

CHAPTER-4

PART-A

4. Illustrate with an example, Multi-Dimensional scaling can work as long as we have the pairwise distances between objects. We do not actually need to represent the objects as vectors at all as long as we have some measure of similarity.

ANSWER:

Multidimensional Scaling

Multidimensional Scaling is a family of statistical methods that focus on creating mappings of items based on distance. Inside Multidimensional Scaling, there are methods for different types of data:

- **Metric Multidimensional Scaling**, also called **Principal Coordinate Analysis**, is a subtype of Multidimensional Scaling that deals with *numerical distances*, in which there is *no measurement error* (you have exactly one distance measure for each pair of items).
- **Nonmetric Multidimensional Scaling** is a subcategory of Multidimensional Scaling that deals with *non-numerical distances between items*, in which there is *no measurement*

error (you have exactly one distance measure for each pair of items).

- **Individual Differences Scaling** is a type of Multidimensional Scaling that applies when you have *multiple (different) estimates of the distances* between items. This is often the case when multiple individuals each give an estimation of the distances between all the pairs.
- **Multidimensional Analysis of Preference** is a type of Multidimensional Scaling that applies when you have *multiple (different) estimates of the distances* between items. This method does not work on distance matrices yet rather it works on *ranking data* (multiple participants each rank items from best to worst).

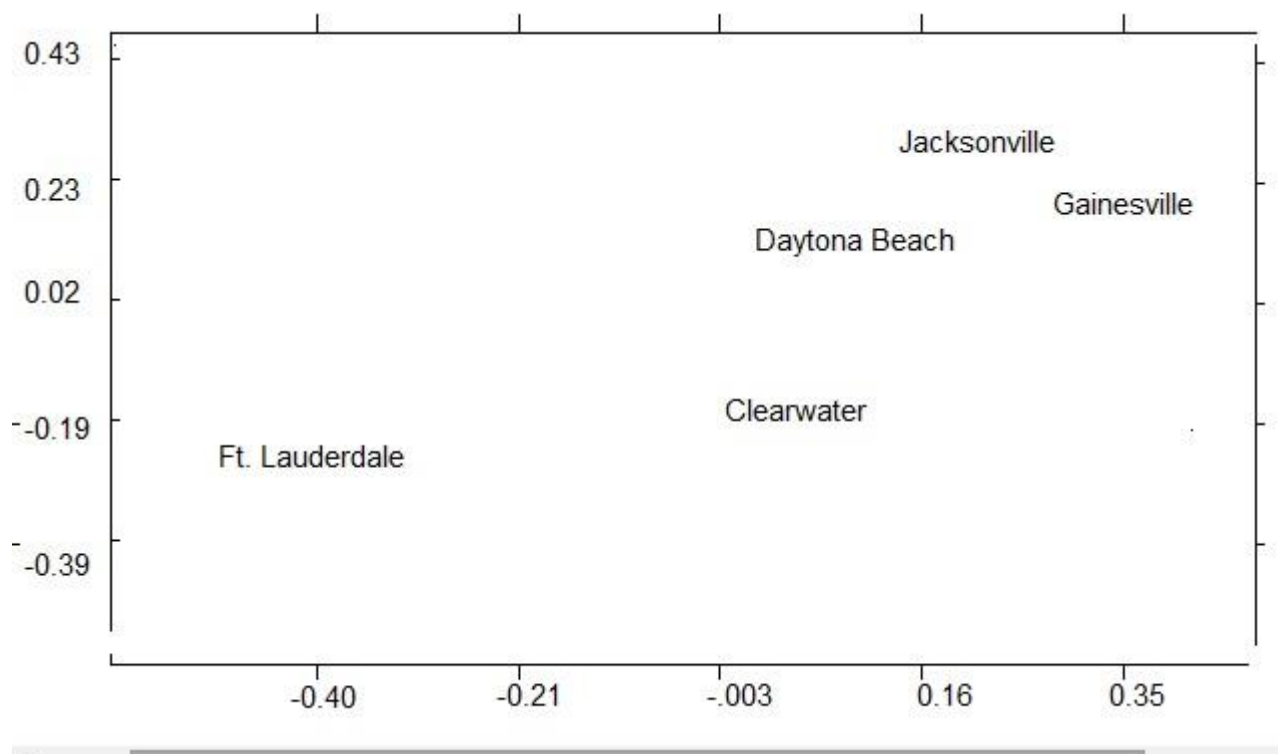
The most interesting (statistically speaking) applications of Multidimensional Scaling are those in which you have multiple participants giving (slightly or very) different estimates. Therefore, I will go a bit faster with Metric and Nonmetric Multidimensional Scaling and I'll spend a bit more time on Individual Differences Scaling and Multidimensional Analysis of Preference.

A Simple Example

Multidimensional scaling uses a square, [symmetric matrix](#) for input. The matrix shows relationships between items. For a simple example, let's say you had a set of cities in Florida and their distances:

CITY	Clearwater	Daytona Beach	Ft. Lauderdale	Gainesville	Jacksonville
Clearwater	0	159	247	131	197
Daytona Beach	159	0	230	97	89
Ft. Lauderdale	247	230	0	309	317
Gainesville	131	97	309	0	68
Jacksonville	197	89	317	68	0

The scaling produces a graph like the one below.



The very simple example above shows cities and distances, which are easy to visualize as a map. However, multidimensional scaling can work on “theoretically” mapped data as well. For example, Kruskal and Wish (1978) outlined how the method could be used to uncover the answers to a variety of questions about people’s viewpoints on political candidates. This could be achieved by reducing the data and issues (say, partisanship and ideology) to a two-dimensional map.

5. How can we incorporate class information into Isomap and LLE such that instances of the same class are mapped to nearby locations in the new space?

ANSWER:

Q.How does Isometric Mapping (Isomap) work?

Isomap is a technique that combines several different algorithms, enabling it to use a non-linear way to reduce dimensions while preserving local structures.

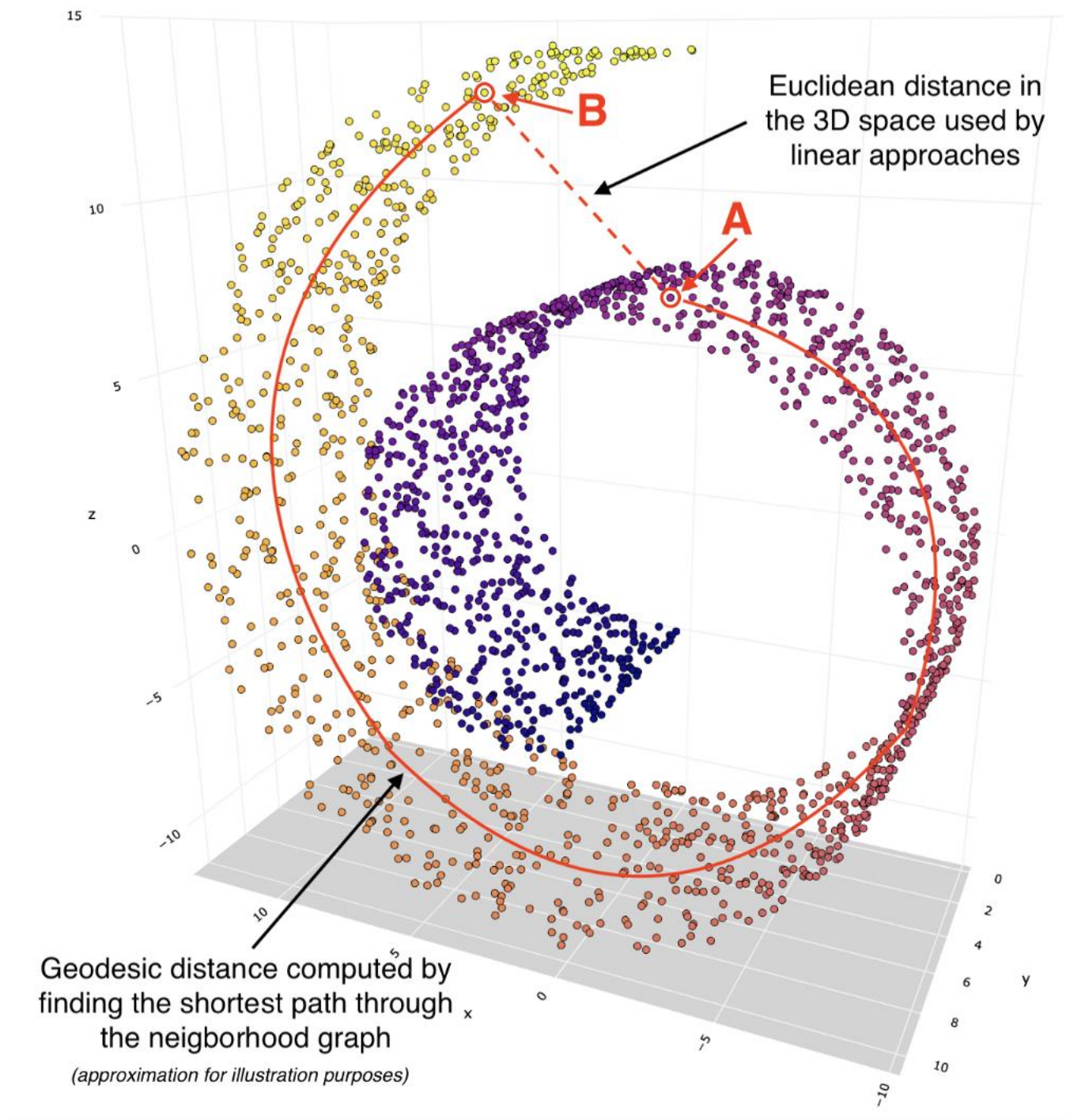
Before we look at the example of Isomap and compare it to a linear method of Principal Components Analysis (PCA), let's list the high-level steps that Isomap performs:

1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, "k" is an arbitrary number of neighbors that you can specify within model hyperparameters.
2. Once the neighbors are found, **construct the neighborhood graph** where points are connected to each other if they are each other's neighbors. Data points that are not neighbors remain unconnected.
3. **Compute the shortest path** between each pair of data points (nodes). Typically, it is either Floyd-Warshall or Dijkstra's algorithm that is used for this task. Note, this step is also commonly described as finding a **geodesic distance** between points.
4. Use [multidimensional scaling \(MDS\)](#) to **compute lower-dimensional embedding**. Given distances between each pair of points are known, MDS places each object into the N-dimensional space (N is specified as a hyperparameter) such that the between-point distances are preserved as well as possible.

For our example, let's create a 3D object known as a Swiss roll. The object is made up of 2,000 individual data points. The chart is **interactive** so make sure to rotate it to familiarize yourself with its exact shape.

Interactive 3D swiss roll. Graph by [author](#).

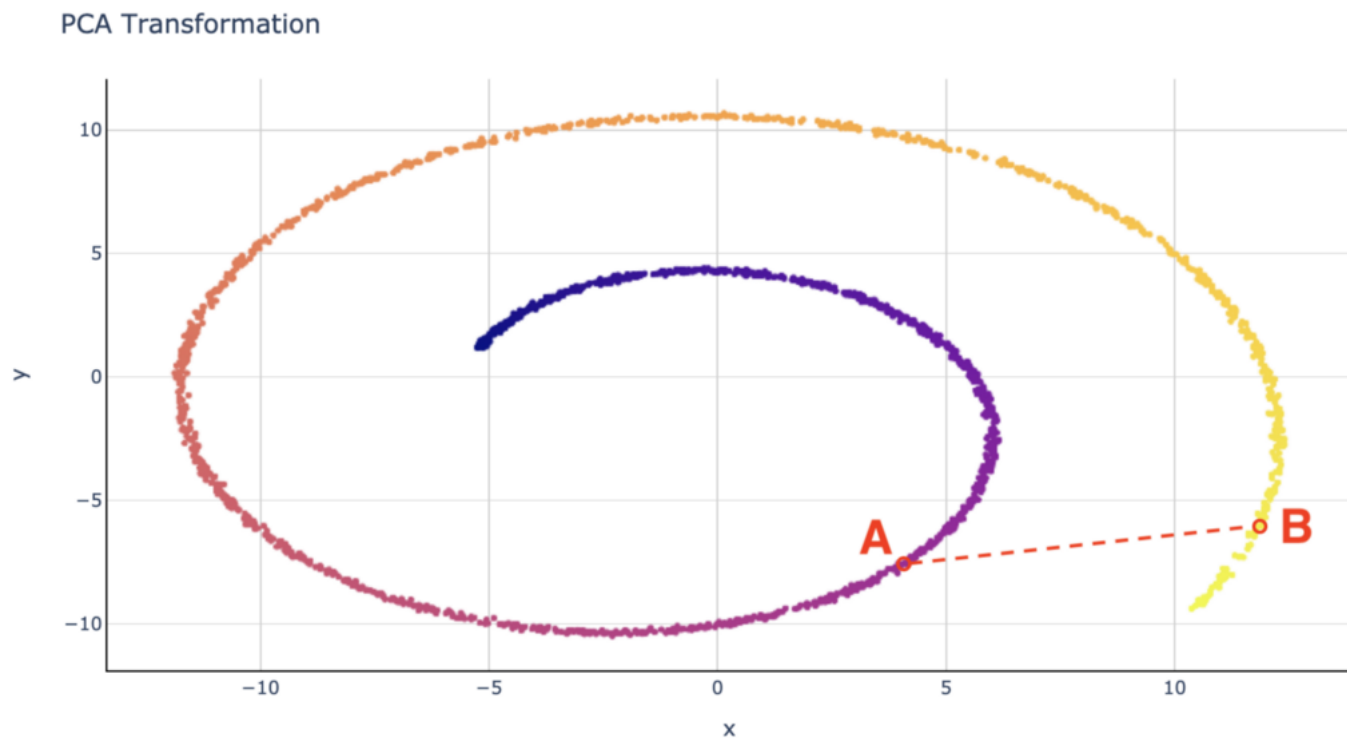
Next, we want to map this 3-dimensional swiss roll 2 dimensions using Isomap. To track what happens during this transformation, let's select two points: A and B.



Euclidean vs. geodesic distances between points on a 3D Swiss roll. Image by [author](#).

We can see that these two points are relatively close to each other within the 3D space. If we used a linear dimensionality reduction approach such as PCA, then the Euclidean distance between these two

points would remain somewhat similar in lower dimensions. See PCA transformation chart below:

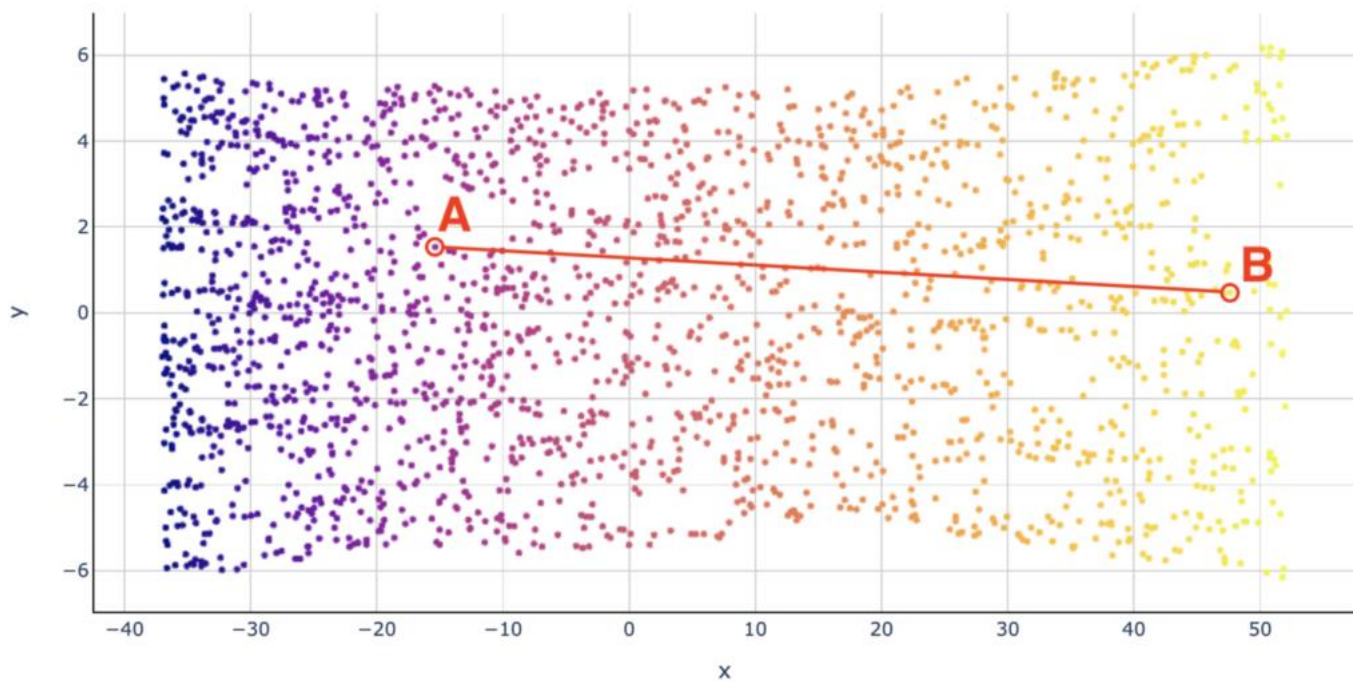


3D swiss roll reduced to 2 dimensions using PCA. Image by [author](#).

Note, the shape of a 2D object in PCA looks like a picture taken of the same 3D object but from a specific angle. This is a feature of linear transformation.

Meanwhile, non-linear approaches such as Isomap give us a very different result. We can describe such transformation as unrolling the Swiss roll and laying it flat on a 2D surface:

Isomap transformation




3D swiss roll reduced to 2 dimensions using Isomap. Image by [author](#).

We can see that the distance between points A and B in a 2D space is based on the geodesic distance computed by going through the neighborhood connections.

That's the secret to Isomap's ability to perform non-linear dimensionality reduction while balancing the relationship between local and global structures

Locally Linear Embedding (LLE) within the universe of Machine Learning algorithms

Even an experienced Data Scientist can easily get lost amongst hundreds of different Machine Learning algorithms used in the industry. Hence, I believe there is value in creating a structure by putting some of the most frequently used algorithms into categories.

It will never be perfect because some algorithms are flexible enough to perform multiple tasks. Nevertheless, below is my attempt to create such categorization. Make sure to explore this **interactive** graph by clicking  on different sections **to reveal more**.

Machine Learning algorithm classification. Interactive chart created by the [author](#).

As you can see, Locally Linear Embedding (LLE) sits under the **Unsupervised** branch of Machine Learning within the group of **dimensionality reduction** algorithms.

It means that you do not need to have a target variable to perform dimensionality reduction with LLE, unlike supervised techniques such as Linear Discriminant Analysis (LDA).

Q.How does Locally Linear Embedding work?

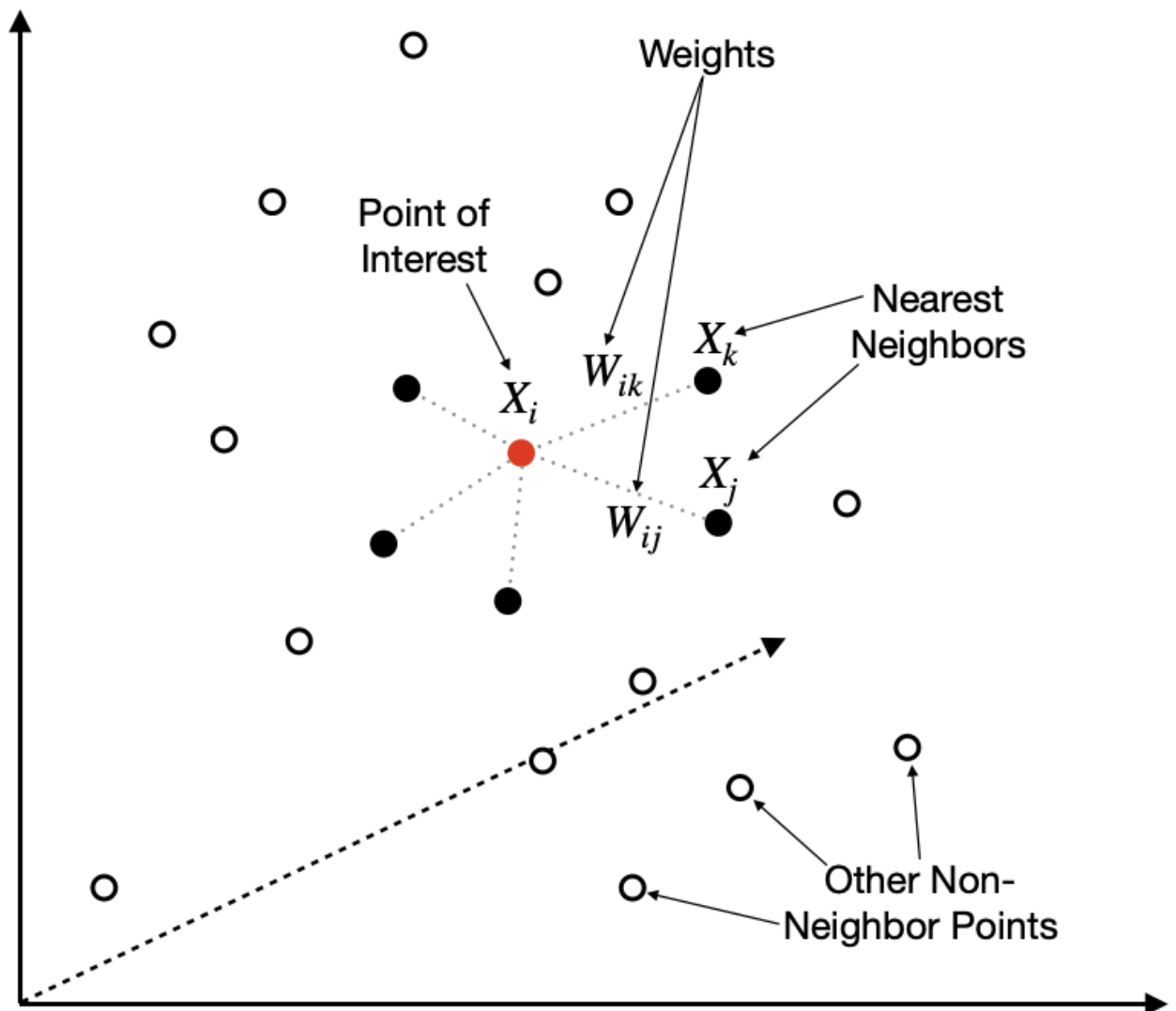
ANSWER:

The high-level steps

Similar to Isomap, LLE combines several steps to produce the lower-dimensional embedding. These are:

1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, “k” is an arbitrary number of neighbors that you can specify within model hyperparameters.
2. **Construct a weight matrix** where every point has its weights determined by minimizing the error of the cost function shown below. Note that every point is a linear combination of its neighbors, which means that **weights for non-neighbors are 0**.

Original High-Dimensional Space



We know the
position of the
Point of Interest

We know the
positions of all the
Nearest Neighbors

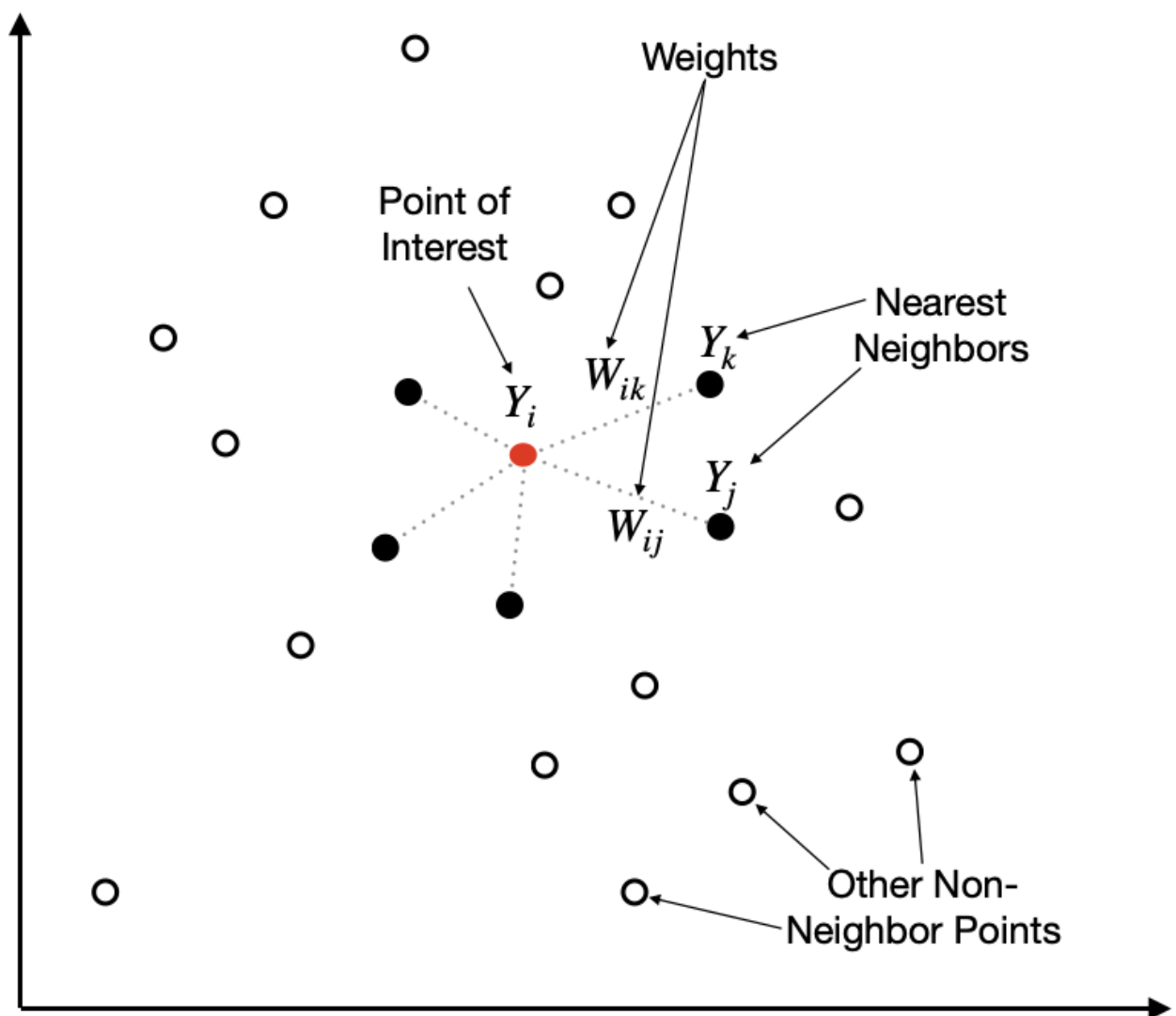
$$E(W) = \sum_i |X_i - \sum_j W_{ij} X_j|^2$$

The cost function is solved to find the weights,
where the sum of weights for each X_i is set to equal to 1

Finding weights in the original high-dimensional space. Image by [author](#).

3. Find the positions of all the points in the **new lower-dimensional embedding** by minimizing the cost function shown below. Note, here we use weights (W) from step two and solve for Y . The actual solving is performing using Partial Eigenvalue Decomposition.

New Lower-Dimensional Space



We know the weights from the previous step

$$C(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2$$

Arrows point from the text 'We know the weights from the previous step' to the W_{ij} term in the equation. Another arrow points from the Y_i term to the text below.

The cost function is solved to find the positions of Y_i and its neighbors in the new lower-dimensional space using weights from the previous step.

Finding the position of points in the new lower-dimensional embedding. Image by [author](#).

With the above steps completed, we get a lower-dimensional representation of the data, which we can typically visualize using standard scatterplots, provided we reduce the dimensionality to 3D or less.

6. Identify the discriminant for this case. Another possibility of using Gaussian densities is to have them all diagonal but allow them to be different.

ANSWER:

7. What type of boundaries can be defined? Let us say in two dimensions, we have two classes with exactly the same mean.

ANSWER:

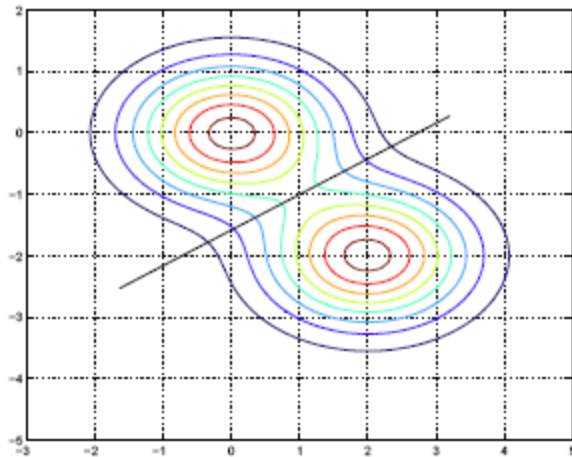
Binary Classification

In binary classification in particular, for instance if we let ($k = 1, l = 2$), then we would define constant a_0 , given below, where π_1 and π_2 are prior probabilities for the two classes and μ_1 and μ_2 are mean vectors.

- Binary classification ($k = 1, l = 2$):
 - Define $a_0 = \log \pi_1 \pi_2 - \frac{1}{2}(\mu_1 + \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2)$
 - Define $(a_1, a_2, \dots, a_p)^T = \Sigma^{-1}(\mu_1 - \mu_2)$
 - Classify to class 1 if $a_0 + \sum_{j=1}^p a_j x_j > 0$; to class 2 otherwise.
 - An example

- * $\pi_1=\pi_2=0.5$
- * $\mu_1=(0,0)^T, \mu_2=(2,-2)^T$
- * $\Sigma=(1.00.00.00.5625)$
- *Decision boundary: $5.56-2.00x_1+3.56x_2=0.0$

Here is a contour plot of this result:



We have two classes and we know the within-class density. The marginal density is simply the weighted sum of the within-class densities, where the weights are the prior probabilities. Because we have equal weights and because the covariance matrix two classes are identical, we get these symmetric lines in the contour plot. The black diagonal line is the decision boundary for the two classes. Basically, if you are given an x above the line, then we would classify this x into the first-class. If it is below the line, we would classify it into the second class.

There is a missing piece here, right?

For all of the discussion above we assume that we have the prior probabilities for the classes and we also had the within-class densities given to us. Of course, in practice, you don't have this. In practice, what we have is only a set of training data.

8.How can we find the remaining ones if we already know some of the factors in Factor Analysis,

ANSWER:

Factor Analysis

Factor analysis is a technique that is used to reduce a large number of variables into fewer numbers of factors. This technique extracts maximum common variance from all variables and puts them into a common score. As an index of all variables, we can use this score for further analysis. Factor analysis is part of **general linear model (GLM)** and this method also assumes several assumptions: there is linear relationship, there is no **multicollinearity**, it includes relevant variables into analysis, and there is true correlation between variables and factors. Several methods are available, but principal component analysis is used most commonly.

Types of factoring:

There are different types of methods used to extract the factor from the data set:

1. **Principal component analysis:** This is the most common method used by researchers. PCA starts extracting the maximum variance and puts them into the first factor. After that, it removes that variance explained by the first factors and then starts extracting maximum variance for the second factor. This process goes to the last factor.
2. **Common factor analysis:** The second most preferred method by researchers, it extracts the common variance and puts them into factors. This method does not include the unique variance of all variables. This method is used in SEM.
3. **Image factoring:** This method is based on correlation matrix. OLS Regression method is used to predict the factor in image factoring.
4. **Maximum likelihood method:** This method also works on correlation metric but it uses maximum likelihood method to factor.
5. **Other methods of factor analysis:** Alfa factoring outweighs least squares. Weight square is another regression based method which is used for factoring.

Factor loading:

Factor loading is basically the correlation coefficient for the variable and factor. Factor loading shows the variance explained by the variable on that particular factor. In the SEM approach, as a rule of thumb, 0.7 or higher factor loading represents that the factor extracts sufficient variance from that variable.

Eigenvalues: Eigenvalues is also called characteristic roots. Eigenvalues shows variance explained by that particular factor out of the total variance. From the commonality column, we can know how much variance is explained by the first factor out of the total variance. For example, if our first factor explains 68% variance out of

the total, this means that 32% variance will be explained by the other factor.

Factor score: The factor score is also called the component score. This score is of all row and columns, which can be used as an index of all variables and can be used for further analysis. We can standardize this score by multiplying a common term. With this factor score, whatever analysis we will do, we will assume that all variables will behave as factor scores and will move.

Criteria for determining the number of factors: According to the Kaiser Criterion, Eigenvalues is a good criteria for determining a factor. If Eigenvalues is greater than one, we should consider that a factor and if Eigenvalues is less than one, then we should not consider that a factor. According to the variance extraction rule, it should be more than 0.7. If variance is less than 0.7, then we should not consider that a factor.

Rotation method: Rotation method makes it more reliable to understand the output. Eigenvalues do not affect the rotation method, but the rotation method affects the Eigenvalues or percentage of variance extracted. There are a number of rotation methods available: (1) No rotation method, (2) Varimax rotation method, (3) Quartimax rotation method, (4) Direct oblimin rotation method, and (5) Promax rotation method. Each of these can be easily selected in **SPSS**, and we can compare our variance explained by those particular methods.

Assumptions:

1. **No outlier:** Assume that there are no outliers in data.
2. **Adequate sample size:** The case must be greater than the factor.
3. **No perfect multicollinearity:** Factor analysis is an interdependency technique. There should not be perfect multicollinearity between the variables.
4. **Homoscedasticity:** Since factor analysis is a linear function of measured variables, it does not require homoscedasticity between the variables.
5. **Linearity:** Factor analysis is also based on linearity assumption. Non-linear variables can also be used. After transfer, however, it changes into linear variable.
6. **Interval Data:** Interval data are assumed.

9. Demonstrate an application where there are hidden factors (not necessarily linear) and where factor analysis would be expected to work well.

ANSWER:

You perform a factor analysis to see if there are really these three factors. If they do, you will be able to create three separate scales, by summing the items on each dimension.

Factor analysis is based on a correlation table. If there are k items in the study (e.g. k questions in the above example) then the correlation table has $k \times k$ entries of form r_{ij} where each r_{ij} is the correlation coefficient between item i and item j . The main diagonal consists of entries with value 1.

Closely related to factor analysis is **principal component analysis**, which creates a picture of the relationships between the variables useful in identifying common factors.

Factor analysis is based on various concepts from Linear Algebra, in particular eigenvalues, eigenvectors, orthogonal matrices and the spectral theorem. We review these concepts first before explaining how principal component analysis and factor analysis work.

Factor analysis is a way to take a mass of data and shrinking it to a smaller data set that is more manageable and more understandable. It's a way to find hidden patterns, show how those patterns overlap and show what characteristics are seen in multiple patterns. It is also used to create a set of [variables](#) for similar items in the set (these sets of variables are called dimensions). It can be a very useful tool for complex sets of data involving psychological studies, socioeconomic status and other involved concepts.

A "factor" is a set of [observed variables](#) that have similar response patterns; They are associated with a hidden variable (called a [confounding variable](#)) that isn't directly measured. Factors are listed according to factor loadings, or how much variation in the data they can explain

10. Illustrate the computer program that does this for different values of k and c . In image compression, k-means can be used as follows: The image is divided into non-overlapping c cross c windows and these c^2 -dimensional vectors make up the sample. For a given k , which is generally a power of two, we do k-means clustering. The reference vectors and the indices for each window is sent over the communication line. At the receiving end, the image is then re-constructed by reading from the table of reference vectors using the indices. For each case, calculate the re-construction error and the compression rate.

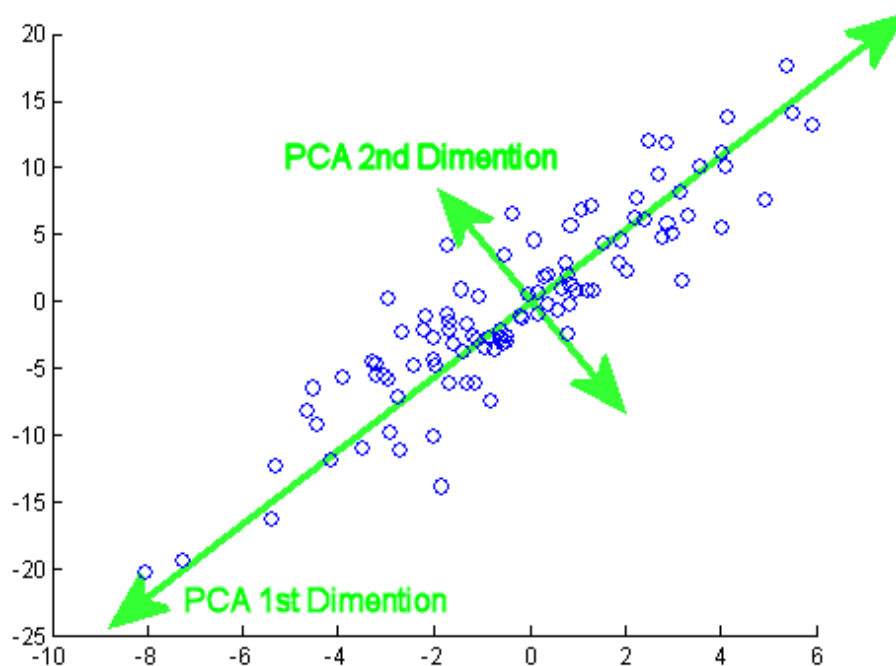
PART-B

1 What is the relationship between PCA and K-Means Clustering?

ANSWER:

Principal Component Analysis(PCA) is a tool for dimension reduction. This technique is to transform the larger dataset into a smaller dataset by identifying the correlations and patterns with preserving most of the valuable information.

This is need for feature selection of a model. PCA aims to capture valuable information explaining high variance which results in providing the best accuracy.



It makes the data visualizations easy to handle. It decreases the complexity of the model and increases computational efficiency.

Steps Involved in the PCA analysis

Step 1: Standardize the dataset.

Step 2: Calculate the covariance matrix for the features in the dataset.

Step 3: Calculate the eigenvalues and eigenvectors for the covariance matrix.

Step 4: Sort eigenvalues and their corresponding eigenvectors.

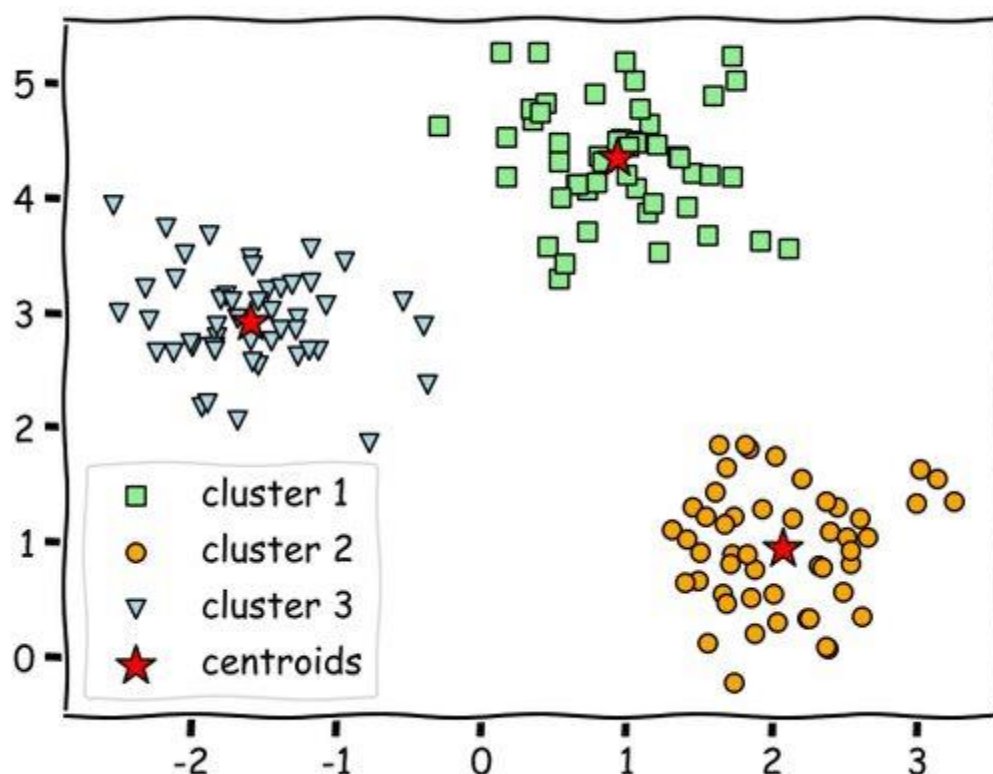
K Means Clustering:

Before you know that What is Clustering? Clustering is an unsupervised learning algorithm. It is used to identify groups of similar objects in a multivariate dataset collected from various industries.

There are many algorithms in the Clustering algorithm. One of the most popular algorithms is k means clustering algorithm.

K-Means Clustering:

It is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups(clusters) where each data point belongs to only one group.



The way K means algorithm works is as follows:

- Specify the number of clusters K.

- Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
- Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

Relationship between PCA and K Means Clustering:

Both uses are in dimensionality reduction for visualizing patterns in data from parameters(variables).

PCA in conjunction with K-means is a powerful method for visualizing high dimensional data.

To better understand the PCA, let's see the basic steps:

Step 1: Reduce Dimensionality(PCA)

Step 2: Find the Clusters(from PCA(lower dimensionality) to K means clusters)

Step3: Visualize and Interpret the Clusters(K means clustered using PCA components(variances))

All the best, continue to learn

2. How to find the best subset of selection of features?

ANSWER:

Understand Best Subset Selection

When building a regression model, removing irrelevant variables will make the model easier to interpret and less prone to overfit the data, therefore more generalizable.

Best subset selection is a method that aims to find the subset of independent variables (X_i) that best predict the outcome (Y) and it does so by considering all possible combinations of independent variables.

We will start by explaining how this method works and then we will discuss its advantages and limitations.

So let's get started!

How best subset selection works

If we have k independent variables under consideration, best subset selection:

1. **Starts** by considering all possible models with 1 variable, 2 variables, ..., k variables
2. **Then** chooses the best model of size 1, the best model of size 2, ..., the best model of size k
3. **Lastly**, from these finalists, it chooses the best overall model

Best subset selection ends up selecting 1 model from 2^k possible models.

Here's an example of best subset selection with 3 variables:

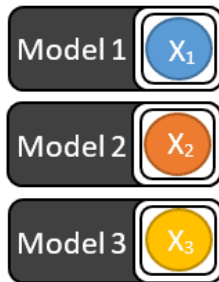
Best Subset Selection: Example with 3 Variables



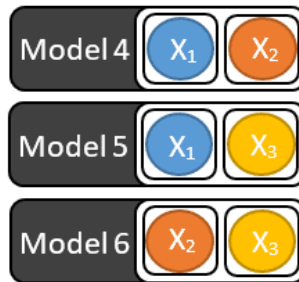
Step 1: Consider All Possible Models

By listing all possible combination of variables

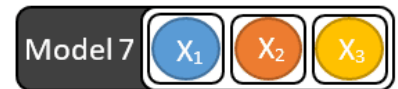
Models with 1 variable:



Models with 2 variables:



Models with 3 variables:



Step 2: Identify the Best Model of Each Size

By choosing the one with the lowest sum of squared errors or the highest R^2

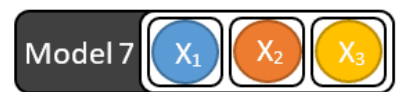
Best model with 1 variable:



Best model with 2 variables:



Best model with 3 variables:



Step 3: Identify the Best Overall Model

By choosing the one with the lowest AIC (or BIC) or the highest adjusted R^2

Best overall model:



Other than enumerating all possible models (1st step in the figure above), we have 2 key steps that we need to understand in order to develop an intuition on how the best subset algorithm is working under the hood:

1. How to identify the best model of each size (2nd step in the figure above)

2. How to identify the best overall model (3rd step in the figure above)

1. Determine the best model of each size

The best model can be:

- The one that predicts the outcome with less errors — so the one with the lowest RSS (Residuals Sum of Squares)
- Or the one that explains the largest percentage of variance of the outcome — so the one that has the largest R^2

Note that in both cases the selected model will be the same.

2. Determine the best overall model

Unlike step 2, where we compared models of the same size, in step 3 we will be comparing models of different sizes, so we can no longer use RSS or R^2 to select the best overall model.

Why not?

Because the model with most variables is always going to have the lowest sum of squared errors and the highest R^2 . So we cannot lose by adding more predictors, but something worse can happen which is choosing a model that overfits the data — that is closely fitting the sample data while having a low out-of-sample accuracy.

In order to deal with this problem, one solution is to choose the best overall model according to a statistic that imposes some sort of penalty on bigger models, especially when they contain additional variables that barely provide any improvement.

This is where the AIC, BIC and adjusted R^2 come into play.

What these will do, in simple terms, is estimate the out-of-sample accuracy.

Of course you can do better than estimating:

You can leave out part of your dataset to test these models and get a direct measure of the out-of-sample accuracy. But this can only be done in special cases where data collection is not expensive and time consuming (which is almost never the case, especially in medical research).

BOTTOM LINE:

Depending on which parameter you choose, the best overall model will be the one that has the lowest AIC, BIC, or the highest adjusted R^2 .

Advantages and limitations of best subset selection

Advantages of best subset selection:

Best subset selection:

1. Improves out-of-sample accuracy (generalizability) of the regression model by eliminating unnecessary predictors
2. Yields a simple and easily interpretable model
3. Provides a reproducible and objective way to reduce the number of predictors compared to manually choosing variables which can be manipulated to serve one's own hypotheses and interests

Note that automated variable selection is not meant to replace expert opinion. In fact, important variables judged by background knowledge should still be entered in the model even if they are statistically non-significant.

Limitations of best subset selection:

Best subset selection suffers from 2 major limitations:

1. Computational limitation:

The number of models the best subset algorithm has to consider grows exponentially with the number of predictors under consideration.

For example:

- For 3 predictors, the best subset algorithm has to consider $2^3 = 8$ models
- For 10 predictors, $2^{10} = 1024$ models
- For 20 predictors, $2^{20} = 1,048,576$ models!

2. Theoretical limitation:

The idea of considering all possible combinations of independent variables can both be seen as:

- Good, because we are choosing the best model from ALL possible solutions
- Bad, because choosing from thousands even millions of models can be considered p-hacking (a.k.a. data dredging), as one model can look better just by chance and end up underperforming when we try to validate it on new data

In recent years, Bertsimas et al. developed a new algorithmic approach that runs best subset selection with a sample size of 1000 and number of predictors of 100 in a matter of minutes!

According to our calculations above, for 100 predictors, the computer needs to run 1.26×10^{30} models — which is absolutely ridiculous!

Not only did they solve the computational problem (and by the way made the algorithm available in an R package called *leaps*), it turned out that the problem of overfitting the sample data (the theoretical limitation) is not that limiting in practice, as the model they obtained via best subset selection outperformed LASSO and forward stepwise selection in out-of-sample prediction accuracy.

3.What are the similarities and Differences between Avererage link clustering and K- Means

ANSWER:

[k-means](#) is method of cluster analysis using a pre-specified no. of clusters. It requires advance knowledge of 'K'.

[Hierarchical clustering](#) also known as hierarchical cluster analysis (HCA) is also a method of cluster analysis which seeks to build a hierarchy of clusters without having fixed number of cluster.

Main differences between K means and Hierarchical Clustering are:

k-means Clustering

k-means, using a pre-specified number of clusters, the method assigns records to each cluster to find the mutually exclusive cluster of spherical shape based on distance.

K Means clustering needed advance knowledge of K i.e. no. of clusters one want to divide your data.

Hierarchical Clustering

Hierarchical methods can be either divisive or agglomerative.

In hierarchical clustering one can stop at any number of clusters, one find appropriate by interpreting the dendrogram.

One can use median or mean as a cluster centre to represent each cluster.

Agglomerative methods begin with 'n' clusters and sequentially combine similar clusters until only one cluster is obtained.

Methods used are normally less computationally intensive and are suited with very large datasets.

Divisive methods work in the opposite direction, beginning with one cluster that includes all the records and Hierarchical methods are especially useful when the target is to arrange the clusters into a natural hierarchy.

In K Means clustering, since one start with random choice of clusters, the results produced by running the algorithm many times may differ.

In Hierarchical Clustering, results are reproducible in Hierarchical clustering

K- means clustering a simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset).

A hierarchical clustering is a set of nested clusters that are arranged as a tree.

K Means clustering is found to work well when the structure of the clusters is hyper spherical (like circle in 2D, sphere in 3D).

Hierarchical clustering don't work as well as, k means when the shape of the clusters is hyper spherical.

Advantages: 1. Convergence is guaranteed.

Advantages: 1 .Ease of handling of any forms of similarity or distance.

2. Specialized to clusters of different sizes and shapes.

2. Consequently, applicability to any attributes types.

Disadvantages: 1. K-Value is difficult to predict 2. Didn't work well with global cluster.

Disadvantage: 1. Hierarchical clustering requires the computation and storage of an $n \times n$ distance matrix. For very large datasets, this can be expensive and slow

4.How is Dimension Reduction performed on High Dimension Data?

ANSWER:

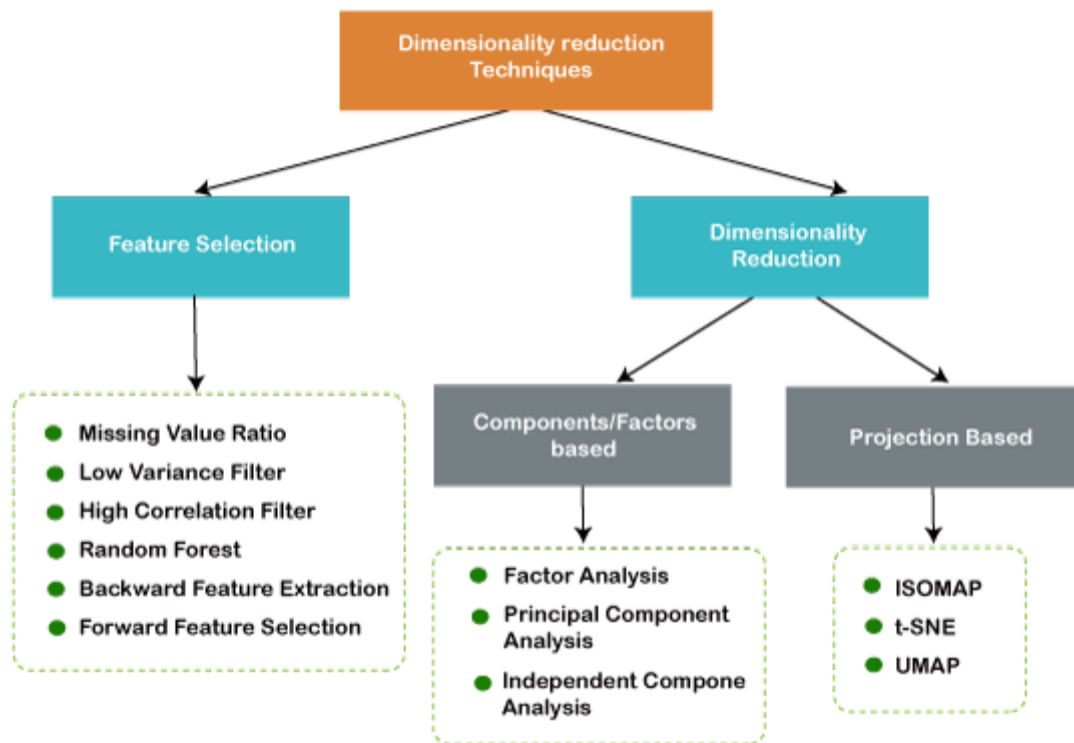
The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.

A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.

Dimensionality reduction technique can be defined as, "***It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information.***" These techniques are widely used in [machine learning](#)

for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc.** It can also be used for **data visualization, noise reduction, cluster analysis**, etc.



The Curse of Dimensionality

Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality*. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance.

Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

Benefits of applying Dimensionality Reduction

Some benefits of applying dimensionality reduction technique to the given dataset are given below:

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- It removes the redundant features (if present) by taking care of multicollinearity.

Disadvantages of dimensionality Reduction

There are also some disadvantages of applying the dimensionality reduction, which are given below:

- Some data may be lost due to dimensionality reduction.
- In the PCA dimensionality reduction technique, sometimes the principal components required to consider are unknown.

Approaches of Dimension Reduction

There are two ways to apply the dimension reduction technique, which are given below:

Feature Selection

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset.

Three methods are used for the feature selection:

1. Filters Methods

In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are:

- **Correlation**
- **Chi-Square Test**
- **ANOVA**
- **Information Gain, etc.**

2. Wrappers Methods

The wrapper method has the same goal as the filter method, but it takes a machine learning model for its evaluation. In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are:

- Forward Selection
- Backward Selection

- Bi-directional Elimination

3. Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are:

- **LASSO**
- **Elastic Net**
- **Ridge Regression, etc.**

Feature Extraction:

Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

Some common feature extraction techniques are:

- a. Principal Component Analysis
- b. Linear Discriminant Analysis
- c. Kernel PCA
- d. Quadratic Discriminant Analysis

Common techniques of Dimensionality Reduction

- a. **Principal Component Analysis**
- b. **Backward Elimination**
- c. **Forward Selection**
- d. **Score comparison**
- e. **Missing Value Ratio**
- f. **Low Variance Filter**
- g. **High Correlation Filter**
- h. **Random Forest**

- i. **Factor Analysis**
- j. **Auto-Encoder**

Principal Component Analysis (PCA)

Principal Component Analysis is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***

Backward Feature Elimination

The backward feature elimination technique is mainly used while developing Linear Regression or Logistic Regression model. Below steps are performed in this technique to reduce the dimensionality or in feature selection:

- In this technique, firstly, all the n variables of the given dataset are taken to train the model.
- The performance of the model is checked.
- Now we will remove one feature each time and train the model on $n-1$ features for n times, and will compute the performance of the model.
- We will check the variable that has made the smallest or no change in the performance of the model, and then we will drop that variable or features; after that, we will be left with $n-1$ features.
- Repeat the complete process until no feature can be dropped.

In this technique, by selecting the optimum performance of the model and maximum tolerable error rate, we can define the optimal number of features require for the machine learning algorithms.

Forward Feature Selection

Forward feature selection follows the inverse process of the backward elimination process. It means, in this technique, we don't eliminate the feature; instead, we will find the best features that can produce the highest increase in the performance of the model. Below steps are performed in this technique:

- We start with a single feature only, and progressively we will add each feature at a time.
- Here we will train the model on each feature separately.
- The feature with the best performance is selected.
- The process will be repeated until we get a significant increase in the performance of the model.

Missing Value Ratio

If a dataset has too many missing values, then we drop those variables as they do not carry much useful information. To perform this, we can set a threshold level, and if a variable has missing values more than that threshold, we will drop that variable. The higher the threshold value, the more efficient the reduction.

Low Variance Filter

As same as missing value ratio technique, data columns with some changes in the data have less information. Therefore, we need to calculate the variance of each variable, and all data columns with variance lower than a given threshold are dropped because low variance features will not affect the target variable.

High Correlation Filter

High Correlation refers to the case when two variables carry approximately similar information. Due to this factor, the performance of the model can be degraded. This correlation between the independent numerical variable gives the calculated value of the correlation coefficient. If this value is higher than the threshold value, we can remove one of the variables from the dataset. We can consider those variables or features that show a high correlation with the target variable.

Random Forest

Random Forest is a popular and very useful feature selection algorithm in machine learning. This algorithm contains an in-built feature importance package, so we do not need to program it separately. In this technique, we need to generate a large set of trees against the target variable, and with the help of usage statistics of each attribute, we need to find the subset of features.

Random forest algorithm takes only numerical variables, so we need to convert the input data into numeric data using **hot encoding**.

Factor Analysis

Factor analysis is a technique in which each variable is kept within a group according to the correlation with other variables, it means variables within a group can have a

high correlation between themselves, but they have a low correlation with variables of other groups.

5.Explain the K-Means Algorithm for the given data set?

ANSWER:

K-Means Clustering is an [Unsupervised Learning algorithm](#)

, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if $K=2$, there will be two clusters, and for $K=3$, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means [clustering](#)

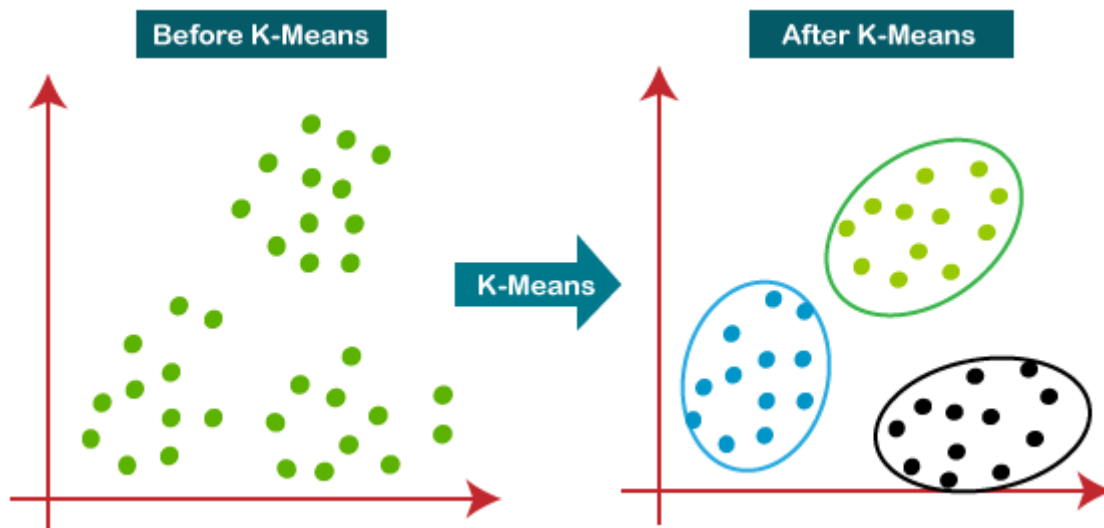
algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.

- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

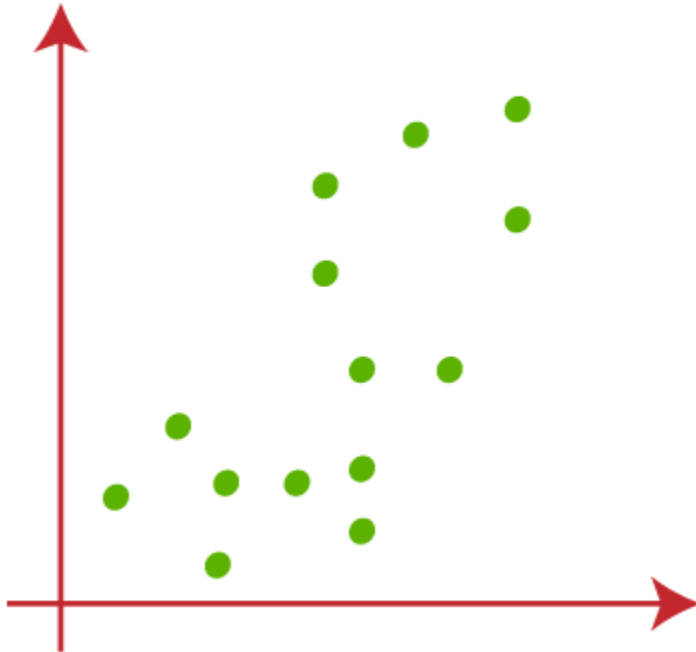
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Let's understand the above steps by considering the visual plots:

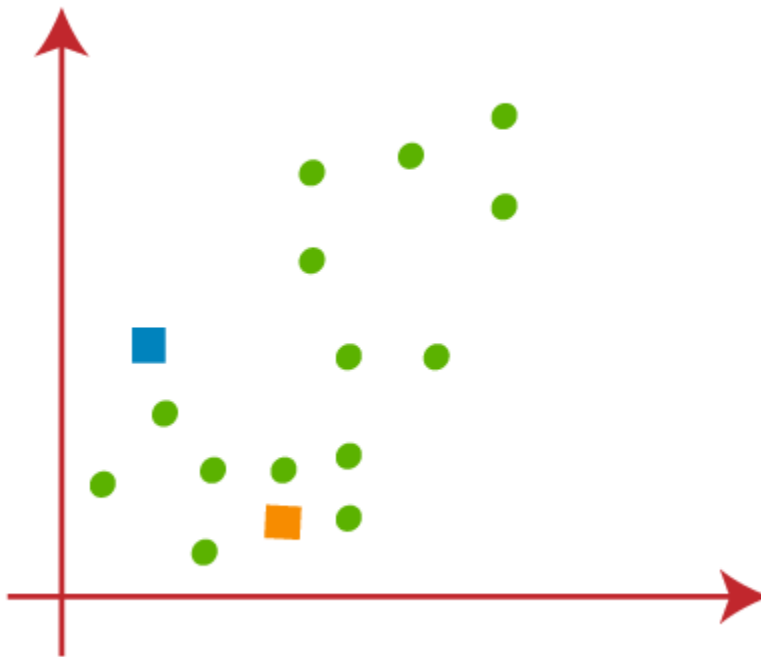
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



- Let's take number k of clusters, i.e., $K=2$, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the

below

image:



- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the

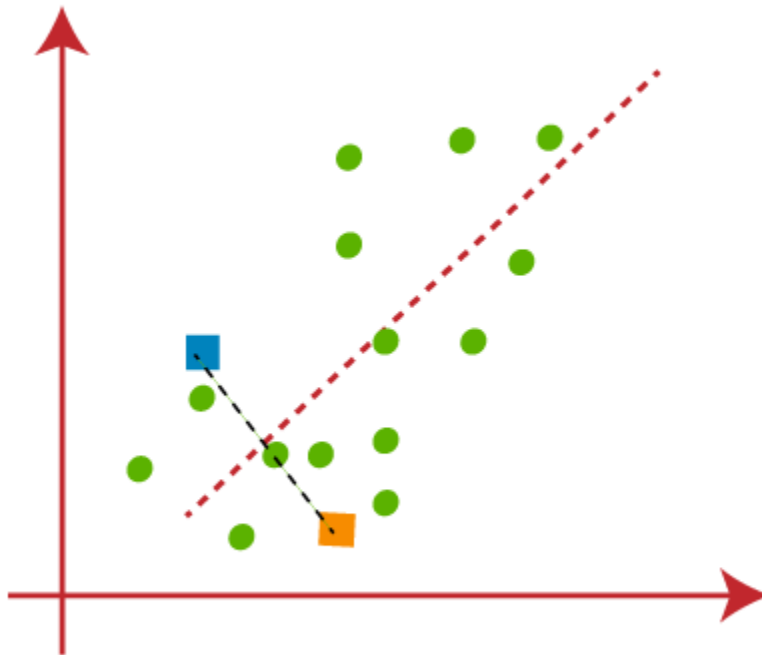
centroids.

Consider

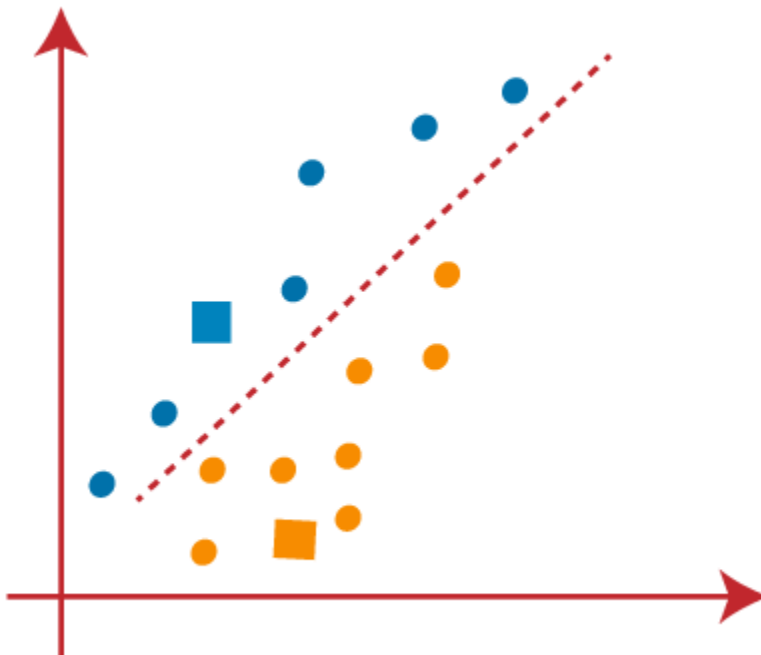
the

below

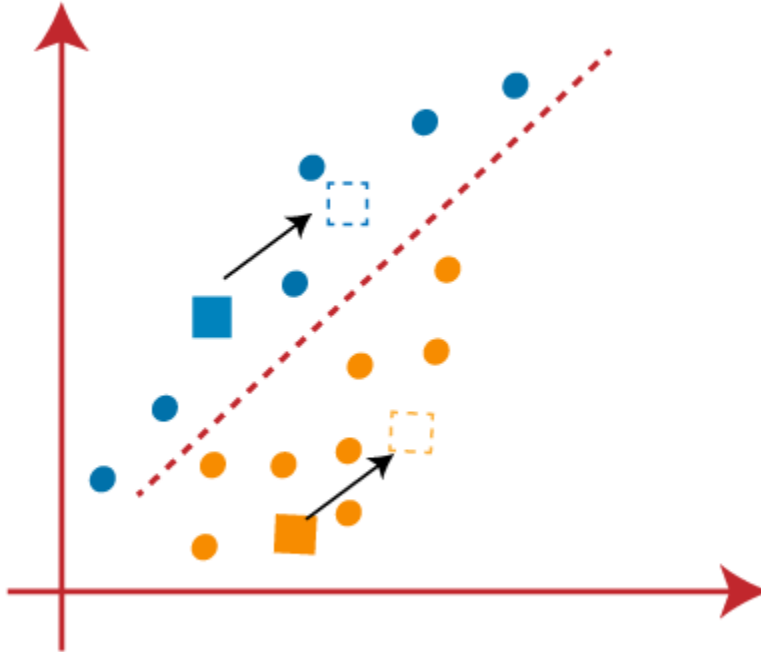
image:



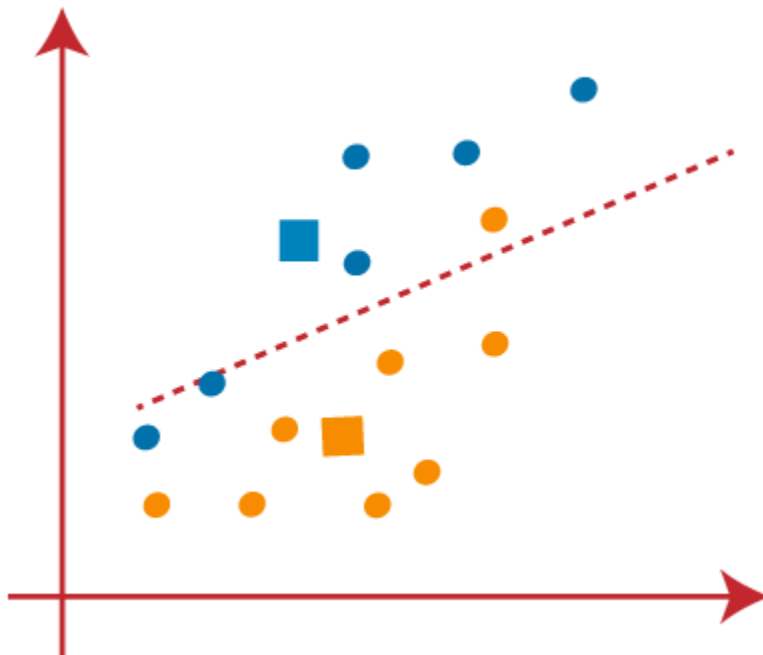
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



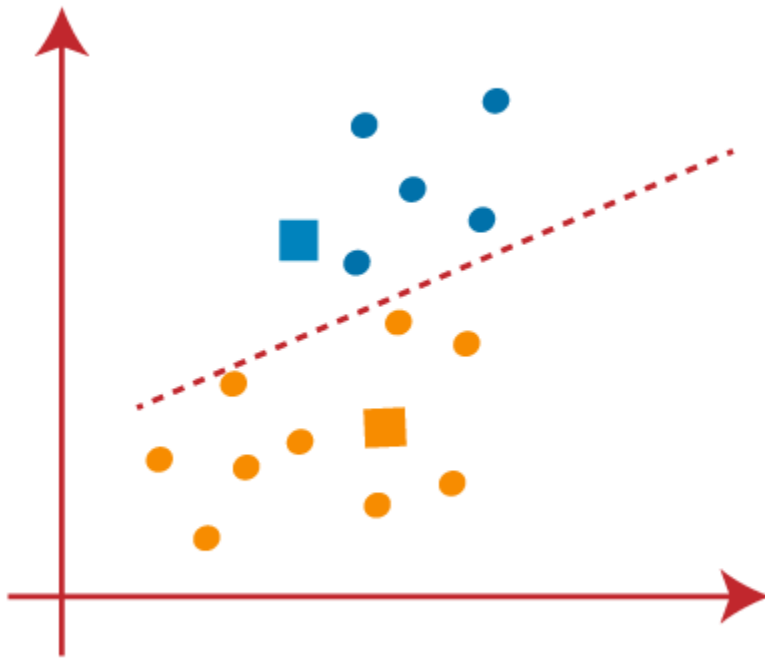
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:

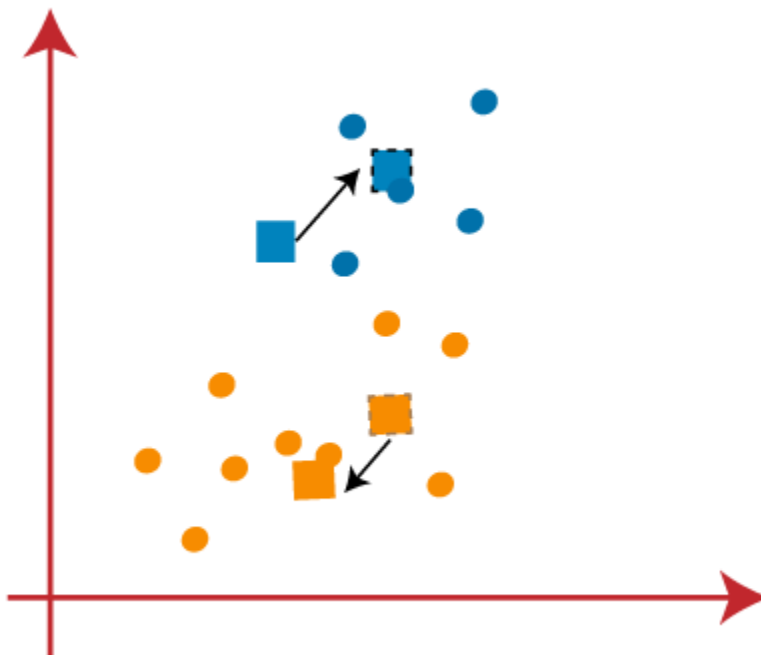


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

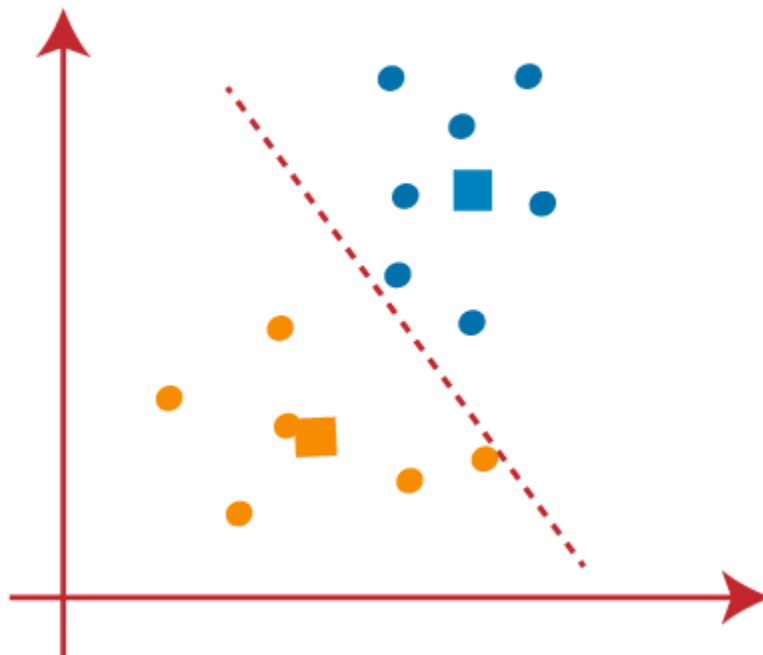


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

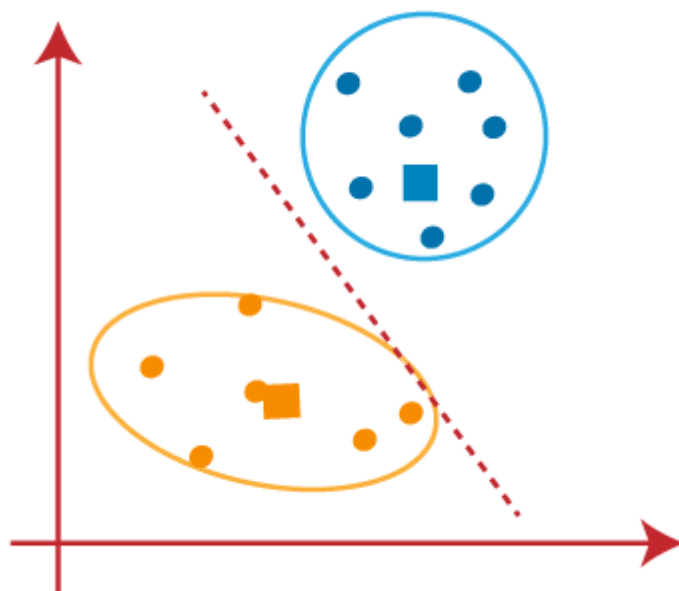
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



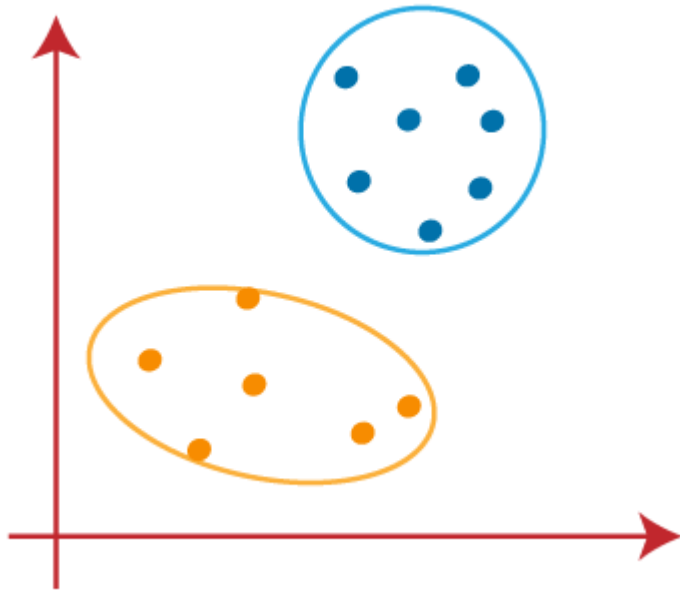
- As we got the new centroids so again will draw the median line and reassign the data points.
So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS = \sum_{P_i \text{ in Cluster1}} \text{distance}(P_i C_1)^2 + \sum_{P_i \text{ in Cluster2}} \text{distance}(P_i C_2)^2 + \sum_{P_i \text{ in Cluster3}} \text{distance}(P_i C_3)^2$$

In the above formula of WCSS,

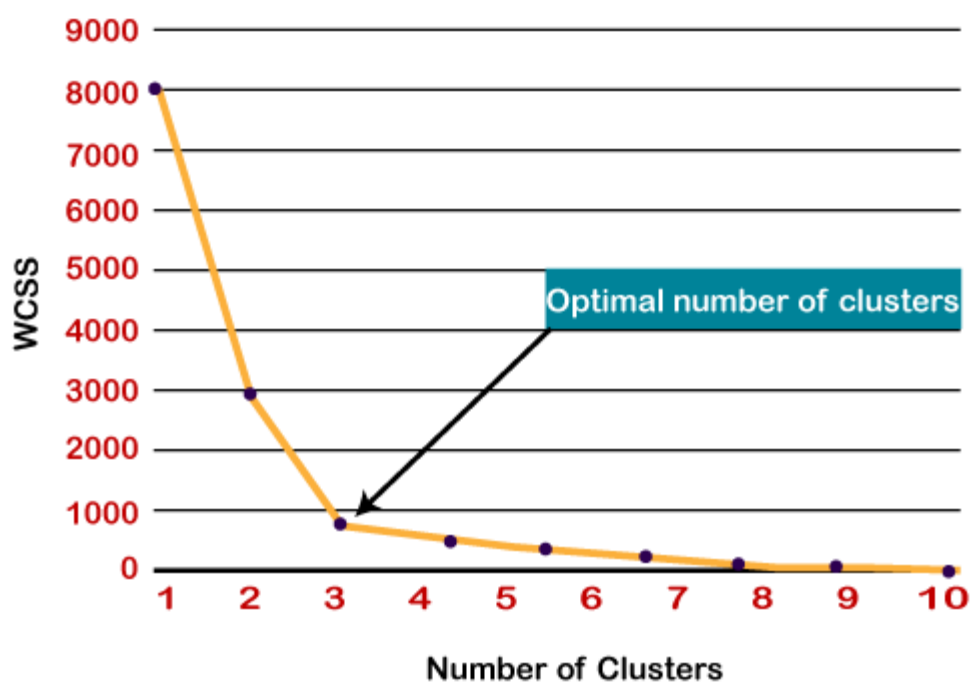
$\sum_{P_i \text{ in Cluster1}} \text{distance}(P_i C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



6.Explain Principal Component Analysis for the given sample?

ANSWER:

Principal Component Analysis

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in [machine learning](#). It is a statistical process that

converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels***. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M , and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v .

- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm

1. **Getting the dataset**
Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**
Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance. If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

5. Calculating the Eigen Values and Eigen Vectors

Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z . Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

6. Sorting the Eigen Vectors

In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P^* .

7. Calculating the new features Or Principal Components

Here we will calculate the new features. To do this, we will multiply the P^* matrix to the Z . In the resultant matrix Z^* , each observation is the linear combination of original features. Each column of the Z^* matrix is independent of each other.

8. Remove less or unimportant features from the new dataset.

The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such as **computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

7.Explain AGNES Algorithm in detail

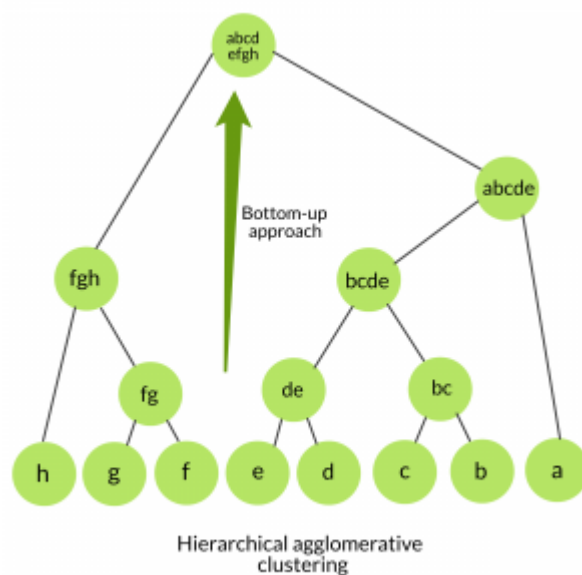
ANSWER:

In data mining and statistics, hierarchical clustering analysis is a method of cluster analysis that seeks to build a hierarchy of clusters i.e. tree-type structure based on the hierarchy.

Basically, there are two types of hierarchical cluster analysis strategies –

1. Agglomerative Clustering: Also known as bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerates pairs of clusters until all clusters have been merged into a single cluster that contains all data.

Algorithm : Merge the two clusters having minimum distance



Python implementation of the above algorithm using the scikit-learn library:

- Python3

```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

# randomly chosen dataset
X = np.array([[1, 2], [1, 4], [1, 0],
              [4, 2], [4, 4], [4, 0]])

# here we need to mention the number of clusters
# otherwise the result will be a single cluster
```

```
# containing all the data

clustering = AgglomerativeClustering(n_clusters = 2).fit(X)

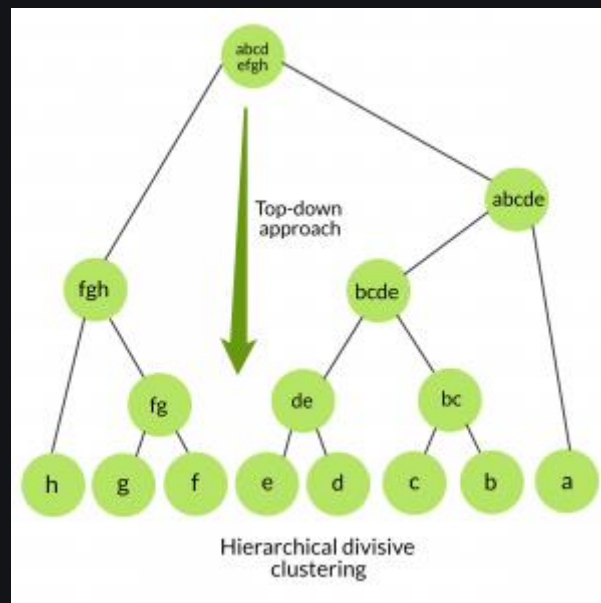
# print the class labels
print(clustering.labels_)
```

Output :

8.Explain DIANA Algorithm in detail.

ANSWER:

Divisive clustering: Also known as a top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.



Algorithm :

Hierarchical Agglomerative vs Divisive clustering –

- Divisive clustering is more *complex* as compared to agglomerative clustering, as in the case of divisive clustering we need a flat

clustering method as “subroutine” to split each cluster until we have each data having its own singleton cluster.

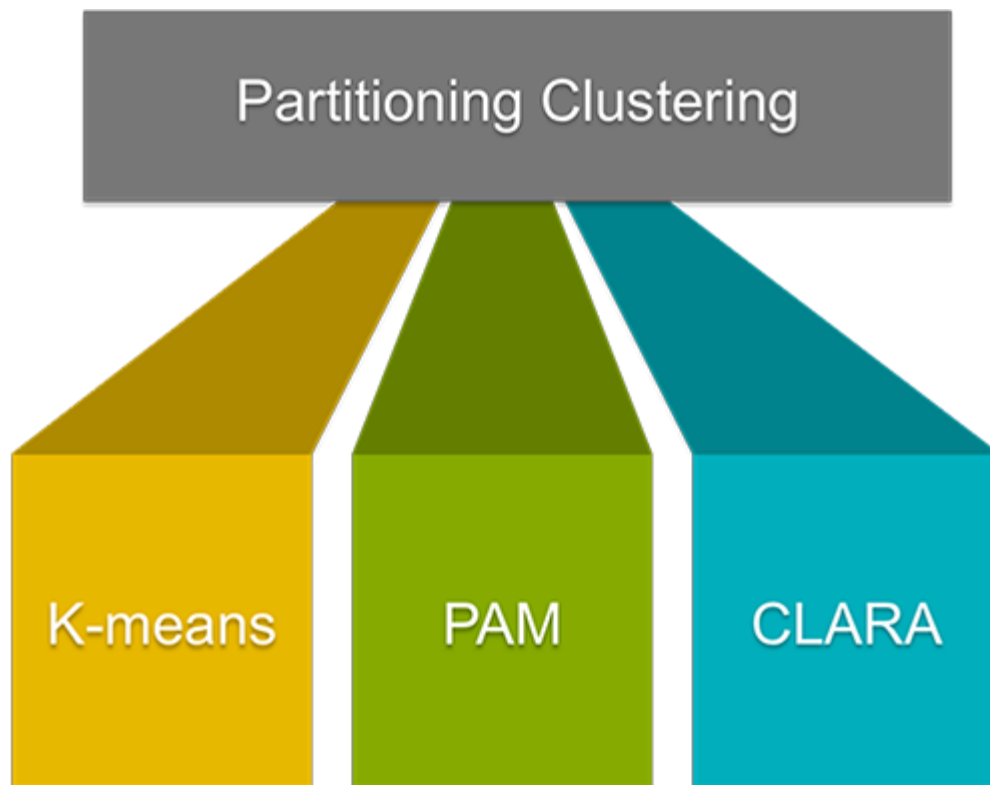
- Divisive clustering is more *efficient* if we do not generate a complete hierarchy all the way down to individual data leaves. The time complexity of a naive agglomerative clustering is $O(n^3)$ because we exhaustively scan the $N \times N$ matrix `dist_mat` for the lowest distance in each of $N-1$ iterations. Using priority queue data structure we can reduce this complexity to $O(n^2 \log n)$. By using some more optimizations it can be brought down to $O(n^2)$. Whereas for divisive clustering given a fixed number of top levels, using an efficient flat algorithm like K-Means, divisive algorithms are linear in the number of patterns and clusters.
- A divisive algorithm is also more *accurate*. Agglomerative clustering makes decisions by considering the local patterns or neighbor points without initially taking into account the global distribution of data. These early decisions cannot be undone. whereas divisive clustering takes into consideration the global distribution of data when making top-level partitioning decisions.

9.Explain Partitional Clustering Algorithm in detail

ANSWER:

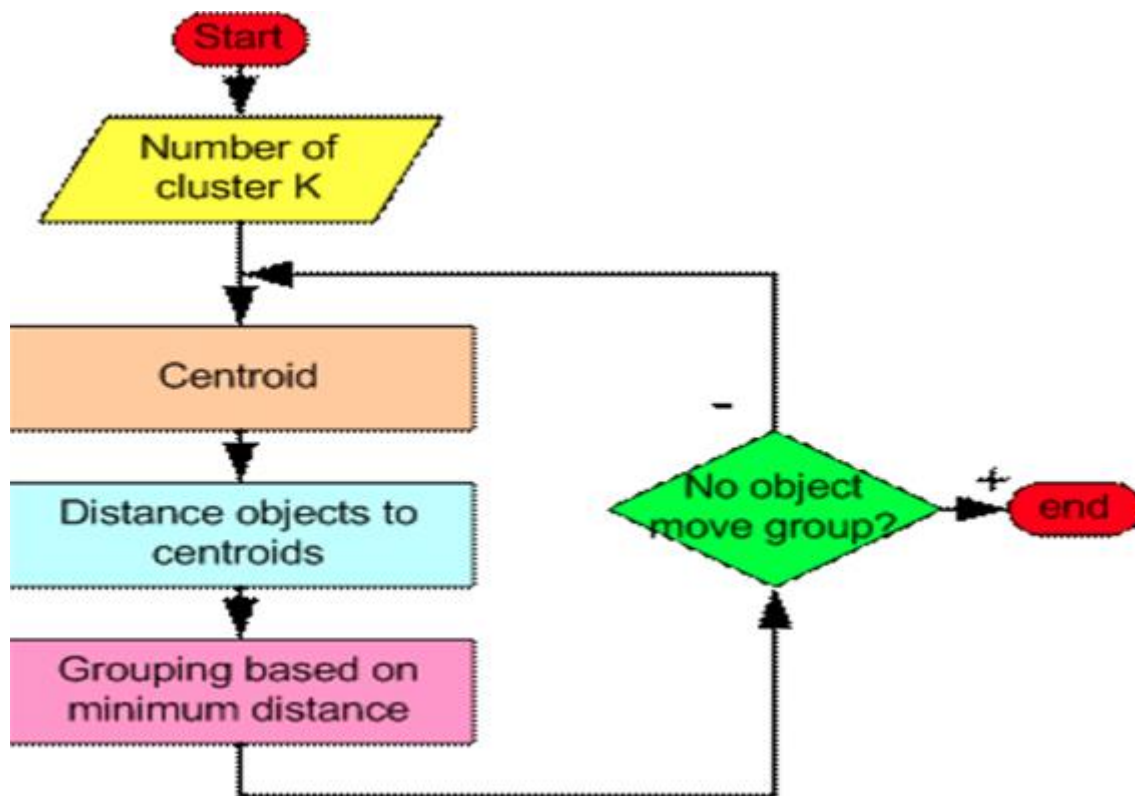
The most popular class of clustering algorithms that we have is the iterative relocation algorithms. These algorithms minimize a given clustering criterion by iteratively relocating data points between clusters until a (locally) optimal partition is attained. There are many algorithms that come under partitioning method some of the popular ones are K-Means, PAM(k-Medoid), CLARA algorithm (Clustering Large Applications) etc.

Partitioning Algorithms used in Clustering -



Types of Partitional Clustering

K-Means Algorithm (A centroid based Technique): It is one of the most commonly used algorithm for partitioning a given data set into a set of k groups (i.e. k clusters), where k represents the number of groups. It classifies objects in multiple groups (i.e., clusters), such that objects within the same cluster are as similar as possible (i.e., high *intra-class similarity*), whereas objects from different clusters are as dissimilar as possible (i.e., low *inter-class similarity*). In k-means clustering, each cluster is represented by its center (i.e., *centroid*) which corresponds to the mean of points assigned to the cluster. The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized.



Process for K-means Algorithm

Steps involved in K-Means Clustering :

1. The first step when using k-means clustering is to indicate the number of clusters (k) that will be generated in the final solution.
2. The algorithm starts by randomly selecting k objects from the data set to serve as the initial centers for the clusters. The selected objects are also known as cluster means or centroids.
3. Next, each of the remaining objects is assigned to its closest centroid, where closest is defined using the [Euclidean distance](#) between the object and the cluster mean. This step is called “cluster assignment step”.
4. After the assignment step, the algorithm computes the new mean value of each cluster. The term cluster “centroid

update” is used to design this step. Now that the centers have been recalculated, every observation is checked again to see if it might be closer to a different cluster. All the objects are reassigned again using the updated cluster means.

5. The cluster assignment and centroid update steps are iteratively repeated until the cluster assignments stop changing (i.e until *convergence* is achieved). That is, the clusters formed in the current iteration are the same as those obtained in the previous iteration.

Step 1 ->

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 2 ->

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 3 ->

Step	Cluster 1		Cluster 2	
	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Step 4 ->

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

Step 5 ->

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Step 6 ->

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

Example for K-Means Clustering

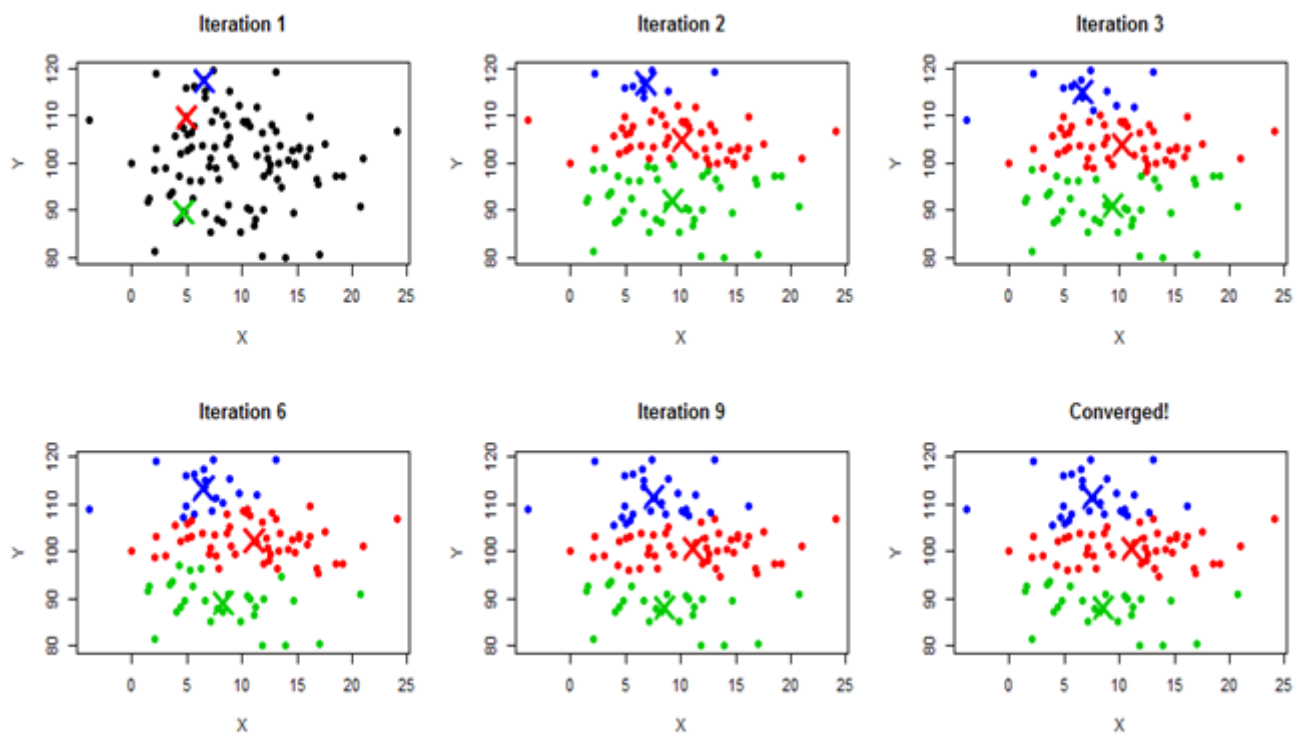
Step 5 ->

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Step 6 ->

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

Example for K-Means Clustering

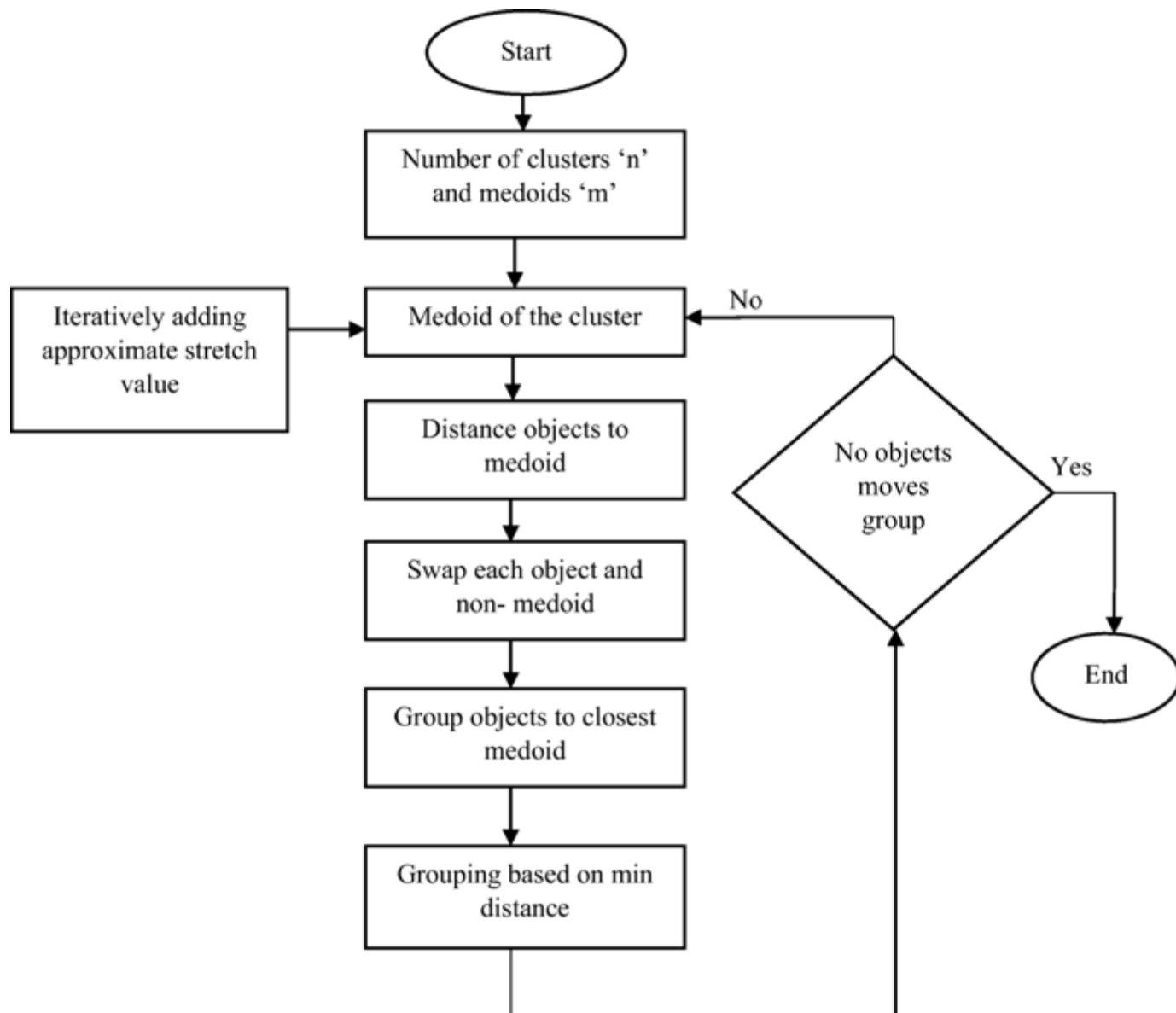


Plotting of K-means Clustering

K-Medoids Algorithm (Partitioning Around Medoid) :

1. A medoid can be defined as the point in the cluster, whose similarities with all the other points in the cluster is maximum.

2. In k-medoids clustering, each cluster is represented by one of the data point in the cluster. These points are named cluster medoids. The term medoid refers to an object within a cluster for which average dissimilarity between it and all the other the members of the cluster is minimal. It corresponds to the most centrally located point in the cluster.
3. These objects (one per cluster) can be considered as a representative example of the members of that cluster which may be useful in some situations. Recall that, in k-means clustering, the center of a given cluster is calculated as the mean value of all the data points in the cluster.
4. K-medoid is a robust alternative to k-means clustering. This means that, the algorithm is less sensitive to noise and outliers, compared to k-means, because it uses medoids as cluster centers instead of means (used in k-means).



Steps involved in K-Medoid Clustering

Steps involved in K-Medoids Clustering :

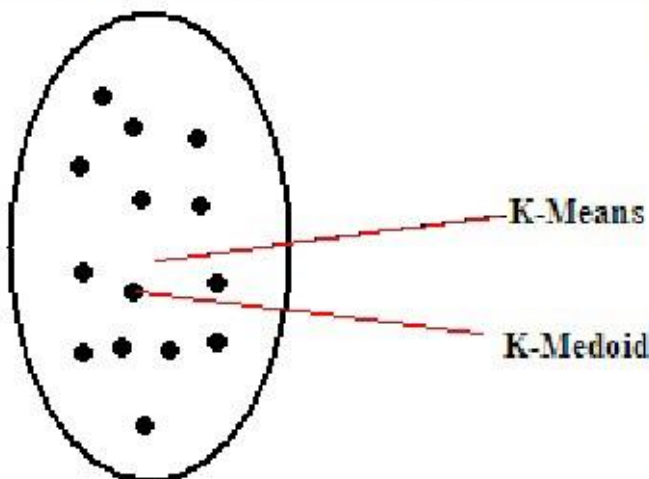
1. The PAM algorithm is based on the search for k representative objects or medoids among the observations of the data set.
2. After finding a set of k medoids, clusters are constructed by assigning each observation to the nearest medoid.
3. Next, each selected medoid m and each non-medoid data point are swapped and the objective function is computed.

The objective function corresponds to the sum of the dissimilarities of all objects to their nearest medoid.

4. The SWAP step attempts to improve the quality of the clustering by exchanging selected objects (medoids) and non-selected objects. If the objective function can be reduced by interchanging a selected object with an unselected object, then the swap is carried out. This is continued until the objective function can no longer be decreased. The goal is to find k representative objects which minimize the sum of the dissimilarities of the observations to their closest representative object.

Difference between K-Means & K-Medoids Clustering -

K-Means & K-medoid



Cluster ==>

2 5 6 9 13

K-mean's Centroid = 7

K-medoid's Centroid = 6

K-means clustering use the exact center of a cluster (means or the center of gravity) while K-medoid uses the most centrally located object in a cluster (medoid).

K-medoid is less sensitive to outliers Compared to K-means.

K value (number of clusters) has to be determined a-priori.

1. *K*-means attempts to minimize the total [squared error](#), while *k*-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the *k*-means algorithm, *k*-medoids chooses data points as centers ([medoids](#) or exemplars).
2. *K*-medoid is more robust to noise and outliers as compared to *K*-means because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distances.
3. *K*-medoids has the advantage of working on **distances other than numerical** and lends itself well to analyse mixed-type data that include both numerical and categorical features.

k-means	k-medoids
Complexity is $O(ikn)$	Complexity is $O(ik(n-k)^2)$
More efficient	Comparatively less efficient
Sensitive to outliers	Not Sensitive to outliers
Convex shape is required	Convex shape is not must
Number of clusters need to be specified in advance	Number of clusters need to be specified in advance
Efficient for separated clusters	Efficient for separated clusters and small data sets

10. Define Dendograms. can we prune Dendograms.

ANSWER:

The implementation of the Agglomerative Clustering algorithm accepts the number of desired clusters. There are several ways to find the optimal number of clusters such that the population is divided into k clusters in a way that:

Points in the same cluster are closer to each other.

Points in the different clusters are far apart.

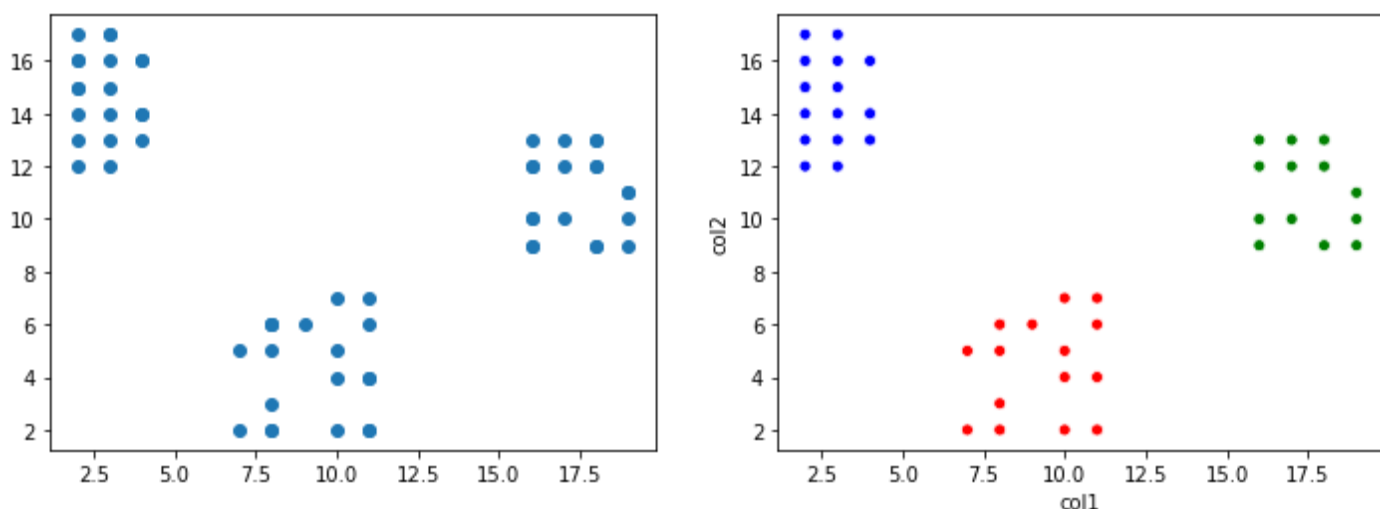
By observing the dendrograms, one can find the desired number of clusters.

Dendrograms are a diagrammatic representation of the hierarchical relationship between the data-points. It illustrates the arrangement of the clusters produced by the corresponding analyses and is used to observe the output of hierarchical (agglomerative) clustering.

Implementation of Dendrograms:

(Code by Author)

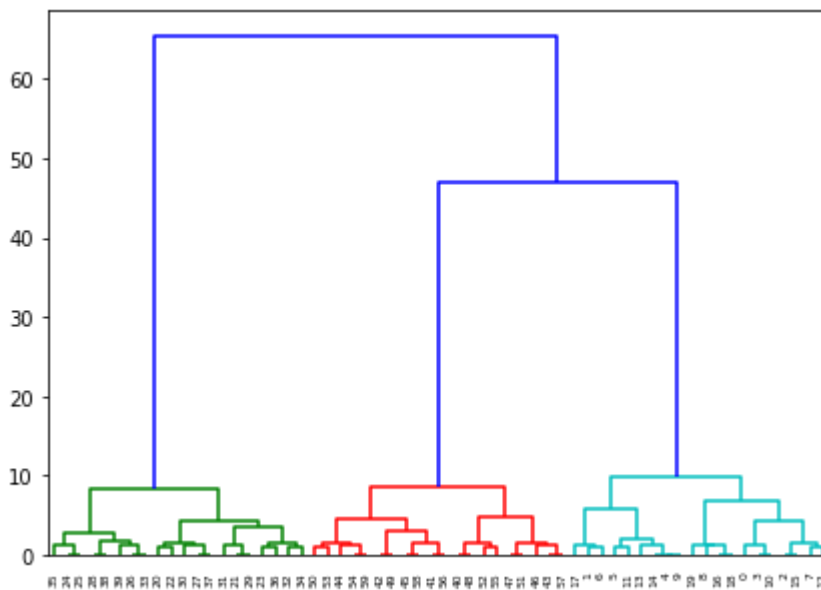
Download the sample 2-dimension dataset from [here](#).



(Image by Author), Left Image: Visualize the sample dataset, Right Image: Visualize 3 cluster for the sample dataset

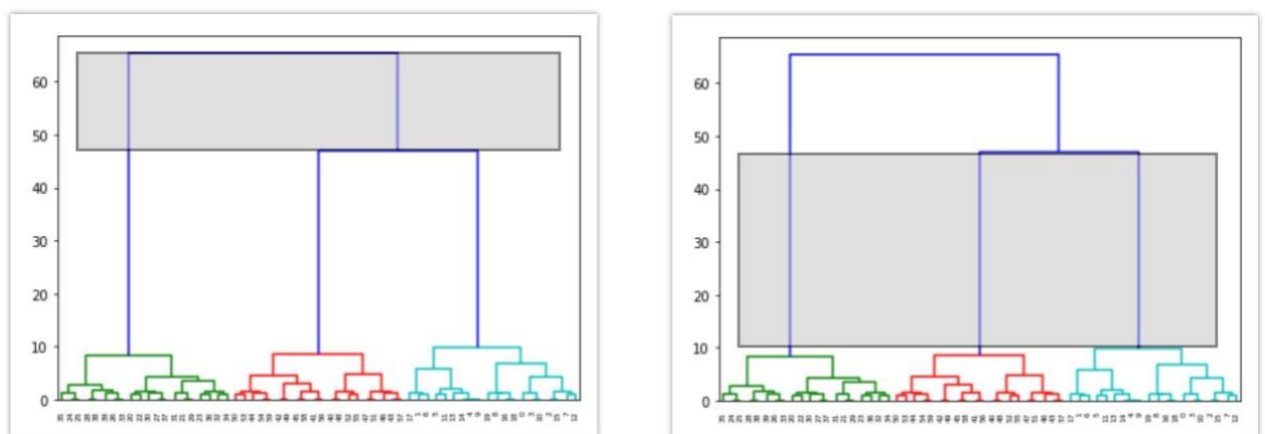
For the above sample dataset, it is observed that the optimal number of clusters would be 3. But for high dimension dataset where visualization of the dataset is not possible dendrograms play an important role to find the optimal number of clusters.

How to find the optimal number of clusters by observing the dendrograms:



(Image by Author), Dendrogram for the above sample dataset

From the above dendrogram plot, find a horizontal rectangle with max-height that does not cross any horizontal vertical dendrogram line.



(Image by Author), Left: Separating into 2 clusters, Right: Separating into 3 clusters

The portion in the dendrogram in which rectangle having the max-height can be cut, and the optimal number of clusters will be 3 as observed in the right part of the above image. Max height rectangle is

chosen because it represents the maximum Euclidean distance between the optimal number of clusters

11.Explain K-Mode Clustering Algorithm in detail

ANSWER:

This article was published as a part of the [Data Science Blogathon](#)



≡ KMODES CLUSTERING ≡

Introduction:

[Clustering](#) is an unsupervised learning method whose task is to divide the population or data points into a number of groups, such that data points in a group are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects based on similarity and dissimilarity between them.

KModes clustering is one of the unsupervised Machine Learning algorithms that is used to cluster **categorical variables**.

You might be wondering, why KModes when we already have KMeans.

KMeans uses mathematical measures (distance) to cluster continuous data. The lesser the distance, the more similar our data points are. Centroids are updated by Means. But for categorical data points, we cannot calculate the distance. So we go for KModes algorithm. It uses the dissimilarities(total mismatches) between the data points. The lesser the dissimilarities the more similar our data points are. It uses Modes instead of means.


How does the KModes algorithm work?

Unlike [Hierarchical clustering](#) methods, we need to upfront specify the K.

1. Pick K observations at random and use them as leaders/clusters
2. Calculate the dissimilarities and assign each observation to its closest cluster
3. Define new modes for the clusters
4. Repeat 2–3 steps until there are is no re-assignment required

I hope you got the basic idea of the KModes algorithm by now. So let us quickly take an example to illustrate the working step by step.

Example: Imagine we have a dataset that has the information about hair color, eye color, and skin color of persons. We aim to group them based on the available information(maybe we want to suggest some styling ideas)

Hair color, eye color, and skin color are all categorical variables. Below  is how our dataset looks like.

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Image of our data

Alright, we have the sample data now. Let us proceed by defining the number of clusters(K)=3

Step 1: Pick K observations at random and use them as leaders/clusters

I am choosing P1, P7, P8 as leaders/clusters

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair
person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Leaders and

Observations

Step 2: Calculate the dissimilarities(no. of mismatches) and assign each observation to its closest cluster

Iteratively compare the cluster data points to each of the observations. Similar data points give 0, dissimilar data points give 1.

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair
person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing leader/Cluster P1 to the observation P1 gives 0 dissimilarities.

Leaders			
P1	blonde	amber	fair
P7	red	green	fair
P8	black	hazel	fair
person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing leader/cluster P1 to the observation P2 gives $3(1+1+1)$ dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix as shown below and assign the observations to their closest cluster(cluster that has the least dissimilarity)

	Cluster 1 (P1)	Cluster 2 (P7)	Cluster 3 (P8)	Cluster
P1	0 ✓	2	2	Cluster 1
P2	3 ✓	3	3	Cluster 1
P3	3	1 ✓	3	Cluster 2
P4	3	3	1 ✓	Cluster 3
P5	1 ✓	2	2	Cluster 1
P6	3	3	2 ✓	Cluster 3
P7	2	0 ✓	2	Cluster 2
P8	2	2	0 ✓	Cluster 3

Dissimilarity matrix (Image by Author)

After step 2, the observations P1, P2, P5 are assigned to cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to cluster 3.

Note: If all the clusters have the same dissimilarity with an observation, assign to any cluster randomly. In our case, the observation P2 has 3 dissimilarities with all the leaders. I randomly assigned it to Cluster 1.

Step 3: Define new modes for the clusters

Mode is simply the **most observed value**.

Mark the observations according to the cluster they belong to. Observations of Cluster 1 are marked in Yellow, Cluster 2 are marked in Brick red, and Cluster 3 are marked in Purple.

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Looking for Modes (Image by author)

Considering one cluster at a time, for each feature, look for the Mode and update the new leaders.

Explanation: Cluster 1 observations(P1, P2, P5) has brunette as the most observed hair color, amber as the most observed eye color, and fair as the most observed skin color.

Note: If you observe the same occurrence of values, take the mode randomly. In our case, the observations of Cluster 3(P3, P7) have one occurrence of brown, fair skin color. I randomly chose brown as the mode.

Below are our new leaders after the update.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

Obtained new leaders

Repeat steps 2-4

After obtaining the new leaders, again calculate the dissimilarities between the observations and the newly obtained leaders.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown
person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing Cluster 1 to the observation P1 gives 1 dissimilarity.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown
person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Comparing Cluster 1 to the observation P2 gives 2 dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix. Assign each observation to its closest cluster.

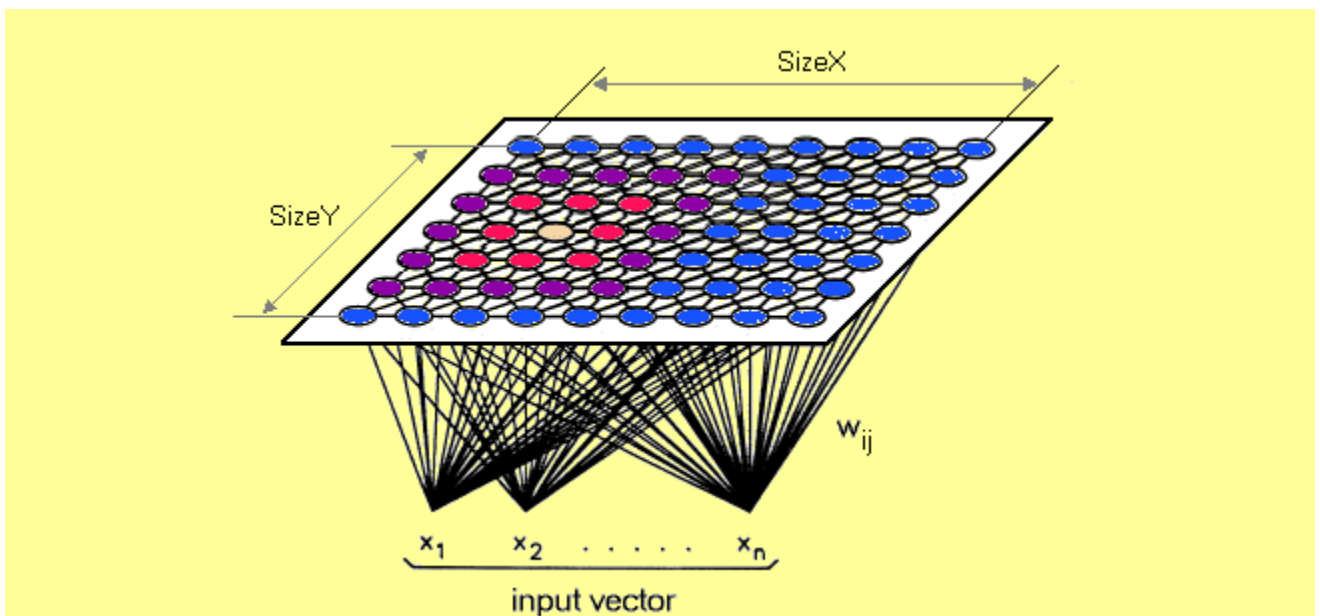
	Cluster 1	Cluster 2	Cluster 3	Cluster
P1	1 ✓	2	3	Cluster 1
P2	2 ✓	3	2	Cluster 1
P3	3	1 ✓	2	Cluster 2
P4	3	3	0 ✓	Cluster 3
P5	0 ✓	2	3	Cluster 1
P6	3	3	1 ✓	Cluster 3
P7	2	0 ✓	3	Cluster 2
P8	2	2	1 ✓	Cluster 3

The observations P1, P2, P5 are assigned to Cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to Cluster 3.

12. Explain about Self Organizing Maps (SOM)

ANSWER:

A **self-organizing map (SOM)** is a type of [artificial neural network](#) (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a **map**, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space.



Dimensionality reduction in SOM

SOM was introduced by Finnish professor Teuvo Kohonen in the 1980s and is sometimes called a **Kohonen map**.

What really happens in SOM ?

Each data point in the data set recognizes themselves by competing for representation. SOM mapping steps starts from initializing the weight vectors. From there a sample vector is selected randomly and the map of weight vectors is searched to find which weight best represents that sample. Each weight vector has neighboring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. The neighbors of that weight are also rewarded by being able to become more like the chosen sample vector. This allows the map to grow and form different shapes. Most generally, they form square/rectangular/hexagonal/L shapes in 2D feature space.

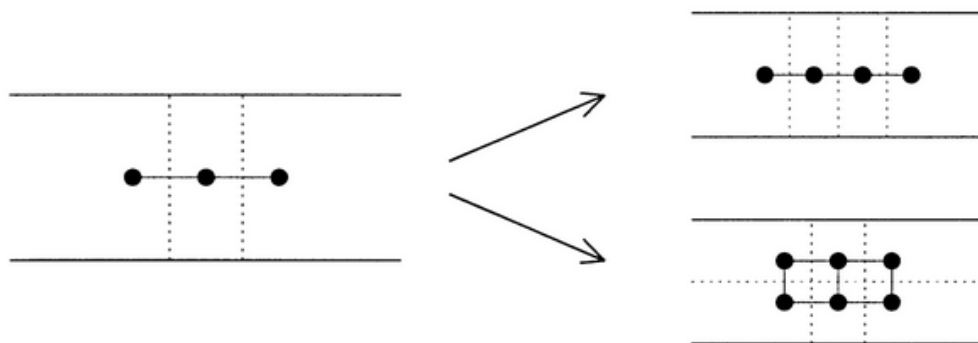


Fig. 1. Illustration of the basic decision to be made during the growth procedure: an output space lattice can either grow in an existing direction or it can open a new direction.

Referece: Applications of the growing self-organizing map, Th. Villmann, H.-U. Bauer, May 1998

The Algorithm:

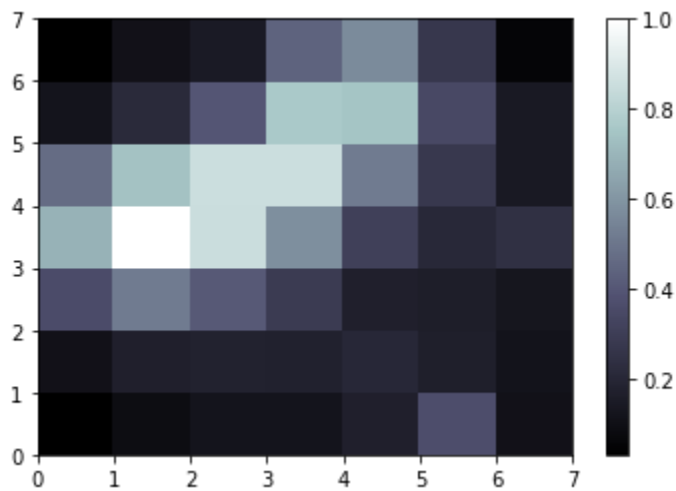
1. Each node's weights are initialized.
2. A vector is chosen at random from the set of training data.

3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the **Best Matching Unit** (BMU).
4. Then the neighbourhood of the BMU is calculated. The amount of neighbors decreases over time.
5. The winning weight is rewarded with becoming more like the sample vector. The neighbors also become more like the sample vector. The closer a node is to the BMU, the more its weights get altered and the farther away the neighbor is from the BMU, the less it learns.
6. Repeat step 2 for N iterations.

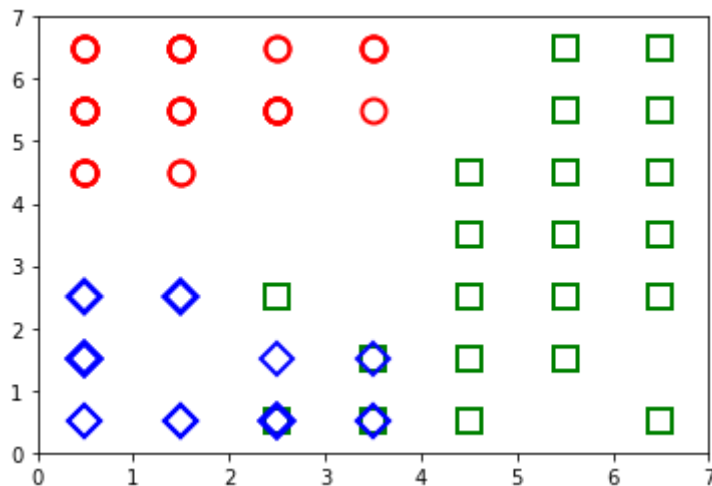
Best Matching Unit is a technique which calculates the distance from each weight to the sample vector, by running through all weight vectors. The weight with the shortest distance is the winner. There are numerous ways to determine the distance, however, the most commonly used method is the [*Euclidean Distance*](#), *and that's what is used in the following implementation.*

Implementation:

Coming to implementation part, there are various Python libraries (minisom, sompy) out there which you could directly use to implement SOM. You could also write your own implementation of it. I've implemented both on the [*iris dataset*](#). Here are the results:



In simpler terms, the darker parts represent clusters, while the lighter parts represent the division of the clusters.



(Red Circles , Iris-setosa), (Green, Iris-versicolor) , (Blue, Iris-virginica)

Inference:

If the average distance is high, then the surrounding weights are very different and a light color is assigned to the location of the weight. If the average distance is low, a darker color is assigned. The resulting maps show that the concentration of different clusters of species are more predominant in three zones. First figure tells us only about where the density of species is greater (darker regions) or less (lighter

regions). The second visualisation tells us how they are specifically clustered.

Cons of Kohonen Maps:

1. It does not build a generative model for the data, i.e, the model does not understand how data is created.
2. It does not behave so gently when using categorical data, even worse for mixed types data.
3. The time for preparing model is slow, hard to train against slowly evolving data

13. What do you mean by mixture Densities? Explain the need of it in Clustering.

ANSWER:

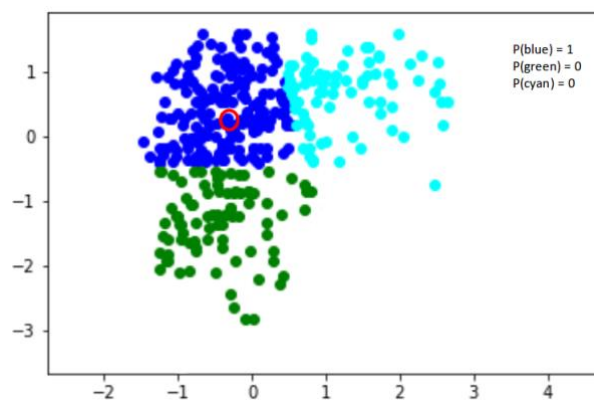
Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together.

Let's say we have three Gaussian distributions (more on that in the next section) – GD1, GD2, and GD3. These have a certain mean (μ_1 , μ_2 , μ_3) and variance (σ_1 , σ_2 , σ_3) value respectively. For a given set of data points, our GMM would identify the probability of each data point belonging to each of these distributions.

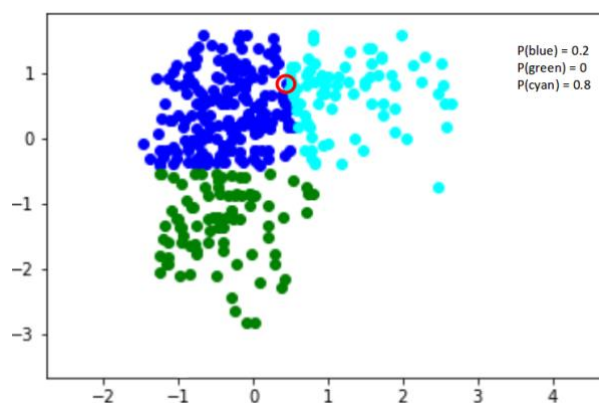
Wait, [probability](#)?

You read that right! **Gaussian Mixture Models** are probabilistic models and use the **soft clustering approach** for distributing the points in different **clusters**. I'll take another example that will make it easier to understand.

Here, we have three clusters that are denoted by three colors – Blue, Green, and Cyan. Let's take the data point highlighted in red. The probability of this point being a part of the blue cluster is 1, while the probability of it being a part of the green or cyan clusters is 0.



Now, consider another point – somewhere in between the blue and cyan (highlighted in the below figure). The probability that this point is a part of cluster green is 0, right? And the probability that this belongs to blue and cyan is 0.2 and 0.8 respectively.

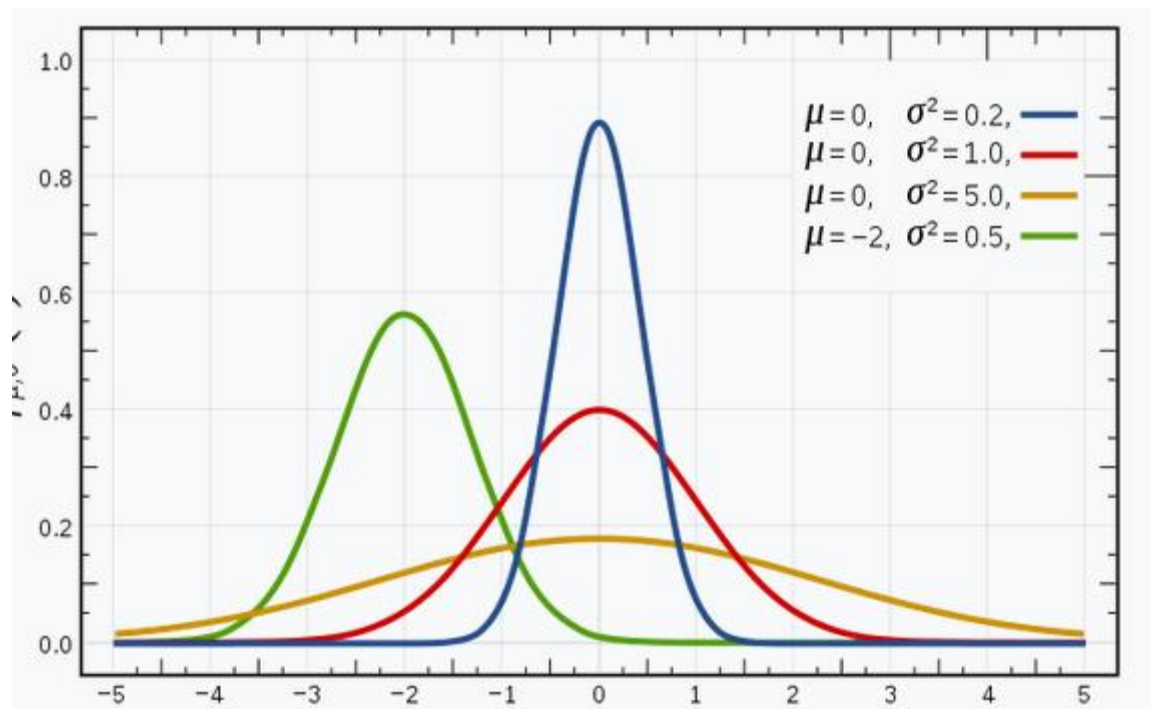


Gaussian Mixture Models use the soft clustering technique for assigning data points to Gaussian distributions. I'm sure you're wondering what these distributions are so let me explain that in the next section.

The Gaussian Distribution

I'm sure you're familiar with Gaussian Distributions (or the Normal Distribution). It has a bell-shaped curve, with the data points symmetrically distributed around the mean value.

The below image has a few Gaussian distributions with a difference in mean (μ) and variance (σ^2). Remember that the higher the σ value more would be the spread:



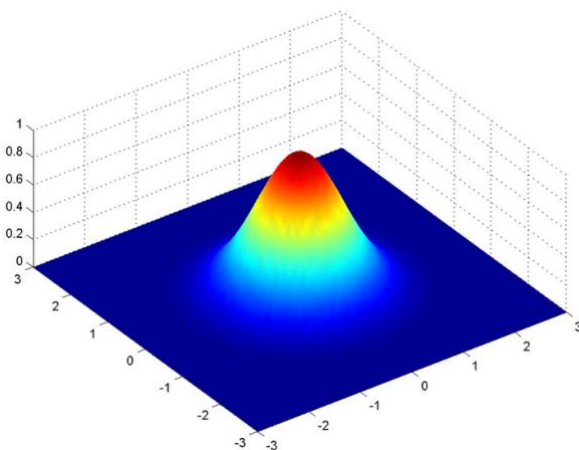
Source: Wikipedia

In a one dimensional space, the [probability density function](#) of a Gaussian distribution is given by:

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

where μ is the mean and σ^2 is the variance.

But this would only be true for a single variable. In the case of two variables, instead of a 2D bell-shaped curve, we will have a 3D bell curve as shown below:



The probability density function would be given by:

$$f(x | \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp \left[-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right]$$

where x is the input vector, μ is the 2D mean vector, and Σ is the 2×2 covariance matrix. The covariance would now define the shape of this curve. We can generalize the same for d -dimensions.

Thus, this multivariate Gaussian model would have x and μ as vectors of length d , and Σ would be a $d \times d$ covariance matrix.

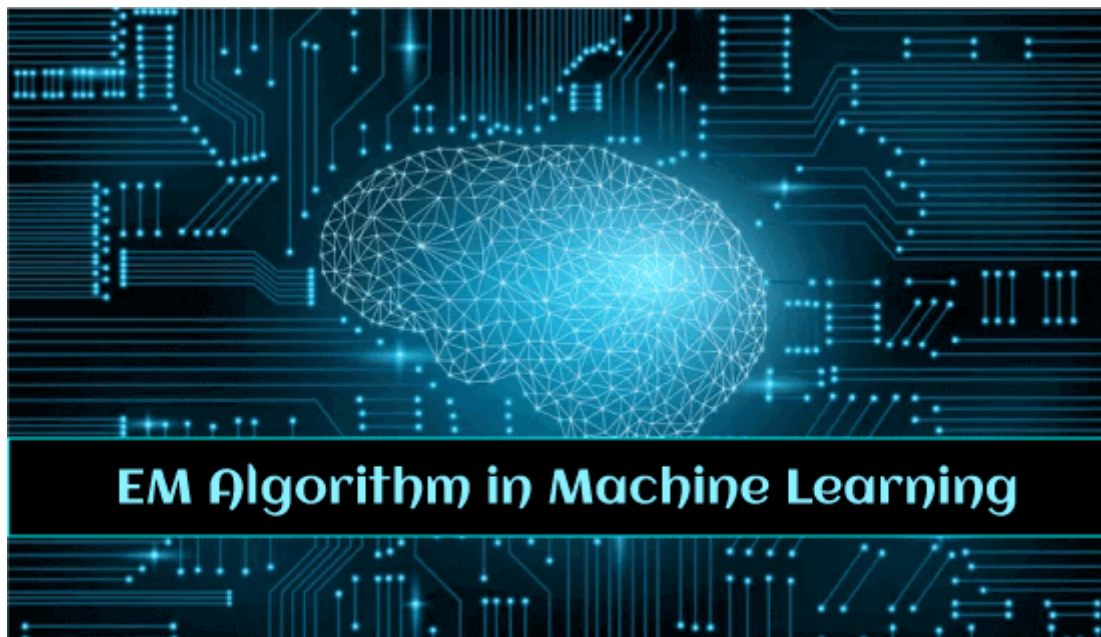
Hence, for a dataset with d features, we would have a mixture of k Gaussian distributions (where k is equivalent to the number of clusters), each having a certain mean vector and variance matrix. But wait – how is the mean and variance value for each Gaussian assigned?

These values are determined using a technique called Expectation-Maximization (EM). We need to understand this technique before we dive deeper into the working of Gaussian Mixture Models.

14. Describe about Expectation-Maximization Algorithm in detail

ANSWER:

The EM algorithm is considered a latent variable model to find the local maximum likelihood parameters of a statistical model, proposed by Arthur Dempster, Nan Laird, and Donald Rubin in 1977. The EM (Expectation-Maximization) algorithm is one of the most commonly used terms in machine learning to obtain maximum likelihood estimates of variables that are sometimes observable and sometimes not. However, it is also applicable to unobserved data or sometimes called latent. It has various real-world applications in statistics, including obtaining the **mode of the posterior marginal distribution of parameters in machine learning and data mining applications.**



In most real-life applications of machine learning, it is found that several relevant learning features are available, but very few of them are observable, and the rest are unobservable. If the variables are observable, then it can predict the value using instances. On the other hand, the variables which are latent or directly not observable, for such variables Expectation-Maximization (EM) algorithm plays a vital role to predict the value with the condition that the general form of probability distribution governing those latent variables is known to us. In this topic, we will discuss a basic introduction to the EM algorithm, a flow chart of the EM algorithm, its applications, advantages, and disadvantages of EM algorithm, etc.

What is an EM algorithm?

The Expectation-Maximization (EM) algorithm is defined as the combination of various unsupervised machine learning algorithms, which is used to determine the **local maximum likelihood estimates (MLE)** or **maximum a posteriori estimates (MAP)** for unobservable variables in statistical models. Further, it is a technique to find maximum likelihood estimation when the latent variables are present. It is also referred to as the **latent variable model**.

A latent variable model consists of both observable and unobservable variables where observable can be predicted while unobserved are inferred from the observed variable. These unobservable variables are known as latent variables.

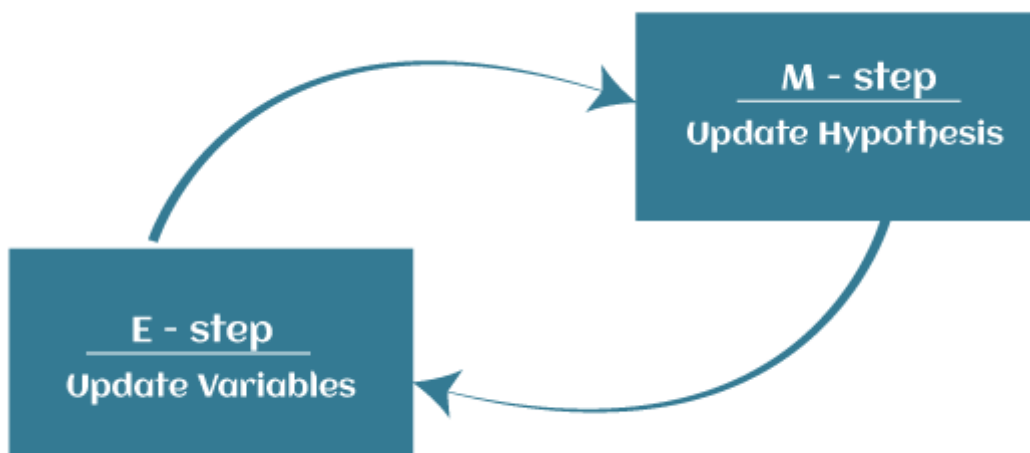
Key Points:

- It is known as the latent variable model to determine MLE and MAP parameters for latent variables.

- It is used to predict values of parameters in instances where data is missing or unobservable for learning, and this is done until convergence of the values occurs.

EM Algorithm

The EM algorithm is the combination of various unsupervised ML algorithms, such as the **k-means clustering algorithm**. Being an iterative approach, it consists of two modes. In the first mode, we estimate the missing or latent variables. Hence it is referred to as the **Expectation/estimation step (E-step)**. Further, the other mode is used to optimize the parameters of the models so that it can explain the data more clearly. The second mode is known as the **maximization-step or M-step**.



- **Expectation step (E - step):** It involves the estimation (guess) of all missing values in the dataset so that after completing this step, there should not be any missing value.
- **Maximization step (M - step):** This step involves the use of estimated data in the E-step and updating the parameters.
- **Repeat E-step and M-step** until the convergence of the values occurs.

The primary goal of the EM algorithm is to use the available observed data of the dataset to estimate the missing data of the latent variables and then use that data to update the values of the parameters in the M-step.

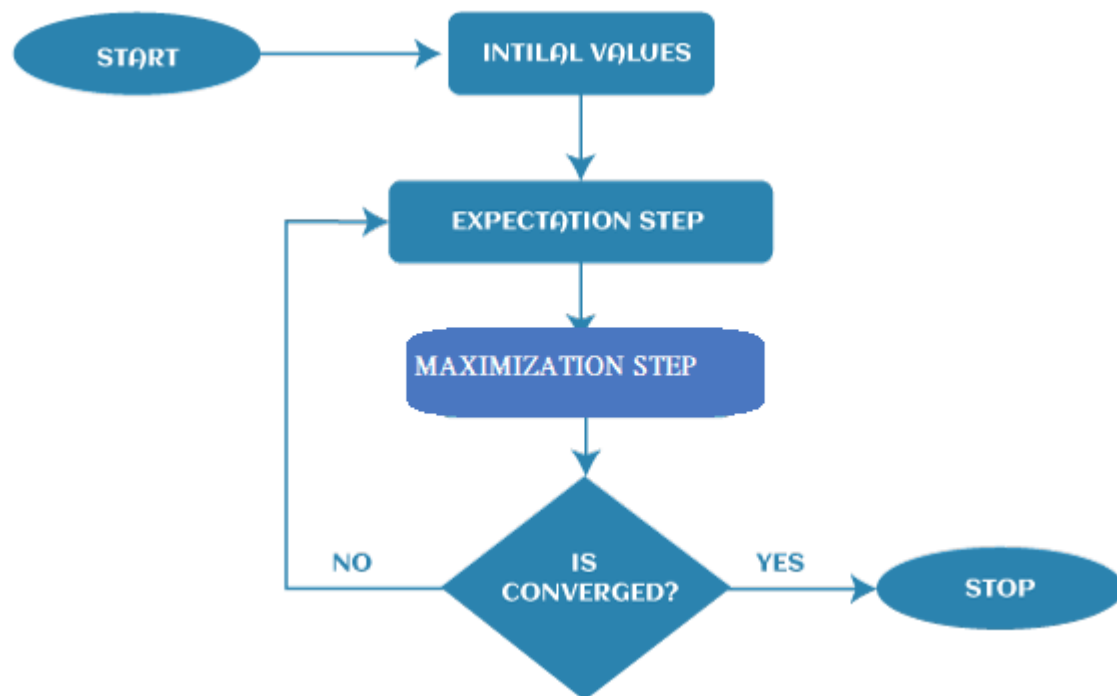
What is Convergence in the EM algorithm?

Convergence is defined as the specific situation in probability based on intuition, e.g., if there are two random variables that have very less difference in their probability,

then they are known as converged. In other words, whenever the values of given variables are matched with each other, it is called convergence.

Steps in EM Algorithm

The EM algorithm is completed mainly in 4 steps, which include **Initialization Step**, **Expectation Step**, **Maximization Step**, and **convergence Step**. These steps are explained as follows:



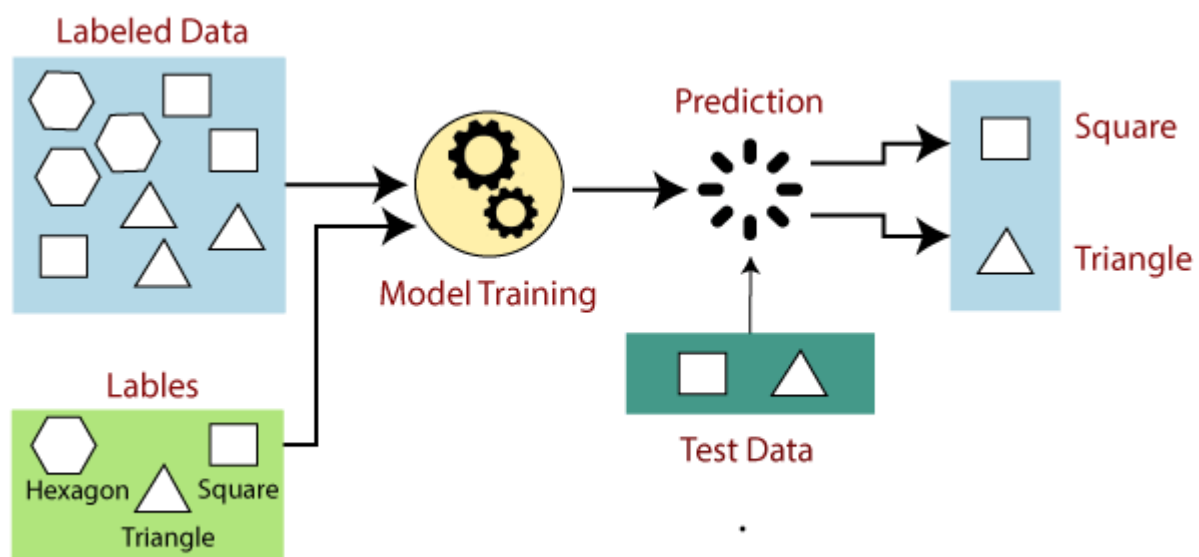
- **1st Step:** The very first step is to initialize the parameter values. Further, the system is provided with incomplete observed data with the assumption that data is obtained from a specific model.
- **2nd Step:** This step is known as Expectation or E-Step, which is used to estimate or guess the values of the missing or incomplete data using the observed data. Further, E-step primarily updates the variables.
- **3rd Step:** This step is known as Maximization or M-step, where we use complete data obtained from the 2nd step to update the parameter values. Further, M-step primarily updates the hypothesis.
- **4th step:** The last step is to check if the values of latent variables are converging or not. If it gets "yes", then stop the process; else, repeat the process from step 2 until the convergence occurs.

15. Explain in detail about Supervised Learning and Clustering

ANSWER:

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

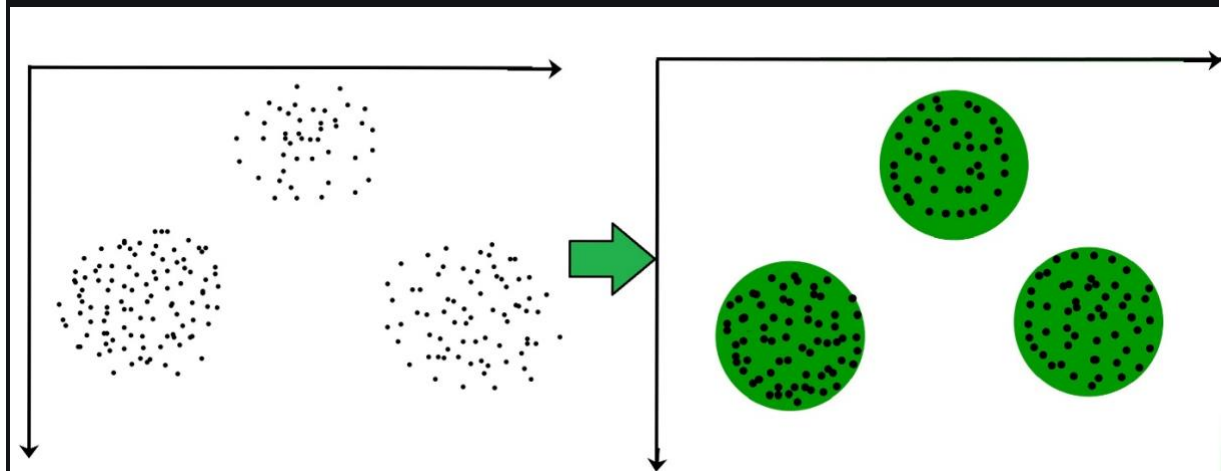
- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset.**
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate

CLUSTERING:

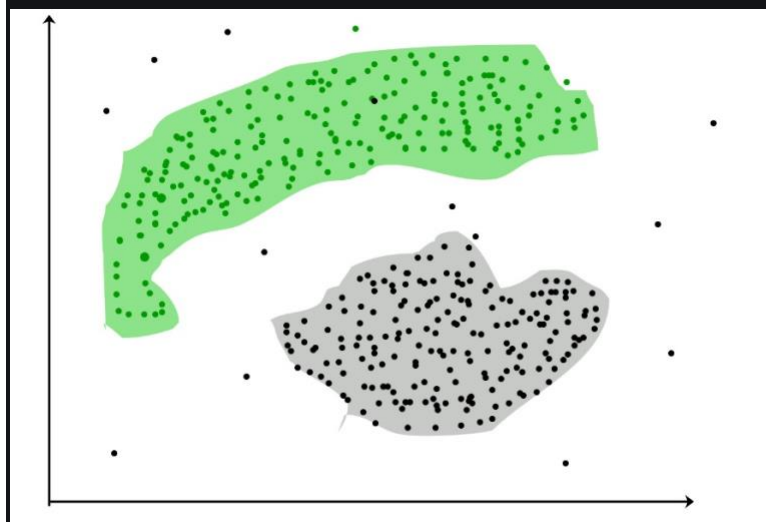
It is basically a type of [unsupervised learning method](#). An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



It is not necessary for clusters to be spherical. Such as :



DBSCAN: Density-based Spatial Clustering of Applications with Noise

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for the calculation of the outliers.

Why

Clustering?

Clustering is very much important as it determines the intrinsic grouping among the unlabelled data present. There are no criteria for good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions that

constitute the similarity of points and each assumption make different and equally valid clusters.

Clustering Methods :

- **Density-Based Methods:** These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters. Example *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)*, *OPTICS (Ordering Points to Identify Clustering Structure)*, etc.
- **Hierarchical Based Methods:** The clusters formed in this method form a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category
 - **Agglomerative** (bottom-up approach)
 - **Divisive** (top-down approach)

examples *CURE (Clustering Using Representatives)*, *BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies)*, etc.

- **Partitioning Methods:** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example *K-means*, *CLARANS (Clustering Large Applications based upon Randomized Search)*, etc.
- **Grid-based Methods:** In this method, the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operations done on these grids are fast and independent of the number of data objects example *STING (Statistical Information Grid)*, *wave cluster*, *CLIQUE (CLustering In Quest)*, etc.

16 How do you choose the number of clusters to perform Clustering ?

ANSWER:

When we use clustering algorithms, choosing the number of clusters is always a challenging task. While there are some existing approaches that can help with this task, they are usually being used separately in order to take the decision. Moreover, you will probably also need to manually choose the final number of clusters, based on the approaches' outcome.

Here I suggest an approach that considers the common trade-off between the inner-distances and the number of clusters- and automatically choose the number of clusters.

Existing Approaches

In this section you can find the two most common approaches for choosing the number of clusters. Each has its own advantages and limitations.

Silhouette Analysis

This is a well-known approach that also provides very cool visualizations. It allows us to interpret and validate the consistency within the clusters. The full Python implementation and explanation can be found in this [scikit-learn page](#):

Selecting the number of clusters with silhouette analysis on KMeans clustering - scikit-learn...

Silhouette analysis can be used to study the separation distance between the resulting clusters. The silhouette plot...

scikit-learn.org

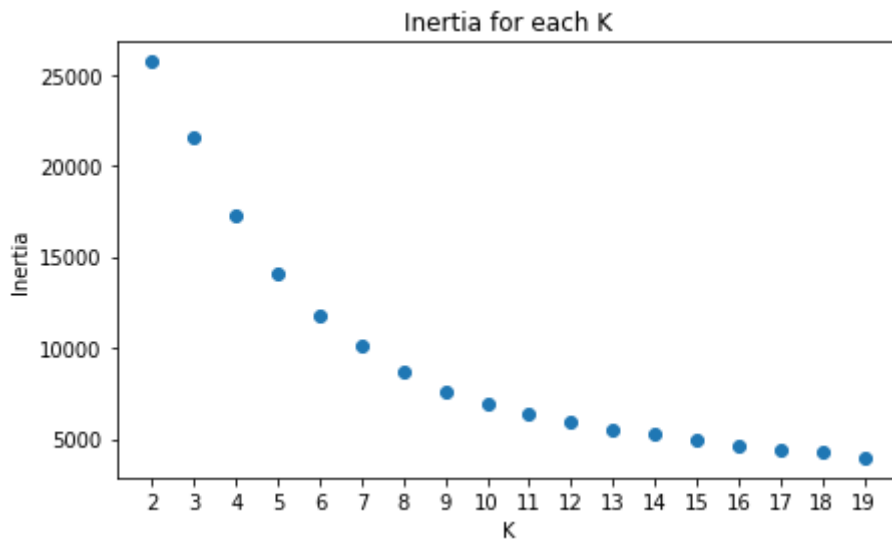
This scikit-learn Python code produces the below visualization that help us understand whether the chosen number of clusters is a good choice or not.

A drawback of this approach (which is also presented in the link) is that it can produce more than one possible number of clusters, so you will need to choose by yourself from the different options.

Inertia and Elbow Method

Inertia is the sum of squared distance of samples to their closest cluster center. We would like this number to be as small as possible. But, if we choose K that is equal to the number of samples we will get $\text{inertia}=0$. This is the smallest inertia value we can achieve, however we miss our goal of clustering the data into the optimal number of clusters.

The value of inertia decreases as the number of clusters increase- so we will need to manually pick K while considering the trade-off between the inertia value and the number of clusters. For that, we usually use the Elbow Method- and we choose the elbow point in the inertia graph. After that point the improvement in the inertia value is not significant.



Inertia Graph

Based on this approach, the chosen K is around 8–11, but it needs to be manually picked.

Here is a very good article summarizing this approach:

K-Means Clustering: From A to Z

Everything you need to know about K-means clustering

towardsdatascience.com

Suggested Approach- Scaled Inertia

The two approaches above require your manual decision for the number of clusters. Based on what I have learned from these approaches, I have developed an automatic process for choosing K- the number of clusters.

The suggested approach takes into account the inertia value for each possible K and weights it by a penalty parameter. This parameter

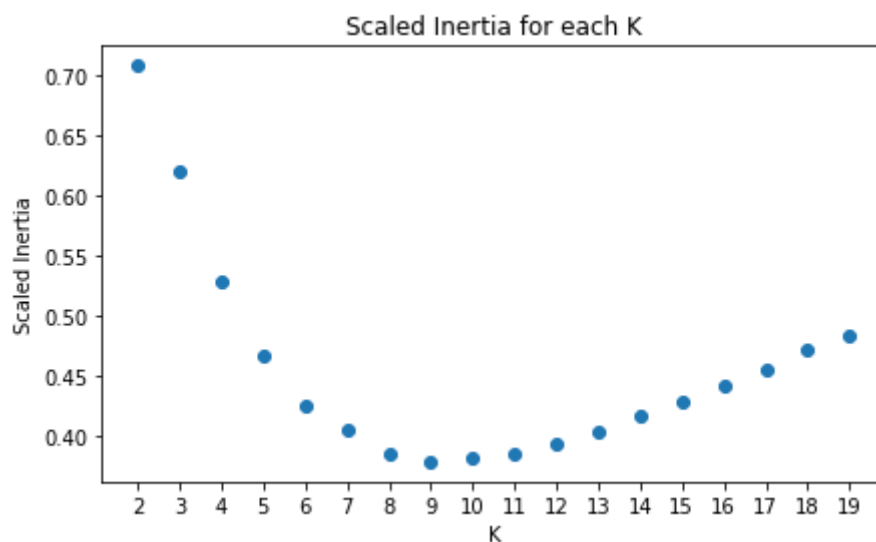
represents the trade-off between the inertia and the number of clusters.

Instead of using the inertia by itself, we computed a weighted version of it:

$$\text{Scaled Inertia} = \frac{\text{Inertia}(K)}{\text{Inertia}(K = 1)} + \alpha \cdot K$$

Scaled Inertia Formula

- Inertia- sum of squared distanced of samples to their closest cluster center
- Alpha- manually tuned factor that gives penalty to the number of clusters
- Inertia(K=1)- inertia for the basic situation in which all data points are in the same cluster



Scaled Inertia Graph


Alpha is manually tuned because as I see it, the penalty for the number of clusters is a business decision that should be incorporated into the analysis.

Using the Scaled Inertia, the chosen K is obvious and can be done automatically. In the above case $K=9$.

17. What do you mean by Dimensionality Reduction ? Explain about Isomap?

ANSWER:

Isomap within the family of Machine Learning algorithms

There are so many Machine Learning algorithms that it may never be possible to collect and categorize them all. However, I have attempted to do it for some of the most commonly used ones, which you can find in the **interactive** sunburst chart below. Make sure to explore the chart by clicking  on different categories to **enlarge and reveal more**.

Machine Learning algorithm classification. Interactive chart created by the [author](#).

If you enjoy Data Science and Machine Learning, please [subscribe](#) to get an email whenever I publish a new story.

As you can see, Isomap is an **Unsupervised Machine Learning** technique aimed at **Dimensionality Reduction**.

It differs from a few other techniques in the same category by using a **non-linear** approach to dimensionality reduction instead of linear mappings used by algorithms such as PCA. We will see how linear vs. non-linear approaches differ in the next section.

How does Isometric Mapping (Isomap) work?

Isomap is a technique that combines several different algorithms, enabling it to use a non-linear way to reduce dimensions while preserving local structures.

Before we look at the example of Isomap and compare it to a linear method of Principal Components Analysis (PCA), let's list the high-level steps that Isomap performs:

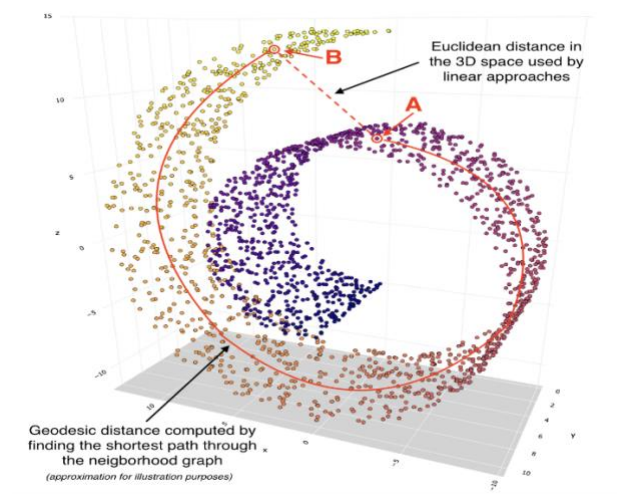
1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, "k" is an arbitrary number of neighbors that you can specify within model hyperparameters.
2. Once the neighbors are found, **construct the neighborhood graph** where points are connected to each other if they are each other's neighbors. Data points that are not neighbors remain unconnected.
3. **Compute the shortest path** between each pair of data points (nodes). Typically, it is either Floyd-Warshall or Dijkstra's algorithm that is used for this task. Note, this step is also commonly described as finding a **geodesic distance** between points.
4. Use [multidimensional scaling \(MDS\)](#) to **compute lower-dimensional embedding**. Given distances between each pair of points are known, MDS places each object into the N-dimensional space (N is specified as a hyperparameter) such that the between-point distances are preserved as well as possible.

For our example, let's create a 3D object known as a Swiss roll. The object is made up of 2,000 individual data points. The chart

is **interactive** so make sure to rotate it to familiarize yourself with its exact shape.

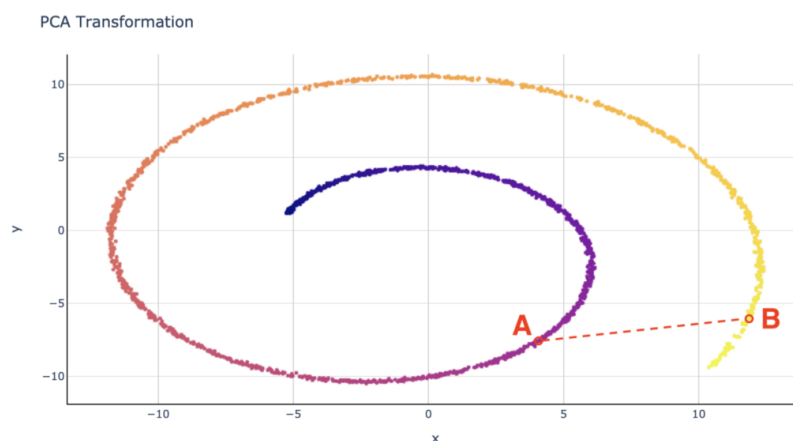
Interactive 3D swiss roll. Graph by [author](#).

Next, we want to map this 3-dimensional swiss roll 2 dimensions using Isomap. To track what happens during this transformation, let's select two points: A and B.



Euclidean vs. geodesic distances between points on a 3D Swiss roll. Image by [author](#).

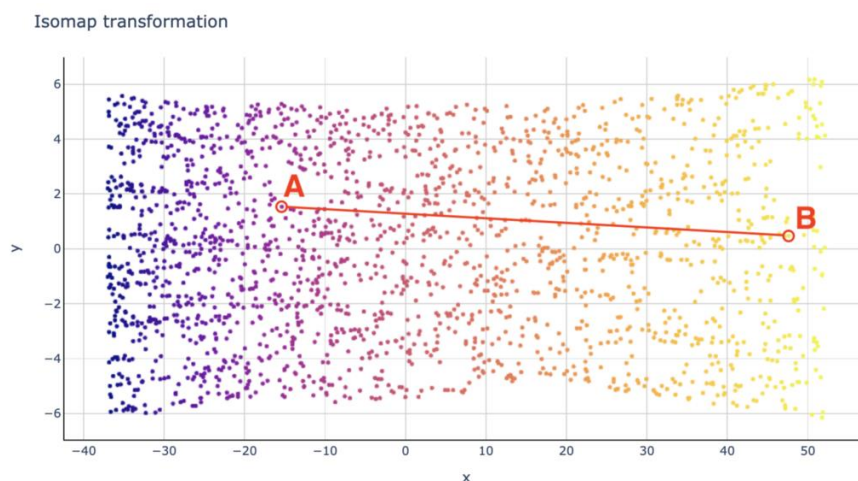
We can see that these two points are relatively close to each other within the 3D space. If we used a linear dimensionality reduction approach such as PCA, then the Euclidean distance between these two points would remain somewhat similar in lower dimensions. See PCA transformation chart below:



3D swiss roll reduced to 2 dimensions using PCA. Image by [author](#).

Note, the shape of a 2D object in PCA looks like a picture taken of the same 3D object but from a specific angle. This is a feature of linear transformation.

Meanwhile, non-linear approaches such as Isomap give us a very different result. We can describe such transformation as unrolling the Swiss roll and laying it flat on a 2D surface:



3D swiss roll reduced to 2 dimensions using Isomap. Image by [author](#).

We can see that the distance between points A and B in a 2D space is based on the geodesic distance computed by going through the neighborhood connections.


That's the secret to Isomap's ability to perform non-linear dimensionality reduction while balancing the relationship between local and global structures.

18 Explain about Locally Linear Embedding Process in detail

ANSWER:

Locally Linear Embedding (LLE) within the universe of Machine Learning algorithms

Even an experienced Data Scientist can easily get lost amongst hundreds of different Machine Learning algorithms used in the industry. Hence, I believe there is value in creating a structure by putting some of the most frequently used algorithms into categories.

It will never be perfect because some algorithms are flexible enough to perform multiple tasks. Nevertheless, below is my attempt to create such categorization. Make sure to explore this **interactive** graph by clicking  on different sections **to reveal more**.

Machine Learning algorithm classification. Interactive chart created by the [author](#).

As you can see, Locally Linear Embedding (LLE) sits under the **Unsupervised** branch of Machine Learning within the group of **dimensionality reduction** algorithms.

It means that you do not need to have a target variable to perform dimensionality reduction with LLE, unlike supervised techniques such as Linear Discriminant Analysis (LDA).

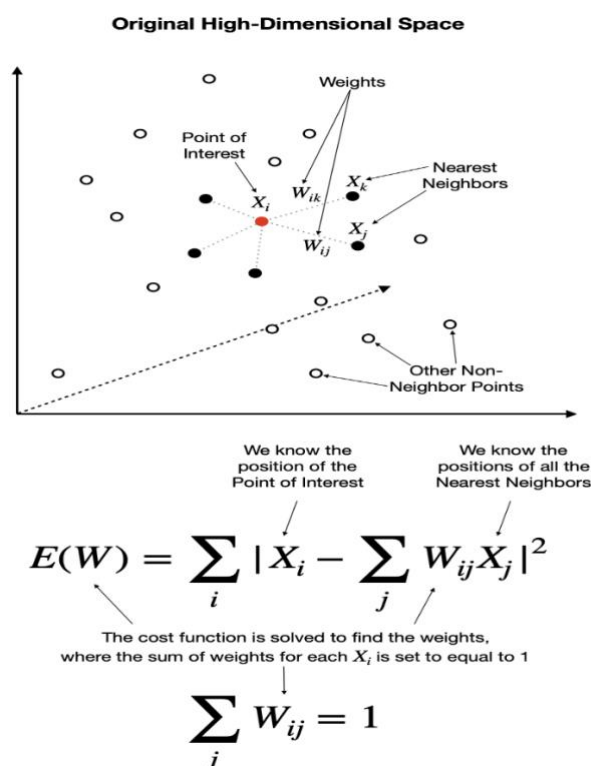
If you enjoy Data Science and Machine Learning, please [subscribe](#) to get an email whenever I publish a new story.

How does Locally Linear Embedding work?

The high-level steps

Similar to Isomap, LLE combines several steps to produce the lower-dimensional embedding. These are:

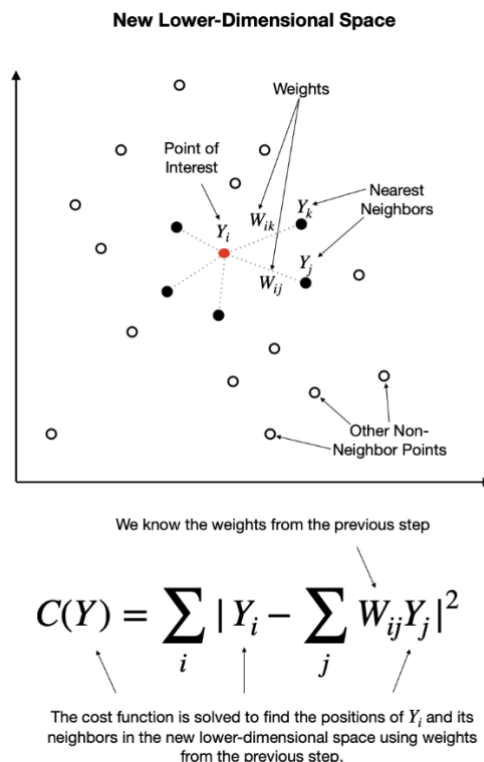
1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, “k” is an arbitrary number of neighbors that you can specify within model hyperparameters.
2. **Construct a weight matrix** where every point has its weights determined by minimizing the error of the cost function shown below. Note that every point is a linear combination of its neighbors, which means that **weights for non-neighbors are 0**.



Finding weights in the original high-dimensional space. Image by [author](#).

3. **Find the positions** of all the points in the **new lower-dimensional embedding** by minimizing the cost function shown below. Note, here we use weights (W) from step two and solve for Y.

The actual solving is performed using Partial Eigenvalue Decomposition.



Finding the position of points in the new lower-dimensional embedding. Image by [author](#).

With the above steps completed, we get a lower-dimensional representation of the data, which we can typically visualize using standard scatterplots, provided we reduce the dimensionality to 3D or less.

LLE variants

You should be aware of a few LLE variants, which improve upon the original setup. However, note that these improvements come at the cost of efficiency, making the algorithm slower. Here is how [scikit-learn](#) describes these variants:

- **Modified LLE (MLLE)** — One well-known issue with LLE is the regularization problem. A way to address it is to

use **multiple weight vectors** in each neighborhood. This is the essence of MLLE.

- **Hessian LLE (HLLE)**— Hessian Eigenmapping is another method of solving the regularization problem of LLE. It revolves around a **hessian-based quadratic form** at each neighborhood used to recover the locally linear structure.

While I will not go into details, I recommend you experiment with them to see which variant yields the best results for your data. Personally, I find MLLE to perform well in most scenarios (see an example of this in the next section).

Difference between LLE and Isomap

The two algorithms are similar in the way they approach dimensionality reduction, but they do have their differences.

Similar to LLE, Isomap also uses KNN to find the nearest neighbors in the first step. However, the second step constructs neighborhood graphs instead of describing each point as a linear combination of its neighbors. Then it uses these graphs to compute the shortest path between every pair of points.

Finally, Isomap uses those pairwise distances between all points to construct a lower-dimensional embedding.

Should I choose LLE over Isomap?

In general, LLE is a more efficient algorithm as it eliminates the need to estimate pairwise distances between widely separated data points. Furthermore, it assumes that the manifold is linear when viewed

locally. Thus it recovers the non-linear structure from locally linear fits.

19. Explain in detail about Factor Analysis

ANSWER:

Factor Analysis in Machine Learning :

1. Reduce a large numbers of variables into fewer numbers of factors.

2. Puts maximum common variance into a common score.

3. Associates multiple observed variables with a latent variable.

4. Has the same numbers of factors and variables,where each factor contains a certain amount of overall variance .

Eigenvalue : A measure of the variance that a factor explains for observed variables. A factor with eigenvalue < 1 explains less variance than a single observed value.

Factor Analysis Process :

• ***Principal Component Analysis (PCA)***

Extract the hidden factor from the dataset.

Defines data using less numbers of components,explaining the variance in your data

Reduce the computation complexity .

Determine that the new data is the part of the group of data points from the training set.

• **Linear Discriminant Analysis(LDA)**

Reduces dimensions.

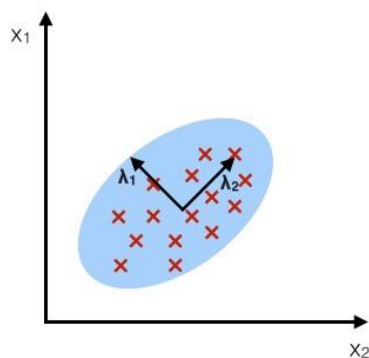
Search the linear combination of variables that best separates two class.

Reduce degree of overfitting.

Determine how to classify the new observation out of group of classes.

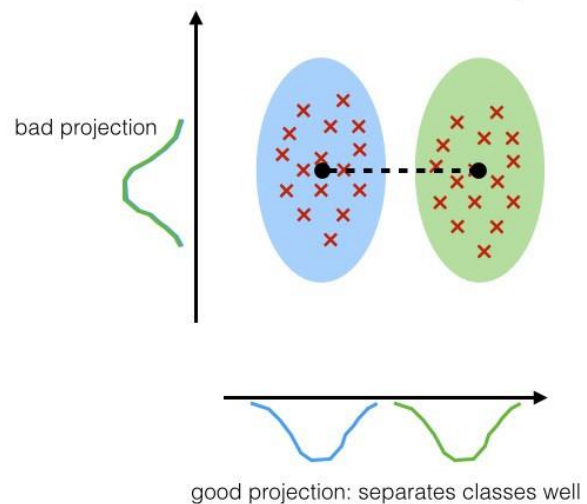
PCA:

component axes that maximize the variance



LDA:

maximizing the component axes for class-separation



Direction of Maximum Variance:

1. PCA seeks the linear combination of variables in order to extract the maximum variance.

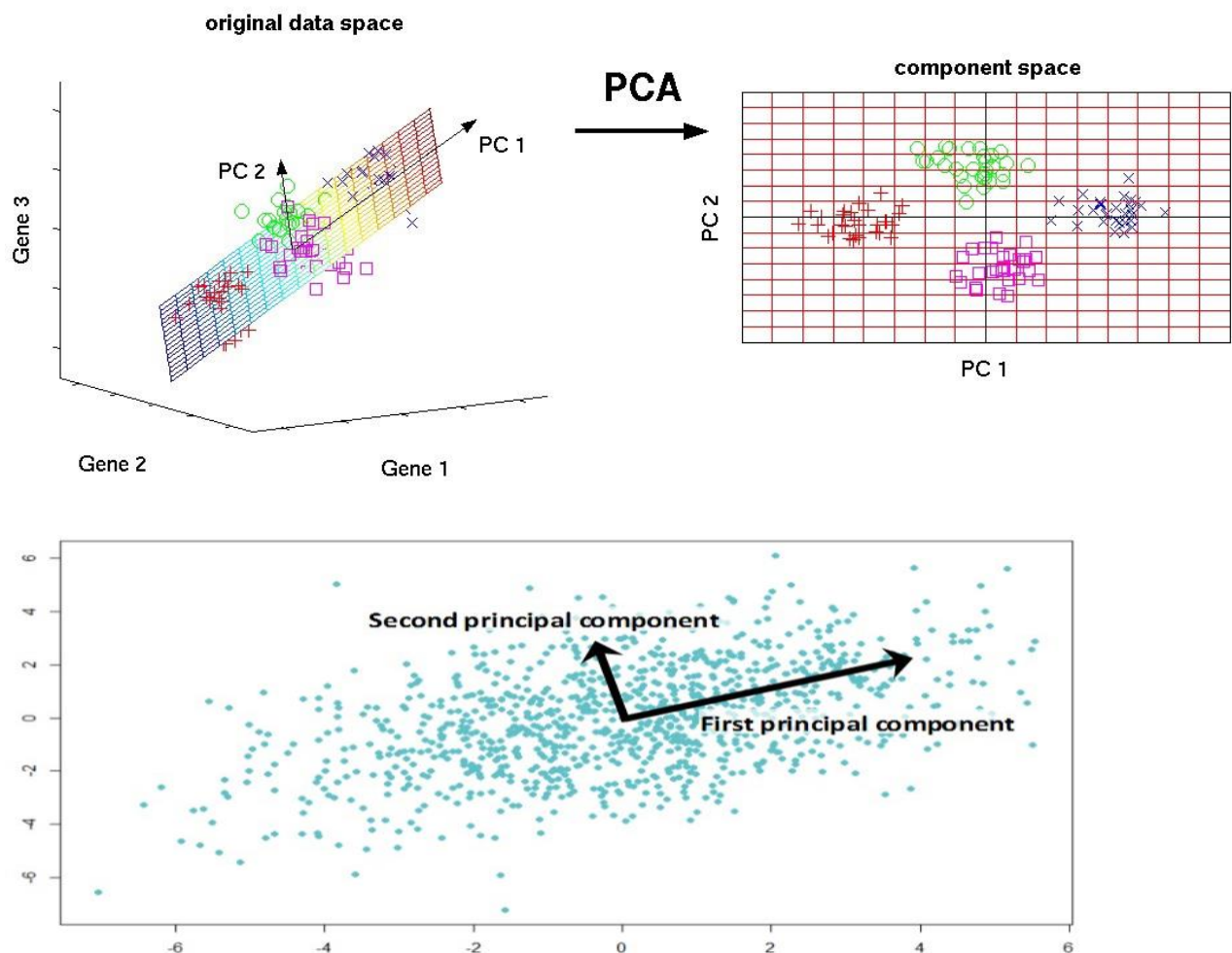
2. 2. Compute Eigenvector that are principal components of the dataset and collect them in projection matrix.
3. 3. Each of the Eigenvector is associate with Eigenvalue, which is magnitudes.
4. 4. Reduce the dataset into smaller dimensional subspace by dropping the less informative Eigenpairs.

PCA finds line depending on two criteria :

1. The variation of values should be maximal along this line.

2. The error should be minimum if you don't reconstruct original two positions of a blue dot from the new position of the red dot.

First Principle Component :



The first principle component (PC1) is the direction of the maximum variance and is obtained by solving Eigenvector .

Finding PC1 :

PC1 (Mathematically) : $a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n$

Constraint: $a_1^2 + a_2^2 + a_3^2 + \dots + a_k^2$

Eigen decomposition to solve the equation.

NOTE : Eigen decomposition is the factorisation of the matrix into a canonical form, where the matrix is represented in terms of Eigenvectors or Eigenvalues.

Eigenvalues and PCA. :

Eigenvalues are the variances of the principal component arranged in descending order.

Summary of PCA Process :

1. Standardize the data PCA : Requires that the input variables have similar scales of the measurement.

2. Build the correlation matrix : This summarizes how your variables all relate to one another.

3. Obtain the Eigenvalue and Eigenvector from correlation matrix : Break the matrix down in direction and magnitude . Sort Eigenvalues in descending order and choose Eigenvectors that corresponds to the largest Eigenvalue.

4. Construct the projection matrix from selected Eigenvector : Reduce the dataset by dropping less informative Eigenpairs.
5. Transform the original dataset to obtain a k -dimensional feature sub space : Compress your data into smaller space by excluding less important directions

20. Explain the importance of Subset selection in Dimensionality Reduction

ANSWER:

REFER Q.2 PART-B