→ If FP is more important FN

$\beta = 0.5$ then $F_{0.5}$ score $= (1+0.25)\frac{P \times R}{P+R}$

→ If FN >> FP

$\beta = 2$ then F_2 score $= 1+4\frac{P \times R}{P+R}$

Majorly we calculate F1 Beta score

28th Dec

Decision Tree

DTC — ID, CART] Both

(Splits binary basis)

Binary Classification

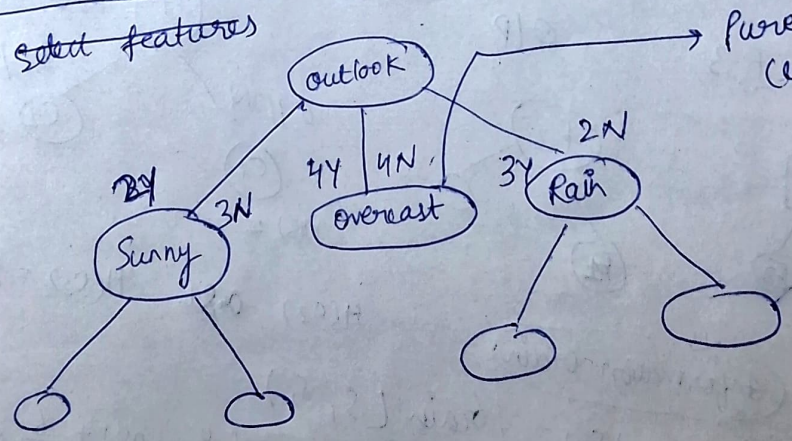
↓

- we have to train our model until ~~the~~ we get a single leaf node.
- where is my leaf node we'll check it through Entropy, Gini Impurity

Entropy & Gini split feature to select

Steps to design a decision tree

~~Select features~~

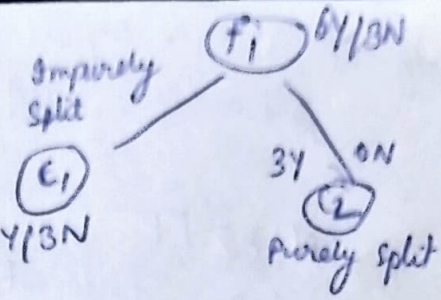


→ Pure split (leaf node)

Purity : Pure or Impure

- Entropy
- Gini Impurity

Step 2 What feature you need select for splitting (Information gain)

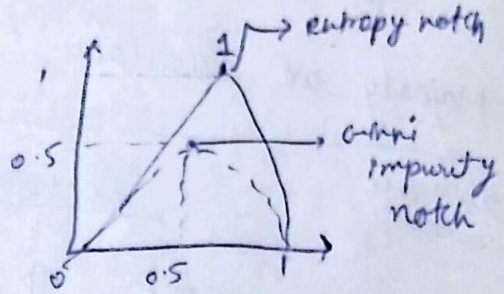


- Entropy (0, 1) Best Suitable

$H(S) = -p * \log_2 p + -(p * \log_2 p)$

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

$= 0$



- Gini Impurity

Pure $1 - \sum_{i=1}^{n} (P)^2 \rightarrow 1 - \left(\frac{3}{3}\right)^2 = 0$

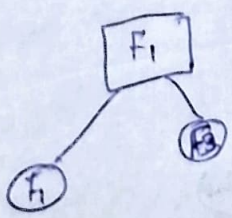
Impure $1 - ((p+)^2 + (p-)^2) \rightarrow 1 - ((1/2)^2 + (1/2)^2) = \underline{\underline{0.5}}$

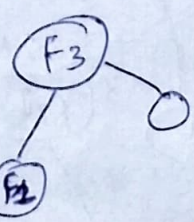
Decision Tree

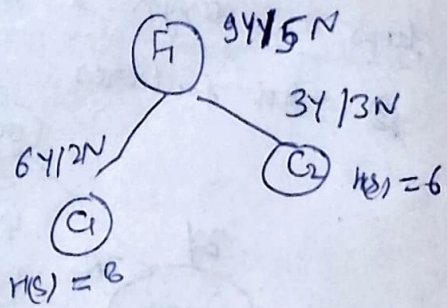
Multiple feature & O/P → Supervised learning

F_1 F_2 F3 O/P

DT







$H(C_2) = 0.81$ $H(C_2) = 1$

Information Gain

c → belongs to

$Gain(S, F_2) = H(S) - \frac{\sum |S_v|}{|S|}$ Variance

$H(S) = -p\log p_+ - P_-\log p_-$

$Gain(S, F_1)$

$= 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$

$= 0.049$

$Gain(S, F_2) = 0.051$

choose this one

$\boxed{Gain(S, F_2) > Gain(S, F_1)}$

Small dataset

Fixed classification

1, 2, 3, 4

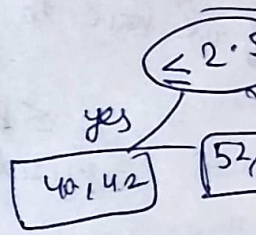
Entropy

Big Dataset

Gini Impurity

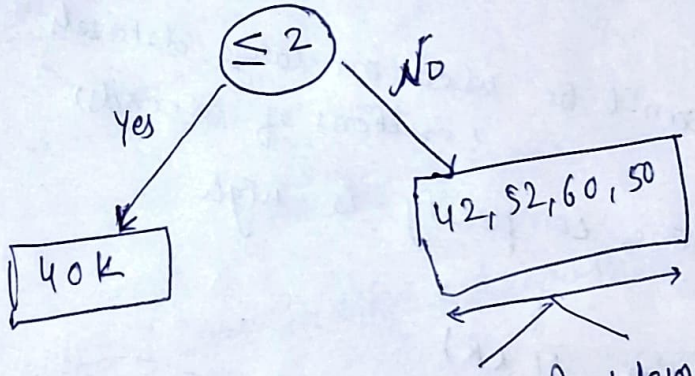
DTR (Decision Tree Regression)

Exp	Dataset Career Grap	Salary
-2	Yes	40 K
2.5	Yes	42 K
3	No	52 K
4	No	60 K
4.5	Yes	56 K



C1 = 100

C2 = 42



Variance Reduction {Regression Problem}

$$\text{variance} = \frac{1}{n} \sum_{i=1}^{n} \underbrace{(y-\hat{y})^2}_{\text{Average}} \Rightarrow \text{MSE}$$

50 → Predi

$$\text{Var (root)} = \frac{1}{5} \rightarrow 60.8$$

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

DT Parameter

$Var(CI) = \frac{1}{2}((40-50)^2 + (42-50)) = \frac{1}{2}(100+64) = 82$

$Var(CI) = \frac{1}{3}[4+100+36] = 46.66$ Child I (C1)

Variance Reduction

$60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.66\right] = 0.04$

$60.8 - \left[\frac{1}{5} \times 100 + \frac{4}{5} \times 42\right]$

$60.8 - [20 + \quad] = 0$

If we have to make multiple choices through regression

0.04 → variance high

0 ≤ 2 leftside

≤ 2.5 Right side

Disadvantages → can't be used on large datasets (millions of records)

→ time complexity is high

Ensemble Modelling (L&R)

Types → 1) Bagging → Random forest → multiple decision tree

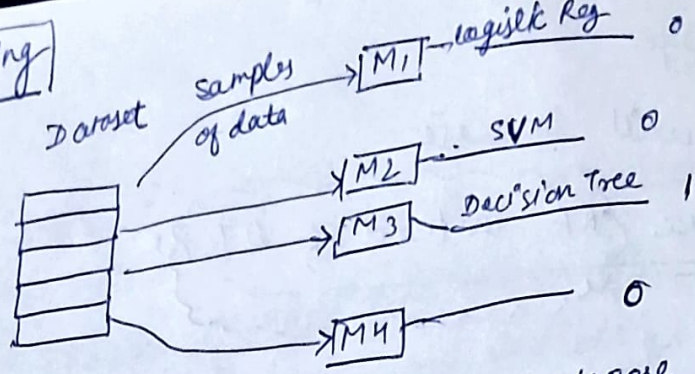
2) Boosting → Ada Boost

CAT Boost

Gradient Boost

XG Boost

Bagging



$d' < D$
sample data — original Data

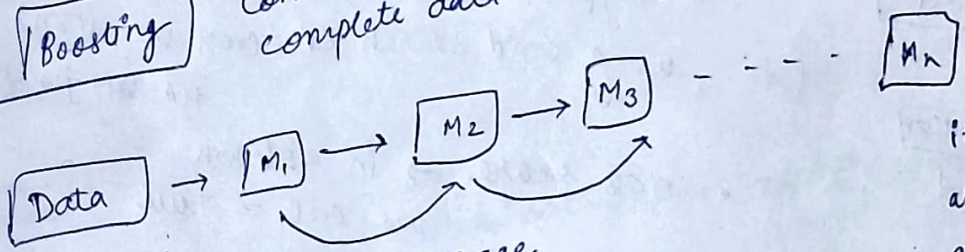
Classification →
Majority of voting → zero & choose your model

Regression → Average of all outputs (models combined)

Models are connected parallely in bagging

Boosting

Combinations of all weak learners
complete data is being used

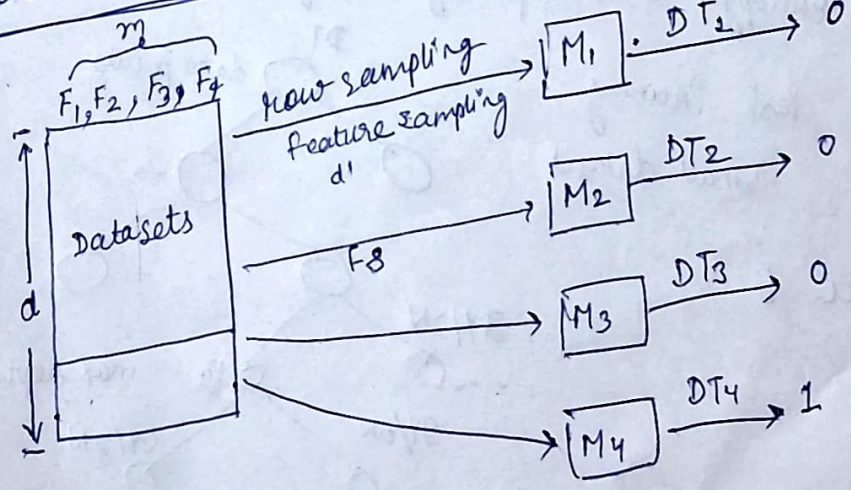


series

at the end it will club all the weak le as well.

weak learners → Average
↓
strong learners

Random Forest (Classification & Regression)



$d' < D$ m'

Classification → majority voting classifier

Regression → Average

Time complexity will increase

Q. Why should we use RF instead of DT ?

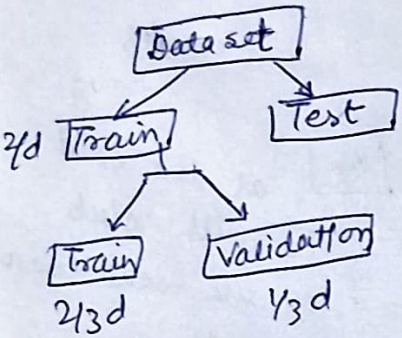
DT → overfitting

- Train Accuracy ↑ low Bias $\xrightarrow{RF}$ low bias
- Test Accuracy ↓ high variance $\xrightarrow{RF}$ low variance

---

Out of Bag Evaluation (OOB)

sklearn



100 → data points were not used

- hyper parameters (tuning) using Grid Search CV (cross validation) 40 data points
- OOB score → in sklearn use OOB = True

Percentage %

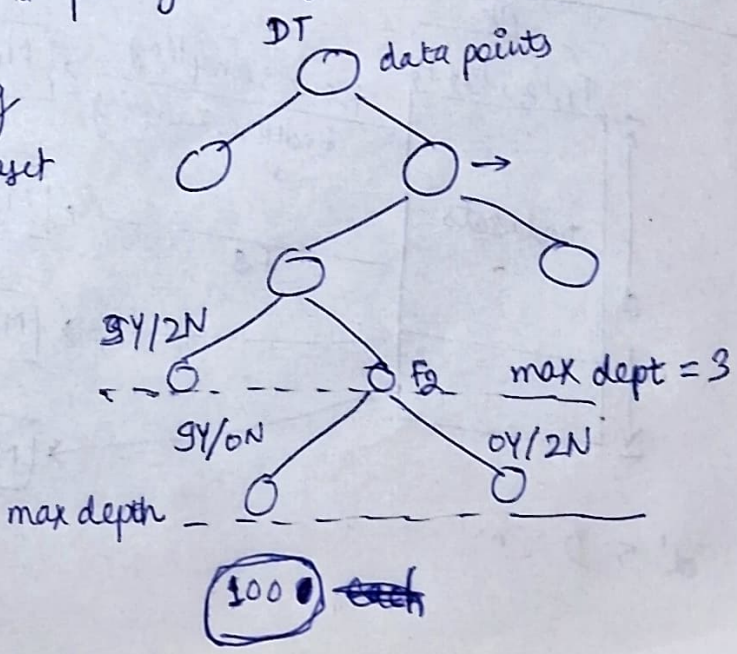
Random forest (Classification & Regression)

Decision Tree → Entropy, Gini Impurity & Information

Pre pruning & Post Pruning

large datasets small dataset

overfitting reduce ↓



Grid Search CV

- It is a technique in machine learning used to find the best combination of hyper parameters for a model.
- It systematically ~~tag~~ tries every possible combination of hyper parameter values you specify and evaluate the model performance for each combination using cross validation.

Ex: Hyperparameter C's values & kernels for SVM (code)

Cross Validation → Split the data into multiple folds to ensure the model perform well on unseen data.

Grid: → A list of all combination of hyper parameters to try.

CV → Evaluate each combinations on multiple data split.

Deadlog (decision tree) ?? Dedd logram.

Grid Search CV

- It is a technique in machine learning used to find the best combination of hyperparameters for a model.
- It systematically ~~tag~~ tries every possible combination of hyper parameter values you specify and evaluate the model performance for each combination using cross validation.

Ex: Hyperparameter C's values & kernels for SVM (code)

Cross Validation → Split the data into multiple folds to ensure the model perform well on unseen data.

Grid: → A list of all combination of hyper parameters to try.

CV → Evaluate each combinations on multiple data split.

Dendrog (decision tree) ?? Dendrogram.

Classification → majority voting classifier

Regression → Average

Time complexity will increase

Q why should we use RF instead of DT ?

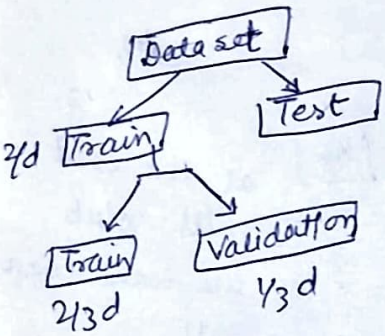
DT → overfitting

- Train Accuracy ↑ low Bias $\xrightarrow{RF}$ low bias
- Test Accuracy ↓ high variance $\xrightarrow{RF}$ low variance

---

Out of Bag Evaluation (OOB)

sklearn



100 → data points were not used

- Hyper parameters (tuning) using Grid Search CV (Cross Validation) 40 data points
- OOB score → in sklearn use OOB = True

Percentage %

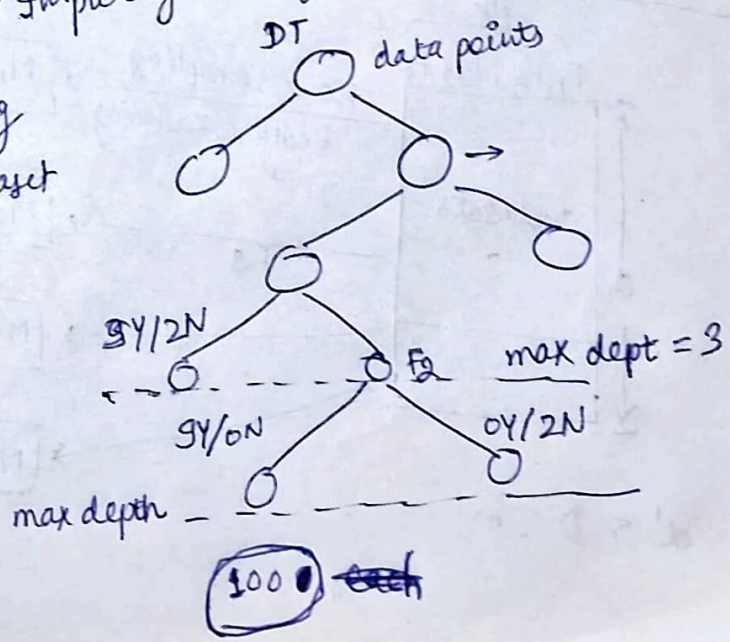
Random forest (Classification & Regression)

Decision Tree → Entropy, Gini Impurity & Information

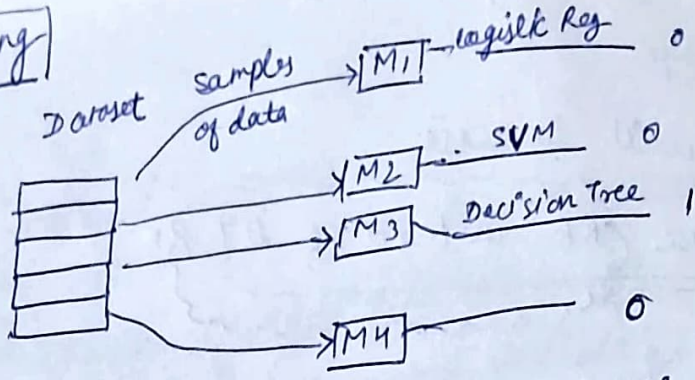
Pre pruning & Post Pruning

large datasets small dataset

overfitting reduce ↓



Bagging



$d' < D$
sample data — original Data

Classification →
Majority of voting → zero & choose your model

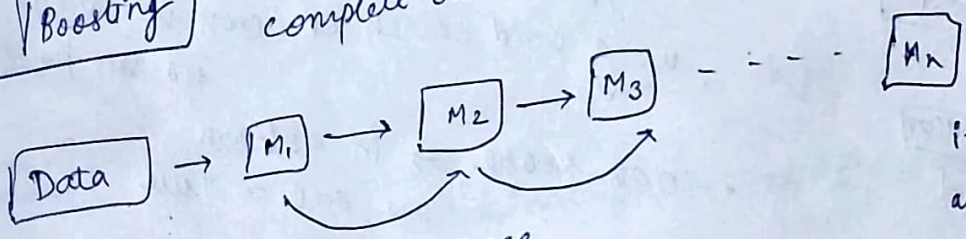
Regression → Average of all outputs (models combined)

Models are connected parallely in bagging

Boosting

Combinations of all weak learners
complete data is being used

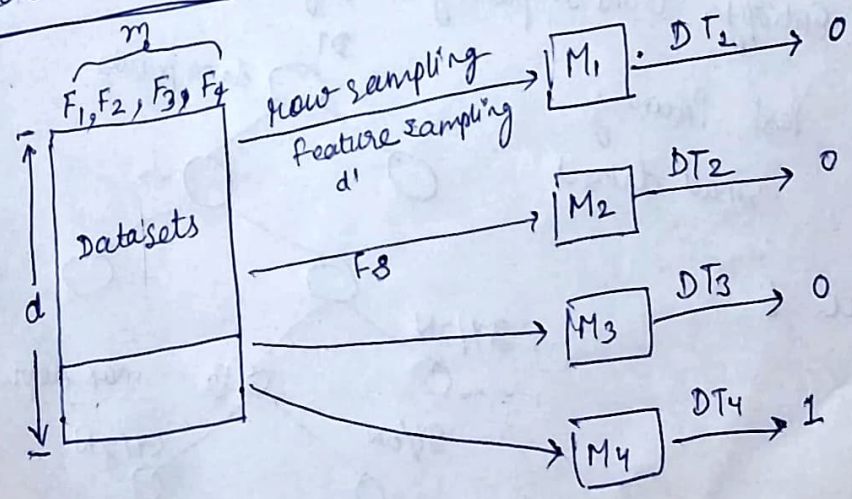
Series



at the end it will club all the weak le as well.

weak learners → Average
↓
strong learners

Random Forest (Classification & Regression)



$d' < D$ m'

DT Parameter

$Var(CI) = \frac{1}{2}((40-50)^2 + (42-50)) = \frac{1}{2}(100+64) = 82$

$Var(CI) = \frac{1}{3}[4+100+36] = 46.66$ Child I (C1)

Variance Reduction

$60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.66\right] = 0.04$

$60.8 - \left[\frac{1}{5} \times 100 + \frac{4}{5} \times 42\right]$

$60.8 - [20 + \quad] = 0$

If we have to make multiple choices through regression

0.04 → variance high

0

≤ 2 leftside

≤ 2.5 Right side

Disadvantages → can't be used on large datasets (millions of records)

→ time complexity is high

Ensemble Modelling (L&R)

Types → 1) Bagging → Random forest → multiple decision tree

2) Boosting → Ada Boost

CAT Boost

Gradient Boost

XG Boost

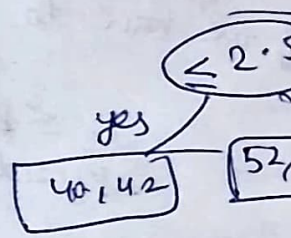
Small dataset
Fixed classification
1, 2, 3, 4
Entropy

Big dataset
Gini Impurity

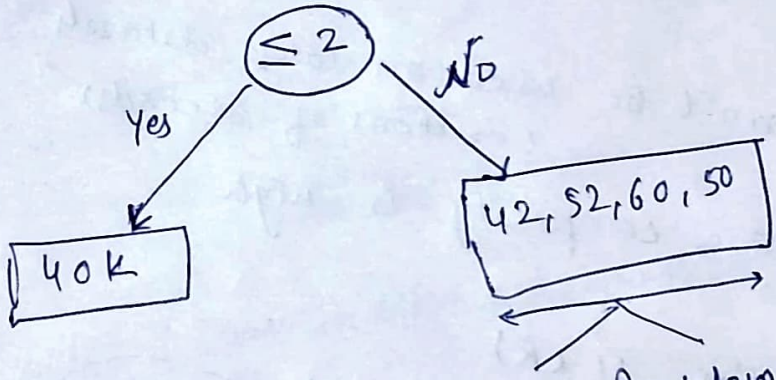
DTR (Decision Tree Regression)

Dataset

Exp	Career Grop	Salary
-2	Yes	40 K
2.5	Yes	42 K
3	No	52 K
4	No	60 K
4.5	Yes	56 K



C1 = 100
C2 = 42



Variance Reduction {Regression Problem}

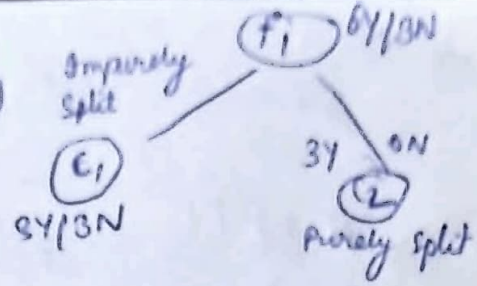
$$\text{variance} = \frac{1}{n} \sum_{i=1}^{n} \underbrace{(y-\hat{y})^2}_{\text{Average}} \Rightarrow \text{MSE}$$

50 → Predi

$$\text{Var (root)} = \frac{1}{5} \rightarrow 60.8$$

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

Step 2 What feature you need select for splitting (Information gain)

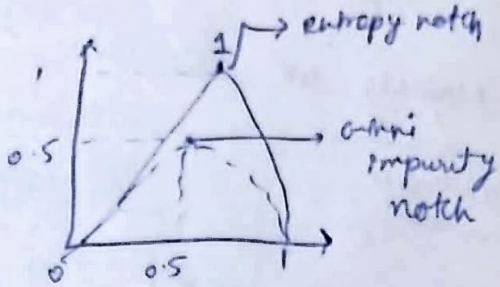


- Entropy (0, 1) Best Seatable

$H(S) = -p * \log_2 p + -(p * \log_2 p)$

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

$= 0$



- Gini Impurity

Pure $1 - \sum_{i=1}^{n} (P)^2 \rightarrow 1 - \left(\frac{3}{3}\right)^2 = 0$

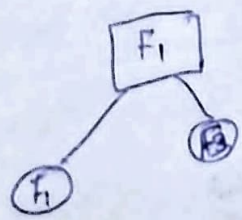
Impure $1 - ((p_+)^2 + (p_-)^2) \rightarrow 1 - ((1/2)^2 + (1/2)^2) = \underline{\underline{0.5}}$

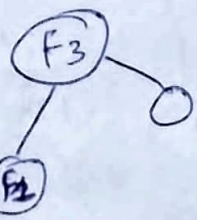
Decision Tree

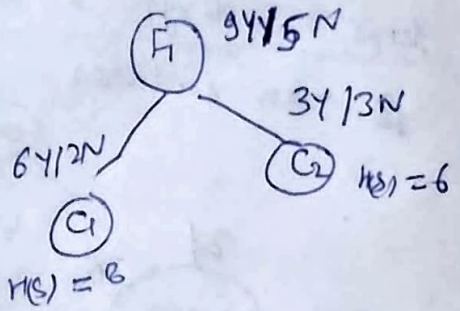
Multiple feature & O/P → Supervised learning

F1 F2 F3 O/P

DT







$H(C_2) = 0.81$ $H(C_2) = 1$

Information Gain

C → belongs to

$Gain(S, F_2) = H(S) - \sum_{v \in values} \frac{|S_v|}{|S|}$

$H(S) = -p\log p_+ - P_-\log p_-$

$Gain(S, F_1)$

$= 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$

$= 0.049$

$Gain(S, F_2) = 0.051$

choose this one

$\boxed{Gain(S, F_2) > Gain(S, F_1)}$

→ If FP is more important FN
$\beta = 0.5$ then $F_{0.5}$ score $= (1+0.25)\frac{P \times R}{P+R}$

→ If FN >> FP
$\beta = 2$ then F_2 score $= 1+4\frac{P \times R}{P+R}$

Majorly we calculate F1 Beta score

28th Dec

Decision Tree

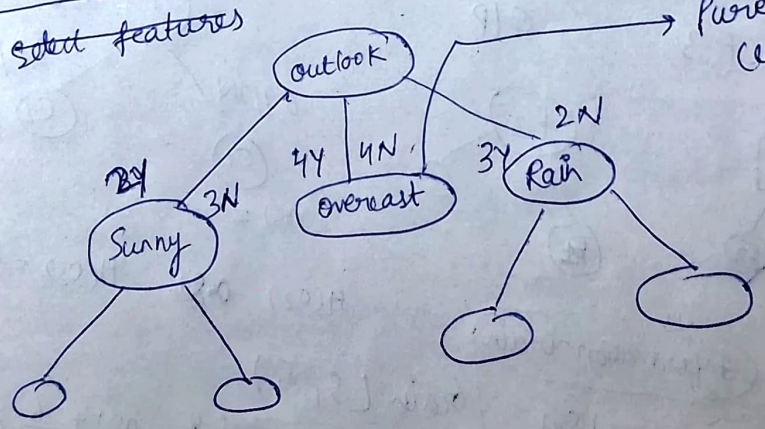
Binary classification
↓
- we have to train our model until ~~the~~ we get a single leaf node.
- Where is my leaf node we'll check it through Entropy / Gini Impurity

DTC — ID, CART } Both
(Splits binary basis)

Entropy & Gini split feature to select

Steps to design a decision tree

~~Select features~~



- Purity : Pure or Impure
  - Entropy
  - Gini Impurity

Ques Overcast & Mild (play or not ?)

P(yes) + P(overcast/yes) + P(overcast/No)

$\frac{9}{14}$

P(No) . P(Mild/yes) . P(Mild . No)

$\frac{5}{14}$

100% will play

{ Data Preprocessing }

In this step the data analyses pipeline

- It involves transforming raw data into a clean & usable format.
- Primary step in this process include -
  - → Data cleaning (Handling missing values, removing duplicates and correcting inconsistencies)
  - → Data Integration (combining data from different sources)
  - → Data Transformation (Normalizing, scaling and transforming data for better analysis)
  - → Data Reduction (Reducing the number of attributes or dimensions)
  - → Data Rescaling (It involves adjusting the range of data featured to a standard range)

<u>Min-Max Scaling</u> : Rescale the data to fixed range 0 to 1.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

→ Data standardization

Standardizing transforms the data to have a mean of zero and standard deviation of one.

$$K_{stand} = \frac{x - \mu}{\sigma}$$

$\mu \rightarrow$ mean

$\sigma \rightarrow$ standard deviation

→ Normalizing Data : It is scaling the data such that it has a unit known (norm), typically used in algorithms that involves distance like KNN. The most common form is the N2 normalization.

$$K_{Norm} = \frac{K}{\|X\|}$$ → Euclidean Norms.

Euclidean Distance → It is measured by the straight line distance between two points in Euclidean space for two points $A(X_1, Y_1)$ and $B(X_2, Y_2)$

$$ED = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

→ Manhattan distance : It is also known as L1 distance. It is the sum of the absolute difference of the coordinates for two points $A(X_1, Y_1)$ & $B(X_2$

$$d(A, B) = |X_2 - X_1| + |Y_2 - Y_1|$$

→ Handling categorical Data : It needs to transform ~~m~~ a numerical format for most machine learning algori

One Hot Encoder : It converts categorical ~~values~~ vari into binary matrix. for example

"Red", "Green", "Blue"

would be transformed into three columns.

	Red	Green	Blue
1	1	0	0
2	0	1	0
3	0	0	1

Label Encoding → It converts into integer (Binary Values)

Red: 0 Green: 1 Blue: 2

takes a lot of time

~~Typically used~~

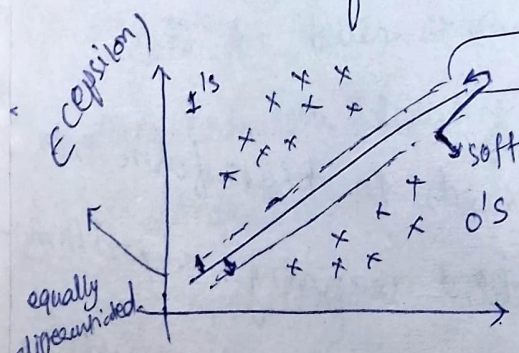
F1 score is also called F-1 beta score?

→ if given option we classify (cat's, species)

→ if to tell a numeric value b/w a range its regression

Support Vector Machine

Tasks → classification & Regression



marginal plane

Hard marginal plane

For better accuracy of your model make sure to keep high distance between marginal plane & soft marginal

We don't work on Hard marginal plane. (not exists in real life)

Support vector classification

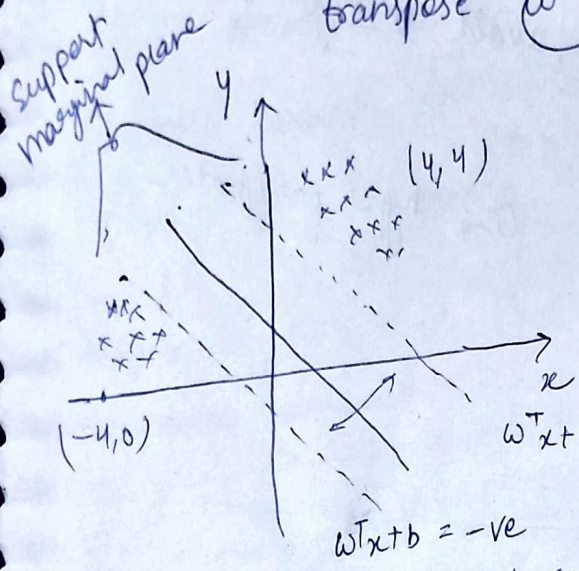
$x \rightarrow$ magnitude or vector (direction)

$y = mx + c$ $\quad y = \theta_0 + \theta_1 x$ $\quad y = \beta_0 + \beta_1 x$ ↑ straight line

$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$ $\quad ax + by + c = 0$

transpose $(w^T x)$ slope $\quad y = -\frac{a}{b}x + \frac{c}{b}$

$mx + c$



for ex → $3x + 2y + 4 = 0$

$-\frac{a}{b}x = -8 \quad \frac{c}{b} = 24$

$(w) \rightarrow$ magnitude of vector

data points will always lie b/w zeroes & ones

$w^T x_1 + b = +1 \qquad w^T x_2 + b = -1$

$\frac{w^T x_1 + b}{\vec{w}} \rightarrow \frac{w^T(x_1 - x_2)}{\|w\|} = \frac{2}{\|w\|}$ → diff b/w both planes, maximise

$\text{Maximum}(w, b) = \frac{2}{\|w\|}$

marginal plane distance

Constraints such that

$y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$

for all alternate data points

$y * (w^T x_i + b) \geq 1$

constraints

Miss classification (when the maximize value or marginal plane distance is high)

$\frac{2}{\|w\|}$ maximise $\qquad \frac{\|w\|}{2}$ minimise

Cost function (SVM)

$$\text{minimize}(w,b) = \frac{\|w\|}{2} + \underbrace{C_i \sum_{i=1}^{w} \xi_i}_{\text{summation of the distance}}$$

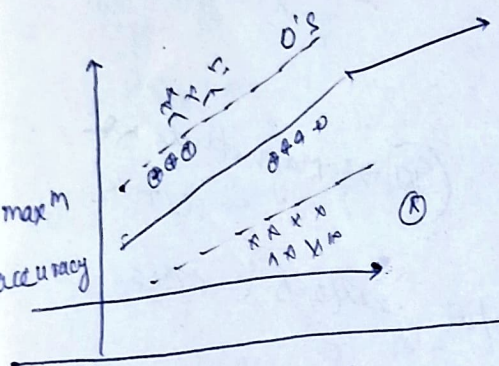
$\xi_i \rightarrow$ eta

of miss classification points from marginal plane.

$C_i \rightarrow$ How many point we can avoid ~~through~~ miss classification

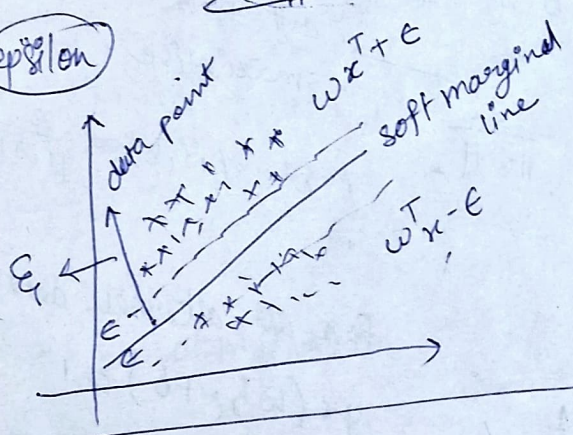
$w \rightarrow$ width of marginal plane

$C_i \rightarrow$ Hyper parameter



Support Vector Regression

ϵ epsilon



SVR

Cost f/n : Minimize $\frac{\|w\|}{2}$

constraint {MSE}

$|y - w^T x_i| \leq \epsilon$ It is good

Cost function

$$\frac{\|w\|}{2} + C \sum_{i=1}^{w} |\xi| \Big\} \text{ ringe loss ??}$$

Penalty Parameter

$\downarrow$

C in code

Hyper parameter

$\xi \rightarrow$ eta

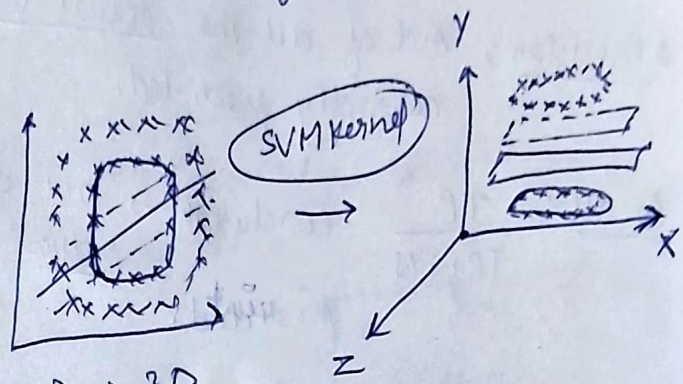
Constraint

$$[|y_i - w_i x_i| \leq \epsilon + |\xi_i|]$$

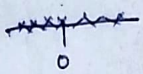
Transformation through SVM Kernel ~~is done~~ performed to convert the model into 3D. (for left out data points)

Types of SVM kernels

3) Polynomial kernel
RBF Kernel
Sigmoid Kernel



1D → 2D → 3D



- Polynomial Kernel → convert data into whatever dimension you need. 1D → 2D

Measure Parameters

Confusion Matrix

predicted \ Actual	1	0
1	TP	FP
0	FN	TN

- Accuracy
- Precision
- Recall
- F1 Beta Score

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

Predicts for any new data point. → How much is TP & FP → based on this How much the model has predicted.

Precision

Datasets : Binary Classification

1000 data points: 900 1's, 100 0's } Imbalance datasets

After training the model How good can it predict more → Precision

$$\text{Precision} = \frac{TP}{TP + FP \downarrow\downarrow}$$

decreasing FP will give you better precision.

Recall → ~~and~~ you have to decrease the FN values.

• **Precision**: out of all the actual value how many are correctly predicted.

Recall: $\frac{TP}{TP+FN}$ conclusion → out of all the predicted value how many are correctly predicted.

Example or Use Case 1

Spam classification

Email → Spam
Model → Spam

Good classification

Mail → Not Spam
Model → Spam

Blunder

	1	0
1	TP	FP
0	TN	FN

FP → reduce it (what if any imp email is recieved & we are unaware of that)

Use Case 2:- To predict whether person has cancer or not

Truth → Yes (Concerious)
Model → No

Truth → Cancer
Model → Cancer

$\boxed{FN}$ (Blunder is cuz of model machine)

Domain expert for Medical field

Stock Market Analysis → Reduce both ↓↓

F1 Beta Score: combines both precision & recall if FP & FN are both imp.

$1 + \beta^2 \left(\frac{P \times R}{P+R}\right)$

↳ Harmonic Mean

$\beta = 1$

F1 score ⇒ $2 \times \frac{P \times R}{P+R}$

→ If FP is more important FN

$\beta = 0.5$ then $F_{0.5}$ score $= (1+0.25)\frac{P \times R}{P+R}$

→ If FN >> FP

$\beta = 2$ then F_2 score $= 1+4\frac{P \times R}{P+R}$

Majorly we calculate F1 Beta score

28th Dec

Decision Tree

DTC — ID, CART] Both
(Splits binary basis)

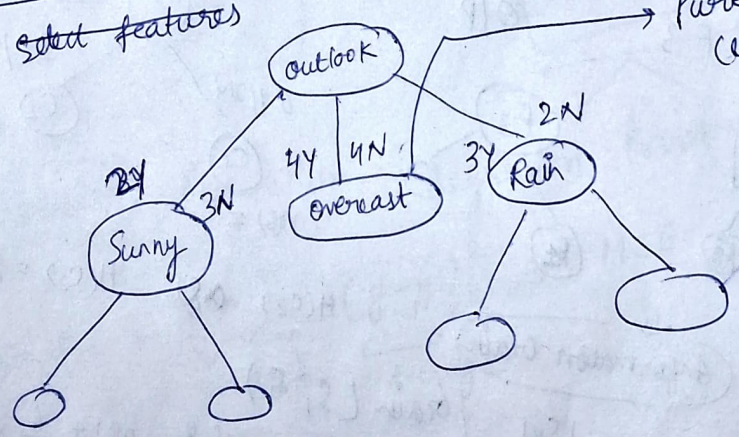
Binary classification

- we have to train our model until ~~the~~ we get a single leaf node.
- Where is my leaf node we'll check it through Entropy, Gini Impurity

Entropy & Gini split feature to select

Steps to design a decision tree

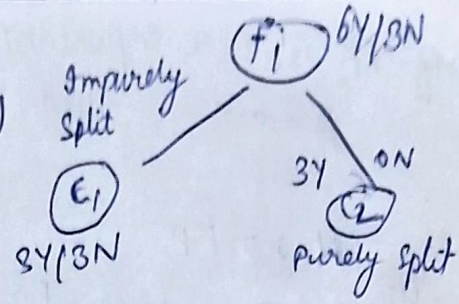
~~Select features~~



Purity : Pure or Impure

- Entropy
- Gini Impurity

Step 2 What feature you need select for splitting (Information gain)

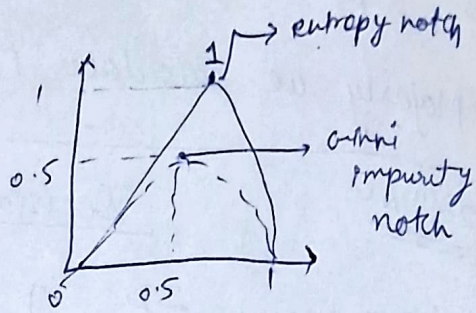


- Entropy (0, 1) Best Suitable

$H(S) = -p * \log_2 p + -(p * \log_2 p)$

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

$= 0$



- Gini Impurity

Pure $1 - \sum_{i=1}^{n} (P)^2 \rightarrow 1 - \left(\frac{3}{3}\right)^2 = 0$

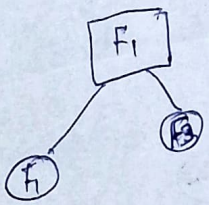
Impure $1 - ((p+)^2 + (p-)^2) \rightarrow 1 - ((1/2)^2 + (1/2)^2) = 0.5$

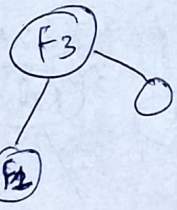
Decision Tree

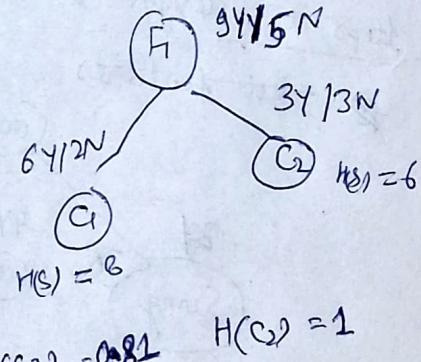
Multiple feature & O/P → Supervised learning

F1 F2 F3 O/P

DT







$H(S) = 6$

$H(C_2) = 0.81$ $H(C_2) = 1$

Information Gain

→ belongs to

$Gain(S, F_2) = H(S) - \frac{\sum |S_v|}{|S|}$ Variance

$H(S) = -p\log p_+ - p_-\log p_-$

$Gain(S, F_1)$

$= 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$

$= 0.049$

$Gain(S, F_2) = 0.051$

choose this one

$\boxed{Gain(S, F_2) > Gain(S, F_1)}$

Small dataset

Fixed classification

1, 2, 3, 4

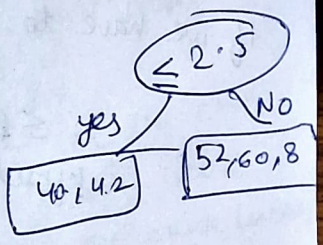
Entropy

Big Dataset

Gini Impurity

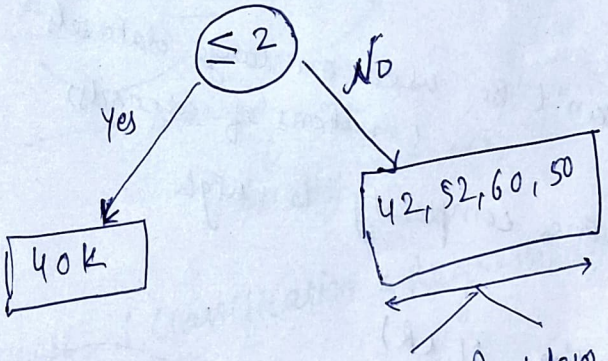
DTR (Decision Tree Regression)

Exp	Dataset carrer Grap	Salary
-2	Yes	40 K
2.5	Yes	42 K
.3	NO	52 K
4	NO	60 K
4.5	Yes	56 K



C1 = 100

C2 = 42



Variance Reduction {Regression Problem}

$$\text{variance} = \frac{1}{n} \sum_{i=1}^{n} \underbrace{(y-\hat{y})^2}_{\text{Average}} \Rightarrow \text{MSE}$$

50 → Predicted value

$$\text{Var (root)} = \frac{1}{5} \rightarrow 60.8$$

$$\text{Var(root)} = \frac{1}{5}\{(40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2\}$$

DT Parameter

$Var(CI) = \frac{1}{2}\left((40-50)^2 + (42-50)\right) = \frac{1}{2}(100+64) = 82$

$Var(CI) = \frac{1}{3}\left[4+100+36\right] = 46.66$

Child I (C1)

Classification

Variance Reduction

$60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.66\right] = 0.04$

$60.8 - \left[\frac{1}{5} \times 100 + \frac{4}{5} \times 42\right]$

$60.8 - [20 + \quad] = 0$

If we have to make multiple choices through regression

0 $\quad$ 0.04 → variance high

≤ 2 leftside $\quad$ ≤ 2.5 Right side

Disadvantages → can't be used on large datasets (millions of records)

→ time complexity is high

Ensemble Modelling (L & R)

Types → 1) Bagging → Random forest → multiple decision tree

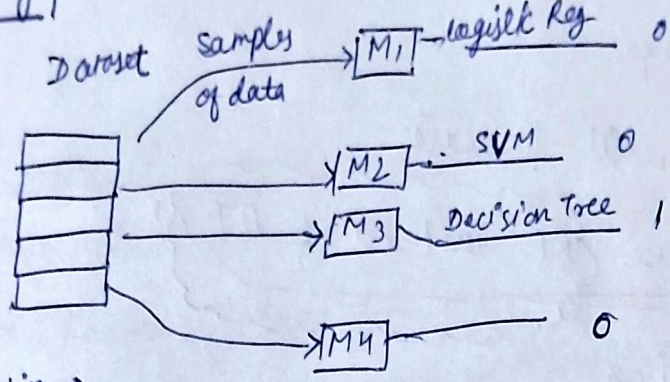
2) Boosting → Ada Boost

CAT Boost

Gradient Boost

XG Boost

Bagging



$d' < D$ original data

sample data

Classification →

Majority of voting → zero & choose your model

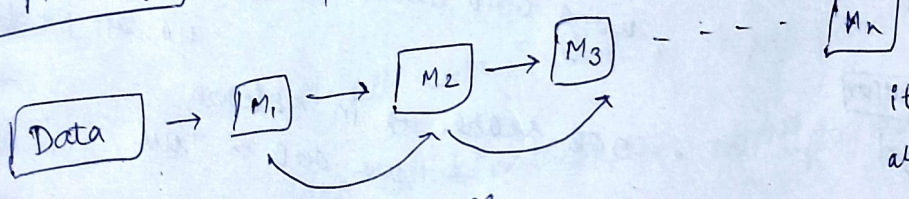
Regression → Average of all outputs (models combined)

Models are connected parallelly in bagging

Boosting

Combinations of all weak learners
complete data is being used

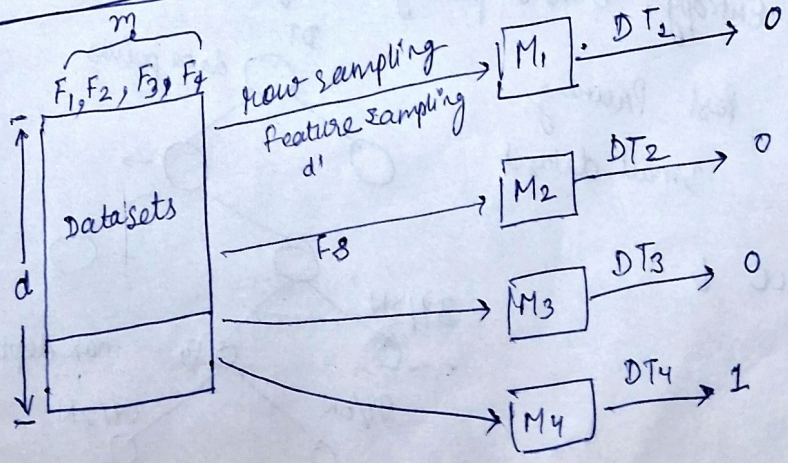
series



at the end it will club all the weak learners as well.

weak learners → Average → strong learners

Random Forest (Classification & Regression)



$d' < D$ m'

Classification → majority voting classifier

Regression → Average

Time complexity will increase

Q why should we use RF instead of DT ?

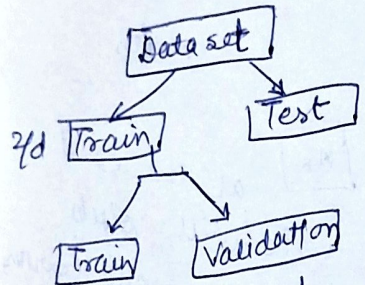
DT → overfitting

- Train Accuracy ↑ low Bias $\xrightarrow{RF}$ low bias
- Test Accuracy ↓ high variance $\xrightarrow{RF}$ low variance

---

Out of Bag Evaluation (OOB)

sklearn



100 → data points were not used

- hyper parameters (tuning) using Grid Search CV (Cross Validation) 40 data points

- OOB score → in sklearn use OOB = True

Percentage %

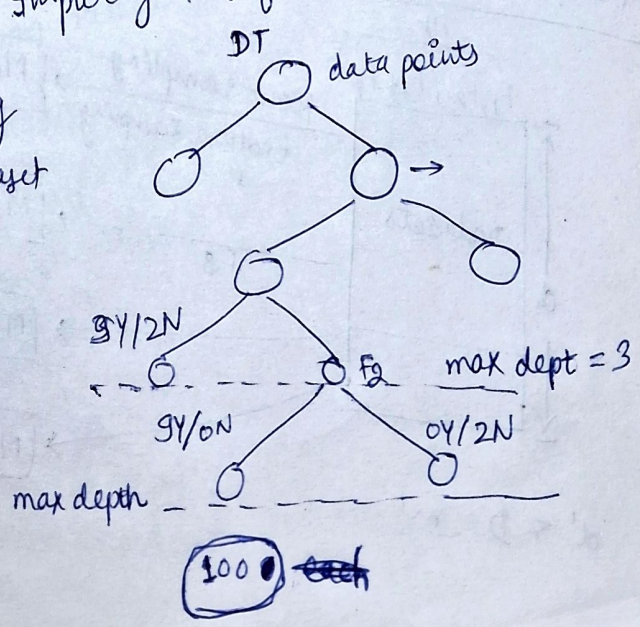
Random forest (Classification & Regression)

Decision Tree → Entropy, Gini Impurity & Information

Pre pruning & Post Pruning

large datasets small dataset

overfitting reduce ↓



Grid Search CV

- It is a technique in machine learning used to find the best combination of hyper parameter for a model.
- It systematically ~~try~~ tries every possible combination of hyper parameter values you specify and evaluate the model performance for each combination using cross validation.

Ex: Hyperparameter C's values & kernels for SVM (code)

Cross Validation → Split the data into multiple folds to ensure the model perform well on unseen data.

Grid: → A list of all combination of hyper parameters to try.

CV → Evaluate each combinations on multiple data split.

Deadlog (decision tree) ?? Dedd logram.

Train, Test & Validation

Data set

f_1 f_2 f_3 f_4 O/P

CV = 5

CV = 1 | Valid 1600 | Train 6400 | → Acc1

CV = 2 | Train | 1600 | Train | Acc2

} Average mean

Hyperparameter tuning

random - state → divisible by 2 (pick that value only)

Types of CV

1 → LOOCV [Leave one out CV] → not recommended for using

2 → HOCV (Hold out CV)

3 → K fold CV

4 → Stratified K fold CV

5 → Time Series CV.

Dataset 1300 records

300 (data leakage)

Test

LOOCV

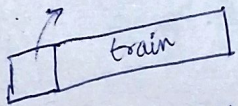
1000

Train

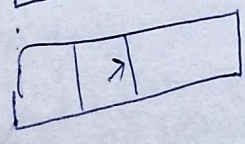
Train

Valid

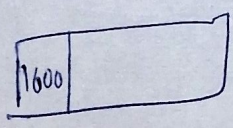
Leave one out → CV=1



CV=2



HOCV CV = 1



Disadvantage

- Time Consuming Task
- overfitting

Same for HOCV

No validation dataset

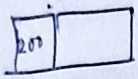
Train dataset accuracy

HOCV → holds the data points

70% & 30% ratio → Train accuracy will be ↑

with the help of random state it will for some time.

200 800 → Experiments.



K fold CV

k = 10 experiments

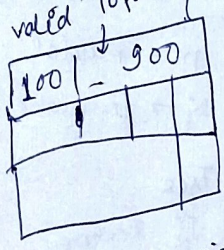
k → hyperparameter

$\frac{1000}{11} = 100$

→ These cannot work on imbalanced dataset. → HOCV, → LOOCV

K fold CV

$\frac{1000}{11} = 100$



CV = 1 Acc1

CV = 2 Acc2

CV = 3 Acc3

K → value for k is 10

iterations → cv values

10 folds

[ ] → data points

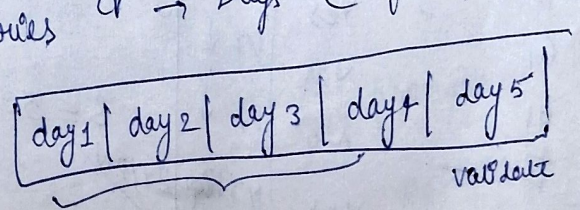
K fold doesn't work with imbalanced dataset.

Stratified K fold

Train & validate → Equal or ≈ Equal all the categories of output

→ Imbalanced dataset upto some extent

Time Series CV → Days (Sequence) or Series (for better accuracy use hyperparameter)



Boosting

- Bagging
  - RFC
  - RFR
- Boosting
  - AdaBoost → Adaptive
  - Gradient → XG Boost
  - CAT Boost

sequential DT connect

Predict → Acc

Datasets → DT_1 — DT_2 — DT_3 ---- DT_n → Avg of wt

DT_2 → weak learner

weak learners → Haven't learnt much from training dataset

AdaBoost → Assignment a weight to the weak learner

$f = \alpha_1 m_1 + \alpha_2 m_2 + \alpha_3 m_3 \text{ ---- } \alpha_n m_n$

m = model

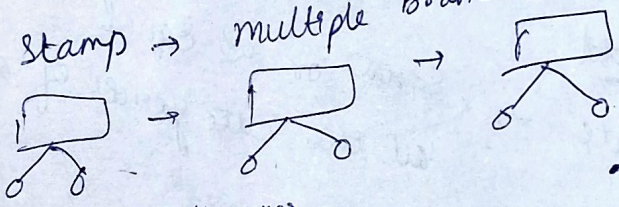
α → to make WL → SL

m_n → Decision Tree

classification, regression

$\alpha_1, \alpha_2, \alpha_3$ → weights

Decision Stamp → multiple branches will not be formed



↳ weak learner

- we create DT stamp & we select Best DT stump using entropy, gini, information gain.

Normalized wt	Dataset updated wt	Salary	Credit	Approved	error
0.08	0.058	<= 50K	B	No	1/7 ↓
	"	<= 50K	G	Yes	1/7 ↓
	"	<= 50K	G	Yes	1/7 ↓
	"	<= 50K	B	No	1/7 ↓
	"	> 50K	G	Yes	1/7 ↑
		> 50K	N	Yes	1/7 ↑
0.08	0.349	> 50K	N	No	1/7 ↑
	0.058	<= 50K			

$S \le 50K$

- 2/12N → Yes
- 2/12N → NO

which condition isn't satisfied

C == G >50 N (yes) → error

3Y/6N 1Y/3NO

(Y) (N)

Step one → Sum of the total error & performance of stump.

Total sum of error = 1/7

Performance = $\frac{1}{2} \ln \left\{ \frac{1 - TE}{TE} \right\} = \frac{1}{2} \ln (6) \approx 0.896$

$\alpha_1 = 0.896$ → weight

3) update the weights correctly & incorrectly classified points

a) wt × e^{-Performance of stump} (increase wt of errors)

$1/7 \times e^{-0.896} = 0.058$

0.058
×8
0.348
0.349
0.497

b) wt × e^{performance of stump}

$1/7 \times e^{-0.896} = 0.349$

4) Normalize wt computation & assigning bins

Assigned bins

0 – 0.08

0.08 – 0.16

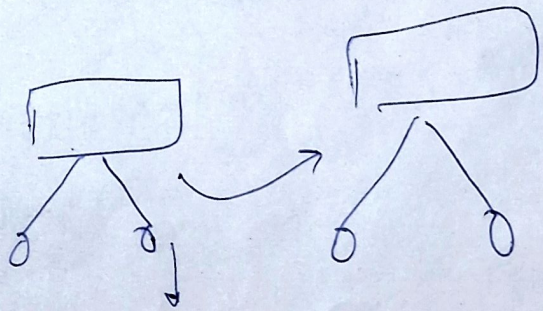
0.16 – 0.24

0.24 – 0.32

0.32 – 0.40

0.40 – 0.90

0.90 – 0.98



To prepare datapoints

5) Select datapoint to send to Next stamp

* Iteration process selecting random value b/w 0 & 1.

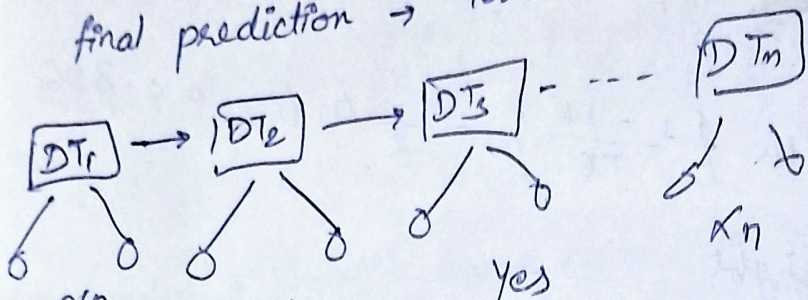
Random

$\alpha = 0.75$

$f = \alpha_1 m_1 + \alpha_2 m_2 +$

$f = 0.896$

final prediction → Test data ≤ 50K, Gr



$\alpha_2 = 0.62$ $\alpha_3 = 0.34$

$\alpha = 0.896$

$f = 0.896\,(Yes) + 0.62\,(No) + 0.24\,(Yes) + \cdots \alpha_n\,(Yes)$

$f = 1.136\,(Yes) + 0.350\,(No)$

previously weak learners → weight & assign bins values

adapt → strong learners.

Gradient Boosting

- Regression
- Classification

Reg Dataset

Exp	Degree	Salary	Predicted $\hat{y}$	Residual error $y-\hat{y}$ (R_1)	Predicted R_2
2	BE	50K	75K	-25K	-23K
3	Master	70K	75K	5K	-3K
5	Master	80K	75K	5K	3
6	Phd.	100K	75K	25K	23K
		mean 75K			↓ Assumed from previous error

Step 1 → Create Base Model

$$\frac{50K + 70K + 80K + 100K}{4} = 75K$$

75K

learning rate [0, 1]

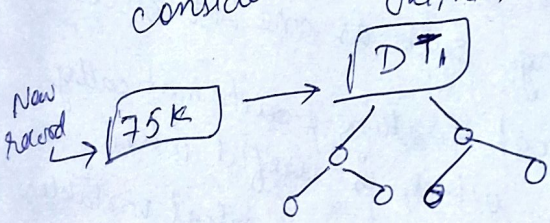
α

Step 2 Compute residual error

Step 3 Construct a DT

consider I/P x_i & O/P R_i

(x_i, R_i)



Predict O/P =

$75 + (-23) = 52$

$75 + (-3) = 72$

} Boosting weak learner

generalized mode

$$f(x) = \alpha_0 h_0(x) + \alpha_1 h(x_1) + \alpha_2 h(x_3) + \cdots \alpha_n h(x)$$

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

XG Boost

~~extent~~ extension gradient boosting

- It is optimised distributed gradient boosting library designed to be highly efficient, flexible and portable. It provide a parallel tree boosting (also known as gradient boosting decision tree, GBDT, GBM (machine)

Key features

- Regularisation → Help to reduce overfitting
- Parallel processing → utilize multiple CPU cores to train model faster.
- Handling missing values → automatically learn the best way to handle missing data.
- Cross validation → Built in CV support.
- ~~CAT Boost~~.

CAT Boost → This ~~strong~~ gradient boosting algo handles categorical features automatically, without the need for explicit pre processing such as one hot encoding.

Key features, Handling categorical feature; automatically process categorical feature which is useful when dealing with dataset that contains many categorical variables

Robust to overfitting → provide better generalization model on test set

Efficient → faster training due to order boosting boosting.

Comparison of XG Boost, Gradient Boosting & cat Boost

features	XGBoost	Gradient Boost (sklearn)	catBoost
Categorical data handling	Requires preprocessing	Requires preprocessing	Built in Support
Speed	Fast with ll processing	moderate	fast, optimized for categorical data
Model Tuning	Highly customizable	Simple	highly customizable
Visualization Support	Tree & feature importance	feature importance	"
Training time	Moderate	Longer	faster

Principal component Analysis (PCA)

Dimension Reduction

x	y	$x-\bar{x}$ x'	$y-\bar{y}$ y'	$x'y'$	x_i^2	y_i^2
2.5	2.4	0.63	0.4	0.33	0.4761	
0.5	0.7	-1.31	-1.26	1.5585	1.7111	
2.2	2.9	0.39	0.99	0.3866	0.1521	
1.9	2.2	0.09	0.29	0.026	0.0081	
3.1	3.0	1.29	1.09	1.4061		
2.3	2.7	0.49	0.79	0.3871		
2	1.6	0.19	-0.31	-0.0589		
1	1.1	-0.81	-0.81	0.6561		
1	1.6	-0.31	-0.31	0.0961		
1	0.9	-1.01	-1.01	0.7172		
MEAN 1.81	1.91			5.539	5.545	6.449

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

$$AI = \lambda I = |A - \lambda I|$$

characteristic equation.

$$\begin{bmatrix} 0.5549 & 0.5539 \\ 0.5539 & 0.64499 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0.5549 & 0.5539 \\ 0.5539 & 0.64499 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 0.5549 - \lambda & 0.5539 \\ 0.5539 & 0.64499 - \lambda \end{bmatrix}$$

$$\det[A - \lambda I] = (0.5549 - \lambda)(0.6449 - \lambda) - (0.5539)$$

$$= (\lambda - 0.04908)(\lambda - 1.2840) = 0$$

λ.

$\lambda_1 = 0.04008$ $\quad$ $\lambda_2 = 1.2848$ $\rightarrow$ Eigen values

Eigen vector $\rightarrow$ Direction

$\lambda_1 = 0.04908$,

$$B = \begin{bmatrix} 0.50582 & 0.5539 \\ 0.5539 & 0.59502 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$Bx = 0$ Augmented matrix

$0.50582\, x_1 + 0.5539\, x_1 = 0$

$0.5539\, x_1 + 0.59582\, x_2 = 0$

Eigen vector $\lambda_1 = 0.04908 = \begin{bmatrix} -0.735178 \\ 0.67787 \end{bmatrix}$

Eigen vector $\lambda_2 = 1.2840 = \begin{bmatrix} -0.67787 \\ -0.735178 \end{bmatrix}$

$\lambda_2 >> \lambda_1 \rightarrow$ PC 1

$PC_1 \quad PC_2$

larger λ value is PC1

smaller λ value is PC2

$$\begin{bmatrix} 0.5549 & 0.5539 \\ 0.5539 & 0.64499 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0.5549 & 0.5539 \\ 0.5539 & 0.64499 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 0.5549 - \lambda & 0.5539 \\ 0.5539 & 0.64499 - \lambda \end{bmatrix}$$

$$\det[A - \lambda I] = (0.5549 - \lambda)(0.6449 - \lambda) - (0.5539)$$

$$= (\lambda - 0.04908)(\lambda - 1.2840) = 0$$

λ.

$\underline{\lambda_1 = 0.04008}$ $\quad$ $\underline{\lambda_2 = 1.2848}$ $\rightarrow$ Eigen values

Eigen vector $\rightarrow$ Direction

$\lambda_1 = 0.04908$,

$$B = \begin{bmatrix} 0.50582 & 0.5539 \\ 0.5539 & 0.59502 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$Bx = 0$ Augmented matrix

$0.50582\, x_1 + 0.5539\, x_1 = 0$

$0.5539\, x_1 + 0.59582\, x_2 = 0$

Eigen vector $\lambda_1 = 0.04908 = \begin{bmatrix} -0.735178 \\ 0.67787 \end{bmatrix}$

Eigen vector $\lambda_2 = 1.2840 = \begin{bmatrix} -0.67787 \\ -0.735178 \end{bmatrix}$

$\lambda_2 >> \lambda_1 \rightarrow$ PC 1

$PC_1 \quad PC_2$

larger λ value is PC1

smaller λ value is PC2

Feature selection vs feature extraction (Dimensionality Reduction)

3D data → 2D data

100 d → 3D to 2D

- Prevent → curse of dimensionality
- Improve → The performance of the model
- Visualize the data → understand the data.

After calculating the covariance (it's used for feature selection) we ~~do~~ use feature extraction for dimensionality reduction.

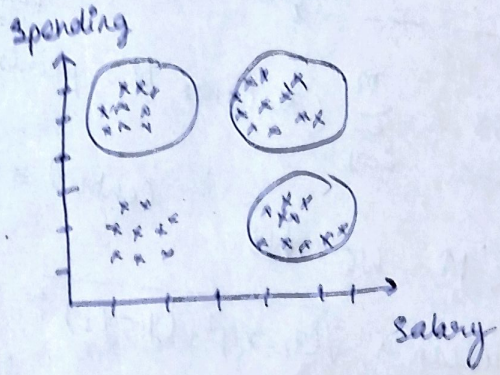
Unsupervised Learning

no o/ps only features.

Supervised learning

f_2	f_1	f_3	f_4	f_5	O/P
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

f_1 f_2 f_3 f_4 samsung



K-means clustering

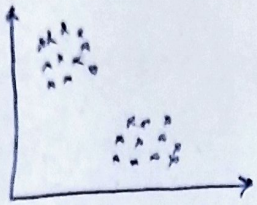
Hierarchical clustering

DBSCAN clustering.

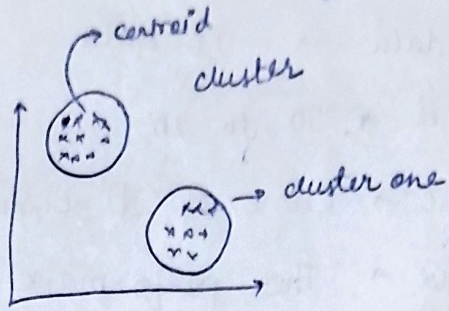
Unsupervised Learning

K=2

k-means clustering algo



K-means →



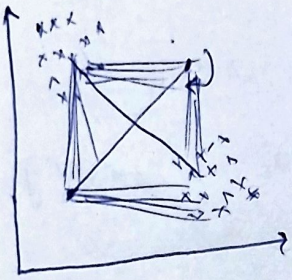
Step one: Initialize some k centroids

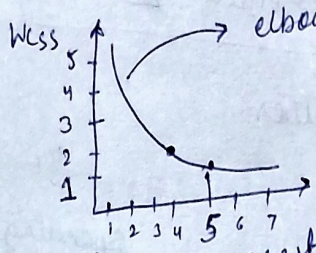
Step two: Points that are nearest to the centroid → Group

Step three: Move the centroids → Average.

How do we select the K-value?

WCSS = within the cluster sum of square





$$WCSS = \sum_{i=1}^{m} (\text{distance b/w points to nearest centroid})^2$$

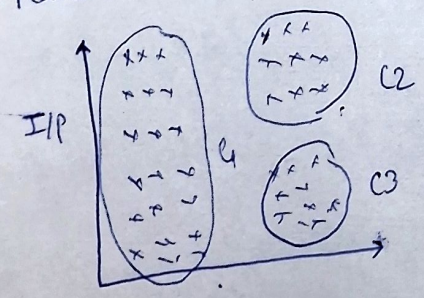
Euclidean & Manhattan Distance.

Centroid → K

K ↑↑ WCSS ↓↓

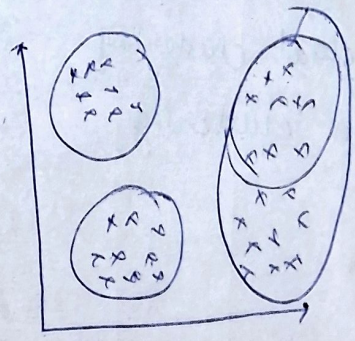
Euclidean → $\sqrt{(x_2-x_1)^2+(y_1-y_2)^2}$ Manhattan = $|x_2-x_1| + (y_2-y_1)$

Random Intialization TRAP (K-means ++)



K-mean

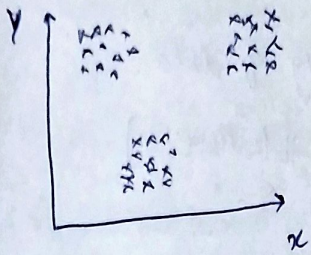
O/P

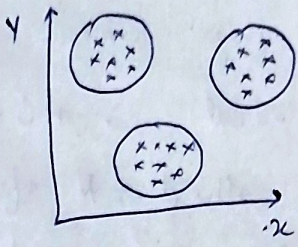


Takes wrong (Predicted) values → corrects them

Unsupervised Learning

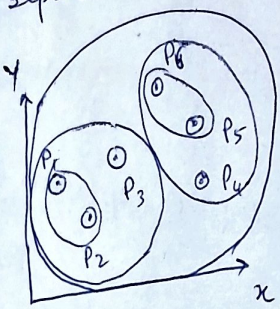
Hierarchical Clustering → No centroid available





Step 1 → for each point initialize we will consider it as a separate center

Step 2 → find the nearest point create a new center.



Step 3 → Keep on doing the same step first process until we get a single cluster.

Top to bottom ↑ (aggro)

Bottom to Top ↓ (divisive)

Euclidean distance.

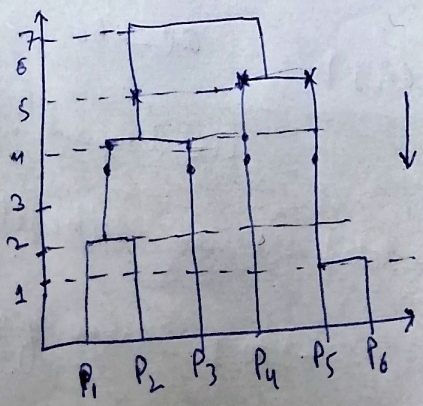
agglomerative
divisive
} Geometrical Intuition

Threshold value → select the longest vertical line such that no horizontal line passes through it.

Euclidean distance

Dendrogram

Dendro →



agglomerative

such vertical line from which no horizontal line is passing through.

threshold = 5

K means vs hierarchical

- Stability and flexibility
- Dataset size — small HC, large K-means
- K-means → Numerical · HC → variety of data

Centroid → Elbow method : No. of centroid [K means]

HC Centroid : not present

DB Scan

min points = 4

ϵ = radius.

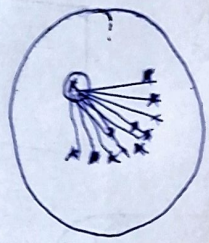
- Core point
- Border point
- outlier

Core point → No of points within the ϵ (epsilon) should be greater than & equal to four.

Border point → No of data points within the radius will be less than minimum points.

* DBScan is robust to the outliers

Method for validation → for all clustering using silhouette clustering.



a(i)

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} (d_{i,j})$$

C_i = datapoint i in the cluster

$a_i << b_i$

$$2)\ b(i) = \min_{J \neq i} \frac{1}{|C_j|} \sum_{i \in C_j} d(i, j)$$

C_i = datapoint i in the cluster

In clustering score is calculated through silhouette score

Score $S(i) = b(i) - a(i)$ / $\max\{a(i), b(i)\}$

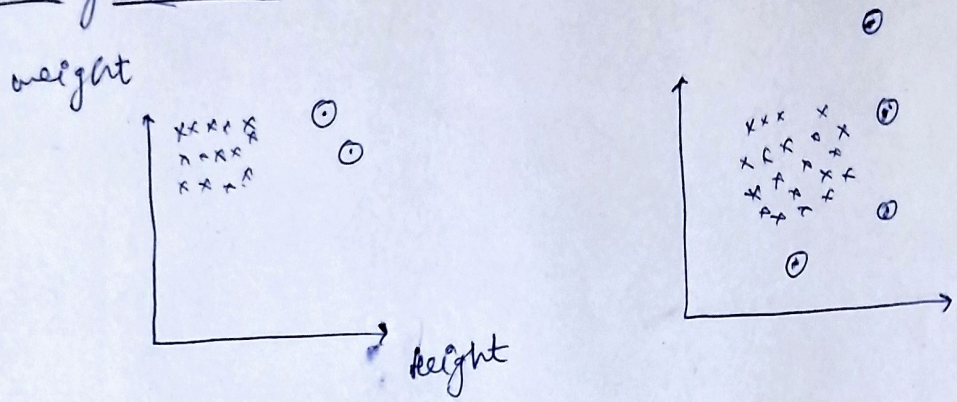
-1 to 1 Range of silhouette score.

Criteria $S(i)$ $\begin{cases} 1 - a_i/b_i & \text{if } a_i < b_i \\ 0 & \text{if } a_i = b_i \\ b_i/a_i - 1 & \text{if } a_i > b_i \end{cases}$

if $a_i > b_i$ = good cluster

if $a_i < b_i$ = bad cluster

<u>Anomaly Detection</u> : To detect outliers

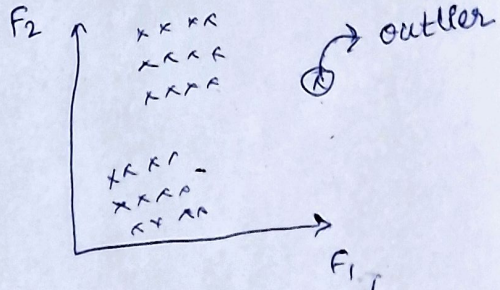


It important for problem statement

• Isolation Forest : Decision Tree (making different type of isolation tree)

lets consider

f_1	f_2	f_3	f_4
–	–	–	–
–	–	–	–
–	–	–	–



will work until it gets single leaf node not found.

math $S(x, m) = 2\left[\frac{-E\,h(x)}{c(m)}\right]$ since m = total no. of data points

x = data points (isolation data points)

$E(h(x))$ = Average search depth of x from isolated tree.

$C(m)$ = Average depth of all data points

$E\,h(x) << C(m) \Rightarrow S(x, m) \approx 1$ or near about one. outlier

$E(h(x)) >> C(m) \Rightarrow S(x, m) \approx 0.5$ normal datapoints

anamoly score

if it is near one than its outlier

• first we define threshold 0.5

Associate Rule Mining (Learning)

- Market - Basket Analysis → work on transactions
- Aim → Interset patterns [m/extend frequent patterns]
- Terminologies → Item set K/item {bread}, {milk} and etc.

→ Support count (Σ) frequency of dataset given set of transactions

→ frequent item set greater than and equal to minimum support threshold.

→ Support ("s") s is the percentage of transactions in dataset that contains: ~~A and B~~ A & B (A ∪ B) ⇒ s = Prob(A ∪ B)

→ Confidence (c) c is the percentage of transaction containing A that also contain B.

$C = P\left(\frac{P}{P(A)}\right)$ or $P\left(\frac{P(A \cup B)}{P(A)}\right)$

Concept: Lift 1 A → B Lift 1 = $\frac{S(A \& B)}{S(A) * S(B)}$ lift 2 = independent items

~~Support A & B~~

lift is > 1 {positive effect of item on each other}

lift is < 1 {negative effect of item on each other}

Strong Association Rule (which item to be picked)

calculate s & c equal. we define a minimum threshold. less number of association rule in dataset when increase satisfied both minimum threshold both support and decrease minimum threshold ~~coff~~ confidence also. (I am sleepy :( )

2:36 pm 3rd Jan 2025

So much sleepy!! UGHHH ...

→ Mining process →←

- find all frequent item sets
- generate strong associate rules from the frequent item set.
- Support count :- frequency of item sets

Algorithm used → Apriori

uses K item set, (K-1) → used to find frequent item set

<u>Properties of Apriori</u> → All non empty subset of frequent item set must also be frequent. For example

{A, B} → {A} {B}

<u>Anti monotonocity Property</u> → If item set i doesn't satisfied the minimum support threshold than i is not frequent. If an item A is added to the item set i. (i ∪ A) can't occur more frequently than i.

<u>Conclusion</u> → If our confidence is approx. equal to 1 we say it's a strong association analysis.

Perceptron Model

↓

Single layer

linear seperable (Binary classification)

Non linear seperable

↓

Multilayer NN.

↓

Multilayer (Artificial intelligence)

- Forward Propogation
- Backward Propogation
- Loss function
- Activation function
- Optimizers (Gradient Descent)

8th Jan

Forward Propagation, Backgward propagation, vanishing gradient, ~~clear~~ chain rule propagation

dataset	IQ	StudyHr	PlayHr	Pass/Fail O/P
	95	4	4	1
	100	5	2	1
	95	2	7	0

bias → always applied on the output of hidden layer

Input → o/p : forward propagation

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

Let us $\begin{Bmatrix} w_1, & w_2, & w_3 \\ 0.01, & 0.02, & 0.03 \end{Bmatrix}$ $\begin{Bmatrix} b \\ 0.001 \end{Bmatrix}$

$= 95 \times 0.01 + 0.001$

$100 \times \cancel{0.02} + 0.001$

$\cancel{95 \times 0.03 + 0.001}$

$Z = 1.151$

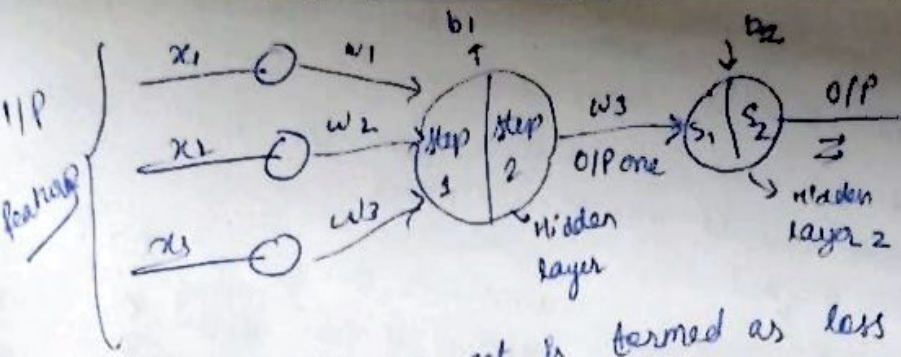
$Z = [95 \times 0.01 + 4 \times 0.02 + 4 \times 0.03 + 1 \times 0.001]$

$= 1.151$

Step 2 → Sigmoid fn $= \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-1.151}} = 0.759$

$o_1 w_4 + b$ → add this to hidden layer's o/p to get the final output

Ayushi



FORWARD PROPAGATION

whatever final o/p we get is termed as loss as compared to real data the value will be lesser. (loss Error)

Loss fn

Regression
- MSE
- MAE
- Huber loss

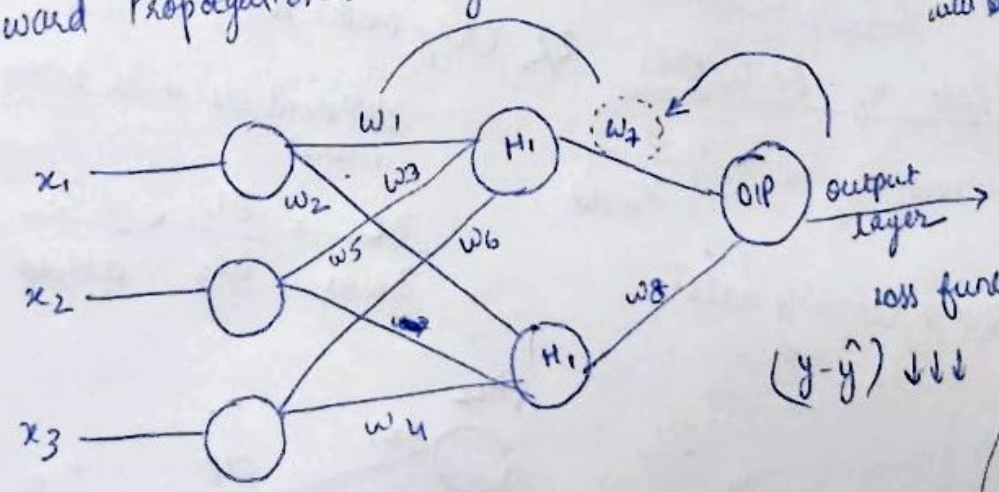
classification
- Binary cross Entropy
- Categorical cross entropy

Vanishing → sigmoid function (0 to 1) sometimes it reduces too much so it makes tough to identify the global minima. [Gradient Descent]. (Replace with ReLU)

middle value 0.25

sigmoid reduces too much the value will ~~shift~~ jump

Backward Propagation & weight updation

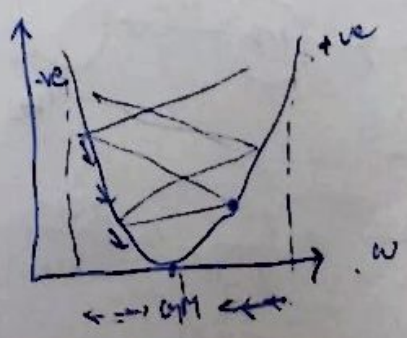


Updation formula

$$w_{7_{new}} = w_{7_{old}} - \eta \frac{\partial L}{\partial w_{7_{old}}}$$

slope loss

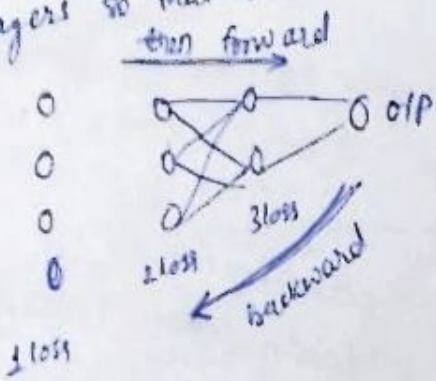
learning rate



$\eta = 0.002$

Back propagation → whenever ~~the~~ come moving backwards & reduce thousands of hidden layers & drop out layers

O/P loss should be minimize, but there's some loss within your hidden layers so that should also be minimized.



whatever loss you get try to reduce it near about 0. ↓ Global minima

continuous cycle.

update weight & bias to reduce loss.

optimizer (Gradient Descent) → to reduce the loss value

$w_{7new} > w_{7old}$ $\boxed{w_{new} > w_{old}}$ **

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

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

vanishing when sigmoid fn stops giving o/p.

When w reaches global minima :-

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

∵ Chain Rule of Derivatives ??. (why maths ∵)

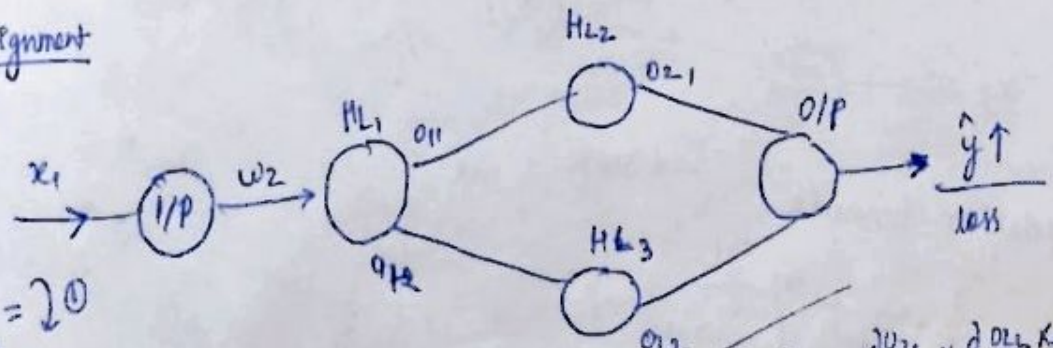
$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

differentiate → at every point (loss)

$$\frac{\partial L}{\partial w_{old}} = \frac{\partial L}{\partial o_3} \times \frac{\partial o_3}{\partial w_{old}}$$

movement → learning rate

Assignment



$\frac{\partial L}{\partial w_{old}} = 2①$

$$\frac{\partial L}{\partial o_{31}} \times \frac{\partial o_{31}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial o_{ld}}$$

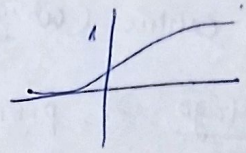
$$\frac{\partial L}{\partial o_{31}} \times \frac{\partial o_{31}}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial w_{old}}$$

②

Activation function

① Tan h ② Relu ③ Prelu ④ Swish

majorly used in place of sigmoid



Sigmoid f/n [0 to 1] $\sigma(x) = \frac{1}{1+e^{-z}} \Rightarrow 0$ to 0.25

differentiate

$$z = \sum_{i=1}^{n} w^T x + b$$

$\sigma(z) = 0, 1$ $\frac{\partial \sigma(z)}{\partial (z)} = 0$ to 0.25

$\phi(z)$

Advantages → It's suitable for binary classification

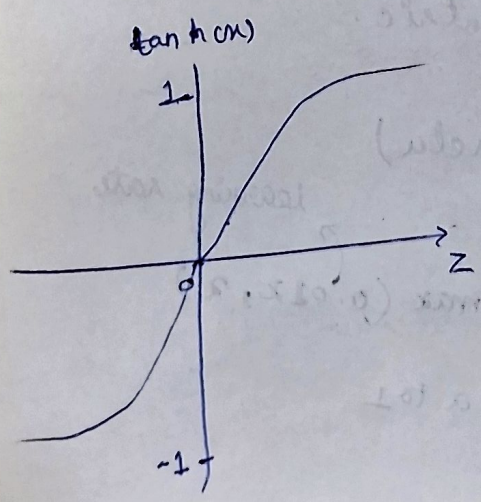
clear predictions that are very close between 0 to 1

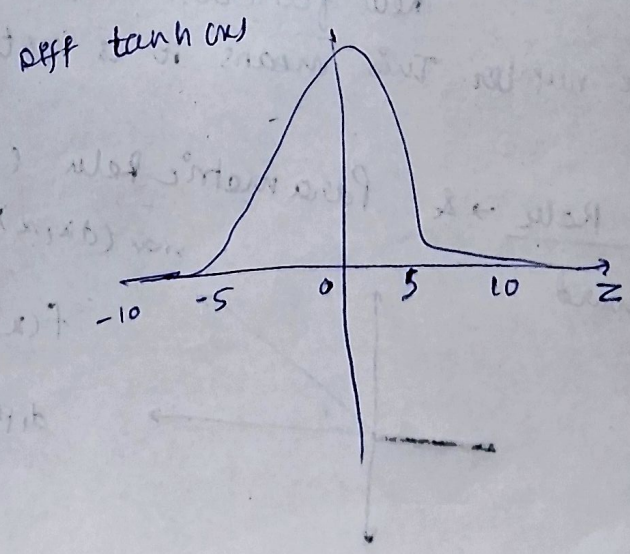
Disadvantages → prone to vanishing gradient descent

- function output is not zero centered
- Mathematical operations are relatively time consuming.

Tan (H) activation function [Tangent Hyperbola]

forward propagation $\tan h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ [-1 to 1]



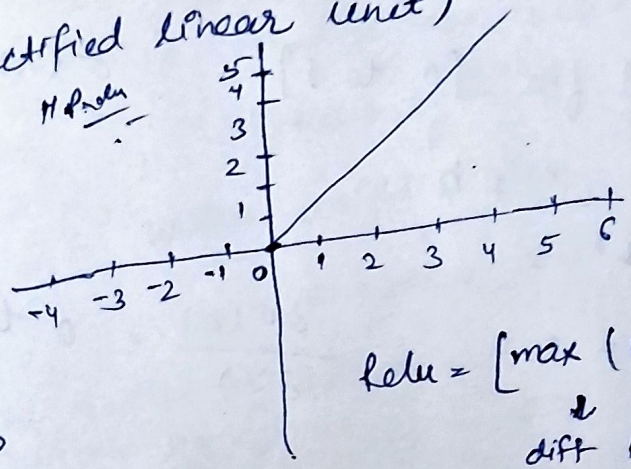


Advantages

zero centric (weight updation efficient)

Disadvantage → prone to vanishing gradient problems.
Time complexity.

Relu Activation fn (Rectified linear unit)



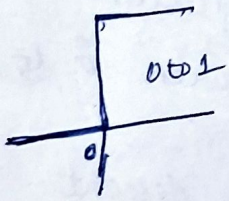
- tanh → -1 to 1

df → 0 to 1

- sigmoid → 0 to 1

diff → 0 to 2.5

Relu = [max (0, x)]
↓
diff Relu
stepwise



x = 0
dead neuron
$w_1 x_1 = 0$

Advantage → solving vanishing gradient problems
maximum (0, x) → calculation is super fast.

The Relu function has a linear relationship
It is much faster than sigmoid or tan h

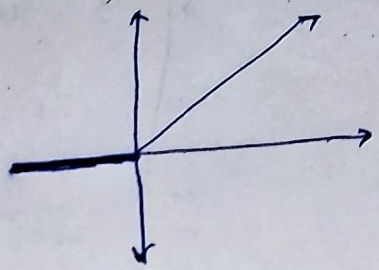
Disadvantage → dead neurons
Relu function output 0, x is equal to zero or positive number. This means it is not centric.

Leaky ~~Leaky~~ Relu → & Parametric Relu (P-relu)

max (dx, x)

learning rate

forward



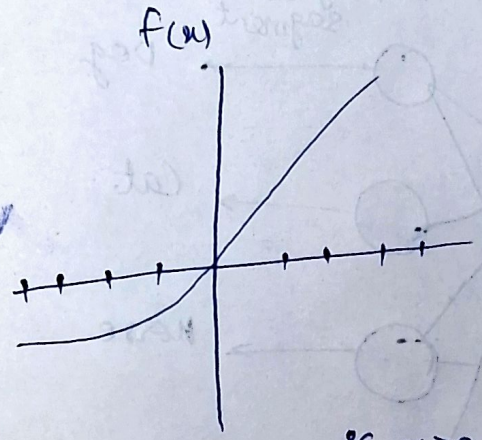
f(x) = max (0.01x, x)

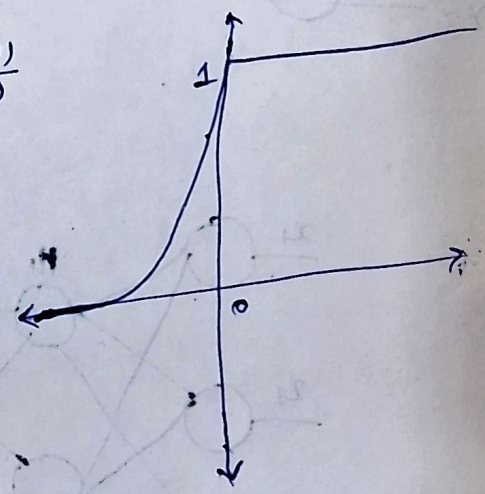
diff = α to 1

Relu → dead neuron → dead neuron problem

- Leaky relu has all the advantage of Relu.
- It removes dead relu problems
  ↳ neuron
- Parametric & leaky relu are same (working based)

Exponential Linear Unit (ELU)





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

It is used to solve relu problems

Advantage → No dead neuron issue
zero centric

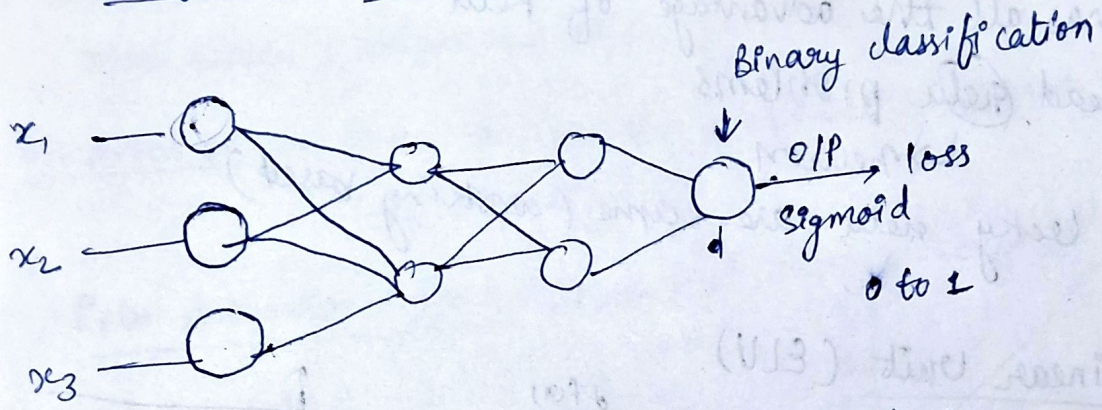
Disadvantage → slightly more computationally intensive

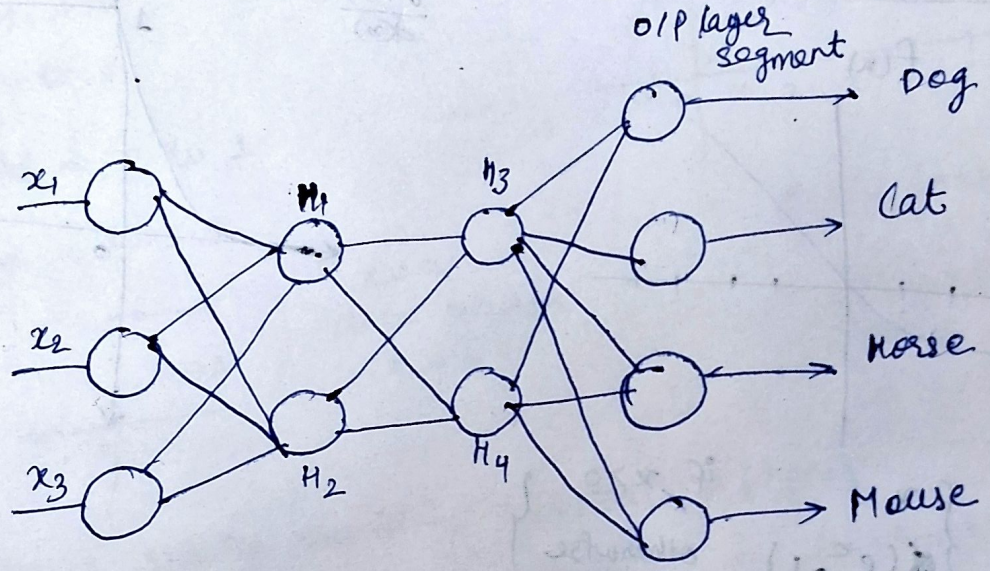
All activation fn used for binary classification

For multiclass classification, we use Soft-Max Activation function.

all Activation fn → diff → backward

Soft Max Activation function





Activation formula $\frac{e^{y_i}}{\sum_{k=0}^{n} e^{y_k}}$ $y_i = O \times w + b$

↳ output

$k \to ?$

Optimizers
- GD
- SGD
- Mini batch SGD
- Adagrad and RMS PROP
- Adam optimizer **

GD O/P loss ↓↓

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

↓

iteration

1000 datapoints
↓
all in forward
↓
all in backward
weight updation

1 Epoch = 1 iteration

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

Advantages of GD → convergence will happen

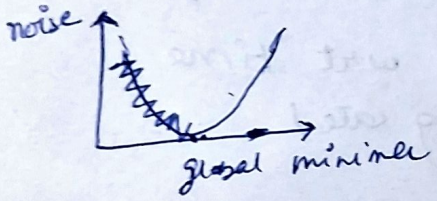
disadvantages → Huge usage of RAM & GPU

3:03 PM × worst time after lunch

~~Sto~~ stochastic GD :(Sleepy head → Ayushi

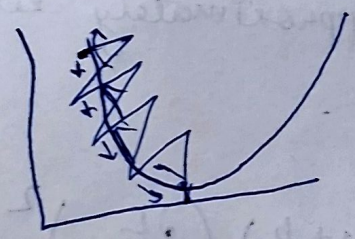
Advantage → solves resource issue –

Noise : (Bias?)



disadvantage → time complexity is increased.
convention or convergence will also
noise get introduced..

Mini Batch



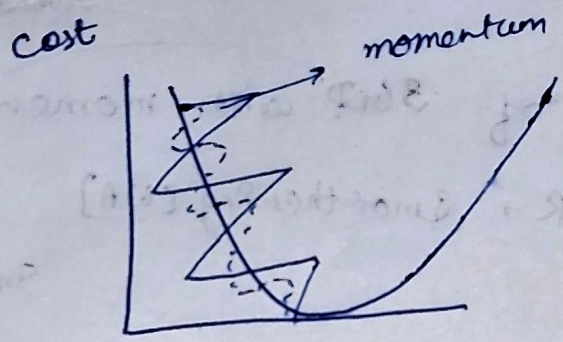
Advantages
convergence speed will increase
noise will be less when compared to SGD.

Advantage → sufficient resource uses (RAM, Ram)

disadvantages. Noise still exist
→ noise still exist

Experimental wt Avg.

Moment



~~Smoothening~~
for updating batch as increase every data point
no dp should be missed
used in ARIMA
SARIMA

Advantage :- Reduce the noise and quick conversion

As the conversion happens the learning rate should be changed (Ada grad)

ada delta , RMS Promp ??

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$

$$\eta' = \frac{\eta}{\sqrt{\alpha^t + \epsilon}} \rightarrow \text{epsilon}$$

↳ changes wrt time (learning rate)

whenever,

$\eta'' \uparrow\uparrow \quad \alpha_t \downarrow\downarrow$

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

Disadvantage → η' = possibility to become a very small value approximately zero.

Ada delta And RMS Prop

$$S_{dw_t} = \beta \times S_{w_{t-1}} + (1+\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

$\beta = 0.95 \qquad S_{dw_t} = 0$

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

Dynamic LR + Smoothening EWA

Exponential weight Average

ADAM → Combinations of SGD with momentum + RMS Prop [Dynamic LR + Smoothening EWA]

Smoothening Gradient Descent

(Best optimizer)

Weight Initialization Technique

1. Uniform Distribution
2. Xavier / Glorot ~~Distribution~~ / Initialization
3. Kaiming He Initialization

Key points why we use ??

→ weight should be small

→ weight should not be same

→ weight should have good variance

Uniform Distribution $w_{ij} \approx$ Uniform $\left[\frac{-1}{\sqrt{input}}, \frac{1}{\sqrt{input}}\right]$ $-1/\sqrt{3}$, $1/\sqrt{3}$

Xavier & Glorot → Research based

Normal

$w_{ij} \approx N(0, \sigma)$

$\sigma = \sqrt{\frac{2}{I/P + O/P}}$

uniform

$\left(\frac{\sqrt{-6}}{\sqrt{I/P + O/P}}, \frac{\sqrt{6}}{\sqrt{I/P + O/P}}\right)$

Kaiming He Initialization

He normal

$\sigma = \sqrt{\frac{2}{I/P}}$

He uniform

$\left[\sqrt{\frac{-6}{I/P}}, \sqrt{\frac{6}{I/P}}\right]$