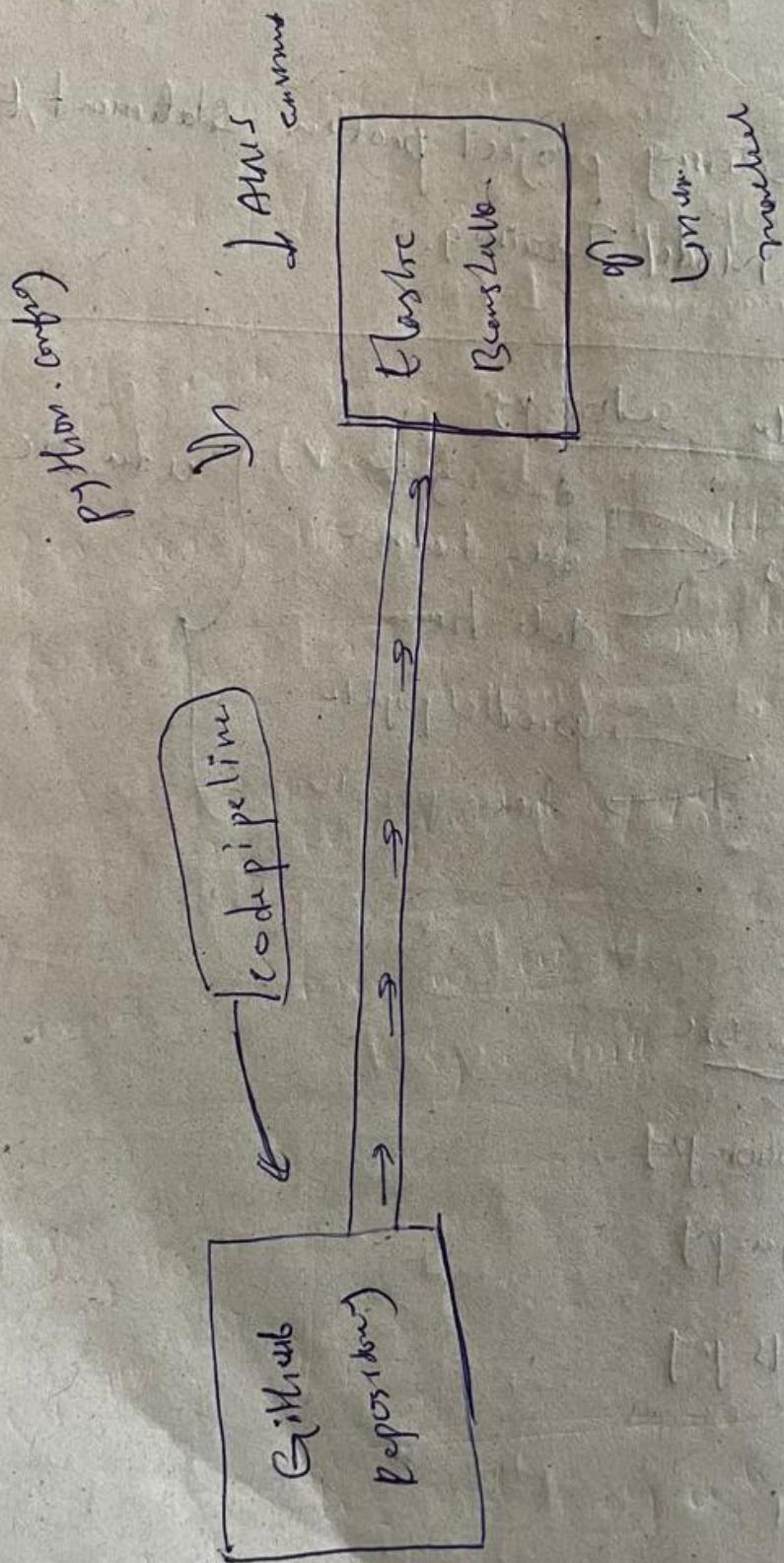
- ① packaging the project
- ② installing Dependencies
- ③ organizing the project
- ④ providing project info
- in simple words setup.py is like a set of
instructions that tells other how to install
and use your project, along with crypto
it needs to work properly in different

place

→ give how we install library for
python pypis

after model building completed

ML project Deployment using AWS Lambda



→ with mlflow.start_run():

- mlflow.log_param → helps you to remember which setting you had for each argument
- mlflow.log_metric

↳ give you feedback on how well your model performed with the parameters.

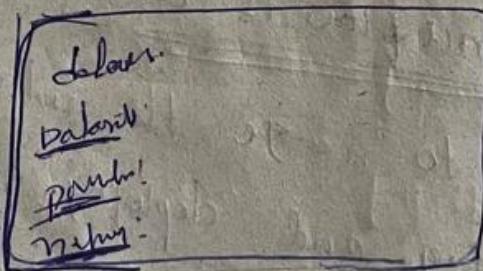
→ mlflow.get_tracking_uri()

after you run the code python app.py

a folder mlruns folder is going to be created

↓
Content, evaluating which project
in end → mlflow ui
↳ give link

you won't
get the
UI like this



Changing parameters and running code again
python app.py 0.3 0.7

MLflow: focus on experiment tracking and model lifecycle management
↳ (Local)

Dagshub: It is a collaboration platform that integrates MLflow with data and code versioning tools, ideal for team-based ML projects.

↳ (Remote)

Sign in dagshub with git hub account

MLops: managing the whole ML lifecycle for production.

MLflow: Tracking and deploying ML experiments.

Dagshub: collaborating and version-controlling ML projects (supports MLflow, DVC)

DVC: Version control for data and models in ml.

Q.

→ requirements.txt — dvc

data folder

↳ data.txt

dvc init

↳ .dvc folder created.

delete requirements.txt

git commit -m "DVC init".

to have data.txt

dvc add data / data.txt

Now data.txt is only tracked with dvc

not by git

↳ Create gitignore file inside that data folder
containing data.txt

↳ and also create .data.txt.dvc file
which contains hash key of data file.

git add data / data.txt.dvc

git add data / signature

→ Similarly you can make multiple changes
in data.txt

Step 1 → dvc add data / data.txt

→ And track data.txt.txt through git

git add ~~data.txt~~ data.txt

Step 2

↓
new updated hash key is going to be
tracked.

Similarly you can do multiple changes
and do above 2 steps.

Suppose we have done three (3)

commits

git log

commit 1

commit 1

commit

Copy

commit 2

commit

commit 3

Now you want see hash w.r.t to
Second const

↳ git checkout 'paste'

To see the date w.r.t that
hash key

dvc checkout

=====

↓

=====

you will see the date w.r.t the
particular hash key.

| To come out of that

for git git checkout master

for dvc: ~~git~~ dvc checkout

Building Datasource project with

BentoML

↓ platform for software

BentoML is a ~~Self~~ ~~Engine~~ Engineers to build AI products.

→ Bento me is a tool that makes it easy to turn ml models into deployable applications.

- ① Model Packaging
- ② API creation. → helps to create REST API
- ③ Deployment Option $\xrightarrow{\text{Cloud Service}}$ VM, Local machine
- ④ Model Registry and healing

→ Bento app with any AI Model

cryptorch, tensorflow, scikit-learn

→ Inference Optimization for AI Apps.

using flake for code api may not be that optimized mem(COL)

↳ Build once Deploy anywhere

How does BentoML work

- ↳ Define model
- ↳ Save a model
- ↳ Create a Service
- ↳ Build a Bento
- ↳ Deploy a Bento

few commands

bentoml models list

to run Service.py

bentoml serve ~~service~~ service.py:svc --reload

Simularity fn (W)

$$w_{1\text{ new}} = w_{1\text{ old}} - \eta \frac{\partial L}{\partial w_{1\text{ old}}}$$

$$\frac{\partial L}{\partial w_{1\text{ old}}} = \frac{\partial L}{\partial o_2} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{1\text{ old}}}$$

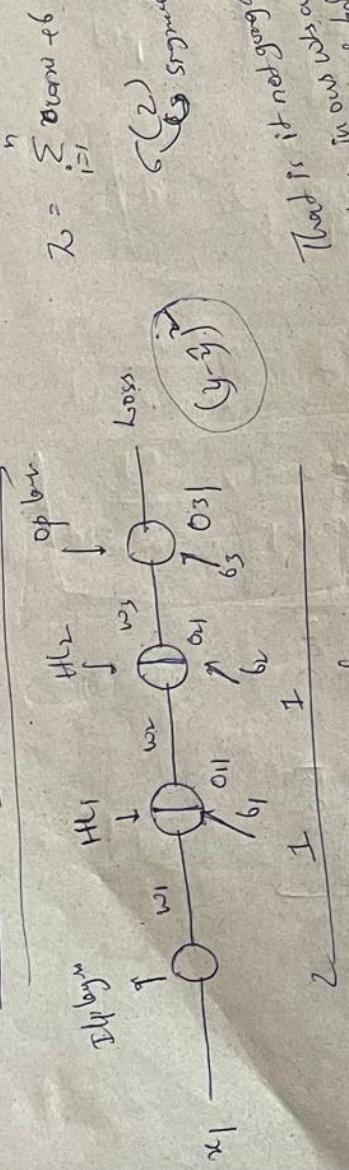
$$w_{2\text{ new}} = w_{2\text{ old}} - \eta \frac{\partial L}{\partial w_{2\text{ old}}}$$

$$\frac{\partial L}{\partial w_{2\text{ old}}} = \frac{\partial L}{\partial o_2} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{2\text{ old}}}$$

diff activation function

- ① Sigmoid $\rightarrow \frac{1}{1+e^{-x}} \rightarrow [0, 1]$
- ② tanh $\rightarrow \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (circled 1) $\rightarrow [-1, 1]$
- ③ ReLU $\rightarrow [0, \infty)$
- ④ Leaky ReLU $\rightarrow [-\infty, \infty)$
- ⑤ ELU
- ⑥ Softmax $\rightarrow [0, 1]$
- ⑦ Swish.

Vanishing Gradient problem and activation functions



① Sigmoid activation function

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

derivative of activation function is very small near zero value whenever we apply sigmoid

$$w_{1, \text{new}} = w_{1, \text{old}} - \eta \left(\frac{\partial L}{\partial w_{1, \text{old}}} \right)$$

Derivative value $\frac{\partial L}{\partial w_{1, \text{old}}} = 0.01 * 0.025 * 0.25$

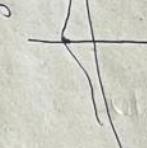
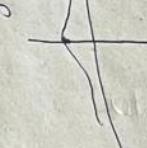
$$\frac{\partial L}{\partial w_{1, \text{old}}} = \frac{\partial L}{\partial o_{1, \text{old}}} * \frac{\partial o_{1, \text{old}}}{\partial w_{1, \text{old}}} = 0.01 * \frac{\partial L}{\partial o_{1, \text{old}}} * \frac{\partial o_{1, \text{old}}}{\partial w_{1, \text{old}}} = 0.01 * 0.025 * 0.25$$

$$o_{1, \text{new}} = \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1)$$

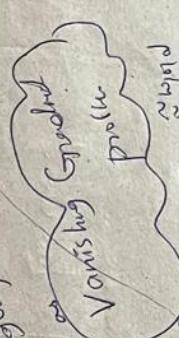
$$o_{1, \text{new}} = w_{1, \text{old}} * o_{1, \text{old}} + b_1$$

$$o_{1, \text{new}} = \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1)$$

$$\frac{\partial o_{1, \text{new}}}{\partial w_{1, \text{old}}} = \frac{\partial \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1)}{\partial (w_{1, \text{old}} * o_{1, \text{old}} + b_1)} = \boxed{0 \leq \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1) \leq 0.25}$$



That is it not going to converge
when in our weights are small
when to be updated are small



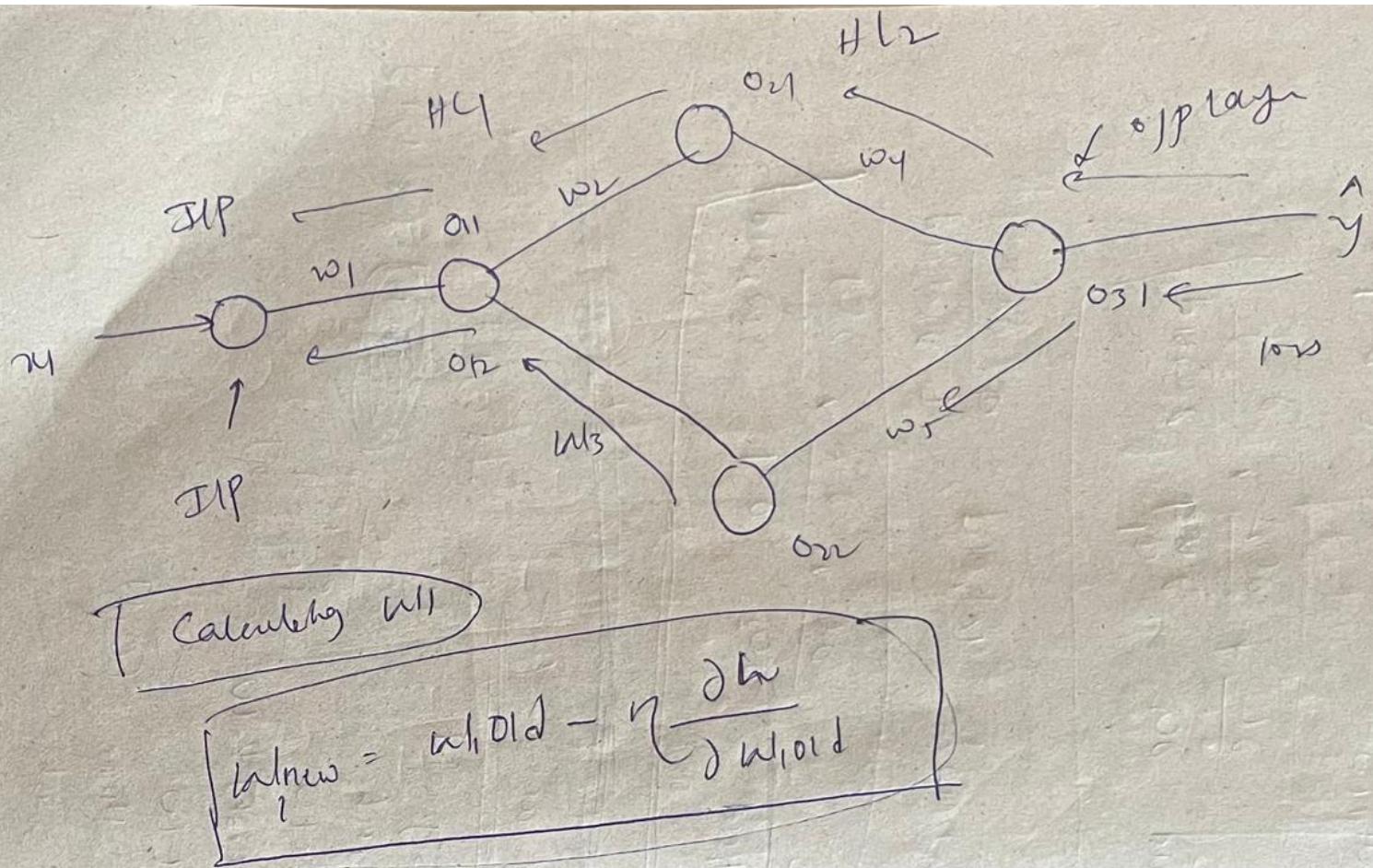
if we applying derivative of
sigmoid function

$$\boxed{0 \leq \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1) \leq 0.25}$$

$$o_{1, \text{new}} = \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1)$$

$$o_{1, \text{new}} = \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1)$$

$$\boxed{0 \leq \sigma(w_{1, \text{old}} * o_{1, \text{old}} + b_1) \leq 0.25}$$



$$\frac{\partial L}{\partial w_{11\text{old}}} = \left[\frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} + \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{11\text{old}}} \right] - \text{upper one}$$

+

$$\left[\frac{\partial L}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} + \frac{\partial o_{22}}{\partial o_{12}} * \frac{\partial o_{12}}{\partial w_{11\text{old}}} \right] - \text{lower one.}$$

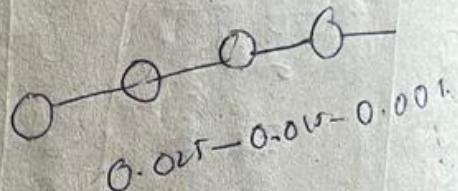
① To fix this problem research started
employing other activation function

→ Tanh

→ ReLU

→ pre ReLU

↳ Swiss



Sigmoid Activation Function

activation function for reducing Z value from (0-1)

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

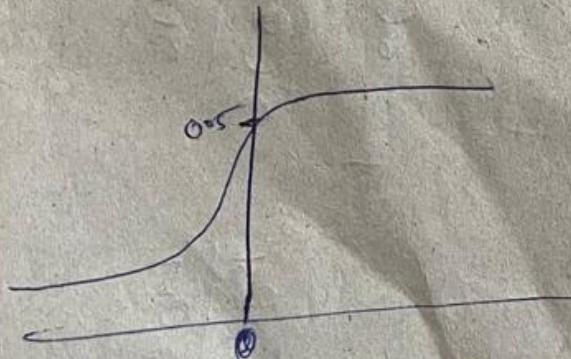
Vanishing Gradient
problem

$$0-0.25$$

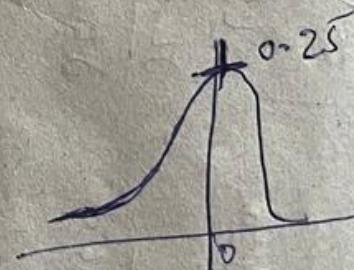
$$\sigma(z)$$

$$\sigma(u)$$

$$\Phi \frac{d\sigma(u)}{du}$$



$$0 \leq \sigma(z) \leq 1$$



$$0 \leq \sigma(z) \leq 1$$

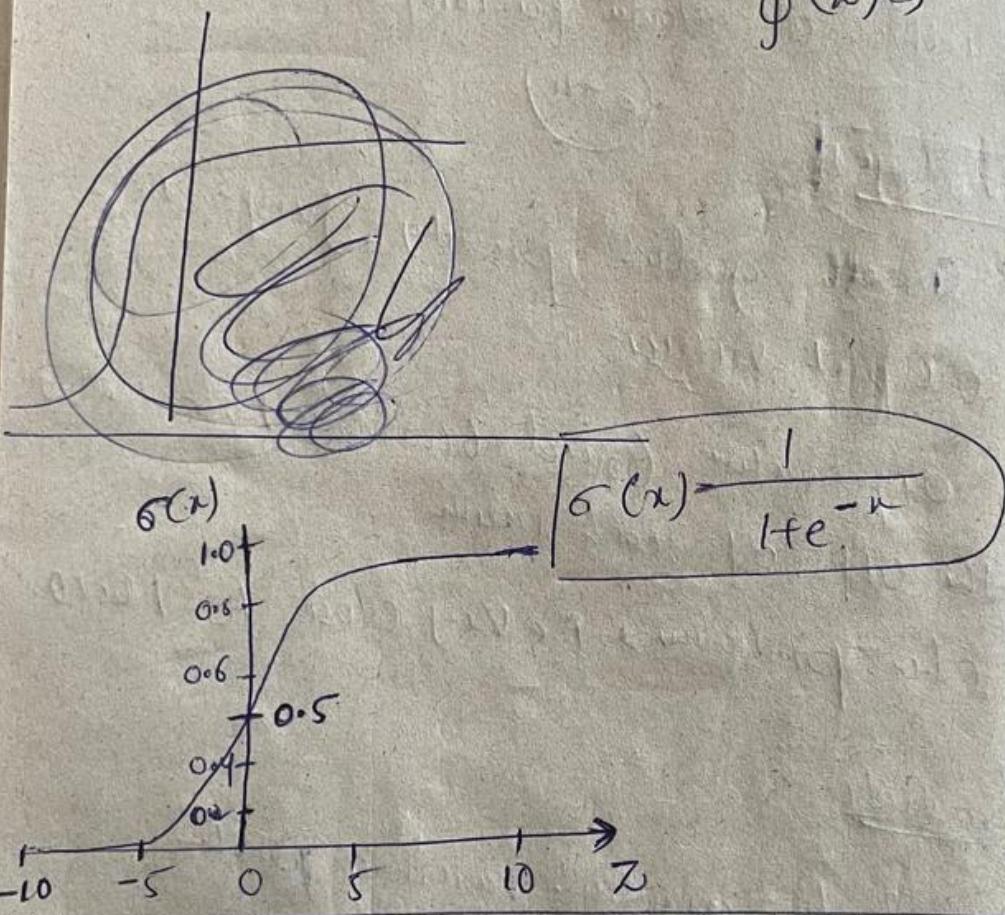
Activation function

① Sigmoid activation function [0 to 1]

$$z_i = \sum_{j=1}^n w_j^T x_j + b$$

$$\sigma(z) \Rightarrow 0 \text{ to } 1$$

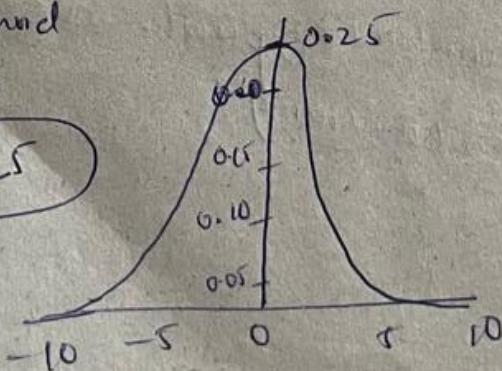
$$f(z) =$$



$$d\sigma(x)/dx$$

derivative wrt Sigmoid
funcn we get

$$0 \rightarrow 0.25$$



defects itself has some defects

- ① Leads to vanishing Gradient problem
- ② function output is not centred on 0, which will reduce the efficacy of wt update
- ③ Sigmoid performs exponential operation, which is slow for computers.

$$e^{-u}$$

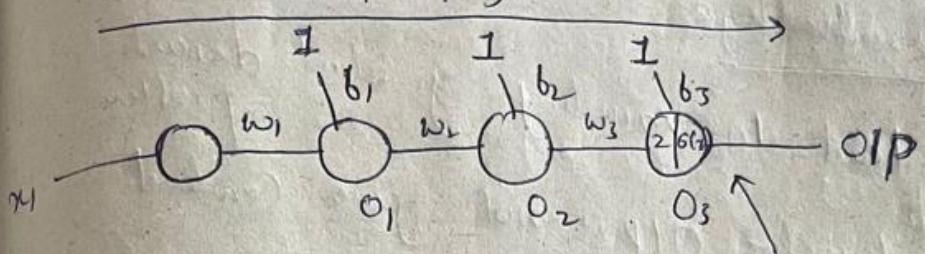
Advantages

- ① Smooth gradient, preventing jump in output values.
- ② OIP values bounded b/w 0 & 1, normalizing the OIP of each neuron
- ③ One production, i.e. very close to 100%.

disadvantages

- Leads to gradient vanishing problem
- function is not zero centred
- exponential operations are relatively time consuming.

Forward propagation



Backward propagation

$$w_{2nw} = w_{2old} - \left[\frac{\partial L}{\partial w_{2old}} \right] \quad \text{if } w_{2nw} \approx w_{2old}$$

$$\frac{\partial L}{\partial w_{2old}} = \frac{\partial L}{\partial o_3} * \frac{\partial o_3}{\partial z} + \frac{\partial o_2}{\partial z}$$

when we multiply it is going to be very small value.

$$\frac{\partial o_3}{\partial z} = \frac{\partial (\sigma(z))}{\partial z} * \frac{\partial z}{\partial o_2}$$

Let
 $z = (o_2 * w_3) + b_3$

$$= [0 - 0.25] * \frac{\partial [((o_2 * w_3) + b_3)]}{\partial o_2}$$

$$= [0 - 0.25] * w_3 \Rightarrow$$

advantages:

- Suitable for binary classification
- Clear prediction i.e. very close to 0 or 1

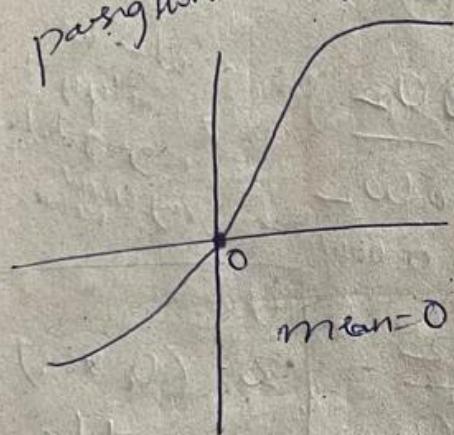
disadvantage

- leads to vanishing gradient descent problem
- function O(p) is not zero centered.
- math ops are relatively time consuming (expn)

$$\rightarrow e^{-x}$$

Zero centered

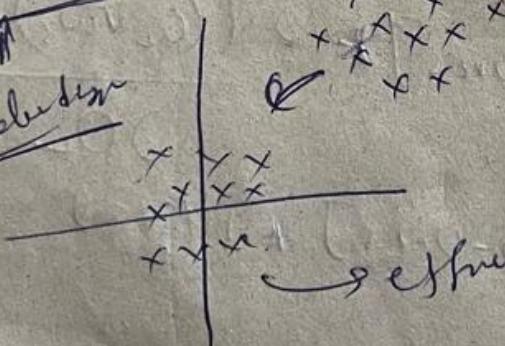
passing through origin



If it is zero centered:
eff. weight update is
going to happen

Zero centered

Stochastic



eff. weight update

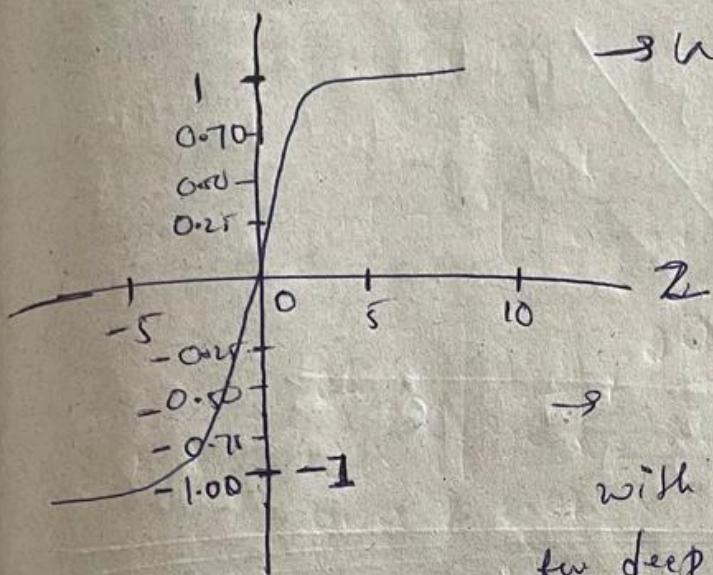
tanh activation function

()

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

[−1, 1]

$a \Delta V$



→ whole funcn is
○ centric

→ it may be better
with medium NN, but
for deep ~~funcn~~ Neural Netw.
it may cause vanishing grad
~~grad~~

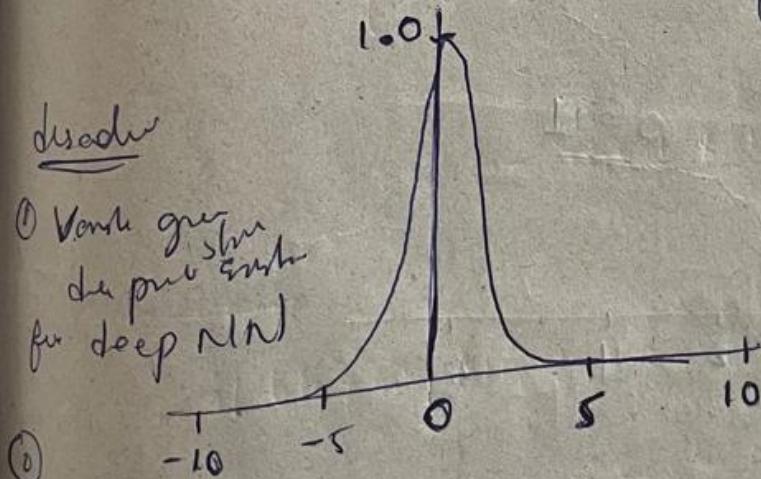
derivative of $\tanh(x)/\Delta V$ [0 to 1] put

[0 1]

disadv

with

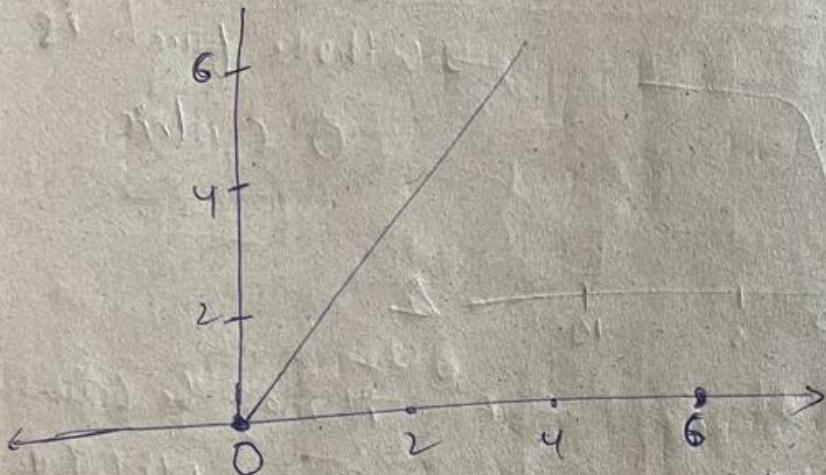
- ① Vanish grad
due to zero
for deep NN



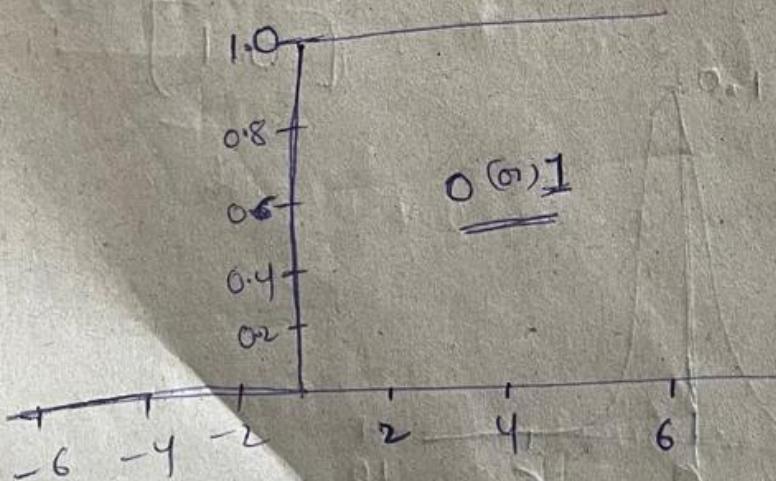
compl. funcn
compl. 0 → 0 0 0 0 0
0.005 0.0202 0.5 0.5

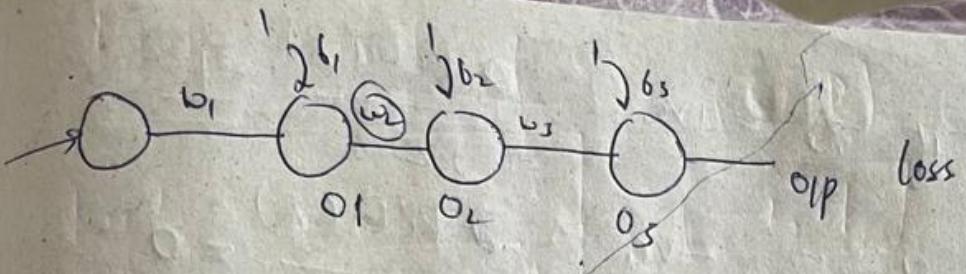
③ ReLU activation Function

$$\text{ReLU} = \max(0, x)$$



derivative of ReLU $\frac{d}{dx}(\text{ReLU}(x))$





$$\frac{\partial L}{\partial w_{2,old}} = \frac{\partial L}{\partial o_3} \times \boxed{\frac{\partial o_3}{\partial o_2}} \times \frac{\partial o_2}{\partial w_{2,old}}$$

$$\frac{\partial o_3}{\partial o_2} = \gamma \left(f_{\text{elu}}(o_2) \right) * \frac{\partial f_{\text{elu}}(o_2)}{\partial o_2} \quad z = (o_2 \times w_3) + b_3$$

$$= \begin{bmatrix} -ve & +ve \\ 0 or 1 \end{bmatrix} \times \frac{\gamma(o_2 \times w_3) + b_3}{\partial o_2}$$

$$= [0 \text{ or } 1] * w_3$$

if sum o/p is $\boxed{1} \rightarrow$ wt update will happen

if sum o/p is $\boxed{0} \rightarrow$ Dead neuron

$$w_{2,new} = w_{2,old} - \eta \boxed{\frac{\delta h}{\partial w_{2,old}}} \Rightarrow 0$$

if Derivative of Relu(z) is 0

$w_{2,new} \approx w_{2,old} \rightarrow$ Dead neuron

a neuron weighted any function

$$\text{if } z = +ve \quad \frac{\partial \text{ReLU}(z)}{\partial z} = 1$$

$$\text{if } z = -ve \quad \frac{\partial \text{ReLU}(z)}{\partial z} = 0$$

if you will get OIP of relu function as [1]
it solve vanishing gradient problem

advantages

- Solving vanishing gradient problem
- max(0, n) → calculate is Super fast.
- Since relu function has a linear relationship
- $\Re(z)$ is much faster than sigmoid

tanh

disadvantages

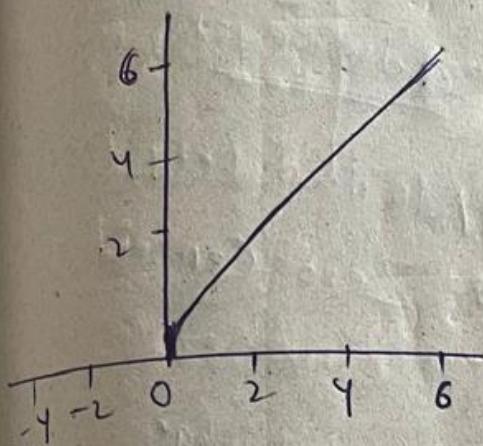
- Dead neuron.
- ReLU function OIP
(0m)
↳ $\Rightarrow 0$ is +ve number

$\Re(z)$ is not zero value

Leaky Relu and Parametric Relu

$$f(x) = \max(0.01x, u)$$

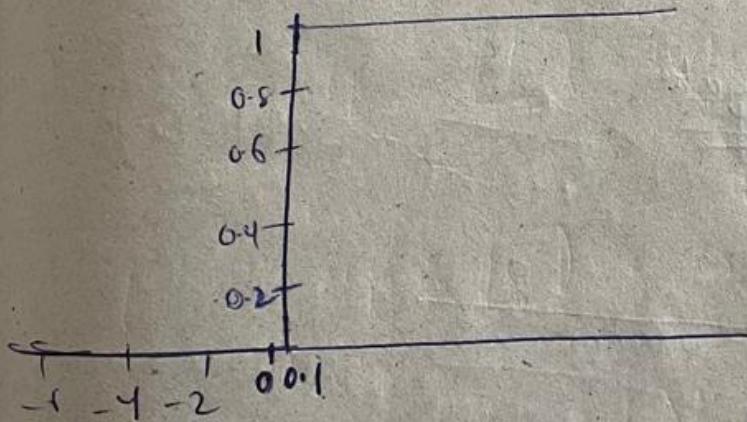
max($d_n x, u$)



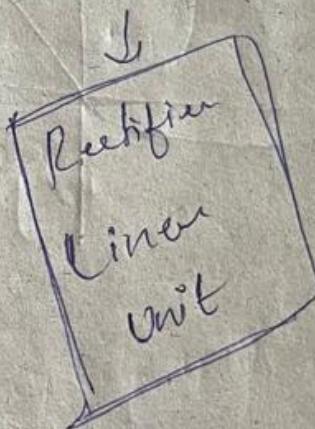
Relu \rightarrow Dead Neuron

↓
Dead ReLU problem.

derivative: $\frac{d f(x)}{dx}$



RELU



disadvantages

advantages
→ show all advantages of Relu

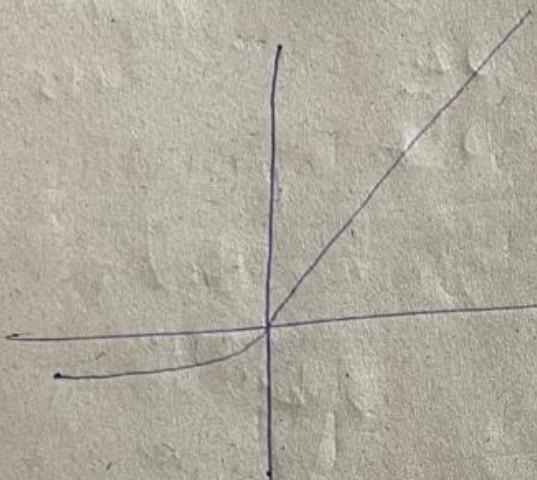
→ not going to zero
at centre

→ It removes the Dead
ReLU problem.

⑤ ECU (exponential linear units)

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

forward propagation



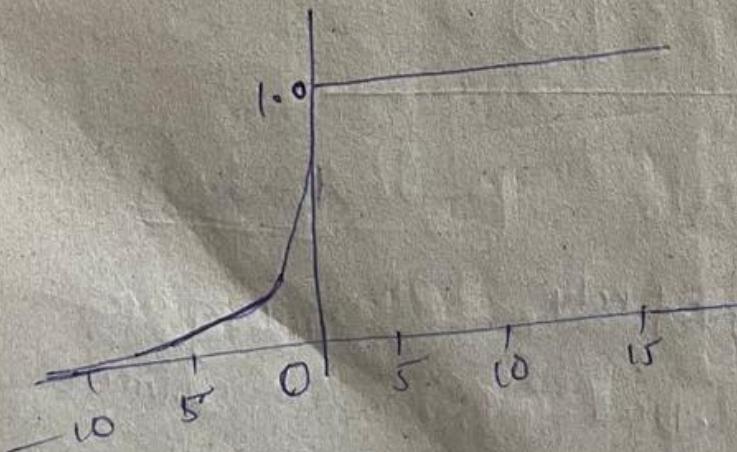
advantages

- No dead ReLU layer
- Zero Centred

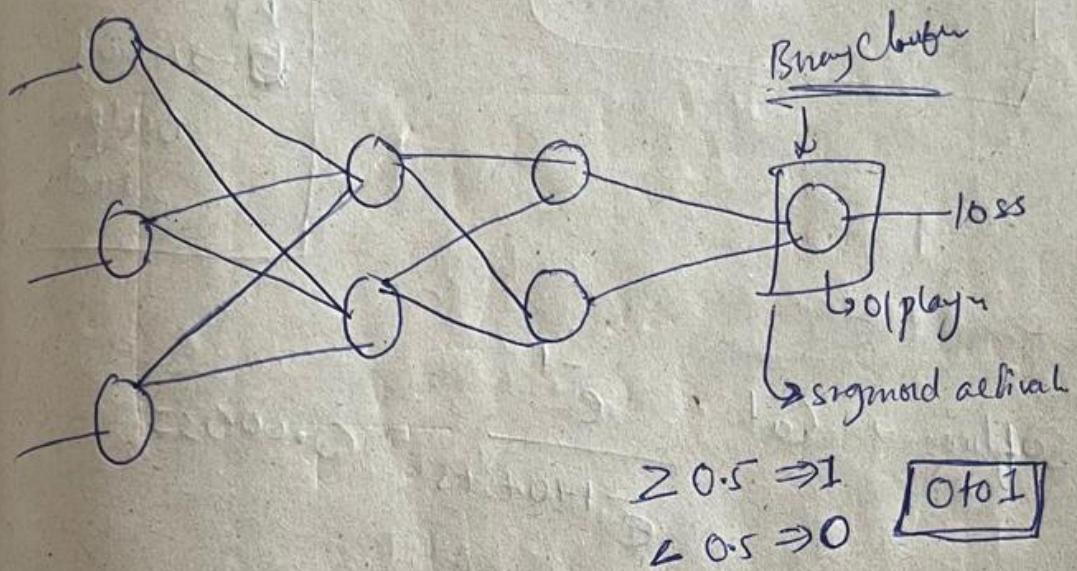
Disadvantage

- ① Slightly more computationally intensive

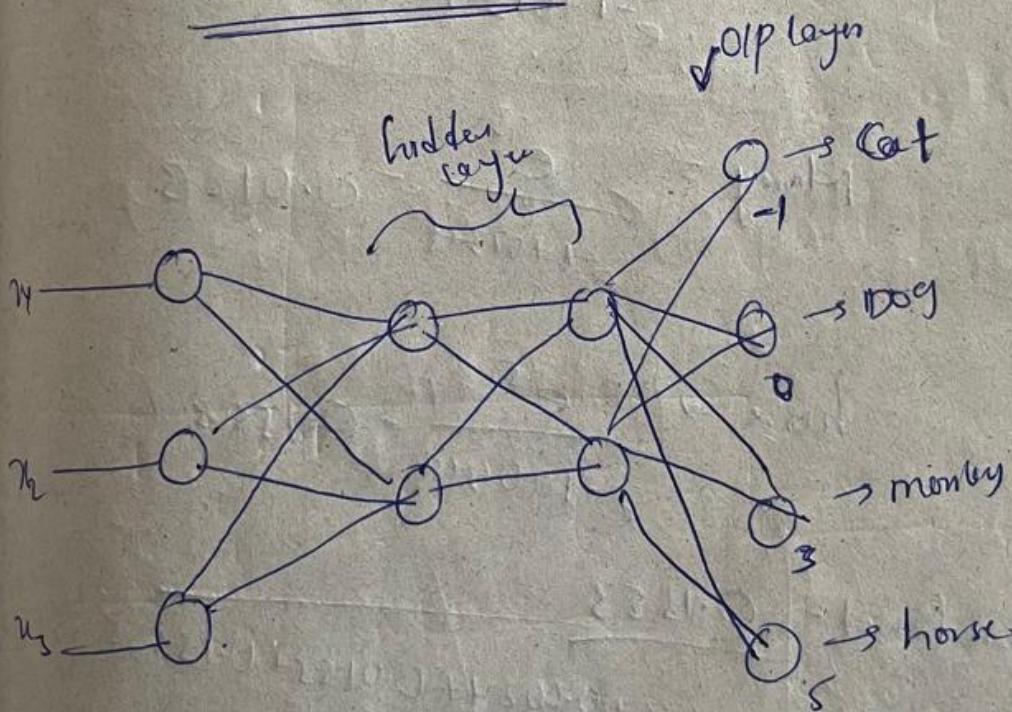
backward propagation



Softmax activation function (multiple classifer problem)



multiple classification



In these cases we should use softmax activation

$$\text{softmax} = \frac{e^{y_i}}{\sum_{k=0}^n e^{y_k}}$$

i → i^{th} node

node

2nd

3rd

n

② → no. of
output
node

$$\text{softmax} \rightarrow \text{Cat} = \frac{e^{-1}}{e^{-1+0+3+5}} = 0.00033.$$

$$\text{Dog} = \frac{e^0}{e^{-1+0+3+5}} = 0.0024$$

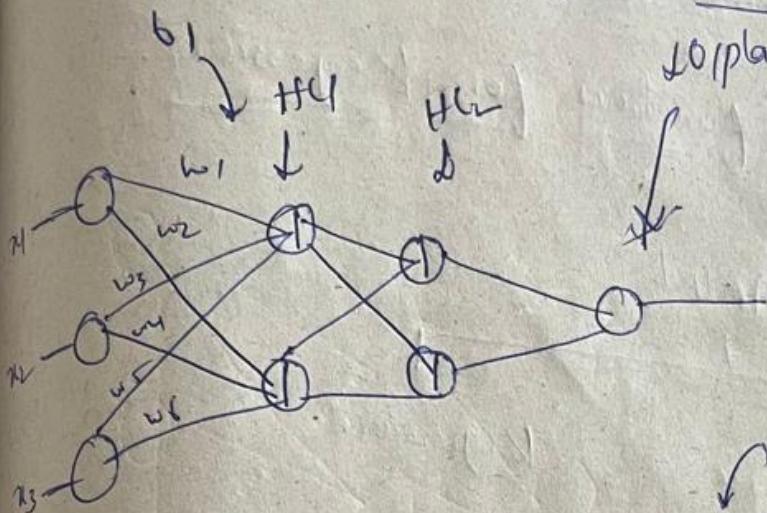
$$\text{Money} = \frac{e^3}{e^{-1+0+3+5}} = 0.0183$$

$$\text{horse} \rightarrow \frac{e^5}{e^{-1+0+3+5}} = 0.1353$$

$$\text{prob(horse)} = \frac{0.1353}{0.00033 + 0.0024 + 0.0183 + 0.1353}$$

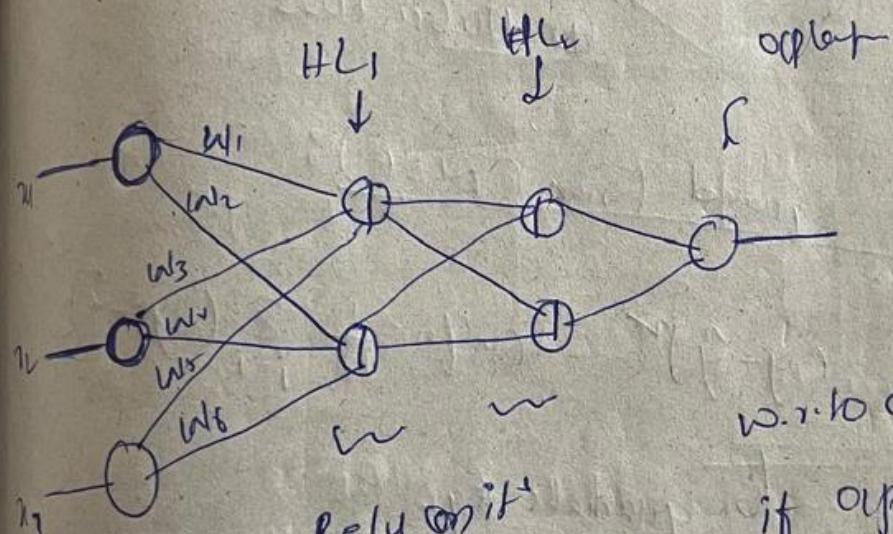
$$\approx = 86\%$$

which activation function to apply when.



more hidden layer \rightarrow sigmoid or tanh

Vanishing Gradient Descent prob



w.r.t. to op layer

if op layer is

binary class

use sigmoid

but if op

multiclass
class use

softmax or

hidden layer

Relu & its variants

op layer

Sigmoid \rightarrow binary class

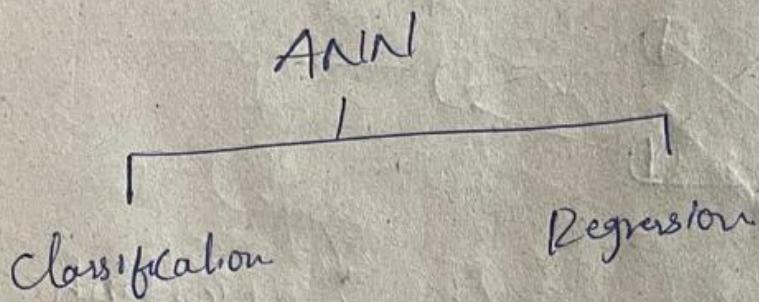
Softmax \rightarrow multiclass class

ReLU or its
various Leaky ReLU,
ELU, prelu.

in
hidden layer

loss function and cost fn

loss & Cost functions for Regression.

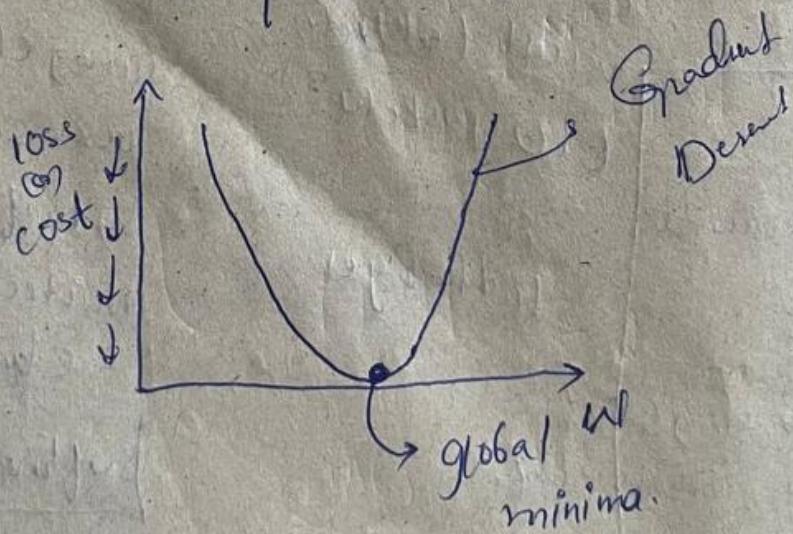


- ① MSE
- ② MAE
- ③ RMSE
- ④ Huber loss

① Mean Squared Error (MSE):

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

↳ quadratic eqn.



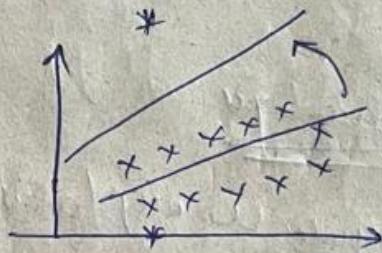
Weight update \rightarrow $W_{\text{new}} = W_{\text{old}} - \eta \frac{\partial L}{\partial W_{\text{old}}}$

advantages

- ① MSE is differentiable
- ② It has 1 (local) global minima.
- ③ It converges faster.

disadvantages

- ① Not robust to outliers



penalizing the error

Since we are squaring the error

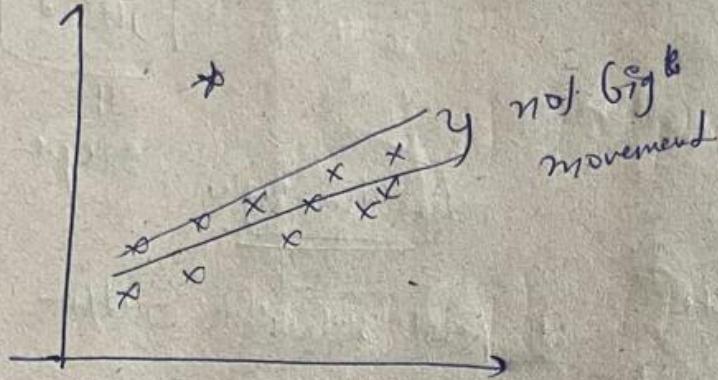
② Mean absolute error (MAE)

$$\text{loss fn} = |y - \hat{y}|$$

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

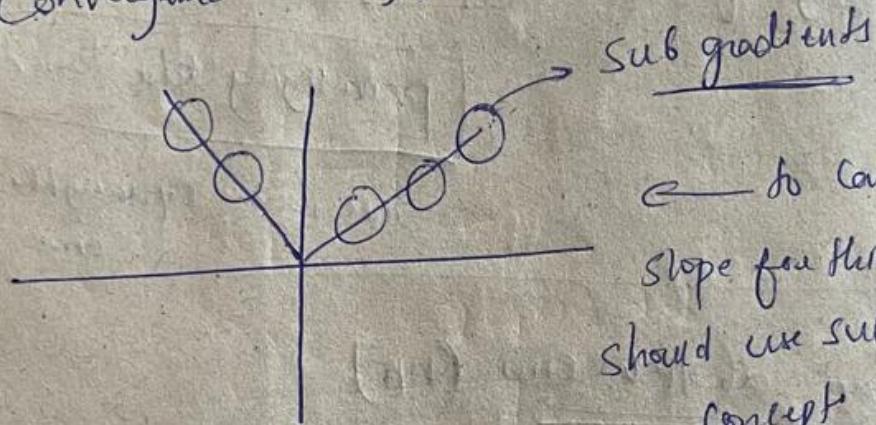
advantages

- ① Robust to outliers because we are not penalizing the error i.e. nothing but not squaring the error
- Even though there is a movement towards outliers, but not as much as in MSE, since here we are not squaring



disadvantages

→ Convergence usually takes time in MAE



Huber loss

Combination of MSE & MAE

$$\text{Cost function} = \begin{cases} \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \frac{1}{2} \int_{\text{outlier}}^{\text{outlier}} |y_i - \hat{y}_i|^2 & \text{otherwise} \end{cases}$$

↓
MAE

MSE

④ RMSE

$$\text{cost function} = \sqrt{\frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{N}}$$

Advantages

→ Remains in
Same unit

→ highlights big
mistake

Disadvantages

Not robust to outliers

Loss(④) Cost functions For classification problem

Classification \rightarrow cross entropy.

Binary cross

entropy {Binary}

Categorical cross

entropy {multiclass}

Spane categorical

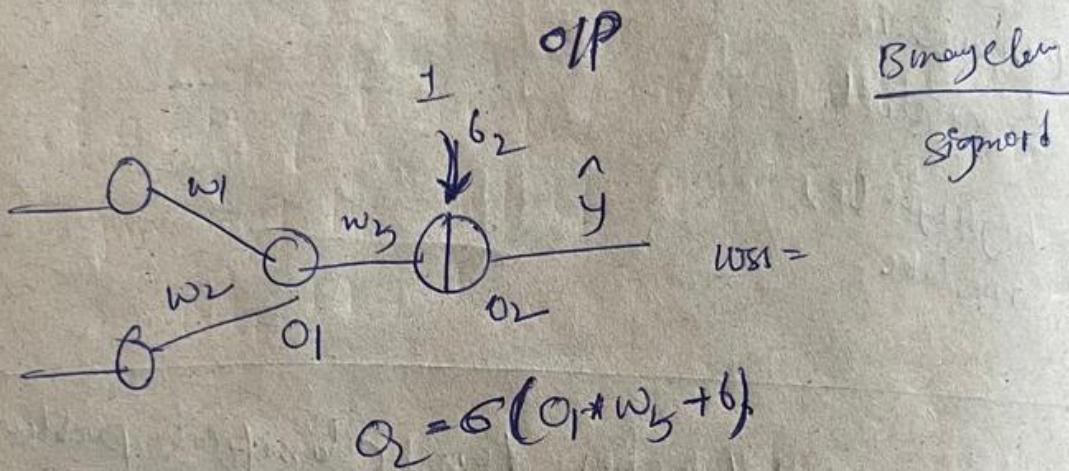
cross entropy

{multiclass}

① Binary cross entropy (log loss)

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

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



$$Q_2 = \sigma(O_1 * w_1 + b)$$

$$\hat{y} = \frac{1}{1+e^{-z}} \Rightarrow \text{Sigmoid activation function}$$

② Categorical cross Entropy (multi-classify)

f_1	f_2	f_3	Q_j	$\sum_{j=1}^3 Q_j$	Good	Bad	Neutral	Other
1	2	3	4	Good	1	0	0	
5	6	7	8	Bad	0	1	0	
9	10			Neutral	0	0	1	
				Other				

$$d(x_i, y_i) = - \sum_{j=1}^c y_{ij} * \ln(\hat{y}_{ij})$$

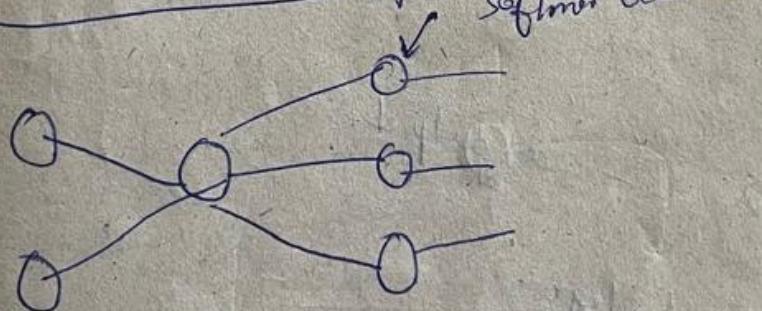
$\Rightarrow y_{ij} = [y_{11} \ y_{12} \ y_{13} \ \dots \ y_{1c}]$

$$= [y_{21}, y_{22}, y_{23}, \dots, y_{2c}]$$

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

product $\Leftrightarrow \hat{y}_{ij} \Rightarrow$ Softmax activation \Rightarrow

$$\text{soft}(z) = \frac{e^{z_i}}{\sum_{j=1}^c e^{z_j}}$$



OIP $\hat{y}_{ij} = \text{probab}$

Category 1 $\Rightarrow [0.2, 0.3, 0.5]$

Cross entropy
[this also gives the probab of other categories]

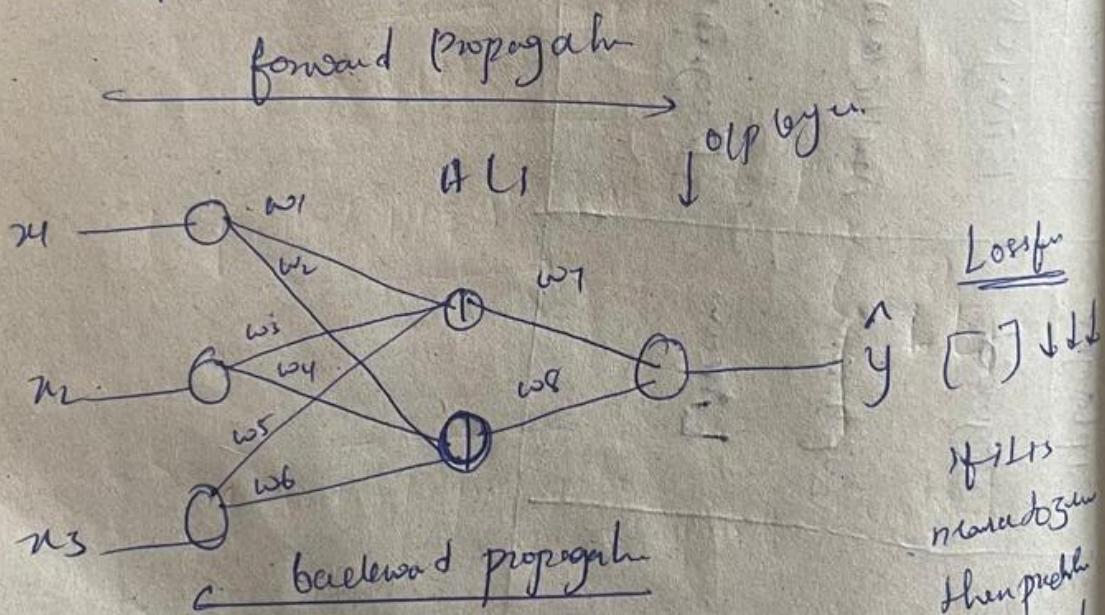
which loss function to use

Hidden layers	Output layer	loss function	
		problem	loss function
1	ReLU / tanh	Sigmoid	Binary cross entropy.
2	ReLU / tanh	Softmax	Categorical / sparse cross entropy.
3	ReLU / tanh	Linear	MSLE, MAE, huber loss, PPLS.
		Regression	

Optimizers

- ① Gradient Descent Optimizer
- ② Stochastic Gradient Descent (SGD)
- ③ Mini batch SGD
- ④ SGD with momentum
- ⑤ Adagrad and RMSprop
- ⑥ Adam Optimizers

① Gradient Descent Optimizer



Optimizers are responsible for reducing the

loss

③ Sparse Categorical Cross entropy

$\begin{matrix} 0 & 1 & 2 \end{matrix} \rightarrow \text{category}$
 $[0.2, 0.3, 0.5]$

↳ it returns the index of highest probability
i.e. 2 index

at the end Sparse Categorical Cross entropy
does not find all probabilities, instead it
gives only the index of highest probability
i.e output result

$\begin{matrix} 0.2, 0.3, 0.1, 0.2, 0.2 \end{matrix}$
 $\begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix}$

↳ 1 as output

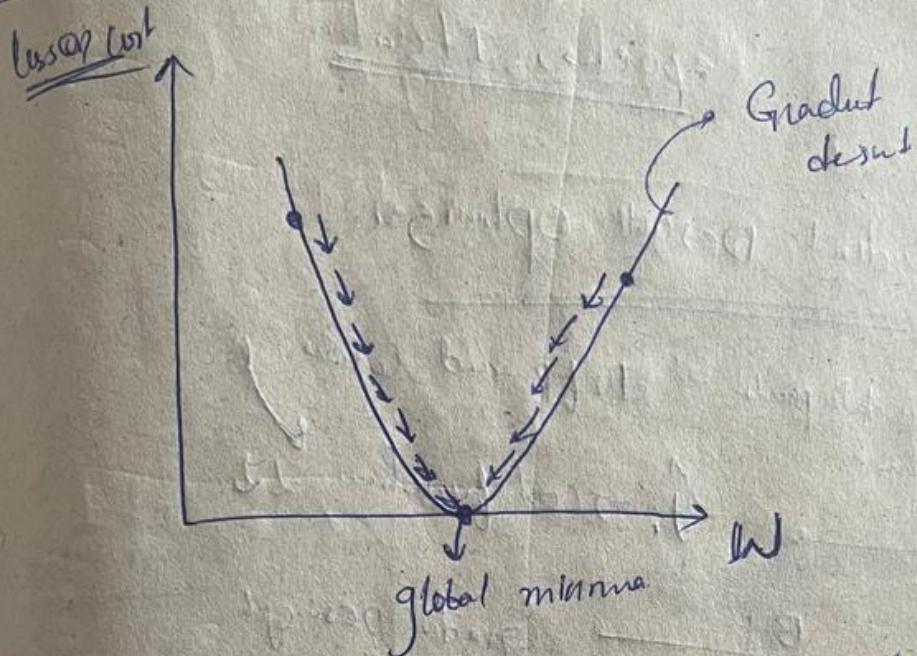
disadvantage

→ losing info about the probability of other
Categories

Weight update

$$\text{Weight} = \text{Weight} - \eta \frac{\partial L}{\partial \text{Weight}}$$

Learning rate



Once it comes to global minima, slope is zero then

$$\text{Weight} \approx \text{Weight}$$

no weight updation is going to happen
at that point Loss \downarrow

MSE

Sum of all the errors

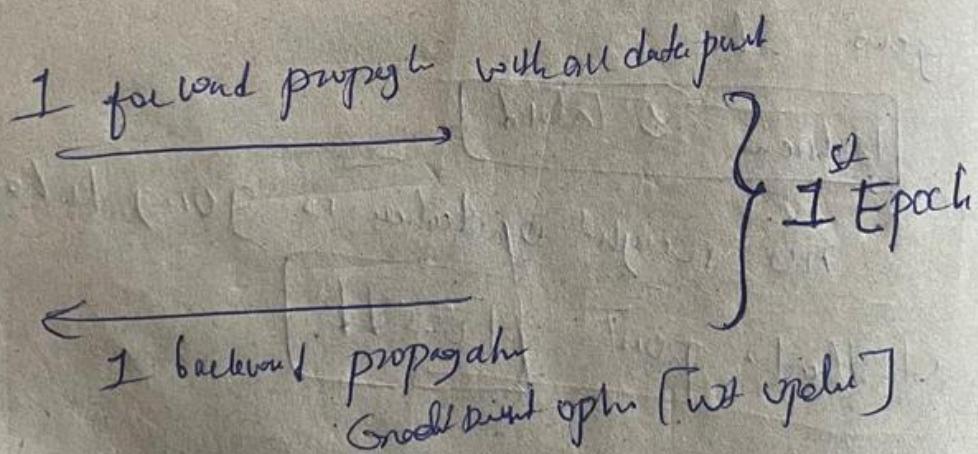
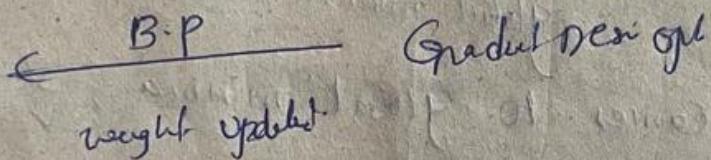
$$\text{Loss fn} = (y - \hat{y})^2 \quad \text{Cost fn: } \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

↓
subject to
epochs, Iterations

Gradient Descent Optimizer.

Take all data points do f.P. and calculate J.
at a time.

Value ... $\hat{y}_i \rightarrow$ cost function $J \downarrow$

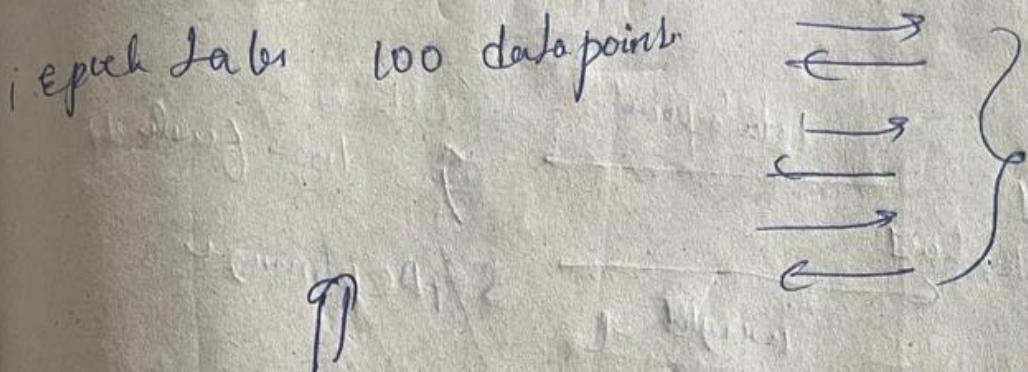


$\hat{y}_i \rightarrow$ cost fn. } 2nd epoch
Grad

$\hat{y}_i \rightarrow$ cost ≈ 0 nth epoch

Batch

$$10 \text{ Batches} = \frac{1000}{10} = 100$$



but in gradient descent one epoch will take
all the data points

advantage:

→ convergence will happen

disadvantage:

- ① Huge Resource RAM, GPU

if we take

1 minibatch

1 minibatch

and then we update

↓

Resource Intensive

ADAM Optimizer (Best optimizer)

SGD with momentum + RMSprop [Dynamic LR + Smoothing]

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

$$b_t = b_{t-1} - \eta' Vdb$$

$$\eta' = \frac{\eta}{\sqrt{sdw_t + E}}$$

$$sdw_t = 0$$

$$sdw_t = \beta * sdw_{t-1} + (1-\beta) \left(\frac{\partial L}{\partial w_t} \right)^2$$

wt update

$$Vdw_t = \beta * Vdw_{t-1} + (1-\beta) \frac{\partial L}{\partial w_t}$$

bd update

$$Vdb_t = \beta * Vdb_{t-1} + (1-\beta) \frac{\partial L}{\partial b_t}$$

↳ momentum

y

smoothing

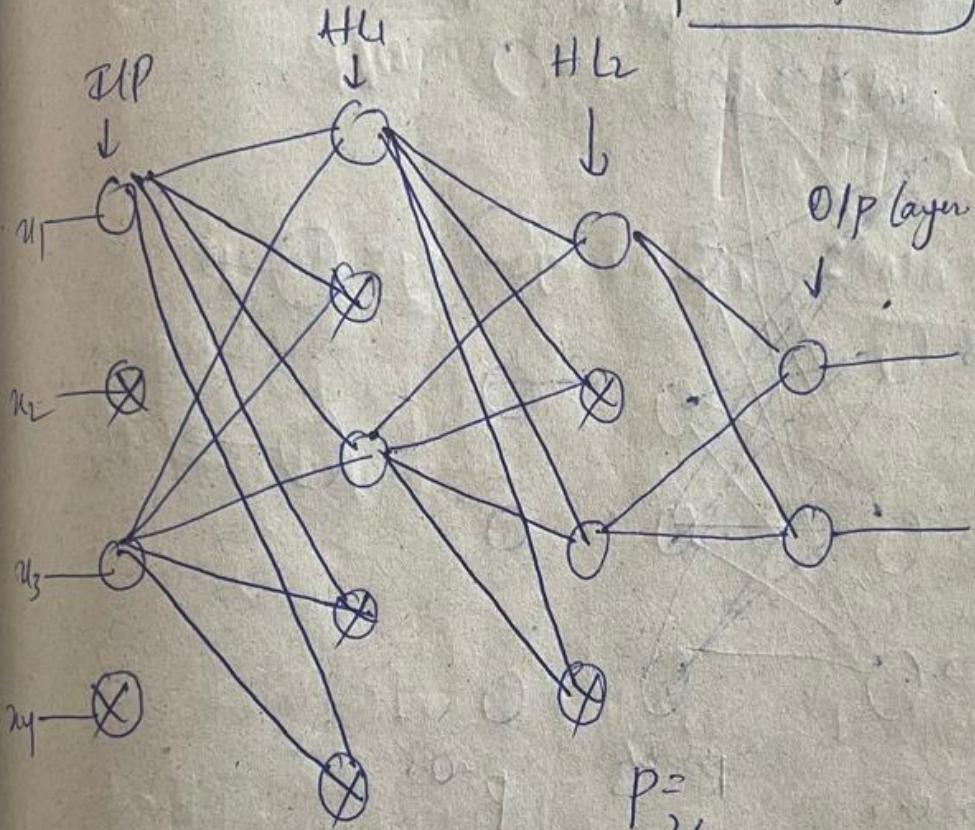
Best optimizer to use in any ANN!

Similarly how ML models undergo Overfitting
 Neural networks also undergo Overfitting.
 so to handle it we use concept called.

Drop Out layer.

drop out rate (P)

$$0 \leq p \leq 1$$



$$\frac{2}{4} \text{ or } p = 0.5$$

$$P = 3/5$$

$$P = \frac{1}{2}$$

$$= 0.5$$

50% of input
nodes are

deactivated.

No propagation happens with it.

We are going to

deactivate few neurons
along with few feature samples

so that this entire
model does not overfit

Padding in CNN

Used to avoid loss of information.

$$\begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 \end{array}
 *
 \begin{array}{|c|c|c|} \hline
 +1 & 0 & -1 \\ \hline
 +2 & 0 & -2 \\ \hline
 +1 & 0 & -1 \\ \hline
 \end{array}$$

6x6

$$= \begin{array}{|c|c|c|c|} \hline
 0 & -4 & -4 & 0 \\ \hline
 0 & -4 & -4 & 0 \\ \hline
 0 & -4 & -4 & 0 \\ \hline
 0 & -4 & -4 & 0 \\ \hline
 \end{array}$$

adding cushions, may be 1, 2, 3 ---

based on output

after applying one layer of padding

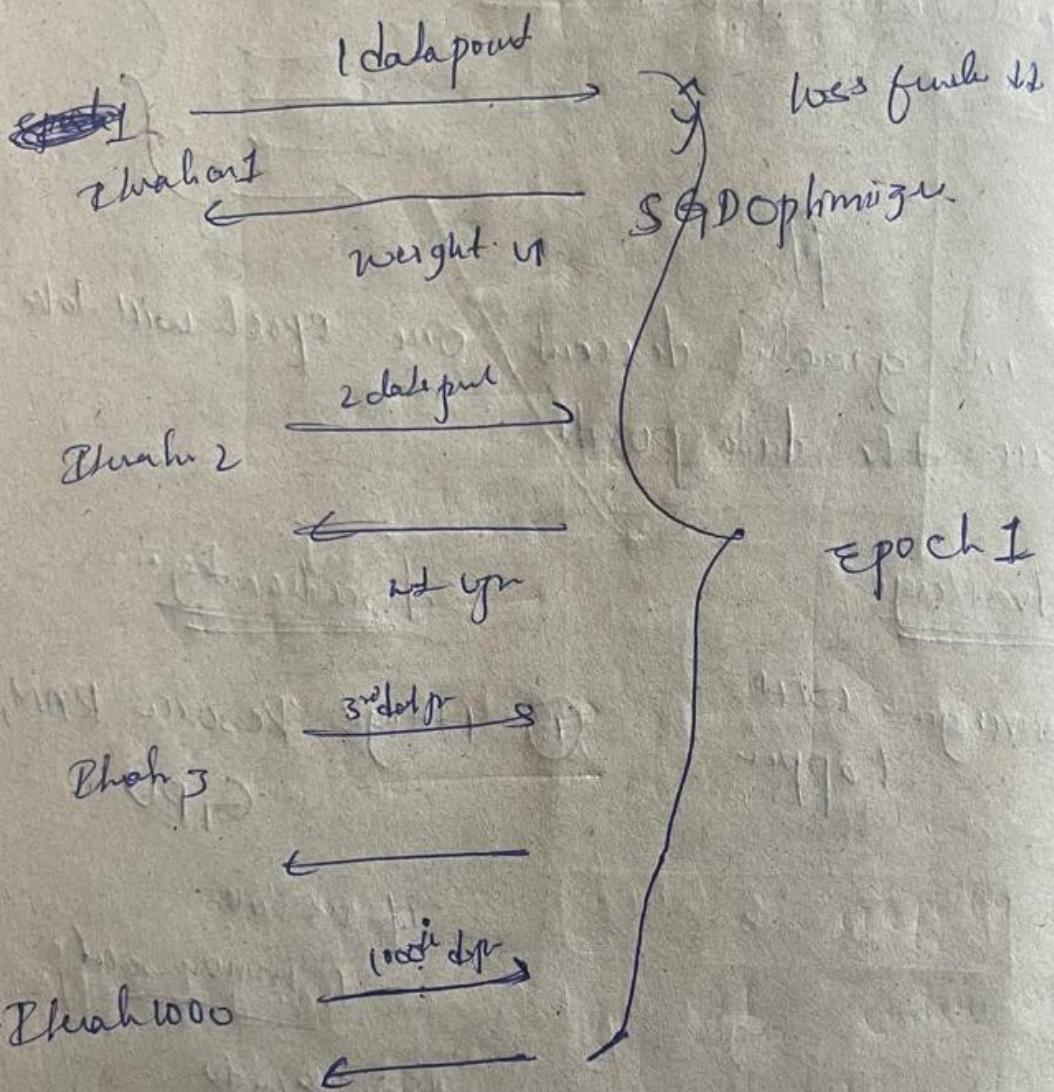
$$\begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 \end{array}
 *
 \begin{array}{|c|c|c|} \hline
 +1 & 0 & 1 \\ \hline
 +2 & 0 & -2 \\ \hline
 +1 & 0 & 1 \\ \hline
 \end{array}$$

6x6

8x8 3x3

Stochastic Gradient Descent (SGD)

epochs +
streak



Only we decrease 1
reduce loss
function

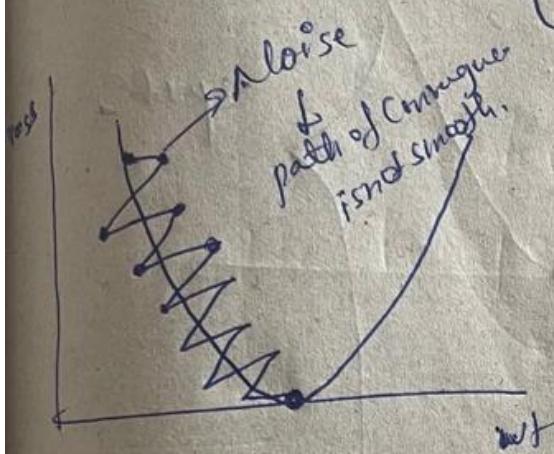
advantage

- ① Solves Resource issues

Disadvantage

- ① Time complexity $T T$

- ② Convergence won't also take more time
- ③ Noise gets involved.



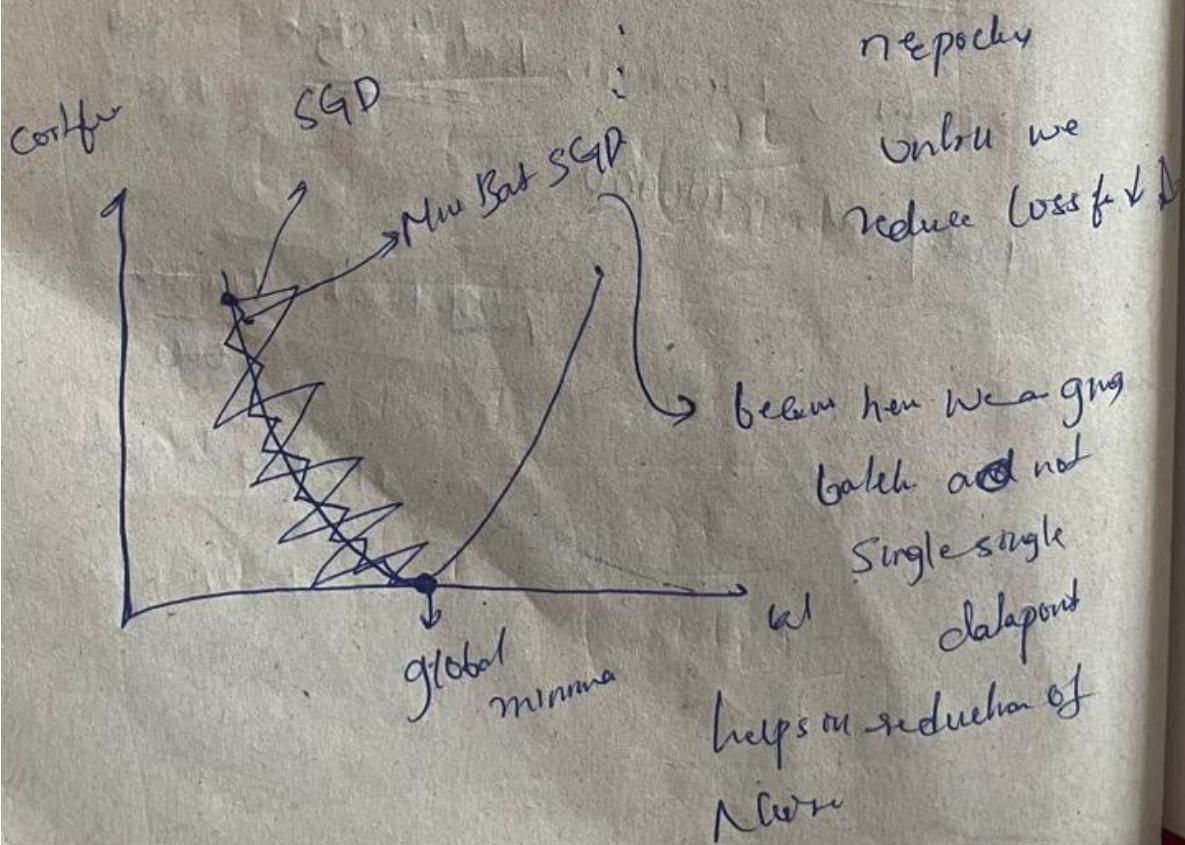
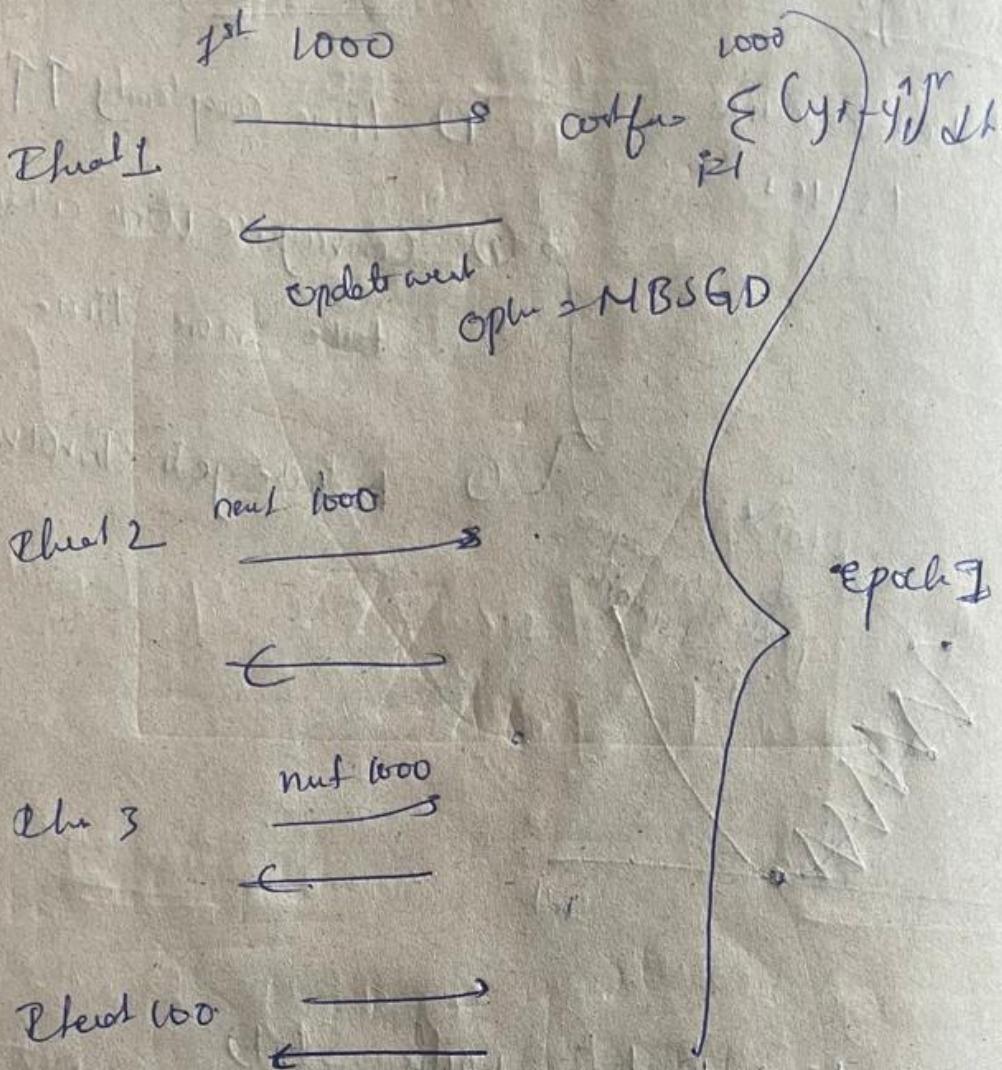
Mini Batch with SGD

epochs, Iterations, Batch-size = 1000

Data points = 100000

$$\text{No. of Iterations} = \frac{100000}{1000}$$

$$= 100 \text{ Iterations}$$



advantages

- 1. converges faster
- 2. more memory required compared to SGD
- 3. less noise compared to SGD
- 4. efficient resources (RAM)

disadvantages

→ More stability

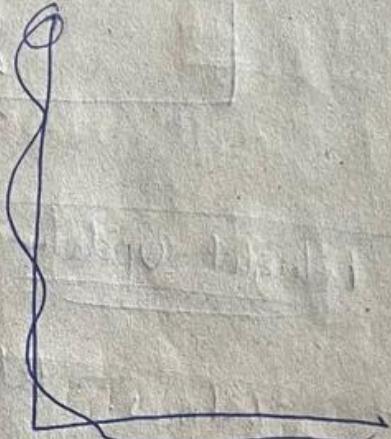
↓ need

Smoothing the
curve

(sgd) → for 1 day ✓

(sgd) → for 1000 days ✓

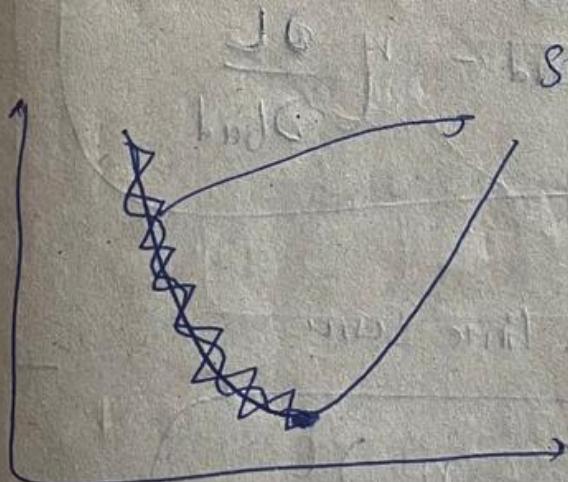
(sgd) → 5000 days



Smoothing the
curve

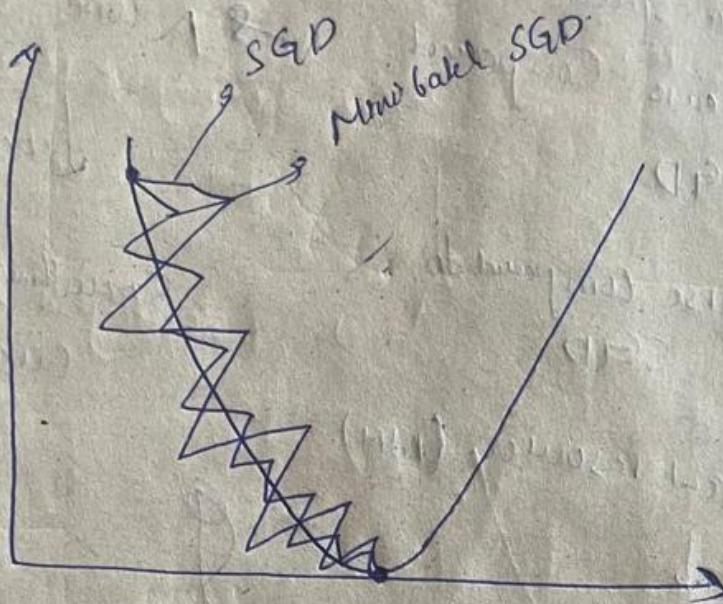
using SGD with

momentum



④

SGD with Momentum



Weight update:

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

Stochastic Grav update

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

W.r.t time series

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$

Smoothing is going to be happen



Exponential Weighted Average (to reduce noisy)

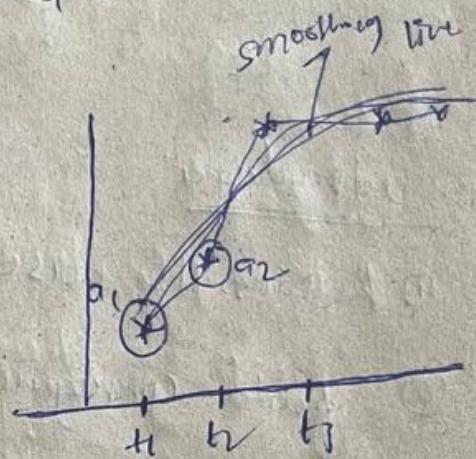
Time = $t_1 \ t_2 \ t_3 \ t_4 \ \dots \ t_n$ } time

Value = $a_1 \ a_2 \ a_3 \ a_4 \ \dots \ a_m$ } series

$$V_{t_1} = a_1$$

$$\beta = 0.95$$

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



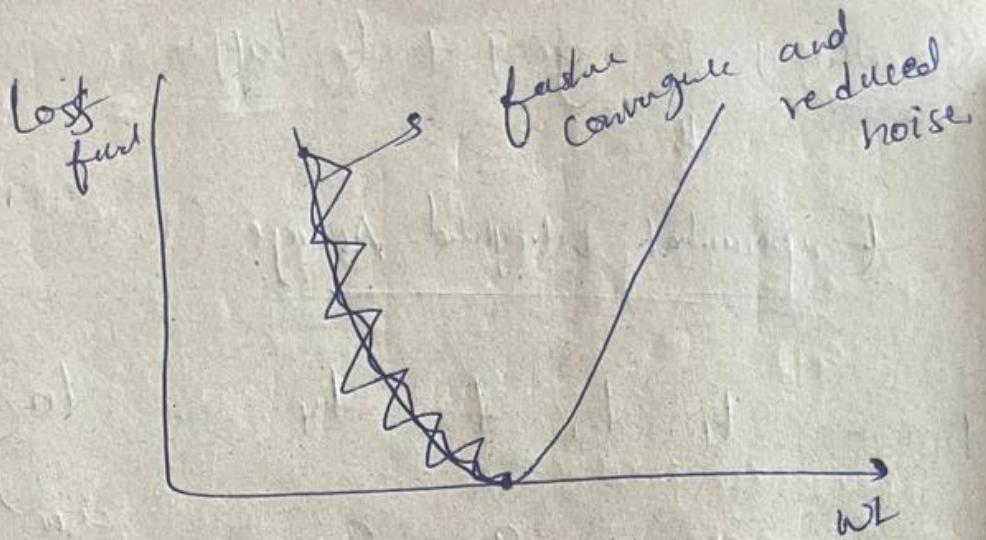
$$\beta = 0.90$$

$$V_{t_2} = 0.95 * a_1 + (0.05) * a_2$$

↓
it is given high so that previous point controls the hold / smoothing and current point has less control towards smoothing

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

$$= 0.95 [0.95 * a_1 + (0.05) * a_2] + (0.05) * a_3$$



advantages

- ① Reduces the noise
- ② faster convergence

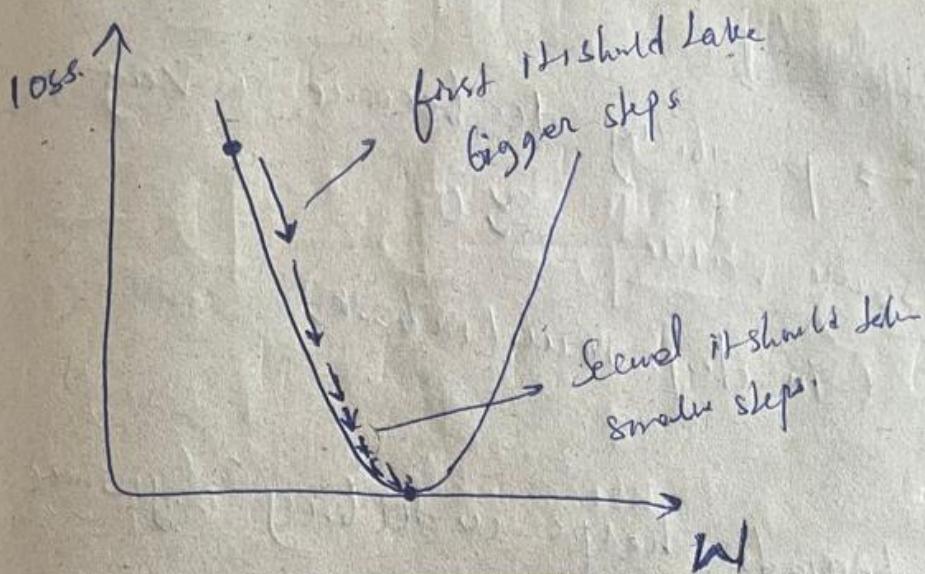
Adagrad :

Adaptive gradient Descent

$$w_t = w_{t+1} - \eta \frac{\partial L}{\partial w_t}$$

learning rate = 0.001

until now $\eta = \text{fixed} \rightarrow$ lets more of dynamic



as the convergence happen the learning rate should change

This is what called as decay

$$w_t = w_{t-1} - \eta' \frac{\partial L}{\partial w_{t-1}}$$

$$\eta' = \frac{\eta}{\sqrt{d_t + \epsilon}} \quad \boxed{\eta' \downarrow \downarrow \quad \epsilon \uparrow \uparrow}$$

\rightarrow epsilon \approx small value

$$d_t = \sum_{t=1}^t \left(\frac{\partial L}{\partial w_t} \right)^2$$

$\overbrace{t=1} \quad \overbrace{t=2} \quad \overbrace{t=3}$

$\eta = 0.01 \quad \eta = 0.005 \quad \eta = 0.003$

disadvantage:

- ① $\eta' \rightarrow$ possibility to become a very small value ≈ 0

(in deep neural network)

\therefore adagrad helps us do long dynamic learning rate, that means initially learning rate will be high and it will keep on decreasing towards global minima

Adadelta and RMSprop

$\beta = 0.95$ exponential weight average

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

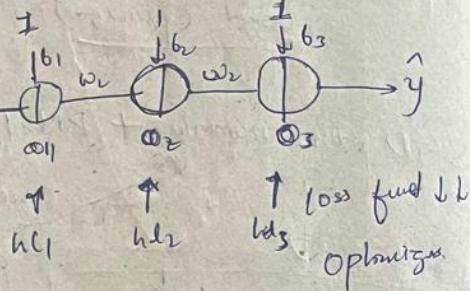
$$S_{dw} = \beta * S_{dw-1} + (1-\beta) \left(\frac{\partial L}{\partial w_t} \right)^2$$

[Dynamic LR + smoothing EWA]

$$w_t = w_{t-1} - \eta' \frac{\partial L}{\partial w_t}$$

Exploding

Gradient problem



w_1 update

$$\textcircled{1} \quad w_{1,\text{new}} = w_{1,\text{old}} - \eta \left[\frac{\partial L}{\partial w_{1,\text{old}}} \right]$$

$$\frac{\partial L}{\partial w_{1,\text{old}}} = \frac{\partial L}{\partial o_3} * \left[\frac{\partial o_3}{\partial o_2} \right] + \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{1,\text{old}}}$$

Chain rule of derivative

$$\rightarrow l^{(4)} = z = (o_2 * w_2 + b_2) \\ \rightarrow \sigma(z)$$

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial (\sigma(z))}{\partial z} * \frac{\partial z}{\partial o_2}$$

derivative of sigmoid

$$= [0 - 0.25] * \frac{\partial (o_2 * w_2 + b_2)}{\partial o_2}$$

$$= [0 - 0.25] * w_2$$

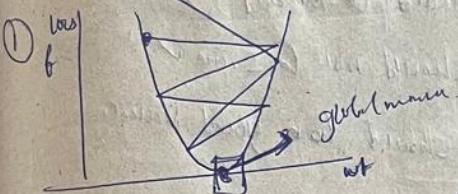
weight initialized

Very high value

lets have what if i
initialized w_1 very high
old
500 - 1000

two scenarios:

- ① $w_{1,\text{new}} \gg w_{1,\text{old}}$
- ② $w_{1,\text{new}} \ll w_{1,\text{old}}$



Since weight operation is having with a very high value, it may explode here and there may not reach to global minima and chance of going outside the gradient descent

this leads to

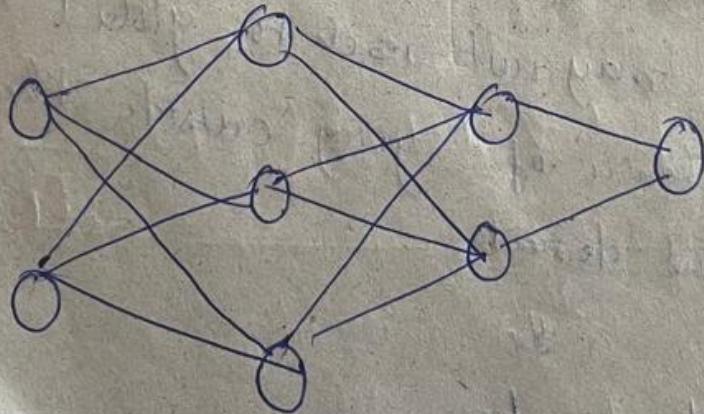
Exploding Gradient problem

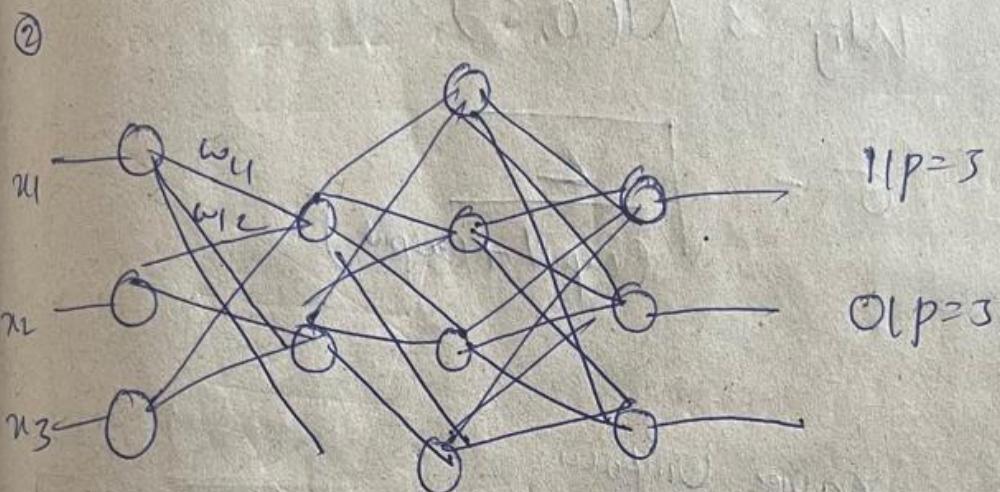
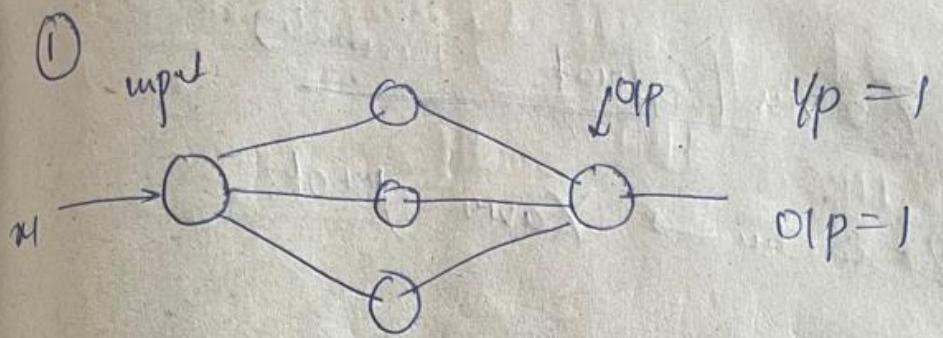
Weight Initialization Technique

- ① uniform Distribution ✓
- ② Xavier/Glorot Initialization ✓
- ③ Kaiming He Initialization. ✓

Key points:

- ① Weights should be small. ✓
- ② Weights should not be same ✓
- ③ Weights should have good variance





① Uniform distribution

$$w_{ij} \approx \text{uniform distribution} \left[-\frac{1}{\sqrt{\text{input}}}, \frac{1}{\sqrt{\text{input}}} \right]$$

$$\left[-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}} \right]$$

② Xavier Glorot Initialization

Rescale \rightarrow Xavier glorot.

① Xavier Normal Init

$$w_{ij} \approx N(0, \sigma^2)$$

$$\sigma = \sqrt{\frac{2}{(\text{input} + \text{output})}}$$

② Xavier Uniform

$w_{ij} \approx \text{Uniform}$
Distribution

$$\left[\frac{-\sqrt{6}}{\sqrt{\text{input} + \text{output}}}, \frac{\sqrt{6}}{\sqrt{\text{input} + \text{output}}} \right]$$

Kaiming lie Punktation.

⑤

① He Normal

$$w_{ij} \approx N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{\text{input}}}$$

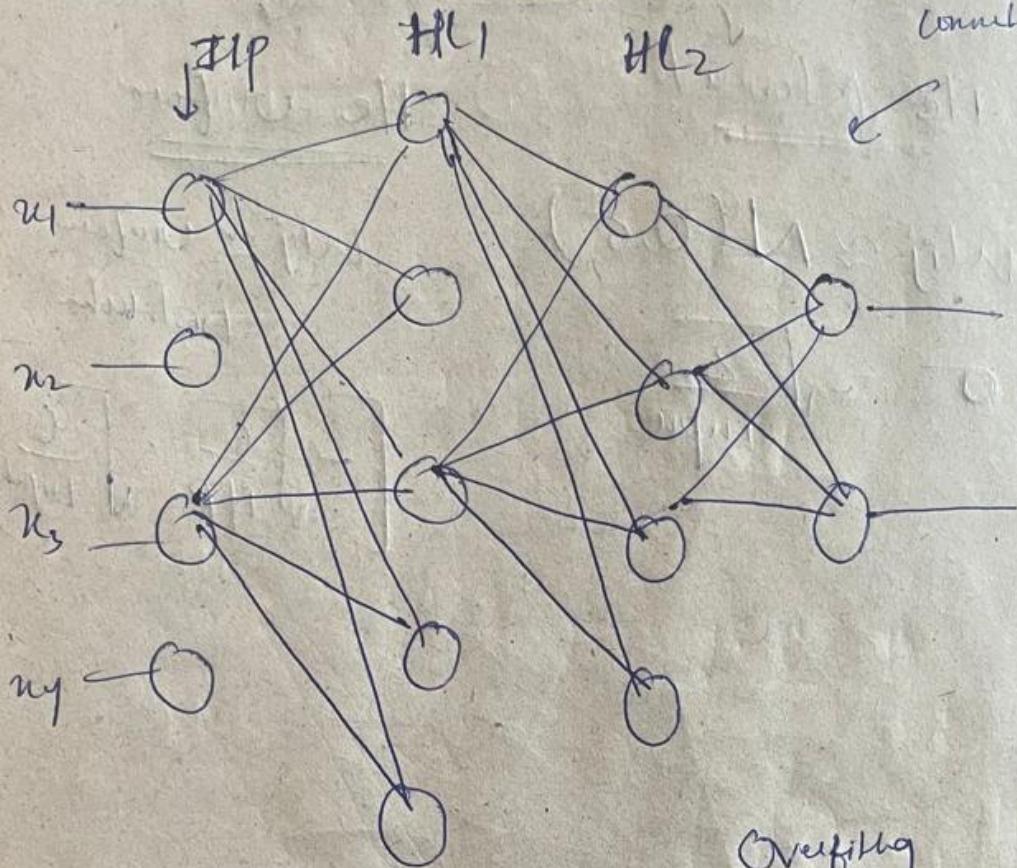
He uniform

$$w_{ij} \approx \text{uniform}$$

Distributie

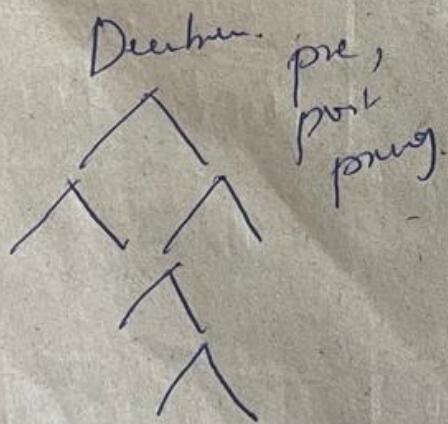
$$\left[-\sqrt{\frac{6}{\text{input}}} \dots \sqrt{\frac{6}{\text{input}}} \right]$$

Drop out layer



Overfitting

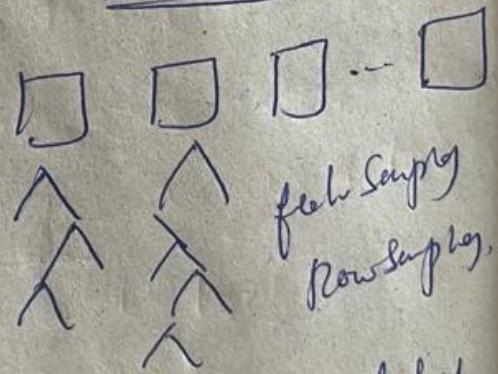
Machine Learning



is constructed with
despite leads to overfitting

model \rightarrow Train acc $\uparrow \uparrow 90\%$
Test acc $\downarrow \downarrow 60\%$

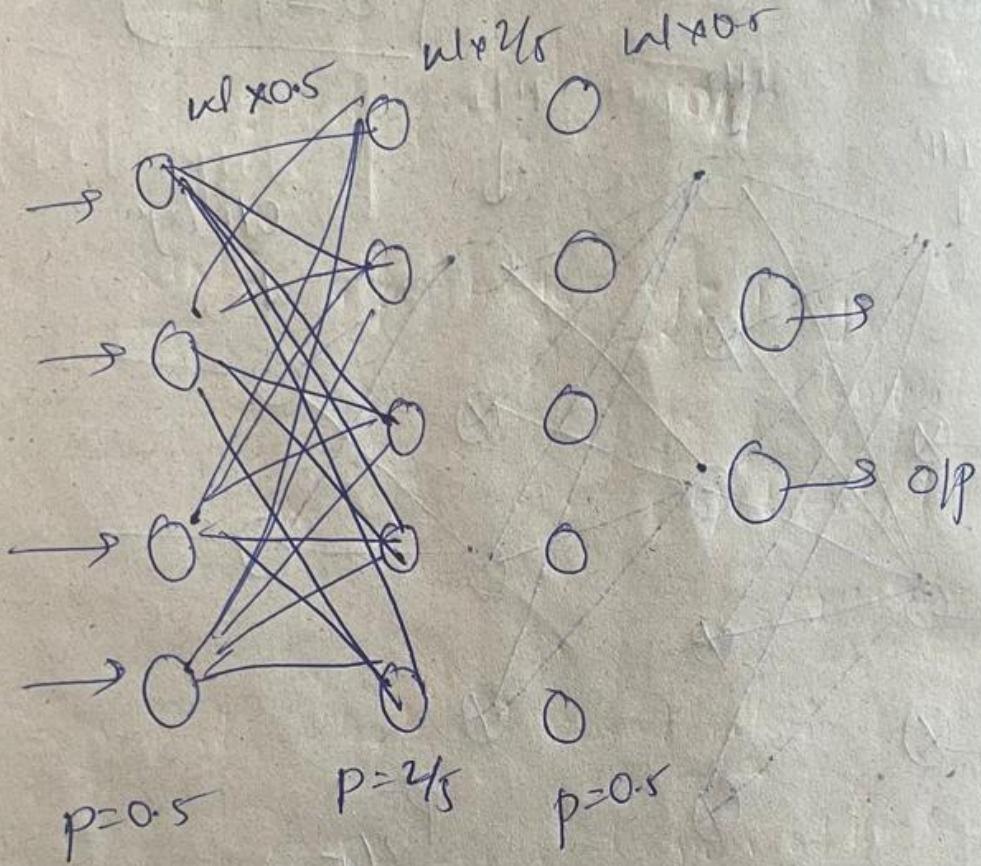
Random Forest



subset of dataset

after model got trained and weights
get fixed

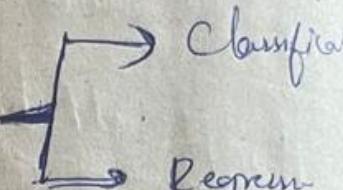
for test data



We don't deactivate the neurons in test data

} at the end dropout layers are used to
create generalized model, to prevent
Overfitting.

CONVOLUTIONAL Neural Network (CNN).

① ANN → Supervised Learning  Classification
Regression

Dataset: IIP features OIP

② CNN: IIP \rightarrow Image e.g.: Image classification

Object detection,
Segmentation

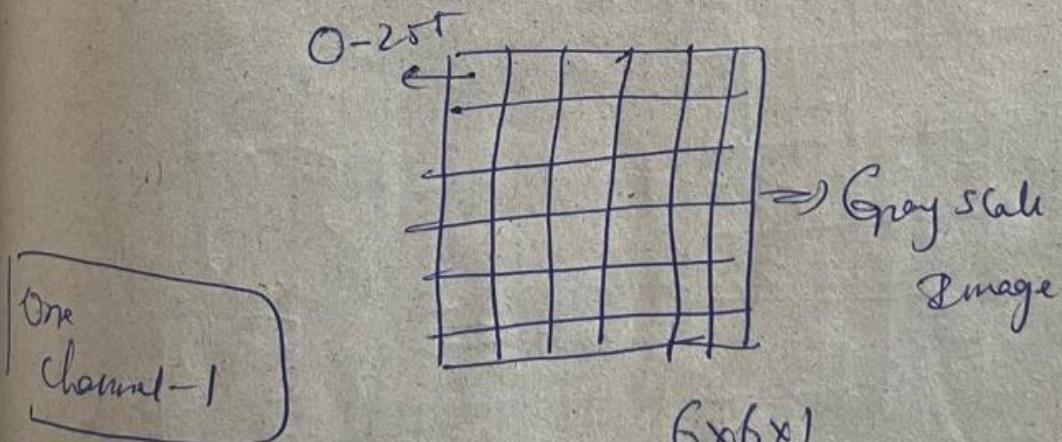
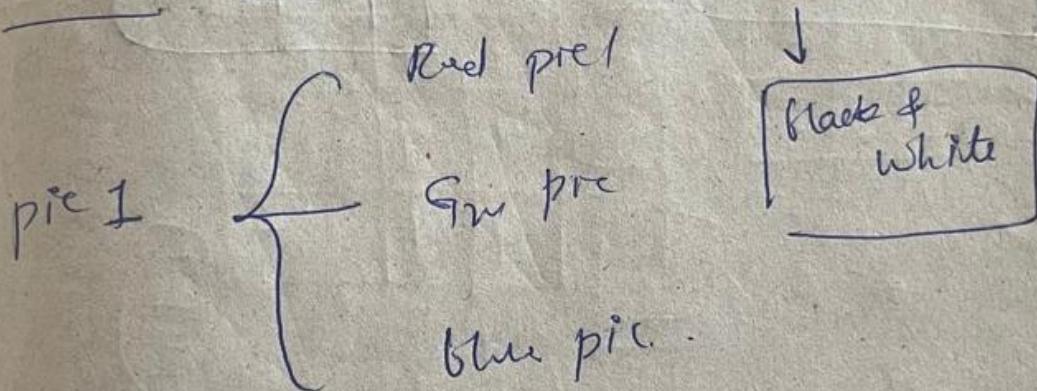
Cerebral cortex and visual cortex:

Visual cortex ($V_1 - V_T$) [Region of the brain
that receives, integrates and
processes visual information relayed
from the retina].

$\rightarrow V_1 \rightarrow$ primary visual cortex [Orientation of
 edges and lines].
 V_2 [Difference in color, complex patterns,
 $V_3 \quad V_4 \quad V_5$ Object Orientation].
 [Object Recognition].

visualize the image

RGB Images and Grayscale Images



Stride = 1 \Rightarrow jump.

0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1

$$m=6$$

Vertical edge filter

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

$$3 \times 3$$

$$f=5$$

filter size

4x4

Input

$$\begin{aligned} & 1 \times 0 + 0 \times 0 + (-1) * 0 \\ & + 2 \times 0 + 0 \times 0 + (-1) * 0 = 0 \\ & + 1 \times 0 + 0 \times 0 + (-1) * 0 \end{aligned}$$

Jump = 1

Image

If size = 6 $\left\{ \begin{array}{l} \text{loss of information} \\ \text{out size = 4} \end{array} \right.$

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

+

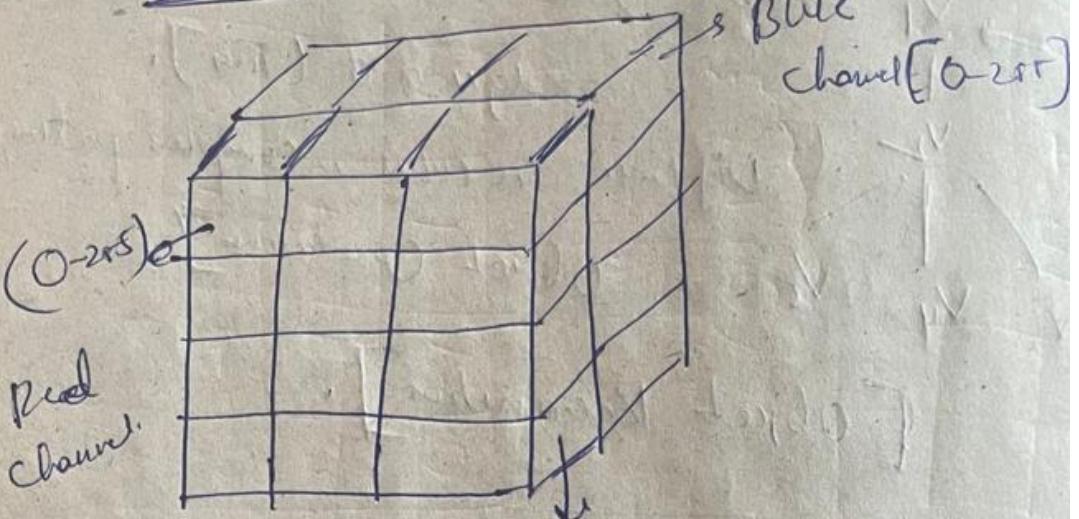
Output

formula:

$$n-f+1$$

$$0 + 0 - 1 + 0 + 0 - 2 + 0 + 0 = -4$$

for RGB



Green
chan [0-255]

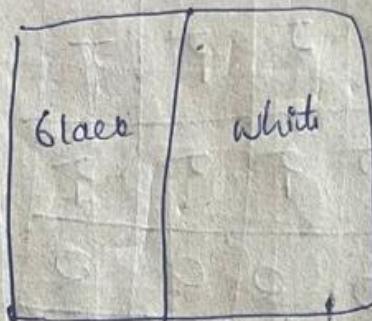
Channels = 3

$4 \times 4 \times 3$

Convolution Operation In CNN

0 → black
255 → white

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



$6 \times 6 \times 1$
or
grayscale

Step 1

- Normalize the image
- Convolve all these pixels (0 to 1)
- by dividing all pixels by 255

↓
2 filter multiplying

0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1

Stride = 1

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

3×3

filters.
Can be of any size.

to prevent loss of info
as we use padding

filters are used applied on

the images to find edges.

Vertical edge filter

$$\begin{array}{|c|c|c|} \hline +1 & 0 & -1 \\ \hline +2 & 0 & -2 \\ \hline +1 & 0 & -1 \\ \hline \end{array}$$

find ~~horizontal~~
vertical edges.

Horizontal edge filter

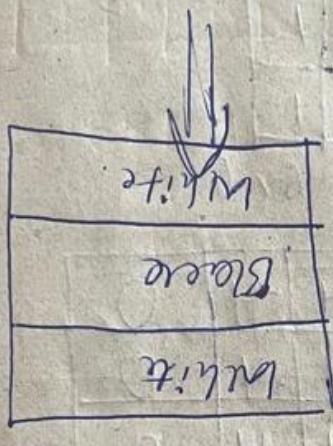
$$\begin{array}{|c|c|c|} \hline +1 & +2 & +1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

thus ~~filter~~ are already predefined

we are able to find diff types
of ~~edges~~, vertical, horizontal,
diagonal etc...

Denormalization

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

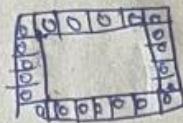


usually in CNN we give random values in
filters. wrt to forward and backward propagation
they are going to be updated.

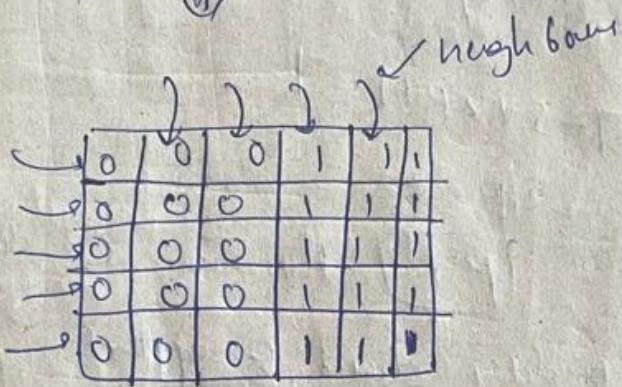
Convolution Operation in CNN

values used in padding

① zero padding.



② Neighbor padding



padding formula $n-f+2pt+1$

to know how much padding should I

do $6-3+2p+1 \Rightarrow 6$

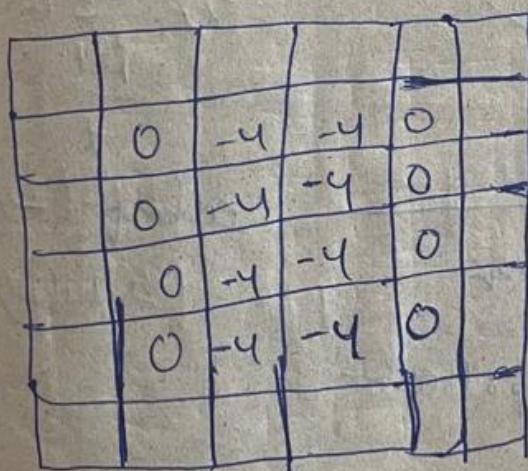
$- 3+2p+1=6$

$2p=2$

$p=1$

let say if
want output
as 6×6

y
 $m=6$



~~4x4~~ 4×4
 4×4
 6×6

CNN

$v_1 \rightarrow$ find edges
 \downarrow
 $v_2 \rightarrow$ find corner like
 $v_3 \text{ and } v_5 \rightarrow$ irregular patterns, 0 or 1

ReLU activation function

on its result

0	0	0	1	1	1
0	0	1	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1



randomly initialize



f_1

8	-4	-4	16
6	-4	-4	0
0	-4	-4	0
0	-4	-4	0

select max

f_2

randomly initialize



f_3

With the help of forward & backward propagation
we try to update this filter value

based on number of filters we add, we are
going to get that much of output

asm

$$n=7 \quad f=3 \quad o/p: 7$$

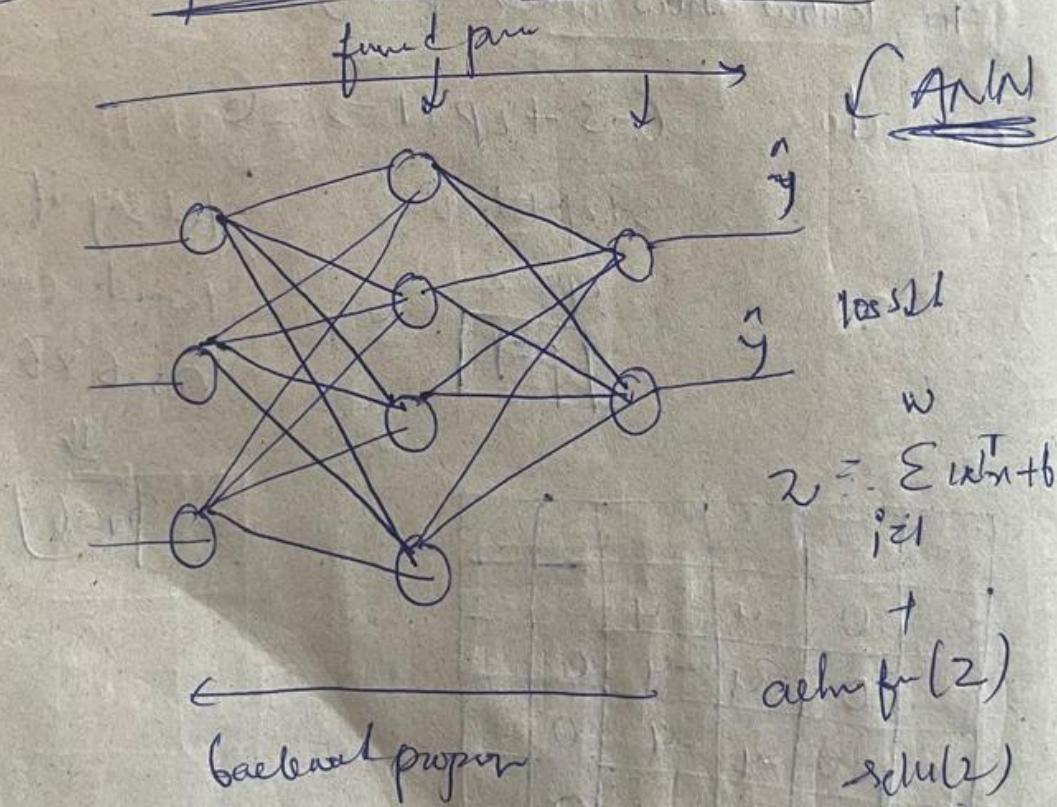
$$n-f+2p+1=7$$

$$7-3+2p+1=7$$

$$2p=2$$

TPY thus mean one layer of padding should be applied

Opuations of CNN vs ANN

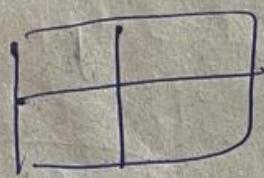
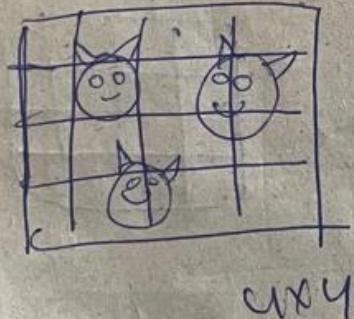


with the help of op.m

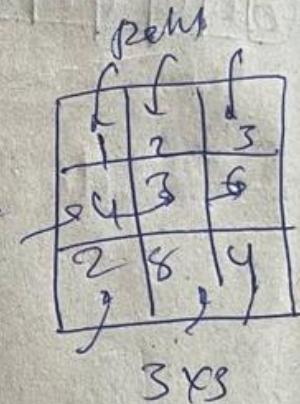
Max pooling, Mm pooling, Average/Mean
pooling

Single convolution layer

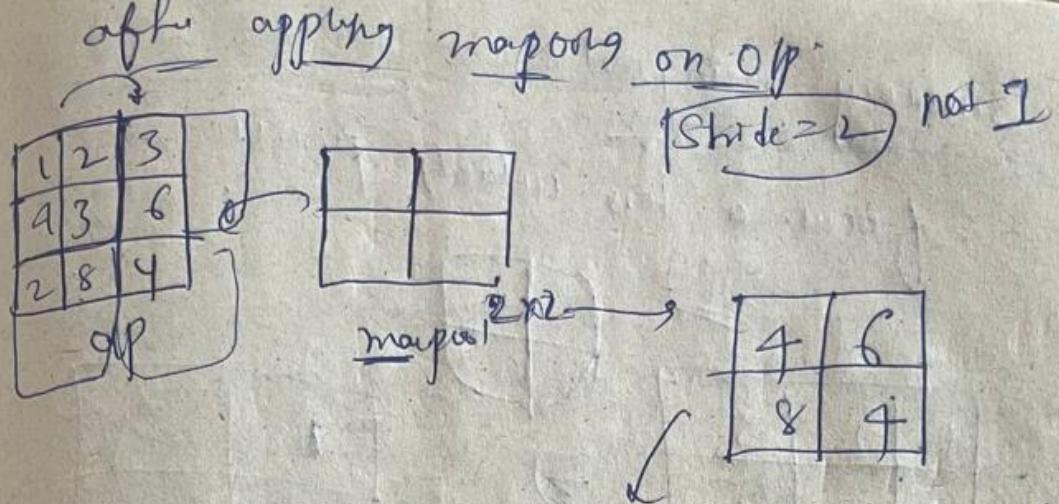
filter is applied on image → We get the output then activation function is applied on top of it



$p=1$ fil
 $s=1$

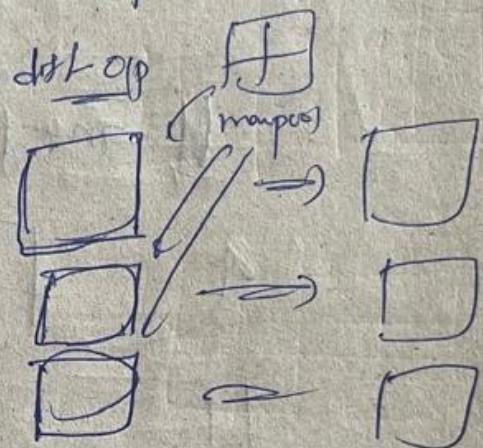
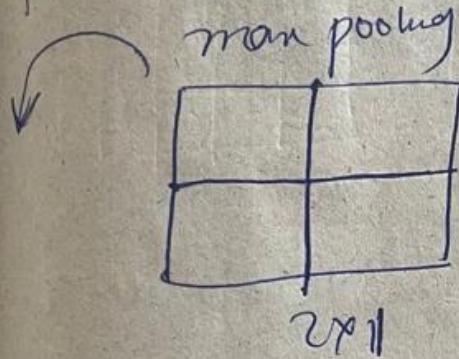


Convolution layers
((convolution layer + relu function))



bright intensity as most imp feature we are going to capture here

applied on top of O/P layer



0 Implication of what follows we are, if we apply max pooling on top of O/P layer if given clear images (like say cat)

Location invariant: When are those specific shape of images are there, max pool should capture that

Summary min pool

Focus on less intensity pixels.

Shdr

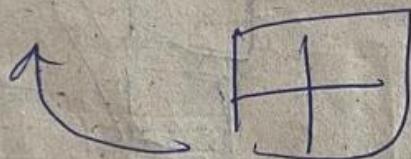
1	2	3
4	3	6
2	8	9



→
minpool

1		3
2		4

↓ minpool



2.5	4.5
5	4

brighter (more intensity)

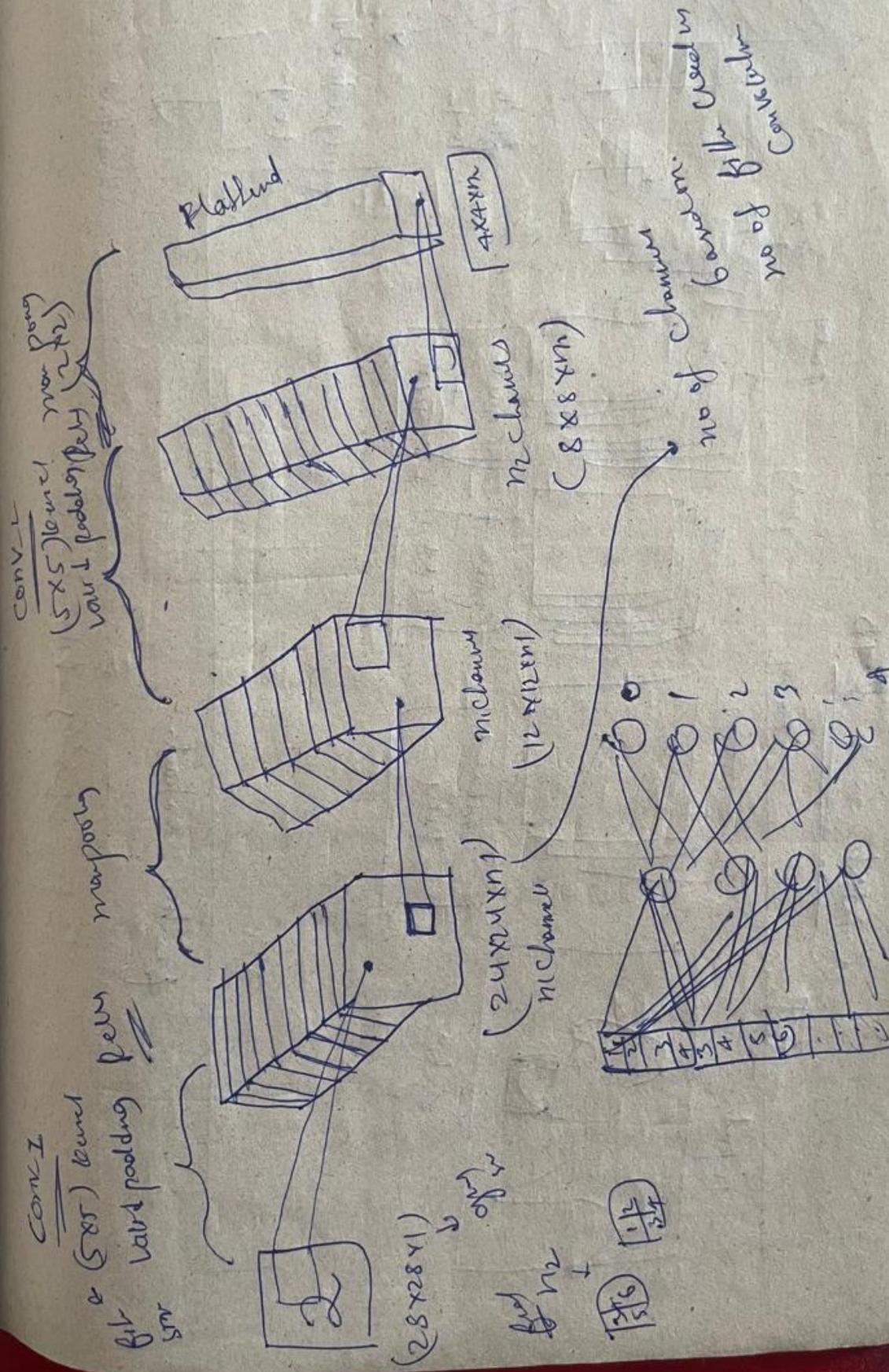
Overall

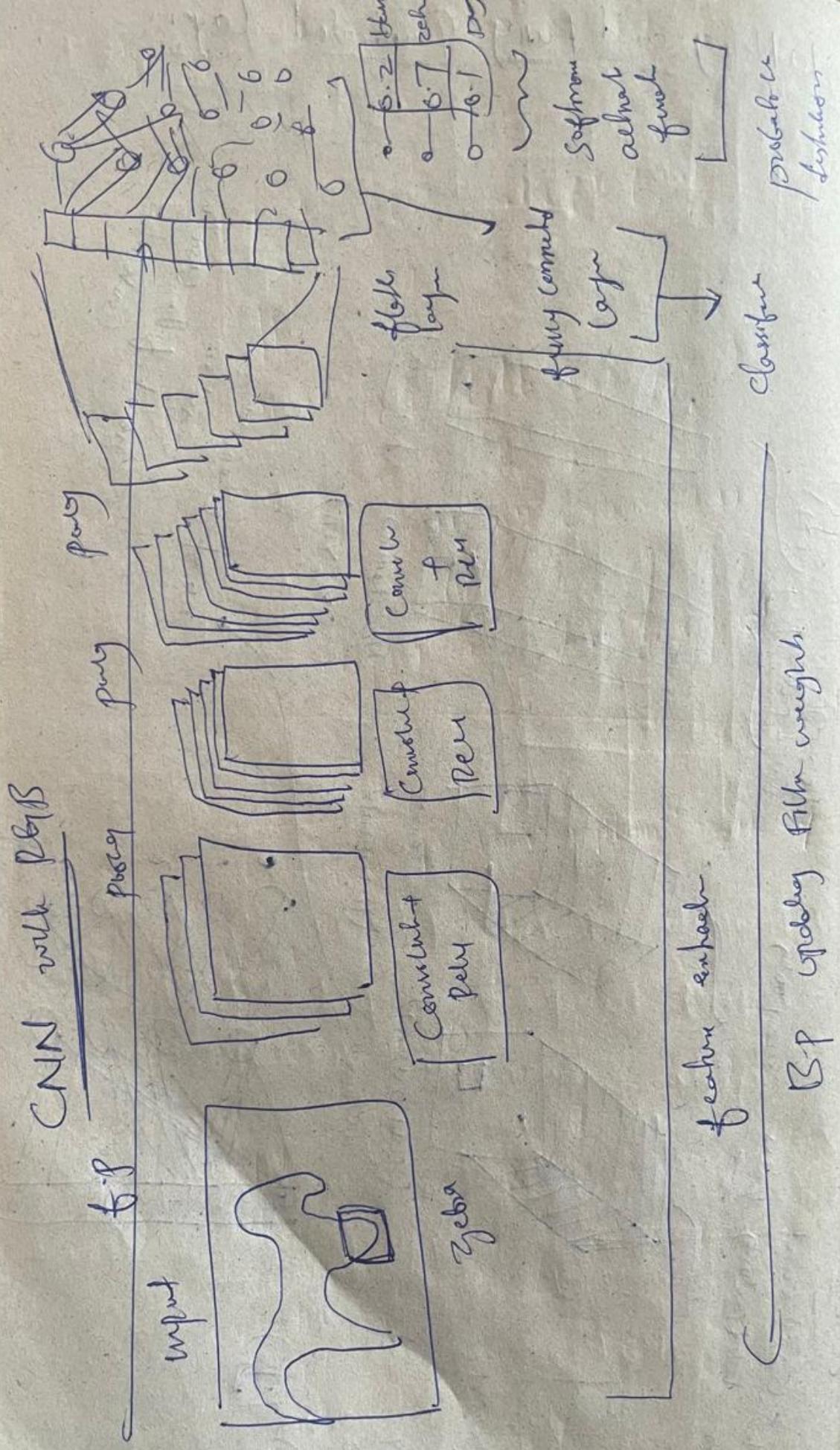
max pooling: Select highest pixel value in each region.

min pooling: Selects the lowest pixel value in each region.

average pooling: Calculate avg pixel value by Smoothing out details.

Flattening and Fully Connected Layer





after model is trained

1) Convert

pickle file .h5 file found

2) STREAMLIT → to Create Web app

Integrate all Our models

3) Deploy in Streamlit Cloud

VS Code Setup

① Conda Create -P venv python==3.11.4

② requirements.txt

③ tensorflow ==2.15.0

④ pandas

⑤ numpy

⑥ scikit-learn

⑦ tensorboard

⑧ matplotlib

⑨ Streamlit

⑩ Seaborn

END-TO-END- Deep Learning project using ANN: (tensorflow and keras)

Keras is like a lakappa on top of tensorflow
which is simple, flexible and powerful

CHURN MODELING
DATASET

exit the bank @
not

Classification \Rightarrow Basic FE



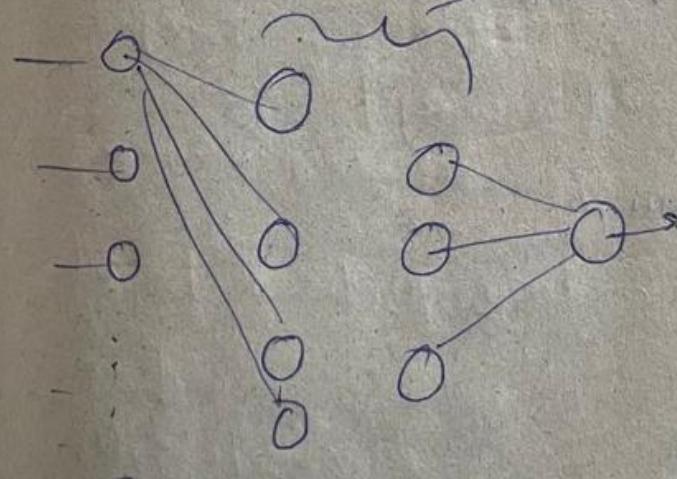
\rightarrow Convert Categorical

Var into Num

\rightarrow Standardization

11 input features

⑤ no of hidden layers
options



how to
apply dropout

(1) input nodes

& Conda activate venv1

→ pip install & requirements.txt.

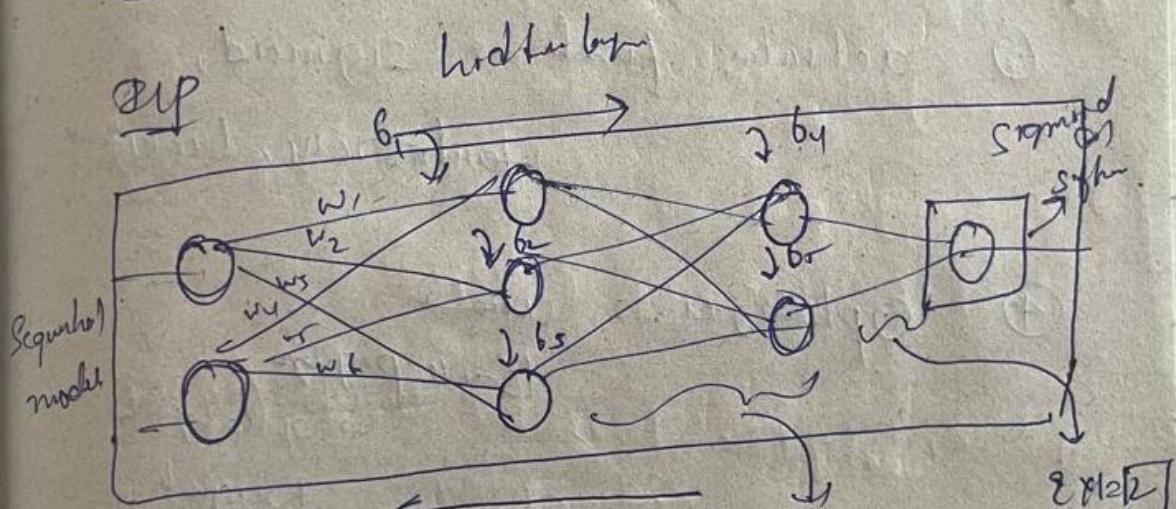
→ pip install ipykernel

→ done some feature engineering

→ Created few pickle files

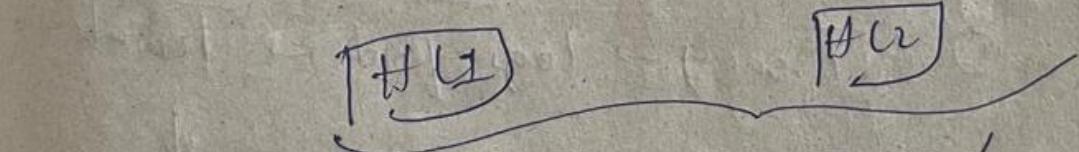
Applying ANN using TensorFlow

ANN → Sequential model
H1 H2 O/P



$$2 \times 3 = 6 \text{ weights. } 3 \times 2 = 6 \text{ weights. } 16 \text{ total.}$$

3 bias. 2 bias.



Trainable parameters = 20 parameters due to f.p. and b.f.

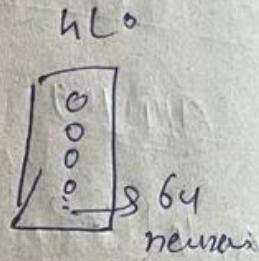
no of input nodes ~~decides~~ in ANN
is given by no of input features
in the data set

Steps in Creating ANN

① Sequential N/W (initialize)

② To create hidden neuron

Dense = 64



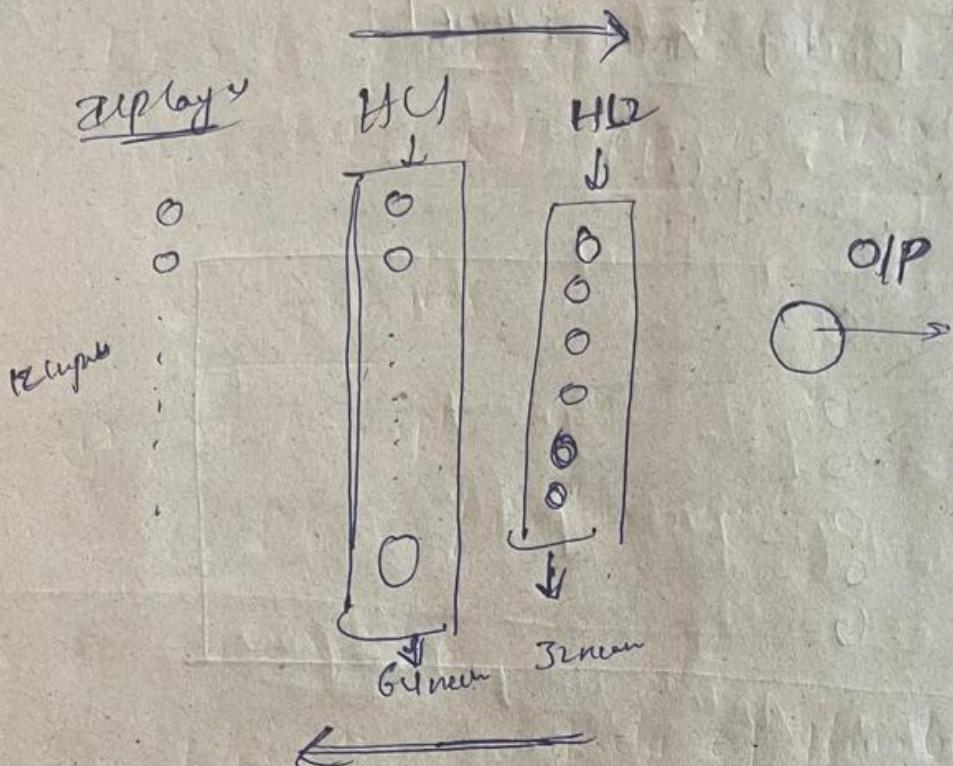
③ activation function \rightarrow sigmoid,
tanh, relu, LeakyRelu,

④ Optimizers \rightarrow Adam
Back propagation
 \hookrightarrow gradient descent

⑤ Metrics \rightarrow [accuracy], [auc]
[max, min]

⑥ Training \rightarrow loop - fold \rightarrow Tensorboard

\downarrow
some visualizations



- ② Which optmize and loss function going to use
- ③ Comprised the model ~~with~~ by assigning the above opt + loss

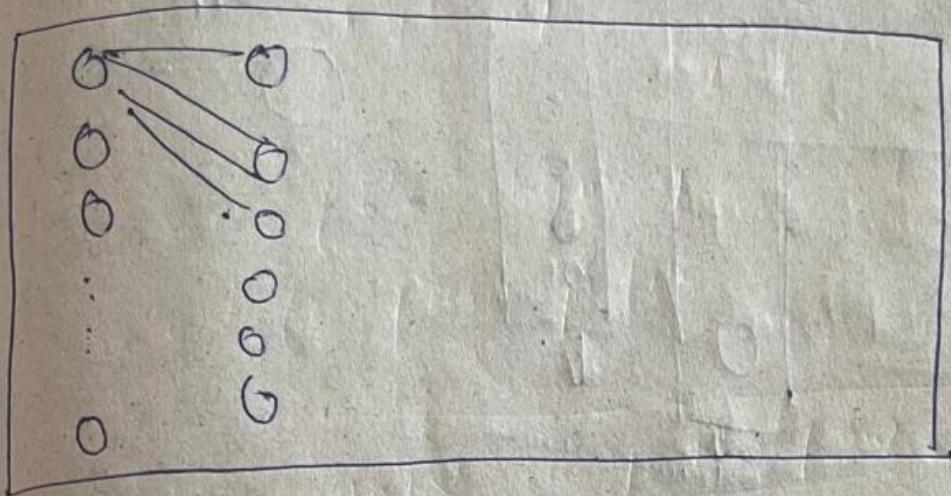
④ ~~Tensorflow~~ Tensorboard setup for logs

⑤ Early stopping callback

Q Suppose you are assigned 100 epochs, but after 10 epochs, there is no much change in loss function, then we are going to restart one then

For Building ANN MODEL

(x-ham-shape (ij))



n ham-shape

y

where

Code

model. survey (j)

How to find optimal hidden layers and hidden Neuron in ANN

Some guidelines

- ① Start simple
- ② Grid Search / Random Search
- ③ Cross validation
- ④ Heuristics and Rule of Thumb.

↳ These provide starting point such as

↳ no. of neurons in the hidden layer

Should be b/w size of the input layer
and size of the output layer.

↳ a common practice to start with
1-2 hidden layers

Keras is used for hyperparametering

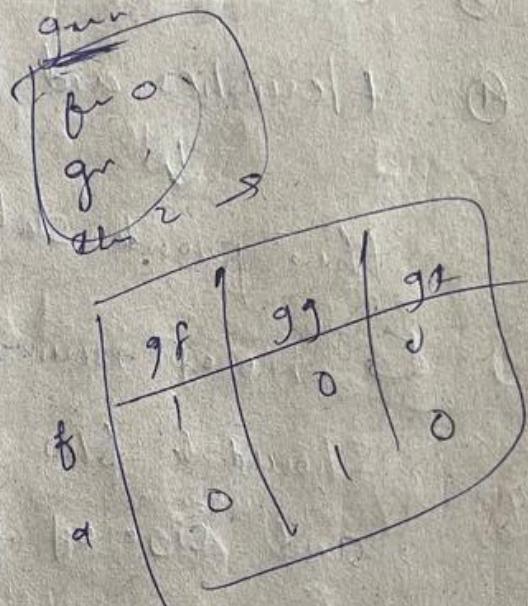
the ANN Model

Keras Commands

⑥ Trained the model

Predicting with Trained ANN MODELS

- Load the model
- Load the pickle file which you have created.



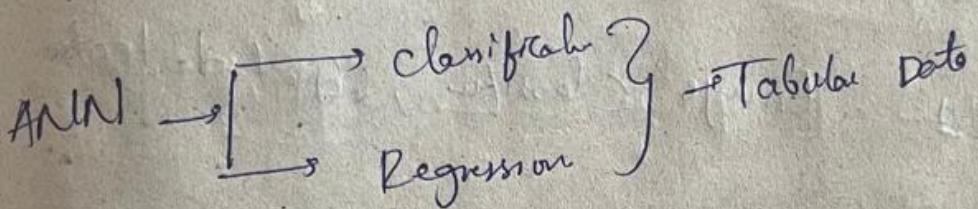
Integrating ANN model with streamlit app.

NLP in deep learning

Text data \rightarrow Vectors \rightarrow Numerical Representations

- ① OHE
- ② BOW
- ③ TF-IDF

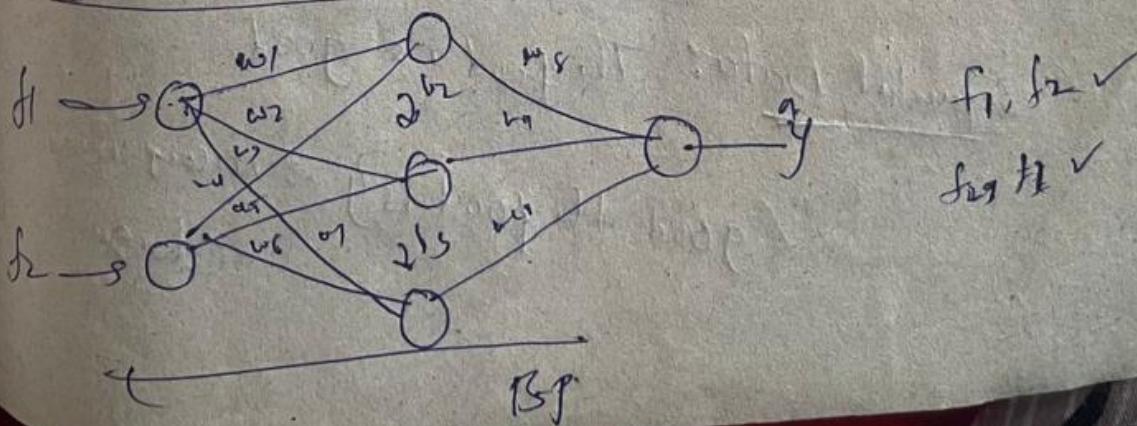
④ word2vec, Avg word2vec
L Sentiment analysis, Text (Cluster)



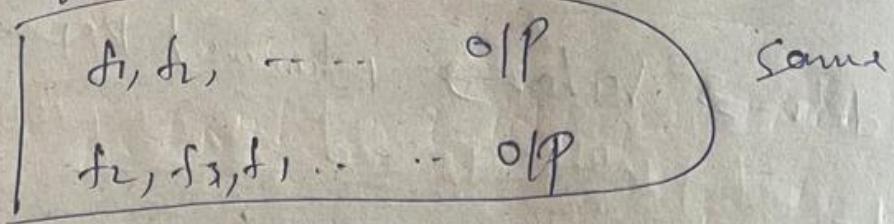
e.g. House price prediction
 ↘ continuous ↗ categorical

House size	No of rooms	price region	class
2 ^{b1}	2 ^{b2}	2 ^{b3}	2 ^{b4}

cost($y - \hat{y}$) add



→ Sequence of data doesn't matter



②

CNN

Data
↓

input data → Images, Video frames

e.g.: Image classifier, Object detector

③

Data is sequential type data.

④

Tell grammar → → jule.

SLP

This is a apple —

Sequence

⑤

Chatbot Conversation → Q&A Q w/ +
answer

Sequential Data: The food is good.

L good the food is y → means may
change

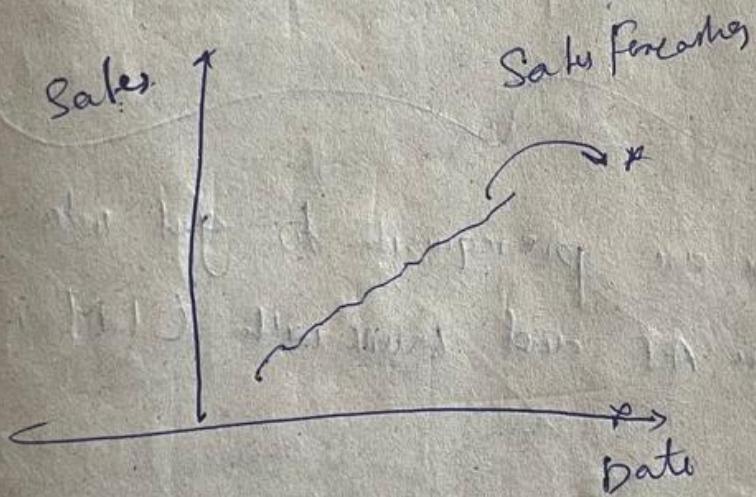
⑥ Language translation →



⑦ auto suggestion

(LinkedIn, Gmail) ...

⑧ Sales data: → Data



Can we use ANN to solve the problem?

which has Sequential data

things to learn NLP in deep learning:

① Simple RNN → LSTM & GRU RNN →

Pearl date: Sequence informed is important

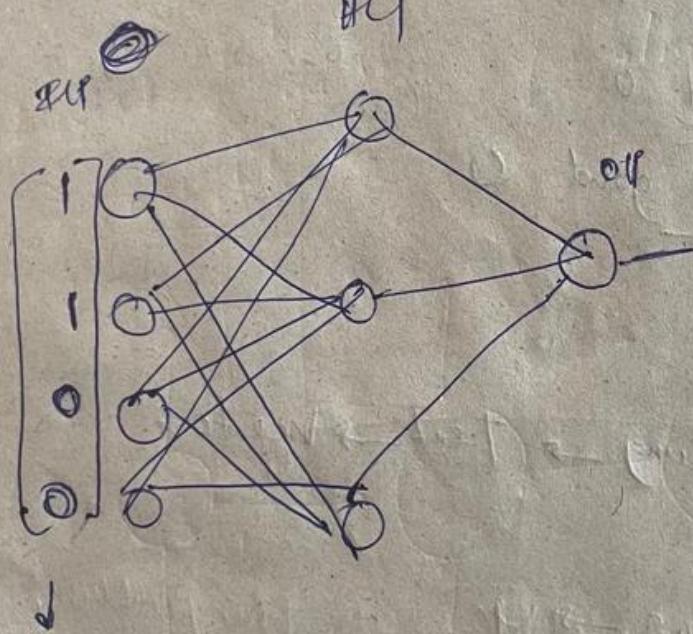
When we use this point from ANN

meaning of the Sequence is lost

[any [Bom, TDF, B. which will]]

input sign

* Sequence info is lost



The food is good

$t=1, t=2, t=3,$

$t=4$

we should
give one word
at a time but
by using ANN

we can give 100s
word at a time.

• Input is given at a time

by this sequence information is also maintained.

(ANN)

google him

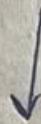
Engg → Book

① hi ② book

① - - ② book

Thing to learn NLP in deep learning

① Single RNN \rightarrow LSTM / GRU RNN



Bidirectional RNN



Encoder Decoder



Transformer \leftarrow Self Attention

then are prequisite to get into
generative AI and working with LLM models

Can ink some using with ANN

→ sequential data

RNN

Dataset L Sentiment Analysis

Text

O/P

s1 The food is good 1

s2 The food is bad 0

s3 The food is not good 0

ANN

Text preparing → Text → Vectors

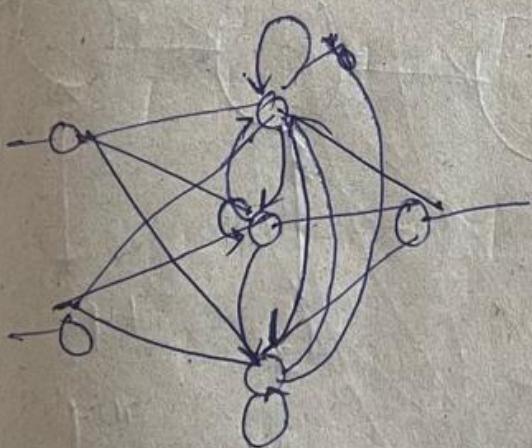
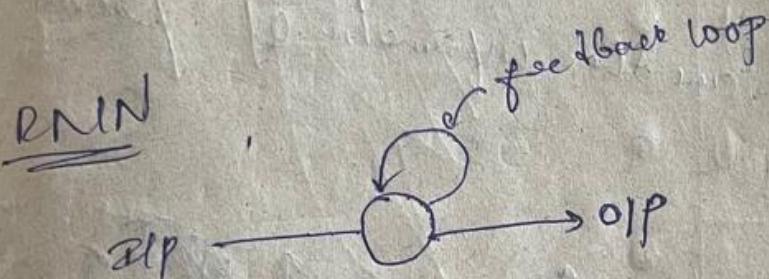
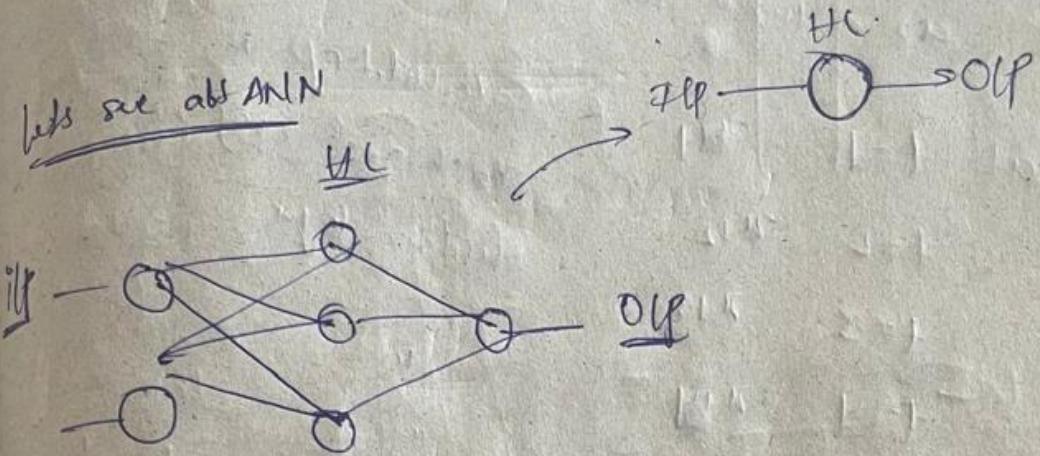
Vocab size → 4

	Bad	good	bad	not	
s1	0	1	0	0	8 firms water
s2	1	0	1	0	8 firms water
s3	0	0	0	1	

So that is when we will use Single RNN

Recurrent Neural Network:

→ so our data → Sequential



whenever OLP means
getting that is going to
start with an hidden
neuron including itself

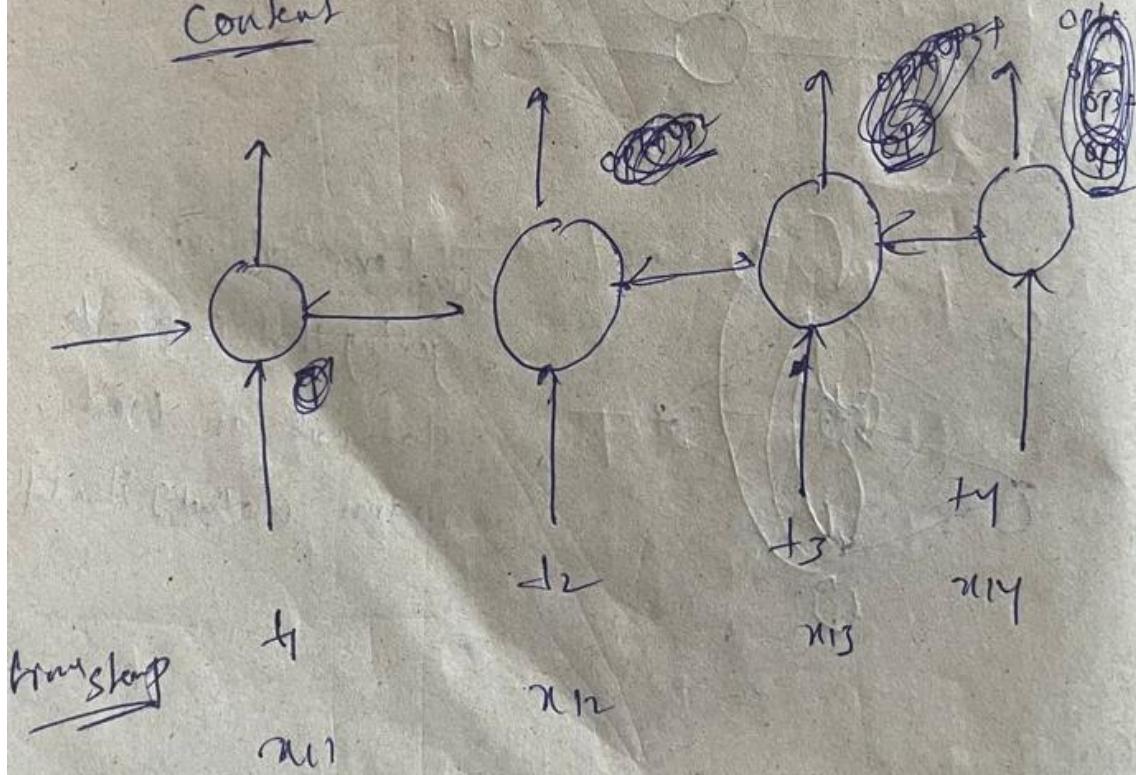
if
 $x_{11} \ x_{12} \ x_{13} \ x_{14}$
The food is good.

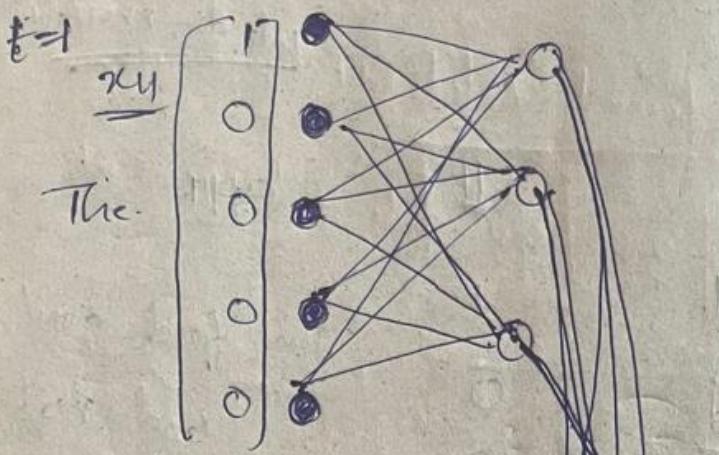
Based on 4 times lamp, single word is given
as a input at a time

at $t=1$ x_{11} t Calculat OP is should will
 $t=2$ x_{12} each and every never with
 $t=3$ x_{13} hidden layer
 $t=4$ x_{14} "

RNN refine the information of previous

Context

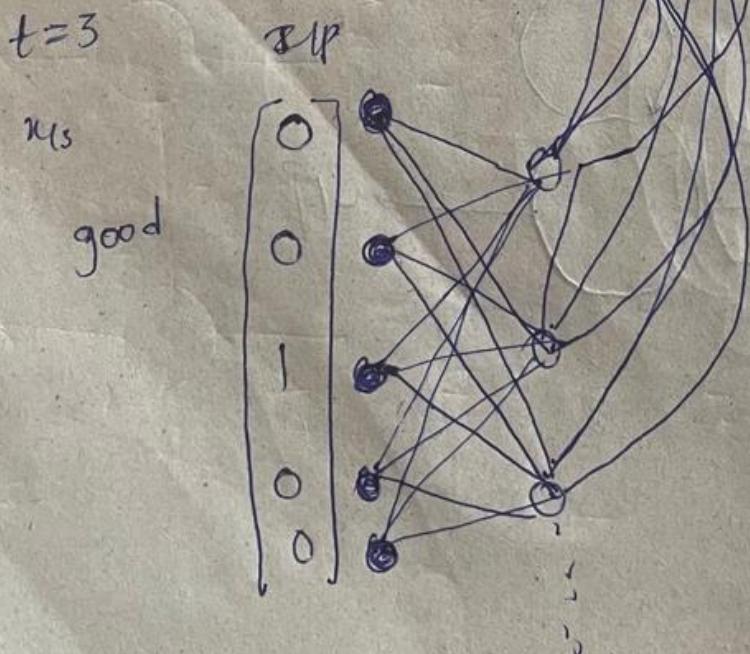
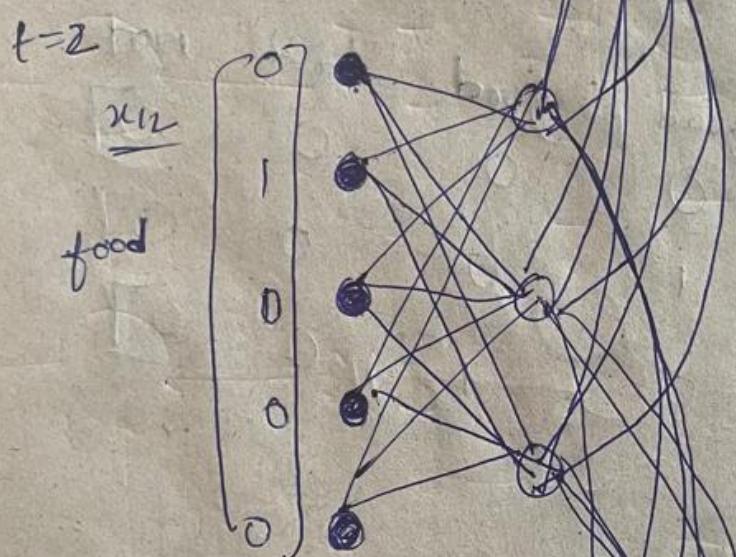




This cellular
RNN has
Content/Info
of all previous
words
from

time step
 t_1 to
 t_n .

RNN



Forward propagation with time in RNN

Datalist

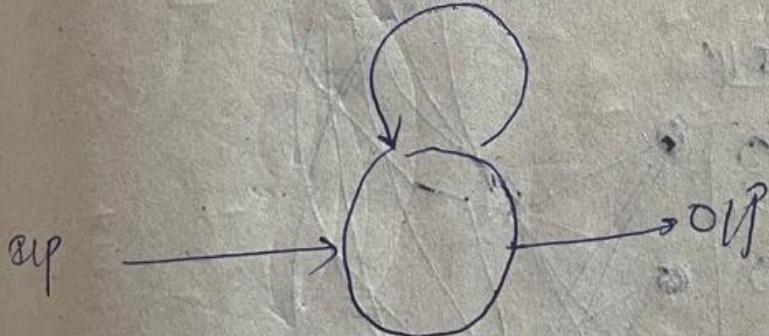
- text
- x₁ x₂ x₃
1. The food is good.
2. The food is bad
3. The food is not good

o/p

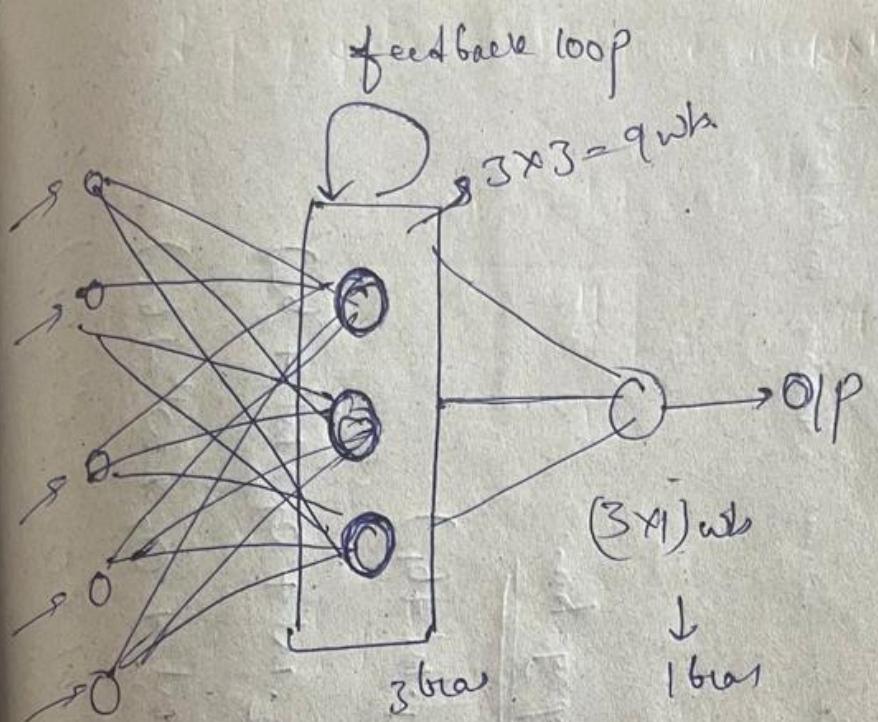
o/t/e

The food good
bad not

	The	food	good	bad	not
the	[1]	0	0	0	0]
food	[0]	1	0	0	0]
good	[0]	0	1	0	0]



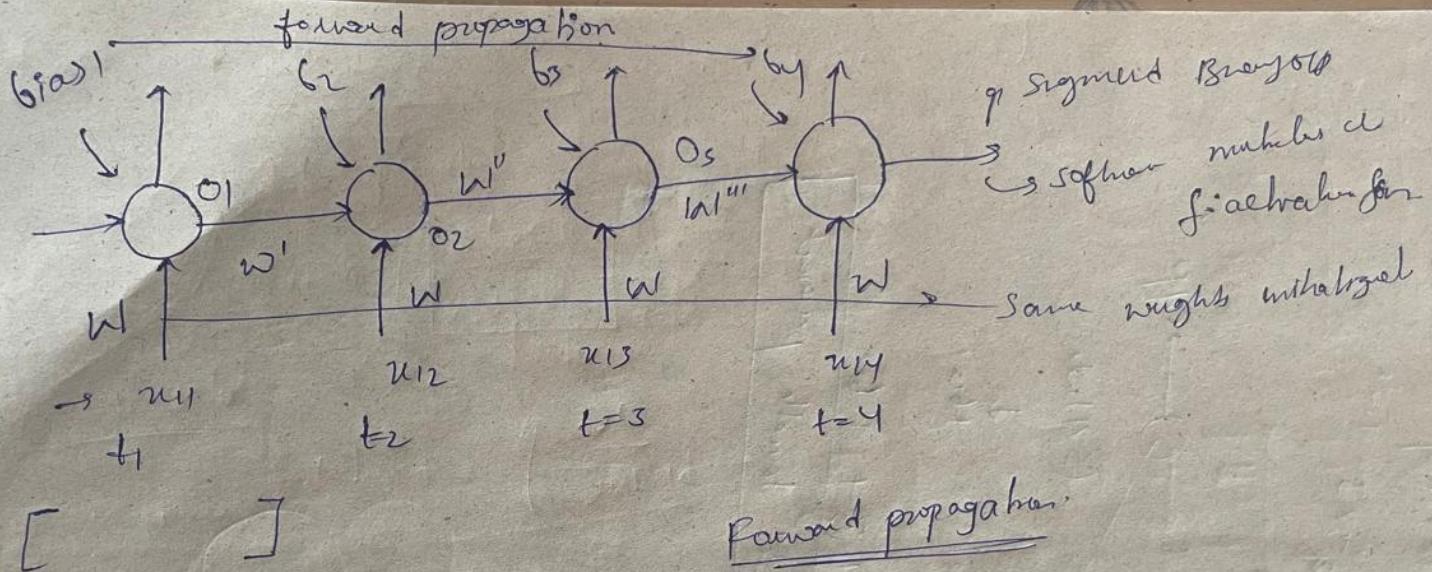
Words & vectors



(5×3)

$$= 15 \text{ weeks} + 9 + 3 + 4 \text{ bras}$$

= 31 no. of total trainable parameters



Dataset:

The foot is good

$x_{11} \quad x_{12} \quad x_{13} \quad x_{14}$

O/P

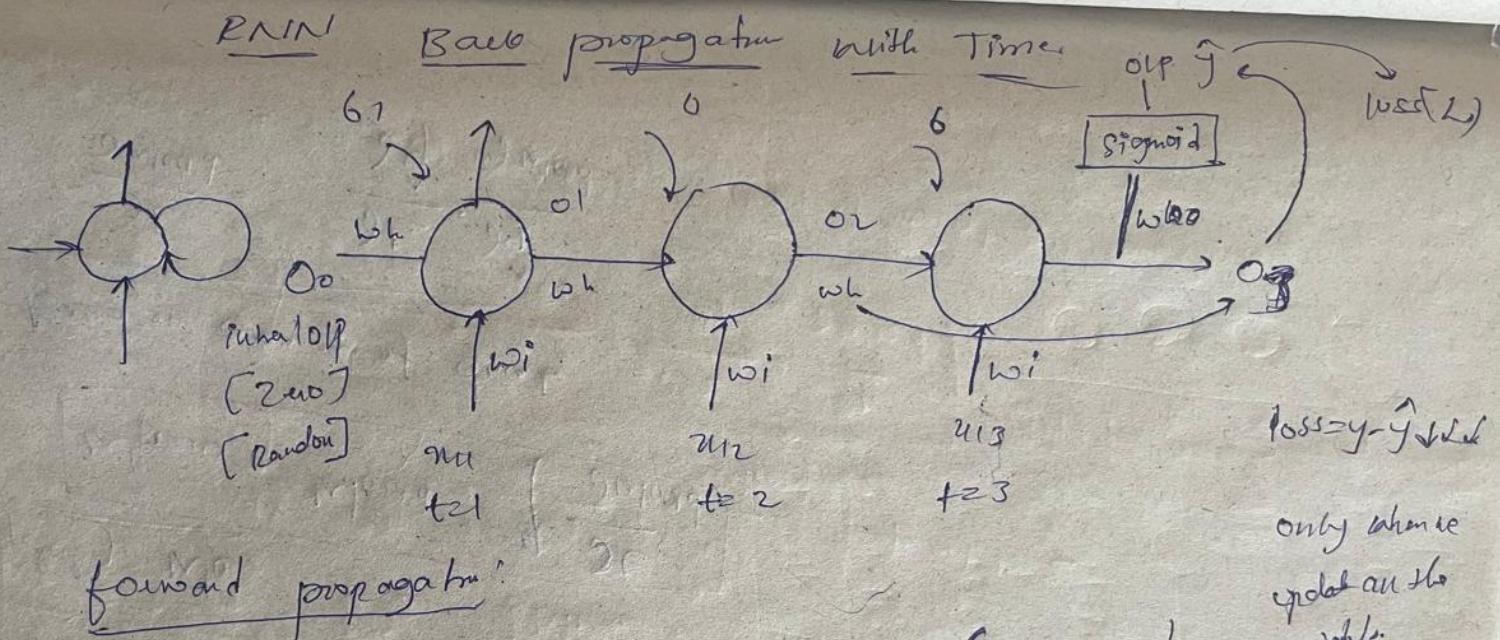
1

$$o_1 = f(u_{11} \cdot w + b_1)$$

$$o_2 = f(u_{12} \cdot w + o_1 \cdot w' + b_2)$$

$$o_3 = f(u_{13} \cdot w + o_2 \cdot w'' + b_3)$$

$$o_4 = f(u_{14} \cdot w + o_3 \cdot w''' + b_4) \text{ sign}$$



$$o_1 = f(u_{11} \cdot w_i + o_0 \cdot w_h + b_1)$$

$$\hat{y} = \sigma(o_3 \cdot w_o)$$

$[w_i, w_h, w_o]$

$$o_2 = f(u_{12} \cdot w_i + o_1 \cdot w_h + b_2)$$

during back propagation

$$o_3 = f(u_{13} \cdot w_i + o_2 \cdot w_h + b_3)$$

Back propagation with line update $[w_i, w_h, w]$

enlight update formula

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

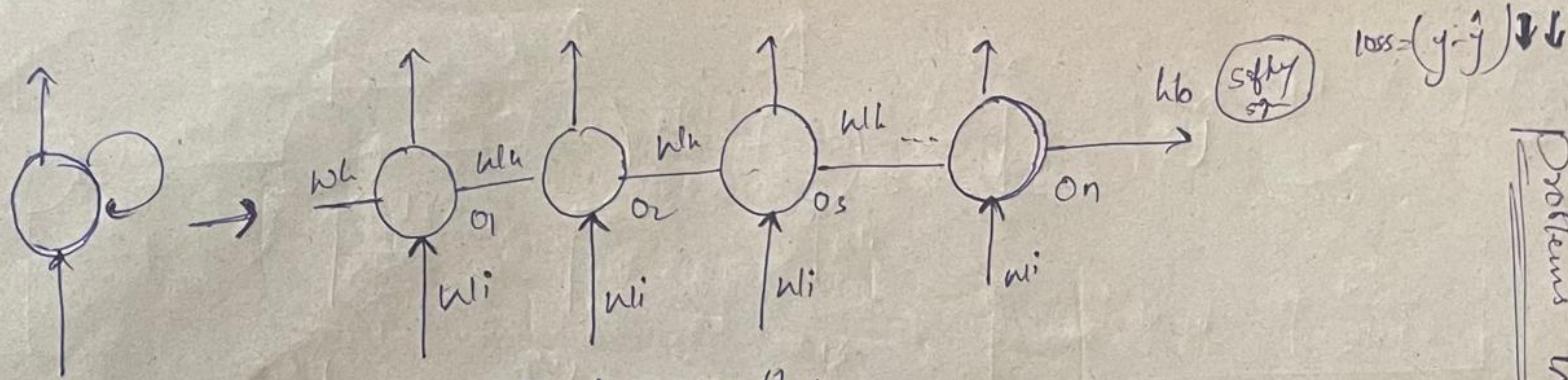
Derivative, slope of GRADIENT DESCENT

① update w_0

$$w_{new} = w_{old} - \eta \boxed{\frac{\partial L}{\partial w_{old,0}}}$$

Based on change of derivative

$$\frac{\partial L}{\partial w_{old,0}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{old}}$$



ANNL → vanishing gradient problem

Tent
The food is good
The food is bad

0
1
0

Tent germain has short term dependence.
 $s_t = I \text{ like to play}$

for $t=2$ as $t=4$ $t=5$
 My name is Brahma and I like sports
 like Cricket, volleyball, and also like

LSTM RNN, GRU RNN

So, if values are used

Problems with RNN

to make
 $t=16$ \rightarrow food
 videos.

Sentence is long

↓ which may have two words

long term dependencies cannot be captured by RNN, hence it
 cannot provide good accuracy w.r.t. this kind of dependency

② Updating w_{lh} [hidden layer weight] \rightarrow Time Stamps $t=1, 2, 3$

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

$$\frac{\partial L}{\partial w_{lh\text{ old}}} = \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial w_{lh}} \right] + \alpha t=3$$

$$\left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial o_2} * \frac{\partial o_2}{\partial w_{lh}} \right] + \alpha t=2$$

$$\left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial o_2} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{lh}} \right]$$

all hidden layers

This is how the weights are going to update in backpropagation

③ Updating w_i

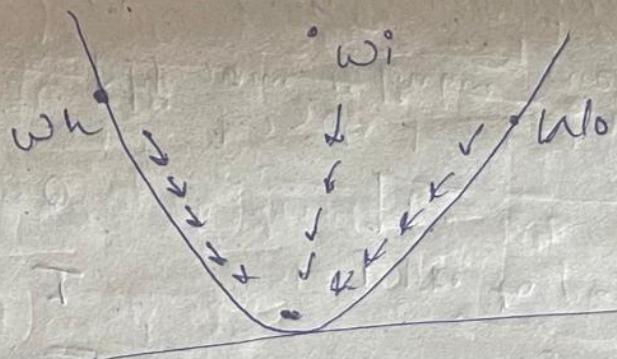
$$w_{linew} = w_{lold} - \eta \left[\frac{\partial L}{\partial w_{lold}} \right]$$

$$\frac{\partial L}{\partial w_{lold}} = \left[\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_3} \times \frac{\partial o_3}{\partial w_{lold}} \right] + \text{at } t=3$$

$$\left[\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_3} \times \frac{\partial o_3}{\partial o_2} \times \frac{\partial o_2}{\partial w_{lold}} \right] + \text{at } t=2$$

$$\left[\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_3} \times \frac{\partial o_3}{\partial o_2} \times \frac{\partial o_2}{\partial o_1} + \frac{\partial o_1}{\partial w_{lold}} \right] \text{ at } t=3$$

↳ Updates all w_i weights in backpropagation at all time stamps.

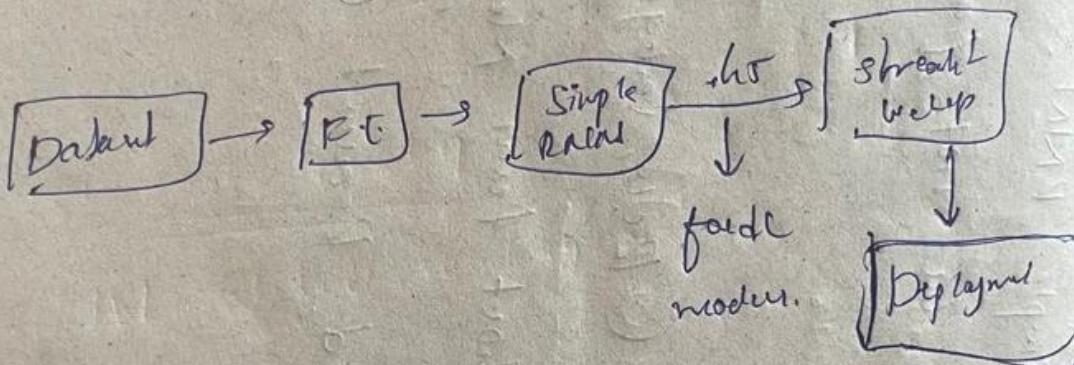


End-to-end deep learning project using Simple RNN

IMDb Dataset [Movie Reviews Dataset]

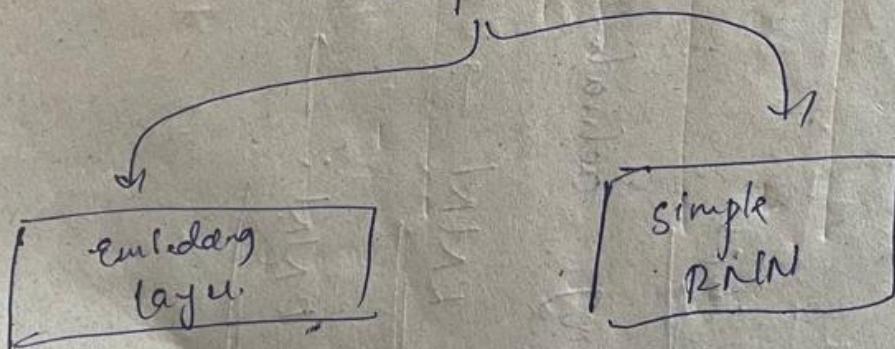
Tent OIP
Reviews pos / Neg

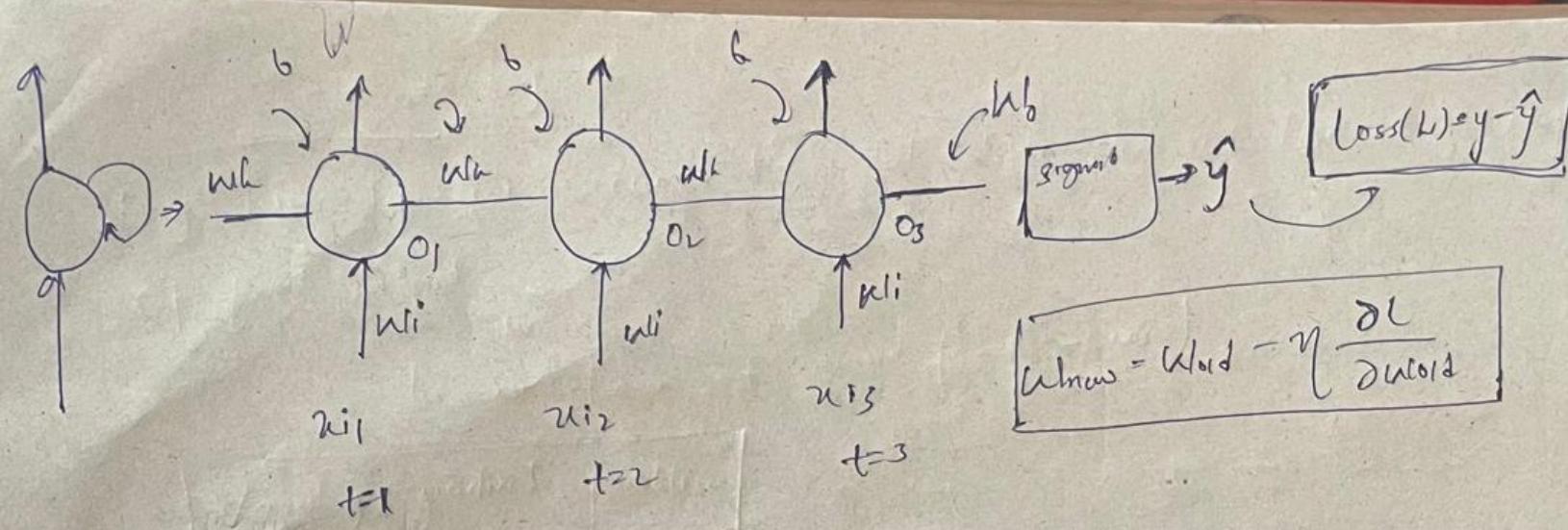
google colab \Rightarrow Create the model
on free GPU's



major components in

Simple RNN





$$\frac{\partial L}{\partial w_{31}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{31}}$$

Now let's calculate update weight for w_{31}

$$\frac{\partial L}{\partial w_{31}} = \left[\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_{31}} \right] + \left[\dots \right]$$

Length of sentence \Rightarrow 50 words.

$$\frac{\partial L}{\partial w_{31}} = \left[\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_{31}} \right] + \left[\dots \right]$$

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial (\sigma(u_{31} + w_{31} \cdot o_2 + b)))}{\partial o_2} = o_2$$

Dynaline of sigmoid = [0-0.25]

That means at $t=1$ the word is not participating in weight update.

When chain is long:

long term dependency occurs, the initial words do not have much impact of final output.

because derivative of sigmoid fun ($0-0.25$)

Vanishing gradient problem

at $t=50$ (small chain)

$$\frac{\partial L}{\partial \text{hidden}} = \left[\frac{\partial L}{\partial \hat{y}}, \frac{\partial \hat{y}}{\partial o_{50}}, * \frac{\partial o_{50}}{\partial \text{hidden}} \right]$$

≈ 0

$t=49 > t=48$

$t=47 > t=46$

$t=45 > t=44$

$t=43 > t=42$

$t=41 > t=40$

$t=39 > t=38$

$t=37 > t=36$

$t=35 > t=34$

$t=33 > t=32$

$t=31 > t=30$

$t=29 > t=28$

$t=27 > t=26$

$t=25 > t=24$

$t=23 > t=22$

$t=21 > t=20$

$t=19 > t=18$

$t=17 > t=16$

$t=15 > t=14$

$t=13 > t=12$

$t=11 > t=10$

$t=9 > t=8$

$t=7 > t=6$

$t=5 > t=4$

$t=3 > t=2$

$t=1 > t=0$

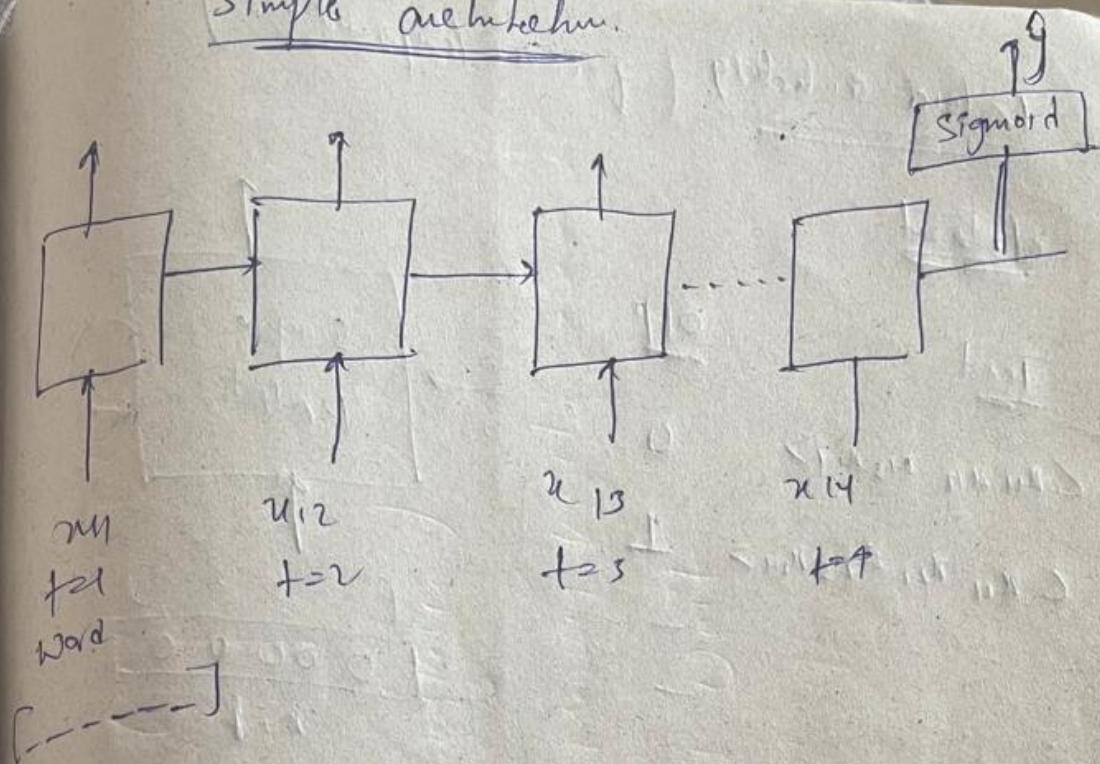
Major

This is the problem with Simple RNN. if it is having
dependency with nearest words, but not farest words.

How to solve this

- by changing activation function (Relu, tanh, ReLU...)
- LSTM RNN → Long short term Memory RNN
- GRU RNN: → Gated Recurrent Unit

Simple architecture.



Tanh

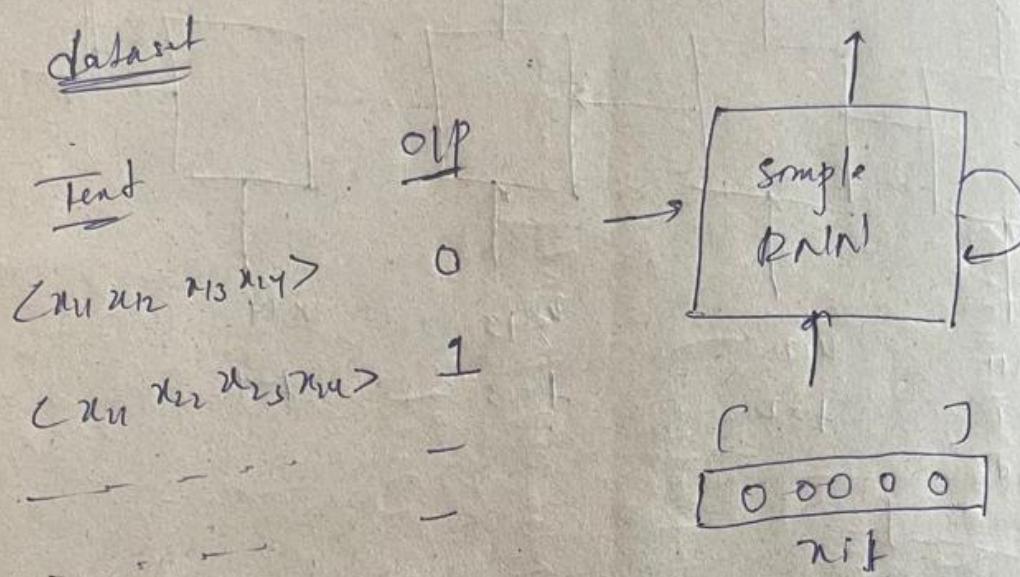
$(u_{11}, u_{12}, u_{13}, u_{14})$

words \rightarrow vector

embedding layer is responsible for conveg.

how does embedding layer works

Word embedding [feature Representation]



One hot encoding

$|V| = 10,000$ vocabulary size

One hot representation

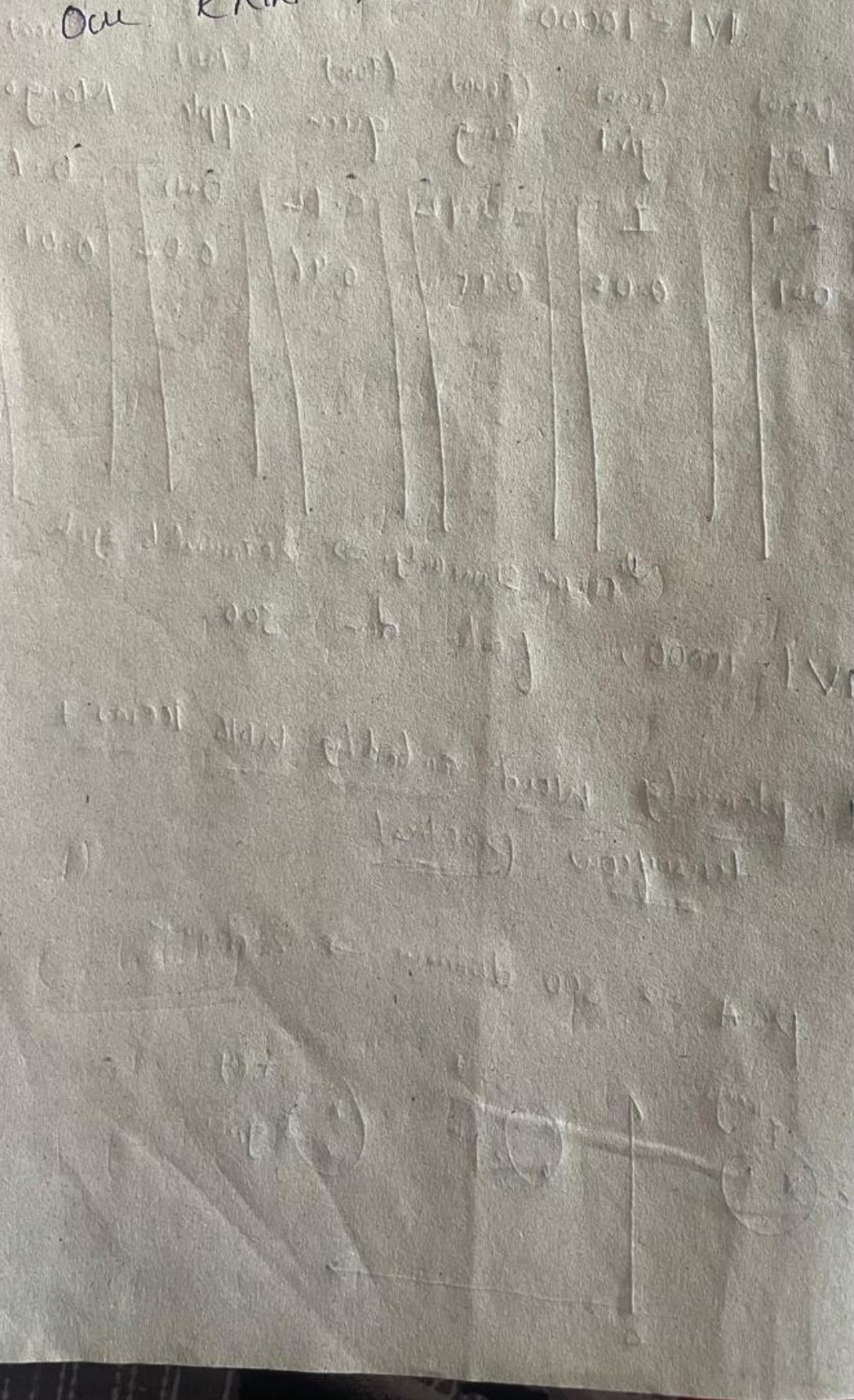
$$\text{Man} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}_{1 \times 10000}$$

$$\text{Boy} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}_{1 \times 10000}^{82000}$$

sparse man \Rightarrow overflow.

+ To overcome this we use word embeddings

We need to make all the sentences of equal
size, otherwise we are not able to have
one RNN model.



Word embedding

Word2vec is one type of embedding layer

$$|V| = 10000$$

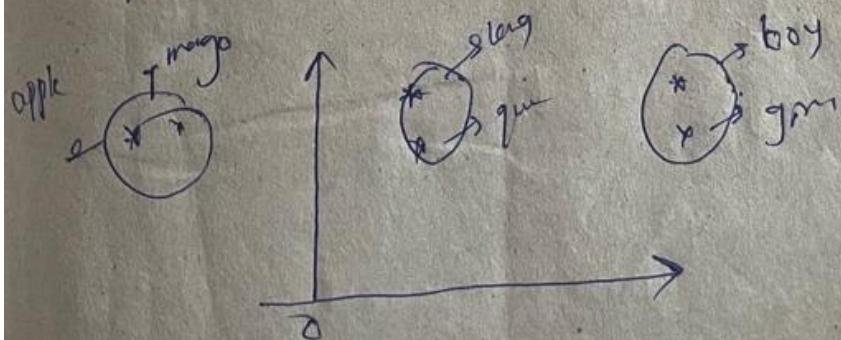
(2000)	(8000)	(6000)	(9000)	(1000)	(7000)
Boy	girl	king	queen	apple	Mango
-1	1	-0.92	0.92	0.0	0.1
0.01	0.03	0.95	0.96	-0.02	0.01
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-

↳ cosine similarity \Rightarrow recommend system

$$|V| = 10000, \text{ featur_dim} = 300$$

Implementation word embedding with keras +
Tensorflow practical

PCA \rightarrow 300 dimens \rightarrow 2 dimens



End-to-End Deep learning Project using

Simple RNN

- Impactbrane
- word mat for tensorflow.

few input things to learn:

① pad_sequences [ff.keras.utils.pad_sequences]

↳ ensure they have the same length. This is important because neural networks require inputs of consistent dimensions.

parameters:

① sequences (list of lists): $\rightarrow [(1, 2), (3, 4, 5), (6)]$

② maxlen : max length of sequences after padding.

shorter seq will be padded, and longer ones will be truncated.

③ padding: pre, post

④ value: the value to use ~~as~~ for padding.

Default is '0'

Return type: 2D numpy array with padded sequence

② Sequential: [tf. bias. modus, sequent]

a linear stack of layers in keras, where layers
are added one by one in order.

parameters:

None explicitly for initializations, but layers
are added sequentially

Return type: a model object

ex: model = Sequential([

Dense(32, input_shape=(10,),
activation='relu'),

Dense(1, activation='sigmoid')

I)

③ Embedding:

Converts word indices into dense vectors
of fixed size (and for word embeddings)

parameters:

input_dim = Vocabulary size

→ output_dim : Dimension of dense embedding vectors.

→ input_length : Length of input sequences.

return type: 3D tensor with shape

(batch_size, input_length, Output_dim)

④ SIMPLE RNN. [tf.keras.layers.RNN]

→ basic recurrent neural network layer that processes
sequential data.

Parameters:

- ① units: hidden units (int)
- ② activation: (str) sigmoid, tanh, softmax
- ③ return_sequences: if True, returns OIP for all time steps, False only last time step is returned.

Returns:

return_sequences = False \rightarrow (batch_size, units)

return_sequences = True \rightarrow (batch_size, time_steps, units)

⑤ Dense

fully connected layer where each neuron
is connected to all inputs.

Parameters:

units: no of neurons in the layer.

activation: activation function

Return type: 2D tensor of shape (batch_size, units)

Model: Compr.

compr.:

used to config the learning proc of your model.

parameters:

optimizers: 'sgd', 'adam', 'rmsprop', 'adagrad'.

loss: 'mean_sq_err', 'mean_abs_err', 'huber_loss'.

Regns

Classification: 'binary_crossentropy' \rightarrow Binary Cross Entropy

or 'categorical_crossentropy' \rightarrow which
spare - (categorical) - crossents.

metrics: 'accuracy' \rightarrow classifier.

'mae', 'mse', 'auc'.

Early stopping

is it used to stop training a model when
model performance stops improving. This prevents
overfitting and saves computation time.

Early stopping Parameters

- ① monitor: specify metric to track
'val loss', 'val accuay', 'loss', 'accuracy'.
- ② patience: number of epochs to wait for an improvement in the metric before stopping training
- ③ restore_best_weights: (T/F)
restores model weights for the epoch with the best monitored metric.

model.fit()

fit(): used to train a model using the provided input data and labels. It adjusts the model weights to minimize the specified loss function by running through multiple epochs of data.

Parameters

- ⊗ x = input features (numpy array)
- ⊗ y = target labels (numpy array)

batch size:

no. of samples per batch for gradient update

→ epochs:

Number of times model iterates over the entire dataset during training.

→ callback's:

List of callback functions to execute during training.

(early stopping),

→ validation split:

fraction of training data to use for validation

= 0.2 20% of training data is used for validation
(unseen testing data)

III. Metrics

[loss, accuracy, val-loss, val-accuracy]			
↓ low (0.1-1.0)	↓ high (80%-100%)	↓ low (0.1-1.0)	↓ high (70%-90%)

model has min error
with the error on
the

① loss (train loss) Should be low ↓↓

→ a low loss indicate the model has minimized the error on the ~~the~~ training data.

→ extremely low loss could signal Overfitting.

② accuracy: (train accuracy) Should be high ↑↑

→ high accuracy shows the model is correctly predicting a large part of training data.

→ high accuracy ↑↑ by suggesting low validation accuracy ↓↓ overfitting

③ val loss (validation loss) Should be low ↓↓

→ low validation loss means the model is performing well on unseen data.

→ It should be close to train-loss, large gap indicates model is overfitting.

train-loss: 0.25] 0.05 val-loss: 0.30
large gap high loss: 0.20 val-loss: 0.85
overfitting model

val-loss = 0.30
[generalization model]

overfitting model

④ val accuracy (validation accuracy) Should be high, 91%

→ a high validation accuracy shows the model

is fitting well on unseen data.

→ it should be close to training accuracy. A large gap indicates Overfitting on data now

$$\text{Trn-acc} = 85\%$$

$$\text{Val-acc} = 87\%$$

$$\begin{matrix} \text{gap} \\ \downarrow \\ y_1 \text{ vs. } y_2 \end{matrix}$$

$$\begin{matrix} \text{Trn-acc} = 95\% \\ \text{Val-acc} = 70\% \end{matrix}$$

Overfitted model

after early model

→ model.compile()

→ model.fit()

→ model.evaluate() → loss, acc = model.evaluate(x_val, y_val)

→ model.predict() → pred = model.predict(x_val)

→ model.summary() → tells model architecture

→ model.save()

→ model.load_model()

→ model.get_weights()

RNN

↓ Vanish

LSTM RNN [Long short Term Memory
RNN]

RNN \rightarrow long term dependency \rightarrow Vanishing Gradient problem

① RNN \rightarrow problem?

② Why LSTM RNN?

Long Term memory

③ How LSTM RNN \rightarrow short term memory.

④ LSTM architecture

⑤ Working of LSTM RNN

Colah's blog

Problems with RNN

Taste

Next word in a sentence

gap is less.

① the color of the sky is _____
blue

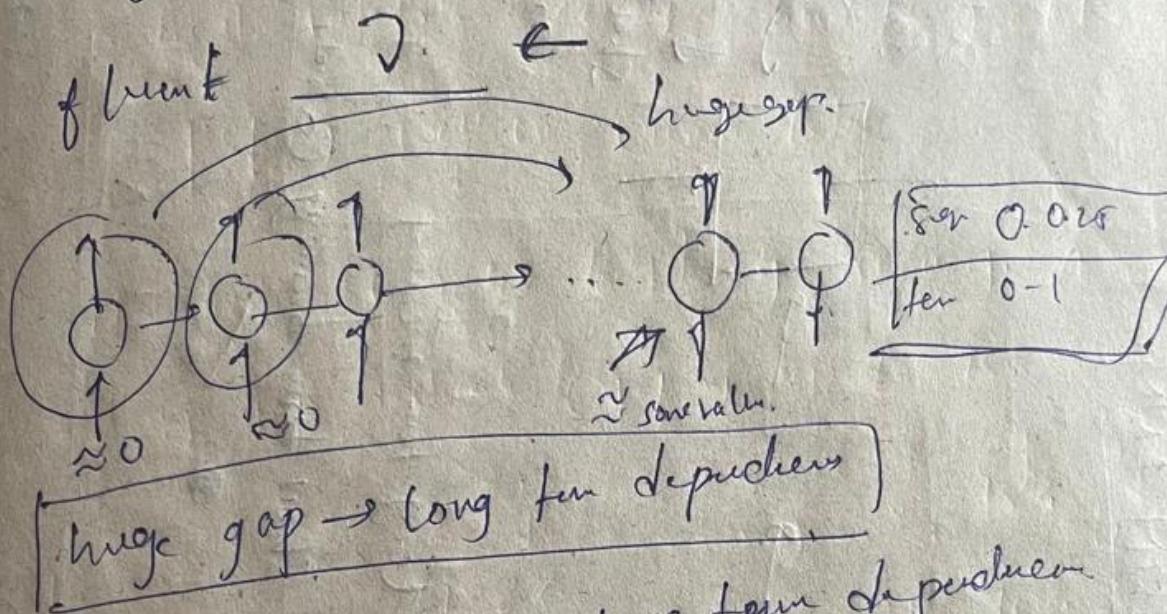
small sentence

does not face any vanishing gradient problem

dependency \rightarrow stay, come

(2) large square

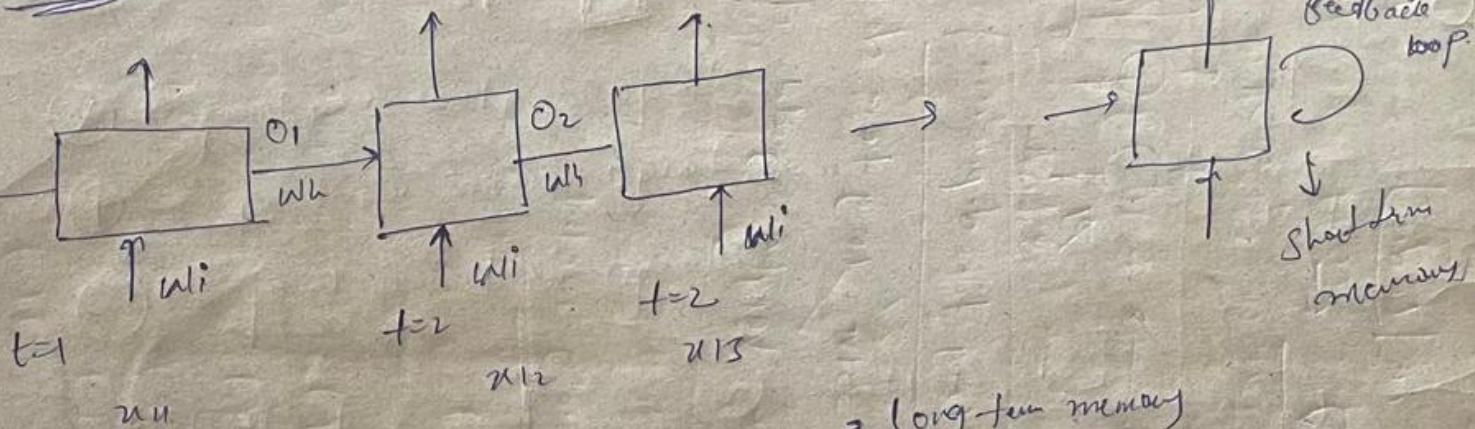
I grew up in India. --- I speak



RNIN cannot solve long term depression because it fails during great descent

Basic Representation of RNN and LSTM RNN.

RNN



LSTM RNN → Long term memory
 → Short term memory (Context prod)

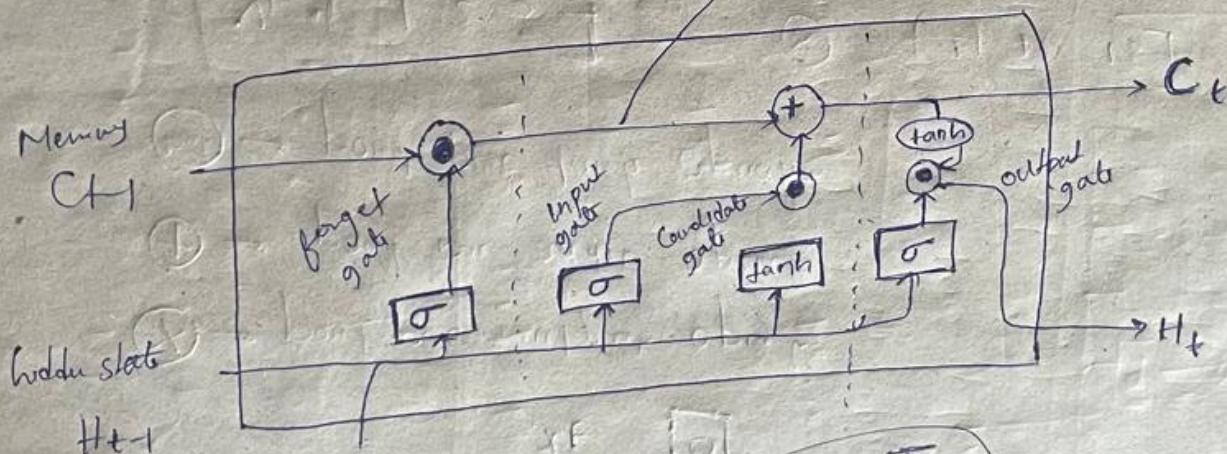
~~Conveyance belt~~: Luggages

memory cell. → LSTM → long term memory ← adds what context is required and removes what context is not required

I grew up in [India]. ... I speak fluent

Saves this context in memory cell.

LSTM architecture:

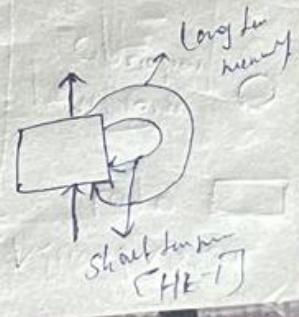
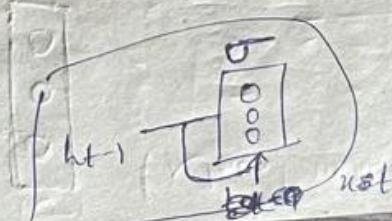


three up gates

forget gate

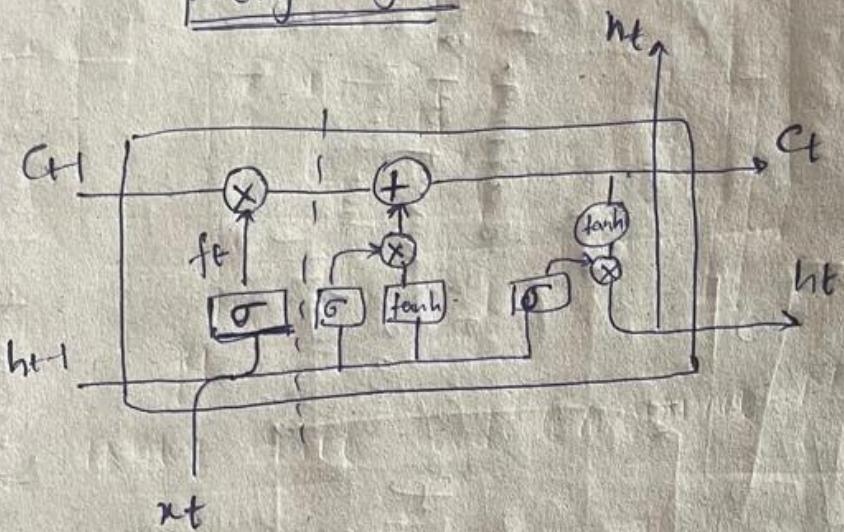
input gate

output gate

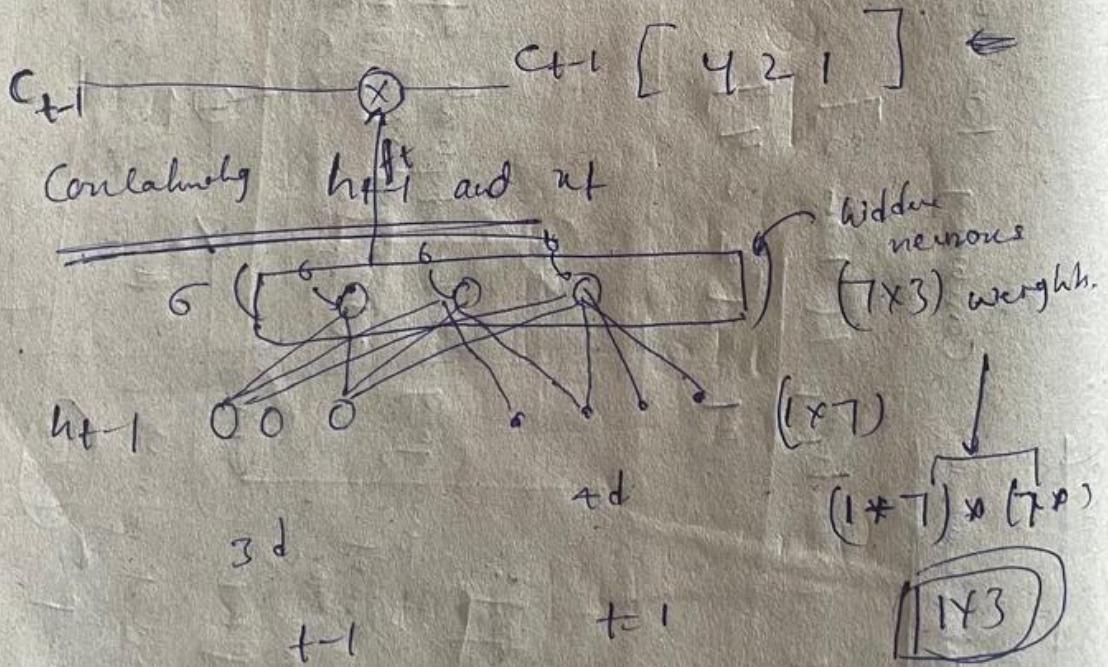


all about gates

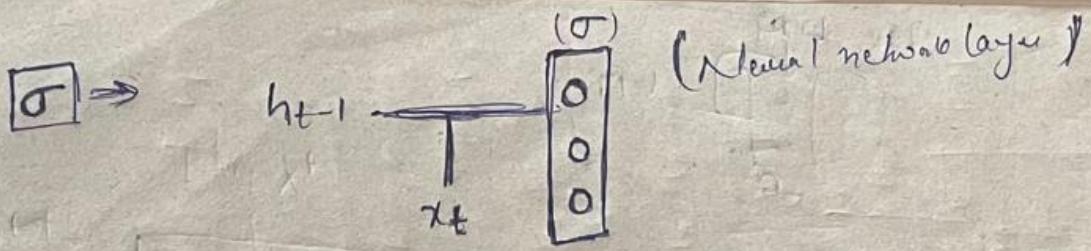
Forget gate



forget next word
 $y_{15}^{nt} = [0 \ 2 \ 4 \ 0 \ 1 \ 5 \ 1 \ 0 \ 2 \ 4] \dots$
 $h_{t+1} [1 \ 2 \ 4]$ also ^{3d}



let $ft = [0 \ 0 \ 0]$ $[-, -, -]$
 $h_{t-1} [0 \ 0 \ 0]$ Output dimen 1×3

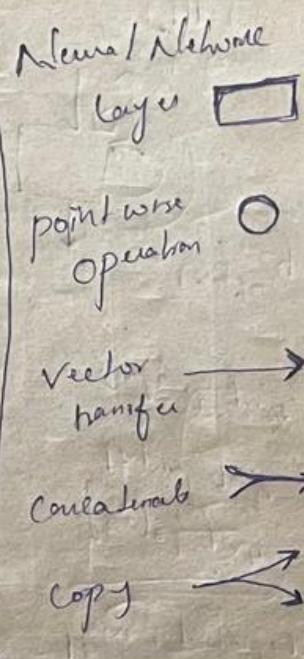


- $\times \rightarrow$ point wise muliply operation
- $+$ → point wise add operation
- $\tanh \rightarrow$ point wise tanh operation

$$v_1 [1 \ 2 \ 3 \ 4] \quad \times \rightarrow [4 \ 10 \ 18]$$

$$v_2 [4 \ 5 \ 6] \quad + \rightarrow [5, 7, 9]$$

$$\tanh \rightarrow [\tanh(1) \ \tanh(2) \ \tanh(3)]$$

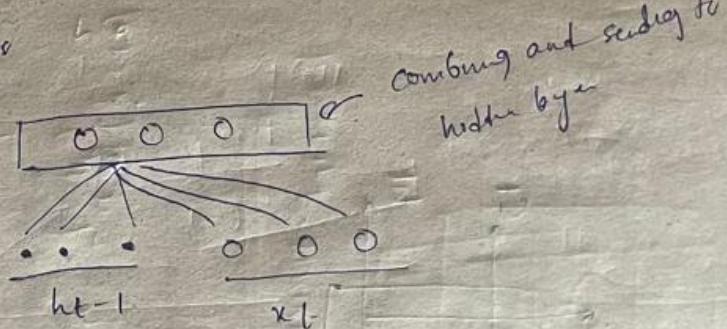


Vector transp: we are just transposing vector to one state to another.

Concatenation: Combining two vectors

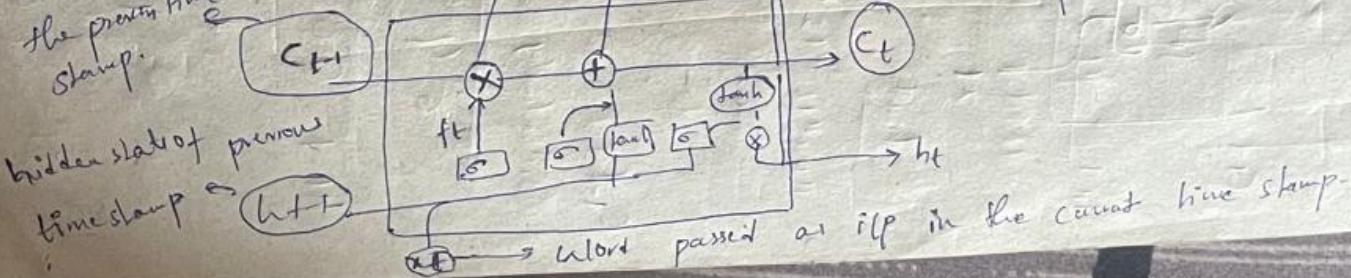
$$h_{t-1} = [1 \ 2 \ 3]$$

$$w_t = [2 \ 3 \ 4]$$



Copy:

We are going to take duplicate of a vector after removing some context adding context bit



after removing and adding the new state of memory cell at that particular time stamp.

hidden state of previous time stamp $\rightarrow h_{t-1}$ word passed as input in the current time stamp

let ft

H_1

$$\textcircled{1} \quad C_{t-1} = [689] \otimes [000]$$

$$= [0.00] \leftarrow \text{Removing all the previous content}$$

$$\textcircled{2} \quad C_t = [689] \otimes [111]$$

$$= [689] \leftarrow \text{not removing any content.}$$

$$\textcircled{3} \quad C_{t-1} = [689] \otimes [011, 011]$$

$$= [38411] \Rightarrow \text{we are removing some content info } (6, 9)$$

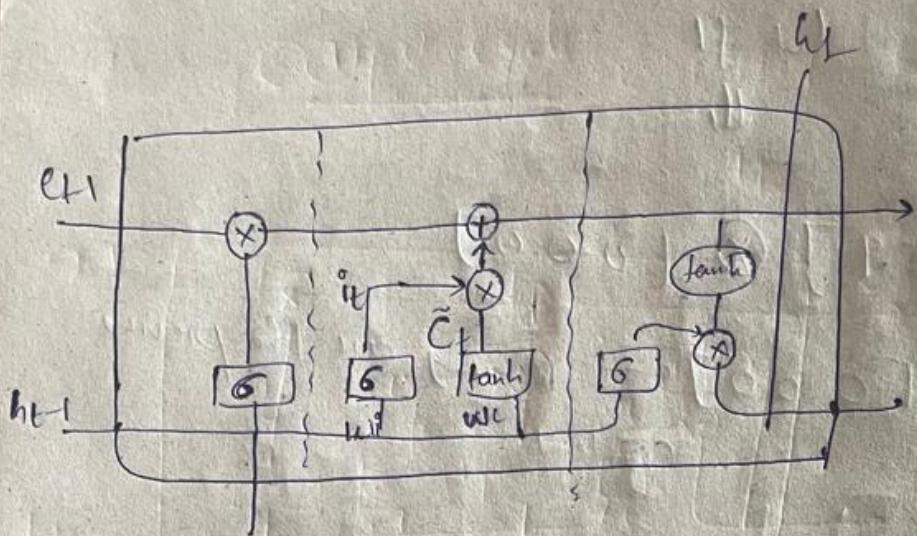
Conclusion about forged gat:

Based on the content \rightarrow

forged gat will let go some information \oplus)

not let go some info $\&$ forgeshtg \oplus

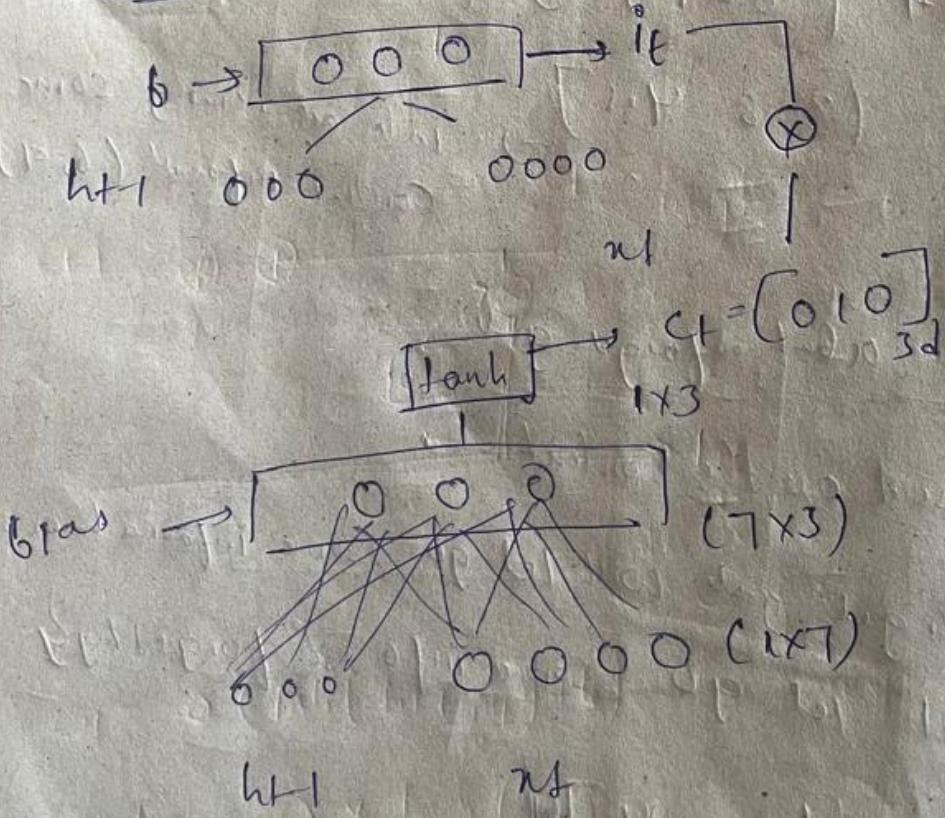
$$G \left[\begin{smallmatrix} w_f & [h_{t-1}, n_t] \\ \downarrow w_f & \downarrow h_{t-1} \end{smallmatrix} \right]$$



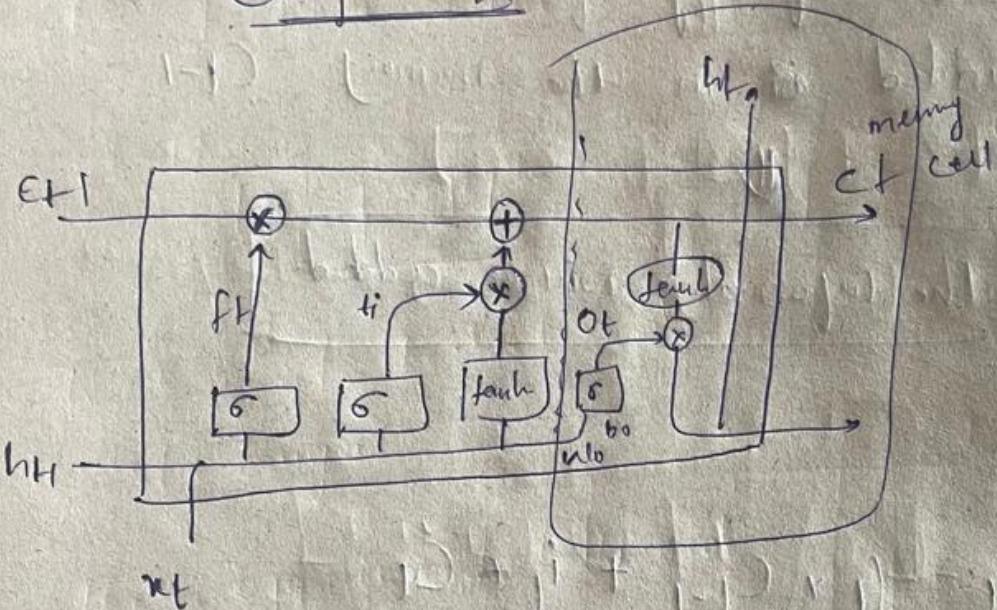
INPUT GATE and CANDIDATE MEMORY

$$i_t = [2 \times 6]$$

Simulate OLP & how we get info in gate

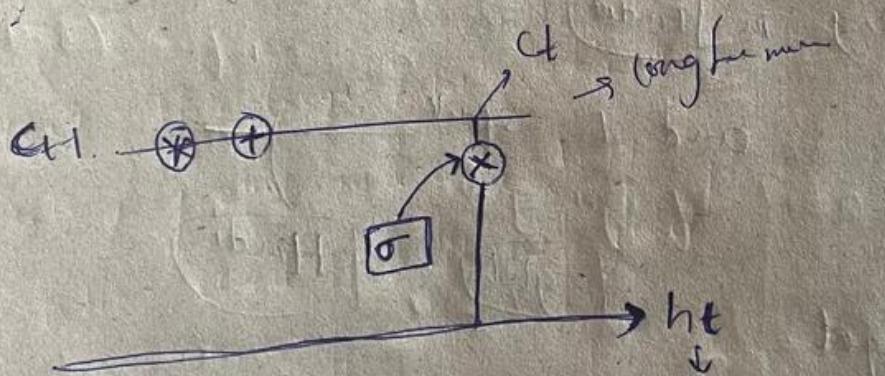


Output Gate



forget gate: forget some info based on the
Content

input gate: add some info based on the
Content



$$o_t = \sigma(w_o(h_{t-1}, i_t) + b)$$

$$h_t = \alpha \times \tanh((i_t) \cdot 2)$$

(writable to get shall have more)

Content: If any information needed to be added ~~is~~ in the memory C_{t-1}



The information will be added.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Removing

(①)

forgetting

Some info

②

Candidate

memory

I stay in India → save this content in the
memory cell

and I spell →

English

Hindi

Forgot gat

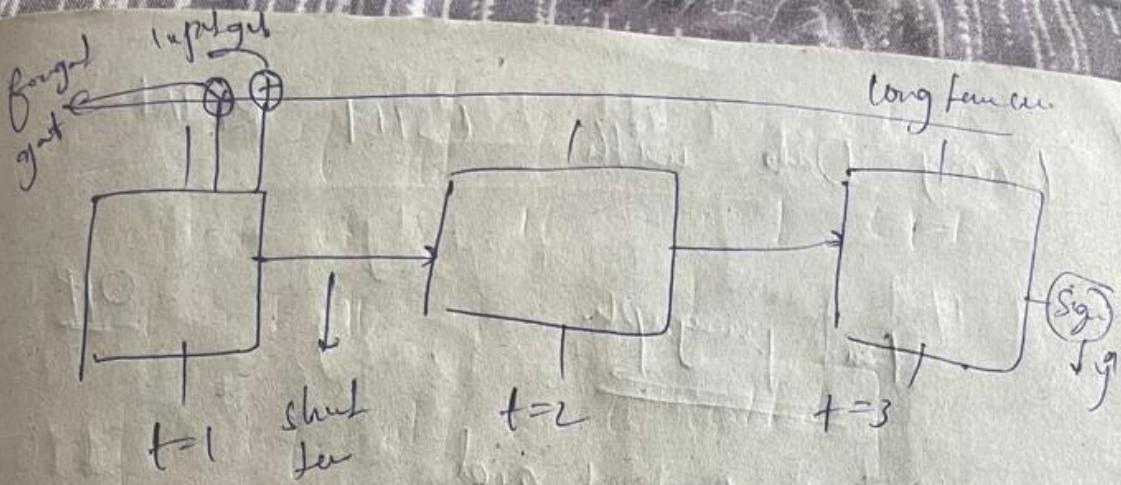
Input gat ③ Candidate memory

⊕

C_{t-1}

④

C_t



vali, inc, inc. \rightarrow updatig them with
• using game propagation

Overall

forget gate: forgets the information, based on
the content

input gate: add some information based on
the content

output gate: used to distinguish properly about
memory cell and hidden slate

\downarrow
which reports
short term
memory.

Training Data with LSTM RNN

Tent paragraph.

OIP (go
bad)

I went to Restaurant and
order Burger.

\$10

The Burger looked fresh and
Crispy

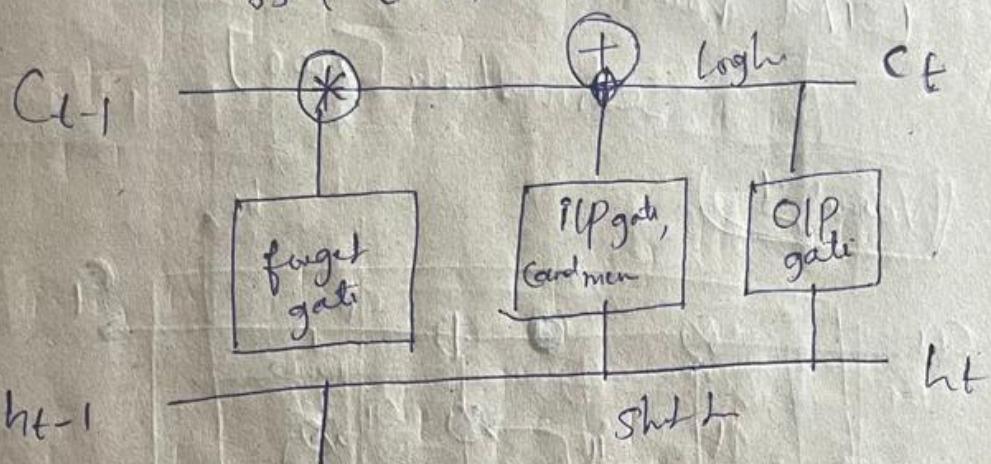
But Burger is not good for
health.

It has lot of fats, cholesterol
But this Burger was made with
whole protein and only veggies
were used, so it was good.

S_1 [good, bad, healthy] \rightarrow Tasty + crisp

S_2 [0.7, 0.4, 0.0] not good.

S_3 [0.4, 0.6, 0.0] \rightarrow lots of fast & clear.



Step 1:
words \rightarrow vector \rightarrow embedding layer.

Suppose word2vec (3 dim vec)

	good	bad	healthy	\leftarrow	black
Tasty	[0.9]	0.0	0.1]	sd	long
				do not	know
				what	happening
				in my	

$$S_4 = [0.7, 0.3, 0.9]$$

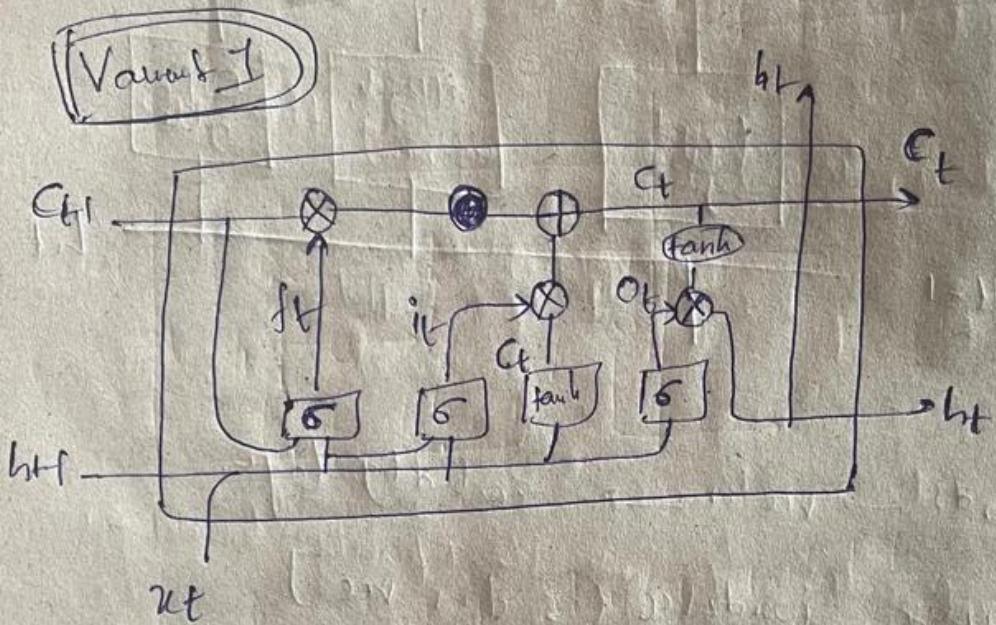
\hookrightarrow made with Wiley print terms

$$[0.0, 0.2, 0.1] \quad [0.4, 0.1, 0.4] \quad [0.2, 0.4, 0.1]$$

Variants of LSTM RNN

(LSTM
[1970-80])

LSTM Variant introduced by Gers &
Schmidhuber [2000]



Connections

from memory cell to

forget cell

ipg gate

opg gate

Peephole

connection.

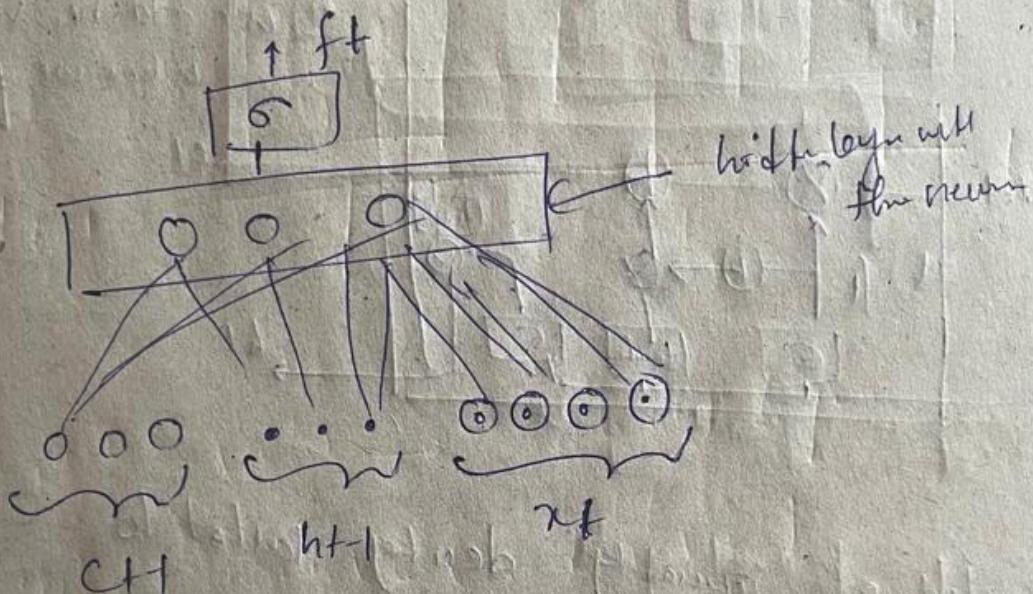


Peephole connection: we let the gates layer
look at the cell state

$$f_t = \sigma(w_f \cdot [c_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(w_i \cdot [c_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(w_o \cdot [c_t, h_t, x_t] + b_o)$$



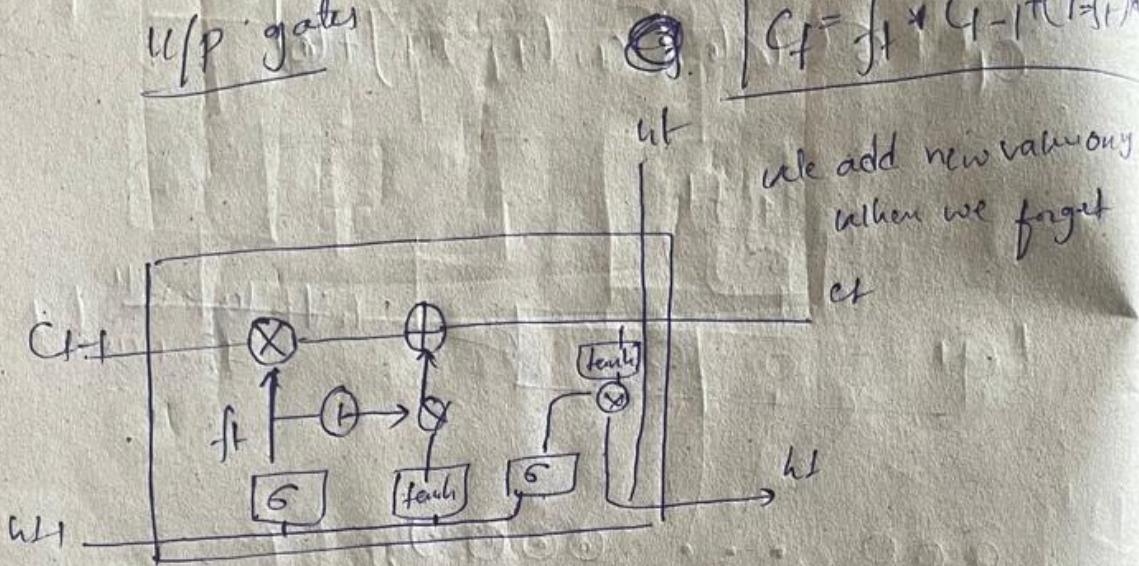
Variant 2

when we are combining

W/P gates

Forged and

$$C_f = f_t * C_1 - 1 + (1-f_t) * C_0$$



we add new value only
when we forget

Instead of separately destroying what is
forged and what we should add
new info. we make this decision
together

goal is: we only forget, when we
are going to help something in its
place.

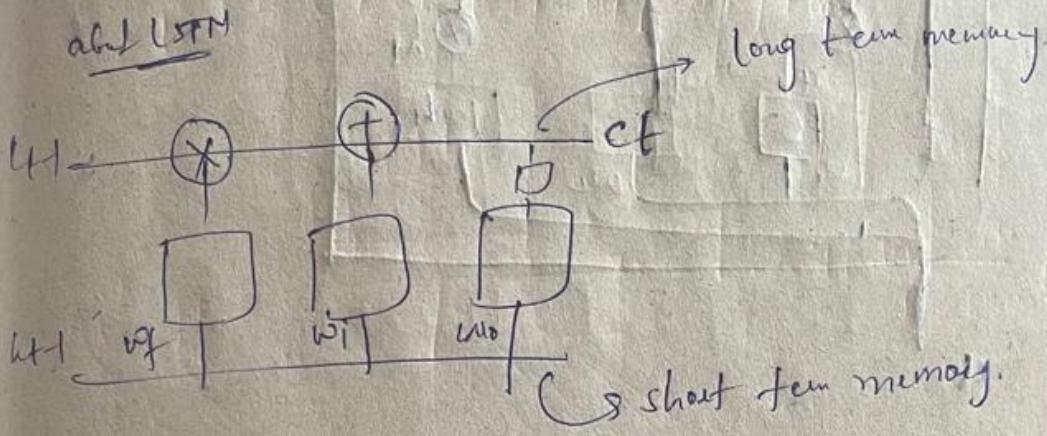
→ we only help new new value to the slots when
we find something older

GRU RNN

② 1980 → LSTM
↓
2000 → vanila

Gated Recurrent Unit

2014 → GRU



LSTM: complex architecture.

forget, input + candidate \rightarrow O/P

$$[(w_f, w_i, w_o)]$$

6x3 ✓

Training time ↑↑

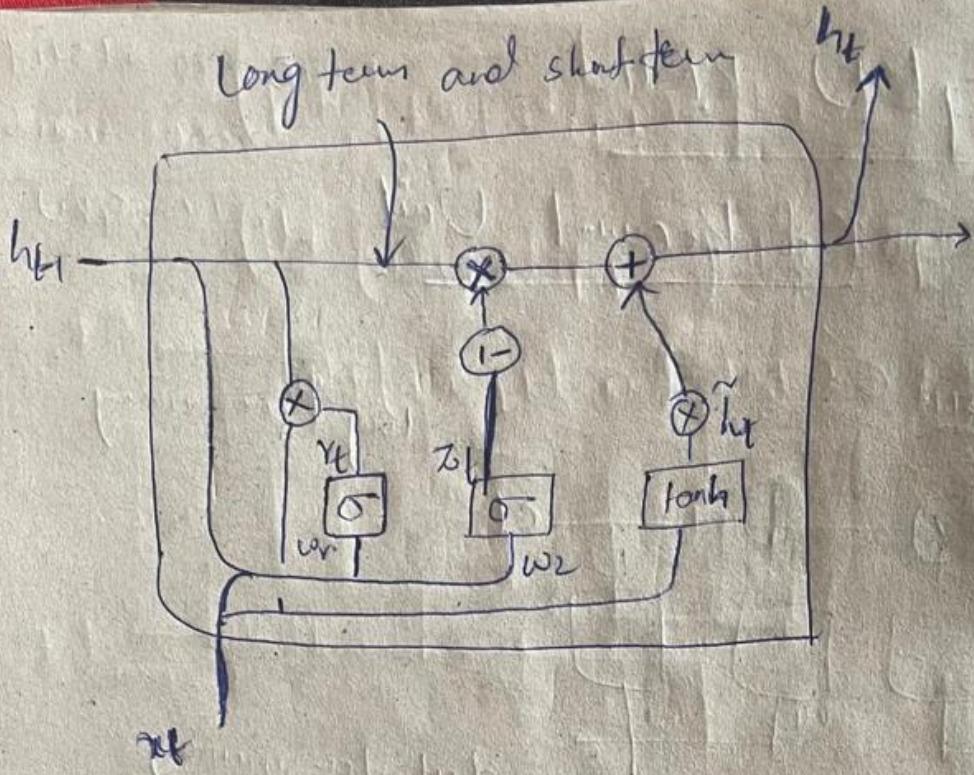
① Training per T ↑

Training time ↑↑

Trainable parameters ↑↑.

Cho, et al (2014) came up with

GRU



z_t = update gate

r_t = Reset gate

\tilde{h}_t = Temporary hidden state

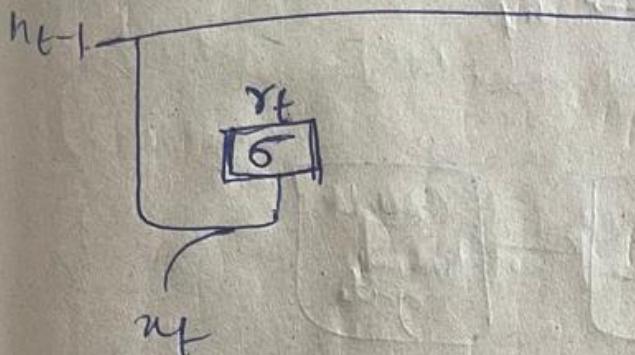
$$z_t = \sigma(w_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(w_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(w_l \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Reset gate (r_t)

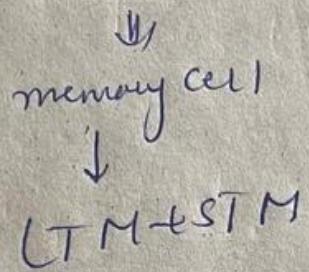


$$\begin{aligned}
 h_{t-1} & [0.6 \ 0.5 \ 0.3 \ 0.9] \\
 r_t & [0.2 \ 0.4 \ 0.8 \ 0.2] \\
 & \downarrow \ 1 \ 4 \ 8 \ 4 \\
 & 0.12 \ 0.20 \ 0.24 \ 0.18
 \end{aligned}$$

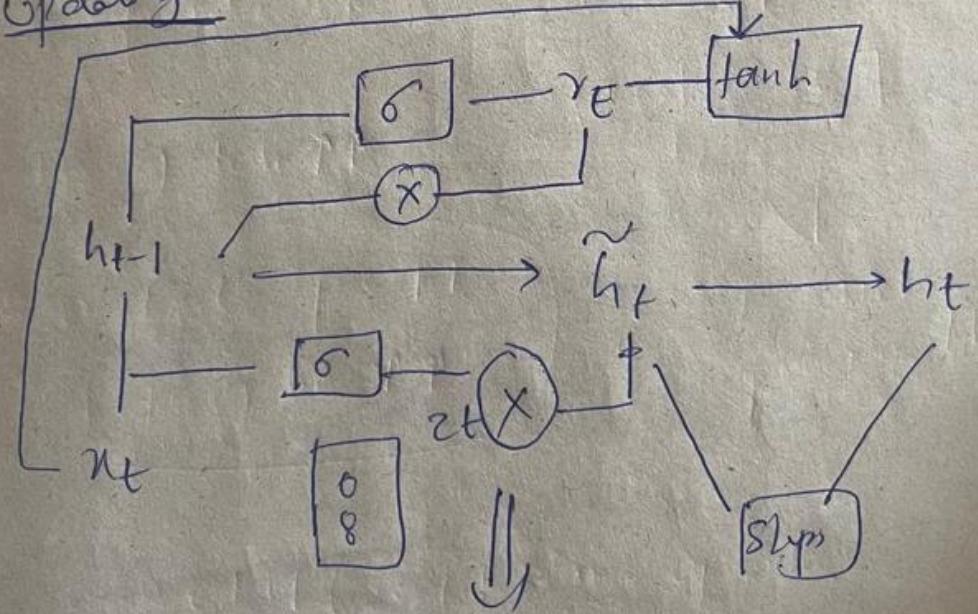
resetting value

resetting content

→ Resetting some info for h_{t+1}



update gate:

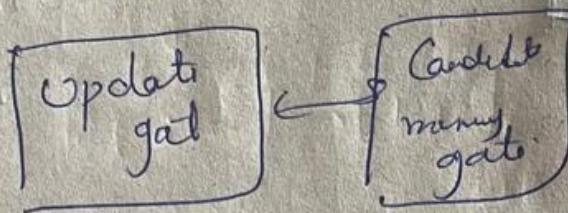


What content info needs to be added

$z_t \rightarrow$ less info. Candidate hidden slate [current content]
 $\xrightarrow{\text{imp}} \text{add info}$

in short to
update

h_{t-1} \longrightarrow h_t



H^t \downarrow
 H^{t-1}, H^t, H^t



END-TO-END - Deep learning project using

LSTM RNN and GRU RNN

① Data collection:

② Load the dataset

③ Data processing.

Tokenizing:

if convert text into sequence of integers,
where each integer represents unique token

text = ["I love Mar Lang"; "Leaves is fun with
python"]

tokenize = Tokenizer(num_words=10, oov_token='<OOV>')

tokenize.fit_on_texts(texts)

↳ [`'<OOV':1, 'lang':2, 'i':3, 'love':4`
`... 3`]

Sequences = tokenizer.texts_to_sequences(texts)

↓
[(3, 4, 1, 7, 6, 6, 7, 8, 9)]

(ii) $\text{tokensList} = [1, 2, 3] \rightarrow$ we prove,
 for i in range ($1, (\text{len(tokensList}) + 1)$): $i=1, 2$,
 $\text{in_gre} = \text{tokens}[:i+1] = \begin{cases} [1, 2] & \text{if } i=1 \\ [1, 2, 3] & \text{if } i=2 \end{cases}$
 \downarrow
 $\text{tokensList}[:2] \rightarrow [1, 2]$
 $\text{tokensList}[:3] \rightarrow [1, 2, 3]$ In_gre

after
Pad sequence
 array $\begin{bmatrix} \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{bmatrix}$
Creating predictors and labels

$x, y = \text{input 1, input 2}$

convert y to one-hot-encoded form.

$y = \text{tf. onehot}(\text{y}, \text{num_vocab})$
 essential for using neural network model

Embedding \rightarrow Convert indices to dense word vectors

LSTM \rightarrow Captures dependency b/w words

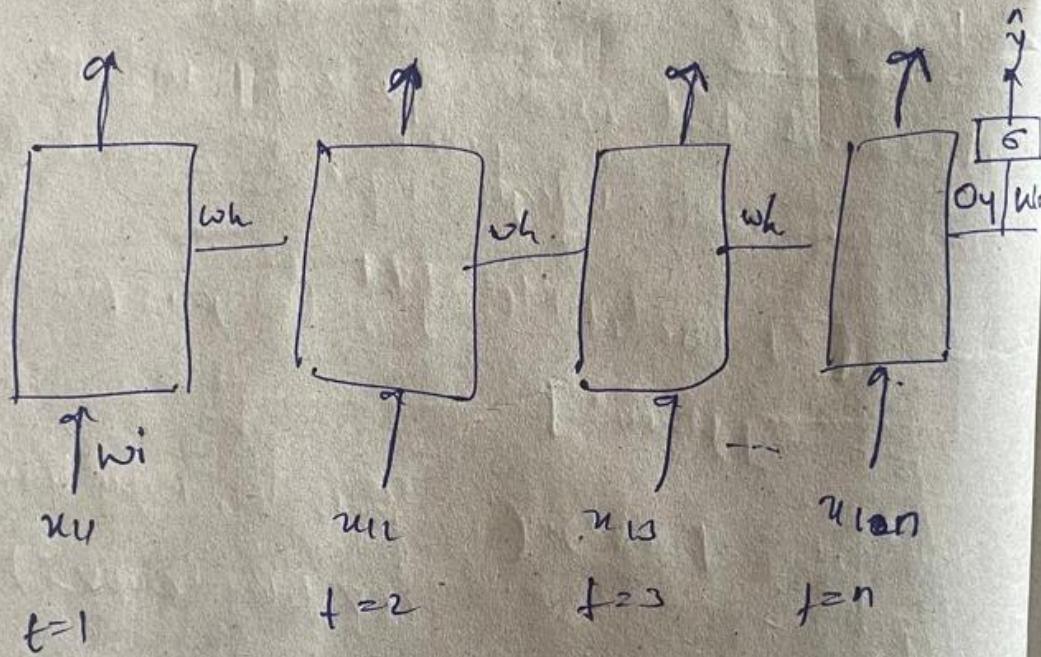
dropout \rightarrow prevent Overfitting

(LSTM(2)) \rightarrow Summarizing the sequence in one vector

Dense \rightarrow predicts next word as probability

Bi-Directional RNN architecture and intuition

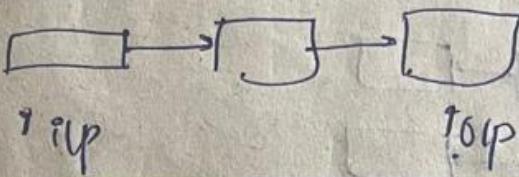
- ① Simple RNN
- ② LSTM, GRU RNN
- ③ Bidirectional RNN [LSTM RNN]



Type of RNN

- ① One to many RNN
- ② Many to one RNN
- ③ Many to Many RNN
- ④ One to One RNN

① One to one:



one i/p

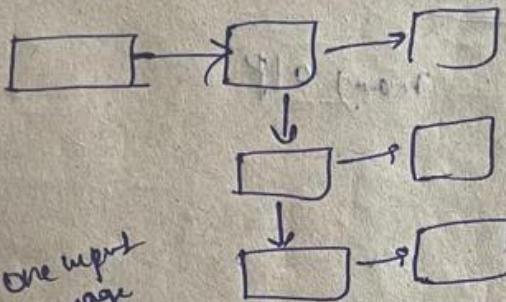
one o/p

1 i/p

1 o/p

② One to many:

loss



one i/p
many o/p

one input
image

eg:-

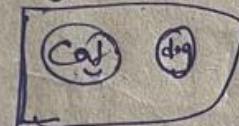
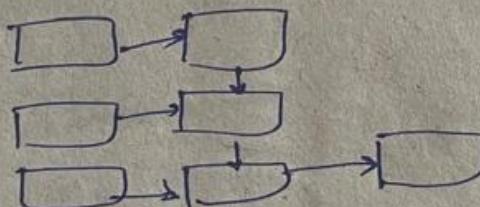


Image Capturing

↳ There is a dog and Cat ← many outputs in form of words

③

many to one:

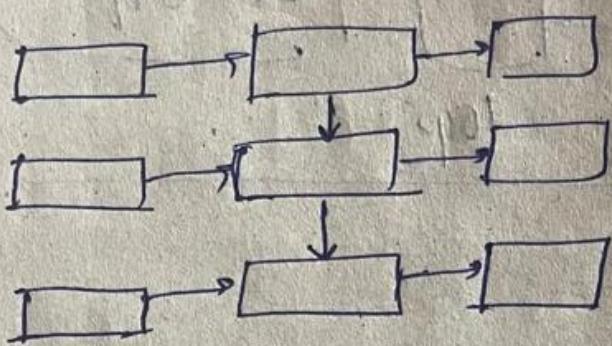


many i/p
one o/p

Ex image search [Cat during walk] I/P

↳ Image → one o/p

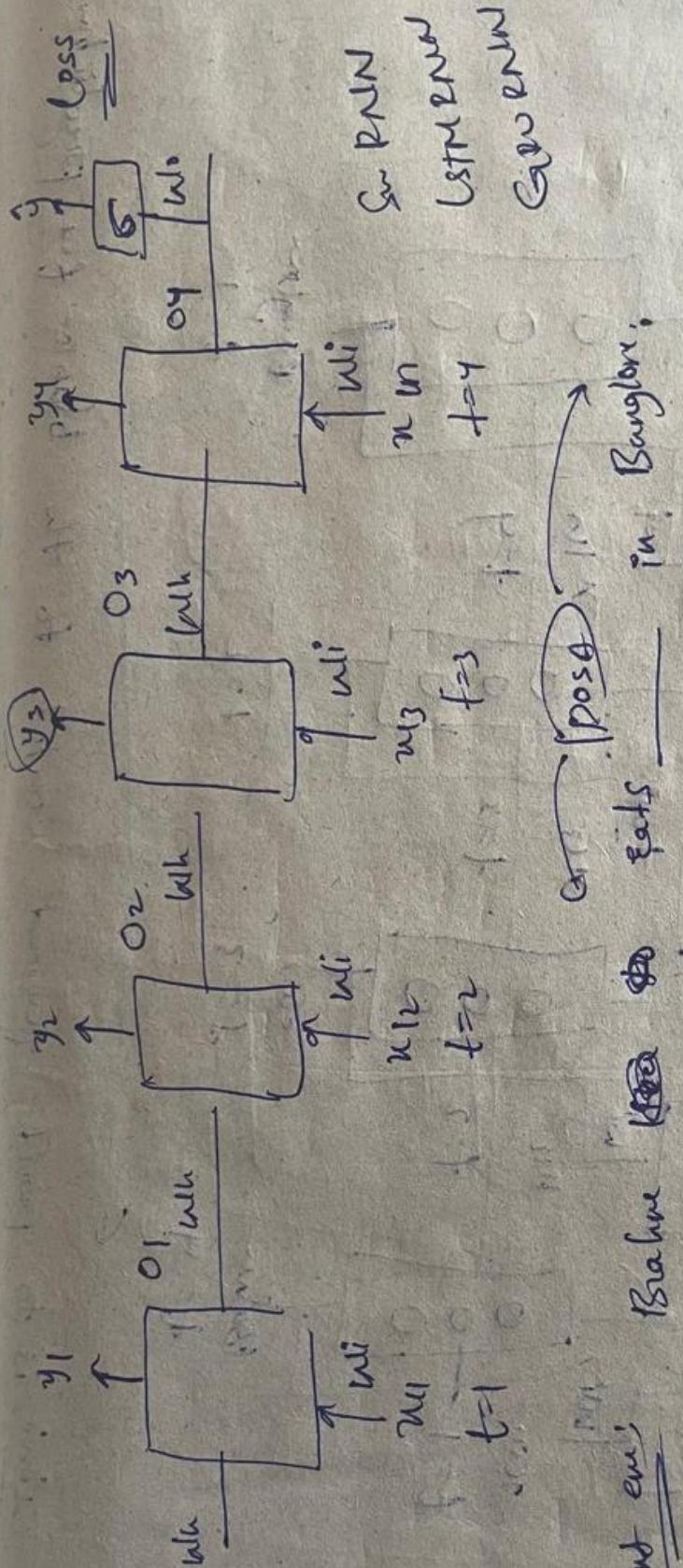
Many to many



Language Translation:

many IIP many OLP

Business → [to] Business → [to] Business
in English in French in German



Tent en: Brahme ~~W~~ ^W W

Q But this deputy on
further sentence

to find thus kind of problems

and directly on
further sentence

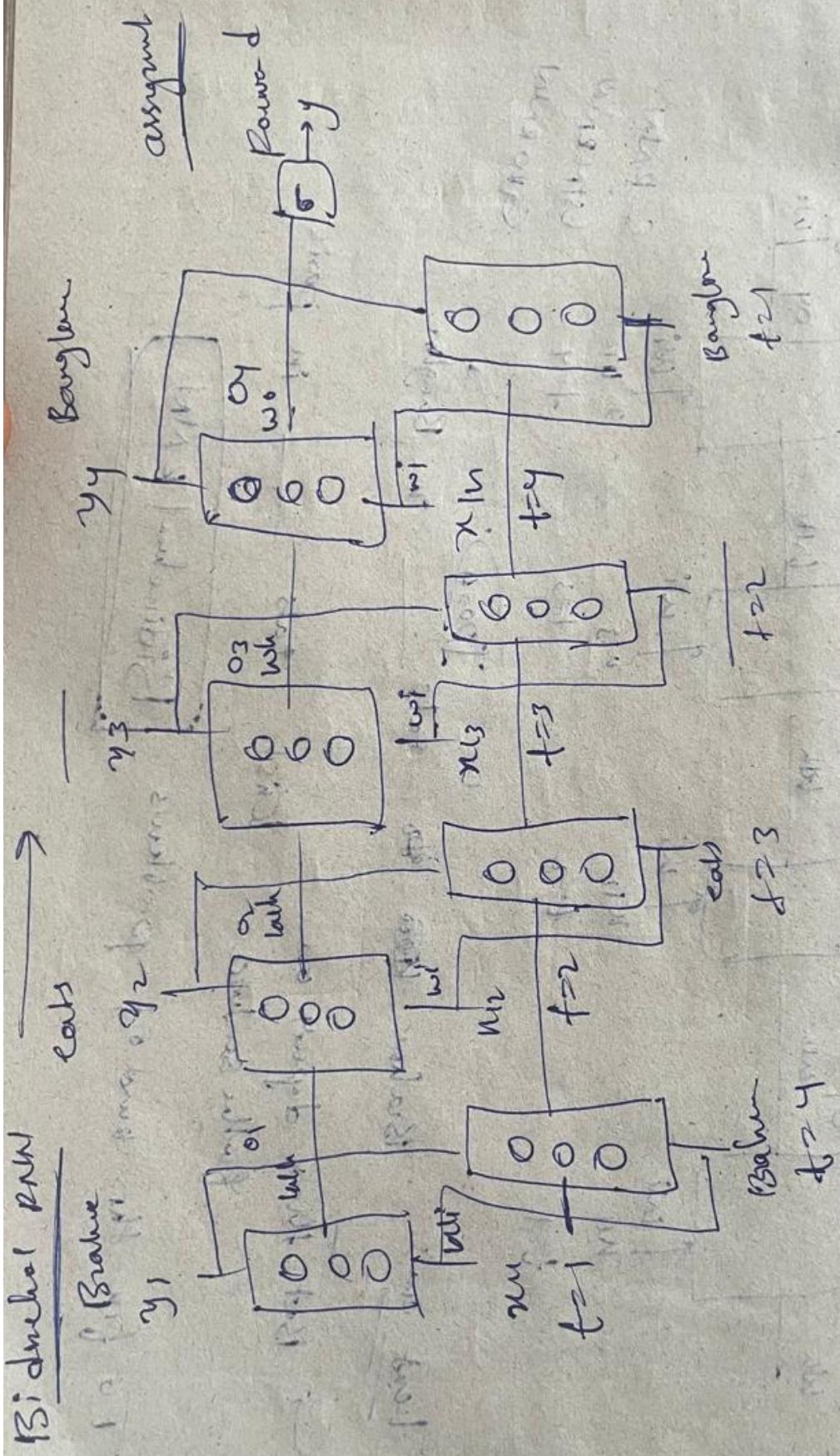
15

Wlich ^{reals}

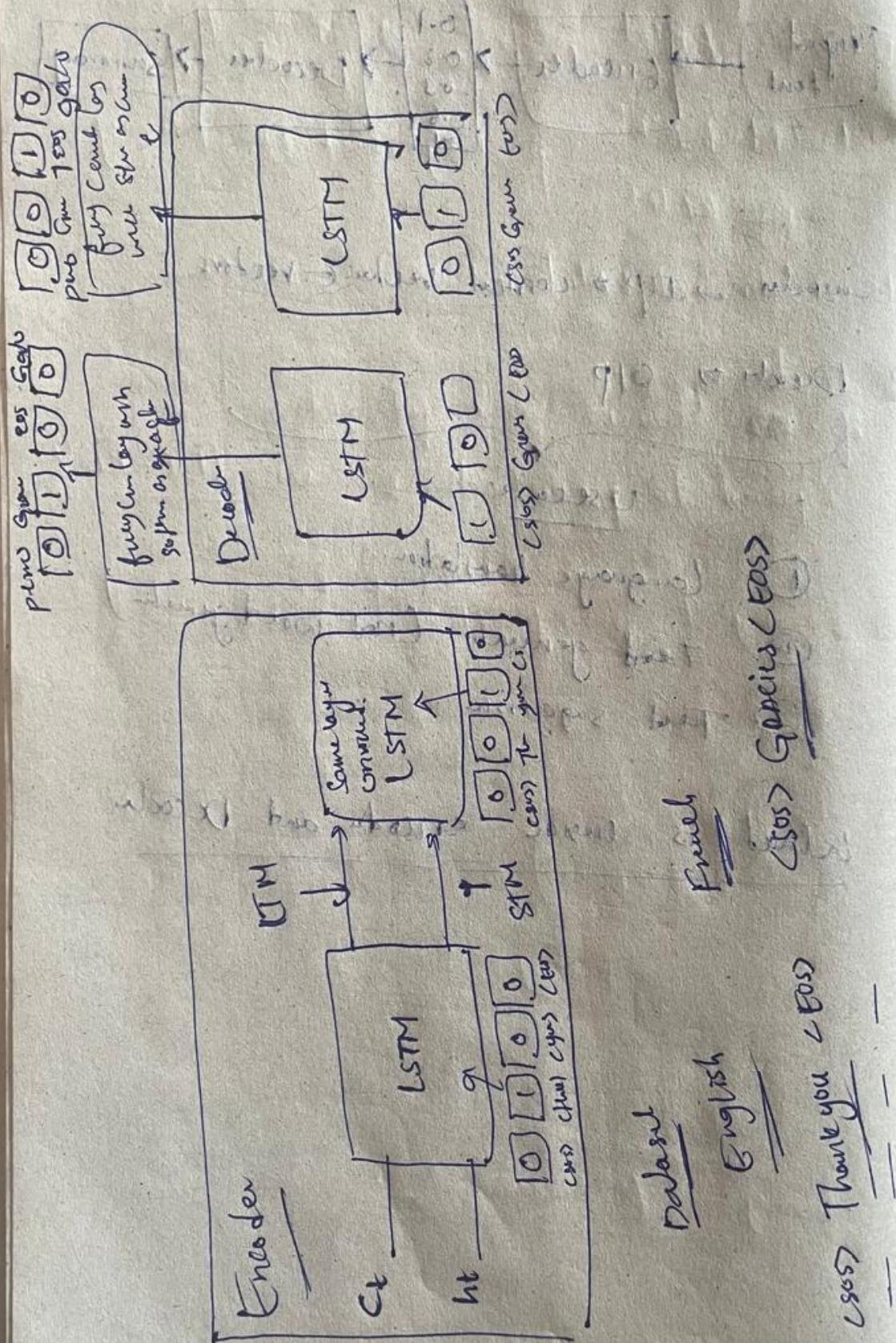
10

To find thus kind of problems

Bidirectional RNN



aim is to provide forward contact to the bank for prediction



→ The encoder compresses all the inputs
informed into a single fixed length vector.

That vector is passed to the decoder.

→ For long sequences, fixed length
vector struggle to retain all information
leading to poor performance

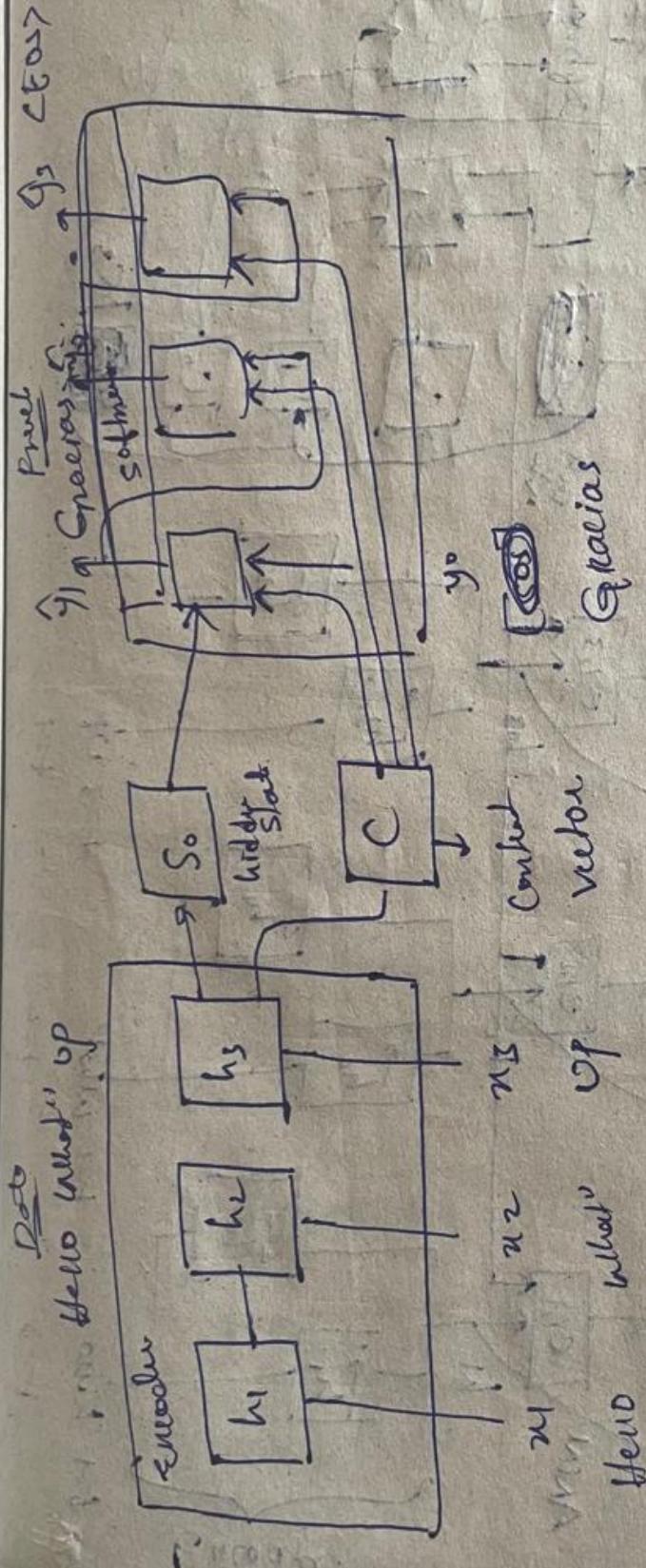
↓
To avoid this
we use

Attention mechanism → Seq2Seq network

longer paragraph → L. Context vector

+
additional context vector is
passed

Attention Mechanism | Seq2Seq Network



Problem - decoder needs to look back in time

Problem with encoder and decoder is, Content vector is not going to capture the essence of information. If the sentence is too long, then slow = $t = 100 \times$

④ Scaling:

We take up the slopes and scale down by
downing the slopes by the $\sqrt{d_k}$ (key
downing)

$$d_k = 4$$

$$\sqrt{d_k} = 2$$

Scaling in the attention mechanism is used
to prevent the dot product from growing
too large \Rightarrow ensure stable gradients

during Training

① Gradient exploding

② Software saturation by vanishing gradients

$$Q = [2, 3, 4, 1] \quad k_1 = [1, 0, 1, 0] \quad k_2 = [0, 1, 0, 1]$$

constant scaling:

$$Q \cdot k_1^T = 2 \times 1 + 3 \times 0 + 4 \times 1 + 1 \times 0 = 6$$

$$Q \cdot k_2^T = 2 \times 0 + 3 \times 1 + 4 \times 0 + 1 \times 1 = 4$$

Scale = [6, 4]

↳ scaling not applied

$$\text{softmax}([6, 4]) = \left[\frac{e^6}{e^6 + e^4}, \frac{e^4}{e^6 + e^4} \right]$$

$$= \left[\frac{e^6}{(1+e^{-2})}, \frac{1}{e^2+1} \right]$$

$$\approx [0.88, 0.12]$$

most of the weight is assigned to the first key vector, very little to the second vector.

Smoothly

property of softmax

$$([10, 1]) = [0.99, 0.001]$$

does not help in weight update

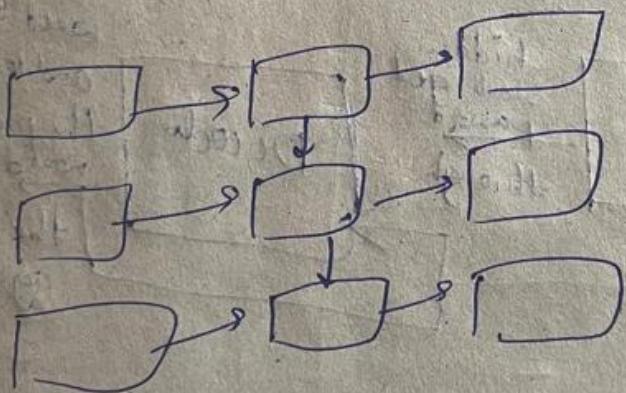
Causes vanishing gradient descent

Encoder and Decoder

- ① simple RNN \rightarrow vanishing Gradient problem
- ② LSTM RNN \rightarrow Long short Term Memory
- ③ GRU RNN
- ④ Bidirectional RNN

if there is some forward context is required to predict the we use BSI-RNN

Many to many RNN

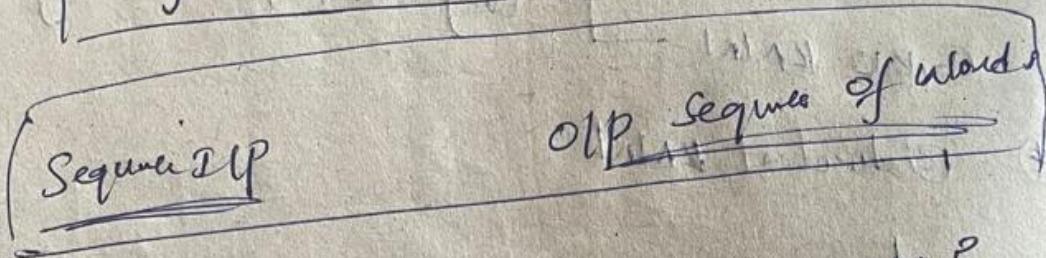
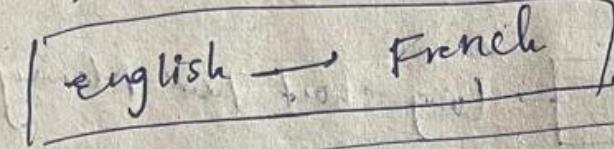


e.g: language translate..

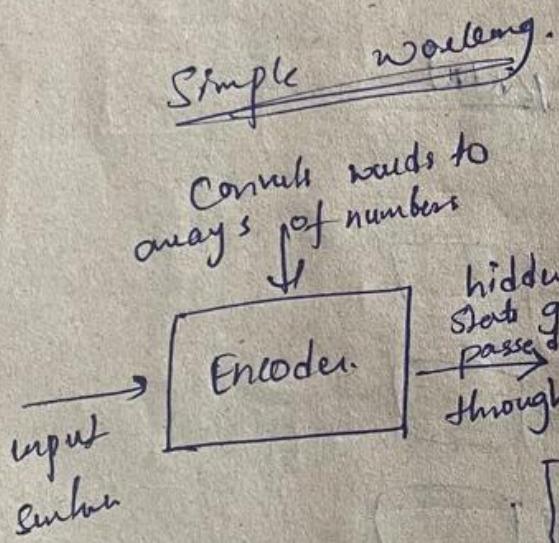
Eng \rightarrow french.

Encoder and Decoder

Creating a model e.g. one lang to other

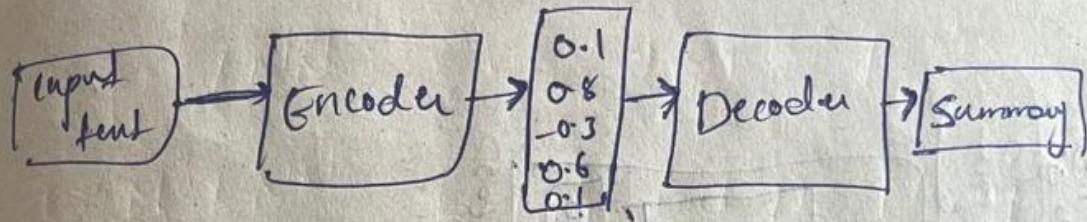


e.g. Indian Chat → hi, how are you?



The decoder generates the word by word, and keep feeding the previous word into the decoder again.

Sequence to Sequence architecture



① encoder \rightarrow IIP \Rightarrow Content vector & vectors

② Decoder \Rightarrow OIP

usecase

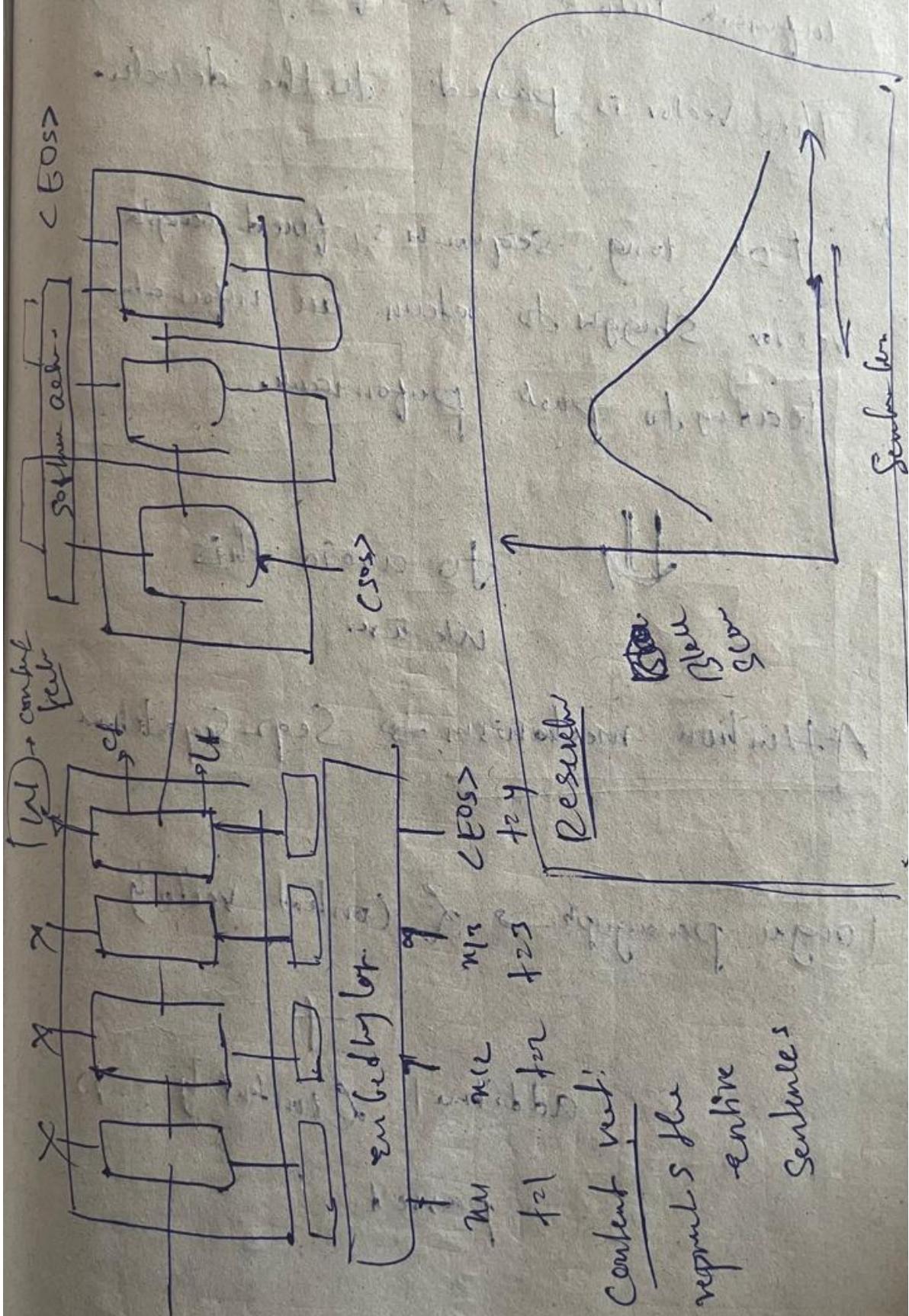
① Language translation.

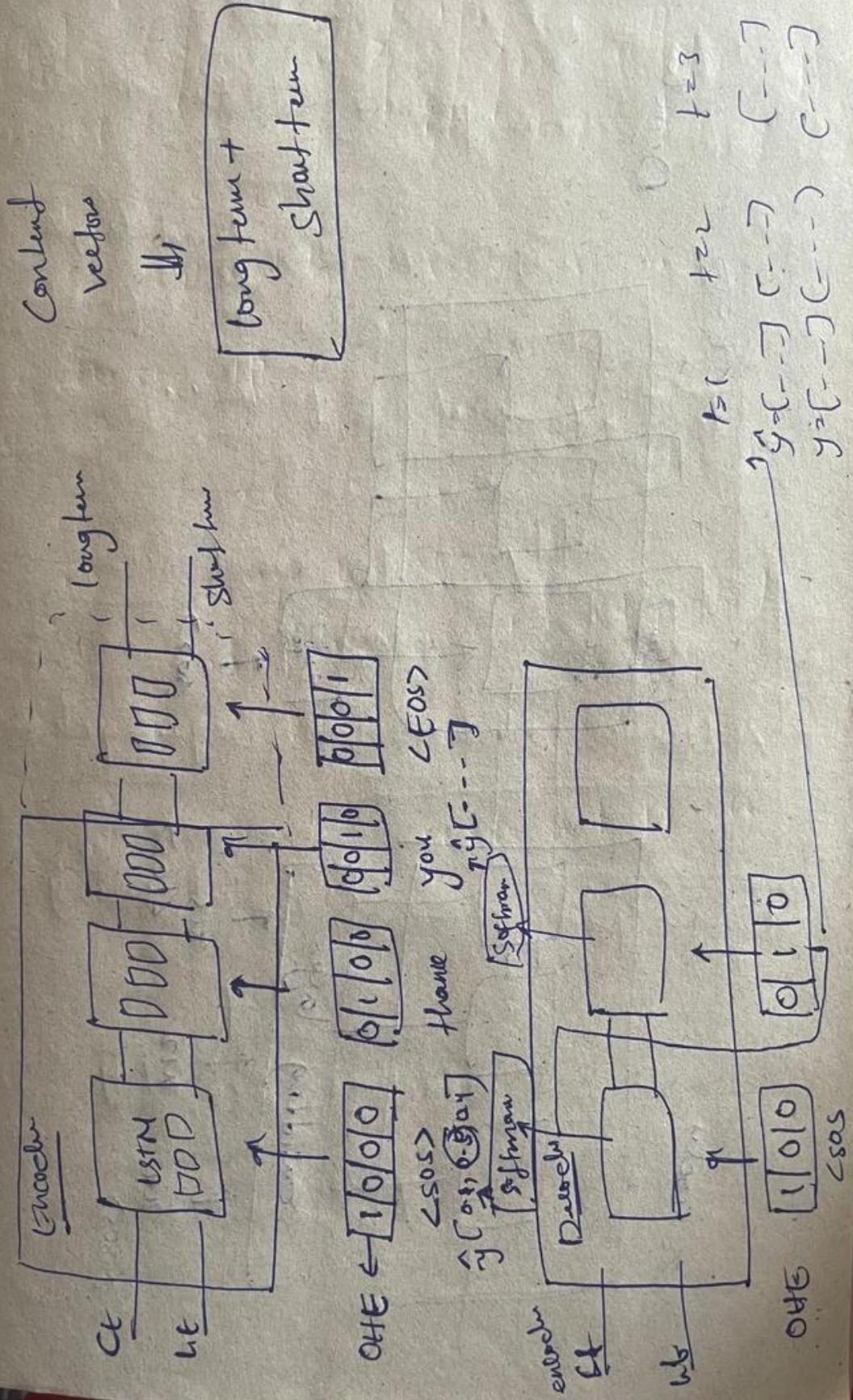
② Text generator (not word generator)

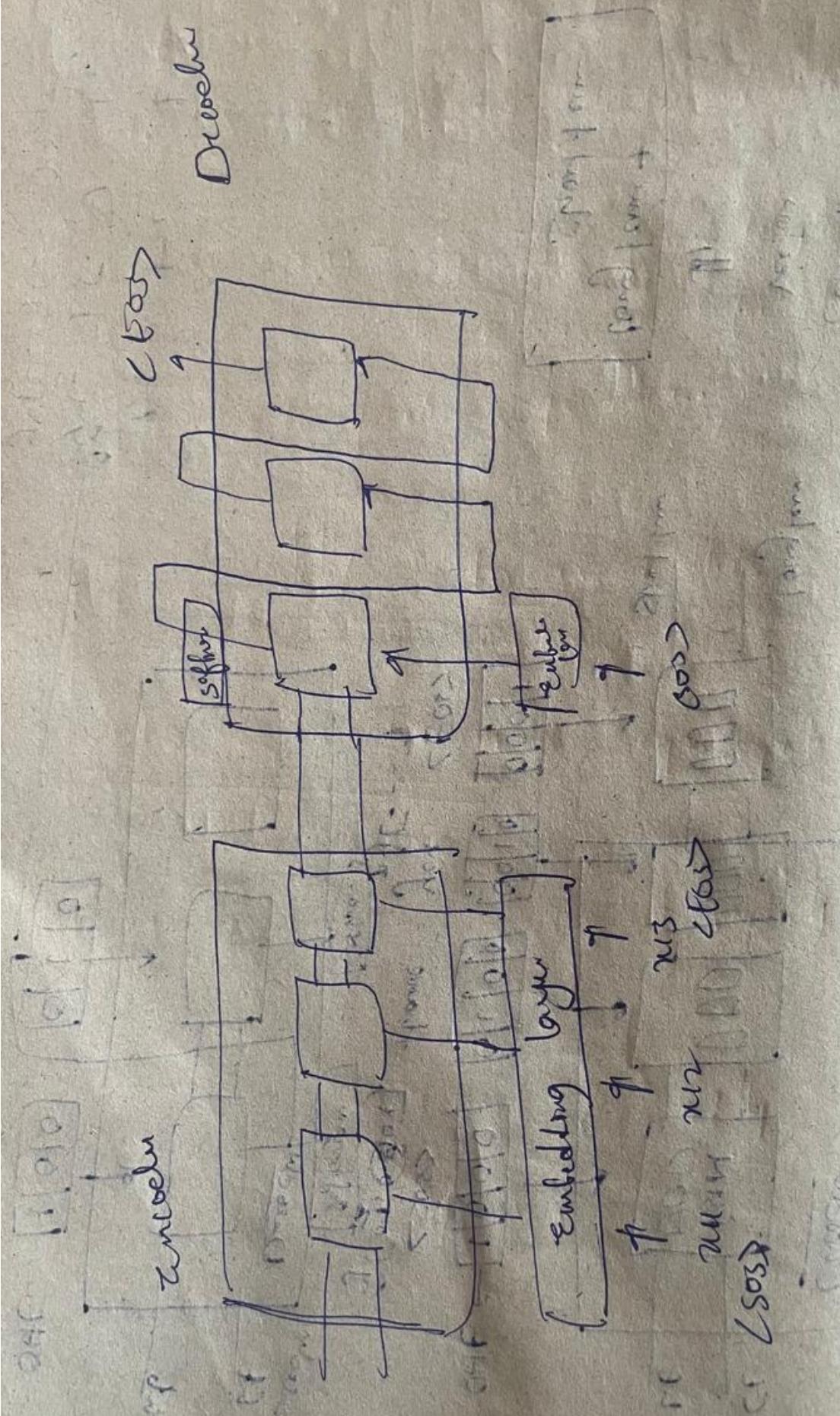
③ Text suggestion

What is inside encoder and decoder

Problems with encoder & Decoder







TRANSFORMERS

Introduction to transformer

- ① RNN / LSTM (Seq2seq)
- ② Encoder Decoder architecture
- ③ attention mechanism.
- ④ Transformers.

① Why transformers?

② Architecture of transformers?

- ① Self attention $\rightarrow Q, K, V$
 - ② Positional encoding
 - ③ Multihead attention
 - ④ Combing the working of transformers.
- } plan of action

Generative AI \rightarrow LLM, multimodel

Bert, GPT \leftarrow



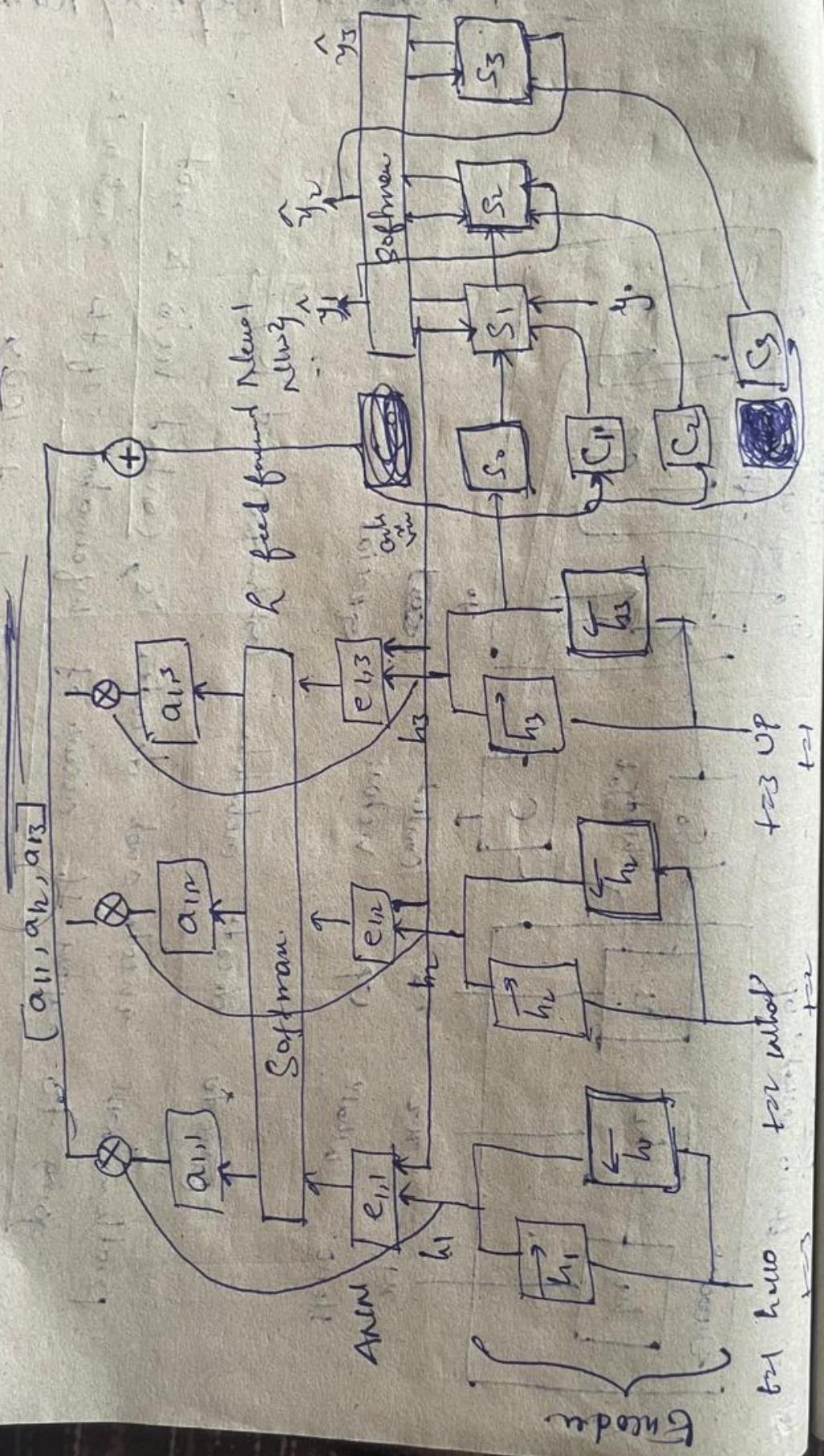
Open-AI \rightarrow Chat GPT

\leftarrow GPT-4.0

use transformer

Attention Mechanism

Attention weights



What and Why \rightarrow Transformer

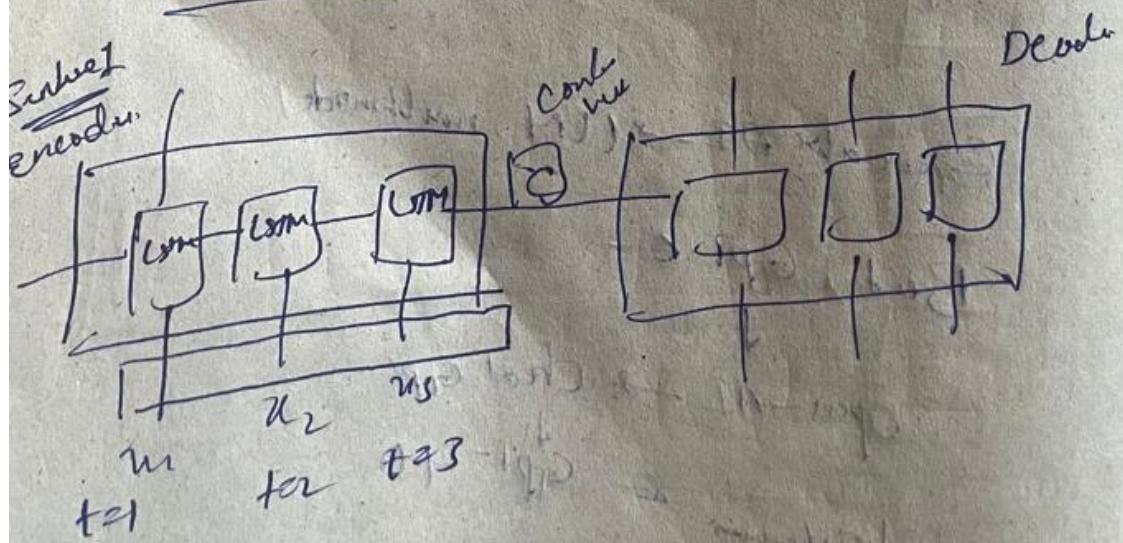
Transformer in Natural Language processing (NLP) are a type of deep learning that uses self-attention mechanisms to analyze and process natural language data. They are encoder-decoder models that can be used for many app's, including machine translation \Rightarrow Seq2seq model

e.g.: language translation \rightarrow googletrans

english \rightarrow French

if many \rightarrow off many L length of the sentence

Encoder - Decoder



if sentence is more ~~or~~ the context vector
is not able to capture the whole meaning

length of sent ↑↑

bluescore ↓↓

→ That we ~~came up~~ came up with the Attention Mechanism

+ But in attention mechanism, we cannot
possibly send all the words in a sentence.
we are going to send words based on time

stamp.

Distant → huge → selective words ^{having} ~~being~~

Transformer X LSTM X-RNN

↳ uses
self attention module

all the words will be parallelly
sent to encoder.

Position encoding.

Transform QT dataset \rightarrow imaging \rightarrow NLP
only

Transfer learning \rightarrow multi model task \rightarrow

NLP + Image

Transformers

A2 space \rightarrow SOTA model \rightarrow

Bert Gpt \rightarrow by using transfer learning

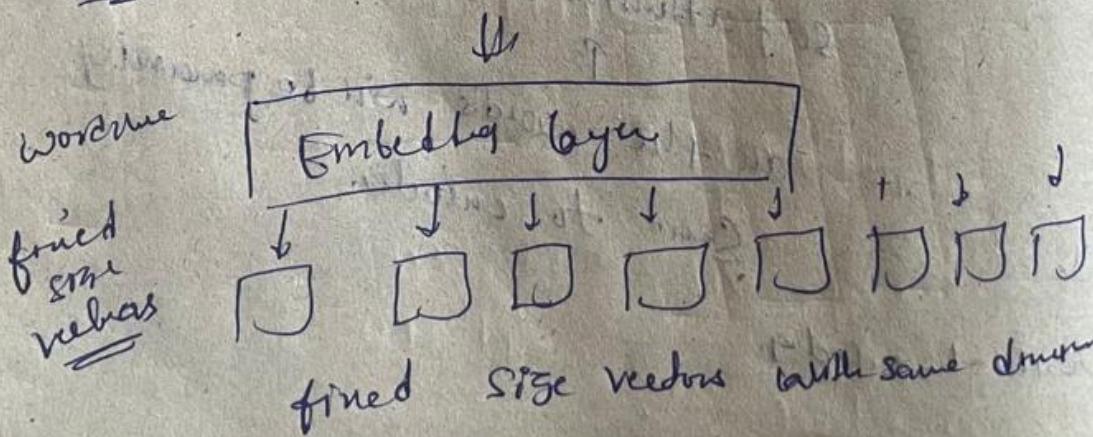
Train huge
data.

SOTA models
Can make

State of art models

② Contextual embedding:

e.g. my name is Brahma, and I play cricket



Contextual vectors

Contextual vector embedding

I I I I I I I I

We are going to get the vectors by carrying
the relationship with other words in the
sentence.

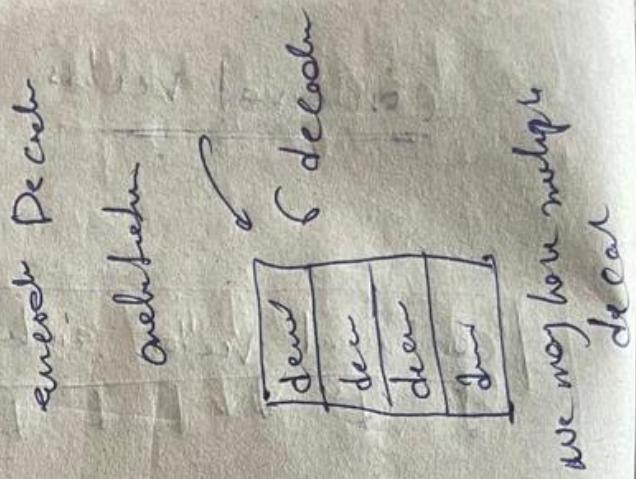
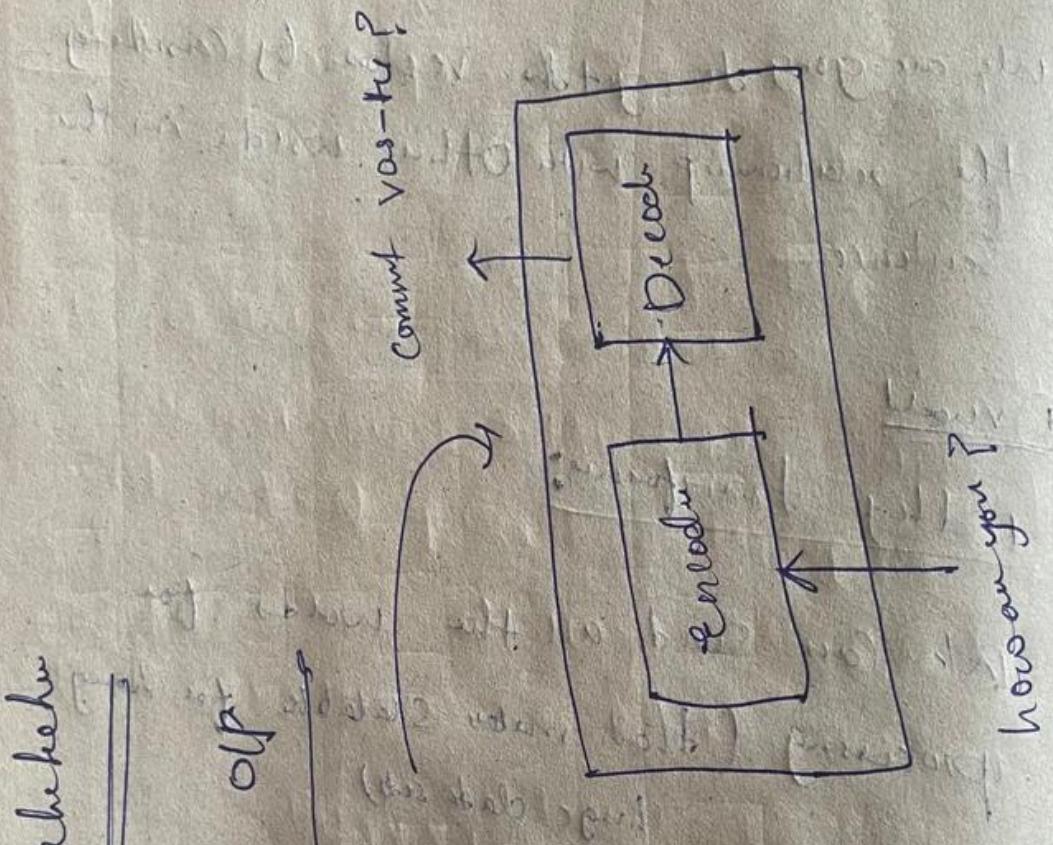
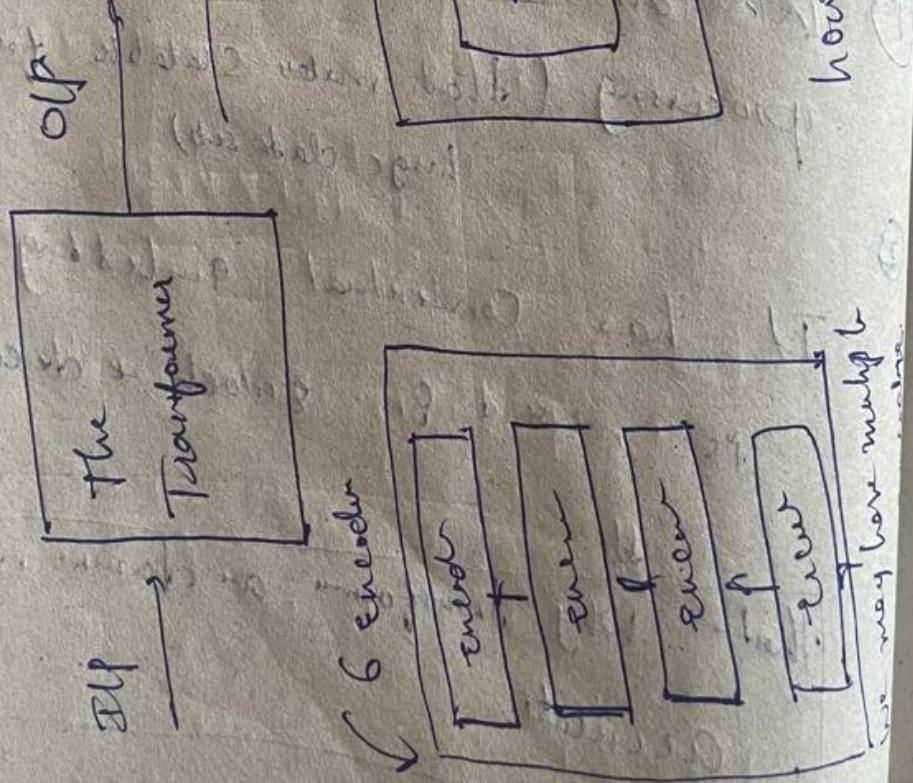
Overall

why transformer?

- ① We can send all the words for processing (that makes suitable for huge data sets)
- ② If has Contextual embedding which is passed on through a decoder.

That's why transformers are damn more accurate

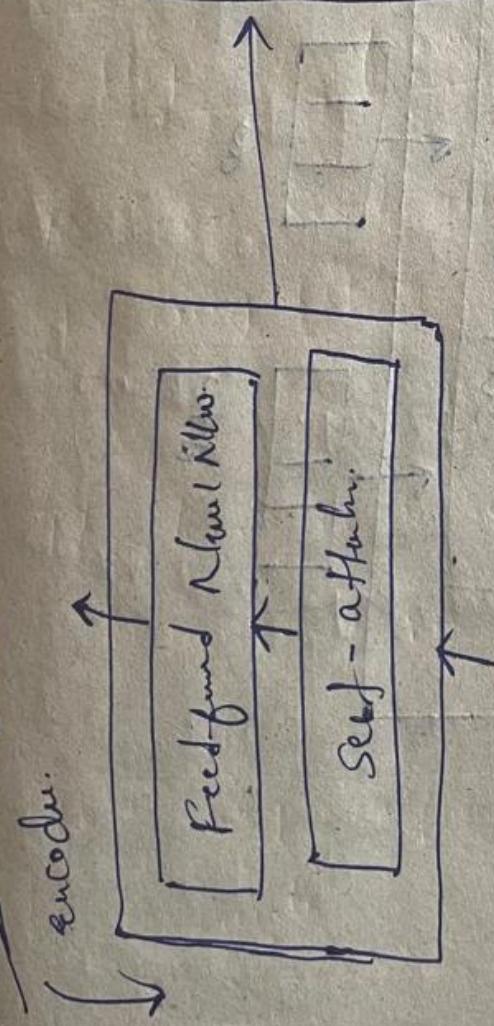
Transformers architecture



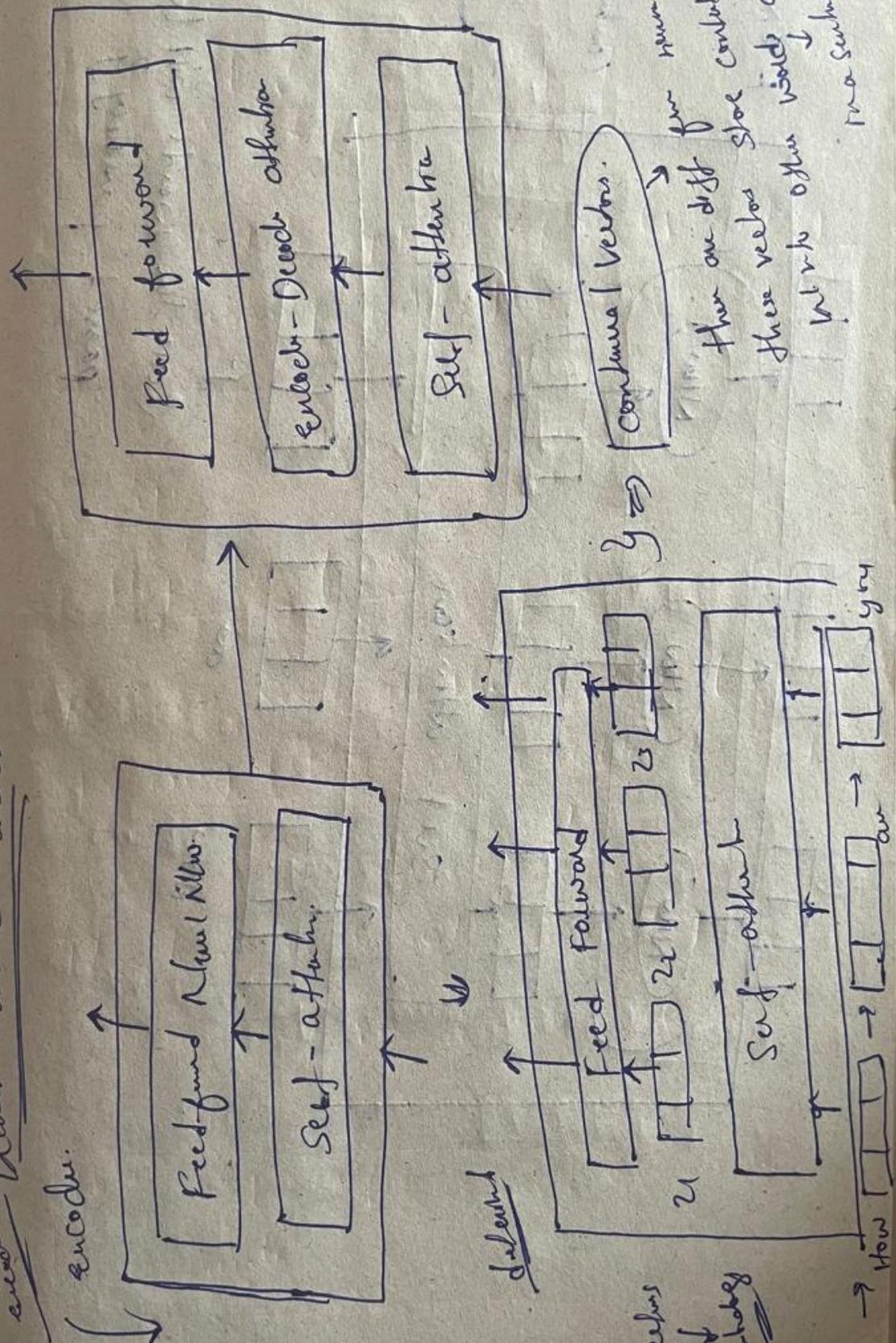
We may have multiple
decoder

case 1,

Singel 102 Valka wurde anerob.



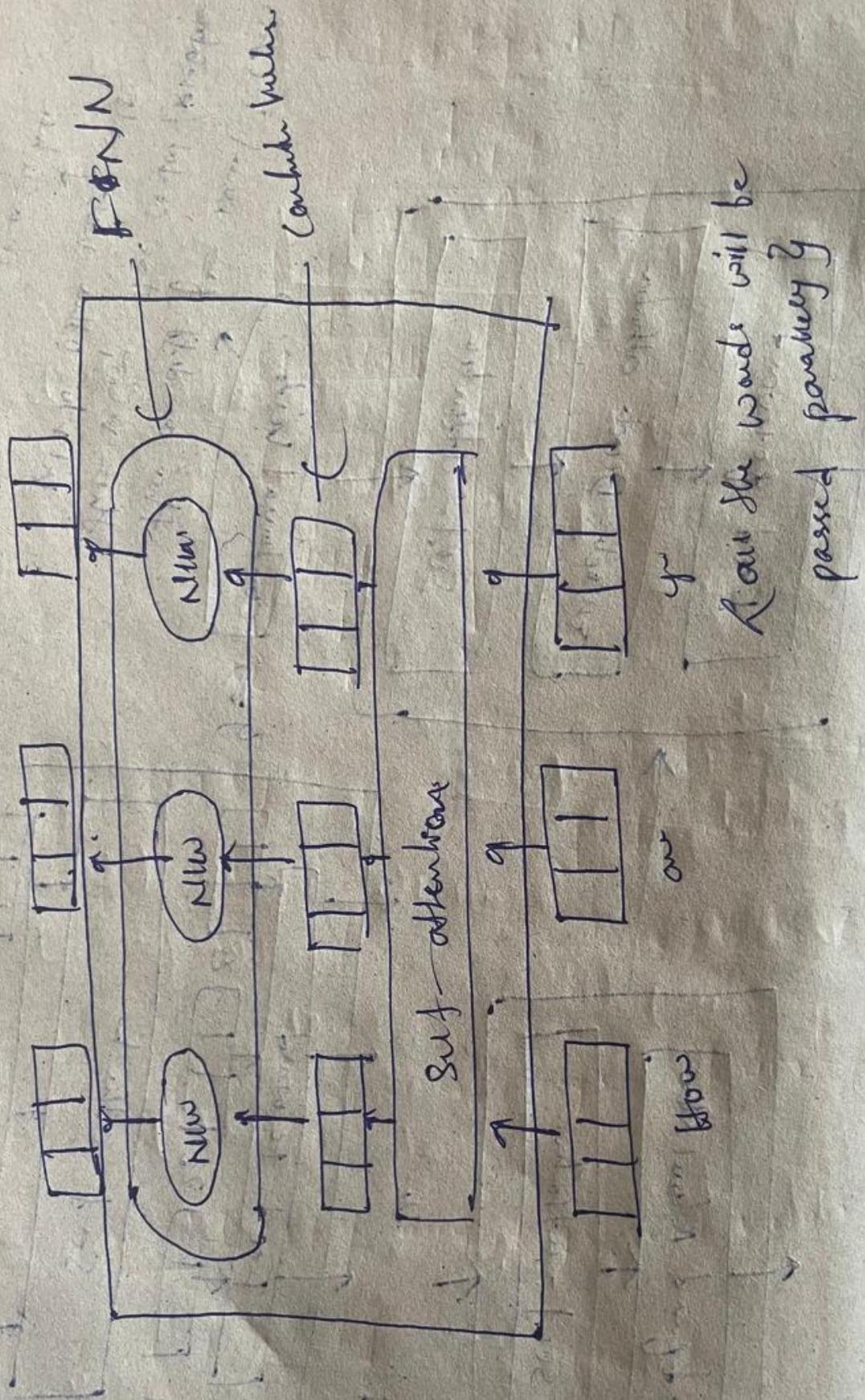
en codice



Venus

→ from new velocities
than one drift from constant acceleration
these velocities also involve other variables
like $s = ut + \frac{1}{2}at^2$

→ from new velocities
than one drift from constant acceleration
These velocities show violations also
with other violations →
no such



How the words will be
passed paravety

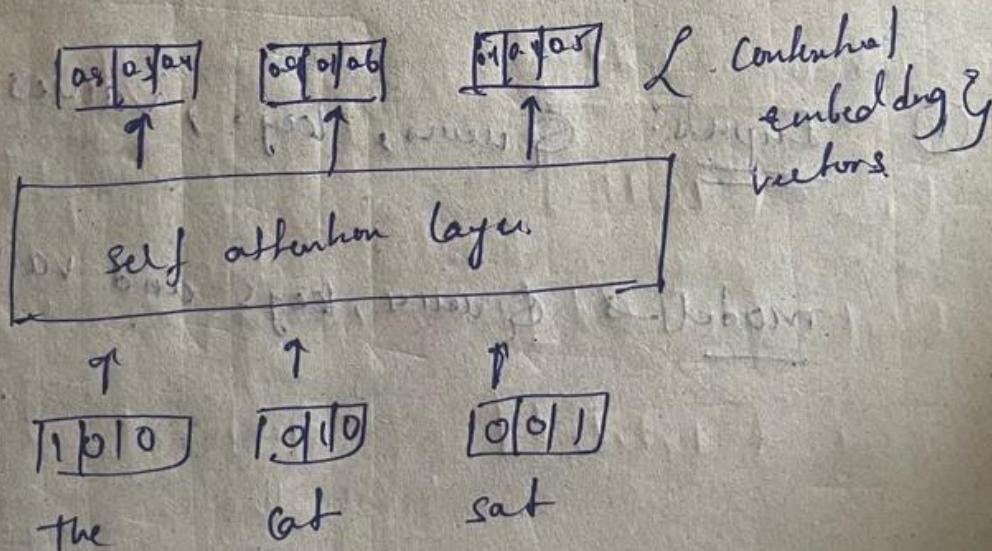
Self attention layer, involving

Self attention at high and detailed level

Self-attention, also known as scaled dot-product attention, is a general mechanism in the transformer architecture that allows the model to weigh the importance of different tokens in the input sequence relative to each other.

Idea

here we are going to take the importance of different tokens and create the vector



Value Vectors (V):

These vectors hold the actual information that will be aggregated to form the output of the attention mechanism importance:

→ Information aggregation

→ Content preservation.

$$\text{input query} = [\text{"The"}, \text{"cat"}, \text{"SAT"}]$$

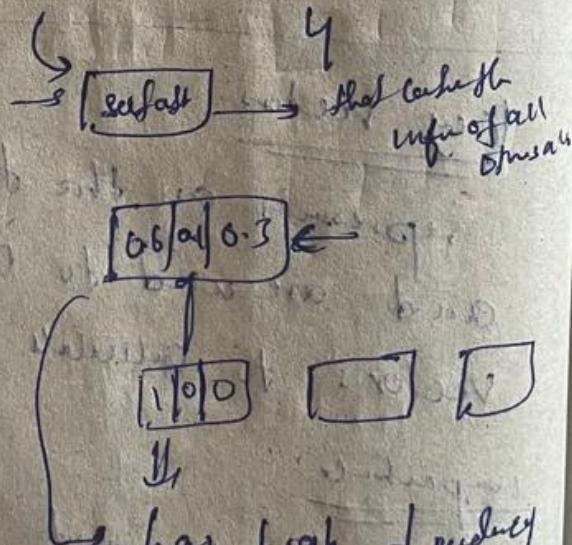
embedding size = 4

① Token embedding

$$E_{\text{the}} = [1, 0, 1, 0]$$

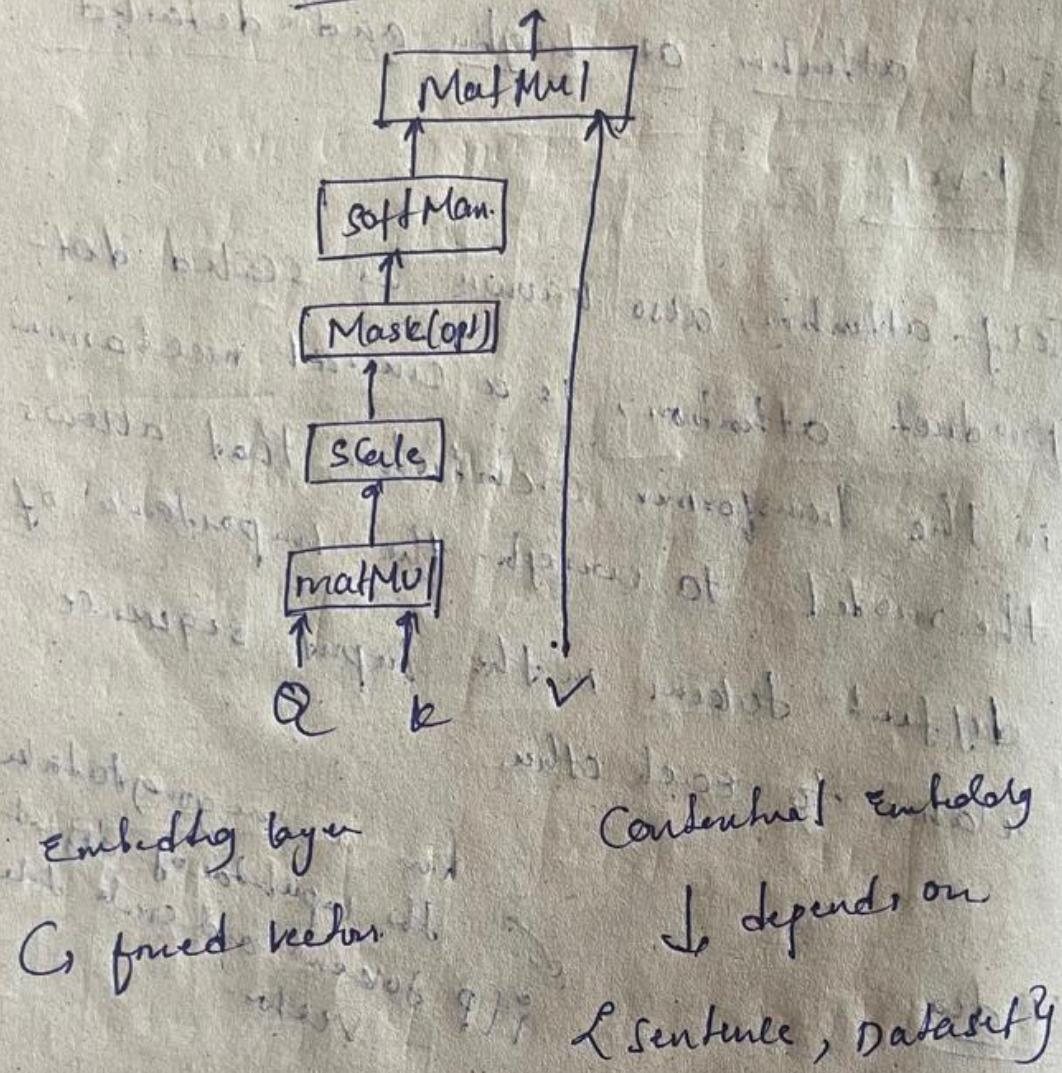
$$E_{\text{cat}} = [0, 1, 0, 1]$$

$$E_{\text{sat}} = [1, 1, 1, 1]$$



has high dependency
on Sentence, Dataset

Scaled dot product attention



① Inputs: Q uers, keys , values

model → Q uers, keys and values.

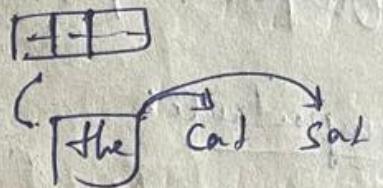
Query vectors (Q):

represents the token for which we are calculating the attention. They help determine the importance of other ~~word~~ tokens in the context of the current token.

importance:

- Focus Determination
- Contextual understanding

↓ Concrete
relationship b/w the current & token.



model will able to
know how much
attention to give on the
other tokens relative
to this token

rest of the sequence, which is essential for
Capturing dependency and context

Key vectors:

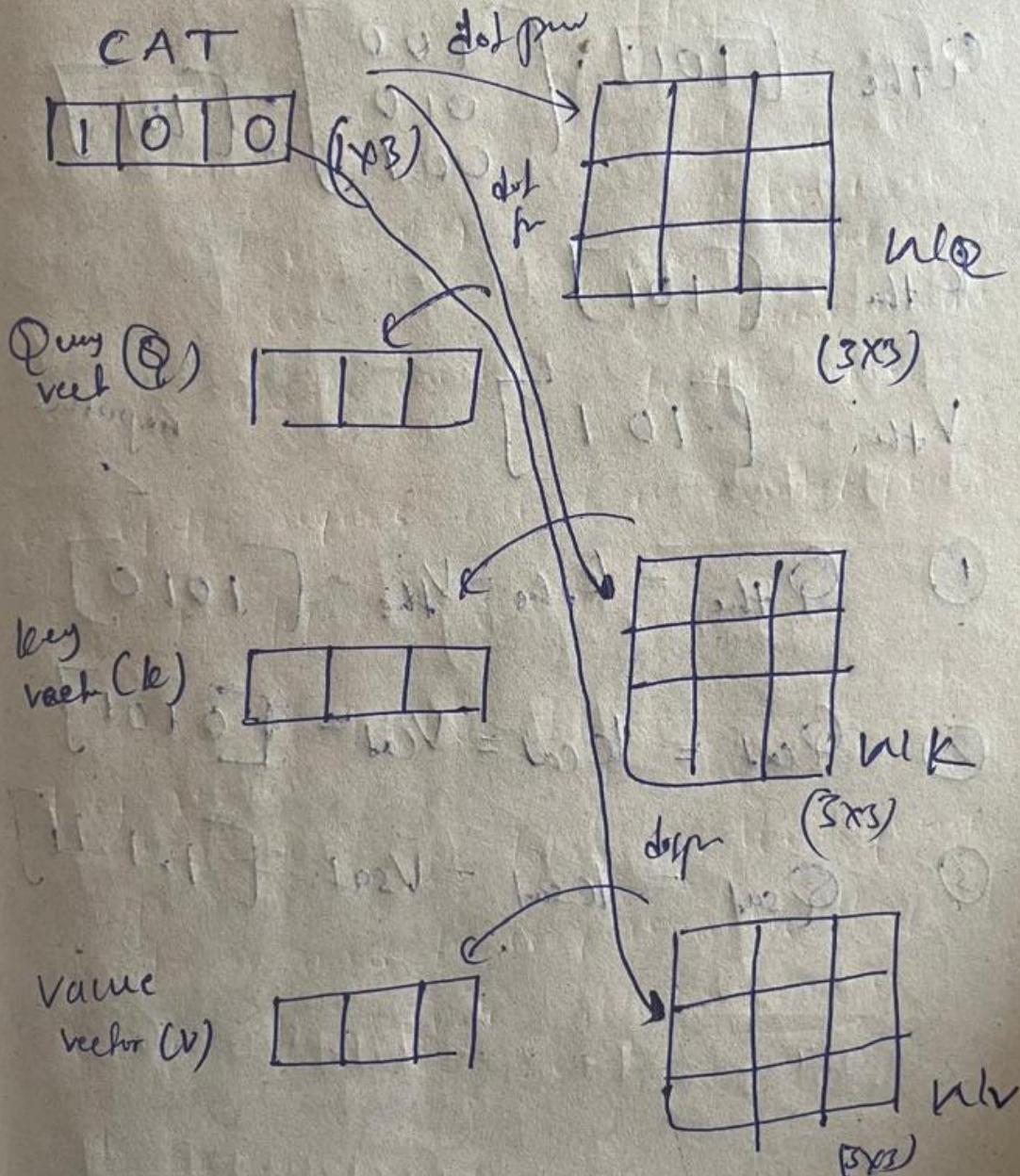
represents all the tokens in the sequence
and are used to compare with the query
vectors to calculate attention scores.

importance:

- Relevance Measurement
- Information Retrieval.

② Linear Transformation

we create $Q_k v$ by multiplying
the embedding by learned weights
matrices W_Q , W_K and W_V .



practically long

$$w_Q = w_K = w_V = I$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Q_{\text{The}} = [101] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [101]$$

$$k_{\text{the}} = [101]$$

$$V_{\text{the}} = [101] \quad \text{no path}$$

$$\textcircled{1} \quad Q_{\text{the}} = k_{\text{the}} = V_{\text{the}} = [1010]$$

$$\textcircled{2} \quad Q_{\text{cat}} = k_{\text{cat}} = V_{\text{cat}} = [0101]$$

$$\textcircled{3} \quad Q_{\text{sat}} = k_{\text{sat}} = V_{\text{sat}} = [1, 1, 1]$$

Compute attention scores for each token

Calculated by doing dot product of

Query vector with all key vectors.

for the token

$$[1010]^T = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{Score } (Q_{\text{the}}, K_{\text{the}}) = [1010] \cdot [1010]^T = 2$$

$$\text{Score } (Q_{\text{the}}, K_{\text{cat}}) = [1010] \cdot [0101]^T = 0$$

$$\text{Score } (Q_{\text{the}}, K_{\text{sat}}) = [1010] \cdot [1111]^T = 2$$

① For 'Cat' token:

$$\text{Score } (Q_{\text{cat}}, K_{\text{the}}) = [0101] \cdot [1010]^T = 0$$

$$\text{Score } (Q_{\text{cat}}, K_{\text{cat}}) = [0101] \cdot [0101]^T = 2$$

$$\text{Score } (Q_{\text{cat}}, K_{\text{sat}}) = [0101] \cdot [1111]^T = 2$$

② for the token sat

$$\text{Score } (Q_{\text{sat}}, K_{\text{the}}) = [111111] \cdot [1010]^T = 2$$

$$\text{Score } (Q_{\text{sat}}, K_{\text{cat}}) = 2$$

$$\text{Score } (Q_{\text{sat}}, K_{\text{sat}}) = 4$$

with scaling:

$$d_k = 4$$

$$\sqrt{d_k} = 2$$

① Compute scaled Dot product

$$[6, 4] \rightarrow \text{scale} \left(\frac{6}{2}, \frac{4}{2} \right) \\ = [3, 2]$$

$\uparrow \uparrow$

$$\text{softmax} \left([3, 2] \right) = \begin{bmatrix} \frac{e^3}{e^3 + e^2}, \frac{e^2}{e^3 + e^2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{1+e^{-1}}, \frac{1}{1+e^1} \end{bmatrix}$$

$$= [0.73, 0.27] \xrightarrow{\text{attention weights.}}$$

Now here we can see a very ~~less~~ ~~less~~ smooth curve, when we are back propagating, weight are going to update in very smooth manner, ~~so~~ no problem of Vanshing Gradient

Descent problem

** often, attention weights are more balanced compared to Unscaled Case.

Without scaling: $\begin{bmatrix} 0.88, 0.12 \end{bmatrix}$ Unbalanced weights
 $\begin{bmatrix} 0.99, 0.01 \end{bmatrix}$

With scaling: $\begin{bmatrix} 0.73, 0.27 \end{bmatrix}$ balanced weights

why $\sqrt{\sum k_i^2}$ key dimension
↓
based on variance.

$$\boxed{\text{dimensions} \uparrow \quad \text{variance} \uparrow}$$

⊕ Scaling: $\sqrt{\sum k_i^2} = \sqrt{4} = 2$

$$\text{scaled score } (\alpha_{\text{the}}, k_{\text{the}}) = 2/2 = 1$$

$$\text{scaled score } (\alpha_{\text{the}}, k_{\text{sat}}) = 0/2 = 0$$

$$\text{scaled score } (\alpha_{\text{the}}, k_{\text{sat}}) = \frac{2}{2} = 1$$

→ Similarly, story can be done for
all other tokens

⑥

apply Softmax

$$\text{attention weights} = \text{Softmax}([1, 0, 1])$$
$$[\text{the}] = [0.4223, 0.1554, 0.4223]$$

$$\text{attention weights}_{\text{Cat.}} = \text{Softmax}([0, 2, 2]).$$

$$= [0.1554, 0.4223, 0.4223]$$

$$[\text{I}]$$

$$\text{attention weights}_{\text{Sat.}} = \text{Softmax}([2, 2, 4])$$

$$[\text{I}] = [0.2119, 0.2119, 0.565]$$

$$[\text{I}] = [0.2119, 0.2119, 0.565]$$

$$[\text{I}] = [0.2119, 0.2119, 0.565]$$

⑥ Calculated Sum of Values

We multiply the attention weights by corresponding value vectors for the token.

$$\underline{v_{the}} = \underline{v_{sat}} \times \underline{v_{cat}}$$

$$\begin{aligned}
 \text{Output}_{(the)} &= 0.4223 * v_{the} + 0.1554 * v_{cat} + \\
 &\quad 0.4223 * v_{sat} \\
 &= 0.4223 [1, 0, 1] + 0.1554 [0, 1, 0] \\
 &\quad + 0.4223 [1, 1, 1] \\
 &= [0.4223, 0, 0.4223, 0] + [0, 0.1554, 0, 0.1554] + \\
 &\quad [0.4223, 0.4223, 0.4223, 0.4223] \\
 &= \boxed{[1.2669, 0.9999, 1.2669, 0.9999]}
 \end{aligned}$$

Contextual vector.

self ↑ other han

initially The [1 0 1 1 0]

Mathematics in self attention

→ Calculated Q, K, V values

Initialized $\text{W}_Q, \text{W}_K, \text{W}_V$

→ attention score.

→ Scaled the values.

→ Softmax.

→ weighted sum of Values

(Softmax \times V)

Multi head Attention:

self attention with multiple heads

Jabammar.github.io.

blog

$$\begin{matrix} X \\ \text{the input} \end{matrix} \times W_Q = Q \text{ (query)}$$

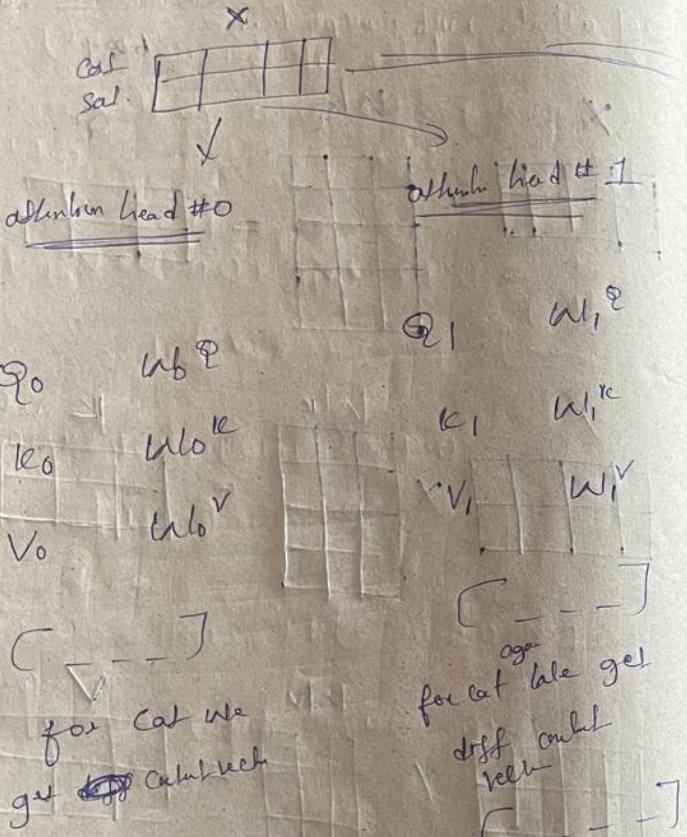
$$X \times W_K = K \text{ (key)}$$

$$X \times W_V = V \text{ (value)}$$

$$\text{Softmax} \left(\frac{Q \times K^T}{\sqrt{dk}} \right) \cdot V$$

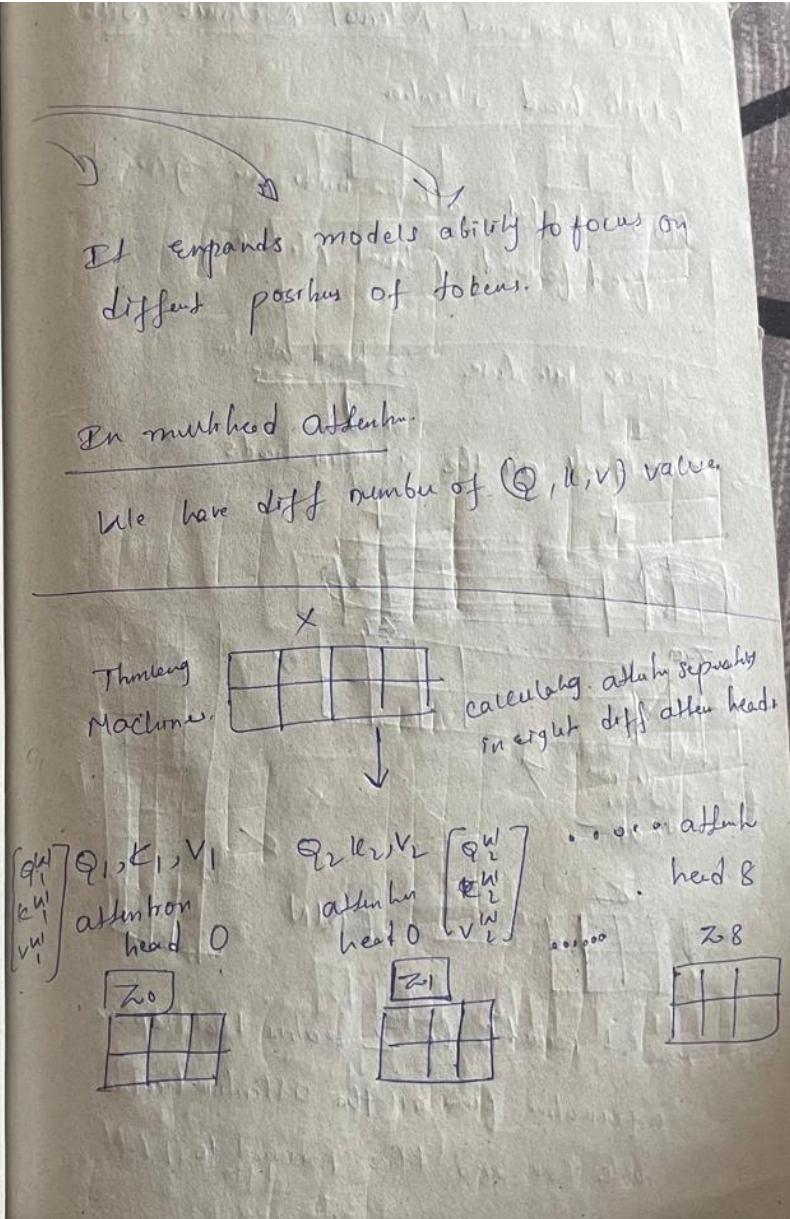
→ scaling.

$$\text{OUP} = \sum Z \leftarrow \text{attention head}$$



Simply: Multi-head att

Same
 for ~~single~~ word (token) case we are going to
 get multiple attention heads (context vector)
 with diff. values (Q, K, V)



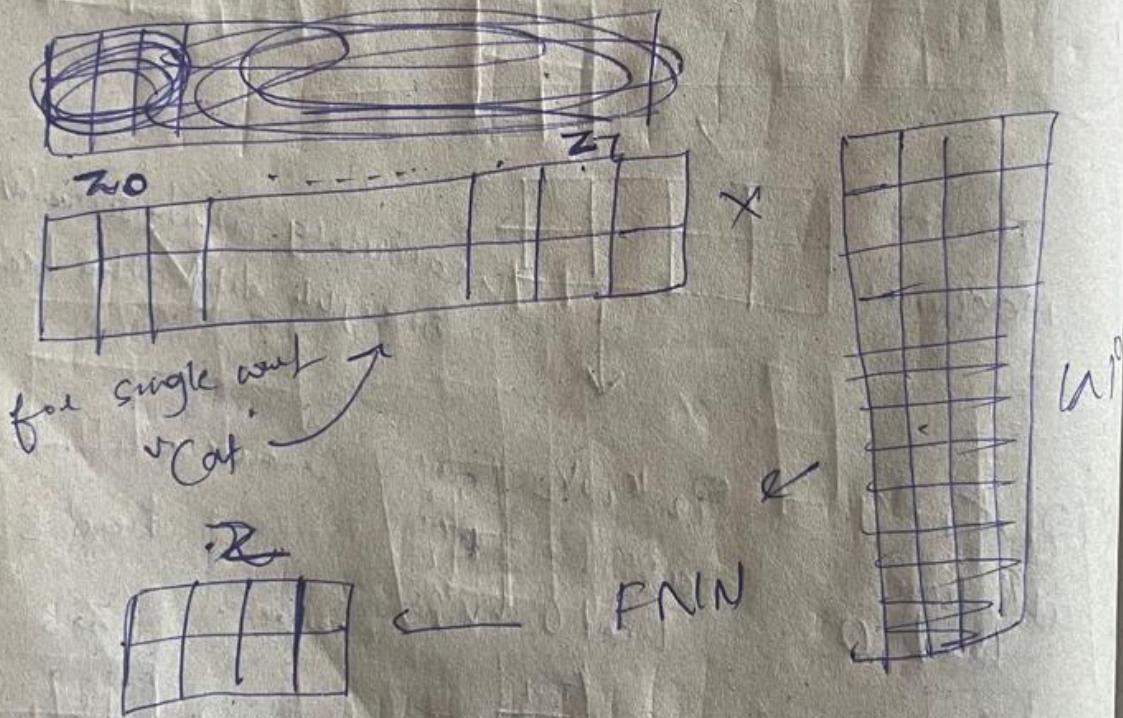
Feed Forward Neural Network with

Multi head attention

Now different attention heads are given to feed forward Neural Network

Suppose we have 8 head attention

→ Concatenates all the attention heads



→ result won't be the $2 \times m$ matrix that captures information from all the attention heads. We can send this forward to FNN

Positional Encoding - Representing order of

Sequence.

adv of using transform

- ① Input tokens it can process parallelly.



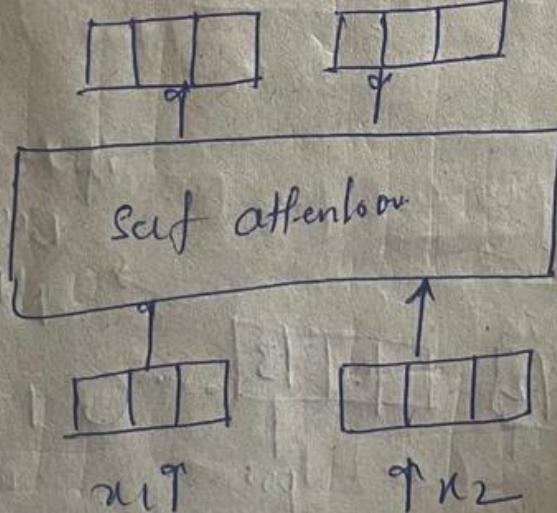
Draw facts

Lack of sequential structure of the words.
Order

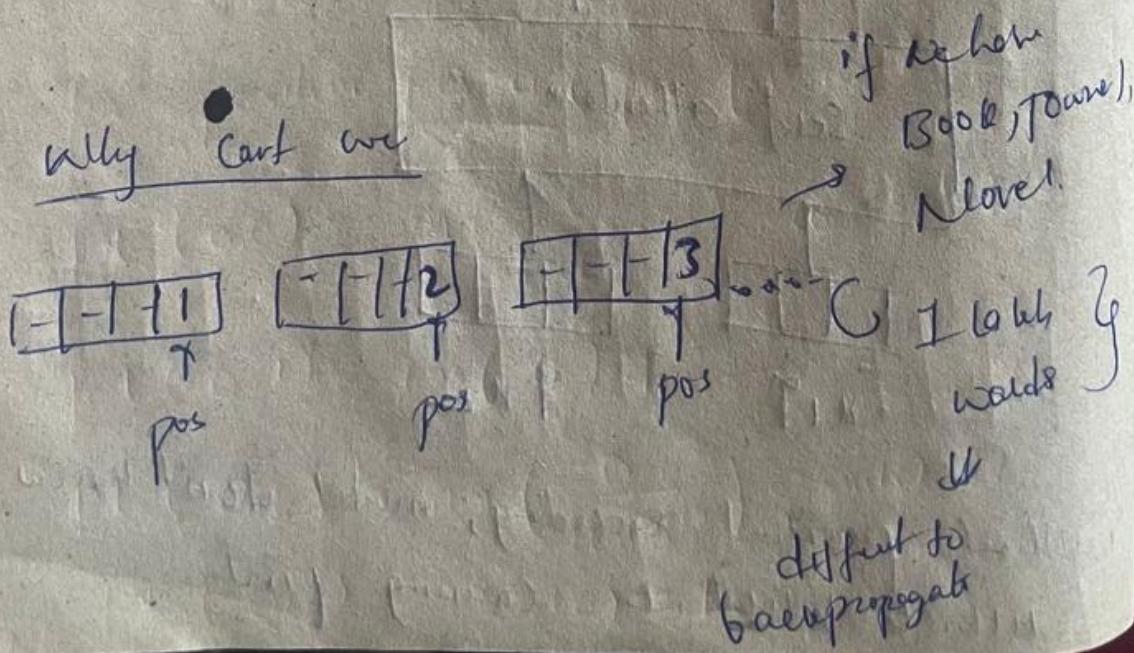
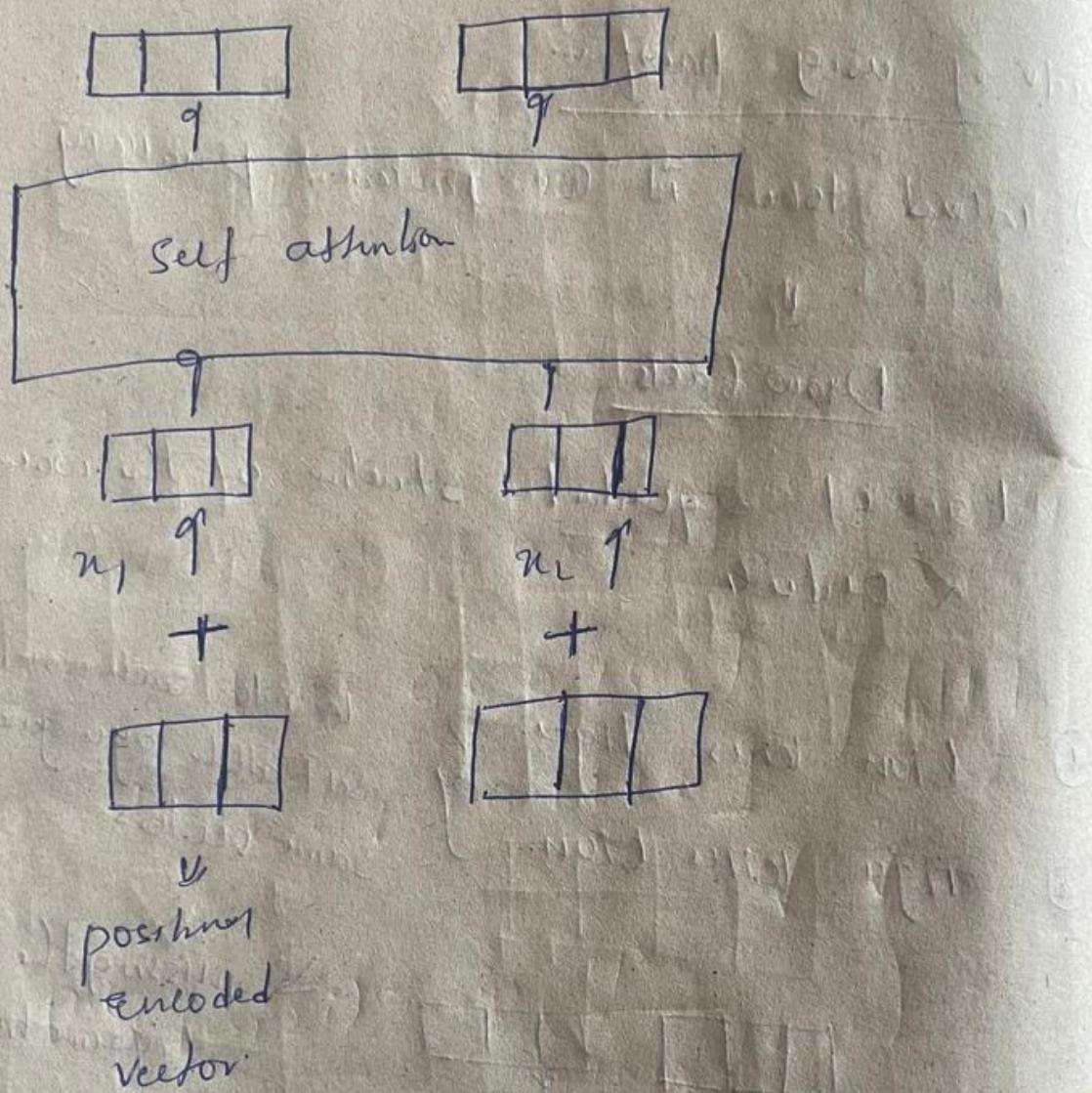
- ① Lion kills tiger }
- ② Tiger kills lion.

W.r.t each word
Self attention generate
Same vectors.

→ missing the
information



Without position encoding, model don't know which token is coming first.



How do we create positional encoding vectors.

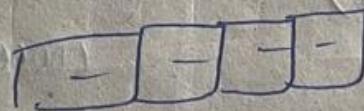
Types of position encoding

- ① Sinusoidal position encoding
- ② Learned positional encoding.

① Sinusoidal positional encoding:

If uses Sine and cosine functions of different frequencies to create positional encodings.

Formula:



$$P.E(pos, z^i) = \sin \left(\frac{pos \cdot z^i}{10000} \right)$$

$$P.E(pos, z^{i+1}) = \cos \left(\frac{pos \cdot z^{i+1}}{10000} \right)$$

later

pos is the position of word
i is the dimension

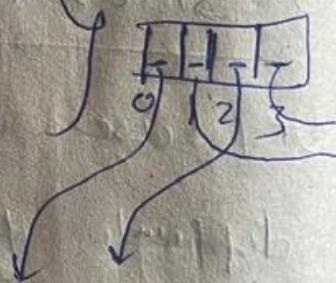
dmodel is the dimensionality
of the embedding

Eg: The Cat Sat

The $\rightarrow [0.1 \ 0.2 \ 0.3 \ 0.4]$ } embedding vectors.

Cat $\rightarrow [0.5 \ 0.6 \ 0.7 \ 0.8]$

Sat $\rightarrow [0.9 \ 1.0 \ 1.1 \ 1.2]$



$$P.E(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

for position pos=0 (for first word). Let say

$$P.E(0,0) = \sin\left(\frac{0}{10000^{2*0}}\right) - \textcircled{2} = \sin(0) = 0,$$

$$P.E(0,1) = \cos(0) = 1$$

$$P.E(0,2) = \sin\left(\frac{0}{10000^{2*2}}\right) = \sin(0) = 0$$

$$P.E(0,3) = \cos(0) = 1$$

for pos1 (cos)

$$P.E(1,0) = \sin\left(\frac{1}{10000^{2/4}}\right) = \sin(1) = 0.8415$$

$$P.E(1,1) = \cos\left(\frac{1}{10000^{2/4}}\right) \approx 0.5403$$

$$P.E(1,2) = \sin\left(\frac{1}{10000^{3/4}}\right) \approx 0.01$$

$$P.E(1,3) = \cos\left(\frac{1}{10000^{3/4}}\right) \approx 0.99995$$

0.8415	0.5403	0.01	0.99995
Position Encoding			

 \Rightarrow

0.5	0.6	0.7	0.8
Cat embedding			

alternatively we should

calculate:



$$P.E(pos, 2^{j+1}) = \cos\left(\frac{pos}{10000^{2^{j+1}/4}}\right)$$

The

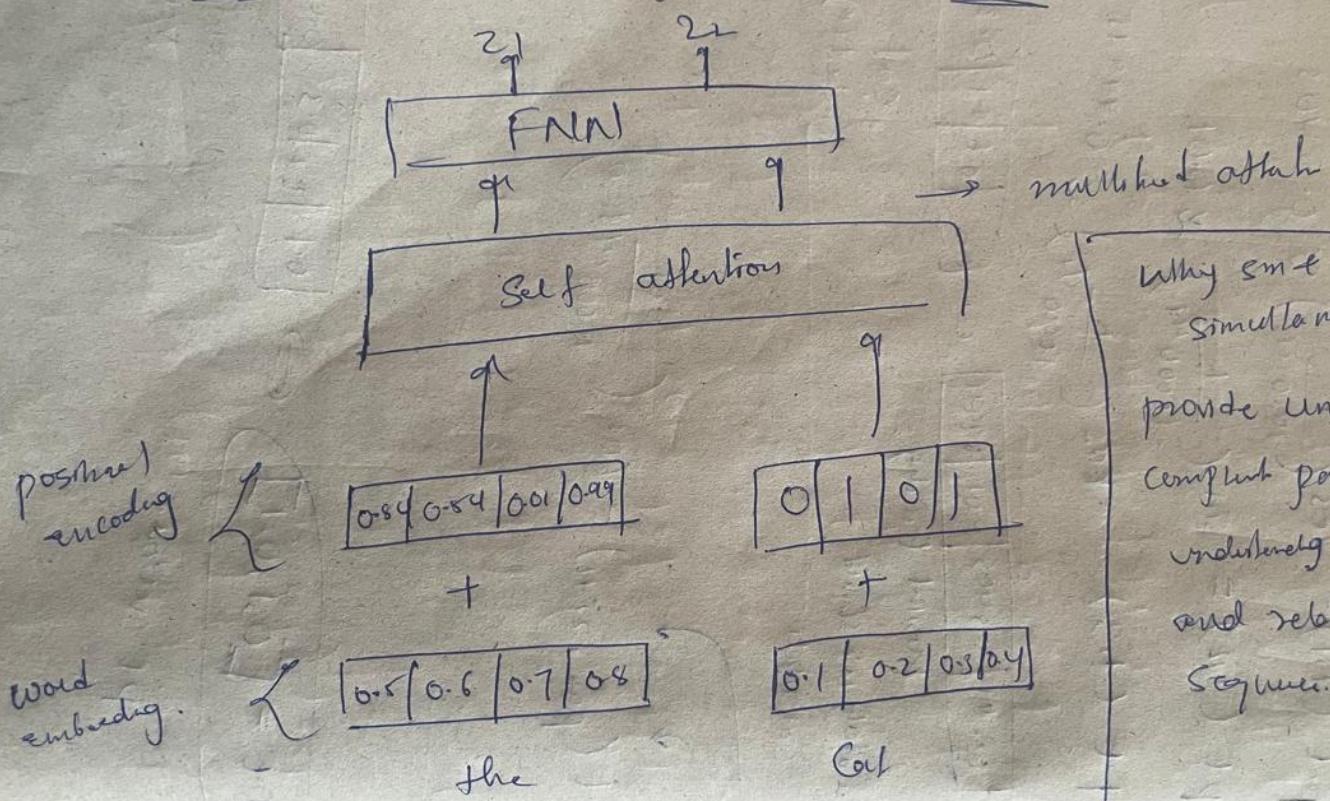
$$P.E = [0, 1, 0, 1]$$

PE vector

$$\Rightarrow [0.1 | 0.2 | 0.3 | 0.4] \text{ The}$$

embeded
vec

This $\xrightarrow{}$ is how we give to self attention layer

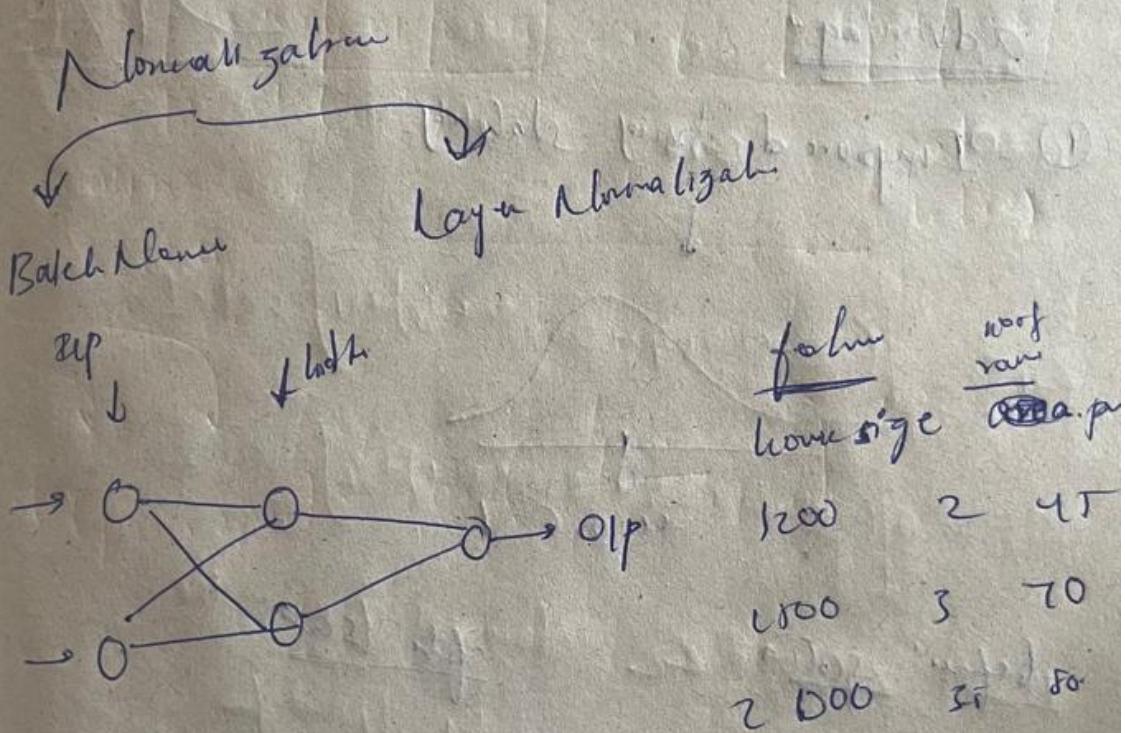


Why same is used
simultaneously.

provide unique, small and
complimentary paths for
understanding both absolute
and relative position in
sequence.

Layer Normalization Dr. Hanfoune

- ① self atten. layer
- ② multi head attention
- ③ position encoding
- ④ Layer Normalization (add and normalize)

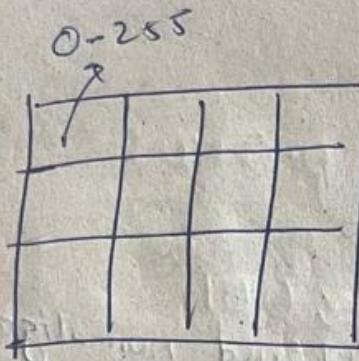


Normaliz. Standard scores

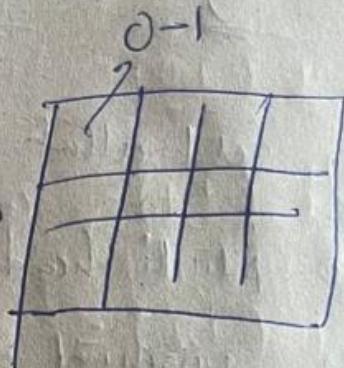
$$Z_{i,m} = \frac{x_i - \mu}{\sigma}$$

after apply to $f_i \xrightarrow{\text{standard}} f'_i$ where $\mu=0, \sigma=1$

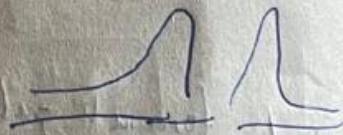
Deep learning: if P image \Rightarrow min max scalar



\Rightarrow min max scaling

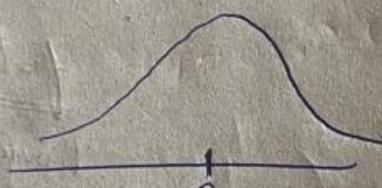


Specifically do for input data



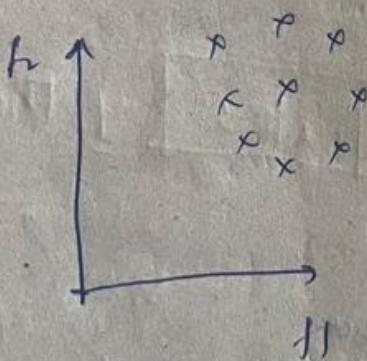
advantages

- ① Improved training stability.

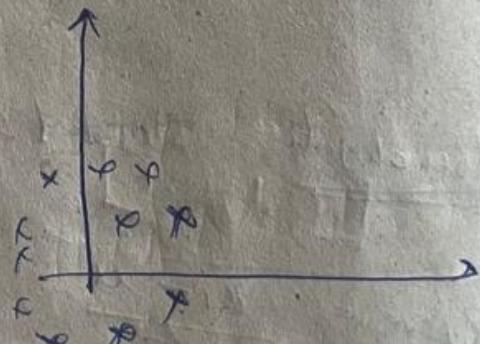


$$\mu=0, \sigma=1$$

before scale

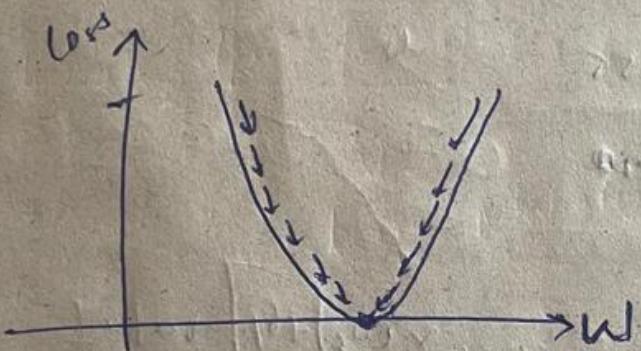


after scale



→ Because of this we are not going to face
Vanishing and Exploding gradient problem.
during backpropagation

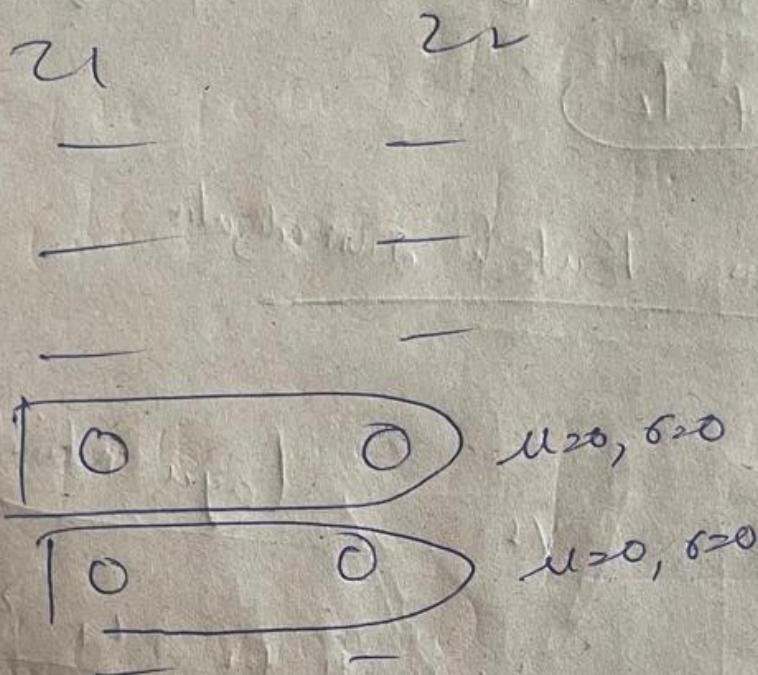
② faster Convergence



Backpropagation state update:

Summary of applying Batch normalization

There is no input



Normalized

\hat{u}

γ, β

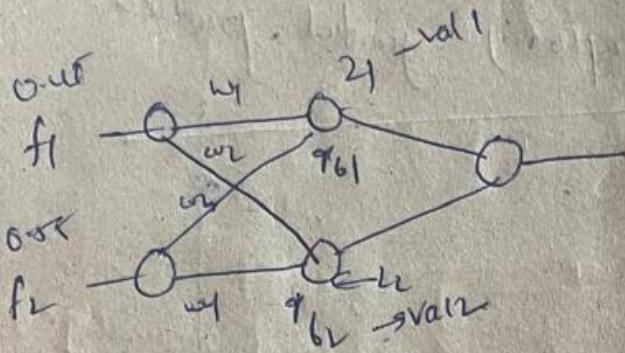
$$z_1 = \sigma(w_1^T y + b_1)$$

$$y = \frac{\gamma}{\sigma} \left[\frac{z_1 - \mu_1}{\sigma} \right] + \beta$$

Learnable parameter

if you don't want to normalize, then by using this pair (γ, β) we say no so do not normalize

Batch normalization



$$\begin{array}{ll}
 \text{Scaled } f_1 & \text{Scaled } f_2 \\
 \text{here size}(f_1) & \text{Rooms}(f_2) \\
 \hline
 0.45 & 0.55 \\
 0.60 & 0.40 \\
 \vdots & \vdots
 \end{array}
 \quad = \quad \frac{z_1}{z_2} = \frac{z_1}{z_2}$$

$$z_1 = [(0.45 \times w_1 + 0.55 \times w_3) + b] = \text{value 1}$$

$$z_2 = [(0.60 \times w_2 + 0.40 \times w_4) + b] = \text{value 2}$$

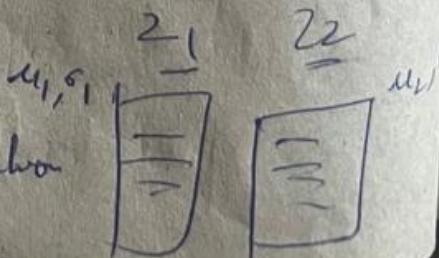
Then z_1 and z_2 don't follow the distribution flat followed by f_1 and f_2

→ So without making much importation

having further go and applying Normalisation

on every O.P.

↳ Batch Normalization



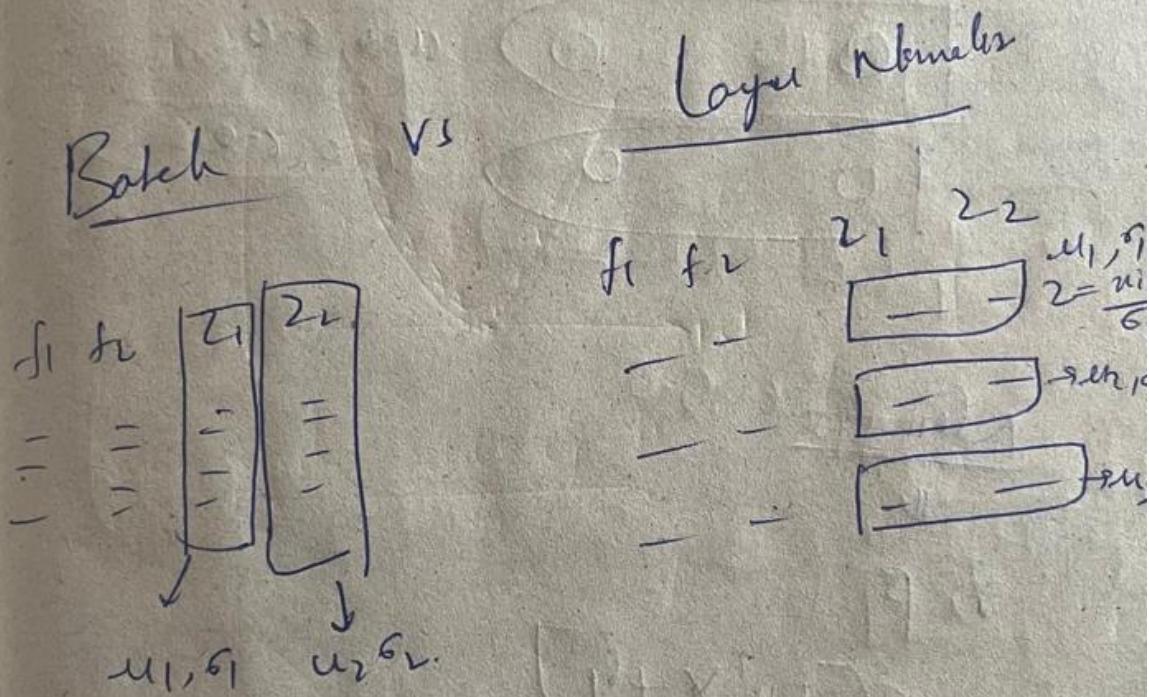
Now for z_1, z_2

$$\text{for } \mu = 0, \sigma = 1$$

that matches with one input feature

$$\mu_1, \sigma_1$$

This is Batch normalization



inp param

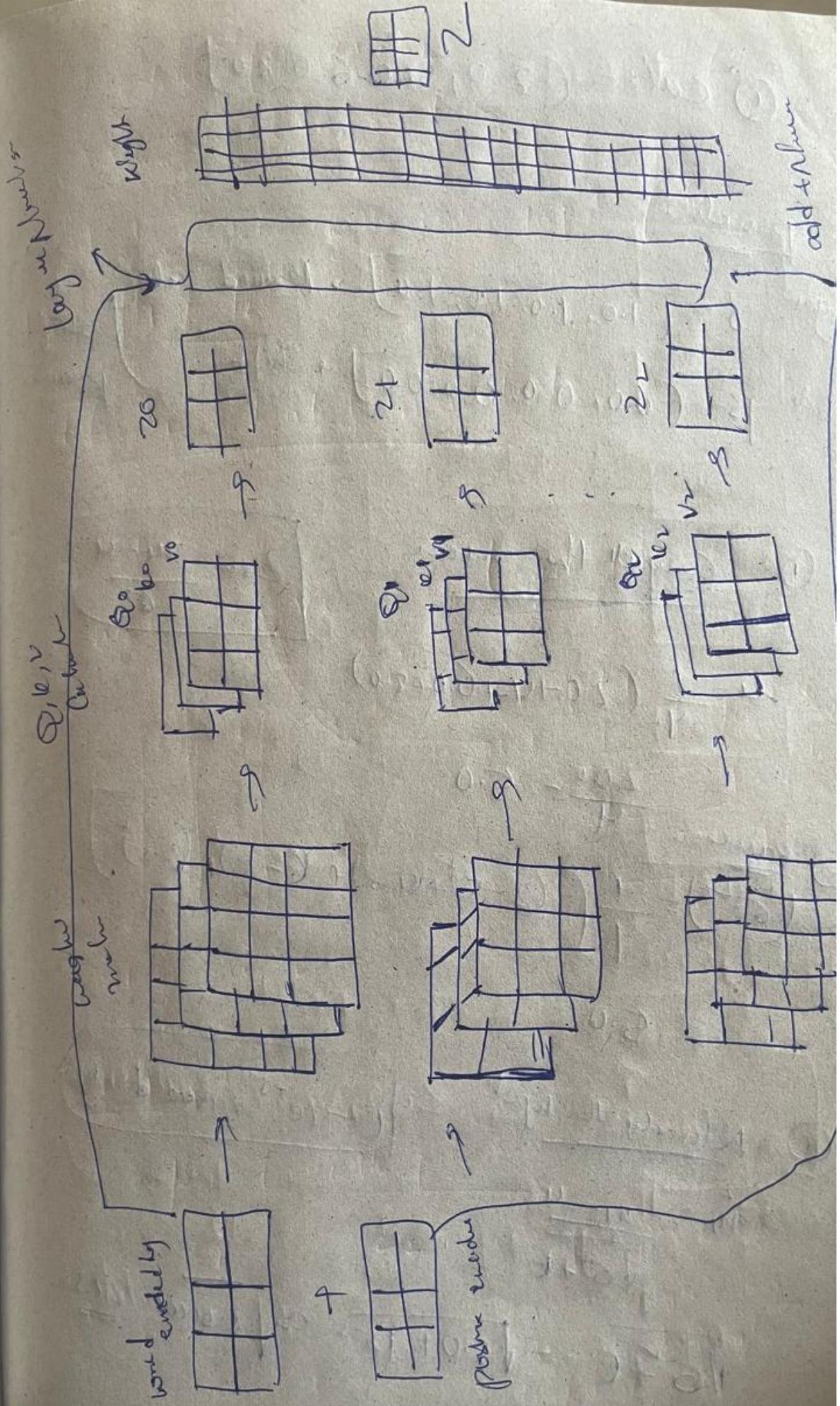
$\gamma, \beta \rightarrow$ Learnable parameters

Batch Number

	z_{11}	z_{21}
=	-	-
$\rightarrow 0$	-	-
$\rightarrow 0$	-	-
$\rightarrow 0$	-	-

after appr BN

like as going to change
zero that going to
impact on z_1



$$\textcircled{1} \quad \text{cal} = [2.0, 4.0, 6.0, 8.0]$$

Param:

given $\gamma = [1.0, 1.0, 1.0, 1.0] \rightarrow$ learned scale }
 & $\beta = [0.0, 0.0, 0.0, 0.0] \rightarrow$ shift }
 { Scale & shift
 param

\textcircled{1} Compute the mean

$$2_{\text{sum}} = \frac{n-1}{\sigma}$$

$$\mu = \frac{1}{4} (2.0 + 4.0 + 6.0 + 8.0)$$

$$= \frac{20.0}{4} = 5.0$$

\textcircled{2} vare

$$(6^2)^2 = \frac{1}{4} \left[(2-5)^2 + (4-5)^2 + (6-5)^2 + (8-5)^2 \right]$$

$$= 5.0$$

\textcircled{3} Normalise the input

$$U = (e^{-5} \rightarrow \text{avoid div by 0})$$

$$\hat{u}_i = \frac{u_i - \mu}{\sqrt{\sigma^2 + E}}$$

$$\sqrt{\sigma^2 + E} = \sqrt{5.0 + 1e^{-5}} \approx \sqrt{5.00001} \approx 2.236$$

$$\hat{x}_1 = \frac{2-5}{2.236} \approx -1.34$$

$$\hat{x}_2 = \frac{4.0-5.0}{2.236} \approx -0.45$$

$$\hat{x}_3 = \frac{6-5}{2.236} = 0.45$$

$$\hat{x}_4 = \frac{8.0-5.0}{2.236} \approx 1.34$$

Number vector = $\left[-1.34, -0.45, 0.45, 1.34 \right]$

④ scale and shift

$$y_i = \gamma_i \hat{x}_i + \beta_i \quad \gamma = [1, 1, 1, 1], \beta = [0, 0, 0, 0]$$

$$y = [-1.34, -0.45, 0.45, 1.34] \quad \text{Some disturbance}$$

if you want to
change disturb. change the γ, β
values

wide self ash

$$Q = 64 \text{ dm.}$$

$$k = 64 \text{ dm}$$

$$\sqrt{= 64 \text{ dm}}$$

$$\sqrt{64} = 8$$

why $\sqrt{25} = 5$

Segm to seg of very complex

why to add residuals, why FNIN

① residual / connections: skip Connec. NN

① addressing the vanishing gradient problem.

let there: we have 8 layers of encoder.

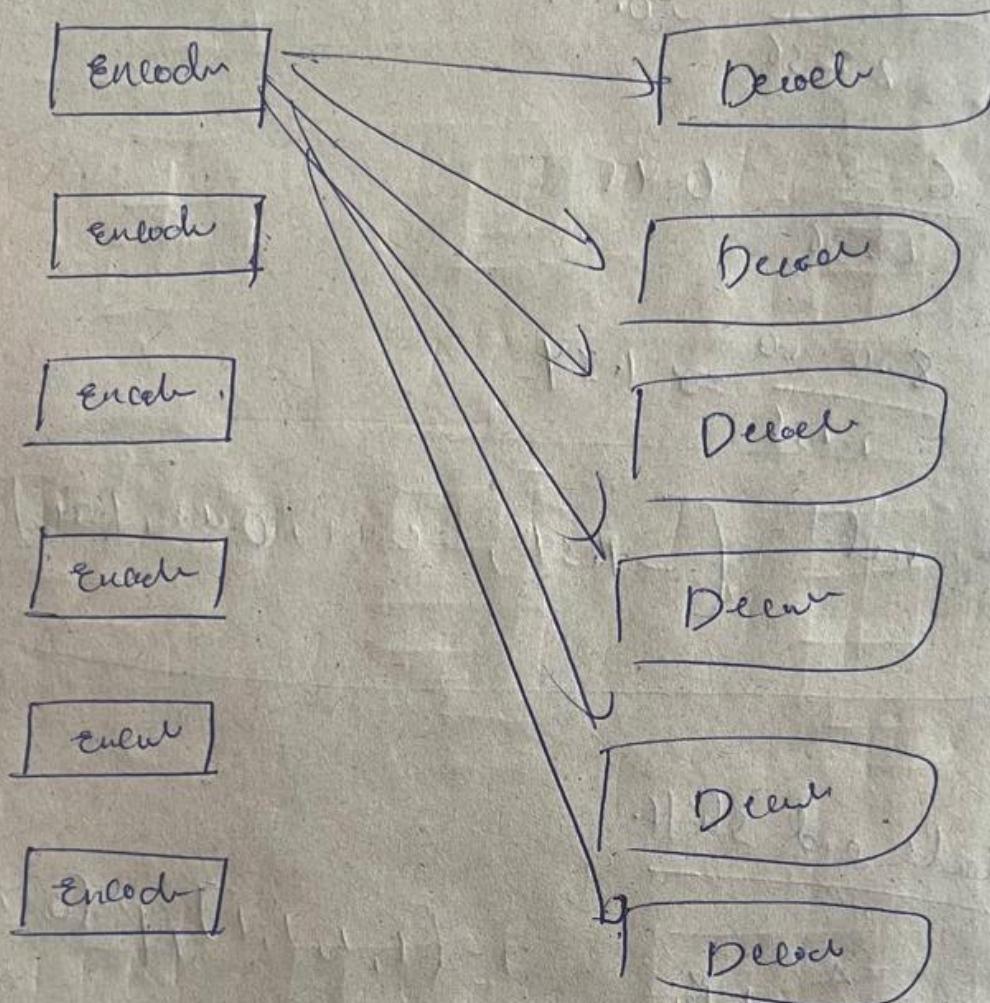
as the number of layers increase, there a
chance of ^{grad of} loss function w.r.t. wts can
be small.

how residual helps:

residual make a short paths for gradients to
flow directly through the NN. that
makes gradient norms sufficiently large.

Encoder architecture

out / I am a student



Input / jesus is student

(add and norm) \rightarrow FNN.
 multi head att \rightarrow $2_1, 2_2, 2_3, 2_4, 2_5, 2_6, 2_7, 2_8$

Residual \oplus denses \rightarrow
 Text embedding + position encoding $\Rightarrow 512$ (Repeat 12 times)
 RNN sequence

① Empire Grids plus

compute will be faster

② Enables training of deeper networks

why FFN

① adding Non-linearity.

② processing each token independently
self attn \rightarrow capture relationships

FFN \rightarrow each token represents independently

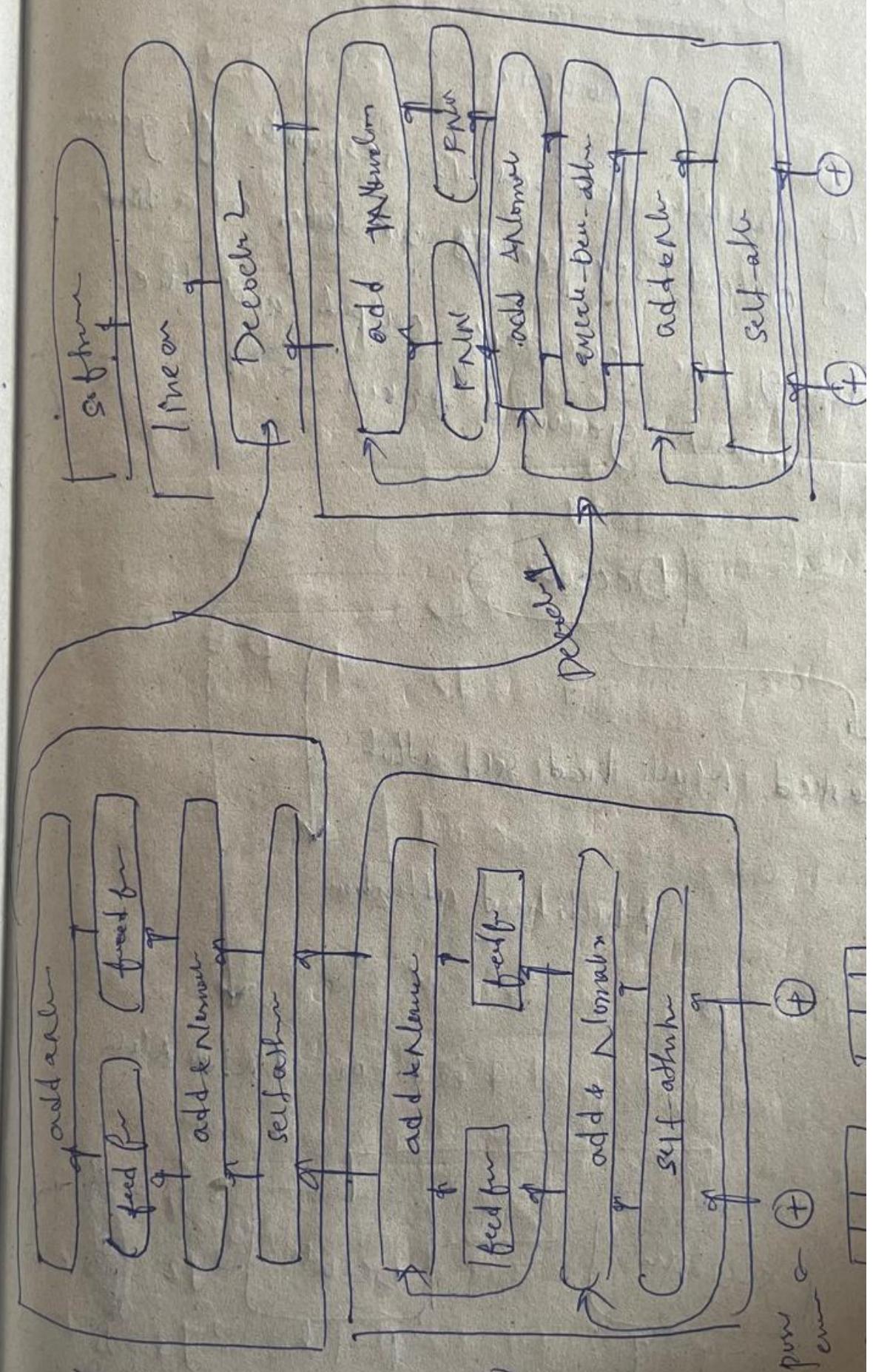
↓

Transforming these representations further
and allow the model to learn
 richer representations

① processing each token individually

② adding non-linearity

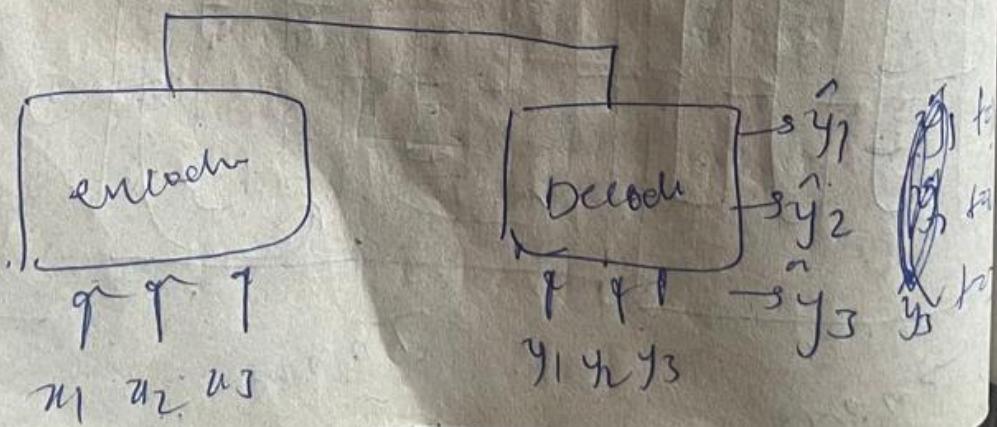
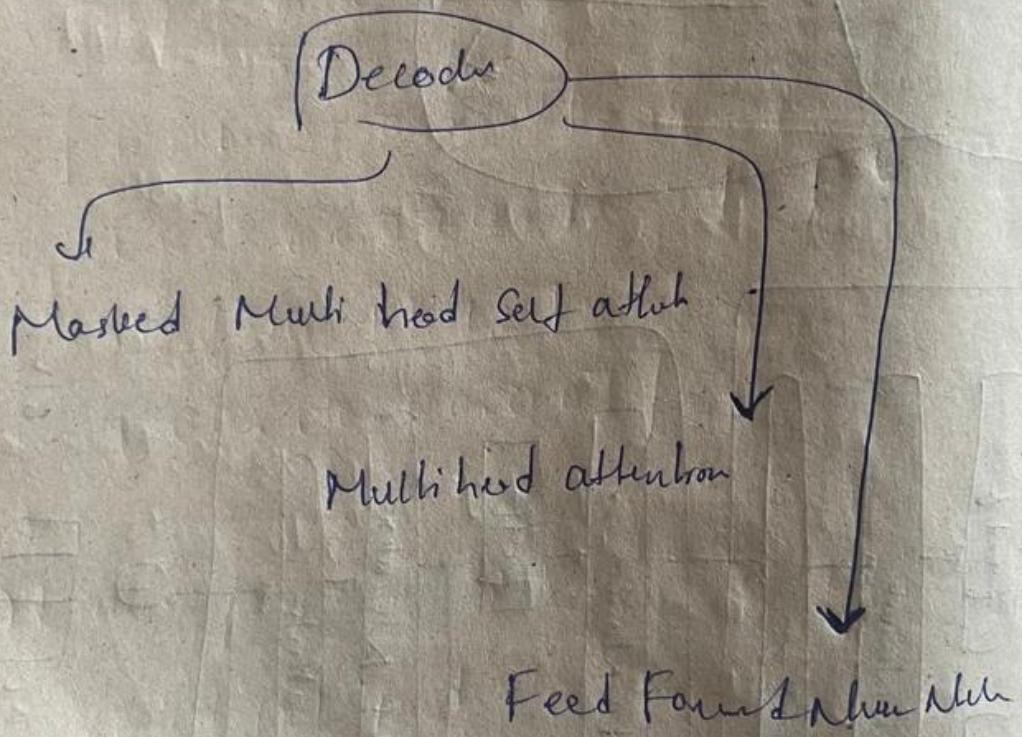
③ linear transformation



Decodes in Transfomers

3 main components

The hardware decode is responsible for generating the output sequence of one token at a time, using the encoder's output and the previously generated token.



Masked multi-head Self attention

- ① DIP embedding and positional embedding
- ② Linear projections for Q, K, V
- ③ Scaled dot product attention
- ④ Mask application
- ⑤ Multi-head attention
- ⑥ Concatenation and final linear projection
- ⑦ Residual Connection and Layer Normalization

Dataset:

eng:

$\langle x_1, x_2, x_3 \rangle$



Encoded.

hindi

$\langle y_1, y_2 \rangle \leftarrow$



$\langle y_1, y_2 \rangle$

↑
Zero padding.

Decod

Zero padding → To make our sequence ~~of~~ length equal

Masked
multi-head
attention

Q, P
 $\begin{bmatrix} \text{[4567]} \end{bmatrix}$ $\begin{bmatrix} \text{[123]} \end{bmatrix}$
 \downarrow
 $\begin{bmatrix} \text{[1230]} \end{bmatrix}$
 \downarrow
+ dim vec

① Input embeddg, and positional encodg

Output embeddg

Step 1

$\begin{bmatrix} [0.1, 0.2, 0.3, 0.4], \\ [0.5, 0.6, 0.7, 0.8], \\ [0.9, 1.0, 1.1, 1.2], \\ [0.0, 0.0, 0.0, 0.0] \end{bmatrix}$ + $P \cdot E^{= 40}$ (4×4)

① Linear projections Q, K, V

② scaled dot product attention

③ Mask application \rightarrow look-ahead mask
 \rightarrow padding mask

Step 2: Linear project for Q, K, V

Create query(Q), key(K) and value(V)
vectors.

$Q = \text{output Embedding} \times W_Q = \text{output embedding}$

$K = \text{"} \times W_K \text{"}$

$V = \text{"} \times W_V \text{"}$

$$Q = K = V = \begin{bmatrix} [0.1, 0.2, 0.3, 0.4], \\ [0.5, 0.6, 0.7, 0.8], \\ [0.9, 1.0, 1.1, 1.2], \\ [0.0, 0.0, 0.0, 0.0] \end{bmatrix}$$

③ scaled Dot product attention calculate

$$\text{Score} = Q \times K^T / \sqrt{d_K}$$

$$= Q \times K^T / 2$$

(v) english

$\langle n_1, n_2, n_3 \rangle$

hindi

$\langle y_1, y_2 \rangle$

$\langle y_1, y_2, 0 \rangle$

\hookrightarrow zero padding

Masked multi-head self attention

① zip embedding and position embedding

\hookrightarrow zero padding \rightarrow to make sequence length equal

Masked
multi head
attention

zip

$[4, 5, 6, 7]$

oip

$[1 \ 2 \ 3]$

\downarrow

$[1 \ 2 \ 3 \ 0]$

i) input embedding and position encoding.

output embedding:

step 1

$[(0.1, 0.2, 0.3, 0.4),$

$(0.5, 0.6, 0.7, 0.8)], \text{ TPF} \rightarrow 0$

$(0.9, 1.0, 1.1, 1.2),$

$(0.0, 0.0, 0.0, 0.0)]$

Same
matrix

Steps involved in masked multhead

attention

Calculus

- ① Linear projection $\rightarrow Q, k, v$
- ② scaled Dot product attention.
- ③ Mask application
 - \rightarrow look ahead mask.
 - \rightarrow padding mask.

Step 2:- Linear project for Q, k and v .

Create Q, k, v vector.

$$Q = \text{OIP embedding} * W_Q = \text{OIP embedding}$$

$$K = \text{OIP embedding} * W_K = \text{OIP embedding}$$

$$V = \text{OIP embedding} * W_V = \text{OIP embedding}$$

$$Q = k = v = \left[\begin{bmatrix} [0.1, 0.2, 0.3, 0.4], \\ [0.5, 0.6, 0.7, 0.8], \\ [0.9, 1.0, 1.1, 1.2], \\ [0.0, 0.0, 0.0, 0.0] \end{bmatrix} \right]$$

③ Scaled Dot product attention Calculation

$$\text{Scores} = Q \times K^T / \sqrt{dk}$$

$$= Q \times K^T / 2$$

$$\underline{\text{Scores}} =$$

~~Q Environ~~

$$\begin{bmatrix} [0.1, 0.2, 0.3, 0.4], \\ [0.5, 0.6, 0.7, 0.8], \\ [0.9, 1.0, 1.1, 1.2], \\ [0.0, 0.0, 0.0, 0.0] \end{bmatrix} \times \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.6 \\ 0.7 \\ 0.8 \end{bmatrix} \begin{bmatrix} 0.9 \\ 1.0 \\ 1.1 \\ 1.2 \end{bmatrix}$$

$$\text{Score} = \begin{bmatrix} [0.3, 0.7, 1.1, 0.0] \\ [0.7, 1.9, 3.1, 0.0] \\ [1.1, 3.1, 5.1, 0.0] \\ [0.0, 0.0, 0.0, 0.0] \end{bmatrix}$$

$$\begin{bmatrix} 0.1 \times 0.1 + 0.2 \times 0.2 + 0.3 \times 0.3 + 0.4 \times 0.4, \\ 0.1 \times 0.5 + 0.2 \times 0.6 + 0.3 \times 0.7 + 0.4 \times 0.8, \\ 0.1 \times 0.9 + 0.2 \times 1.0 + 0.3 \times 1.1 + 0.4 \times 1.2, \\ 0.1 \times 0 + 0.2 \times 0 + 0.3 \times 0 + 0.4 \times 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.7 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

skip³
Masked approach

It helps managing the structure of the sentences being processed and ensures the models behave correctly during long and ~~softmax~~ inferencing.

Reasons:

- ① handling variable length sequences
with padding mask

Purpose:

- ① to handle sequences of diff length in batch
- ② To ensure that padding tokens, which are added to make sequences of uniform length, do not affect the model prediction.

e.g: Sequence 1 [1, 2, 3]

Sequence 2 [4, 5, 0]

Suppose

input

y

100

01P

10.00.90

↑ 0 is the
padding token

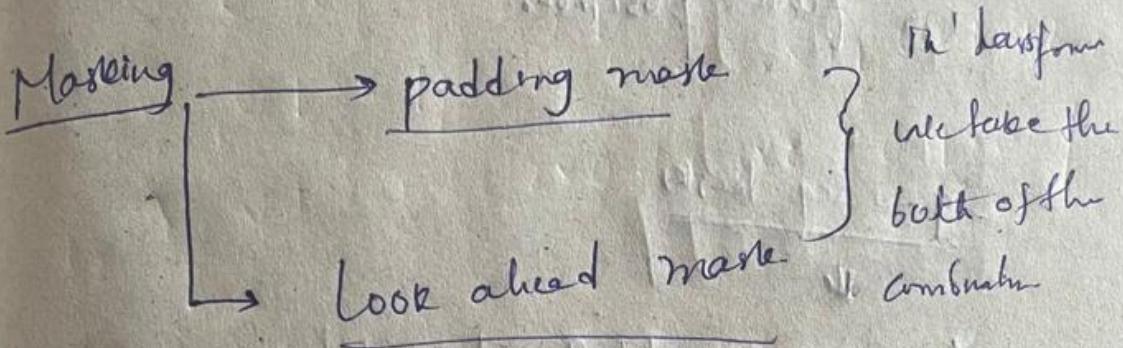
influence other

tokens

lead to incorrect (0)
prediction.

a padding mask

ii)
The tokens are ignored.



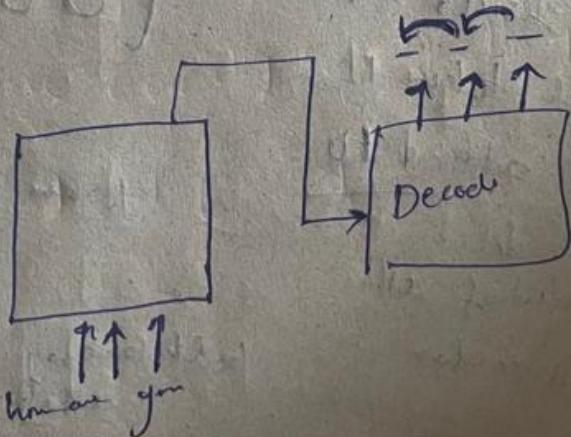
① padding mask :

$$[1, 1, 1, 1] \leftarrow [1, 2, 3]$$

$$[1, 1, 0] \leftarrow [4, 5, 0]$$

② Look ahead mask:

maintain auto regen property.



Look ahead mask \rightarrow Decoder output

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Combine padding and look ahead mask

Element wise multiply of 2 mask.

$$\text{combine mask} = \begin{bmatrix} [1, 0, 0] \\ [1, 1, 0] \\ [0, 0, 0] \end{bmatrix}$$

look ahead mask

(S)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Scw: $\begin{bmatrix} [0.3, 0.7, 1.1, 0.0], \\ [0.7, 1.9, 3.1, 0.0] \end{bmatrix}$

$\begin{bmatrix} 1.1, 3.1, 5.1, 0.0 \end{bmatrix}$

$\begin{bmatrix} 0.0, 0.0, 0.0, 0.0 \end{bmatrix}$

15

14

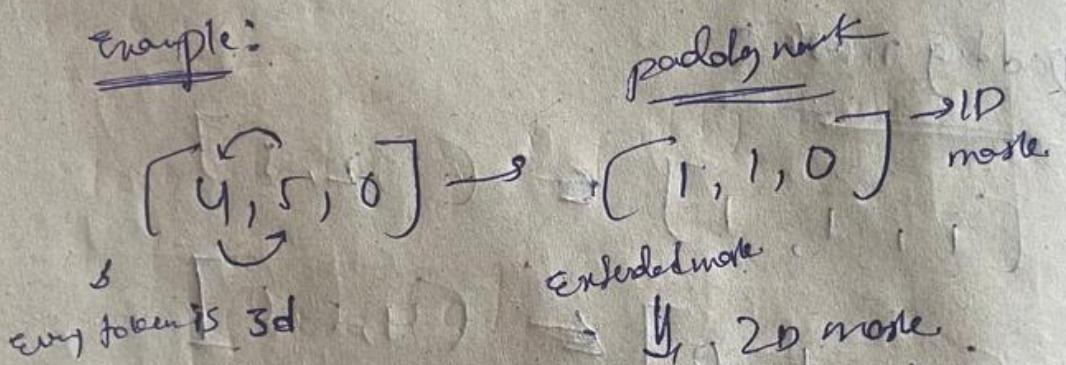
13

① To ensure that each position in the decoder output sequence can only attend to the previous position, but no future position.

Seq → Seq tasks

language modeling, Translation.

example:



for each token in
the sequence the
mask should indicate
which token it
can attend to

&
in the context of other
numbers

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \leftarrow \text{1st token (1)} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \leftarrow \text{2nd token (2)}$$

token 1 attend to token

1, 2 4, 5

2, 1 5, 4

both attending
other w.r.t. each other

paddy mask [enriched so 2D. first]

$$[(1,1,1,0)]$$

cross

$$[(1,1,1,0)]$$

$$[(1,1,1,0)]$$

$$[(0,0,0,0)]$$

combine mask

look ahead mask * paddy mask

$$[(1,0,0,0)]$$

$$[(1,1,0,0)]$$

Counting to
 ∞

$$[(1,1,1,0)]$$

$$[(1,1,1,0)]$$

marked as

$$[(1, -\infty, -\infty, -\infty)]$$

zero out the
the influence

$$[(1, -\infty, -\infty, -\infty)]$$

when the
softmax is
applied

$$[(1,1,1,-\infty)]$$

$$[(1,1,1,-\infty)]$$

Softmax

Score = softmax (masked score)

$$= \left[\begin{bmatrix} 1.0, 0.0, 0.0, 0.0 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} 0.3, 0.7, 0.0, 0.0 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} 0.1, 0.3, 0.6, 0.0 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} 0.0, 0.0, 0.6, 0.0 \end{bmatrix} \right]$$

④ weighted sum of value:

where $\text{Op} = \text{softmax} * \text{Cem} * V$

Major reason for masking

- ① handling variable-length sequences with padding mask
- ② preserving autoregressive property with look-ahead mask

Encoder - Decoder multihop attention

- ① encoder OIP \rightarrow set of attributes

Veeh's Law

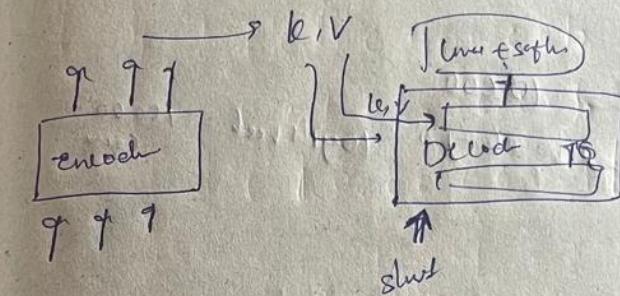
- ② Planted mulberry after test &
(Guaynoly)

These are to be used by each decor
in its "encore decor" after layer

keeps the decoder to focus on appropriate place on the top sequence

Jalam Mai - 918 Hub. 10

Glog.



Pinal Detachment and Southern Ledge

log-poles -

The diagram illustrates a neural network layer. At the bottom, a box labeled "Linear" contains three input units. An arrow points from this box to a horizontal row of seven boxes above it, labeled "logits". The first five boxes in this row are labeled with numbers: 0, 1, 2, 3, 4, 5. An arrow points from the number 5 to a box labeled "softmax" at the top. From the softmax box, an arrow points down to the "logits" row. To the right of the "logits" row is a box containing a single tickmark.

Convol layer is a simple fully connected
model that projects the vector produced
by the stack of Deconv.

model \Rightarrow 10⁴ 000
 \Rightarrow vocabulary \Rightarrow logits vocab = 100 000
cens

logits

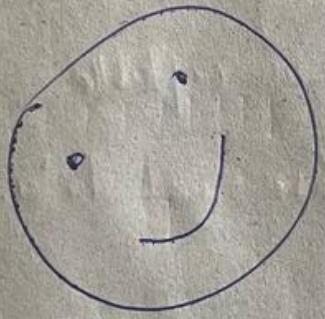
softmax

sent of unique word.

softmax

softmax:

Completed
Data Science ML, DL
NLP BootCamp



~~~kish Naiz~~

~~Done !!~~