Unit I
Unsupervised Learning

Content

- An Overview on Unsupervised Learning
- Clustering
- K-Means Clustering
- Hierarchical Clustering
- Association Rule Mining
- Apriori Algorithm
- F-p growth algorithm
- Biclustering
- Gaussian Mixture Model

Unsupervised Learning

- What is Unsupervised Learning?
- Definition
- Goal
- Example
- Why use Unsupervised Learning
- Working
- Types
- Unsupervised Learning Algorithms
- Advantages
- Disadvantages

Unsupervised Machine Learning: An overview

- In the previous topic, we learned supervised machine learning in which models are trained using labeled data under the supervision of training data. But there may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.

What is Unsupervised Learning?

- As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Definition:

- *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.*

Goal

- The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

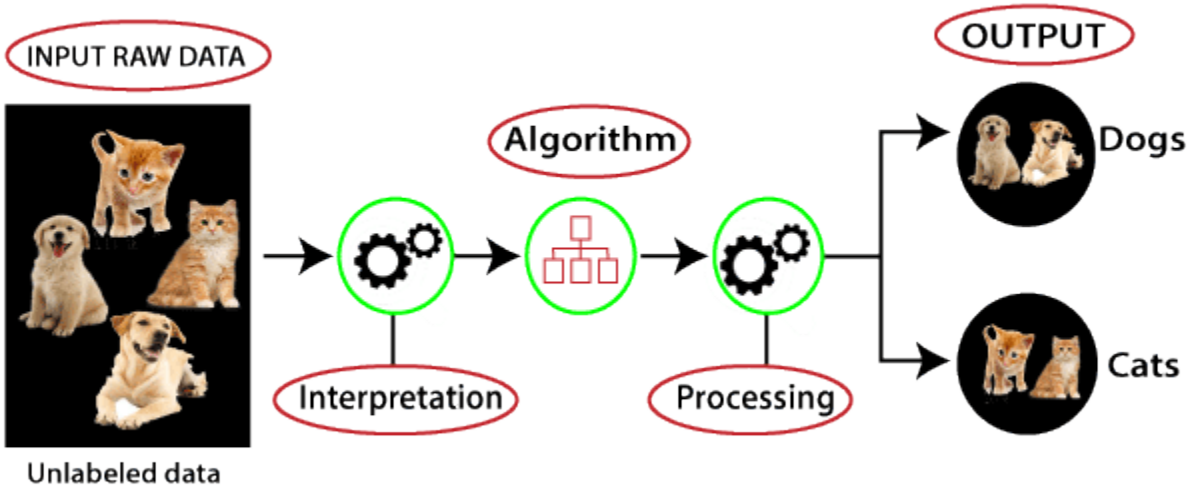
Example:

- 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.

Why use Unsupervised Learning?

- Below are some main reasons which describe the importance of Unsupervised Learning:
- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

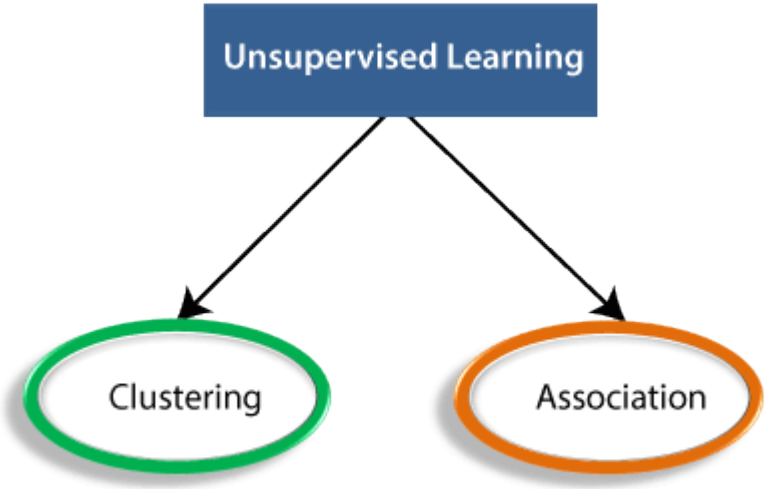
Working of Unsupervised Learning



Working of Unsupervised Learning

- Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.
- Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:



Types:

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.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchal clustering**
- **Anomaly detection**
- **Neural Networks**
- **Principle Component Analysis**
- **Independent Component Analysis**
- **Apriori algorithm**
- **Singular value decomposition**

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

Clustering

- Definition
- Explanation
- Example
- Working
- Types
- Applications

Clustering : Definition

- ***"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."***

Explanation

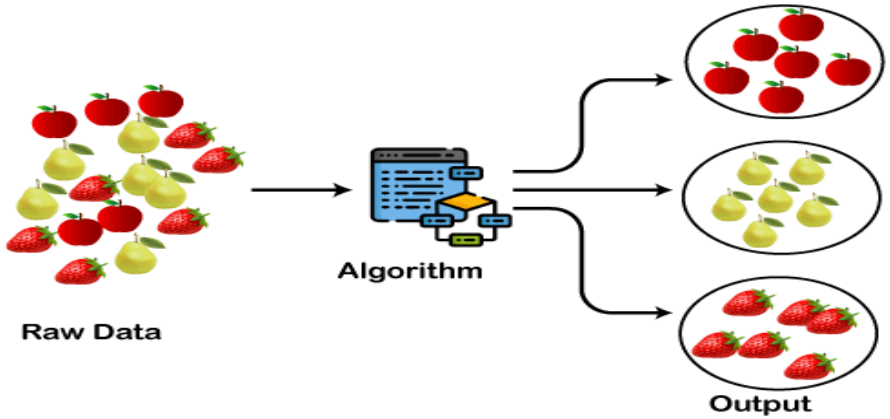
- Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset.
- It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.
- It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled 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.

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.

Working

- The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.

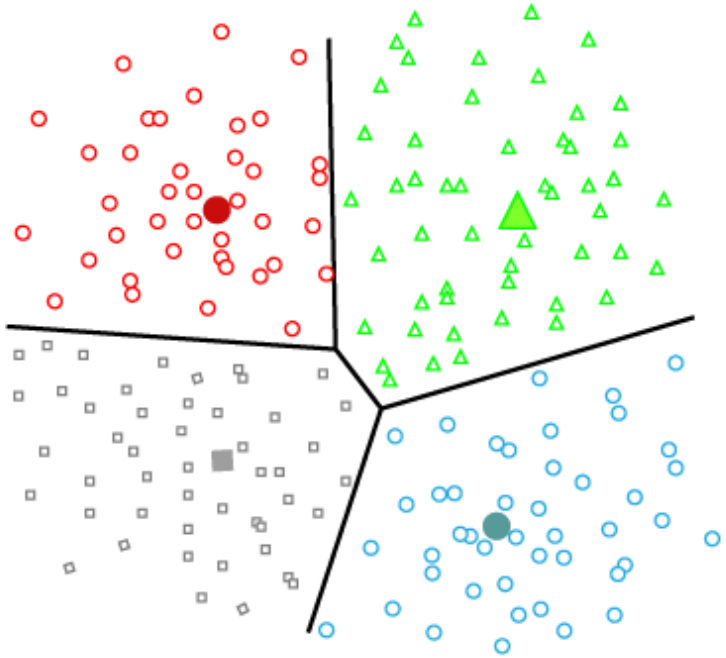


Types of Clustering Methods

- 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:
- **Partitioning Clustering**
- **Density-Based Clustering**
- **Distribution Model-Based Clustering**
- **Hierarchical Clustering**
- **Fuzzy Clustering**

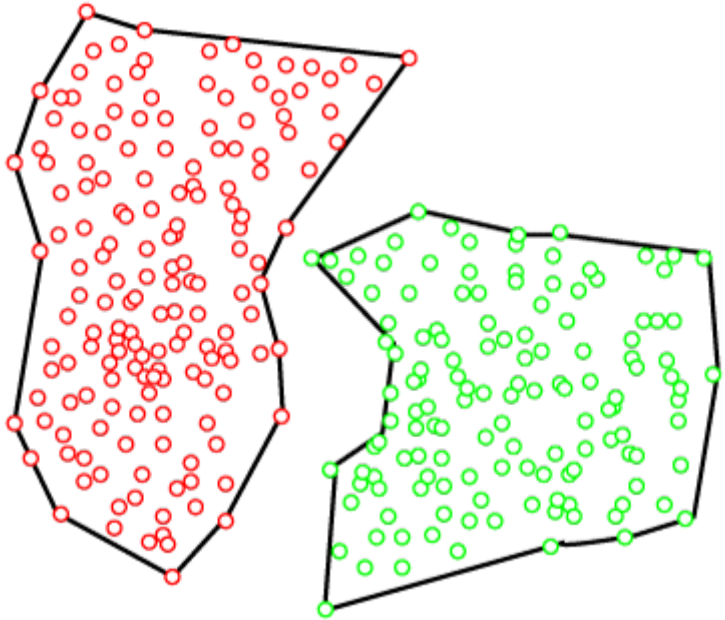
Partitioning Clustering

- 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.



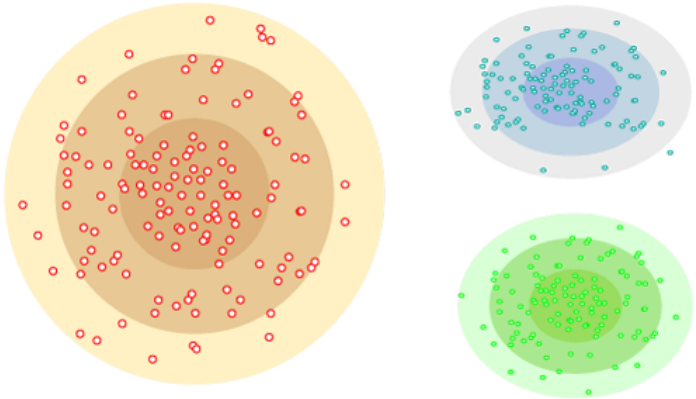
Density –Based Clustering

- 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.



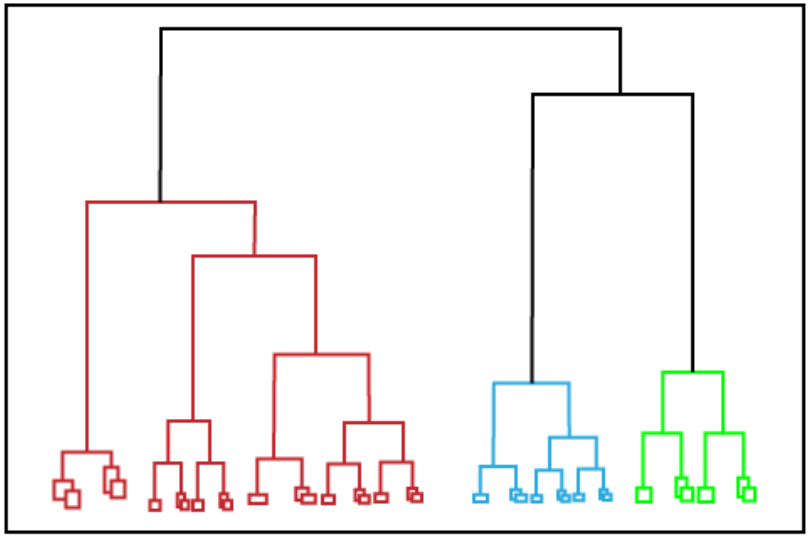
Distribution Model-Based Clustering

- 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.

Applications of Clustering

Below are some commonly known applications of clustering technique in Machine Learning:

- **In Identification of Cancer Cells:** The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.
- **In Search Engines:** Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurate result of a query depends on the quality of the clustering algorithm used.
- **Customer Segmentation:** It is used in market research to segment the customers based on their choice and preferences.
- **In Biology:** It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- **In Land Use:** The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.

K-Means Clustering

- Definition
- Explanation
- Example
- Working
- How to choose the value of "K number of clusters" in K-means Clustering?
- Elbow Method

Definition

- 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.

Explanation

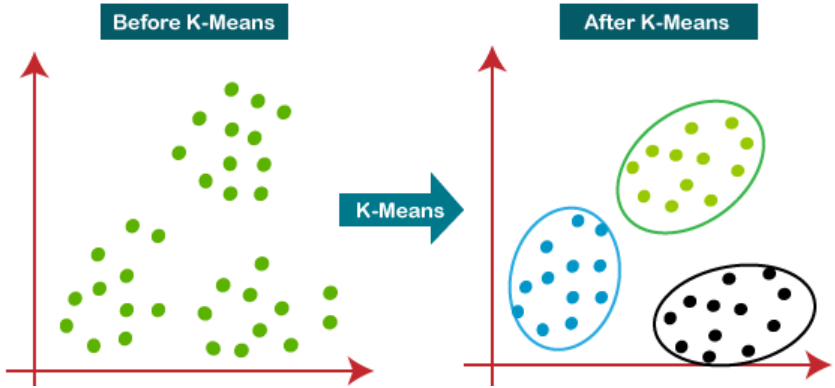
- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.
- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

Example

- The k-means clustering algorithm mainly performs two tasks:

1. Determines the best value for K center points or centroids by an iterative process.
2. Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

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

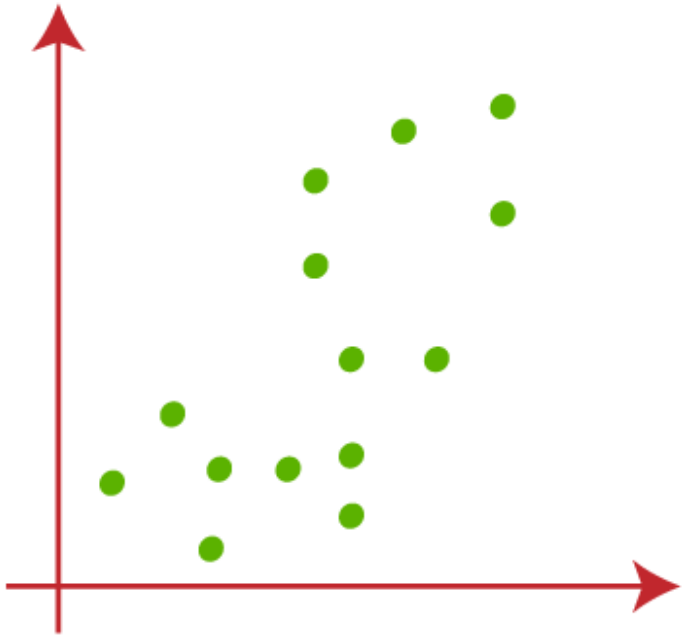


How does the K-Means Algorithm Work?

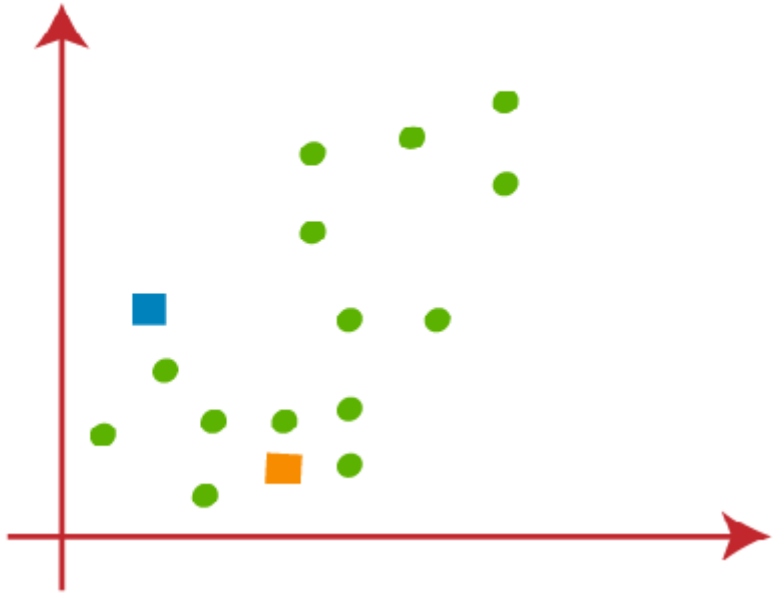
- The working of the K-Means algorithm is explained in the below steps:
- **Step-1:** Select the number K to decide the number of clusters.
- **Step-2:** Select random K points or centroids. (It can be other from the input dataset).
- **Step-3:** Assign each data point to their closest centroid, which 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.

Working

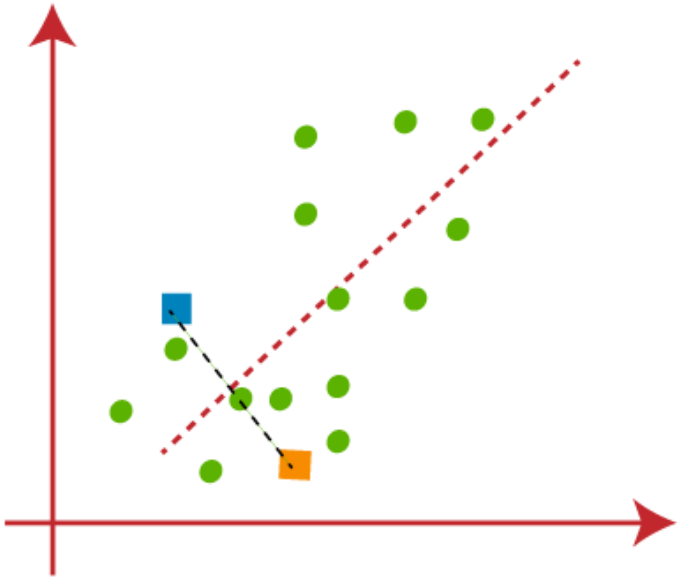
- Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



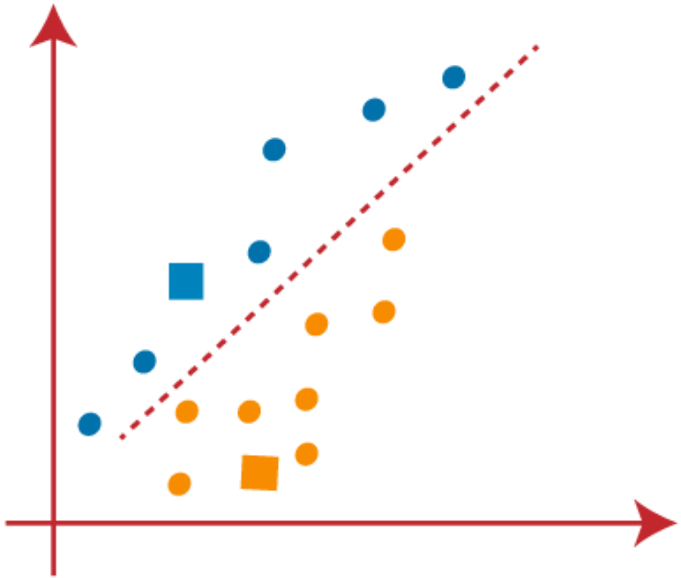
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points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



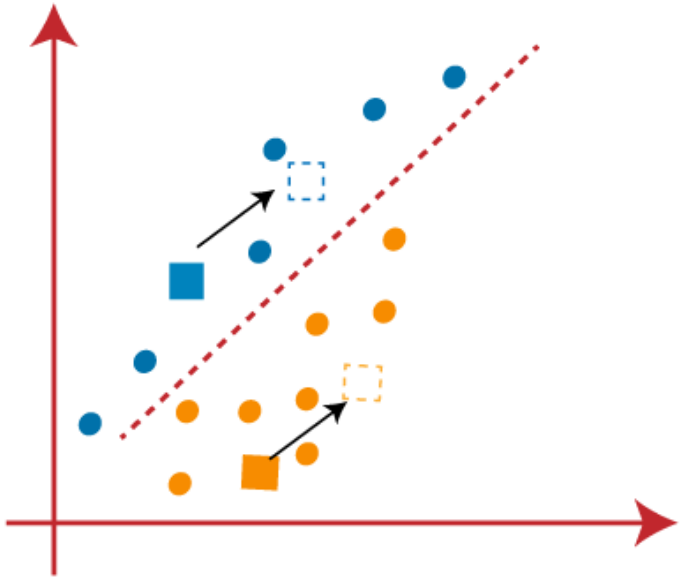
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



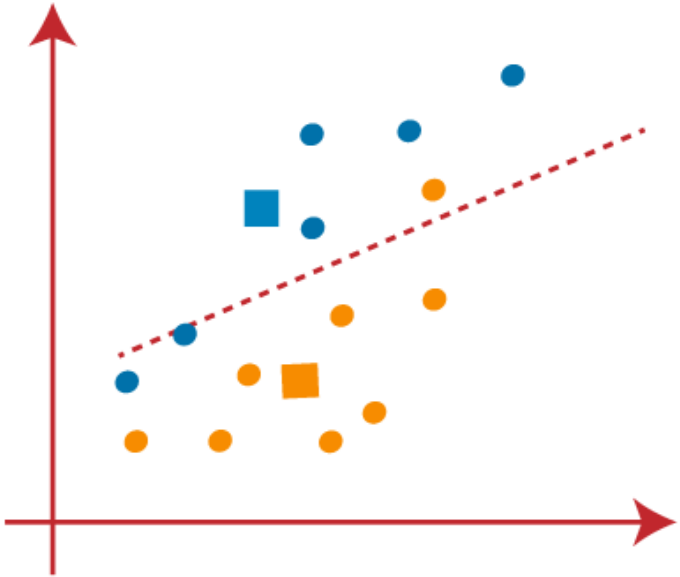
- From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



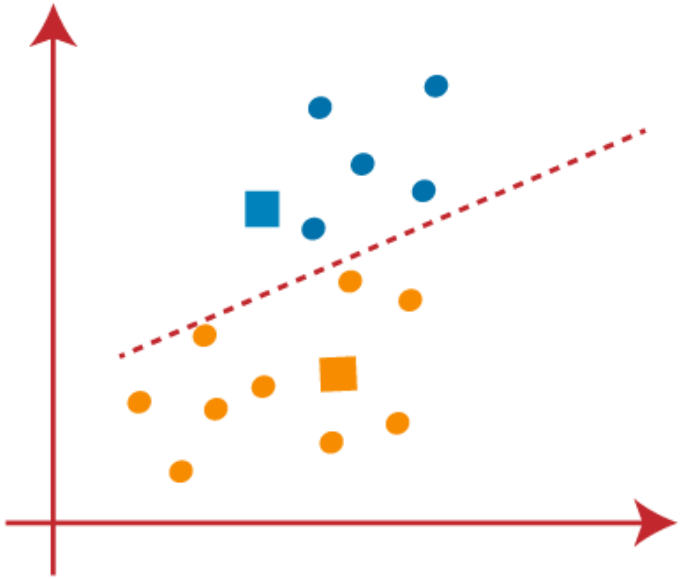
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



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

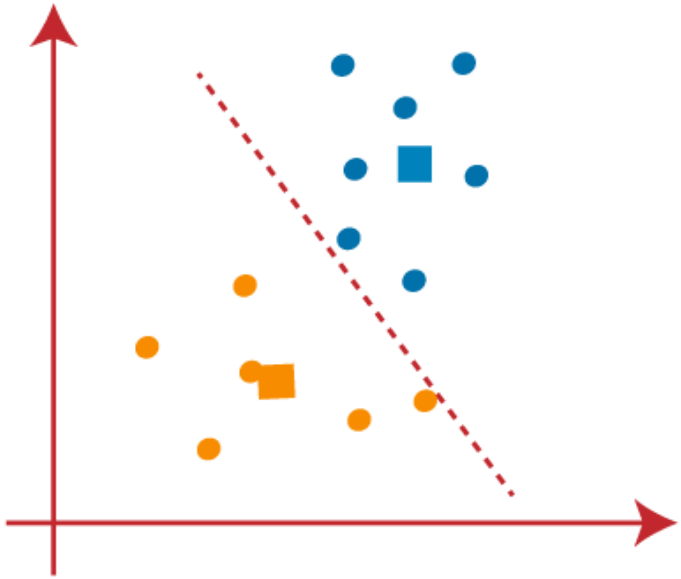


- From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

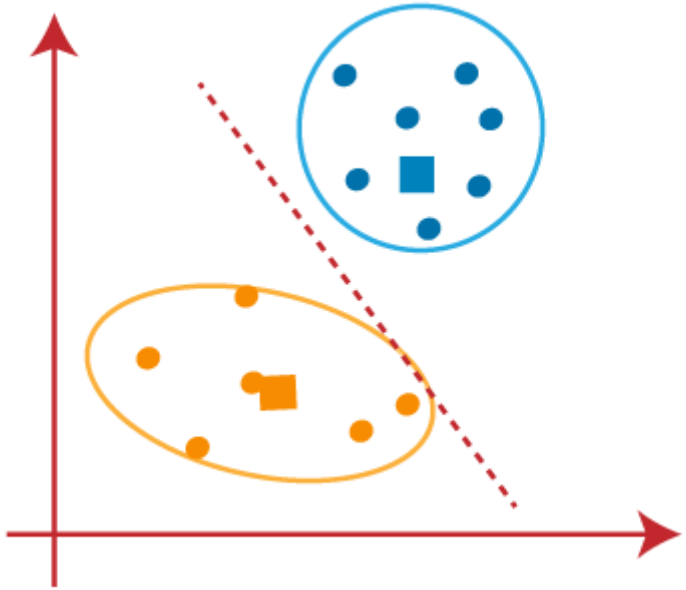


- As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:

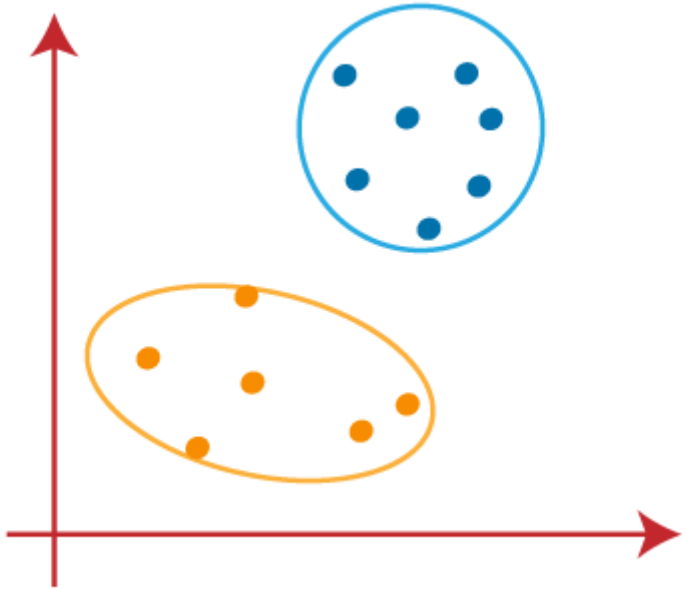
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



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



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



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

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

Elbow Method

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

$$\mathbf{WCSS} = \sum_{Pi\ in\ Cluster1} distance(P_i\ C_1)^2 + \sum_{Pi\ in\ Cluster2} distance(P_i\ C_2)^2 + \sum_{Pi\ in\ CLuster3} distance(P_i\ C_3)^2$$

In the above formula of WCSS,

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

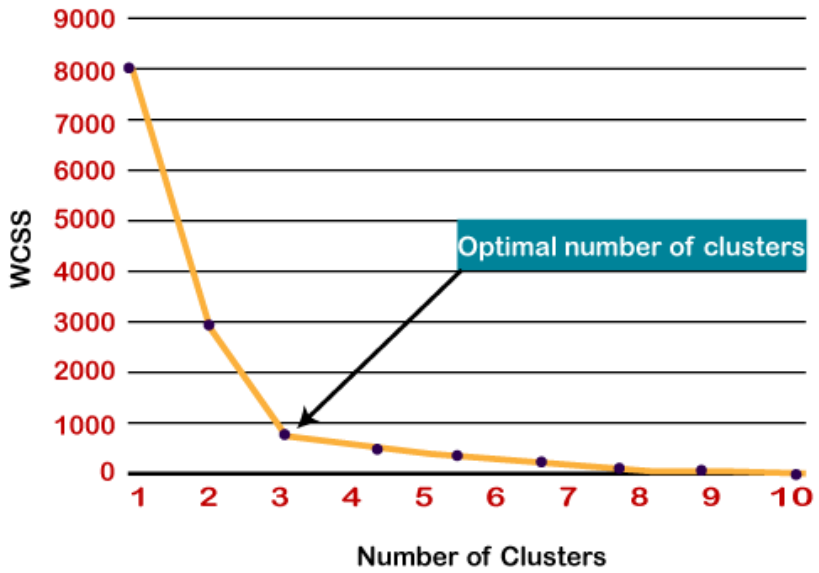
To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

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

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

Elbow Method

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



Hierarchical Clustering

- Definition
- Explanation
- Hierarchical Clustering Approaches
- Need
- Agglomerative Hierarchical Clustering
- How the Agglomerative Hierarchical clustering Work?
- Working of Dendrogram in Hierarchical clustering
- Measure for the distance between two clusters

Definition

- 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.

Explanation

- 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.

Hierarchical clustering approaches

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.
- **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:

a. which are a predetermined number of clusters

b. It always tries to create the clusters of the same size.

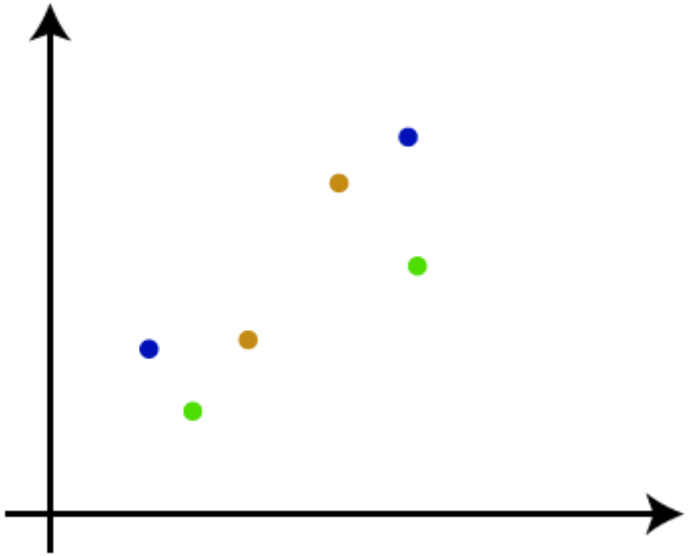
To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

Agglomerative Hierarchical clustering

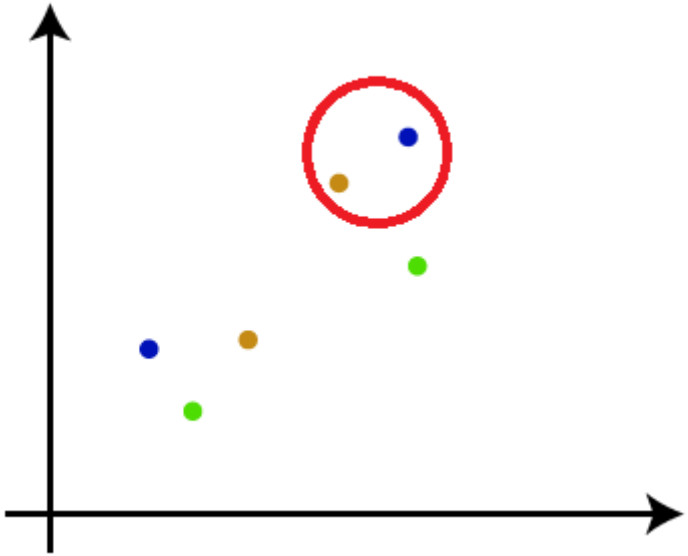
- The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. 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?

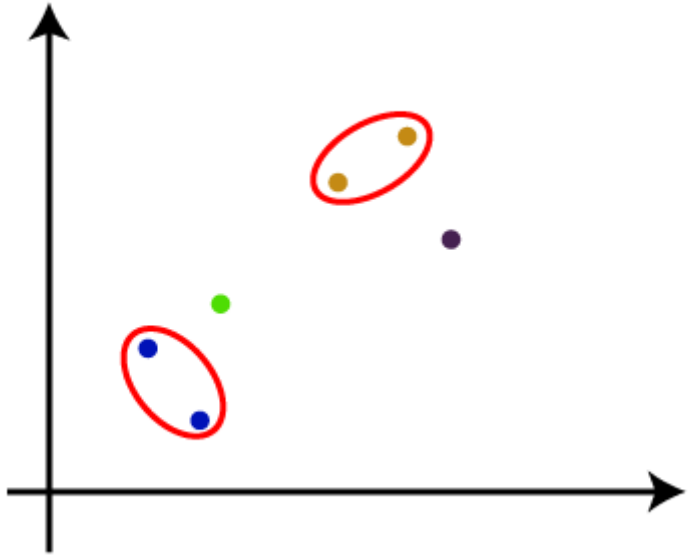
- The working of the AHC algorithm can be explained using the below steps:
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



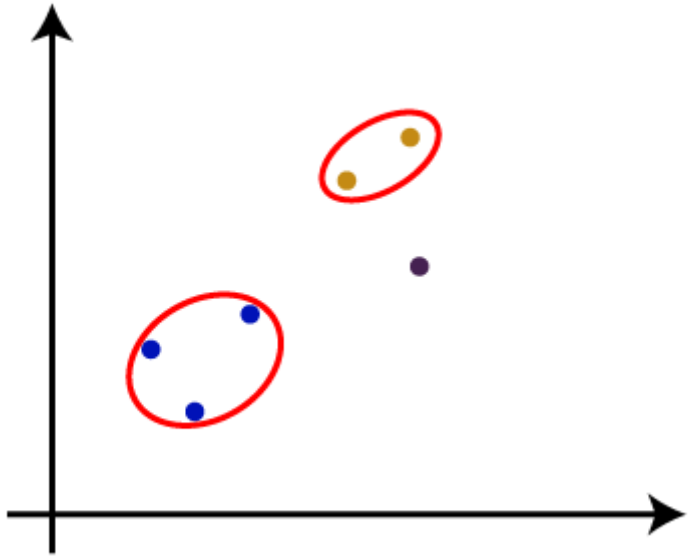
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

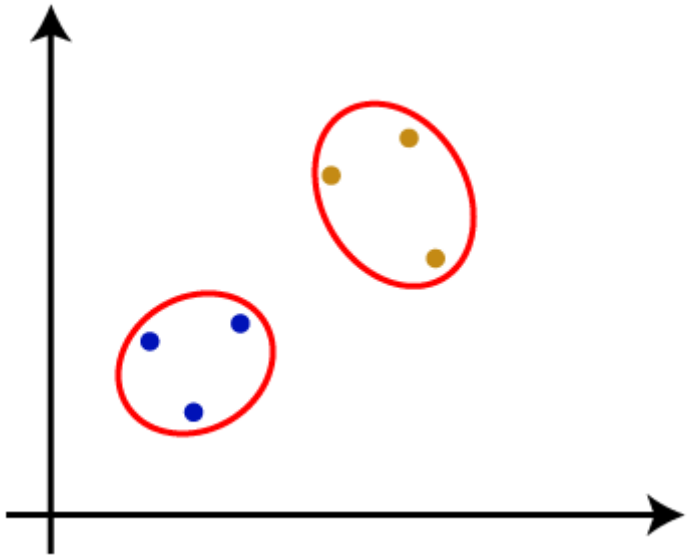


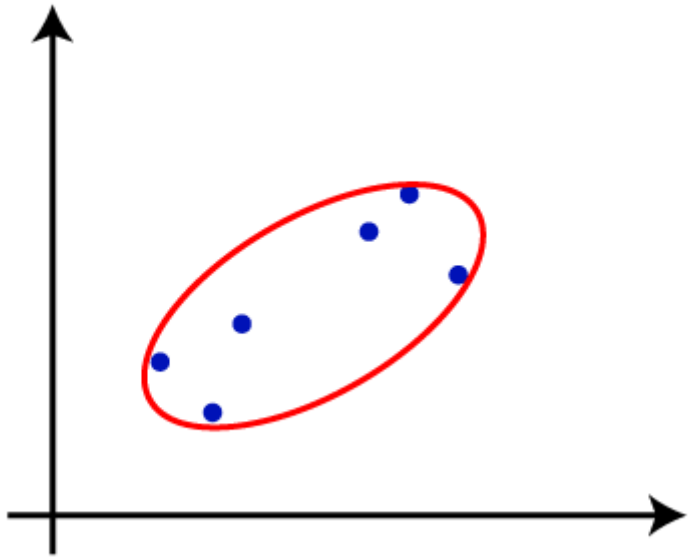
- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



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







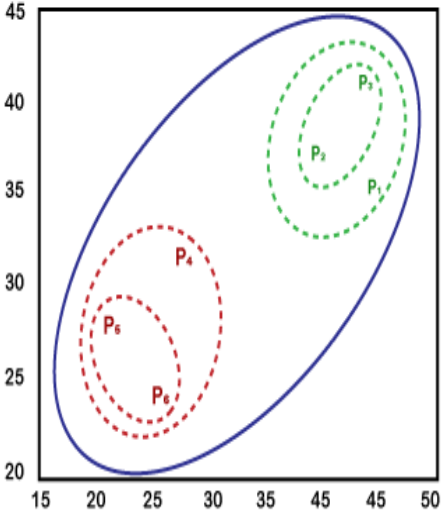
- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

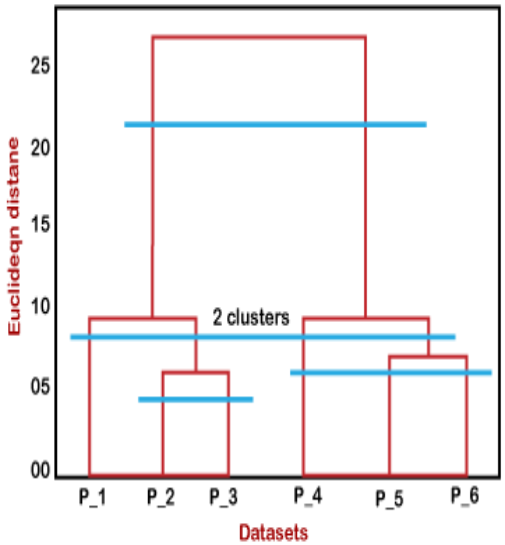
Working of Dendrogram in Hierarchical clustering

- 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.

After step 5

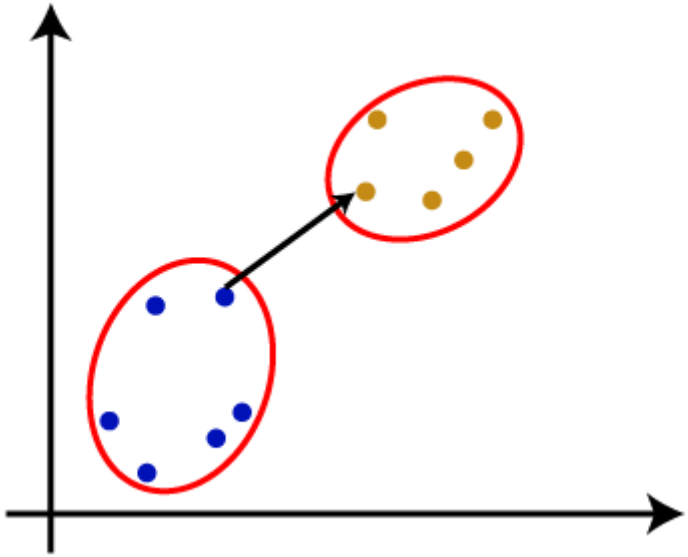
- In the diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.
- As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
- In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
- Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
- At last, the final dendrogram is created that combines all the data points together.
- We can cut the dendrogram tree structure at any level as per our requirement.



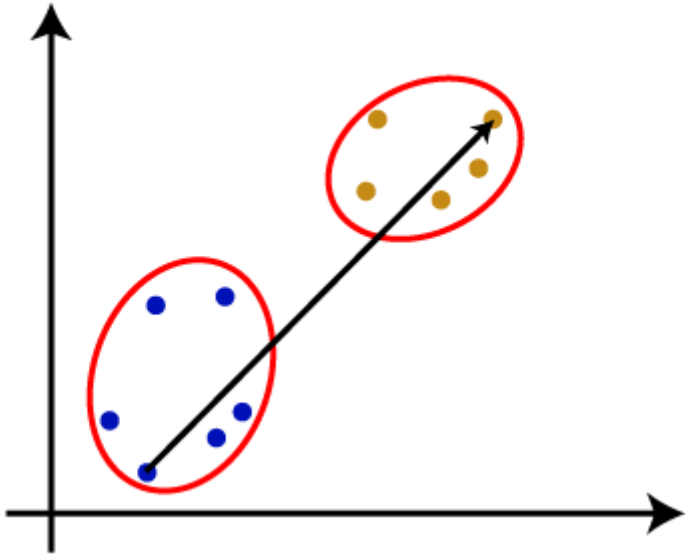


Measure for the distance between two clusters

- As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

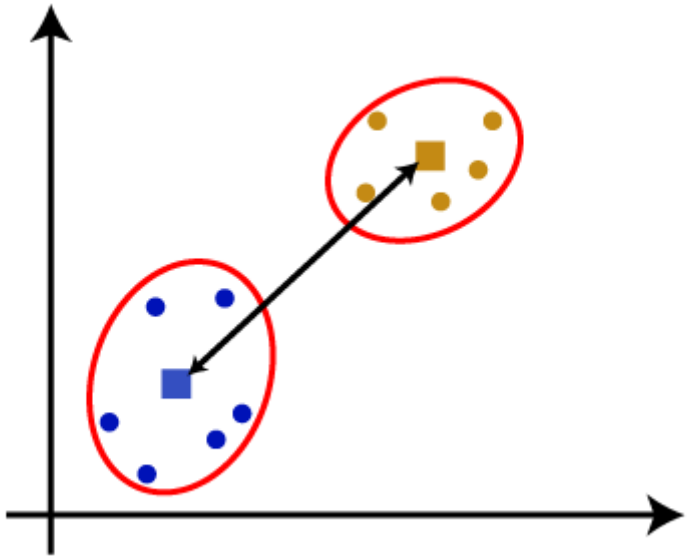


- **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



- **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

- **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



Association Rule Mining

- Definition
- Explanation
- Types
- Working
- Applications

Definition

- Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable.

Explanation

- The association rule learning is one of the very important concepts of machine learning, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc. Here market basket analysis is a technique used by the various big retailer to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put together.
- For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby.

Types

- Association rule learning can be divided into three types of algorithms:

1. **Apriori**
2. **Eclat**
3. **F-P Growth Algorithm**

Apriori Algorithm

- This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the item set efficiently.
- It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

Eclat Algorithm

- Eclat algorithm stands for **Equivalence Class Transformation**. This algorithm uses a depth-first search technique to find frequent itemsets in a transaction database. It performs faster execution than Apriori Algorithm.

F-P Growth Algorithm

- The F-P growth algorithm stands for **Frequent Pattern**, and it is the improved version of the Apriori Algorithm. It represents the database in the form of a tree structure that is known as a frequent pattern or tree. The purpose of this frequent tree is to extract the most frequent patterns.

How does Association Rule Learning work?

- Association rule learning works on the concept of If and Else Statement, such as if A then B.

How does Association Rule Learning work?

- Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:
- **Support**
- **Confidence**
- **Lift**

How does Association Rule Learning work?

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp}(X) = \frac{Freq(X)}{T}$$

How does Association Rule Learning work?

Confidence

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

How does Association Rule Learning work?

Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift}=\frac{Supp(X,Y)}{Supp(X)\times Supp(Y)}$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.

Lift>1: It determines the degree to which the two itemsets are dependent to each other.

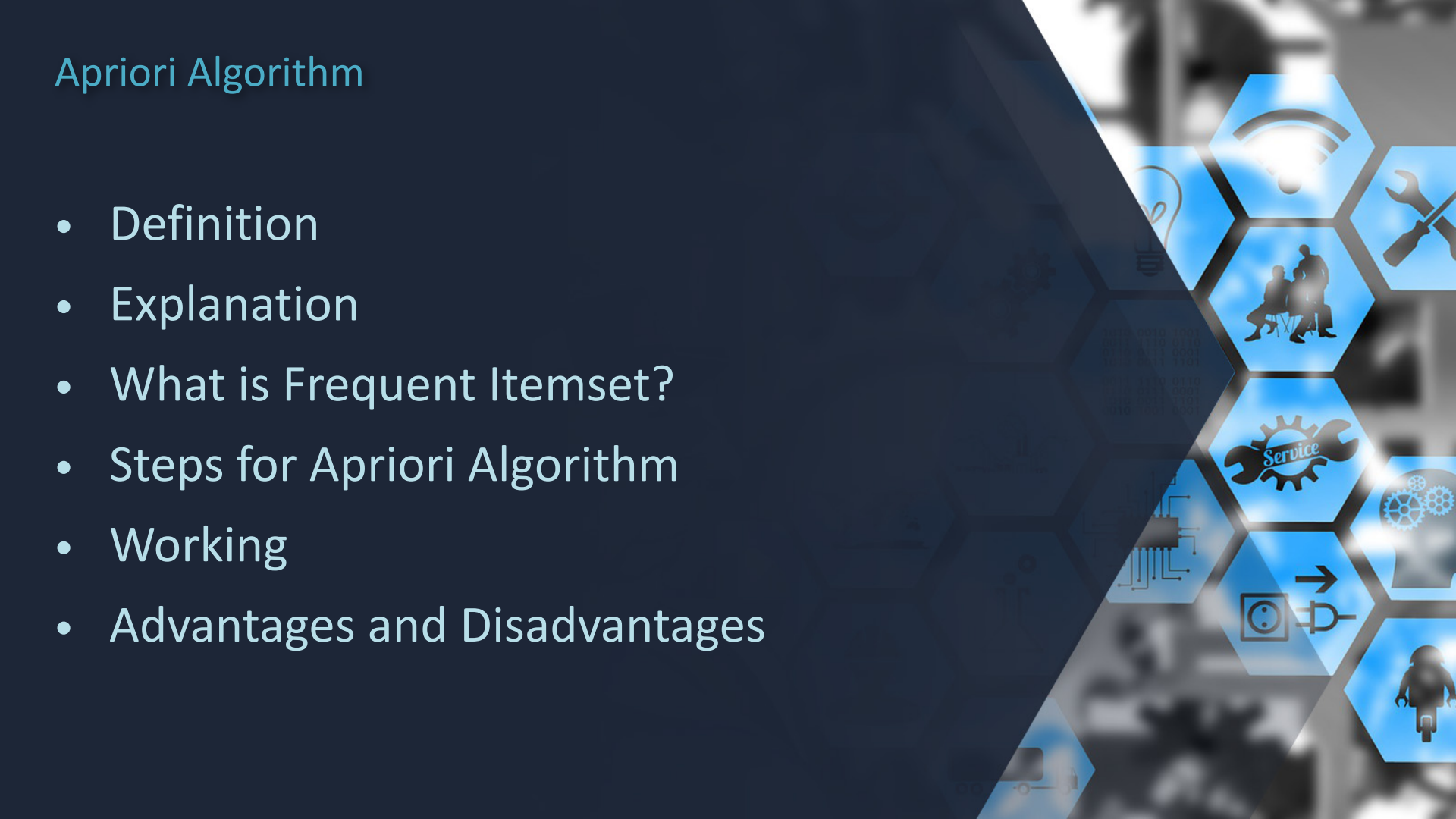
Lift<1: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

Applications of Association Rule Learning

- It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:
- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

Apriori Algorithm

- Definition
- Explanation
- What is Frequent Itemset?
- Steps for Apriori Algorithm
- Working
- Advantages and Disadvantages



Definition

- The Apriori algorithm is an algorithm which uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions.

Explanation

- With the help of these association rule, it determines how strongly or how weakly two objects are connected.
- This algorithm uses a **breadth-first search** and **Hash Tree** to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset.
- This algorithm was given by the **R. Agrawal** and **Srikant** in the year **1994**.
- It is mainly used for *market basket analysis* and helps to find those products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

What is Frequent Itemset?

- Frequent itemsets are those items whose support is greater than the threshold value or user-specified minimum support. It means if A & B are the frequent itemsets together, then individually A and B should also be the frequent itemset.
- Suppose there are the two transactions: A= {1,2,3,4,5}, and B= {2,3,7}, in these two transactions, 2 and 3 are the frequent itemsets.

Steps for Apriori Algorithm

- Below are the steps for the apriori algorithm:
- **Step-1:** Determine the support of itemsets in the transactional database, and select the minimum support and confidence.
- **Step-2:** Take all supports in the transaction with higher support value than the minimum or selected support value.
- **Step-3:** Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.
- **Step-4:** Sort the rules as the decreasing order of lift.

Apriori Algorithm Working

- We will understand the apriori algorithm using an example and mathematical calculation:
- **Example:** Suppose we have the following dataset that has various transactions, and from this dataset, we need to find the frequent itemsets and generate the association rules using the Apriori algorithm:

TID	ITEMSETS
T1	A, B
T2	B, D
T3	B, C
T4	A, B, D
T5	A, C
T6	B, C
T7	A, C
T8	A, B, C, E
T9	A, B, C

Given: Minimum Support= 2, Minimum Confidence= 50%

Solution:

Step-1: Calculating C1 and L1:

- In the first step, we will create a table that contains support count (The frequency of each itemset individually in the dataset) of each itemset in the given dataset. This table is called the **Candidate set or C1.**

Itemset	Support_Count
A	6
B	7
C	5
D	2
E	1

- Now, we will take out all the itemsets that have the greater support count that the Minimum Support (2). It will give us the table for the **frequent itemset L1.**

  Since all the itemsets have greater or equal support count than the minimum support, except the E, so E itemset will be removed.

Itemset	Support_Count
A	6
B	7
C	5
D	2

Step-2: Candidate Generation C2, and L2:

- In this step, we will generate C2 with the help of L1. In C2, we will create the pair of the itemsets of L1 in the form of subsets.
- After creating the subsets, we will again find the support count from the main transaction table of datasets, i.e., how many times these pairs have occurred together in the given dataset. So, we will get the below table for C2:

Itemset	Support_Count
{A, B}	4
{A,C}	4
{A, D}	1
{B, C}	4
{B, D}	2
{C, D}	0

- Again, we need to compare the C2 Support count with the minimum support count, and after comparing, the itemset with less support count will be eliminated from the table C2. It will give us the below table for L2

Itemset	Support_Count
{A, B}	4
{A, C}	4
{B, C}	4
{B, D}	2

A, B, C, D

Step-3: Candidate generation C3, and L3:

- For C3, we will repeat the same two processes, but now we will form the C3 table with subsets of three itemsets together, and will calculate the support count from the dataset. It will give the below table:

Itemset	Support_Count
{A, B, C}	2
{B, C, D}	1
{A, C, D}	0
{A, B, D}	0

- Now we will create the L3 table. As we can see from the above C3 table, there is only one combination of itemset that has support count equal to the minimum support count. So, the L3 will have only one combination, i.e., **{A, B, C}.**

Step-4: Finding the association rules for the subsets:

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B.C}. For all the rules, we will calculate the Confidence using formula **sup(A ^B)/A.** After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

Consider the below table:

Rules	Support	Confidence
A ^B → C	2	Sup{(A ^B) ^C}/sup(A ^B)= 2/4=0.5=50%
B^C → A	2	Sup{(B^C) ^A}/sup(B ^C)= 2/4=0.5=50%
A^C → B	2	Sup{(A ^C) ^B}/sup(A ^C)= 2/4=0.5=50%
C→ A ^B	2	Sup{(C^(A ^B)}/sup(C)= 2/5=0.4=40%
A→ B^C	2	Sup{(A^(B ^C)}/sup(A)= 2/6=0.33=33.33%
B→ B^C	2	Sup{(B^(B ^C)}/sup(B)= 2/7=0.28=28%

As the given threshold or minimum confidence is 50%, so the first three rules **A ^B → C, B^C → A, and A^C → B** can be considered as the strong association rules for the given problem.

Advantages and Disadvantages

- **Advantages of Apriori Algorithm**
- This is easy to understand algorithm
- The join and prune steps of the algorithm can be easily implemented on large datasets.
- **Disadvantages of Apriori Algorithm**
- The apriori algorithm works slow compared to other algorithms.
- The overall performance can be reduced as it scans the database for multiple times.
- The time complexity and space complexity of the apriori algorithm is $O(2^D)$, which is very high. Here D represents the horizontal width present in the database.

Task-1

Suppose we have the rule {Milk, Bread} → {Butter} with:
Support = 0.3

Confidence = 0.75

Lift = 1.25

Interpret what this rule means in simple terms.

Solution

Step 1: Support

Formula:

$$\text{Support}(X \to Y) = \frac{\text{Number of transactions containing } (X \cup Y)}{\text{Total transactions}}$$

Here:

- $X = \{Milk, Bread\}$
- $Y = \{Butter\}$
- $X \cup Y = \{Milk, Bread, Butter\}$

Given: Support = **0.3**

👉 This means **30% of all transactions contain Milk, Bread, and Butter together.**

Step 2: Confidence

Formula:

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

Here:

- Support($X \cup Y$) = 0.3 (Milk, Bread, Butter)
- Confidence = 0.75

So:

$$0.75 = \frac{0.3}{\text{Support}(\{Milk, Bread\})}$$

$$\text{Support}(\{Milk, Bread\}) = \frac{0.3}{0.75} = 0.4$$

👉 40% of transactions contain **Milk and Bread**.

👉 Among those, 75% also have Butter.

Step 3: Lift

Formula:

$$\text{Lift}(X \to Y) = \frac{\text{Confidence}(X \to Y)}{\text{Support}(Y)}$$

Given: Lift = 1.25, Confidence = 0.75.

So:

$$1.25 = \frac{0.75}{\text{Support}(\{Butter\})}$$

$$\text{Support}(\{Butter\}) = \frac{0.75}{1.25} = 0.6$$

👉 60% of transactions overall contain **Butter**.

- **Support = 0.3** → 30% of transactions include Milk, Bread, and Butter together.
- **Confidence = 0.75** → If a customer buys Milk and Bread, there's a 75% chance they also buy Butter.
- **Lift = 1.25** → Buying Milk and Bread increases the likelihood of buying Butter by 25% compared to buying Butter randomly (since Butter alone has support 0.6).

Task-2

A supermarket database has 10,000 transactions. An itemset {Shampoo, Soap} occurs in 1,000 transactions, and {Shampoo, Soap, Toothpaste} occurs in 600 transactions.

1. Compute the **support** and **confidence** of the rule {Shampoo, Soap} → {Toothpaste}.
2. Should this rule be considered strong if minimum support = 5% and minimum confidence = 50%?

Solution

Given :

Total transactions = 10,000

$\{Shampoo, Soap\}$ = 1,000 transactions

$\{Shampoo, Soap, Toothpaste\}$ = 600 transactions

Rule: $\{Shampoo, Soap\} \rightarrow \{Toothpaste\}$

Minimum support = 5%

Minimum confidence = 50%

Step 1: Support of the rule

Formula:

$$\text{Support}(X \to Y) = \frac{\text{Transactions containing } (X \cup Y)}{\text{Total transactions}}$$

Here:

- $X = \{Shampoo, Soap\}$
- $Y = \{Toothpaste\}$
- $X \cup Y = \{Shampoo, Soap, Toothpaste\} = 600$

$$\text{Support} = \frac{600}{10,000} = 0.06 = 6\%$$

Step 2: Confidence of the rule

Formula:

$$\text{Confidence}(X \to Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

- Support($X \cup Y$) = 600 / 10,000 = 0.06
- Support(X) = 1,000 / 10,000 = 0.1

$$\text{Confidence} = \frac{0.06}{0.1} = 0.6 = 60\%$$

Check thresholds

- Minimum support required = 5% → Our support = 6% (meets requirement).
- Minimum confidence required = 50% → Our confidence = 60% (meets requirement).

Since both exceed the minimum thresholds (5% and 50%), the rule is considered strong.

F-p growth algorithm

- Definition
- Need
- Working

Definition

"The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance."

- For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

Need

The two primary drawbacks of the Apriori Algorithm are:

- At each step, candidate sets have to be built.
- To build the candidate sets, the algorithm has to repeatedly scan the database.

- These two properties inevitably make the algorithm slower.
- To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm.
- It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a Trie Data Structure.

Basic Terms :

1. Transaction Database : A collection of transactions, where each transaction contains a set of items.
2. Itemset : A group of one or more items.
3. Minimum Support Threshold : A user-defined cutoff.
4. Frequent Itemset : An itemset whose support ≥ minimum support threshold.

5. FP-Tree (Frequent Pattern Tree) : A **compressed prefix-tree structure** that stores transactions in a compact way. It groups common prefixes of transactions and records counts.Example: If transactions are {A,B,E} and {A,B,C}, FP-Tree will share the prefix {A,B}.
6. Header Table : A table storing each frequent item, its total frequency, and pointers to its occurrences in the FP-tree. Used to navigate the FP-tree quickly.
7. Conditional Pattern Base : The set of prefix paths in the FP-tree that lead to a specific item. Example: For item “C”, we find all paths that end with C.
8. Mining Frequent Patterns : Process of recursively extracting frequent itemsets from the FP-tree using conditional FP-trees. Example: If {B,E} is frequent, we then check which items can join {B,E} to form larger frequent itemsets.

Example

Consider the following transaction database:

The minimum support threshold = 3.

- Tasks:

1. Construct the **FP-tree** from the given transactions.
2. Show the **header table** for the items.
3. Find the conditional pattern base for each item.
4. Generate all the **frequent itemsets** using the FP-Growth algorithm.

TID	Items
T1	{A, B, D, E}
T2	{B, C, E}
T3	{A, B, C, E}
T4	{B, E}
T5	{A, B, C, E}

Solution

Step 1: Transaction Database

TID	Items
T1	{A, B, D, E}
T2	{B, C, E}
T3	{A, B, C, E}
T4	{B, E}
T5	{A, B, C, E}

Step 2: Count Item Frequencies

Count occurrences of each item across all transactions:

- A → 3
- B → 5
- C → 3
- D → 1
- E → 5

Since `min support = 3`, **D is discarded** (frequency = 1 < 3).

Step 3: Sort Each Transaction by Frequency

We sort items in each transaction according to **global descending frequency order**:

`B (5), E (5), A (3), C (3)`

So transactions become:

- T1: {B, E, A}
- T2: {B, E, C}
- T3: {B, E, A, C}
- T4: {B, E}
- T5: {B, E, A, C}

Step 4: Build the FP-Tree

We insert transactions one by one:

Insert T1: {B, E, A}

```
scss

B(1)
  └── E(1)
        └── A(1)
```

Insert T2: {B, E, C}

```
scss

B(2)
  └── E(2)
        ├── A(1)
        └── C(1)
```

Insert T3: {B, E, A, C}

```
scss

B(3)
  └── E(3)
        ├── A(2)
        │     └── C(1)
        └── C(1)
```

Insert T4: {B, E}

```
scss

B(4)
  └── E(4)
        ├── A(2)
        │     └── C(1)
        └── C(1)
```

Insert T5: {B, E, A, C}

```
scss

B(5)
  └── E(5)
        ├── A(3)
        │     └── C(2)
        └── C(2)
```

✅ Final FP-Tree:

```
scss

B(5)
  └── E(5)
        ├── A(3)
        │     └── C(2)
        └── C(2)
```

Step 5: Create Header Table

The header table links each item to its occurrences in the FP-Tree.

Item	Frequency	Node Links
B	5	Root node
E	5	Child of B
A	3	Child of E
C	3	Child of A, child of E

Step 6: Mining the FP-Tree
We mine from the least frequent item to the most frequent.

(a) Mine item C

- Conditional pattern base for C:
  - Path 1: B → E → A (count = 2)
  - Path 2: B → E (count = 1)
- So conditional transactions:
  - {B, E, A} : 2 times
  - {B, E} : 1 time
- From this we get frequent sets with C:
  - {C} (3)
  - {E, C} (3)
  - {B, C} (3)
  - {A, C} (2 → not frequent)
  - {B, E, C} (2 → not frequent)

(b) Mine item A

- Conditional pattern base for A:
  - Path: B → E (count = 3)
- So conditional transactions:
  - {B, E} : 3 times
- Frequent sets with A:
  - {A} (3)
  - {B, A} (3)
  - {E, A} (3)
  - {B, E, A} (3)

(c) Mine item E

- Conditional pattern base for E:
  - Path: B (count = 5)
- Frequent sets with E:
  - {E} (5)
  - {B, E} (5)

(d) Mine item B

- B is root, appears 5 times.
- So {B} (5)

Step 7: Collect All Frequent Itemsets

Final frequent itemsets (support ≥ 3):

- {B} (5)
- {E} (5)
- {A} (3)
- {C} (3)
- {B, E} (5)
- {B, A} (3)
- {B, C} (3)
- {E, A} (3)
- {E, C} (3)
- {B, E, A} (3)

Points to Remember :

1. FP-Growth is more efficient than Apriori because it avoids candidate generation.
2. It compresses the database into a prefix-tree (FP-Tree).
3. Then it recursively mines conditional trees to generate frequent patterns.

Example

Consider the following data:-

Transaction ID	Items
T1	{E, K, M, N, O, Y}
T2	{D, E, K, N, **O**, Y}
T3	{A, E, K, M}
T4	{C, K, M, U, Y}
T5	{C, E, I, K, O, O}

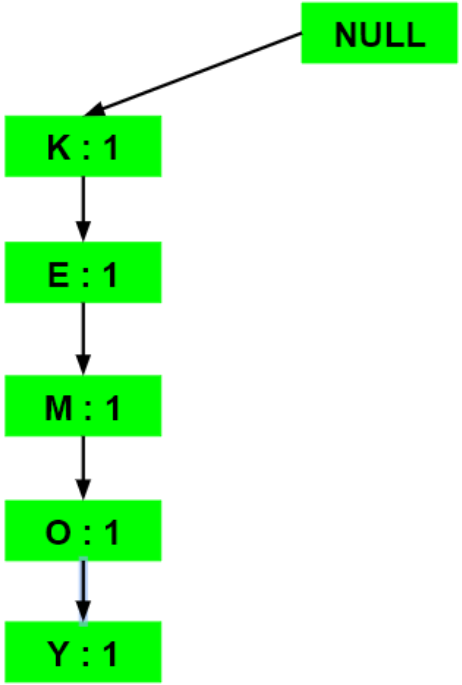
The above-given data is a hypothetical dataset of transactions with each letter representing an item. The frequency of each individual item is computed:-

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

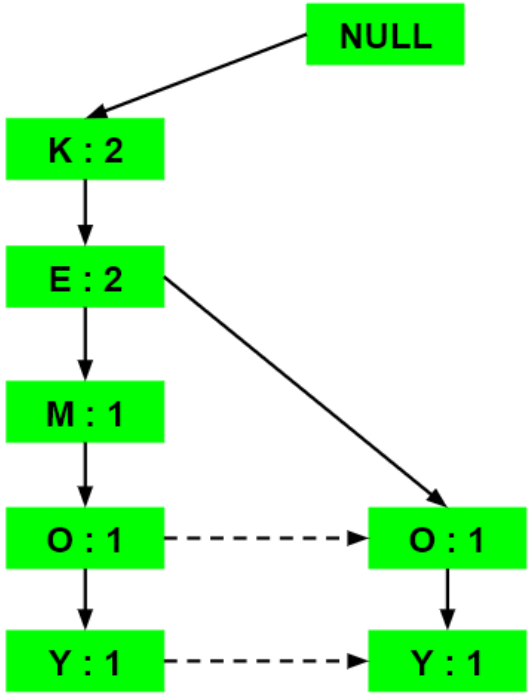
- Let the minimum support be 3. A **Frequent Pattern set** is built which will contain all the elements whose frequency is greater than or equal to the minimum support. These elements are stored in descending order of their respective frequencies. After insertion of the relevant items, the set L looks like this:-
- **L = {K : 5, E : 4, M : 3, O : 3, Y : 3}**
- Now, for each transaction, the respective **Ordered-Item set** is built. It is done by iterating the Frequent Pattern set and checking if the current item is contained in the transaction in question. If the current item is contained, the item is inserted in the Ordered-Item set for the current transaction. The following table is built for all the transactions:

Transaction ID	Items	Ordered-Item Set
T1	{E, K, M, N, O, Y}	{K, E, M, O, Y}
T2	{D, E, K, N, O, Y}	{K, E, O, Y}
T3	{ A, E, K, M}	{K, E, M}
T4	{C, K, M, U, Y}	{K, M, Y}
T5	{C, E, I, K, O, O}	{K, E, O}

- Now, all the Ordered-Item sets are inserted into a Trie Data Structure.
- a) **Inserting the set {K, E, M, O, Y}:**
- Here, all the items are simply linked one after the other in the order of occurrence in the set and initialize the support count for each item as 1.

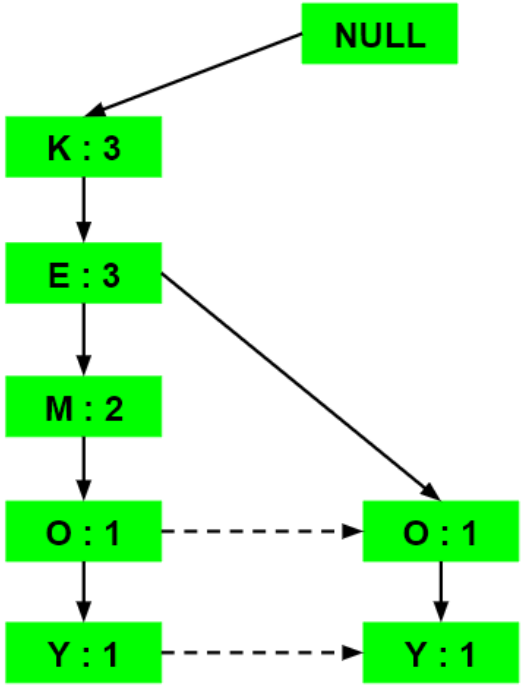


- b) **Inserting the set {K, E, O, Y}:**
- Till the insertion of the elements K and E, simply the support count is increased by 1. On inserting O we can see that there is no direct link between E and O, therefore a new node for the item O is initialized with the support count as 1 and item E is linked to this new node. On inserting Y, we first initialize a new node for the item Y with support count as 1 and link the new node of O with the new node of Y.

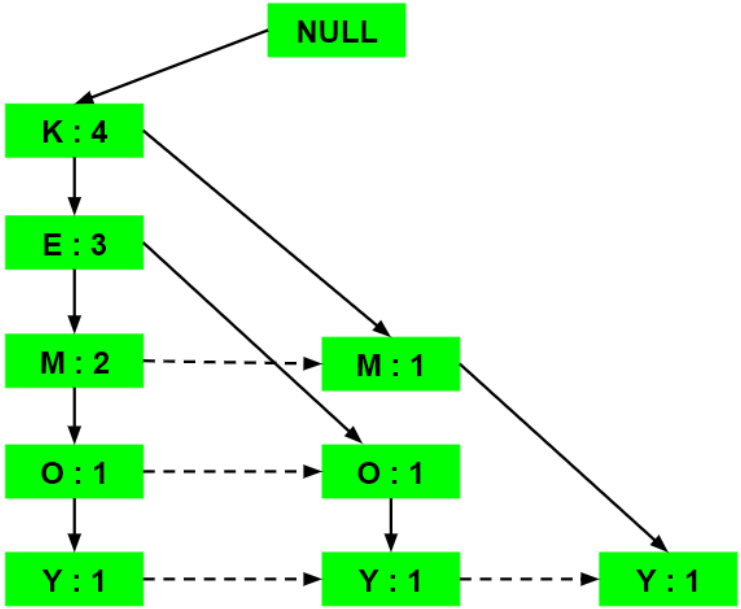


c) **Inserting the set {K, E, M}:**

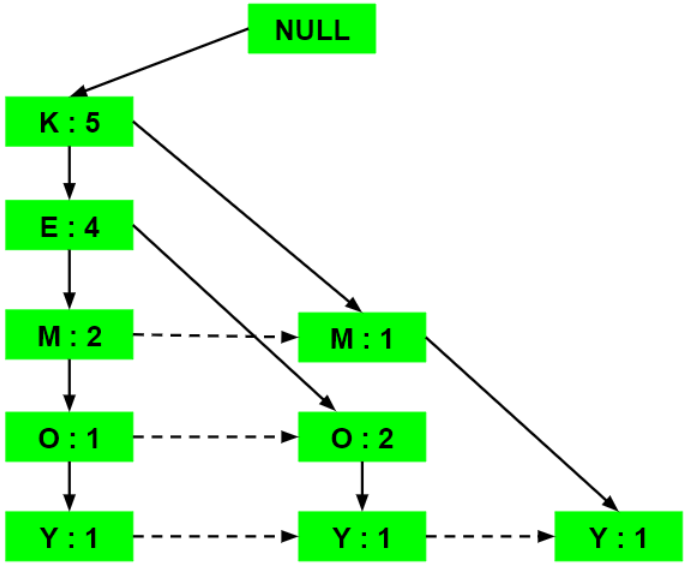
- Here simply the support count of each element is increased by 1.



- d) **Inserting the set {K, M, Y}:**
- Similar to step b), first the support count of K is increased, then new nodes for M and Y are initialized and linked accordingly.



- e) **Inserting the set {K, E, O}:**
- Here simply the support counts of the respective elements are increased. Note that the support count of the new node of item O is increased.



- Now, for each item, the **Conditional Pattern Base** is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree. Note that the items in the below table are arranged in the ascending order of their frequencies.

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

- Now for each item, the **Conditional Frequent Pattern Tree is built.** It is done by taking the set of elements that is common in all the paths in the Conditional Pattern Base of that item and calculating its support count by summing the support counts of all the paths in the Conditional Pattern Base.

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

- From the Conditional Frequent Pattern tree, the **Frequent Pattern rules** are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding to the item as given in the below table.

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 4>}
K	

- For each row, two types of association rules can be inferred for example for the first row which contains the element, the rules K -> Y and Y -> K can be inferred. To determine the valid rule, the confidence of both the rules is calculated and the one with confidence greater than or equal to the minimum confidence value is retained.

BICLUSTERING

Definition

- Bi-clustering also finds patterns involving subsets of rows and columns, but unlike co-clustering, it focuses on discovering **local patterns**: subsets of rows that are similar **only under a subset of columns** (not necessarily the whole dataset).

- **Use case:** Widely used in **bioinformatics** (gene expression analysis), where a subset of genes may show similar expression patterns under only some experimental conditions.

CO CLUSTERING

- Co-clustering is a technique that simultaneously clusters **rows and columns** of a data matrix.
- Instead of clustering only objects (rows) or only features (columns), it looks for **coherent row–column groups** where subsets of rows are strongly associated with subsets of columns.

Aspect	Co-Clustering	Bi-Clustering
Definition	Simultaneously clusters **all rows and columns** into groups (global partition).	Finds **local submatrices**: subsets of rows similar only on subsets of columns.
Approach	Divides the whole matrix into **blocks**. Each row belongs to exactly one cluster, and each column to one cluster.	Allows **overlap**. A row or column can belong to multiple biclusters depending on context.
Pattern type	**Global structure** – full partition of data.	**Local structure** – subgroups within subsets.
Use cases	Text mining (documents × words), recommender systems (users × items).	Bioinformatics (genes × conditions), microarray analysis.
Advantages	Simple, interpretable, covers entire dataset.	More flexible, can reveal hidden/local patterns.
Disadvantages	May miss local relationships.	Computationally harder; overlapping results can be complex.

Example

Suppose we have a matrix of **students vs subjects** (marks or interest levels):

Student / Subject	Math	Physics	Literature	History
S1	90	85	20	30
S2	88	80	25	35
S3	15	20	95	90
S4	18	22	92	88

Co-Clustering Output

- Groups **(S1, S2)** with **(Math, Physics)**
- Groups **(S3, S4)** with **(Literature, History)**

```
   Math   Physics | Literature History
S1  [Block 1]     |
S2  [Block 1]     |
-----------------------------------
S3                | [Block 2]
S4                | [Block 2]
```

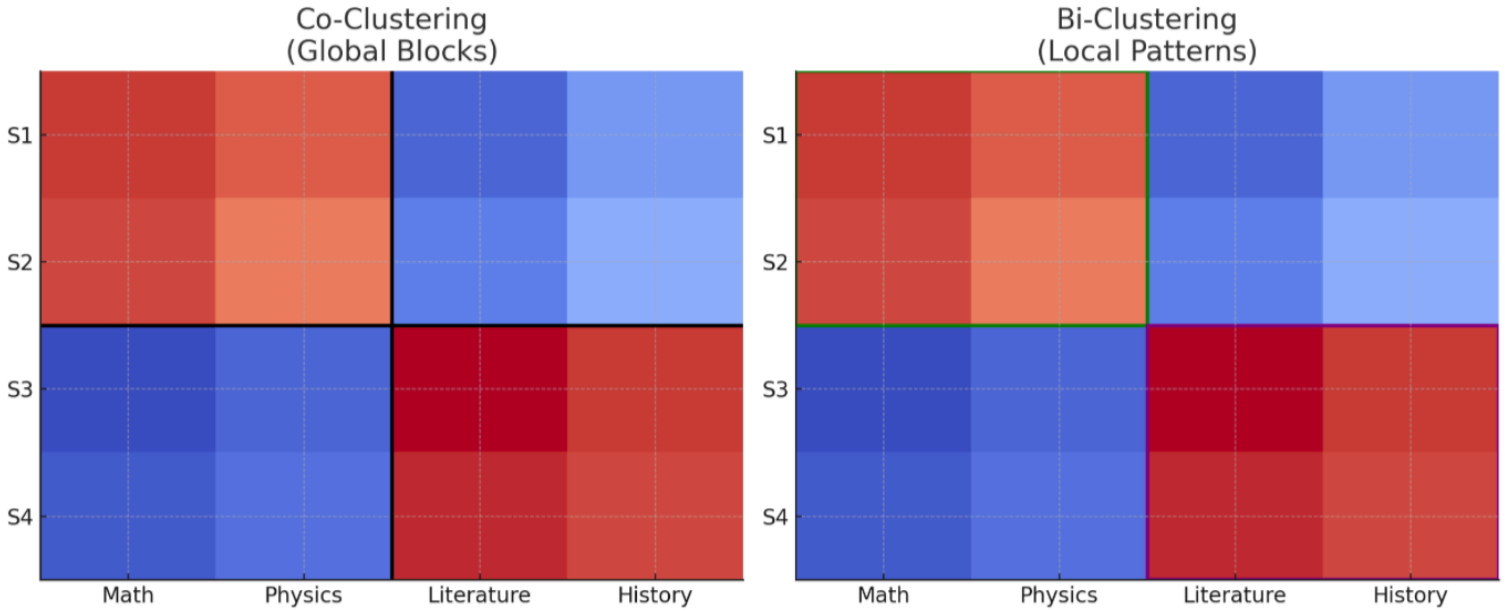
Bi-Clustering Output

- Finds **(S1, S2)** are similar **only for Math & Physics** (STEM bicluster).
- Finds **(S3, S4)** are similar **only for Literature & History** (Arts bicluster).
- But it could also find smaller patterns, e.g. **(S1, S2, S3)** have medium scores under **History**.

Co-clustering = divides whole dataset into **non-overlapping blocks**.

Bi-clustering = finds **flexible, overlapping submatrices** with meaningful patterns.

Visual comparison between Co-Clustering and Bi-Clustering:



- **Left (Co-Clustering):** Whole matrix is divided into 2 **global blocks** (STEM vs Arts).
- **Right (Bi-Clustering):** Flexible **local biclusters** highlighted (green = STEM, purple = Arts), showing overlapping or context-specific patterns.

Applications

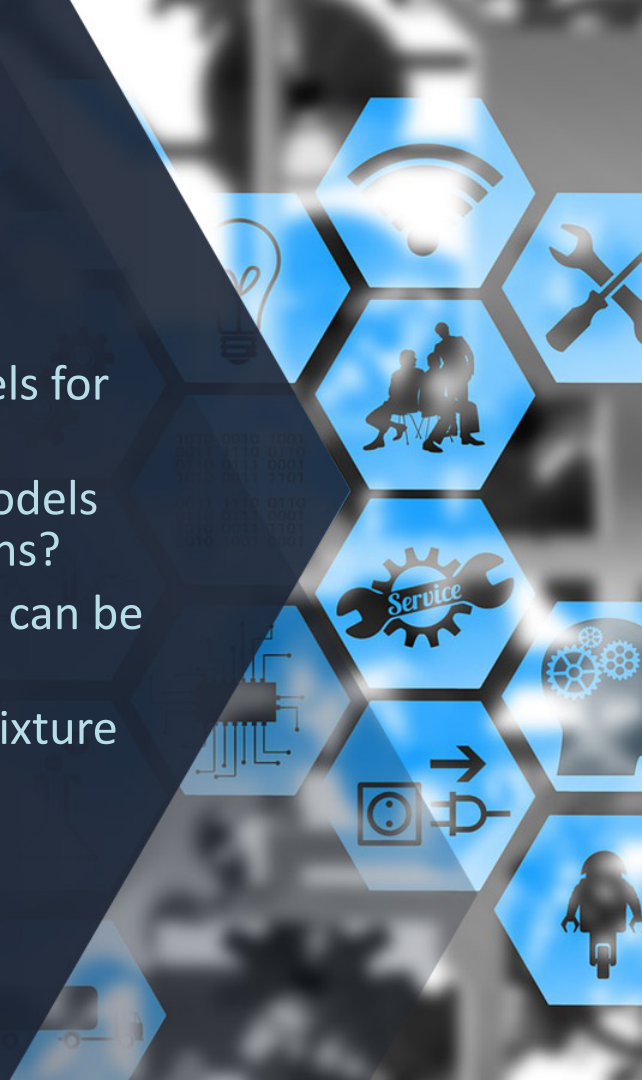
- **Gene expression analysis** → Find subsets of genes that behave similarly under subsets of conditions.
- **Text mining** → Find groups of documents that share specific vocabulary.
- **Recommendation systems** → Find groups of users who like subsets of products.

Types

- **Exclusive-Row & Exclusive-Column**: Each row and column belongs to exactly one bicluster.
- **Overlapping**: Rows/columns can belong to multiple biclusters.
- **Checkerboard structure**: The matrix is reorganized into visible blocks of dense submatrices.

Gaussian Mixture Model

- Definition
- Explanation
- Applications
- What are the key steps of using Gaussian mixture models for clustering?
- What are the differences between Gaussian mixture models and other types of clustering algorithms such as K-means?
- What are the scenarios when Gaussian mixture models can be used?
- What are some real-world examples where Gaussian mixture models can be used?



Definition

"A **Gaussian Mixture Model (GMM)** is a **probabilistic model** used to represent a dataset that is assumed to be generated from a mixture of several Gaussian (Normal) distributions. "

Explanation

- Gaussian mixture models (GMM) are a probabilistic concept used to model real-world data sets.
- GMMs are a generalization of Gaussian distributions and can be used to represent any data set that can be clustered into multiple Gaussian distributions.

Explanation

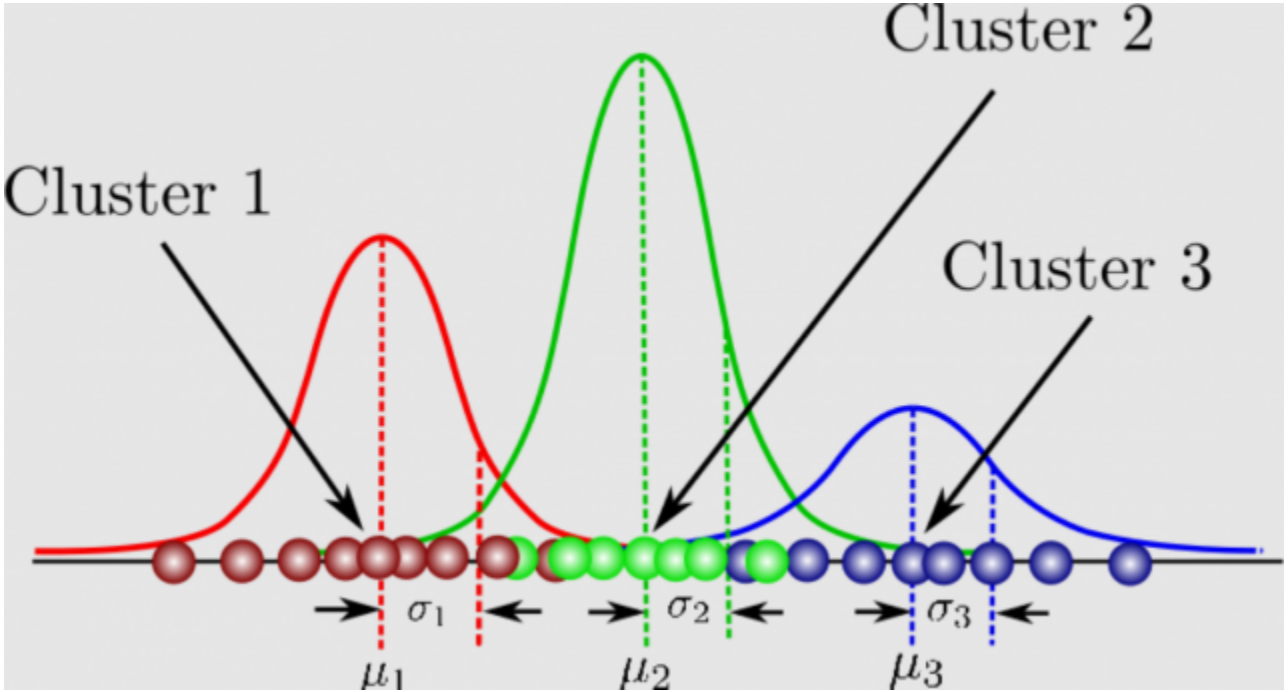
- A Gaussian mixture model can be used for clustering, which is the task of grouping a set of data points into clusters.
- GMMs can be used to find clusters in data sets where the clusters may not be clearly defined. Additionally, GMMs can be used to estimate the probability that a new data point belongs to each cluster.
- Gaussian mixture models are also relatively robust to outliers, meaning that they can still yield accurate results even if there are some data points that do not fit neatly into any of the clusters.
- This makes GMMs a flexible and powerful tool for clustering data. It can be understood as a probabilistic model where Gaussian distributions are assumed for each group and they have means and covariances which define their parameters.

Explanation

GMM consists of two parts

- – mean vectors (μ)
- & covariance matrices (Σ).
- A Gaussian distribution is defined as a continuous probability distribution that takes on a bell-shaped curve. Another name for Gaussian distribution is the normal distribution.

Here is a picture of Gaussian mixture models:



Applications

- GMM has many applications, such as density estimation, clustering, and image segmentation.
- For density estimation, GMM can be used to estimate the probability density function of a set of data points.
- For clustering, GMM can be used to group together data points that come from the same Gaussian distribution.
- And for image segmentation, GMM can be used to partition an image into different regions.
- Gaussian mixture models can be used for a variety of use cases, including identifying customer segments, detecting fraudulent activity, and clustering images.
- In each of these examples, the Gaussian mixture model is able to identify clusters in the data that may not be immediately obvious.
- As a result, Gaussian mixture models are a powerful tool for data analysis and should be considered for any clustering task.

What are the key steps of using Gaussian mixture models for clustering?

The following are three different steps to using gaussian mixture models:

- Determining a covariance matrix that defines how each Gaussian is related to one another. The more similar two Gaussians are, the closer their means will be and vice versa if they are far away from each other in terms of similarity. A gaussian mixture model can have a covariance matrix that is diagonal or symmetric.
- Determining the number of Gaussians in each group defines how many clusters there are.
- Selecting the hyperparameters which define how to optimally separate data using gaussian mixture models as well as deciding on whether or not each gaussian's covariance matrix is diagonal or symmetric.

What are the differences between Gaussian mixture models and other types of clustering algorithms such as K-means?

Aspect	K-Means	Gaussian Mixture Model (GMM)
Clustering Type	**Hard clustering** → each point belongs to exactly one cluster.	**Soft clustering** → each point has a probability of belonging to each cluster.
Cluster Shape Assumption	Assumes **spherical clusters** (equal size, isotropic). Boundaries are circular in 2D.	Can model **elliptical clusters** (different shapes, sizes, orientations) because it uses covariance matrices.
Assignment Rule	Assigns a point to the **nearest centroid** (using Euclidean distance).	Assigns a point based on **maximum probability density** from Gaussian components.
Model Assumption	No probabilistic model, just minimizes **within-cluster variance**.	Probabilistic model → data is generated from a mixture of Gaussian distributions.
Parameters Learned	Only **centroids (μ)** of clusters.	Learns **means (μ), covariances (Σ), and mixing weights (π)**.
Output	Gives cluster labels (e.g., point → Cluster 1).	Gives **cluster probabilities** (e.g., point → 70% Cluster 1, 30% Cluster 2).

Flexibility	Rigid, works well when clusters are roughly spherical and similar in size.	More flexible, works well for **elongated, overlapping, or differently sized clusters**.
Optimization Method	Iteratively minimizes **Sum of Squared Distances**.	Uses the **Expectation-Maximization (EM) algorithm** to maximize likelihood.
Scalability	Fast and efficient, scales well to large datasets.	Slower than K-means (due to probability computations & covariance estimation).
Use Cases	Simple clustering, image compression, customer segmentation when clusters are well-separated.	More complex clustering, density estimation, anomaly detection, speech recognition.

What are the scenarios when Gaussian mixture models can be used?

- The following are different scenarios when GMMs can be used:
- Gaussian mixture models can be used in a variety of scenarios, including when data is generated by a mix of Gaussian distributions when there is uncertainty about the correct number of clusters, and when clusters have different shapes. In each of these cases, the use of a Gaussian mixture model can help to improve the accuracy of results. For example, when data is generated by a mix of Gaussian distributions, using a Gaussian mixture model can help to better identify the underlying patterns in the data. In addition, when there is uncertainty about the correct number of clusters, the use of a Gaussian mixture model can help to reduce the error rate.
- Gaussian mixture models can be used for anomaly detection; by fitting a model to a dataset and then scoring new data points, it is possible to flag points that are significantly different from the rest of the data (i.e. outliers). This can be useful for identifying fraud or detecting errors in data collection.
- In the case of time series analysis, GMMs can be used to discover how volatility is related to trends and noise which can help predict future stock prices. One cluster could consist of a trend in the time series while another can have noise and volatility from other factors such as seasonality or external events which affect the stock price. In order to separate out these clusters, GMMs can be used because they provide a probability for each category instead of simply dividing the data into two parts such as that in the case of K-means.
- Another example is when there are different groups in a dataset and it’s hard to label them as belonging to one group or another which makes it difficult for other machine learning algorithms such as the K-means clustering algorithm to separate out the data. GMMs can be used in this case because they find Gaussian mixture models that best describe each group and provide a probability for each cluster which is helpful when labeling clusters.
- Gaussian mixture models can generate synthetic data points that are similar to the original data, they can also be used for data augmentation.

What are some real-world examples where Gaussian mixture models can be used?

- **Finding patterns in medical datasets**: GMMs can be used for segmenting images into multiple categories based on their content or finding specific patterns in medical datasets. They can be used to find clusters of patients with similar symptoms, identify disease subtypes, and even predict outcomes. In one recent study, a Gaussian mixture model was used to analyze a dataset of over 700,000 patient records. The model was able to identify previously unknown patterns in the data, which could lead to better treatment for patients with cancer.
- **Modeling natural phenomena**: GMM can be used to model natural phenomena where it has been found that noise follows Gaussian distributions. This model of probabilistic modeling relies on the assumption that there exists some underlying continuum of unobserved entities or attributes and that each member is associated with measurements taken at equidistant points in multiple observation sessions.
- **Customer behavior analysis**: GMMs can be used for performing customer behavior analysis in marketing to make predictions about future purchases based on historical data.
- **Stock price prediction**: Another area Gaussian mixture models are used is in finance where they can be applied to a stock's price time series. GMMs can be used to detect changepoints in time series data and help find turning points of stock prices or other market movements that are otherwise difficult to spot due to volatility and noise.
- **Gene expression data analysis**: Gaussian mixture models can be used for gene expression data analysis. In particular, GMMs can be used to detect differentially expressed genes between two conditions and identify which genes might contribute toward a certain phenotype or disease state.