PATTERN RECOGNITION

MODULE - 1

Pattern is everything around in this digital world. A pattern can either be seen physically or it can be observed mathematically by applying algorithms.

Example: The colours on the clothes, speech pattern, etc. In computer science, a pattern is represented using vector feature values.

What is Pattern Recognition?

Pattern recognition is the process of recognizing patterns by using a machine learning algorithm. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of pattern recognition is its application potential.

Examples: Speech recognition, speaker identification, multimedia document recognition (MDR), automatic medical diagnosis.

In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. Pattern recognition involves the classification and cluster of patterns.

In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated using a set of training patterns or domain knowledge. Classification is used in supervised learning.

Clustering generated a partition of the data which helps decision making, the specific decision-making activity of interest to us. Clustering is used in unsupervised learning.

What is a Feature?

Features may be represented as continuous, discrete, or discrete binary variables. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristics of the object.

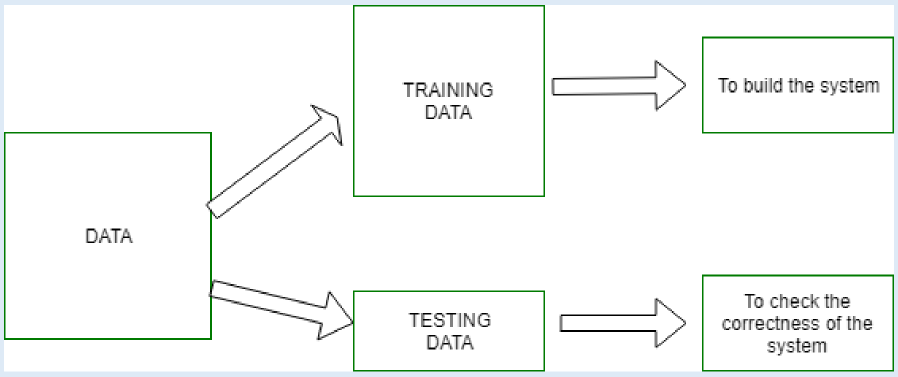
Example: consider our face then eyes, ears, nose, etc are features of the face.
A set of features that are taken together, forms the **features vector**

Training and Learning in Pattern Recognition

Learning is a phenomenon through which a system gets trained and becomes adaptable to give results in an accurate manner. Learning is the most important phase as to how well the system performs on the data provided to the system depends on which algorithms are used on the data. The entire dataset is divided into two categories, one which is used in training the model i.e. Training set, and the other that is used in testing the model after training, i.e. Testing set.

- **Training set:**
  The training set is used to build a model. It consists of the set of images that are used to train the system. Training rules and algorithms are used to give relevant information on how to associate input data with output decisions. The system is trained by applying these algorithms to the dataset, all the relevant information is extracted from the data, and results are obtained. Generally, 80% of the data of the dataset is taken for training data.

- **Testing set:**
  Testing data is used to test the system. It is the set of data that is used to verify whether the system is producing the correct output after being trained or not. Generally, 20% of the data of the dataset is used for testing. Testing data is used to measure the accuracy of the system. For example, a system that identifies which category a particular flower belongs to is able to identify seven categories of flowers correctly out of ten and the rest of others wrong, then the accuracy is 70 %



Advantages:

- Pattern recognition solves classification problems
- Pattern recognition solves the problem of fake biometric detection.
- It is useful for cloth pattern recognition for visually impaired blind people.
- We can recognize particular objects from different angles.

Disadvantages:

- The syntactic pattern recognition approach is complex to implement and it is a very slow process.
- Sometimes to get better accuracy, a larger dataset is required.
- It cannot explain why a particular object is recognized.
- Example: my face vs my friend’s face.

What are the steps of Pattern Recognition?

Pattern recognition involves the process of identifying and classifying patterns or regularities in data. While there can be variations in the specific steps depending on the domain or problem at hand, here are the general steps involved in pattern recognition:

1. **Data Acquisition**: The first step is to gather or acquire the data that contains the patterns to be recognized. This data can come from various sources such as sensors, images, text, or any other form of structured or unstructured data.
2. **Pre-processing (Segmentation):** Once the data is acquired, it often requires pre-processing to clean and enhance it. This step may involve removing noise, normalizing data, scaling, or transforming it to a suitable format for further analysis.
3. **Feature Extraction:** In this step, relevant features or characteristics of the data are extracted to represent the patterns of interest. The choice of features is crucial and depends on the specific problem. Feature extraction techniques can involve statistical methods, signal processing techniques, or domain-specific knowledge.
4. **Feature Selection:** Sometimes, the extracted features may contain redundant or irrelevant information, which can negatively impact the recognition process. Feature selection involves identifying and selecting the most informative and discriminative features for pattern recognition.
5. **Training or Learning**: In this step, a pattern recognition model is trained or learned using a labelled dataset. The model learns the underlying patterns and relationships between the features and the corresponding class labels. Various machine learning algorithms, such as decision trees, support vector machines, or neural networks, can be employed for training.
6. **Classification or Recognition:** After the model has been trained and evaluated, it is ready to be used for pattern recognition on new, unseen data. The model applies the learned patterns and classifies or recognizes the input data into predefined classes or categories.
7. **Post-processing and Decision Making:** In some cases, post-processing steps may be necessary to refine the recognition results. This can involve techniques such as smoothing, filtering, or combining the outputs from multiple classifiers. The final decision or action based on the recognized patterns is made using the processed results.

Applications of Pattern Recognition:

- **Image processing, segmentation, and analysis**
  Pattern recognition is used to give human recognition intelligence to machines that are required in image processing.
- **Computer vision**
  Pattern recognition is used to extract meaningful features from given image/video samples

and is used in computer vision for various applications like biological and biomedical imaging.

- **Seismic analysis**
  The pattern recognition approach is used for the discovery, imaging, and interpretation of temporal patterns in seismic array recordings. Statistical pattern recognition is implemented and used in different types of seismic analysis models.
- **Radar signal classification/analysis**
  Pattern recognition and signal processing methods are used in various applications of radar signal classifications like AP mine detection and identification.
- **Speech recognition**
  The greatest success in speech recognition has been obtained using pattern recognition paradigms. It is used in various algorithms of speech recognition which tries to avoid the problems of using a phoneme level of description and treats larger units such as words as pattern
- **Fingerprint identification**
  Fingerprint recognition technology is a dominant technology in the biometric market. A number of recognition methods have been used to perform fingerprint matching out of which pattern recognition approaches are widely used.

MODULE – 2 (Bayesian Decision Theory)

Bayesian Decision Theory is the statistical approach to pattern classification. It leverages probability to make classifications, and measures the risk (i.e. cost) of assigning an input to a given class. Bayesian Decision Theory makes better predictions by using the prior probability, likelihood probability, and evidence to calculate the posterior probability.

Bayesian Decision Theory (i.e. the Bayesian Decision Rule) predicts the outcome not only based on previous observations, but also by taking into account the current situation. **The rule describes the most reasonable action to take based on an observation**.

The formula for Bayesian (Bayes) decision theory is given below:

$$P(C_i|X) = \frac{P(C_i)P(X|C_i)}{P(X)}$$

The elements of the theory are:

- **P(Ci): Prior probability.** This accounts for how many times the class Ci occurred independently from any conditions (i.e. regardless of the input X).
- **P(X|Ci)): Likelihood.** Under some conditions X, this is how many times the outcome Ci occurred.
- **P(X) Evidence.** The number of times the conditions X occurred.

- **P(Ci|X) Posterior.** The probability that the outcome Ci occurs given some conditions X.

Prior Probability

The probability is calculated according to the past occurrences of the outcomes (i.e. events). This is called the **prior probability** ("prior" meaning "before"). In other words, the prior probability refers to the probability in the past.

Assume that someone asks who will be the winner of a future match between two teams. Let A and B refer to the first or second team winning, respectively.

In the last 10 cup matches, A occurred 4 times and B occurred the remaining 6 times. So, what is the probability that A occurs in the next match? Based on the experience (i.e., the events that occurred in the past), the prior probability that the first team (A) wins in the next match is:

$P(A)$=10/4=0.4

But the past events may not always hold, because the situation or context may change. For example, team A could have won only 4 matches because there were some injured players. When the next match comes, all of these injured players will have recovered. Based on the current situation, the first team may win the next match with a higher probability than the one calculated based on past events only.

The prior probability measures the probability of the next action without taking into consideration a current observation (i.e. the current situation). It's like predicting that a patient has a given disease based only on past doctors visits.

Likelihood Probability

The likelihood helps to answer the question: given some conditions, what is the probability that an outcome occurs? It is denoted as follows:

$P(X|C_i)$

Where X refers to the conditions, and Ci refers to the outcome. Because there may be multiple outcomes, the variable C is given the subscript i.

According to our example of predicting the winning team, the probability that the outcome A occurs does not only depend on past events, but also on current conditions. The likelihood relates the occurrence of an outcome to the current conditions at the time of making a prediction.

Assume the conditions change so that the first team has no injured players, while the second team has many injured players. As a result, it is more likely that A occurs than B. Without considering the current situation and using only the prior information, the outcome would be B, which is not accurate given the current situation.

What is Bayesian Classifier ?

A Bayesian classifier, also known as a Naive Bayes classifier, is a probabilistic machine learning model based on Bayes' theorem. It is a simple and effective classification algorithm that is widely used in pattern recognition and text categorization tasks. The classifier assumes that the features are conditionally independent given the class label, which is known as the "naive" assumption.

The Bayesian classifier calculates the posterior probability of each class given the observed features and selects the class with the highest probability. It makes predictions by combining prior probabilities, likelihoods, and evidence using Bayes' theorem.

Formula: The formula for the Bayesian classifier can be derived from Bayes' theorem:

$P(C \mid x) = (P(x \mid C) * P(C)) / P(x)$

where:

- $P(C \mid x)$ is the posterior probability of class C given the features x.
- $P(x \mid C)$ is the likelihood of observing the features x given class C.
- $P(C)$ is the prior probability of class C.
- $P(x)$ is the evidence or marginal likelihood of observing the features x.

The naive assumption of independence among features given the class allows us to simplify the calculation of the likelihoods as follows:

$P(x \mid C) = P(x_1 \mid C) * P(x_2 \mid C) * \ldots * P(x_n \mid C)$

where $x_1, x_2, \ldots, x_n$ represent the individual features of x.

Working of Bayesian Classifier:

1. Training Phase
2. Calculation of Prior Probabilities:
3. Calculation of Likelihoods
4. Prediction Phase

The Bayesian classifier, also known as the Naive Bayes classifier, has been shown to be an optimal classifier under certain assumptions. It is considered optimal in terms of minimizing the misclassification rate when the assumptions of the model are satisfied. Here are the reasons why the Bayesian classifier is considered optimal

1. **Bayes' Decision Rule:** The Bayesian classifier applies Bayes' decision rule, which is based on the principle of minimizing the expected misclassification rate. It selects the class with the highest posterior probability given the observed features. This decision rule is optimal under the assumption that the misclassification cost is equal for all classes.

2. **Asymptotic Consistency:** As the sample size increases, the Bayesian classifier converges to the true class conditional probability distributions. This property is known as asymptotic consistency. This means that given enough training data, the Bayesian classifier will approach the true underlying distribution and produce increasingly accurate predictions.

DISCRIMINANT FUNCTIONS

Discriminant functions in pattern recognition are mathematical equations used to discriminate or classify patterns into different classes. The specific form of the discriminant function depends on the problem and the underlying assumptions about the data

1. **Linear Discriminant Function (Fisher's LDA):** The linear discriminant function assumes that the decision boundaries between classes are linear. It can be derived using Fisher's linear discriminant analysis (LDA). For a binary classification problem, the linear discriminant function can be represented as:

$g(x) = w^T * x + w_0$

where:

- $g(x)$ is the discriminant function value for pattern x.
- w is the weight vector or coefficients.
- x is the feature vector of the pattern.
- w_0 is the bias or threshold.

2. **Quadratic Discriminant Function:** The quadratic discriminant function relaxes the linearity assumption and allows for quadratic decision boundaries. It can be derived using quadratic discriminant analysis (QDA). For a binary classification problem, the quadratic discriminant function can be represented as:

$g(x) = x^T * W * x + w^T * x + w_0$

where:

- $g(x)$ is the discriminant function value for pattern x.
- W is the quadratic term matrix.
- w is the linear term vector.
- x is the feature vector of the pattern.
- w_0 is the bias or threshold.

3. **Nonlinear Discriminant Function (Kernel SVM):** Nonlinear discriminant functions are used when the decision boundaries between classes are highly nonlinear. Kernel

Support Vector Machines (SVM) are a popular approach for nonlinear classification. The discriminant function for kernel SVM can be represented as:

$$g(x) = \sum \alpha_i y_i K(x_i, x) + b$$

where:

- $g(x)$ is the discriminant function value for pattern x.
- α_i are the support vector coefficients.
- y_i are the class labels.
- $K(x_i, x)$ is the kernel function that measures the similarity between patterns x_i and x.
- b is the bias term.

4. **Probabilistic Discriminant Function (Gaussian Mixture Model):** Probabilistic discriminant functions assign probabilities to each class based on probabilistic models. Gaussian Mixture Models (GMMs) are a common approach for probabilistic classification. The discriminant function for GMM can be represented as the posterior probability of the class given the pattern's features:

$$P(C\_i \mid x) = P(x \mid C\_i) * P(C\_i) / P(x)$$

where:

- $P(C\_i \mid x)$ is the posterior probability of class C_i given the features x.
- $P(x \mid C\_i)$ is the likelihood of observing the features x given class C_i.
- $P(C\_i)$ is the prior probability of class C_i.
- $P(x)$ is the evidence or marginal likelihood of observing the features x.

Decision Surfaces

In pattern recognition, the decision surface refers to the boundary or surface that separates different classes or categories in a feature space. It represents the region where the classifier or decision rule assigns different patterns to different classes based on their observed features. The decision surface can take different forms, such as a line, curve, or manifold, depending on the complexity of the problem and the underlying classifier used. The goal of pattern recognition is to find an optimal decision surface that effectively separates the patterns belonging to different classes, enabling accurate classification of new, unseen patterns.

Discrete and Continuous Feature of Bayes Decision Theory.

In Bayesian decision theory, features are the observed attributes or measurements of a pattern that are used to make a classification decision. Features can be classified into two main types: continuous and discrete.

1. **Continuous Features:** Continuous features are those that can take on any value within a range or interval. They are often represented by real numbers and can have an infinite number of possible values. Examples of continuous features include measurements such as height, weight, temperature, or time.

   In Bayesian decision theory, when dealing with continuous features, probability density functions (PDFs) are used to describe the likelihood of observing a particular feature value given a class. The conditional probability of a feature given a class, P(x | C), is typically modelled using continuous probability distributions such as Gaussian (normal) distributions.

2. **Discrete Features**: Discrete features, on the other hand, are those that can only take on a finite or countable set of values. They are often represented by integers or categories. Examples of discrete features include binary variables (yes/no), categorical variables (color, shape), or nominal variables (class labels)

   In Bayesian decision theory, when dealing with discrete features, probability mass functions (PMFs) are used to describe the probabilities of observing each possible feature value given a class. The conditional probability of a feature given a class, P(x | C), is modelled using these discrete probability distributions.

Explain Normal Distribution with its characteristics

A normal distribution refers to a probability distribution where the values of a random variable are distributed symmetrically. These values are equally distributed on the left and the right side of the central tendency. Thus, a bell-shaped curve is formed.

Normal Distribution Formula :-

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

Where,

• x is the variable

• μ is the mean

• σ is the standard deviation

Characteristics :-

• In a normal distribution, the mean, median and mode are equal.(i.e., Mean = Median= Mode).

• The total area under the curve should be equal to 1.

• The normally distributed curve should be symmetric at the centre.

• The normal distribution should be defined by the mean and standard deviation.

• The normal distribution curve must have only one peak. (i.e., Unimodal)

• The curve approaches the x-axis, but it never touches, and it extends farther away

from the mean

In pattern recognition, the normal density function is often used to model the conditional probability distribution of continuous features given a specific class. By estimating the parameters (μ and σ) from the training data of each class, the normal density function can be used to calculate the likelihood of observing a particular feature value given a class.

Bayesian decision theory utilizes the normal density function as part of the probabilistic modelling to make optimal classification decisions based on observed feature values. By comparing the likelihoods across different classes, the classification decision can be made by selecting the class with the highest probability density for a given feature value.

MODULE – 3 (Parameter Estimation Methods)

Parameter

Parameter is a characteristic or value that defines the behavior or properties of a population or probability distribution. It represents an unknown quantity that we seek to estimate or infer based on observed data. Parameters are used to summarize and describe the underlying properties of a population and are essential in statistical modelling and inference. Common examples of parameters include the population mean, variance, success probability, regression coefficients, and correlation coefficients.

Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a statistical method used to estimate the parameters of a probability distribution or statistical model. It seeks to find the values of the parameters that maximize the likelihood of observing the given data.

The likelihood function represents the probability of observing the given data for different values of the parameters. MLE aims to find the parameter values that make the observed data most probable. In other words, it finds the parameter values that maximize the likelihood of observing the data.

Example:

Suppose you are interested in estimating the success probability (p) of a biased coin. You have a dataset consisting of 100 coin tosses, where the coin landed on heads (success) 70 times and tails (failure) 30 times.

To estimate the success probability (p), you can use MLE. In this case, you assume that the coin tosses are independent and identically distributed (i.i.d.) Bernoulli trials.

The likelihood function, denoted as L(p), represents the probability of observing the given dataset for a specific value of p. Since the coin tosses are independent**, the likelihood can**

be calculated as the product of the probabilities of each individual outcome. In this example, the likelihood function for the given dataset is:

L(p) = p^70 * (1-p)^30

To find the maximum likelihood estimate (MLE) of p, you seek the value of p that maximizes the likelihood function L(p).

To simplify the calculations, it is often convenient to work with the logarithm of the likelihood function, known as the log-likelihood function. Taking the logarithm allows for easier computation and does not change the location of the maximum point. So, taking the logarithm of the likelihood function, we get:

ln(L(p)) = 70 * ln(p) + 30 * ln(1-p)

To find the MLE, we differentiate the log-likelihood function with respect to p, set the derivative to zero, and solve for p:

d/dp [ln(L(p))] = 70/p - 30/(1-p) = 0

Solving the equation yields:

70/(p*(1-p)) - 30/(1-p) = 0

Rearranging and simplifying further, we find:

70 - 100p = 0

p = 70/100 = 0.7

Therefore, the maximum likelihood estimate (MLE) for the success probability of the biased coin, based on the given dataset, is p = 0.7.

Gaussian Mixture Model

Gaussian Mixture Model or Mixture of Gaussian as it is sometimes called, is not so much a model as it is a probability distribution. It is a universally used model for generative unsupervised learning or clustering. It is also called Expectation-Maximization Clustering or EM Clustering and is based on the optimization strategy. Gaussian Mixture models are used for representing Normally Distributed subpopulations within an overall population. The advantage of Mixture models is that they do not require which subpopulation a data point belongs to. It allows the model to learn the subpopulations automatically. This constitutes a form of unsupervised learning.

A Gaussian distribution is a type of distribution where half of the data falls on the left of it, and the other half of the data falls on the right of it. It’s an even distribution, and one can notice just by the thought of it intuitively that it is very mathematically convenient.

1. **Single Gaussian Distribution:** The normal density function, also known as the Gaussian distribution, is often represented as:

f(x | μ, σ^2) = (1 / sqrt(2πσ^2)) * exp(-(x - μ)^2 / (2σ^2))

where:

- f(x | μ, σ^2) represents the probability density function (PDF) of the normal distribution.
- x is the random variable.
- μ is the mean of the distribution.
- σ^2 is the variance of the distribution.

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

2. **Mixture of Gaussian Distributions:** A Gaussian Mixture Model extends the single Gaussian distribution to a mixture of multiple Gaussian distributions. Instead of having a single mean and variance, we have multiple means (μ) and variances (σ^2), each associated with a weight or mixing coefficient (π) that represents the proportion of the data belonging to that component.

The density function of a GMM is given by:

P(x) = Σ [π_k * N(x | μ_k, Σ_k)]

where:

- P(x) is the probability density of observing data point x.
- Σ represents the summation over all K components of the mixture.
- π_k represents the weight or mixing coefficient of the k-th Gaussian component. The weights satisfy the condition Σ π_k = 1, and they determine the relative contribution of each component to the overall mixture.
- N(x | μ_k, Σ_k) is the probability density function of the k-th Gaussian component, characterized by its mean vector μ_k and covariance matrix Σ_k.

The term N(x | μ_k, Σ_k) represents the probability density function of a single Gaussian distribution, given by:

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

3. **Estimating Parameters:** To estimate the parameters of a GMM, including the weights (π_k), means (μ_k), and variances (σ_k^2), the Expectation-Maximization (EM) algorithm is commonly used. The EM algorithm iteratively updates the parameters to maximize the likelihood of the observed data under the GMM.
4. **Inference and Applications:** Once the GMM is trained, it can be used for various tasks such as clustering, density estimation, and generating new data samples. The GMM can assign probabilities or posterior probabilities to new data points, indicating the likelihood of belonging to each component. This allows for tasks like clustering data points into different groups based on their highest probabilities.

Applications

GMM is widely used in the field of signal processing.

GMM provides good results in language Identification.

Customer Churn is another example.

GMM founds its use case in Anomaly Detection.

GMM is also used to track the object in a video frame.

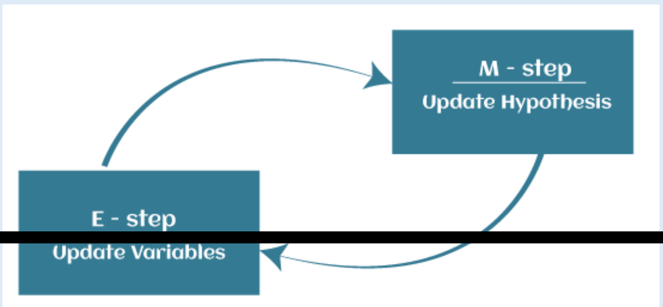
GMM can also be used to classify songs based on genres.

What is an Expectation Maximization 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.

- **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.

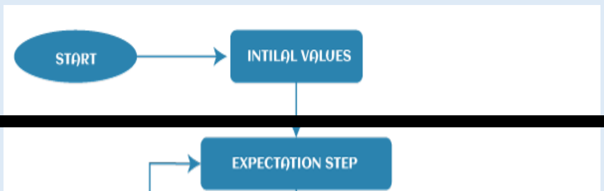
What is Convergence in the EM algorithm?

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.

The EM algorithm consists of two steps: the E-step (Expectation step) and the M-step (Maximization step). Here's an overview of how the algorithm works:

1. **Initialization:** Start by initializing the parameters of the model. This could involve randomly assigning initial values or using some other informed initialization strategy.
2. **E-step (Expectation step):** In this step, the algorithm computes the expected values of the latent variables based on the current parameter estimates. It calculates the posterior probabilities or responsibilities of the hidden variables given the observed data and the current parameter values.
3. **M-step (Maximization step):** Using the expected values of the latent variables obtained from the E-step, the algorithm updates the model parameters to maximize the expected log-likelihood. It finds the parameter values that maximize the log-likelihood of the complete data (both observed and latent variables).
4. **Iteration:** Steps 2 and 3 are repeated iteratively until convergence is achieved. Convergence is typically determined by checking the change in log-likelihood or the change in parameter values between iterations. The algorithm continues until the change falls below a predefined threshold or until a maximum number of iterations is reached.
5. **Output:** After convergence, the algorithm outputs the estimated parameters of the model, which are expected to provide a maximum likelihood estimation given the available data and the underlying model assumptions.

The EM algorithm is particularly useful in cases where there are missing or unobserved data points or when the likelihood function is not directly maximizable due to the presence of hidden variables. It is widely used in various areas of machine learning, statistics, and data analysis, including Gaussian Mixture Models, hidden Markov models, and many other models involving latent variables.



How EM related to 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 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 Model** and quantitative genetics.

Bayesian Estimation

Bayesian estimation in pattern recognition refers to a statistical approach for estimating unknown parameters or making predictions based on prior knowledge and observed data. It involves using Bayes' theorem to update prior beliefs or knowledge about the parameters of interest with new evidence provided by the observed data.

Bayesian estimation involves updating prior beliefs about the parameters of interest using observed data. It employs Bayes' theorem to calculate the posterior distribution of the parameters given the data. Here's the formula and an example to illustrate Bayesian estimation:

Formula: The posterior distribution of the parameters θ given the observed data D can be calculated using Bayes' theorem as follows:

$P(\theta \mid D) = (P(D \mid \theta) * P(\theta)) / P(D)$

where:

- $P(\theta \mid D)$ is the posterior distribution of the parameters θ given the data D.
- $P(D \mid \theta)$ is the likelihood function, representing the probability of observing the data D given the parameters θ.
- $P(\theta)$ is the prior distribution, representing the initial beliefs or knowledge about the parameters before observing the data.
- $P(D)$ is the marginal likelihood or evidence, representing the probability of observing the data regardless of the specific parameter values

MODULE – 4 (Hidden Markov models for sequential pattern classification)

Hidden Markov Model (HMM) is a statistical model that is used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states. It is often used in situations where the underlying system or process that generates the observations is unknown or hidden, hence it got the name “Hidden Markov Model.”

It is used to predict future observations or classify sequences, based on the underlying hidden process that generates the data.

An HMM consists of two types of variables: hidden states and observations.

- The **hidden states** are the underlying variables that generate the observed data, but they are not directly observable.
- The **observations** are the variables that are measured and observed.

The relationship between the hidden states and the observations is modelled using a probability distribution. The Hidden Markov Model (HMM) is the relationship between the hidden states and the observations using two sets of probabilities: the transition probabilities and the emission probabilities.

- The **transition probabilities** describe the probability of transitioning from one hidden state to another.
- The **emission probabilities** describe the probability of observing an output given a hidden state

What are Hidden Markov Models?

A Hidden Markov Model (HMM) is a statistical model used to describe systems that are assumed to be Markovian, meaning that the current state of the system depends only on the previous state and not on any earlier states. It is called "hidden" because the true states of the system are not directly observable, but are inferred from observable outputs or observations.

- A set of N hidden states, S = {s1, s2, ..., sN}.
- A set of M observations, O = {o1, o2, ..., oM}.
- An initial state probability distribution, ? = {?1, ?2, ..., ?N}, which specifies the probability of starting in each hidden state.
- A transition probability matrix, A = [aij], defines the probability of moving from one hidden state to another.
- An emission probability matrix, B = [bjk], defines the probability of emitting an observation from a given hidden state.

The basic idea behind an HMM is that the hidden states generate the observations, and the observed data is used to estimate the hidden state sequence. This is often referred to as the **forward-backwards algorithm.**

Let's consider an example of weather prediction using an HMM. Suppose we are interested in predicting the weather conditions for a particular location, and we can observe whether it is raining or sunny each day. We assume that the weather can be in one of two states: "rainy" or "sunny". However, we do not have direct access to the true weather state; instead, we only observe whether it is wet or dry on a given day.

In this example, the hidden states of the HMM are the weather conditions ("rainy" or "sunny"), and the observed states are the wet or dry observations. The HMM consists of two key components: the transition probabilities and the emission probabilities.

1. **Transition probabilities:** These represent the probabilities of transitioning from one weather state to another. For example, the probability of transitioning from a rainy day to another rainy day could be 0.7, while the probability of transitioning from a sunny day to a rainy day could be 0.4.
2. **Emission probabilities:** These represent the probabilities of observing a particular output (wet or dry) given the current weather state. For instance, the probability of

observing a wet day when it is raining might be 0.8, while the probability of observing a wet day when it is sunny might be 0.2.

Given this setup, we can use the HMM to make predictions about the weather. Let's say we start with a sunny day and want to predict the weather for the next three days. We can use the transition probabilities to calculate the probabilities of transitioning from one state to another, and the emission probabilities to calculate the probabilities of observing wet or dry days.

Discrete Hidden Markov Model.

A Discrete Hidden Markov Model (DHMM) is a specific type of Hidden Markov Model where both the states and observations are discrete variables. It is used to model and analyze sequential data, where the underlying states generating the observed data are not directly observable.

In a DHMM, the states and observations take on a finite number of distinct values. The model assumes that the system being modelled exhibits the Markov property, meaning that the probability of transitioning to a particular state only depends on the current state and not on the history of previous states.

The basic components of a DHMM are as follows:

1. **States:** These are the hidden or unobserved variables that generate the observed data. States are represented by discrete symbols or labels. For example, in speech recognition, the states could represent different phonemes.
2. **Observations:** These are the visible or observable variables that we can directly measure. Observations are emitted by the states but do not directly reveal the state itself. Observations are also represented by discrete symbols or labels. In speech recognition, the observations could be the acoustic features of the spoken words.
3. **Transition probabilities:** These define the probabilities of transitioning from one state to another. They capture the likelihood of moving from one state to another at any given time step. Transition probabilities are typically represented by a matrix, where each element represents the probability of transitioning from one state to another.
4. **Emission probabilities:** These specify the probabilities of emitting particular observations from each state. Each state has its own set of probabilities associated with the different possible observations it can generate. Emission probabilities are also represented by a matrix, where each element represents the probability of emitting a specific observation given a state.

Given a DHMM, the goal is to estimate or infer the most likely sequence of hidden states given a sequence of observations. This can be achieved using algorithms such as the Viterbi algorithm, which calculates the most probable state sequence based on the transition and emission probabilities.

For example, the states could represent weather conditions like "sunny," "cloudy," or "rainy," and the observations could be whether a person is carrying an umbrella or not.

The discrete hidden markov model can be trained using algorithms like expectation-maximization (EM) algorithm or forward-backward algorithm.

Continuous density hidden Markov model

Continuous Density Hidden Markov Model (HMM) is a statistical model that is used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states where both the states and observations are continous in nature.

It is called "hidden" because we cannot directly observe the underlying states but can only observe the outcomes or observations associated with those states.

For example, the state could represent marks of a student and the observation could be whether the cut off marks to be released for selection.

The CDHMM can be trained using algorithms like expectation-maximization (EM) algorithm.

Applications of HMM:-

1. Speech Recognition
2. Natural Language Processing
3. Bioinformatics
4. Finance

Limitations of HMM:-

1. Limited Modeling Capabilities-

One of the key limitations of HMMs is that they are relatively limited in their modelling capabilities. HMMs are designed to model sequences of data, where the underlying structure of the data is represented by a set of hidden states.

MODULE – 5 (DIMESION REDUCTION TECHNIQUE)

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 modelling 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."***

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.

Benefits of applying Dimensionality Reduction

- 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.
- **I**t removes the redundant features (if present) by taking care of multicollinearity.

Approaches of Dimension Reduction

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

Feature Selection

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

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**

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.

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:

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. Fisher Linear Discriminant Analysis

c. Kernel PCA

d. Quadratic Discriminant Analysis

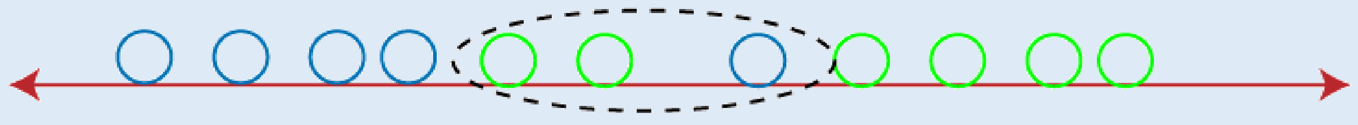
Linear Discriminant Analysis (LDA)

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.

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 modelling 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.

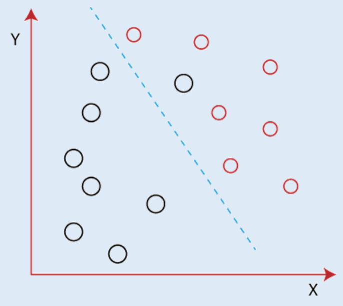


Overlapping

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:



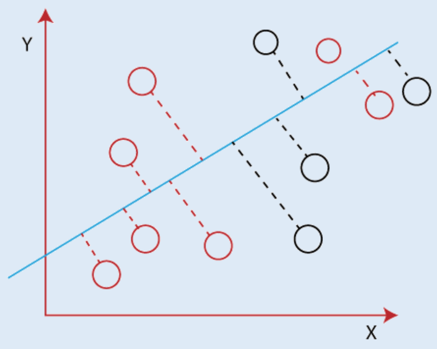
However, it is impossible to draw a straight line in a 2-d plane that can separate these data points efficiently but using linear Discriminant analysis; we can dimensionally reduce the 2-D plane into the 1-D plane. Using this technique, we can also maximize the separability between multiple classes.

<u>How Linear Discriminant Analysis (LDA) works?</u>

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.

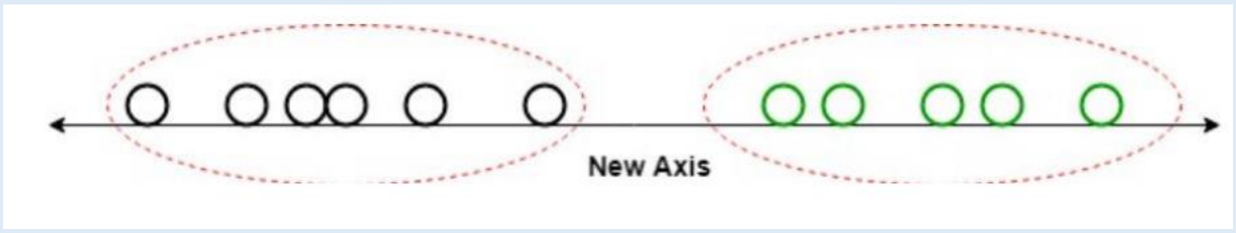


To create a new axis, Linear Discriminant Analysis uses the following criteria:

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

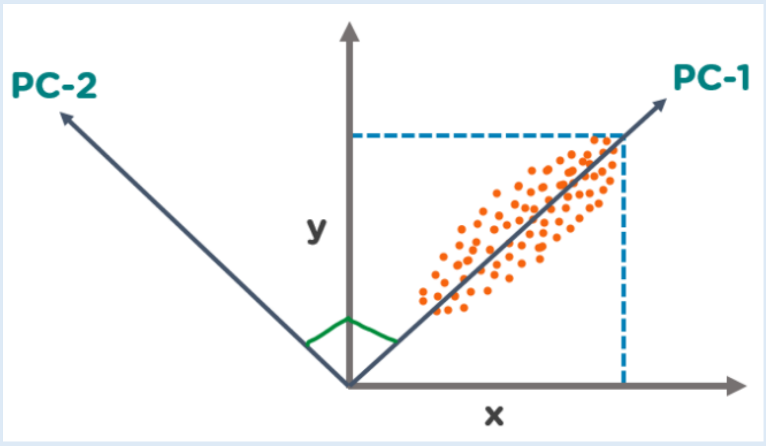
Using the above two conditions, LDA generates a new axis in such a way that it can maximize the distance between the means of the two classes and minimizes the variation within each class.

In the above graph, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each class. In simple terms, this newly generated axis increases the separation between the data points of the two classes. After generating this new axis using the above mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in the figure given below.



PRINCIPAL COMPONENT ANALYSIS (PCA)

The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss. It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables.



In the above figure, we have several points plotted on a 2-D plane. There are two principal components. PC1 is the primary principal component that explains the maximum variance in the data. PC2 is another principal component that is orthogonal to PC1.

What is a Principal Component?

The Principal Components are a straight line that captures most of the variance of the data. They have a direction and magnitude. Principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space.

Dimensionality

The term "dimensionality" describes the quantity of features or variables used in the research. It can be difficult to visualize and interpret the relationships between variables when dealing with high-dimensional data, such as datasets with numerous variables. While reducing the number of variables in the dataset, dimensionality reduction methods like PCA are used to preserve the most crucial data.

Correlation

A statistical measure known as correlation expresses the direction and strength of the linear connection between two variables. The covariance matrix, a square matrix that displays the pairwise correlations between all pairs of variables in the dataset, is calculated in the setting of PCA using correlation.

Steps for PCA Algorithm

1. **Standardize the data:** PCA requires standardized data, so the first step is to standardize the data to ensure that all variables have a mean of 0 and a standard deviation of 1.
2. **Calculate the covariance matrix:** The next step is to calculate the covariance matrix of the standardized data. This matrix shows how each variable is related to every other variable in the dataset.
3. **Calculate the eigenvectors and eigenvalues:** The eigenvectors and eigenvalues of the covariance matrix are then calculated. The eigenvectors represent the directions in which the data varies the most, while the eigenvalues represent the amount of variation along each eigenvector.
4. **Choose the principal components:** The principal components are the eigenvectors with the highest eigenvalues. These components represent the directions in which the data varies the most and are used to transform the original data into a lower-dimensional space.
5. **Transform the data:** The final step is to transform the original data into the lower-dimensional space defined by the principal components.

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.

MODULE – 6 (NON-PARAMETRIC ESTIMATION TECHNIQUE FOR DENSITY ESTIMATION)

Non-parametric density estimation techniques are statistical methods used to estimate the probability density function (PDF) of a random variable without making strong assumptions about the underlying distribution. Unlike parametric methods that assume a specific functional form for the distribution, non-parametric techniques aim to estimate the density directly from the data.

Difference between Parametric and Non-Parametric Methods

Parametric Methods: The basic idea behind the parametric method is that there is a set of fixed parameters that uses to determine a probability model that is used in Machine Learning as well. Parametric methods are those methods for which we priory knows that the population is normal, or if not then we can easily approximate it using a normal distribution which is possible by invoking the Central Limit Theorem. Parameters for using the normal distribution is as follows:

- Mean
- Standard Deviation

Parameter estimation: Parametric methods estimate the parameters of the chosen distribution, such as mean, variance, or shape parameters, from the data. Maximum likelihood estimation or Bayesian inference is commonly used for parameter estimation.

Advantages:

Parametric methods are computationally efficient as they involve estimating a small number of parameters.

With well-specified distributions, parametric methods can provide accurate density estimates even with limited data.

Parametric models provide interpretable parameters that can be useful for understanding the underlying data distribution.

Nonparametric Methods: Non-parametric methods make minimal assumptions about the underlying data distribution. They do not assume any specific functional form for the PDF, allowing for greater flexibility.

Estimation from data: Non-parametric methods estimate the density directly from the data, without assuming a specific distribution. The density estimate is constructed based on the observed data points or their distances.

Advantages:

Non-parametric methods can handle complex and unknown data distributions, as they do not rely on any predefined functional form.

They can capture intricate patterns and variations in the data, making them more suitable for diverse datasets.

Non-Parameter Estimation : Density Estimation

- Density Estimation is a Non-Parameter Estimation technique which is used to determine the probability density function for a randomly chosen variable among a data set.
- The idea of calculating unknown probability density function can be done by:

$$P = \int_R P(x)\,dx$$

where,

“x” Denotes sample data i.e. x1, x2, x3,..,xn on region R.

P(X) denotes the estimated density and P denotes the average estimated density.

- In order to calculate probability density estimation on sample data “x”, it can be achieved by:

$$P_K = \binom{n}{K} P^K (1-P)^{n-K} \quad , \quad \text{where} \quad \binom{n}{K} = \frac{n!}{K!(n-K)!}$$

- Histogram is one of the simplest way used for density estimation.
- Other approaches used for non-parametric estimation of density are:
    - Parzen Windows.
    - K-nearest Neighbor**.**

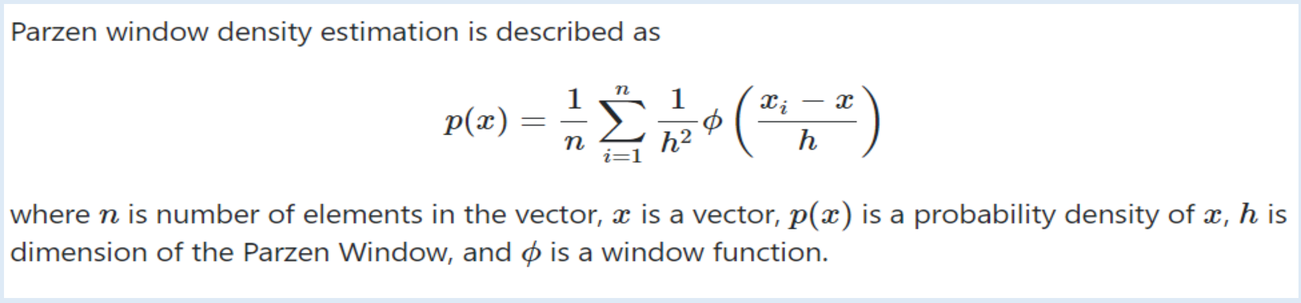
PARZEN WINDOW

Parzen Window is a non-parametric density estimation technique. The Parzen window is a mathematical concept used in statistics and pattern recognition to estimate the probability distribution of a set of data points. It helps us understand how likely it is for a new data point to belong to a certain category or group.

The parzen windows classification algorithm does not require any training phase and the lake of sparseness in the data set makes the testing phase quite slower. Parzen window can be regarded as the generalization of k-nearest neighbor technique. Rather than choosing the k nearest neighbor of the test point and labelling the test point with the weighted majority class, one can consider all points voting scheme and assign the weight by means of the kernel function

- Parzen windows is considered to be a classification technique used for non-parameter estimation technique.
- Generalized version of k-nearest neighbour classification technique can be called as Parzen windows.

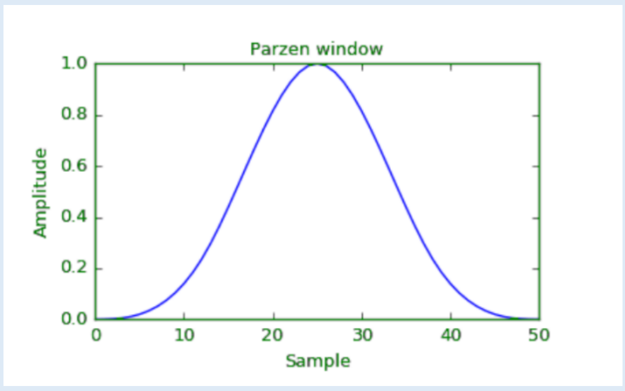
- Parzen Windows algorithm is based upon the concept of support vector machines and is considered to be extremely simple to implement.
- Parzen Windows works on the basis of considering all sample points of given sample data based on scheme of voting and assigning weights w.r.t the kernel function. It does not consider the neighbors and labelled weights.
- Also, it does not requires any training data as it can affect the speed of operation.



Parzen window density estimation is described as

$$p(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^2}\phi\left(\frac{x_i - x}{h}\right)$$

where n is number of elements in the vector, x is a vector, $p(x)$ is a probability density of x, h is dimension of the Parzen Window, and ϕ is a window function.



K-Nearest Neighbour method:-

KNN is a non-parametric density estimation technique. It is a supervised learning technique. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

The K-NN working can be explained on the basis of the below algorithm:-

Step-1: Select the number K of the neighbors

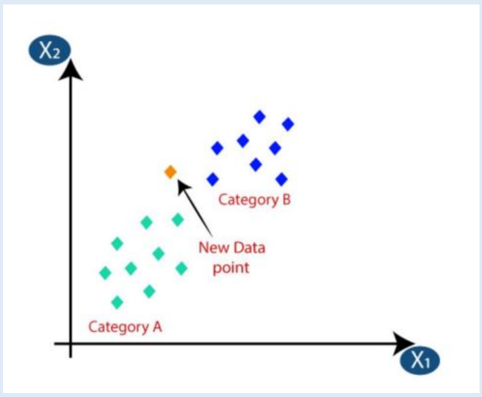
Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

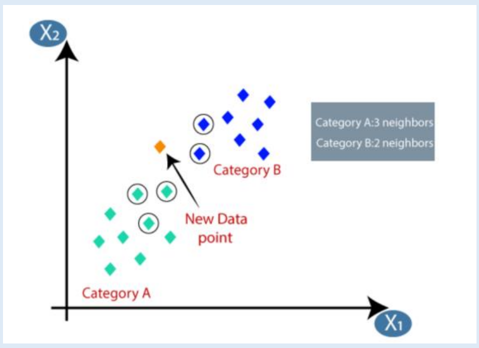
Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready. Suppose we have a new data point and we need to put it in the required category. Consider the below image:



Firstly, we will choose the number of neighbors, so we will choose the k=5. Next, we will calculate the Euclidean distance between the data points. By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A

MODULE – 7 (Linear discriminant function based classifier)

Perceptron:- Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations. Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks.

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components.

These are as follows:

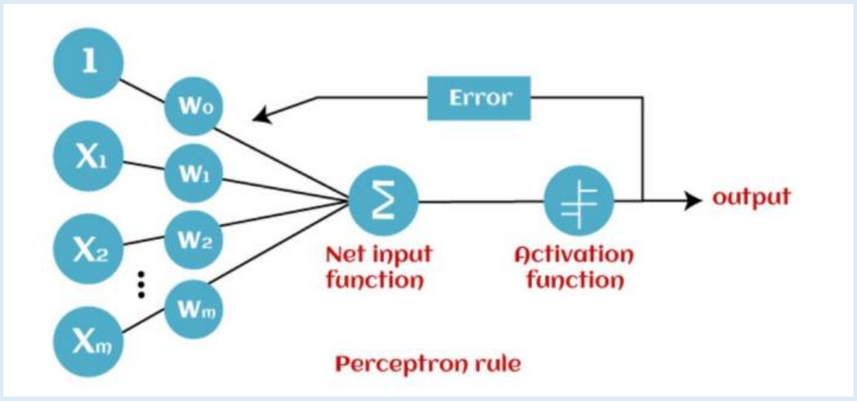
1. Input Nodes or Input Layer:- This is the primary component of Perceptron which accepts the initial data into the system for further processing.

2. Weight and Bias:- Weight parameter represents the strength of the connection between units. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

3. Activation function:-These are the final and important components that help to determine whether the neuron will fire or not.

Types of Activation functions:

- Sign function
- Step function, and
- Sigmoid function



How does Perceptron work?

In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f

Perceptron model works in two important steps as follows:

Step-1- In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum.

Mathematically, we can calculate the weighted sum as follows:

$\sum wi*xi = x1*w1 + x2*w2 + \ldots wn*xn$

Add a special term called bias 'b' to this weighted sum to improve the model's performance. $\sum wi*xi + b$

Step-2- In the second step, an activation function is applied with the above-

mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

$Y = f(\sum wi*xi + b)$

Perceptron Function:- Perceptron function ''f(x)'' can be achieved as output by multiplying the input 'x' with the learned weight coefficient 'w'.

Mathematically, we can express it as follows:

f(x)=1; if w.x+b>0

otherwise, f(x)=0

'w' represents real-valued weights vector

'b' represents the bias

'x' represents a vector of input x values

Characteristics of Perceptron:-

1. Perceptron is a machine learning algorithm for supervised learning of binary classifiers.

2. In Perceptron, the weight coefficient is automatically learned.

3. If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

Limitations of Perceptron Model:-

1. The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.

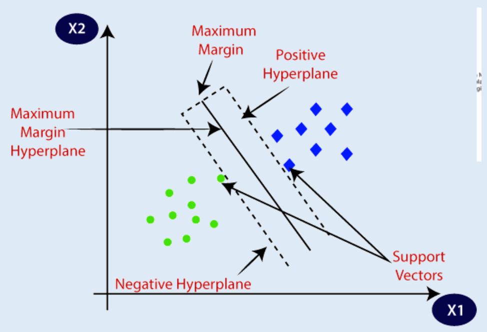
2. Perceptron can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them properly.

Support Vector Machine Algorithm

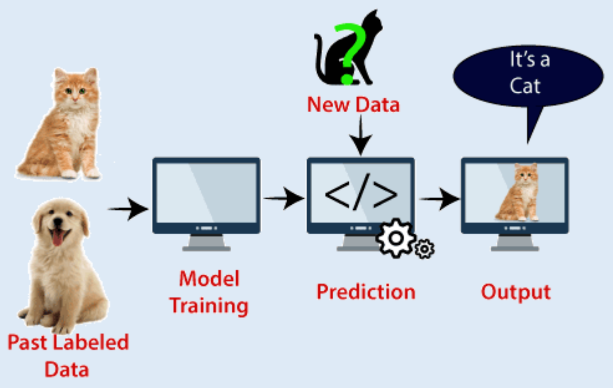
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram

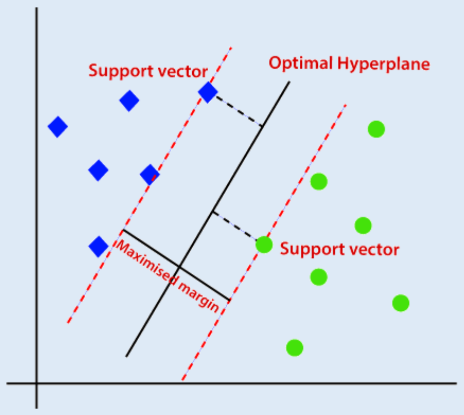


SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

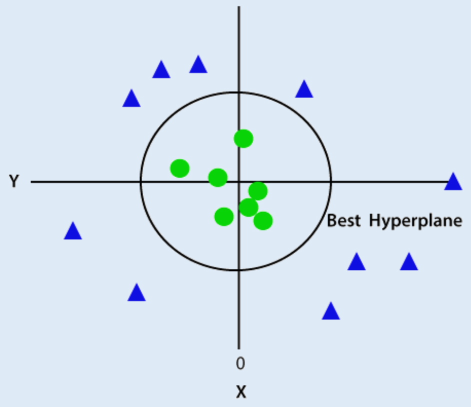
Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.



Linear SVM



Non- linear SVM

MODULE – 8 (Non-metric methods for pattern classification)

In pattern recognition, non-numeric data or nominal data refers to data that consists of categories or labels rather than numerical values. This type of data is often encountered when dealing with qualitative or categorical variables. Examples of nominal data include the type of animal (cat, dog, bird), colors (red, blue, green), or categories such as "yes" or "no".

When working with non-numeric data in pattern recognition, there are a few considerations to keep in mind:

1. **Encoding:** Non-numeric data needs to be encoded into a numerical representation that can be used by pattern recognition algorithms. One common approach is to use one-hot encoding, where each category is represented by a binary vector of 0s and 1s. For example, if we have three colors (red, blue, green), one-hot encoding would represent red as [1, 0, 0], blue as [0, 1, 0], and green as [0, 0, 1].
2. **Feature Extraction:** Depending on the specific problem, feature extraction techniques may be applied to transform the non-numeric data into a set of meaningful numerical features. This process involves extracting relevant information from the nominal data that can be used for pattern recognition tasks. For example, in text classification, features could be extracted based on the presence or frequency of specific words or phrases.
3. **Similarity Metrics:** When dealing with non-numeric data, it is important to define appropriate similarity or distance metrics. These metrics quantify the similarity or dissimilarity between different instances based on their non-numeric attributes. For example, when comparing two categorical variables, the Hamming distance or Jaccard similarity coefficient can be used.
4. **Algorithm Selection:** Different pattern recognition algorithms may handle non-numeric data differently. Some algorithms, such as decision trees or random forests, can directly handle categorical variables without requiring specific encoding. Others, like k-nearest neighbors or support vector machines, may require the non-numeric data to be converted to numerical form. It is important to select an algorithm that is suitable for the specific type of non-numeric data and the problem at hand.

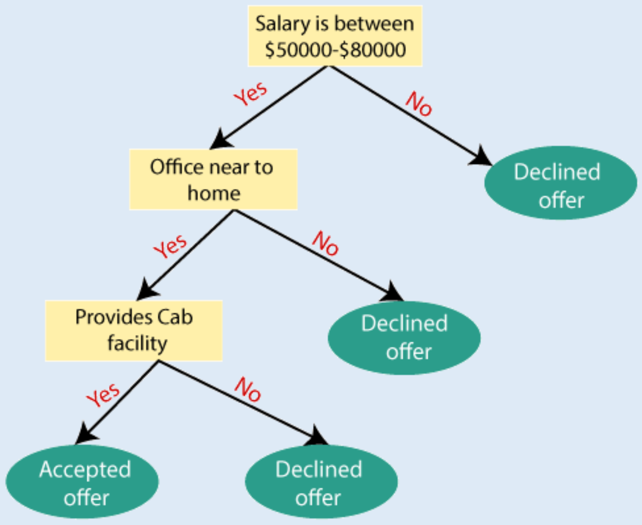
Overall, handling non-numeric or nominal data in pattern recognition involves appropriate encoding, feature extraction, selection of similarity metrics, and choosing algorithms that can effectively handle such data. By considering these aspects, non-numeric data can be successfully incorporated into pattern recognition tasks.

Decision Tree:-

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The branches represents the decision rule.

Decision Tree Terminologies

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

```
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
```

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Module – 9 (Unsupervised learning and clustering)

What are the criterion function for clustering algorithm

Clustering algorithms aim to group similar data points together based on certain criteria. The choice of criterion function, also known as the clustering objective or evaluation function, depends on the specific clustering algorithm being used. Here are some common criterion functions used in clustering:

1. **K-Means Clustering:**
    - Within-cluster sum of squares (WCSS): Also known as inertia, it measures the sum of squared distances between each data point and the centroid of its assigned cluster. The objective is to minimize the WCSS, indicating tight and compact clusters.
2. **Hierarchical Clustering:**
    - Linkage-based criteria: Hierarchical clustering algorithms often use different linkage criteria to determine the distance between clusters. Some commonly used linkage criteria include:
        - **Single linkage:** The distance between clusters is based on the minimum distance between any two points in the clusters.
        - **Complete linkage:** The distance between clusters is based on the maximum distance between any two points in the clusters.
        - **Average linkage:** The distance between clusters is based on the average distance between all pairs of points in the clusters.
3. **Density-Based Clustering (e.g., DBSCAN):**
    - Density-reachability: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) uses density-reachability to identify dense regions separated by sparser areas. Points are classified as core points, border points, or noise points based on their density and reachability within a specified radius.
4. **Gaussian Mixture Models (GMM):**
    - Maximum likelihood: GMMs aim to fit a mixture of Gaussian distributions to the data. The criterion function involves maximizing the likelihood of the data given the model parameters. This is typically achieved using the Expectation-Maximization (EM) algorithm.
5. **Spectral Clustering:**
    - Graph-based criteria: Spectral clustering algorithms leverage the eigenvectors of a similarity graph to partition the data. The criterion function involves

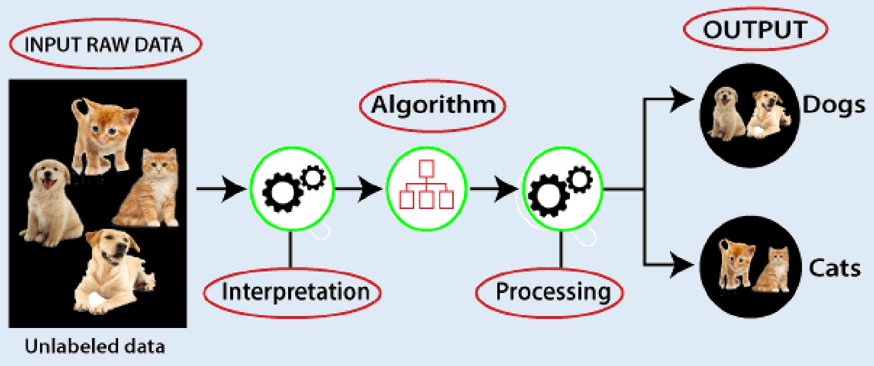
finding a cut that maximizes the similarity between points within clusters and minimizes the similarity between points across clusters.

It's important to note that the choice of criterion function can vary based on the clustering algorithm and the specific problem at hand. The objective is to select a criterion that aligns with the clustering goals, such as maximizing intra-cluster similarity and minimizing inter-cluster similarity. Additionally, some criterion functions may require tuning of parameters, such as the number of clusters or a density threshold, to achieve optimal clustering results

What is Unsupervised Learning?

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision. Unsupervised learning model finds the hidden patterns in data. In unsupervised learning, only input data is provided to the model.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images



Types of Unsupervised Learning Algorithm:

- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the

set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Clustering:- Clustering is an unsupervised Machine Learning-based Algorithm that comprises a group of data points into clusters so that the similar objects belong to the same cluster. Clustering is a method of partitioning a set of data or objects into a set of significant subclasses called clusters. A cluster is a subset of similar objects. Categorization of Clustering Methods Clustering methods can be categorized into several types based on their underlying algorithms and approaches. Here are some common categorizations of clustering methods along with examples:

1. **Partitioning Methods:**

Examples: k-means, k-medoids (PAM - Partitioning Around Medoids)

Partitioning methods aim to partition the data into a fixed number of clusters by iteratively optimizing an objective function. They assign each data point to the nearest cluster centroid based on distance measures such as Euclidean distance.

2.Hierarchical Methods: Examples: Agglomerative Hierarchical Clustering, Divisive 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. Agglomerative hierarchical clustering starts with each data point as a separate cluster and merges the most similar clusters iteratively. Divisive hierarchical clustering starts with all data points in a single cluster and splits them into smaller clusters.

3. Density-Based Methods: Examples: DBSCAN (Density-Based Spatial Clustering of Applications with Noise).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.

4. Model-Based Methods: Examples: Gaussian Mixture Models (GMM), Expectation-Maximization (EM) algorithm Model-based methods assume that the data is generated from a probabilistic model. They aim to fit a model to the data and identify clusters based on the model parameters. Gaussian Mixture Models (GMM) assume that the data points come from a mixture of Gaussian distributions.

5. Fuzzy Clustering Methods: Examples: Fuzzy C-Means (FCM)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:- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties. 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).

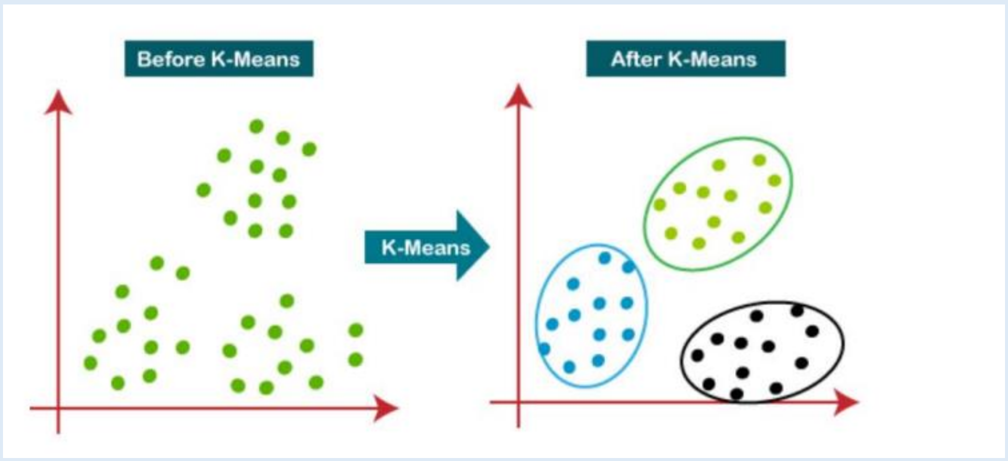
Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters. This is done by finding the Euclidian distance of the given data set form the centroid of each cluster.

Step-4: Find the shortest distance of the given data set among all the distance from form all the clusters.

Step-5: Place the data set in the cluster which have the closest distance from it. Update the centroid of the belonging class.

Step-6: Repeat the steps for all the given data set.

Step-7: Your K-means model is ready.



Hierarchical clustering:- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled 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. The hierarchical clustering technique has two approaches

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. 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.

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

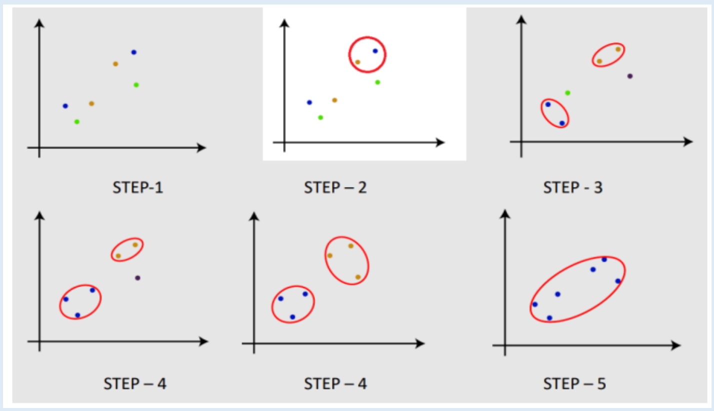
Step-1: Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.

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

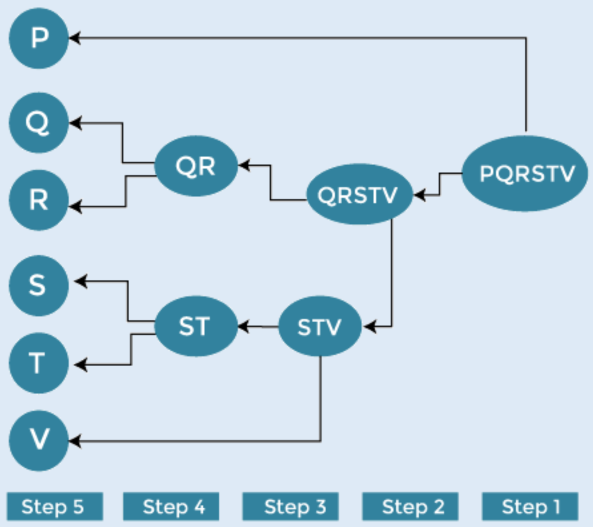
Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.

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

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



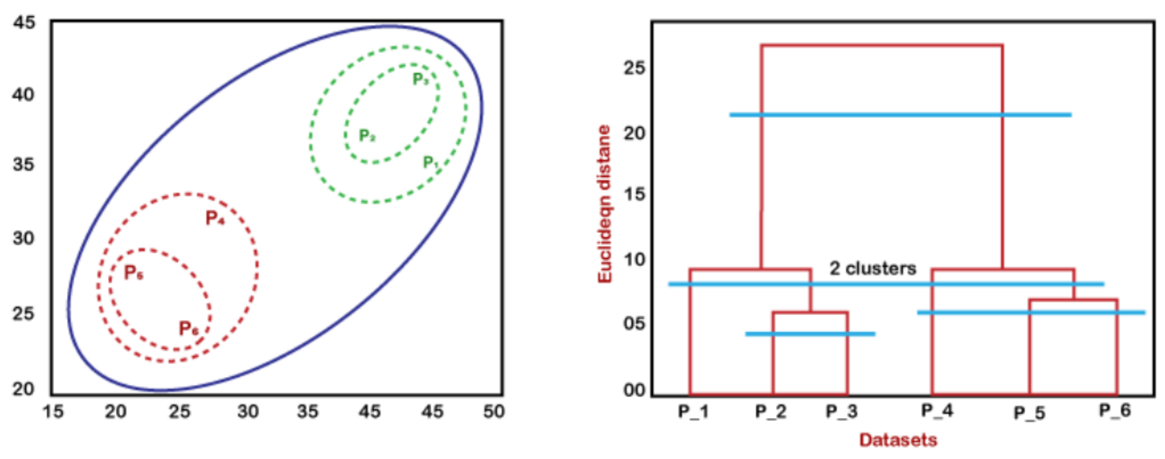
2. Divisive: Divisive algorithm is the reverse of the agglomerative algorithm as it is a top-down approach. In Divisive Hierarchical clustering, all the data points are considered an individual cluster, and in every iteration, the data points that are not similar are separated from the cluster. The separated data points are treated as an individual cluster. Finally, we are left with N clusters.



Woking of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



45
40
35
30
25
20
15
20
25
30
35
45
45
50
P_3
P_2
P_1
P_4
P_5
P_6
Euclideqn distane
25
20
15
10
05
00
2 clusters
P_1
P_2
P_3
P_4
P_5
P_6
Datasets