UNIT IV

UNSUPERVISED LEARNING

Clustering in Machine Learning

- Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset.
- It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points.
- The objects with the possible similarities remain in a group that has less or no similarities with another group."
- It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.
- It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabelled dataset.
- After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.
- The clustering technique is commonly used for **statistical data analysis**

Example:

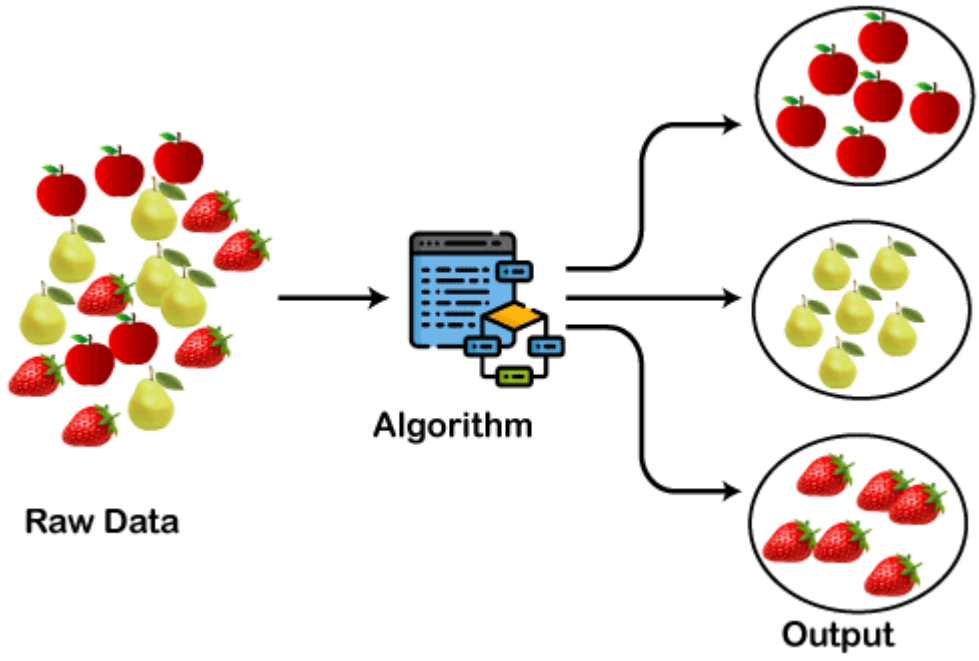
Let's understand the clustering technique with the real-world example of Mall:

- When we visit any shopping mall, we can observe that the things with similar usage are grouped together.
- Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., are grouped in separate sections, so that we can easily find out the things.
- The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

- Apart from these general usages, it is used by the Amazon in its recommendation system to provide the recommendations as per the past search of products.
- Netflix also uses this technique to recommend the movies and web-series to its users as per the watch history.
- The below diagram explains the working of the clustering algorithm.
- We can see the different fruits are divided into several groups with similar properties.



Types of Clustering Methods:

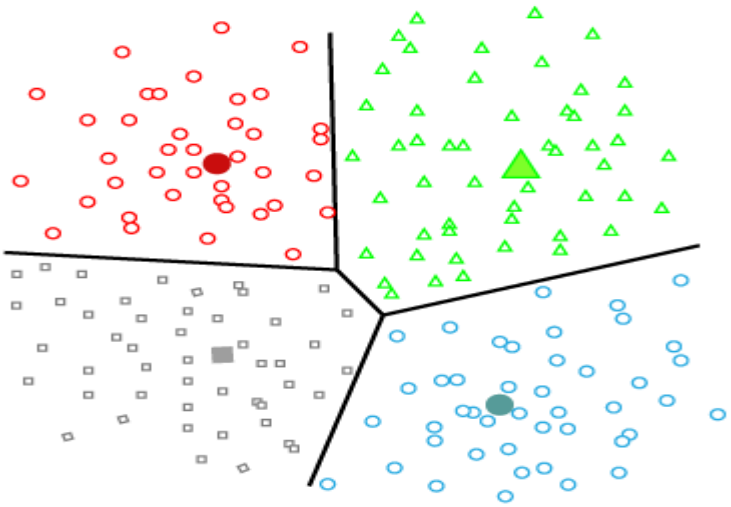
- The clustering methods are broadly divided into Hard clustering (datapoint belongs to only one group) and Soft Clustering (data points can belong to another group also).

But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. Partitioning Clustering
2. Density-Based Clustering
3. Distribution Model-Based Clustering
4. Hierarchical Clustering
5. Fuzzy Clustering

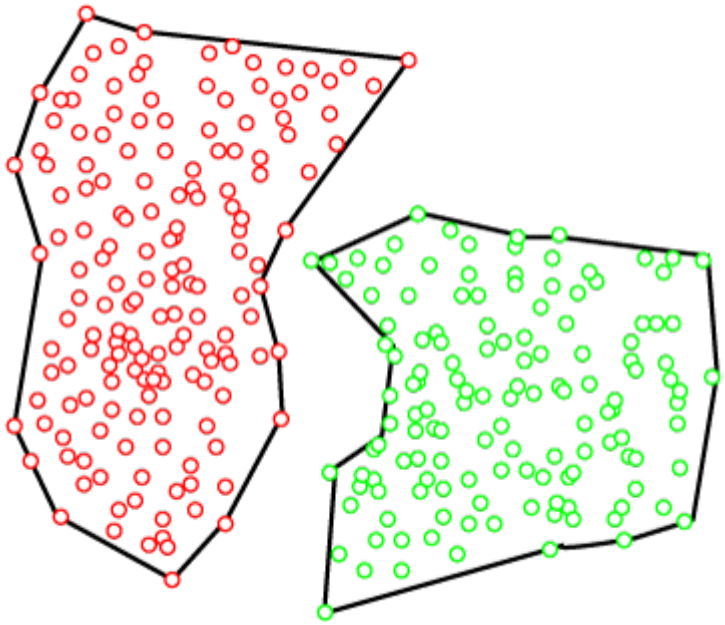
Partitioning Clustering:

- It is a type of clustering that divides the data into non-hierarchical groups.
- It is also known as the centroid-based method.
- The most common example of partitioning clustering is the K-Means Clustering algorithm.
- In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups.
- The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



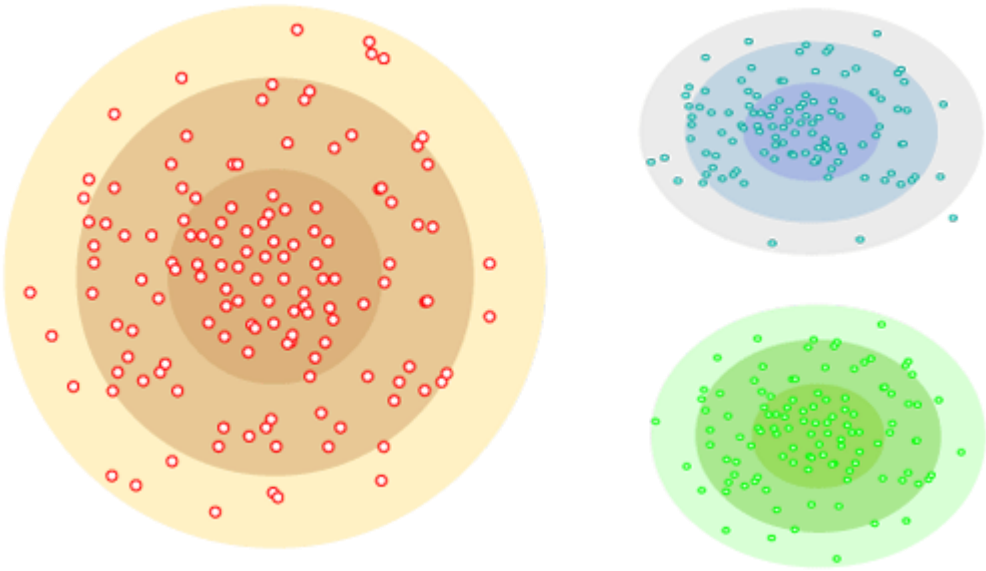
Density-Based Clustering:

- The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected.
- This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters.
- The dense areas in data space are divided from each other by sparser areas.
- These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



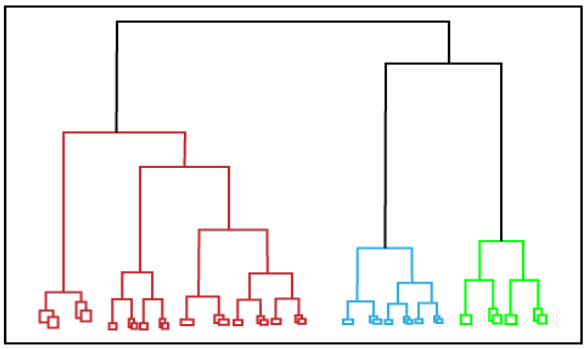
Distribution Model-Based Clustering:

- In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution.
- The grouping is done by assuming some distributions commonly Gaussian Distribution.
- The example of this type is the Expectation-Maximization Clustering algorithm that uses Gaussian Mixture Models (GMM).



Hierarchical Clustering:

- Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created.
- In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a dendrogram.
- The observations or any number of clusters can be selected by cutting the tree at the correct level.
- The most common example of this method is the Agglomerative Hierarchical algorithm.



Fuzzy Clustering:

- Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster.

- Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster.
- Fuzzy C-means algorithm is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

K-Means Clustering Algorithm

- K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.
- In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

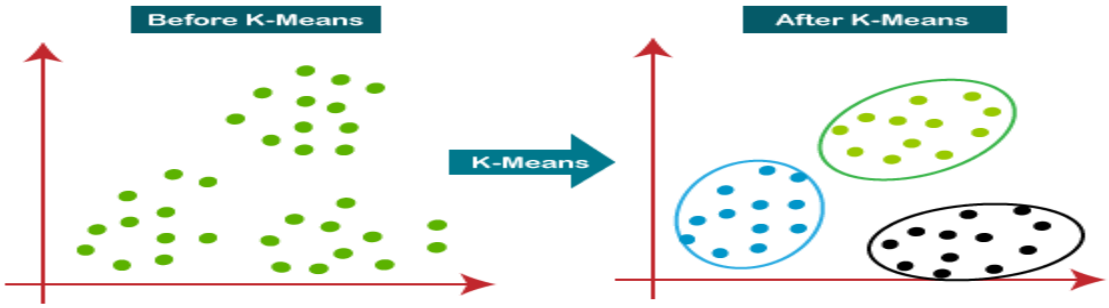
- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It is an iterative algorithm that divides the unlabelled 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 unlabelled 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 unlabelled 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 centre points or centroids by an iterative process.
- Assigns each data point to its closest k-centre. 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.

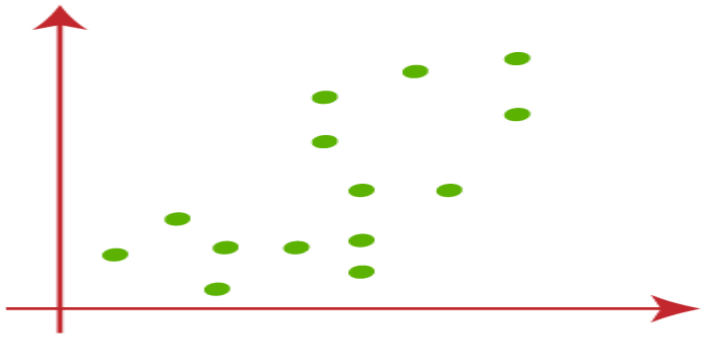
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

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

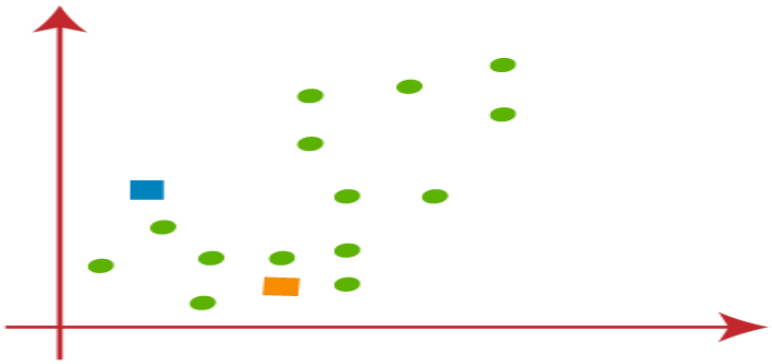
Step-7: The model is ready.

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:

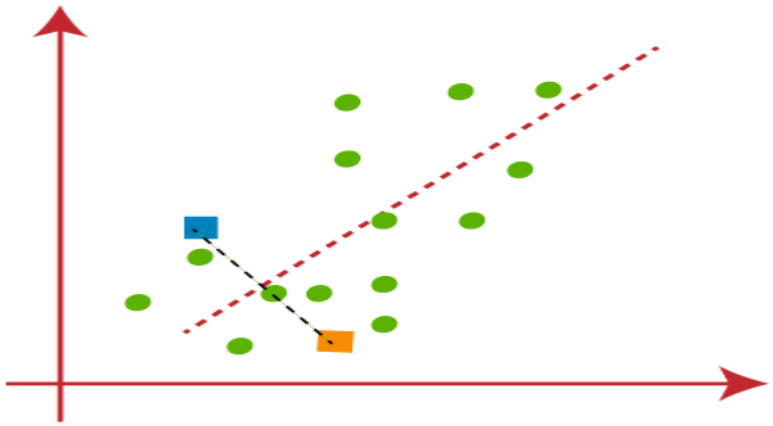


- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points 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:

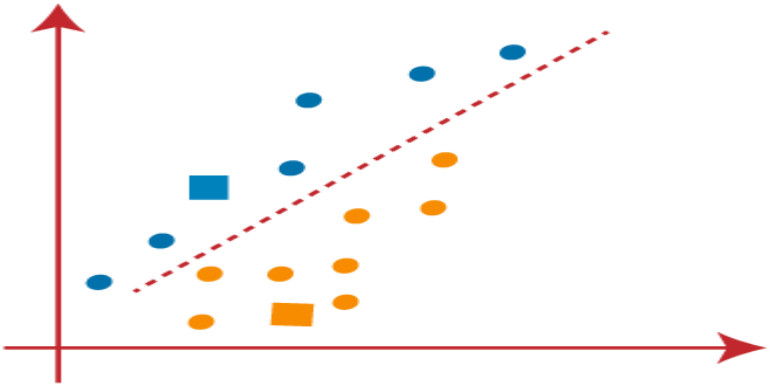


- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both

  the centroids. Consider the below image:

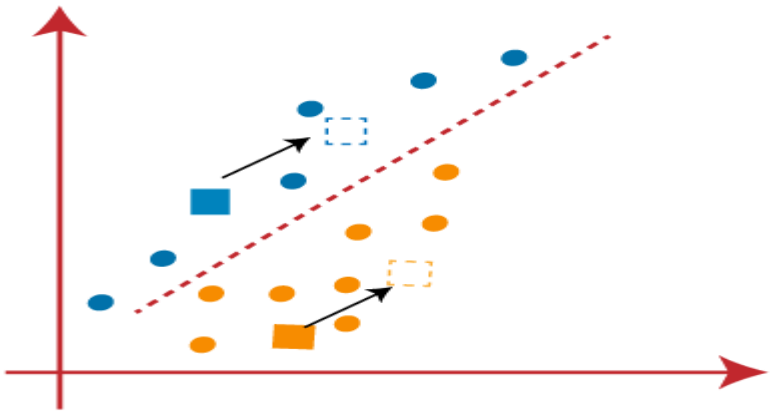


From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.

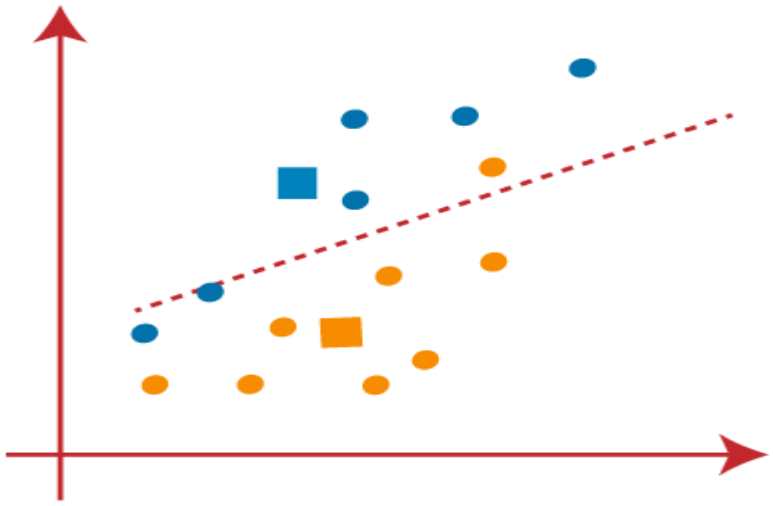


- As we need to find the closest cluster, so we will repeat the process by choosing a new centroid. To choose the new centroids, we will compute the center of gravity of

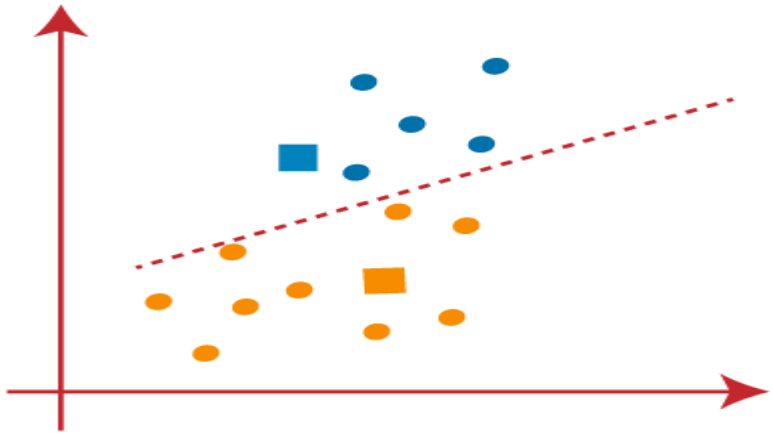
these centroids, and will find new centroids as below:



- 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:

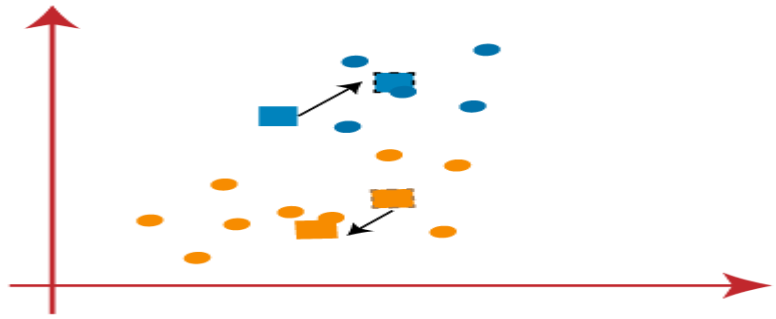


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

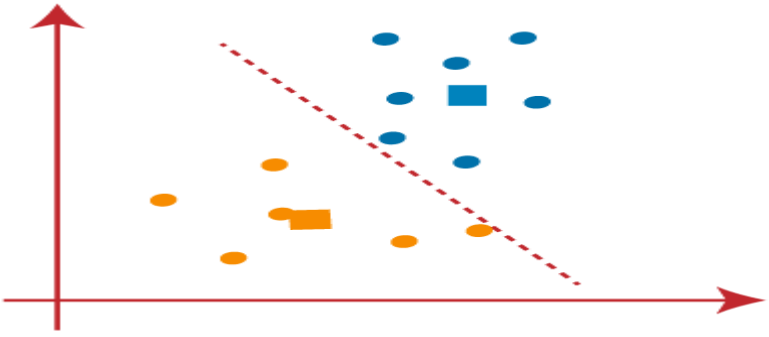


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

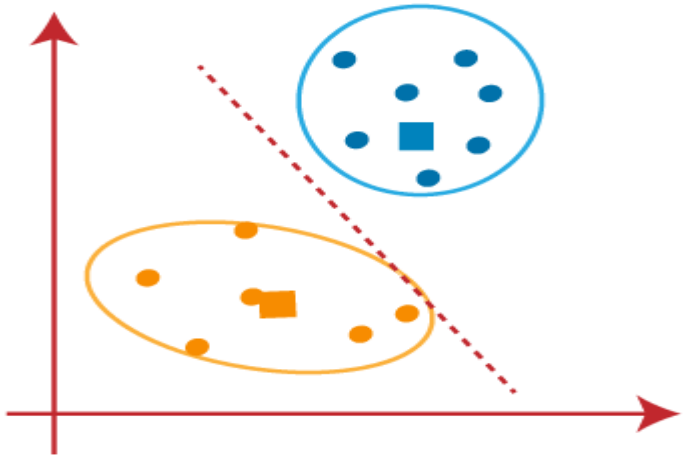
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



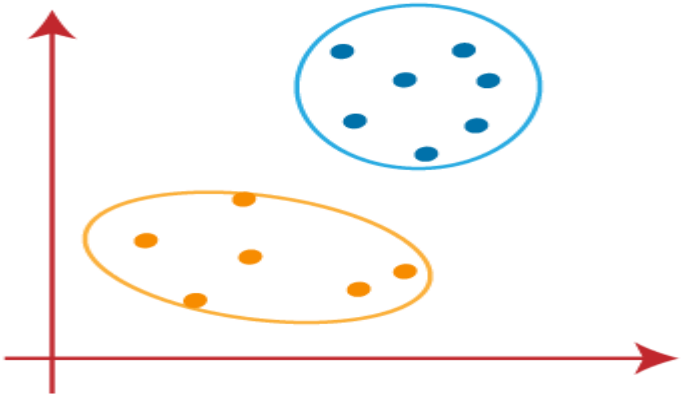
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- 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:



EM Algorithm in Machine Learning

- 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.
- n 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.

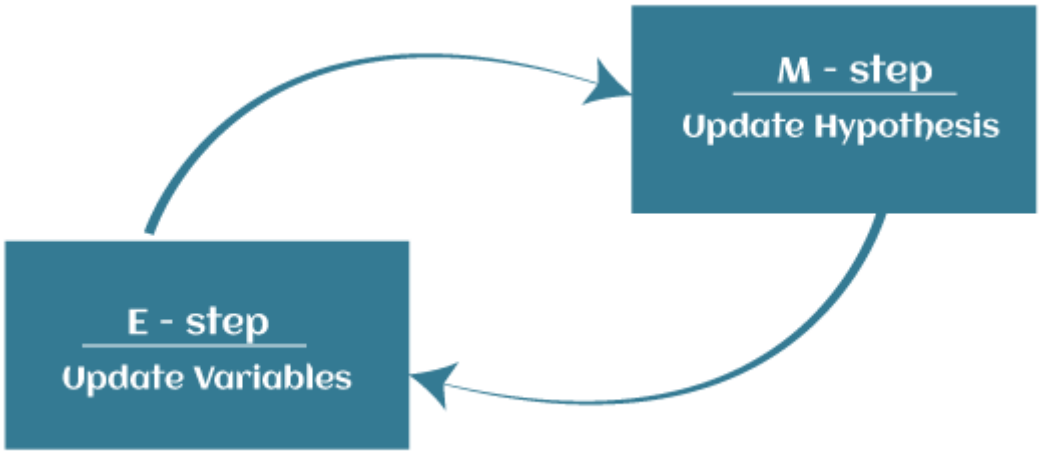
- A latent variable model consists of both observable and unobservable variables where observable can be predicted while unobserved are inferred from the observed variable.
- These unobservable variables are known as latent variables.

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.

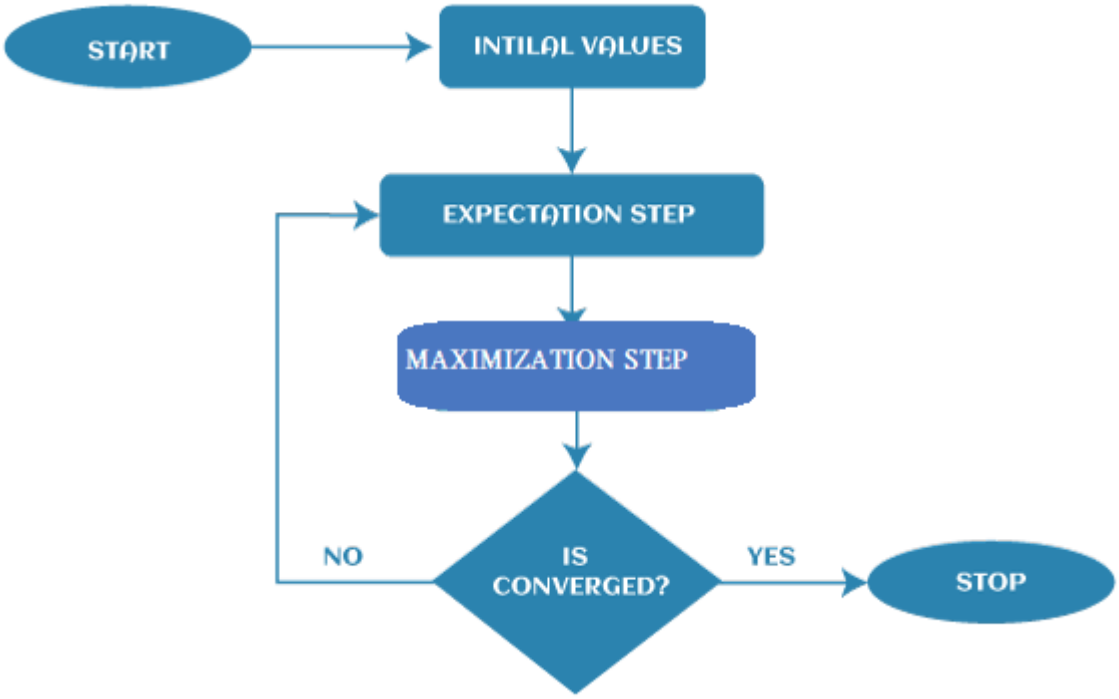
What is Convergence in the EM algorithm?

- 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.

Gaussian Mixture Model (GMM)

- The Gaussian Mixture Model or GMM is defined as a mixture model that has a combination of the unspecified probability distribution function.

- Further, GMM also requires estimated statistics values such as mean and standard deviation or parameters.
- It is used to estimate the parameters of the probability distributions to best fit the density of a given training dataset.
- Although there are plenty of techniques available to estimate the parameter of the Gaussian Mixture Model (GMM), the Maximum Likelihood Estimation is one of the most popular techniques among them.
- Let's understand a case where we have a dataset with multiple data points generated by two different processes.
- However, both processes contain a similar Gaussian probability distribution and combined data. Hence it is very difficult to discriminate which distribution a given point may belong to.
- The processes used to generate the data point represent a latent variable or unobservable data. In such cases, the Estimation-Maximization algorithm is one of the best techniques which helps us to estimate the parameters of the gaussian distributions.
- In the EM algorithm, E-step estimates the expected value for each latent variable, whereas M-step helps in optimizing them significantly using the Maximum Likelihood Estimation (MLE). Further, this process is repeated until a good set of latent values, and a maximum likelihood is achieved that fits the data.

Applications of EM algorithm:

- The primary aim of the EM algorithm is to estimate the missing data in the latent variables through observed data in datasets.

The EM algorithm or latent variable model has a broad range of real-life applications in machine learning. These are as follows:

- The EM algorithm is applicable in data clustering in machine learning.
- It is often used in computer vision and NLP (Natural language processing).
- It is used to estimate the value of the parameter in mixed models such as the Gaussian Mixture Modeland quantitative genetics.
- It is also used in psychometrics for estimating item parameters and latent abilities of item response theory models.

- It is also applicable in the medical and healthcare industry, such as in image reconstruction and structural engineering.
- It is used to determine the Gaussian density of a function.

Advantages of EM algorithm

- It is very easy to implement the first two basic steps of the EM algorithm in various machine learning problems, which are E-step and M- step.
- It is mostly guaranteed that likelihood will enhance after each iteration.
- It often generates a solution for the M-step in the closed form.

Disadvantages of EM algorithm

- The convergence of the EM algorithm is very slow.
- It can make convergence for the local optima only.
- It takes both forward and backward probability into consideration. It is opposite to that of numerical optimization, which takes only forward probabilities.

Hierarchical Clustering in Machine Learning

- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabelled datasets into a cluster and also known as hierarchical cluster analysis or HCA.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.
- Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work.
- As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach.

Why hierarchical clustering?

- As we already have other clustering algorithms such as K-Means Clustering, then why we need hierarchical clustering?
- So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size.
- To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.
- In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

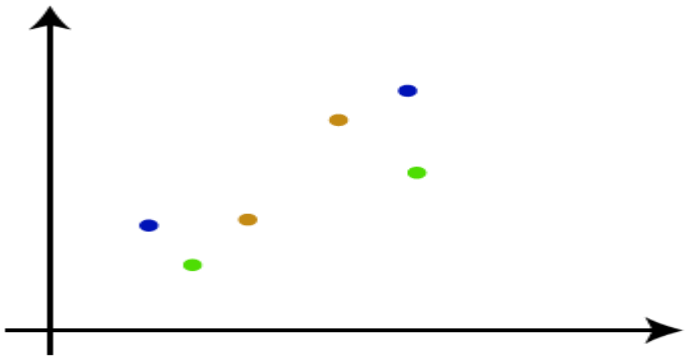
Agglomerative Hierarchical clustering:

- The agglomerative hierarchical clustering algorithm is a popular example of HCA.
- To group the datasets into clusters, it follows the bottom-up approach.
- It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together.
- It does this until all the clusters are merged into a single cluster that contains all the datasets.
- This hierarchy of clusters is represented in the form of the dendrogram.

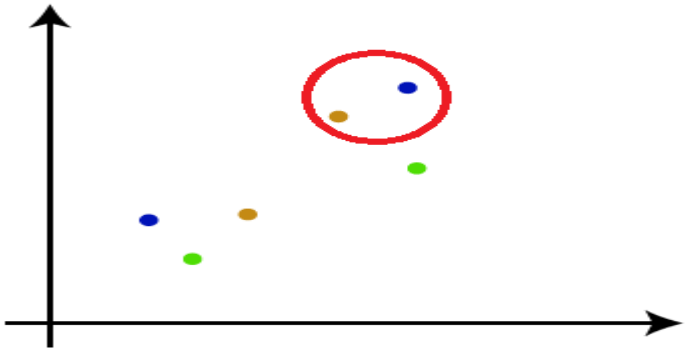
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

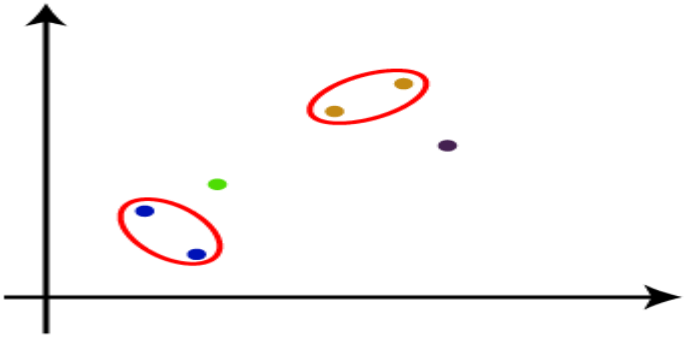
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



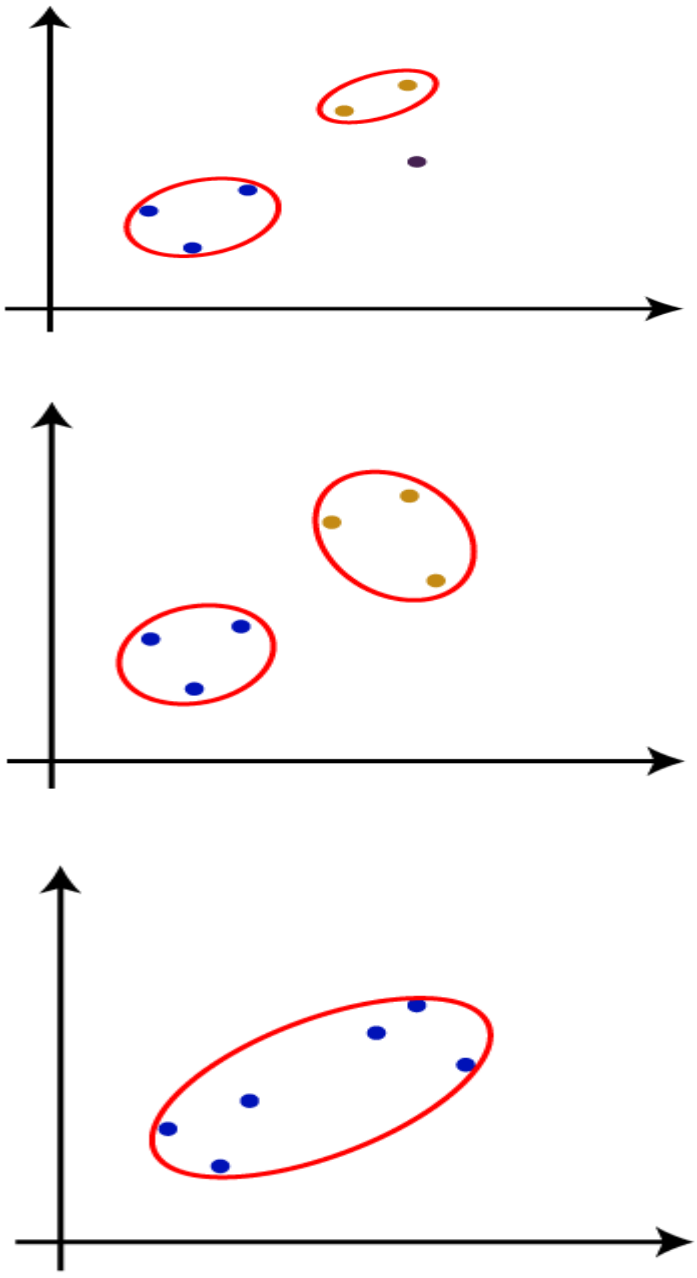
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.



- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:



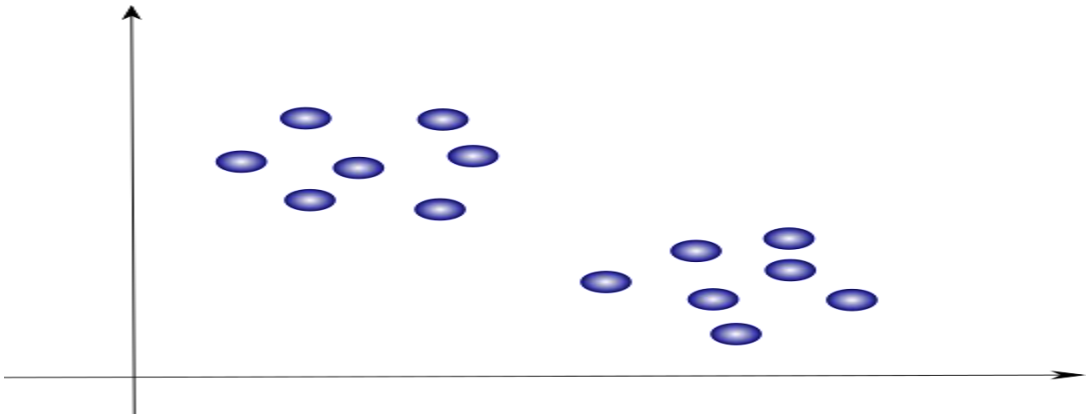
- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

Gaussian Mixture Models

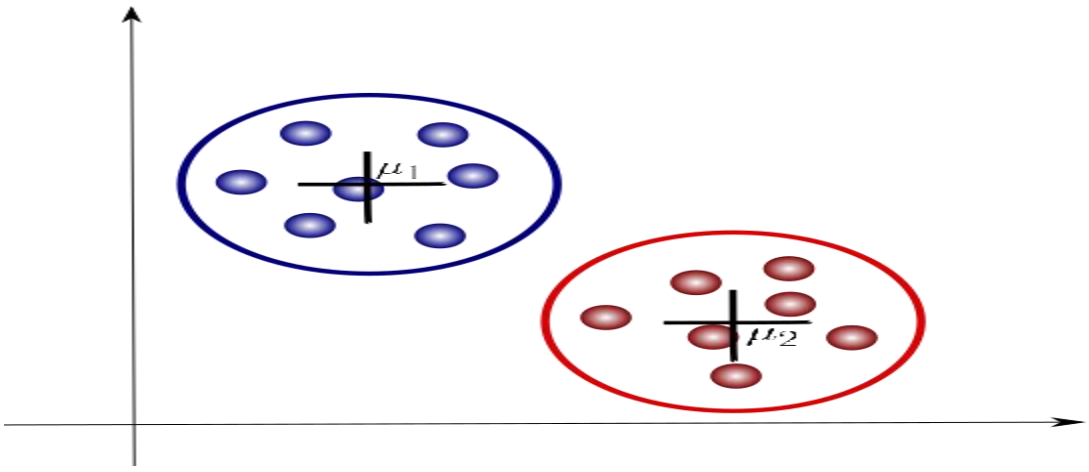
In the world of Machine Learning, we can distinguish two main areas:

- Supervised and unsupervised learning.
- The main difference between both lies in the nature of the data as well as the approaches used to deal with it.
- Clustering is an unsupervised learning problem where we intend to find clusters of points in our dataset that share some common characteristics.

Let's suppose we have a dataset that looks like this:



Our job is to find sets of points that appear close together. In this case, we can clearly identify two clusters of points which we will colour blue and red, respectively:



Please note that we are now introducing some additional notation. Here, $\mu 1$ and $\mu 2$ are the centroids of each cluster and are parameters that identify each of these. A popular clustering algorithm is known as K-means, which will follow an iterative approach to update the parameters of each clusters. More specifically, what it will do is to compute the means (or centroids) of each cluster, and then calculate their distance to each of the data points. The latter are then labeled as part of the cluster that is identified by their closest centroid. This process is repeated until some convergence criterion is met, for example when we see no further changes in the cluster assignments.

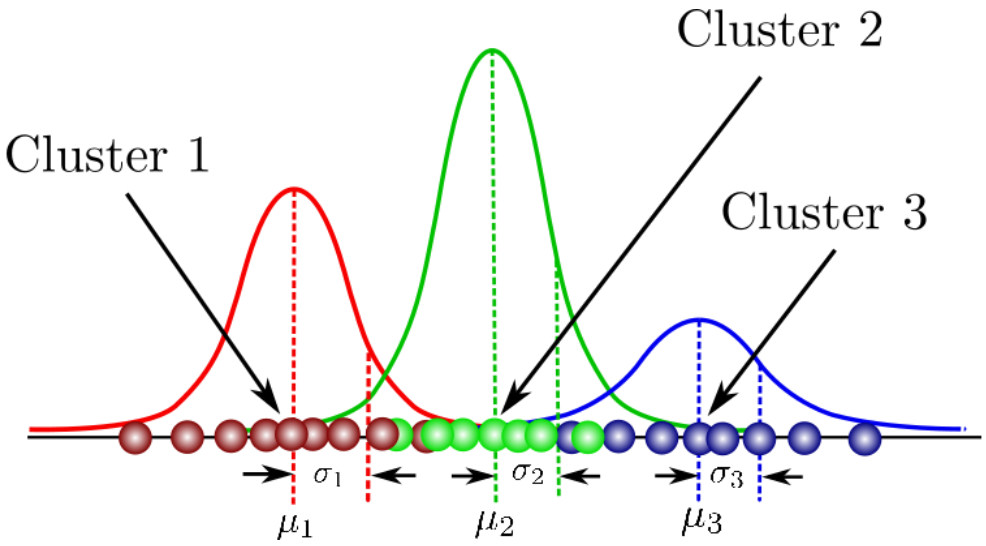
One important characteristic of K-means is that it is a hard clustering method, which means that it will associate each point to one and only one cluster. A limitation to this approach is that there is no uncertainty measure or probability that tells us how much a data point is associated with a specific cluster. So what about using a soft clustering instead of a hard one? This is exactly what Gaussian Mixture Models, or simply GMMs, attempt to do. Let's now discuss this method further.

Definitions

A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by $k \in \{1,\ldots, K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.

Let us now illustrate these parameters graphically:



Here, we can see that there are three Gaussian functions, hence K = 3. Each Gaussian explains the data contained in each of the three clusters available. The mixing coefficients are themselves probabilities and must meet this condition:

$$\sum_{k=1}^{K} \pi_k = 1 \qquad (1)$$

Now how do we determine the optimal values for these parameters? To achieve this we must ensure that each Gaussian fits the data points belonging to each cluster. This is exactly what maximum likelihood does.

In general, the Gaussian density function is given by:

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Where x represents our data points, D is the number of dimensions of each data point. μ and Σ are the mean and covariance, respectively. If we have a dataset comprised of N = 1000 three-dimensional points (D = 3), then x will be a 1000 × 3 matrix. μ will be a 1 × 3 vector, and Σ will be a 3 × 3 matrix. For later purposes, we will also find it useful to take the log of this equation, which is given by:

$$\ln \mathcal{N}(\mathbf{x}|\mu, \Sigma) = -\frac{D}{2} \ln 2\pi - \frac{1}{2} \ln \Sigma - \frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu) \qquad (2)$$

If we differentiate this equation with respect to the mean and covariance and then equate it to zero, then we will be able to find the optimal values for these parameters, and the solutions will correspond to the Maximum Likelihood Estimates (MLE) for this setting. However, because we are dealing with not just one, but many Gaussians, things will get a bit complicated when time comes for us to find the parameters for the whole mixture. In this regard, we will need to introduce some additional aspects that we discuss in the next section.

Initial derivations

We are now going to introduce some additional notation. Just a word of warning. Math is coming on! Don't worry. I'll try to keep the notation as clean as possible for better understanding of the derivations. First, let's suppose we want to know what is the probability that a data point xn comes from Gaussian k. We can express this as:

$$p(z_{nk} = 1|\mathbf{x}_n)$$

Which reads "given a data point x, what is the probability it came from Gaussian k?" In this case, z is a latent variable that takes only two possible values. It is one when x came from Gaussian k, and zero otherwise. Actually, we don't get to see this z variable in reality, but knowing its probability of occurrence will be useful in helping us determine the Gaussian mixture parameters, as we discuss later.

Likewise, we can state the following:

$$\pi_k = p(z_k = 1)$$

Which means that the overall probability of observing a point that comes from Gaussian k is actually equivalent to the mixing coefficient for that Gaussian. This makes sense, because the

bigger the Gaussian is, the higher we would expect this probability to be. Now let z be the set of all possible latent variables z, hence:

$$\mathbf{z} = \{z_1, ..., z_K\}$$

We know beforehand that each z occurs independently of others and that they can only take the value of one when k is equal to the cluster the point comes from. Therefore:

$$p(\mathbf{z}) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} ... p(z_K = 1)^{z_K} = \prod_{k=1}^{K} \pi_k^{z_k}$$

Now, what about finding the probability of observing our data given that it came from Gaussian k? Turns out to be that it is actually the Gaussian function itself! Following the same logic we used to define p(z), we can state:

$$p(\mathbf{x}_n|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)^{z_k}$$

Ok, now you may be asking, why are we doing all this? Remember our initial aim was to determine what the probability of z given our observation x? Well, it turns out to be that the equations we have just derived, along with the Bayes rule, will help us determine this probability. From the product rule of probabilities, we know that

$$p(\mathbf{x}_n, \mathbf{z}) = p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z})$$

Hmm, it seems to be that now we are getting somewhere. The operands on the right are what we have just found. Perhaps some of you may be anticipating that we are going to use the Bayes rule to get the probability we eventually need. However, first we will need p(xn), not p(xn, z). So how do we get rid of z here? Yes, you guessed it right. Marginalization! We just need to sum up the terms on z, hence

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)$$

This is the equation that defines a Gaussian Mixture, and you can clearly see that it depends on all parameters that we mentioned previously! To determine the optimal values for these we need to determine the maximum likelihood of the model. We can find the likelihood as the joint probability of all observations xn, defined by:

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)$$

Like we did for the original Gaussian density function, let's apply the log to each side of the equation:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \qquad (3)$$

Great! Now in order to find the optimal parameters for the Gaussian mixture, all we have to do is to differentiate this equation with respect to the parameters and we are done, right? Wait! Not so fast. We have an issue here. We can see that there is a logarithm that is affecting the second summation. Calculating the derivative of this expression and then solving for the parameters is going to be very hard!

What can we do? Well, we need to use an iterative method to estimate the parameters. But first, remember we were supposed to find the probability of z given x? Well, let's do that since at this point we already have everything in place to define what this probability will look like.

From Bayes rule, we know that

$$p(z_k = 1|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(\mathbf{x}_n|z_j = 1)p(z_j = 1)}$$

From our earlier derivations we learned that:

$$p(z_k = 1) = \pi_k, \qquad p(\mathbf{x}_n|z_k = 1) = \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)$$

So let's now replace these in the previous equation:

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}) \qquad (4)$$

And this is what we've been looking for! Moving forward we are going to see this expression a lot. Next we will continue our discussion with a method that will help us easily determine the parameters for the Gaussian mixture.

Expectation — Maximization algorithm

Well, at this point we have derived some expressions for the probabilities that we will find useful in determining the parameters of our model. However, in the past section we could see that simply evaluating (3) to find such parameters would prove to be very hard. Fortunately, there is an iterative method we can use to achieve this purpose. It is called the Expectation — Maximization, or simply EM algorithm. It is widely used for optimization problems where the objective function has complexities such as the one we've just encountered for the GMM case.

Let the parameters of our model be

$$\theta = \{\pi, \mu, \Sigma\}$$

Let us now define the steps that the general EM algorithm will follow[1].

Step 1: Initialise θ accordingly. For instance, we can use the results obtained by a previous K-Means run as a good starting point for our algorithm.

Step 2 (Expectation step): Evaluate

$$\mathcal{Q}(\theta^*, \theta) = \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z}|\theta^*)] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta) \ln p(\mathbf{X}, \mathbf{Z}|\theta^*) \qquad (5)$$

Well, actually we have already found p(Z|X, θ). Remember the γ expression we ended up with in the previous section? For better visibility, let's bring our earlier equation (4) here:

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}) \qquad (4)$$

For Gaussian Mixture Models, the expectation step boils down to calculating the value of γ in (4) by using the old parameter values. Now if we replace (4) in (5), we will have:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{\mathbf{Z}} \gamma(z_{nk}) \ln p(\mathbf{X}, \mathbf{Z}|\theta^*) \qquad (6)$$

Sounds good, but we are still missing p(X, Z|θ*). How can we find it? Well, actually it's not that difficult. It is just the complete likelihood of the model, including both X and Z, and we can find it by using the following expression:

$$p(\mathbf{X}, \mathbf{Z}|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi^{z_{nk}} \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_{nk}}$$

Which is the result of calculating the joint probability of all observations and latent variables and is an extension of our initial derivations for p(x). The log of this expression is given by

$$\ln p(\mathbf{X}, \mathbf{Z}|\theta^*) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} [\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)] \qquad (7)$$

Nice! And we have finally gotten rid of this troublesome logarithm that affected the summation in (3). With all of this in place, it will be much easier for us to estimate the parameters by just maximizing Q with respect to the parameters, but we will deal with this in the maximization step. Besides, remember that the latent variable z will only be 1 once everytime the summation is evaluated. With that knowledge, we can easily get rid of it as needed for our derivations.

Finally, we can replace (7) in (6) to get:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) [\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)] \qquad (8)$$

In the maximization step, we will find the revised parameters of the mixture. For this purpose, we will need to make Q a restricted maximization problem and thus we will add a Lagrange multiplier to (8). Let's now review the maximization step.

Step 3 (Maximization step): Find the revised parameters θ* using:

$$\theta^* = \arg\max_{\theta} \mathcal{Q}(\theta^*, \theta)$$

Where

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) [\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)] \qquad (8)$$

Which is what we ended up with in the previous step. However, Q should also take into account the restriction that all π values should sum up to one. To do so, we will need to add a suitable Lagrange multiplier. Therefore, we should rewrite (8) in this way:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) [\ln \pi_k + \ln \mathcal{N}(x_n | \mu_k, \Sigma_k)] - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \qquad (8)$$

And now we can easily determine the parameters by using maximum likelihood. Let's now take the derivative of Q with respect to π and set it equal to zero:

$$\frac{\partial \mathcal{Q}(\theta^*, \theta)}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(z_{nk})}{\pi_k} - \lambda = 0$$

Then, by rearranging the terms and applying a summation over k to both sides of the equation, we obtain:

$$\sum_{n=1}^{N} \gamma(z_{nk}) = \pi_k \lambda \implies \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma(z_{nk}) = \sum_{k=1}^{K} \pi_k \lambda$$

From (1), we know that the summation of all mixing coefficients π equals one. In addition, we know that summing up the probabilities γ over k will also give us 1. Thus we get λ = N. Using this result, we can solve for π:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$$

Similarly, if we differentiate Q with respect to μ and Σ, equate the derivative to zero and then solve for the parameters by making use of the log-likelihood equation (2) we defined, we obtain:

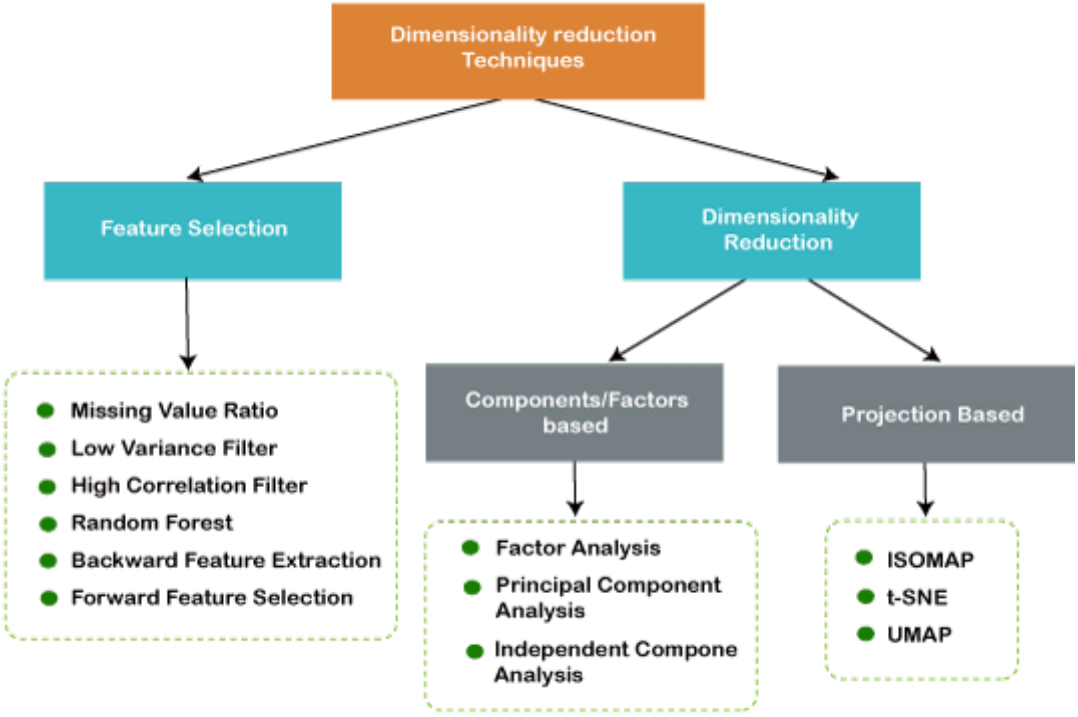
$$\mu_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}^n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \qquad \Sigma_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}^n - \mu_k)(\mathbf{x}^n - \mu_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

And that's it! Then we will use these revised values to determine γ in the next EM iteration and so on and so forth until we see some convergence in the likelihood value. We can use equation (3) to monitor the log-likelihood in each step and we are always guaranteed to reach a local maximum.

Dimensionality Reduction

What is Dimensionality Reduction?

- 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:

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

Linear Discriminant Analysis (LDA) in Machine Learning

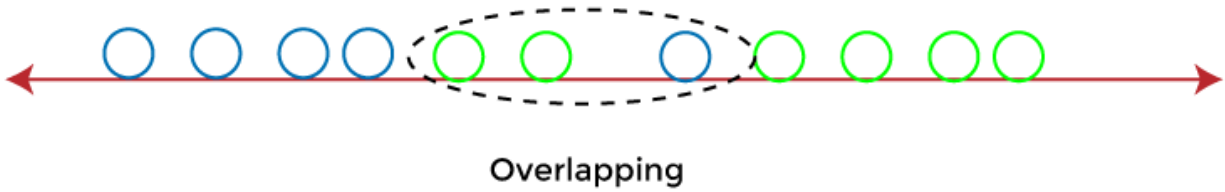
- Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems.
- It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).
- This can be used to project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

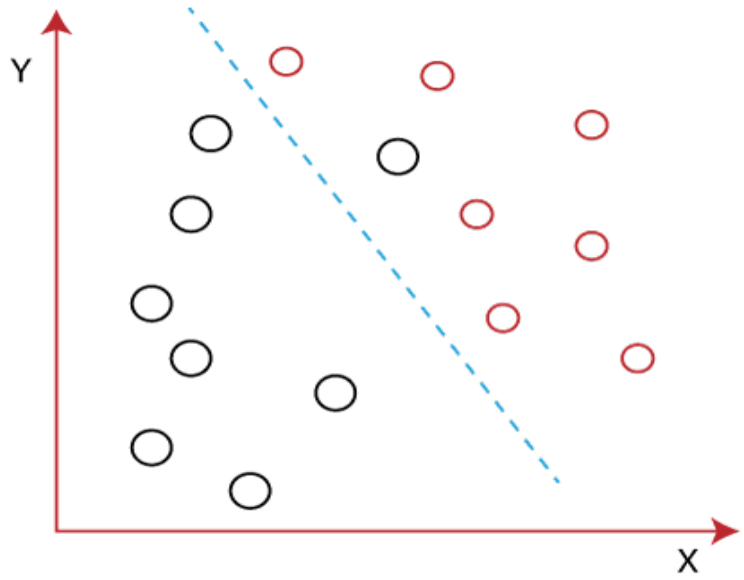
- Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems.
- Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification 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.

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:

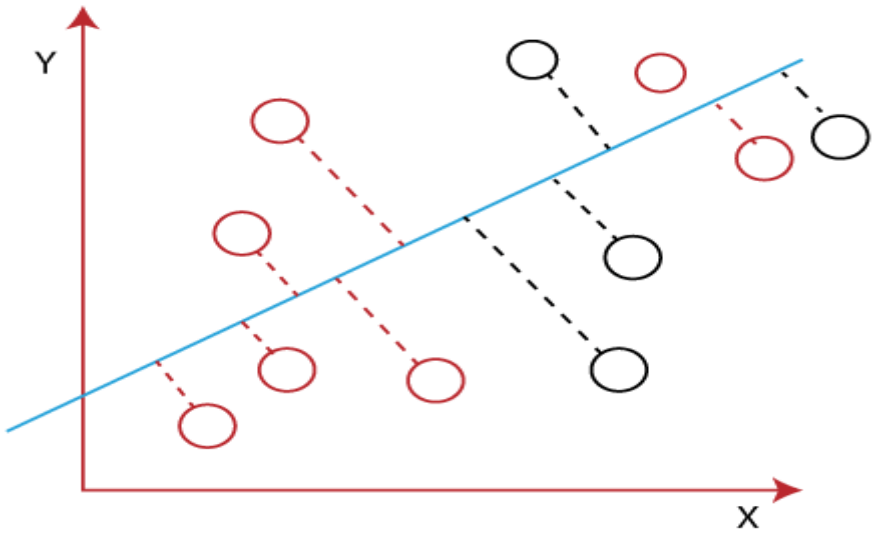


- However, it is impossible to draw a straight line in a 2-d plane that can separate these data points efficiently but using linear Discriminant analysis;
- we can dimensionally reduce the 2-D plane into the 1-D plane. Using this technique, we can also maximize the separability between multiple classes.

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.

Difference between Linear Discriminant Analysis and PCA

Below are some basic differences between LDA and PCA:

- PCA is an unsupervised algorithm that does not care about classes and labels and only aims to find the principal components to maximize the variance in the given dataset. At the same time, LDA is a supervised algorithm that aims to find the linear discriminants to represent the axes that maximize separation between different classes of data.
- LDA is much more suitable for multi-class classification tasks compared to PCA. However, PCA is assumed to be an as good performer for a comparatively small sample size.
- Both LDA and PCA are used as dimensionality reduction techniques, where PCA is first followed by LDA.

How to Prepare Data for LDA

Below are some suggestions that one should always consider while preparing the data to build the LDA model:

- **Classification Problems:** LDA is mainly applied for classification problems to classify the categorical output variable. It is suitable for both binary and multi-class classification problems.
- **Gaussian Distribution:** The standard LDA model applies the Gaussian Distribution of the input variables. One should review the univariate distribution of each attribute and transform them into more Gaussian-looking distributions. For e.g., use log and root for exponential distributions and Box-Cox for skewed distributions.

- **Remove Outliers:** It is good to firstly remove the outliers from your data because these outliers can skew the basic statistics used to separate classes in LDA, such as the mean and the standard deviation.
- **Same Variance:** As LDA always assumes that all the input variables have the same variance, hence it is always a better way to firstly standardize the data before implementing an LDA model. By this, the Mean will be 0, and it will have a standard deviation of 1.

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.
- Factor Analytics is a special technique reducing the huge number of variables into a few numbers of factors is known as factoring of the data, and managing which data is to be present in sheet comes under factor analysis.
- It is completely a statistical approach that is also used to describe fluctuations among the observed and correlated variables in terms of a potentially lower number of unobserved variables called factors.
- The factor analysis technique extracts the maximum common variance from all the variables and puts them into a common score.
- It is a theory that is used in training the machine learning model and so it is quite related to data mining.
- The belief behind factor analytic techniques is that the information gained about the interdependencies between observed variables can be used later to reduce the set of variables in a dataset.
- Factor analysis is a very effective tool for inspecting changeable relationships for complex concepts such as social status, economic status, dietary patterns, psychological scales, biology, psychometrics, personality theories, marketing, product management, operations research, finance, etc.

- It can help a researcher to investigate the concepts that are not easily measured in a much easier and quicker way directly by the cave in a large number of variables into a few easily interpretable fundamental factors.

Types of factor analysis:

1. **Exploratory factor analysis (EFA) :**
    - It is used to identify composite inter-relationships among items and group items that are the part of uniting concepts.
    - The Analyst can't make any prior assumptions about the relationships among factors.
    - It is also used to find the fundamental structure of a huge set of variables.
    - It lessens the large data to a much smaller set of summary variables.
    - It is almost similar to the Confirmatory Factor Analysis(CFA).

    Similarities are:

    - Evaluate the internal reliability of an amount.
    - Examine the factors represented by item sets. They presume that the factors aren't correlated.
    - Investigate the grade/class of each item.

    - However, some common differences, most of them are concerned about how factors are used.
    - Basically, EFA is a data-driven approach, which allows all items to load on all the factors, while in CFA you need to specify which factors are required to load.
    - EFA is really a nice choice if you have no idea about what common factors might exist. EFA is able to generate a huge number of possible models for your data, something which is not possible is, if a researcher has to specify factors.
    - If you have a bit idea about what actually the models look like, and then afterwards you want to test your hypotheses about the data structure, in that case, the CFA is a better approach.

2. **Confirmatory factor analysis (CFA) :**

- It is a more complex(composite) approach that tests the theory that the items are associated with specific factors.
- Confirmatory Factor Analysis uses a properly structured equation model to test a measurement model whereby loading on the factors allows for the evaluation of relationships between observed variables and unobserved variables.
- As we know, the Structural equation modelling approaches can board measurement error easily, and these are much less restrictive than least-squares estimation thus provide more exposure to accommodate errors.
- Hypothesized models are tested against actual data, and the analysis would demonstrate loadings of observed variables on the latent variables (factors), as well as the correlation between the latent variables.
- Confirmatory Factor Analysis allows an analyst and researcher to figure out if a relationship between a set of observed variables (also known as manifest variables) and their underlying constructs exists.
- It is similar to the Exploratory Factor Analysis.

The main difference between the two is:

1. Simply use Exploratory Factor Analysis to explore the pattern.
2. Use Confirmatory Factor Analysis to perform hypothesis testing.

- Confirmatory Factor Analysis provides information about the standard quality of the number of factors that are required to represent the data set.
- Using Confirmatory Factor Analysis, you can define the total number of factors required.
- For example, Confirmatory Factor Analysis is able to answer questions like Does my thousand question survey can able to measure accurately the one specific factor.
- Even though it is technically applicable to any kind of discipline, it is typically used in social sciences.

3. **Multiple Factor Analysis :**

- This type of Factor Analysis is used when your variables are structured in changeable groups.
- For example, you may have a teenager's health questionnaire with several points like sleeping patterns, wrong addictions, psychological health, mobile phone addiction, or learning disabilities.

The Multiple Factor Analysis is performed in two steps which are:-

- Firstly, the Principal Component Analysis will perform on each and every section of the data. Further, this can give a useful eigenvalue, which is actually used to normalize the data sets for further use.
- The newly formed data sets are going to merge into a distinctive matrix and then global PCA is performed.

4. **Generalized Procrustes Analysis (GPA) :**

The Procrustes analysis is actually a suggested way to compare then the two approximate sets of configurations and shapes, which were originally developed to equivalent to the two solutions from Factor Analysis, this technique was actually used to extend the GP Analysis so that more than two shapes could be compared in many ways. The shapes are properly aligned to achieve the target shape. Mainly GPA (Generalized Procrustes Analysis) uses geometric transformations.

Geometric progressions are :

- Isotropic rescaling,
- Reflection,
- Rotation,
- Translation of matrices to compare the sets of data.

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 modelling.
- 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.

Independent Component Analysis

- Independent Component Analysis (ICA) is a statistical and computational technique used in machine learning to separate a multivariate signal into its independent non-Gaussian components.
- ICA assumes that the observed data is a linear combination of independent, non-Gaussian signals.
- The goal of ICA is to find a linear transformation of the data that results in a set of independent components.

1. ICA is a powerful technique used for a variety of applications, such as signal processing, image analysis, and data compression. ICA has been used in a wide range of fields, including finance, biology, and neuroscience.
2. The basic idea behind ICA is to identify a set of basis functions that can be used to represent the observed data. These basis functions are chosen to be statistically independent and non-Gaussian. Once these basis functions are identified, they can be used to separate the observed data into its independent components.
3. ICA is often used in conjunction with other machine learning techniques, such as clustering and classification. For example, ICA can be used to pre-process data before performing clustering or classification, or it can be used to extract features that are then used in these tasks.
   - ICA has some limitations, including the assumption that the underlying sources are non-Gaussian and that they are mixed linearly.
   - Additionally, ICA can be computationally expensive and can suffer from convergence issues if the data is not properly pre-processed.

- Despite these limitations, ICA remains a powerful and widely used technique in machine learning and signal processing

Advantages of Independent Component Analysis (ICA):

1. **Ability to separate mixed signals: ICA** is a powerful tool for separating mixed signals into their independent components. This is useful in a variety of applications, such as signal processing, image analysis, and data compression.
2. **Non-parametric approach:** ICA is a non-parametric approach, which means that it does not require assumptions about the underlying probability distribution of the data.
3. **Unsupervised learning:** ICA is an unsupervised learning technique, which means that it can be applied to data without the need for labeled examples. This makes it useful in situations where labeled data is not available.
4. **Feature extraction:** ICA can be used for feature extraction, which means that it can identify important features in the data that can be used for other tasks, such as classification.

Disadvantages of Independent Component Analysis (ICA):

1. **Non-Gaussian assumption:** ICA assumes that the underlying sources are non-Gaussian, which may not always be true. If the underlying sources are Gaussian, ICA may not be effective.
2. **Linear mixing assumption:** ICA assumes that the sources are mixed linearly, which may not always be the case. If the sources are mixed nonlinearly, ICA may not be effective.
3. **Computationally expensive:** ICA can be computationally expensive, especially for large datasets. This can make it difficult to apply ICA to real-world problems.
4. **Convergence issues:** ICA can suffer from convergence issues, which means that it may not always be able to find a solution. This can be a problem for complex datasets with many sources.

- Independent Component Analysis (ICA) is a machine learning technique to separate independent sources from a mixed signal.
- Unlike principal component analysis which focuses on maximizing the variance of the data points, the independent component analysis focuses on independence, i.e. independent components.

Problem: To extract independent sources' signals from a mixed signal composed of the signals from those sources.

- Independent Component Analysis (ICA) is a technique for separating independent signals from a multi-dimensional signal.
- It is used for signal processing, data analysis, and machine learning applications.
- The goal of ICA is to find a linear transformation of the data such that the transformed data is as close to being statistically independent as possible.

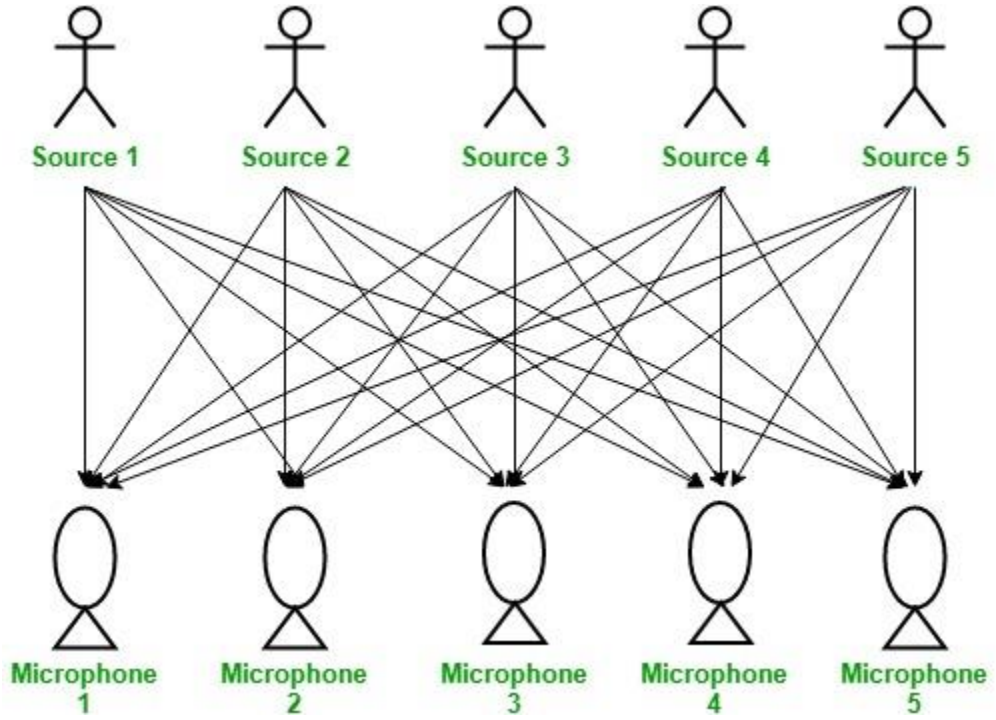
1. The underlying idea of ICA is to find a set of basis functions that are as independent as possible, and to represent the data in terms of these basis functions. The transformed data is then assumed to be statistically independent, and can be used for various applications, such as denoising, feature extraction, and source separation.
2. There are various algorithms that can be used to perform ICA, including FastICA, JADE, and infomax. These algorithms differ in their optimization objectives and the methods used to estimate the independent components.

Given: Mixed signal from five different independent sources.

Aim: To decompose the mixed signal into independent sources:

- Source 1
- Source 2
- Source 3
- Source 4
- Source 5

Solution: Independent Component Analysis (ICA). Consider Cocktail Party Problem or Blind Source Separation problem to understand the problem which is solved by independent component analysis.



Here, There is a party going into a room full of people.

There is 'n' number of speakers in that room and they are speaking simultaneously at the party.

In the same room, there are also 'n' microphones placed at different distances from the speakers which are recording 'n' speakers' voice signals.

Hence, the number of speakers is equal to the number must of microphones in the room. Now, using these microphones' recordings, we want to separate all the 'n' speakers' voice signals in the room given each microphone recorded the voice signals coming from each speaker of different intensity due to the difference in distances between them. Decomposing the mixed signal of each microphone's recording into an independent source's speech signal can be done by using the machine learning technique, independent component analysis. [X1, X2, ….., Xn] => [Y1, Y2, ….., Yn] where, X1, X2, …, Xn are the original signals present

in the mixed signal and Y1, Y2, …, Yn are the new features and are independent components which are independent of each other.

Restrictions on ICA:

1. The independent components generated by the ICA are assumed to be statistically independent of each other.
2. The independent components generated by the ICA must have non-gaussian distribution.
3. The number of independent components generated by the ICA is equal to the number of observed mixtures.

Advantages of Independent Component Analysis (ICA):

1. **Non-Gaussianity:** ICA assumes that the source signals are non-Gaussian, which makes it well-suited for separating signals that are not easily separable by other methods, such as linear regression or PCA.
2. **Blind Source Separation:** ICA is capable of separating signals without any prior knowledge about the sources or their relationships. This is useful in many applications where the sources are unknown, such as in speech separation or EEG signal analysis.
3. **Computationally Efficient:** ICA algorithms are computationally efficient and can be applied to large datasets.
4. **Interpretability:** ICA provides an interpretable representation of the data, where each component represents a single source signal. This can help in understanding the underlying structure of the data and in making informed decisions about the data.

Disadvantages of Independent Component Analysis (ICA):

1. **Non-uniqueness:** There is no unique solution to the ICA problem, and the estimated independent components may not match the true sources. This can lead to suboptimal results or incorrect interpretations.
2. **Non-deterministic:** Some ICA algorithms are non-deterministic, meaning that they can produce different results each time they are run on the same data.

3. **Limitations of Gaussianity:** If the source signals are not non-Gaussian, then ICA may not perform well, and other methods such as PCA or linear regression may be more appropriate.