STUDY OF VARIOUS DATA MINING TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

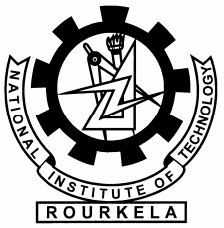
in

Computer Science and Engineering

By

JNANARANJAN MALLICK

MANMOHAN TUDU



Department of Computer Science Engineering

National Institute of Technology

Rourkela

2007

STUDY OF VARIOUS DATA MINING TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Computer Science and Engineering

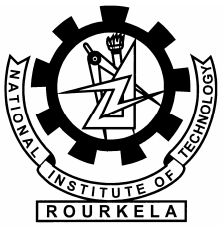
By

JNANARANJAN MALLICK

MANMOHAN TUDU

Under the Guidance of

Prof. S.K. Rath



Department of Computer Science Engineering

National Institute of Technology

Rourkela

2007

2007

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "Study of various data mining techniques" submitted by Jnanaranjan Mallick and Manmohan Tudu in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Computer Science Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. S. K. Rath
Dept. of Computer Science Engineering
National Institute of Technology
Rourkela - 769008

ACKNOWLEDGEMENT

We would like to articulate our deep gratitude to our project guide Prof. Rath who has always been our motivation for carrying out the project. It is our pleasure to refer Microsoft Word exclusive of which the compilation of this report would have been impossible. An assemblage of this nature could never have been attempted with out reference to and inspiration from the works of others whose details are mentioned in reference section. We acknowledge out indebtedness to all of them. Last but not the least our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.

JNANARANJAN MALLICK
ROLL NO. : 10306024

MANMOHAN TUDU
ROLL NO. : 10306011

CONTENTS

Chapter 1

INTRODUCTION

INTRODUCTION:

The advent of computing technology has significantly influenced our lives and two major impacts of this effect are Business Data Processing and Scientific Computing. During the initial years of the development of computer techniques for business, computer professionals were concerned with designing files to store the data so that information could be efficiently retrieved. There were restrictions on storage size for storing data and on speed of accessing the data. Needless to say, the activity was restricted to a very few, highly qualified professionals. Then came the era when the task was simplified by a DBMS [1]. The responsibilities of intricate tasks, such as declarative aspects of the program were passed on to the database administrator and the user could pose his query in simpler languages such as query languages. Thus, due to widespread computerization and also due to affordable storage facilities, there is a enormous wealth of information embedded in huge databases belonging to different enterprises. Every organization is now accumulating a large volume of daily transaction data and storing them in archival files.

Business was inherently competitive and in this competitive world of business one is constantly on the lookout for ways to beat the competition. A question that naturally arose is whether the enormous data that is generated and stored as archives can be used for improving the efficiency of business performance.

Such collections of data, whether their origin is business enterprise or scientific experiment, have recently spurred a tremendous interest in the areas of knowledge discovery and data mining. Statistians and data miners now have faster analysis tools that can help sift and analyze the stockpiles of data, turning up valuable and often surprising information. As a result, a new discipline in Computer Science, Data Mining, gradually evolved. Data mining is the exploration and analysis of large data sets, in order to discover meaningful patterns and rules. The idea is to find effective ways to combine the computer's power to process the data with the human eye's ability to detect patterns. The techniques of data mining are designed for, and work best with, large data sets.

Chapter 2

DATA MINING BASICS

2.1 Definition

Data mining, the extraction of the hidden predictive information from large databases, is a powerful new technology with great potential to analyze information in the data warehouse. Data mining scours databases for hidden patterns, finding predictive information that experts may miss, as it goes beyond their expectations. When implemented on a high performance client/server or parallel processing computers, data mining tools can analyze massive databases to deliver answer to questions such as which clients are most likely to respond to the next promotional mailing. There is an increase in the desire to use this new technology in the new application domain, and a growing perception that these large passive databases can be made into useful actionable information.

The term 'data mining' refers to the finding of relevant and useful information from databases. Researchers have defined 'data mining' in many ways. Some are mentioned below:

1. *Data mining or knowledge discovery in databases, as it is also known, is the non-trivial extraction of implicit, previously unknown and potentially useful information from the data. This encompasses a number of technical approaches, such as clustering, data summarization, classification, finding dependency networks, analyzing changes, and detecting anomalies.*
2. *Data mining refers to using a variety of techniques to identify nuggets of information or decision-making knowledge in the database and extracting these in such a way that they can be put use in areas such as decision support, prediction, forecasting and estimation. The data is often voluminous, but it has low value and no direct use can be made of it. It is the hidden information in the database that is useful.*
3. *Data mining is the process of discovering meaningful, new correlation patterns and trends by sifting through large amount of data stored in repositories, using pattern recognition techniques as well as statistical and mathematical techniques.*

2.2 Data Mining

The wide-spread use of distributed information systems leads to the construction of large data collections in business, science and on the Web. These data collections contain a wealth of information, which however needs to be discovered. Businesses can learn from their transaction data more about the behavior of their customers and therefore can improve their business by exploiting this knowledge. Science can obtain from observational data (e.g. satellite data) new insights on research questions. Web usage information can be analyzed and exploited to optimize information access. Data mining provides methods that allow to extract from large data collections unknown relationships among the data items that are useful for decision making. Thus data mining generates novel, unsuspected interpretations of data.

In practice the two fundamental goals of data mining tend to be: *prediction* and *description.*

1. Prediction makes use of existing variables in the database in order to predict unknown or future values of interest
2. Description focuses on finding patterns describing the data and the subsequent presentation for user interpretation.

The relative emphasis of both prediction and description differ with respect to the underlying application and the technique. There are several data mining techniques fulfilling these objectives. Some of these are associations, classifications, and clustering.

2.2.1 DISCOVERY OF ASSOCIATION RULES

An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are the sets of items. The intuitive meaning of such a rule is that the transaction of the database which contains X tends to contain Y. Given a database, the goal is to discover all the rules that have the support and confidence greater than or equal to the minimum support and confidence, respectively.

Let $L = \{l_1, l_2, ..., l_m\}$ be a set of literals called items. Let D, the database, be a set of transactions, where each transaction T is a set of items. T supports and item x, if x is in T.

T is said to support a subset of items *X*, if *T* supports each item *x* in *X*. $X \Rightarrow Y$ holds with confidence c, if c% of the transactions in *D* if s% of the transactions in *D* support $X \cup Y$. *Support* means how often *X* and *Y* occur together as a percentage of the total transactions. *Confidence* measures how much a particular item is dependent on another.

Thus, the association with a very high support and confidence is a pattern that occurs often in the database that should be obvious to the end user. Patterns with extremely low support and confidence should be regarded as of no significance. Only patterns with a combination of intermediate values of confidence and support provide the user with interesting and previously unknown information.

2.2.2 CLUSTERING

Clustering is a method of grouping data into different groups, so that the data in each group share similar trends and patterns. Clustering constitutes a major class of data mining algorithms. The algorithm attempts to automatically partition the data space into a set of regions or clusters, to which the examples in the table are assigned, either deterministically or probability-wise. The goal of the process is to identify all sets of similar examples in the data, in some optimal fashion.

Clustering according to similarity is a concept which appears in many disciplines. If a measure of similarity is available, then there are a number of techniques for forming clusters. Another approach is to build set functions that measure some particular property of groups. This latter approach achieves what is known as optional partitioning.

The objectives of clustering are:

- To uncover natural groupings
- To initiate hypothesis about the data
- To find consistent and valid organization of the data.

A retailer may want to know where similarities exist in his customer base, so that he can create and understand different groups. He can use the existing database of the different customers or more specifically, different transactions collected over a period of time. The clustering methods will help him in identifying different categories of customers. During the discovery process, the differences between data sets can be discovered in order to

separate them into different groups, and similarity between data sets can be used to group similar data together.

2.2.3 DISCOVERY OF CLASSIFICATION RULES

Classification involves finding rules that partition the data into disjoint groups. The input for the classification is the training data set, whose class labels are already known. Classification analyzes the training data set and constructs a model based on the class label, and aims to assign a class label to the future unlabelled records. Since the class field is known, this type of classification is known as supervised learning. A set of classification rules are generated by such a classification process, which can be used to classify future data and develop a better understanding of each class in the database. We can term this as *supervised learning* too.

There are several classification discovery models. They are: the decision tress, neural networks, genetic algorithms and the statistical models like linear/geometric discriminates. The applications include the credit card analysis, banking, medical applications and the like. Consider the following example.

The domestic flights in our country were at one time only operated by Indian Airlines. Recently, many other private airlines began their operations for domestic travel. Some of the customers of Indian Airlines started flying with these private airlines and, as a result, Indian Airlines lost these customers. Let us assume that Indian Airlines wants to understand why some customers remain loyal while others leave. Ultimately, the airline wants to predict which customers it is most likely to lose its competitors. Their aim to build a model based on the historical data of loyal customers versus customers who have left. This becomes a classification problem. It is a supervised learning task as the historical data becomes the training set which is used to train the model. The decision tree is the most popular classification technique.

Chapter 3

ASSOCIATION RULES

3.1 INTRODUCTION

Among the areas of data mining, the problem of deriving associations from data has received a great deal of attention. The problem was formulated by Agrawal *et al.* in 1993 and is often referred to as the *market-basket problem.* In this problem, we are given a set of items and a large collection of transactions, which are subsets (baskets) of these items. The task is to find relationships between the presences of various items within these baskets.

There are numerous applications of data mining which fit to this framework. The classic example from which the problem gets its name is the supermarket. In this context, the problem is to analyze customers' buying habits by finding associations between the different items that customers place in their shopping baskets. The discovery of such association rules can help the retailer develop marketing strategies. It also helps inventory management, sale promotion strategies, etc.

3.2 WHAT IS AN ASSOCIATION RULE?

In this section, the basic definitions and concepts of the association rule are introduced.

Let $A = \{l_1, l_2, \ldots\ldots, l_m\}$ be a set of items. Let *T,* the transaction database, be a set of transactions, where each transaction *t* is a set of items. Thus, *t* is a subset of *A*.

3.2.1 SUPPORT

A transaction *t* is said to support an item l_i, if l_i is present in *t. t* is said to support a subset of items $X \subseteq A$, if *t* supports each item *l* in X. An itemset $X \subseteq A$ has a support *s* in *T*, denoted by $s(X)_T$, if *s%* of transactions in *T* support *X.* In other words *Support* means how often *X* and *Y* occur together as a percentage of the total transactions.

3.2.2 CONFIDENCE

Confidence measures how much a particular item is dependent on another.

3.2.3 ASSOCIATION RULE

For a given transaction database *T*, an *association rule* is an expression of the form $X \Rightarrow Y$, where *X* and *Y* are subsets of *A* and $X \Rightarrow Y$ holds with confidence τ, if $\tau\%$ of transactions in D that support *X* also support *Y*. The rule $X \Rightarrow Y$ has support σ in the transaction set *T* if $\sigma\%$ of transactions in *T* support $X \cup Y$.

3.3 METHODS TO DISCOVER ASSOCIATION RULES

The discovery of association rules is the most well studied problem in data mining. There are many interesting algorithms proposed recently. One of the key features of all algorithms is that each of these methods assume that the underlying database size is enormous and they require multiple passes over the database.

3.2.1 PROBLEM DECOMPOSITION

The problem of mining association rules can be decomposed into two sub-problems:

- Find all sets of items (*itemsets*) whose support is greater than the user-specified minimum support, σ. Such itemsets are called *frequent* itemsets.
- Use the frequent itemsets to generate the desired rules.

3.2.2 DEFINITIONS

FREQUENT SET: Let T be the transaction database and σ be the user-specified minimum support. An itemset $X \subseteq A$ is said to be frequent set in T with respect to σ, if $s(X)_T \geq \sigma$.

DOWNWARD CLOSURE PROPERTY: Any subset of a frequent set is a frequent set.

UPWARD CLOSURE PROPERTY: Any superset of an infrequent set is an infrequent set.

MAXIMAL FRQUENT SET: A frequent set is a maximal frequent set if it is a frequent set and no superset of this is a frequent set.

BORDER SET: An itemset is a border set if it is not a frequent set, but all its proper subsets are frequent sets.

3.3 A PRIORI ALGORITHM

A Priori algorithm [1] is also called the *level-wise* algorithm. It was proposed by Agrawal and Srikant in 1994. It is the most popular algorithm to find all the frequent sets. It makes use of the downward closure property.

The first pass of the algorithm simply counts item occurrence to determine the frequent sets. A subsequent pass, say pass k, consists of two phases. First, the frequent itemsets L_{k-1} found in the $(k-1)th$ pass are used to generate the candidate itemsets C_k, using the candidate generation procedure. Next, the database is scanned and the support

of candidates in C_k is counted. The set of candidate itemsets is subjected to a pruning process to ensure that all the subsets of the candidate sets are already known to be frequent itemsets.

3.4.1 CANDIDATE GENERATION

Given L_{k-1}, the set of all frequent (k-1)-itemsets, we want to generate a superset of the set of all frequent k-itemsets. The intuition behind a priori candidate-generation procedure is that if an itemset *X* has a minimum support, so do all subsets of *X*.

$gen\_candidate\_itemsets$ with the given L_{k-1} as follows:

$C_k = \varnothing$

for all itemsets $l_1 \in L_{k-1}$ *do*

for all itemsets $l_2 \in L_{k-1}$ *do*

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

then $c = l_1[1], l_1[2] .... l_1[k-1], l_2[k-1]$

$C_k = C_k \cup \{c\}$

3.4.2 PRUNNING

The pruning step eliminates the extensions of (k-1)-itemsets which are not found to be frequent, from being considered for counting support.

$prune(C_k)$

for all $c \in C_k$

for all $(k-1) - subsets$ d *of* c *do*

if $d \notin L_{k-1}$

then $C_k = C_k \setminus \{c\}$

A Priori Algorithm

initialize $k := 1, C_1 =$ all the 1-itemsets:

read teh database to count the support of C_1 to determine L_1.

$L_1 := \{frequent\ 1-itemsets\}$;

$k := 2$ // represents the pass number //

while$(L_{k-1} \neq \phi)$ *do*

begin

$C_k := gen\_candidate\_itemsets$ with the given L_{k-1}

prune(C_k)

for all transactions $t \in T$ *do*

increment the count of all the candidates in C_k that are contained in t;

$L_k :=$ all candidates in C_k with minimum support;

$k := k + 1$;

end

answer:= $\cup_k L_k$;

EXAMPLE:

A1	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**
1	0	0	0	1	0	0	1	0
0	1	0	1	0	0	0	1	0
0	0	0	1	1	0	1	0	0
0	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0
0	1	1	1	0	0	0	0	0
0	1	0	0	0	1	1	0	1
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	0	0	0	1	1	0	1	0
0	1	0	1	0	1	1	0	0
1	0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	0	1

FIG. 1 SAMPLE DATABASE

We illustrate the working of the algorithm with the above transaction database with A = {A1, A2, A3, A4, A5, A6, A7, A8, A9} and σ = 20%. Since T contains 15 records, it means that an itemset that is supported by at least three transactions is a frequent set.

k := 1

Read the database to count the support of 1-itemsets (FIG. 1). The frequent 1-itemsets and their support count is given below.

{1}	2
{2}	6
{3}	6
{4}	4
{5}	8
{6}	5
{7}	7
{8}	4
{9}	2

FIG. 2 SUPPORT COUNT

$L_1 := \{\{2\} \to 6.\{3\} \to 6, \{4\} \to 4, \{5\} \to 8, \{6\} \to 5, \{7\} \to 7, \{8\} \to 4\}.$

k := 2

In the candidate generation step, we get

$C_2 := \{\{2,3\},\{2,4\},\{2,5\},\{2,6\},\{2,7\},\{2,8\},\{3,4\},\{3,5\},\{3,6\},\{3,7\},\{3,8\},\{4,5\},\{4,6\},\{4,7\},$
$\{4,8\},\{5,6\},\{5,7\},\{5,8\},\{6,7\},\{6,8\},\{7,8\}\}$

The pruning step doesn't change C_2.

Read the database to get the support count of elements in C_2 to get

$L_2 := \{\{2,3\} \to 3, \{2,4\} \to 3, \{3,5\} \to 3, \{3,7\} \to 3, \{5,6\} \to 3, \{5,7\} \to 5, \{6,7\} \to 3\}$

k := 3

In the candidate generation step,

Using {2, 3} and {2, 4}, we get {2, 3, 4}

Using {3, 5} and {3, 7}, we get {3, 5, 7} and

Similarly from {5, 6} and {5, 7}, we get {5, 6, 7}.

$C_3 := \{\{2,3,4\},\{3,5,7\},\{5,6,7\}\}.$

The pruning step prunes {2, 3, 4} as not all the subsets of size 2, i.e., {2, 3}, {2, 4} and {3, 4} are present in L_2. The other two itemsets are retained.

Thus the pruned C_3 is {{3, 5, 7}, {5, 6, 7}}.

Read the database to count the support of the itemsets in C_3 to get

$L_3 := \{\{3,5,7\} \to 3\}.$

k := 4

Since L_3 contains only one element, C_4 is empty and hence the algorithm stops, returning the asset of frequent sets along with their respective support values as

$L := L_1 \cup L_2 \cup L_3$

3.5 PARTION ALGORITHM

The partition algorithm [1] is based on the observation that the frequent sets are normally very few in number compared to the set of all itemsets. As a result, if we partition the set of transactions to smaller segments such that each segment can be accommodated in the main memory, then we can compute the set of frequent sets of each of these partitions. It is assumed that these sets (set of local frequent sets) contain a reasonably small number of itemsets. Hence, we can read the whole database (the unsegmented one) once, to count the support of the set of all local frequency sets.

The partition algorithm uses two scans of the database to discover all frequent sets. In one scan, it generates a set of all potentially frequent itemsets by scanning the database once. This set is a superset of all frequent itemsets, i.e., it may contain false positives; but no false negatives are reported. During the second scan, counters for each of these itemsets are set up and their actual support is measured in one scan of the database.

The algorithm executes in two phases. In the first phase, the partition algorithm logically divides the database into a number of non-overlapping partitions. The partitions are considered one at a time and all frequent itemsets for the partition are generated. Thus, if there are *n* partitions, phase I of the algorithm takes *n* iterations. At the end of phase I, these frequent itemsets are merged to generate a set of all potential frequent itemsets. In this step, the local frequent itemsets of same lengths from all *n* partitions are combined to generate the global candidate itemsets. In phase II, the actual support for these item sets are generated and the frequent itemsets are identified. The algorithm reads the entire database once during phase I and once during phase II. The partition sizes are chosen such that each partition can be accommodated in the main memory, so that the partitions are read only once in each phase.

Partition Algorithm

P = partiton_database(T); n = Number of partitions

// Phase I

for $i = 1$ *to n do begin*

read in partition(T_i *in* P)

L^i = generate all frequent itemsets of T_i using a priori method in main memory.

end

//Merge Phase

for$(k = 2; L_k^i \neq \varnothing, i = 1, 2, ..., n; k++)$ *do begin*

$$C_k^G = \bigcup_{i=1}^{n} L_i^k$$

end

//Phase II

for $i = 1$ *to n do begin*

read _in _ partiton(T_i *in* P)

for all candidates $c \in C^G$ compute $s(c)_{T_i}$

end

$L^G = \{c \in C^G \mid s(c)_{T_i} \geq \sigma\}$

Answer = L^G

EXAMPLE:

Let us take the same database given in FIG. 1, and the same σ mentioned in the above algorithm. For the sake of illustration, let us partition T into three partitions T_1, T_2, T_3 each containing 5 transactions. The first partition T_1 contains transactions 1 to 5 and so on. We fix the local support as equal to the given support , that is 20%. Thus $\sigma_1 = \sigma_2 = \sigma_3 = \sigma = 20\%$. Any itemset that appears in just one of the transactions in any partition is a local frequent set in that partition.

$L^1 := \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{1,5\},\{1,6\},\{1,8\},\{2,3\},\{2,4\},\{2,8\},\{4,5\},\{4,7\},\{4,8\},$
$\{5,6\},\{5,8\},\{5,7\},\{6,7\},\{6,8\},\{1,6,8\},\{1,5,6\},\{1,5,8\},\{2,4,8\},\{4,5,7\},\{5,6,8\},\{5,6,7\},$
$\{1,5,6,8\}\}$

Similarly,

$$L^2 := \{\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{2,3\},\{2,4\},\{2,6\},\{2,7\},\{2,9\},\{3,4\},\{3,5\},\{3,7\},\{5,7\}$$
$$,\{6,7\},\{6,9\},\{7,9\},\{2,3,4\},\{2,6,7\},\{2,6,9\},\{2,7,9\},\{3,5,7\},\{2,6,7,9\}\}$$

$$L^3 := \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{1,3\},\{1,5\},\{1,7\},\{2,3\},\{2,4\},\{2,6\},\{2,7\},\{2,9\},$$
$$\{3,5\},\{3,7\},\{3,9\},\{4,6\},\{4,7\},\{5,6\},\{5,7\},\{5,8\},\{6,7\},\{6,8\},\{1,3,5\},\{1,3,7\},\{1,5,7\},$$
$$\{2,3,9\},\{2,4,6\},\{2,4,7\},\{3,5,7\},\{4,6,7\},\{5,6,8\},\{1,3,5,7\},\{2,4,6,7\}\}$$

In phase II we have the candidate set as

$C := L^1 \cup L^2 \cup L^3$.

Then read the database once to compute the global support of the sets in C and get the final set of frequent sets.

Chapter 4

CLUSTERING TECHNIQUES

4.1 Introduction

Clustering is a useful technique for the discovery of data distribution and patterns in the underlying data. The goal of clustering is to discover both the dense and the sparse regions in a data set. The earlier approaches to clustering do not adequately consider the fact that the data set can be too large to fit in the main memory. In particular, they do not recognize that the problem must be viewed in terms of how to work with limited resources. The main emphasis has been to cluster with as high as accuracy possible, while keeping the I/O costs high. Thus it's not relevant to apply the classical clustering algorithms in the context of data mining and it's necessary to investigate the principle of clustering to devise efficient algorithms which meet the specific requirement of minimizing the I/O operations. It is also to be noted that the basic principle of clustering hinges on a concept of distance metric or similarity metric.

This chapter deals with various types of clustering techniques for data mining. The clustering algorithms for numerical data can be categorized into two classes: partition and hierarchical.

4.2 Clustering Paradigms

There are two main approaches to clustering-hierarchical clustering and partitioning clustering. The partition clustering techniques partition the database into a predefined number of clusters. They attempt to determine k partitions that optimize a certain criterion function. The partition clustering are of two types: k-means and k-medoid algorithms [1, 4].

The hierarchical clustering techniques do a sequence of partitions, in which each partition is nested into the next partition in the sequence. It creates a hierarchy of clusters from small to big and from big to small. The hierarchical techniques are of two types-*agglomerative* and *divisive* clustering techniques. Agglomerative clustering techniques start with as many as clusters as there are records, with each cluster having only one record. Then pairs of clusters are successfully merged until the numbers of clusters reduces to k. If the merging continues, it terminates in a hierarchy of clusters which is built with just a single cluster containing all the records at the top of the hierarchy. Divisive clustering techniques take the opposite approach. This starts with all the records

in one cluster, and then tries to split that cluster into small pieces. DBSCAN is an example of hierarchical clustering.

4.3 Partitioning Algorithms

Partitioning algorithms construct partition of a database of N objects into a set of k clusters. The construction involves determining the optimal partition with respect to an objective function. The partitioning clustering algorithm usually adopts the *Iterative Optimization* paradigm. It starts with an initial partition and uses an iterative control strategy. It tries swapping data points to see if such a swapping improves the quality of clustering. When swapping doesn't yield any improvements in clustering, it finds a locally optimal partition. This quality of clustering is very sensitive to the initially selected partition. There are two main categories of partitioning algorithms. They are

(i) k-means algorithms, where each cluster is represented by the center of gravity of the cluster.

(ii) k-medoid algorithms, where each cluster is represented by one of the objects of the cluster located near the center.

These two algorithms work in a very similar fashion. But most of special clustering algorithms designed for data mining are k-medoid algorithms.

4.4 k-Medoid Algorithms

PAM (Partition Around Medoid) uses a k-medoid method to identify the clusters. PAM selects k objects arbitrarily from the data as medoids. Each of these k objects are representative of k classes. Other objects in the database are classified based on their distance to these k-medoids. The algorithm starts with arbitrarily selected k-medoids and iteratively improves upon the selection. In each step, a swap between a selected object Oi and a non-selected object Oh is made, as long as such a swap results in an improvement in the quality of clustering. To calculate the effect of such a swap between Oi and Oh a cost C_{ih} is computed, which is related to the quality of partitioning the non-selected objects to k clusters represented by the medoids. The algorithm has two important has two important modules-the partitioning of the database for a given set of medoids and the iterative selection of medoids.

PAM Algorithm

Input: Database of objects D.

select arbitrarily k representative objects Kmed.

mark these objects as "selected" and mark the remaining as "non-selected".

do for all selected object O_i

do for all non-selected objects O_h

Compute C_{ih}

end do.

end do.

Select $_{imin,hmin}$ such that $C_{imin,hmin}=Min_{i,h}C_{i,h}$

if $C_{imin,hmin}<0$

then swap: mark O_i as non-selected and O_h as selected.

repeat

find clusters $C_1,C_2,C_3,C_4\ldots.C_k$.

EXAMPLE:

The following figures illustrate the working of the swapping in PAM. There are 12 objects which are required to be classified into 4 clusters. Initially, the selected medoids are O_1, O_2, O_3 and O_5. Based on closest distance, the remaining objects are classified as shown in the figures. Let us assume that O_4 is introduced as a medoid in place of O_5. For convenience we d_{ij} for $d(O_i, O_j)$ and the distance metric is symmetric. The three cases are illustrated below.

Case 1: The object O_8 which was in the cluster C5 before swapping is now in the cluster C_4 after swapping.

Hence, the cost $C_{854} = d_{58}-d_{48}$.

Case 2: The object O_9 was in C_5 before swapping and is now in C_1 after swapping.

The cost for O_9 is $C_{954} = d_{19}-d_{59}$

Case 3: The object O_7 was in the cluster C_1 before swapping and is in C_4 after swapping.

Thus the cost is $C_{754} = d_{47}-d_{17}$

Define the total cost of swapping O_i and O_h as

$C_{ih} = \sum_j C_{jih}$,

Where the sum is taken for all objects *j*.

In the above example, the sum is taken over all the 12 objects to compute C_{54}.

If C_{ih} is negative, then the quality of clustering is improved by making Oh a medoid in place of *O*i. PAM computes C_{ih} for all selected objects O_i and non-selected objects O_h. The process is repeated until we cannot find a negative C_{ih}.

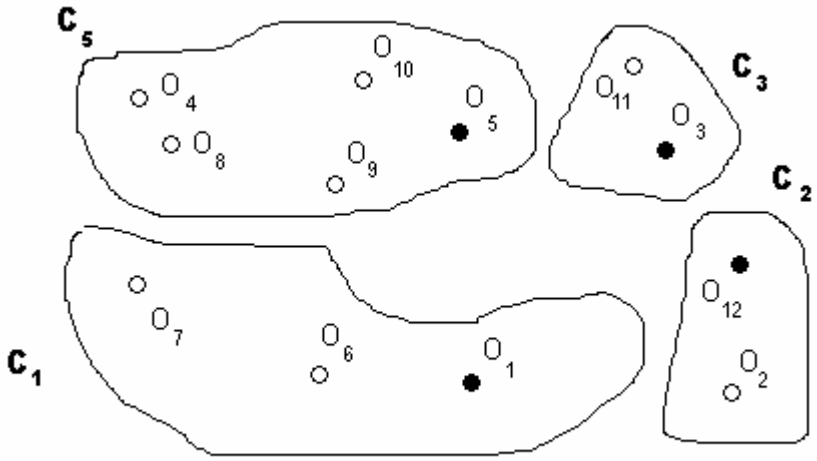


FIG. 3 PARTITIONING BEFORE SWAPPING

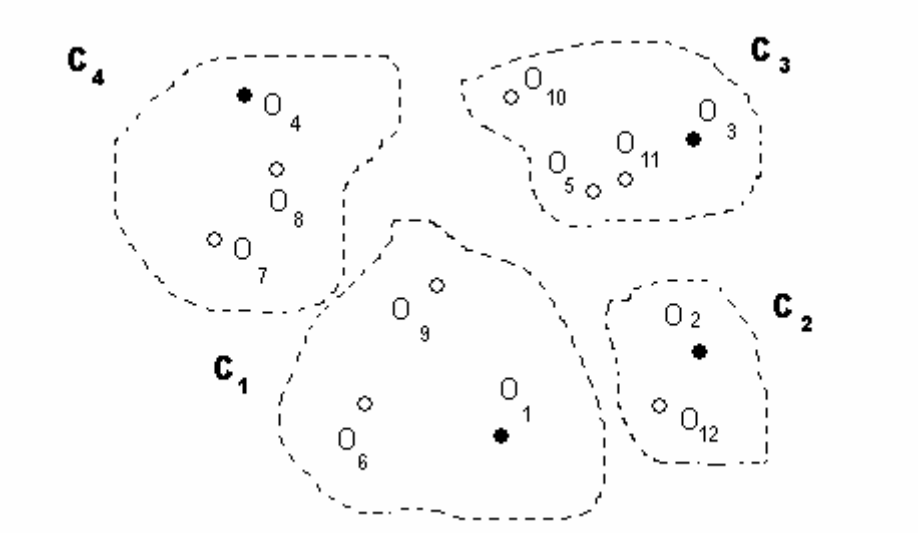


FIG. 4 CLUSTERING AFTER SWAPPING O_5 BY O_4

4.5 Hierarchical Clustering

Hierarchical algorithms create a hierarchical decomposition of the database. The algorithms iteratively split the database into smaller subsets, until some termination condition is satisfied. The hierarchical algorithms do not need k as an input parameter, which is an obvious advantage over partitioning algorithms. However the disadvantage of the hierarchical algorithm is that the termination condition is to be specified. There are two approaches to hierarchical clustering.

(i) Bottom-up (Agglomerative) approach and

(ii) Top-down (Divisive) approach.

4.6 DBSCAN

DBSCAN (Density Based Spatial Clustering of Application of Noise) [1] uses a density-based notion of clusters to discover clusters of arbitrary shapes. The key idea of DBSCAN is that, for each object of a cluster, the neighborhood of a given radius has to contain at least a minimum number of data objects. In other words, the density of the neighborhood must exceed a threshold. The critical parameter here is the distance function for the data objects.

We identify two different kinds of objects in DBSCAN-core objects and non-core objects. Non-core objects in turn, are either border objects or noise objects. Two border objects are possibly not density-reachable from each other. However, a border object is always density-reachable from a core object. A noise object is a non-core object, which is not density-reachable from other core objects. The DBSCAN algorithm maintains the set of objects in three different categories. These are: classified, unclassified and noise. Each classified objects has an associated cluster-id. A noise object may also have an associated dummy cluster-id. For both classified and noise objects, the ε-neighborhoods are already computed. The unclassified category of objects doesn't have any cluster-id and their neighborhoods are not computed. The algorithm gradually converts an unclassified object into a classified or noise object.

At each step, the algorithm starts with an unclassified object and a new cluster-id associated with it. We study its ε-neighborhood to see if it's adequately dense or not. If its density does not exceed the threshold MinPts, then it's marked as a noise object.

Otherwise all the objects that are within its ε-neighborhood are retrieved and put into a list of candidate objects. These objects may be either unclassified or noise. But it can't contain any classified object. If the object is a noise object, the current cluster-id is assigned to it. If the object is unclassified, then the current cluster-id is assigned to it and it is included in the list of candidate objects for which the ε-neighborhoods are to be obtained. The algorithm continues till the list of candidate objects is empty. Thus, one cluster with the given cluster-id is determined. The algorithm repeats this process for other unclassified objects and terminates when all the objects are marked as either classified or noise.

DBSCAN Algorithm

Algorithm DBSCAN (D, ε, *MinPts*)
Input: Database of objects D
do for all $O \in D$
    if O is unclassified
        call function *expand_cluster*(O, D, ε, *MinPts*)
end do.

Function expand_cluster (O, D, ε, *MinPts*):
get the ε-neighbourhood of O as $N_\varepsilon(O)$
    if $|N_\varepsilon(O)| <$ *MinPts*,
        mark O as noise
        return
    else
        select a new *cluster_id* and mark all the objects of $N_\varepsilon(O)$ with this *cluster_id*
        and put them into *candidate-objects*.
        do while candidate-objects are not empty
            select an object from *candidate-objects* as *current-object*
            delete *curent-object* from *candidate-objects*
            retrieve N_ε(*current-object*)
            if $|N_\varepsilon(current\text{-}object)| \geq MinPts$
                select an object N_ε(*current*-object) not yet classified or marked as noise,
                mark all of the objects with *cluster_id*,
                include the unclassified objects into *candidate-objects*
        end do
return.

EXAMPLE:

An illustration of expand-cluster phase of DBSCAN algorithm is given below. In the figure below we assume MinPts = 6 and we start with an unclassified object O_1. We find that there are 6 objects in the ε-neighborhood of O_1. These are: O_1, O_2, O_3, O_4, O_5 and O_6. So put all these points in the candidate-objects and associate them with cluster-id of O_1 (say C_1). We select an object from the candidate-objects, say O_2. We find that $N_\varepsilon(O_2)$ does not contain adequate number of points. Hence we mark this as a noise object. The object O_4 is already marked as a noise object, so there is no further action for this point. Let us select the object O_3 from the candidate objects. $N\varepsilon(O_3)$ contains 7 points O_1,O_3,O_5, O_6,O_9,O_{10} and O_{11}. Among these O_9, and O_{10} are already marked as noise objects, O_1 and O_3 are already classified and the others are unclassified. All the seven objects are associated with new cluster-id C_1. The unclassified objects are included in the candidate-objects for the next iteration.

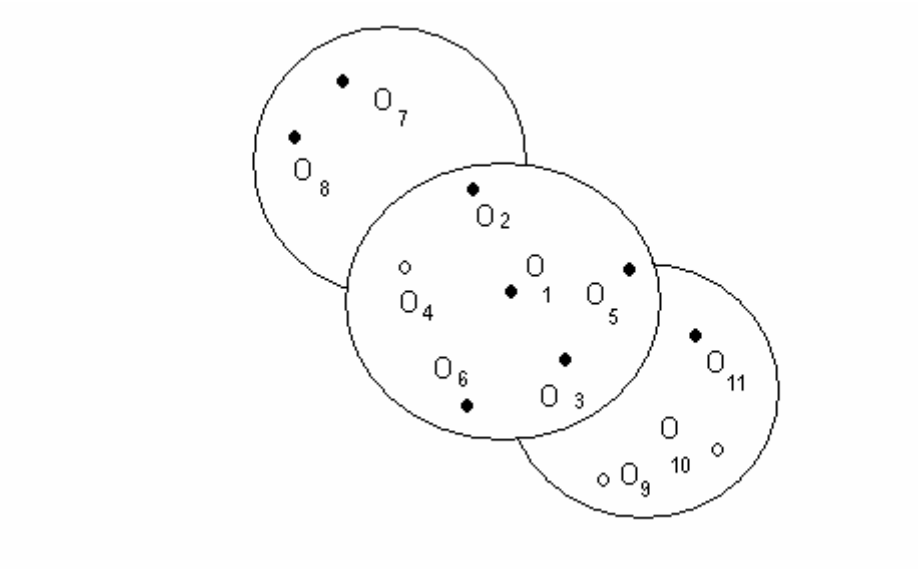


FIG. 5 ILLUSTRATION OF DBSCAN ALGORITHM

Chapter 5

IMPLEMENTATION AND RESULTS

5.1 A PRIORI ALGORITHM

The algorithm is implemented in Java platform and analyzed by their execution time taken to generate the result.

For A-Priori algorithm it is seen that for a support count of 6 on a 60 itemset dataset it took on an average of 550 ms to get the frequent set. But the time increased when the support count is decreased. When the support is decreased to 4 the algorithm took on an average of 590 ms to complete the task. The time also gradually increased with the increase in the size of the dataset.

5.2 PARTITION ALGORITHM

This algorithm is also implemented using Java and analyzed by the above mentioned parameters.

The same database is put into consideration for the partitioning algorithm too. It is seen that it took an average of 1862 ms to generate an output. The decrease in the support count didn't affect much the execution time. Also with the increase in the dataset, the increase in the execution time was acceptable.

5.3 COMPARISON BETWEEN A-PRIORI AND PARTITION ALGORITHMS

From the above, it is seen that for a small database with a large support count A-Priori algorithm stands taller than the Partition algorithm. With the increase in the datasets, the execution time increased for both the algorithms. But the increase in time was huge for A-Priori algorithm, and it was greater than Partition algorithm. So, it is seen that with large databases Partition algorithm fares better than A-Priori algorithm.

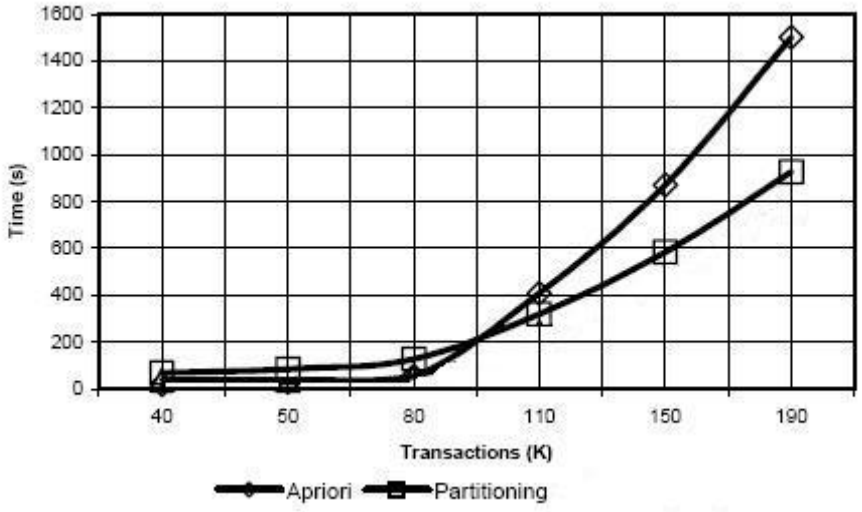


FIG. 6 GRAPH FOR COMP-ISION OF ALGO-RITHMS

5.4 K-MEANS AND K-MEDOID

Both the clustering algorithms are implemented using Java. In k-means algorithm the total number of interchanges is found to be 116 for a database of 60 objects. The execution time is found to be 3976 ms.

K-medoid algorithm is implemented using the same database used in K-means. The total number of interchanges comes out to be 171. The execution time is more than that of k-means. Here the execution time is found to be 6029 ms.

5.5 DBSCAN

The DBSCAN algorithm is implemented using Java. For a database of 200 objects the number of clusters is found to be 55 and the execution time to be 64.7 sec. For a database of 400 the number of clusters produced is 75. The execution time is 216 sec. For a database of 600 objects this program produces 77 clusters. The execution time is 333 sec. For a database of 800 objects the number of clusters is found to be 81 and the execution time to be 462 sec. For 1000 objects the number of clusters is found to be 82 and the execution time to be 589 sec. In all the cases the intracluster distance remains less than 0.1.

5.6 COMPARISON BETWEEN K-MEANS AND DBSCAN ALGORITHMS

For a larger database DBSCAN shows more efficiency than the k-means algorithm as the execution time is less in DBSCAN. For a smaller database the number of clusters is more in DBSCAN. The ratio of number of clusters to the number of objects gradually decreases as the database becomes larger in DBSCAN.

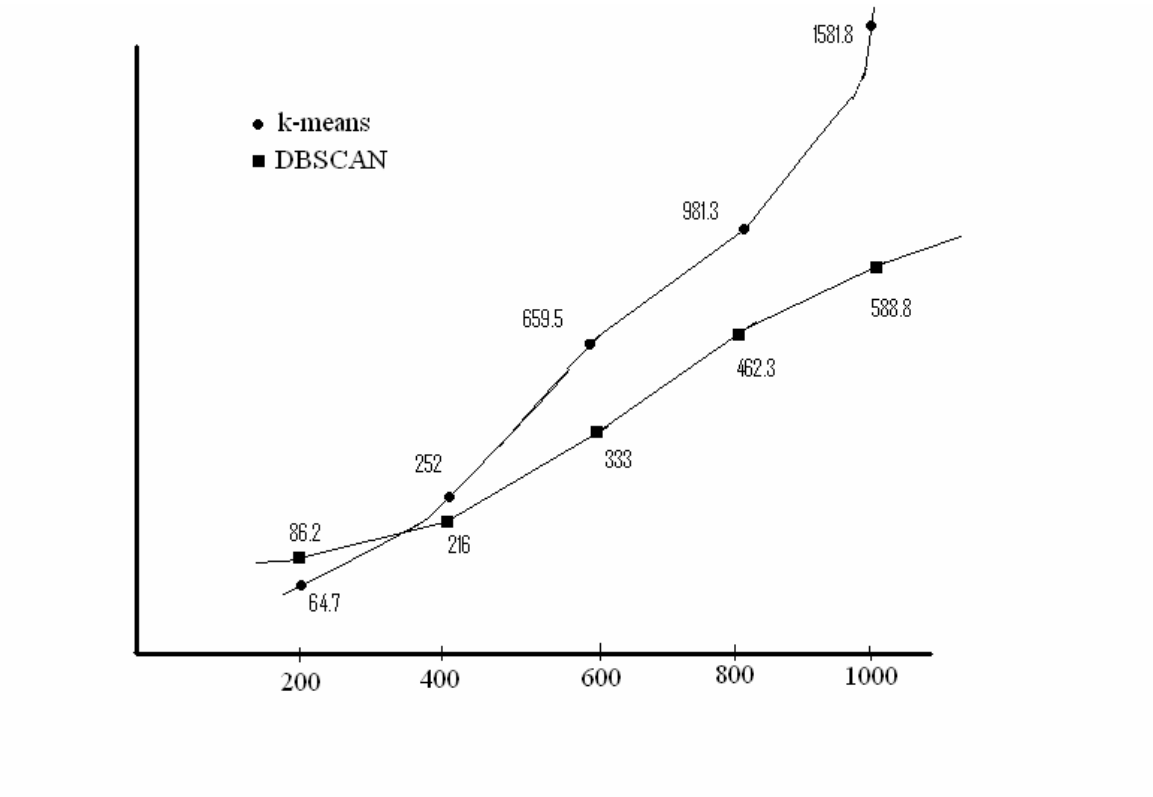


FIG. 7 COMPARISION OF CLUSTERING ALGORITHMS

The above graph shows a comparison between k-means and DBSCAN algorithms. From the graph it is clear that for larger databases the execution time of DBSCAN is less than that of k-means.

Chapter 6

CONCLUSION

CONCLUSION

In this project we implemented some well known algorithms under association rules and clustering techniques. The algorithms were compared against each other and seen for their efficiency. For association rule, the algorithms implemented are A-Priori and Partitioning algorithm were implemented and compared. The results show that for small databases A-Priori is better and takes less time than the other, where as in case of large databases Partition algorithm fares better with executable time better than the other. To sum it all, both the algorithms can be used efficiently according to usage in various fields. Under the clustering techniques of data mining various algorithms namely- k-means, k-medoid and DBSCAN algorithms are implemented using Java. The results are compared and analyzed in accordance to their efficiencies.

REFRENCES

1. PUJARI A.K. Data mining Techniques, University Press

2. TWO CROWS CORPORATION Introduction to Data mining and Knowledge Discovery, pp. 6-9

3. GYORODI ROBERT S. A Comparative Study of Iterative Algorithms in Association Rules Mining, Department of Computer Science, Faculty of Electrotehnics and Informatics, University of Oradea, Romania

4. SALEM SAMEH A. AND NANDI ASOKE K. New Assessment Criteria for Clustering Algorithms, Signal Processing and Communications Group, Department of Electrical Engineering and Electronics, The University of Liverpool, Brownlow hill, Liverpool, U.K.