

## Module 5

Association Rules Mining:- Concepts, Apriori and FP-Growth algorithm.

Cluster Analysis:- Introduction, Concepts, Types of data in cluster analysis, Categorization of clustering methods, Partitioning method:- k-means and k-Meetoid clustering



# NOTES

*the learning companion*

## Cluster Analysis

- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.
- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- Clustering is also called data segmentation.
- Clustering can also be used for outlier detection.
- Clustering is an example of unsupervised learning.
- Requirements of clustering in data mining

i. Scalability: Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions of objects. Clustering on a sample

of a given large data set may lead to biased results. Highly scalable clustering algorithms are needed.

2. Ability to deal with different types of attributes

Many algorithms are designed to cluster interval-based (numerical) data. However, applications may require clustering other types of data, such as binary, categorical (nominal) and ordinal data or mixtures of these data types.

3. Discovery of clusters with arbitrary shape.

Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures. Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could be of any shape. It is important to develop algorithms that can detect clusters

of arbitrary shape

#### 4. Minimal requirements of domain knowledge to determine input parameters

Many clustering algorithms require users to input certain parameters in cluster analysis (such as the number of desired clusters). The clustering results can be quite sensitive to input parameters. Parameters are often difficult to determine, especially for datasets containing high dimensional objects.

#### 5. Ability to deal with noisy data

Most real-world database contain outliers or missing, unknown or erroneous data.

Some clustering algorithms are sensitive to such data and may lead to clusters of poor quality.

#### 6. Incremental clustering and insensitivity to the order of input records

Some clustering algorithms can not

Incorporate newly inserted data to existing clustering structure and instead, must determine a new clustering from scratch.

Some clustering algorithms are sensitive to the order of input data. That is given a set of data objects, such an algorithm may return dramatically different clustering depending on the order of presentation of input objects. It is important to develop incremental clustering algorithms and algorithms that are insensitive to the order of input.

#### 7. High dimensionality:-

A database or a data warehouse can contain several dimensions or attributes.

Many clustering algorithms are good at handling low dimensional data, involving only two or three dimensions. Finding clusters of data objects in high dimensional space is challenging.

## 8. Constraint-based clustering:

Real world applications may need to perform clustering under various kinds of constraints. A challenging task is to find groups of data with good clustering behaviour that satisfy specified constraints.

## 9. Interpretability & Usability:

Users expect clustering results should be interpretable, comprehensible and usable.

## Types of data in cluster Analysis

→ Main memory based clustering algorithms typically operate on either of the following two data structures.

### 1. Data Matrix (Object - by - Variable Structure)

This represents  $n$  objects, such as persons with  $p$  variables (also called measurements or attributes), such as age, height, weight, gender and so on.

The structure is in the form of a relational table or n by p matrix  
(n objects x p variables)

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ x_{21} & \dots & x_{2f} & \dots & x_{2p} \\ \vdots & & & & \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

### 2. Similarity

2. Dissimilarity Matrix [Object by object structure]

This stores a collection of proximities that are available for all pairs of n objects. It is often represented by an n by n table.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & & & & \\ d(n,1) & d(n,2) & & \dots & 0 \end{bmatrix}$$

$d(i,j)$  is the measured difference or dissimilarity between object  $i$  and  $j$ . In general  $d(i,j)$  is a nonnegative no that close to 0 when objects  $i$  and  $j$  are highly similar or near each other.

→ Rows and columns of the data matrix represent different entities while those of the dissimilarity matrix represent the same entity.

So data matrix is called two-mode matrix dissimilarity matrix is called one-mode matrix

→ Many clustering algorithms operate on dissimilarity matrix. If data are represented in the form of a data matrix, it can first be transformed into a dissimilarity matrix before applying such clustering algorithms.

# Different Types of data in cluster analysis

## Interval-Scaled Variables

→ Interval scaled variables are continuous measurements of a roughly linear scale.

e.g.: height, weight, temperature.

→ The measurement unit used can affect the clustering analysis.

→ To avoid the dependence on the choice of measurement unit, the data should be standardized.

→ To standardize measurements, one choice is to convert the original measurements to unitless variables. Given measurements for a variable  $f$ , this can be performed as follows:

1. Calculate the mean absolute deviation  $s_f$

$$s_f = \frac{1}{n} (|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)$$

where  $x_{if}, x_{2f}, \dots, x_{nf}$  are  $n$  measurements of  $f$  and  $m_f$  is the mean value of  $f$ .

$$m_f = \frac{1}{n} (x_{if} + x_{2f} + \dots + x_{nf})$$

2. Calculate the standardized measurement or z-score

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

→ After standardization or without standardization in certain applications, the dissimilarity (or similarity) between the objects described by interval scaled variables is typically computed based on the distance between each pair of object. The most popular distance measure is Euclidean distance

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{in} - x_{jn})^2}$$

where  $i = (x_{i1}, x_{i2}, \dots, x_{in})$  &  $j = (x_{j1}, x_{j2}, \dots, x_{jn})$  are two  $n$ -dimensional data objects.

Another well-known metric is Manhattan (city block) distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{in} - x_{jn}|$$

Both the Euclidean and Manhattan distance satisfy the following mathematic requirements of a distance function:

1.  $d(i,j) \geq 0$ , Distance is a nonnegative no.
2.  $d(i,i) = 0$ , The distance of an object to itself is 0.
3.  $d(i,j) = d(j,i)$ ; Distance is a symmetric function.
4.  $d(i,j) \leq d(i,h) + d(h,j)$

Going directly from object  $i$  to object  $j$  in space is no more than making a detour over any other object  $h$ .

→ Minkowski distance is a generalization of both Euclidean distance and Manhattan distance

$$d(i,j) = \left( |x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \dots + |x_{in} - x_{jn}|^p \right)^{1/p}$$

$p$  is positive integer

$p = 1$  = Manhattan distance

$p = 2$  = Euclidean distance

→ If a variable is assigned a weight according to its perceived importance,

the weighted Euclidean distance can be

computed as

$$d(i,j) = \sqrt{w_1 |x_{i1} - x_{j1}|^2 + w_2 |x_{i2} - x_{j2}|^2 + \dots + w_m |x_{in} - x_{jn}|^2}$$

## 2. Binary Variables.

A binary variable has only two states: 0 or 1, where 0 means that variable is absent and 1 means that it is present.

→ How to compute dissimilarity between two binary variables?

\* One approach involves compiling a dissimilarity matrix from the given binary data.

\* If all binary variables are thought of as having the same weight, we can create a contingency table

		Object j		Sum
		1	0	
Object i	1	r	s	r+s
	0	t	u	t+u
Sum	q+s	r+t	p	

$q$  = no of variables that equal 1 for both object i and j

$r$  = no of variables that equal 1 for object i but that are for object j

$s$  = no of variables that are equal 0 for object i and equal 1 for object j

$t$  = no of variables that are equal 0 for both object i and j.

$P = q + r + s + t \rightarrow$  total no of variables.

→ A binary variable is symmetric if both of its states are equally valuable and carry the same weight:

e.g.: Gender      Male 0  
                        Female 1.

→ Dissimilarity that is based on symmetric binary variable is called symmetric binary dissimilarity

$$d(ij) = \frac{r+s}{q+r+s+t}$$

→ A binary variable is asymmetric if the outcome of the states are not equally important.

eg: Cancerous - non cancerous.

→ The dissimilarity based on such variables is called asymmetric binary dissimilarity.

$$d(i,j) = \frac{r+s}{q+r+s}$$

→ asymmetric binary similarity b/w object i & j

$$\text{Sim}(i,j) = \frac{q}{q+r+s} = 1 - d(i,j)$$

\* Sim(i,j) - Taccard coefficient.

eg: Suppose that a patient record table contains the attributes name, gender, fever, cough, test-1, test-2, test-3 and test-4. Gender is symmetric attribute & remaining are asymmetric.

Name	gender	fever	cough	test 1	test 2	test 3	test 4
Jack	M	Y	N	P	N	N	N
Mae	F	Y	N	P	N	P	N
Tim	M	Y	Y	N	N	N	N

→ If distance is computed based on asymmetric variables.

$$d(\text{Jack}, \text{Mae}) = \frac{0+1}{2+0+1} = 0.33$$

$$d(\text{Jack}, \text{Tim}) = \frac{1+1}{1+1+1} = 0.67$$

$$d(\text{Mae}, \text{Tim}) = \frac{1+2}{1+1+2} = 0.75$$

### 3. Categorical Variables:

A categorical variable is a generalization of the binary variable in that it can take on more than two states.

e.g: color

→ The states of the categorical variables can be denoted by letters, symbols or a set of integers.

→ Dissimilarity computed between objects described by categorical variables is

$$d(i, j) = \frac{P-m}{P}$$

$m$  = no of matches (no of variables for which  $i$  and  $j$  are in same state)

$P$  = total no of variables.

Object Identifier	test - 1 (categorical)
Code 1	Code A
2	Code B
3	Code C
4	Code A

$$P = 4$$

$$d(1, 1) = 0$$

$$d(2, 1) = 1 \left( \frac{P-0}{P} = \frac{4-0}{4} = 1 \right)$$

Dissimilarity matrix is

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

#### 4. Ordinal Variables.

→ A discrete ordinal variable resembles categorical variable, except that the states of the ordinal value are ordered in a meaningful sequence.

e.g.: Professional rank - Assistant Professor  
Associate Professor  
Professor

→ A continuous ordinal variable looks like a set of continuous data of an unknown scale; i.e. the relative ordering of the values is essential but their actual magnitude is not.

e.g.: relative ranking in a sports (gold  
silver  
bronze)

→ Suppose that  $f$  is a variable from a set of ordinal variables describing  $n$  objects. The dissimilarity computation with respect to  $f$  involves the following steps:

1. The value of  $f$  for the  $i^{\text{th}}$  object is  $x_{if}$  and  $f$  has  $M_f$  ordered states, representing  $1, \dots, M_f$ . Replace each  $x_{if}$  by its corresponding rank.

$$x_{if} \in \{1, \dots, M_f\}$$

2. Since each ordinal variable can have a different number of states, it is often necessary to map the range of each variable onto  $[0.0, 1.0]$ , so that each variable has equal weight. This can be achieved by replacing the rank  $x_{if}$  of the  $i^{\text{th}}$  object in the  $f^{\text{th}}$  variable by

$$z_{if} = \frac{x_{if}}{M_f - 1}$$

3. Dissimilarity can then be computed using any of the distance measures.

e.g.: Object  
Identifier

test - 2  
(Cordinal)

1  
2  
3  
4

Excellent (3)  
Fair (2)  
Good (2)  
Excellent (3)

$$M_f = 3$$

Object

test 2

1  
2  
3

Object	test 2
1	1
2	0
3	0.5
4	1

Normalize:

1  
2  
3  
4

## Dissimilarity matrix

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

## 5. Ratio-Scaled Variables

A ratio scaled variable makes a positive measurement on a non linear scale, such as an exponential scale, approximately following the formula

$$A e^{Bt} \text{ or } A e^{-Bt} \quad A, B - +ve \text{ constants}$$

$t$  - time

e.g.: Growth of bacteria population

## How to compute ratio-scaled variables?

Three approaches are used:

1. Treat ratio-scaled variables like interval scaled variables.
2. Treat  $x_{if}$  as continuous ordinal data and treat their ranks as interval valued.
3. Apply logarithmic transformation to a ratio-scaled variable  $f$  having value  $x_{pf}$  for object  $i$  by using formula

$y_{if} = \log(x_{if})$ . The  $y_{if}$  values can be treated as interval valued.

e.g.: Object identifiers

test 3  
(ratio scaled)

1 445

2 22

3 164

4 1210

Take logarithmic transformation

1	algm 2.65
2	1.34
3	2.21
4	3.08

Using Euclidean distance, dissimilarity measure matrix

$$\begin{bmatrix} 0 & & & \\ 1.31 & 0 & & \\ 0.44 & 0.87 & 0 & \\ 0.43 & 1.74 & 0.87 & 0 \end{bmatrix}$$

## 6 Variables of Mixed Types

→ One approach to calculate dissimilarity between objects of mixed variable types is to group each kind of variable together, performing a separate cluster analysis for each variable type.

→ A more preferable approach is to process all <sup>types of</sup> variables together, performing a single cluster analysis. One such technique: combine the different variables into a single dissimilarity matrix, bringing all of the variables onto a common scale of the interval  $[0.0, 1.0]$

→ Q4 Suppose that the dataset contains P variables of mixed type. The dissimilarity  $d(i,j)$  between objects  $i$  &  $j$  is defined as

$$d(i,j) = \frac{\sum_{f=1}^P s_{ij}(f) d_{ij}(f)}{\sum_{f=1}^P s_{ij}(f)}$$

$f$  represents a variable where the indicator  $s_{ij}(f) = 0$  if either

(1)  $x_{if}$  or  $x_{jf}$  is missing

(2)  $x_{if} = x_{jf} = 0$  and variable  $f$  is asymmetric binary.

Otherwise  $s_{ij}(f) = 1$ .

\* The contribution of variable  $f$  to the dissimilarity between  $i$  and  $j$ , that is  $d_{ij}(f)$  is computed dependent on its type. broad based on

1. If  $f$  is interval based

$$d_{ij}^{(f)} = \frac{(\alpha_{if} - \alpha_{jf})}{\max_h \alpha_{hf} - \min_h \alpha_{hf}}$$

where  $h$  runs over all non missing objects for variable  $f$ .

2. If  $f$  is binary or categorical

$$d_{ij}^{(f)} = 0 \text{ if } \alpha_{if} = \alpha_{jf}$$

$$\text{otherwise } d_{ij}^{(f)} = 1$$

3. If  $f$  is ordinal - Compute the

$$\text{sanks } \alpha_{if} \text{ and } z_{if} = \frac{\alpha_{if} - 1}{M_f - 1} \text{ and}$$

treat  $z_{if}$  as interval-scaled.

4. If  $f$  is ratio scaled, - either

perform logarithmic transformation

and treat the transformed data as interval based or treat  $f$  as

continuous cardinal data, compute

$\alpha_{if}$  and  $z_{if}$  and then treat  $z_{if}$  as interval based.

	Object identifier	test 1 (categorical)	test 2 (ordinal)	test 3 (ratio-scaled)
1	Code A	Excellent	445	
2	Code B	Fair		22
3	Code C	Good		
4	Code A	Excellent	164	1210

Dissimilarity Matrix for test 1

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Dissimilarity matrix for test 2

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

For test 3, logarithmic transformation is applied and we got 2.65, 1.34

2.21 and 3.08

$$\max h \chi_h = 3.08$$

$$\min h \chi_h = 1.34$$

We then normalize the dissimilarity matrix

$$\begin{bmatrix} 0 & (2.65 - 1.34) & & \\ 1.34 & 0 & & \\ 2.21 - 2.65 & 0 & (2.21 - 1.34) & \\ 0.44 & 0.87 & 0 & \\ 0.43 & 1.74 & 0.87 & 0 \end{bmatrix}$$

by dividing by  $3.08 - 1.34 = 1.74$

Finally dissimilarity matrix for test 3 is

$$\begin{bmatrix} 0 & & & & \\ 0.75 & 0 & & & \\ 0.25 & 0.50 & 0 & & \\ 0.25 & 1.00 & 0.50 & 0 & \end{bmatrix}$$

Now we can use these three dissimilarity matrices in

$$d(i,j) = \frac{\sum_{f=1}^P s_{ij}(f) d_{ij}(f)}{\sum_{f=1}^P s_{ij}(f)}$$

$$d(2,1) = \frac{s_{21}(test) \times d_{21}(test)}{s_{21}(test) + s_{21}(test)}$$

$$d(2,1)$$

$$d(2,1) = \frac{1 \times 1 + 1 \times 1 + 1 \times 75}{3}$$

$$= .92$$

$$d(3,1) = \frac{1 \times 1 + 1 \times 5 + 1 \times 25}{3}$$

$$= \frac{1 + 5 + 25}{3}$$

$$= .58.$$

Final Matrix will be

$$\begin{bmatrix} 0 & .92 & .58 \\ .92 & 0 & .08 \\ .58 & .08 & 0 \end{bmatrix}$$

## 7. Vector objects

→ Popular similarity function to compare two vectors  $x$  and  $y$  is cosine measure

$$S(x, y) = \frac{x^t \cdot y}{\|x\| \|y\|}$$

$x^t$  = transposition of vector  $x$

$\|x\|$  &  $\|y\|$ , Euclidean norm  
if  $n = (n_1, n_2, \dots, n_m)$

$$\|n\| = \sqrt{n_1^2 + n_2^2 + \dots + n_m^2}$$

$S$  is the cosine of angle between  $x$  &  $y$ .

eg.:  $x = (1, 1, 0, 0)$   $n = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$   $\|n\| = \sqrt{1^2 + 1^2}$   
 $y = (0, 1, 1, 0)$   $y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$   $\|y\| = \sqrt{0^2 + 1^2 + 1^2 + 0^2} = \sqrt{2}$

$$S(x, y) = \frac{x^t y}{\|x\| \|y\|}$$

$$= \frac{0+1+0+0}{\sqrt{2} \sqrt{2}} \\ = \underline{\underline{.5}}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$1 \times 4$        $4 \times 1$

A simple variable to above measure is

$$S(x, y) = \frac{x^T y}{\sqrt{x^T x} \sqrt{y^T y}}$$

It is called tanimoto coefficient or tanimoto distance.

## Categorization of clustering Methods

### clustering Methods.

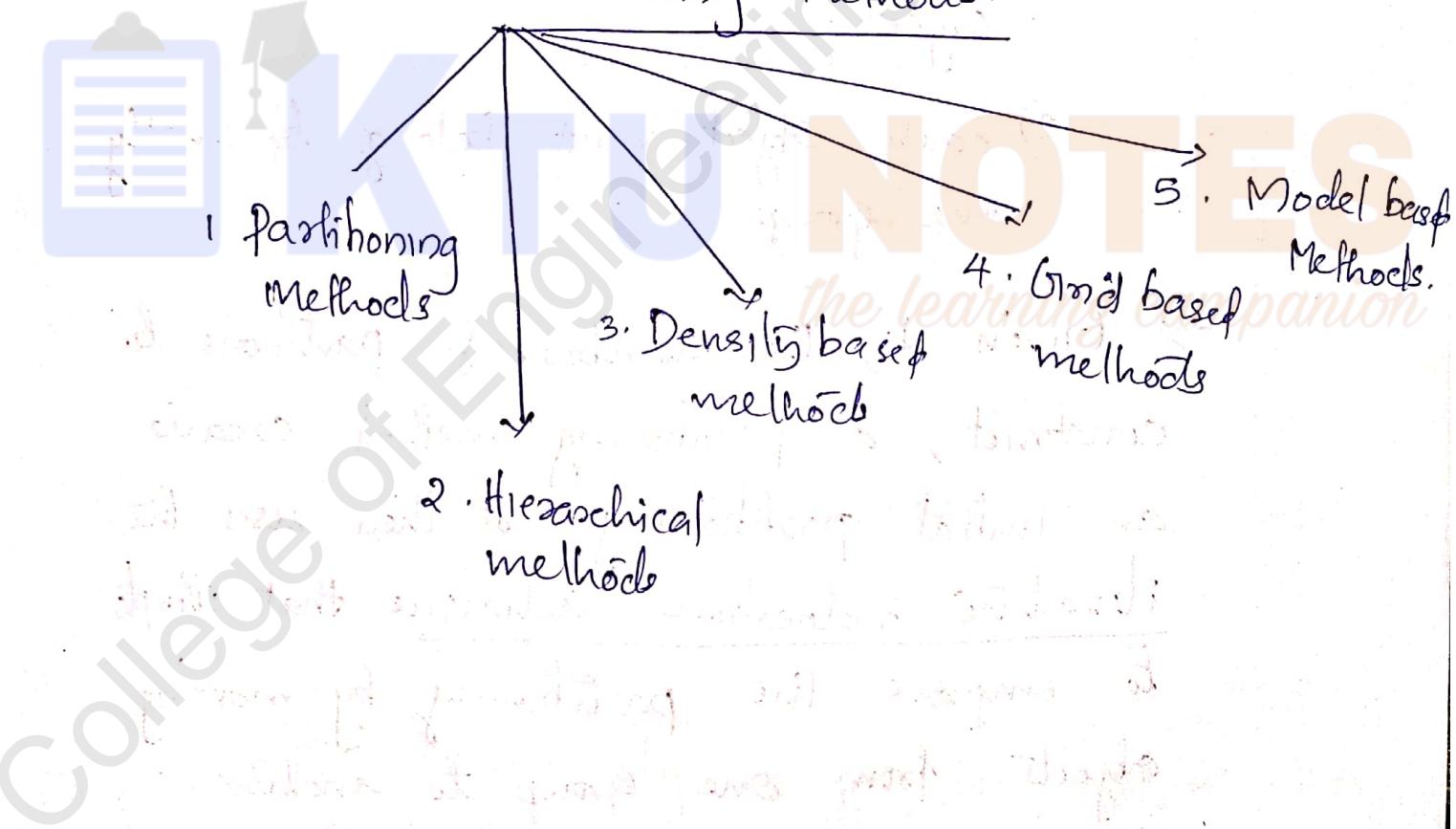
1 Partitioning  
Methods

2. Hierarchical  
methods

3. Density based  
methods

4. Grid based  
methods

5. Model based  
Methods.



## 1. Partitioning Methods.

→ Given a database of  $n$  objects or data tuples, a partitioning method construct  $k$  partitions of the data, where each partition represents a cluster and  $k \leq n$ .

→ That is, it classifies the data into  $k$  groups, which together satisfy the following requirements -

1) Each group must contain at least one object.

2) Each object must belong to exactly one group.

→ Given  $k$ , the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses the iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another.

→ The general criterion of a good partitioning is that objects in the same cluster are close or related to each other whereas objects of different clusters are far apart or very different.

eg: k-means algorithm  
cluster is represented by the mean value of objects in the cluster

k-medoids algorithm

Each cluster is represented by one of the objects located near the center of the cluster.

## 2. Hierarchical Methods-

A hierarchical method creates a hierarchical decomposition of the given set of data objects

Hierarchical methods

Agglomerative approach  
Divisive.

→ Agglomerative approach also known as bottom up approach, starts with each object forming a separate group.

It successively merges the objects or groups that are close to one another until all of the groups are merged into one or a termination condition hold.

→ The divisive approach is also called a top down approach, starts with all of the objects in the same cluster.

In each iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster or until a termination condition holds.

→ Hierarchical methods suffer from the fact that once a step is done, it can never be undone. The such techniques can not correct erroneous decisions.

→ There are two approaches for improving the quality of hierarchical clustering.

1. Perform careful analysis of object linkages at each hierarchical partitioning

2. Integrate hierarchical agglomeration and other approaches by first using a hierarchical agglomeration algm to group objects into microclusters and then performing microclustering on the macroclusters using another methods.

### 3. Density-based Methods

→ Most partitioning methods clusters objects based on their distance between the objects. Such methods can only find spherical shaped clusters.

→ General approach in density based methods is to continue growing the given cluster as long as the density (no of objects or data points) in the neighborhood exceeds a some threshold; that is for each data points within a given cluster, the neighborhood of a given radius has to contain atleast a minimum no of points.

e.g.: DBSCAN, OPTICS, DENCLUE

### 4. Grid-based Methods

Grid based methods quantize the object space into finite no of cells that form a grid structure. All the clustering operations are performed on the grid structure (on quantized space).

The main advantage of this approach is fast processing time, which is independent of the no. of data objects and depends only on the no. of cells in each dimension in the quantized space

eg: STINGER

WaveCluster - both grid based & density based

## 5. Model based Methods

→ Model based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model.

→ A model based algm may locate clusters by constructing a density function that reflects the spatial distribution of the data points.

eg: EM algm - performs expectation - maximization analysis

COBWEB - performs probability analysis

SOM - Self organizing feature map

clustering tasks :- that requires special attention  
are

### clustering high-dimensional data

→ clustering high dimensional data is challenging due to the curse of dimensionality

→ Many dimensions may not be relevant.

As the no. of dimensions increases the data become increasingly sparse so that distance measurement between pairs of points become meaningless and the average density of points anywhere in data is likely to be low.

→ CLIQUE and PROCLUS are two subspace clustering methods, which search for clusters in ~~entire~~ subspaces of data rather than the entire data space.

→ Frequent pattern based clustering, extracts distinct frequent patterns among the subset of dimensions that occurs frequently.  
eg: pcluster

## 2. Constraint based clustering:

- performs clustering by incorporation of user specified or application oriented constraints.
- A constraint expresses a user's expectation or describes properties of the desired clustering results.
- Various kinds of constraints can be specified, either by a user or as per application requirements.

## Partitioning Methods

K-means Method :- Centroid based Technique

- The K-means algorithm takes the input parameter  $k$ , and partitions a set of  $n$  objects into  $k$  clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured in regard to the mean value of

the objects is a cluster, which can be viewed as the cluster's centroid or center of gravity.

### Algorithm:

#### Input

$k$ : the no. of ~~sup~~ clusters

$D$ : Data set containing  $n$  objects

#### Output:

A set of  $k$  clusters

#### Method

(1) arbitrarily choose  $k$  objects from  $D$  as the initial cluster centers.

(2) repeat

(3) (re) assign each object to the cluster to which the object is similar, based on the mean value of the objects in the cluster

(4) update the cluster means; calculate the mean value of the objects for each cluster

(5) Until no change.

First, it randomly selects  $k$  of the objects, each of which initially represents a cluster's mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and cluster mean. It then computes the new mean for each cluster. This process repeats until ~~converges~~ the criterion function converges.

Typically square error criterion is used

$$E = \sum_{i=1}^k \sum_{P \in C_i} \|P - m_i\|^2$$

$P$  = represents an element in cluster  $C_i$ ;  
 $m_i$  = mean of cluster  $C_i$ .

→ Computational Complexity-

$O(nkt)$

$n$  = no of objects

$k$  = no of clusters

$t$  = no of iterations.

## Disadvantages

- can not be applied for data containing categorical attributes.
- Users need to specify the no of clusters in advance.
- not suitable for discovering clusters with nonconvex shapes or clusters of very different size.
- It is sensitive to noise and outlier datapoints.

## Variants of K-means algorithm

1. K-modes Method :- which extends the K-means paradigm to cluster categorical data by replacing means of the data clusters with modes, using new dissimilarity measures to deal with categorical objects and a frequency based to update modes of clusters.
2. EM [Expectation - Maximization] algorithms extend K-means paradigm by assigning each object to clusters according to a weight representing its probability of membership.

→ How can we make k-means algorithm

more scalable?

One approach is

→ 1. Identify 3. kinds of regions in data

1. Regions that are discasable

- An object is discasable

if its membership in a cluster  
is ~~not~~ ascertained

2. Regions that are compressible

- An object that is not  
discasable but it belongs to  
a subcluster

3. Regions that must be retained in

main memory

Objects that are not discasable  
& compressible

\* A data structure known as a clustering  
feature is used to summarize objects  
that have been discarded or ~~compressed~~

\* To achieve scalability, the iterative  
algm only uses clustering feature of  
compressed objects & objects that have to be  
retained in main mly

→ The ~~next~~<sup>another</sup> approach is to first group nearby objects into microclusters and then performs k-means clustering on the microclusters.

### Problems

$$1. A_1(2, 10) \quad A_2(2, 5) \quad A_3(8, 4)$$

$$B_1(5, 8) \quad B_2(7, 5) \quad B_3(6, 4)$$

$$C_1(1, 2) \quad C_2(4, 9)$$

clusters the points into 3 clusters.

Initial centers are  $A_1, B_1$ , &  $C_1$ .

The distance function is Euclidean distance.

$$A_1(2, 10) \quad B_1(5, 8) \quad C_1(1, 2)$$

$$d(A_1, A_2) = \sqrt{0+5^2} = \sqrt{25}$$

$$d(B_1, A_2) = \sqrt{5^2+5^2} = \sqrt{50}$$

$$d(A_1, A_3) = \sqrt{6^2+6^2} = \sqrt{72}$$

$$d(A_1, B_3) = \sqrt{4^2+6^2} = \sqrt{60}$$

$$d(A_1, C_2) = \sqrt{2^2+1^2} = \sqrt{5}$$

$$d(B_1, A_2) = \sqrt{3^2+3^2} = \sqrt{18}$$

$$d(B_1, A_3) = \sqrt{3^2+4^2} = \sqrt{25}$$

$$d(B_1, B_2) = \sqrt{2^2+3^2} = \sqrt{13}$$

$$d(B_1, B_3) = \sqrt{1^2+4^2} = \sqrt{17}$$

$$d(C_1, C_2) = \sqrt{2}$$

$$d(C_1, C_2) = \sqrt{3^2 + 7^2} = \sqrt{58}$$

$$d(C_1, A_2) = \sqrt{1^2 + 3^2} = \sqrt{10}$$

$$d(C_1, A_3) = \sqrt{7^2 + 2^2} = \sqrt{53}$$

$$d(C_1, B_2) = \sqrt{6^2 + 3^2} = \sqrt{45}$$

$$d(C_1, B_3) = \sqrt{5^2 + 2^2} = \sqrt{29}$$

clusters

$$\{A_1\}, \text{ Mean: } (2, 10)$$

$$\{B_1, B_2, A_3, B_3, C_2\}, \text{ Mean: } (6, 6)$$

$$\{C_1, A_2\}, \text{ Mean: } (1.5, 3.5)$$

Final clusters

$$\{A_1, C_2, B_1\}, \text{ Mean: } (2, 10)$$

$$\{A_3, B_2, B_3\}, \text{ Mean: } (6, 6)$$

$$\{C_1, A_2\}, \text{ Mean: } (1.5, 3.5)$$

## K-medoids Method: - Representative Object Based Technique

K-means algorithm is sensitive to outliers. To diminish such sensitivity, instead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters, using one representative object per cluster. Each remaining object is reclustered with the representative object to which it is most similar. The partitioning method is then performed based on the principle of minimizing the sum of dissimilarities between each object and its corresponding reference point. That is, an absolute error criterion is used.

$$E = \sum_{j=1}^k \sum_{p \in C_j} |p - o_j|$$

Medoid : - representative object is a point in the space representing a set, if has smallest average dissimilarity to other objects in it.

$p$  = point in the space representing a given object in cluster  $G_j$

$o_j$  = representative object of  $G_j$

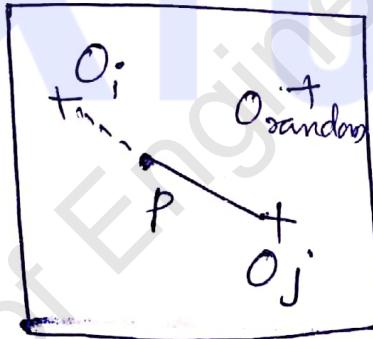
The algorithm iterates until each representative element is actually medoid, or most centrally located object of its cluster. This is the basis of the K-medoids method.

### Working of K-medoid algorithm

- The initial representative objects are chosen arbitrarily.
- The iterative process of replacing representative objects by non representative objects continues as long as quality of resulting cluster is improved.
- The quality is estimated using a cost function that measures the average dissimilarity between an object and the representative object of its cluster.

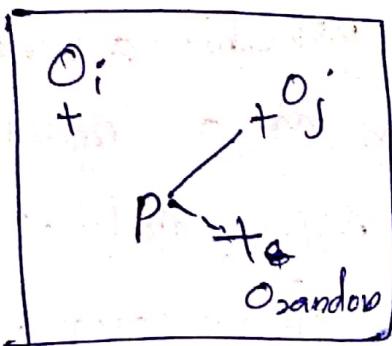
→ To determine whether a nonrepresentative object  $O_{\text{random}}$  is a good replacement for a current representative object  $O_j$ , the following four cases are examined.

Case 1:  $P$  is currently belongs to representative object  $O_j$ . If  $O_j$  is replaced by  $O_{\text{random}}$  as a representative object and  $P$  is closest to one of the other representative object  $O_i$ ,  $i \neq j$ , then  $P$  is reassigned to  $O_i$ .

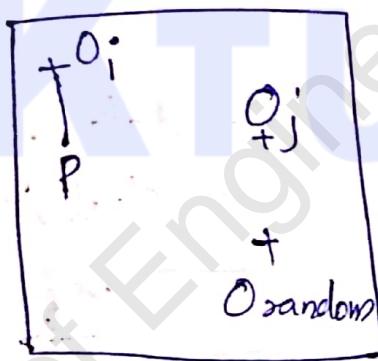


- data object
- + cluster center
- before swapping
- after swapping

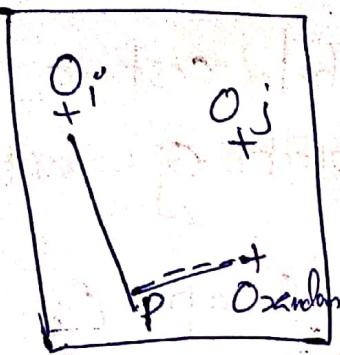
Case 2:  $P$  is currently belongs to representative object  $O_j$ . If  $O_j$  is replaced by  $O_{\text{random}}$  as a representative object and  $P$  is closest to  $O_{\text{random}}$  then  $P$  is reassigned to  $O_{\text{random}}$ .



Case 3:  $P$  is currently belonging to representative object  $O_i$ ,  $i \neq j$ . If  $O_j$  is replaced by  $O_{\text{random}}$  as a representative object and  $P$  is still closest to  $O_i$ , then the assignment does not change.



Case 4:  $P$  is currently belongs to representative object  $O_i$ ,  $i \neq j$ . If  $O_j$  is replaced by  $O_{\text{random}}$  as a representative object and  $P$  is closest to  $O_{\text{random}}$  then  $P$  is reassigned to  $O_{\text{random}}$ .



## PAM (Partitioning around Around Medoids)

→ It is the first k-medoid algm introduced

### Algorithm

#### Input

$k$  : the no of clusters

$D$  : a dataset containing  $n$  objects

Output : A set of  $k$  clusters

#### Method

- (1) arbitrarily choose  $k$  objects in  $D$  as the initial representative objects or seeds.
- (2) repeat
- (3) assign each remaining object to the cluster with the nearest representative object.
- (4) randomly select a nonrepresentative object  $O_{random}$ .

- (5) compute the total costs, of swapping representative object  $O_j$  with  $O_{random}$ .
- (6) if  $S < 0$  then swap  $O_j$  with  $O_{random}$  to form the new set of  $k$  representative objects
- (7) Until no change

If attempts to determine  $K$  partitions for  $n$  objects. After an initial random selection of  $k$  representative objects, the algm repeatedly tries to make a better choice of cluster representatives. All of the possible pairs of objects are analyzed, where one object in each pair is considered a representative object and the other is not. The quality of the resulting clustering is calculated for each such combination. An object  $O_j$  is replaced with object causing the greatest reduction in error.

The set of best objects for each cluster in one iteration forms the representative objects for the next iteration. The final set of representative objects are the representative medoids of the clusters.

Complexity of each iteration -  $O(k(n-k)^2)$

### Advantages of k-medoid algorithm

→ More robust than k-means in the presence of noise and outliers.

#### Drawback

→ Processing is costly.

In both k-means & k-medoids, we need to specify no of clusters k.

# Association Rules Mining

Frequent items, closed items and  
Association Rules.

Let  $I = \{I_1, I_2, \dots, I_m\}$  be a set of items. Let  $D$  be a set of database transactions where each transaction  $T$  is a set of items such that  $T \subseteq I$ .

Each transaction is associated with an identifier  $TID$ . Let  $A$  be a set of items.

A transaction  $T$  is said to contain  $A$  if and only if  $A \subseteq T$ .

→ An association rule is an implication of the form

$$A \Rightarrow B \text{ where } A \subset I$$

$$B \subset I$$

$$A \cap B \neq \emptyset$$

→ The support  $s$  of the association rule

$A \Rightarrow B$  is the percentage of transactions in  $D$  that contain  $A \cup B$ , (both  $A \neq B$ )

This is taken to be the probability  $A \cup B$

→ The confidence  $c$  of the association rule  $A \Rightarrow B$  is the percentage of transactions in  $B$  containing  $A$  that also contain  $B$ . This is taken to be the conditional probability  $P(B|A)$ .

→ Rules that satisfy both minimum support and minimum confidence is called strong.

→ A set of items is referred to as an itemset. An itemset that contains  $k$ -items is a  $k$ -itemset.

→ The occurrence frequency of an itemset is the no. of transactions that contains the itemset.

If it is also known as frequency, support count or count of the itemset.

→  $\text{Support}(A \Rightarrow B) = P(A \cup B)$  ← This is called relative support.

Occurrence frequency is called absolute support.

→ If the relative support of an itemset  $I$  satisfies a prespecified minimum support threshold (i.e., the absolute support of  $I$  satisfies the corresponding minimum support count threshold), then  $I$  is a frequent itemset. The set of frequent  $k$ -itemsets is denoted by  $L_k$ .

$$\text{Confidence } (A \Rightarrow B) = P(B|A)$$

$$= \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$$

$$= \frac{\text{Support-count}(A \cup B)}{\text{Support-count}(A)}$$

→ A typical example of frequent itemset mining is market basket Analysis.

This process analyzes the customer buying habits by finding associations between the different items that customers place in their shopping basket.

"in which items are frequently purchased together"

→ Association rule mining can be viewed as a two step process:

1. Find all frequent itemsets:

2. Generate strong association rules from the frequent itemsets.

→ An itemset  $X$  is closed in a dataset  $S$  if there exists no proper super-itemset  $Y$  such that  $Y$  has the same support count as  $X$  in  $S$ .

→ An itemset  $X$  is a closed frequent itemset in  $S$  if  $X$  is both closed and frequent in  $S$ .

→ An itemset  $X$  is maximal frequent itemset in set  $S$  if  $X$  is frequent and there exists no super itemset  $Y$  such that  $X \subset Y$  and  $Y$  is frequent in  $S$ .

## The Apriori Algorithm: - Finding frequent items using candidate generation

- First the set of frequent 1-items is found by scanning the database to accumulate the count of each item and collecting those items that satisfy minimum threshold. The resulting set is denoted by  $L_1$ .
- $L_1$  is used to find  $L_2$  and so on until no more frequent k-items can be found.
- Finding of each  $L_k$  requires one full scan of the database.
- To improve the efficiency of the level wise generation of frequent items, an important property called Apriori property is used to reduce the search space.

## Apriori

Apriori property: All nonempty subset of a frequent itemset must also be frequent.

i.e. If  $I$  is not frequent, and if an item  $A$  is added to the itemset  $I$ , then  $I \cup A$  is also not frequent.

→ This property belongs to a special category of properties called antimonotone in the sense that if a set cannot pass a test, all of its supersets will fail in the same test as well.

→ How  $L_{k-1}$  is used to find  $L_k$ ,  $k > 2$ .

A two step process is followed.

1. The join step: To find  $L_k$ , a set of candidate  $k$ -itemsets is generated by joining  $L_{k-1}$  with itself. This set of candidates is denoted  $C_k$ .

Let  $l_1$  and  $l_2$  be itemsets in  $L_{k-1}$

The notation  $l_i[j]$  refers to the  $j^{th}$  item in  $l_i$ . Apriori assumes that neither a transaction or elements are sorted in lexicographic order. For  $(k-1)$  element,  $l_i[1] < l_i[2] \dots < l_i[k-1]$ .

The form of  $L_{k-1} \bowtie L_{k-1}$  is performed where members of  $L_{k-1}$  are joinable if their first  $(k-2)$  elements are in common.

If members  $l_1$  and  $l_2$  of  $L_{k-1}$  are joined

$$\begin{aligned} \text{if } (l_1[1] = l_2[1] \wedge l_1[2] \wedge l_2[2] \\ \dots \wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k]) \end{aligned}$$

The resulting itemsets are formed by joining  $l_1$  &  $l_2$  is  $l_1[1], l_1[2], l_1[3] \dots l_1[k-2], l_1[k-1], l_2[k]$

2. The prune step:  $C_k$  is superset of  $L_k$ ; i.e. its members may or may not be frequent! but all the frequent  $k$ -itemsets are included in  $C_k$ . A scan of the database to

determine the count of each candidate.  $C_k$  would result in determination of  $L_k$  [all candidates having count not less than minimum support count are frequent, therefore belong to  $L_k$ ].

$C_k$ , can be huge. To reduce size of  $C_k$ , the a priori property is used.

Any  $k-1$  itemset that is not frequent cannot be a subset of a frequent  $k$ -itemset. Hence if  $(k-1)$  subset of a candidate  $k$ -itemset is not in  $L_{k-1}$ , then it cannot be frequent and can be removed from  $C_k$ .

Eg

TID

List of item-IDs

T<sub>1</sub>

I<sub>1</sub>, I<sub>2</sub>, I<sub>5</sub>

T<sub>2</sub>

I<sub>2</sub>, I<sub>4</sub>

T<sub>3</sub>

I<sub>2</sub>, I<sub>3</sub>

T<sub>4</sub>

I<sub>1</sub>, I<sub>2</sub>, I<sub>4</sub>

T<sub>5</sub>

I<sub>1</sub>, I<sub>3</sub>

T<sub>6</sub>

I<sub>2</sub>, I<sub>3</sub>

T<sub>7</sub>

I<sub>1</sub>, I<sub>3</sub>

T<sub>8</sub>

I<sub>1</sub>, I<sub>2</sub>, I<sub>3</sub>, I<sub>5</sub>

T<sub>9</sub>

I<sub>1</sub>, I<sub>2</sub>, I<sub>3</sub>

In the first iteration, each item is a member of the set of candidate 1-itemset. The algorithm simply scans all of the transactions in order to count no of occurrences of each item.

$C_1$

Itemset	Sup.Count
$\{I_1\}$	6
$\{I_2\}$	7
$\{I_3\}$	6
$\{I_4\}$	2
$\{I_5\}$	2

If min. Then  $L_1$  is formed by selecting candidate 1-itemsets satisfying minimum support (min-sup)

Eg: If the minimum support = 2

Then  $L_1$

Itemset	Sup. count
$\{I_1\}$	6
$\{I_2\}$	7
$\{I_3\}$	6
$\{I_4\}$	2
$\{I_5\}$	2

3. To find frequent 2-itemset  $L_2$ :

$L_1 \bowtie L_p$  is performed to generate a candidate set of 2-itemsets.

$C_2$	itemset
	$\{I_1, I_2\}$
	$\{I_1, I_3\}$
	$\{I_1, I_4\}$
	$\{I_1, I_5\}$
	$\{I_2, I_3\}$
	$\{I_2, I_4\}$
	$\{I_2, I_5\}$
	$\{I_3, I_4\}$
	$\{I_3, I_5\}$
	$\{I_4, I_5\}$

No candidates are removed from  $C_2$  during pruning because each subset of the candidates is also frequent.

4: Transactions in D are scanned and support count of each candidate itemset in  $C_2$  is accumulated.

$C_2$  itemset sup. count

$\{I_1, I_2\}$	4	$\{I_2, I_4\}$	2
$\{I_1, I_3\}$	4	$\{I_2, I_5\}$	2
$\{I_1, I_4\}$	1	$\{I_3, I_4\}$	0
$\{I_1, I_5\}$	2	$\{I_3, I_5\}$	1
$\{I_2, I_3\}$	4	$\{I_4, I_5\}$	0

5. The set of frequent 2-itemsets  $L_2$  is their determined, consisting of those candidate items in  $C_2$  having minimum support.

$L_2$

Itemset	Sup-count
$\{I_1, I_2\}$	4
$\{I_1, I_3\}$	4
$\{I_1, I_5\}$	2
$\{I_2, I_3\}$	4
$\{I_2, I_4\}$	2
$\{I_2, I_5\}$	2

6. Generation of Candidate 3-itemsets.

(a) To do  $L_2 \bowtie L_2$ .

[first  $I_1 = l_1[1] = l_2[1]$ ]

[first  $I_{k-1}$  item should be common]

Here first 1 item should be common

$\{I_1, I_2, I_3\}$

$\{I_1, I_2, I_5\}$

$\{I_1, I_3, I_5\}$

$\{I_2, I_3, I_4\}$

$\{I_2, I_3, I_5\}$

$\{I_2, I_4, I_5\}$

## b) Pause Using Apriori property:

1. The 2-itemset subset of  $\{I_1, I_2, I_3\}$  are  $\{I_1, I_2\}$ ,  $\{I_1, I_3\}$ ,  $\{I_2, I_3\}$ . These subsets are in  $L_2$ .

So  $\{I_1, I_2, I_3\}$  is in  $C_3$

2. The 2-item subset of  $\{I_1, I_2, I_5\}$  are also in  $L_2$

So  $\{I_1, I_2, I_5\}$  is in  $C_3$

3. For the remaining 3-itemsets, this property is not kept.

So they are not included in  $C_3$ .

Finally  $C_3$  is

elements
$\{I_1, I_2, I_3\}$
$\{I_1, I_2, I_5\}$

elements	sup-count
$\{I_1, I_2, I_3\}$	2
$\{I_1, I_2, I_5\}$	2

$C_3$  is formed by selecting candidate 3-itemset having minimum support

elements	sup-count
$\{I_1, I_2, I_3\}$	2
$\{I_1, I_2, I_5\}$	2

8 If  $L_3 \setminus L_3$  is performed to generate  
a candidate  $C_4$  then result is  $\{I_1, I_2, I_3, I_5\}$

The subset  $\{I_2, I_3, I_5\}$  is not in  $L_3$ .

So it is pruned.

So  $C_4 = \emptyset$ , the algorithm terminates.

Alg: Apriori

Input:

$D$ , a database of transactions.

min-sup: - the minimum support count threshold

Method

$L_1$  = find frequent-1-items( $D$ )

for ( $k=2$  ;  $L_{k-1} \neq \emptyset$  ;  $k++$ ) {

$C_k$  = apriori-gen( $L_{k-1}$ );

for each transaction  $t \in D$  { // scan  $D$   
for count

$C_t$  = subset( $C_k, t$ ) // get the subset  
of  $t$  that are candidates)

for each candidate  $c \in C_t$

$c.count++$ ;

}  $L_k = \{ c \in C_k \mid c.count \geq \text{min-sup} \}$

return  $L = \bigcup_k L_k$ .

Procedure apriori\_gen ( $L_{k-1}$ : Frequent  $(k-1)$  elemset)

for each elemset  $l_1 \in L_{k-1}$

for each elemset  $l_2 \in L_{k-1}$

if ( $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \dots$

$\wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k-1]$ )

{

$c = l_1 \bowtie l_2$

if has-infrequent-subset ( $c, L_{k-1}$ ) then

delete  $c$  // prune step

else add  $c$  to  $C_k$ ;

return  $C_k$ ;

Procedure has-infrequent-subset ( $c$ : candidate  $k$ -itemset)

$L_{k-1}$ : frequent  $(k-1)$  elemset); // use prior knowledge

for each  $(k-1)$  subset  $s$  of  $c$

if  $s \notin L_{k-1}$  then

return TRUE

else

return FALSE

## → Generating Association Rules from frequent itemsets

Association rules are generated as follows.

1. For each frequent itemset  $l$ , generate all non-empty subset of  $l$

2. For every non empty subset  $s$  of  $l$ ,

output rule  $s \rightarrow (l - s)$ ,  $\frac{\text{support}(s)}{\text{support}(l)}$

min-conf; where min-conf is the minimum confidence threshold.

e.g.  $\{I_1, I_2, I_3\}$ ,  $\{I_1, I_2, I_5\}$  are two frequent itemsets.

Take  $\{I_1, I_2, I_5\}$

Subset are

$\{I_1, I_2\}$ ,  $\{I_1, I_5\}$ ,  $\{I_2, I_5\}$

$\{I_1\}$ ,  $\{I_2\}$ ,  $\{I_5\}$

s  $I_1 \wedge I_2$   $\Rightarrow I_5$  Confidenc =  $2/4 = 50\%$

$$I_1 \wedge I_5 \Rightarrow I_2 = \frac{2}{2} = 100\%$$

$$I_2 \cap I_5 \Rightarrow I_1 = 2/2 = 100\%$$

$$I_1 \Rightarrow I_2 \wedge I_5$$

$$I_2 \Rightarrow I_1 \cap I_5 = 2/7 = 29\%$$

$$I_5 \Rightarrow I_1 \cap I_2 \Rightarrow 2/2 = 100\%.$$

i.f. minimum confidence threshold is 70%.

There are only three entries are selected

$$I_1 \wedge I_5 \Rightarrow I_2$$

$$I_2 \cap I_5 \Rightarrow I_1$$

$$I_S \Rightarrow I_1 \wedge I_2$$

Take  $\{I_1, I_2, I_3\}$ .

Subsets are  $\{\{I_1\}, \emptyset, \{I_2\}, \{I_3\}, \{I_1, I_2\}\}$

$$\{I_2, I_3\}, \{I_1, I_3\}.$$

$$I_4 \Rightarrow I_2 \wedge I_3 = 2/6 = 33\%$$

$$I_2 \Rightarrow I_1 \wedge I_3 = 2/7 = 29\%. \quad \text{no ends}$$

$$I_3 \Rightarrow I_1 \cap I_2 = 2/6 = 33\% \text{ are selected}$$

$$I_1, I_2 \Rightarrow I_3 = 2 \mid 4 = 50\%$$

$$\frac{I_1}{I_1 + I_2} \Rightarrow I_2 = 2/4 = 50\%$$

$$I_2 \cap I_3 \Rightarrow I_1 = 2 \mid 4 \quad \approx 50\%$$

## Mining Frequent Patterns [Items]

### without Candidate Generation

The main disadvantages of Apriori algm are

1. It may need to generate a huge no of candidate sets
2. It may need to repeatedly scan the database and check a large set of candidates by pattern matching.

## FP growth [Frequent Pattern Growth]

algm mines the complete set of frequent items without candidate generation.

### FP Growth Algorithm

- It adopts a divide and conquer strategy.

- First it compresses the database representing frequent items into a frequent pattern tree or FP tree, which retains the itemset association information.
- Then divides the compressed database into a set of conditional databases, each associated with one frequent item or pattern fragment and mines such database separately.

Eg:

TID

$T_1$

$T_2$

$T_3$

$T_4$

$T_5$

$T_6$

$T_7$

$T_8$

$T_9$

List of item-IDs

$I_1, I_2, I_5$

$I_2, I_4$

$I_2, I_3$

$I_1, I_2, I_4$

$I_1, I_3$

$I_2, I_3$

$I_1, I_3$

$I_1, I_2, I_3, I_5$

$I_1, I_2, I_3$

Find frequent  
itemsets?

Minimum  
Support Count  
is 2

→ Scan the database and find set of frequent 1-itemsets and their support count

→ The set of frequent itemsets are sorted in the order of descending support count.

→ The resulting set as list is denoted by L

$$L = \{ \{I_2: 7\}, \{I_1: 6\}, \{I_3: 6\}, \\ \{I_4: 2\}, \{I_5: 2\} \}$$

→ Then FP tree is constructed.

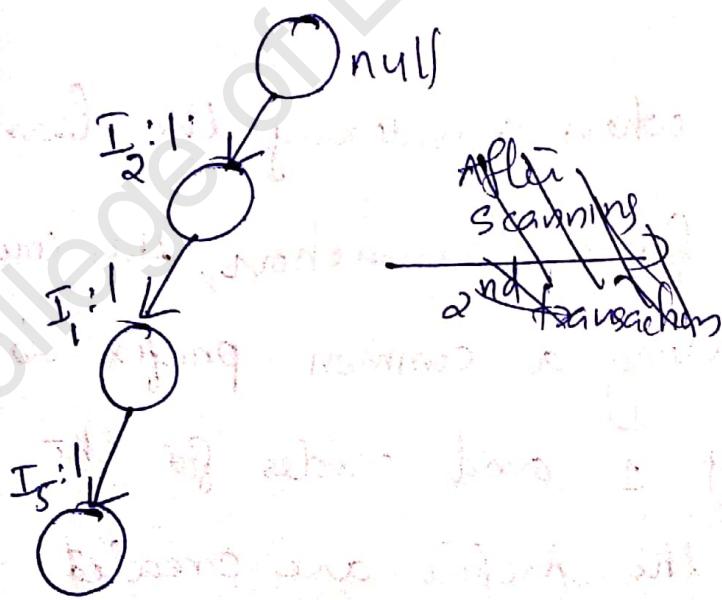
• First create the root of the tree labeled with null

○ null.

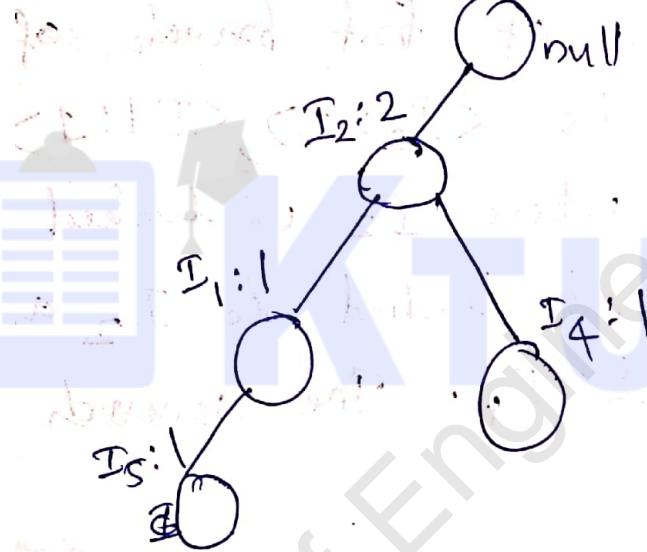
- Scan the database D a second time.
- The items in each transaction are processed in L order [sorted according to descending support count] and a branch is created for each transaction.

e.g. T1: I<sub>1</sub>, I<sub>2</sub>, I<sub>5</sub> which contains

I<sub>2</sub>, I<sub>1</sub> & I<sub>5</sub> in L order leads to the construction of first branch of tree with nodes  $\langle I_2 : 1 \rangle$ ,  $\langle I_1 : 1 \rangle$  and  $\langle I_5 : 1 \rangle$  where  $I_2$  is linked to the root,  $I_1$  is linked to  $I_2$  and  $I_5$  is linked to  $I_1$ . The second transaction

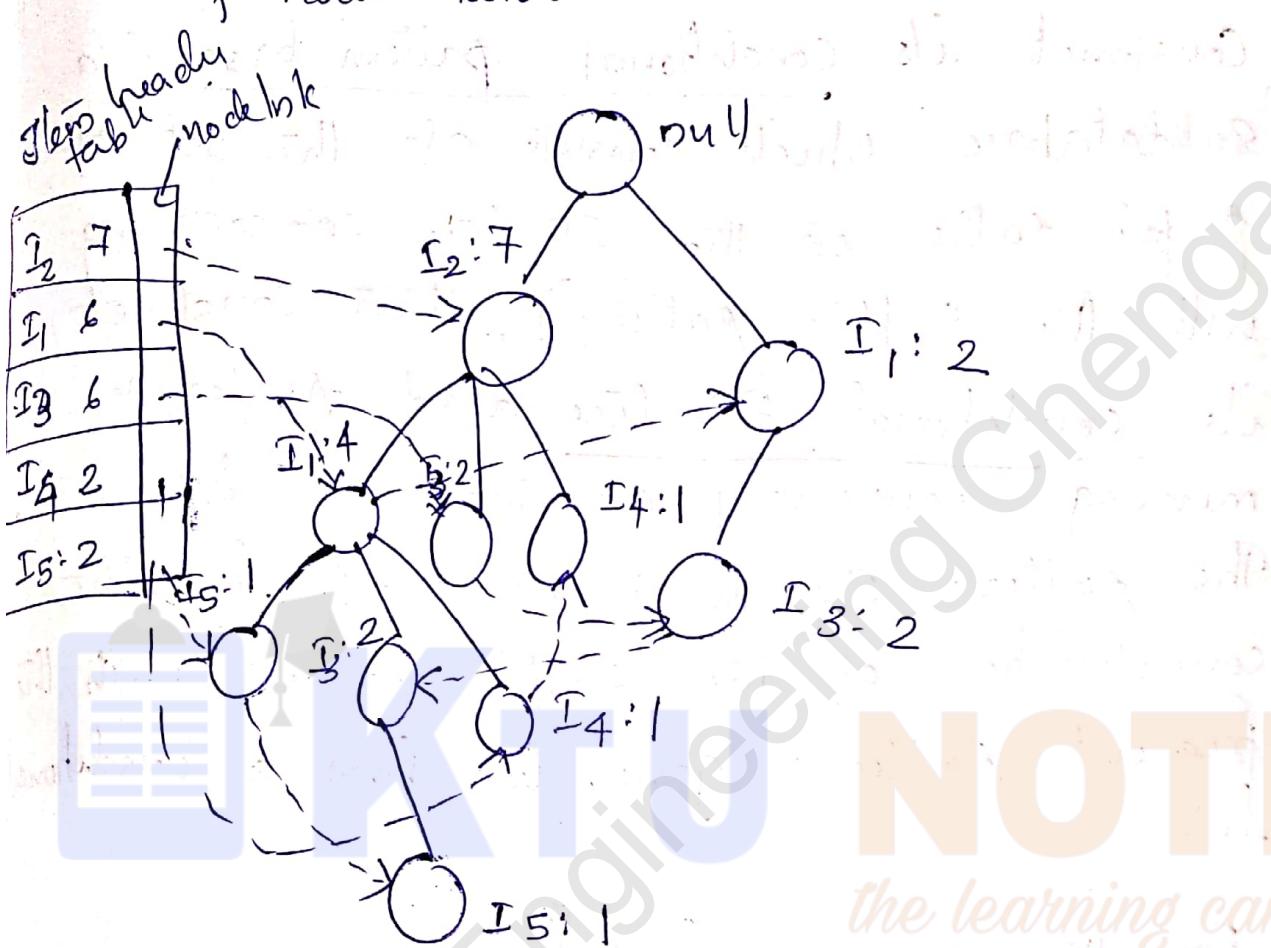


The second transaction  $T_2$ , contains  $I_2$  and  $I_4$  in L order which would result in a branch where  $I_2$  is linked to the root and  $I_4$  is linked to  $I_2$ . This branch share a common prefix  $I_2$ , with the path for  $T_1$ . Therefore instead increment count of  $I_2$  node and create new node  $\langle I_4 : 1 \rangle$  which is linked to  $I_2$ .



In general when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1 and nodes for the items following the prefix are created and linked accordingly.

To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node links.



In this way, the problem of mining frequent patterns in databases is transformed to that of mining FP-tree.

The FP tree is mined as follows:

Start from each frequent length-1

Pattern [ item with minimum support count ]

Construct its conditional pattern base (a subdatabase which consists of the set of prefix paths in the FP tree co-occurring with the suffix pattern), then construct its conditional FP tree and perform mining recursively on such a tree.

The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional tree.

Eg: From the previous FP tree.

We first consider I5. I5 occurs in two branches of FP tree.

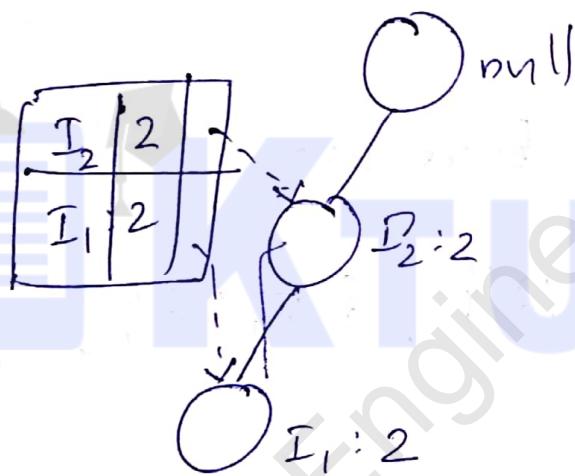
The path followed formed by these branches are  $\langle I2, I1, I5:1 \rangle$  and  $\langle I2: I1: I3: I5:1 \rangle$

Considering I5 as suffix, its corresponding frequent paths are  $\langle I2, I1 : 1 \rangle$  and  $\langle I2, I1, I3 : 1 \rangle$  which forms its conditional pattern base.

Then construct conditional FP tree.

<u>Support count</u>	
I <sub>2</sub>	2
I <sub>1</sub>	2
<del>I<sub>3</sub></del>	

~~I<sub>3</sub>~~ → If it is eliminated because support count is less than minimum support count.



Conditional FP tree >

$\langle I2: 2, I1: 2 \rangle$

The single path generates all the combination of frequent patterns:

$\{I2, I5: 2\}, \{I1, I5: 2\}, \{I_d, I_2\}$   
 $\{I2, I1, I5: 2\}$

Causes tree  $\langle S: 1 \rangle$  &  $\langle I \rangle$

For I4,

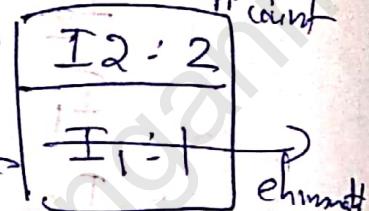
The prefix path form the conditional pattern base

$$\{ \{ I_2, I_1 : 1 \}, \{ I_2 : 1 \} \}$$

which a single node conditional

FP tree  $\langle I_2 : 2 \rangle$  and derive

one frequent pattern,  $\{ I_2, I_4 : 2 \}$



For I3

Conditional pattern bases are

$$\{ \{ I_2 : 2 \}, \{ I_2, I_1 : 2 \}, \{ I_1 : 2 \} \}$$



From same

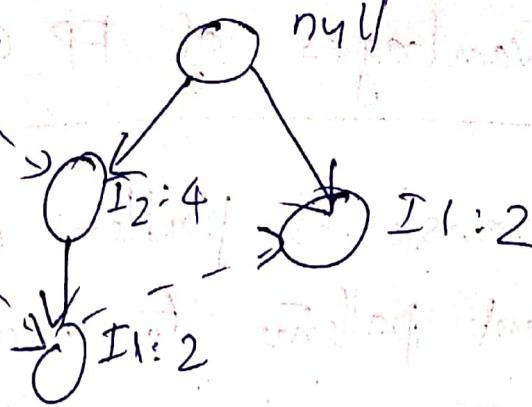
branch of

FP tree

Here the conditional FP tree contains two branches.

$\langle I_2 : 4, I_1 : 2 \rangle$  and  $\langle I_1 : 2 \rangle$

I <sub>2</sub>	4
I <sub>1</sub>	4



which generates a set of patterns

$$\{ \{I_2, I_3:4\}, \{I_1, I_3:4\}, \{I_2 \cup I_1, I_3:2\} \}$$

For I<sub>1</sub>, the frequent pattern is I<sub>1</sub>:2.

Condition base is {I<sub>2</sub>:4} whose FP tree contains only one node <I<sub>2</sub>:4>, which generates frequent pattern {I<sub>2</sub>, I<sub>1</sub>:4}

Summary:

Item	Conditional Pattern base	conditional FP tree	Frequent patterns Generated
I <sub>5</sub>	{ {I <sub>2</sub> , I <sub>1</sub> :1}, {I <sub>2</sub> , I <sub>1</sub> , I <sub>3</sub> :1} }	<I <sub>2</sub> :2, I <sub>1</sub> :2>	{ {I <sub>2</sub> , I <sub>5</sub> :2}, {I <sub>1</sub> , I <sub>5</sub> :2} }
I <sub>4</sub>	{ {I <sub>2</sub> , I <sub>1</sub> :1}, {I <sub>2</sub> , I <sub>3</sub> } }	<I <sub>2</sub> :2>	{ I <sub>2</sub> , I <sub>4</sub> :2 }
I <sub>3</sub>	{ {I <sub>2</sub> , I <sub>1</sub> :2}, {I <sub>2</sub> :2}, {I <sub>1</sub> :2} }	<I <sub>2</sub> :4, I <sub>1</sub> :2> <I <sub>1</sub> :2>	{ I <sub>2</sub> , I <sub>3</sub> :4 }
I <sub>1</sub>	{ {I <sub>2</sub> :4} }	& I <sub>2</sub> :4	{ I <sub>1</sub> , I <sub>3</sub> :4 } { I <sub>2</sub> , I <sub>1</sub> , I <sub>3</sub> :2 } { I <sub>2</sub> , I <sub>1</sub> :4 }

## Advantages of FP Growth Algo

- It transforms problem of finding long frequent patterns to searching for shorter ones recursively and then concatenating suffix
- It is efficient and scalable for mining both long and short frequent patterns
- It is faster than Apriori algorithm
- When database is large, and it is not possible to consume main memory based FP tree, first partition the database into set of projected database and construct FP tree and mine patterns in each projected database.