Chapter No 6 Dimensionality Reduction

Principal Component Analysis

Introduction:

Principal Component Analysis is an unsupervised learning algorithm that is used for 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.

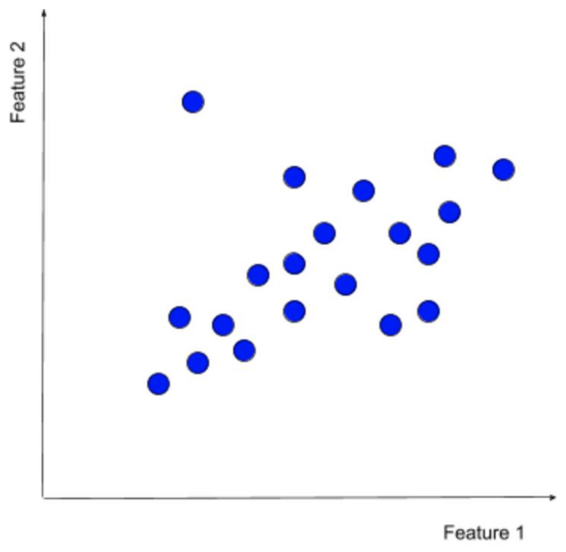
Applications of Principal Component Analysis

- 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.

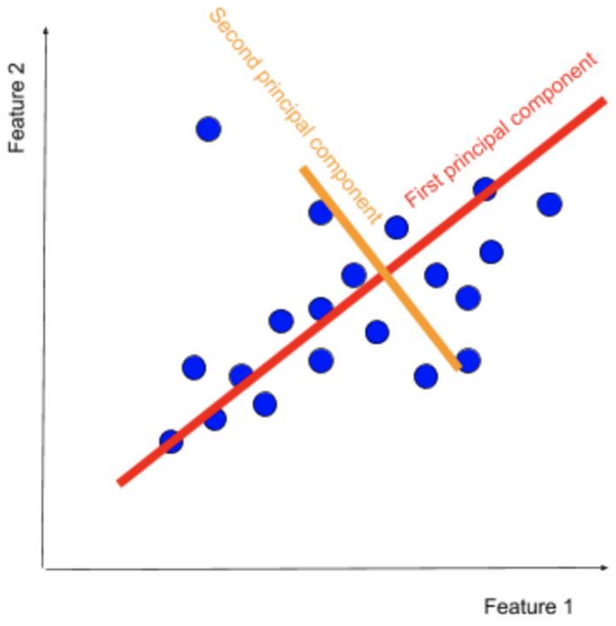
Imagine we have a 2-dimensional dataset. Each dimension can be represented as a feature column:

Feature 1	Feature 2
4	2
6	3
13	6
...	...

We can represent the same dataset as a scatterplot:



The main aim of PCA is to find such principal components, which can describe the data points with a set of... well, principal components.



The principal components are vectors, but they are not chosen at random. The first principal component is computed so that it explains the greatest amount of variance in the original features. The second component is orthogonal to the first, and it explains the greatest amount of variance left *after* the first principal component.

The original data can be represented as feature vectors. PCA allows us to go a step further and represent the data as linear combinations of principal components. Getting principal components is equivalent to a linear transformation of data from the feature1 x feature2 axis to a PCA1 x PCA2 axis.

Why is this useful?

In the small 2-dimensional example above, we do not gain much by using PCA, since a feature vector of the form (feature1, feature2) will be very similar to a vector of the form (first principal component (PCA1), second principal component (PCA2)). But in very large datasets (where the number of dimensions can surpass 100 different variables), principal components remove noise by reducing a large number of features to just a couple of principal components. Principal components are orthogonal projections of data onto lower-dimensional space.

In theory, PCA produces the same number of principal components as there are features in the training dataset. In practice, though, we do not keep all of the principal components. Each successive principal component explains the variance that is left after its preceding component, so picking just a few of the first components sufficiently approximates the original dataset *without* the need for additional features.

SVD Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices.

It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations.

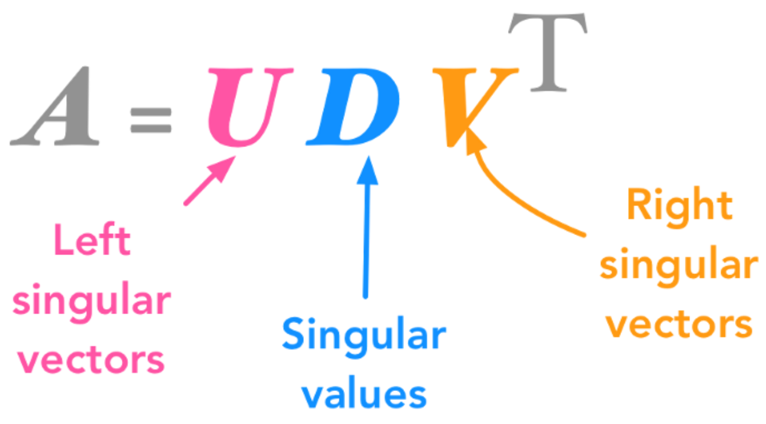
It also has some important applications in data science.

Mathematics behind SVD

The SVD of mxn matrix A is given by the formula :

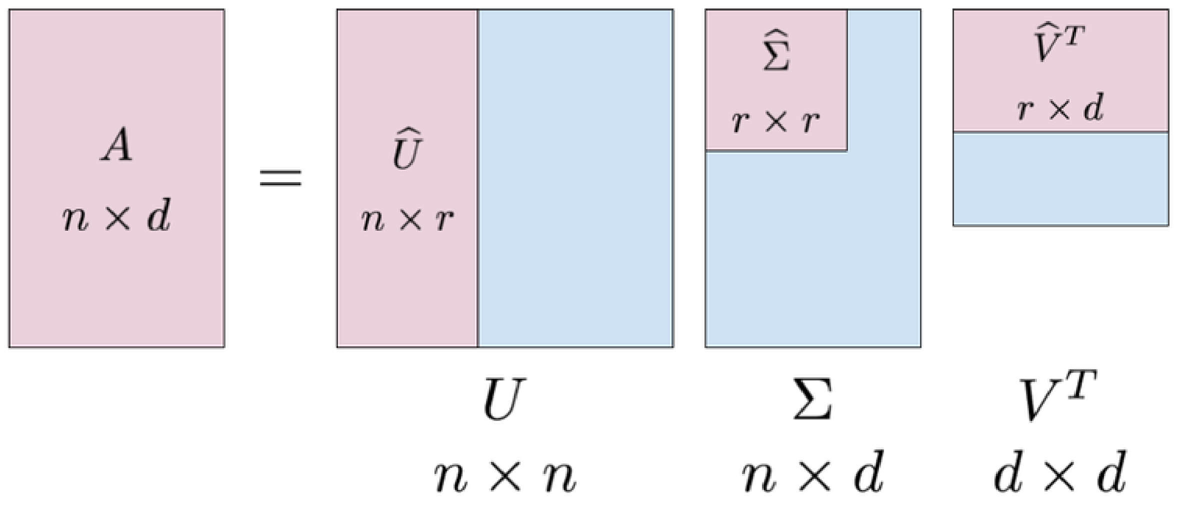
Mathematics behind SVD

The SVD of mxn matrix A is given by the formula :

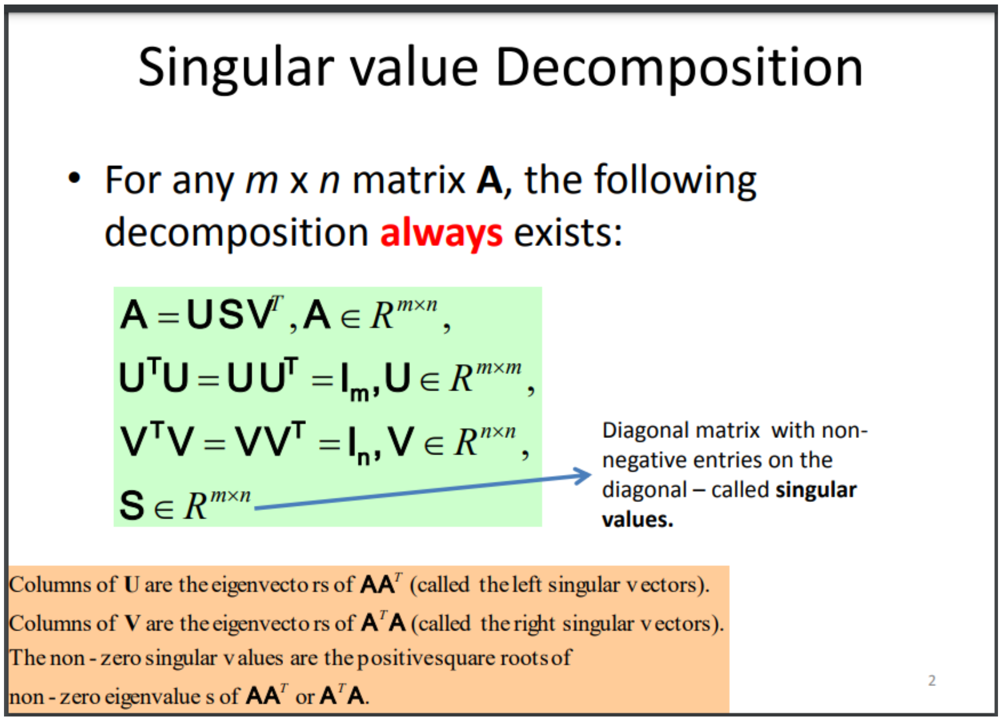


The SVD of an mxn matrix A with real values is a factorization of A as U∑V^T, where

- U is an mxm orthogonal matrix,
- V is an nxn orthogonal matrix, and
- ∑ is a diagonal matrix with nonnegative real entries on the diagonal.



A $n \times d$
$=$
$\widehat{U}$ $n \times r$
$\widehat{\Sigma}$ $r \times r$
$\widehat{V}^T$ $r \times d$
U $n \times n$
Σ $n \times d$
V^T $d \times d$



Singular value Decomposition

- For any *m* x *n* matrix **A**, the following decomposition **always** exists:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \mathbf{A} \in R^{m\times n},$$
$$\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}_m, \mathbf{U} \in R^{m\times m},$$
$$\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}_n, \mathbf{V} \in R^{n\times n},$$
$$\mathbf{S} \in R^{m\times n}$$

Diagonal matrix with non-negative entries on the diagonal – called **singular values.**

Columns of U are the eigenvectors of $\mathbf{AA}^T$ (called the left singular vectors).
Columns of V are the eigenvectors of $\mathbf{A}^T\mathbf{A}$ (called the right singular vectors).
The non-zero singular values are the positive square roots of non-zero eigenvalues of $\mathbf{AA}^T$ or $\mathbf{A}^T\mathbf{A}$.

2

Linear Discriminant Analysis (LDA) in Machine Learning

Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).

This can be used to project the features of higher dimensional space into lower-dimensional space in order to reduce resources and dimensional costs. In this topic, "**Linear Discriminant Analysis (LDA) in machine learning"**, we will discuss the LDA algorithm for classification predictive modeling problems, limitation of logistic regression, representation of linear Discriminant analysis model, how to make a prediction using LDA, how to prepare data for LDA, extensions to LDA and much more. So, let's start with a quick introduction to Linear Discriminant Analysis (LDA) in machine learning.

What is Linear Discriminant Analysis (LDA)?

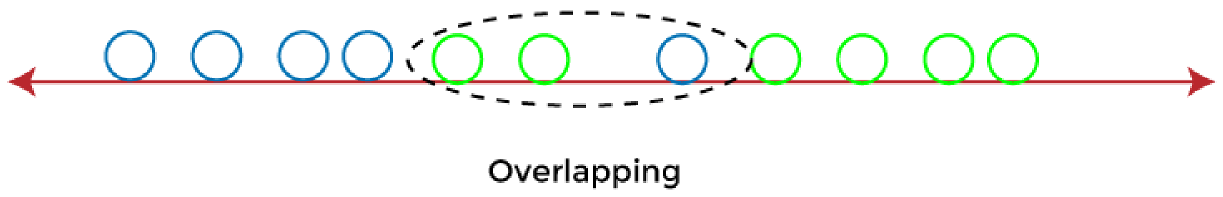
Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems.

Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning.

It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification.

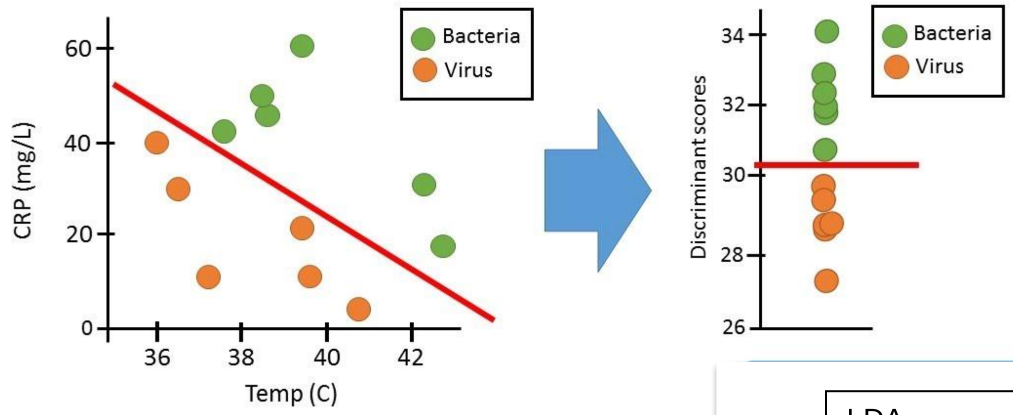
Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems.

For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.

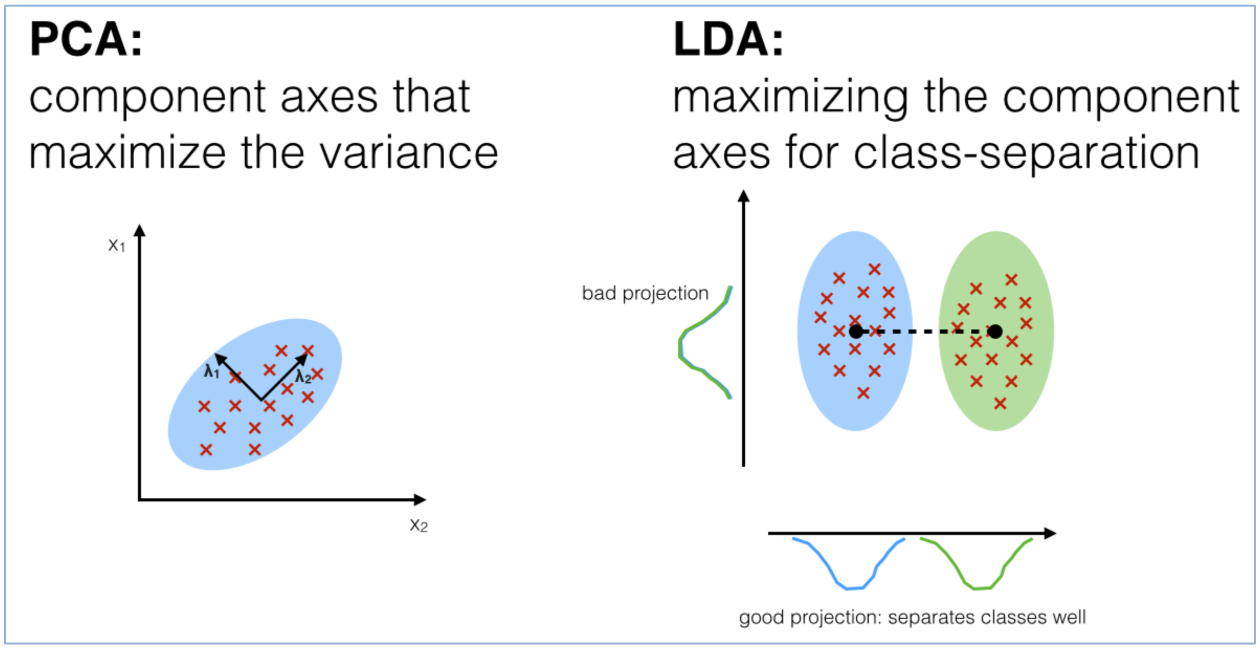


To overcome the overlapping issue in the classification process, we must increase the number of features regularly.

LDA



LDA



PCA:
component axes that maximize the variance

LDA:
maximizing the component axes for class-separation

Calculate the ‘separability’ between the classes.

“Between-class scatter matrix”

Overall mean

$$S_b = \sum_{i=1}^{g} N_i(\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

Sample size of class *i*

Sample mean of class *i*

Compute the within-class variance, or the distance between the mean and the sample of every class.

“Within-class scatter matrix”

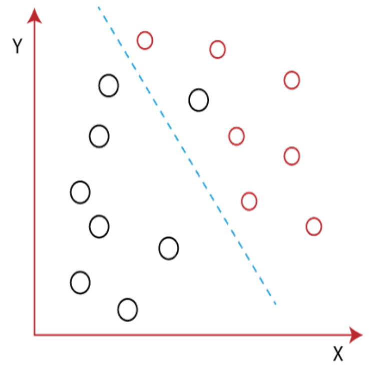
Sample size

$$S_w = \sum_{i=1}^{g}(N_i - 1)S_i = \sum_{i=1}^{g}\sum_{j=1}^{N_i}(x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

Scatter matrix for class *i*

Example:

Let's assume we have to classify two different classes having two sets of data points in a 2-dimensional plane as shown below image:



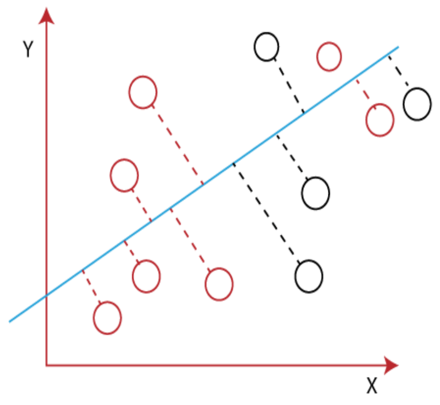
However, it is impossible to draw a straight line in a 2-d plane that can separate these data points efficiently but using linear Discriminant analysis; we can dimensionally reduce the 2-D plane into the 1-D plane. Using this technique, we can also maximize the separability between multiple classes.

How Linear Discriminant Analysis (LDA) works?

Linear Discriminant analysis is used as a dimensionality reduction technique in machine learning, using which we can easily transform a 2-D and 3-D graph into a 1-dimensional plane.

Let's consider an example where we have two classes in a 2-D plane having an X-Y axis, and we need to classify them efficiently. As we have already seen in the above example that LDA enables us to draw a straight line that can completely separate the two classes of the data points. Here, LDA uses an X-Y axis to create a new axis by separating them using a straight line and projecting data onto a new axis.

Hence, we can maximize the separation between these classes and reduce the 2-D plane into 1-D.



To create a new axis, Linear Discriminant Analysis uses the following criteria:

- It maximizes the distance between means of two classes.
- It minimizes the variance within the individual class.

Using the above two conditions, LDA generates a new axis in such a way that it can maximize the distance between the means of the two classes and minimizes the variation within each class.

In other words, we can say that the new axis will increase the separation between the data points of the two classes and plot them onto the new axis.

Why LDA?

- Logistic Regression is one of the most popular classification algorithms that perform well for binary classification but falls short in the case of multiple classification problems with well-separated classes. At the same time, LDA handles these quite efficiently.
- LDA can also be used in data pre-processing to reduce the number of features, just as PCA, which reduces the computing cost significantly.
- LDA is also used in face detection algorithms. In Fisherfaces, LDA is used to extract useful data from different faces. Coupled with eigenfaces, it produces effective results.

Drawbacks of Linear Discriminant Analysis (LDA)

Although, LDA is specifically used to solve supervised classification problems for two or more classes which are not possible using logistic regression in machine learning. But LDA also fails in some cases where the Mean of the distributions is shared. In this case, LDA fails to create a new axis that makes both the classes linearly separable.

To overcome such problems, we use **non-linear Discriminant analysis** in machine learning.

Extension to Linear Discriminant Analysis (LDA)

Linear Discriminant analysis is one of the most simple and effective methods to solve classification problems in machine learning. It has so many extensions and variations as follows:

1. **Quadratic Discriminant Analysis (QDA):** For multiple input variables, each class deploys its own estimate of variance.
2. **Flexible Discriminant Analysis (FDA):** it is used when there are non-linear groups of inputs are used, such as splines.
3. **Flexible Discriminant Analysis (FDA):** This uses regularization in the estimate of the variance (actually covariance) and hence moderates the influence of different variables on LDA.

Real-world Applications of LDA

Some of the common real-world applications of Linear discriminant Analysis are given below:

- **Face Recognition**
  Face recognition is the popular application of computer vision, where each face is represented as the combination of a number of pixel values. In this case, LDA is used to minimize the number of features to a manageable number before going through the classification process. It generates a new template in which each dimension consists of a linear combination of pixel values. If a linear combination is generated using Fisher's linear discriminant, then it is called Fisher's face.
- **Medical**
  In the medical field, LDA has a great application in classifying the patient disease on the basis of various parameters of patient health and the medical

treatment which is going on. On such parameters, it classifies disease as mild, moderate, or severe. This classification helps the doctors in either increasing or decreasing the pace of the treatment.

- **Customer Identification**

  In customer identification, LDA is currently being applied. It means with the help of LDA; we can easily identify and select the features that can specify the group of customers who are likely to purchase a specific product in a shopping mall. This can be helpful when we want to identify a group of customers who mostly purchase a product in a shopping mall.

- **For Predictions**

  LDA can also be used for making predictions and so in decision making. For example, "will you buy this product" will give a predicted result of either one or two possible classes as a buying or not.

- **In Learning**

  Nowadays, robots are being trained for learning and talking to simulate human work, and it can also be considered a classification problem. In this case, LDA builds similar groups on the basis of different parameters, including pitches, frequencies, sound, tunes, etc.

—--

Chapter No 5 Learning with clustering

Expectation maximization algorithm:

In the real-world applications of machine learning, it is very common that there are many relevant features available for learning but only a small subset of them are observable.

So, for the variables which are sometimes observable and sometimes not, then we can use the instances when that variable is visible is observed for the purpose of learning and then predict its value in the instances when it is not observable.

On the other hand,

Expectation-Maximization algorithm can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables) too in order to predict their values with the condition that the general form of probability distribution governing those latent variables is known to us.

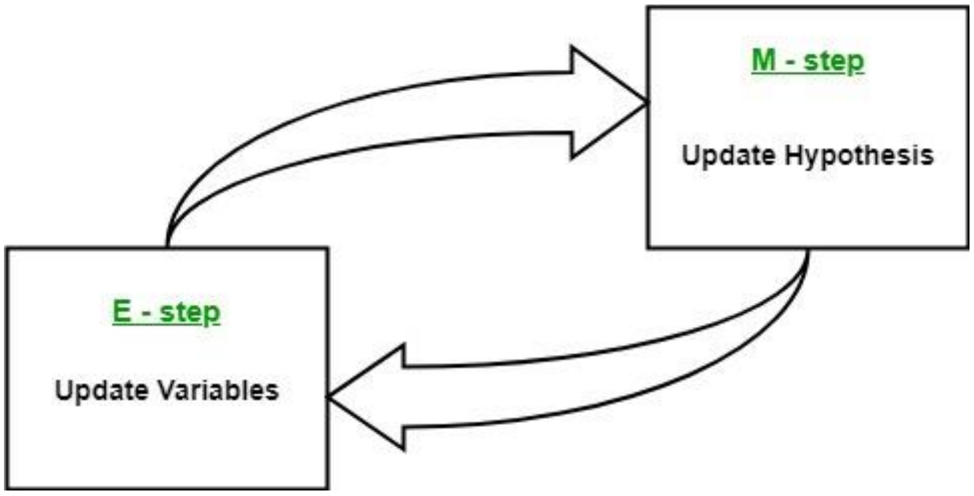
This algorithm is actually at the base of many unsupervised clustering algorithms in the field of machine learning.

It was explained, proposed and given its name in a paper published in 1977 by Arthur Dempster, Nan Laird, and Donald Rubin. It is used to find the *local maximum likelihood parameters* of a statistical model in the cases where latent variables are involved and the data is missing or incomplete.

Algorithm:

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.

3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.

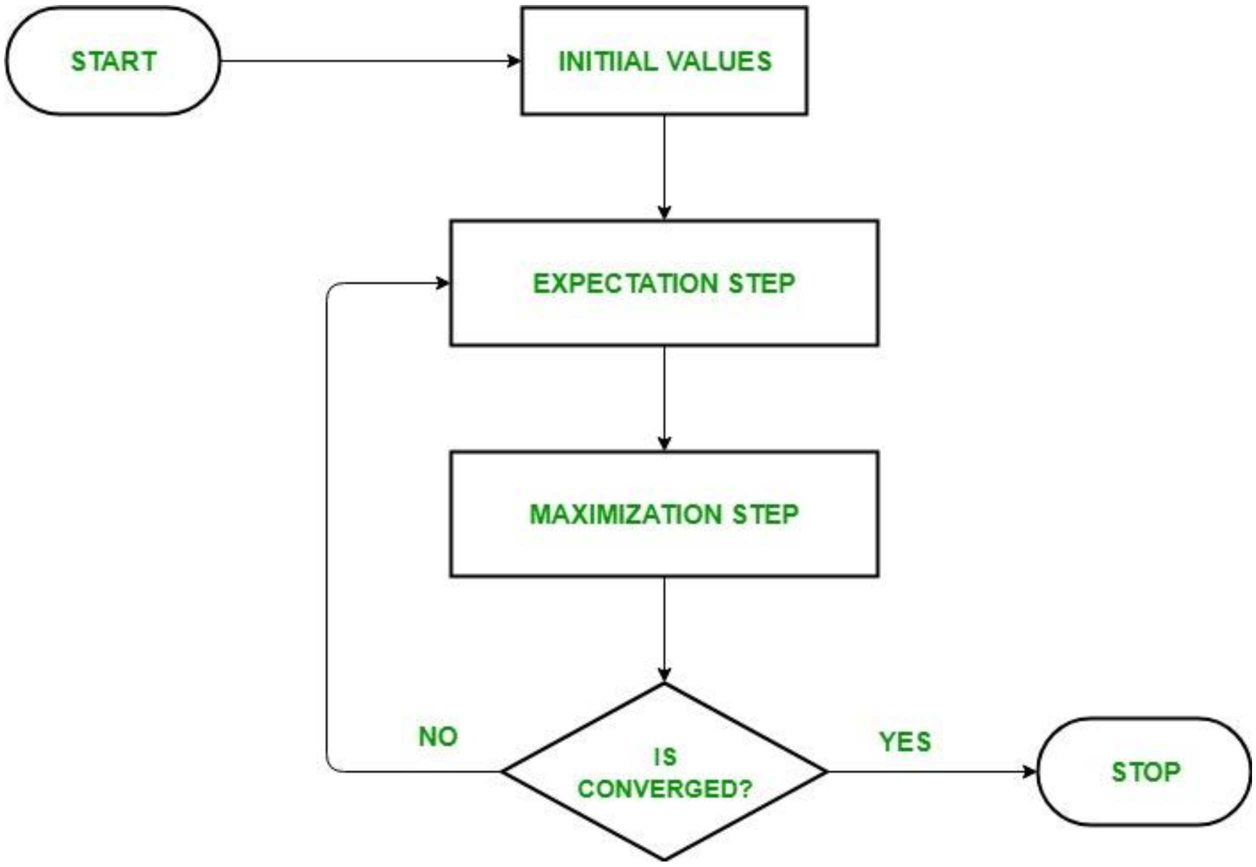


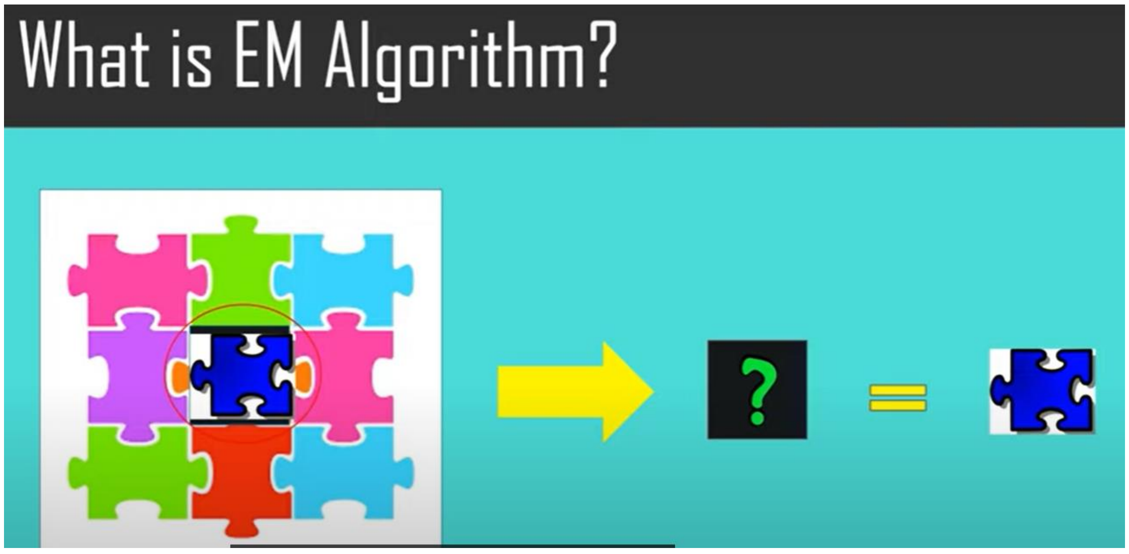
The essence of Expectation-Maximization algorithm is to use the available observed data of the dataset to estimate the missing data and then using that data to update the values of the parameters. Let us understand the EM algorithm in detail.

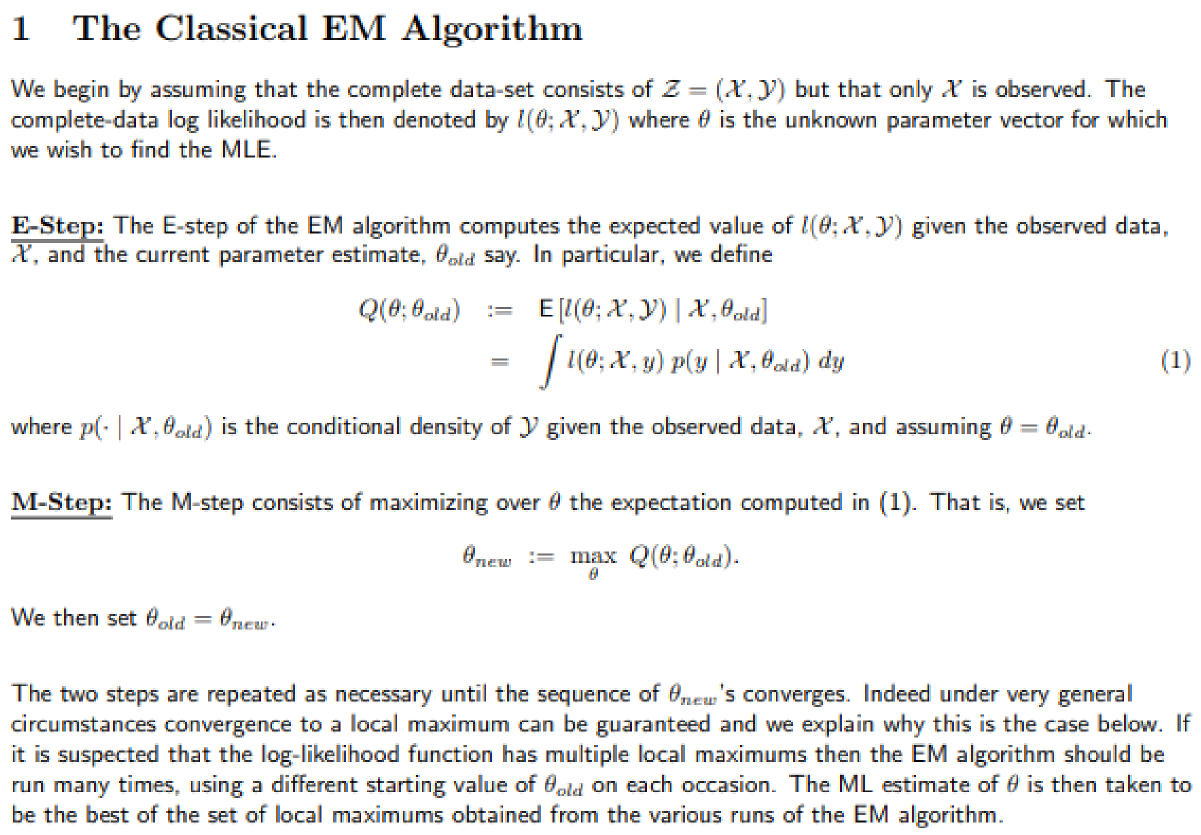
- Initially, a set of initial values of the parameters are considered. A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.
- The next step is known as “Expectation” – step or *E-step*. In this step, we use the observed data in order to estimate or guess the values of the missing or incomplete data. It is basically used to update the variables
- The next step is known as “Maximization”-step or *M-step*. In this step, we use the complete data generated in the preceding “Expectation” – step in order to update the values of the parameters. It is basically used to update the hypothesis.

- Now, in the fourth step, it is checked whether the values are converging or not, if yes, then stop otherwise repeat *step-2* and *step-3* i.e. “Expectation” – step and “Maximization” – step until the convergence occurs.

Flow chart for EM algorithm –







1 The Classical EM Algorithm

We begin by assuming that the complete data-set consists of $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ but that only $\mathcal{X}$ is observed. The complete-data log likelihood is then denoted by $l(\theta; \mathcal{X}, \mathcal{Y})$ where θ is the unknown parameter vector for which we wish to find the MLE.

E-Step: The E-step of the EM algorithm computes the expected value of $l(\theta; \mathcal{X}, \mathcal{Y})$ given the observed data, $\mathcal{X}$, and the current parameter estimate, θ_{old} say. In particular, we define

$$\begin{aligned} Q(\theta; \theta_{old}) &:= \mathsf{E}\left[l(\theta; \mathcal{X}, \mathcal{Y}) \mid \mathcal{X}, \theta_{old}\right] \\ &= \int l(\theta; \mathcal{X}, y)\, p(y \mid \mathcal{X}, \theta_{old})\, dy \end{aligned} \tag{1}$$

where $p(\cdot \mid \mathcal{X}, \theta_{old})$ is the conditional density of $\mathcal{Y}$ given the observed data, $\mathcal{X}$, and assuming $\theta = \theta_{old}$.

M-Step: The M-step consists of maximizing over θ the expectation computed in (1). That is, we set

$$\theta_{new} := \max_{\theta}\; Q(\theta; \theta_{old}).$$

We then set $\theta_{old} = \theta_{new}$.

The two steps are repeated as necessary until the sequence of θ_{new}'s converges. Indeed under very general circumstances convergence to a local maximum can be guaranteed and we explain why this is the case below. If it is suspected that the log-likelihood function has multiple local maximums then the EM algorithm should be run many times, using a different starting value of θ_{old} on each occasion. The ML estimate of θ is then taken to be the best of the set of local maximums obtained from the various runs of the EM algorithm.

Usage of EM algorithm –

- It can be used to fill the missing data in a sample.
- It can be used as the basis of unsupervised learning of clusters.

- It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM).
- It can be used for discovering the values of latent variables.

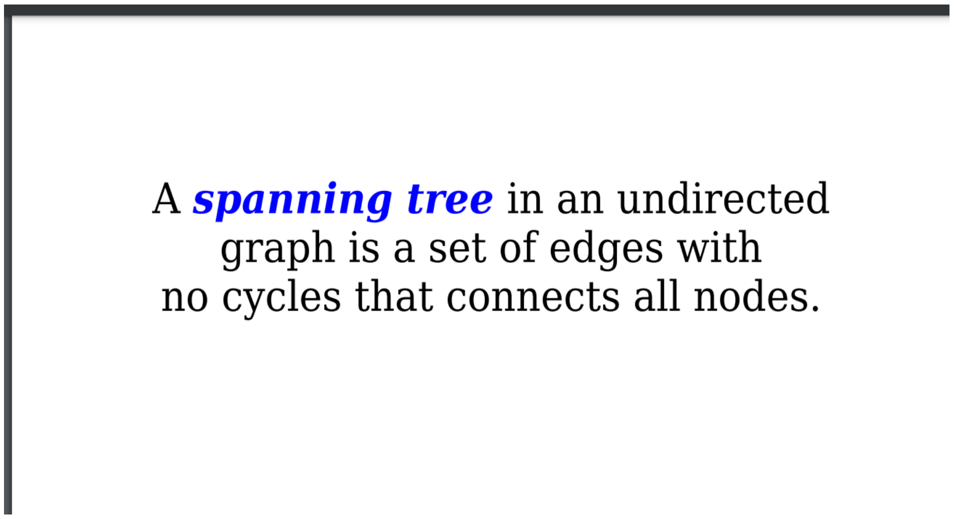
Advantages of EM algorithm –

- It is always guaranteed that likelihood will increase with each iteration.
- The E-step and M-step are often pretty easy for many problems in terms of implementation.
- Solutions to the M-steps often exist in the closed form.

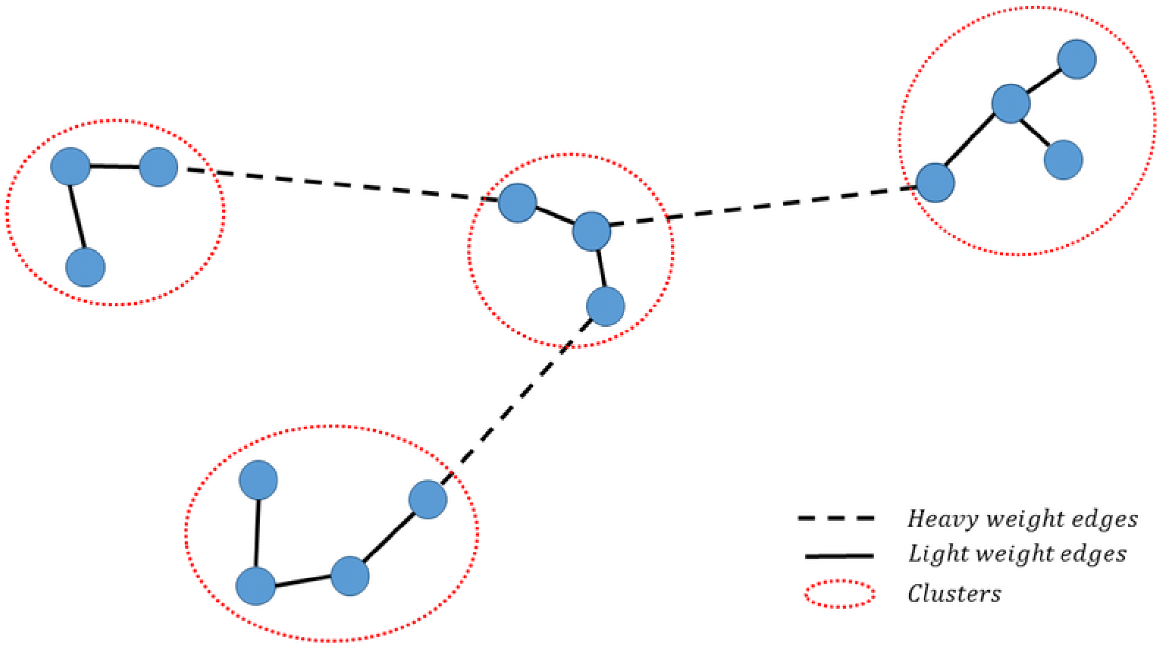
Disadvantages of EM algorithm –

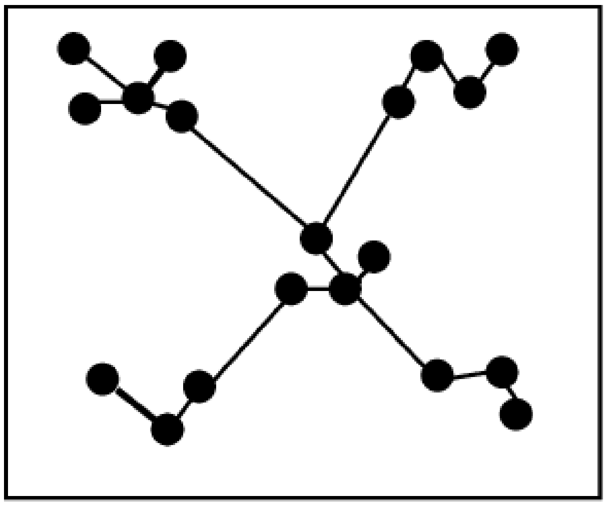
- It has slow convergence.
- It makes convergence to the local optima only.
- It requires both the probabilities, forward and backward (numerical optimization requires only forward probability).

Graph Based Clustering: Using Minimum Spanning Tree:

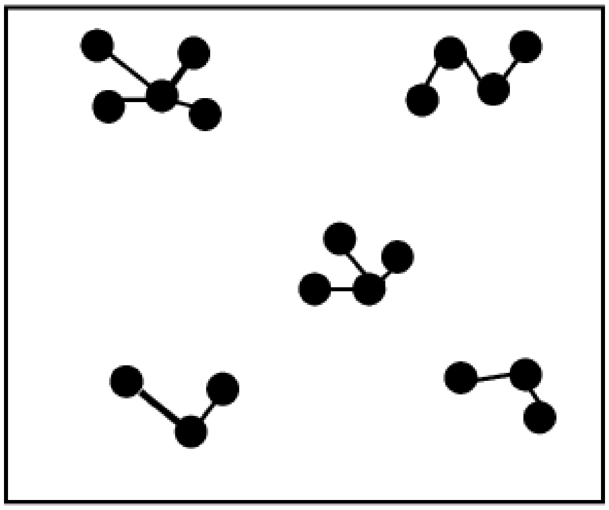


A ***spanning tree*** in an undirected graph is a set of edges with no cycles that connects all nodes.





(b) An MST connecting all the data points



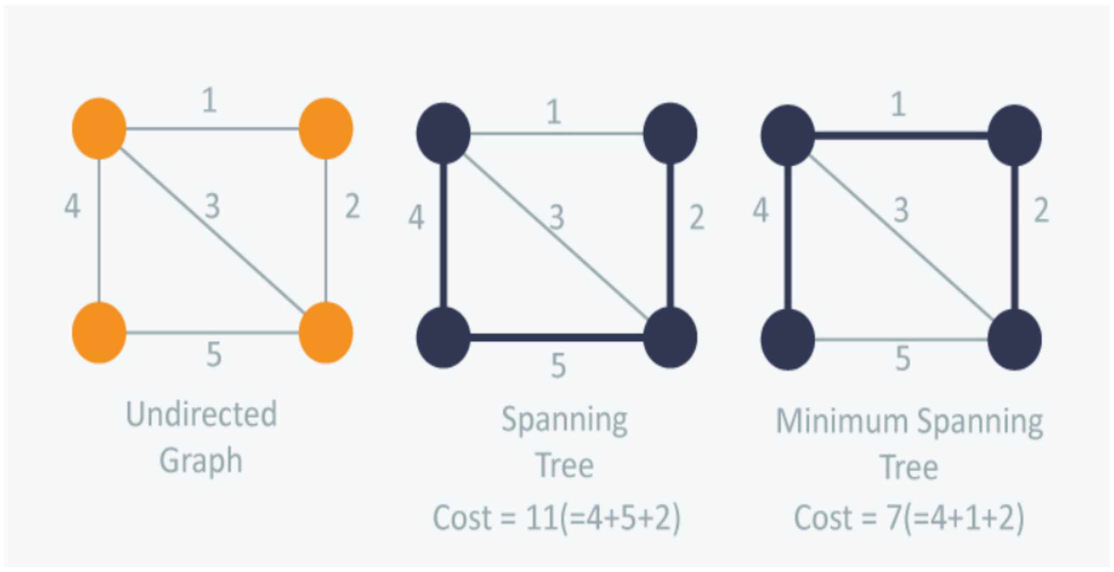
(a) Clusters after removal of longest edges

What is a Minimum Spanning Tree?

The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees.

Minimum spanning tree has direct application in the design of networks. It is used in algorithms approximating the travelling salesman problem, multi-terminal minimum cut problem and minimum-cost weighted perfect matching. Other practical applications are:

1. Cluster Analysis
2. Handwriting recognition
3. Image segmentation



DBSCAN Clustering:**Density-Based Spatial Clustering of Applications with Noise**

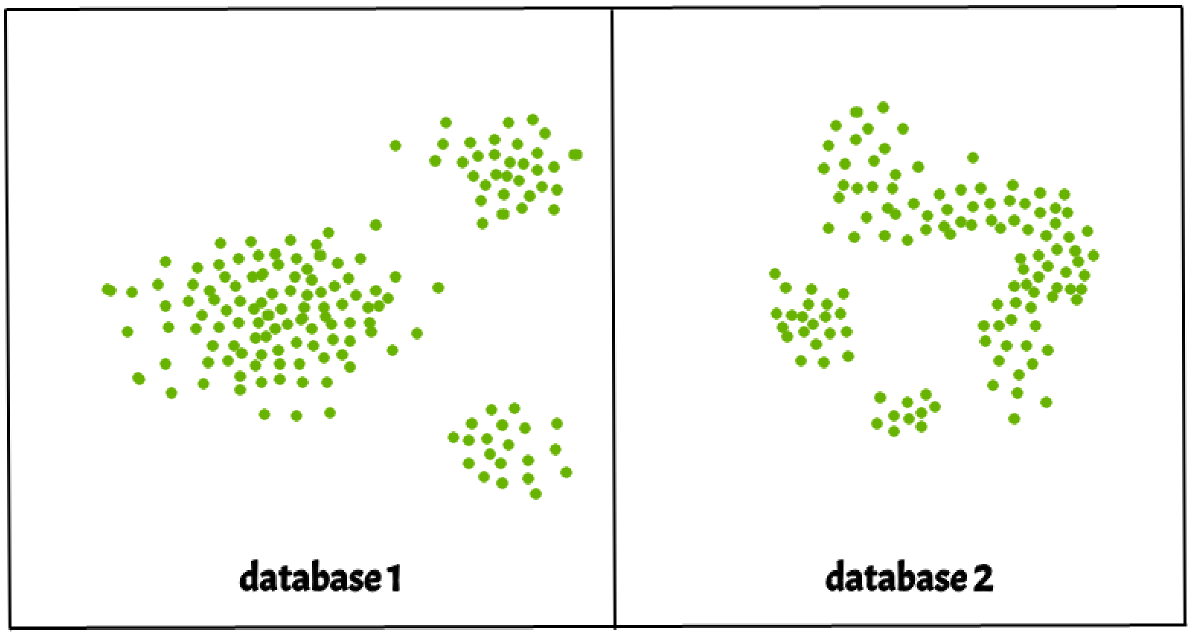
Clustering analysis or simply Clustering is basically an Unsupervised learning method that divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense.

It comprises many different methods based on differential evolution.

E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), Spectral clustering (graph distance) etc.

Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches. Here we will focus on **Density-based spatial clustering of applications with noise** (DBSCAN) clustering method.

Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of "clusters" and "noise". The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

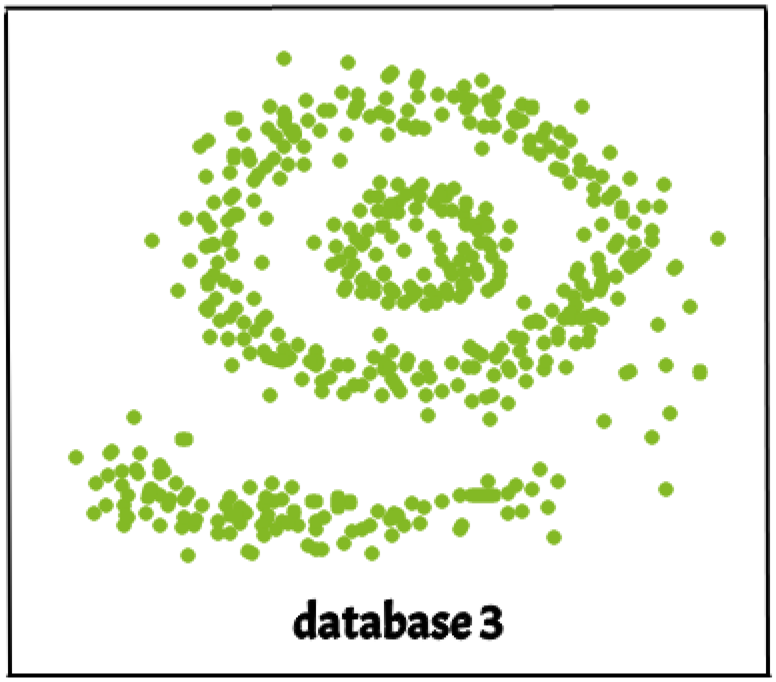


Why DBSCAN?

Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.

Real life data may contain irregularities, like:

1. Clusters can be of arbitrary shape such as those shown in the figure below.
2. Data may contain noise.



The figure below shows a data set containing nonconvex clusters and outliers/noises. Given such data, k-means algorithm has difficulties in identifying these clusters with arbitrary shapes.6

DBSCAN algorithm requires two parameters:

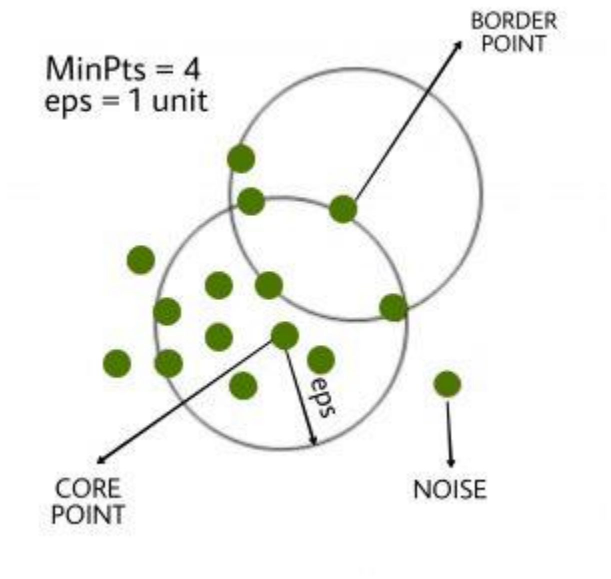
1. **eps** : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters. One way to find the eps value is based on the ***k-distance graph***.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

In this algorithm, we have 3 types of data points.

Core Point*: A point is a core point if it has more than MinPts points within eps.*

Border Point*: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.*

Noise or outlier*: A point which is not a core point or border point.*



DBSCAN algorithm can be abstracted in the following steps:

1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.
   A point *a* and *b* are said to be density connected if there exist a point *c* which has a sufficient number of points in its neighbors and both the points *a* and *b* are within the *eps distance*. This is a chaining process. So, if *b* is neighbor of *c*, *c* is neighbor of *d*, *d* is neighbor of *e*, which in turn is neighbor of *a* implies that *b* is neighbor of *a*.

4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.