Lecture - 06:

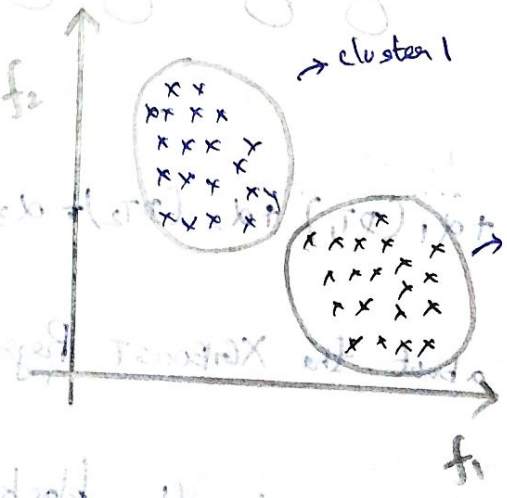
Unsupervised ML:

- → k means clustering.
- → Hierarchical Clustering.
- → Silhoutte score.
- → DB Scan clustering.

Unsupervised ML ? It means actually there is no output. We need to form a clusters.

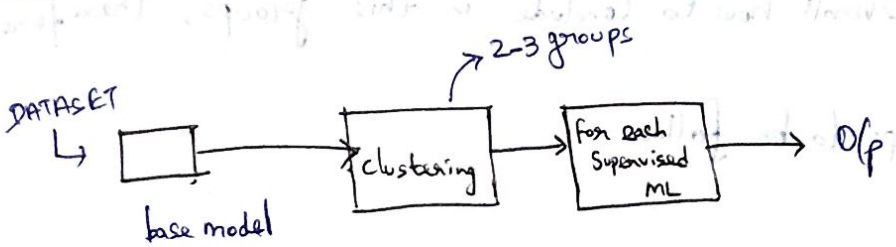
O/P	f_1	f_2
We form Clusters to make O/p Category	-	-
	-	-
	-	+
	-	-
	-	-

Clusters ⇓ { similar kind of data }



We already know Custom Ensemble technique,

In this technique we create,



First clustering than can able to apply Supervised ML algorithm.

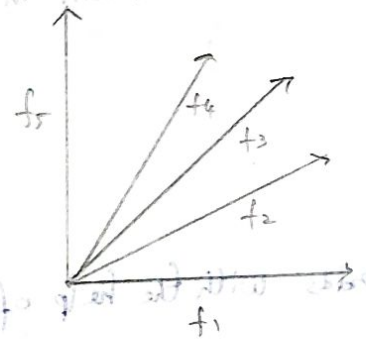
Most of the time Clustering is applied in ensemble technique.

The "k means" → Centroids.

Suppose k=2, we get the two cluster groups.

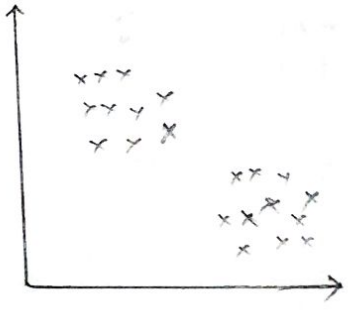
Suppose k=3, we get the three cluster groups like that.

Suppose we have <u>High Dimension data</u> we cannot able to see how the clusters will be formed.



Unable to see

Consider this,



Overall how to Conclude in this groups, Therefore Some Steps to be follow.

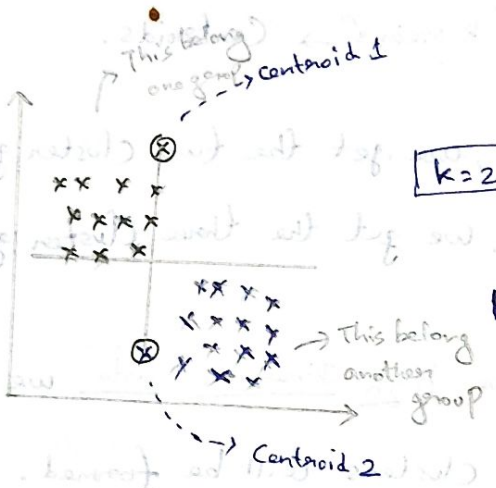
Steps:

1. We try k values ⇒ Suitable k=2 just Consider.

2. Initialize the k number of Centroids.

3. Update the Centroids.

Step 1 and 2

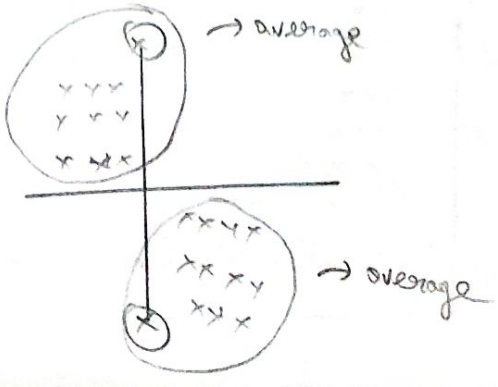


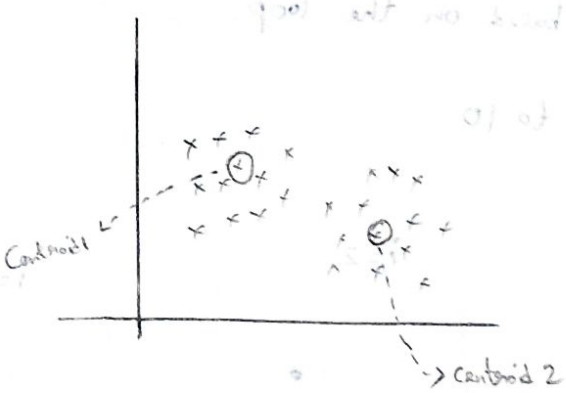
Each Centroid to data points are Calculated with the help of Euclidian distance.

The Centroids are initialize in random in the graph Space.

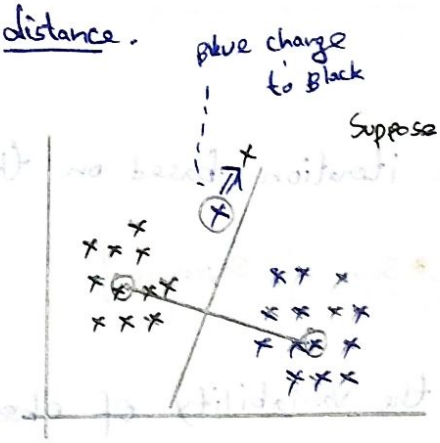
Step 3:

Update the Centroids with the help of taking the averages.





Again the perpendicular line make, It means calculate <u>eucleidian distance</u>.



Suppose some blue point in black cluster group. That is also considered as black.

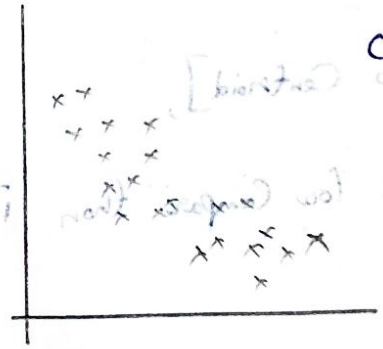
Because, After updating all, still some blue points in blackgroup. It is also considered as black point.

How to decide the K-value?

With the help of Elbow method to find the optimized K-Value.

<u>Elbow method:</u>

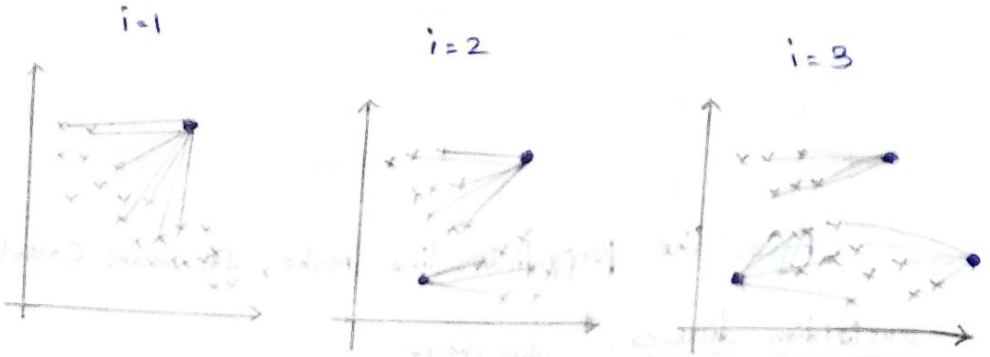
The elbow method is used to find the optimized K-value.



Consider this, we cannot directly split clusters. In this time use elbow method.

The working is based on the loop.

Initialise for i=1 to 10



Like this try some iteration based on the iteration and Wcss [within Clusters Sum of Squares].

Wcss measures the variability of observations within each cluster using Euclidian distance.

Bcss [between clusters Sum of Squares] which measures the squared average distance between all Centroids.

To calculate Bcss, you find the euclidean distance from a given Cluster Centroid to all other cluster centroids.

Obiviously i=1 [It means one centroid],

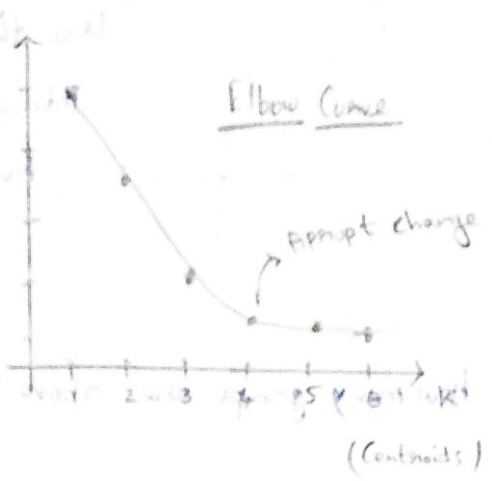
The wcss is quite large.

i=2 [It means two Centroid],

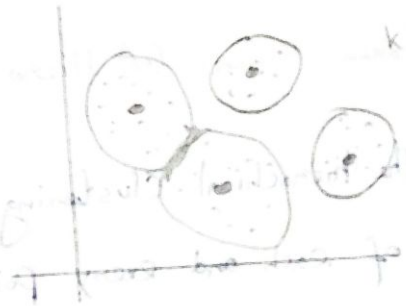
The wcss is low Compare than i=1.

Because, Based on Centroids the whole data points splits to calculate WCSS. Obivously the more Centroids the distance will be low.

for i = 1 to 10



The apropt change happened at Centroid 4. It means actually 4 groups are available in the dataset.

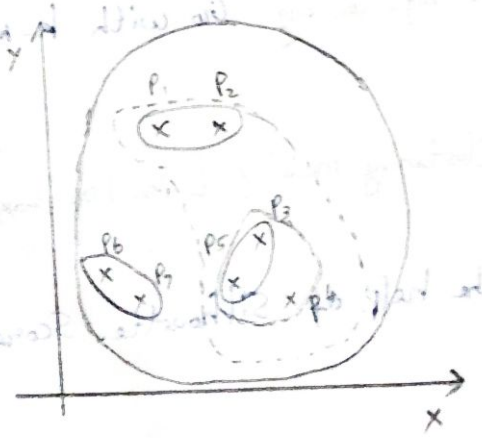


• → Every group have own Centroid.

This is all about the k-means clustering.

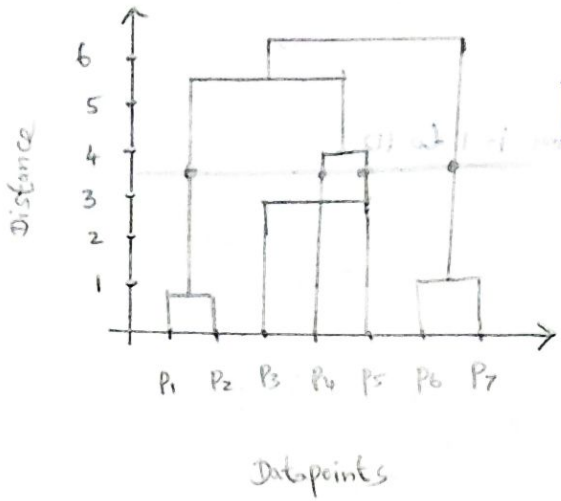
Hierarchical Clustering:

Consider,



This clustering make Clusters based on minimum distance to maximum distance point.

This is called as Dendogram.



k=4

Draw the line with point intercept without no horizontal line.

How do you find how many groups are above?

. You need find the longest vertical line that has no horizontal line passed through it.

Maximum time is taken Kmeans (or) Hierarchical Clustering?

Obiviously, The hierachical clustering because it has the process of each and every point to make dendogram is high time complexity.

If the dataset is small, Go with hierarchical Clustering.

If the dataset is high (or) big, Go with K-means clustering.

How to validate Clustering model? [Like ~~Performance~~ Confusion metrics, R^2].

Ans: With the help of Silhouette Score.

Validating Clustering Models:

The Silhouette Score is used to validate the cluster models.

Some steps to be follow,

→ To find out the a(i) and b(i).

Consider a(i) is nothing but one group,

i → Centroid
j → datapoints

The Centroid (i) calculates the differences and squares of all data points with summation.

$$\sum d(i,j)$$

Then calculate the overall averages of each with the formula,

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

Samlike Calculate b(i) with the help of considering other group.

After that,

Each and every points are calculated with each every point of another group.

Based on above,

Good Model

$b(i) >> a(i)$ It is considered as good model

{ It means because,

The distance between $b(i)$ is greater than the distance of $a[i]$.

The more distance, the clusters are correct group without any point.

Bad model

$a(i) >> b(i)$ → It means the ~~greatest distance~~ of $a(i)$ is greater than $b(i)$.

The value of the Silhoute Score its ranges between (-1 to 1).

The value towards +1 => It considered as good model.

The value towards -1 => If considered as Bad model.

The formula will be,

Silhoutte Score,

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

⇓

By this formula gives the value (-1 to +1).

DBScan Clustering:

Density - Based Spatial clustering of Applications with Noise (DBSCAN)

The four comparats of DBScan Clustering,

→ Epsilon
→ Min Points
→ Core points
→ Border points
→ Noise points

Epsilon;

Epsilon refers to the radius of the circle.

Min points:

Core points:

Atleast present 4 points.

[min points = 4]

In this case,

Core → points.

Border Points:

Atleast one point present, we call it is border points.

Noise Point:

→ without any points, we can say Noise points.

[It means the points are present outside]

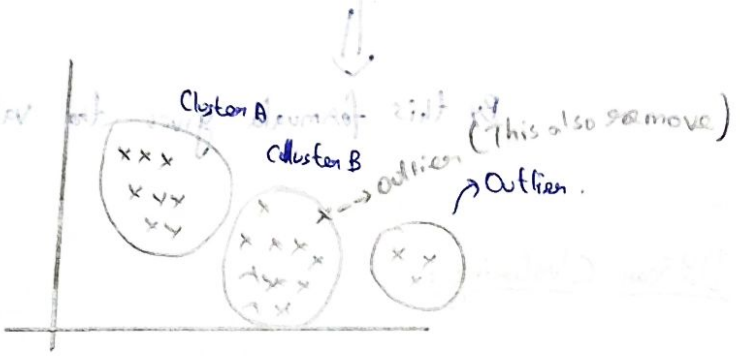
↳ Outliers.

The Min pts is decided by hyperparameter.

The ε-epsilon value also decided in some specific way.

Why use?

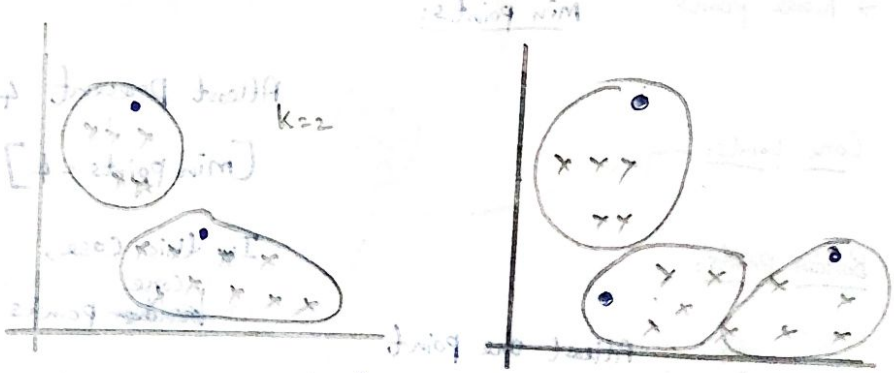
To detect the outliers efficient way,



Once find the Outlier, we don't need to take the group for analyze, Just neglecting whole Outliered group.

One issue with K-means:

If initialize the centroids in k-means it should be very very far. Otherwise, Suppose initialize Centroids in short distance it may be take two groups of two centroid instead of three Centroid of three groups.



Centroid use in short distance.

Centroid use in long distance.

⇓

Effiient One

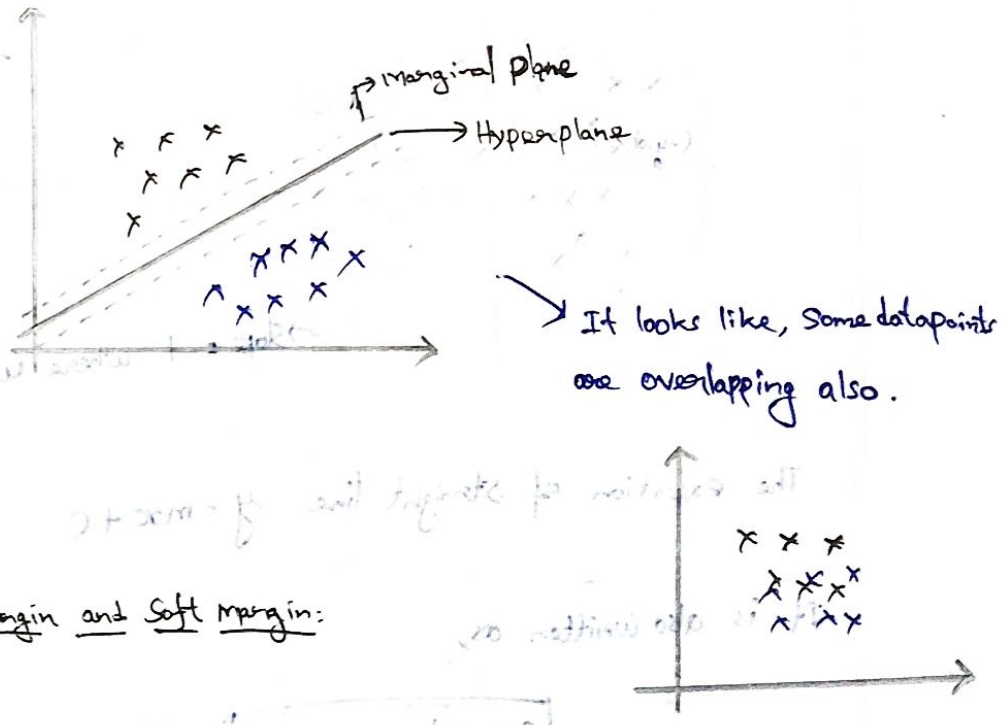
To set the Centroid very very far in random space, Something algorithm is used Called as K means ++.

Support Vector Machine: (SVM)

The main aim of the Support vector machine Create best fit line with marginal plane. In this way, we Create or decide the output.

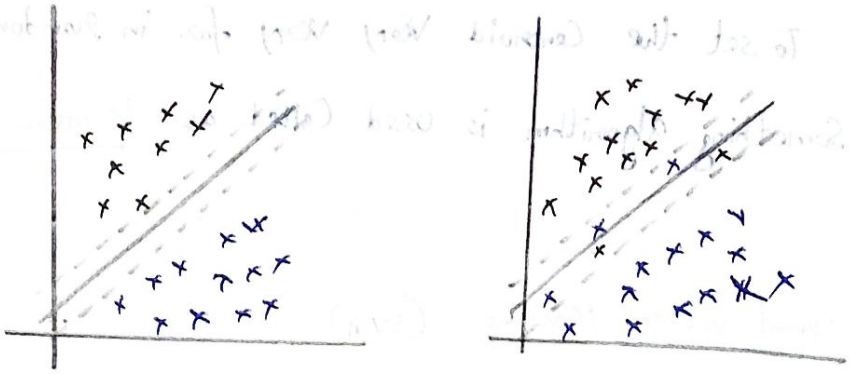
It is supervised machine learning algorithm for Classification.

Looks like,



Hard margin and Soft margin:

The data is linearly Separable, and we don't want to have any misclassification, we Use SVM with a hard margin.



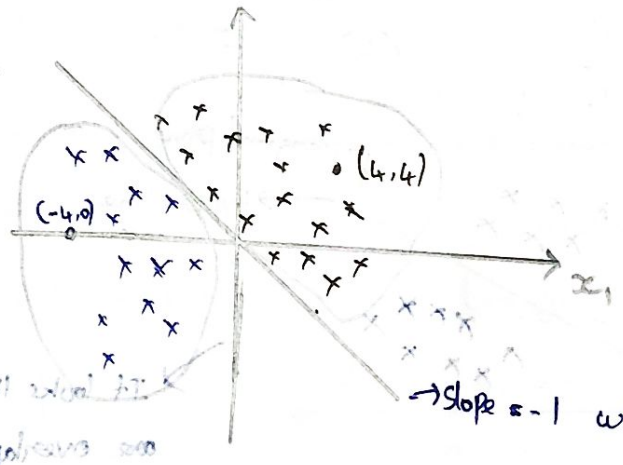
Hard margin

Soft margin

However, when a linear boundary is not feasible, or we want some misclassification is called <u>Soft margin.</u>

Generally, In normal dataset only contains <u>soft margin</u>.

Consider,



→ Slope = -1 where $w = (-1, 0)$

↳ Intercept

The equation of straight line $y = mx + c$

It is also written as,

$$\boxed{ax + by + c = 0}$$

$$\Downarrow$$

$$y = -\frac{c}{b} - \frac{ax}{b} \quad \text{where } m = -\frac{a}{b},\ c = \frac{-c}{b}$$

$y = mx + c$ (slope, intercept)

Same like written as,

$$y = w_1x_1 + w_2x_2 + w_3x_3 + \dots w_nx_n + b$$

(why write this ? because based on feature (x))

$$\boxed{y = w^Tx + b}$$

Consider (-4, 0),

Apply in y,

$$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -4 & 0 \end{bmatrix} \Rightarrow -4 + 0 = 4 \quad \boxed{y = 4}$$

$$y = \begin{bmatrix} -4 & -4 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \end{bmatrix} = 4 \text{ (Positive Value)}$$

and Consider (4, 4),

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

$$y = \begin{bmatrix} -4 & -4 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -8 \\ 4 \end{bmatrix}$$

$$y = -4 + 0$$

$$\boxed{y = -4}$$

(Negative Value)

So, we Conclude that the points in below the linear line. It gives positive value. So that the each data points below are give positive value.

The value above the points in the linear line. If gives y as negative value. So all the points are Considered as negative.

How to Create best marginal plane?

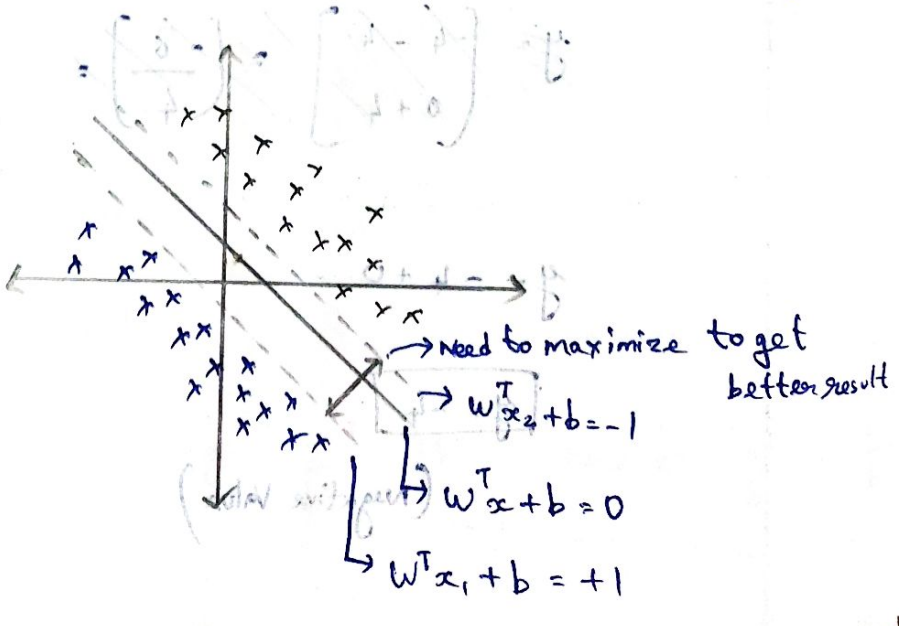
The answer is very simple. According to above Calculation,

$$ax + by + c = 0$$

$$\Downarrow$$

$$w^T x + b = 0$$ [It indicates the $mx+c=y$]

↳ straight line



The line equation is written with the help of previous Calculation.

To find out the distance between two marginal plane, we do,

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

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

$$(-) \quad (-) \quad (+)$$

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

In Generally, The marginal plane are the vectors. We don't need any mg magnitude. So, remove with the help of $|w|$.

$$\frac{w^T (x_1 - x_2)}{|w|} = \frac{2}{|w|} \quad \text{[Just dividing]}$$

So, our aim is to maximize the marginal, It means that

$$\text{Maximize} \left[\frac{2}{||w||} \right].$$

We need to maximize $\left(\frac{2}{|w|} \right)$ with respect to (w, b).

$$\underset{(w,b)}{\text{Maximize}} \quad \frac{2}{||w||}$$

Such that,

$$y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$$ (by write with the help of diagram)

$\Downarrow$

So, this overall written as [for getting positive]

$$\boxed{y_i * (w^T x_i + b) \geq 1}$$ → Constraint

$\Downarrow$

for correct point it means we get the positive value.

With the help of above equation to get the positive value.

So,

$$\boxed{\text{Maximize}_{(w,b)} \frac{2}{\|w\|} \iff \text{Min}_{(w,b)} \frac{\|w\|}{2}}$$

The both are equal, so find minimize the function,

$$\boxed{\text{Min}_{(w,b)} \frac{\|w\|}{2}}$$ with adding two more parameter.

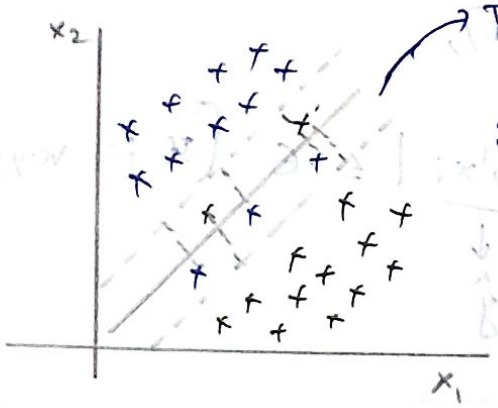
$\xi_i \rightarrow$ eeta

So,

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

C_i → How many errors we can have.

ξ_i → Summation of the distance of the wrong datapoints.

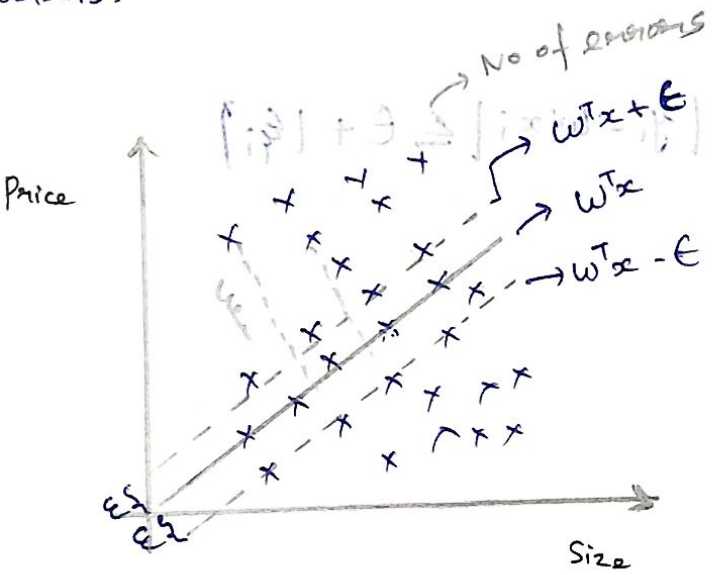


Summation of distance of wrong points,

The distance between wrong and marginal plane of correct group.

Support Vector Regression (SVR):

The SVR is basically tells about the Regression problems.



ϵ - epsilon denotes the distance

So Cost function,

$$\underset{(w,b)}{\text{Minimize}} \quad \frac{||w||}{2}$$

Constraint,

$$|y - w^T x_i| \leq \epsilon$$

$\downarrow$

$\hat{y}$

Constraint (MSE),

$$|y_i - \underbrace{w^T x_i}_{\downarrow \hat{y}}| \leq \epsilon \quad \text{[If is very good]}$$

So total cost function,

→ Hinge loss

$$\underset{(w,b)}{\text{Minimize}} \quad \frac{\|w\|}{2} + \boxed{C \sum_{i=1}^{n} |\xi_i|}$$

Hyperparameter

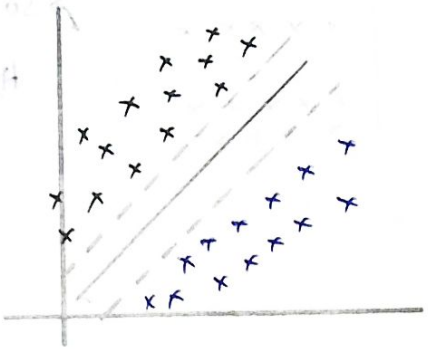
So, Constraint would be

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

Some Concepts in Machine Learning:

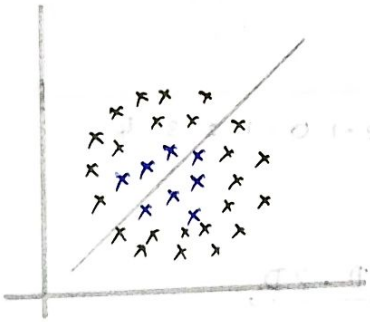
SVM Kernels:

Consider a SVM for datapoints,



The datapoints may be an Soft margin (or) Hard Margin.

Suppose, we have a data look like below,



In this data, How do Split, Because these data are no correalation.

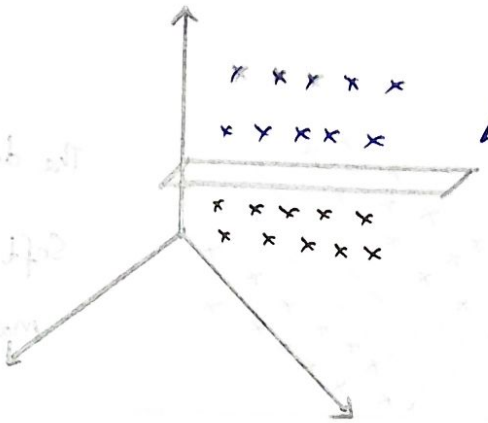
The above linear line split is wrong, So therefore my problem is that Create the linear line for the problem. So, SVM kernel which comes in place.

What Can I do for above Problem?

We need to apply some transformation. It means we need to change to lower dimension to higher dimension

The transformation is done with the help of Mathematical formula.

After applying transformation, The diagram would be



So easily create Hyperplane.

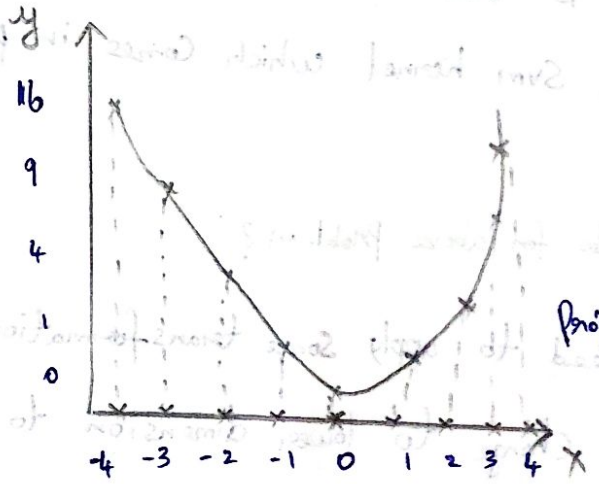
Suppose we have One dimensional,

1D

one dimension (f_1)

We need to change 1D - 2D,

Applying $y = f(x)$, $f(x) = x^2$



$y = (-4)^2 = 16$

$y = (-3)^2 = 9$

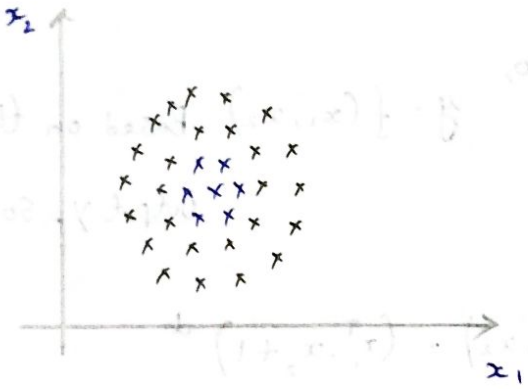
. . . .

$y(4) = 16$

After plotting the value, we need to project the one dimensional data.

Same like, the 1D - 2D, 2D - 3D, 3D - 4D like this the line is fitted properly and make split.

Suppose we have,



We have this type of the data, This is 2D. We need to change 3D.

Some types of kernels are,

→ polynomial kernel.

→ RBF kernel.

→ Sigmaid kernel.

Based on situation of data, we use the kernel types.

In this, we are use the polynomial kernels.

The above dataset, we have the features like x_1, x_2 and o/p as y.

x_1	x_2	y

The formula for polynomial kernel,

$d \rightarrow$ dimensions

$$f(x_1, x_2) = (x_1^T \cdot x_2 + 1)^d$$

So,

$y = f(x_1, x_2)$ Based on the formula the output y. So assign this.

So,

$$f(x_1, x_2) = (x_1^T \cdot x_2 + 1)^d$$

Apply feature,

$$[x_1 \; x_2]^T * [x_1 \; x_2]$$

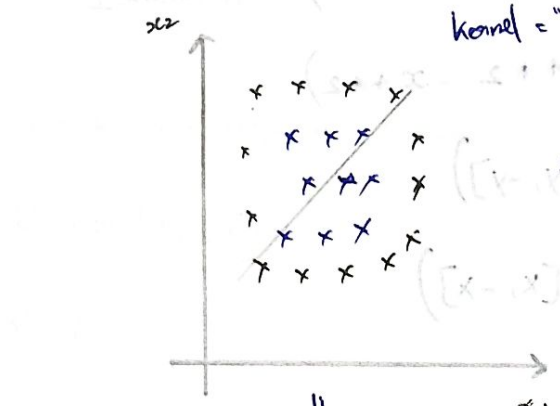
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} * [x_1 \; x_2]$$

$$= \begin{bmatrix} x_1^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 \end{bmatrix}$$

Take the unique elements for creating 3D.

So, After taking this now we have five features,

x_1	x_2	y	x_1^2	x_2^2	$x_1 x_2$
2	4	1	4	16	8
3	3	0	9	9	9
4	5	1			
-	-	-	-	-	-

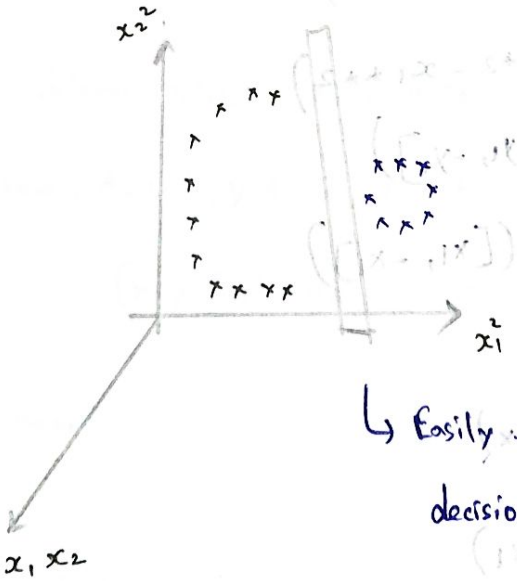


kernel = "Linear"

↳ But it not efficient for below data.

This will change into below

2D to 3D



kernel = "poly"

↳ Easily split and take the decision.

The kernel type "poly", "rbf", "Sigmoid" are taking with the hyperparameter tuning.

This is all about the SVM Kernels.

PRACTICAL IMPLEMENTATION:

Svm kernels and Practical

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5.0, 5.0, 100)          # Feature 1
y = np.sqrt(10**2 - x**2)
y = np.hstack([y, -y])
x = np.hstack([x, -x])

x1 = np.linspace(-5.0, 5.0, 100)         # Feature 2
y1 = np.sqrt(5**2 - x1**2)
y1 = np.hstack([y1, -y1])
x1 = np.hstack([x1, -x1])

plt.scatter(y, x)
plt.scatter(y1, x1)

# Create Dataframes with Pandas
```

```
import pandas as pd

df1 = pd.DataFrame(np.vstack([y, x]).T, columns=['x1', 'x2'])

df1['y'] = 0

df2 = pd.DataFrame(np.vstack([y1, x1]).T, columns=['x1', 'x2'])

df2['y'] = 1

df = df1.append(df2)

df.head()

# Independent and dependent feature

X = df.iloc[:, :2]

y = df.y

# Split the dataset into train and Split.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split
                                   (x, y, test_size = 0.25, random_state = 0)

y_train

# Applying in Algorithm

from sklearn.svm import SVC

Classifier = SVC(kernel = "linear")
```

If linear for above datasets.

Check the accuracy.

```
from sklearn.metrics import accuracy_score
y_pred = classifier.predict(x_test)
accuracy_score(y_test, y_pred)
```

Output: 0.45

Therefore use Polynomial kernel

we need to find components for the polynomial kernel.

x_1, x_2, x_1-square, x_2-square, $x_1 * x_2$

```
df['x1_square'] = df['x1'] ** 2
df['x2_square'] = df['x2'] ** 2
df['x1*x2'] = (df['x1'] * df['x2'])
df.head()
```

Independent and dependent feature

```
x = df[['x1', 'x2', 'x1_square', 'x2_square', 'x1*x2']]
y = df['y']
```

```
y
```

```
X_train, X_test, y_train, y_test = train_test_split
                    (x, y, test_size = 0.25, random_state = 0)

X_train

#(visualize the 3D Graph)

import plotly.express as px

fig = px.scatter_3d (df, x = 'x1', y = 'x2', z = 'x1 * x2',
                     color = 'y')

fig.show()


Classifier = SVC (kernel = "linear")

Classifier.fit (x_train, y_train)

y_pred = Classifier.predict (x_test)

accuracy_score (y_test, y_pred)
```

Output: 1.0

It denotes 100% accuracy with Correct linear Split.

Principal Component Analysis: [PCA]

Dimensionality Reduction

Why we are Use PCA?

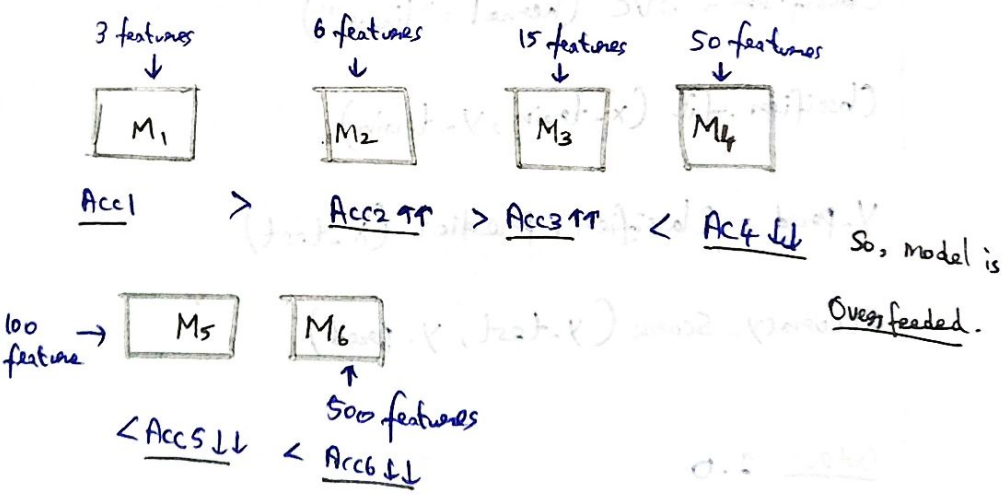
Some reasons are used to perform PCA.

① Curse of Dimensionality:

Suppose we have 500 features in the dataset.

we want to train ML models with these features,

Suppose,



Note that, The increases in features means. Some features are not important at all. So, the accuracy will decreases.

The features would be,

→ House Size

→ No of bedrooms.

→ No of bathrooms. like that.

② Model Performance Degrade:

Suppose Consider a human being, The person thinks about not only the prices of the room appartment and also thinks about the facilities which comes along the room.

feature 1

Loc A	Price
2 Bhk	450k - 500k
3 Bhk	500k - 600k
Near Beach	↑↑↑
near to celebrity house	↑↑↑
Grocery shop	It is not more important but machine also train this.
School	↑↑↑

This says curse of dimensionality

Sometime, The domain expert also Confused to give accurate data. This will shows the many features are feeded into the machines. So, the model is Overfeeded.

How to remove and prevent this Curse of dimensionality?

Two different ways are,

→ Feature Selection.

→ fe Dimensionality Reduction (PCA).

Feature Selection

⇓

Take out and select the important features.

Dimensionality Reduction (PCA)

⇓

Feature Extraction.

⇓

It says we will derive the new feature from the set of features.

The feature extraction would be an,

original

f_1 f_2 f_3 O/p

If must be lesser dimension ⇓ Feature Extraction

D_1 D_2 O/p

This is all about the feature engineering on PCA.

We will see about the feature selection and feature Extraction.

Feature Selection Vs Feature Extraction:

Why Dimensionality Reduction?

→ Prevent → Curse of dimensionality.

→ Improve the performance of model.

→ Visualise the data → understand the data.

3d | 2d visualization.

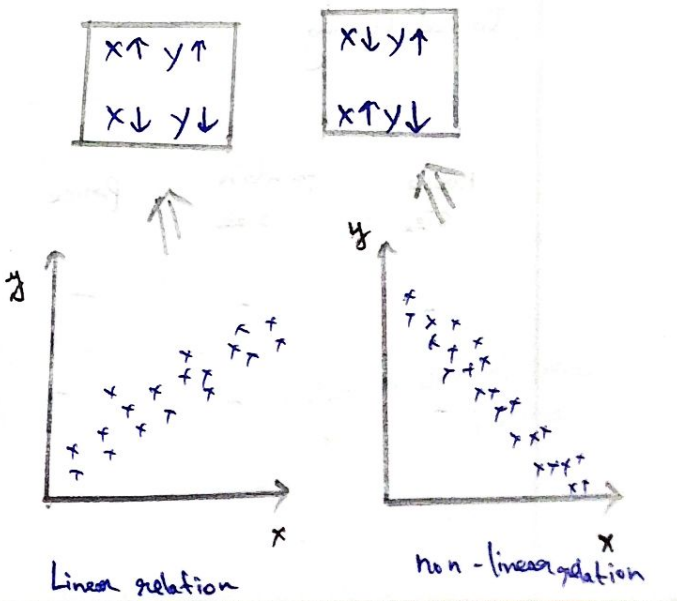
load → 3d (or) 2d then visualize.

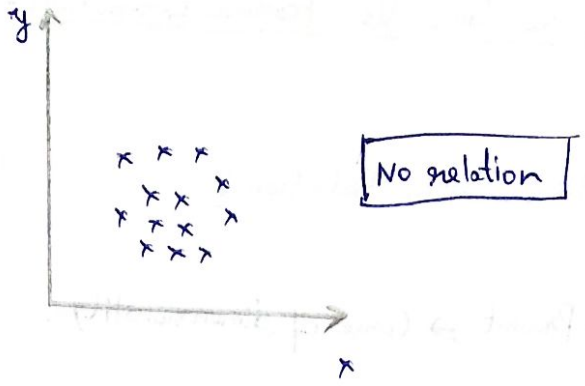
Feature Selection:

Feature Selection is the technique to select the most important feature from set of features.

Consider,

I/p	O/p
x	y
-	-
-	-
-	-
-	-





$$Cov(x,y) = \sum_{i=1}^{n} \frac{(x_i - \bar{x}) * (y_i - \bar{y})}{N-1}$$ = +ve (or) 0 (or) -ve

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

More towards +1 the feature x and y are more relationship.

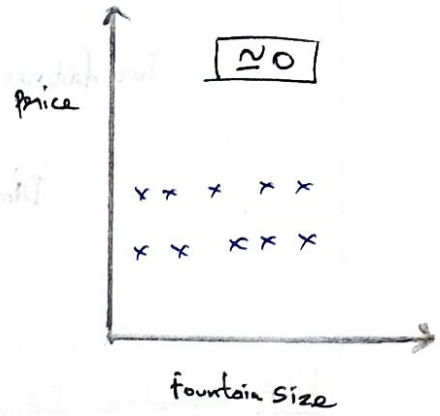
More towards -1 the feature x and y are no relationship.

Is equal to 0. The Two features are no-correlated.

Example,

Dataset housing

House size	fountain size	Price
-	-	-
-	-	-



⇓ This is linear and more Correlated.

⇓ This is not an linear and Correlated.

Therefore, The house size is very important feature. Select that house size and neglect the fountain size for an analysis.

Feature Extraction:

Feature Extraction is the technique to extract the new feature from set of features.

Consider that,

Room Size	No of Rooms	Price
-	-	-
-	-	-
-	-	-

⇓ Transformation applied to extract new feature

House Size	Price
-	-
-	-

Two features → One feature

Dimensionality Reduction

PCA Geometric Intuition:

Consider the housing dataset,

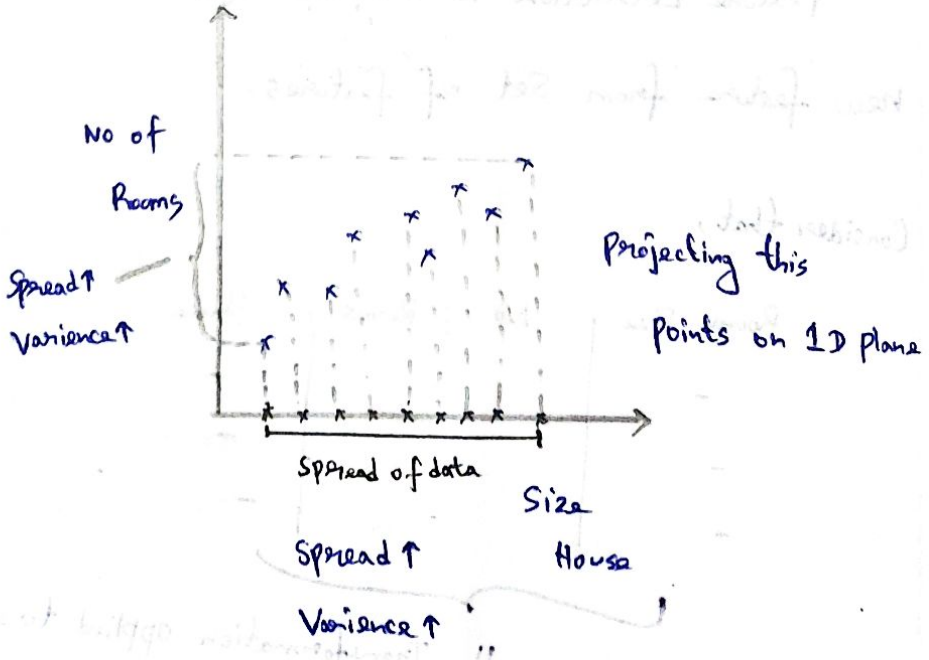
Size House	No of Room	Price (o/p)
-	-	-
-	-	-

PCA,

2 dimensions → 1 dimensions

Obiviously, we know the size house and no of Room are the linear relationship.

So, plot the points,



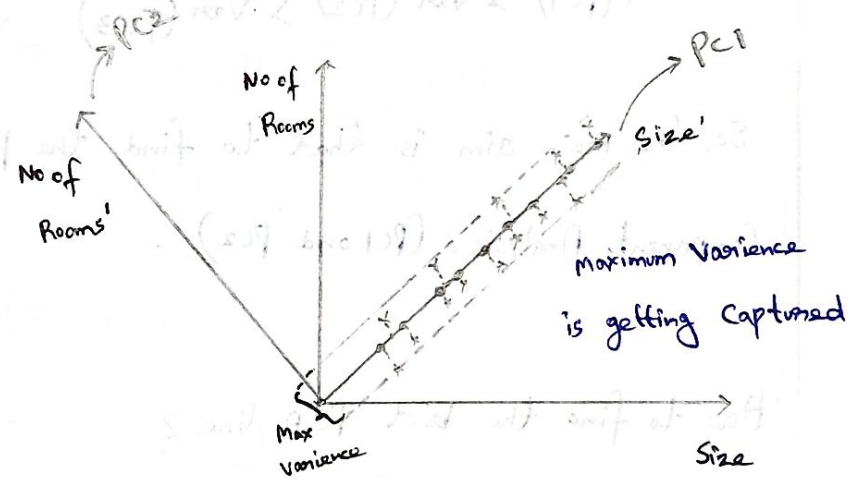
So, the main disadvantages of this approach is that the No of rooms information will be lost.

So, the model unable to predict and perform well.

→ Loss of information (No of Rooms).

Therefore, the different approach was used in PCA.

That different approach is nothing but the some transformation is applied to capture maximum varience of data.



How the axis line created?

It is created with the help of some transformation is nothing but Eigen decomposition on Matrix.

So, in this approach we tried 2D → 1D.

Therefore much information is not lost.

2 Dimensions

PC1, PC2

Var (PC1) > Var (PC2)

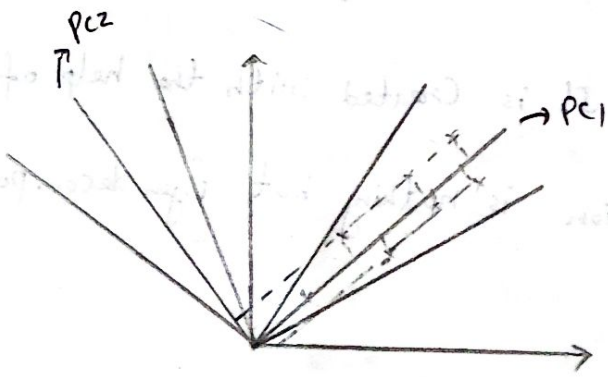
3 dimensions

PC1, PC2, PC3

Var (PC1) > Var (PC2) > Var (PC3)

So, the main aim is that to find the Principal Component Analysis. (PC1 and PC2).

How to find the best PCA line?



2D - 1D

2 Best PCA

The finding of PCA is very simple to check the which Principal Component lines Captures more Variance of data points.

That data PCA lines are selected.

To get the best Principal Component which captures Maximum Variance.

Suppose we have,

$3D \rightarrow 1D$

⇓

we know, $PC_1, PC_2, PC_3 \rightarrow 1D$

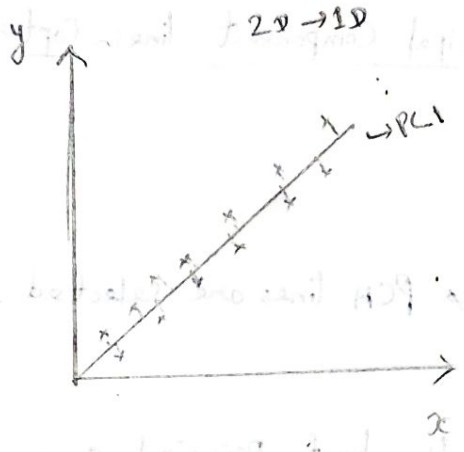
$Var(PC_1) > Var(PC_2) > Var(PC_3)$

Suppose, $3D \rightarrow 2D$,

PC_1, PC_2, PC_3

↳ This two taken and Convert to 2D.

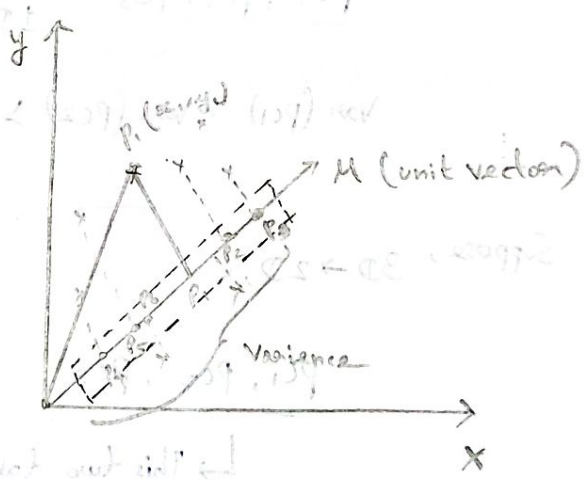
Maths Intuition behind PCA algorithm.



The two important things are,

1. Projection.
2. Cost function related to Variance.

Consider that,



Suppose I want to project $P_1(x_1, y_1)$ on unit vector,

$$\text{Proj}_{P_1} M = \frac{P_1 \cdot M}{\|u\|} \qquad \text{where, } \|u\| = 1$$

Therefore, $\text{Proj}_{P_1} M = P_1 \cdot M$ ⇒ Give Scalar value.

⇓

P_1'

Scalar refers to only magnitude.

So, In our case we want to find the variance. (i.e) variance which comes under the distance. Therefore we say magnitude. (Scalar).

So, Computing every points we need to project. It looks like,

$P_0', P_1', P_2', P_3', P_4', \ldots P_n'$ → Scalar value

⇓

Variance (distance) spread.

The projecting point on the unit vector we say P_1'.

Let's take,

$P_0', P_1', P_2', P_3', P_4', \ldots P_n'$

We use different notation as,

$x_0', x_1', x_2', x_3', x_4' \ldots x_n'$.

Goal:

Find the best unit vector which captures maximum variance.

$\bar{x}$ → Sample mean.

So, Max variance $= \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}$

⇓

So, this is the cost function of PCA.

Then, we cannot obiviously try to find which dirf vector line is capturing maximum variance.

Inorder to find that, we something used called as Eigen Values and Eigen Vectors.

Eigen Vectors and Eigen Values:

To calculate this,

→ Firstly, Covariance matrix between features need to be find.

→ Then the eigen vectors and eigen values will found (or) find out from this Covariance Matrix.

→ Eigen Vector → Eigen Value → It shows the magnitude of eigen vector. Using this to capture the maximum variance.

The second point shows some mathematical equation to find eigen value and eigen vector. Something are the, $\boxed{Av = \lambda v}$ → It is nothing but linear Transformation of matrix.

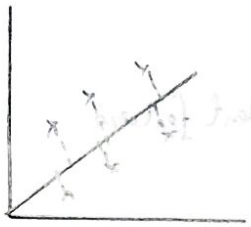
Using these, to find out the maximum variance of the datapoints captured.

Eigen Vectors and Eigen Values: [Linear Transformation]

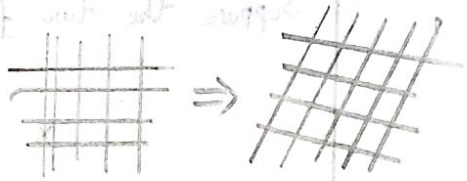
[Eigen decomposition of Covariance Matrix]

⇓

Eigen Vector and Eigen values



The linear transformation would be,



See the difference in some website..

$$[\ ] * [V] = \lambda * V$$

↓

Eigen value

So,

$$A * V = \lambda * V$$

⇓ So this give

Eigen Vector

↓ which gives

Maximum Magnitude

↳ from this we get principal components and get the maximum variance captured.

The graphical visualization of this available in,

https://shad.io/MatVis/

Steps to Calculate Eigen Value and Vectors:

1. Firstly, Covariance of features.

Consider,

x	y	→ z (dependent feature)

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

Suppose the two feature, represented as 2×2,

	x	y
x	Var(x)	Cov(x,y)
y	Cov(y,x)	Var(y)

why Var(x), because Cov(x,x),

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

$$= \sum_{i=1}^{n} \frac{x_i^2 - x_i\bar{x} + \bar{x}^2 - x_i\bar{x} + x_i}{N-1}$$

$$= \sum_{i=1}^{n} \frac{(x_i - \bar{x})}{N-1} \Rightarrow Var(x)$$

That's why we put Cov(x,x) and Cov(y,y) are mentioned at Var(x) and Var(y).

Suppose, the same as 3x3 matrix

	X	Y	Z
X	Var(x)	Cov(x,y)	Cov(x,z)
Y	Cov(y,x)	Var(y)	Cov(y,z)
Z	Cov(z,x)	Cov(z,y)	Var(z)

Then Consider,

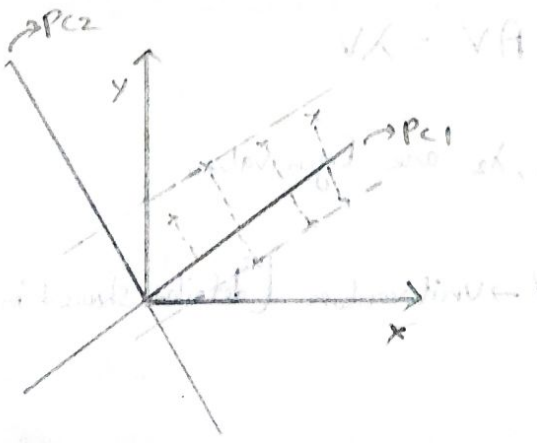
$$A = \begin{bmatrix} Var(x) & Cov(x,y) \\ Cov(y,x) & Var(y) \end{bmatrix}$$

$$\boxed{A.V = \lambda.V}$$

Where λ denotes λ_1, λ_2 i.e (f_1 and f_2)

$\lambda_1 \downarrow$ PC1 $\quad \lambda_2 \downarrow$ PC2

Then the $\boxed{\lambda_1, \lambda_2}$ are the Eigen Values.

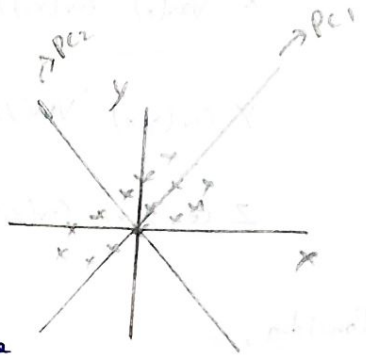


Overall steps for Calculations:

1] Change dimension

2D → 1D

2] Standardize the data.



Once, Standardization is applied. It Comes to the Center.

3] Covariance matrix x and y

$$A = \begin{array}{c|cc} & x & y \\ x & \text{var}(x) & \text{cov}(x,y) \\ y & \text{cov}(y,x) & \text{var}(y) \end{array}$$

4] Find out Eigen vectors and Value.

$$AV = \lambda V.$$

λ_1, λ_2 are Eigen values.

V → Unit vector [detailed showed in graph in website]

λ_1, λ_2 $\lambda_1 - PC_1$

$\lambda_2 - PC_2$

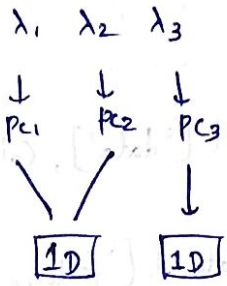
λ_1 denotes the Magnitude of eigen Vector.

⇓ In this way

Capture maximum Variance.

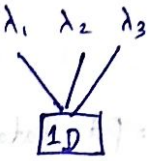
Suppose,

Change 3D → 2D

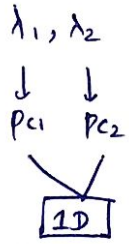


$\Rightarrow 2D$

Change 3D → 1D



Change 2D → 1D



Likewise, we change the dimension on PCA.

PRACTICAL IMPLEMENTATION ON PCA:

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

```
# Load the dataset

from Sklearn.datasets import load_breast_cancer

Cancer_datasets = load_breast_cancer()

Cancer_dataset.keys()

Print (Cancer_dataset.DESCR)


df = pd.Dataframe (Cancer_dataset['data'], Columns =
                        Cancer_dataset['feature_names'])
df.head()

# Feature Extraction (PCA)

# Standardization

from Sklearn.preprocessing import StandardScaler

Scalar = StandardScaler()

Scalar.fit(df)

Scaled_data = Scalar.transform(df)

Scaled_data
```

Applying PCA algorithms

```
from Sklearn.decomposition import PCA

pca = PCA(n_components = 2)  # n_components are the change dimension need.

data_pca = pca.fit_transform(scaled_data)

data_pca
```

Output: array([[9.128 , 1.9458
 - -
 - -
 -]])

Change to two Columns.

```
pca.explained_variance_
```

Output: array([13.3049907, 5.7013746])

↓ feature 1 Captured Variance of data

↓ Feature 2 Captured Variance of data

fit PCA without n_components. it gives more feature and explained -variance for each and every feature.

Finally, plot the figure

```
plt.figure(figsize=(8,6))

plt.scatter(data_pca[:,0], data_pca[:,1], c= cancer_dataset['target'], cmap = 'plasma')
```

```
plt.xlabel ('First Principal Component')
plt.ylabel ('Second Principal Component')
```

Practical implementation of k-means on kaish naik Day-06

[This is very long code].

Definition of Bias and Variance:

Suppose, Training Dataset = 90% Acc
Test Dataset = 70% Acc
} => Overfitting.

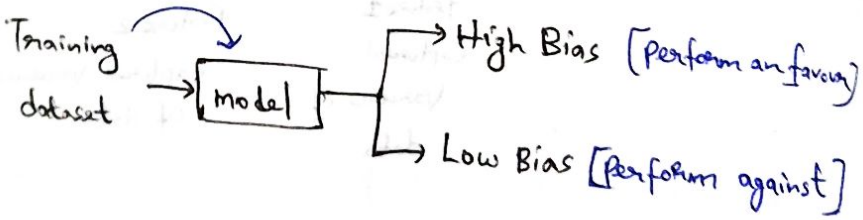
⇓

Low Bias
High Variance

Bias:

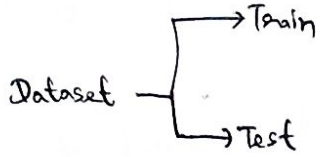
It is a phenomenon that shows the result of an algorithm in favour or against an idea.

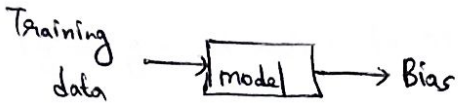
↳ Training dataset.

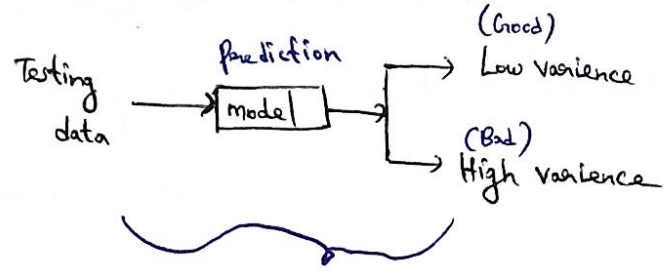


Variance:

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







⇓

This says about the perfect definition.

Model 1	model 2	model 3
Training Acc = 90%	Train Acc = 60%	Train Acc = 90%
Test Acc = 75%	Test Acc = 55%	Test Acc = 92%
⇓	⇓	⇓
{ Low Bias, High Variance }	{ High Bias, High Variance }	{ Low Bias, Low Variance }
⇓	⇓	⇓
Overfitting	Underfitting	Generalised Model