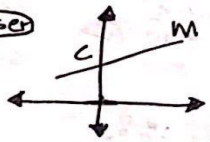
Supervised Learning: TERM-1

Fitting a line through Data:

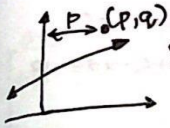


$y = mx + c$

C: Intercept
m: slope

OR $\boxed{y = w_1 x + w_2}$

Methods: ① Absolute Trick:-



$y = mx + c$
$+p\alpha \quad +1\alpha$

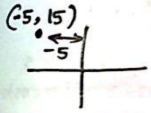
α: learning rate

∴ New equation: $\boxed{y = (m + p\alpha)x + (c + \alpha)}$

But what if point is Below the line?

→ Instead of Adding we Subtract ⊖

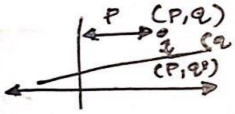
$\therefore y = (m - p\alpha)x + (c - \alpha)$



$\alpha = 0.1$ $\quad y = \underline{2x} + \underline{3}$
$\quad\quad -0.5 \quad +0.1$

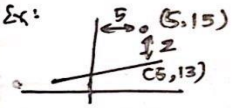
$\therefore y = (1.5)x + (3.1)$

② Square Trick:



$y = mx + c$
$+p(q-q') + \alpha(q-q')$

Ex:

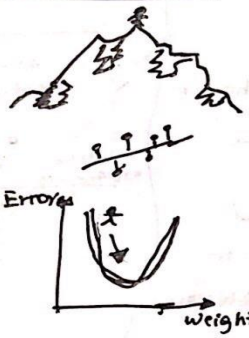


$\alpha = 0.01$
$y = mx + c$
$\quad 0.1 + 0.02.$

$\therefore y = (2.1)x + (3.02).$

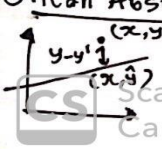
Gradient Descent:

$\boxed{w_i \to w_i - \alpha \frac{\partial}{\partial w_i} \text{Error}}$



Error Functions: (For linear Regression)

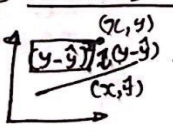
① Mean Absolute Error



$\text{Error} = \frac{1}{m}\sum_{i=1}^{m} |y - \hat{y}|$

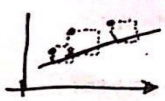
m = no. of points

② Mean Squared Error

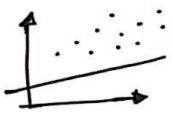


$\text{Error} = \frac{1}{2m}\sum_{i=1}^{m} (y - \hat{y})^2$

Ex:



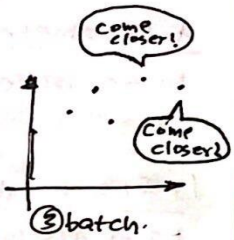
Batch & Mini-Batch & Stochastic:



• come close Mag by All points ☺

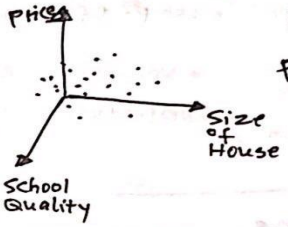
① Batch

② Stochastic.



③ batch.

Higher Dimensions:

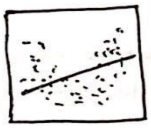


Predictions = A Plane

$\text{Price} = m_1(SQ) + m_2(SH) + c$

$\boxed{\hat{y} = m_1x_1 + m_2x_2 + m_3x_3 \cdots + m_{n-1}x_{n-1} + c}$

Linear Regression works best if the data is Linear

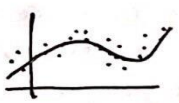


← Not in this case.

• If outliers are present ; → Drammatic changes in Model's behaviour!

∴ Polynomial comes In!!!

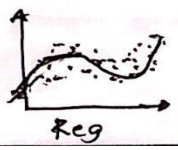
Polynomial Regression:

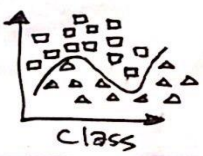


$\boxed{\hat{y} = m_1x^3 + m_2x^2 + m_3x + c}$

Polynomial can be used to both

1. Classification
2. Regression





Simple or Complex Model?

Depends on Application:

Requires low error; OK with its Complexity

Requires simplicity and fast.

λ-Parameter:

Miss classified x Added Error ⇒ Complex Model wins

L1 & L2 Regularization:

L1 → Absolute value

L2 → Squared value.

L1
- computationally inefficient.
- Sparse outputs
- Feature selection

L2
- comp- efficient
- Non-Sparse o/ps.
- NO feature selection

Feature scaling:

- FS is a way of classifying data into a common range of values. 2 common scaling:
  1. Sandardizing
  2. Normalizing

1. **Standardizing:** is taking each value of the column and Devide by Std of the column

2. **Normalization:** Data is scaled beteween 0 and 1.

When to use FS?:

In Most ML Algo the result will charge depending on the units of the Data. This happens in 2 cases:

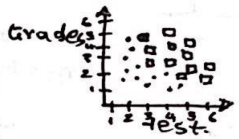
1. when Algo uses Distance based Metrics to predict Ex Knn, SVM etc.
2. when you incorporate regularization ex.

CLASSIFICATION:

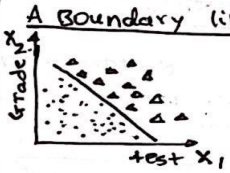
Example: Student Acceptance to college:

Student 1
test: 9/10
Grade: 8/10

Student 2
test: 3/10
Grade: 4/10



A Boundary line:



A line: $2x_1 + x_2 - 18 = 0$

i.e 2(test) + Grade − 18 = score

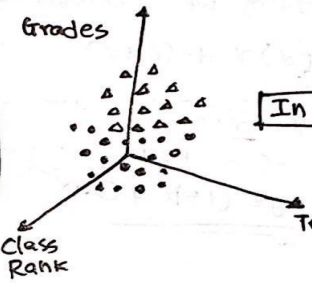
$m_1x_1 + m_2x_2 = 0$

Prediction:

score > 0 : Accept

score < 0 : Reject

What if there are 3 features?



In 3D

Now Boundary is Not line But a Plane.

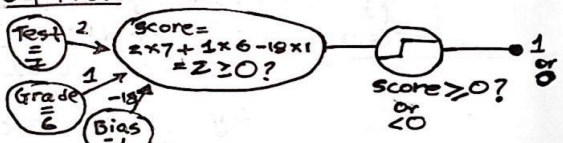
If n features then n−1 dimension hyperplane

i.e $m_1x_1 + m_2x_2 + m_3x_3 + b_2 \ldots + m_nx_n + b = 0$

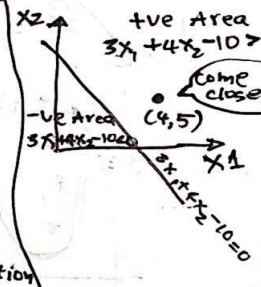
Predictions:

$$\hat{y} = \begin{cases} 1 & ; W^T x + b \geq 0 \\ 0 & ; W^T x + b < 0 \end{cases}$$

Perceptron:



Score = 2 * test + 1 * Grades − 18



3	4	−10
−4	5	1

Now to take small steps → use Learning rate: 0.1

∴

3	4	−10
−0.4	−0.5	−0.1

A line eqn which will be closeee to Point

If miss classified point is in Area 0 Add ⊕

If in 1 Area then ⊖

Decision Trees:

less Entropy + High knowledge Gain

log → will convert Product to Sum

$\therefore \boxed{\log(ab) = \log(a) + \log(b)}$

□□□□□ ○○○
5 3

$\text{Entropy} = -\frac{5}{8}\log_2\left(\frac{5}{8}\right) - \frac{3}{8}\log_2\left(\frac{3}{8}\right) = 0.9544$

or

$$\text{Entropy} = \frac{-m}{m+n}\log_2\left(\frac{m}{m+n}\right) - \frac{n}{m+n}\log_2\left(\frac{n}{m+n}\right)$$

if $P_1 = \frac{m}{m+n}$, $P_2 = \frac{n}{m+n}$

$$\text{Entropy} = -\sum_{i=1}^{n} P_i \log_2(P_i)$$

Information Gain: (Change in Entropy)

i.e. Parent 1 → Child1 0.72, Child2 0.72

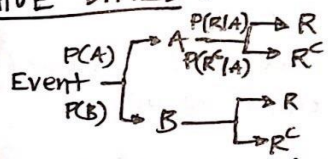
IG = Entropy(Parent) − Entropy(Child)

$IG = 1 - 0.72 = 0.28$

Hyper parameter:

1. Max_depth : Max No. of levels in the tree
2. Min_samples_leaf : Min. No. of samples Allowed in leaf
3. Min_samples_split : Min. No. of dp's Allowed in split
4. Max_features : The Number of features to consider when looking for best split.

NAIVE BAYES:



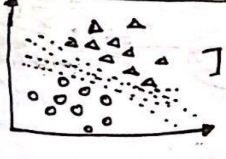
R: New event Happens

R^c: New event Doest Happen

W.K $P(R|A) = P(R \cap A) = P(A).P(R|A)$

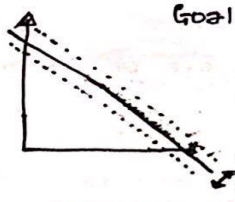
$$P(A|R) = \frac{P(A).P(R|A)}{P(A)P(R|A) + P(B)P(R|B)}$$

SUPPORT VECTOR MACHINE: (SVM)



] Many option to Boundary line But which one is Best?

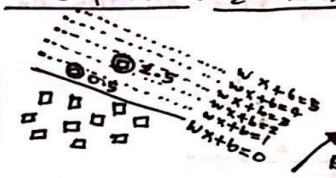
- keep the line which have large distance from closest dp's of both sides.

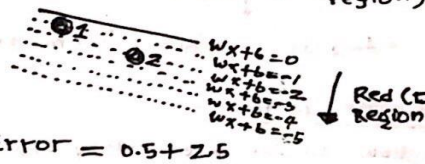


Goal:
- Minimize Error
- large margin width

ERROR = Classification Error + Margin Error

Perceptron Algorithm:





Total Error = 0.5 + 2.5 + 1 + 2 = 6

Idea is to use Gradient Descent to give the best possible cut.

→ Perceptron Algorithm.

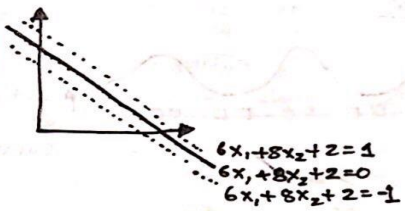
Margin Error:

- Small margin → Large Error
- Large margin → Small Error

Ex:

$m_1x_1 + m_2x_2 + b = 0$ $\quad |W|^2 = 3^2 + 4^2 = 25$

$3x_1 + 4x_2 + 1 = 0$ $\quad \Rightarrow$ Error = 25



This gives same error as L1+L2 Regularisation.

C-Parameter:

It's a constant that Attaches to Error

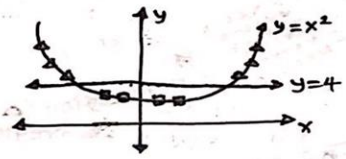
Error = C * Classification Error + Margin Error

1. If C is very large → we are focusing on Classification Error
2. If C is small then we are focusing on Margin Error
- small C → Large Margin → May make Classification Error
- Large C → May have small margin → classifies points well.

① Polynomial Kernel :-1.

- What if the points are on a line; not on 2D plane ?

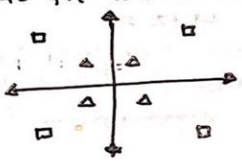
To seperate this with a boundary line we need to use a model → "Polynomial kernel"; it converts the 1D → 2D ; puts all dp's on a Parabola.



- And then convert from 2D → 1D.

② Kernel trick-2:

How to fit a boundary to this.

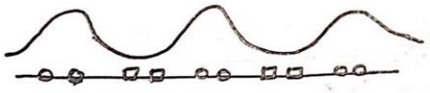


1. Fit All (Δ) in a circle.
2. Send (Δ)s to Another Dimension (lets say z-Axis).

RBF KERNEL :-

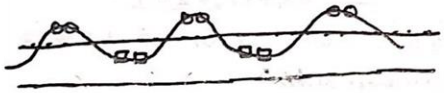
- Radial Basis Function (RBF)

Scenario:



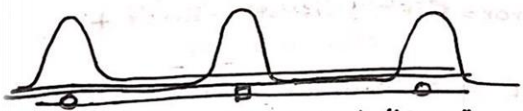
fit All dps in the curve

then



But the question is how to Do this ?

- Build mountains on top of every DP.



this technique is called "RBF"

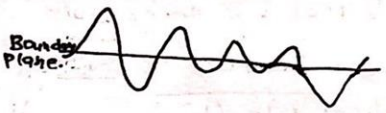
- Now combine all the Mountains

In Higher Dimension:

- Draw a Gaussian Parabaloid

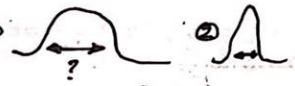
the seperate with a plane ; this plane will cut this Gaussian Parabaloid at some height ; And this → is a circle

This circle will give the boundary equation



In 3D. / N-D

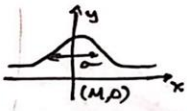
But How to decide the width of the Mountain ?



γ → decides their width.

∴ we use "γ Parameter"

- large γ → Narrow curve ②
- large γ → wide curve ①



"NORMAL DISTRIBUTION"

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

$$\gamma = \frac{1}{2\sigma^2}$$

Hyper Parameter : In SVC

- C : The C Parameter
- kernel : The kernel : The most common ones are 'linear', 'poly' and 'rbf'
- degree : If kernel is Polynomial; → This is the Maximum degree of the monomial in the kernel
- gamma : If kernel is RBF; → This is 'γ'

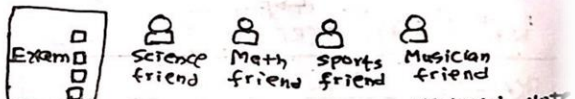
ENSEMBLE :

1. Bagging (Bootstrap aggregating)
2. Boosting

Bagging

BAGGING :

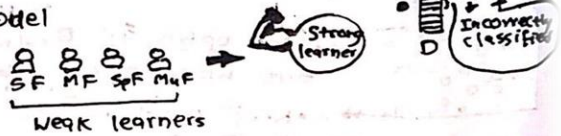
Use Many Model for a Job ; combine / Avg / Add etc.) the output of all



- which Answer have Highest vote that will be Chosen.

BOOSTING :

- Combine the Models → to make a strong model



BIAS :

- If a Model have high bias → means it wont fit well with the Data

VARIANCE :

- If a Model have High V → means it changes drastically

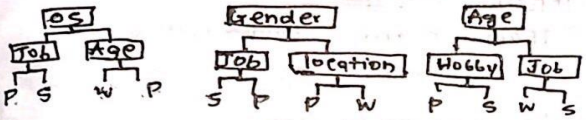
2 Methods to solve:

① Bootstrap the Data: Sampling the Data
② subset the features:

RANDOM FOREST: (BAGGING)

Gender	Age	location	os	Job	Hobby	App

are features ∴ then Randomly Pick Any 3 features for Many times ∴ the most Appeared Answer will be used as output.



Bagging: Row subsets given to Models and voting of the Models is considered. (K-Fold).

ADABOOST: (BOOSTING TECHNIQUE)

① Decision trees are created; with only one depth

f_1	f_2	f_3	o/p

② sample weight $w = \frac{1}{n}$ → 1/n, 1/n, 1/n

n: no. of records / rows

③ create stumps / 1 level trees for 4 feats

Compare the Entropy of all stumps → choose less Entropy stump

② calculate total Error of stump:

→ Total Error (T.E) = $\sum$ weight of Incorrectly classified records.

③ Calculate the Performance of stump:

→ Performance = $\frac{1}{2} \log_e \left[\frac{1-TE}{TE}\right]$

④ Increase the weights of Incorrectly Classified records
- Decrease the weights of correctly Classified.

New sample weight = old weight $\times e^{Performance}$

New sample weight = old weight $\times e^{-Performance}$

f_1	f_2	f_3	o/p	sample weight	Updated weight	Normalized weight

Normalized weight for row(i) = $\frac{\text{updated weight of row(i)}}{\sum \text{updated weights}}$

NOTE: $\sum$ Normalized weights = 1

⑤ consider Normalized weight further

f_1	f_2	f_3	o/p	Normalized weight

- create buckets

Normalized weights	Buckets
0.07	0 → 0.07
0.51	0.07 → 0.58 (0.51+0.07)
0.07	0.58 → 0.65
0.07	0.65 → 0.72

⑥ New Dataset is Created:-
- This Dataset will consider some random Number (say 0.43)
- It takes row belonging to the bucket value range 0.43
- Repeats this with other random values (0.31 ...) till 'n' number of Iterations

④ Repeat from step 1 to 6 for all the Learners

⑤ Majority of votes from all the stumps of all learners will be considered as output.

Hyperparameters: In sklearn

① base_estimator: The model utilized for weak learners

② n_classifier: The Maximum Number of weak learners used.

MODEL EVALUATION METRICS:

- when an Algorithm is working well with the Data And.
- How to tweak them?

→ Training set and Testing set → to test

Golden Rule: Thou shalt Never use your Testing Data for training

CONFUSION MATRIX:-

Medical Model

	Diagnosed as sick	Diagnosed as Healthy
Sick	TP	FN
Healthy	FP	TN

② Accuracy:

$$Acc = \frac{\text{Correctly classified}}{\text{Total records.}}$$

③ Precision and Recall:

In Medical Model	In Spam Dector Model
FP is OK	FP is NotOK
FN is NotOK	FN is ok

Precision:

	Diagnosed as sick	Diagnosed as healthy
Sick	1000	200
Healthy	800	9000

} Recall

$$\text{Precision} = \frac{1000}{1000+800} = \frac{TP}{TP+FP}$$

Recall:

$$\text{Recall} = \frac{1000}{1000+200} = 83.3\%$$

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

④ F1 score:

- Its used to combine Precision & Recall ; by taking its Average
- Although its Not Pure Avg.

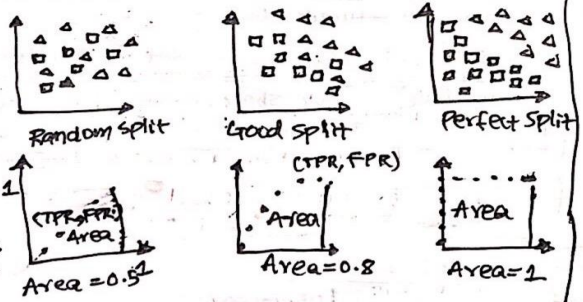
$$\therefore \text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

⑤ F-β score:

$$F_\beta \text{ SCORE} = (1+\beta^2) \cdot \frac{\text{Precision} * \text{Recall}}{\beta^2 * \text{Precision} + \text{Recall}}$$

Precision $F_{0.5}$ score F_1 score F_2 score Recall

⑥ ROC CURVE:



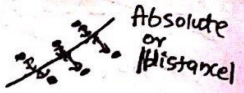
Calculate True Positive Rate and False Positive Rate

$$TPR = \frac{TP}{\text{All Positives}}$$

$$FPR = \frac{FP}{\text{All Negatives}}$$

Regression Metrics:

1. Mean Absolute Error

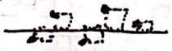


2. Mean Squared Error;

3. R2-score:

$$R2\text{-score} = 1 - \frac{\text{Mean squared Error}}{\text{Simplest model squared Error}}$$

Simplest Model:



If R2score close to '0' } BAD Model

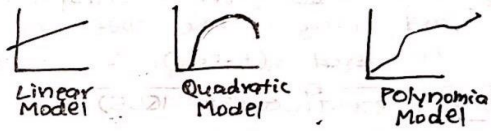
If R2-score close to 1 } GOOD Model.

TRAINING AND TUNING:

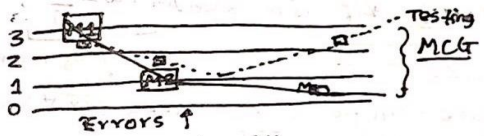
① TRaining error : Underfitting

② Testing Error : Overfitting

Model Complexity Graph:



We calculate training and testing Error and plot that in graph → called → "MCG"



M1 → underfits
M2 → OK
M3 → overfits

K-Fold Cross Validation:-

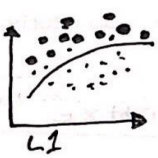
Split 'D' into 'K' number of splits and then train and test with it.

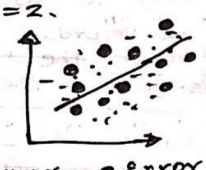
LEARNING CURVES:

Used to check overfitting, Underfitting of Model

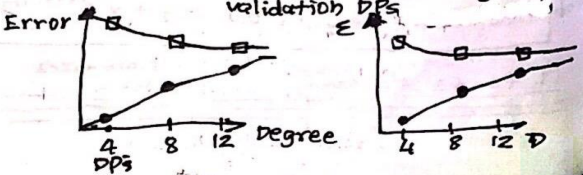
Learner-1 : Linear Model

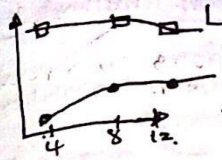
Learner-2 : Polynomial Model with deg-ree=2.





□ Error ● Error with cross validation DPs





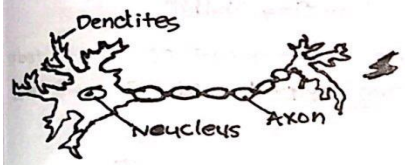
L3 → An Overfitting Model

- It Never gets good while testing with Cross-validation Data Points (DPs)

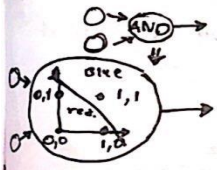
GRID SEARCH CROSS VALIDATION:

- For decision Tree try all Depths (ex: Depth = 1, 2, 3, 4, ...) Pick the one with Highest F1 score.
- For polynomial models try all Degree → Pick Highest F1 scored model
- This is called "Grid search"

DEEP LEARNING

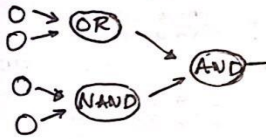


- University student Admission classification from Perceptron Algorithm.
- Perceptron as logical operator



XOR:

- we Need Multilayered Perceptron.



← DP Misclassified.

1. DP → wants the line to come closer!

Perceptron trick :- (In S. Learning section)

Perceptron Algorithm :- (In to)

Non-linear regions:-

- what if the Data cannot be seperated by a single linear line ?

← This cannot be seperated by a Straight line

- we Need something complex

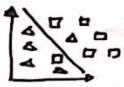
3. ERROR FUNCTION

- To calculate the error

Error Function = Distance

- Look around all the Direction → to (use) Dicrease Error
- But small steps (learning rate η)
- so we use "Gradient Descent"

Log-loss Error Function:

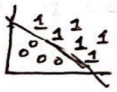


1 Error = □□□+△△△

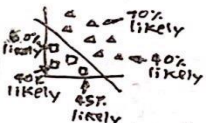
2. Adjust line & measure Error Again.; the Error gets smaller, ∴ till zero.

- But we Need continuous values in Prediction / dependant variable to do Any Gradient Descent ; so If it is Discreate → convert to → continuous

Discrete	Continuous
yes/NO	63.8% likely

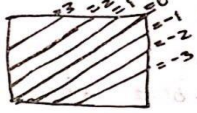


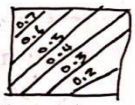
] Discrete



] CONTINUOUS

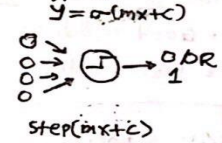
- DP Far → low % likely





- Sigmoid Applied.

$\hat{y} = \sigma(mx+c)$



SOFTMAX:

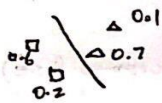
- What if we want to classify multiclass Data points like cat, dog, horse etc... ?
- Exp: This will convert -ve to +ve Number
- Bcz Probability should not have Any value outside the 0 → 1

ONE hot ENCODING:

- Categorical → Numerical

Animal	Cat?	DoG?	Horse?	...
Cat	1	0	0	
DOG	0	1	0	
Horse	0	0	1	
⋮				

Maximum Likelihood:



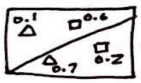
Maximul likelihood.

$0.6 * 0.2 * 0.1 * 0.7 = 0.0084$

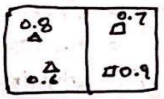
$0.7 * 0.9 * 0.8 * 0.6 = 0.3024$

- The likelihood that the Model is correct
- Good Model will give good Probability

But How to Maximize it ??



$0.6 * 0.2 * 0.1 * 0.7 = 0.0084$



$0.7 * 0.9 * 0.8 * 0.6 = 0.3024$ ← Max

meaning: (Δ 0.8) The model is 0.8 or 80% confident that the dp is Δ or Red Point

CROSS ENTROPY:

But Products will take long if lot of DPs Present ∴ we convert it to sums by using log() function.

$\log(ab) = \log(a) + \log(b)$

we use log of base 10 or ln()

- The ln() -ve of Prob of All dp's is called

-ve Bcz ln() will give o/p in Perfect -ve Numbers.

Ex: $0.7 * 0.9 * 0.8 * 0.6 = 0.3024$

$\ln(0.7) + \ln(0.9) + \ln(0.8) + \ln(0.6)$

$-0.36 \quad -0.1 \quad -0.22 \quad -0.51$

∴ -ve is used

$-\ln(0.7) \quad -\ln(0.9) \quad -\ln(0.8) \quad -\ln(0.6)$

$+0.36 \quad +0.1 \quad +0.22 \quad +0.51$

This is called "Cross entropy"

- Good Model: less entropy
- Bad Model: large entropy

$$\text{Cross Entropy} = -\sum_{i=1}^{m} y_i \ln(p_i) + (1-y_i)\ln(1-p_i)$$

Multi-Class-Cross-Entropy :-

Animal	DOOR1		DOOR2		DOOR3	
CAT	P_{11}	0.7	P_{12}	0.3	P_{13}	0.1
DOG	P_{21}	0.2	P_{22}	0.4	P_{23}	0.5
HORSE	P_{31}	0.1	P_{32}	0.3	P_{33}	0.4

{ Prob. of Animal behind the door

If the Actual thing is: DOOR1 → CAT
DOOR2 → Horse
DOOR3 → Horse

then the Prob table Made by Model

$P = 0.7 * 0.3 * 0.4 = 0.084$

Cross.En $= -\ln(0.7) - \ln(0.3) - \ln(0.4)$

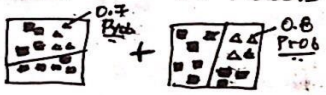
$$\therefore CE = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \ln(P_{ij})$$ for Multiclass

Non-linear Models for Non-linearly seperable Data: [ARCHITECTURE]

we are going to seperate it by using a Probability based seperator

→ This line represents the DP to be equally likely to both ■ (red) and (Δ) blue

- But how do we do this!?
- combine 2 linear Models



$0.7 + 0.8 = 1.5$

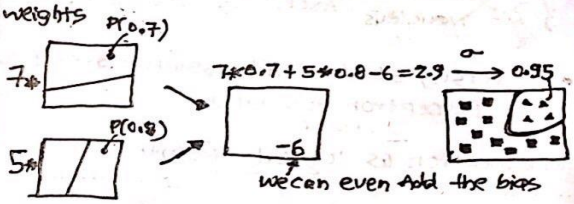
$\therefore \sigma(1.5) = 0.82$

- same dp in 2 different linear model have 2 different values; what do we do when we combine them?

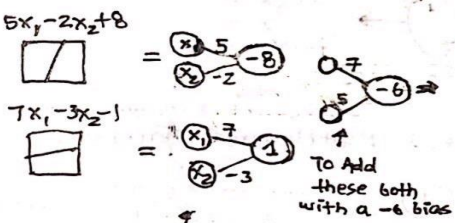
∴ we use Sigmoid (:Bcz $0.7 + 0.8 = 1.5$ → (a)

$= \sigma(1.5) = 0.82$ → This is the Prob. of that dp is resulting Model

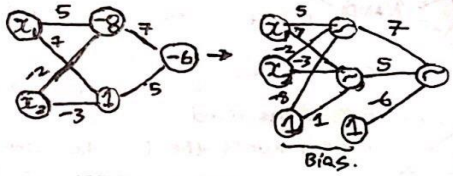
- But what if we want more of first Model and little less of second Model? ∴ we add weights



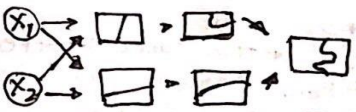
- we can Represent the Above in Neural Networks:



- lets join this:



- Input layer, Hidden layer, Output layer.
- If Input layer have 'n' Nodes then the DPs will be in 'n' Dimension Space.
- One Node at output layer represents one class of classification [Ex cat, Dog etc.]



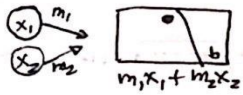
- Combining more Non linear model → Complex Non-linear model

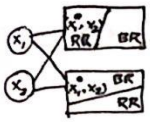
Multiclass Classification:

① NN to find Cat ② NN. to find Dog ③ NN to Identify Horse

combine all this and Add softmax function to obtain the Probability of o/p.

Feed Forward:





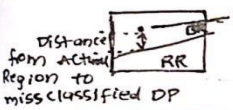
← It gives small Probability value. Since the DP is Blue

• - Blue DP
BR - Blue Region
RR - Red Region

ERROR FUNCTION:

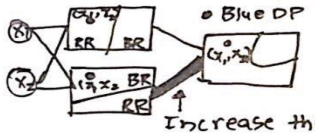
$$E(W) = \frac{-1}{m}\sum_{i=1}^{m} y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)$$

$$\hat{y} = \sigma(Wx+b)$$



∴ Back Propagation. by using Gradient Descent

Backpropagation:



Increase the weight coming from bottom Model Bcz it classified DP correctly (OR Have large Prob value)

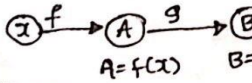
• And decrease the weights of model which gave Misclassification (or less Prob. / σ value).

• In first Model the DP wants or Bcz of DP; the line will come close to DP

• In second Model the DP wants the line to go Away Bcz it correctly classified

• So the output model will Also be affecting from this change

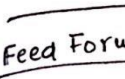
CHAIN RULE:



$A = f(X)$ $B = g(f(x))$

$$\frac{\partial B}{\partial x} = \frac{\partial B}{\partial A}\frac{\partial A}{\partial x}$$

• Gradient Descent, local minima
• to Avoid Local minima add Momentum
• Momentum is like Replacing the Man on top of the hill with A giant Ball; so it can Avoid Local minima
• what if we Dont know the weights



• w_1 & w_2? ∴ we use Error Function $E = \sum(y - \hat{y})$

$\hat{y}$ = N/w o/p
y = Actual o/p.

Sum of the squared Error

$$E = \frac{1}{2}\sum_{\mu}(y^{\mu} - \hat{y}^{\mu})^2$$

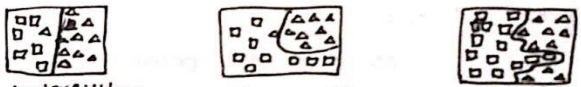
$w_i = w_i + \Delta w_i$

$$\Delta w_i = -\eta\frac{\partial E}{\partial w_i}$$

η → learning rate

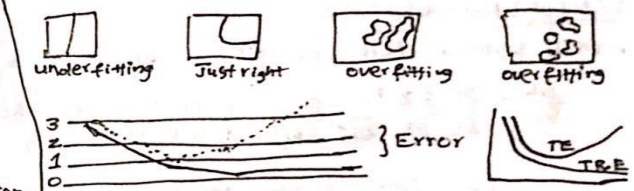
TRAINING OPTIMIZATION:

• Overfitting; underfitting;



• STOPPING CRITERIA:
• Stop epochs before it becomes overfit

MODEL COMPLEXITY GRAPH:



• TR-Testing Error
• TR.E - Training Error

Regularization:

which eqn to use? • x_1+x_2 or • $10x_1+10x_2$

Solution 1: x_1+x_2
$\sigma(1+1) = \sigma(2) = 0.88$ (a)
$\sigma(-1-1) = 0.12$ (□)

Solution 2: $10x_1+10x_2$
Is good one
$\sigma(10+10) = 0.99...79$ (a)
$\sigma(-10-10) = 0.00...21$ (□)

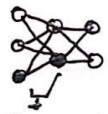
• But the problem is we need to train for large Amount of epochs because it have 10; which the gradient descent should Attain from a smallest Number; with small steps; And it overfits

∴ To Avoid overfitting we use Regularisation

- L1 Error Function $= \frac{-1}{m}\sum_{i=1}^{m}(1-y_i)\ln(1-\hat{y}_i) + y_i\ln(\hat{y}_i) + \boxed{\lambda(|w_1|+\ldots+|w_n|)}$
- L2 Error Function $= \frac{-1}{m}\sum_{i=1}^{m}(1-y_i)\ln(1-\hat{y}_i) + y_i\ln(\hat{y}_i) + \boxed{\lambda(w_1^2+\ldots+w_n^2)}$

DROPOUT:

- Turning some of the Nodes "OFF" randomly
- specified in Probability Value
- EX Dropout Prob. 0.2.



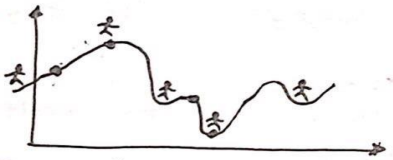
Dropped out for an epoch

LOCAL MINIMA:

- Solution 1:

RANDOM RESTART:

- Start from Many Random points & descend



VANISHING GRADIENT:

- with sigmoid function weight updation is

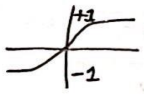
$$\frac{\partial E}{\partial w_{jk}^{(i)}} = \frac{\partial E}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h}\frac{\partial h}{\partial h_j}\frac{\partial h_j}{\partial w_{jk}^{(i)}}$$] Product tiny No. will very tiny Number.

∴ Descending from error mountain takes a lot of time.

How to fix it ? • Best way to fix is changing Activation function

1. tanh :

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



2. Relu : Rectified Linear Unit :

$$relu(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

STOCHASTIC GRADIENT DESCENT:

- take some batch and Feed forward + Back Propagate ; take Another batch ; Repeat this step.

Learning rate Decay:

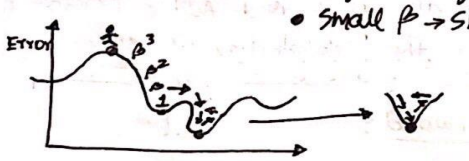
- large η → may give less Accury ; Take tiny η
- Big η → Big steps while descending from the Mountain.

MOMENTUM:

- Take Avg. of previous steps.

β : Blw 0 & 1

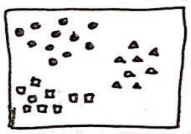
- Big β → Big steps/Ro
- Small β → Small Rolls.



UN-SUPERVISED LEARNING:

- Clustering
- Hierarchical & Density based clustering
- Gaussian Mixture Models and clusture validation
- Principal component Analysis.
- Random Projection and Independent Component Analysis.

CLUSTERING: & DIMENSIONALITY REDUCTION:

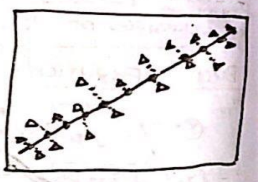


unsupervised learning task concerned with putting similar data into groups "CLUSTERING"

DIMENSIONALITY REDUCTION :

- unsupervised task

Concerned with reducing the Number of variables used



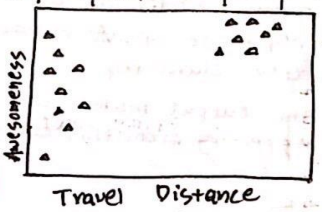
K-Means:

Use cases:

- Netflix grouping similar movies for recommending
- grouping books / things for recommendation
- cluster the group of customers who are similar to each other.
- similar movies / music

K-Means:

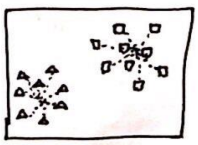
to see family Travelling to work.



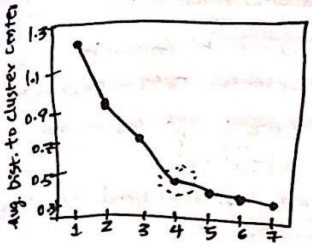
- In K-Means we group the DP's to K- number of groups

- How to choose the value of 'K'?

→ Elbow Method:



K=1, avg.dist= 1.261
K=2, avg.dist= 0.923
K=3, avg.dist= 0.639
K=4, avg.dist= 0.382
⋮ ⋮



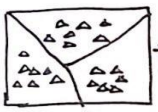
K=7 ; avg dist = 0.298

- we don't Need smallest avg. dist value ;
- Bcz As K ↑ses Avg. distance from center to dp ↓ses.
- so we Need a point where "Elbow" Here it is at (4)
- After which the Avg. dist significantly decreased.

How K-Means works?

1. Randomly place K-centroids amongst your data
2. Assign each point to the closest centroid
3. Move the centroid to the center of the points assigned to it.

Repeat 2,3 untill convergence

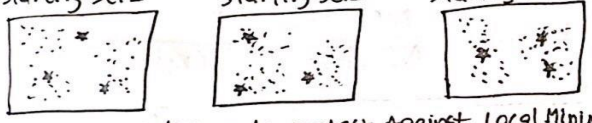


→ For K=3

Is that the optimal solution? :

- Start Kmeans with different starting points

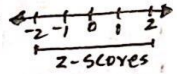
Starting set 1 | Starting set 2 | Starting set 3



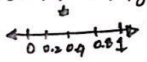
- use repeated runs to protect Against Local Minima
- Choose best grouping sets ; (less Avg. Dist to centroids).

Feature Scaling :

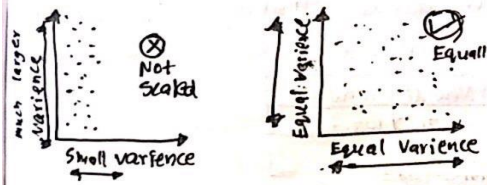
- Standardizing / z-score scaling →



- Normalizing / Min Max

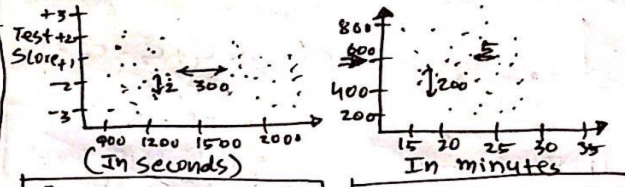


- we use Normalizing → when Scaling the Color of an Image.
- The features with much larger variance will Dominate in the clustering ; we do not want this to happen ; so we scale.



- Standardizing : moves variables to mean = 0 & Std = 1
- Normalizing : moves variable b/w 0 and 1

Feature Scaling Example :



5k completion time(s) | 5k completion time (m)

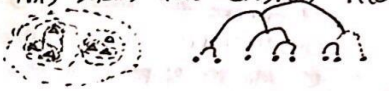
But how to cluster for some Datasets that look like this

But what if this doesn't yield anything to us? or we don't want this ?!!

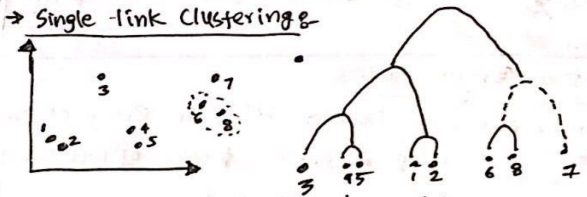
So we use

1. Hierarchical clustering :

- This shows how clusters related to each other



→ Single link Clustering :



- every DP is a cluster at initial
- cluster 2 DPs of closest distance ; continue this with all DPs.
- In (6,8) cluster ; what about 7? ; calculate Distance b/w closest Point in the cluster (Here 6) to 7 make a New cluster of (6,8) and 7 as shown Above Dendogram
- We can decide how many clusters we want ; ∴ so if cluster = 2 then there will be No cluster b/w (687) to (345 12)

→ Complete link clustering :

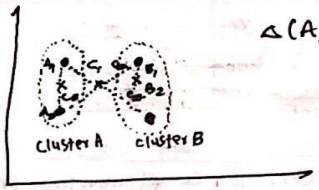
- Everything is same as single linked clustering But In 6,8 to 7 ; calculate longest dist. between (6,8) ; make a cluster to longest dist DP to 7 ; Here 8 and 7 (68,7)

→ Average link clustering :

→ Every thing is same as single link but in case of (6,8) and 7 ; calculate every possible distance from every DP to every other DPs in other cluster

] Avg. of Distance. is the Distance between 2 Clusters.

4. Ward's Method:



Distance b/w Clust. A and B:

$$\Delta(A,B) = C_1^2 + C_2^2 + C_3^2 + C_4^2 - A_1^2 - A_2^2 - B_1 - B_2$$

- whichever is least we merge those 2 clusters

- This is the Default Hierarchical Clustering in Sklearn.

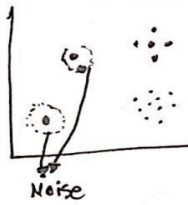
Cons of All Hierarchical Clustering:

- computationally intensive $O(N^2)$
- sensitive to outliers.

2. DBSCAN:

takes I/P: ϵ : search distance Around point

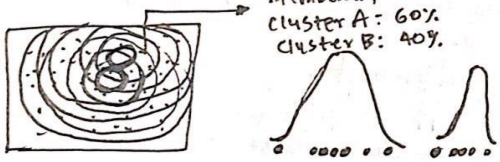
MinPts: Minimum No. of Points required to form a density cluster



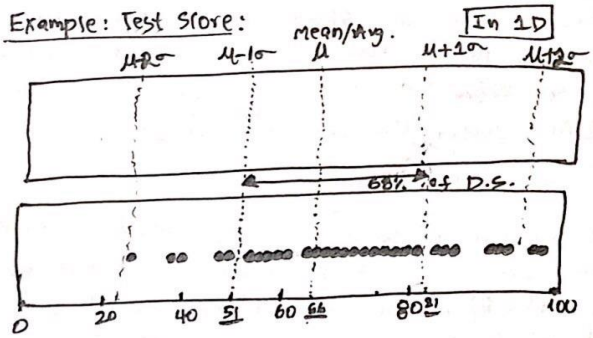
- It checks the Neighbor DP counts for all DPs and if Min No. of pts (i.e Here 5) present; makes it as a cluster.
- Border points; core point

Gaussian Mixture Models:

- Every points in Dataset belongs to every cluster in DS. ; But have different levels of member-ship.

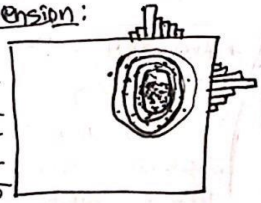


Example: Test Score:

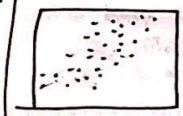


Gaussian Distribution I 2 Dimension:

Student ID	Chemistry score	Geology score
1	66	83
2	61	73
...	...	...
99		



Expected Maximization (EM) Algorithm:



D.S to Cluster into 2 clusters

Step 1 : Initialize 'K' Gaussian Distribution

Step 2: Soft-cluster Data - "Expectations"

Step 3: Re-estimate the Gaussians - "Maximization"

Step 4: Evaluate log-likelihood to check for convergence

Repeat from step#2 until converged

Explanation:

1. Initial each

Cluster	μ	σ^2
A	68, 76.3	100
B	46, 51.2	57

- either by μ & σ of the DS or
- By using k-means (use the centroids as mean)

2. Soft Cluster DPs:

Point#	feature1	feature2	ClusterA	ClusterB
1	62	71	0.99976	0.00024
2	58	81		
3	52	74		
...	...	...		
N	52	78		

- i.e. the Membership Percentage of DP to each of Clusters. (Expectation)

= Expectation of (Membership) of point 1 belonging to cluster A' is denoted as follows:

$$E[z_{1A}] = \frac{N(x_i|\mu_A, \sigma_A^2)}{N(x_i|\mu_A, \sigma_A^2) + N(x_i|\mu_B, \sigma_B^2)}$$

$$N(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \cdot e^{\frac{-1}{2\sigma^2}(x-\mu)^2}$$

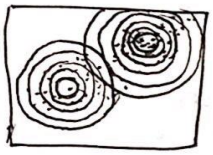
$$E[z_{1A}] = \frac{0.001288}{0.001288 + 0.0000038}$$

- x_i is row in the DS.

$$= 0.99976$$

$$E[z_{1B}] = 1 - E[z_{1A}]$$

- This means that we are 99.97% sure that the point belongs to cluster A.



Step#3: Make New (σ) and New (μ) by using Above Gaussian clusters

New Gaussian parameters:

Cluster	New μ	New σ^2
A	(64..., 76...)	103....
B	(46..., 51...)	67.10...

$$\text{New } \mu_A = \frac{\sum_{i=1}^{N} E[z_{ij}] x_i}{\sum_{i=1}^{N} E[z_{ij}]} = 0.99937 \times (62\ 71) +$$

$$= \frac{0.99937 \times (62\ 71) + 0.9998 \times (58\ 81) + \ldots}{0.99937 + 0.9998 + 0.55818 + \ldots}$$

point #	feature 1	feature 2	cluster A	cluster B
1	62	71	0.99937	0.00063
2	58	81	0.9998	0.0002
3	52	74	0.55618	0.44382
...	...	...	...	...
N	52	78	0.99133	0.00867

→ New $\mu_A = (64.48724579, 76.30745900)$

$$\text{New } \sigma_A^2 = \frac{\sum_{i=1}^{N} E[z_{iA}](x_i - \mu_A^{new})(x_i - \mu_A^{new})^T}{\sum_{i=1}^{n} E[z_{iA}]}$$

$$= 103.92494594$$

- Now we have new Gaussians; the cluster will move up or down; Do this until it converges

Step#4: Evaluate log-likelihood

$$\ln p(X|\mu, \sigma^2) = \sum_{i=1}^{n} \ln\left(\sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \sigma_k^2)\right)$$

- π_k → mixing co-efficient.

- The better gaussian → good or Maximum log likelihood it have; we Repeat Entire process until we get better likelihood

---

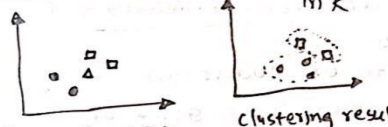
CLUSTER VALIDATION:

1. External validation Indices:
2. Internal validation Indices:

1. External validation Indices:

$$\text{Rand Index} = \frac{a+b}{\binom{n}{2}}$$

a: Number of pairs in the same cluster C and in the same cluster in K



Actual label (C) | clustering result (K)

b: number of pairs in a different cluster C and in a different cluster in K.

n: number of samples or points

$$\therefore \text{Rand Index (RI)} = \frac{2+4}{\binom{5}{2}} = \frac{6}{10} = 0.6$$

$$ARI = \frac{RI - \text{Expected Index}}{\max(RI) - \text{Expected Index}}$$

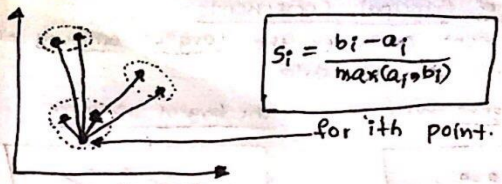
High ARI → Good cluster (Near to 1)
low ARI → Bad cluster (Near to 0)

INTERNAL INDICES:

- Working with Dses that ... labelled.

1. Silhouette coefficient:

- Between −1 and +1



$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

for ith point.

a: avg. distance to other samples in the same cluster

b: Avg. distance to samples in the closest neighboring cluster

$S = \text{average}(s_1, s_2, \ldots s_n)$

Principal Component Analysis (PCA):

- PCA is a technique aimed at dimensionality reduction
- this technique is about taking the Full Dataset and reducing it to only the parts that hold the most Information.

Have you had Break fast?

→ so, I was running late and.. I stopping by the coffee shop, and I had a coffee, and then... I grabbed a pastry → "I had a coffee and a Pastry"

x_1	x_2	x_3	.....	x_n

→

x_1'	x_2'	x_3'

- Size of House
  - lot size x
  - number of rooms x
  - floor plan size
- Neighbourhood Quality
  - local crime rate
  - no. of school in 5mil. x
  - Property tax rate x

→ FloorPlanSize, localCrime rate

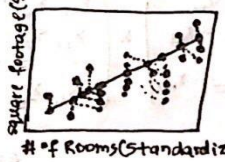
- But makes us loose information; ∴ Rather than dropping features to reduce the dimension-ality of dataset, if we were able to combine the features in some way to maximize the amount of information we retain; → This is the Goal of "PCA"

Now,

$b_1(\text{lot size}) + b_2(\#\text{rooms}) + \ldots$

- size feature
- Neighborhood Quality feature

$b_3(\text{crime rate}) + b_4(\text{schools}) + \ldots$

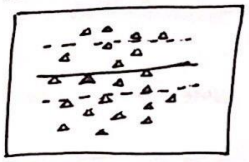


Dimensionality Reduction.

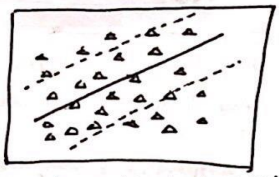
Properties of Principal Components:

① Each component captures the largest amount of variance left in the data.

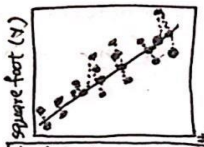
Not the largest variance



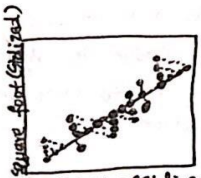
The largest variance



- you loose least Amount of Information
- The Information lost is the sum of dps to the components.



total distance = 5.41



total distance = 5.28

- This will loose less info

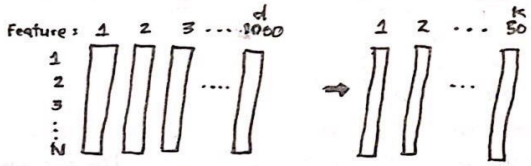
② Components are orthogonal to one another:

- PC1 & PC2 are orthogonal if Angle b/w them is 90°

Principal components are Also known as "eigens vector"

RANDOM PROJECTION:

- Is a powerful Dimensionality Reduction Method



- Each columns Here captures multiple columns of previous

d is dimension → into 'K' (either user given or computed) dimension

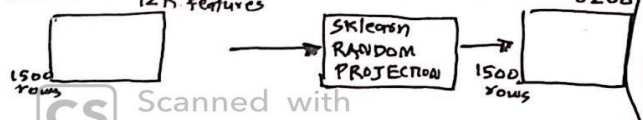
$$X^{RP}_{K\times N} = R_{K\times d} \; X_{d\times N}$$

[original feature Matrix]

$X^{RP}_{K\times N}$: New Matrix; $R_{K\times d}$: Matrix of Random Numbers

$$\begin{bmatrix} X^{RP}_{11} & X^{RP}_{12} & \cdots & X^{RP}_{1n} \\ X^{RP}_{21} & X^{RP}_{22} & \cdots & X^{RP}_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ X^{RP}_{k1} & X^{RP}_{k2} & \cdots & X^{RP}_{kn} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \cdots r_{1d} \\ r_{21} & r_{22} \cdots r_{2d} \\ \cdots & \cdots \\ r_{k1} & r_{k2} \cdots r_{kd} \end{bmatrix} \times \begin{bmatrix} X_{11} & X_{12} \cdots X_{1n} \\ X_{21} & X_{22} \cdots X_{2n} \\ \cdots & \cdots \\ X_{d1} & X_{d2} \cdots X_{dn} \end{bmatrix}$$

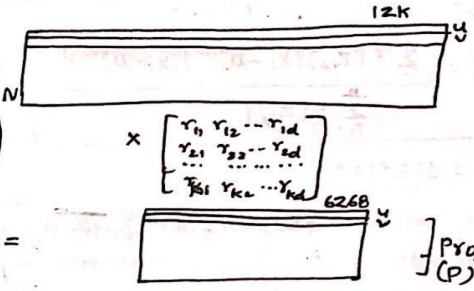
Example:



What is the theory behind RANDOM PROJECTION?

- **Johnson-Lindenstrauss lemma:**

A Dataset of N points in High Dimensional Euclidean space can be mapped down to a space in much lower dimension in a way that preserves the distance between the points to a large degree.



The distance b/w 2 point u,v. in new D.S is:

$$\|P(u) - P(v)\|^2 > (1-eps)\|u-v\|^2 < (1+eps)\|u-v\|^2$$

eps: level of error that we allow in Random projection;

∴ Distance b/w DP u,v in Projection will be larger than Distance b/w u,v in Original Dataset

In sklearn Default value of epselon (eps) is 0.1

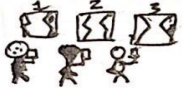
Use cases:

- Random Projection before Applying K-means clustering to 400 face images of 64×64 dimensions.
- Application of RP. in image & text data.

INDEPENDENT COMPONENT ANALYSIS: (ICA):

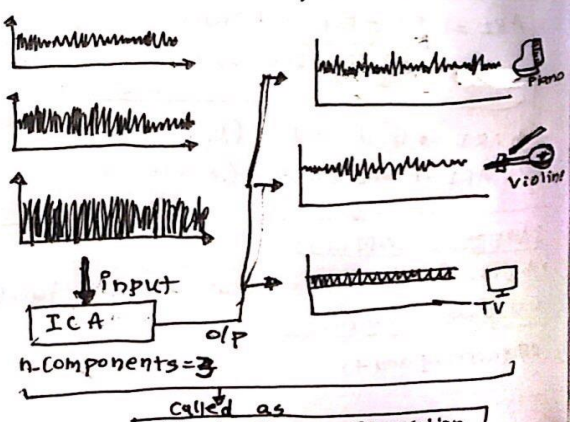
Use case scenario:

- 3 persons go to an Auditorium with 3 shows from 3 curtains going on; one from tv & from Piano & from violine



- All of them record their favorite show sounds by using their phone recorder;

Now is it possible to seperate individual of sound from All of these recordings !?



called as Blind source seperation

FastICA Algorithm:

$w^+ = E\{x g(w^T x)\} - E\{g'(w^T x)\} w$

$g_1(u) = \tanh(a_1 u)$.

Algorithm:

#1: Center, whiten X.

#2: Choose initial Random weight Matrix $w_1, w_2 .. w_n$

#3: Estimate W_p containing vectors.

#4: Decorrelate W

#5: Repeat from step#3 until converged.

- NOTE: There should be n-number of recordings in order to get n-number of components out of it.

ICA Applications:

- Medical fields:
- In Brain scanner ~~is~~ EEG or MEG (like the one In CHAPPIE movie), where each of these sensors sends the electric signals from the brain.

Electroencephalographic data → EEG

↓

Apply ICA to seperate the mixtures.

- In MRIscan Images.
- In finacial Data (stock signals). ⇒ what facts that causes stock to go up & down.

TERM 1 OVER