RGPVNOTES.IN

Please do not share these notes on apps like WhatsApp or Telegram.

The revenue we generate from the ads we show on our website and app funds our services. The generated revenue **helps us prepare new notes and improve the quality of existing study materials**, which are available on our website and mobile app.

If you don't use our website and app directly, it will hurt our revenue, and we might not be able to run the services and **have to close them**. So, it is a humble request for all to **stop sharing the study material** we provide on various apps. Please **share the website's URL instead.**

CS-703(B) Open Elective, Data Mining and Warehousing

- UNIT-V: Clustering & Association Rule mining: Hierarchical algorithms, Partitional algorithms,

Clustering large databases – BIRCH, DBSCAN, CURE algorithms. Association rules: Parallel and distributed algorithms such as Apriori and FP growth algorithms.

Introduction:

Cluster Analysis in Data Mining

Cluster Analysis in Data Mining means that to find out the group of objects which are similar to each other in the group but are different from the object in other groups.

In clustering, a group of different data objects is classified as similar objects. One group means a cluster of data. Data sets are divided into different groups in the cluster analysis, which is based on the similarity of the data. After the classification of data into various groups, a label is assigned to the group. It helps in adapting to the changes by doing the classification.

- Clustering is the method of converting a group of abstract objects into classes of similar objects.
- Clustering is a method of partitioning a set of data or objects into a set of significant subclasses called clusters.
- It helps users to understand the structure or natural grouping in a data set and used either as a stand-alone instrument to get a better insight into data distribution or as a pre-processing step for other algorithms.

Association Rule Mining:

Association rules are 'if/then' statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository. An example of an association rule would be "If a customer buys a dozen eggs, he is 80% likely to also purchase milk." Association rule mining is the data mining process of finding the rules that may govern associations and causal objects between sets of items.

Association rules analysis is a technique to uncover how items are associated to each other. There are three common ways to measure association.

Measure 1: **Support**

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

Measure 2: Confidence

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

Measure 3: Lift

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

The main applications of association rule mining:

- Basket data analysis
- Cross marketing
- Catalog design
- Web Mining
- Medical Analysis
- Bioinformatics
- Network Analysis
- Programming Pattern Finding
- Clustering, Classification, etc.

Hierarchical algorithms:

Hierarchical clustering is an alternative approach to partitioning clustering for identifying groups in the dataset. It does not require to pre-specify the number of clusters to be generated.

The result of hierarchical clustering is a tree-based representation of the objects, which is also known as dendrogram. Observations can be subdivided into groups by cutting the dendrogram at a desired similarity level.

R code to compute and visualize hierarchical clustering:

Compute hierarchical clustering

res.hc <- USArrests %>% scale() %>%

Scale the data dist(method = "euclidean") %>%

Compute dissimilarity matrix hclust(method = "ward.D2")

Compute hierachical clustering # Visualize using factoextra

Cut in 4 groups and color by groups fviz_dend(res.hc, k = 4,

Cut in four groups cex = 0.5,

label size k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"), color_labels_by_k = TRUE,

color labels by groups rect = TRUE # Add rectangle around groups)

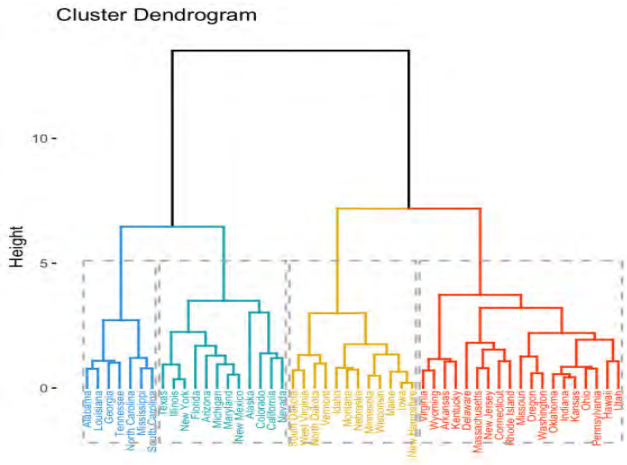

Figure 1: Hierarchical method

The method will create a hierarchical decomposition of a given set of data objects. Based on how the hierarchical decomposition is formed, we can classify hierarchical methods. This method is given as follows:

- Agglomerative Approach
- Divisive Approach

Agglomerative Approach is also known as Button-up Approach. Here we begin with every object that constitutes a separate group. It continues to fuse objects or groups close together.

Divisive Approach is also known as the Top-Down Approach. We begin with all the objects in the same cluster. This method is rigid, i.e., it can never be undone once a fusion or division is completed.

Approaches to Improve Quality of Hierarchical Clustering

Here are the two approaches that are used to improve the quality of hierarchical clustering −

- Perform careful analysis of object linkages at each hierarchical partitioning.
- Integrate hierarchical agglomeration by first using a hierarchical agglomerative algorithm to group objects into micro-clusters, and then performing macro-clustering on the micro-clusters.

Partitional Algorithms:

Suppose, given a database of 'n' objects and the partitioning method constructs 'k' partition of data. Each partition will represent a cluster and k ≤ n. It means that it will classify the data into k groups, which satisfy the following requirements −

- Each group contains at least one object.
- Each object must belong to exactly one group.

Points to remember −

- For a given number of partitions (say k), the partitioning method will create an initial partitioning.
- Then it uses the iterative relocation technique to improve the partitioning by moving objects from one group to other.

Partitioning algorithms are clustering techniques that subdivide the data sets into a set of k groups, where k is the number of groups pre-specified by the analyst.

There are different types of partitioning clustering methods. The most popular is the K-means clustering (MacQueen 1967), in which, each cluster is represented by the center or means of the data points belonging to the cluster. The K-means method is sensitive to outliers.
An alternative to k-means clustering is the K-medoids clustering or PAM (Partitioning Around Medoids, Kaufman & Rousseeuw, 1990), which is less sensitive to outliers compared to k-means.
The following R codes show how to determine the optimal number of clusters and how to compute k-means and PAM clustering in R.

library ("factoextra") fviz_nbclust(my_data, kmeans, method = "gap_stat")
set.seed(123) km.res <- kmeans(my_data, 3, nstart = 25)
# Visualize library("factoextra") fviz_cluster(km.res, data = my_data, ellipse.type = "convex", palette = "jco", ggtheme = theme_minimal())
# Compute PAM library("cluster") pam.res <- pam(my_data, 3)
# Visualize fviz_cluster(pam.res)

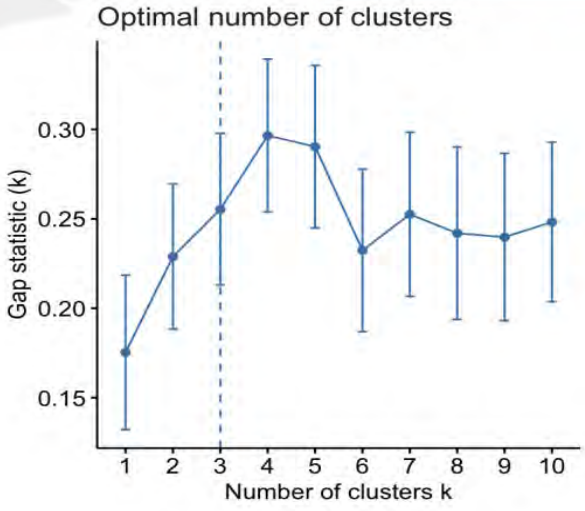

Figure 2: Partitional Clustering

Clustering Large Databases:

BIRCH Algorithms

BIRCH stands for Balanced Iterative Reducing and Clustering Using Hierarchies, which uses hierarchical methods to cluster and reduce data.

BIRCH only needs to scan the data set in a single pass to perform clustering. The BIRCH algorithm is more suitable for the case where the amount of data is large and the number of categories K is relatively large. It runs very fast, and it only needs a single pass to scan the data set for clustering.
The BIRCH algorithm uses a tree structure to create a cluster. It is generally called the Clustering Feature Tree (CF Tree). Each node of this tree is composed of several Clustering Features (CF).
Clustering Feature tree structure is similar to the balanced B+ tree from the figure-3 below, we can see what the clustering feature tree looks like.
Each node including leaf nodes has several CFs, and the CFs of internal nodes have pointers to child nodes, and all leaf nodes are linked by a doubly linked list.
A CF Tree structure is given as below:

- Each non-leaf node has at most B entries.
- Each leaf node has at most L CF entries which satisfy threshold T, a maximum diameter of radius.
- P (page size in bytes) is the maximum size of a node.
- Compact: each leaf node is a sub-cluster, not a data point.

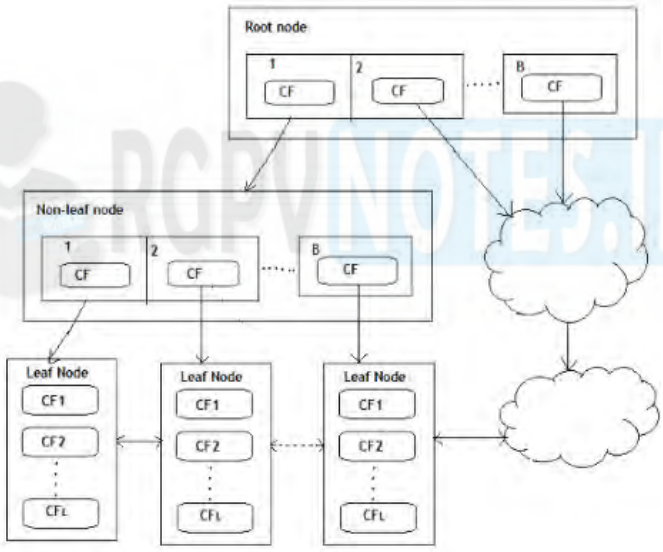

Figure 3: CF Tree

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

- It is a scalable clustering method.
- Designed for very large data sets.
- Only one scan of data is necessary.
- It is based on the notation of CF (Clustering Feature) tree.
- CF tree is a height balanced tree that stores the clustering features for a hierarchical clustering.
- Cluster of data points is represented by a triple of numbers (N,LS,SS), where

N= Number of items in the sub cluster, LS=Linear sum of the points, SS=sum of the squared of the points

DBSCAN Algorithm:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a very famous density-based clustering algorithm. Intuitively, the DBSCAN algorithm can find all the dense regions of the sample points and treat these dense regions as clusters one by one.

The DBSCAN algorithm has the following characteristics:

- Density-based, robust to noise points which are away from the density core
- No need to know the number of clusters
- Can create clusters of arbitrary shape

DBSCAN is usually suitable for cluster analysis of lower-dimensional data.

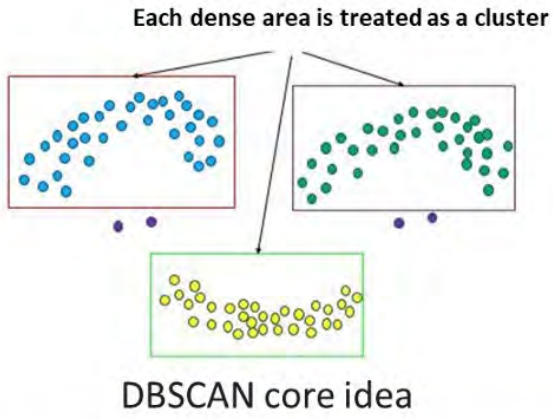

Figure 4: DBSCAN

Basic concept

The basic concepts of DBSCAN can be summarized by 1, 2, and 3.

1. Core idea: Based on density

2. Algorithm parameters: Neighborhood radius (eps/ε) and the minimum number of points (min_points).

These two algorithm parameters can describe what is dense in the cluster.

When the number of points within the neighborhood radius (eps) is greater than the minimum number of points (min_points), it is dense.

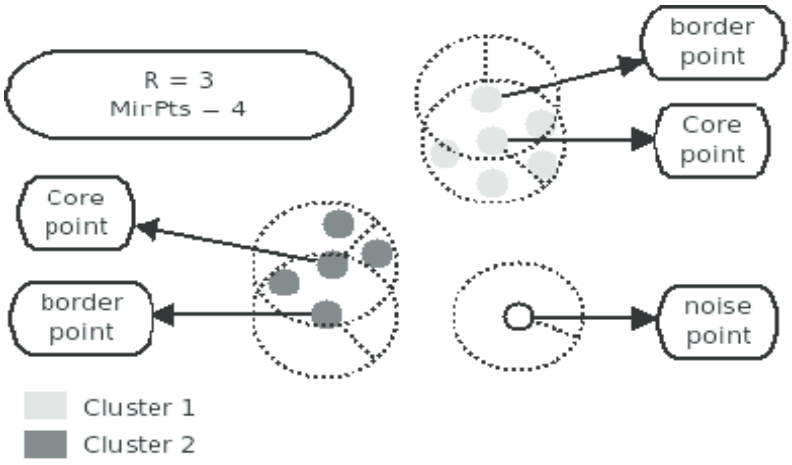

Figure 5: DBSCAN parameter

3. Types of points: Core points, boundary points, and noise points

- The point where the number of sample points in the neighborhood radius (eps) is greater than or equal to min points is called the core point.
- Points that do not belong to the core point but are within the neighborhood of a certain core point are called boundary points.
- Noise points are neither core points nor boundary points.

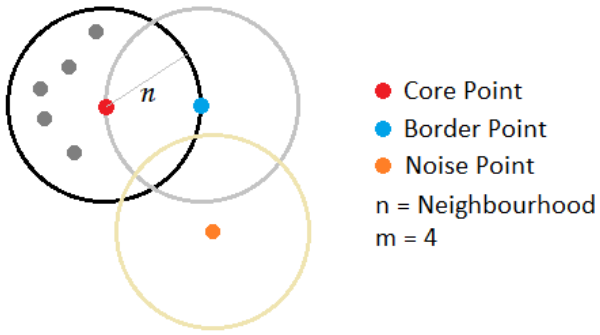

Figure 6: DBSCAN points

DBSCAN Algorithm Steps:

Pseudo Code for main function

DBSCAN (data, Eps, MinPts) :

For x_i in the data set:

    ClusterID =1

    If x_i is UNCLASSIFIED

        ExpandCluster(x_i ,*ClusterID*)

        If *ExpandCluster* successful

            ClusterID = *ClusterID* +1

        End

    End

End *//end main function*

The algorithm step of DBSCAN is divided into two steps:

1**.** Find the core point to form a temporary cluster:

Scan all sample points. If the number of points within a radius of a certain sample point is> = MinPoints, It will be included in the list of core points, and the points with direct density will form corresponding temporary clusters.

2. Combine temporary clusters to obtain clusters:

For each temporary cluster, check whether the point in it is a core point. If so, merge the temporary cluster corresponding to the point with the current temporary cluster to obtain a new temporary cluster.

This operation is repeated until each point in the current temporary cluster is either not in the core point list, or the point with a direct density is already in the temporary cluster, and the temporary cluster is upgraded to a cluster.

Continue to perform the same merge operation on the remaining temporary clusters until all the temporary clusters are processed.

CURE Algorithms

CURE (Clustering Using REpresentatives) is an efficient data clustering algorithm for large databases. Compared with K-means clustering it is more robust to outliers and able to identify clusters having non-spherical shapes and size variances.

- It is a hierarchical based clustering technique that adopts a middle ground between the centroid based and the all-point extremes. Hierarchical clustering is a type of clustering that starts with a single point cluster and moves to merge with another cluster, until the desired number of clusters are formed.
- It is used for identifying the spherical and non-spherical clusters.
- It is useful for discovering groups and identifying interesting distributions in the underlying data.
- Instead of using one point centroid, as in most of data mining algorithms, CURE uses a set of well-defined representative points, for efficiently handling the clusters and eliminating the outliers.

Six steps in CURE algorithm:

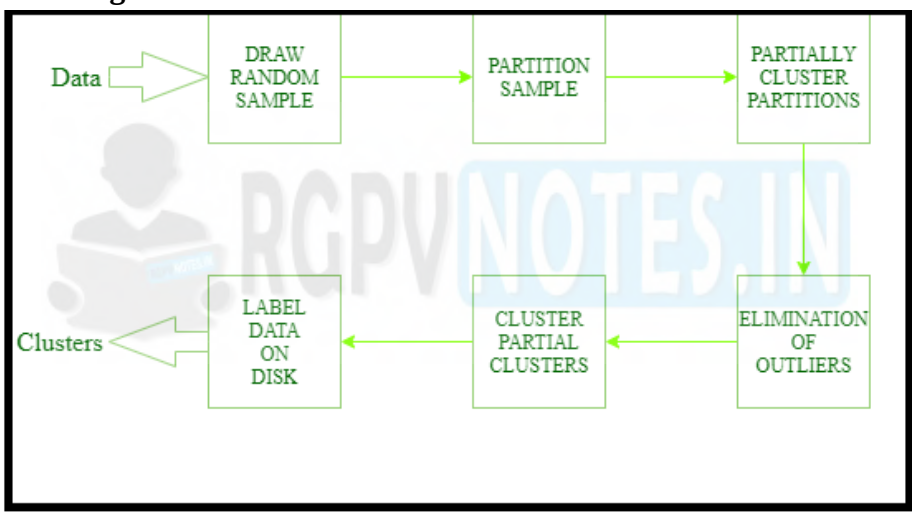

Figure 7: Six steps in CURE algorithm

CURE Algorithm

• Input: k, the number of clusters
• Draw a random sample of the points
• Each point is its own cluster initially
• Each cluster stores a point of representative points and a mean
• Build a kd-tree of all clusters
• Use the tree to build a heap that stores u.closest for each cluster u
• While size(heap) > k:
    - Merge together the two closest clusters in the heap
    - Update the representative points in each cluster
    - Update the tree and the heap

Merging step:

- New mean is a mean of the means of the clusters being merged

- Select c well-scattered points based on their distance from the new mean
- Shrink each representative point closer to the mean

The two stages of CURE implementation are as follows:

1) Initialization in CURE

• Take a small sample data and cluster it in main memory using hierarchical clustering algorithm.

• Select a small set of points from each cluster to be representative points. These points should be chosen to be as far from one another as possible.

• Move each of the representative points a fixed fraction of the distance between its location and the centroid of its cluster. The fraction could be about 20% or 30% of the original distance.

2) Completion of CURE Algorithm

- Once initialization is complete, we have to cluster the renaming points and output the final cluster.
- The final step of CURE is to merge two clusters if they have a pair of representative points, one from each cluster, that are sufficiently close. The user may pick the distance threshold. The merging step can be repeated until there are no more sufficiently close clusters.
- Each point P is brought from secondary storage and compared with the representative points. We assign p to the cluster of the representative point that is closes to P.

Figures illustrates CURE Algorithm:

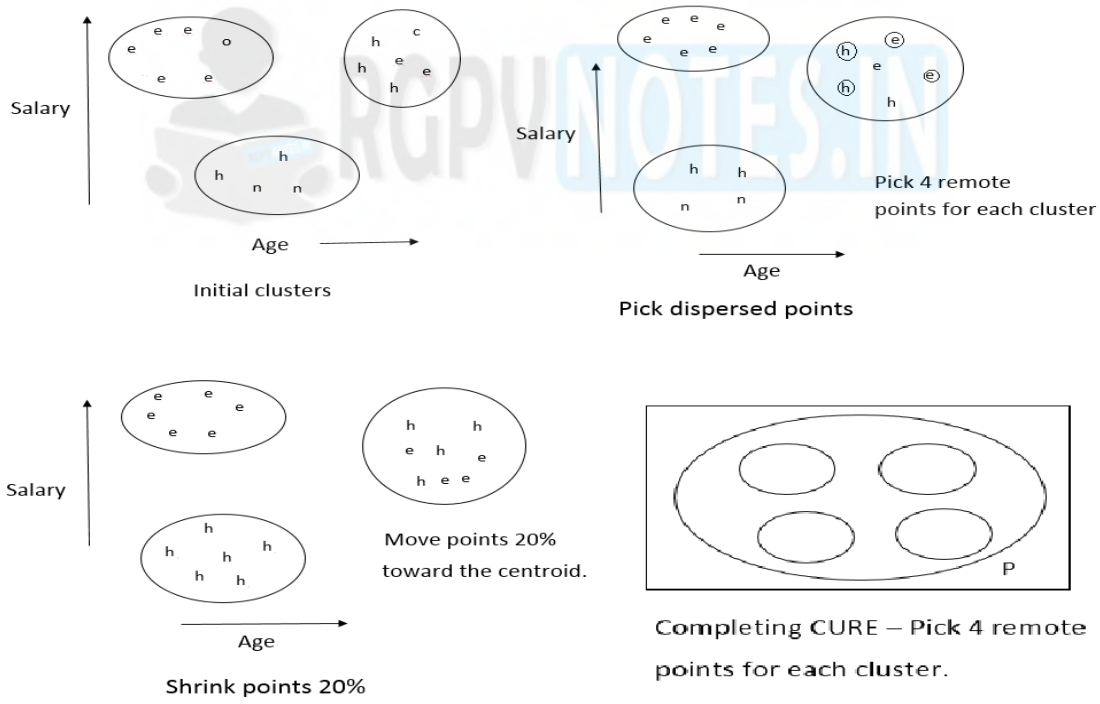

Figure 8: Representation of partitioning and clustering

Association rules:

Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently itemset occurs in a transaction. A typical example is Market Based Analysis.

Market Based Analysis is one of the key techniques used by large relations to show associations between items. It allows retailers to identify relationships between the items that people buy together frequently.

Association rules are "if-then" statements, which help to show the probability of relationships between data items, within large data sets in various types of databases.

Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. **Association rule** generation is usually split up into two separate steps: A minimum support threshold is applied to find all frequent itemsets in a database.

Association Rules find all sets of items (itemsets) that have ***support*** greater than the minimum support and then using the large itemsets to generate the desired rules that have ***confidence*** greater than the minimum confidence. The ***lift*** of a rule is the ratio of the observed support to that expected if X and Y were independent.

$$Support = \frac{frq(X,Y)}{N}$$

$$Rule: X \Rightarrow Y \longrightarrow Confidence = \frac{frq(X,Y)}{frq(X)}$$

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

Example

Rule	Support	Confidence	Lift
$A \Rightarrow D$	2/5	2/3	10/9
$C \Rightarrow A$	2/5	2/4	5/6
$A \Rightarrow C$	2/5	2/3	5/6
$B \& C \Rightarrow D$	1/5	1/3	5/9

Figure 9: Association Rule Example

Parallel and Distributed Algorithms:

Parallel and distributed computing is expected to relieve current mining methods from the sequential bottleneck, providing the ability to scale to massive datasets, and improving the response time. Achieving good performance on today's multiprocessor systems is a non-trivial task. The main challenges include synchronization and communication minimization, work-load balancing, finding good data layout and data decomposition, and disk I/O minimization, which is especially important for data mining.

Apriori Algorithms:

Apriori algorithm is a classical algorithm in data mining. It is used for mining frequent itemsets and relevant association rules. It is devised to operate on a database containing a lot of transactions, for instance, items brought by customers in a store.

1. Candidate itemsets are generated using only the large itemsets of the previous pass without considering the transactions in the database.
2. The large itemset of the previous pass is joined with itself to generate all itemsets whose size is higher by 1.
3. Each generated itemset that has a subset which is not large is deleted. The remaining itemsets are the candidate ones.

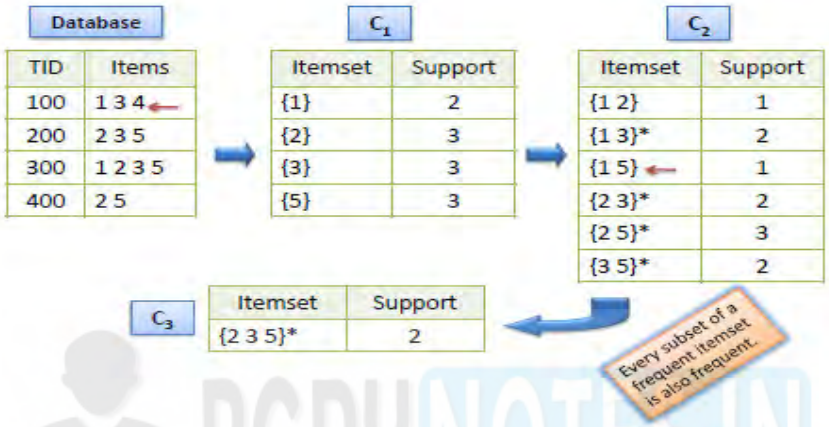

Figure 10: Apriori Example

The Apriori algorithm takes advantage of the fact that any subset of a frequent itemset is also a frequent itemset. The algorithm can therefore, reduce the number of candidates being considered by only exploring the itemsets whose support count is greater than the minimum support count. All infrequent itemsets can be pruned if it has an infrequent subset.

FP Growth Algorithms:

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.

This tree structure will maintain the association between the itemsets. The database is fragmented using one frequent item. This fragmented part is called “pattern fragment”. The itemsets of these fragmented patterns are analyzed. Thus with this method, the search for frequent itemsets is reduced comparatively.

The FP-Growth Algorithm, proposed by Han, is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree).

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

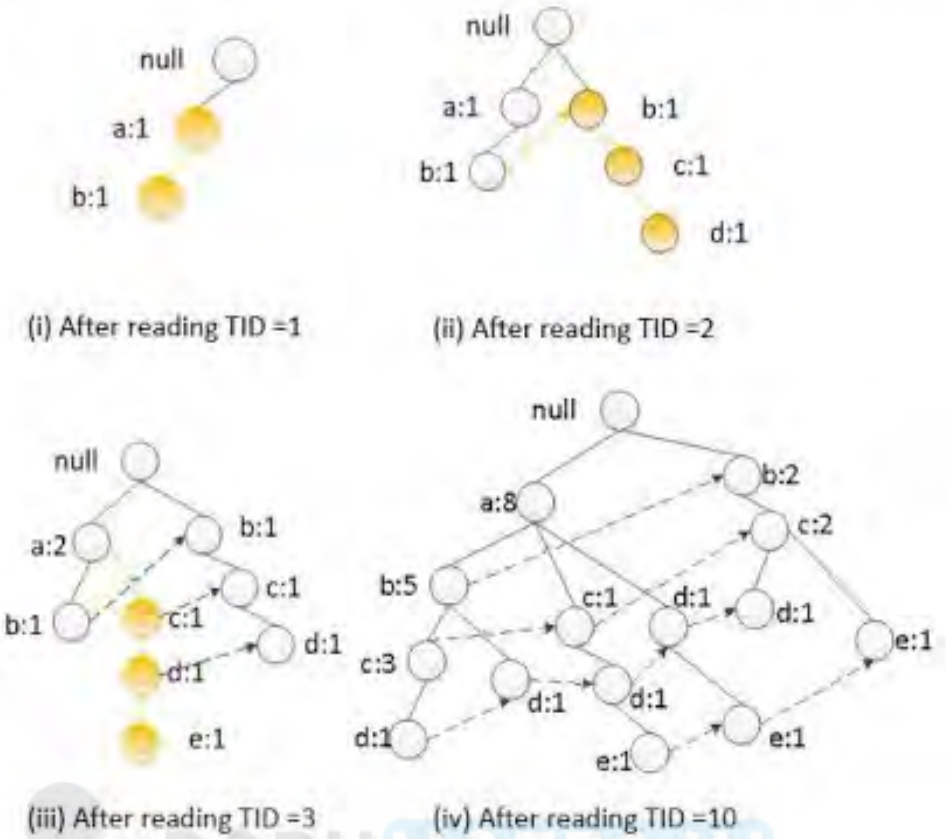

Figure 11: FP Growth (Example-1)

Frequent Itemset Using FP-Growth (Example)

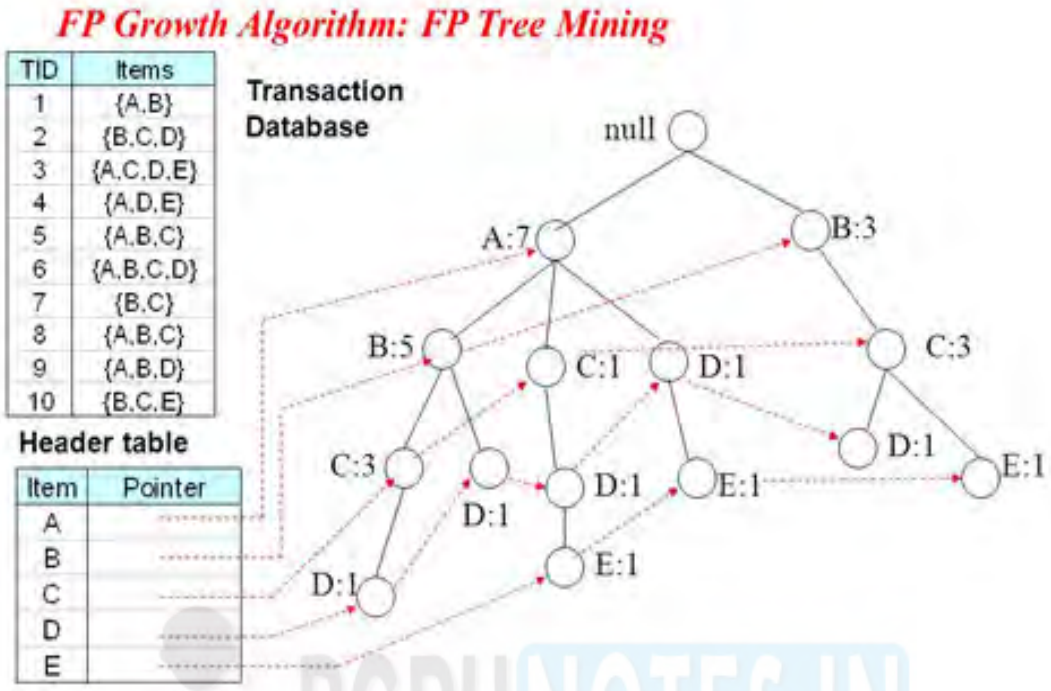

Figure 12: FP Growth (Example-2)

Advantages of FP Growth Algorithm

1. This algorithm needs to scan the database only twice when compared to Apriori which scans the transactions for each iteration.
2. The pairing of items is not done in this algorithm and this makes it faster.
3. The database is stored in a compact version in memory.
4. It is efficient and scalable for mining both long and short frequent patterns.

Disadvantages of FP-Growth Algorithm

1. FP Tree is more cumbersome and difficult to build than Apriori.
2. It may be expensive.
3. When the database is large, the algorithm may not fit in the shared memory.

RGPVNOTES.IN

Thank you for using our services. Please support us so that we can improve further and help more people.
https://www.rgpvnotes.in/support-us

If you have questions or doubts, contact us on
WhatsApp at +91-8989595022 or by email at hey@rgpvnotes.in.

For frequent updates, you can follow us on
Instagram: https://www.instagram.com/rgpvnotes.in/.