

- ✓ To comprehend the concept of clustering, its applications, and features.
- ✓ To understand various distance metrics for clustering of data.
- ✓ To comprehend the process of K-means clustering.
- ✓ To comprehend the process of hierarchical clustering algorithms.
- ✓ To comprehend the process of DBSCAN algorithms.

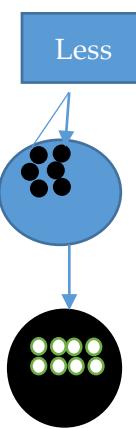
Less

7.1 Introduction to Cluster Analysis

Generally, in the case of large datasets, data is not labeled because labeling a large number of records requires a great deal of human effort. The unlabeled data can be analyzed with the help of clustering techniques. Clustering is an unsupervised learning technique which does not require a labeled dataset.

Clustering is defined as grouping a set of similar objects into classes or clusters. In other words, during cluster analysis, the data is grouped into classes or clusters, so that records within a cluster (**intra-cluster**) have high similarity with one another but have high dissimilarities in comparison to objects in other clusters (**inter-cluster**), as shown in Figure 7.1.

The similarity of records is identified on the basis of values of attributes describing the objects. Cluster analysis is an important human activity. The first human beings Adam and Eve actually learned through the process of clustering. They did not know the name of any object, they simply observed each and every object. Based on the similarity of their properties, they identified these objects in groups or clusters. For example, one group or cluster was named as trees, another as fruits and so on. They further classified the fruits on the basis of their properties like size, colour, shape, taste, and others. After that, people assigned labels or names to these objects calling them mango, banana, orange, and so on. And finally, all objects were labeled. Thus, we can say that the first



human beings used clustering for their learning and they made clusters or groups of physical objects based on the similarity of their attributes.

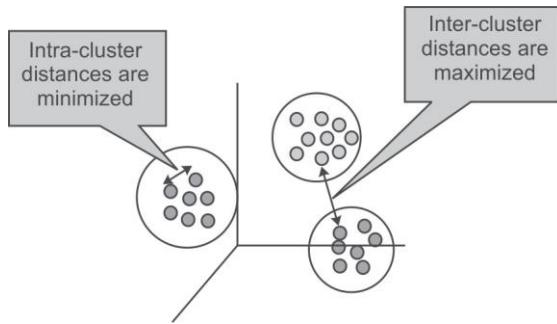


Figure 7.1 Characteristics of clusters

7.2 Applications of Cluster Analysis

Cluster analysis has been widely used in various important applications such as:

- Marketing: It helps marketers find out distinctive groups among their customer bases, and this knowledge helps them improve their targeted marketing programs.
- Land use: Clustering is used for identifying areas of similar land use from the databases of earth observations.
- Insurance: Clustering is helpful for recognizing clusters of insurance policyholders with a high regular claim cost.
- City-planning: It also helps in identifying clusters of houses based on house type, geographical location, and value.
- Earthquake studies: Clustering is helpful for analysis of earthquakes as it has been noticed that earthquake epicenters are clustered along continent faults.
- Biology studies: Clustering helps in defining plant and animal classifications, identifying genes with similar functionalities, and in gaining insights into structures inherent to populations.
- Web discovery: Clustering is helpful in categorizing documents on the web for information discovery.
- Fraud detection: Clustering is also helpful in outlier detection applications such as credit card fraud detection.

7.3

7.4

7.5

7.6

7.7

7.8

7.9

7.10 Desired Features of Clustering

The desired feature of an ideal clustering technique is that intra-cluster distances should be minimized and inter-cluster distances should be maximized. Following are the other important features that an ideal cluster analysis method should have:

- **Scalability:** Clustering algorithms should be capable of handling small as well as large datasets smoothly.
- **Ability to handle different types of attributes:** Clustering algorithms should be able to handle different kinds of data such as binary, categorical and interval-based (numerical) data.
- **Independent of data input order:** The clustering results should not be dependent on the ordering of input data.
- **Identification of clusters with different shapes:** The clustering algorithm should be capable of identifying clusters of any shape.
- **Ability to handle noisy data:** Usually, databases consist of noisy, erroneous or missing data, and algorithm must be able to handle these.
- **High performance:** To have a high performance algorithm, it is desirable that the algorithm should need to perform only one scan of the dataset. This capability would reduce the cost of input-output operations.
- **Interpretability:** The results of clustering algorithms should be interpretable, logical and usable.
- **Ability to stop and resume:** For a large dataset, it is desirable to stop and resume the task as it can take a huge amount of time to accomplish the full task and breaks may be necessary.
- **Minimal user guidance:** The clustering algorithm should not expect too much supervision from the analyst, because commonly the analyst has a limited knowledge of the dataset.

In clustering, distance metrics play a vital role in comprehending the similarity between the objects. In the next section, we will discuss different distance metrics that play an important role in the process of clustering of objects.

7.11 Distance Metrics

A distance metric is a function $d(x, y)$ that specifies the distance between elements of a set as a non-negative real number. Two elements are equal under a particular metric if the distance between them is zero. Distance functions present a method to measure the closeness of two elements. Here, elements can be matrices, vectors or arbitrary objects and do not necessarily need to be numbered.

In the following subsections, important distance metrics used in the measuring similarity among objects have been illustrated.

7.11.1 Euclidean distance

Euclidean distance is mainly used to calculate distances. The distance between two points in the plane with coordinates (x, y) and (a, b) according to the Euclidean distance formula is given by:

$$\text{Euclidean dist}((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

For example, the (Euclidean) distance between points $(-2, 2)$ and $(2, -1)$ is calculated as

$$\text{Euclidean dist}((-2, 2), (2, -1)) = \sqrt{(-2 - (2))^2 + (2 - (-1))^2}$$

$$\begin{aligned} &= \sqrt{(-4)^2 + (3)^2} \\ &= \sqrt{16 + 9} \\ &= \sqrt{25} \\ &= 5 \end{aligned}$$

Let's find the Euclidean distance among three persons to find the similarities and dissimilarities among them, on the basis of two variables. The data is given in Table 7.1.

Table 7.1 Data to calculate Euclidean distances among three persons

| | Variable 1 | Variable 2 |
|----------|------------|------------|
| Person 1 | 30 | 70 |
| Person 2 | 40 | 54 |
| Person 3 | 80 | 50 |

Using the formula of Euclidean distance, we can calculate the similarity distance among persons.

The calculation for the distance between person 1 and 2 is:

$$\begin{aligned}
 \text{Euclidean dist}((30, 70), (40, 54)) &= \sqrt{(30 - 40)^2 + (70 - 54)^2} \\
 &= \sqrt{(-10)^2 + (16)^2} \\
 &= \sqrt{100+256} \\
 &= \sqrt{356} \\
 &= 18.86
 \end{aligned}$$

The calculation for the distance between person 1 and 3 is:

$$\begin{aligned}
 \text{Euclidean dist}((30, 70), (80, 50)) &= \sqrt{(30 - 80)^2 + (70 - 50)^2} \\
 &= \sqrt{(-50)^2 + (20)^2} \\
 &= \sqrt{2500 + 400} \\
 &= \sqrt{2900} \\
 &= 53.85
 \end{aligned}$$

The calculation for the distance between person 2 and 3 is:

$$\begin{aligned}
 \text{Euclidean dist}((40, 54), (80, 50)) &= \sqrt{(40 - 80)^2 + (54 - 50)^2} \\
 &= \sqrt{(-40)^2 + (4)^2} \\
 &= \sqrt{1600+16} \\
 &= \sqrt{1616} \\
 &= 40.19
 \end{aligned}$$

This indicates that the persons 1 and 2 are most similar while person 1 and person 3 are most dissimilar. Thus, Euclidean distance is used to determine the dissimilarity between two objects by comparing them across a range of variables. These two objects might be profiles of two persons, e.g., a person and a target profile, or in fact, any two vectors taken across equivalent variables.

In the Euclidean distance, the attribute with the largest value may dominate the distance. Thus, it's important that attributes should be properly scaled before the application of the formula. It is also possible to use the Euclidean distance formula without the square root if one wishes to place greater weightage on large differences. Euclidean distance can also be represented as shown in Figure 7.2.

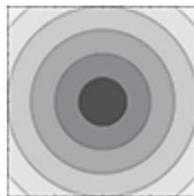


Figure 7.2 Representation of Euclidean distance [see colour plate]

[Credits: <https://numerics.mathdotnet.com/Distance.html>]

7.11.2 Manhattan distance

Manhattan distance is also called L1-distance. It is defined as the sum of the lengths of the projections of the line segment between the two points on the coordinate axes.

For example, the distance between two points in the plane with coordinates (x, y) and (a, b) according to the Manhattan distance formula, is given by:

$$\text{Manhattan dist}((x, y), (a, b)) = |x - a| + |y - b|$$

Let's do the calculations for finding the Manhattan distance among the same three persons, on the basis of their scores on two variables as shown in Table 7.1.

Using the formula of Manhattan distance, we can calculate the similarity distance among persons.

The calculation for the distance between person 1 and 2 is:

$$\begin{aligned}\text{Manhattan dist}((30, 70), (40, 54)) &= |30 - 40| + |70 - 54| \\ &= |-10| + |16| \\ &= 10+16 \\ &= 26\end{aligned}$$

The calculation for the distance between person 1 and 3 is:

$$\begin{aligned}\text{Manhattan dist}((30, 70), (80, 50)) &= |30 - 80| + |70 - 50| \\ &= 50+20 \\ &= 70\end{aligned}$$

The calculation for the distance between person 2 and 3 is:

$$\begin{aligned}\text{Manhattan dist}((40, 54), (80, 50)) &= |40 - 80| + |54 - 50| \\ &= 40+4 = 44\end{aligned}$$

This indicates that the persons 1 and 2 are most similar while person 1 and person 3 are most dissimilar and it produces the same conclusion as Euclidean distance.

Manhattan distance is also called city block distance because like Manhattan, it is the distance a car would drive in a city laid out in square blocks. In Manhattan city, one-way, oblique streets and real streets only exist at the edges of blocks. The Manhattan distance can be represented as shown in Figure 7.3.

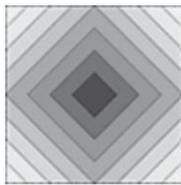


Figure 7.3 Representation of Manhattan distance [see colour plate]

[Credits: <https://numerics.mathdotnet.com/Distance.html>]

In Manhattan distance, the largest valued attribute can again dominate the distance, although not as much as in the Euclidean distance.

7.11.3 Chebyshev distance

It is also called as chessboard distance because, in a game of chess, the minimum number of moves required by a king to go from one square to another on a chessboard equals Chebyshev distance between the centers of the squares. Chebyshev distance is defined on a vector space, where the distance between two vectors is the maximum value of their differences along any coordinate dimension.

Formula of Chebyshev distance is given by:

$$\text{Chebyshev dist}((r_1, f_1), (r_2, f_2)) = \max(|r_2 - r_1|, |f_2 - f_1|)$$

Using the formula of Chebyshev distance, let us find the distance between object A and object B.

| Features | Coord1 | Coord2 | Coord3 | Coord4 |
|----------|--------|--------|--------|--------|
| Object A | 0 | 1 | 2 | 3 |
| Object B | 6 | 5 | 4 | -2 |

Object A coordinate = {0,1,2,3}

Object B coordinate = {6,5,4,-2}

According to Chebyshev distance formula

$$\begin{aligned}
 D &= \max(|r_2 - r_1|, |f_2 - f_1|) \\
 &= \max(|6-0|, |5-1|, |4-2|, |-2-3|) \\
 &= \max(6, 4, 2, 5) = 6
 \end{aligned}$$

Let's compute the calculations for finding the Chebyshev distance among three persons, on the basis of their scores on two variables as shown in Table 7.1.

The calculation for the distance between person 1 and 2 is:

$$\begin{aligned}
 \text{Chebyshev dist}((30, 70), (40, 54)) &= \max(|30 - 40|, |70 - 54|) \\
 &= \max(|-10|, |16|) = 16
 \end{aligned}$$

The calculation for the distance between person 1 and 3 is:

$$\begin{aligned}
 \text{Chebyshev dist}((30, 70), (80, 50)) &= \max(|30 - 80|, |70 - 50|) \\
 &= \max(50, 20) = 50
 \end{aligned}$$

The calculation for the distance between person 2 and 3 is:

$$= \max(|40 - 80|, |54 - 50|)$$

$$\text{Chebyshev dist}((40, 54), (80, 50)) = \max(40, 4) = 40$$

This indicates that the persons 1 and 2 are most similar while person 1 and person 3 are most dissimilar and it produces the same conclusion as Euclidean and Manhattan distance.

The Chebyshev distance can be represented as shown in Figure 7.4.

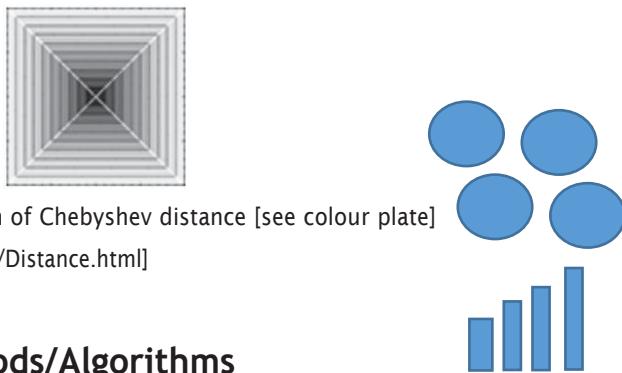


Figure 7.4 Representation of Chebyshev distance [see colour plate]

[Credits: <https://numerics.mathdotnet.com/Distance.html>]

7.12 Major Clustering Methods/Algorithms

Clustering methods/algorithms can be categorized into five categories which are given as follows:

- **Partitioning method:** It constructs random partitions and then iteratively refines them by some criterion.
- **Hierarchical method:** It creates a hierarchical decomposition of the set of data (or objects) using some criterion.
- **Density-based method:** It is based on connectivity and density functions.
- **Grid based method:** It is based on a multiple-level granularity structure.
- **Model based method:** A model is considered for each of the clusters and the idea is to identify the best fit for that model.

The categorization of clustering algorithms is shown in Figure 7.5.

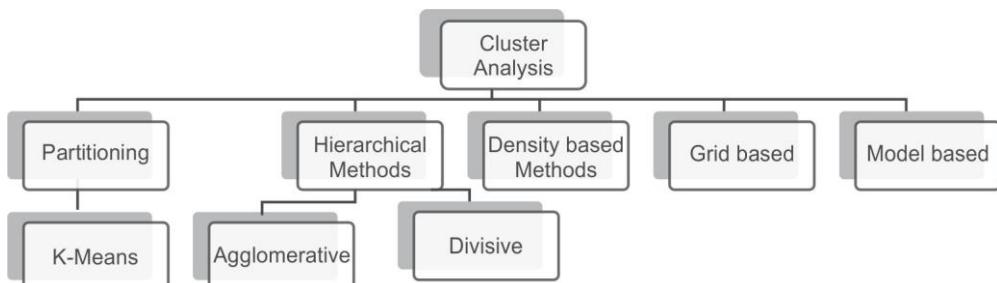
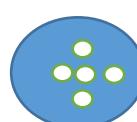
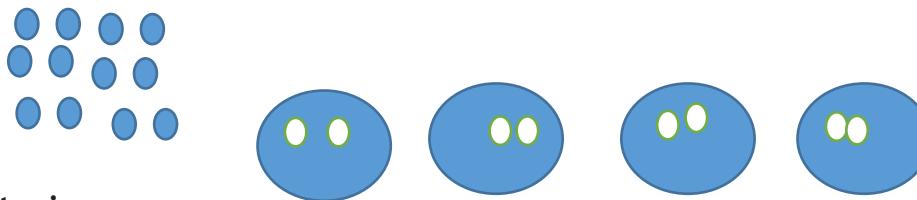


Figure 7.5 Major clustering methods/algorithms

Now, let us understand each method one by one.



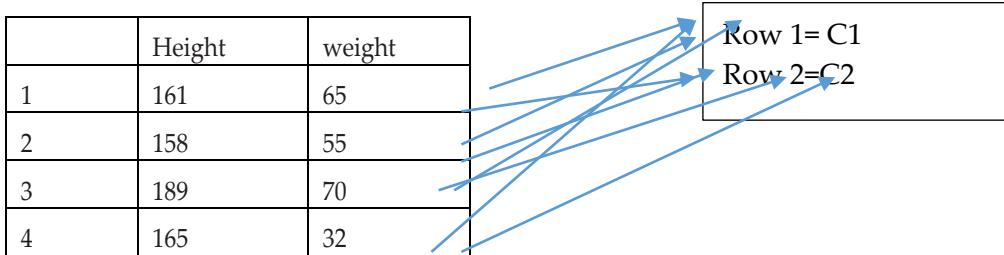


7.13 Partitioning Clustering

K=2 centroid=2 no of K =no of cluster

Take any 2 rows from the dataset 1 as group1 and 2nd group 2

Centroid of rows each rows will be a group



Clustering is the task of splitting a group of data or dataset into a small number of clusters. For example, the items in a grocery store are grouped into different categories (butter, milk, and cheese are clustered in dairy products). This is a qualitative kind of partitioning. A quantitative approach on the other hand, measures certain features of the products such as the percentage of milk and suchlike, i.e., products having a high percentage of milk would be clustered together.

In the partitioning method, we cluster objects based on attributes into a number of partitions. The k-means clustering is an important technique which falls under partitioning clustering.

7.6.1. k-means clustering

In the k-means clustering algorithm, n objects are clustered into k clusters or partitions on the basis of attributes, where $k < n$ and k is a positive integer number. In simple words, in k-means clustering algorithm, the objects are grouped into ' k ' number of clusters on the basis of attributes or features. The grouping of objects is done by minimizing the sum of squares of distances, i.e., a Euclidean distance between data and the corresponding cluster centroid.

Working of the k-means algorithm

The working of k-means clustering algorithm can be illustrated in five simple steps, as given below.

Step 1: Start with a selection of the value of k where $k = \text{number of clusters}$

In this step, the k centroids (or clusters) are initiated and then the first k training samples out of n samples of data are taken as single-element clusters. Each of the remaining ($n-k$) training samples are assigned to the cluster with the nearest centroid and the centroid of the gaining cluster is recomputed after each assignment.

Step 2: Creation of distance matrix between the centroids and each pattern

In this step, distance is computed from each sample of data to the centroid of each of the clusters. The heavy calculation involved is the major drawback of this step since there are k centroids and n samples, the algorithm will have to compute $n*k$ distances.

Step 3: Assign each sample in the cluster with the closest centroid (minimal distance)

Now, the samples are grouped on the basis of their distance from the centroid of each of the clusters. If currently, the sample is not in the cluster then switch it to the cluster with the closest centroid. When there is no movement of samples to another cluster anymore, the algorithm will end.

Step 4: Update the new centroids for each cluster

In this step, the locations of centroids are updated. Update the location for each centroid of the cluster that has gained or lost a sample by computing the mean of each attribute of all samples belonging to respective clusters.

Step 5: Repeat until no further change occurs

Return to Step 2 of the algorithm and repeat the updating process of each centroid location until a convergence condition is satisfied, which is until a pass through the training sample causes no new changes.

The flow chart of k-means algorithm is illustrated in Figure 7.6.

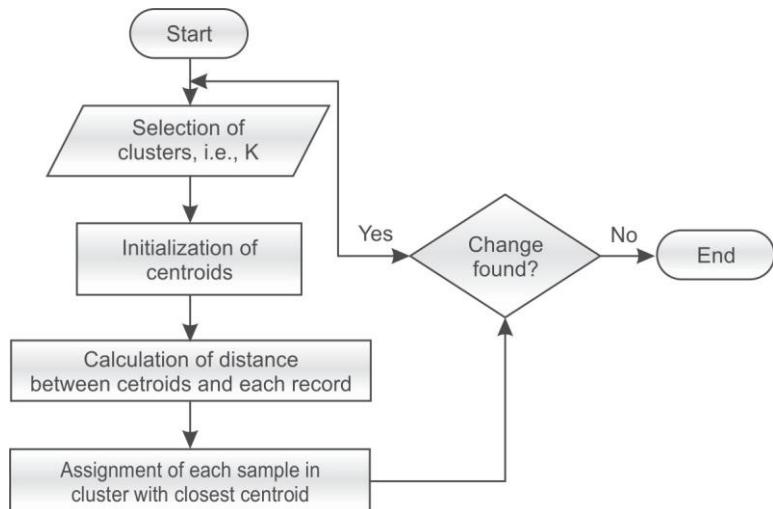


Figure 7.6 Flowchart for k-means algorithm

Example 7.1: Let's understand this procedure through an example; the database for which is given in Table 7.2. Here $k=2$ (there are just going to be two clusters).

Table 7.2 Database for the k-means algorithm example

| Individual | Variable 1 | Variable 2 |
|------------|------------|------------|
| 1 | 1.0 | 1.0 |
| 2 | 1.5 | 2.0 |
| 3 | 3.0 | 4.0 |
| 4 | 5.0 | 7.0 |
| 5 | 3.5 | 5.0 |
| 6 | 4.5 | 5.0 |
| 7 | 3.5 | 4.5 |

Step 1:

The first step will be to choose the value of k . In this problem, let's suppose, $k=2$, i.e., we are going to create two clusters C1 and C2 from this database. Let's suppose, record number 1st and 4th are randomly chosen as candidates for clusters (By just browsing the database given in Table 7.2, we can conclude that 1st and 4th are the most dissimilar records or farthest from each other. Otherwise, we can choose any two records as the starting point for initialization of this algorithm).

In this case, the two centroids are: centroid 1 (first record) = (1.0,1.0) and centroid 2 (fourth record) = (5.0,7.0) as shown in Figure 7.7.

| <i>Individual</i> | <i>Variable 1</i> | <i>Variable 2</i> |
|-------------------|-------------------|-------------------|
| 1 | 1.0 | 1.0 |
| 2 | 1.5 | 2.0 |
| 3 | 3.0 | 4.0 |
| 4 | 5.0 | 7.0 |
| 5 | 3.5 | 5.0 |
| 6 | 4.5 | 5.0 |
| 7 | 3.5 | 4.5 |

| | <i>Individual</i> | <i>Mean Vector</i> |
|---------|-------------------|--------------------|
| Group 1 | 1 | (1.0, 1.0) |
| Group 2 | 4 | (5.0, 7.0) |

Figure 7.7 Database after initialization

Step 2:

Now, the distance of each object is calculated from the other objects in this database on the basis of Euclidean distance as shown in Table 7.3.

The Euclidean distance of record 2 (1.5,2.0) from centroid 1 and centroid 2 has been given below.

Table 7.3 Database after first iteration

| <i>Individual</i> | <i>Centroid₁ (1,1) for C1</i> | <i>Centroid₂ (5,7) for C2</i> | <i>Assigned Cluster</i> |
|-------------------|--|---|-------------------------|
| 1 (1.0,1.0) | 0 | $\sqrt{ 1-5 ^2+ 1-7 ^2}$ = $\sqrt{16+36}$ = $\sqrt{52}$ = 7.21 | C1 |
| 2 (1.5,2.0) | $\sqrt{ 1.5-1 ^2+ 2.0-1 ^2}$ = $\sqrt{0.25+1}$ = $\sqrt{1.25}$ = 1.12 | $\sqrt{ 1.5-5 ^2+ 2.0-7 ^2}$ = $\sqrt{12.25+25}$ = $\sqrt{37.25}$ = 6.10 | C1 |

Contd.

| Individual | Centroid ₁ (1,1) for C1 | Centroid ₂ (5,7) for C2 | Assigned Cluster |
|--------------|--|--|------------------|
| 3 (3.0,4.0) | $\sqrt{ 3-1 ^2+ 4-1 ^2}$ = $\sqrt{4+9}$ = $\sqrt{13}$ = 3.61 | $\sqrt{ 3-5 ^2+ 4.0-7 ^2}$ = $\sqrt{4+9}$ = $\sqrt{13}$ = 3.61 | C1 |
| 4 (5.0, 7.0) | $\sqrt{ 5-1 ^2+ 7-1 ^2}$ = $\sqrt{16+36}$ = $\sqrt{52}$ = 7.21 | $\sqrt{ 5-5 ^2+ 7.0-7 ^2}$ = $\sqrt{0}$ = 0 | C2 |
| 5 (3.5, 5.0) | $\sqrt{ 3.5-1 ^2+ 5-1 ^2}$ = $\sqrt{6.25+16}$ = $\sqrt{22.25}$ = 4.72 | $\sqrt{ 3.5-5 ^2+ 5.0-7 ^2}$ = $\sqrt{2.25+4}$ = 2.5 | C2 |
| 6 (4.5, 5.0) | $\sqrt{ 4.5-1 ^2+ 5.0-1 ^2}$ = $\sqrt{12.25+16}$ = $\sqrt{28.25}$ = 5.31 | $\sqrt{ 4.5-5 ^2+ 5.0-7 ^2}$ = $\sqrt{0.25+4}$ = 2.06 | C2 |
| 7 (3.5, 4.5) | $\sqrt{ 3.5-1 ^2+ 4.5-1 ^2}$ = $\sqrt{6.25+12.25}$ = $\sqrt{18.5}$ = 4.30 | $\sqrt{ 3.5-5 ^2+ 4.5-7 ^2}$ = $\sqrt{2.25+6.25}$ = $\sqrt{8.5}$ = 2.92 | C2 |

Thus, we obtain two clusters containing: {1,2,3} and {4,5,6,7}.

The updated centroid for cluster 1 will be average of instances 1, 2 and 3 and for cluster 2 it will be average of instances 4, 5, 6 and 7 as shown below.

$$\text{centroid}_1 = \left(\frac{1}{3}(1.0 + 1.5 + 3.0), \frac{1}{3}(1.0 + 2.0 + 4.0) \right) = (1.83, 2.33)$$

$$\text{centroid}_2 = \left(\frac{1}{4}(5.0 + 3.5 + 4.5 + 3.5), \frac{1}{4}(7.0 + 5.0 + 5.0 + 4.5) \right) = (4.12, 5.38)$$

Now, the distance of each object to the modified centroid is calculated as shown in Table 7.4.

Table 7.4 Database after the second iteration

| Individual | Centroid ₁ (1.83,2.33) for C1 | Centroid ₂ (4.12,5.38) for C2 | Assigned Cluster |
|-------------|---|---|------------------|
| 1 (1.0,1.0) | $\sqrt{ 1.0-1.83 ^2+ 1.0-2.33 ^2} = 1.56$ | $\sqrt{ 1.0-4.12 ^2+ 1.0-5.83 ^2} = 4.83$ | C1 |
| 2 (1.5,2.0) | $\sqrt{ 1.5-1.83 ^2+ 2.0-2.33 ^2} = 0.46$ | $\sqrt{ 1.5-4.12 ^2+ 2.0-5.83 ^2} = 4.64$ | C1 |
| 3 (3.0,4.0) | $\sqrt{ 3.0-1.83 ^2+ 4.0-2.33 ^2} = 2.03$ | $\sqrt{ 3.0-4.12 ^2+ 4.0-5.83 ^2} = 2.14$ | C2 |
| 4 (5.0,7.0) | $\sqrt{ 5.0-1.83 ^2+ 7.0-2.33 ^2} = 5.64$ | $\sqrt{ 5.0-4.12 ^2+ 7.0-5.83 ^2} = 1.84$ | C2 |
| 5 (3.5,5.0) | $\sqrt{ 3.5-1.83 ^2+ 5.0-2.33 ^2} = 3.14$ | $\sqrt{ 3.5-4.12 ^2+ 5.0-5.83 ^2} = 1.03$ | C2 |
| 6 (4.5,5.0) | $\sqrt{ 4.5-1.83 ^2+ 5.0-2.33 ^2} = 3.77$ | $\sqrt{ 4.5-4.12 ^2+ 5.0-5.83 ^2} = 0.91$ | C2 |

The clusters obtained are: {1,2} and {3,4,5,6,7}

The updated centroid for cluster 1 will be average of instances 1 and 2 and for cluster 2 it will be average of instances 3, 4, 5, 6 and 7 as shown below.

$$\text{centroid}_1 = (1/2 (1.0+1.5), 1/2 (1.0+2.0)) = (1.25, 1.5)$$

$$\text{centroid}_2 = (1/5 (3.0+5.0+3.5+4.5+3.5), 1/5 (4.0+7.0+5.0+5.0+4.5)) = (3.9, 5.1)$$

Now, the distance of each object to the modified centroid is calculated as shown in Table 7.5.

Table 7.5 Database after the second iteration

| Individual | Centroid ₁ (1.25, 1.5) for C1 | Centroid ₂ (3.9, 5.1) for C2 | Assigned Cluster |
|--------------|--|---|------------------|
| 1 (1.0,1.0) | $\sqrt{ 1-1.25 ^2+ 1.0-1.5 ^2} = 0.56$ | $\sqrt{ 1-3.9 ^2+ 1-5.1 ^2} = 5.02$ | C1 |
| 2 (1.5,2.0) | $\sqrt{ 1.5-1.25 ^2+ 2.0-1.5 ^2} = 0.56$ | $\sqrt{ 1.5-3.9 ^2+ 2.0-5.1 ^2} = 3.92$ | C1 |
| 3 (3.0,4.0) | $\sqrt{ 3-1.25 ^2+ 4-1.5 ^2} = 3.05$ | $\sqrt{ 3-3.9 ^2+ 4.0-5.1 ^2} = 1.42$ | C2 |
| 4 (5.0, 7.0) | $\sqrt{ 5-1.25 ^2+ 7-1.5 ^2} = 6.66$ | $\sqrt{ 5-3.9 ^2+ 7.0-5.1 ^2} = 2.20$ | C2 |
| 5 (3.5, 5.0) | $\sqrt{ 3.5-1.25 ^2+ 5-1.5 ^2} = 4.16$ | $\sqrt{ 3.5-3.9 ^2+ 5.0-5.1 ^2} = 0.41$ | C2 |
| 6 (4.5, 5.0) | $\sqrt{ 4.5-1.25 ^2+ 5.0-1.5 ^2} = 4.78$ | $\sqrt{ 4.5-3.9 ^2+ 5.0-5.1 ^2} = 0.61$ | C2 |
| 7 (3.5, 4.5) | $\sqrt{ 3.5-1.25 ^2+ 4.5-1.5 ^2} = 3.75$ | $\sqrt{ 3.5-3.9 ^2+ 4.5-5.1 ^2} = 0.72$ | C2 |

The clusters obtained are: {1,2} and {3,4,5,6,7}

At this iteration, there is no further change in the clusters. Thus, the algorithm comes to a halt here and final result consists of 2 clusters, {1,2} and {3,4,5,6,7} respectively. These instances can be plotted as two clusters as shown in Figure 7.8.

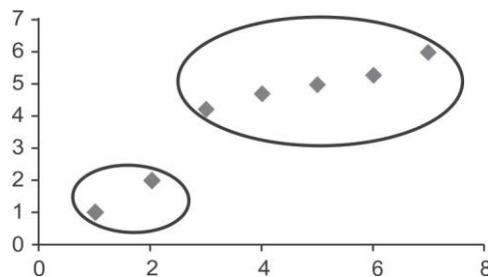


Figure 7.8 Plot of data for k=2 [see colour plate]

Step 3: Implementation of the k-means algorithm (using k=3)

For the same database, if k = 3; it means that there will be three clusters for this database. Let's randomly choose first three records as candidates for clusters. This time, readers are advised to perform the calculations and compare their results with final clusters data given in Table 7.6.

Table 7.6 Initial dataset for k = 3

| <i>Individual</i> | <i>Centroid1 = 1</i> | <i>Centroid2 = 2</i> | <i>Centroid3 = 3</i> | <i>Assigned cluster</i> |
|-------------------|----------------------|----------------------|----------------------|-------------------------|
| 1 | 0 | 1.11 | 3.61 | C1 |
| 2 | 1.12 | 0 | 2.5 | C2 |
| 3 | 3.61 | 2.5 | 0 | C3 |
| 4 | 7.21 | 6.10 | 3.61 | C3 |
| 5 | 4.72 | 3.61 | 1.12 | C3 |
| 6 | 5.31 | 4.24 | 1.80 | C3 |
| 7 | 4.30 | 3.20 | 0.71 | C3 |

Clustering with initial centroids (1,2,3)

Now, if we calculate the centroid distances as discussed above, the algorithm will stop at the state of the database as shown in Table 7.7, by putting instance 1 in cluster 1, instance 2 in cluster 2 and instances 3,4,5,6 and 7 in cluster 3 as shown below.

Table 7.7 Final assigned cluster for k = 3

| <i>Individual</i> | <i>Centroid1(0.56,0.55,3.05) for C1</i> | <i>Centroid2(4.21, 3.10, 0.61) for C2</i> | <i>Centroid3(6.26,5.17,0.56) for C3</i> | <i>Assigned cluster</i> |
|-------------------|---|---|---|-------------------------|
| 1 | 0 | 1.11 | 5.02 | C1 |
| 2 | 1.12 | 0 | 3.92 | C2 |
| 3 | 3.61 | 2.5 | 1.42 | C3 |
| 4 | 7.21 | 6.10 | 2.20 | C3 |
| 5 | 4.72 | 3.61 | 0.41 | C3 |
| 6 | 5.31 | 4.24 | 0.61 | C3 |
| 7 | 4.30 | 3.20 | 0.72 | C3 |

The plot of this data for $k = 3$ is shown below in Figure 7.9.

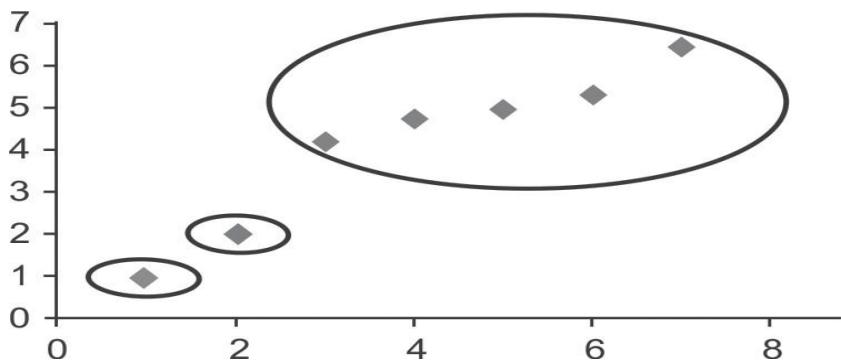


Figure 7.9 Plot of data for $k=3$ [see colour plate]

Example 7.2: Suppose the visitors visiting a website are to be grouped on the basis of their age given as follows:

15,15,16,19,19,20,20,21,22,28,35,40,41,42,43,44,60,61,65

To perform clustering for the given data, the first step will be to choose the value of k. In this case, let's suppose, k = 2, i.e., we are going to create just two clusters for this database and suppose we randomly choose the 3rd record i.e, 16 and the 9th, i.e., 22 as the initial centroids for the two clusters.

Thus: Centroid for C1 = 16 [16]

Centroid for C2 = 22 [22]

The distances of each record from centroid 1 and centroid 2 is given in Table 7.8. Here, the Manhattan distance has been used to calculate the distance metric which is the absolute distance between the points in this case. It has been used to make the calculations simple.

Table 7.8 Dataset after first iteration

| <i>Instances</i> | <i>Centroid 16 for C1</i> | <i>Centroid 22 for C2</i> | <i>Assigned Cluster</i> |
|------------------|---------------------------|---------------------------|-------------------------|
| 15 | 1 | 7 | C1 |
| 15 | 1 | 7 | C1 |
| 16 | 0 | 6 | C1 |
| 19 | 3 | 3 | C2 |
| 19 | 3 | 3 | C2 |
| 20 | 4 | 2 | C2 |
| 20 | 4 | 2 | C2 |
| 21 | 5 | 1 | C2 |
| 22 | 6 | 0 | C2 |
| 28 | 12 | 6 | C2 |
| 35 | 19 | 13 | C2 |
| 40 | 24 | 18 | C2 |
| 41 | 25 | 19 | C2 |
| 42 | 26 | 20 | C2 |
| 43 | 27 | 21 | C2 |
| 44 | 28 | 22 | C2 |
| 60 | 44 | 38 | C2 |
| 61 | 45 | 39 | C2 |
| 65 | 49 | 43 | C2 |

Thus, we obtain two clusters containing:

$$\{15, 15, 16\} \text{ and } \{19, 19, 20, 20, 21, 22, 28, 35, 40, 41, 42, 43, 44, 60, 61, 65\}.$$

The updated centroid for clusters will be as follows:

$$\text{Centroid1} = \text{avg}(15, 15, 16) = 15.33$$

$$\text{Centroid2} = \text{avg}(19, 19, 20, 20, 21, 22, 28, 35, 40, 41, 42, 43, 44, 60, 61, 65) = 36.25$$

Now, the distance of each object to modified centroid is calculated as shown in Table 7.9.

The least actual distances from the instance to the centroid have been highlighted.

Table 7.9 Dataset after second iteration

| Instances | Centroid 15.33 for C1 | Centroid 36.25 for C2 | Assigned Cluster |
|-----------|-----------------------|-----------------------|------------------|
| 15 | 0.33 | 21.25 | C1 |
| 15 | 0.33 | 21.25 | C1 |
| 16 | 0.67 | 20.25 | C1 |
| 19 | 3.67 | 17.25 | C1 |
| 19 | 3.67 | 17.25 | C1 |
| 20 | 4.67 | 16.25 | C1 |
| 20 | 4.67 | 16.25 | C1 |
| 21 | 5.67 | 15.25 | C1 |
| 22 | 6.67 | 14.25 | C1 |
| 28 | 12.67 | 8.25 | C2 |
| 35 | 19.67 | 1.25 | C2 |
| 40 | 24.67 | 3.75 | C2 |
| 41 | 25.67 | 4.75 | C2 |
| 42 | 26.67 | 5.75 | C2 |
| 43 | 27.67 | 6.75 | C2 |
| 44 | 28.67 | 7.75 | C2 |
| 60 | 44.67 | 23.75 | C2 |
| 61 | 45.67 | 24.75 | C2 |
| 65 | 49.67 | 28.75 | C2 |

Now, we have obtain two fresh clusters containing:

$$\{15, 15, 16, 19, 19, 20, 20, 21, 22\} \text{ and } \{28, 35, 40, 41, 42, 43, 44, 60, 61, 65\}.$$

The updated centroids for these fresh clusters will be as follows:

$$\text{Centroid 1} = \text{avg}(15,15,16,19,19,20,20,21,22) = 18.55$$

$$\text{Centroid 2} = \text{avg}(28,35,40,41,42,43,44,60,61,65) = 45.9$$

Next, the distance of each object to the modified centroid is calculated as shown in Table 7.10

Table 7.10 Dataset after third iteration

| Instances | Centroid 18.55 for C1 | Centroid 45.9 for C2 | Assigned Cluster |
|-----------|-----------------------|----------------------|------------------|
| 15 | 3.55 | 30.9 | C1 |
| 15 | 3.55 | 30.9 | C1 |
| 16 | 2.55 | 29.9 | C1 |
| 19 | 0.45 | 26.9 | C1 |
| 19 | 0.45 | 26.9 | C1 |
| 20 | 1.45 | 25.9 | C1 |
| 20 | 1.45 | 25.9 | C1 |
| 21 | 2.45 | 24.9 | C1 |
| 22 | 3.45 | 23.9 | C1 |
| 28 | 9.45 | 17.9 | C1 |
| 35 | 16.45 | 10.9 | C2 |
| 40 | 21.45 | 5.9 | C2 |
| 41 | 22.45 | 4.9 | C2 |
| 42 | 23.45 | 3.9 | C2 |
| 43 | 24.45 | 2.9 | C2 |
| 44 | 25.45 | 1.9 | C2 |
| 60 | 41.45 | 14.1 | C2 |
| 61 | 42.45 | 15.1 | C2 |
| 65 | 46.45 | 19.1 | C2 |

Here, we obtain two clusters with still fresher changes containing:

{15,15,16,19,19,20,20,21,22,28} and {35,40,41,42,43,44,60,61,65}.

The updated centroid for clusters will be as follows:

$$\text{Centroid 1} = \text{avg}(15,15,16,19,19,20,20,21,22,28) = 19.5$$

$$\text{Centroid 2} = \text{avg}(35,40,41,42,43,44,60,61,65) = 47.89$$

Now, the distance of each object to the most recently modified centroids is calculated as shown in Table 7.11.

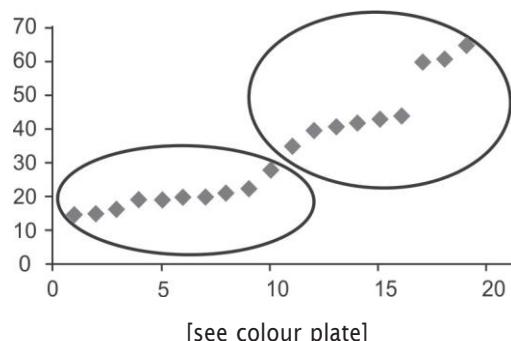
Table 7.11 Dataset after fourth iteration

| Instances | Centroid 19.50 for C1 | Centroid 47.89 for C2 | Assigned Cluster |
|-----------|-----------------------|-----------------------|------------------|
| 15 | 4.5 | 32.89 | C1 |
| 15 | 4.5 | 32.89 | C1 |
| 16 | 3.5 | 31.89 | C1 |
| 19 | 0.5 | 28.89 | C1 |
| 19 | 0.5 | 28.89 | C1 |
| 20 | 0.5 | 27.89 | C1 |
| 20 | 0.5 | 27.89 | C1 |
| 21 | 1.5 | 26.89 | C1 |
| 22 | 2.5 | 25.89 | C1 |
| 28 | 8.5 | 19.89 | C1 |
| 35 | 15.5 | 12.89 | C2 |
| 40 | 20.5 | 7.89 | C2 |
| 41 | 21.5 | 6.89 | C2 |
| 42 | 22.5 | 5.89 | C2 |
| 43 | 23.5 | 4.89 | C2 |
| 44 | 24.5 | 3.89 | C2 |
| 60 | 40.5 | 12.11 | C2 |
| 61 | 41.5 | 13.11 | C2 |
| 65 | 45.5 | 17.11 | C2 |

At this iteration, no object has shifted from one cluster to another, i.e., no change has been found between iterations 3 and 4. Thus, the algorithm will stop at fourth iteration and the final clusters will be as follows.

Cluster 1 = [15,15,16,19,19,20,20,21,22,28]

Cluster 2 = [35,40,41,42,43,44,60,61,65]



Hence, two groups, i.e., 15-28 and 35-65 have been found by using clustering. The output clusters can be affected by the initial choice of centroids. The algorithm is run several times with different starting conditions in order to get a fair view of clusters.

Example 7.3: Clustering of Students record with the k-means algorithm

Let us consider another database containing the result of students' examination in a given course. Here, we wish to cluster the students according to their performance in the course. The students' marks (obtained out of 100) across Quiz1, MSE (Mid Semester Exam), Quiz2, and ESE (End Semester Exam) are given in Table 7.12.

Table 7.12 Record of students' performance

| Sr. No | Quiz1 | MSE | Quiz2 | ESE |
|--------|-------|-----|-------|-----|
| S1 | 8 | 20 | 6 | 45 |
| S2 | 6 | 18 | 7 | 42 |
| S3 | 5 | 15 | 6 | 35 |
| S4 | 4 | 13 | 5 | 25 |
| S5 | 9 | 21 | 8 | 48 |
| S6 | 7 | 20 | 9 | 44 |
| S7 | 9 | 17 | 8 | 49 |
| S8 | 8 | 19 | 7 | 39 |
| S9 | 3 | 14 | 4 | 22 |
| S10 | 6 | 15 | 7 | 32 |

Steps 1 and 2:

Let's consider the first three students as the three seeds as shown in Table 7.13.

Table 7.13 Seed records

| Sr. no | Quiz1 | MSE | Quiz2 | ESE |
|--------|-------|-----|-------|-----|
| S1 | 8 | 20 | 6 | 45 |
| S2 | 6 | 18 | 7 | 42 |
| S3 | 5 | 15 | 6 | 35 |

Steps 3 and 4:

In this step, calculate the distance using the four attributes as well as the sum of absolute differences for simplicity, i.e., Manhattan distance. Table 7.14 shows the distance values for all the objects, wherein columns 7, 8 and 9 represent the three distances from the three seed records respectively.

On the basis of distances computed, each student is allocated to the nearest cluster, i.e., C1, C2 or C3. The result of the first iteration is given in Table 7.14.

Table 7.14 First iteration-allocation of each object to its nearest cluster

| Seed Records | | Quiz1 | MSE | Quiz2 | ESE | Distances from Cluster | | | Allocation to the nearest Cluster | |
|--------------|----|-------|-----|-------|-----|--------------------------------------|--------------------------------------|--------------------------------------|-----------------------------------|--|
| | C1 | 8 | 20 | 6 | 45 | | | | | |
| | C2 | 6 | 18 | 7 | 42 | | | | | |
| | C3 | 5 | 15 | 6 | 35 | C1 | C2 | C3 | | |
| S1 | 1 | 8 | 20 | 6 | 45 | $(8-8 + 20-20 + 6-6 + 45-45) = 0$ | $(8-6 + 20-18 + 6-7 + 45-42) = 8$ | $(8-5 + 20-15 + 6-6 + 45-35) = 18$ | C1 | |
| S2 | 2 | 6 | 18 | 7 | 42 | $(6-8 + 18-20 + 7-6 + 42-45) = 8$ | $(6-6 + 18-18 + 7-7 + 42-42) = 0$ | $(6-5 + 18-15 + 7-6 + 42-35) = 12$ | C2 | |
| S3 | 3 | 5 | 15 | 6 | 35 | $(5-8 + 15-20 + 6-6 + 35-45) = 18$ | $(5-6 + 15-18 + 6-7 + 35-42) = 12$ | $(5-5 + 15-15 + 6-6 + 35-35) = 0$ | C3 | |
| S4 | 4 | 4 | 13 | 5 | 25 | 32 | 26 | 14 | C3 | |
| S5 | 5 | 9 | 21 | 8 | 48 | 7 | 13 | 25 | C1 | |
| S6 | 6 | 7 | 20 | 9 | 44 | 5 | 7 | 19 | C1 | |
| S7 | 7 | 9 | 17 | 8 | 49 | 10 | 12 | 22 | C1 | |
| S8 | 8 | 8 | 19 | 7 | 39 | 8 | 6 | 12 | C2 | |
| S9 | 9 | 3 | 14 | 4 | 22 | 36 | 30 | 18 | C3 | |
| S10 | 10 | 6 | 15 | 7 | 32 | 21 | 13 | 5 | C3 | |

Instances in Cluster 1, i.e., C1 are 1, 5, 6 and 7

Instances in Cluster 2, i.e., C2 are 2 and 8

Instances in Cluster 3, i.e., C3 are 3, 4, 9 and 10

The first iteration results in four, two and four students in the first, second and third cluster, respectively.

Step 5:

New centroids for each attribute/column are updated after the first iteration as shown in Table 7.15.

Table 7.15 Updated centroids after first iteration

| | Quiz1 | MSE | Quiz2 | ESE |
|--------------------------|----------------------|---------------------------|----------------------|--------------------------|
| C1 (record 1,5,6,7) | 8.25 Avg(8,9,7,9) | 19.5 Avg(20,21,20,17) | 7.75 Avg(6,8,9,7) | 46.5 Avg(45,48,44,39) |
| C2 (record 2,8) | 7 Avg(6,8) | 18.5 Avg(18,19) | 7 Avg(7,7) | 40.5 Avg(42,39) |
| C3 (record 3,4,9, 10) | 4.5 Avg(5,4,3,6) | 14.25 Avg(15,13,14,15) | 5.5 Avg(6,5,4,7) | 28.5 Avg(35,25,22,32) |

It is interesting to note that the mean marks for cluster C3 are significantly lower than for clusters C1 and C2. So, the distances of each object to each of the means using new cluster are again re-computed and allocate each object to the nearest cluster. The results of the second iteration are shown in Table 7.16.

Table 7.16 Second iteration-allocation of each object to its nearest cluster

| Seed Records | | Quiz1 | MSE | Quiz2 | ESE | Distances from Cluster | | | Allocation to the nearest Cluster | | |
|-----------------|----|-------|------|-------|------|--|--|---|--|--|--|
| | | C1 | 8.25 | 19.5 | 7.75 | 46.5 | | | | | |
| | | C2 | 7 | 18.5 | 7 | 40.5 | | | | | |
| | | C3 | 4.5 | 14.25 | 5.5 | 28.5 | C1 | C2 | C3 | | |
| Student 1 | 1 | 8 | 20 | 6 | 45 | $\begin{aligned} &(8-8.25 + 20-19.5 + 6-7.75 + 45-46.5) \\ &= 4 \end{aligned}$ | $\begin{aligned} &(8-7 + 20-18.5 + 6-7 + 45-40.5) \\ &= 8 \end{aligned}$ | $\begin{aligned} &(8-4.5 + 20-14.25 + 6-5.5 + 45-28.5) \\ &= 26.25 \end{aligned}$ | C1 | | |
| Student 2 | 2 | 6 | 18 | 7 | 42 | $\begin{aligned} &(6-8.25 + 18-19.5 + 7-7.75 + 42-46.5) \\ &= 9 \end{aligned}$ | $\begin{aligned} &(6-7 + 18-18.5 + 7-7 + 42-40.5) \\ &= 3 \end{aligned}$ | $\begin{aligned} &(6-4.5 + 18-14.25 + 7-5.5 + 42-28.5) \\ &= 20.25 \end{aligned}$ | C2 | | |
| Student 3 | 3 | 5 | 15 | 6 | 35 | 21 | 12 | 8.25 | C3 | | |
| Student 4 | 4 | 4 | 13 | 5 | 25 | 35 | 26 | 5.75 | C3 | | |
| Student 5 | 5 | 9 | 21 | 8 | 48 | 4 | 13 | 33.25 | C1 | | |
| Student 6 | 6 | 7 | 20 | 9 | 44 | 5.5 | 7 | 27.25 | C1 | | |
| Student 7 | 7 | 9 | 17 | 8 | 49 | 6 | 13 | 30.25 | C1 | | |
| Student 8 | 8 | 8 | 19 | 7 | 39 | 9 | 3 | 20.25 | C2 | | |
| Student 9 | 9 | 3 | 14 | 4 | 22 | 39 | 30 | 9.75 | C3 | | |
| Student 10 | 10 | 6 | 15 | 7 | 32 | 22 | 13 | 7.25 | C3 | | |

Again, there are four, two and four students in clusters C1, C2 and C3, respectively. A more careful look shows that there is no change in clusters at all. Therefore, the method has converged quite quickly for this very simple dataset. For more clarity, the final assigned cluster has been reproduced in Table 7.17.

Table 7.17 Final allocation

| Seed Records | | Quiz1 | MSE | Quizz2 | ESE | Allocation to Cluster |
|-----------------|----|-------|-------|--------|------|--------------------------|
| | C1 | 8.25 | 19.5 | 7.75 | 46.5 | |
| | C2 | 7 | 18.5 | 7 | 40.5 | |
| | C3 | 4.5 | 14.25 | 5.5 | 28.5 | |
| S1 | 1 | 8 | 20 | 6 | 45 | C1 |
| S2 | 2 | 6 | 18 | 7 | 42 | C2 |
| S3 | 3 | 5 | 15 | 6 | 35 | C3 |
| S4 | 4 | 4 | 13 | 5 | 25 | C3 |
| S5 | 5 | 9 | 21 | 8 | 48 | C1 |
| S6 | 6 | 7 | 20 | 9 | 44 | C1 |
| S7 | 7 | 9 | 17 | 8 | 49 | C1 |
| S8 | 8 | 8 | 19 | 7 | 39 | C2 |
| S9 | 9 | 3 | 14 | 4 | 22 | C3 |
| S10 | 10 | 6 | 15 | 7 | 32 | C3 |

The membership of clusters is as follows:

Cluster 1, i.e., C1 - 1,5,6,7

Cluster 2, i.e., C2 - 2, 8

Cluster 3, i.e., C3 - 3, 4, 9, 10

It should be noted that the above method was k-median rather than k-means because we wanted to use a simple distance measure that a reader could check without using a computer or a calculator.

Interestingly, when k-means is used, the result is exactly the same and the method again converges rather quickly.

Another point worth noting is about the intra (within) cluster variance and inter (between) cluster variance. The average Euclidean distance of objects in each cluster to the cluster centroids is presented in Table 7.18. The average distance of objects within clusters C1, C2, and C3 from their centroids is 2.89, 1.87 and 5.51, respectively. Although these numbers do not indicate which result is better, they help however, in analyzing that the clustering method has performed well in maximizing inter (between) cluster variance and minimizing intra (within) cluster variance. Here, one can notice that the Euclidean distance between C1 and C2 is 6.89, while it is 44.89 in case of C1 and C3. The intra cluster distance in the case of C1, is the minimum i.e., 2.89 between the instances of C1. Similarly, the intra cluster distance is minimum in case of C2 and C3.

Consider the student performance of your class having following attributes.Rno, Name, Quiz1, MSE, Quiz2, Lab, ESE, Total

Insert 30 dummy records in this excel file.

- Perform clustering on this dataset.
- Identify optimum number of clusters.
- After identification of optimum number of clusters, prepare clustering on this number.

We can get different results with different seed records.

Table 7.18 Within (intra) cluster and between (inter) clusters distance

| Cluster | C1 | C2 | C3 |
|---------|-------|-------|-------|
| C1 | 2.89 | 6.89 | 19.46 |
| C2 | 6.52 | 1.87 | 13.19 |
| C3 | 20.02 | 14.17 | 5.51 |

The detail of calculations for the identification of within (intra) cluster and between (inter) cluster variance is shown in Table 7.19.

Table 7.19 Calculations for within-cluster and between-cluster variance using Euclidean distance

| Calculation | Quiz1 | MSE | Quiz2 | ESE | Description |
|--------------------------|---|--|-------------------|-------------------|----------------|
| $x_i - y_i$ for S1 of C1 | -0.25 [8-8.25] | 0.5 [20-19.5] | -1.75 [6-7.75] | -1.5 [45-46.5] | For Cluster C1 |
| $x_i - y_i$ for S5 of C1 | 0.75 | 1.5 | 0.25 | 1.5 | |
| $x_i - y_i$ for S6 of C1 | -1.25 | 0.5 | 1.25 | -2.5 | |
| $x_i - y_i$ for S7 of C1 | 0.75 | -2.5 | 0.25 | 2.5 | |
| sumsqr($x_i - y_i$) | 5.625 sumsqr(-0.25,0.5,-1.75,-1.5) | for S1 | | | |
| sumsqr($x_i - y_i$) | 5.125 sumsqr(0.75,1.5,0.25, 1.5) | for S5 | | | |
| sumsqr($x_i - y_i$) | 9.625 | for S6 | | | |
| sumsqr($x_i - y_i$) | 13.125 | for S7 | | | |
| avg(1,56,7) | 8.375 avg (5.625, 5.125, 9.625, 13.125) | for 1,5,6,7 of c1 from Cluster C1 | | | |
| sqrt (avg(1,56,7)) | 2.893959 | for 1,5,6,7 of c1, Thus within-cluster variance of C1 is 2.89 | | | |
| $x_i - y_i$ for S1 of C1 | 1 | 1.5 | -1 | 4.5 | For Cluster C2 |
| $x_i - y_i$ for S5 of C1 | 2 | 2.5 | 1 | 7.5 | |
| $x_i - y_i$ for S6 of C1 | 0 | 1.5 | 2 | 3.5 | |
| $x_i - y_i$ for S7 of C1 | 2 | -1.5 | 1 | 8.5 | |

Contd.

| <i>Calculation</i> | <i>Quiz 1</i> | <i>MSE</i> | <i>Quiz 2</i> | <i>ESE</i> | <i>Description</i> |
|-------------------------------------|-----------------|------------|--|------------|--------------------|
| $\text{sqr}(x_i - y_i)$ | 24.5 | | for S1 | | |
| $\text{sqr}(x_i - y_i)$ | 67.5 | | for S5 | | |
| $\text{sqr}(x_i - y_i)$ | 18.5 | | for S6 | | |
| $\text{sqr}(x_i - y_i)$ | 79.5 | | for S7 | | |
| $\text{avg}(1, 56, 7)$ | 47.5 | | for 1, 5, 6, 7 of C1 from C2 | | |
| $\text{sqrt}(\text{avg}(1, 56, 7))$ | 6.892024 | | for 1, 5, 6, 7 of C1 from C2 | | |
| | | | Thus variance between C1 and C2 is 6.89 | | |
| $x_i - y_i$ for S1 of C1 | 3.5 | 5.75 | 0.5 | 16.5 | For Cluster C3 |
| $x_i - y_i$ for S5 of C1 | 4.5 | 6.75 | 2.5 | 19.5 | |
| $x_i - y_i$ for S6 of C1 | 2.5 | 5.75 | 3.5 | 15.5 | |
| $x_i - y_i$ for S7 of C1 | 4.5 | 2.75 | 2.5 | 20.5 | |
| $\text{sqr}(x_i - y_i)$ | 317.8125 | | for S1 from C3 | | |
| $\text{sqr}(x_i - y_i)$ | 452.3125 | | for S5 from C3 | | |
| $\text{sqr}(x_i - y_i)$ | 291.8125 | | for S6 from C3 | | |
| $\text{sqr}(x_i - y_i)$ | 454.3125 | | for S7 from C3 | | |
| $\text{avg}(1, 56, 7)$ | 379.0625 | | for 1, 5, 6, 7 of C1 from C3 | | |
| $\text{sqrt}(\text{avg}(1, 56, 7))$ | 19.4695 | | for 1, 5, 6, 7 of C1 from C3 | | |
| | | | Thus variance between C1 and C3 is 19.4695 | | |
| $x_i - y_i$ for S2 of C2 | -2.25 | -1.5 | -0.75 | -4.5 | For Cluster C1 |
| $x_i - y_i$ for S8 of C2 | -0.25 | -0.5 | -0.75 | -7.5 | |
| $\text{sqr}(x_i - y_i)$ | 28.125 | | for S2 of C2 from C1 | | |
| $\text{sqr}(x_i - y_i)$ | 57.125 | | for S8 of C2 from C1 | | |
| $\text{avg}(2, 8)$ | 42.625 | | for 2, 8 of C2 from C1 | | |
| $\text{sqrt}(\text{avg}(2, 8))$ | 6.528782 | | for 2, 8 of C2 from C1 | | |
| | | | Thus variance between C2 and C1 is 6.52 | | |
| $x_i - y_i$ for S2 of C2 | -1 | -0.5 | 0 | 1.5 | For Cluster C2 |
| $x_i - y_i$ for S8 of C2 | 1 | 0.5 | 0 | -1.5 | |
| $\text{sqr}(x_i - y_i)$ | 3.5 | | for S2 of C2 from C2 | | |
| $\text{sqr}(x_i - y_i)$ | 3.5 | | for S8 of C2 from C2 | | |
| $\text{avg}(2, 8)$ | 3.5 | | for 2, 8 of C2 from C2 | | |

Contd.

| <i>Calculation</i> | <i>Quiz 1</i> | <i>MSE</i> | <i>Quiz 2</i> | <i>ESE</i> | <i>Description</i> |
|-------------------------------------|-----------------|------------|---------------|------------|---|
| $\text{sqrt}(\text{avg}(2,8))$ | 1.870829 | | | | for 2,8 of C2 from C2 Thus within cluster variance of C2, i.e. from C2 to C2 is 1.87 |
| $x_i \cdot y_i$ for S2 of C2 | 1.5 | 3.75 | 1.5 | 13.5 | For Cluster C3 |
| $x_i \cdot y_i$ for S8 of C2 | 3.5 | 4.75 | 1.5 | 10.5 | |
| $\text{sqr}(x_i \cdot y_i)$ | 200.8125 | | | | for S2 of C2 from C3 |
| $\text{sqr}(x_i \cdot y_i)$ | 147.3125 | | | | for S8 of C2 from C3 |
| $\text{avg}(2,8)$ | 174.0625 | | | | for 2,8 of C2 from C3 |
| $\text{sqrt}(\text{avg}(2,8))$ | 13.19327 | | | | for 2,8 of C2 from C3 Thus variance between C2 and C3 is 13.19 |
| $x_i \cdot y_i$ for S3 of C3 | -3.25 | -4.5 | -1.75 | -11.5 | For Cluster C1 |
| $x_i \cdot y_i$ for S4 of C3 | -4.25 | -6.5 | -2.75 | -21.5 | |
| $x_i \cdot y_i$ for S9 of C3 | -5.25 | -5.5 | -3.75 | -24.5 | |
| $x_i \cdot y_i$ for S10 of C3 | -2.25 | -4.5 | -0.75 | -14.5 | |
| $\text{sqr}(x_i \cdot y_i)$ | 166.125 | | | | for S3 of C3 from C1 |
| $\text{sqr}(x_i \cdot y_i)$ | 530.125 | | | | for S4 of C3 from C1 |
| $\text{sqr}(x_i \cdot y_i)$ | 672.125 | | | | for S9 of C3 from C1 |
| $\text{sqr}(x_i \cdot y_i)$ | 236.125 | | | | for S10 of C3 from C1 |
| $\text{avg}(3,4,9,10)$ | 401.125 | | | | for 3,4,9,10 of C3 from C1 |
| $\text{sqrt}(\text{avg}(3,4,9,10))$ | 20.02811 | | | | for 3,4,9,10 of C3 from C1 Thus variance between C3 and C1 is 20.02 |
| $x_i \cdot y_i$ for S3 of C3 | -2 | -3.5 | -1 | -5.5 | For Cluster C2 |
| $x_i \cdot y_i$ for S4 of C3 | -3 | -5.5 | -2 | -15.5 | |
| $x_i \cdot y_i$ for S9 of C3 | -4 | -4.5 | -3 | -18.5 | |
| $x_i \cdot y_i$ for S10 of C3 | -1 | -3.5 | 0 | -8.5 | |
| $\text{sqr}(x_i \cdot y_i)$ | 47.5 | | | | for S3 of C3 from C2 |
| $\text{sqr}(x_i \cdot y_i)$ | 283.5 | | | | for S4 of C3 from C2 |
| $\text{sqr}(x_i \cdot y_i)$ | 387.5 | | | | for S9 of C3 from C2 |
| $\text{sqr}(x_i \cdot y_i)$ | 85.5 | | | | for S10 of C3 from C2 |
| $\text{avg}(3,4,9,10)$ | 201 | | | | for 3,4,9,10 of C3 from C2 |

Contd.

| <i>Calculation</i> | <i>Quiz 1</i> | <i>MSE</i> | <i>Quiz 2</i> | <i>ESE</i> | <i>Description</i> |
|-------------------------------------|-----------------|------------|---------------|------------|--|
| $\text{sqrt}(\text{avg}(3,4,9,10))$ | 14.17745 | | | | for 3,4,9,10 of C3 from C2 Thus variance between C3 and C2 is 14.17 |
| $x_i - y_i$ for S3 of C3 | 0.5 | 0.75 | 0.5 | 6.5 | For Cluster C3 |
| $x_i - y_i$ for S4 of C3 | -0.5 | -1.25 | -0.5 | -3.5 | |
| $x_i - y_i$ for S9 of C3 | -1.5 | -0.25 | -1.5 | -6.5 | |
| $x_i - y_i$ for S10 of C3 | 1.5 | 0.75 | 1.5 | 3.5 | |
| $\text{sqr}(x_i - y_i)$ | 43.3125 | | | | for S3 of C3 from C3 |
| $\text{sqr}(x_i - y_i)$ | 14.3125 | | | | for S4 of C3 from C3 |
| $\text{sqr}(x_i - y_i)$ | 46.8125 | | | | for S9 of C3 from C3 |
| $\text{sqr}(x_i - y_i)$ | 17.3125 | | | | for S10 of C3 from C3 |
| $\text{avg}(3,4,9,10)$ | 30.4375 | | | | for 3,4,9,10 of C3 from C3 |
| $\text{sqrt}(\text{avg}(3,4,9,10))$ | 5.517019 | | | | for 3,4,9,10 of C3 from C3 Thus within cluster variance of C3, i.e. from C3 to C3 is 5.51 |

7.6.2 Starting values for the k-means algorithm

Normally we have to specify the number of starting seeds and clusters at the start of the k-means algorithm. We can use an iterative approach to overcome this problem. For example, first select three starting seeds randomly and choose three clusters. Once the final clusters have been found, the process may be repeated several times with a different set of seeds. We should select the seed records that are at maximum distance from each other or are as far as possible. If two clusters are identified to be close together during the iterative process, it is appropriate to merge them. Also, a large cluster may be partitioned into two smaller clusters if the intra-cluster variance is above some threshold value.

7.6.3 Issues with the k-means algorithm

There are a number of issues with the k-Means algorithm that should be understood. Some of these are:

- The results of the k-means algorithm intensively depends on the initial guesses of the seed records.
- The k-means algorithm is sensitive to outliers. Thus, it may give poor results if an outlier is selected as a starting seed record.
- The k-means algorithm works on the basis of Euclidean distance and the mean(s) of the attribute values of intra-cluster objects. Thus, it is only suitable for continuous data as it is restricted to data for which there is the notion of a center (centroid).
- The k-means algorithm does not take into account the size of the clusters. Clusters may be either large or small.

- It does not deal with overlapping clusters.
- It does not work well with clusters of different sizes and density.

7.6.4 Scaling and weighting

Sometimes during clustering, the value of one attribute may be small as compared to other attributes. In such cases, that attribute will not have much influence on determining which cluster the object belongs to, as the values of this attribute are smaller than other attributes and hence, the variation within the attribute will also be small.

Consider the dataset given in Table 7.20 for the chemical composition of wine samples. Note that the values for different attributes cover a range of six orders of magnitude. It is also important to notice that for wine density, values vary by only 5% of the average value. It turns out that clustering algorithms struggle with numeric attributes that exhibit such ranges of values.



WineQuality.arff file available in
WineQuality.zip at
www.technologyforge.net/Datasets

| fined acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | |
|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|
| 4.6 | 0.520 | 0.15 | 2.1 | 0.054 | 8 | 65 | 0.99340 | 3.9 |
| 4.7 | 0.600 | 0.17 | 2.3 | 0.058 | 17 | 106 | 0.99320 | 3.9 |
| 4.9 | 0.420 | 0.00 | 2.1 | 0.048 | 16 | 42 | 0.99154 | 3.71 |
| 5.0 | 0.380 | 0.01 | 1.6 | 0.048 | 26 | 60 | | 0.74 |
| 5.0 | 0.400 | 0.50 | 4.3 | 0.046 | 29 | 80 | | 14.0 |
| 5.0 | 0.420 | 0.24 | 2.0 | 0.060 | 19 | 50 | | R |
| 5.0 | 0.740 | 0.00 | 1.2 | 0.041 | 16 | 46 | | 7 |
| 5.0 | 1.020 | 0.04 | 1.4 | 0.045 | 41 | 85 | 0.99380 | 3.75 |
| 5.0 | 1.040 | 0.24 | 1.6 | 0.050 | 32 | 96 | 0.99340 | 3.74 |
| 5.1 | 0.420 | 0.00 | 1.8 | 0.044 | 18 | 88 | 0.99157 | 3.68 |
| 5.1 | 0.470 | 0.02 | 1.3 | 0.034 | 18 | 44 | 0.99210 | 3.90 |
| 5.1 | 0.510 | 0.18 | 2.1 | 0.042 | 16 | 101 | 0.99240 | 3.46 |

10⁶ range of magnitudes

Max – Min = 5% of average

Table 7.20 Chemical composition of wine samples

[Source: Witten et. al.]

All attributes should be transformed to a similar scale for clustering to be effective unless we want to give more weight to some attributes that are comparatively large in scale. Commonly, we use two techniques to convert the attributes. These are Normalization and Standardization.

Normalization

In case of normalization, all the attributes are converted to a normalized score or to a range (0, 1). The problem of normalization is an outlier. If there is an outlier, it will tend to crunch all of the other values down toward the value of zero. In order to understand this case, let's suppose the range of students' marks is 35 to 45 out of 100. Then 35 will be considered as 0 and 45 as 1, and students will be distributed between 0 to 1 depending upon their marks. But if there is one student having marks 90, then it will act as an outlier and in this case, 35 will be considered as 0 and 90 as 1. Now, it will crunch most of the values down toward the value of zero. In this scenario, the solution is standardization.

Standardization

In case of standardization, the values are all spread out so that we have a standard deviation of 1.

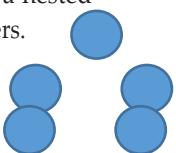
Generally, there is no rule for when to use normalization versus standardization. However, if your data does have outliers, use standardization otherwise use normalization. Using standardization tends to make the remaining values for all of the other attributes fall into similar ranges since all attributes will have the same standard deviation of 1.

In next section, another important clustering technique, i.e., Hierarchical Clustering has been discussed.

7.14 Hierarchical Clustering Algorithms (HCA)

Hierarchical clustering is a type of cluster analysis which seeks to generate a hierarchy of clusters. It is also called Hierarchical Cluster Analysis (HCA). Hierarchical clustering methods build a nested series of clusters in comparison to partitioned methods that generate only a flat set of clusters.

There are two types of Hierarchical clustering: agglomerative and divisive.



Agglomerative

This is a 'bottom-up' approach. In this approach, each object is a cluster by itself at the start and its nearby clusters are repetitively combined resulting in larger and larger clusters until some stopping criterion is met. The stopping criterion may be the specified number of clusters or a stage at which all the objects are combined into a single large cluster that is the highest level of hierarchy as shown in Figure 7.10.

Divisive

This is a 'top-down' approach. In this approach, all objects start from one cluster, and partitions are performed repeatedly resulting in smaller and smaller clusters until some stopping criterion is met or each cluster consists of the only object in it as shown in Figure 7.10. Generally, the mergers and partitions are decided in a greedy manner.

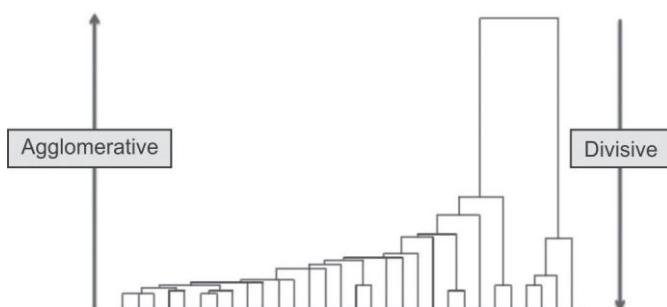


Figure 7.10 Illustration of agglomerative and divisive clustering

[Credits: http://chem-eng.utoronto.ca/~datamining/dmc/clustering_hierarchical.htm]

| | | | |
|----------------------|---|--|--|
| | 1 | | |
| Let's disc uss | 2 | | |
| | 3 | | |

agglomerative and divisive clustering in detail.

7.14.1 Agglomerative clustering

In agglomerative clustering, suppose a set of N items are given to be clustered, the procedure of clustering is given as follows:

1. First of all, each item is allocated to a cluster. For example, if there are N items then, there will be N clusters, i.e., each cluster consisting of just one item. Assume that the distances (similarities) between the clusters are same as the distances (similarities) between the items they contain.
2. Identify the nearest (most similar) pair of clusters and merge that pair into a single cluster, so that now we will have one cluster less.
3. Now, the distances (similarities) between each of the old clusters and the new cluster are calculated.
4. Repeat steps 2 and 3 until all items are merged into a single cluster of size N .

As this type of hierarchical clustering merges clusters recursively, it is therefore known as agglomerative.

Once the complete hierarchical tree is obtained, then there is no point of grouping N items in a single cluster. You just have to cut the $k-1$ longest links to get k clusters.

Let's understand the working of agglomerative clustering by considering the following raw data given in Figure 7.11.

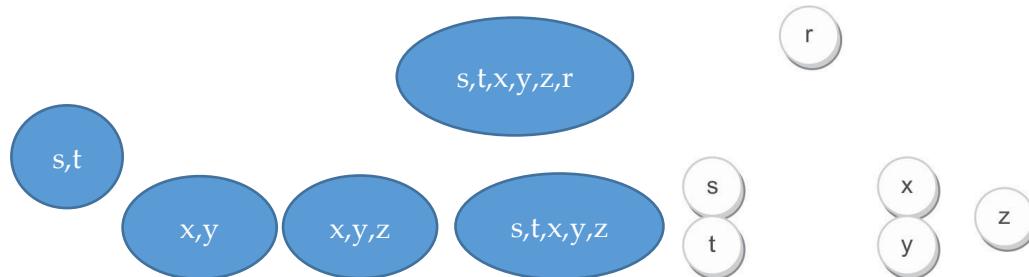


Figure 7.11 Raw data for agglomerative clustering

This method generates the hierarchy by recursively combining clusters from the individual items. For example, there are six items $\{r\}$, $\{s\}$, $\{t\}$, $\{x\}$, $\{y\}$ and $\{z\}$. The first step is to identify the items to be combined into a cluster. Normally, the two closest elements are taken on the basis of the chosen distance.

When the tree is cut at a given height, it gives a partitioning clustering at a selected precision. In this case, cutting the tree after the second row produces clusters $\{r\}$ $\{s\} \{t\}$ $\{x\} \{y\}$ and $\{z\}$ as shown in Figure 7.12. Cutting the tree after the third row will produce clusters $\{r\}$ $\{s\} \{t\}$ and $\{x\} \{y\} \{z\}$, which is called as **coarser clustering**, with a smaller number of clusters, but which are larger in size. The complete hierarchical clustering tree diagram is as given in Figure 7.12.

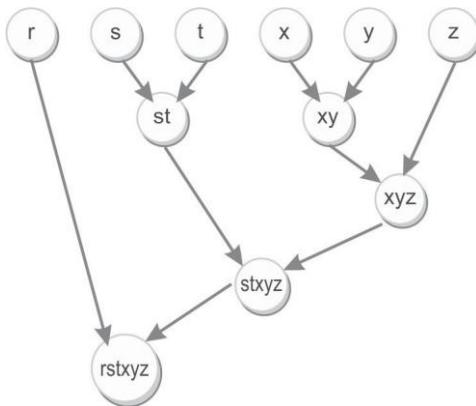


Figure 7.12 Complete hierarchical clustering tree diagram

7.7.1.1 Role of linkage metrics

It is required to compute the distance between clusters by using some metrics before any clustering is performed. These metrics are called linkage metrics.

It will be beneficial to identify the proximity matrix consisting of the distance between each point using a distance function.

The important linkage metrics to measure the distance between each cluster are given below:

Single linkage

In single linkage hierarchical clustering, the shortest distance between two points in each cluster is considered as the distance between two clusters. For example, the distance between two clusters 'r' and 's' to the left is equal to the length of the arrow between their two closest points as shown in Figure 7.13 and the formula of calculation of distance using single-linkage hierarchical clustering is given by:

$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

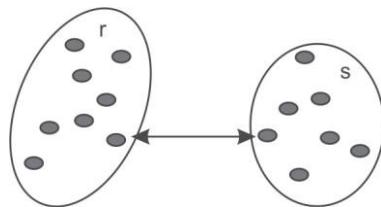


Figure 7.13 Single linkage

[Credits: http://chem-eng.utoronto.ca/~datamining/dmc/clustering_hierarchical.htm]

Complete linkage

In complete linkage hierarchical clustering, the longest distance between two points in each cluster is considered as the distance between two clusters. For example, the distance between clusters 'r' and 's' to the left is equal to the length of the arrow between their two farthest points as shown in Figure 7.14 and the formula of calculation of distance using complete linkage hierarchical clustering is given by:

$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

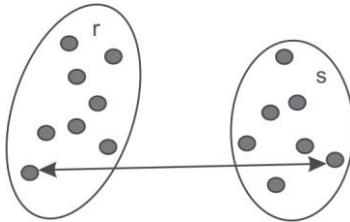


Figure 7.14 Complete linkage

[Credits: http://chem-eng.utoronto.ca/~datamining/dmc/clustering_hierarchical.htm]

Average linkage

In average linkage hierarchical clustering, the average distance between each point in one cluster to every point in the other cluster is considered as the distance between two clusters. For example, the distance between clusters 's' and 'r' to the left is equal to the average length of each arrow connecting the points of one cluster to the other as shown in Figure 7.15 and the formula of calculation of distance using average linkage hierarchical clustering is given by:

$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

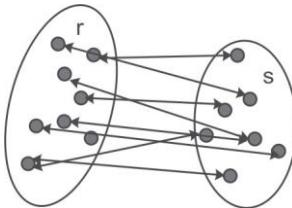


Figure 7.15 Average linkage

[Credits: http://chem-eng.utoronto.ca/~datamining/dmc/clustering_hierarchical.htm]

Example of Single-linkage hierarchical clustering

Let's perform a hierarchical clustering of some Indian cities on the basis of distance in kilometers between them. In this case, single-linkage clustering is used to calculate the distance between clusters.

The distance matrix (with a sequence number, m: 0) shown in Table 7.21 represents the distance in kilometers among some of the Indian cities. Suppose initially the level (L) is 0 for all the clusters.

Table 7.21 Input distance matrix ($L = 0$ for all the clusters)

| | <i>Delhi</i> | <i>Jammu</i> | <i>Srinagar</i> | <i>Patiala</i> | <i>Amritsar</i> | <i>Pahalgam</i> | <i>S/p</i> |
|-------------------|--------------|--------------|-----------------|----------------|-----------------|-----------------|------------|
| <i>Delhi</i> | 0 | 575 | 853 | 267 | 452 | 842 | |
| <i>Jammu</i> | 575 | 0 | 294 | 359 | 203 | 285 | |
| <i>Srinagar</i> | 853 | 294 | 0 | 627 | 469 | 98 | |
| <i>Patiala</i> | 267 | 359 | 627 | 0 | 235 | 610 | |
| <i>Amritsar</i> | 452 | 203 | 469 | 235 | 0 | 456 | |
| <i>Pahalgam</i> | 842 | 285 | 98 | 610 | 456 | 0 | |
| <i>Srinagar/p</i> | 842 | 285 | 0 | 610 | 456 | 0 | |

The nearest pair of cities is Srinagar and Pahalgam as the distance between them is 98. These cities are combined into a single cluster called 'Srinagar/Pahalgam'. Then, the distance is calculated from this new compound object to all other objects. In case of single link clustering, the shortest distance from any object of the cluster to the outside object is considered as the distance from the compound object to another object. Therefore, distance from 'Srinagar/Pahalgam' to Delhi is picked to be 842, i.e., $\min(853, 842)$, which is the distance from Pahalgam to Delhi.

Now, the level of the new cluster is $L(\text{Srinagar/Pahalgam}) = 98$ and **m: 1 is the new sequence number**. By following the same method, new distance from the compound object is calculated as shown in Table 7.22 and the description of calculation is given as follows.

$$*\min(853, 842) \quad **\min(294, 285) \quad ***\min(627, 610)$$

Table 7.22 Input distance matrix, with m: 1

| | <i>Delhi</i> | <i>Jammu</i> | <i>Srinagar/ Pahalgam</i> | <i>Patiala</i> | <i>Amritsar</i> |
|--------------------------|--------------|--------------|-------------------------------|----------------|-----------------|
| <i>Delhi</i> | 0 | 575 | 842 | 267 | 452 |
| <i>Jammu</i> | 575 | 0 | 285 | 359 | 203 |
| <i>Srinagar/Pahalgam</i> | 842* | 285** | 0 | 610*** | 456 |
| <i>Patiala</i> | 267 | 359 | 610 | 0 | 235 |
| <i>Amritsar</i> | 452 | 203 | 456 | 235 | 0 |
| <i>J/A</i> | 452 | 0 | 285 | 235 | 0 |

At the level of m:1, the minimum distance ($\min(\text{Jammu}, \text{Amritsar})$) in the distance matrix is 203, i.e., the distance between Jammu and Amritsar so, Jammu and Amritsar will be merged into

a new cluster called Jammu/Amritsar with new $L(\text{Jammu/Amritsar}) = 203$ and $m = 2$ as shown in

Table 7.23. The description of the calculation of the new distance from the compound object to other objects is given as follows.

$$* \min(575, 452) \quad ** \min(285, 456) \quad *** \min(359, 235)$$

Table 7.23 Input distance matrix, with m: 2

| | <i>Delhi</i> | <i>Jammu/Amritsar</i> | <i>Srinagar/Pahalgam</i> | <i>Patiala</i> |
|--------------------------|--------------|-----------------------|--------------------------|----------------|
| <i>Delhi</i> | 0 | 452 | 842 | 267 |
| <i>Jammu/Amritsar</i> | 452* | 0 | 285** | 235** |
| <i>Srinagar/Pahalgam</i> | 842 | 285 | 0 | 610 |
| <i>Patiala</i> | 267 | 235 | 610 | 0 |
| j/a/p | 267 | 0 | 285 | |

At the level of m:2, the minimum distance ($\min(Jammu/Amritsar, Patiala)$) in the distance matrix is 235, i.e., distance between Jammu/Amritsar and Patiala so, Jammu/Amritsar will be merged with Patiala into a new cluster called Jammu/Amritsar/Patiala with $L(Jammu/Amritsar/Patiala) = 235$

and m = 3 as shown in Table 7.24. The description of the calculation of the new distance from the compound object to other objects is given as follows.

$$* \min(452, 267) \quad ** \min(285, 610)$$

Table 7.24 Input distance matrix, with m: 3

| | <i>Delhi</i> | <i>Jammu/Amritsar/Patiala</i> | <i>Srinagar/Pahalgam</i> |
|-------------------------------|--------------|-------------------------------|--------------------------|
| <i>Delhi</i> | 0 | 267 | 842 |
| <i>Jammu/Amritsar/Patiala</i> | 267* | 0 | 285** |
| <i>Srinagar/Pahalgam</i> | 842 | 285 | 0 |

At the level of m:3, the minimum distance ($\min(Jammu/Amritsar/Patiala, Delhi)$) in the distance matrix is 267, i.e., the distance between Jammu/Amritsar/Patiala and Delhi so, Jammu/Amritsar/Patiala will be merged with Delhi into a new cluster called Delhi/Jammu/Amritsar/Patiala with $L(Delhi/Jammu/Amritsar/Patiala) = 267$ and m = 4 as shown in Table 7.25. The description of the calculation of the new distance from the compound object to other objects is given as follows.

$$* \min(285, 842)$$

Table 7.25 Input distance matrix, with m: 4

| | <i>Delhi/ Jammu/Amritsar/Patiala</i> | <i>Srinagar/Pahalgam</i> |
|-------------------------------------|--------------------------------------|--------------------------|
| <i>Delhi/Jammu/Amritsar/Patiala</i> | 0 | 285* |
| <i>Srinagar/Pahalgam</i> | 285 | 0 |

Finally, the last two clusters are merged at level 285. The whole procedure is outlined by the hierarchical tree as shown in Figure 7.16.

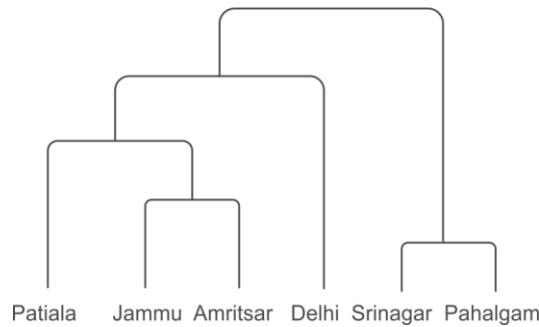


Figure 7.16 Hierarchical tree of clustering of Indian cities on the basis of distance

Example of clustering of students record with the agglomerative algorithm

Let us apply the Agglomerative algorithm on the database containing results of students' examination in a given course (earlier discussed for the k-mean algorithm). The students' performance on the basis of marks obtained out of 100 marks distributed across Quiz1, MSE, Quiz2, and ESE is given again in Table 7.26.

Table 7.26 Record of students' performance

| <i>Instances</i> | <i>Quiz 1</i> | <i>MSE</i> | <i>Quiz2</i> | <i>ESE</i> |
|------------------|---------------|------------|--------------|------------|
| S1 | 8 | 20 | 6 | 45 |
| S2 | 6 | 18 | 7 | 42 |
| S3 | 5 | 15 | 6 | 35 |
| S4 | 4 | 13 | 5 | 25 |
| S5 | 9 | 21 | 8 | 48 |
| S6 | 7 | 20 | 9 | 44 |
| S7 | 9 | 17 | 8 | 49 |
| S8 | 8 | 19 | 7 | 39 |
| S9 | 3 | 14 | 4 | 22 |
| S10 | 6 | 15 | 7 | 32 |

Step 1:

In this step, the distance matrix is computed from the above data using the centroid method between every pair of instances that are to be clustered. Due to the symmetric nature of the distance matrix (because the distance between p and q is same as the distance between q and p), only the lower triangular matrix is shown in Table 7.27 as the upper triangle can be filled in by reflection. The distance matrix consists of zeroes on the diagonal because every instance is at zero distance from itself.

The description of some of the calculations for cells of the second and third row of the distance matrix given in Table 7.27 drawn from Table 7.26 is given as follows.

$$\text{*Second row S2: } [S2-S1] = [|6-8| + |18-20| + |7-6| + |42-45|] = 8$$

$$\text{**Third row S3: } [S3-S1] = [|5-8| + |15-20| + |6-6| + |35-45|] = 18$$

$$\text{***Third row S3: } [S3-S2] = [|5-6| + |15-18| + |6-7| + |35-42|] = 12$$

Fourth row S4: [S4-s1], [s4-s2], [s4-s3]

Table 7.27 Distance matrix at m: 0

| Instances | S1 | S2 | S3 | S4 | S5/s 7 | S6 | S7 | S8 | S9 | S10 |
|-----------|------|-------|----|----|-----------|----|----|----|----|-----|
| S1 | 0 | | | | | | | | | |
| S2 | 8* | 0 | | | | | | | | |
| S3 | 18** | 12*** | 0 | | | | | | | |
| S4 | 32 | 26 | 14 | 0 | | | | | | |
| S5 | 7 | 13 | 25 | 39 | 0 | | | | | |
| S6 | 5 | 7 | 19 | 33 | 8 | 0 | | | | |
| S7 | 10 | 12 | 22 | 36 | 5 | 11 | 0 | | | |
| S8 | 8 | 6 | 12 | 26 | 13 | 9 | 14 | 0 | | |
| S9 | 36 | 30 | 18 | 6 | 43 | 37 | 40 | 30 | 0 | |
| S10 | 21 | 13 | 5 | 13 | 26 | 20 | 23 | 13 | 17 | 0 |
| S5/s7 | 7 | 12 | 22 | 36 | 0 | | | | | |
| | | | | | 8 | | | | | |
| | | | | | 9 | | | | | |
| | | | | | 37 | | | | | |
| | | | | | 20 | | | | | |

Step 2:

Apply the agglomerative method on this distance matrix and calculate the minimum distance between clusters and combine them. The minimum distance in the distance matrix is 5. Let's randomly select the minimum distance, i.e., 5 between S7 and S5.

Thus, the instances S7 and S5 are merged into a cluster called C1 and the distance is calculated

from this new compound object to all other objects by following the single link hierarchical clustering approach for calculation. To attain a new distance matrix, the instances S5 and S7 need to be removed and are replaced by cluster C1. Since single link hierarchical clustering is used, the distance between cluster C1 and every other instance is the shortest of the distance between an instance and S5; and an instance and S7. The cells involved in merging record number S5 and S7 for the creation of a new cluster are highlighted in Table 7.28.

Table 7.28 Cells involved in C1

| Instances | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 |
|-----------|------|-------|----|----|--------------|----|----|----|----|-----|
| S1 | 0 | | | | | | | | | |
| S2 | 8* | 0 | | | | | | | | |
| S3 | 18** | 12*** | 0 | | | | | | | |
| S4 | 32 | 26 | 14 | 0 | | | | | | |
| S5 | 7 | 13 | 25 | 39 | 0 | | | | | |
| S6 | 5 | 7 | 19 | 33 | 8 | 0 | | | | |
| S7 | 10 | 12 | 22 | 36 | 5 | 11 | 0 | | | |
| S8 | 8 | 6 | 12 | 26 | 13 ← 9 → 14 | 0 | | | | |
| S9 | 36 | 30 | 18 | 6 | 43 ← 37 → 40 | 30 | 0 | | | |
| S10 | 21 | 13 | 5 | 13 | 26 ← 20 → 23 | 13 | 17 | 0 | | |

The description of some of the calculations to create a new distance matrix with m: 2 is given as follows.

For example, to create a new distance matrix after merging S5 and S7, calculations will be as:

$$d(S1, S5) = 7 \text{ and } d(S1, S7) = 10, \text{ so } d(S1, C1) = \min(7, 10) = 7^*$$

$$\text{Similarly, } d(S2, S5) = 13 \text{ and } d(S2, S7) = 12, \text{ so } d(S2, C1) = \min(13, 12) = 12^{**}$$

The distance between S6 and C1 will be (minimum of distance between S6-S5 and S7-S6, i.e. min (8, 11), i.e., 8***). By using the highlighted cells, the data in C1 is filled as shown in Table 7.29.

$$d(S8, S5) = 13 \text{ and } d(S8, S7) = 14, \text{ so } d(S8, C1) = \min(13, 14) = 13****$$

$$\text{Similarly, } d(S9, S5) = 43 \text{ and } d(S9, S7) = 40, \text{ so } d(S9, C1) = \min(43, 40) = 40*****$$

The instances with the smallest distance get clustered next.

Table 7.29 Input distance matrix, with m: 2

| Instances | S1 | S2 | S3 | S4 | C1 | S6 | S8 | S9 | S10 |
|-----------|----|------|----|----|---------|----|----|----|-----|
| S1 | 0 | | | | | | | | |
| S2 | 8 | 0 | | | | | | | |
| S3 | 18 | 12 | 0 | | | | | | |
| S4 | 32 | 26 | 14 | 0 | | | | | |
| C1 | 7* | 12** | 22 | 36 | 0 | | | | |
| S6 | 5 | 7 | 19 | 33 | 8*** | 0 | | | |
| S8 | 8 | 6 | 12 | 26 | 13**** | 9 | 0 | | |
| S9 | 36 | 30 | 18 | 6 | 40***** | 37 | 30 | 0 | |
| S10 | 21 | 13 | 5 | 13 | 23 | 20 | 15 | 17 | 0 |
| S3/10 | 18 | 12 | 0 | | | | | | |



Step 3: Repeat Step 2 until all clusters are merged.

Combine S3 and S10 clusters into C2 as it has the minimum distance. The cells involved in merging record number 3 and 10 for the creation of C2 have been highlighted in Table 7.30.

Table 7.30 Cells involved in C2

| Instances | S1 | S2 | S3 | S4 | C1 | S6 | S8 | S9 | S10 |
|-----------|----|----|----|----|----|----|----|----|-----|
| S1 | 0 | | | | | | | | |
| S2 | 8 | 0 | | | | | | | |
| S3 | 18 | 12 | 0 | | | | | | |
| S4 | 32 | 26 | 14 | 0 | | | | | |
| C1 | 7 | 12 | 22 | 36 | 0 | | | | |
| S6 | 5 | 7 | 19 | 33 | 8 | 0 | | | |
| S8 | 8 | 6 | 12 | 26 | 13 | 9 | 0 | | |
| S9 | 36 | 30 | 18 | 6 | 43 | 37 | 30 | 0 | |
| s10 | 21 | 13 | 5 | 13 | 23 | 20 | 13 | 17 | 0 |

The description of the some of the calculations to create a new distance matrix with m:3 is given as follows.

For example, to create a new distance matrix after merging S3 and S10, calculations will be as:

$d(S1, S3) = 18$ and $d(S1, S10) = 21$, so $d(S1, C2) = \min(18, 21) = 18^*$

Similarly, $d(S_2, S_3) = 12$ and $d(S_2, S_{10}) = 13$, so $d(S_2, C_2) = \min(12, 13) = 12^{**}$

The distance between S4 and C2 will be (minimum of distance between S4-S3 and S10-S4, i.e. min(14,13), i.e., 13***). By using the highlighted cells, the data in C2 is filled as shown in Table 7.30.

$d(C1, S3) = 22$ and $d(C1, S10) = 23$, so $d(C1, C2) = \min(22, 23) = 22^{***}$

Similarly, $d(S6, S3) = 19$ and $d(S6, S10) = 20$, so $d(S6, C2) = \min(19, 20) = 19^{***}$

The input matrix at m: 3 is shown in Table 7.31.

Table 7.31 Input distance matrix, with m: 3

At the level of m: 3, the minimum distance in the distance matrix is 5 between S6 and S1, so S6 and S1 will be merged into C3. The cells involved in the merging of S6 and S1 for the creation of C3 have been highlighted in Table 7.32.

Table 7.32 Cells involved in creating C3

| Instances | S1 | S2 | C2 | S4 | C1 | S6 | S8 | S9 |
|-----------|-----|------|--------|----|----|----|----|----|
| S1 | 0 | | | | | | | |
| S2 | 8 | 0 | | | | | | |
| C2 | 18* | 12** | 0 | | | | | |
| S4 | 32 | 26 | 13*** | 0 | | | | |
| C1 | 7 | 12 | 22**** | 36 | 0 | | | |
| S6 | 5 | 7 | 19 | 33 | 8 | 0 | | |
| S8 | 8 | 6 | 12 | 26 | 13 | 9 | 0 | |
| S9 | 36 | 30 | 17 | 6 | 40 | 37 | 30 | 0 |

The description of calculation of some of new distances from compound object C3 to other objects is given below.

$$d(S2, S1) = 8 \text{ and } d(S2, S6) = 7 \text{ so, } d(S2, C3) = \min(8, 7) = 7^*$$

$$d(C2, S1) = 18 \text{ and } d(C2, S6) = 19 \text{ so, } d(C2, C3) = \min(18, 19) = 18^{**}$$

$$\text{Similarly, } d(S8, S1) = 8 \text{ and } d(S8, S6) = 9 \text{ so, } \min(S8, C3) = \min(8, 9) = 8^{***}$$

The input matrix at m: 4 is shown in Table 7.33.

Table 7.33 Input distance matrix, with m: 4

| Instances | C3 | S2 | C2 | S4 | C1 | S8 | S9 |
|-----------|------|----|----|----|----|----|----|
| C3 | 0 | | | | | | |
| S2 | 7* | 0 | | | | | |
| C2 | 18** | 12 | 0 | | | | |
| S4 | 32 | 26 | 13 | 0 | | | |
| C1 | 7 | 12 | 22 | 36 | 0 | | |
| S8 | 8*** | 6 | 12 | 26 | 13 | 0 | |
| S9 | 36 | 30 | 17 | 6 | 40 | 30 | 0 |

At the level of m: 4, the minimum distance in the distance matrix is 6, so S9 and S4 will be merged into C4, and the cells involved in C4 are highlighted in Table 7.34.

| Instances | C3 | S2 | C2 | S4 | C1 | S8 | S9 |
|-----------|---------|------|------|----|----|----|----|
| C3 | 0 | | | | | | |
| S2 | 7* | 0 | | | | | |
| C2 | 18 | 12 | 0 | | | | |
| { S4 | 32] | 26] | 13] | 0 | | | |
| C1 | 7 | 12 | 22 | 36 | 0 | | |
| { S8 | 8 | 6 | 12 | 26 | 13 | 0 | |
| S9 | 36***] | 30] | 17] | 6 | 40 | 30 | 0 |

The calculation of some of new distances from the compound object C4 to other objects is given below.

$$d(C3, S4) = 32 \text{ and } d(C3, S9) = 36 \text{ so, } d(C3, C4) = \min(32, 36) = 32^*$$

$$d(C2, S4) = 13 \text{ and } d(C2, S9) = 17 \text{ so, } d(C2, C4) = \min(13, 17) = 13^{**}$$

Similarly, $d(C1, S4) = 36$ and $d(C1, S9) = 40$ so, $\min(C1, C4) = \min(36, 40) = 36^{***}$

The input matrix at m: 5 is shown in Table 7.35.

Table 7.35 Input distance matrix, with m: 5

| Instances | C3 | S2 | C2 | C4 | C1 | S8 |
|-----------|-----|----|------|-------|----|----|
| C3 | 0 | | | | | |
| S2 | 7 | 0 | | | | |
| C2 | 18 | 12 | 0 | | | |
| C4 | 32* | 26 | 13** | 0 | | |
| C1 | 7 | 12 | 22 | 36*** | 0 | |
| S8 | 8 | 6 | 12 | 26 | 13 | 0 |

At the level of m: 5, the minimum distance in the distance matrix is 6, so S8 and S2 will be merged into C5, and the cells involved in C5 are highlighted in Table 7.36.

Table 7.36 Cells involved in creating C5

| Instances | C3 | S2 | C2 | C4 | C1 | S8 |
|-----------|-----|----|------|-------|----|----|
| C3 | 0 | | | | | |
| { S2 | 7 | 0 | | | | |
| C2 | 18 | 12 | 0 | | | |
| { C4 | 32* | 26 | 13** | 0 | | |
| C1 | 7 | 12 | 22 | 36*** | 0 | |
| S8 | 8 | 6 | 12 | 26 | 13 | 0 |

The description of calculation of new distances from compound object C5 to other objects is given below. The input matrix at m:6 is shown in Table 7.37.

*min(12,12) **min(26,26) ***min(12,13)

Table 7.37 Input distance matrix, with m: 6

| Instances | C3 | C5 | C2 | C4 | C1 |
|-----------|----|-------|----|----|----|
| C3 | 0 | | | | |
| C5 | 7 | 0 | | | |
| C2 | 18 | 12* | 0 | | |
| C4 | 32 | 26** | 13 | 0 | |
| C1 | 7 | 12*** | 22 | 36 | 0 |
| C6 | C6 | | | | |

At the level of m: 6, the minimum distance in the distance matrix is 7, so C1 and C3 will be merged into C6, and the cells involved in C6 are highlighted in Table 7.38.

Table 7.38 Cells involved in creating C6

| Instances | C3 | C5 | C2 | C4 | C1 |
|-----------|----|-------|----|----|----|
| C3 | 0 | | | | |
| C5 | 7 | 0 | | | |
| C2 | 18 | 12* | 0 | | |
| C4 | 32 | 26** | 13 | 0 | |
| C1 | 7 | 12*** | 22 | 36 | 0 |

The calculation of some of new distances from compound object C6 to other objects is given below. The input matrix at m: 7 is shown in Table 7.39.

*min(12,7) **min(22,18) ***min(36,32)

Table 7.39 Input distance matrix, with m: 7

| Instances | C6 | C5 | C2 | C4 |
|-----------|-------|----|----|----|
| C6 | 0 | | | |
| C5 | 7* | 0 | | |
| C2 | 18** | 12 | 0 | |
| C4 | 32*** | 26 | 13 | 0 |

At the level of m: 7, the minimum distance in the distance matrix is 7, so C5 and C6 will be

merged into C7, and the cells involved in C7 are highlighted in Table 7.40.

| Instances | C6 | C5 | C2 | C4 |
|-----------|--------------------------------|----|----|----|
| C6 | 0 | | | |
| C5 | 7* | 0 | | |
| C2 | 18** \longleftrightarrow 12 | | 0 | |
| C4 | 32*** \longleftrightarrow 26 | | 13 | 0 |

The calculation of some of new distances from compound object C7 is given below. The input matrix at m: 8 is shown in Table 7.41.

$$* \min(12, 18) \quad ** \min(26, 32)$$

Table 7.41 Input distance matrix, with m: 8

| Instances | C7 | C2 | C4 |
|-----------|------|----|----|
| C7 | 0 | | |
| C2 | 12* | 0 | |
| C4 | 26** | 13 | 0 |

At the level of m:8, the minimum distance in the distance matrix is 12, so C2 and C7 will be merged into C8, and the cells involved in C8 are highlighted in Table 7.42.

Table 7.42 Cells involved in creating C8

| Instances | C7 | C2 | C4 |
|-----------|----------------------------|----|----|
| C7 | 0 | | |
| C2 | 12* | 0 | |
| C4 | 26** \longleftrightarrow | 13 | 0 |

The input matrix at m: 9 is shown in Table 7.43.

Table 7.43 Input distance matrix, with m: 9

| Instances | C8 | C4 |
|-----------|----|----|
| C8 | 0 | |
| C4 | 13 | 0 |

Finally, the last two clusters are merged at level 8. The whole procedure is outlined by the hierarchical tree as shown in Figure 7.17. Thus, all clusters are eventually combined.

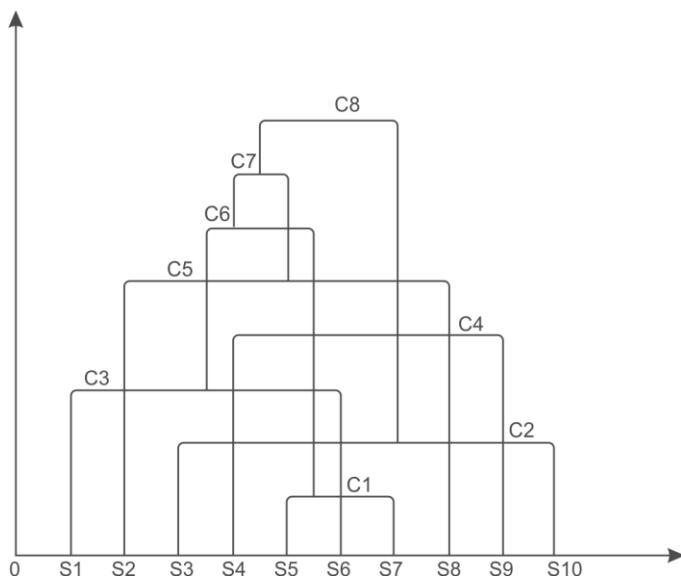


Figure 7.17 Hierarchical tree of clustering of students on the basis of their marks

7.1.12 Weakness of agglomerative clustering methods

The major weaknesses of agglomerative clustering methods are given as follows:

- Agglomerative clustering methods do not scale well as they have a high time complexity of at least $O(n^2)$ as the algorithm will have to compute $n*n$ distances, where n is the number of total objects.
- There is no undo mechanism to undo what was done previously.

7.1.2 Divisive clustering

As discussed earlier, in divisive clustering, all the objects or observations are assigned to a single cluster and then that single cluster is partitioned into two least similar clusters. This approach is repeatedly followed on each cluster until there is only one cluster for each object or observation.

Let us now apply a divisive algorithm on the same database discussed earlier for k-means and agglomerative clustering that contains the results of student examination in a given course. The students' performance on the basis of marks obtained out of 100 marks distributed across Quiz1, MSE, Quiz2, and ESE is given again in Table 7.44.

*Second row S2: $[S2-S1] = [|6-8| + |18-20| + |7-6| + |42-45|] = 8$

**Third row S3: $[S3-S1] = [|5-8| + |15-20| + |6-6| + |35-45|] = 18$

***Third row S3: $[S3-S2] = [|5-6| + |15-18| + |6-7| + |35-42|] = 12$

Fourth row S4: $[S4-S1], 4-8+13-20+5-6+25-45=32$

$[S4-S2]$,

$[S4-S3]$

Table 7.44 Record of students' performance

| Roll no | Quiz1 | MSE | Quiz2 | ESE |
|---------|-------|-----|-------|-----|
| 1 | 8 | 20 | 6 | 45 |
| 2 | 6 | 18 | 7 | 42 |

| Roll no | Quiz1 | MSE | Quiz2 | ESE |
|---------|-------|-----|-------|-----|
| 3 | 5 | 15 | 6 | 35 |
| 4 | 4 | 13 | 5 | 25 |
| 5 | 9 | 21 | 8 | 48 |
| 6 | 7 | 20 | 9 | 44 |
| 7 | 9 | 17 | 8 | 49 |
| 8 | 8 | 19 | 7 | 39 |
| 9 | 3 | 14 | 4 | 22 |
| 10 | 6 | 15 | 7 | 32 |

Step 1:

The distance matrix is calculated in the same way as we did in agglomerative clustering by using the centroid method to calculate the distance between the clusters. For representation, Roll numbers have been converted to Instances S1 to S10. Due to the symmetric nature of the distance matrix, we need to represent only the lower triangular matrix as shown in Table 7.45.

Table 7.45 Distance matrix at m: 0

| <i>Instances</i> | <i>S1</i> | <i>S2</i> | <i>S3</i> | <i>S4</i> | <i>S5</i> | <i>S6</i> | <i>S7</i> | <i>S8</i> | <i>S9</i> | <i>S10</i> |
|------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|
| S1 | 0 | | | | | | | | | |
| S2 | 8 | 0 | | | | | | | | |
| S3 | 18 | 12 | 0 | | | | | | | |
| S4 | 32 | 26 | 14 | 0 | | | | | | |
| S5 | 7 | 13 | 25 | 39 | 0 | | | | | |
| S6 | 5 | 7 | 19 | 33 | 8 | 0 | | | | |
| S7 | 10 | 12 | 22 | 36 | 5 | 11 | 0 | | | |
| S8 | 8 | 6 | 12 | 26 | 13 | 9 | 14 | 0 | | |
| S9 | 36 | 30 | 18 | 6 | 43 | 37 | 40 | 30 | 0 | |
| S10 | 21 | 13 | 5 | 13 | 26 | 20 | 23 | 13 | 17 | 0 |

Step 2:

Pick up the two objects with the largest distance, i.e., 43 in this case between S5 and S9. Now, these two objects become the seed records of two new clusters. Split the entire group into two clusters based on the distances in the distance matrix as shown in Figure 7.18.

The details about some of the calculations of the distance matrix (referring to Table 7.45) for cluster C1 have been shown in Figure 7.18 (b) and similarly, the distances, for instance S9, i.e., C2 from other instances are computed.

(a)

| Instances | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 |
|-----------|----|----|----|----|----|----|----|----|----|-----|
| S1 | 0 | | | | | | | | | |
| S2 | 8 | 0 | | | | | | | | |
| S3 | 18 | 12 | 0 | | | | | | | |
| S4 | 32 | 26 | 14 | 0 | | | | | | |
| S5 | 7 | 13 | 25 | 39 | 0 | | | | | |
| S6 | 5 | 7 | 19 | 33 | 8 | 0 | | | | |
| S7 | 10 | 12 | 22 | 36 | 5 | 11 | 0 | | | |
| S8 | 8 | 6 | 12 | 26 | 13 | 9 | 14 | 0 | | |
| S9 | 36 | 30 | 18 | 6 | 43 | 37 | 40 | 30 | 0 | |
| S10 | 21 | 13 | 5 | 13 | 26 | 20 | 23 | 13 | 17 | 0 |

(a)

| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 |
|------------------|----|----|------|----|----|----|----|----|----|-----|
| S5, i.e., C1 | 7* | 13 | 25** | 39 | 0 | 8 | 5 | 13 | 43 | 26 |
| S9, i.e., C2 | 36 | 30 | 18 | 6 | 43 | 37 | 40 | 30 | 0 | 17 |
| Assigned Cluster | C1 | C1 | C2 | C2 | C1 | C1 | C1 | C1 | C2 | C2 |

Figure 7.18 (a) Distance matrix at m: 0 (b) Data objects split after two clusters

In Figure 7.18 (b), the instances with minimum distance from cluster C1 and C2 are assigned to that cluster. For example,

* $d(S1, C1) = 7$ and $d(S1, C2) = 36$, so, $\min(7, 36) = 7$, i.e., C1

**Similarly, $d(S3, C1) = 25$ and $d(S3, C2) = 18$, so, $\min(25, 18) = 18$, i.e., C2

From the figure, it has been observed that

Cluster C1 includes – S1, S2, S5, S6, S7, S8, and

Cluster C2 includes – S3, S4, S9, S10

Since the stopping criterion hasn't been met, we will split up the largest cluster, i.e., C1. For this distance matrix will be computed for C1 as given in Table 7.46.

Table 7.46 Distance matrix for cluster C1

| | S1 | S2 | S5 | S6 | S7 | S8 |
|----|----|----|----|----|----|----|
| S1 | 0 | | | | | |
| S2 | 8 | 0 | | | | |
| S5 | 7 | 13 | 0 | | | |
| S6 | 5 | 7 | 8 | 0 | | |
| S7 | 10 | 12 | 5 | 11 | 0 | |
| S8 | 8 | 6 | 13 | 9 | 14 | 0 |

Since, the largest distance is 14, so C1 is split with S7 and S8 as seeds. They become the seeds of two new clusters. Splitting the entire group into these two clusters based on the distances in the distance matrix is shown in Table 7.47.

Table 7.47 Splitting of cluster C1 into two new clusters of S7 and S8

| | S1 | S2 | S5 | S6 | S7 | S8 |
|------------------|----|----|----|----|----|----|
| S7,i.e., C3 | 10 | 12 | 5 | 11 | 0 | 14 |
| S8, i.e., C4 | 8 | 6 | 13 | 9 | 14 | 0 |
| Assigned Cluster | C4 | C4 | C3 | C4 | C3 | C4 |

Here, Cluster C3 includes S5, S7 and cluster C4 includes S1, S2, S6, S8.

We already have another cluster as C2 (as described in Figure 7.18 (b)) which includes S3, S4, S9, S10. We can divide the cluster C3 further into two elementary clusters.

Now, the largest cluster, in this case is either C2 or C4 (both have the same number of objects). Here, we are considering cluster C2 as a candidate for a further split. For this the distance matrix is computed for cluster C2 as given in Table 7.48.

Table 7.48 Distance matrix for cluster C2

| | S3 | S4 | S9 | S10 |
|-----|----|----|----|-----|
| S3 | 0 | | | |
| S4 | 14 | 0 | | |
| S9 | 18 | 6 | 0 | |
| S10 | 5 | 13 | 17 | 0 |

The largest distance is 18, so C2 has S3 and S9 as seeds for a further split. The result of splitting the entire group into these two clusters based on the distance in the distance matrix is given in Table 7.49.

Table 7.49 Splitting of cluster C2 into two new clusters of S3 and S9

| | S3 | S4 | S9 | S10 |
|------------------|----|----|----|-----|
| S3, i.e., C5 | 0 | 14 | 18 | 5 |
| S9, i.e., C6 | 18 | 6 | 0 | 17 |
| Assigned Cluster | C5 | C6 | C6 | C5 |

Here, cluster C5 includes S3, S10 and cluster C6 includes S4, S9.

Both clusters C5 and C6 can be divided into two more elementary clusters. We already have other clusters namely cluster C3 that includes S5, S7 and cluster C4 that includes S1, S2, S6, S8. Now, the largest cluster is C4 thus, C4 is the candidate for the further split. For this the distance matrix is computed for C4 as given in Table 7.50.

Table 7.50 Distance matrix for cluster C4

| | S1 | S2 | S6 | S8 |
|----|----|----|----|----|
| S1 | 0 | | | |
| S2 | 8 | 0 | | |
| S6 | 5 | 7 | 0 | |
| S8 | 8 | 6 | 9 | 0 |

The largest distance is 9, so C4 has S6 and S8 as seeds for a further split. The result of splitting the entire group into these two clusters based on the distance in the distance matrix is given in Table 7.51.

Table 7.51 Splitting of cluster C4 into two new clusters of S6 and S8

| | S1 | S2 | S6 | S8 |
|------------------|----|----|----|----|
| S6, i.e., C7 | 5 | 7 | 0 | 9 |
| S8, i.e., C8 | 8 | 6 | 9 | 0 |
| Assigned Cluster | C7 | C8 | C7 | C8 |

Here, Cluster C7 includes S1, S6 and cluster C8 includes S2, S8.

We already have other clusters as cluster C5 includes S3, S10 and cluster C6 includes S4, S9. At this stage each cluster can be divided into two clusters each of one object. Thus, the stopping criterion has been reached.

7.14.3 Density-based clustering

Density-based clustering algorithms perform the clustering of data by forming the cluster of nodes on the basis of the estimated density distribution of corresponding nodes in the region. In density-

based clustering, clusters are dense regions in the data space. These clusters are separated by regions of lower object density. A cluster is defined as a maximal set of density-connected points.

In 1996, Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu proposed a Density-based clustering algorithm. The major strength of this approach is that it can identify clusters of arbitrary shapes. A Density-based clustering method is known as DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

Familiarity with the following terms is a must in order to understand the workings of the DBSCAN algorithm.

Neighborhood (ε)

Neighborhood is an important term used in DBSCAN. It represents objects within a certain radius of from a centroid type object. The high-density neighborhood results if an object contains at least MinPts (minimum points) of objects in its neighborhood.

This concept is illustrated in Figure 7.19.

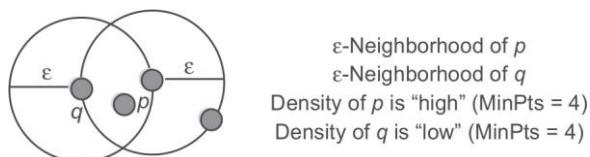


Figure 7.19 Concept of neighborhood and MinPts

Core, Border, and Outlier

A point is known as a core point if it has more than a specified number of points (MinPts) within neighborhood (ε). These points must lie at the interior of a cluster. A border point is a point if it has fewer than MinPts within neighborhood (ε), but it is in the neighborhood of a core point. A point is a noise or outlier point if it is neither a core point nor a border point. The concept of Core, Border, and Outlier is illustrated by considering Minpts = 4 in Figure 7.20.

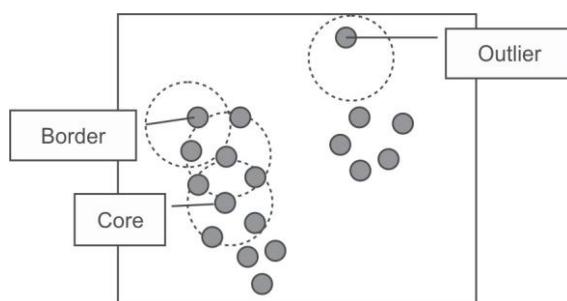


Figure 7.20 Concept of core, border and outlier

Density reachability; Directly and Indirectly density-reachable

DBSCAN's definition of a cluster is defined based on the concept of density reachability. Generally, a point q is directly density-reachable from a point p if it is not more distant than a given distance ε (i.e., is part of its ε -neighborhood). However, one may consider that p and q belong to the same cluster if point p is surrounded by necessarily many points.

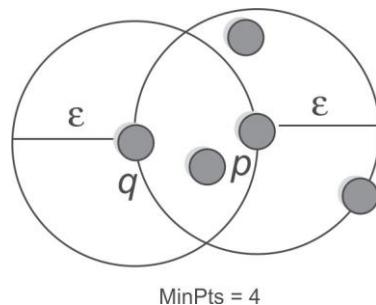


Figure 7.21 Concept of density reachability

As shown in Figure 7.21, if p is a core point and point q is in neighborhood ε of p then we can say that point q is directly density-reachable from point p .

Following are some important observations that can be drawn from data points given in Figure 7.21.

- Point q is directly density-reachable from point 'core point' p .
- Point q is not a core point as it has only 3 nodes in the neighborhood while MinPts is 4, therefore, p is not directly density-reachable from point q .
- Thus, Density-reachability is asymmetric

To further understand the concept of directly and indirectly density-reachable, let us consider Figure 7.22. Here, MinPts are 7.

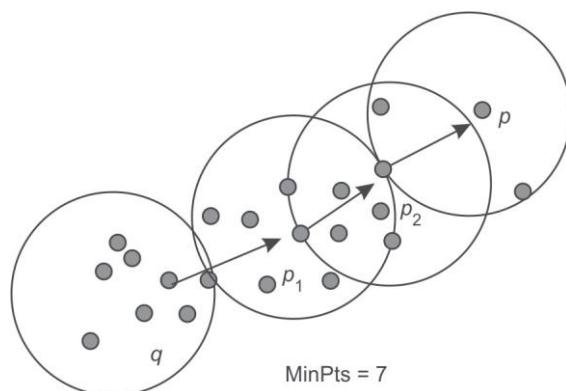


Figure 7.22 Concept of directly and indirectly density-reachable

As shown in Figure 7.22, following are some important observations:

- Point p_2 is a core point as it has more than 7 points within its neighborhood and point p is directly density-reachable from it.
- Point p_1 is also a core point and point p_2 is directly density-reachable from it.
- Point q is also a core point and point p_1 is directly density-reachable from it.
- Here, $p \rightarrow p_2 \rightarrow p_1 \rightarrow q$ form a chain.
- Point p is indirectly (through intermediate points p_1 and p_2) density-reachable from point q .
- Because p is not a core point as point q is not density-reachable from it. Point p has 6 nodes in the neighborhood while MinPts value is 7.

It is important to notice that the relation of density-reachable is not symmetric.

Let us consider another case given in Figure 7.23.

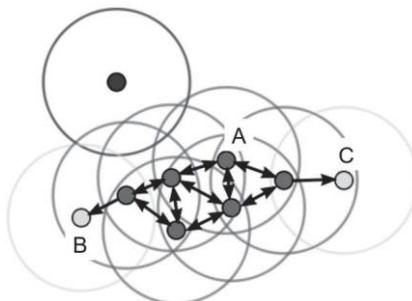


Figure 7.23 Another case of density reachability [see colour plate]

Here, in Figure 7.23, minPts = 4. Point A and the other red points are core points, because the area surrounding these points in an epsilon (ϵ) radius, contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.

Some more examples of DBSCAN are presented in Figure 7.24.

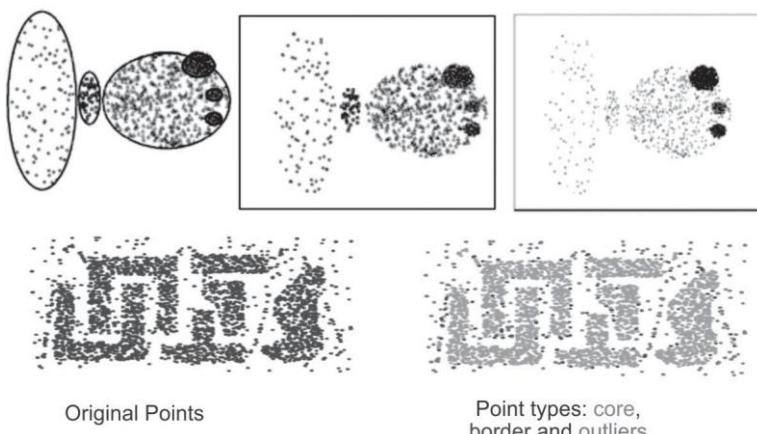


Figure 7.24 Some more examples of DBSCAN [see colour plate]

Thus, we can conclude that, a cluster then satisfies two properties:

- All points within the cluster are mutually density-connected.
- If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

7.14.4 **DBSCAN algorithm**

After the understanding of concepts of neighborhood, core, border, outlier points and density reachability, DBSCAN algorithm can be summarized as given below.

Algorithm for DBSCAN

1. Find the ϵ (epsilon) neighbors of every point, and identify the core points with more than minPts neighbors.
2. Find the connected components of core points on the neighbourhood graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an ϵ (eps) neighbor, otherwise assign it to noise.

7.14.5 **Strengths of DBSCAN algorithm**

DBSCAN algorithm has following advantages.

- It is not required to specify the number of clusters in the data at the start in case of the DBSCAN algorithm.
- It requires only two parameters and does not depend on the ordering of the points in the database.
- It can identify clusters of arbitrary shape. It is also able to identify the clusters completely surrounded by a different cluster.
- DBSCAN is robust to outliers and has a notion of noise.

7.14.6 **Weakness of DBSCAN algorithm**

DBSCAN algorithm has following disadvantages.

- The DBSCAN algorithm is sensitive to the parameter, i.e., it is difficult to identify the correct set of parameters.
- The quality of the DBSCAN algorithm depends upon the distance measures like neighborhood (ϵ) and MinPts.

The DBSCAN algorithm cannot cluster datasets accurately with varying densities or large differences in densities

9.1 Introduction to Association Rule Mining

Association rule mining often known as 'market basket' analysis is very effective technique to find the association of sale of item X with item Y. In simple words, market basket analysis consists of examining the items in the baskets of shoppers checking out at a market to see what types of items 'go together' as illustrated in Figure 9.1.

It would be useful to know, when people make a trip to the store, what kind of items do they tend to buy during that same shopping trip? For example, as shown in Figure 9.1, a database of customer transactions (i.e., shopping baskets) is given where each transaction consists of a set of items (i.e., products purchased during a visit). Association rule mining is used to identify groups of items which are frequently purchased together (customers' purchasing behavior). For example, 'IF one buys bread and milk, THEN he/she also buys eggs with high probability.' This information is useful for the store manager for better planning of stocking items in the store to improve its sale and efficiency.

Let us suppose that the store manager, receives customer complaints about heavy rush in his store and its consequent slow working. He may then decide to place associated items such as bread and

milk together, so that customers can buy the items easier and faster than if these were at a distance. It also improves the sale of each product. In another scenario, let us suppose his store is new and the store manager wishes to display all its range of products to prospective customers. He may then decide to put the associated items, i.e., bread and milk, at opposite ends of the store and would place other new items and items being promoted in between them, thereby ensuring that many customers, on the way from bread to milk, would take notice of these and some would end up buying them.

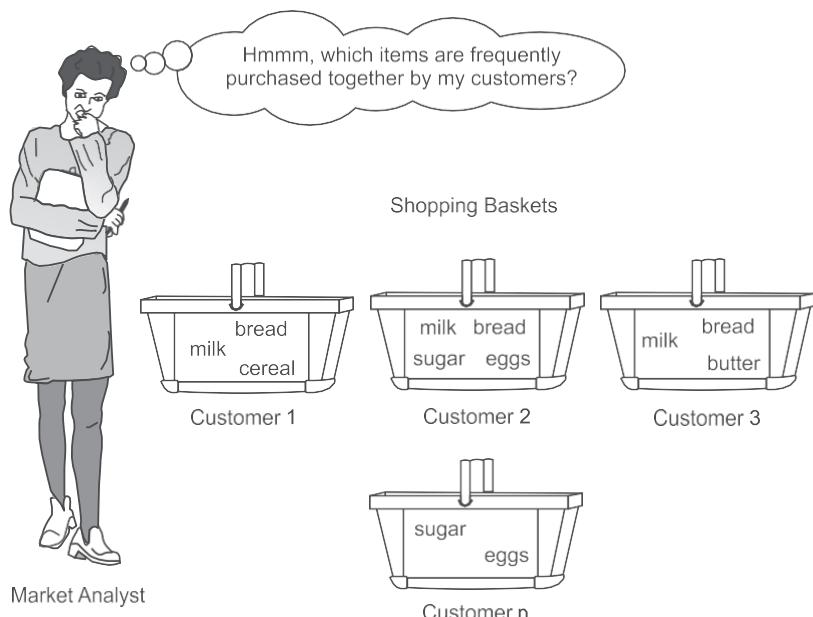


Figure 9.1 Need for association mining

Sometimes analysis of sale records leads to very interesting and unexpected results. In one very popular case study by Walmart USA, they had identified that people buying diapers often also bought beer. By putting the beer next to the diapers in their stores, it was found that the sales of each skyrocketed as depicted in Figures 9.2, 9.3 and 9.4.

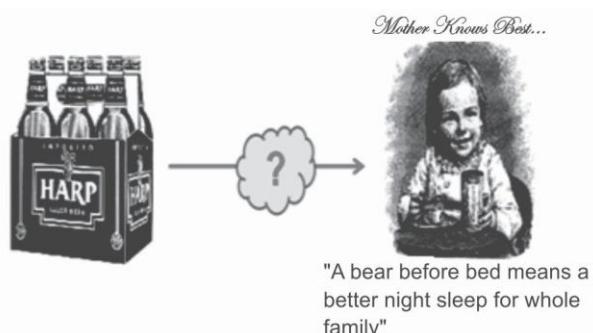


Figure 9.2 Association of sale of beer and diapers

One may make multiple conclusions about the association of beer and diapers. It may be funny to assume that 'A beer before bed means a better night's sleep for the whole family.' But serious analysis could lead to 'Young couples prepare for the weekend by stocking up on diapers for the infants and beer for dad'.



Figure 9.3 Association of sale of beer and diapers

Nevertheless, this story aptly illustrates the market basket concept of things 'going together', which is the basis of the data mining association function.



Figure 9.4 Association of sale of beer and diapers

Some other interesting findings of association mining may be trivial, for example, 'Customers who purchase maintenance agreements are very likely to purchase large appliances.' Or it may be unexplicable and unexpected, as in 'When a new restaurant opens, one of the most sold items is coffee'. This may be due to the reason that coffee is the cheapest item in the menu. So, to get the

feel of the restaurant, people only order coffee. This type of finding suggests to the manager to increase the cost of coffee. It happens to be the reason why in many good restaurants, the price of coffee is alarming high.

Often, while conducting market basket analysis or association rule mining, the only source of data available on customers is the bills of purchase, which tells what items go together in a shopping cart, and can also show what items 'go together' at certain times of the day, days of the week, seasons of the year, credit versus cash versus check payment, geographical locations of stores, and so on. Discovering associations among these attributes can lead to fact-based marketing strategies for things like store floor plans, special discounts, coupon offerings, product clustering, catalog design, identifying items that need to be put in combo packs and can be used to compare stores as shown in Figure 9.5.

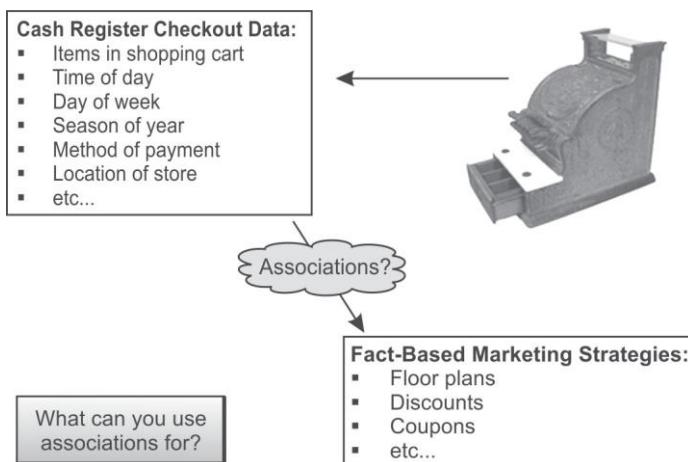


Figure 9.5 Association and customer purchase bills

Nowadays, recommendations given by online stores like Amazon and Flipkart also make use of association mining to recommend products related with your purchase and the system offers the list of products that others often buy together with the product you have just purchased. Besides the examples from market basket analysis given above, association rules are used today in many application areas such as intrusion detection, Web usage mining, bioinformatics and continuous production. Programmers use association rules to build programs capable of machine learning.

This is commonly known as market basket data analysis.

Association rule mining can also be used in applications like marketing, customer segmentation, web mining, medicine, adaptive learning, finance and bioinformatics, etc.

9.2 Defining Association Rule Mining

Association rule mining can be defined as identification of frequent patterns, correlations, associations, or causal structures among sets of objects or items in transactional databases, relational databases, and other information repositories.

Association rules are generally *if/then* statements that help in discovering relationships between seemingly unrelated data in a relational database or other information repository. For example, 'If a customer buys a dozen eggs, he is 80% likely to also purchase milk.'

An association rule consists of two parts, i.e., an antecedent (if) and a consequent (then). An antecedent is an object or item found in the data while a consequent is an object or item found in the combination with the antecedent.

Association rules are often written as $X \rightarrow Y$ meaning that whenever X appears Y also tends to appear. X and Y may be single items or sets of items. Here, X is referred to as the rule's antecedent and Y as its consequent.

For example, the rule found in the sales data of a supermarket could specify that if a customer buys onions and potatoes together, he or she will also like to buy burgers. This rule will be represented as onions, potatoes \rightarrow burger.

The concept of association rules was formulated by two Indian scientists Dr Rakesh Agrawal and Dr R. Srikant. The detail about their brief profile is given in Section 9.7.1.

9.3 Representations of Items for Association Mining

Let us assume that the number of items in the shop stocks is n. In Table 9.1, there are 6 items in stock, namely, bread, milk, diapers, beer, eggs, cola, thus $n = 6$ for this shop.

The item list is represented by I and its items are represented by {i₁, i₂, ..., i_n}.

The number of transactions are represented by N transactions, i.e., $N = 5$ for the shop data as given in Table 9.1.

Table 9.1 Sale database

| <i>TID</i> | <i>Items</i> |
|------------|------------------------------|
| 1 | {Bread, Milk} |
| 2 | {Bread, Diapers, Beer, Eggs} |
| 3 | {Milk, Diapers, Beer, Cola} |
| 4 | {Bread, Milk, Diapers, Beer} |
| 5 | {Bread, Milk, Diapers, Cola} |

Each transaction is denoted by T {t₁, t₂, ..., t_N} each with a unique identifier (TID) and each transaction consists of a subset of items (possibly a small subset) purchased by one customer.

Let each transaction of m items be {i₁, i₂, ..., i_m}, where $m \leq n$ [number of items in a transaction should be less than or equal to total items in the shop]. Typically, transactions differ in the number of items.

As shown in Table 9.1, the first transaction is represented as T₁ and it has two items, i.e., $m = 2$, with i₁ = Bread and i₂ = Milk. Similarly transaction T₄, has four items, having $m = 4$, with i₁ = Bread, i₂ = Milk, i₃ = Diapers and i₄ = Beer.

Our task will be to find association relationships, given a large number of transactions, such that items that tend to occur together are identified. It is important to note that association rules mining does not take into account the quantities of items bought.

9.4 The Metrics to Evaluate the Strength of Association Rules

The metrics to judge the strength and accuracy of the rule are as follows:

- Support
- Confidence
- Lift

9.4.1 Support

Let N is the total number of transactions. Support of X is represented as the number of times X appears in the database divided by N , while the support for X and Y together is represented as the number of times they appear together divided by N as given below.

$$\text{Support}(X) = (\text{Number of times } X \text{ appears}) / N = P(X)$$

$$\text{Support}(XY) = (\text{Number of times } X \text{ and } Y \text{ appear together}) / N = P(X \cap Y)$$

Thus, Support of X is the probability of X while the support of XY is the probability of $X \cap Y$.

Table 9.1, has been reproduced as Table 9.2 to calculate the supports for each item as given below:

Table 9.2 Sale database

| TID | Items |
|-----|------------------------------|
| 1 | {Bread, Milk} |
| 2 | {Bread, Diapers, Beer, Eggs} |
| 3 | {Milk, Diapers, Beer, Cola} |
| 4 | {Bread, Milk, Diapers, Beer} |
| 5 | {Bread, Milk, Diapers, Cola} |

$$\text{Support}(\text{Bread}) = \text{Number of times Bread appears} / \text{total number of translations} = 4/5 = P(\text{Bread})$$

$$\text{Support}(\text{Milk}) = \text{Number of times Milk appears} / \text{total number of translations} = 4/5 = P(\text{Milk})$$

$$\text{Support}(\text{Diapers}) = \text{Number of times Diapers appears} / \text{total number of translations} = 4/5 = P(\text{Diapers})$$

$$\text{Support}(\text{Beer}) = \text{Number of times Beer appears} / \text{total number of translations} = 3/5 = P(\text{Beer})$$

$$\text{Support}(\text{Eggs}) = \text{Number of times Eggs appears} / \text{total number of translations} = 1/5 = P(\text{Eggs})$$

$$\text{Support}(\text{Cola}) = \text{Number of times Cola appears} / \text{total number of translations} = 2/5 = P(\text{Cola})$$

$$\text{Support}(\text{Bread, Milk}) = \text{Number of times Bread, Milk appear together} / \text{total number of translations} = 3/5 = P(\text{Bread} \cap \text{Milk})$$

$\text{Support}(\text{Diapers}, \text{Beer}) = \text{Number of times Diapers, Beer appears together} / \text{total number of translations} = 3/5 = P(\text{Diapers} \cap \text{Beer})$

A high level of support indicates that the rule is frequent enough for the business to take interest in it.

Support is very important metric because if a rule has low support then it may be the case that the rule occurs by chance and it will not be logical to promote items that customers seldom buy together. But if a rule has high support then that association becomes very important and if implemented properly will result in increase in revenue, efficiency and customer satisfaction.

9.4.2 Confidence

To understand the concept of confidence, let us suppose that support for $X \rightarrow Y$ is 80%, then it means that $X \rightarrow Y$ is very frequent and there are 80% chances that X and Y will appear together in a transaction. This would be of interest to the sales manager.

Let us suppose we have another pairs of items (A and B) and support for $A \rightarrow B$ is 50%.

Of course it is not as frequent as $X \rightarrow Y$, but if this was higher, such as whenever A appears there is 90% chance that B also appears, then of course it would be of great interest.

Thus, not only the probability that A and B appear together matters, but also the conditional probability of B when A has already occurred plays a significant role. This conditional probability that B will follow when A has already been occurred is considered during determining the confidence of the rule.

Thus, Support and Confidence are important metrics to judge the quality of the association mining rule.

A high level of confidence shows that the rule is true often enough to justify a decision based on it.

Confidence for $X \rightarrow Y$ is defined as the ratio of the support for X and Y together to the support for X (which is same as the conditional probability of Y when X has already been occurred). Therefore if X appears much more frequently than X and Y appearing together, the confidence will be low.

$\text{Confidence of } (X \rightarrow Y) = \text{Support}(XY) / \text{Support}(X) = P(X \cap Y) / P(X) = P(Y|X)$

$P(Y|X)$ is the probability of Y once X has taken place, also called the conditional probability of Y.

Let us consider Table 9.3 and Ttable 9.4 to further understand the concept of support and confidence.

Table 9.3 Example of the support measure

| TID | Items | Support = Occurrence / Total Support |
|-----|-------|--------------------------------------|
| 1 | ABC | |
| 2 | ABD | Total Support = 5 |
| 3 | BC | Support {AB} = 2/5 = 40% |
| 4 | AC | Support {BC} = 3/5 = 60% |
| 5 | BCD | Support {ABC} = 1/5 = 20% |

Table 9.4 Example of the confidence measure

| TID | Items | Given $X \Rightarrow Y$ Confidence = Occurrence {X and Y} / Occurrence of (X) |
|-----|-------|--|
| 1 | ABC | |
| 2 | ABD | Confidence $\{A \Rightarrow B\} = 2/3 = 66\%$ |
| 3 | BC | Confidence $\{B \Rightarrow C\} = 3/4 = 75\%$ |
| 4 | AC | Confidence $\{AB \Rightarrow C\} = 1/2 = 50\%$ |
| 5 | BCD | |

Let us consider the database given in Table 9.5 for further understanding support and confidence.

Table 9.5 Database for identification of association rules

| Antecedent | Consequent |
|------------|------------|
| A | 0 |
| A | 0 |
| A | 1 |
| A | 0 |
| B | 1 |
| B | 0 |
| B | 1 |

There are two rules derived from the association of these combinations:

Rule 1: A implies 0, i.e., $A \rightarrow 0$

Rule 2: B implies 1, i.e., $B \rightarrow 1$

The support for Rule 1 is the Number of times A and 0 appear together / Total Number of transactions.

Thus, Support of Rule 1 = $3/7$

The Support for Rule 2 is the Number of times B and 1 appear together / Total Number of transactions.

Support of Rule 2 = $2/7$

The Confidence for Rule 1 is Support of $(A, 0)$ / Support A, i.e.,

(Number of times A and 0 appear together / Total number of items) DIVIDED BY
(Number of times A appears / Total number of items)

Here, total number of items get cancelled.

Thus, the Confidence for Rule 1 is the Number of times A and 0 appear together / Number of times A appears.

So the Confidence for Rule 1 is $3/4$

Similarly, the Confidence for Rule 2 is $2/3$

Going back to our general example involving 'X' (the Antecedent) and 'Y' (the Consequence), the Confidence of the rule does not depend on the frequency of 'Y' appearing. But in order to identify the strength of the rule it is important to consider the frequency of X and Y; X only; and Y only. In simple words, it is important to consider the whole dataset.

TheLift of the rule however considers the whole dataset, by taking into account the probability of Y also, in deciding the strength of the rule. So, Lift is the third important metric to check the strength or power of the rule.

9.4.3 Lift

It is very important to consider the frequency of Y or probability of Y for the effectiveness of the association mining rule $X \rightarrow Y$.

As already explained, Confidence is the conditional probability of Y when X has already occurred. It is very important to consider how frequent Y is to gauge the effectiveness of the confidence.

Thus, Lift is the ratio of conditional probability of Y when X is given to the unconditional probability of Y in the dataset. In simple words, it is Confidence of $X \rightarrow Y$ divided by the probability of Y.

$$\text{Lift} = P(Y|X) / P(Y)$$

Or

$$\text{Lift} = \text{Confidence of } (X \rightarrow Y) / P(Y)$$

Or

$$\text{Lift} = (P(X \cap Y) / P(X)) / P(Y)$$

Thus, lift can be computed by dividing the confidence by the unconditional probability of consequent Y.

Let us suppose that Coke is a very common sales item in a store and that it usually appears in most of the transactions. Let us suppose that we have a rule of Candle \rightarrow Coke which has a support of 20% and has a confidence of 90%. It is very logical to think that if coke is very popular and it appears in 95% of transactions, then obviously it also appears quite often with the candle as well. So, the rule for association of candle and coke will not be all that useful. But if we find that Candle \rightarrow Matchbox also has a support of 20% and a confidence of 90% then it is logical to suppose that the frequency of matchbox sales is very little as compared to the sale of coke. And the rule suggests that when we make a sale of candles, 90% chance indicates that a matchbox will also be sold in the same transaction. It is more effective and logical to conclude that when we sell a candle then we also sell a coke (coke is popular and will appear with every item not just with candle). As support and confidence are unable to handle this case, it is handled by the lift of the rule.

In this case, the probability of Y is very low in case of Candle \rightarrow Matchbox (Here, Y is matchbox) and will be very high in case of Candle \rightarrow Coke (Here, Y is coke).

One can note that the low probability of Y, makes the $X \rightarrow Y$ rule more effective as compared to high probability of Y. Lift takes the note of this, and is defined as follows.

$$\text{Lift} = P(Y|X) / P(Y)$$

Or

$$\text{Lift} = \text{Confidence of } (X \rightarrow Y) / P(Y)$$

Or

$$\text{Lift} = (P(X \cap Y) / P(X)) / P(Y)$$

Thus, the high probability of Y, i.e., P(Y) makes lift less effective and its low value makes it more effective.

Let us again consider the data given in Table 9.5, which has been reproduced in Table 9.6 for quick reference to calculate the lift of Rule 1 and Rule 2 as discussed earlier.

Table 9.6 Dataset

| Antecedent | Consequent |
|------------|------------|
| A | 0 |
| A | 0 |
| A | 1 |
| A | 0 |
| B | 1 |
| B | 0 |
| B | 1 |

$$\text{Lift for Rule1, i.e., } A \rightarrow 0 = P(0 | A) / P(0) = (P(A \cap 0) / P(A)) / P(0) = (3/4) / (4/7) = 1.3125$$

$$\text{Lift for Rule2, i.e., } B \rightarrow 1 = P(1 | B) / P(1) = (P(B \cap 1) / P(B)) / P(1) = (2/3) / (3/7) = 1.55$$

As discussed earlier, the confidence for Rule 1 is $3/4=0.75$ and confidence for Rule 2 is 0.66 .

It should be observed that although the Rule 1 has higher confidence as compared to Rule 2, but it has lower lift as compared to Rule 2.

Naturally, it would appear that Rule1 is more valuable because of having higher confidence; it appears more accurate (better supported). But, the accuracy of the rule can be misleading if it is independent of the dataset. Lift, as a metric is important because it considers both the confidence of the rule and the overall dataset.

In order to understand the importance of lift let us consider modified dataset given in Table 9.7.

In this modified database another five records have been added to the existing database (the five at the bottom). In these five records 0 appears four times, i.e., in the modified database 0 becomes more frequent; so, its probability has been increased, while the probability of 1 has been reduced.

Thus for first rule, i.e., $A \rightarrow 0$ the lift of the rule has been reduced in the modified database while it has same value of confidence as calculated below.

$$\text{Support of } A \rightarrow 0 = 3/12$$

$$\text{Confidence of } A \rightarrow 0 = \text{Support}(A,0) / \text{Support}(A) = 3/4 = 0.75$$

$$\text{Lift} = \text{Confidence}(A,0) / \text{Support}(0) = (3/4) / (8/12) = 36/32 = 1.125$$

Here, the lift of rule 1 has been reduced from 1.312 in the original dataset to 1.125 in the modified dataset, so the strength of rule 1 has been decreased in the modified dataset.

For Rule 2, i.e., $B \rightarrow 1$, the lift rule has been improved in the modified database while it has same value of confidence as calculated below.

$$\text{Support of } B \rightarrow 1 = 2/12$$

$$\text{Confidence of } B \rightarrow 1 = \text{Support}(B,1) / \text{Support}(B) = 2/3 = 0.66$$

$$\text{Lift} = \text{Confidence} / \text{Support}(1) = (2/3) / (4/12) = 2$$

Here, the lift of Rule 2 has been increased from 1.555 in the original dataset to 2 in the modified dataset, so the strength of Rule 2 has been improved in the modified dataset. Thus, lift considers both the confidence of the rule and the overall dataset.

Table 9.7 Modified dataset

| Antecedent | Consequent |
|------------|------------|
| A | 0 |
| A | 0 |
| A | 1 |
| A | 0 |
| B | 1 |
| B | 0 |
| B | 1 |
| C | 0 |
| D | 0 |
| C | 0 |
| C | 1 |
| E | 0 |

} Five new records inserted into the database

Another representation of association rules

Sometimes, the representation of association rules also includes support and confidence as shown in Figure 9.6.

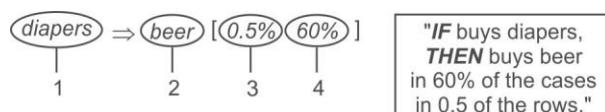


Figure 9.6 Representation of association rules

$$\text{diapers} \Rightarrow \text{beer} [0.5\%, 60\%]$$

Here,

1. Antecedent, left-hand side (LHS), body
2. Consequent, right-hand side (RHS), head
3. Support, frequency
4. Confidence, strength

9.5 The Naïve Algorithm for Finding Association Rules

To understand the Naïve Algorithm, let us consider the sales record of a grocery store. For simplicity, only four sale transactions of it are considered which deal with only four items for sale (Bread, Cornflakes, Jam and Milk) as shown in Table 9.8 and a naïve brute force algorithm is used to perform the task to identify the association of items in this sale record.

Let us find the association rules having minimum ‘support’ of 50% and minimum ‘confidence’ of 75%.

Table 9.8 Sale record of grocery store

| <i>Transaction ID</i> | <i>Items</i> |
|-----------------------|------------------------|
| 100 | Bread, Cornflakes |
| 101 | Bread, Cornflakes, Jam |
| 102 | Bread, Milk |
| 103 | Cornflakes, Jam, Milk |

9.5.1 Working of the Naïve algorithm

In order to find association rules, the first step of the naïve algorithm is to list all the combinations of the items that are in stock and then identify the frequent combinations or combinations having frequency more than or equal to a specified support limit. Using this approach, the association rules that have the ‘confidence’ more than the threshold limit are identified.

Here, we have four items in stock, thus there are 2^4 possible combinations that we can create by considering null as one combination and if we ignore null then the following 15 combinations are possible as given in Table 9.9 along with their frequencies of occurrence in the transaction database.

Table 9.9 List of all itemsets and their frequencies

| <i>Itemsets</i> | <i>Frequency</i> |
|---------------------|------------------|
| Bread | 3 |
| Cornflakes | 3 |
| Jam | 2 |
| Milk | 2 |
| (Bread, Cornflakes) | 2 |
| (Bread, Jam) | 1 |
| (Bread, Milk) | 1 |
| (Cornflakes, Jam) | 2 |

Contd.

| Itemsets | Frequency |
|--------------------------------|-----------|
| (Cornflakes, Milk) | 1 |
| (Jam, Milk) | 1 |
| (Bread, Cornflakes, Jam) | 1 |
| (Bread, Cornflakes, Milk) | 0 |
| (Bread, Jam, Milk) | 0 |
| (Cornflakes, Jam, Milk) | 1 |
| (Bread, Cornflakes, Jam, Milk) | 0 |

Here, the minimum required support is 50% and we have to identify the frequent itemsets that appear in at least two transactions. The list of frequencies shows that all four items Bread, Cornflakes, Jam and Milk are frequent. It has been observed that the frequency goes down when we look at 2-itemsets and only two 2-itemsets such as (Bread, Cornflakes) and (Cornflakes, Jam) are frequent. All the frequent itemsets has been shown in bold in Table 9.8. There are no three or four itemsets that are frequent. The frequent itemsets are provided in Table 9.10.

Table 9.10 The set of all frequent items

| Itemsets | Frequency |
|---------------------|-----------|
| Bread | 3 |
| Cornflakes | 3 |
| Jam | 2 |
| Milk | 2 |
| (Bread, Cornflakes) | 2 |
| (Cornflakes, Jam) | 2 |

As we are interested in association rules that can only occur with item pairs, thus individual frequent items Bread, Cornflakes, Jam and Milk are ignored, and item pairs (Bread, Cornflakes) and (Cornflakes, Jam) are considered for association rule mining.

Now, the association rules for the two 2-itemsets (Bread, Cornflakes) and (Cornflakes, Jam) are determined with a required confidence of 75%.

It is important to note that every 2-itemset (A, B) can lead to two rules A→B and B→A if both satisfy the required confidence. As stated earlier, confidence of A→B is given by dividing the support for A and B together, by the support for A.

Therefore, we have four possible rules which are given as follows along with their confidence:

Bread→Cornflakes

Confidence = Support of (Bread, Cornflakes) / Support of (Bread) = 2/3 = 67%

Cornflakes→Bread

Confidence = Support of (Cornflakes, Bread) / Support of (Cornflakes) = 2/3 = 67%

Cornflakes → Jam

Confidence = Support of (Cornflakes, Jam) / Support of (Cornflakes) = 2/3 = 67%

Jam → Cornflakes

Confidence = Support of (Jam, Cornflakes) / Support of (Jam) = 2/2 = 100%

Therefore, only the last rule Jam → Cornflakes has more than the minimum required confidence, i.e., 75% and it qualifies. The rules having more than the user-specified minimum confidence are known as confident.

9.5.2 Limitations of the Naïve algorithm

We can observe that in Table 9.7, we had a small number of items. However, the number of all possible combinations for which we have to calculate the frequency grows enormously as the number of items increase in the dataset. When dealing with 4 itemsets, it produces 15 different combinations in Table 9.8 and if we include a null combination (a combination with no items) as well, we would obtain 16. If there are six items it would have produced 64 combinations and if the number of items were 10 then we would have a table with 1024 combinations.

This simple algorithm works well with four items but if the number of items is say 100, the number of combinations is much larger, in billions. The number of combinations becomes about a million with 20 items since the number of combinations is 2^n with n items. The naive algorithm can be improved to deal more effectively with larger datasets.

9.5.3 Improved Naïve algorithm to deal with larger datasets

Rather than counting all the possible item combinations in the stock it will be better to focus only on the items that are sold in transactions. Because, we may have hundreds of items in stock but we are concerned with finding associations only for the items that are being sold together. Thus, it is natural to focus only on the items that are sold in transactions instead of all the items available in stock.

We can look at each transaction and count only the combinations that actually occur; with this approach we do not count itemsets with zero frequency. And that will be an improvement.

As an example, Table 9.11 lists all the actual combinations occurring within the transactions given in Table 9.8.

Table 9.11 All possible combinations with nonzero frequencies

| Transaction ID | Items | Combinations |
|----------------|------------------------|--|
| 100 | Bread, Cornflakes | (Bread, Cornflakes) |
| 200 | Bread, Cornflakes, Jam | (Bread, Cornflakes), (Bread, Jam), (Cornflakes, Jam), (Bread, Cornflakes, Jam) |
| 300 | Bread, Milk | (Bread, Milk) |
| 400 | Cornflakes, Jam, Milk | (Cornflakes, Jam), (Cornflakes, Milk), (Jam, Milk), (Cornflakes, Jam, Milk) |

We therefore only need to look at the following combinations and their frequencies are given in Table 9.12.

Table 9.12 Frequencies of all itemsets with nonzero frequencies

| Itemsets | Frequency |
|--------------------------|-----------|
| Bread | 3 |
| Cornflakes | 3 |
| Jam | 2 |
| Milk | 2 |
| (Bread, Cornflakes) | 2 |
| (Bread, Jam) | 1 |
| (Cornflakes, Jam) | 2 |
| (Bread, Cornflakes, Jam) | 1 |
| (Bread, Milk) | 1 |
| (Cornflakes, Milk) | 1 |
| (Jam, Milk) | 1 |
| (Cornflakes, Jam, Milk) | 1 |

We can now proceed as before. This would work better since the list of item combinations is significantly reduced from 15 (as given in Table 9.8) to 12 (as given in Table 9.11) and this reduction is likely to be much larger for bigger problems. Regardless of the extent of the reduction, this list will also become very large for, say, 1000 items since it includes all the nonzero itemsets that exist in the transactions database whether they have minimum support or not. Therefore, the naïve algorithm or any improvement of it is not suitable for large problems. We will discuss a number of algorithms for more efficiently solving the association rule problem starting with the classical *Apriori algorithm*.

Before starting with the Apriori algorithm, it is important to discuss different ways to store our transaction database in computer memory because, it will have an impact on performance if we are processing a large number of transactions. In the next section, implementation issues of transition storage have been discussed.

9.6 Approaches for Transaction Database Storage

It is important to understand different ways to store a dataset of transactions before processing it using algorithms, as storage is an important issue for its performance.

There are three ways to store datasets of transactions. These are as follows.

- Simple Storage
- Horizontal Storage
- Vertical Storage

Let us suppose the number of items be five; to demonstrate the different options. Let them be $\{I_1, I_2, I_3, I_4, I_5\}$. Let there be only seven transactions with transaction IDs $\{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$. This set of seven transactions with five items can be stored using three different methods given as follows.

9.6.1 Simple transaction storage

The first representation is commonly used and known as Simple Transaction Storage. Each row of the table shows the transaction ID and the purchased items as given in Table 9.13.

Table 9.13 A simple representation of transactions as an item list

| <i>Transaction ID</i> | <i>Items</i> |
|-----------------------|-----------------|
| T1 | I_1, I_2, I_4 |
| T2 | I_4, I_5 |
| T3 | I_1, I_3 |
| T4 | I_2, I_4, I_5 |
| T5 | I_4, I_5 |
| T6 | I_2, I_3, I_5 |
| T7 | I_1, I_3, I_4 |

9.6.2 Horizontal storage

In this representation, each row is still a transaction, but columns have been created for each item. In the cell, 1 is filled against the item that occurs in a transaction and 0 against the rest as shown in Table 9.14.

Table 9.14 Horizontal storage representation

| <i>TID</i> | <i>I1</i> | <i>I2</i> | <i>I3</i> | <i>I4</i> | <i>I5</i> |
|------------|-----------|-----------|-----------|-----------|-----------|
| T1 | 1 | 1 | 0 | 1 | 0 |
| T2 | 0 | 0 | 0 | 1 | 1 |
| T3 | 1 | 0 | 1 | 0 | 0 |
| T4 | 0 | 1 | 0 | 1 | 1 |
| T5 | 0 | 0 | 0 | 1 | 1 |
| T6 | 0 | 1 | 1 | 0 | 1 |
| T7 | 1 | 0 | 1 | 1 | 0 |

The advantage of this storage system is that we can count the frequency of each item by counting the '1's in the given column. As we can easily calculate the frequency of item I1 as 3 and item I4 as

-
1. We can also easily calculate the frequency of item pairs by counting the 1's that result after an 'AND' operation on corresponding columns. For example, the frequency of item pairs I1 and I2 is 1 (By AND operation on column of I1 and I2 we will get only one 1, i.e., T1).

9.6.3 Vertical representation

In this representation as given in Table 9.15, the transaction list is turned around. Rather than using each row to represent a transaction of the items purchased, each row now represents an item and it indicates transactions in which the item appears. The columns now represent the transactions. This representation is also called a TID-list since for each item it provides a list of TIDs (Transition Ids).

Table 9.15 Vertical storage representation

| Item | TID | | | | | | |
|------|-----|----|----|----|----|----|----|
| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
| I1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| I2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| I3 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| I4 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| I5 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

A vertical representation also facilitates counting of items by counting the number of 1s in each row and in case of 2-itemsets, where you want to find out the frequency of occurrences of 2 items together in a transaction, you have to refer to the intersection of 2 rows corresponding to the items, and inspect one column at a time.

Example: If you want to find out frequency of item pairs (I1,I2) just look at rows I1 and I2 for each column if values of both I1 and I2 is 1 then this means I1 and I2 are occurring together in a transaction hence write their count as 1, move further in some way for each transaction. Finally you will get Table 9.16.

From the table it can be seen that the count for item pair {I1,I2} is 1; similarly the count of item pair {I1, I4} is 2. (Table 9.16)

Table 9.16 Frequency of item pairs

| | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|--------|----|----|----|----|----|----|----|
| I1, I2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| I1, I4 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

The vertical representation is not storage efficient in case of very large number of transactions. Also, if an algorithm needs data in the vertical representation then there would be a cost in transforming data from the horizontal representation to the vertical one.

With this we can move on to the concept of the Apriori algorithm.

9.7 The Apriori Algorithm

The Apriori algorithm was developed by two Indians Rakesh Agrawal and Ramakrishnan Srikant in 1994, to mine frequent itemsets for identifying association rules.

It should be a matter of pride and motivation to have two Indians as inventors of association mining. Their work is the base for commonly used recommendation systems for modern applications such as product recommendations for online purchases and video recommendation and so on.

A brief profile of these two scientists, based on Wikipedia, has been presented below to motivate readers and in appreciation of their efforts.

9.7.1 About the inventors of Apriori

Rakesh Agrawal is the President and Founder of the Data Insights Laboratories. He is also the President of the Professor Ram Kumar Memorial Foundation.



Rakesh is an innovator and thought leader driven by the desire to make the world better through scientific breakthroughs and by building practical working systems. He is the recipient of the ACM-SIGKDD Inaugural Innovation Award, ACM-SIGMOD Edgar F. Codd Innovations Award, VLDB 10-Yr Most Influential Paper Award. Scientific American has included him in its first list of 50 top scientists and technologists.

Until recently, Rakesh was a Microsoft Technical Fellow and headed the Search Labs in Microsoft Research. Prior to joining Microsoft in March 2006, Rakesh was an IBM Fellow and led the Quest group at the IBM Almaden Research Center. Earlier, he was with the Bell Laboratories, Murray Hill from 1983 to 1989. He also worked for three years at a leading Indian, namely Bharat Heavy Electricals Ltd. He received the M.S. and Ph.D. degrees in Computer Science from the University of Wisconsin-Madison in 1983. He also holds a B.E. degree in Electronics and Communication Engineering from IIT-Roorkee, and a two-year Post Graduate Diploma in Industrial Engineering from the National Institute of Industrial Engineering (NITIE), Bombay. Both IIT-Roorkee and NITIE have decorated him with their distinguished alumni awards.

Rakesh has been granted 83 patents. He has published more than 200 research papers, many of them considered seminal. He has written the 1st as well as the 2nd highest cited of all papers in the fields of databases and data mining (18th and 26th most cited across all computer science). Wikipedia lists one of his papers as one of the most influential database papers. His papers have been cited more than 80,000 times, with more than 25 of them receiving more than 500 citations each and three of them receiving 5,000 citations each (Google Scholar). He is the most cited author in the field of database systems and the 26th most cited author across all of Computer Science (Citeseer). His research has been featured in N.B.C., New York Times, and several other media channels.

It is rare that a researcher's work creates not only a product, but a whole new industry. IBM's data mining product, Intelligent Miner, grew straight out of Rakesh's research. IBM's introduction of Intelligent Miner and associated services created a new category of software and services. His research has been incorporated into many other commercial products, including DB2 Mining Extender, DB2 OLAP Server, WebSphere Commerce Server, and Microsoft Bing Search engine, as well as many research prototypes and applications.

To know more, please visit <http://rakeshagrwal.org/>

Ramakrishnan Srikant is a Distinguished Research Scientist at Google. His research interests include data mining, online advertising, and user modeling.

He previously managed the Privacy Research group, part of the Intelligent Information Systems Research department at IBM's Almaden Research Center.

Srikant has published more than 30 research papers that have been extensively cited. He has been granted 25 patents. Dr Srikant was a key architect for the IBM Intelligent Miner, and contributed the association rules and sequential patterns modules to the product.



Srikant received the ACM SIGKDD Innovation Award (2006) for his seminal work on mining association rules and privacy preserving data mining, the 2002 ACM Grace Murray Hopper Award for his work on mining association rules, the VLDB 2004 Ten Year Best Paper Award, and the ICDE 2008 Influential Paper Award. He was named an IBM Research Division Master Inventor in 1999. He has received 2 Outstanding Technical Achievement Awards for his contributions to the Intelligent Miner, and an Outstanding Innovation Award for his work on Privacy in Data Systems.

Srikant received his M.S. and Ph.D. from the University of Wisconsin, Madison and his B. Tech. from the Indian Institute of Technology, Madras.

To know more, please visit <http://www.rsrikant.com/index.html>

9.7.2 Working of the Apriori algorithm

The Apriori algorithm has been named so on the basis that it uses prior knowledge of frequent itemset properties. This algorithm consists of two phases. In the first phase, the frequent itemsets, i.e., the itemsets that exceed the minimum required support are identified. In the second phase, the association rules meeting the minimum required confidence are identified from the frequent itemsets. The second phase is comparatively straight forward – therefore, the major focus of research in this field is to improve the first phase.

To understand the Apriori algorithm, let us follow the learn by example approach. We will first apply this algorithm on a simple database to illustrate its working and then we will go into detail to learn the concept more deeply.

Example: Let us consider an example of only five transactions and six items. We want to identify association rules with 50% support and 75% confidence with the Apriori algorithm. The transactions are given in Table 9.17.

Table 9.17 Transactions database

| Transaction ID | Items |
|----------------|------------------------------|
| T1 | Bread, Cornflakes, Eggs, Jam |
| T2 | Bread, Cornflakes, Jam |
| T3 | Bread, Milk, Tea |
| T4 | Bread, Jam, Milk |
| T5 | Cornflakes, Jam, Milk |

For 50% support each frequent item must appear in at least three transactions. The first phase of the Apriori algorithm will be to find frequent itemsets.

Phase 1: Identification of frequent itemsets

It starts with the identification of candidate 1 itemsets, represented by C1 (one itemsets that may be frequent) and it is always the items in the stock which the store deals with. Here, we have six items, so C1 will be as shown below.

Table 9.18 Candidate one itemsets C1

| Item |
|------------|
| Bread |
| Cornflakes |
| Eggs |
| Milk |
| Jam |
| Tea |

From Candidate 1 itemsets, frequent one-itemsets are represented by L1 and found by calculating the count of each candidate item and selecting only those counts which are equal to or more than the threshold limit of support, i.e., 3 in this case. Thus, frequent single itemsets, L1, will be as shown in Table 9.19.

Table 9.19 Frequent items L1

| Item | Frequency |
|------------|-----------|
| Bread | 4 |
| Cornflakes | 3 |
| Milk | 3 |
| Jam | 4 |

Then, the candidate 2-itemsets or C2 are determined per the process illustrated below:

$$C2 = L1 \text{ JOIN } L1.$$

The joining of L1 with itself has been illustrated below.

| | | |
|------------|------|------------|
| Bread | JOIN | Bread |
| Cornflakes | | Cornflakes |
| Milk | | Jam |
| Jam | | Milk |

In the Join operation the first step will be to perform a Cartesian product, i.e., to make all possible pairs between L1 and L1 as shown below.

Bread, Bread
Bread, Cornflakes
Bread, Milk
Bread, Jam
Cornflakes, Bread
Cornflakes, Cornflakes
Cornflakes, Milk
Cornflakes, Jam
Milk, Bread
Milk, Cornflakes
Milk, Milk
Milk, Jam
Jam, Bread
Jam, Cornflakes
Jam, Milk
Jam, Jam

Here, we have 16 possible pairs. In association mining, we are interested in item pairs, so the pairs having same itemsets (like Bread, Bread) need to be considered for removal from the list. Similarly, if we already have an item pair of Milk, Jam then there is no need to list item pair Jam, Milk so it will also be removed from the list.

Thus, to create C2 the following steps are applied:

- Perform Cartesian product of L1 with itself
- Some item pairs may have identical items. Keep just one. Remove the extras.
- Select only those item pairs in which items are in lexical order (so that if we have Milk, Jam then Jam, Milk should not appear).

Thus, the final C2 will be as shown below:

Bread, Cornflakes
Bread, Milk
Bread, Jam
Cornflakes, Milk
Cornflakes, Jam
Milk, Jam

The frequency of each of these item pairs will be found as shown in the table given below.

Table 9.20 Candidate item pairs C2

| <i>Item pairs</i> | <i>Frequency</i> |
|---------------------|------------------|
| (Bread, Cornflakes) | 2 |
| (Bread, Milk) | 2 |
| (Bread, Jam) | 3 |

Contd.

| <i>Item pairs</i> | <i>Frequency</i> |
|--------------------|------------------|
| (Cornflakes, Milk) | 1 |
| (Cornflakes, Jam) | 3 |
| (Milk, Jam) | 2 |

We therefore have only two frequent item pairs in L2 as shown in Table 9.21.

Table 9.21 Frequent two item pairs L2

| <i>Item pairs</i> | <i>Frequency</i> |
|-------------------|------------------|
| (Bread, Jam) | 3 |
| (Cornflakes, Jam) | 3 |

Generation of C3 from L2

C3 is generated from L2 by carrying out a JOIN operation over L2 as shown below.

$$C3 = L2 \text{ JOIN } L2$$

It will involve the same steps as performed for C2, but it has one important pre-requisite for Join, i.e., two items are joinable if their first item is common. In a generalized case:

$$Ck = Lk-1 \text{ JOIN } Lk-1$$

And they are joinable if their first k-2 items are the same. So, in case of C3, the first item should be the same in L2, while in case of C2 there is no requirement of first item similarity because k-2 in the C2 case is 0.

From {Bread, Jam} and {Cornflakes, Jam} two frequent 2-itemsets, we do not obtain a candidate 3-itemset since we do not have two 2-itemsets that have the same first item. This completes the first phase of the Apriori algorithm.

Phase 2: Generation of rules

The two frequent 2-itemsets given above lead to the following possible rules.

Bread → Jam

Jam → Bread

Cornflakes → Jam

Jam → Cornflakes

The confidence of these rules is obtained by dividing the support for both items in the rule by the support for the item on the left-hand side of the rule. The confidence of the four rules therefore are $3/4 = 75\%$, $3/4 = 75\%$, $3/3 = 100\%$, and $3/4 = 75\%$ respectively. Since all of them have a minimum 75% confidence, they all qualify.

In order to generalize the Apriori algorithm, let us define the representation of itemsets.

Representation of Itemsets

To generate C_k , let us first define L_{k-1} .

Let l_1 and l_2 be itemsets in list L_{k-1} and the notation $l_i[j]$ refers to the j th item in l_i .

To generate C_2 from L_1 , C_2 is represented as C_k and L_1 as L_{k-1} . Let us consider L_1 as shown in Table 9.22, the description of each item list has been given below.

Table 9.22 L_1 for generation of C_2 having only one element in each list

| L_1 | |
|-------|---|
| 1 | represented as item list l_1 and its item, i.e., 1 is represented as $l_1[1]$. |
| 2 | represented as item list l_2 and its item, i.e., 2 is represented as $l_2[1]$. |
| 3 | represented as item list l_3 and its item, i.e., 3 is represented as $l_3[1]$. |
| 5 | represented as item list l_4 and its item, i.e., 5 is represented as $l_4[1]$. |

Let us consider L_2 given in Table 9.23.

Table 9.23 L_2 for generation of C_3 (i.e., $K=3$) having two elements in each list

| L_2 | |
|-------|--|
| 1,2 | represented as item list l_1 and its item, 1 is represented as $l_1[1]$ or $l_1[k-2]$ and 2 is represented as $l_1[2]$ or $l_1[k-1]$ |
| 4,5 | represented as item list l_2 and its item, 4 is represented as $l_2[1]$ or $l_2[k-2]$ and 5 is represented as $l_2[2]$ or $l_2[k-1]$ |
| 2,6 | represented as item list l_3 and its item, 2 is represented as $l_3[1]$ or $l_3[k-2]$ and 6 is represented as $l_3[2]$ or $l_3[k-1]$ |

Let us consider L_3 given in Table 9.24.

Table 9.24 L_3 for generation of C_4 (i.e., $K = 4$) having three elements in each list

| L_3 | |
|-------|---|
| 1,2,3 | represented as item list l_1 and its item, 1 is represented as $l_1[1]$ or $l_1[k-3]$ and 2 is represented as $l_1[2]$ or $l_1[k-2]$ and 3 is represented as $l_1[3]$ or $l_1[k-1]$ |
| 1,2,5 | represented as item list l_2 and its item, 1 is represented as $l_2[1]$ or $l_2[k-3]$ and 2 is represented as $l_2[2]$ or $l_2[k-2]$ and 5 is represented as $l_2[3]$ or $l_2[k-1]$ |
| 2,3,6 | represented as item list l_3 and its item, 2 is represented as $l_3[1]$ or $l_3[k-3]$ and 3 is represented as $l_3[2]$ or $l_3[k-2]$ and 6 is represented as $l_3[3]$ or $l_3[k-1]$ |

Defining Phase 1: Identification of frequent itemsets

The Apriori algorithm uses an iterative approach and is also known as a level-wise search where k -itemsets are used to search $(k+1)$ itemsets. First, the database is scanned to find the set of frequent 1-itemsets to accumulate the count for each item denoted as C_1 , known as candidate 1-itemset, and gathering those items satisfying minimum required support denoted as L_1 , frequent 1-itemset.

Next, L1 is used to find C2, candidate 2-itemsets and collecting those items that satisfy minimum support denoted as L2, frequent 2-itemsets; which is used to find C3, candidate 3-itemsets; which is used further to identify L3, frequent 3-itemsets and so on, until no more frequent k-itemsets can be found. This process has been illustrated in Figure 9.7. The finding of each L_k requires one full scan of the database.

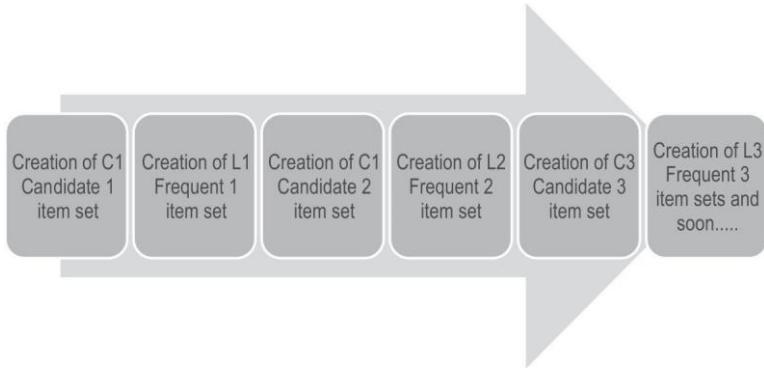


Figure 9.7 Process for identification of frequent itemsets

To proceed further, we have to generate C2 based on L1. To understand the generation of candidate itemsets, it is important to understand the representation of items in the item lists.

Step 3: Form candidate 2 itemset pair, i.e., C2.

C2 is created by joining L1 with itself by using following joining rule.

The join step

To find C_k, a set of candidate k-itemsets denoted as C_k is generated by joining L_{k-1} with itself. Let l₁ and l₂ be itemsets in L_{k-1}, the notation l_i[j] refers to the jth item in l_i, e.g., l₁[k-2] refers to the second last item in l₁. By convention, the prior assumption is that items within a transaction or itemset are sorted in lexicographic order. For the (k-1) itemset, l_i, this means that the items are sorted such that l_i[1] < l_i[2] < : : : < l_i[k-1].

The join, L_{k-1} on L_{k-1}, is performed, where members of L_{k-1} are joinable if their first (k-2) items are in common.

That is, members l₁ and l₂ of L_{k-1} are joined if (l₁[1] = l₂[1]) ^ (l₁[2] = l₂[2]) ^ : : : ^ (l₁[k-2] = l₂[k-2]) ^ (l₁[k-1] < l₂[k-1]). The condition l₁[k-1] < l₂[k-1] simply ensures that no duplicates are generated.

The resulting itemset formed by joining l₁ and l₂ is l₁[1], l₁[2], : : : , l₁[k-2], l₁[k-1], l₂[k-1].

Thus, there are three important points that make items of L_{k-1} joinable and these are as follows.

Point 1: The members of L_{k-1}, are joinable if their first (k-2) items are in common

It means for C2 = L1 JOIN L1, with k = 2, there is no requirement for having the first common data item, because k-2 = 0.

For C3 = L2 JOIN L2, with k = 3, the first k-2, i.e., first 1 item should be the same in the itemset.

For $C4 = L3 \text{ JOIN } L3$, having $k = 4$, the first $k-2$, i.e., first 2 items should be the same in the itemset.

For example, Let us verify whether $L1$ given in Table 9.25 is joinable or not, to create $C2$.

Table 9.25 L1

| L1 |
|----|
| 1 |
| 2 |
| 3 |

$C2 = L1 \text{ JOIN } L1$

*There is no requirement for first common item because $k-2 = 0$ in this case and all itemsets, i.e., 1, 2 and 3 are joinable.

Now, let us verify whether $L2$ given in Table 9.26 is joinable or not to create $C3$.

Table 9.26 L2

| L2 | |
|-----|---|
| 1,2 | represented as item list $l1$ and its item, i.e., 1 is represented as $l1[1]$ or $l1[k-2]$ and 2 is represented as $l1[2]$ or $l1[k-1]$ |
| 1,3 | represented as item list $l2$ and its item, i.e., 1 is represented as $l2[1]$ or $l2[k-2]$ and 3 is represented as $l2[2]$ or $l2[k-1]$ |
| 2,5 | represented as item list $l3$ and its item, 2 is represented as $l3[1]$ or $l3[k-2]$ and 5 is represented as $l3[2]$ or $l3[k-1]$ |

$C3 = L2 \text{ JOIN } L2$ [Here, $k = 3$]

Itemsets are joinable only if their first $k-2$ itemsets, i.e., first 1 item is common. Thus, itemsets 1, 2 and 1, 3 are joinable as their first element is 1 in both the itemsets.

Now, let us identify that $L3$ given in Table 9.27 is joinable or not to create $C4$.

Table 9.27 L3

| L3 | |
|-------|--|
| 1,2,3 | represented as item list $l1$ and its item, i.e., 1 is represented as $l1[1]$ or $l1[k-3]$ and 2 is represented as $l1[2]$ or $l1[k-2]$ and 3 is represented as $l1[3]$ or $l1[k-1]$ |
| 1,2,5 | represented as item list $l2$ and its item, i.e., 1 is represented as $l2[1]$ or $l2[k-3]$ and 2 is represented as $l2[2]$ or $l2[k-2]$ and 5 is represented as $l2[5]$ or $l2[k-1]$ |
| 1,3,5 | represented as item list $l3$ and its item, 1 is represented as $l3[1]$ or $l3[k-3]$ and 3 is represented as $l3[2]$ or $l3[k-2]$ and 5 is represented as $l3[3]$ or $l3[k-1]$ |

$C4 = L3 \text{ JOIN } L3$ [Here, $k = 4$]

Itemsets are joinable only if their first $k-2$ itemsets, i.e., first 2 items are common in this case. Thus, itemsets 1, 2, 3 and 1, 2, 5 are joinable as their first two elements, i.e., 1 and 2 are common in both the itemsets.

Point 2

In joinable item lists $l1(k-1)$ should be less than $l2(k-1)$, i.e., $l1(k-1) < l2(k-1)$: this rule ensures that items in the itemset should be in lexicographic order and there is no repetition.

To understand this condition, let us find C2 for given L1 in Table 9.28.

Table 9.28 L1

| L1 |
|----|
| 1 |
| 2 |
| 3 |
| 5 |

As discussed earlier, there is no requirement for the first common item, because $k-2 = 0$ in this case and all itemsets, i.e., 1, 2, 3 and 5 are joinable.

Thus, $C2 = L1 \text{ JOIN } L1$ (the base for join is the Cartesian product of two itemsets) as illustrated in Table 9.29.

Table 9.29 Generation of C2

| | | |
|---|------|---|
| 1 | JOIN | 1 |
| 2 | | 2 |
| 3 | | 3 |
| 5 | | 5 |

Possible pairs for 1 are 11*, 12, 13&15

*11 will not appear in the final list because, here $l1(1) = l2(1)$ and according to the joining rule $l1(1)$ should be less than $l2(1)$ [1 is not less than 1].

Thus, selected pairs for 1 are 12, 13 & 15

Possible pairs for 2 are 21*, 22*, 23 & 25

21 and 22 will not be selected because, $l1(1)$ is not less than $l2(1)$ [2 is not less than 1 and 2 is not less than 2].

Thus, selected pairs for 2 are 23 & 25

Possible pairs for 3 are 31*, 32*, 33* & 35

31, 32 and 33 will not be selected because , $l1(1)$ is not less than $l2(1)$ [3 is not less than 1; 3 is not less than 2; and 3 is not less than 3].

Thus, the selected pair for 3 is 35

Possible pairs for 5 are 51*, 52*, 53* & 55*

*No item will be selected because, l1(1) is not less than l2(1) [5 is not less than 1,2,3 and 5].

Thus, C2 will be as shown in Table 9.30.

Table 9.30 Generated C2

| C2 |
|----|
| 12 |
| 13 |
| 15 |
| 23 |
| 25 |
| 35 |

Point 3

The resulting itemset formed by joining $l1$ and $l2$ is $l1[1], l1[2], \dots, l1[k-2], l1[k-1], l2[k-1]$.

It means that, resulting itemset will contain all the items of the first item list, i.e., from item 1 to k-1 and last item of the second item list, i.e., k-1 item of second list, because other k-2 items of second list are already there, as they are common with first list.

To illustrate this point, let us identify C3 for given L2 in Table 9.31.

Table 9.31 L2

| L2 |
|-----|
| 1,2 |
| 1,3 |
| 2,3 |

Here, itemsets 1, 2 (first list) and 1, 3 (second list) are joinable. The final item list for C3 will be all items of first list and last item of second list.

Thus, C3 will be as shown in Table 9.32.

Table 9.32 C3

| C3 |
|---------|
| 1, 2, 3 |

Let us identify C4 for the given L3 in Table 9.33.

Table 9.33 L3

| L3 |
|---------|
| 1, 2, 4 |
| 1, 2, 5 |
| 1, 2, 7 |
| 1, 3, 6 |

Here, itemsets (1, 2, 4), (1, 2, 5) and (1, 2, 7) are joinable. The resultant C4 will be as shown in Table 9.34.

Table 9.34 C4

| C4 | |
|------------|---|
| 1, 2, 4, 5 | Considering (1, 2, 4) as first list and (1, 2, 5) second list |
| 1, 2, 4, 7 | Considering (1, 2, 4) as first list and (1, 2, 7) second list |
| 1, 2, 5, 7 | Considering (1, 2, 5) as first list and (1, 2, 7) second list |

Example

Let us understand the process for identification of frequent itemsets for the given transaction database of Table 9.35. The threshold value of support is 50% and confidence is 70%,

Table 9.35 Transaction database

| | |
|----|---------|
| T1 | 1,3,4 |
| T2 | 2,3,5 |
| T3 | 1,2,3,5 |
| T4 | 2,5 |

Step 1

Create C1 candidate 1-itemsets. It is all the items in the transaction database as shown in Figure 9.11. Figure 9.8 also contains the frequency of each item.

| C ₁ | Count |
|----------------|-------|
| 1 | 2 |
| 2 | 3 |
| 3 | 3 |
| 4 | 1 |
| 5 | 3 |

Figure 9.8 C1, candidate 1-itemset and their count

Step 2

Create frequent 1-itemset, i.e., list of 1-itemsets whose frequency is more than the threshold value of support, i.e., 2 as shown in Figure 9.9.

| C ₁ | Count | L ₁ |
|----------------|-------|----------------|
| 1 | 2 | 1 |
| 2 | 3 | 2 |
| 3 | 3 | 3 |
| 4 | 1 | 5 |
| 5 | 3 | |

Figure 9.9 L₁, frequent 1-itemset

Candidate two itemsets, C₂ with its frequency count as given in Table 9.36.

Table 9.36 Generation of C₂

| C ₂ | Count |
|----------------|-------|
| 1, 2 | 1 |
| 1, 3 | 2 |
| 1, 5 | 1 |
| 2, 3 | 2 |
| 2, 5 | 3 |
| 3, 5 | 2 |

L₂ is created by selecting those candidate pairs having support of 2 or more as shown in Table 9.37.

Table 9.37 Generation L₂

| L ₂ |
|----------------|
| 1, 3 |
| 2, 3 |
| 2, 5 |
| 3, 5 |

From the given L₂, the next step will be to generate C₃ by using L₂ JOIN L₂. Thus, C₃ will be as shown in Table 9.38.

Table 9.38 Generation of C₃

| C ₃ |
|----------------|
| 2, 3, 5 |

The frequency of candidate three itemset 2, 3, 5 is 2, so C3 is qualified as L3. Thus, frequent three itemset is 2, 3, 5 for given dataset.

This is the final frequent item list and it completes the first phase of the Aproori algorithm to find the frequent itemsets.

Phase 2: Generating association rules from frequent itemsets

To understand the process of generation of association rules from the frequent itemset, let us consider frequent itemset l .

- For each frequent itemset l generates all non-empty subsets of l .
- For every non-empty subset s of l , output the rule $s \rightarrow l-s$ and if the confidence of the rule is more than the threshold value of the confidence, then, this rule will be selected as the final association rule.

Let us generate the rules for the frequent itemset (2, 3, 5) represented as l . Here, non-empty subsets are $\{2\}, \{3\}, \{5\}, \{(2, 3)\}, \{(2, 5)\}, \{(3, 5)\}$.

For every non-empty subset, the rule will be generated as follows.

$2 \rightarrow 3, 5$ [Here, (2) is s and (3, 5) is $l-s$]

$3 \rightarrow 2, 5$

$5 \rightarrow 2, 3$

$2, 3 \rightarrow 5$

$2, 5 \rightarrow 3$

$3, 5 \rightarrow 2$

The next step will be to calculate the confidence for each rule as shown below.

$2 \rightarrow 3, 5$; Confidence = $S(2 \cap 3 \cap 5) / S(2) = 2/3 = 0.67$

$3 \rightarrow 2, 5$; Confidence = $S(2 \cap 3 \cap 5) / S(3) = 2/3 = 0.67$

$5 \rightarrow 2, 3$; Confidence = $S(2 \cap 3 \cap 5) / S(5) = 2/3 = 0.67$

$2, 3 \rightarrow 5$; Confidence = $S(2 \cap 3 \cap 5) / S(2 \cap 3) = 2/2 = 1.0$

$2, 5 \rightarrow 3$; Confidence = $S(2 \cap 3 \cap 5) / S(2 \cap 5) = 2/3 = 0.67$

$3, 5 \rightarrow 2$; Confidence = $S(2 \cap 3 \cap 5) / S(3 \cap 5) = 2/2 = 1.0$

Here, the minimum threshold for confidence is 70%, thus selected association rules are as follows.

$2, 3 \rightarrow 5$;

$3, 5 \rightarrow 2$;

The possible rules from $2, 3 \rightarrow 5$

It is intuitive to create two new rules as $2 \rightarrow 5$ and $3 \rightarrow 5$ from the given rule. Since $2, 3 \rightarrow 5$ has confidence more than the threshold limit, in this case, it is not implicit or guaranteed that $2 \rightarrow 3$ and $3 \rightarrow 5$ will have confidence more than the threshold limit as there is no correlation between denominator and quotient of both the rules as shown below.

Confidence of $2, 3 \rightarrow 5$ = $S(2 \cap 3 \cap 5) / S(2 \cap 3)$

Confidence of $2 \rightarrow 5$ = $S(2 \cap 5) / S(2)$

Same is the case for $3 \rightarrow 5$ as shown below.

Confidence of $3 \rightarrow 5$ = $S(3 \cap 5) / S(3)$

Thus, the given high confidence rule $2, 3 \rightarrow 5$ does not implicitly state that $2 \rightarrow 5$ and $3 \rightarrow 5$ will have its confidence more than the threshold limit. Thus, there is a need to calculate the confidence of these rules as shown in Table 9.39.

Similarly, the given high confidence rule $3, 5 \rightarrow 2$ does not implicitly state that $3 \rightarrow 2$ and $5 \rightarrow 2$ will have its confidence more than the threshold limit. Thus, again there is a need to calculate the confidence of these rules as shown in Table 9.39.

Table 9.39 Calculation of confidence

| Rule | Confidence |
|----------------------|-----------------------------------|
| $2, 3 \rightarrow 5$ | 1.0 |
| $2 \rightarrow 5$ | $S(2 \cap 5) / S(2) = 3/3 = 1.0$ |
| $3 \rightarrow 5$ | $S(2 \cap 5) / S(3) = 3/3 = 1.0$ |
| $3, 5 \rightarrow 2$ | 1.0 |
| $3 \rightarrow 2$ | $S(3 \cap 2) / S(3) = 2/3 = 0.67$ |
| $5 \rightarrow 2$ | $S(5 \cap 2) / S(5) = 3/3 = 1.0$ |

Thus, final rules having confidence more than the threshold limit, i.e., 75% are as follows.

| |
|--------------------------------|
| Selected Association rules are |
| $2, 3 \rightarrow 5$ |
| $2 \rightarrow 5$ |
| $3 \rightarrow 5$ |
| $3, 5 \rightarrow 2$ |
| $5 \rightarrow 2$ |

Apriori property

Apriori property states that all nonempty subsets of a frequent itemset must also be frequent.

It means all the subsets of the candidate itemset should be frequent otherwise that itemset will be removed from the candidate itemset. This step is called pruning of the candidate itemset and it will help to reduce the search space and to improve the performance of the algorithm because now the frequency of fewer item pairs need to be computed.

Lattice structure of frequent itemsets

The Apriori property that defines an itemset can be frequent only if all the subsets of the itemset are frequent. This can be illustrated by a lattice structure. Let us suppose there are four items A, B, C and D in the dataset. Then, the four-itemset ABCD will be frequent only if three itemsets

ABC, ABD, ACD and BCD are frequent. These three itemsets are frequent only if their subsets, i.e., AB, AC, BC, BD, AD and CD are frequent. And these two itemsets are frequent only if their subsets, i.e., A, B, C and D items are also frequent as shown in Figure 9.10.

From the lattice structure we can conclude that the Apriori algorithm works bottom up one level at a time, i.e., it first computes frequent 1-itemsets, the candidate 2-itemsets and then frequent 2-itemsets and so on. The number of scans of the transaction database is equal to the maximum number of items in the candidate itemsets.

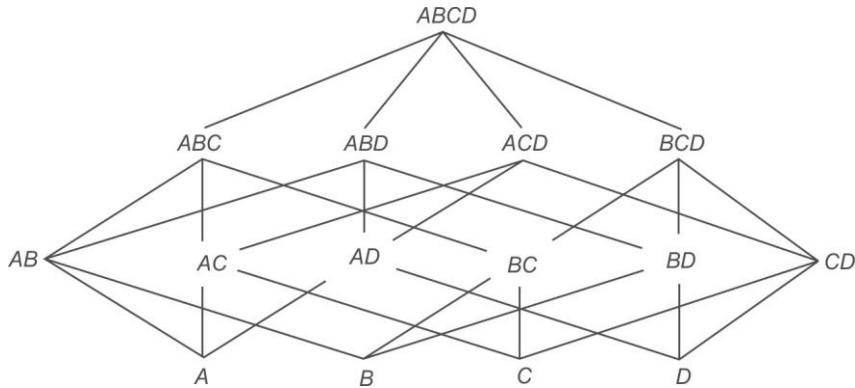


Figure 9.10 Lattice structure of frequent itemsets

For example, Apriori property states that for a frequent three itemset (1, 2, 3) all of its non empty subsets, i.e., (1, 2), (2, 3) and (1, 3) must be frequent, because, support of a superset is always less than or equal to support of its subset. It means that if (1, 3) is not frequent then (1, 2, 3) will also not be frequent.

This Apriori property also explains the significance of having first k-2 items common in a joining principle.

Let us suppose, we have L2 that is (1, 2) and (2, 3), then, one may suppose that it could produce a set (1,2,3) as C3. But as discussed earlier, (1, 2, 3) should be frequent only if all of its nonempty subsets, i.e., (1, 2), (2, 3) and (1, 3) are frequent. Here, as given in L2 (1, 3) is not frequent. So, itemsets given in L2 are considered as non joinable and the condition of having first k-2 items common is enforced to ensure it.

Now, what would happen if L2 has (1, 2) and (1, 3) but not (2, 3)? Then, according to the joining principle C3 will be generated as (1, 2, 3), but it will be discarded by the Apriori property because one of its nonempty subset, i.e., (2, 3) is not frequent. Thus, the combination of joining principle and apriori property make the whole process complete.

In conclusion, it is not the only joining principle that decides about the candidate itemsets, the joining principle can produce some extra items sets that can be removed further by the pruning or apriori property.

Example: Find association rules for the transaction data given in Table 9.40 for having support 15% and confidence 70%.

Table 9.40 Transaction database for identification of association rules

| TID | List of Items |
|-----|----------------|
| 10 | I1, I2, I5 |
| 20 | I2, I4 |
| 30 | I2, I3 |
| 40 | I1, I2, I4 |
| 50 | I1, I3 |
| 60 | I2, I3 |
| 70 | I1, I3 |
| 80 | I1, I2, I3, I5 |
| 90 | I1, I2, I4 |

The frequency count for each item, C1 is given in Table 9.41.

Table 9.41 C1

| C1 | Count |
|----|-------|
| I1 | 6 |
| I2 | 7 |
| I3 | 5 |
| I4 | 3 |
| I5 | 2 |

All the items have frequency more than the specified support limit, thus all 1-item candidate sets given as C1 qualify as 1-item frequent itemset L1. The next step will be to generate C2.

$C_2 = L_1 \text{ JOIN } L_1$

The process of creation of C2 and L2 is illustrated in Figure 9.11.

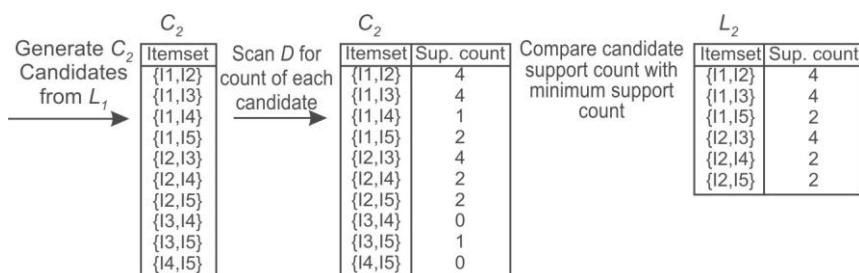


Figure 9.11 Generation of C2 and L2

From the given L2, C3 is generated by considering those item pairs whose first k-2 items, i.e. first 1 item is common as shown in Table 9.42.

Table 9.42 Generation of C3

| C3 |
|------------|
| I1, I2, I3 |
| I1, I2, I5 |
| I1, I3, I5 |
| I2, I3, I4 |
| I2, I3, I5 |
| I2, I4, I5 |

Before finding the frequency of each candidate item pair, the Apriori property will be applied to prune the candidate list.

Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent? Let us check that as shown in Table 9.43.

Table 9.43 Pruning of candidate itemset C3

| | | |
|------------|--|------------------|
| I1, I2, I3 | The 2-item subsets of I1, I2 and I3 are (I1, I2), (I1, I3), and (I2, I3). All 2-item subsets are members of L2. Therefore, keep I1, I2 and I3 in C3. | Qualify |
| I1, I2, I5 | The 2-item subsets of I1, I2 and I5 are (I1, I2), (I1, I5), and (I2, I5). All 2-item subsets are members of L2. Therefore, keep I1, I2 and I5 in C3. | Qualify |
| I1, I3, I5 | The 2-item subsets of I1, I3 and I5 are (I1, I3), (I1, I5), and (I3, I5). Since, (I3, I5) is not a member of L2, and so it is not frequent. Therefore, remove I1, I3 and I5 from C3. | Does not qualify |
| I2, I3, I4 | The 2-item subsets of I2, I3 and I4 are (I2, I3), (I2, I4), and (I3, I4). Since, (I3, I4) is not a member of L2, and so it is not frequent. Therefore, remove I2, I3 and I4 from C3. | Does not qualify |
| I2, I3, I5 | The 2-item subsets of I2, I3 and I5 are (I2, I3), (I2, I5), and (I3, I5). Since, (I3, I5) is not a member of L2, so it is not frequent. Therefore, remove I2, I3 and I5 from C3. | Does not qualify |
| I2, I4, I5 | The 2-item subsets of I2, I4 and I5 are (I2, I4), (I2, I5), and (I4, I5). Since, (I4, I5) is not a member of L2, it is not frequent. Therefore, remove I2, I4 and I5 from C3. | Does not qualify |

Thus pruned C3 will be as shown in Table 9.44.

Table 9.44 Pruned C3

| Pruned C3 |
|------------|
| I1, I2, I3 |
| I1, I2, I5 |

It is important to note that pruning results into improving the efficiency of the algorithm, as in this case instead of finding the count of six three item pairs, there is a need to find the count for only 2 three item pairs (pruned list) which is a significant improvement in the efficiency of the system.

The process of creation of L3 from C3 by identifying those 3 itemsets that has a frequency more than or equal to the given value of threshold value of support, i.e., 2 has been shown below in Figure 9.12.

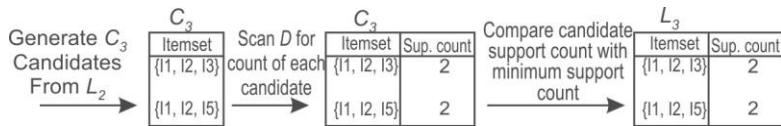


Figure 9.12 Generation of C3 and L3

The next step will be to generate C4 as shown below.

$$C4 = L3 \text{ JOIN } L3$$

C4 will be I1, I2, I3 & I5, because 2 items are common as shown in Table 9.45.

Table 9.45 C4

| C4 |
|----------------|
| I1, I2, I3, I5 |

But this itemset is pruned by the Apriori property because its subset (I2, I3, I5) is not frequent as it is not present in L3. Thus, C4 is null and the algorithm terminates at this point, having found all of frequent itemsets as shown in Table 9.46.

Table 9.46 Pruned C4

| C4 |
|------|
| NULL |

In this case, instead of finding the count of 1 four-item pair, pruning indicates that there is no need to find its count as all its subsets are not frequent.

With this, the first phase of the Apriori algorithm has been completed. Now the next phase to generate association rules from frequent itemsets will be performed.

Phase 2: Generating association rules from frequent itemsets

Let us apply this rule to frequent 3-itemsets (I1, I2, I3) and (I1, I2, I5) found in case of example given in Table 9.40.

For first frequent itemset (I1, I2, I3), non-empty subsets are $\{\{I1\}, \{I2\}, \{I3\}, \{(I1, I2)\}, \{(I1, I3)\}, \{(I2, I3)\}\}$.

For every non-empty set, the rule will be generated as follows:

I1→I2, I3 [Here, (I1) is s and I2 and I3 are l-s]

I2→I1, I3

I3→I1, I2

I1, I2→I3

I1, I3→I2

I2, I3→I1

The next will be to calculate the confidence for each rule as shown below.

I1→I2, I3; Confidence = $S(I1 \cap I2 \cap I3) / S(I1) = 2/6 = 0.3$

I2→I1, I3; Confidence = $S(I1 \cap I2 \cap I3) / S(I2) = 2/7 = 0.28$

I3→I1, I2; Confidence = $S(I1 \cap I2 \cap I3) / S(I3) = 2/5 = 0.4$

I1, I2→I3; Confidence = $S(I1 \cap I2 \cap I3) / S(I1 \cap I2) = 2/4 = 0.5$

I1, I3→I2; Confidence = $S(I1 \cap I2 \cap I3) / S(I1 \cap I3) = 2/4 = 0.5$

I2, I3→I1; Confidence = $S(I1 \cap I2 \cap I3) / S(I2 \cap I3) = 2/4 = 0.5$

Since, the minimum threshold is 70%, there are no rules that qualify from the frequent itemset {1, 2, 3}.

Now, let us apply this rule to second frequent 3-itemset (I1, I2, I5). For this frequent itemset, non-empty subsets are $\{I1\}, \{I2\}, \{I5\}, \{(I1, I2)\}, \{(I1, I5)\}, \{(I2, I5)\}$.

For every non-empty set the rule will be generated as follows:

I1→I2, I5 [Here, (I1) is s and I2 and I5 are l-s]

I2→I1, I5

I5→I1, I2

I1, I2→I5

I1, I5→I2

I2, I5→I1

The next step will be to calculate the confidence for each rule as shown below.

I1→I2, I5; Confidence = $S(I1 \cap I2 \cap I5) / S(I1) = 2/6 = 0.3$

I2→I1, I5; Confidence = $S(I1 \cap I2 \cap I5) / S(I2) = 2/7 = 0.28$

I5→I1, I2; Confidence = $S(I1 \cap I2 \cap I5) / S(I5) = 2/2 = 1$

I1, I2→I5; Confidence = $S(I1 \cap I2 \cap I5) / S(I1 \cap I2) = 2/4 = 0.5$

I1, I5→I2; Confidence = $S(I1 \cap I2 \cap I5) / S(I1 \cap I5) = 2/2 = 1$

I2, I5→I1; Confidence = $S(I1 \cap I2 \cap I5) / S(I2 \cap I5) = 2/2 = 1$

Now, there are three rules whose confidence is more than minimum threshold value of 70%, and these rules are as follows:

I5→I1, I2

I1, I5→I2

I2, I5→I1

The possible rules from $I_5 \rightarrow I_1, I_2$

It is intuitive to create two new rules as $I_5 \rightarrow I_1$ and $I_5 \rightarrow I_2$ from the given rule.

Since, $I_5 \rightarrow I_1, I_2$ has confidence more than threshold limit, so it is implicit that $I_5 \rightarrow I_1$ and $I_5 \rightarrow I_2$ will also have their confidence more than the threshold limit as justified below.

$$\text{Confidence of } I_5 \rightarrow I_1, I_2 = S(I_5 \cap I_1 \cap I_2) / S(I_5)$$

$$\text{Confidence of } I_5 \rightarrow I_1 = S(I_5 \cap I_1) / S(I_5)$$

Since, Support of subset (I_5, I_1) will be more than or equal to Support of superset (I_5, I_1, I_2), it is implicit that Confidence of $I_5 \rightarrow I_1$ will be more than equal to Confidence of $I_5 \rightarrow I_1, I_2$.

Similarly, Confidence of $I_5 \rightarrow I_2 = S(I_5 \cap I_2) / S(I_5)$ will be more than equal to Confidence of $I_5 \rightarrow I_1, I_2$.

Thus, $I_5 \rightarrow I_1$ and $I_5 \rightarrow I_2$ will also have their confidence more than the threshold limit.

Discussing possible rules from $I_2, I_5 \rightarrow I_1$

It is intuitive to create two new rules as $I_2 \rightarrow I_1$ and $I_5 \rightarrow I_1$ from the given rule. $I_2, I_5 \rightarrow I_1$ has confidence more than the threshold limit, but it is not implicit or guaranteed that $I_2 \rightarrow I_1$ and $I_5 \rightarrow I_1$ will have their confidence more than the threshold limit as explained below.

$$\text{Confidence of } I_2, I_5 \rightarrow I_1 = S(I_2 \cap I_5 \cap I_1) / S(I_2 \cap I_5)$$

$$\text{Confidence of } I_2 \rightarrow I_1 = S(I_2 \cap I_1) / S(I_2)$$

Here, there is no correlation between denominator and quotient of both the rules, so it is not implicit that this rule will also have its confidence more than the threshold.

$$\text{Similarly, the Confidence of } I_5 \rightarrow I_1 = S(I_5 \cap I_1) / S(I_5)$$

Again, there is no correlation between denominator and quotient of this rule with $I_2, I_5 \rightarrow I_1$ rules, so it is not implicit that this rule will also have its confidence more than the threshold.

The given high confidence rule $I_2, I_5 \rightarrow I_1$ does not implicitly state that $I_2 \rightarrow I_1$ and $I_5 \rightarrow I_1$ will have their confidence more than the threshold limit. Thus, there is a need to calculate the confidence of these rules.

Similarly, the given high confidence rule $I_1, I_5 \rightarrow I_2$ does not implicitly state that $I_1 \rightarrow I_2$ and $I_5 \rightarrow I_2$ will have their confidence more than the threshold limit. Thus, again there is a need to calculate the confidence of these rules.

The process of generation of all association rules from $I_5 \rightarrow I_1, I_2; I_2, I_5 \rightarrow I_1$ and $I_1, I_5 \rightarrow I_2$ rules has been illustrated in Table 9.47.

Table 9.47 Generation of association rules

| Association Rule | Discussion | Confidence | More than threshold limit Or Not |
|----------------------------|--------------------|--|----------------------------------|
| $I_5 \rightarrow I_1, I_2$ | Already identified | 1.0 | Yes |
| $I_5 \rightarrow I_1$ | Implicit | No need to calculate it will be more than or equal to $I_5 \rightarrow I_1, I_2$ | Yes |
| $I_5 \rightarrow I_2$ | Implicit | No need to calculate it will be more than or equal to $I_5 \rightarrow I_1, I_2$ | Yes |

Contd.

| Association Rule | Discussion | Confidence | More than threshold limit Or Not |
|------------------|---|---|----------------------------------|
| I2, I5→I1 | Already identified | 1.0 | Yes |
| I2→I1 | Not found earlier, confidence need to be calculated | Confidence of I2→I1 $= S(I2 \cap I1) / S(I1) = 4/6 = 67\%$ | No |
| I5→I1 | Already found from I5→I1, I2 given in row 2 | No need to calculate it will be more than or equal to I5→I1, I2 | Yes (Already listed) |
| I1, I5→I2 | Already identified | 1.0 | Yes |
| I1→I2 | Not found earlier, so confidence needs to be calculated | Confidence of I1→I2 $= S(I2 \cap I1) / S(I2) = 4/7 = 57\%$ | No |
| I5→I2 | Already found from I5→I1, I2 given in row 3 | | Yes (Already listed) |

During generation of the final rules, it is important to look at L2 because there may be some frequent 2-itemsets that have not appeared in three itemsets and may be left out. The frequent 2 itemsets generated earlier have been reproduced in Table 9.48 from Figure 9.10.

Table 9.48 Frequent 2-itemsets, i.e., L2

| L2 | Discussion | New Rules Identified |
|--------|---|---|
| I1, I2 | This item pair has already been covered as it is a sub set of I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules discussed in Table 9.47. | NIL |
| I1, I3 | This item pair has not already been covered. So, confidence needs to be calculated for possible rules of this item pair I1, I3. I1→I3, Confidence = $S(I1 \cap I3) / S(I1) = 4/6 = 66\%$ I3→I1, Confidence = $S(I3 \cap I1) / S(I3) = 4/5 = 80\%$ | Since, I3→I1 has confidence more than the threshold limit. So, the new rule identified is I3→I1. |
| I1, I5 | This item pair has already been covered as it is a sub set of I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules. | NIL |
| I2, I3 | This item pair has not already been covered. So, confidence needs to be calculated for possible rules of this item pair I2, I3. I2→I3, Confidence = $S(I2 \cap I3) / S(I2) = 4/7 = 57\%$ I3→I2, Confidence = $S(I2 \cap I3) / S(I3) = 4/5 = 80\%$ | I3→I2 |
| I2, I4 | This item pair has not already been covered. So, confidence needs to be calculated for possible rules of this item pair I2, I4. I2→I4, Confidence = $S(I2 \cap I4) / S(I2) = 2/7 = 28\%$ I4→I2, Confidence = $S(I4 \cap I2) / S(I4) = 2/3 = 66\%$ | NIL, because the confidence of both the rules is less than threshold limit; so, no new rule has been found. |
| I2, I5 | This item pair has already been covered as it is a sub set of I5→I1, I2; I2, I5→I1 and I1, I5→I2 rules. | NIL |

Here, we have identified two new rules by considering L2 and these are $I3 \rightarrow I1$ and $I3 \rightarrow I2$, both of which has its confidence more than threshold limit of 70%.

The final association rules having support more than 20% and confidence more than 70% are given below.

$I5 \rightarrow I1, I2$

$I2, I5 \rightarrow I1$

$I1, I5 \rightarrow I2$

$I5 \rightarrow I1$

$I5 \rightarrow I2$

$I3 \rightarrow I1$

$I3 \rightarrow I2$.

Example: Consider a store having 16 items for sale as listed in Table 9.49. Now consider the 25 transactions on the sale of these items as given in Table 9.50. As usual, each row in the table represents one transaction, that is, the items bought by one customer. In this example, we want to find association rules that satisfy the requirement of 25% support and 70% confidence.

Table 9.49 List of grocery items

| Item number | Item name |
|-------------|------------|
| 1 | Biscuit |
| 2 | Bournvita |
| 3 | Bread |
| 4 | Butter |
| 5 | Coffee |
| 6 | Cornflakes |
| 7 | Chocolate |
| 8 | Curd |
| 9 | Eggs |
| 10 | Jam |
| 11 | Juice |
| 12 | Milk |
| 13 | Rice |
| 14 | Soap |
| 15 | Sugar |
| 16 | Tape |

In Table 9.50 showing 25 transactions, we list the names of items of interest but it would be more storage efficient to use item numbers.

Table 9.50 Transaction data

| <i>TID</i> | <i>Items</i> |
|------------|--|
| 1 | Biscuit, Bournvita, Butter, Cornflakes, Tape |
| 2 | Bournvita, Bread, Butter, Cornflakes |
| 3 | Butter, Coffee, Chocolate, Eggs, Jam |
| 4 | Bournvita, Butter, Cornflakes, Bread, Eggs |
| 5 | Bournvita, Bread, Coffee, Chocolate, Eggs |
| 6 | Jam, Sugar |
| 7 | Biscuit, Bournvita, Butter, Cornflakes, Jam |
| 8 | Curd, Jam, Sugar |
| 9 | Bournvita, Bread, Butter, Coffee, Cornflakes |
| 10 | Bournvita, Bread, Coffee, Chocolate, Eggs |
| 11 | Bournvita, Butter, Eggs |
| 12 | Bournvita, Butter, Cornflakes, Chocolate, Eggs |
| 13 | Biscuit, Bournvita, Bread |
| 14 | Bread, Butter, Coffee, Chocolate, Eggs |
| 15 | Coffee, Cornflakes |
| 16 | Chocolate |
| 17 | Chocolate, Curd, Eggs |
| 18 | Biscuit, Bournvita, Butter, Cornflakes |
| 19 | Bournvita, Bread, Coffee, Chocolate, Eggs |
| 20 | Butter, Coffee, Chocolate, Eggs |
| 21 | Jam, Sugar, Tape |
| 22 | Bournvita, Bread, Butter, Cornflakes |
| 23 | Coffee, Chocolate, Eggs, Jam, Juice |
| 24 | Juice, Milk, Rice |
| 25 | Rice, Soap, Sugar |

Computing C1

All 16 items are candidate items for a frequent single-itemset. So, all the items listed in Table 9.49 will act as C1.

Computing L1

To find the frequent single itemset, we have to scan the whole database to count the number of times each of the 16 items has been sold. Scanning transaction database only once, we first set up the list of items and one transaction at a time, update the count of every item that appears in that transaction as shown in Table 9.51.

Table 9.51 Frequency count for all items

| <i>Item no.</i> | <i>Item name</i> | <i>Frequency</i> |
|-----------------|------------------|------------------|
| 1 | Biscuit | 4 |
| 2 | Bournvita | 13 |
| 3 | Bread | 9 |
| 4 | Butter | 12 |
| 5 | Coffee | 9 |
| 6 | Cornflakes | 9 |
| 7 | Chocolate | 10 |
| 8 | Curd | 2 |
| 9 | Eggs | 11 |
| 10 | Jam | 6 |
| 11 | Juice | 2 |
| 12 | Milk | 1 |
| 13 | Rice | 2 |
| 14 | Soap | 1 |
| 15 | Sugar | 4 |
| 16 | Tape | 2 |

The items that have the necessary support (25% support in 25 transactions) must occur in at least 7 transactions. The frequent 1-itemset or L1 is now given in Table 9.52.

Table 9.52 The frequent 1-itemset or L1

| <i>Item</i> | <i>Frequency</i> |
|-------------|------------------|
| Bournvita | 13 |
| Bread | 9 |
| Butter | 12 |

Contd.

| <i>Item</i> | <i>Frequency</i> |
|-------------|------------------|
| Coffee | 9 |
| Cornflakes | 9 |
| Chocolate | 10 |
| Eggs | 11 |

Computing C2

C2 = L1 JOIN L1

This will produce 21 pairs as listed in Table 9.53. These 21 pairs are the only ones worth investigating since they are the only ones that could be frequent.

Table 9.53 The 21 candidate 2-itemsets or C2

| |
|-------------------------|
| {Bournvita, Bread} |
| {Bournvita, Butter} |
| {Bournvita, Coffee} |
| {Bournvita, Cornflakes} |
| {Bournvita, Chocolate} |
| {Bournvita, Eggs} |
| {Bread, Butter} |
| {Bread, Coffee} |
| {Bread, Cornflakes} |
| {Bread, Chocolate} |
| {Bread, Eggs} |
| {Butter, Coffee} |
| {Butter, Cornflakes} |
| {Butter, Chocolate} |
| {Butter, Eggs} |
| {Coffee, Cornflakes} |
| {Coffee, Chocolate} |
| {Coffee, Eggs} |
| {Cornflakes, Chocolate} |
| {Cornflakes, Eggs} |
| {Chocolate, Eggs} |

Computing L2

This step is the most resource intensive step. We scan the transaction database once and look at each transaction and find out which of the candidate pairs occur in that transaction. This then requires scanning the list C2 for every pair of items that appears in the transaction that is being scanned. Table 9.54 presents the number of times each candidate 2-itemset appears in the transaction database example given in Table 9.50.

Table 9.54 Frequency count of candidate 2-itemsets

| Candidate 2-itemset | Frequency |
|-------------------------|-----------|
| {Bournvita, Bread} | 8 |
| {Bournvita, Butter} | 9 |
| {Bournvita, Coffee} | 4 |
| {Bournvita, Cornflakes} | 8 |
| {Bournvita, Chocolate} | 4 |
| {Bournvita, Eggs} | 6 |
| {Bread, Butter} | 5 |
| {Bread, Coffee} | 5 |
| {Bread, Cornflakes} | 3 |
| {Bread, Chocolate} | 4 |
| {Bread, Eggs} | 5 |
| {Butter, Coffee} | 4 |
| {Butter, Cornflakes} | 8 |
| {Butter, Chocolate} | 4 |
| {Butter, Eggs} | 6 |
| {Coffee, Cornflakes} | 2 |
| {Coffee, Chocolate} | 7 |
| {Coffee, Eggs} | 7 |
| {Cornflakes, Chocolate} | 1 |
| {Cornflakes, Eggs} | 2 |
| {Chocolate, Eggs} | 9 |

From this list we can find the frequent 2-itemsets or the set L2 by selecting only those item pairs from Table 9.54 given above, having a frequency equal to or more than 7. Therefore the list of frequent-2 itemsets or L2 is given in Table 9.55. Usually L2 is smaller than L1 but in this example they both have seven itemsets and each member of L1 appears as a subset of some member of L2. This is unlikely to be true when the number of items is large.

Table 9.55 The frequent 2-itemsets or L2

| Frequent 2-itemset | Frequency |
|-------------------------|-----------|
| {Bournvita, Bread} | 8 |
| {Bournvita, Butter} | 9 |
| {Bournvita, Cornflakes} | 8 |
| {Butter, Cornflakes} | 8 |
| {Coffee, Chocolate} | 7 |
| {Coffee, Eggs} | 7 |
| {Chocolate, Eggs} | 9 |

Computing C3

C3 = L2 JOIN L2 (First item should be common)

To find the candidate 3-itemsets or C3, we combine the appropriate frequent 2-itemsets from L2 (these must have the same first item) and obtain four such itemsets as given in Table 9.56.

Table 9.56 Candidate 3-itemsets or C3

| Candidate 3-itemset |
|---------------------------------|
| {Bournvita, Bread, Butter} |
| {Bournvita, Bread, Cornflakes} |
| {Bournvita, Butter, Cornflakes} |
| {Coffee, Chocolate, Eggs} |

Pruning C3

Before processing further to calculate the frequency of each 3-item pairs, it is important to apply the Apriori property to prune the candidate 3-itemsets as shown in Table 9.57.

Table 9.57 Pruning of candidate itemset C3

| Candidate Item Pair | Discussion | Qualify or Not |
|---------------------------------|---|----------------------------------|
| {Bournvita, Bread, Butter} | Its subsets are (Bournvita, Bread), (Bournvita, Butter) and (Bread, Butter). Out of these subsets (Bread, Butter) is not frequent as it is not in L2 given in Table 9.55. Thus, {Bournvita, Bread, Butter} will be removed from C3. | Not Qualified, so will be pruned |
| {Bournvita, Bread, Cornflakes} | Its subsets are (Bournvita, Bread), (Bournvita, Cornflakes) and (Bread, Cornflakes). Out of these subsets, (Bread, Cornflakes) is not frequent as it is not in L2. Thus, {Bournvita, Bread, Cornflakes} will be removed from C3. | Not Qualified, so will be pruned |
| {Bournvita, Butter, Cornflakes} | Its subsets are (Bournvita, Butter), (Bournvita, Cornflakes) and (Butter, Cornflakes). Here all subsets are frequent as they are in L2. Thus, {Bournvita, Butter, Cornflakes} will be retained in C3. | Qualified |
| {Coffee, Chocolate, Eggs} | Its subsets are (Coffee, Chocolate), (Coffee, Eggs) and (Chocolate, Eggs). Here, all subsets are frequent as they are in L2. Thus, {Coffee, Chocolate, Eggs} will be retained in C3. | Qualified |

Thus, pruned C3 will be as shown in Table 9.58.

Table 9.58 Pruned candidate itemset C3

{Bournvita, Butter, Cornflakes}

{Coffee, Chocolate, Eggs}

As observed, this pruning step has reduced the candidate items from 4 to 2 and it improves the performance of the algorithm. Table 9.59 shows candidate 3-itemsets or C3 and their frequencies.

Table 9.59 Candidate 3-itemsets or C3 and their frequencies

| Candidate 3-itemset | Frequency |
|---------------------------------|-----------|
| {Bournvita, Butter, Cornflakes} | 8 |
| {Coffee, Chocolate, Eggs} | 7 |

Computing L3

Table 9.60 now presents the frequent 3-itemsets or L3. Note that only one member of L2 is not a subset of any itemset in L3. That 2-itemset is {Bournvita, Bread}.

Table 9.60 The frequent 3-itemsets or L3

| Frequent 3-itemset | Frequency |
|---------------------------------|-----------|
| {Bournvita, Butter, Cornflakes} | 8 |
| {Coffee, Chocolate, Eggs} | 7 |

Computing C4

C4 = L3 JOIN L3 (First two items should be common)

Since, no first two items are common, thus, C4 is null.

This is the end of frequent itemset generation since there cannot be any candidate 4-itemsets.

Finding the rules

The rules obtained from frequent 3-itemsets would always include some rules that we could obtain from L2 embedded in them, but some additional rules will be obtainable from itemsets in L2 which are not subsets of itemsets in L3, for example, {Bournvita, Bread} in this case.

Here, the frequent itemset is {Bournvita, Butter, Cornflakes} whose resulting subsets are as followed:

{Bournvita}, {Butter}, {Cornflakes}, {Bournvita, Butter}, {Butter, Cornflakes}, {Bournvita, Cornflakes} [represented by s , and the superset or whole list {Bournvita, Butter, Cornflakes} is represented as l , and the rules will be generated in the form of $s \rightarrow l-s$].

The rules generated from {Bournvita, Butter, Cornflakes} are as follows.

Bournvita \rightarrow Butter, Cornflakes

Butter \rightarrow Bournvita, Cornflakes

Cornflakes \rightarrow Bournvita, Butter

Bournvita, Butter \rightarrow Cornflakes

Bournvita, Cornflakes \rightarrow Butter

Butter, Cornflakes \rightarrow Bournvita

Now, let us calculate the confidence of these rules as shown in Table 9.61.

Table 9.61 Confidence of association rules from {Bournvita, Butter, Cornflakes}

| Rule | Support of Both together | Frequency of Antecedent LHS (Refer to Table 9.52 and 9.55) | Confidence | Whether Confidence is more than threshold limit or not |
|--|--------------------------|--|-------------|--|
| Bournvita \rightarrow Butter, Cornflakes | 8 | 13 | 8/13 = 0.61 | No |
| Butter \rightarrow Bournvita, Cornflakes | 8 | 12 | 8/12 = 0.67 | No |
| Cornflakes \rightarrow Bournvita, Butter | 8 | 9 | 8/9 = 0.89 | Yes |
| Bournvita, Butter \rightarrow Cornflakes | 8 | 9 | 8/9 = 0.89 | Yes |
| Bournvita, Cornflakes \rightarrow Butter | 8 | 8 | 8/8 = 1.0 | Yes |
| Butter, Cornflakes \rightarrow Bournvita | 8 | 8 | 8/8 = 1.0 | Yes |

We also have two rules from L2 for a frequent item pair {Bournvita, Bread} let us calculate its confidence as shown in Table 9.62.

Table 9.62 Confidence of association rules from {Bournvita, Bread}

| Rule | Support of Both together | Frequency of Antecedent LHS (Refer to Tables 9.52 and 9.55) | Confidence | Whether Confidence is more than threshold limit or not |
|-------------------|--------------------------|---|-------------|--|
| Bournvita → Bread | 8 | 13 | 8/13 = 0.61 | No |
| Bread → Bournvita | 8 | 9 | 8/9 = 0.89 | Yes |

Thus, from this discussion we have found out five rules having confidence more than 70% as shown below in Table 9.63.

Table 9.63 Identified rules from {Bournvita, Butter, Cornflakes} having confidence more than 70%

| |
|--------------------------------|
| Cornflakes → Bournvita, Butter |
| Bournvita, Butter → Cornflakes |
| Bournvita, Cornflakes → Butter |
| Butter, Cornflakes → Bournvita |
| Bread → Bournvita |

Now, let us find all possible rules from these identified rules as shown in Table 9.64.

Table 9.64 List of all possible rules from rules given in Table 9.61

| Rules | Discussion | Rules having confidence more than the threshold value |
|--------------------------------|---|---|
| Cornflakes → Bournvita, Butter | It has confidence 0.89 as discussed in Table 9.61. [Formula = S (Cornflakes, Bournvita, Butter) / S(Cornflakes)] | Cornflakes → Bournvita, Butter |
| Cornflakes → Bournvita | It is intuitive that this rule will have confidence greater than the threshold limit as its formula is S (Cornflakes, Bournvita) / S(Cornflakes). Since, support of subset, i.e, Cornflakes, Bournvita, should be equal to or more than superset, Cornflakes, Bournvita, Butter so this rule qualifies. (Resulting confidence = 0.89) | Cornflakes → Bournvita |

Contd.

| Rules | Discussion | Rules having confidence more than the threshold value |
|--------------------------------|---|---|
| Cornflakes → Butter | <p>It is also intuitive that this rule will have confidence greater than the threshold limit as its formula is $S(\text{Cornflakes}, \text{Butter}) / S(\text{Cornflakes})$. Since, support of subset, i.e., Cornflakes, Butter should be equal to or more than superset, Cornflakes, Bournvita, Butter so this rule qualifies. (Resulting confidence = 0.89)</p> | Cornflakes → Butter |
| Bournvita, Butter → Cornflakes | <p>It has confidence 0.89 as shown in Table 9.61. [Formula = $S(\text{Bournvita}, \text{Butter}, \text{Cornflakes}) / S(\text{Bournvita}, \text{Butter})$]</p> | Bournvita, Butter → Cornflakes |
| Bournvita → Cornflakes | <p>It is not intuitive that this rule will have confidence more than the threshold limit as its formula is $S(\text{Bournvita}, \text{Cornflakes}) / S(\text{Bournvita})$.</p> <p>So, let us calculate its confidence. Confidence = $S(\text{Bournvita}, \text{Cornflakes}) / S(\text{Bournvita}) = 8/13 = 0.61$. This rule has confidence less than threshold limit, so it will be discarded.</p> | Discarded |
| Butter → Cornflakes | <p>It is not intuitive that this rule will have confidence more than the threshold limit as its formula is $S(\text{Butter}, \text{Cornflakes}) / S(\text{Butter})$.</p> <p>So, let us calculate its confidence.</p> <p>Confidence = $S(\text{Butter}, \text{Cornflakes}) / S(\text{Butter}) = 8/12 = 0.67$</p> <p>This rule has confidence less than threshold limit, so it will be discarded.</p> | Discarded |
| Bournvita, Cornflakes → Butter | <p>It has confidence 1.0 as shown in Table 9.61.</p> | Bournvita, Cornflakes → Butter |
| Bournvita → Butter | <p>It is not intuitive that this rule will have confidence more than the threshold limit.</p> <p>So, let us calculate its confidence.</p> <p>Confidence = $S(\text{Bournvita}, \text{Butter}) / S(\text{Bournvita}) = 9/13 = 0.69$</p> <p>This rule has confidence less than threshold limit, so it will be discarded.</p> | Discarded |
| Cornflakes → Butter | <p>It is not intuitive that this rule will have confidence more than the threshold limit.</p> <p>So, let us calculate its confidence.</p> <p>Confidence = $S(\text{Cornflakes}, \text{Butter}) / S(\text{Cornflakes}) = 8/9 = 0.88$</p> <p>This rule has confidence more than threshold limit, so it will be selected.</p> | Cornflakes → Butter |

Contd.

| <i>Rules</i> | <i>Discussion</i> | <i>Rules having confidence more than the threshold value</i> |
|-------------------------------------|---|--|
| Butter, Cornflakes→ Bournvita | It has confidence 1.0 as shown in Table 9.61. | Butter, Cornflakes→ Bournvita |
| Butter→ Bournvita | <p>It is not intuitive that this rule will have confidence more than the threshold limit. So, let us calculate its confidence. Confidence = $S(\text{Butter, Bournvita}) / S(\text{Butter}) = 8/12 = 0.67$ This rule has confidence less than threshold limit, so it will be discarded.</p> | Discarded |
| Cornflakes→ Bournvita | <p>It is not intuitive that this rule will have confidence more than the threshold limit. So, let us calculate its confidence. Confidence = $S(\text{Cornflakes, Bournvita}) / S(\text{Cornflakes}) = 8/9 = 0.89$ This rule has confidence more than threshold limit, so it will be selected.</p> | Cornflakes→ Bournvita |

Similarly, we will examine the other 3-itemset {Coffee, Chocolate, Eggs}.

Table 9.65 Confidence of association rules from {Coffee, Chocolate, Eggs}

| <i>Rule</i> | <i>Support</i> | <i>Frequency of LHS (Antecedent)</i> | <i>Confidence</i> |
|------------------------|----------------|--------------------------------------|-------------------|
| Coffee→Chocolate, Eggs | 7 | 9 | 0.78 |
| Chocolate→Coffee, Eggs | 7 | 10 | 0.70 |
| Eggs→Coffee, Chocolate | 7 | 11 | 0.64 |
| Chocolate, Eggs→Coffee | 7 | 9 | 0.78 |
| Coffee, Eggs→Chocolate | 7 | 7 | 1.0 |
| Coffee, Chocolate→Eggs | 7 | 7 | 1.0 |

Again, the confidence of all these rules, except the third, is higher than 70%, and so five of them also qualify.

List of all possible rules for three item pairs {Coffee, Chocolate, Eggs} has been explained in Table 9.66, given below.

Table 9.66 List of all possible rules from rules given in Table 9.65

| Rules | Discussion | Rules having confidence more than the threshold value |
|------------------------|--|---|
| Coffee→Chocolate, Eggs | It has confidence of 0.78 as shown in Table 9.65. | Coffee→Chocolate, Eggs |
| Coffee→Eggs | It is intuitive from rule Coffee→Chocolate, Eggs | Coffee→Eggs |
| Chocolate→Coffee, Eggs | It has confidence of 0.70 as shown in Table 9.65. | Chocolate→Coffee, Eggs |
| Chocolate→Coffee | It is intuitive from rule Chocolate→Coffee, Eggs | Chocolate→Coffee |
| Chocolate→Eggs | It is intuitive from rule Chocolate→Coffee, Eggs | Chocolate→Eggs |
| Chocolate, Eggs→Coffee | It has confidence of 0.78 as shown in Table 9.65. | Chocolate, Eggs→Coffee |
| Chocolate→Coffee | It is non intuitive, so let us calculate its confidence. Confidence = $S(\text{Chocolate}, \text{Coffee}) / S(\text{Coffee}) = 7/9 = 0.77$ Since, confidence is more than the threshold so this rule will be selected. | Chocolate→Coffee |
| Eggs→Coffee | It is non intuitive, so let us calculate its confidence. Confidence = $S(\text{Eggs}, \text{Coffee}) / S(\text{Eggs}) = 7/11 = 0.64$ Since, confidence is less than the threshold so this rule will be discarded. | Discarded |
| Coffee, Eggs→Chocolate | It has confidence 1.0 as discussed in Table 9.65. | Coffee, Eggs→Chocolate |
| Coffee→Chocolate | It is non intuitive, so let us calculate its confidence. Confidence = $S(\text{Coffee}, \text{Chocolate}) / S(\text{Coffee}) = 7/9 = 0.77$ Since, confidence is more than the threshold so this rule will be selected. | Coffee→Chocolate |
| Eggs→Chocolate | It is non intuitive, so let us calculate its confidence. Confidence = $S(\text{Eggs}, \text{Chocolate}) / S(\text{Eggs}) = 9/11 = 0.81$ Since, confidence is more than the threshold so this rule will be selected. | Eggs→Chocolate |

Contd.

| <i>Rules</i> | <i>Discussion</i> | <i>Rules having confidence more than the threshold value</i> |
|------------------------|---|--|
| Coffee, Chocolate→Eggs | It has confidence 1.0 as shown in Table 9.65. | Coffee, Chocolate→Eggs |
| Coffee→Eggs | This rule is non intuitive, but it is already selected as given in row 2 from rule Coffee→Chocolate, Eggs | Coffee→Eggs |
| Chocolate→Eggs | This rule is non intuitive, but it is already selected as given in row 5 from rule Chocolate→Coffee, Eggs | Chocolate→Eggs |

Thus, all association mining rules for the given database and having confidence of more than 70% are listed in Table 9.67.

Table 9.67 All association rules for the given database

| <i>Rule No.</i> | <i>Rule</i> | <i>Confidence</i> |
|-----------------|-------------------------------|-------------------|
| 1 | Bread→Bournvita | 0.89 |
| 2 | Cornflakes→ Bournvita, Butter | 0.89 |
| 3 | Cornflakes→ Bournvita | 0.89 |
| 4 | Cornflakes→ Butter | 0.89 |
| 5 | Bournvita, Butter→ Cornflakes | 0.89 |
| 6 | Bournvita, Cornflakes→ Butter | 1 |
| 7 | Cornflakes→ Butter | 0.89 |
| 8 | Butter, Cornflakes→ Bournvita | 1 |
| 9 | Coffee→Chocolate, Eggs | 0.78 |
| 10 | Coffee→Chocolate | 0.78 |
| 11 | Coffee→Eggs | 0.78 |
| 12 | Chocolate→Coffee, Eggs | 0.7 |
| 13 | Chocolate→Coffee | 0.7 |
| 14 | Chocolate→Eggs | 0.9 |
| 15 | Chocolate, Eggs→Coffee | 1 |
| 16 | Coffee, Eggs→Chocolate | 1 |
| 17 | Eggs→Chocolate | 0.82 |
| 18 | Coffee, Chocolate→Eggs | 1 |

9.8 Closed and Maximal Itemsets

To further improve the efficiency of the association mining process, frequent itemsets can be divided into closed and maximal itemsets.

A frequent closed itemset is a frequent itemset X such that there exists no superset of X with the same support count as X.

A frequent itemset Y is maximal if it is not a proper subset of any other frequent itemset. Therefore a maximal itemset is a closed itemset, but a closed itemset is not necessarily a maximal itemset as shown in Figure 9.13.

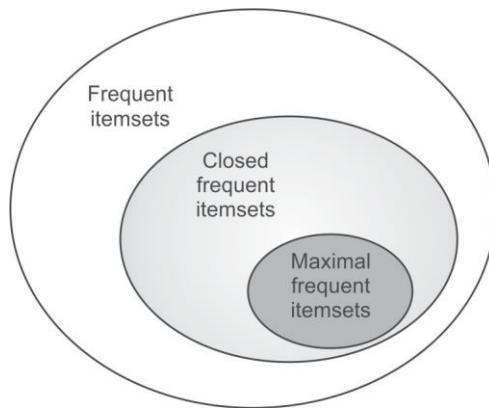


Figure 9.13 Illustration of closed and maximal frequent itemsets

The concept of Closed and maximal itemsets have been explained with an example database.

Example: Consider the transaction database in Table 9.68 with minimum support required to be 40%.

Table 9.68 A transaction database to illustrate closed and maximal itemsets

| | |
|-----|---------------------------|
| 100 | Butter, Curd, Jam |
| 200 | Butter, Curd, Jam, Muffin |
| 300 | Curd, Jam, Nuts |
| 400 | Butter, Jam, Muffin, Nuts |
| 500 | Muffin, Nuts |

There are a total of 14 frequent itemsets as shown in Table 9.69 and the closed and maximal itemsets for the given database have also been presented there.

Table 9.69 Frequent itemsets for the database in Table 9.68

| Itemset | Support | Closed? | Maximal? | Both? |
|------------------|---------|--|--|-------|
| {Butter} | 3 | No, there exists a superset of Butter with the same support 3, i.e., {Butter, Jam} | No, it is a proper subset of another frequent itemset, i.e., {Butter, Curd, Jam} | No |
| {Curd} | 3 | No, there exists a superset of Curd with the same support 3, {Curd, Jam} | No, it is a proper subset of another frequent itemset, i.e., {Butter, Curd, Jam} | No |
| {Jam} | 4 | Yes, there exists no superset of Jam with the same support 4 | No, it is a proper subset of another frequent itemset, i.e., {Butter, Curd, Jam} | No |
| {Muffin} | 3 | Yes, there exists no superset of Muffin with the same support 3 | No, it is a proper subset of another frequent itemset, i.e., {Muffin, Nuts} | No |
| {Nuts} | 3 | Yes, there exists no superset of Nuts with the same support 3 | No, it is a proper subset of another frequent itemset, i.e., {Muffin, Nuts} | No |
| {Butter, Curd} | 2 | No, there exists a superset of Butter, Jam with the same support 2, i.e., {Butter, Curd, Jam} | No, it is a proper subset of another frequent itemset, i.e., {Butter, Curd, Jam} | No |
| {Butter, Jam} | 3 | Yes, there exists no superset of Butter, Jam with the same support 3 | No, it is a proper subset of another frequent itemset, i.e., {Butter, Curd, Jam} | No |
| {Butter, Muffin} | 2 | No, there exists a superset of Butter, Muffin with the same support 2, i.e., {Butter, Jam, Muffin} | No, it is a proper subset of another frequent itemset, i.e., {Butter, Jam, Muffin} | No |
| {Curd, Jam} | 3 | Yes, there exists no superset of Curd, Jam with the same support 3 | No, it is a proper subset of another frequent itemset, i.e., {Butter, Curd, Jam} | No |
| {Jam, Muffin} | 2 | No, there exists a superset of Jam, Muffin with the same support 2, i.e., {Butter, Jam, Muffin} | No, it is a proper subset of another frequent itemset, i.e., {Butter, Jam, Muffin} | No |
| {Jam, Nuts} | 2 | Yes, there exists no superset of Jam, Nuts with the same support 2 | Yes, it is not proper subset of any other frequent itemset | Yes |

Contd.

| Itemset | Support | Closed? | Maximal? | Both? |
|-----------------------|---------|--|--|-------|
| {Muffin, Nuts} | 2 | Yes, there exists no superset of Muffin, Nuts with the same support 2 | Yes, it is not proper subset of any other frequent itemset | Yes |
| {Butter, Curd, Jam} | 2 | Yes, there exists no superset of Butter, Curd, Jam with the same support 2 | Yes, it is not proper subset of any other frequent itemset | Yes |
| {Butter, Jam, Muffin} | 2 | Yes, there exists no superset of Butter, Jam, Muffin with the same support 2 | Yes, it is not proper subset of any other frequent itemset | Yes |

One can observe that there are 9 closed frequent itemsets and 4 maximal frequent itemsets as shown in Table 9.69.

It is important to note that closed and maximal frequent itemsets are often smaller than all frequent itemsets and association rules can be generated directly from them. Thus, it is desirable that instead of mining every frequent itemset, an efficient algorithm should mine only maximal frequent itemsets. The association of frequent itemsets, closed frequent itemsets and maximal frequent itemsets has been illustrated in Figure 9.16 for better understanding. From this one can observe that a maximal itemset is a closed itemset, but a closed itemset is not necessarily a maximal itemset.

9.9 The Apriori-TID Algorithm for Generating Association Mining Rules

Apriori-TID uses the Apriori algorithm to generate the candidate and frequent itemsets, but the difference is that it uses the TID or vertical storage approach (discussed in Section 9.6.3) for its implementation.

The working of the Apriori-TID algorithm is given as follows.

1. Scan the whole transaction database to convert it into TID storage and represent it as T1 (i.e., each entry of T1 consists of all items in the transaction along with the corresponding TID).
2. Calculate the frequent 1-itemset L1 using T1.
3. Obtain C2 by applying the joining principle discussed under the Apriori algorithm.
4. Calculate the support for the candidates in C2 with the help of T1 by intersecting corresponding rows and a new TID dataset is created and represent it as T2 containing two item pairs.
5. Generate L2 from C2 by the usual means and then generate C3 from L2, again by using the joining principle.
6. Generate T3 using T2 and C3. This process is repeated until the set of candidate k-itemsets is an empty set.

Example of Apriori-TID: Let us understand this concept with an example database of transactions given in Table 9.70 for finding frequent itemsets with the Apriori-TID algorithm. Here, minimum support is 50% and minimum confidence is 75%.

Table 9.70 Transaction database

| | |
|-----|-------------------------|
| 100 | Butter, Curd, Eggs, Jam |
| 200 | Butter, Curd, Jam |
| 300 | Butter, Muffin, Nuts |
| 400 | Butter, Jam, Muffin |
| 500 | Curd, Jam, Muffin |

Step 1:

Scan the entire transaction database to convert it into TID storage represented as T1 as given in Table 9.71.

Table 9.71 Transaction database T1

| | 100 | 200 | 300 | 400 | 500 |
|--------|-----|-----|-----|-----|-----|
| Butter | 1 | 1 | 1 | 1 | 0 |
| Curd | 1 | 1 | 0 | 0 | 1 |
| Eggs | 1 | 0 | 0 | 0 | 0 |
| Jam | 1 | 1 | 0 | 1 | 1 |
| Muffin | 0 | 0 | 1 | 1 | 1 |
| Nut | 0 | 0 | 1 | 0 | 0 |

Step 2:

Frequent 1-itemset L1 is calculated using T1.

Since, the minimum support is 3, all items have been selected as frequent items having support equal or more than threshold limit of support, i.e., 3 with the help of T1.

Thus, L1 is as shown in Table 9.72.

Table 9.72 L1

| Itemset | Support |
|---------|---------|
| Butter | 4 |
| Curd | 3 |
| Jam | 4 |
| Muffin | 3 |

Step 3:

C2 is obtained by applying the joining principle, i.e., L1 JOIN L1

Thus, C2 will be as given in Table 9.73.

Table 9.73 C2

| Itemsets |
|----------------|
| Butter, Curd |
| Butter, Jam |
| Butter, Muffin |
| Curd, Jam |
| Curd, Muffin |
| Jam, Muffin |

Step 4:

Support for the candidates in C2 is then calculated with the help of T1 by intersecting corresponding rows and a new TID dataset is created represented as T2 containing two item pairs as illustrated in Table 9.74.

Table 9.74 Transaction database T2

| | 100 | 200 | 300 | 400 | 500 |
|----------------|-----|-----|-----|-----|-----|
| Butter, Curd | 1 | 1 | 0 | 0 | 0 |
| Butter, Jam | 1 | 1 | 0 | 1 | 0 |
| Butter, Muffin | 0 | 0 | 1 | 1 | 0 |
| Curd, Jam | 1 | 1 | 0 | 0 | 1 |
| Curd, Muffin | 0 | 0 | 0 | 0 | 1 |
| Jam, Muffin | 0 | 0 | 0 | 1 | 1 |

Thus, support for C2 is given in Table 9.75.

Table 9.75 Support for C2

| Itemsets | Support |
|----------------|---------|
| Butter, Curd | 2 |
| Butter, Jam | 3 |
| Butter, Muffin | 2 |
| Curd, Jam | 3 |
| Curd, Muffin | 1 |
| Jam, Muffin | 2 |

Step 5:

L2 is then generated from C2 as shown in Table 9.76.

Table 9.76 L2

| Itemsets | Support |
|-------------|---------|
| Butter, Jam | 3 |
| Curd, Jam | 3 |

Thus, {Butter, Jam} and {Curd, Jam} are the frequent pairs and they generate L2. C3 may also be generated but it is observed that C3 is empty because the first item is not common in the L2 item pairs. If C3 was not empty then it would be used to identify T3 using the transaction set T2. That would result in a smaller T3 and it might have resulted in the removal of a transaction or two. In the example database, all the records of T2 have been removed and T3 is null.

The generation of association rules from the derived frequent set can be done in the usual way.

Advantages of the Apriori-TID algorithm

The advantage of Apriori-TID algorithm is in the storage structure because it facilitates counting of items by counting the number of 1s in each row and the number of 2-itemsets can be counted by finding the intersection of the two rows.

The other main advantage of the Apriori-TID algorithm is that the size of T_k is generally smaller than the transaction database.

Because the support for each candidate k-itemset is counted using the corresponding T_k , therefore this algorithm runs faster than the basic Apriori algorithm. It is important to note that both Apriori and Apriori-TID use the same candidate generation algorithm, and therefore they count the same itemsets.

9.10 Direct Hashing and Pruning (DHP)

A major shortcoming of the Apriori algorithm is that it cannot prune 2 itemset candidate items. As C2 is derived from L1 by using following formula:

$$C_2 = L_1 \text{ Join } L_1$$

Since, every subset of C2 is a member of L1, C2 cannot be pruned by the Apriori property, i.e., subset of frequent itemsets should also be frequent. In case of C2, all the subsets are frequent, i.e., member of L1, so it cannot be pruned. However, Apriori property is very helpful in pruning higher order candidate itemsets like C3 or C4.

The DHP algorithm is useful as it overcomes this limitation since it has the capability to prune C2; this is important to improve performance.

Working of the DHP algorithm

This algorithm employs a hash-based technique for reducing the count of candidate itemsets generated in the first pass (i.e., a significantly smaller C2 is generated). The number of itemsets in C2 generated using DHP can be smaller in magnitude; therefore the scan to identify L2 is more efficient.

The steps to be taken to operate DHP are:

1. Find all frequent 1-itemsets and all the candidate 2-itemsets by using the same steps as the Apriori algorithm, i.e., C2 is generated by L1 JOIN L1.
2. Generate all possible 2-itemsets in each transaction and assign a code to each item and itemsets.
3. All the possible 2-itemsets are hashed to a hash table by using the code of each itemset. For hashing, a hash function is applied on each code and a bucket is assigned to each itemset based on the output of the hash function. The count of each bucket in the hash table is increased by one, each time when an itemset is hashed to that bucket. When different itemsets are hashed to the same bucket then collisions can occur. To assign a flag for each bucket, a bit vector is associated with the hash table. If the bucket count is equal or above the minimum required support count, then accordingly flag in the bit vector is set to 1, otherwise it is set to 0.
4. In this step C2 is pruned. Each item pair of C2 generated in step 1 is checked for the bit vector of its corresponding bucket to which it was assigned in step 3. If its bit vector is 0, then the item pair will be pruned and removed from C2, while all those item pairs having a bit vector of 1 are retained in C2. This step helps to prune C2 and this pruning of C2 is the major objective of the DHP algorithm.

It is important to note that, having the corresponding bit vector or bit set does not guarantee that the itemset is frequent; this is due to collisions. The hash table filtering does reduce C2, significantly.

5. Then, from the pruned C2, frequent item pairs, L2 is generated and process continues as in Aproiroi property.
6. In this step, the hash table for the next step is generated. Only those itemsets that are frequent, i.e. in L2, are kept in the database. The pruning not only trims each transaction by removing the unwanted itemsets but also removes transactions that have no itemsets that could be frequent.
7. The algorithm is continued till no more candidate itemsets are identified. The whole transaction database is scanned to identify the support count for each itemset despite using a hash table to identify the frequent itemsets. Now, the database seems relatively smaller because of the pruning. The algorithm identifies the frequent itemsets as earlier by checking against the minimum support when the support count is calculated. The algorithm then produces candidate itemsets similar to what the Apriori algorithm does.

To get a clear idea about the working of DHP algorithm, let us consider some case studies by way of examples. After going through these examples, this algorithm will become clear to you.

Example: Using the DHP algorithm

To understand this concept, let us identify the association rules satisfying 50% support and 75% confidence for the transaction database given in Table 9.77 with the DHP algorithm.

Table 9.77 Transaction database

| | |
|-----|-------------------------|
| 100 | Butter, Curd, Eggs, Jam |
| 200 | Butter, Curd, Jam |
| 300 | Butter, Muffin, Nuts |
| 400 | Butter, Jam, Muffin |
| 500 | Curd, Jam, Muffin |

Let us first find all frequent 1-itemsets and in this case it will all those items that appear 3 or more times in the database. The L1 is given in Table 9.78.

Table 9.78 Frequent 1-itemset L1

| |
|--------|
| Butter |
| Curd |
| Jam |
| Muffin |

Then, generate C2 from L1, i.e., L1 JOIN L1. This is given in Table 9.78. Here, for simplicity we are representing each item by its first alphabet.

Table 9.79 Candidate 2 itemsets C2

| |
|--------------------------------------|
| BC ('B' for Butter and 'C' for Curd) |
| BJ |
| BM |
| CJ |
| CM |
| JM |

Since, the size of C2 is 6 and the major advantage of DHP is to prune C2 so hashing is applied on each item pair as discussed below.

To apply hashing, the next step is to generate all possible 2-itemsets for each transaction as shown in Table 9.80.

Table 9.80 Possible 2-itemsets for each transaction

| | |
|-----|---|
| 100 | (Butter, Curd), (Butter, Eggs), (Butter, Jam), (Curd, Eggs), (Curd, Jam), (Eggs, Jam) |
| 200 | (Butter, Curd), (Butter, Jam), (Curd, Jam) |
| 300 | (Butter, Muffin), (Butter, Nuts), (Muffin, Nuts) |
| 400 | (Butter, Jam), (Butter, Muffin), (Jam, Muffin) |
| 500 | (Curd, Jam), (Curd, Muffin), (Jam, Muffin) |

For a support of 50%, the frequent items are Butter, Curd, Jam, and Muffin as shown in Table 9.81.

Now, to hash the possible 2-itemsets, a code is assigned to each item as given in Table 9.82.

Table 9.81 Frequent itemsets for a support of 50%

| |
|--------|
| Bread |
| Curd |
| Jam |
| Muffin |

Table 9.82 Code for each item

| Code | Item Name |
|------|-----------|
| 1 | Butter |
| 2 | Curd |
| 3 | Eggs |
| 4 | Jam |
| 5 | Muffin |
| 6 | Nuts |

For each pair, a numeric value is obtained by representing Butter by 1, Curd by 2 and so on as shown in the above table. Then, each pair is represented by a two-digit number, for example, (Butter, Curd) is represented by 12; (Curd, Jam) is represented by 24 and so on.

The coded value for each item pair is given in Table 9.83.

Table 9.83 Coded representation for each item pair

| | |
|-----|------------------------|
| 100 | 12, 13, 14, 23, 24, 34 |
| 200 | 12, 14, 24 |
| 300 | 15, 16, 56 |
| 400 | 14, 15, 45 |
| 500 | 24, 25, 45 |

Now, the hash function is applied on coded values so that these items can be assigned to different buckets. Here, we are using modulo 8 number, i.e., dividing the code by 8 and using the remainder as hash function for the given dataset.

Hash function: Modulo 8, i.e., divide the concerned item pair code by 8 and use the remainder as its bucket number.

The guidelines to decide about hash function have been discussed later. For the moment, let's apply hash function to each item pair and assign the item pairs to each bucket on the basis of modulo 8 as shown in Table 9.60.

Applying the hash function to each item pair in transaction 100, the first item pair is 12, and when modulo 8 hash function is applied on 12, it gives a remainder of 4, so 12 is assigned to bucket 4. The next item pair in transaction 100 is 13, and gives a remainder of 5 after dividing it by 8, so 13 is assigned to bucket 5. Similarly, 14 has a remainder of 6, 23 has a remainder of 7, 24 has the remainder 0 and 34 has a remainder of 2, so these item pairs are assigned corresponding bucket numbers as shown in Table 9.84. This process is repeated for each transaction and every item pair is assigned to its corresponding bucket.

Table 9.84 Assigning item pairs to buckets based on hash function modulo 8

| Bucket address | | | | | | | | |
|----------------|----|----|---|----|----|----|----|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 24 | 25 | 34 | | 12 | 13 | 14 | 23 | Item pairs |
| 24 | | | | 12 | 45 | 14 | 15 | |
| 16 | | | | 45 | 14 | | 15 | |
| 56 | | | | | | | | |
| 24 | | | | | | | | |
| 5 | 1 | 1 | 0 | 2 | 3 | 3 | 3 | Support of bucket |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Bit Vector (it refers to a sequence of numbers which are either 0 or 1 and stored contiguously in memory) |

After assigning every item pair to its corresponding bucket, the support or count for each bucket has been calculated and if its support is equal to the threshold value then the corresponding bit vector is set 1 as shown in Table 9.84.

Now, C2 is pruned by checking the bit vector of the corresponding bucket for each candidate item pair. And only those item pairs are retained in C2 which have the corresponding bit vector as 1. The process of pruning of C2 has been illustrated in Table 9.85.

It has been noted that pair BC, i.e., 12 belong to the bucket with support of 2, since its support is less than threshold value of support so its corresponding bit vector is 0 and this item pair has been removed from C2; in other words, C2 has been pruned.

Similarly, this analysis is performed for each item pair of C2 as shown in Table 9.85.

Table 9.85 Pruning of C2

| Item Pairs in C2 | Bucket | Bit Vector | Remarks | Pruned C2 |
|------------------|--------|------------|------------------|-----------|
| BC, i.e., 12 | 4 | 0 | Removed from C2 | |
| BJ, i.e., 14 | 6 | 1 | Will be retained | BJ |
| BM, i.e., 15 | 7 | 1 | Will be retained | BM |
| CJ, i.e., 24 | 0 | 1 | Will be retained | CJ |
| CM, i.e., 25 | 1 | 0 | Removed from C2 | |
| JM, i.e., 45 | 5 | 1 | Will be retained | JM |

These candidate pairs are then hashed to the hash table and those candidate pairs are removed that are hashed to locations where the bit vector bit is not set. Table 9.85 illustrates that (B, C) and (C, M) have been eliminated from C2. Now, only four candidate item pairs are left instead of six item pairs in C2 and thus, C2 has been pruned as given in the last column of Table 9.85. After that, analyze the transaction database and modify it to include only these candidate pairs as given in Table 9.86.

Now, from the pruned C2, let us find L2 as shown in Table 9.86.

Table 9.86 Finding L2

| Itemset | Count |
|---------|-------|
| BJ | 3 |
| BM | 2 |
| CJ | 3 |
| JM | 2 |

Thus, L2 has two item pairs, i.e., BJ and CJ. Now, let us use L2 by selecting only those item pairs of the transaction database that qualify by being above the threshold value of support. The method of finding three itemsets is shown in Table 9.87.

Table 9.87 Finding three itemsets

| TID | Original Item groups | Selected Item pairs according to L2 | Three Item groupings (sets) |
|-----|---|-------------------------------------|-----------------------------|
| 100 | (Butter, Curd), (Butter, Eggs), (Butter, Jam), (Curd, Eggs), (Curd, Jam), (Eggs, Jam) | (Butter, Jam), (Curd, Jam) | NIL |
| 200 | (Butter, Curd), (Butter, Jam), (Curd, Jam) | (Butter, Jam), (Curd, Jam) | NIL |
| 300 | (Butter, Muffin), (Butter, Nuts), (Muffin, Nuts) | NIL | NIL |
| 400 | (Butter, Jam), (Butter, Muffin), (Jam, Muffin) | (Butter, Jam) | NIL |
| 500 | (Curd, Jam), (Curd, Muffin), (Jam, Muffin) | (Curd, Jam) | NIL |

As shown above in Table 9.87, there are no 3-Item groups, and this complete the process of identifying frequent item pairs. So, in this case, there are two item pairs (Butter, Jam) and (Curd, Jam). We will find out the rules for selected item pairs in the same way as we did in the Apriori algorithm, previously. Proceed as follow:

These item pairs produce four rules as shown below.

Butter → Jam

Jam → Butter

Curd → Jam

Jam → Curd

Now, let us calculate their confidence as shown below.

Confidence of Butter → Jam = $S(\text{Butter, Jam}) / S(\text{Butter}) = 3/4 = 0.75$

Jam → Butter = $S(\text{Jam, Butter}) / S(\text{Jam}) = 3/4 = 0.75$

Curd → Jam = $S(\text{Curd, Jam}) / S(\text{Curd}) = 3/3 = 1.0$

Jam → Curd = $S(\text{Jam, Curd}) / S(\text{Jam}) = 3/4 = 1.0$

Based on given threshold value of confidence we can select the qualified rules.

It is important to note that, the DHP algorithm usually performs better than the Apriori algorithm during the initial stages, especially during L2 generation. Added to this improvement is also the feature that trims and prunes the transaction set progressively, thereby improving the overall efficiency of the algorithm. The startup phase of DHP does consume some extra resources when the hash table is constructed but this is outweighed by the advantages gained during the later phases when both the number of candidate itemsets and the transaction set are reduced (pruned better).

Guidelines for selection of the hash function

As one of the major objectives of the DHP algorithm is to reduce the size of C2, it is essential that the hash table is large enough so that the collisions are low, i.e., different item pairs should not be hashed into the same bucket. In case of collisions, the bit vector is not very effective because in this case more than one item pair is contributing to support for the bucket and in ideal case it should be the support of only one item pair that should contribute, so that candidate item pairs are close to frequent item pairs.

If the hash table is too small and the collisions are high then more than one itemset will hash to most of the buckets. It will result into loss of effectiveness of the hash table in reducing the number of candidate itemsets C2. This is what happened in the example above in which we had collisions in three of the eight buckets. If the minimum support count is low, it is even more important that all collisions should be avoided.

If the hash table is too large then calculation will increase as we have to calculate the values for each bucket. Thus, the hash function should be selected in such a manner that it results in the least collisions.

Increasing the efficiency of the DHP algorithm

For efficient working of DHP, it is best to ensure that the hash table resides in the main memory. If the hash table does not fit in the main memory, additional disk I/O costs will apply which may again reduce the efficiency. This algorithm is most efficient when the number of itemsets in L2 is lot smaller than C2. This reduction in database size will have a positive impact on the efficiency of the algorithm.

The efficient pruning of the database, particularly C2, helps DHP to achieve a significantly shorter execution time than the Apriori algorithm.

Some more examples of DHP and Apriori algorithms

Another example will help in comparing Apriori with the DHP algorithm. In this example, we will first apply the Apriori algorithm to identify frequent item pairs; then we will apply the DHP algorithm for the same identification to illustrate differences in the process. Here, we have to find frequent item pairs with support more than 40% for the database given in Table 9.88.

By applying both the algorithms on the same database, comparison of the workings of both algorithms will be easy.

Table 9.88 Transaction database for Apriori and DHP

| TID | Items |
|-----|---------|
| 100 | A C D |
| 200 | B C E |
| 300 | A B C E |
| 400 | B E |

The process to identify frequent item pairs through the Apriori algorithm has been illustrated in Figure 9.14.

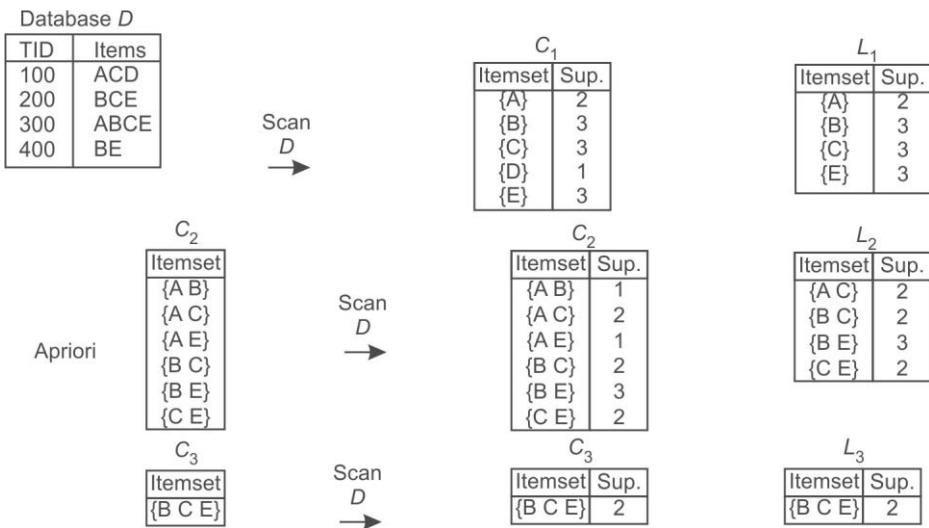


Figure 9.14 Process by the Apriori algorithm method

The process to identify frequent item pairs through DHP algorithm has been illustrated in Figure 9.15.

Database D

| TID | Items | On the fly | C_1 | count | L_1 |
|-----|-------|------------|-------|-------|-------|
| 100 | ACD | | {A} | 2 | {A} |
| 200 | BCE | | {B} | 3 | {B} |
| 300 | ABCE | | {C} | 3 | {C} |
| 400 | BE | | {D} | 1 | {E} |
| | | | {E} | 3 | |

minimum support, $s = 2$

Making a hash table

| | |
|-----|--|
| 100 | {A C}, {A D}, {C D} |
| 200 | {B C}, {B E}, {C E} |
| 300 | {A B}, {A C}, {A E}, {B C}, {B E}, {C E} |
| 400 | {B E} |

$h\{x y\} = ((\text{order of } x) * 10) + (\text{order of } y) \bmod 7;$

Figure 9.15 Process by the DHP algorithm method

Now, to hash the possible 2-itemsets, the code is assigned to each item. Let us assign code to each item as given in Table 9.89.

Table 9.89 Code for each item

| Item | Code |
|------|------|
| A | 1 |
| B | 2 |
| C | 3 |
| D | 4 |
| E | 5 |
| F | 6 |

A numeric value is attained for each pair by representing A by 1, B by 2 and so on as given in above table. Then each pair is represented by a two-digit number, for example, AC by 13 and AD by 14. The coded value for each item pair is given in Table 9.90.

Table 9.90 Coded representation for each item pair

| TID | Item pairs code |
|-----|--|
| 100 | 13 (AC), 14(AD), 34 (CD) |
| 200 | 23 (BC), 25 (BE), 35 (CE) |
| 300 | 12 (AB), 13 (AC), 15 (AE), 23 (BC), 25 (BE), 35 (CE) |
| 400 | 25 (BE) |

Now, hash function is applied on the coded values so that these items can be assigned to different buckets. Here, we are using modulo 7 number, i.e., dividing by 7 and using the remainder as hash function for the given dataset as given in Table 9.91.

Table 9.91 Assigning of item pairs to buckets based on hash function modulo 7

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | Bucket address |
|----|----|----|---|----|----|----|-------------------|
| 14 | 15 | 23 | | 25 | 12 | 13 | Item pairs |
| 35 | | 23 | | 25 | | 34 | |
| 35 | | | | 25 | | 13 | |
| 3 | 1 | 2 | 0 | 3 | 1 | 3 | Support of bucket |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | Bit Vector |

The process of pruning C2 based on bit vector has been shown in Table 9.92.

Table 9.92 Pruning of C2

| Item Pairs in C2 | Bucket | Bit Vector | Remarks | Pruned C2 |
|------------------|--------|------------|------------------|-----------|
| AB, i.e., 12 | 5 | 0 | Removed from C2 | |
| AC, i.e., 13 | 6 | 1 | Will be retained | AC |
| AE, i.e., 15 | 1 | 0 | Removed from C2 | |
| BC, i.e., 23 | 2 | 1 | Will be retained | BC |
| BE, i.e., 25 | 4 | 1 | Will be retained | BE |
| CE, i.e., 35 | 0 | 1 | Will be retained | CE |

This pruned C2 is used to find L2 as shown in Table 9.93.

Table 9.93 Finding L2

| Item pairs | Count |
|------------|-------|
| AC | 2 |
| BC | 2 |
| BE | 3 |
| CE | 2 |

All these item pairs have support more than threshold limit. This L2 will be used for identifying three item pairs by selecting only those item pairs from Table 9.90 which are having support more than the threshold limit, as shown below in Table 9.94

Table 9.94 Identifying three itemsets

| TID | Original Item Pairs | Selected Item Pairs according to L2 | Three Itemsets |
|-----|------------------------|-------------------------------------|----------------|
| 100 | AC, AD, CD | AC | NIL |
| 200 | BC, BE, CE | BC, BE, CE | BCE |
| 300 | AB, AC, AE, BC, BE, CE | AC, BC, BE, CE | BCE |
| 400 | BE | BE | NIL |

Here, BCE has a support of 2; thus, the frequent item group is BCE.

In next phase, association rules for three item group BCE can be found by using the same approach as discussed in the Apriori algorithm.

Example: Let us consider, the database given in Table 9.95 for identification of frequent item pairs with the Apriori algorithm and the DHP algorithm for minimum support of 50%.

Table 9.95 Transaction database

| TID | Item Bought |
|-----|---------------|
| 100 | 1, 2, 3, 4 |
| 200 | 5, 6, 7, 8, 9 |
| 300 | 6, 8, 9, 10 |
| 400 | 2, 3, 9 |

The process to identify frequent item pairs through Apriori algorithm has been illustrated in Figure 9.16.

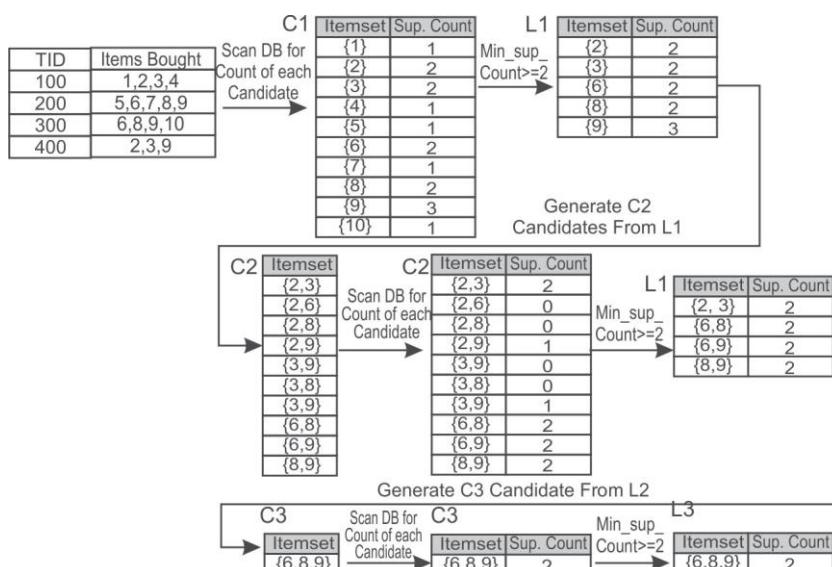


Figure 9.16 Identifying frequent item pairs and groups by Apriori algorithm

Thus, the process of the Apriori algorithm has produced a frequent three item group {6, 8, 9}.

The process to identify frequent item pairs and groups through the DHP algorithm has been illustrated in Table 9.96.

Table 9.96 Coded item pairs for DHP

| TID | Items bought | Item pairs code |
|-----|--------------|-------------------------------|
| 100 | 1,2,3,4 | 12,13,14,23,24,34 |
| 200 | 5,6,7,8,9 | 56,57,58,59,67,68,69,78,79,89 |
| 300 | 6,8,9,10 | 68,69,610,89,810,910 |
| 400 | 2,3,9 | 23,29,39 |

Now, the hash function is applied on coded values so that these items can be assigned to different buckets. Here, we are going to use Modulo 11 number, i.e., dividing by 11 and using the remainder as hash function to assign buckets for the given dataset.

Let's apply hash function to each item pair and assign the item pairs to each bucket on the basis of modulo 11 as shown in Table 9.97.

Table 9.97 Assigning of item pairs to buckets based on hash function modulo 11

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Bucket address |
|----|----|----|----|-----|----|-----|-----|---|---|----|-------------------|
| 12 | 13 | 14 | 59 | 610 | 39 | 29 | 910 | | | | Item pairs |
| 23 | 24 | 58 | | | | 810 | | | | | |
| 34 | 57 | 69 | | | | | | | | | |
| 56 | 68 | 69 | | | | | | | | | |
| 67 | 68 | | | | | | | | | | |
| 89 | | | | | | | | | | | |
| 23 | | | | | | | | | | | |
| 0 | 7 | 5 | 4 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | Support of bucket |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Bit Vector |

The pruning of C2 has been illustrated in Table 9.98.

Table 9.98 Pruning of C2

| Item Pairs in C2 | Bucket | Bit Vector | Remarks | Pruned C2 |
|------------------|--------|------------|------------------|-----------|
| 23 | 1 | 1 | Will be retained | 23 |
| 26 | - | - | Removed from C2 | |
| 28 | - | - | Removed from C2 | |
| 29 | 7 | 1 | Will be retained | 29 |
| 36 | - | - | Removed from C2 | |

Contd.

| Item Pairs in C2 | Bucket | Bit Vector | Remarks | Pruned C2 |
|------------------|--------|------------|------------------|-----------|
| 38 | - | - | Removed from C2 | |
| 39 | 6 | 0 | Removed from C2 | |
| 68 | 2 | 1 | Will be retained | 68 |
| 69 | 3 | 1 | Will be retained | 69 |
| 89 | 1 | 1 | Will be retained | 89 |

This pruned C2 is used to find L2 as shown in Table 9.99.

Table 9.99 Finding L2

| Item pair | Count |
|-----------|-------|
| 23 | 2 |
| 29 | 1 |
| 68 | 2 |
| 69 | 2 |
| 89 | 2 |

Thus, L2 is 23, 68, 69 and 89, which is used to find three item groups as shown in Table 9.100.

Table 9.100 Finding three itemsets

| TID | Original Item pairs | Selected Item pairs according to L2 | Three Itemsets |
|-----|-------------------------------|-------------------------------------|----------------|
| 100 | 12,13,14,23,24,34 | 23 | NIL |
| 200 | 56,57,58,59,67,68,69,78,79,89 | 68,69,89 | 689 |
| 300 | 68,69,610,89,810,910 | 68,69,89 | 689 |
| 400 | 23,39 | 23 | NIL |

Here, 689 has a support of 2 and so the frequent item grouping is 689.

We have found the identical three item grouping by using both the Apriori and the DHP algorithms. Now, the same procedure that was discussed in the section on the Apriori algorithm can be applied to determine rules for the three item group {6, 8, 9}.

9.11 Dynamic Itemset Counting (DIC)

The major issue with the Apriori algorithm is that it requires multiple scans of the transaction database which is equal to the number of items in the last candidate itemset that was checked for its support.

For example, to identify candidate 3-itemset groupings, it will require a minimum of 3 scans of the database and to identify candidate 4-itemset groupings, it will require four scans of the database.

The major advantage of the Dynamic Itemset Counting (DIC) algorithm is that it requires only 2 scans of the database to identify frequent item pairs in the whole database. It reduces the number of scans required by not just doing one scan for the frequent 1-itemset and another for the frequent 2-itemset but combining the counting for a number of itemsets as soon as it appears that it might be necessary to count it.

The basic algorithm is as follows.

1. Divide the transaction database into n number of partitions.
2. In the first partition of the transaction database, count the 1-itemsets.
3. Count the 1-itemsets at the beginning of the second partition of the transaction database and also start counting the 2-itemsets using the frequent 1-itemsets from the first partition.
4. Count the 1-itemsets and the 2-itemsets at the beginning of the third partition and also start counting the 3-itemsets using results from the first two partitions.
5. Continue this process until the whole database is scanned once. There will be final set of frequent 1-itemsets once you scan the database fully.
6. Go back to the starting of the transaction database and continue counting the 2-itemsets and the 3-itemsets, etc.
7. After the end of the first partition in the second scan of the database, the whole database has been scanned for 2-itemsets. Therefore, we have the final set of frequent 2-itemsets.
8. After the end of the second partition in the second scan of the database, the whole database has been scanned for 3-itemsets. Therefore, we have the final set of frequent 3-itemsets.
9. This process is continued in a similar manner until no frequent k-itemsets are identified.

Illustration on working of the DIC algorithm

Let us consider, a dataset having 100 records which is divided into a number of, say 5 equal partitions. Now, the major assumption in DIC is that the data should be homogeneous throughout the file. This means that all the items should appear in partitions and the percentage of each item in the partition should be same as percentage of each item in whole dataset. In simple words, if there are 5 items A, B, C, D and E in the dataset then each of these items should appear in each partition and if the percentage of item A in the whole transaction dataset is 60%, then it should be the same in each partition, i.e. if the count of data item A is 50 in whole dataset of 100 transactions, then the item should have the count of 10 in each partition of 20 records i.e. 50% both ways.

On the assumption that data is homogenous throughout in each partition, data is processed for the DIC algorithm.

Here, we have a database of 100 transactions which is divided into five equal partitions of 20 records each. During counting of the 1-itemsets in the first partition of the transaction database, we have identified data items A, B, C and D in the first partition. The count of these items is indicated as C_A , C_B and so on. Count the 1-itemsets at the beginning of the second partition, and also starts counting the 2-itemsets using the frequent 1-itemsets from the first partition, i.e. we also start counting two item pairs, i.e., AB, AC, AD, BC, BD and CD. The working of the DIC algorithm has been illustrated in Table 9.101.

It has been observed the counting of 1-itemsets, i.e., A, B, C and D has been completed after

Table 9.101 Working of the DIC algorithm for the example database

| Total dataset of 100 records | Partition-1 20 records | PHASE-1 | | PHASE-2 | | PHASE-2 | PHASE-2 | Count of 2-itemset has been completed after first partition of second pass | |
|------------------------------|---------------------------|------------|--|------------|--|--|------------------------|--|--|
| | | 1-itemsets | A-C _A B-C _B C-C _C D-C _D | | 2-itemsets | AB-C _{AB} AC-C _{AC} AD-C _{AD} BC-C _{BC} BD-C _{BD} CD-C _{CD} | 3-itemsets | ABC-C _{ABC} ABD-C _{ABD} ACD-C _{ACD} BCD-C _{BCD} | |
| | | 1-itemsets | A-C _A B-C _B C-C _C D-C _D | 2-itemsets | | 3-itemsets | 4-itemsets | Count of 3-itemset has been completed after second partition of second pass | |
| | | 1-itemsets | A-C _A B-C _B C-C _C D-C _D | 2-itemsets | AB-C _{AB} AC-C _{AC} AD-C _{AD} BC-C _{BC} BD-C _{BD} CD-C _{CD} | ABC-C _{ABC} ABD-C _{ABD} ACD-C _{ACD} BCD-C _{BCD} | ABCD-C _{ABCD} | Count of 4-itemsets has been completed after third partition of second pass | |

Contd.

| | | | | | | |
|--|---------------------------|--|--|--|-------------------------------|---|
| | Partition 4 20 records | 1-itemsets $A-C_A$ $B-C_B$ $C-C_C$ $D-C_D$ | 2-itemsets $AB-C_{AB}$ $AC-C_{AC}$ $AD-C_{AD}$ $BC-C_{BC}$ $BD-C_{BD}$ $CD-C_{CD}$ | 3-itemsets $ABC-C_{ABC}$ $ABD-C_{ABD}$ $ACD-C_{ACD}$ $BCD-C_{BCD}$ | 4-itemsets $ABCD-C_{ABCD}$ | |
| | Partition 5 20 records | 1-itemsets $A-C_A$ $B-C_B$ $C-C_C$ $D-C_D$ | 2-itemsets $AB-C_{AB}$ $AC-C_{AC}$ $AD-C_{AD}$ $BC-C_{BC}$ $BD-C_{BD}$ $CD-C_{CD}$ | 3-itemsets $ABC-C_{ABC}$ $ABD-C_{ABD}$ $ACD-C_{ACD}$ $BCD-C_{BCD}$ | 4-itemsets $ABCD-C_{ABCD}$ | Count of 1-itemset has been completed after one pass |
| | | | | | | |

one pass, the counting of 2-itemset has been completed after the first partition of the second pass, counting of 3-itemsets has been completed after the second partition of the second pass and counting of 4-itemsets has been completed after the third partition of the second pass. It is clear from above illustration that DIC will require only two scans of database while Apriori will require 4 scans for the identification of frequent 4-itemsets.

Major condition to make DIC effective

The DIC algorithm works well when the data throughout the file is relatively homogeneous as it starts counting of the 2-itemset before having a count of final 1-itemset. The algorithm is not able to identify the large itemset till the whole database is scanned if the data distribution is not homogeneous. The major benefit of DIC is that it finishes the counting of the itemset in two scans of the database while Apriori often takes three or more scans.

9.12 Mining Frequent Patterns without Candidate Generation (FP Growth)

As only frequent items are required for identifying the association rules, so it is best to identify the frequent items in the dataset and ignore all others. The FP growth works on the same principle and instead of generating candidate itemsets as in the case of the Aprioiri algorithm, frequent pattern-growth (FP growth) only tests and generates frequent itemsets. Thus, the major difference between the Apriori algorithm and FP growth is that FP growth does not generate candidate itemsets, it only tests, whereas the Apriori algorithm generates candidate itemsets and then tests them.

To improve the performance of the algorithm, it uses the principle to store frequent items in a compact structure which does not require the need to use the original transaction database repeatedly.

The working of the FP growth algorithm is as follows:

1. Like the Apriori algorithm, the transaction database is scanned once to identify all the 1-itemset frequent items and their supports are noted down.
2. Then 1-itemset frequent items are sorted in descending order of their support.
3. After that the FP-tree is created with a 'NULL' root.
4. Then the first transaction is obtained from the transaction database and all the non-frequent items are removed. And the remaining items are listed according to the order in the sorted frequent items.
5. After this, the first branch of the tree is constructed using the transaction with each node corresponding to a frequent item and by setting the frequency of each item as 1 for the first transaction.
6. Then the next transaction is retrieved from the transaction database and all the non-frequent items are removed. And the remaining items are listed according to the order in the sorted frequent items as done in step 4.
7. Now, the transaction is inserted in the tree using any common prefix that may appear and

count of items is increased by 1. If no common prefix is found then the branch of the tree is constructed using the transaction; with each node corresponding to a frequent item and by setting the frequency of each item as 1.

8. Step 6 is repeated until all transactions in the database have been processed.

To understand this algorithm let us find frequent item pairs by building FP-trees for the following database.

Example: Let us consider a transaction database shown in Table 9.102. The minimum support required is 50% and confidence is 75%.

Table 9.102 Transaction database

| | |
|-----|----------------------------|
| 100 | Butter, Curd, Eggs, Jam |
| 200 | Butter, Curd, Jam |
| 300 | Butter, Eggs, Muffin, Nuts |
| 400 | Butter, Eggs, Jam, Muffin |
| 500 | Curd, Jam, Muffin |

According to the algorithm, the first step will be to scan the transaction database to identify all the frequent items in the 1-itemsets and sort them in descending order of their support.

The frequent 1-itemset database sorted on the basis of frequency is shown in Table 9.103.

Table 9.103 Frequency of each item in sorted order

| Item | Frequency |
|--------|-----------|
| Butter | 4 |
| Jam | 4 |
| Curd | 3 |
| Eggs | 3 |
| Muffin | 3 |

Here, Nuts has been removed as it has a support of 1 only, which is less than the threshold limit of support (which is 3 in our case).

To create the FP-tree, the next step will be to exclude all the items that are not frequent from the transactions and sort the remaining frequent items in descending order of their frequency count. The example transaction database will be processed accordingly. The updated database after eliminating the non-frequent items and reorganising the frequent items on the basis of frequency, is shown in Table 9.104.

Table 9.104 Updated database after eliminating the non-frequent items and reorganising it according to support

| Transaction ID | Items |
|----------------|---------------------------|
| 100 | Butter, Jam, Curd, Eggs |
| 200 | Butter, Jam, Curd |
| 300 | Butter, Eggs, Muffin |
| 400 | Butter, Jam, Eggs, Muffin |
| 500 | Jam, Curd, Muffin |

It is important to note that in case of identical support, items are arranged according to lexicographic or sorted order. Now, we create the FP-tree by using the algorithm discussed above. For simplicity of discussion, we are referencing items with their first letter, i.e., Butter by B, Jam by J and so on.

An FP-tree consists of nodes. Here, each node contains three fields, i.e., an item name, a count, and a node link as given in Figure 9.17. To build the FP-tree let us consider first transaction, i.e., 100 having Butter, Jam, Curd, Eggs as items. Each item node will be created from the root node, i.e., NULL as shown in Figure 9.17. The tree is built by making a root node labeled NULL. A node is made for each frequent item in the first transaction and the count is set to 1. For example, to build the FP-tree of Figure 9.17, the first transaction {B, J, C, E} is inserted in the empty tree with the root node labeled NULL. Each of these items is given a frequency count of 1.

Next, the tree is traversed for the next transaction, i.e., 200. If a path already exists then it will follow the same path and corresponding count of item is increased by 1. If a path does not exist then a new path will be created. Next the second transaction, which is {B, J, C} having all the items common with first, is inserted. It changes the frequency of each item, i.e., {B, J, C} to 2. The FP for first two transactions is shown in Figure 9.18.



Figure 9.17 FP-tree for first transaction,
i.e., 100 only



Figure 9.18 FP-tree for the first
two transactions

Similarly, third transaction is considered and {B, E, M} is inserted. This requires that nodes for E and M be created. The counter for B goes to 3 and the counter for E and M is set to 1. The FP-tree for first three transactions is shown in Figure 9.19.

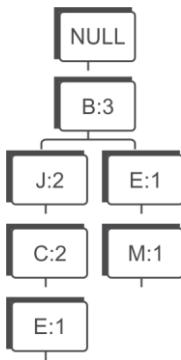


Figure 9.19 FP-tree for first three transactions

The next transaction {B, J, E, M} results in counters for B and J going up to 4 and 3 respectively and a new nodes for E and M have been inserted with count of 1 under node J as shown in Figure 9.20.

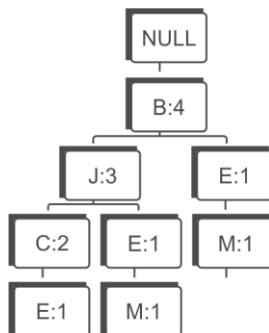


Figure 9.20 FP-tree for first four transactions

The last transaction {J, C, M} results in a brand new branch for the tree which is shown on the right-hand side in Figure 9.21 with a counter of 1 for each item.

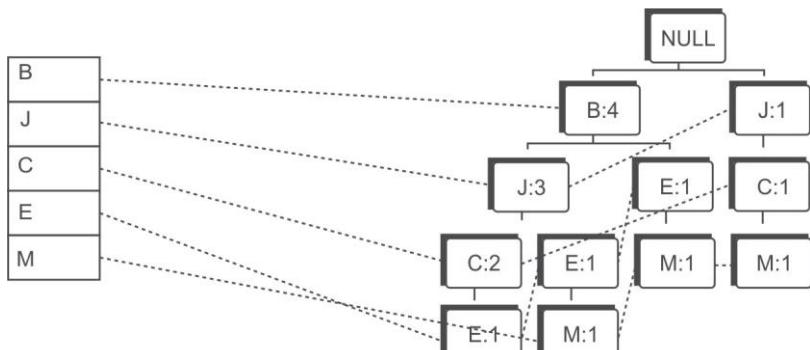


Figure 9.21 The final FP-tree for the example database after five transactions

The count tells the number of occurrences the path (constructed from the root node to this node) has in the transaction database. The node link is a link to the next node in the FP-tree containing the same item name or a null pointer if this node is the last one with this name. The FP-tree also consists of a header table with an entry for each itemset and a link to the first item in the tree with the same name. This linking is done to make traversal of the tree more efficient. Nodes with the same item name in the tree are linked via the dotted node-links.

The nodes near the root in the tree are more frequent than those further down the tree. Each identical path is only stored in the FP-tree once, which often makes the size of an FP-tree smaller than the corresponding transactional database. The height of an FP-tree is always equal to the maximum number of itemsets in a transaction excluding the root node. The FP-tree is compact and often orders of magnitude smaller than the transaction database. Once the FP-tree is constructed, the transaction database is not required.

Identification of frequent itemsets from FP-tree

The mining on the FP-tree structure is performed using the Frequent Pattern growth (FP growth) algorithm. This algorithm starts with the least frequent item, *i.e.*, the last item in the header table. Then the algorithm identifies all the paths from the root to this least frequent item and adjusts the count on the basis of the item's support count.

First, look at Figure 9.21 built using the FP-tree to identify the frequent itemsets in the example.

Start with the item M and identify the patterns given as follows:

Identification of patterns for item M

BEM(1)

BJEM(1)

JCM(1)

From this we can identify that, the support for BM is 2 (1 from BEM and 1 from BJEM), but its support is less than the threshold limit of 3, so this item pair will be discarded and we have identified no frequent item pairs by using item M.

Next consider the item E and identify the patterns given as follows:

Identification of patterns for item E

BJCE(1)

BJE(1)

BE(1)

From this we can identify that the support for BE is 3 (1 from BJCE, 1 from BJE and 1 from BE). However there is also a three item pair BJE having a support of 2 (1 from BJCE and 1 BJE), but its support is less than threshold value so will be discarded.

So, we have identified BE has two item pairs having support more than the threshold limit.

Next consider the item C and identify the following patterns:

Identification of patterns for item C

BJC(2)

JC(1)

From this we can identify that, the support for JC is 3 (2 from BJC and 1 from JC). There is no other item pair having support equal to or more than 3.

So, we have identified JC has two item pairs having supports more than the threshold limit.

Next consider the item J to identify its patterns.

Identification of patterns for item J

BJ(3)

J(1)

From this we can identify that, the support for BJ is 3. There is no other item pair having support equal to or more than 3.

So, we have identified that BJ has two item pairs having support more than the threshold limit.

Since, B is at top of the tree and cannot have any item prior to it to make a pair, so there is no need to follow links for item B. Thus, when the tree only contains one path, the algorithm will find all the possible combinations of the items in the itemset that support the minimum support count. It is therefore important to start with the least frequent item the first time, otherwise the items with the higher frequency will not be considered.

Thus, the final list of frequent item pairs for the given database of transactions is shown in Table 9.105.

Table 9.105 Frequent item pairs for database example given in table

| <i>Frequent Item pairs</i> | <i>Count</i> |
|----------------------------|--------------|
| BE | 3 |
| JC | 3 |
| BJ | 3 |

Finding association rules

To find the association mining rules, the same approach is followed as in Apriori algorithm.

For item pair BE, the possible rules are $B \rightarrow E$ and $E \rightarrow B$. The confidence for each rule will be calculated and if its value is more than given threshold value of confidence then corresponding rule will be selected otherwise it will be discarded. The same approach will be followed for other frequent item pairs JC and BJ to find association rules for the given dataset.

The readers are advised to identify association mining rules that have confidence of more than 80%.

I hope that, you have identified it correctly and answer is $E \rightarrow B$ and $C \rightarrow J$ as explained below.

Confidence of $(B \rightarrow E) = S(B \cap E) / S(B) = 3/4 = 0.75$ [support of individual items is given in Table 9.101]

Confidence of $(E \rightarrow B) = S(E \cap B) / S(E) = 3/3 = 1.0$

Confidence of $(J \rightarrow C) = S(J \cap C) / S(J) = 3/4 = 0.75$

Confidence of $(C \rightarrow J) = S(C \cap J) / S(C) = 3/3 = 1$

Confidence of $(B \rightarrow J) = S(B \cap J) / S(B) = 3/4 = 0.75$

Confidence of $(J \rightarrow B) = S(J \cap B) / S(J) = 3/4 = 0.75$

Example: Let us consider another database given in Table 9.106, to identify frequent item pairs having support more than 50%.

Table 9.106 Transaction database

| TID | Items |
|-----|---------|
| 100 | A C D |
| 200 | B C E |
| 300 | A B C E |
| 400 | B E |

The count for each item is given in Table 9.107.

Table 9.107 Count for each data item

| Item | Count |
|------|-------|
| A | 2 |
| B | 3 |
| C | 3 |
| D | 1 |
| E | 3 |

On the basis of the frequent items given in Table 9.108, those items are selected that have support equal to or more than 2 and these frequent items are sorted on the basis of their frequency as shown in Table 9.108.

Table 9.108 Frequency of each item in sorted order

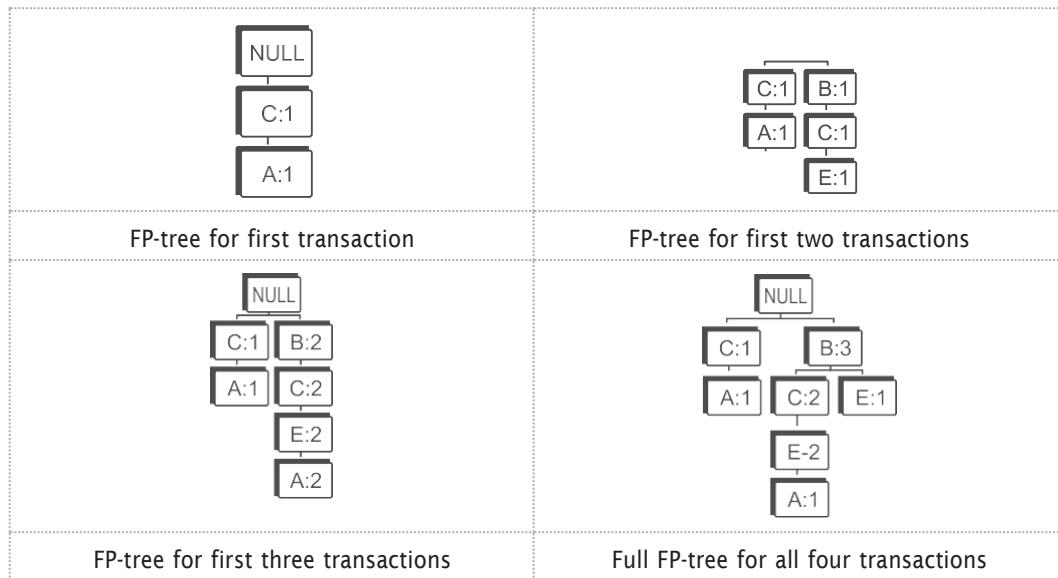
| Item | Count |
|------|-------|
| B | 3 |
| C | 3 |
| E | 3 |
| A | 2 |

Now, the non-frequent items are removed from the transaction database and items are ordered on the basis of their frequency as given in the above table. The database after eliminating the items that are not frequent and reorganising them on the basis of their frequency is shown in Table 9.109.

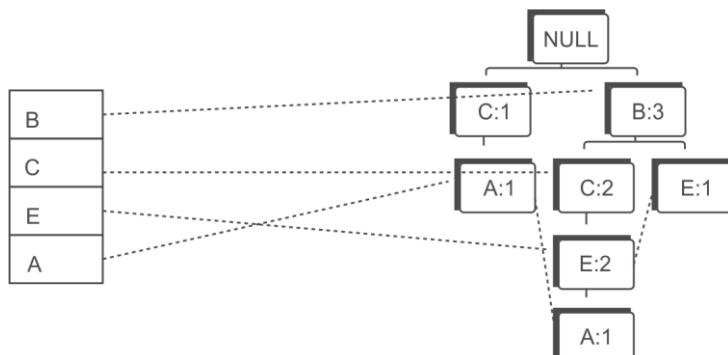
Table 9.109 Modified database after eliminating the non-frequent items and reorganising

| TID | Items |
|-----|---------|
| 100 | C A |
| 200 | B C E |
| 300 | B C E A |
| 400 | B E |

The process of creating the FP-tree for the database given above has been shown in Figure 9.22 on a step-by-step basis.

**Figure 9.22** Illustrating the step-by-step creation of the FP-tree

Thus, the final FP-tree with nodes will be as shown in Figure 9.23.

**Figure 9.23** Final FP-tree for database given in Table 9.105

Identification of Frequent Itemsets from the FP-tree

Start with the item A and identify the patterns given as follows:

Identification of patterns for item A

CA(1),
BCEA(1),

The support for CA is 2 (1 from CA and 1 from BCEA) and it is equal to the threshold limit thus CA is identified as a frequent item pair.

Next consider the item E and identify the following patterns:

Identification of patterns for item E

BCE(2)
BE(1)

The support for BCE is 2 and it is equal to the threshold limit thus BCE is identified as a frequent 3-itemset.

Also from E, BE is identified as a frequent 2-item pair, since it has a support of 3 (2 from BCE and 1 from BE).

Thus, we have selected two frequent item pairs from E, i.e., BCE and BE.

Next consider the item C and examine the following patterns:

Identification of patterns for item C

BC(2)

The support for BC is 2 and it is equal to the threshold limit thus BC is identified as a frequent item pair.

Since, B is at top of the tree and cannot have any item prior to it to make a pair, so there is no need to follow links for item B.

The final list of frequent item pairs for the given database of transactions is as shown in Table 9.110.

Table 9.110 Frequent item pairs for the example database

| <i>Frequent Item pairs</i> | <i>Count</i> |
|----------------------------|--------------|
| CA | 2 |
| BCE | 2 |
| BE | 2 |
| BC | 2 |

Finding association rules

The identification process of association rules having confidence more than 70% has been shown in Table 9.111.

Table 9.111 Calculation of confidence for identification of association rules

| | | |
|-----|---|---|
| CA | $C \rightarrow A = S(C \cap A) / S(C) = 2/3 = 0.67$ $A \rightarrow C = S(A \cap C) / S(A) = 2/2 = 1.0$ | Selected rule is $A \rightarrow C$ |
| BCE | $B \rightarrow CE = S(B \cap C \cap E) / S(B) = 2/3 = 0.67$ | Confidence is less than threshold value, so it will be discarded. |
| | $CE \rightarrow B = S(B \cap C \cap E) / S(CE) = 2/2 = 1.0$ Possible other non-implied rules are: $C \rightarrow E$, $E \rightarrow B$ $C \rightarrow E = S(C \cap E) / S(C) = 2/3 = 0.67$ $E \rightarrow B = S(E \cap B) / S(E) = 2/3 = 0.67$ | Selected rule is $CE \rightarrow B$. Others are discarded |
| BE | $B \rightarrow E = S(B \cap E) / S(B) = 2/3 = 0.67$ $E \rightarrow B = S(E \cap B) / S(E) = 2/3 = 0.67$ | Both the rules are discarded. |
| BC | $B \rightarrow C = S(B \cap C) / S(B) = 2/3 = 0.67$ $C \rightarrow B = S(C \cap B) / S(C) = 2/3 = 0.67$ | Both the rules are discarded. |

Thus, final association rules having confidence more than 70% are $CE \rightarrow B$ and $A \rightarrow C$.

Example: Let us consider another database given in Table 9.112, to identify frequent item pairs having support more than 50%.

Table 9.112 Transaction database

| Tr. | Itemsets |
|-----|-----------|
| T1 | {1,2,3,4} |
| T2 | {1,2,4} |
| T3 | {1,2} |
| T4 | {2,3,4} |
| T5 | {2,3} |
| T6 | {3,4} |
| T7 | {2,4} |

The count for each item is given in Table 9.113.

Table 9.113 Frequency of each item

| Item | Support |
|------|---------|
| {1} | 3 |
| {2} | 6 |
| {3} | 4 |
| {4} | 5 |

On the basis of the frequent items given in Table 9.114, those items are selected that have support equal to or more than 2 and these frequent items are sorted on the basis of their frequency as shown in Table 9.114.

Table 9.114 Frequency of each item in sorted order

| Item | Support |
|------|---------|
| {2} | 6 |
| {4} | 5 |
| {3} | 4 |
| {1} | 3 |

Now, the non-frequent items are removed from the transactions and items are organized according to their frequency as given in the above Table 9.114. Since, all items are frequent, transaction items are reorganized according to their support as shown in Table 9.115.

Table 9.115 Modified database after eliminating the non-frequent items and reorganizing

| | |
|----|---------|
| T1 | 2 4 3 1 |
| T2 | 2 4 1 |
| T3 | 2 1 |
| T4 | 2 4 3 |
| T5 | 2 3 |
| T6 | 4 3 |
| T7 | 2 4 |

The process of creating the FP-tree for this database has been shown in Figure 9.24 on a step-by-step basis.



Contd.

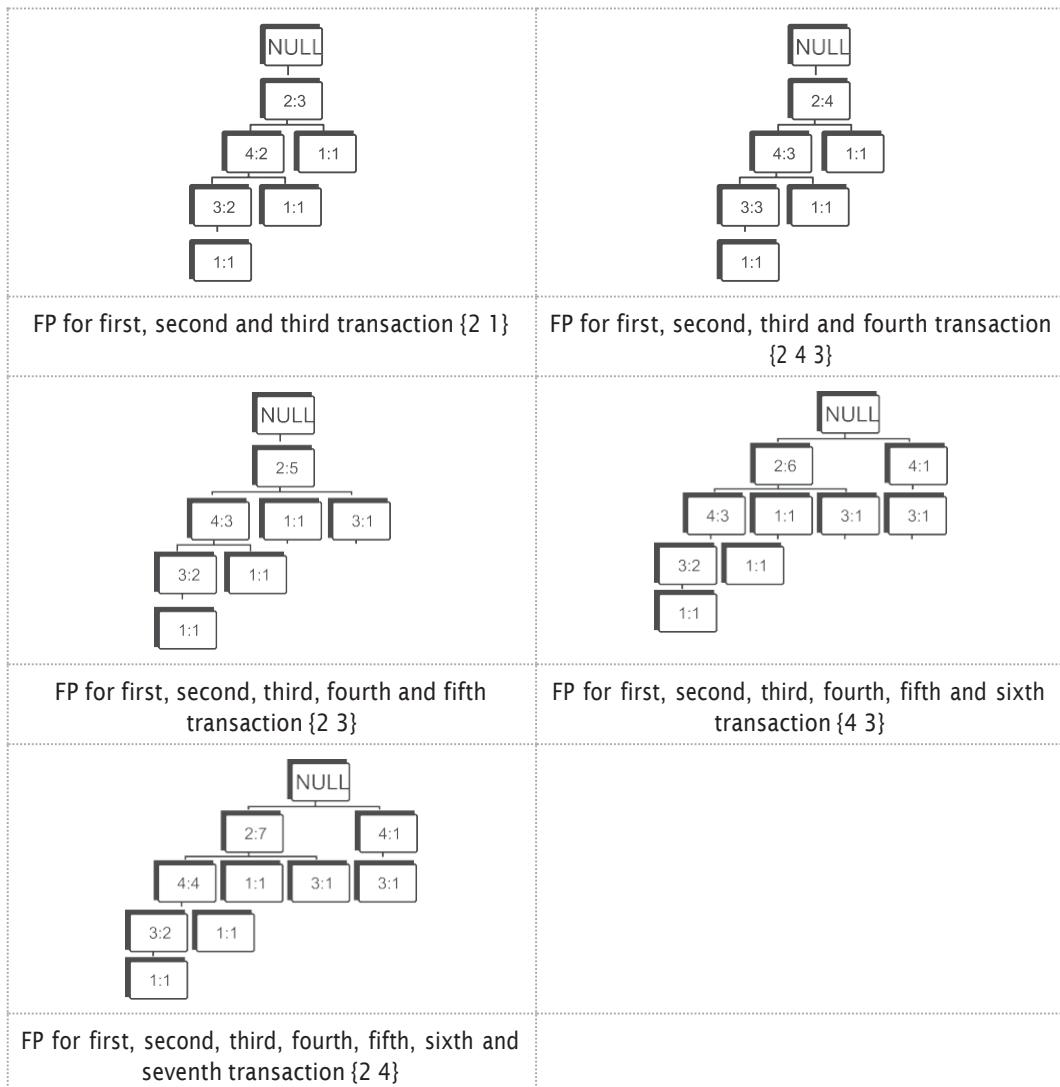


Figure 9.24 Illustrating the step-by-step creation of the FP-tree

Thus, the final FP-tree with nodes will be as shown in Figure 9.25.



Figure 9.25 FP-tree for the example database

Identification of frequent itemsets from the FP-tree

Start with the item 1 and identify the patterns given as follows.

Identification of patterns for item 1

2431(1)

241(1)

21 (1)

The support for 21 is 3 (1 from 2431, 1 from 241 and 1 from 21) and it is equal to the threshold limit thus 21 is identified as frequent item pair.

Next consider the item 3 and identify its patterns.

Identification of patterns for item 3

243(2)

23(1)

43(1)

The support for 43 is 3 (2 from 243 and 1 from 43) and it is equal to the threshold limit thus 43 is identified as a frequent item pair.

The support for 23 is also 3 (2 from 243 and 1 from 23) and it is equal to the threshold limit thus 23 is also identified as a frequent item pair.

Next consider the item 4 and identify its patterns.

Identifications of patterns for item 4

24(4)

The support for 24 is 4 and it is more than threshold limit, thus 24 is identified as a frequent item pair.

The final list of frequent item pairs for given database of transactions is shown in Table 9.116.

Table 9.116 Frequent item pairs for example database

| Item | Support |
|-------|---------|
| {1,2} | 3 |
| {3,4} | 3 |
| {2,3} | 3 |
| {2,4} | 4 |

It is important to note that the order of data items does not matter in listing frequent item pairs.

Finding association rules

The identification process of association rules having confidence more than 70% has been shown in Table 9.117.

Table 9.117 Association rules for database given in Table 9.76

| | | |
|--------|--|--|
| {1, 2} | $1 \rightarrow 2 = S(1 \cap 2) / S(1) = 3/3 = 1.0$ $2 \rightarrow 1 = S(2 \cap 1) / S(1) = 3/3 = 1.0$ | Both the rules, i.e., $1 \rightarrow 2$ and $2 \rightarrow 1$ are selected |
| {3, 4} | $3 \rightarrow 4 = S(3 \cap 4) / S(3) = 3/4 = 0.75$ $4 \rightarrow 3 = S(4 \cap 3) / S(3) = 3/4 = 0.75$ | Both the rules, i.e., $3 \rightarrow 4$ and $4 \rightarrow 3$ are selected |
| {2, 3} | $2 \rightarrow 3 = S(2 \cap 3) / S(2) = 3/6 = 0.50$ $3 \rightarrow 2 = S(3 \cap 2) / S(3) = 3/4 = 0.75$ | $3 \rightarrow 2$ rule has been selected |
| {2, 4} | $2 \rightarrow 4 = S(2 \cap 4) / S(2) = 4/6 = 0.67$ $4 \rightarrow 2 = S(4 \cap 2) / S(4) = 4/5 = 0.80$ | $4 \rightarrow 2$ rule has been selected |

Thus, final association rules having confidence more than 70% are $1 \rightarrow 2$, $2 \rightarrow 1$, $3 \rightarrow 4$, $4 \rightarrow 3$, $3 \rightarrow 2$ and $4 \rightarrow 2$.

9.12.1 Advantages of the FP-tree approach

The advantages of the FP-tree algorithm over the Apriori algorithm are as follows:

- The FP-tree algorithm avoids the need to scan the database more than twice to identify the counts of support.
- The FP-tree completely eliminates the requirement of costly candidate generation, which can be expensive in case of the Apriori algorithm for the candidate set C_2 .
- The FP growth algorithm is better than the Apriori algorithm when the transaction database is huge and the minimum support count is low. Because in case of low minimum support count, there will be more items that will satisfy the support count and hence the size of the candidate sets for Apriori will be large. As the database grows, FP growth uses an efficient structure to mine patterns.

9.12.2 Further improvements of FP growth

In spite of the advantages there are possibilities to make the algorithm even more efficient and scalable. Important observations to further improve the performance of FP growth are as follows:

- It works best if the FP-tree fits in the main memory. However, in large databases this will not always be possible, and then it is better to first divide the database and then constructs FP-trees for each of the databases.
- Use **B+-tree** to index the FP-tree for faster access to items in the FP-tree.
- If the same data is used multiple times with different minimum confidence thresholds in order to yield different association rule mining then, it could be useful to use the same FP-tree results. This will save the initial cost of generating the FP-tree and make the algorithm overall more efficient.
- Sometimes the minimum support count will be different in different tasks, so to overcome this problem, the lowest minimum support count is used in the materialization of the FP-tree. When a higher support count is wanted, only the top of the FP-tree is used. Because of the structure of the FP-tree, it is easy to have different support counts.