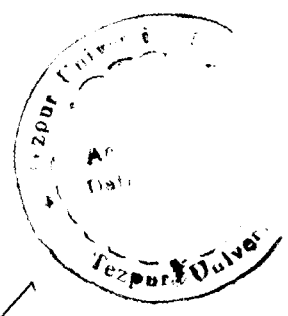


T 123



43052

T 123

27/02/13

A study on Multi Objective Association Rule Mining

*A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy*

Bhabesh Nath

Registration No. 038 of 2000



School of Engineering
Department of Computer Science and Engineering
Tezpur University
July 2009

Abstract

The problem of mining association rules has attracted a lot of attention in the research community. Several techniques for discovery of association rules have been discussed in literature. These algorithms, motivated by Agrawal's approach, handles the rule mining problem as a single objective problem. However, the rule mining based on single objective have some common drawbacks, i.e. rules may be difficult to understand due to the involvement of a lot of conditions, some discovered rules may not be interesting to the user as they were not actually hidden. To overcome these limitations, this dissertation presents an approach for rule mining using multiple objectives so that some interesting and useful rules can be extracted. For this - three different objectives namely *predictive accuracy*, *comprehensibility* and *interestingness* are considered. As a result, the proposed approach is found to be better than the classical approach as some previously unknown, potentially useful and ultimately understandable rules can be discovered.

Moreover, the single objective based classical rule mining approach can be found to be impractical as sometimes they may produce a huge number of rules, that in turn makes the job of decision makers tougher one in deciding which rule to use which to discard. However the proposed approach will give only a few valid rules, which will help the decision maker as the choice is limited.

It was found that *Pareto Based Genetic Algorithm* acts as an efficient tool in handling the multi-objective problems. Since association rule mining is treated as a multi objective problem, the proposed approach uses Pareto based multi objective genetic algorithm to extract the rules.

The dimensionality of the databases plays an important role in the data mining tasks including association rule mining. Appropriate *feature selection* or *dimensionality reduction* techniques can save the cost of computation during the association rule mining over high dimensional space to a grate extent. Hence, this dissertation presents two effective techniques as a preprocessing task to reduce the dimensionality of the databases before applying any data mining techniques.

Rule mining being a time consuming job, it is not appreciated to extract the rules again and again from the whole database, every time the database is updated. So the task of extracting the rules from an incremental database received a lot of research attention. However, my survey reveals that most of the existing works are basically based on the classical approach. To meet this challenge, this dissertation also presents a technique of incremental mining based on multi objective approach.

Keywords — Data mining, Association rule mining, Multi objective rule mining, Genetic algorithms for rule mining, Pareto based rule mining, GA based incremental mining



DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
TEZPUR UNIVERSITY

NAPAAM, TEZPUR – 784028
ASSAM : INDIA

Ph : 03712 - 267007-009
Ext- 5101, 5353
Fax : 03712 - 267005/006
Email: dkb @ tezu.ernet.in

Prof. D. K. Bhattacharyya
Head

Certificate of the Research Supervisor

This is to certify that the thesis entitled “**A study on Multi Objective Association Rule Mining**” submitted to the *Tezpur University* in the *Department of Computer Science and Engineering* under the *School of Engineering* in partial fulfilment for the award of the degree of the **Doctor of Philosophy in Computer Science** is a record of research work carried out by **Mr. Bhabesh Nath** under my supervision and guidance.

All helps received by him from various sources have been duly acknowledged.

No part of this thesis has been reproduced elsewhere for award of any other degree.

Signature of Research Supervisor

Designation : Professor

School : Engineering

Department : Computer Science and Engineering

INDIAN STATISTICAL INSTITUTE

Telephone : (+91) (33) 2575-3110/3100

FAX : (+91) (33) 2578-3357
(+91) (33) 2577-3035

Email : ash@isical.ac.in



203 BARRACKPORE TRUNK ROAD
KOLKATA 700 108, INDIA

Residence : M-4-14 Teachers' Quarter
Jadavpur University
KOLKATA 700 032

Telephones : (+91) (33) 2414 6067

Dr. Ashish Ghosh, M. Tech., Ph.D.
Professor
Machine Intelligence Unit

Certificate of the Joint Research Supervisor

This is to certify that the thesis entitled “**A study on Multi Objective Association Rule Mining**” submitted to the *Tezpur University* in the *Department of Computer Science and Engineering* under the *School of Engineering* in partial fulfilment for the award of the degree of the **Doctor of Philosophy in Computer Science** is a record of research work carried out by **Mr. Bhabesh Nath** under my supervision and guidance.

All helps received by him from various sources have been duly acknowledged.


No part of this thesis has been reproduced elsewhere for award of any other degree.


Ashish Ghosh

(Research Supervisor)

Declaration

I, Bhabesh Nath, hereby declare that the thesis entitled '*A Study on Multi Objective Association Rule Mining*' submitted to the Department of Computer Science and Engineering under the School of Engineering, Tezpur University, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, is based on bonafide work carried out by me. The results embodied in this thesis have not been submitted in part or in full, to any other University or Institute for award of any degree or diploma.


(Bhabesh Nath)

Acknowledgements

There are several people who have in one way or another made this thesis possible, and to whom I wish to express my gratitude. First, I would like to take the opportunity to offer my heartfelt gratitude to Prof. Dhruba Kr. Bhattacharyya, Tezpur University, for his sustained and continuous guidance during the course of this dissertation work. I am grateful to Prof. Ashish Ghosh, ISI, Kolkata for his valuable suggestions and guidance at various stages of this work.

My gratitude is extended to all members of the Department of Computer Science and Engineering, Tezpur University including Ajay and Arun of the technical staff, for their help and support. I would like to express my special gratitude to Dr. Utpal Sharma and Dr. Shyamanta Moni Hazarika. The innumerable discussions, regarding not only research but everything under the sun have created a stimulating atmosphere to work in. Thanks for the many valuable discussions during the tenure of this dissertation and for the companionship in traveling the bumpy road towards the Ph.D. degree.

I would also like to thank my teacher-cum-friends Dr. Chandan Goswami, Dr. Apurba Das, Dr. Tridib Ranjan Sharma and Mr. Pankaj Bora for their encouragement and for sharing some of the additional loads on me.

I shall remain thankful to Mr. Santanu Sharma of Department of Electronics and Communication Engineering of Tezpur University, for his helps in various stages of this dissertation work.

Last but not the least I would like to thank my wife, Rupali, for bringing light in the moment of darkness of my life. Her love and support were one of my main sources of strength in some crucial moments during this dissertation work. Finally I will ask my daughter, Riniki, to forgive me for those times from her childhood days, I owed to her.



Tezpur University

This is to certify that the thesis entitled “A study on Multi Objective Association Rule Mining” submitted by **Mr. Bhabesh Nath** to *Tezpur University* in the *Department of Computer Science and Engineering* under the *School of Engineering* in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science has been examined by us on _____ are found to be satisfactory.

The committee recommends for the award of the degree of Doctor of Philosophy.

Signature of
Supervisor

External Examiners

Date

Contents

1	Introduction	1
1.1	Background	1
1.2	Association Rule Mining	4
1.3	Applications of Association Rules	6
1.4	Multiple Objectives of Association Rules	8
1.5	Motivation of the Work	9
1.6	Work Done	10
1.7	Organization of the Thesis	10
2	Association Rule Mining: The Classical Approach	12
2.1	Classical Algorithms for Mining Frequent Itemsets	13
2.1.1	Apriori	13
2.1.2	SETM	15
2.1.3	Direct Hashing and Pruning	15
2.1.4	Partition Algorithm	17
2.1.5	Dynamic Itemset Counting	19
2.1.6	Pincer Search	21
2.1.7	FP-tree Growth	23
2.2	Algorithms for Generation of Rules	25
2.2.1	Agrawal's Algorithm	26
2.2.2	Srikant's Simple Algorithm	27
2.2.3	Srikant's Faster Algorithm	28

2.3	A Faster Rule Generation Algorithm	31
2.3.1	Experimental Results	33
2.3.2	Observations	35
2.4	Discussion	36
3	Multi Objective Association Rule Mining	41
3.1	Multi Objective Problems	41
3.2	Handling Multi Objective Problems	43
3.3	Genetic Algorithms as a Tool	45
3.4	Multi Objective Genetic Algorithms	46
3.4.1	Vector Evaluated Genetic Algorithm	46
3.4.2	Multi Objective Genetic Algorithm	47
3.4.3	Non-dominated Sorting Genetic Algorithm	48
3.5	Multiple Objectives of Association Rule Mining	49
3.5.1	Efficient MOGA for Association Rule Mining	51
3.5.2	MOGA Based Partitioning Approach	58
3.6	Discussion	64
4	Dimensionality of Databases: Another Challenge	66
4.1	Dimensionality Reduction	67
4.2	Existing techniques	67
4.2.1	Focus	68
4.2.2	LVF	69
4.2.3	Branch and Bound	70
4.2.4	Relief	71
4.2.5	DTM	72
4.2.6	FFC	72
4.2.7	MDLM	73
4.3	Dimensionality Reduction: New Approaches	74
4.3.1	Frequency Count Based Reduction	75
4.3.2	Rule Based Reduction	85

4.4	Discussion	92
5	MORM in Incremental Databases	93
5.1	Need of Incremental Mining	94
5.2	Existing Techniques	95
5.2.1	FUP	97
5.2.2	FUP ₂	98
5.2.3	MAAP	99
5.2.4	Borders	100
5.2.5	Efficient Counting Using TID-lists	102
5.2.6	Maximal Frequent Trend Pattern	103
5.2.7	Modified borders	104
5.3	Proposed Method	106
5.3.1	MORM in Incremental Databases	107
5.3.2	Implementation and Results	111
5.4	Discussion	116
6	Conclusions and Future works	118
6.1	Conclusions	118
6.2	Future works	120

List of Tables

2.1	Frequent Itemsets Derived: An Example	34
2.2	Rules Derived by Agrawal's Algorithm	35
2.3	Rules Derived by Srikant's Simple Algorithm	36
2.4	Rules Derived by Srikant's Faster Algorithm	37
2.5	Frequent Itemset from a Synthetic Dataset	38
2.6	Rules from the Synthetic Dataset	38
2.7	Frequent Itemset from monks-1 Dataset	39
2.8	Rules from monks-1 Dataset	39
2.9	Frequent itemset from monks-3 Dataset	39
2.10	Rules from monks-3 Dataset	40
3.1	Multi Objective Problem : An Example	42
3.2	Summary of results	56
3.3	Rules from kddcup.data.10_percent dataset	57
3.4	Attributes of kddcup dataset involved in the reported rules . .	58
3.5	Some rules from Wisconsin Diagnostic Breast Cancer Database	61
3.6	Some rules from Wisconsin Breast Cancer Database	62
3.7	Some rules from Wisconsin Prognostic Breast Cancer	63
4.1	Symbols used in Above Algorithms	74
4.2	Symbols used in DRUFT	77
4.3	Dimensionality Reduction on Monks-1 and Monks-3 by DRUFT	80
4.4	Comparative Results of Some Existing Algorithms and DRUFT	81

4.5	Reduction in Synthetic Datasets	83
4.6	Time Taken in Deriving Frequent Itemset	85
4.7	Description of the Synthetic Database	89
5.1	A sample database	95
5.2	Frequent itemsets from Table 5.1 with minimum support 50%	96
5.3	Updated sample dataset	96
5.4	Frequent itemsets from Table 5.3 with minimum support of 50%	97
5.5	Symbols and Notations used	107
5.6	Incremental rules from WDBC database	113
5.7	Incremental rules from WBC database	114
5.8	Incremental rules from WPBC database	115
5.9	Comparison of Static and Incremental MORM	116

List of Figures

2.1	Apriori	14
2.2	Direct Hashing and Pruning	17
2.3	Partition Algorithm	18
2.4	Dynamic Itemset Counting	21
2.5	Pincer Search	23
2.6	FP-Tree Growth	25
2.7	Generating rules: Agrawal's Algorithm	26
2.8	Generating Rules: Srikant's Simple Algorithm	27
2.9	Generating Rules: Srikant's Faster Algorithm	30
2.10	NBG: A Faster Rule Generation Algorithm	32
3.1	GA based Multi Objective Rule Mining	54
3.2	MOGA based partitioning algorithm	60
4.1	Focus	68
4.2	LVF	69
4.3	Branch & Bound	70
4.4	Relief	71
4.5	FFC	73
4.6	Dimensionality Reduction Using Frequency Count	77
4.7	Dimensionality Reduction for Association Rule Mining	83
4.8	Dimensionality Reduction with Multi objective GA	88

5.1	FUP2	99
5.2	Borders (addition)	102
5.3	ECUT	103
5.4	MFTP	104
5.5	Modified Borders	106
5.6	MORMI	110

List of Abbreviations/Acronyms

AOF	: Aggregate Objective Function
DHP	: Direct Hashing and Pruning
DIC	: Dynamic Itemset Counting
DRARM	: Dimensionality Reduction for Association Rule Mining
DRMOGA	: Dimensionality Reduction with Multi Objective Genetic Algorithm
DRUFT	: Dimensionality Reduction Using Frequency Count
DTM	: Decision Tree Method
ECUT	: Efficient Counting Using TID-list
FFC	: Feature selection using Frequency Count
FP-tree	: Frequent Pattern tree
FUP	: Frequent Update
KDD	: Knowledge Discovery in Databases
LVF	: Filter version of Las Vegas algorithm
MAAP	: Maintaining Association rules with Apriori Property
MDLM	: Minimum Description Length Method
MFTP	: Maximal Frequent Trend Pattern
MOGA	: Multi Objective Genetic Algorithm
MORM	: Multi Objective Rule Mining
MORMI	: Multi Objective Rule Mining in Incremental Database
NSGA	: Non-dominated Sorting Genetic Algorithm
SETM	: Set oriented Mining
SGA	: Simple Genetic Algorithm
TID	: Transaction Identifier
VEGA	: Vector Evaluated Genetic Algorithm

Chapter 1

Introduction

1.1 Background

With the rapid growth of technologies of data storage and collections, capabilities of collecting data are also increasing rapidly. The widespread use of bar codes for most commercial products, the computerization of many business and government transactions, and the advances in data collection tools have provided us with huge amounts of data. These days, millions of databases are in use in business management, government administration, scientific and engineering data management, and many other applications. Because of availability of powerful but affordable database systems, these databases are growing in an explosive rate. This rapid growth in data and databases has generated an urgent need for new techniques and tools that can automatically and intelligently transform the processed data into useful information and knowledge.

The process of handling these large databases for extracting knowledge is known as knowledge discovery in databases(KDD). *KDD is the process*

of identifying valid, novel, potentially useful and ultimately understandable structure in data [BFM98]. There are several steps in KDD such as data preparation, data selection, data cleaning, incorporating appropriate prior knowledge, extraction of new information, proper interpretation of the information extracted etc.

Due to these massive data collected by the systems of different organizations, it is becoming more difficult to extract the useful information from them. So a new challenge has arose to find efficient techniques to discover useful and interesting patterns from such a huge amount of data. Hence this step of KDD, commonly termed as Data Mining, has emerged as new area to meet this challenge of database research. Recently, data mining attracted a lot of research attentions. Data mining has been defined differently by different researchers, such as *“The efficient discovery of previously unknown patterns in large databases”* [AP95] and *“The non-trivial extraction of explicit, previously unknown and potentially useful information (such as rules, constraints and regularities) from data in databases”* [CHY96]. But many researchers used these two terms, KDD and Data Mining as synonyms.

These discovered patterns help in decision making and to predict the future behaviours. Depending on the users requirement different types of information are to be extracted from these databases leading to different data mining tasks. Some of the commonly known tasks are – classification, clustering, association rule mining, sequential pattern analysis, prediction and data visualization [BL97, CPS00, CHY96, FPSM91] a few of them are briefly discussed below. All these tasks have different types of applications.

Classification: The input to the classification consists of multiple ex-

amples (records), each of them having multiple attributes or features. Every record has been tagged with a special class label. The objective of classification is to analyze the input data, termed as training set, and to develop an accurate description or model for each class in terms of the attributes of the data. These descriptions are used to classify new records, termed as test data, for which class labels are unknown.

Clustering: It is the process of grouping records of '*similar*' type. The input data for clustering is similar to that of classification, except that the records are not tagged. Some times it is termed as unsupervised classification also. Clustering helps in constructing meaningful partitions of a large set of objects (records) based on a "divide and conquer" methodology which decomposes a large scale system into smaller components to simplify design and implementation.

Association rule mining: In a given database with a number of attributes, the different sets of attributes of the database have some interrelations or associations among them. The objective of association rule mining is to discover these associations among the attributes of the database. This extraction process is unsupervised, i.e. no prior information is required during classification.

Time series analysis: Time series data constitutes a large portion of data stored in computers. The capabilities to find time-series (or portions thereof) that are "similar" to a given time-series or to be able to find groups of similar time series has several applications. As for example, identifying the companies with similar growth patterns, finding products with similar selling patterns, discovering stocks with similar price movements, etc.

1.2 Association Rule Mining

The problem of mining for associations over a binary database, known as Market Basket Database, was first introduced in [AIS93]. Association rule mining can be stated as follows [CHY96]: given a database of sales transactions, it is desirable to discover the important associations among items such that the presence of some items in a transaction will imply the presence of other items in the same transaction. An example of an association rule is: 30% of transactions that contain *bread* also contain *butter*; 5% of all transactions contain both of these items. The following formal definition was proposed in [AS94] to address the problem.

Let $Item = \{i_1, i_2, \dots, i_m\}$ be a set of literals called *items*, DB be a database of transactions where each transaction $T \subseteq Item$ and has a unique identifier, TID . Given an itemset $X \subseteq Item$, X is contained in T iff $X \subseteq T$. An association rule is an implication of the form $S_a \Rightarrow S_c$, where both S_a (*rule antecedent*) and S_c (*rule consequent*) are itemsets and $S_a \cap S_c = \phi$. A rule has confidence c iff $c\%$ of the transactions containing S_a also contain $S_a \cup S_c$. An itemset is *frequent* iff its support exceeds a certain support threshold *minsup*.

Given a set of transactions, where each transaction is a set of *items*, associations among two sets of items X and Y can be expressed as a rule of the form of IF- THEN statement. IF <some conditions are satisfied > THEN <predict some values of other attribute(s)>. The conditions associated in the IF part is termed as *Antecedent* and those with the THEN part is called the *Consequent*. Refer them as A and C , respectively, symbolically we can

represent this relation as $A \Rightarrow C$. The intuitive meaning of such rule is that transactions of the database which contain X tend to contain Y also. Association rule mining is the process of finding all association rules that satisfies two user-specified constraints *minimum support* and *minimum confidence*.

The problem of discovering all association rules can be decomposed into two sub-problems [AIS93]:

- **Frequent Itemset Generation:** Find all sets of items that have transaction support above a given minimum support. These are the *frequent* itemsets. Other itemsets are called *infrequent* itemsets.
- **Rule generation:** Use the frequent itemsets to generate the desired rules, having *confidence* more than a user specified *minimum confidence*.

The second phase can be done in a straight forward manner in main memory once the frequent itemsets are found [AS94]. In [Sri96], some better techniques have been proposed. But due to the huge search space (the power set of the set of all items), the first phase becomes more time consuming. That is the reason for the attention of great number of researchers paid to this problem in recent years.

In this dissertation, we examine the problem of mining association rules. We first present a faster algorithm to extract rules using the conventional approach. Then some efficient algorithms are presented to extract a reduced set of rules to help the decision makers.

1.3 Applications of Association Rules

The problem of mining association rules was originally motivated by the decision support problem faced by most of the large retail organizations [SAD⁺93]. But now-a-days it is gaining its popularity among the decision support systems of all those organizations maintaining their transactional databases. Depending on the need of the organization these rules have different applications. Some application specific works can be found in [FL07, SL08, HLS⁺07].

Item Placement: To provide a better service to the customers of a large retail store, the management of the store should be aware of the selling patterns of different items. Knowledge about the items are sold together is an useful information for providing a better service to the customers by placing those items together in the store. But for a large store, retailing thousands of different items to thousands of customers per day, it is not as easy job to find these types of groups of items. Association rules help them to take these types of decisions. A closely related application is **catalog placement**. Mail-order companies can use association rules in determining the items to be placed on the same page of the catalog.

Customized catalog: Rather than sending the same catalog to everyone, direct marketing retailers can use associations to customize the catalog based on the items a person has bought. These customized catalogs are generally much smaller, or may be mailed infrequently, reducing mailing cost. Customized online catalogs are very much helpful to the customers doing web-marketing, as the catalog of the products of his interest only will be produced to him.

Fraud Detection: Insurance companies are interested in finding groups of medical service providers, doctors and clinics, who forces the patients to move between each other for unnecessary tests. Given medical claims data, each patient can be mapped to a transaction, and each doctor/clinic visited by the patient to an item in the transaction. Using the association rules now the insurance company can investigate the claim records for sets of providers who have a large number of common patients to determine, if any, fraudulent activity actually occurred.

Medical diagnosis: The different information stored about the previous patients having a particular disease is very much useful to diagnose a new patient. Rules extracted from these databases help the doctors to diagnose the disease quickly. That normally becomes fruitful in case of several lethal diseases like cancer, where early detection increases the probability of curing.

Medical Research: The symptoms of disease of a patient, any correspond to that of another patient diagnosed by the doctor. The patterns discovered using these data could be of use in research in order to help identify symptoms/diseases that precede certain diseases.

Intrusion detection: For the security of the computer systems connected to the internet it is very much necessary to prevent the malicious requests. Association rules derived from the records of the previous activities handled by the system, can help a system to detect these types of new malicious requests, termed as intrusion, to keep the system secure.

1.4 Multiple Objectives of Association Rules

Though the classical algorithms for association rule mining are giving more emphasis on the first phase of the rule mining problem, i.e. on the frequent itemset generation, however, for the appropriate use of these frequent itemsets, the second phase has to be applied. Once the *frequent itemsets* have been extracted, rules can be extracted from them by using rule generation algorithms [AIS93, Sri96]. The rule generation algorithms calculate the *confidence* of every possible candidate rules. The candidate rules having the confidence more than a user specified threshold *minimum confidence* are declared as generated rules. Confidence is sometimes termed as *predictive accuracy* also. It is the ratio of the support count of the whole rule to that of the antecedent part. For example, confidence of the rule $A \Rightarrow C$ is $SUP(A \cup C) / SUP(A)$, where $SUP(X)$ is the number of records/transactions containing the set of items X .

Minimum support and minimum confidence, these two parameters affect the performance of the rule mining algorithms significantly. The size of rule set generated is highly influenced by these parameters. If their values are not properly tuned, then the rule generation phase may not result any rule, or a huge number of rules may be resulted, making the analysis of rules more complex.

Classical algorithms use the confidence as the only measure to evaluate the rules. Hence the association rule mining is handled as a single objective problem. But these extracted rules may not carry any interesting information within it. If the support count of the rule is very high, then confidence of the rule is also generally higher. But this type of association may be ex-

tracted without using any data mining tasks, because these associations are not hidden from the users. For example, say, in a databases with 100 records if $SUP(A)$, $SUP(A \cup B)$ and $SUP(A \cup C)$ are 90, 80 and 20 respectively. Then the confidence of $A \Rightarrow B$ will be $8/9$, that is higher than $2/9$ which is the confidence of $A \Rightarrow C$. From a simple observation of the databases, occurrence of $A \cup B$ can be detected. But it is not so simple to detect the occurrence of $A \cup C$. Hence the rule $A \Rightarrow C$ carries more interesting information, than $A \Rightarrow B$. To discover these interesting rules some effort was found in [Yun07].

At the same time the rules that were generated may be very long. These long rules may be useful but difficult to understand [XL07]. If the rules are not understandable then no decision makers will use those rules.

So there is a need of handling the problem as multi objective problem. *Interestingness* and *comprehensibility* can be used as two additional objectives of the association rule mining problem. Interestingness defines the *surprisingness* of the rule whereas comprehensibility defines the *understandability* of the rules. If the association rule mining is handled as a multi objective problem, using interestingness, confidence and comprehensibility as the measures, some previously unknown, potentially useful and ultimately understandable rules may be extracted.

1.5 Motivation of the Work

The great practical benefits of mining association rules and its wide area of applications have led to several proposals for fast mining of association rules. Those proposals, although contributed towards making the process

more applicable in practical systems, still suffer from the problem of huge amount of generated rules that can be found to be confusing and most of the time not useful to the user. User will get the maximum benefit if a small set of understandable and practically useful rules are provided to him. This need of the users has motivated us to the work presented in this thesis.

1.6 Work Done

Several challenges of data mining problem were encountered at various stages of the work; and concentration was given to all of them. The significant aspects of rule mining, those were considered during the formulation of this work are –

- Faster generation of association rules.
- Efficient algorithms for mining association rules using Multi Objective Genetic Algorithms.
- Algorithms to reduce dimensions of databases to help the data mining tasks including association rule mining.
- Efficient algorithms to extract rules from incremental databases.

1.7 Organization of the Thesis

Chapter 2 presents some of the classical algorithms used for frequent itemset generation, and rule generation. Then an efficient algorithm to generate rules by the conventional approach is discussed. In this chapter, it is established that association rule mining is not a single objective problem, but a multi-objective problem.

Chapter 3 presents some efficient techniques for generation of association rules, using Multi Objective Genetic Algorithms.

Chapter 4 presents the need of dimensionality reduction in data mining tasks, followed by some algorithms to achieve it.

Chapter 5 presents the need of incremental mining followed by an algorithm to tackle the problem of incremental association rule mining.

Finally, Chapter 6 summarizes the dissertation and presents suggestions for future work.

Chapter 2

Association Rule Mining: The Classical Approach

Classical approach of mining association rules handles the problem as a single objective problem, and works in two phases. First phase being the most time consuming one has attracted the attention of many researchers. Though the second phase is also an important part of the rule mining, only a few work can be found in the literature [AIS93, Sri96].

Starting with *Apriori* [AIS93] a significant number of works have been carried out to attend the first phase of the association rule mining problem. A few of them are presented in the next section. Some more works can be found in [SB04, NLWF05, ZKCY07, CKN08, CTL09]. But the list is not exhaustive.

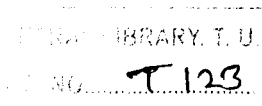
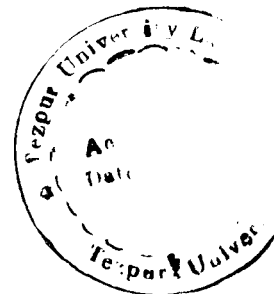
2.1 Classical Algorithms for Mining Frequent Itemsets

Several popular algorithms have been presented in the past decade to handle this problem of frequent itemset generation. These algorithms can be divided into two broad categories depending on their way of finding frequent itemsets, namely bottom up (i.e. agglomerative) and top down (i.e. divisive). Both of these approaches take the benefit from the downward closure property of frequent itemsets, i.e. *if an itemset is frequent, then all of its subsets must also be frequent* [AS94]. Some of the popular frequent itemset generation techniques are reported next.

2.1.1 Apriori

13052

In Apriori [AIS93] candidate itemsets are generated and counted on-the-fly as the database is scanned. After reading a transaction, it determines which of the itemsets that were frequent in the previous passes can be found to be contained in this transaction. New candidate itemsets are generated by extending these frequent itemsets with other items in the transaction. A frequent itemset l is extended with only those items that are frequent and occur later in the lexicographic ordering of items than any of the items in l . The candidates generated from a transaction are added to the set of candidate itemsets maintained for the pass, or the counts of the corresponding entries are increased if they were created by an earlier transaction. The steps of this algorithm are given in Figure 2.1. A faster algorithm, *Apriori-TID*, based on this approach is also found in literature [AS94].



```

Initialize
 $k = 1, C_1 = \text{all the 1-itemsets};$ 
read the database to count the support of  $C_1$  to determine  $L_1$ ;
 $L_1 = \{\text{frequent 1-itemsets}\};$ 
 $k = 2$  ;
while( $L_{k-1} \neq \phi$ )
     $C_k = \text{gen\_candidate\_itemsets}(L_{k-1});$ 
    prune( $C_k$ );
    for all transactions  $t \in T$  do
        increment the count of all candidates in  $C_k$  contained in  $t$ ;
     $L_k = \{c \mid c \in C_k \ \& \ \text{count}(c) \geq \gamma\}$ 
     $k = k + 1$ ;
return  $\forall i L_i$ .

gen_candidate_itemsets( $L_k$ ) :  $C = \phi$ 
for all itemsets  $l_1 \in L_k$  do
    for all itemsets  $l_2 \in L_k$  do
        if ( $l_1[1] = l_2[1]$ ) & ( $l_1[2] = l_2[2]$ ) & ... & ( $l_1[k-1] < l_2[k-1]$ )
            then  $c = l_1[1], l_1[2], \dots, l_1[k-1], l_2[k-1]$ 
             $C = C \cup c$ 
return( $C$ )

prune( $C$ )
for all  $c \in C$ 
    for all  $(k-1)$ -subsets  $d$  of  $c$  do
        if  $d \notin L_{k-1}$ 
            then  $C = C - \{c\}$ 

```

Figure 2.1: Apriori

2.1.2 SETM

Like Apriori [AIS93], SETM [HS95] algorithm also generates candidates on-the-fly based on transactions read from the database. It thus generates and counts every candidate itemset that the Apriori algorithm generates. However, it separates the candidate generation from counting. It saves a copy of the candidate itemset together with the transaction identifier (*TID*) of the generating transaction in a sequential structure. At the end of the pass, the support count of candidate itemsets is determined by sorting and aggregating this sequential structure.

The problem with these two algorithms was the size of the candidate sets generated of which many are often found to be infrequent. However, those algorithms are considered to be the pioneers in handling this problem.

2.1.3 Direct Hashing and Pruning

This algorithm, DHP [PCY95], uses the technique of hashing to filter out unnecessary itemsets for next candidate itemset generation. When the support of candidate k -itemset is counted by scanning the database, it accumulates information about candidate $(k + 1)$ -itemsets in advance in such a way that all possible $(k + 1)$ -itemsets of each transaction after some pruning are hashed to a hash table. Each bucket in the hash table consists of a number to represent how many itemsets have been hashed to this bucket thus far. Based on the resulting hash table, if a bit vector is constructed, where the values of one bit is set to one when the number in the corresponding entry of the hash table is greater than or equal to *minimum support*. This bit vector can be used to further reduce the number of possible candidate itemsets. The algorithmic form of DHP is presented in Figure 2.2


```

/* Part 1*/
s=a minimum support;
set all the buckets of  $H_2$  to zero; /* hash table */
for all transactions  $t \in D$  do
    insert and count 1-item occurrences in a hash tree;
    for all 2-item subsets of  $x$  of  $t$  do
         $H_2[h_2(x)] ++$ ;
 $L_1 = \{c \mid c.count \geq s, c \text{ is in a leaf node of the hash tree}\}$ ;
/*Part 2*/
k=2;
 $D_k = D$ ;
while( $|\{x \mid H_b[x] \geq s\}| \geq LARGE$ ) begin
    /* make a hash table */
    gen_candidate( $L_{k-1}, H_k, C_k$ );
    set all buckets of  $H_{k+1}$  to zero;
     $D_{k+1} = \phi$ ;
    for all transactions  $t \in D_k$  do
        count_support( $t, C_k, k, \hat{t}$ ) /* $\hat{t} \subseteq t^*$ */
        if ( $|\hat{t}| > k$ )
            make_hashtab( $\hat{t}, H_k, k, H_{k+1}, \bar{t}$ );
            if ( $|\bar{t}| > k$ )
                 $D_{k+1} = D_{k+1} \cup \{\bar{t}\}$ ;
     $L_k = \{c \in C_k \mid c.count \geq s\}$ ;
     $k ++$ ;
End while

```

```

/* Part 3 */
gen_candidate( $L_{k-1}, H_k, C_k$ );
while( $| C_k | > 0$ )
     $D_{k+1} = \phi$ ;
    for all transactions  $t \in D_k$  do
        count_support( $t, C_k, k, \hat{t}$ );
        if ( $|\hat{t}| > k$ )
             $D_{k+1} = D_{k+1} \cup \{\hat{t}\}$ ;
     $L_k = \{c \in C_k \mid c.count \geq s\}$ ;
    if( $| D_{k+1} | = 0$ )
        break;
     $C_{k+1} = \text{apriori\_gen}(L_k)$ ;
     $k++$ ;

```

Figure 2.2: Direct Hashing and Pruning

2.1.4 Partition Algorithm

Partition algorithm [SON95] is based on the observation that the frequent itemsets are normally very few in number as compared to the set of all itemsets. The whole database is divided into some partitions in such a way that they can be loaded to the memory. This algorithm works in two phases. In the first phase, frequent itemsets for every partition are derived using apriori algorithm. Since all the data of the partition can be loaded to memory it takes less time to derive the frequent itemsets from the partition. After the frequent itemsets are derived from all the partitions, in the second phase,

the frequent itemsets local to each partition are combined and their global support is counted by reading complete database once again. This algorithm needs maximum two passes over the whole database to derive the frequent itemsets. The major limitation of this algorithm is that if the number of partitions of the database is very big, then after combining the local frequent itemsets, a huge number of itemsets will be resulted, which will demand for a significant amount of memory. The algorithm has been reproduced below in Figure 2.3.

Input: Database T , minimum support γ
Output: Frequent itemsets L^G

```

p=partition_database( $T$ );
 $n$  = Number of partitions;
for  $i = 1$  to  $n$  do
    read_in_partition( $T_i$  in  $p$ )
     $L^i$  = generate all frequent itemsets of  $T_i$  using Apriori
for ( $k = 2; L_k^i \neq \phi, i = 1, 2, \dots, n; k^{++}$ ) do
     $C_k^G = \cup_{i=1}^n L_i^k$ 
for  $i = 1$  to  $n$  do
    read_in_partition( $T_i$  in  $p$ )
    for all candidates  $c \in C^G$ 
        compute  $s(c)_{T_i}$  /*support of  $c$  in all partition  $T_i$  */
 $L^G = \{c \in C^G | s(c)_{T_i} \geq \gamma\}$ 
return  $L^G$ 

```

Figure 2.3: Partition Algorithm

2.1.5 Dynamic Itemset Counting

This algorithm, DIC [BMTU97], mainly differs from the other algorithms in the candidate generation. In the other algorithms, next level candidates are generated when the current pass on the database is over. In DIC, some *stop points* are defined within the databases. If some candidate itemsets become frequent before the whole database is scanned, then using those frequent itemsets next level candidates are generated at the *stop points*. Hence the counting of the frequency of some itemsets may start from a middle position of the database. To ensure that every itemset is counted over the whole database, they are given some *stop number*. When the database is scanned only those itemsets are considered for counting the frequency, which have not completed a complete pass over the database. Four different disjoint lists of itemsets are maintained here:

DC-list of candidate itemsets.

DB-list of itemsets that are frequent but not completed a pass over the database.

SB - list of frequent itemsets which have completed a pass.

SC- list of itemsets that have completed one pass over the database and found infrequent.

During the execution of the algorithm, the following events occur when a stop point is reached.

- Some itemsets from *DC* move into *DB* if its support count has reached the minimum support.
- Some new candidates are added to *DC* which are nothing but the superset of the itemsets newly introduced into *DB*.

- Some itemsets from DC move into SC , which have completed a pass over the database, but still found infrequent.
- Itemsets from DB which have completed a pass are moved to SB .

The algorithm is formally described next in Figure 2.4.

```

SB =  $\phi$ ; // set frequent itemsets
SC =  $\phi$ ; // set of infrequent itemsets
DB =  $\phi$ ;
DC = {1-itemsets with stop number 0}
while DC  $\neq \phi$  do
    while stop point not reached
        read a transaction t
        for all itemsets  $d \in (DB \cup DC)$ 
            increment the support count of  $d$  if it is in t
        increment the current-stop-number;
        for all itemset  $d \in DC$ 
            if stop-number( $d$ )=current-stop-number
                 $SC = SC \cup d$ 
                 $DC = DC - d$ 
            else
                if count( $d$ ) $\geq \gamma$ 
                     $DB = DB \cup d$ 
                     $DC = DC - d$ 
                Generate itemsets  $E$  using  $d$ 
                stop-number( $e$ )=current-stop-number,  $\forall e \in E$ 

```

```

count(e)=0;  $\forall e \in E$ 
 $DC = DC \cup E$ 
for all itemset  $d \in DB$ 
  if stop-number( $d$ )=current-stop-number
     $SB = SB \cup d$ ;
     $DB = DB - d$ ;
return  $SB$ 

```

Figure 2.4: Dynamic Itemset Counting

2.1.6 Pincer Search

Other frequent itemset generation algorithms search for the frequent itemsets using the bottom-up approach. The computation starts from the 1-item frequent itemsets and moves upward till it reaches the largest frequent itemset. The number of database passes is equal to the largest size of the frequent itemset. When any of the frequent itemsets becomes longer, performance decreases as the number of iterations increases. To overcome this difficulty pincer search [LK98] algorithm was developed, which is based on a bidirectional search. It attempts to find the frequent itemsets in a bottom-up manner, at the same time it maintains a list of maximal frequent itemsets. While making a database pass, it also counts the support of these candidate maximal frequent itemsets to see if any one of these is actually frequent. In that event, it can be concluded that all the subsets of these maximal frequent sets are going to be frequent and, hence, they are not verified for the support count in the next pass. This algorithm is advantageous than Apriori, if the cardinality of the longest frequent itemset is large. The steps of this algorithm are given in Figure 2.5.

```

 $L_0 = \phi; k = 1; C_1 = \{\{i\} \mid i \in I\}; S_0 = \phi;$ 
 $MFCS = \{\{1, 2, \dots, n\}\}; MFS = \phi;$ 
while  $C_k \neq \phi$  and  $S_{k-1} \neq \phi$  do
    read database and find  $\text{count}(c) \forall c, c \in C_k \cup MFCS;$ 
     $MFS = MFS \cup \{m \mid m \in MFCS \text{ and } \text{count}(m) \geq \gamma\};$ 
     $S_k = \{c \mid c \in C_k \text{ and } \text{count}(c) < \gamma\}$ 
    if  $S_k \neq \phi;$ 
        call MFCS-gen
        call MFS-prune
        generate candidates  $C_{k+1}$  from  $C_k;$ 
        if any  $\{e \mid e \in C_k \text{ and } \text{count}(e) \geq \gamma\}$  was removed in MFS-prune
            call recovery over  $C_{k+1};$ 
            call MFCS-prune over  $C_{k+1};$ 
         $k = k + 1;$ 
return MFS
MFCS-gen
for all itemsets  $s \in S_k$ 
    for all itemsets  $m \in MFCS$ 
        if  $s \subset m$ 
             $MFCS = MFCS - \{m\};$ 
    for all item  $e \in s$ 
        if  $(m - \{e\} \subset p), \nexists p, p \in MFCS$ 
             $MFCS = MFCS \cup \{m - \{e\}\};$ 
return MFCS

```

Recovery

for all itemsets $l \in L_k$

for all itemsets $m \in MFS$

if first $k - 1$ items in l are also in m

/* suppose $l[k - 1] = m[j]$ */

for $i = j + 1$ to $|m|$

$C_{k+1} = C_{k+1} \cup \{l[1], l[2], \dots, l[k], m[i]\}$

MFS-prune

for all itemsets $l \in L_k$

if $(l \subset p), \exists p, p \in MFS$

$L_k = L_k - l;$

MFCS-prune

for all itemsets $c \in C_k$

if $(c \subset p), \exists p, p \in MFCS$

$C_k = C_k - c;$

Figure 2.5: Pincer Search

2.1.7 FP-tree Growth

In the above mentioned algorithms a significant amount of time is wasted in generating the candidate itemsets; and large amount of memory is required to store these candidate itemsets. For example, if there are 10,000 frequent 1-itemsets for a database then there will be roughly 10^7 number of candidate 2-itemsets. To overcome this difficulty an algorithm based on *Frequent Pattern Tree*, an extended prefix tree structure, was developed and named as FP-Tree Growth algorithm [HPY00]. The FP-Tree maintains the crucial and quantitative information about the frequent itemsets. The tree nodes are

frequent items and are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than the less frequently occurring ones. The method starts from frequent 1-itemsets as an initial prefix pattern and examines only its conditional pattern base, which consists of the set of frequent items co-occurring with the prefix pattern. The algorithm works on two phases, in the first phase, it constructs the conditional FP-Tree with respect to the given minimum support. Construction of this tree requires two passes over the whole database. In the second phase the algorithm uses the FP-Tree constructed earlier and does not use the database any more.

Since this algorithm always needs only two passes over the database, it takes less time than the other counterparts, irrespective of the size of the maximal frequent sets. The major limitation of the algorithm is the memory requirement to maintain the FP-Tree. Algorithm for construction of the FP-Tree is given in Figure 2.6

```

create root and label as null
for every  $t, t \in T$ 
    if  $t \neq \phi$ 
        call insert( $t, root$ );
        link the new nodes with existing nodes with similar label
return FP-Tree

```

```

insert( $t, n$ )
    while  $t \neq \phi$  do
        if  $n$  has a child with label  $head_t$ 
            increment link count by 1 between  $n$  and  $head_t$ 

```

```

else
    create a new child of  $n$  with label  $head_t$  with link count 1
    call insert( $body_t, head_t$ )
end do

```

Figure 2.6: FP-Tree Growth

The algorithms discussed above are capable of finding frequent itemsets from a given database. Though the way of searching for the frequent itemsets are differing, they extract the same frequent itemsets subject to a fixed *minimum support*. After generating the frequent itemsets, rules can be generated from them. For the generation of the rules another user parameter *minimum confidence* is used by the rule generation algorithms. Existing algorithms to generate the rules from a given set of frequent itemsets are discussed in the next section.

2.2 Algorithms for Generation of Rules

Algorithms discussed in the previous section are capable of finding the frequent itemsets from a given database. They provide the frequent itemsets of various sizes along with their support count. Using their support counts rules can be extracted. Here, in this section we have presented some algorithms to extract the rules from the given set of frequent itemsets. These algorithms use the user parameter *minimum confidence* while extracting the rules. Rules having the confidence value more than the specified one are declared as generated rules.

2.2.1 Agrawal's Algorithm

Agrawal et.al. [AIS93] presented the first rule generation algorithm using the frequent itemsets extracted by the first phase of the rule mining process. The algorithm was straight forward and was capable of generating only those rules with one item in the consequent part. For a given frequent itemset $Y = I_1I_2...I_k, k \geq 2$, generate at most k rules that uses the items from the set Y . The antecedent of each of these rules will be a subset X of Y such that X has $k-1$ items, and the consequent will be the *item* $Y-X$. Generate the rules $X \Rightarrow I_j$ with confidence equal or greater than *minconf*. The confidence of the rule is calculated as the ratio of $\text{support}(Y)$ and $\text{support}(X)$, where $X \cup I_j = Y$.

The major drawback of this algorithm is that it is unable to generate all the rules from the frequent itemsets. For a frequent itemset with size n , this algorithm will check maximum n candidate rules, though there can be $2^n - 2$ number of possible rules present. Effectively this algorithm checks only a small portion of the candidate rules. The steps of the algorithm are listed in Figure 2.7

1. **forall** frequent itemsets $l_k, k \geq 2$ **do**
2. **forall** $i \leq k$ **do**
3. $c = l_k[i]$
4. $a = l_k - c$
5. **if** $(\text{support}(l_k)/\text{support}(a)) \geq \text{minconf}$;
6. **declare** $a \Rightarrow c$ is a rule
7. **end do**
8. **end do**

Figure 2.7: Generating rules: Agrawal's Algorithm

2.2.2 Srikant's Simple Algorithm

This algorithm is a simple generalization of the previous algorithm. The size of the consequent part of the rules generated are not limited to one item only. To generate the rules, from a frequent itemset l , all its non-empty subsets are found first. For every such subset a , a rule is generated of the form $a \Rightarrow (l-a)$ if the ratio of $\text{support}(l)$ to $\text{support}(a)$ is at least minconf , the user specified minimum confidence. Since the frequent itemsets are stored in hash tables, the support counts for the subset itemsets can be found efficiently. The algorithm [Sri96] is reproduced in Figure 2.8.

```

for all frequent itemsets  $l_k, k \geq 2$  do
  Call  $\text{genrules}(l_k, l_k)$ 
  procedure  $\text{genrules}(l_k : \text{frequent } k - \text{itemset}, a_m : \text{frequent } m - \text{itemset})$ 
    1.  $A = \{(m-1) - \text{itemsets} \mid a_{m-1} \subset a_m\}$ 
    2. for all  $a_{m-1} \in A$  do begin
    3.    $\text{conf} = \text{support}(l_k) / \text{support}(a_{m-1});$ 
    4.   if  $(\text{conf} \geq \text{minconf})$  then begin
    5.     output the rule  $a_{m-1} \Rightarrow (l_k - a_{m-1})$ , with confidence =  $\text{conf}$ 
     and support =  $\text{support}(l_k)$ 
    6.   if  $(m-1 > 1)$  then
    7.     call  $\text{genrules}(l_k, a_{m-1})$ 
    8.   end
    9. end
  
```

Figure 2.8: Generating Rules: Srikant's Simple Algorithm

This simple algorithm is capable of generating all possible rules. But due to some redundant checking, it wastes a lot of time. For example, when itemset ABCD is used for rule generation, subset ABC, then AB then A

will lead to checking of $ABC \Rightarrow D$, $AB \Rightarrow CD$ and $A \Rightarrow BCD$ as possible rules. If the rule $ABC \Rightarrow D$ have the confidence less than *minconf*, then the confidence of $AB \Rightarrow CD$ cannot be more than *minconf*. Since the support count of AB cannot be smaller than that of ABC , second rule's confidence cannot be larger than that of the first one. But the algorithm checks for the second rule also and hence wastes some amount of time. Similarly, it also checks for the rule $A \Rightarrow BCD$.

2.2.3 Srikant's Faster Algorithm

By eliminating these unnecessary checking of rules a faster algorithm was proposed in [Sri96]. If $\bar{a} \subset a$, then the support of \bar{a} cannot be smaller than that of a . Therefore, the confidence of $\bar{a} \Rightarrow (l - \bar{a})$ cannot be more than the confidence of $a \Rightarrow (l - a)$. These facts can be rewritten as rule $a \Rightarrow (l - a)$ to hold, all the rules of the form $\bar{a} \Rightarrow (l - \bar{a})$ must also hold, $\forall \bar{a}, \bar{a} \in a$. For example, the rule $AB \Rightarrow CD$ may hold, iff both the rules $ABC \Rightarrow D$ and $ABD \Rightarrow C$ holds.

The above property states that for a given frequent itemset, if a rule with consequent c holds then the rules with consequents that are subsets of c will also hold. This property of rules is similar to the downward closure property of frequent itemsets, "*subsets of a frequent item set are also frequent*".

For example, assuming that $ACDE \Rightarrow B$ and $ABCE \Rightarrow D$ are the only one item consequent rules derived, having the minimum confidence, from the itemset $ABCDE$. If the algorithm in Figure 2.8 is used, the recursive call *genrules*($ABCDE$, $ACDE$) will test whether the two-item consequent rules $ACD \Rightarrow BE$, $ADE \Rightarrow BC$, $CDE \Rightarrow AB$ and $ACE \Rightarrow BD$ will

hold or not. But first one of these rules cannot hold, because $E \subset BE$, and the rule $ABCD \Rightarrow E$ does not have the minimum confidence. For the same reason second and third rule also cannot hold. But the call, *genrules*($ABCDE, ABCE$) of the algorithm presented in Section 2.2.2 will test if all these four rules hold or not and will find that first 3 rules do not hold. The only two item consequent rule that can possibly hold is $ACE \Rightarrow BD$, where B and D are the consequents in the valid one-item consequent rules. This is the only rule that will be tested by the algorithm given in Figure 2.9.

From a frequent itemset l , rules with one item in the consequent are generated first. Then the possible consequents with two items are generated using the apriori candidate generation function. If some rules are found here, it will generate the three item consequents, in the same manner.

1. **forall** frequent k-itemsets $l_k, k \geq 2$ **do begin**
2. $H_1 = \{\text{consequents of rules derived from } l_k \text{ with one item in the consequent}\};$
3. **Call** ap-genrules(l_k, H_1);
4. **end**

procedure ap-genrules(l_k : frequent k-itemset, H_m : set of m-item consequent)

1. **if**($k > m + 1$) **then begin**
2. $H_{m+1} = \text{result of calling apriori candidate generation function with } H_m$
3. **forall** $h_{m+1} \in H_{m+1}$ **do begin**
4. $\text{conf} = \text{support}(l_k) / \text{support}(l_k - h_{m+1});$
5. **if**($\text{conf} \geq \text{minconf}$) **then**
6. **output** the rule $(l_k - h_{m+1}) \Rightarrow h_{m+1}$ with confidence conf and $\text{support}(l_k)$

```

7.      else
8.          delete  $h_{m+1}$  from  $H_{m+1}$ ;
9.      end
10.     call ap-genrules( $h_k, H_{m+1}$ );
11. end

```

Figure 2.9: Generating Rules: Srikant's Faster Algorithm

Although, the algorithm in Figure 2.9 is considered to be the best rule generation algorithm, it also can be found to be disadvantageous due to the following difficulties. The candidate consequent to be generated for the rule discovery require a significant amount of memory. Also, a considerable time is wasted by generating the same consequent several times for different antecedents. For example, if $X \subset Y$, while generating the rules using X , all the candidate consequents will be generated. The same operation will be repeated for the set Y also, although many of them have already been generated in the earlier stage since Y is a super set of X .

So to overcome this difficulty, an effective algorithm to generate the rules was developed. This algorithm also avoids unnecessary checking of the rules and at the same time it saves some time by eliminating redundant generation of the same subsets. Since the algorithm in Figure 2.9 uses a recursive function to generate the rules, memory requirement of it is more for a longer frequent itemset. The next section presents a new algorithm to generate the rules that is efficient from both time and memory requirement point of view.

2.3 A Faster Rule Generation Algorithm

This algorithm is capable of generating all possible rules subject to the user specified minimum confidence. During the rule generation process, it avoids the unnecessary checking of some candidate rules, based on similar concepts used in the algorithm in Figure 2.9, which results in significant reduction of the time required to generate the rules. All the rules that can be found out by the third algorithm discussed above will be generated by this new algorithm also. It will use the frequent itemsets that are already stored to the memory and will not generate the subsets of a given frequent itemset. Hence, the memory requirement for this algorithm is far less than the other one. The steps of the proposed algorithm are listed in Figure 2.10.

Input: $L = \{l_k \mid l_k \text{ is the set of frequent } k\text{-itemsets, sorted on support count in descending order, } 1 \leq k \leq \text{maxsize}\}$;

minconf = the minimum confidence specified by the user.

Output: the strong association rules discovered with their support and confidence.

1. **forall** $l_k, l_k \in L, 1 \leq k \leq \text{maxsize} - 1$ **do begin**
2. $\text{reqsup} = \text{support}(l_k) * \text{minconf}$
3. $\text{found} = 0$
4. **forall** $l_m, l_m \in L, (k + 1) \leq m \leq \text{maxsize}$ **do begin**
5. **if**($\text{support}(l_m) \geq \text{reqsup}$) **then begin**
6. **if**($l_k \subset l_m$) **then begin**
7. $\text{found} = \text{found} + 1$
8. $\text{conf} = \text{support}(l_m) / \text{support}(l_k)$
9. generate the rule $l_k \Rightarrow (l_m - l_k)$, with $\text{confidence} = \text{conf}$
and $\text{support} = \text{support}(l_m)$


```

10.         end if
11.     else
12.         if(found<2)
13.             continue step 1 with next k
14.         else
15.             found=0;
16.         endif
17.     endif
18. end do
19. end do

```

Figure 2.10: NBG: A Faster Rule Generation Algorithm

This algorithm is capable of discovering all possible rules from the given set of frequent itemsets subject to a user specified minimum confidence. It discovers all rules with a fixed antecedent and with different consequents. For that it checks only those frequent itemsets which can fulfill the minimum confidence. At the same time, the algorithm will go to the next level of frequent itemset with the same antecedent if in the current level at least two itemsets fulfil the minimum confidence. This eliminates a number of unnecessary checks for the rule.

The algorithm is generating the rules with a fixed antecedent first, starting with single item antecedent. When all the rules with that antecedent are generated then it will go to the next antecedent. For the same antecedent it checks for the rules with equal size consequent, starting with one item consequent, then go to the next level. For, a given antecedent if all rules in level k , where k is the number of items in the consequent, have confidence less

than the threshold, i.e. no rules are generated, then the confidence of any rule in level $k+1$ also cannot be more than threshold. So checking for rules from this level onward can be avoided without missing any rules. If in level k , one rule fulfilled the threshold, then also in $(k+1)^{th}$ level there will be no rule. Because an itemset in the $(k+1)^{th}$ level is generated by combining two of the k^{th} level itemsets. Now the maximum possible confidence of the rule in the $(k+1)^{th}$ level will be minimum confidence of the two itemsets from which this is constructed. Since the confidence of only one of them is larger than the threshold, others must be less than the threshold. So the confidence of the rule in level $k+1$ will be less than the threshold. So, it is needless to check for the rules in the next level without missing any valid rule. It can then be concluded that the algorithm is complete.

2.3.1 Experimental Results

All the above mentioned algorithms were tested with some synthetic databases as well as standard databases. An Intel Core2duo 2.5 GHz processor based computer with 3 GB RAM was used for this purpose. To compare the performances of these algorithms, frequent itemsets derived from a test dataset using Apriori algorithm are presented in Table 2.1. Based on these frequent itemsets, the rules extracted by the algorithms presented in Figure 2.7, 2.8 and 2.9 are reported in Tables 2.2, 2.3 and 2.4 respectively.

Based on these experimental results it can be easily observed that the algorithm in Figure 2.9 gives the best performance among them. So, this algorithm was considered as the counterpart while evaluating the performance of the proposed algorithm.

Table 2.1: Frequent Itemsets Derived: An Example

Itemset Size	Itemset	Support Count
1	1	2
1	2	6
1	3	6
1	4	4
1	5	8
1	6	5
1	7	7
1	8	4
1	9	2
2	5, 6	3
2	5, 7	5
2	6, 7	3
3	5, 6, 7	1

From these experimental results it can be observed that algorithms in Figure 2.8 and Figure 2.9 produces the same rules, may be in different order. But the algorithm in Figure 2.7 missed some of the rules that are derived by the other two. Again the algorithm in Figure 2.9 executes faster than the algorithm in Figure 2.8. The new algorithm, NBG, also produces the same rules, takes less time even than Figure 2.9.

To compare the performance of NBG some other synthetic datasets were considered. Initially, using the *Apriori* algorithm frequent itemsets were discovered from the database. Table 2.5 contains the frequent itemsets derived

Table 2.2: Rules Derived by Agrawal's Algorithm

Rule	Support	Confidence
$5 \Rightarrow 6$	3	0.375000
$5 \Rightarrow 7$	5	0.625000
$6 \Rightarrow 7$	3	0.600000
$5, 6 \Rightarrow 7$	1	0.333333

from a synthetic dataset having 10000 records and 20 attributes in it. While deriving them, a minimum support of 20% was considered.

The frequent itemsets summarized in Table 2.5 are used to discover the rules. Table 2.6 contains the results of different execution of the algorithms with different values of confidence. The timing information provided in Table 2.6 are average of 15 runs of the program.

Similar comparison of above two algorithms was done using two standard datasets namely Monks-1 and Monks-3. Summary of the frequent itemsets with minimum support 10% derived from them were listed in Tables 2.7 and 2.9. Tables 2.8 and 2.10 contains the summary of the rule extraction process with different values of minimum confidence.

2.3.2 Observations

Based on our experimental study, following observations can be made:

- Same numbers of rules are generated by both the algorithms even if

Table 2.3: Rules Derived by Srikant's Simple Algorithm

Rule	Support	Confidence
$6 \Rightarrow 5$	3	0.600000
$5 \Rightarrow 6$	3	0.375000
$7 \Rightarrow 5$	5	0.714286
$5 \Rightarrow 7$	5	0.625000
$7 \Rightarrow 6$	3	0.428571
$6 \Rightarrow 7$	3	0.600000
$6, 7 \Rightarrow 5$	1	0.333333
$5, 7 \Rightarrow 6$	1	0.200000
$5, 6 \Rightarrow 7$	1	0.333333
$6 \Rightarrow 5, 7$	1	0.200000

the orders of generation are different. It is evident from columns 2 & 4 of Tables 2.6, 2.8 and 2.10.

- Time taken by the proposed algorithm is significantly less than the other. It is evident from columns 3 & 5 of Tables 2.6, 2.8 and 2.10.
- The time saved by the NBG is more significant with the increasing size of the maximal frequent set and the total number of frequent itemsets.

2.4 Discussion

In this chapter, some of the algorithms used in the first phase of association rule mining in the classical approach, i.e. frequent itemsets mining, were presented. These algorithms have their own advantages and limitations. They

Table 2.4: Rules Derived by Srikant's Faster Algorithm

Rule	Support	Confidence
$6 \Rightarrow 5$	3	0.600000
$5 \Rightarrow 6$	3	0.375000
$7 \Rightarrow 5$	5	0.714286
$5 \Rightarrow 7$	5	0.625000
$7 \Rightarrow 6$	3	0.428571
$6 \Rightarrow 7$	3	0.600000
$6, 7 \Rightarrow 5$	1	0.333333
$6 \Rightarrow 5, 7$	1	0.200000
$5, 7 \Rightarrow 6$	1	0.200000
$5, 6 \Rightarrow 7$	1	0.333333

can derive the frequent itemsets from any given market basket dataset. After finding those frequent itemsets, rules are generated. This chapter also discussed three existing algorithms used for the generation of rules. Among these three, the third one can be found to be the best, because it does not miss any possible rule, and also works in a faster way.

This chapter finally presents an effective algorithm (NBG) which has been established to be capable of extracting the rules even faster than the previously known best algorithm.

All these algorithms to handle association rule mining problem, presented in this chapter, considers the problem as a single objective problem. The confidence of the rules was the main objective of the problem that is optimized.

Table 2.5: Frequent Itemset from a Synthetic Dataset

Itemset Size	Number of Itemsets
1	18
2	133
3	216
4	43

Table 2.6: Rules from the Synthetic Dataset

Min Conf.	NBG		Srikant's Faster Algorithm	
	Rules	Time(ns)	Rules	Time(ns)
20%	2250	89,666,000	2250	1,263,772,000
30%	1822	25,691,000	1822	97,006,000
40%	1379	21,022,000	1379	47,039,000
50%	951	17,669,000	951	36,186,000
60%	580	12,605,000	580	25,487,000

But the association rule mining can be treated as a multi-objective problem. In the next chapter we discuss about some other objectives of the association rules that should also be optimized along with confidence. The chapter will also discuss about some techniques to handle the multi-objective problems.

Table 2.7: Frequent Itemset from monks-1 Dataset

Itemset Size	Number of Itemsets
1	17
2	94
3	20

Table 2.8: Rules from monks-1 Dataset

Min Conf.	NBG		Srikant's Faster Algorithm	
	Rules	Time(ns)	Rules	Time(ns)
20%	252	7,989,000	252	8,474,000
30%	180	7,307,000	180	7,616,000
40%	99	6,453,000	99	6,674,000
50%	55	6,016,000	55	6,242,000
60%	13	5,498,000	13	5,830,000

Table 2.9: Frequent itemset from monks-3 Dataset

Itemset Size	Number of Itemsets
1	17
2	93
3	19

Table 2.10: Rules from monks-3 Dataset

Min Conf.	NBG		Srikant's Faster Algorithm	
	Rules	Time(ns)	Rules	Time(ns)
20%	241	7,912,000	241	8,285,000
30%	172	7,280,000	172	7,433,000
40%	90	6,391,000	90	6,569,000
50%	52	6,025,000	52	6,251,000
60%	11	5,506,000	11	5,734,000

Chapter 3

Multi Objective Association Rule Mining

Among most of the real life problems of optimization only one measure or objective is needed to be optimized. For handling them, some straight forward algorithms can be used to get the optimal solution. But the situation becomes more complicated when the problem have more than one measures to evaluate and optimize all these measures simultaneously. Due to the demands in different fields of real life applications, multi objective optimization has got a lot of attention.

3.1 Multi Objective Problems

Multi objective optimization problems can be found in various fields: product and process design, finance, aircraft design, the oil and gas industry, automobile design, or wherever optimal decisions need to be taken in the presence of trade-offs between two conflicting objectives. Maximizing profit

and minimizing the cost of a product; maximizing performance and minimizing fuel consumption of a vehicle; and minimizing weight while maximizing the strength of a particular component are examples of multi-objective optimization problems. In these types of problems it is practically impossible to find the *best* solution, but some optimal solutions can be found.

The example in Table 3.1 will clarify the situation. Let us consider the problem of traffic controlling system in a city. One objective of it is to reduce the number of accidents, and the other objective is to reduce the cost incurred to implement the system.

Table 3.1: Multi Objective Problem : An Example

Investment per year (Lakh Rs.)	0	10	20	30	40	50
Number of accidents per year	1000	700	500	350	250	175

For this problem, some of the probable solutions are tabulated in Table 3.1. Each column of the table represents a solution of the problem. But none of them can be called as the best. The problem mentioned here is a minimization problem, where we have to minimize both the cost and number of accidents. But, if cost is reduced number of accidents will increase. Any solution of the problem, if it is better than another; in one objective, than it is worse in the other objective. For example, the last solution seems to be the best from the number of accidents point of view. But this is the worst one from cost point of view. So, none of the solutions here is the best. But every solution works as an optimal solution of the problem and the decision makers can choose any of them depending on the situation.

3.2 Handling Multi Objective Problems

Perhaps the most intuitive approach to solve the multi objective problem is *constructing a single aggregate objective function*. The basic idea is to combine all the objective functions into a single functional form, called the *AOF* [JT02]. There are also different variations in defining the AOF. Simple linear aggregation of the objectives also can be used in some problems. Another well-known combination is the weighted linear sum of the objectives. Here, in this approach some scalar weights for each objective to be optimized are specified, and then they are combined into a single function that can be solved by any single-objective optimizer. Clearly, the solution obtained will depend on the values (more precisely, the relative values) of the weights specified. For example, if we are trying to maximize the strength of a machine component and minimize the production cost, and if we specify a higher weight for the cost objective compared to the strength, our solution will be one that favors lower cost over higher strength. Thus, it may be noticed that the weighted sum method is essentially subjective, in that a decision manager needs to supply the weights [Fla76]. Depending on the weights supplied for the different objectives we will get a different solution. Hence the selection of the suitable weights for different objectives again becomes a crucial problem.

Goal programming is a branch of multi objective optimization. It can be thought of as an extension or generalization of linear programming to handle multiple, conflicting objective measures. Each of these measures is given a goal or target value to be achieved. Unwanted deviations from this set of target values are then minimized in an achievement function. This was first introduced in [CCF55], although the actual name first appeared in [CC61]. The first engineering application of goal programming was the design and

placement of the antennas employed on the second stage of the Saturn V. This was used to launch the Apollo space capsule that landed the first men on the moon.

The initial goal programming formulations ordered the unwanted deviations into a number of priority levels, with the minimization of a deviation in a higher priority level being infinitely more important than any deviations in lower priority levels. This is known as lexicographic goal programming [Ign76]. Lexicographic goal programming should be used when there exists a clear priority ordering amongst the goals to be achieved. A major strength of goal programming is its simplicity and ease of use. This accounts for the large number of goal programming applications in many and diverse fields. Goal programming can hence handle relatively large number of variables, constraints and objectives. A debated weakness is the ability of goal programming to produce solutions that are not *Pareto efficient*.

Since we cannot define the clear priority ordering of the different objectives of the association rule mining, lexicographic goal programming will not help us here.

It is better to find out the solutions for these type of problems depending on non-dominance criterion [Coe99, FF95, ZDT00]. At the time of taking a decision, the solution that seems to fit better depending on the circumstances can be chosen from the set of these candidate solutions. A solution, say a , is said to be dominated by another solution, say b , if and only if the solution b is better or equal with respect to all the corresponding objectives of the solution a , and b is strictly better in at least one objective. Here the solution b is called a non-dominated solution. So it will be helpful for the decision-

maker, if we can find a set of such non-dominated solutions. This approach of solving the multi objective problem was suggested by French economist Vilfredo Pareto [Par96]. After his name this technique of optimization was termed as *Pareto optimization technique*.

3.3 Genetic Algorithms as a Tool

When we are looking for a solution for a problem, it is better to check the complete solution space for the best solution. But in some problems the exhaustive search is practically infeasible, if the search space is too big. When the search space is linear in nature, some kind of filtering can be done to eliminate some subspace of the total search space. But when the search space is multi-dimensional, those types of filtering also do not help too much.

To attend this type of complicated problems, based on natural evolution, Genetic Algorithms were suggest by John Henry Holland [Hol75]. Basically, genetic algorithms are implemented as a computer simulation in which a *population* of abstract representations, called *chromosomes*, of candidate solutions, called *individuals* to an optimization problem evolves toward better solutions. Solutions are represented in binary as strings of 0s and 1s. The evolution usually starts from a population of randomly generated individuals, called *initial population* and moves in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are randomly selected from the current population, based on their fitness. To form a new population, those chromosomes are combined pairwise, which are selected randomly. These new chromosomes are then randomly mutated and is used in the next iteration of the algorithm. Commonly, the algorithm ter-

minates when either a maximum number of generations have been produced, or a satisfactory fitness level has been reached for the population.

3.4 Multi Objective Genetic Algorithms

Several works have been found in the literature to establish it that the genetic algorithms can be used as an efficient tool to handle the multi-objective problem [Sha85, FF93, SD93, Coe96, Deb01, CLV07]. A few of them are discussed here in this section.

3.4.1 Vector Evaluated Genetic Algorithm

David shaffer [Sha85] extended Grefenstette's GENESIS program [xx171xx] to include multiple objective functions. Shaffer's approach was to use an extension of the Simple Genetic Algorithm(SGA) that he called the Vector Evaluated Genetic Algorithm (VEGA), and that differed of the first only in the way in which selection was performed. This operator was modified so that at each generation a number of sub-population was generated by performing proportional selection according to each objective function in turn. Thus, for a problem with k objectives, k sub-populations of size N/k each are generated, assuming a total population size of N . These sub-populations are shuffled together to obtain a new population of size N , on which crossover and mutation operators are applied in the usual way. The solutions generated by this approach are non-inferior in local sense, as their non-inferiority is limited to the current population, and while a locally dominated are globally dominated also. Another problem, termed as "speciation", may also arise in this approach (i.e., evolution of "species" within the population which excel on different aspects of performance). This problem may arise because this

technique selects individuals who excel in one dimension of performance, without looking at other dimensions. The potential danger in doing that is that there may be some individuals with “middling” performance in all objectives, which could be very useful, but that will not survive under this selection scheme, since they are not in the extreme for any objective (i.e., they do not produce best value for any objective function, but only moderately good values for all of them). To deal with this problem some heuristics was suggested. For example, to use a heuristic selection preference approach for non-dominated individuals in each generation, to protect the “middling” individuals. Also, crossbreeding among the “species” could be encouraged by adding some mate selection heuristics instead of using the random mate selection of traditional Genetic Algorithms.

3.4.2 Multi Objective Genetic Algorithm

Based on ranking of the individuals, Fonseca and Fleming [FF93] have proposed this scheme. The rank assigned to an individual of the current population is determined by the number of individuals by which it is dominated. For example, if an individual x of generation g are dominated by p^g individuals, then $Rank(x, g) = 1 + p^g$. Hence, all the non-dominated individuals are ranked as 1. Then the individuals are assigned some fitness based on their ranks. Proper care must be taken so that the chromosomes with same ranks gets equal fitness.

As Goldberg [Gol89] and Deb [SD94] point out, this type of blocked fitness assignment is likely to produce a large selection pressure that might produce premature convergence. To avoid that Fonseca and Fleming used a niche-formation [FF95] method to distribute the population over the Pareto-optimal region. This maintains diversity in the objective function values, but

may not maintain diversity in the parameter set, which is an important issue for a decision maker. Furthermore, this approach may not be able to find multiple solutions in problems where different Pareto-optimal points corresponds to the same objective function value.

In this approach, it is possible to evolve only a certain region of the trade off surface, by combining Pareto dominance with partial preference information in the form of a goal vector.

3.4.3 Non-dominated Sorting Genetic Algorithm

This algorithm, NSGA [SD93], is based on several layers of classification of the individuals. Before the selection is performed, the population is ranked on the basis of non-domination. All non-dominated individuals are classified into one category and assigned some dummy fitness. To maintain the diversity of the population, these classified individuals are shared with their dummy fitness values. Then this group of classified individuals is ignored and another layer of non-dominated individuals is considered. The process continues until all individuals in the population are classified.

A stochastic remainder proportionate selection was used for this approach. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. This allows to search for non-dominated regions, and results in quick convergence of the population towards such regions. The efficiency of NSGA lies in the way multiple objectives are reduced to dummy fitness function using a non-dominated sorting procedure. With this approach, any number of objectives can be solved and both maximization and minimization problems can be handled.

3.5 Multiple Objectives of Association Rule Mining

Association rule mining is a multi objective problem. *Confidence* or *predictive accuracy*, *comprehensibility* and *interestingness* of the rules can be used as the objectives of the rule mining problem.

Confidence

This measure is commonly taken as the objective to be optimized by the classical approach of rule mining. This is defined as the ratio of the support of the rule to the support of the antecedent part of the rule. For example, if $A \Rightarrow C$ is a rule and $SUP(A)$ and $SUP(A \cup C)$ are the support of the antecedent and the rule respectively, then the confidence of the rule, $conf = SUP(A \cup C)/SUP(A)$. It defines the probability of finding C in a record of the database when it contains A . Hence this measure is sometimes termed as *predictive accuracy* also.

Comprehensibility

This measure of evaluating the rules defines how much understandable the rule is. Since the association rules are represented as IF-THEN statements, some conditions are involved there. It is easier to understand the statement, if the number of conditions involved are less. If the rules are extracted by the classical approach, sometimes a large number of conditions are involved there. So it becomes very difficult to understand the rule. As a result the rule will not be used by the decision makers.

Though it is very difficult to quantify the *understandability* or *compre-*

hensibility, a careful study of an association rule will infer that, if the number of conditions involved in the antecedent part is less, the rule is easy to understand, in otherwords more *comprehensible*. To reflect this behaviour, an expression was derived as $comp=N-$ (*number of conditions in the antecedent part*) [FLFG00]. This expression serves well for the classification rule generation [Fre01] where the number of attributes in the consequent part is always one. Since, in the association rules, the consequent part may contain more than one attribute, this expression is not suitable for the association rule mining. So the need for a new expression to quantify the comprehensibility of association rule was felt, where the number of attributes involved in both the parts of the rule has some effects. Several expressions were designed during this dissertation work to quantify the comprehensibility, and finally the following expression was found to be better among them[GN04]. The expression for comprehensibility of an association rule–

$$Comprehensibility = \log(1+ | C |)/\log(1+ | A \cup C |).$$

Here, $| C |$ and $| A \cup C |$ are the number of attributes involved in the consequent part and the total rule, respectively.

Interestingness

Since association rule mining is a part of data mining process that extracts some hidden information, it should extract only those rules that have comparatively less occurrences in the entire database. Such a surprising rule may be more interesting to the users; which again is difficult to quantify. For classification rules it can be defined by information gain theoretic measures [Fre02]. This way of measuring interestingness for the association rules will become computationally inefficient. For finding interestingness, the database is to be divided based on each attribute present in the consequent part. Since

a number of attributes can appear in the consequent part and they are not predefined, this approach is not feasible for association rule mining. So a new expression was defined [GN04] which uses only the support count of the antecedent and the consequent parts of the rules, and is defined as

$$\begin{aligned} \text{Interestingness} = & [SUP(A \cup C)/SUP(A)] \times [SUP(A \cup C)/SUP(C)] \\ & \times [1 - (SUP(A \cup C)/|D|)]. \end{aligned}$$

where $|D|$ is the total number of records in the database.

This expression contains three parts. The first part, $[SUP(A \cup C)/SUP(A)]$, gives the probability of generating the rules depending on the antecedent part, the second part, $[SUP(A \cup C)/SUP(C)]$, gives the probability of generating the rule depending on the consequent part, and $(SUP(A \cup C)/|D|)$ gives the probability of generating the rule depending on the whole database. So, the complement of this probability will be the probability of *not generating* the rule. Thus, a rule having a very high support count will be measured as less interesting.

3.5.1 Efficient MOGA for Association Rule Mining

This section presents a Pareto based genetic algorithm to extract the association rules. To work with the genetic algorithm it is necessary to represent the candidate solutions as chromosomes, for which a suitable encoding/decoding scheme is required. For the association rule mining problem the candidate solutions are nothing but the possible rules. To encode rules two approaches can be adopted. In the *Pittsburgh approach* [GS93] each chromosome represents a set of rules. The length of the chromosome limits the number of rules generated. This approach is more suitable for classification rule mining; as we do not have to decode the consequent part. The other approach

is called the *Michigan approach* [NFL99] where each chromosome represents a separate rule. In the original Michigan approach we have to encode the antecedent and consequent parts separately; and thus this is an inefficient representation from the point of space utilization. As it is not known a priori, which attributes will appear in which part, space should be reserved for every attribute in both the parts, antecedent and consequent. Again the same attribute cannot appear in both the parts. As a result, at least half of the reserved space within the chromosome will never be utilized. So we followed a new approach that is better than this, from the point of storage requirement. With each attribute two extra tag bits are associated. If these two bits are **00** then the attribute, next to these two bits, appears in the antecedent part and if it is **11** then the attribute appears in the consequent part. And the other two combinations, **01** and **10** will indicate the absence of the attribute in either of these parts. For example, the representation of the rule $AC \Rightarrow BF$ from a database with attributes ABCDEF, will become **00A 11B 00C 01D 10E 11F**. In this way we can handle variable length rules with more storage efficiency, with an overhead of only $2k$ bits, where k is the number of attributes in the database.

The next step in the chromosome representation is to find a suitable scheme for encoding/decoding the attribute values of the rules to/from binary chromosomes. In the above representation of the rules the positions of attributes are fixed. Hence, the attribute names are not needed to be encoded in the chromosome. Values of different attributes are to be encoded in the chromosome only. For encoding a *categorical* or *nominal* valued attribute, the market basket encoding scheme is used. But this scheme is not suitable for numeric valued attributes. For a *continuous* or *real* valued attribute their binary representation is used as the encoded value. The range of values of that attribute and the desired accuracy level controls the number of bits re-

quired to encode the attribute's value. Decoding can be performed as–

$$Value = min + (max - min) ((2^i - 1 \times i^{th} \text{ bit value}) / (2^n - 1)),$$
where $1 \leq i \leq n$ and n is the number of bits used for encoding; min and max are minimum and maximum values of the attribute.

Using these encoding schemes values of different attributes can be encoded into the chromosomes. Since in the association rules an attribute may be involved with different relational operators [Oli93], it is better to encode them also within the rule itself. For example, in one rule a numeric attribute A may be involved as $A \geq value1$, but in another rule it may be involved as $A \leq value2$. Similarly, a categorical attribute may be involved with either equal to ($=$) or not equal to (\neq). To handle this situation another bit is used to indicate the operators involved with the attribute. Equality ($=$) and not equality (\neq) are not considered with the numerical attribute. In this way the whole rule can be represented as a binary string, and this binary string will represent one chromosome or a possible rule.

After getting the chromosomes, various genetic operators such as *crossover*, *mutation*, *selection* are performed on it. Presence of large number of attributes in the records result in long chromosomes, which demands for multi-point crossover needed to bring more diversity within the chromosomes.

There are some difficulties to use the standard multi-objective GAs for association rule mining problem. In case of rule mining problem, a set of better rules extracted from the database needs to be stored. If the standard genetic operations are used, then the final population may not contain some better rules generated at some intermediate generations. Hence it is better to keep those better rules of the intermediate generations. For this task, a sep-

arate population can be used [ES91]. In this population no genetic operation is performed. It will simply contain only the non-dominated chromosomes of the previous generations. At the end of first generation, it will contain the non-dominated chromosomes of the first generation. After the next generation, it will contain those chromosomes, which are non-dominated among the current population as well as among the non-dominated solutions till the previous generation.

Steps of the algorithm to extract the rules are enumerated in Figure 3.1.

Input: Database D , number of generations G

Output: A set of nondominated rules

1. Load S , $S \subset D$
2. Generate population P of N chromosomes randomly.
3. Decode(p_i), $\forall i, p_i \in P$
4. find $SUP(A)$, $SUP(C)$ and $SUP(R)$ by scanning S
5. Calculate the confidence, comprehensibility and interestingness values.
6. Rank(p_i), $\forall i, p_i \in P$ based on non-dominance property
7. fitness(p_i)= $| P | - \text{Rank}(p_i)$, $\forall i, p_i \in P$
8. $\forall i, p_i \in P$, if Rank(p_i)=1 then Maintain_Elite(p_i)
9. Based on fitness(p_i) select cromosomes for next generation
10. Perform multi-point crossover, then mutation to get new population O .
11. Replace population P by O
12. If *number of generations* $< G$, then go to Step 3.
13. Decode and return Elite Chromosomes.

Figure 3.1: GA based Multi Objective Rule Mining

Implementation and Results

Many experiments were carried out to set the various parameters of the algorithm. A computer with Intel Core2Duo 2.5 GHz processor and 3 GB RAM was used to perform those experiments. During those experiments, the parameters affecting the genetic algorithm were tuned for different values. From these results it was observed that for all most all datasets the optimal results are derived, if the crossover and mutation probability are tuned nearer to 0.8 and 0.002 respectively. The results presented in this dissertation, are based on these values of the said parameters. Similarly, the population size was fixed at 40. To keep the diversity within the population multi-point crossover was performed, where the number of crossover points varies with chromosome length. Based on various experiments, one crossover point per 100 bits of chromosome was found to be most effective.

The algorithm was tested over several synthetic and standard datasets. The synthetic datasets were generated based on probabilistic measures. Every dataset differs from others in number of attributes, number of records and ranges of attribute values. Satisfactory results were obtained from those experiments. Some results from a dataset, *kddcup.data.10.percent*, publicly available at UCI Machine Learning Repository (<ftp://ftp.ics.uci.edu/pub/machine-learning-databases/>) are given in Table 3.2. The dataset contains total 41 attributes in it, out of them 34 attributes are numeric remaining attributes are symbolic. The dataset is conventionally used for classification problem. Only the numeric valued attributes were considered during rule extraction.

Table 3.2: Summary of results

Sample size	Number of generations	Number of rules generated
1000	100	24
	200	31
	300	31
1000	100	35
	200	40
	300	40
1000	100	27
	200	36
	300	37
2000	100	35
	200	40
	300	40
2000	100	35
	200	40
	300	40

A few rules from this datasets with a sample size of 2000 are given in given in Table 3.3. For better understanding of the rules, the complete name of the attributes involved in these rules are given in Table 3.4.

Discussion

From the rule sets generated for different samples and for different number of generations it can be observed that after 200 generations it ceases to gen-

Table 3.3: Rules from kddcup.data_10_percent dataset

$hot \leq 0.0 \ \& \ count \geq 309.82$	\rightarrow	$Dst_host_count \geq 111.98$
$hot \geq 0.0 \ \& \ count \geq 1.18 \ \& \ Dst_host_error_rate \leq 0.01$	\rightarrow	$srv_diff_host_rate \leq 0.0 \ \& \ Dst_host_count \leq 41.06$
$Dst_host_srv_diff_host_rate \geq 0.12 \ \& \ Dst_host_count \leq 8.05 \ \& \ Dst_host_srv_serror_rate \leq 0.0$	\rightarrow	$srv_diff_host_rate \geq 0.0 \ \& \ Dst_host_srv_count \geq 18.63$
$serror_rate \geq 1.36 \ \& \ dst_bytes \leq 82.48 \ \& \ num_file_creations \leq 0.01$	\rightarrow	$Dst_host_count \leq 6.67 \ \& \ Dst_host_srv_serror_rate \leq 0.0$

erate more rules; in other words after that number of generations the GA converges. From the results given above it can be seen that only for the third sample, it gives an extra rule at the cost of 100 additional generations. Moreover, only a very few number of attributes (4 to 5 attributes on both the antecedent and consequent parts) got involved in the rules, which means that all the attributes are not equally important; and the rules are simple to understand (i.e. comprehensible).

This section presented an effective approach to discover the association rules based on the multi objective genetic algorithm. But that algorithm has a difficulty in it. To save the time by doing multiple disk accesses, it has loaded a sample of the database to the memory. Since the algorithm works on the memory resident data it works faster. But the rules that were discovered by the algorithm may not reflect the whole database and is widely affected by the sampling technique used to select the sample. Hence, it demands for

Table 3.4: Attributes of kddcup dataset involved in the reported rules

No.	Name	Description
6	dst_bytes	Number of data bytes from destination to source
10	hot	Number of “hot” indicators
17	num_file_creations	Number of file creation operations
23	count	Number of connections to the same host as the current connection in the past two seconds
25	serror_rate	Percentage of connections that have “SYN” errors
31	srv_diff_host_rate	Percentage of connections to different hosts
32	Dst_host_count	Number of connections to host
33	Dst_host_srv_count	Number of services requested of host
37	Dst_host_srv_diff_host_rate	Percentage of connections with same service but to different Host
39	Dst_host_srv_serror_rate	Percentage of connections to the same service that have “SYN” errors
40	Dst_host_rerror_rate	Percentage of connections that have “REJ” errors

an algorithm which can handle dataset efficiently at minimum computational cost. Next section presents a horizontal partitioning approach to address this issue.

3.5.2 MOGA Based Partitioning Approach

In this approach the whole database is used during the rule extraction process. To minimize the disk access time, the whole database was divided into some *horizontal partitions*, where the size of the partitions were selected in such

a way that it could be accommodated within the memory. Then the rules from the first partition were extracted. Since memory resident data are used, the rule extraction process becomes faster. After storing these rules, next partition was used to extract the rules from it. And this process continues till all the partitions are used. Finally, the rules from all the partitions are combined, and the redundant rules are eliminated. Then the whole database is read once more from the disk to evaluate the rules.

However, chromosome representation, encoding/decoding scheme and all other genetic operators used in the previous algorithm were strong enough to handle the problem. So they were used in this partitioning approach also.

The steps of the algorithm are given in Figure 3.2.

Input: Database D , number of generations G

Output: A set of non-dominated rules

1. Create n partitions s_i of D such that $\bigcup_{i=1}^n s_i = D$
2. $i = 1$
3. Load the partition p_i to memory
4. Generate population P of N chromosomes randomly
5. $gen = 0$
6. Decode(p_j), $\forall j, p_j \in P$
7. Find $SUP(A)$, $SUP(C)$ and $SUP(R)$ from s_i
8. Calculate *confidence*, *comprehensibility* and *interestingness*
9. Rank(P_j), $\forall j, p_j \in P$
10. Fitness(p_j) = $|P| - \text{Rank}(p_j)$, $\forall j, p_j \in P$
11. $\forall j, p_j \in P$, if Rank(p_j)=1 then Maintain_Elite(p_j)
12. Based on Fitness(p_j) select chromosomes for next generation

13. Perform multi-point crossover, then mutation to get new population O
14. Replace population P by O
15. $gen = gen + 1$
16. if $gen < G$ then go to *Step 6*
17. $P^i = P$
18. $i = i + 1$
19. if $i \leq n$ go to *Step 3*
20. $Q = \bigcup_{i=1}^n P^i$
21. Find $SUP(A)$, $SUP(C)$ and $SUP(R)$ by scanning D , $\forall q_j \in Q$
22. Calculate *confidence*, *comprehensibility* and *interestingness*, $\forall q_j \in Q$
23. return Q

Figure 3.2: MOGA based partitioning algorithm

Implementation and Results

The above mentioned algorithm was implemented and tested over various synthetic as well as standard databases in the same environment as the earlier one given in the previous section. From the results of various databases it was observed that this algorithm can extract the rules from a database that were extracted by the previous one also. But their measures are not the same, as in the first algorithm objectives were evaluated on a sample of the database. Here some of the results from some standard public databases are presented. Crossover probability of 0.8, mutation probability of 0.01, population size of 40 was used during the extraction of the rules. 12 bits were used to encode each attribute. The databases presented below were downloaded from UCI Machine learning repository (<ftp://ftp.ics.uci.edu/pub/machine-learning-databases/>).

A. Wisconsin Diagnostic Breast Cancer Database

This dataset contains 569 instances and 32 attributes. First and second attributes are sample code number and class label respectively. All other 30 attributes are real valued attributes. Out of these 569 instances, 357 are benign and 212 are of malignant types. Some rules extracted from this database by the algorithm are presented in Table 3.5.

Table 3.5: Some rules from Wisconsin Diagnostic Breast Cancer Database

Mean perimeter ≤ 183.023 & Worst area ≤ 3924.124	\Rightarrow	Mean texture ≤ 30.579 & Standard error perimeter ≤ 15.730 & Worst compactness ≤ 0.944 & Worst symmetry ≥ 0.338
Mean radius ≥ 7.029 & Standard error perimeter ≤ 10.94 & Worst symmetry ≥ 361.012	\Rightarrow	Mean area ≥ 249.27 & Standard error smoothness ≤ 0.016
Mean radius ≤ 12.895 & Standard error radius ≤ 0.8346 & Worst area ≤ 901.53	\Rightarrow	Mean concave points ≤ 0.1152 & Standard error perimeter ≤ 6.217 & Standard error fractal dimension ≥ 0.001

When these rules were observed we found that in 84 instances the first rule is satisfied and out of these 84 instances 66 (78.57%) are of malignant type. Similarly, second rule is satisfied by 56 instances out of which 52 (92.86%) instances are Malignant. And out of 248 instances that satisfy the fourth rule 236 (95.16%) instance are of benign type. This is an interesting observation, as these rules may be used for classification also.

B. Wisconsin Breast Cancer Database

This dataset contains 699 instances and 11 attributes. First and last are sample code number and class attribute respectively. All other attributes are real valued attributes. Out of these 699 instances, 458 are Benign type and 241 are of Malignant type. Some of the rules extracted from it are presented in Table 3.6.

Table 3.6: Some rules from Wisconsin Breast Cancer Database

Uniformity of Cell Shape ≤ 8.248	\Rightarrow	Marginal Adhesion ≥ 2.708 & Mitoses ≤ 5.730
Single Epithelial Cell Size ≤ 9.888	\Rightarrow	Marginal Adhesion ≥ 4.571 & Mitoses ≤ 5.207
Bland Chromatin ≤ 8.842	\Rightarrow	Clump Thickness ≥ 0.972 & Normal Nucleoli ≥ 4.774 & Mitoses ≤ 5.642

When these rules were observed, it was found that in 168 instances the first rule is satisfied and out of these 168 instances 122 (72.62%) are of malignant type. Similarly second rule is satisfied by 119 instances out of which 109 (91.6%) instances are Malignant. And out of 63 instances that satisfy the third rule 52 (82.54%) instance are of benign type.

C. Wisconsin Prognostic Breast Cancer Database

This dataset was also taken from the UCI machine learning data repository. This dataset contains 198 instances and 34 attributes. First and second are sample code number and class attribute respectively. All other 32 attributes are real valued attribute, values lying in different ranges. Out of these 198

instances, 151 are non-recurring and 47 are recurring type. When the rule extraction algorithm was applied on it, some interesting rules were discovered. A few of them are presented in Table 3.7.

Table 3.7: Some rules from Wisconsin Prognostic Breast Cancer

Standard error texture ≤ 3.275 & Standard error smoothness ≤ 0.026	\Rightarrow	Mean fractal dimension ≥ 0.093 & Standard error compactness ≤ 0.092 & Worst smoothness ≥ 0.108 & Worst compactness ≥ 0.246 & Worst fractal dimension ≥ 0.068 & & 36 & 107
Standard error symmetry ≤ 0.056 & Worst perimeter ≤ 214.958	\Rightarrow	Mean area ≥ 547.904 & Standard error texture ≤ 3.486 & Worst smoothness ≥ 0.152 & Worst fractal dimension ≥ 0.069 & & 12 & 33
Time ≤ 112.585 & Mean perimeter ≤ 181.616 & Standard error texture ≤ 3.275 & Standard error smoothness ≤ 0.025 & Mean fractal dimension ≤ 0.082	\Rightarrow	Standard error compactness ≤ 0.094 & Worst compactness ≥ 0.264 & Worst fractal dimension ≥ 0.080 & 25 & 80

When these rules were observed we found that in 143 instances the first rule is satisfied and out of these 143 instances, 107 (74.83%) are of non-recurring type. Similarly second rule is satisfied by 45 instances out of which 33 (73.33%) instances are non-recurring. And out of 105 instances that satisfy the third rule, 80 (76.19%) instances are of non-recurring type.

From these results it can be observed that the algorithm presented in this section is capable of extracting some interesting and understandable rules from the databases. Since the algorithm works on memory resident data, it executes quickly. The databases is read from the disk two times only. So the disk access time is less.

3.6 Discussion

Two algorithms are presented in this chapter, based on multi objective genetic algorithms which have been found capable to extract meaningful rules from large datasets. Both the algorithms were designed considering the association rule mining problem as a multi objective problem. Expressions to quantify comprehensibility and interestingness are also presented here. These expressions were used to measure the objectives of the association rule mining problem. From the experimental results it can be observed that the algorithms were capable of discovering some interesting, understandable and valid rules.

While discovering the rules from those databases it was also observed that some of the attributes were never or very rarely used in the derived rules. If those attributes can be eliminated before the rule mining process, the cost of rule extraction process could be saved significantly. This necessitates an appropriate dimensionality reduction technique and the next chapter introduces this issue and also describes some effective dimensionality reduction techniques.

Though it was assumed that the datasets used for rule extraction are static in nature, it may not be always true. Time to time some new records

are added to the datasets. Due to which some new rules may become valid. Since the rule extraction is a time consuming job, it is not appreciated to extract the rules over an incremental database by repeated scanning of the whole database every time it is updated. Extraction of meaningful rules over incremental database is considered to be another challenging job. In a later chapter this issue of rule mining is addressed.

Chapter 4

Dimensionality of Databases: Another Challenge

In the previous chapter, while extracting rules from the databases, it was observed that some of the attributes were never used in any of the rules generated. In other words, presence or absence of those attributes in the database have no effect on the result of the rule mining task. The possible reason for it is that, these attributes are irrelevant. Almost every dataset contains some irrelevant attributes. Presence of these irrelevant attributes increase the storage requirement of the database, but practically carries no useful information. Thus, reduction in the dimensionality of these databases by discarding those redundant or irrelevant attributes will help saving the cost of computation to a great extent.

4.1 Dimensionality Reduction

Selecting the relevant attributes or discarding the irrelevant attributes from a database is a challenging job. Based on our study and experimental analysis it has been found that an irrelevant attribute does not affect the target concept in any way, and a redundant feature does add anything new to the target concept [KP94]. In many applications, the size of the dataset is so huge that learning might not work well before removing the unwanted attributes. Reducing the number of irrelevant or redundant attributes drastically reduces the execution time of a learning algorithm [KS95, KS96].

In reality, relevant attributes are unknown apriori. Therefore, set of candidate attributes are introduced to represent the domain in a better way. But, for a database with n attributes, there will be $2^n - 1$ possible candidate sets available. Hence, selecting the best among them is a time consuming job. This process of eliminating the irrelevant attributes or selecting the relevant features is commonly known as *dimensionality reduction* or *feature selection*.

Dimensionality reduction attempts to remove irrelevant features according to two basic criteria: (i) the accuracy does not significantly decrease and (ii) the resulting concept, given only the values for the selected attributes, is as close as possible to the original concept, given all the attributes.

4.2 Existing techniques

A good number of algorithms were proposed for dimensionality reduction/feature selection over the years [Doa92, FC07, SDZ07, KCN08, GE08, HCX08, RFJT08].

Some of the prominent feature selection algorithms commonly used for reducing the dimensionality of the databases are reproduced for the sake of understanding and for comparison in this section. The notations/symbols used in describing those algorithms are reported in Table 4.1.

4.2.1 Focus

Using consistency measure to evaluate the subsets, Focus [AD92] generates all possible feature subsets. It implements the Min-Features bias that prefers consistent hypothesis definable over as few features as possible. The algorithm is given below.

Focus(D, S) /* D and S are the database and the set of features respectively.*/

1. $T = S$
2. For $i=0$ to N /*N is number of features */
3. For each subset L of size i
4. If no inconsistency in the training set D then
5. $T = L$
6. Return T

Figure 4.1: Focus

The algorithm works well with noise-free data. Presence of noisy data within the dataset affects the performance of the algorithm.

4.2.2 LVF

Using consistency measure to evaluate the subsets, LVF [LS96] generates the candidate subsets randomly. It randomly searches the subset space and calculates an inconsistency count for the subset. An inconsistency threshold is assumed and any subset with inconsistency measure greater than that value is rejected. The algorithm is reproduced in Figure 4.2.

LVF($D, S, MaxTries, \alpha$)

1. $T = S$
2. For $i=1$ to $MaxTries$
3. Randomly choose a subset of features, S_j
4. if $card(S_j) \leq card(T)$
5. if $inConCal(S_j, D) \leq \alpha$
6. $T = S_j$
7. Output S_j
8. else
9. append S_j to T
10. output S_j as 'another solution'
11. endfor
12. return T

Figure 4.2: LVF

This algorithm works well for datasets with smaller number of attributes. Since all feature subsets are not considered best subset may not be found, specially when the number of attributes is high. If $MaxTries$ has been given a larger value many possible feature subsets will be produced as output, selection of the required subset becomes another problem for the user.

4.2.3 Branch and Bound

This algorithm was proposed by Narendra and Fukonaga in 1977 [NF77]. The important requirement of the algorithm is that the evaluation function be monotonic. The algorithm needs input of required number of features (M) and it attempts to find out the best subset. The algorithm is given in Figure 4.3.

```
B&B( $D, S, M$ )
1. if card( $S$ )  $\neq M$  then
   /*subset generation*/
2.  $j=0$ 
3. for all features  $f \in S$  begin
4.  $S_j = S - f$  /*remove one feature at a time */
5. if ( $S_j$  is legitimate) then
6. if isbetter( $S_j, T$ ) then
7.  $T = S_j$ 
   /*recursion*/
8. B&B( $S_j, M$ )
9. endfor
10.  $j++$ 
11. endif
12. return  $T$ 
```

Figure 4.3: Branch & Bound

The algorithm always produces the best feature subset as the output. And useful for datasets with large number of attributes also. But the major difficulty of the algorithm is its exponential time complexity.

4.2.4 Relief

This algorithm [KR92] selects the relevant features by using statistical methods. It is basically a feature weight based algorithm designed on *instance based learning algorithm* [DL97]. It first chooses a sample of instances (where the number of instances i.e. *NoSample* is a user input) at random from the set of training instances and for each instance in it, finds the *NearHit* and *NearMiss* instances based on Euclidian distance measure. *NearHit* of an instance is defined as the instance having minimum Euclidean distance among all instances of the same class as that of the instance. *NearMiss* of an instance is defined as the instance having minimum Euclidean distance among all instances of different class. The algorithm finds the weights of the features from a sample of instances and chooses the features with weight greater than a threshold. The algorithm is given in Figure 4.4.

Relief(*D*, *S*, *NoSample*, *Threshold*)

1. $T = \phi$
2. Initialize all weights, W_j to zero
3. For $i = 1$ to *NoSample*
4. Randomly choose an instance $x \in D$
5. Find its *nearHit* and *nearMiss*
6. For $j = 1$ to N /* N is the number of features */
7. $W_j = W_j - \text{diff}(x_j, \text{nearHit}_j)^2 + \text{diff}(x_j, \text{nearMiss}_j)^2$
8. For $j = 1$ to N
9. If $W_j \geq \text{Threshold}$
10. Append feature f_j to T
11. Return T

Figure 4.4: Relief

Relief works for noisy and correlated features. This algorithm is efficient as only the subset having the number of features smaller than that of the current best subset are checked for inconsistency. Also it is easy to implement and is guaranteed to find the optimal subset.

However, it cannot work with redundant features and hence generates non-optimal features if the database contains redundant features. It works only with binary classes. Another problem is how to choose of the proper value of *NoSample*.

4.2.5 DTM

Decision Tree Method [Car93] uses feature selection in an application on Natural Language Processing. To select the features, it runs C4.5 [Qui93] over a training set and all those features that appear in the pruned decision tree are selected. In other words, the union of the subsets of the features, appearing in the path to any leaf node in the pruned tree is the selected subset.

4.2.6 FFC

Based on the coherence properties of an attribute to the target concept, FFC [DB04] tries to select the relevant itemsets. For selecting them it uses the *coherence frequency count* and *non coherence frequency count* of the attributes. The steps of the algorithm are given in Figure 4.5.

The algorithm works well for binary class datasets.

FFC(D, γ, β, n)

1. F =all the features
2. do while($| F | > n$)
3. $S = \phi$, $L_1 = \{f \mid \text{support}(f) \geq \gamma\}$, $L'_1 = \{f \mid \text{support}(f') \geq \gamma\}$
4. $S = S \cup \{xC \mid x \in L_1 \cup L'_1\}$ // where C is class label//
5. for all instances $i \in D$ do begin
6. $S_i = \text{subset}(S, i)$
7. for all $s \in S_i$ do
8. $s.\text{count}++$
9. $F1 = F$
10. $F = \{f \mid s = fC, s \in S_i \text{ and } s.\text{count} \geq \gamma\}$
11. $\gamma = \gamma + \beta$
12. if $| F | = n$ then return F
13. else return $F1$

Figure 4.5: FFC

4.2.7 MDLM

Minimum Description Length Method [SDN90] tries to eliminate all irrelevant and redundant features. This method is based on the concept that if the features in a subset X can be expressed as a fixed non-class-dependent function F of the features in another subset Y , then once the values in the features in the subset X are known, the features in the subset Y are useless. Minimum Description Length Criterion (MDLC) is used for this purpose.

The algorithm exhaustively searches for all possible subsets and returns the subset satisfying MDLC. Hence, the algorithm takes significant amount

Table 4.1: Symbols used in Above Algorithms

D	=	The Database
S	=	Original set of Features
M	=	Number of features to be selected
$Card(X)$	=	Function to find the cardinality of the set X
$isbetter(X;Y)$	=	A function to check if the set X is better than the set Y
$NoSample$	=	the sample size
$ThresHold$	=	lower limit of a feature's weight to become relevant
N	=	number of features
g, γ	=	Minimum support
W_j	=	weight of j-th feature
$Maxtries$	=	number of iterations
$InConCal$	=	function to calculate inconsistency
l	=	upper level of inconsistency
$diff()$	=	to find difference of same feature in two different records
L_1	=	Features frequent occurrence
L'_1	=	Features whose non occurrence is frequent
β	=	Increment to min-support
$F, F1$	=	Set of selected attributes

of time. Moreover, the method can find all useful features for Gaussian cases only.

4.3 Dimensionality Reduction: New Approaches

The algorithms presented in the previous section are commonly used in data mining for dimensionality reduction. All of them have their own advantages

and limitations. But there is no universally acceptable algorithm for dimensionality reduction, that works on all kinds of databases. Most of the classical association rule mining algorithms work on *Market Basket* databases. The actual transaction databases is first converted to the market basket form. There are different techniques for discretization of real valued attributes. Most commonly used technique is the use of sub-ranges. Using the sub-ranges of all the attributes, the market basket database is created. From this database the classical rule mining algorithms extract the rules. But this database also contains the irrelevant attributes. To save the computational cost during rule extraction, irrelevant attributes of the database can be discarded by using an appropriate dimensionality reduction technique.

After a careful study of this situation, it was observed that it is more beneficial to reduce the dimensionality before the data is encoded in market basket form. Following two subsections are dedicated in describing three different techniques to handle this problem; two of them are based on frequency count and the other is rule based.

4.3.1 Frequency Count Based Reduction

In classical rule mining algorithms, generation of the rules are controlled by two user given parameters, namely *minimum support* and *minimum confidence*. The attributes whose support is less than the minimum support are not relevant, and are not used in any later stage of rule mining. Hence, if those attributes are eliminated during the conversion to market basket database, then the cost of rule mining process reduces significantly. To meet this requirement, the algorithm reported next was designed.

The algorithm DRUFT (**D**imensionality **R**eduction **U**sing **F**requency **C**ount) is meant for reducing the dimensionality of market basket dataset based on frequency count. When the dataset is converted to market basket, all the sub-ranges of all attributes have to be considered. But some sub-ranges of the attributes may be found irrelevant. If these sub-ranges are discarded then the dimensionality will be reduced. Unlike the above mentioned algorithms, DRUFT is capable of finding the relevant sub-ranges of the attributes, resulting in a market basket dataset with a few number of attributes in it.

The algorithm takes the dataset D and maximum number of needed sub-range $MaxAtt$, as input. Table 4.2 describes the symbols used in DRUFT. It reads the dataset only once and finds the frequency count of every sub-range of all attributes. Using these frequency counts it eliminates irrelevant sub-ranges, till the desired number of attributes remain not-eliminated. Finally it produces the not-eliminated sub-ranges as output. Number of such sub-ranges is equal to or less than $MaxAtt$. The algorithm is given Figure 4.6.

Algorithm DRUFT

Input: The dataset D , maximum numbers of sub-ranges needed $MaxAtt$.

Output: Set of selected sub-ranges, $S1$

1. $S = \phi$
2. for all attributes $A_i \in A$
3. for all subranges $P_{i,j} \in A_i$
4. $S = S \cup P_{i,j}$
5. for all $s \in S$
6. find the frequency count, SUP_s
7. $minfreq=1$
8. $S1 = \phi$

9. for all $s \in S$
10. if $(s \in A_i)$ and $(SUP_s * |A_i|) \geq (minfreq * \max(|A|))$
11. $S1 = S1 \cup s$
12. if $|S1| \leq MaxAtt$ go to Step 16
13. $minfreq = minfreq + 1$
14. $S = S1$
15. goto Step 8
16. return $S1$

Figure 4.6: Dimensionality Reduction Using Frequency Count

Table 4.2: Symbols used in DRUFT

A	=	Attributes of original dataset
$ A_i $	=	Number or sub-ranges of i th Attribute
$P_{i,j}$	=	j^{th} sub-range of i^{th} attribute
$\max(A)$	=	Maximum of $ A_i $
S	=	Set of sub-ranges of A
$S1$	=	Set of selected sub-ranges
SUP_s	=	Frequency of sub-ranges
$minfreq$	=	Current value of support count to declare as frequent
$MaxAtt$	=	Maximum no of sub-ranges to be selected

The algorithm works on the original continuous valued database where the number of attributes are generally small, hence requiring less amount of memory for its execution. For every attribute some sub-ranges are considered. These sub-ranges become attributes in the market basket dataset. But the above method will restrict some of these ranges from becoming an

attribute of the market basket dataset. For every sub-range of all the attributes, the frequency of them within the dataset is calculated by reading the dataset once. Afterwards, only those frequency counts are used to reduce the dimensionality of the market basket dataset to the user desired level. The user has to provide his desired number of attributes as input to the algorithm. After calculating the frequency count, those sub-ranges are eliminated; whose frequency count is less than a factor of minimum frequency, *minfreq*. This factor is different for the sub-ranges of different attributes. If the dataset has been reduced to the desired level, it produces the sub-ranges that are found out to be relevant. Otherwise, it eliminates some more sub-ranges by incrementing the minimum frequency count, *minfreq*. This process continues till the number of relevant sub-ranges do not become less than or equal to the user desired number of attributes.

For example, let a dataset contain three attributes X, Y and Z . Values of the attributes lie in the ranges $\{0,10\}$, $\{10,20\}$ and $\{-10,10\}$ respectively. When the dataset is converted to market basket with 10 sub-ranges of all attributes then the market basket dataset will contain 30 attributes in it. But the values of the attributes may not be evenly distributed. For that reason some of the sub-ranges will become irrelevant because of their low frequency. Say, the dataset contains 100 instances and the values of the first attribute are distributed over the sub-ranges like 4,11,17,27,20,10, 6, 3, 0, 2. Similarly, second and third attributes are distributed as 1,1,4,5,11,20,26,23,6,3 and 2,1,0,2,10,18,33,16,15,3 respectively. If the rules are extracted with a minimum support of 15% then only 10 sub-ranges will be used out of 30. The other sub-ranges will not contribute anything to the rule mining process but will simply occupy the memory and increase the data transfer time from the disk.

If the above mentioned algorithm is applied on the original dataset it will select three sub-ranges of the first and the second attribute and four sub-ranges of the third attribute. If the market basket dataset is constructed only for these sub-ranges then the dataset size will be reduced to one third of the original one. Hence the rule mining algorithm will need less memory and less data transfer from the disk, in turn will speed up the execution of the rule extraction process.

Implementation and Results

The algorithm was implemented in an environment described in the previous chapter. And was tested over various synthetic and standard databases. Here some of the results from *Monks-1* and *Monks-3* training databases downloaded from UCI machine learning repository are given. There are 124 and 122 instances in *Monks-1* and *Monks-3* respectively. Both of them have 8 attributes; first one is the class number and the last one is the sample number. Remaining six attributes are numeric values spanning over different ranges. The minimum and maximum values of these attributes are $A1(1,3)$, $A2(1,3)$, $A3(1,2)$, $A4(1,3)$, $A5(1,4)$ and $A6(1,2)$. If these databases are converted to market basket then there will be a total 17 attributes.

The above mentioned algorithm can reduce the dimensionality of the databases to the required level. Table 4.3 gives details of the reduction. From the results in Table 4.4 it can be observed that it selects the sub-ranges of the attributes those were declared as relevant by the existing algorithms also. Only part 2 of attribute 6, denoted in Table 4.3 as A6-2, is coming in addition. Reason for not selecting A6 by other algorithms is that it is a redundant attribute.

Table 4.3: Dimensionality Reduction on Monks-1 and Monks-3 by DRUFT

Desired no attributes	Monks-3		Monks-1	
	Reduced to	Minimum support	Reduced to	Minimum support
10	8	31	10	31
9	8	31	6	32
8	8	31	6	32
7	4	32	6	32
6	4	32	6	32
5	4	32	4	33

From the results in Table 4.3 it can be observed that the algorithm reduces the dimension of the database always to the required level. Since the selected sub-ranges only have a higher frequency over the database, only those will be finally used by the classical rule mining algorithms. From Table 4.4 it can be observed that the algorithm is selecting only sub-ranges of those attributes that were declared as relevant by other techniques.

From these results it can be observed that the algorithm presented above can be of better use to reduce the dimensionality of database specially for those data mining tasks like association rule mining where *frequency* of the attributes has a great role to play.

However, a limitation of the algorithm is that if the user does not have

Table 4.4: Comparative Results of Some Existing Algorithms and DRUFT

Method	Monks-3		Monks-1	
	Selected attributes	MBs dimension	Selected attributes	MBs dimension
Relief	A2,A5 always & one or both of A3,A4	9 or 10 or 12	A1,A2,A5	10
B&B	A1,A3,A4	8	NA	-
DTM	A2,A5	7	NA	-
LVF	A2,A4,A5	10	NA	-
MDLM	A2,A3,A5	9	NA	-
FFC	A1,A2,A4,A5	13	A1,A2,A5	10
DRUFT reduced to 4	A1-1,A3-1,A4-3, A5-1	4	A1-1,A2-3,A5-4, A6-2	4
DRUFT reduced to 6	-	-	A1-1, A2-3, A3-1, A4-3, A5-4, A6-2	6
DRUFT reduced to 8	A1-1, A2-2, A3-1, A4-3, A5-1, A5-2, A5-4, A6-2	8	-	-

sufficient knowledge about the dataset, some rules may be lost during the rule extraction process. If the dataset contains, say, n single item frequent itemsets, and say, the dataset is already reduced to m , where $m < n$, then some rules will be lost during the extraction process due to the reduction.

To overcome this difficulty, another algorithm was developed to help the classical association rule mining technique.

This algorithm **Dimensionality Reduction for Association Rule Mining (DRARM)** is also meant for reducing the dimensionality of market basket dataset based on Frequency count. Instead of using the desired number of attributes as an input parameter, it uses *minimum support* as the user parameter.

The basic idea behind this algorithm is the downward closer property of frequent itemsets. Only the frequent itemsets are used for constructing next level candidate itemsets. So, the infrequent single itemsets are never used in any stage of the rule extraction process. The algorithm *DRARM* hence eliminates those sub-ranges of the attributes that will result in an infrequent item in the target market basket dataset. Since these infrequent items are eliminated before the rule extraction process, they will not occupy the memory unnecessarily.

The steps of the algorithm are enumerated in Figure 4.7.

Algorithm DRARM

Input: The dataset D , minimum support γ .

Output: Set of selected sub-ranges, $S1$

1. $S = \phi$
2. for all attributes $A_i \in A$
3. for all subranges $P_{i,j} \in A_i$
4. $S = S \cup P_{i,j}$
5. for all $s \in S$
6. find the frequency count, SUP_s
7. $T = |D| / *$ total number of records in D $*/$
8. $S1 = \phi$

9. for all $s \in S$
10. if ($SUP_s \geq T * Minsup$)
11. $S1 = S1 \cup s$
12. return $S1$

Figure 4.7: Dimensionality Reduction for Association Rule Mining

The algorithm was tested over *Monks-1* and *Monks-3* datasets mentioned above, as well as some synthetic datasets. Table 4.5 depicts the summary of the reduction by **DRARM** over some synthetic datasets. These datasets were generated randomly, and different in number of records, number of attributes and attribute value ranges. First dataset contains 20000 records and 25 attributes, the second dataset contains 10000 records and 20 attributes and the third one contains 20000 records and 10 attributes. When the first dataset, named as *T20_C25*, was converted to market basket with 4 equal sub-ranges of all attributes then the resultant market basket dataset contained 100 attributes in it. Similarly, dataset *T10_C20* and *T20_C10* resulted in two other market basket datasets with 5 equal sub-ranges of the attributes.

Table 4.5: Reduction in Synthetic Datasets

Dataset	Sub-ranges	MB's dimension	Support	Reduced to
T20_C25	4	100	20%	69
			40%	22
T10_C20	5	100	20%	69
			40%	21
T20_C10	5	50	20%	35
			40%	15

Using the apriori [AIS93] algorithm frequent itemsets were derived from market basket datasets resulted from the above mentioned datasets. For the same original dataset, different market basket datasets were considered that were resulted as the result of reduction with different support. When these results were analyzed, following observations were made.

- With the same support, same itemsets were derived from the original as well as reduced datasets, provided reduction was done with the same or smaller support.
- With the same support, original datasets took significantly more time than that of the reduced datasets. Some results are reported in Table 4.6.
- For higher support, during frequent itemset finding than the reduction, no information is lost due to reduction, and it executes faster.
- For smaller support, during frequent itemset finding than the reduction, some information are lost. Hence care should be taken while providing the minimum support during reduction.

The algorithm was implemented on a computer with Intel Core2Duo 2.5 GHz processor, 3 GB RAM. The timing information presented in Table 4.6 are average of 15 runs of the program on each dataset.

From the Table 4.6, it can be clearly observed that there is a significant reduction of time with **DRARM** while compared with the original dataset. Though it was tested only for the *Apriori* algorithm it is valid for other algorithms also, only amount of time saved may differ for those algorithms that needs less number of scanning of the dataset during the frequent itemset generation. However, to avoid information loss during reduction, care should be taken while providing the user parameter *minimum support*.

Table 4.6: Time Taken in Deriving Frequent Itemset

Dataset	MB's	Size of Dataset	Minimum Support	Time Taken
T20_C25	Dimension			
Original	100	4,020,000 Bytes	20%	57282 ms
			40%	46578 ms
Reduced with 20% support	69	2,780,000 Bytes	20%	37469 ms
			40%	18641 ms
Reduced with 40% support	22	900,000 Bytes	20%	10516 ms
			40%	5313 ms

Next section shall discuss another dimensionality reduction technique which seems to support the Multi-objective Association Rule Mining as well as classical approach.

4.3.2 Rule Based Reduction

In the previous section two dimensionality reduction techniques useful for the rule mining using classical approach were presented, which reduce the dimension of databases to a required level. If only the higher frequency attributes are selected for the final rule mining stage, then some of the interesting rules may be lost. It is obvious that an interesting rule has a lower frequency, so, above two techniques for the dimensionality reduction are not suitable for multi-objective association rule mining. The need of dimensionality reduction suitable for multi-objective association rule mining has motivated us towards the development of this rule based algorithm presented next.

attribute value of the first, second and third attribute respectively. Output of the algorithm are the relevant ranges of the attributes. These ranges can be used to construct a market basket dataset with reduced dimensionality, if classical rule mining approach is used. Otherwise these ranges of the different attributes can be used by the multi objective rule mining algorithm that works on continuous valued attributes.

The candidate range of each attribute is encoded within a chromosome of the genetic algorithm. The lower and upper limit of the value of each attribute is represented separately in a binary form with m number of bits. So the complete chromosome will be the concatenation of n such pair of bit strings, where n is the number of attributes in the database. While decoding the values floating point decoding method described in the Chapter 3. The genetic algorithm will execute for a given number of generations, *MaxGen*. Output of the algorithm will be some combinations of different relevant ranges of attribute values. The steps of the algorithm is next in Figure 4.8.

Algorithm DRMOGA

Input: Continuous valued dataset D , maximum number of generations *MaxGen*

Output: Combinations of useful attribute value sub-ranges

1. Generate population P of N chromosomes randomly.
2. $i = 0$.
3. Do while($i < MaxGen$)
4. Decode(p_j), $\forall j, p_j \in P$.
5. $\forall j, p_j \in P$ find SUP(p_j) and compactness(p_j).
6. Rank(P_j), $\forall j, p_j \in P$.

7. Based on $\text{Rank}(p_j)$ select chromosomes for next generation.
8. Perform multi-point crossover, then mutation to get new population O .
9. Replace population P by O .
10. $i = i + 1$.
11. End do.
12. return P .

Figure 4.8: Dimensionality Reduction with Multi objective GA

This algorithm uses the concept of frequency count and compactness of continuous range of values of the attributes. From the experiments it has been found that the proposed algorithm reduces the dimension of the target market basket dataset considerably, if the dimension of the market basket dataset without reduction is very high. The reduction is less if original market basket dataset's dimension is already very small. It will happen, if the dataset's attribute value range is very small and/or a very few sub-ranges have to be considered while converting to market basket dataset. For example, for an attribute ranging over 1 to 20 and the relevant values within 7 and 15, in both the market basket dataset before reduction and after reduction, there will be 2 features corresponding to this attribute if two sub-ranges are considered while converting. Since the algorithm is producing a number of such combinations, before using these ranges for converting the dataset to market basket such ranges can be combined to get a better result. Though it imposes a little burden on the decision maker, it will provide some control in constructing the final market basket database.

Implementation and Results

The algorithm was implemented in the same environment as the other one. Crossover and mutation probability was taken as 0.9 and 0.002 respectively, Population size was taken as 40 and 10 bits were used to encode every value within the chromosome. The algorithm was tested over several synthetic databases as well as standard databases like Monks-1 and Monks-3. Since in Monks databases ranges of attribute values are small, results from those databases are not included here. Results from a synthetic database are reported here. Table 4.7 contains a description of the synthetic database used.

Table 4.7: Description of the Synthetic Database

Attribute name	Minimum value	Maximum value
Attribute1	1	100
Attribute2	51	150
Attribute3	0	1
Attribute4	1	1000

To conduct the experiments, 1000 records was generated for the database. While generating these records it was ensured that 90 % instances satisfy the conditions ($1 \leq Att1 \leq 25$), ($90 \leq Att2 \leq 145$), ($0.1 \leq Att3 \leq 0.3$) and ($400 \leq Att4 \leq 900$) other instances are random so they may or may not be present within that range. The database finally had 902 records that satisfied the above conditions.

Then the algorithm under discussion was used on this database to find the relevant sub-ranges. ($1 \leq Att1 \leq 20$), ($101 \leq Att2 \leq 150$), ($0.1 \leq Att3 \leq 0.4$) and ($401 \leq Att4 \leq 800$) were some of those sub-ranges found by the algorithm and was found to be very near to the range used during

construction of the database. Total 907 records were there within this range. After analyzing the database, it was found that 9 records that satisfied the original conditions, were not satisfying the new condition but 14 more records satisfy the new conditions which did not satisfy the original conditions. The exclusion was due to the following reasons - 5 records were eliminated since the Attribute1's values were more than 20 but less than 25, 1 record was eliminated since the Attribute2's value was less than 101 but more than 90, 2 more such records were there but they were already eliminated by the first attribute. 3 records were eliminated since the Attribute4's values more than 800 but less than 900, 4 more such records were there but 3 of them were already eliminated by the first attribute and 1 by the second attribute. The inclusion was due to the following reasons - 6 records were included since the Attribute2's values were less than 150 but more than 145. Other 8 records were included since the Attribute3's values were less than 0.4 but more than 0.3, 3 such records were also there but they were already included by Attribute2.

From this analysis it can be observed that the algorithm is able to find out the relevant ranges of values of the attributes. If the above mentioned database is to be converted to market basket using 10 equal sub-ranges of all the four attributes, then the converted dataset will have 40 attributes. But, after discarding the irrelevant ranges, there will be only 14 attributes in it.

Similarly, if 100 equal sub-ranges of all the four attributes have to be considered while encoding, then the converted dataset will have 400 attributes, but after discarding the irrelevant sub-ranges it will restrict to only 150 attributes.

With 10 equal sub-ranges of every attribute, when the previous algorithm was used on this database and tried to reduce the dimensionality to 14, it got reduced to 13 attribute. And all these were lying within the ranges selected by the current algorithm.

The previous algorithm was applied on the database with 100 equal sub-ranges of each attribute, when it was reduced with a support of 10% , it gets reduced to 151. When this reduction was analyzed, an interesting observation was made. Though this algorithm gives one more sub-range, other 150 sub-ranges are not exactly the same. When the sub-ranges were analyzed it was found that 146 were common. Four sub-ranges that were given by the current algorithm are not produced by the previous one, but five new sub-ranges were declared to be relevant. This is because of the reason that the values of the attributes are not evenly distributed in the whole range, as the majority of the sub-ranges were declared to be relevant by both the algorithms. Similar observation was made on the results from other datasets considered during this dissertation.

The difficulty of the previous algorithm is that the decision maker has less control during the reduction process. Sometimes he may be interested in some sub-ranges of attribute value that are not too frequent. But the current algorithm will give him some freedom to select the sub-ranges of his interest, as multiple sub ranges will be produced by the algorithm, as the sub-range considered below.

Another sub-range given by the algorithm was $(7 \leq Att1 \leq 16)$, $(114 \leq Att2 \leq 132)$, $(0.2 \leq Att3 \leq 0.35)$ and $(567 \leq Att4 \leq 714)$ with the frequency count 638. Though the frequency count for this subset range is smaller than

the earlier one, this subset range is more compact. It indicates that the dimensionality of the target dataset has a further scope of reduction. For example, if this range is used then the dimension of the converted dataset will be 11 and 58 only, when 10 and 100 sub-ranges are considered respectively during conversion, making the dataset more compact.

4.4 Discussion

In this chapter influence of dimensionality of databases on the data mining tasks was discussed, and a few existing algorithms for dimensionality reduction were presented. But none of them is universally acceptable. So, to help the association rule mining, three algorithms for reducing the dimensionality of databases were presented in this chapter. From the experiments it has been found that though the first one can reduce the dimensionality of dataset to the desired level, user should have some knowledge about the dataset under consideration. But the other two algorithms are capable of reducing the dimension of the databases and suitable for association rule mining.

Chapter 5

MORM in Incremental Databases

Databases used for extraction of association rules, are assumed to be static in nature, however, it may not be always true. Time to time these databases are updated in terms of deletion of existing records, insertion of new records or modification of existing records. Though all these types of updations are allowed in a transaction database, however, all of them may not occur in a *data warehouse*. Modification in the existing records or deletion of an existing record is normally not recommended here. However, new records are added to it time to time. In other words, the databases used by data mining tasks, may be *incremental* in nature. Extraction of association rules in a cost effective manner from such incremental databases is considered to be a challenging problem.

5.1 Need of Incremental Mining

Due to the insertion of new records into a dataset, some new rules may come into existence. And at the same time some of the existing rules may become invalid. Insertion of these new records will change the support of the rules although the support count for some rules may not be changed.

For example, say X and Y are two distinct itemsets of a dataset having 1000 records with the support count 210 and 190 respectively. Subject to the minimum support of 20%, X is a frequent itemset but Y is not. Now, another 100 records, where 5 and 40 records contain X and Y respectively, are added to the dataset. In the updated dataset, the support of the itemset X falls below the threshold but the support of Y goes above the threshold. In other words X becomes infrequent but Y becomes frequent in the updated dataset. Due to these changes of frequent itemsets, some earlier derived rules will be dropped and some new rules may be found to be relevant. Since the rule extraction is a time-consuming job, it is not appreciated to extract the rules over an incremental database by repeated scanning of the whole database every time it is updated. Hence, the extraction of meaningful rules over an incremental database is considered to be another challenging job.

To meet the challenges of incremental mining, several works based on the classical approach of association rule mining were carried out over the decade. Most of these techniques try to extract the new rules, (if there any), by repeated scans over the newly inserted records, however, with a minimum scan over the old database. The size of the *incremental part* is normally very small as compared to that of the old part. Hence, more number of scanning of the old part leads to wastage of time. For clarity, some of the existing relevant techniques are reproduced in the next section.

5.2 Existing Techniques

Most of the existing techniques for incremental mining work in two phases, i.e. frequent itemset generation and rule generation. These algorithms give more importance to the frequent itemset generation phase. While doing this they use the information that were extracted from the old database so that less number of scanning of the old database is required. To explain the working of most of these algorithms, the sample database given in Table 5.1 is used.

Table 5.1: A sample database

T_ID	A	B	C	D	E	F
100	1	1	1	1	0	1
200	1	1	1	1	1	0
300	0	1	1	1	0	0
400	1	0	0	1	1	0
500	1	0	1	1	0	0
600	1	1	1	0	1	0

The frequent itemsets based on the market basket data of Table 5.1 were then derived using Apriori [AIS93] algorithm with a minimum support of 50%. Table 5.2 reports these frequent itemsets. The algorithms discussed in this section use these frequent itemsets while deriving the frequent itemsets over the incremented data.

Now, to describe the incremental association mining problem, with reference to the database reported in Table 5.1 is updated in the following manner: transaction with ID 400 is deleted and a new transaction with ID

Table 5.2: Frequent itemsets from Table 5.1 with minimum support 50%

Size	Number	Itemsets
1	5	A,B,C,D,E
2	7	AB, AC, AD, AE, BC, BD, CD
3	3	ABC, ACD, BCD

700 is added. The resultant database is reported in Table 5.3.

Table 5.3: Updated sample dataset

T.ID	A	B	C	D	E	F
100	1	1	1	1	0	1
200	1	1	1	1	1	0
300	0	1	1	1	0	0
400	-	-	-	-	-	-
500	1	0	1	1	0	0
600	1	1	1	0	1	0
700	0	1	0	1	1	0

The frequent itemsets derived from this updated database using Apriori algorithm with a minimum support 50% are reported in Table 5.4.

It can be observed that the frequent sets listed in Table 5.4 are not exactly the same with those reported in Table 5.2. In Table 5.4, there is a new frequent itemset of size 2, **BE**, with support count 3, and the itemset **AE** that was frequent earlier, listed in Table 5.2, has become infrequent. Apriori and

Table 5.4: Frequent itemsets from Table 5.3 with minimum support of 50%

Size	Number	Itemsets
1	5	A,B,C,D,E
2	7	AB, AC, AD, BC, BD, BE, CD
3	3	ABC, ACD, BCD

its other counter parts are able to extract these frequent itemsets, but for that those algorithms will simply consider the updated dataset as a new dataset and everything is started from the scratch ignoring the earlier computed result. Since extraction of the frequent itemsets is a time consuming task, the algorithm should be capable to exploit the pre-computed results during generation of the frequent itemsets for the updated databases. It will save the computation time to a great extent. The following algorithms attempt to address this issue.

5.2.1 FUP

The algorithm FUP [CHNW96] first scans the incremental part of the dataset and detects (i) the looser single itemsets, i.e. the itemsets that become infrequent due to the inclusion of the incremented part and (ii) it finds the candidate frequent itemsets. Then the whole database (i.e. the old and new together) is scanned to find their support in the complete database. Next, it performs similar operations iteratively for k -itemsets. This algorithm gets some benefit during the candidate frequent itemset generation, that saves some time than the apriori algorithm. But the algorithm needs multiple number of scanning of the whole database. It needs k number of scanning of the whole database if the largest maximal frequent set's size is k .

5.2.2 FUP₂

FUP₂ [CLK97] works on a dynamic dataset where new records may be inserted and some of the existing records may be deleted. It extracts the rules from the final dataset by considering both the deleted parts and the newly added part. Applying this algorithm on the updated database D' given in Table 5.3, with a minimum support of 50% will proceed as follows. In the first step, candidate set C'_1 is formed that will be the same as C_1 in the old database. That is, $C'_1=C_1=\{A,B,C,D,E,F\}$. Then comparing C'_1 and L_1 (as computed for old dataset D) will lead to breaking of C'_1 into two parts: first part, $P_1=\{A,B,C,D,E\}$, that is common to both C'_1 and L_1 and the second part $Q_1=\{F\}$ that is the difference of C'_1 and P_1 . Then the support of each itemset in P_1 and Q_1 are computed. After that frequent itemsets in P_1 that are still frequent in the new dataset are obtained and included in L'_1 . Then it is checked whether any itemset of Q_1 , that were earlier infrequent has become frequent or not. If yes, add them to L'_1 . Finally, $L'_1=\{A,B,C,D,E\}$ is obtained. Then $C'_2=\{AB, AC, AD, AE, BC, BD, BE, CD, CE, DE\}$ is generated by *Apriori-gen* function using L'_1 . The generated candidate set is again broken to $P_2=\{AB, AC, AD, AE, BC, BD, CD\}$ and $Q_2=\{BE, CE, DE\}$. After computing the support of the itemsets in P_2 and Q_2 , it computes $L'_2=\{AB, AC, AD, BC, BE, CD\}$. Similarly, based on L'_2 , $C'_3=\{ABC, ABD, ACD, BCD, BCE, BDE\}$ is computed and broken to $P_3=\{ABC, ACD, BCD\}$ and $Q_3=\{ABD, BCD, BCE, BDE\}$. Then $L'_3=\{ABC, ACD, BCD\}$ is computed as earlier. Since C'_4 has been found empty, the algorithm stops there. The algorithm is reproduced in Figure 5.1.

1. Obtain a candidate set C_k of itemsets. Halt if $C_k = \phi$.
2. Calculate b_X^+ for each $X \in C_k$.
3. Partition C_k into P_k and Q_k .

4. For each $X \in P_k$, remove it if $\sigma_X + b_X^+ < |D'| \times s\%$.
5. For each $X \in Q_k$, remove it if $b_X^+ \leq (|\Delta^+| - |\Delta^-|) \times s\%$.
6. If $|\Delta^-| \leq |\Delta^+|$, let $R_k = \phi$. Otherwise calculate b_X^- for each $X \in Q_k$ and if $b_X^+ - b_X^- \geq (|\Delta^+| - |\Delta^-|) \times s\%$, move it to R_k and assign b_X^- to δ_X^- .
7. Scan Δ^- to find out δ_X^- for each $X \in P_k \cup Q_k$.
8. Delete from P_k those candidates X where $\sigma_X + b_X^+ - \delta_X^- < |D'| \times s\%$.
9. Delete from Q_k those candidates with $b_X^+ - \delta_X^- \leq (|\Delta^+| - |\Delta^-|) \times s\%$.
10. Scan Δ^+ to find δ_X^+ for each $X \in P_k \cup Q_k \cup R_k$.
11. For each candidate $X \in P_k$, calculate σ'_X .
12. For each candidate $X \in Q_k$, delete X if $\delta_X^+ - \delta_X^- \leq (|\Delta^+| - |\Delta^+|) \times s\%$.
13. For each candidate $X \in R_k$, delete X if $\delta_X^+ \leq (|\Delta^+| - |\Delta^+|) \times s\%$.
14. Scan D^- and get the count of each $X \in Q_k \cup R_k$. Then, add this count to δ_X^+ to get σ'_X .
15. Add to L'_k those candidates X from $P_k \cup Q_k \cup R_k$ where $\sigma'_x \geq |D'| \times s\%$.
16. Halt if $|L'_k| < k+1$.

Figure 5.1: FUP2

5.2.3 MAAP

The algorithm, MAAP [ES02] first finds out the old frequent itemsets that will remain frequent in the updated dataset also. Downward closure property of frequent itemsets makes this job little bit simpler. Then it checks for the possible new frequent itemset, and if found, new candidates are generated. For the above mentioned example, it starts with $L_3 = \{ABC, ACD, BCD\}$ and maintains the lists L'_3 , L'_2 and L'_1 . Support of all itemsets in L_3 are computed

and found that **ABC** is frequent, so all the subset of it must be frequent and hence added to the list of corresponding size. Since **ACD** and **BCD** are also frequent, subsets of them are also treated in the similar way and $L'_3=\{ABC, ACD, BCD\}$, $L'_2=\{AB, AC, BC, AD, CD, BD\}$ and $L'_1=\{A,B,C,D\}$ are resulted. These are some of the itemsets that were frequent earlier and remained frequent after the updation. In the next step, itemsets in $L_i - L'_i$ are tested, whether they are frequent or not. In the above example, $L_1 - L'_1=\{E\}$ and $L_2 - L'_2=\{AE\}$. Since E is frequent, it is added to L_1 but AE is not because it is no longer frequent. In the next step, the algorithm checks for the itemsets which were infrequent earlier and computes $S_i = C_i - L_i$. For example, $S_1 = C_1 - L_1=\{F\}$ and $S_2 = C_2 - L_2=\{BE,CE,DE\}$. After scanning the dataset only **BE** was found to be frequent and included in L'_2 resulting in $L'_2=\{ AB, AC, AD, BC, BD, BE, CD\}$. Since some of these itemsets were infrequent in old database, next level higher additional candidates will be generated. Here, *additional* $C'_3=L'_2 \times \{BE\}=\{BCE, BDE\}$ and resulted in $C'_3=\{ABC, ABD, ACD, BCD, BCE, BDE\}$. Since none of the elements of additional C'_3 is frequent, final $L'_3=\{ABC, ACD, BCD\}$. The final frequent itemsets are $L=\{A, B, C, D, E, AB, AC, AD, BC, BD, BE, CD, ABC, ACD, BCD\}$.

5.2.4 Borders

Borders algorithm [AFLM99] finds the frequent itemsets from the dynamic database using the frequent itemsets already discovered from the old dataset. Here the concept of *aborder set* is used. An infrequent itemset is termed as border set, if all the non empty proper subsets of it are frequent. Due to the insertion of new records to the dataset, some of the border sets may become frequent, and is termed as *promoted border set*. For that, the border sets

of the old dataset also have to be maintained along with the frequent sets derived. Based on the promoted border set, some new candidate itemsets are generated and checked for frequent set. For the above example $B_1=\{F\}$ and $B_2=\{BE, CE, DE\}$ and $B_3=\{ ABD, ABE, ACE, ADE\}$ are the border sets. On the updated dataset only **BE** becomes promoted border set. Then the candidates due to the promoted border set are generated. Database is scanned to get the support of the itemsets and the frequent itemsets are found. The candidates are generated if there is at least one promoted border set. This algorithm may require more than one passes of the old dataset depending on the frequent sets discovered due to the incremented part.

The steps of the Borders algorithm that works on an incremental database are presented in Figure 5.2

Input: $R_N, R_O, \alpha, Borders$ and $FrequentSets$ of R_O and their count

Output: $Borders$ and $FrequentSets$ of $R_O \cup R_N$ and their count

1. Scan new relation R_N and find count $c(X, R_N)$, for all $X \in Borders \cup FrequentSets$.
2. **For all** $X \in Borders \cup FrequentSets$ **do**
3. $c(X, R_O, R_N) = c(X, R_O) + c(X, R_N)$
4. $s(X, R_O, R_N) = c(X, R_O \cup R_N) / (n_O + n_N)$
5. **end do**
6. $PromotedBorders = \{X \in Borders \mid s(X, R_O \cup R_N) \geq \alpha\}$
7. $FrequentSets = \{X \in FrequentSets \mid s(X, R_O \cup R_N) \geq \alpha\} \cup PromotedBorders$
8. $Borders = \{X \mid \forall x \in X, X - \{x\} \in FrequentSets\}$
9. $m = \max\{i \mid PromotedBorders(i) \neq \phi\}$
10. $L_0 = \phi$
11. $i=1$
12. **While** ($L_i \neq \phi$ or $i \leq m$) **do**
13. $C_{i+1} = \{X = S_1 \cup S_2 \mid (i) \mid X \mid = i + 1.$
14. (ii) $\exists x \in X, X - \{x\} \in PromotedBorders(i) \cup L_i,$
15. (iii) $\forall x \in X, X - \{x\} \in FrequentSets(i) \cup L_i\}$

16. Scan $R_N \cup R_O$ and obtain $c(X, R_N \cup R_O)$ for all candidates, $X \in C_{i+1}$
17. $L_{i+1} = \{X \mid X \in C_{i+1} \text{ and } c(X, R_N \cup R_O)/(n_O + n_N) \geq \alpha\}$
18. $FrequentSets = FrequentSets \cup L_{i+1}$
19. $Borders = Borders \cup (C_{i+1} - L_{i+1})$
20. $i = i + 1$
21. **end do**

Figure 5.2: Borders (addition)

5.2.5 Efficient Counting Using TID-lists

To improve the support counting algorithm during the update phase, this algorithm ECUT [GGR00] exploits systematic data evolution and the fact that only a very small number of new candidate itemsets need to be counted. The intuition behind this support counting algorithm is similar to that of an index in that it retrieves only the *relevant* portion of the dataset to count the support of an itemset X . The relevant information consists of the set of TID-lists of items in X . ECUT uses TID-lists $\theta(i_1), \dots, \theta(i_k)$ of all items in an itemset $X = \{i_1, \dots, i_k\}$ to count the support of X . The cardinality of the result of the intersection of these TID-lists equals $\sigma(X)$. Since TID-lists consists of transaction identifiers sorted in increasing order, the intersection can be performed easily; the procedure is exactly the same as the merge phase of merge sort.

ECUT (database increment R_N, R_O , History Log)

1. SCAN db to create its *TIDList* of R_N .
2. Generate L^{db} and $NBd(L^{R_N})$.
3. Update counts of itemsets in L^{R_O} and in $NBd(L^{R_O})$ and specify winners as all itemsets that were in $NBd(L^{R_O})$ and now are in $NBd(L^{D'})$.
4. While winners set is not empty.

5. Generate candidates set and empty the winners set.
6. Use the History Log and R_N to update count of candidates and specify new winners.
7. End while

Figure 5.3: ECUT

5.2.6 Maximal Frequent Trend Pattern

This algorithm, MFTP [GAMH06], first compute L^{R_N} , $NBd(L^{R_N})$, $L^{D'}$ and $NBd(L^{D'})$. All itemsets that were in $NBd(L^{R_o})$ and became large are added to a winners set. This winners set is used with $L^{D'}$ to generate the candidates set. This candidates set is filtered against $NBd(L^{R_o})$, based on the theory proved in [GGR00] that a winner itemset should have at least one subset that belongs to $NBd(L^{R_o})$. After filtering, new winners and new candidates are generated and so on till no more winners could be generated. For each itemset in $L^{D'}$ and $NBd(L^{D'})$, the MFTP algorithm (Figure 5.4) constructs and smoothes the time series, and then transforms it into a trend pattern, which is mined for maximal-frequent-trend-pattern. The maximal frequent trend pattern that matches the current pattern is then used to predict the forthcoming trend, and hence, the forthcoming support range. To match two patterns of length l , a match-factor is used. That is if $(match - factor)\%$ of the first $l - 1$ indicators matches, then the two patterns matches, and the l^{th} indicator is the predicted trend.

MFTP (itemset A , R_N , R_O , $HistoryLog$)

1. From History Log, construct the times series of A : $TS(A)$.
2. Calculate the triangular moving average of $TS(A)$: $MA(A)$.
3. Calculate the trend indicators of $MA(A)$: $TR(A)$.
4. Find maximal-frequent-pattern in $TR(A)$ that matches the current pattern.
5. Based on the maximal matched frequent-pattern, predict next support range of A .

Figure 5.4: MFTP

5.2.7 Modified borders

This modified version of the borders algorithm, named as *modified borders* [DB05], aims to minimize the generation of unnecessary candidate sets. In order to do so, it uses an additional user parameter, apart from the minimum support. Correctness and completeness of the frequent itemsets largely depends on these parameters. With proper tuning of these parameters, *modified borders* can perform better than the *Borders* algorithm. When this additional parameter's value is closer to the support, the algorithm converges to the borders algorithm. Depending on this parameter, the border sets are divided into four different sets B' , B'' , B''' and B'''' . The probability of becoming promoted border set is highest for the elements of B' and lowest for B'''' . The algorithm is reproduced in Figure 5.5.

Input: $T_{new}, T_{old}, \alpha, \beta, L_{old}, B'_{old}, B''_{old}$

Output: $L_{whole}, B'_{whole}, B''_{whole}$

Scan T_{new} and increment the support count of $X \in (L_{old} \cup B'_{old} \cup B''_{old})$

$B' = \{X \mid X \in B'_{old} \text{ and } S(X)_{T_{whole}} \geq \alpha\};$

$B'' = \{X \mid X \in B''_{old} \text{ and } S(X)_{T_{whole}} \geq \alpha\};$
 $L_{whole} = B' \cup B'' \cup \{X \mid X \in L_{old} \text{ and } S(X)_{T_{whole}} \geq \alpha\};$
 $B''' = \{X \mid X \in B''_{old}; \forall x \in X, X - \{x\} \in L_{whole}; (S(X)_{T_{whole}} \geq \beta \text{ and } S(X)_{T_{whole}} < \alpha)\};$
 $B'''' = \{X \mid X \in B'_{old} \cup L_{old}; \forall x \in X, X - \{x\} \in L_{whole}; (S(X)_{T_{whole}} \geq \beta \text{ and } S(X)_{T_{whole}} < \alpha)\};$
 $B'_{whole} = B''' \cup B'''';$
 $B''_{whole} = \{X \mid \forall x \in X, X - \{x\} \in L_{whole} \text{ and } S(X) < \beta\};$
 If $B'' \neq \phi$ then $m = \max \{i \mid B''(i) \neq \phi\}$
Candidate – generation :
 $L_0 = \phi; B_0 = \phi; k = 2;$
 while($L_{k-1} \neq \phi$ or $B_{k-1} \neq \phi$ or $k \leq (m - 1)$) do
 $C_k = \phi$
 $L = B''(k - 1) \cup L_{k-1} \cup B'''(k - 1) \cup B_{k-1}$
 $M = L_{k-1} \cup L_{whole}(k - 1) \cup B'_{whole}(k - 1)$
 For all itemsets in $l_1 \in L$ do begin
 For all itemsets in $l_2 \in M$ do begin
 If $l_1[i] = l_2[i]$ ($1 \leq i \leq k - 2$) and $l_1[k - 1] < l_2[k - 1]$ then
 $C = \{l_1[1], l_1[2], \dots, l_1[k - 2], l_1[k - 1], l_2[k - 1]\}$
 $C_k = C_k \cup C$
 End for
 End for
 Prune C_k : All the subsets of C_k of size $(k - 1)$ must be present in M ;
 Scan T_{whole} and obtain support $S(X)$ for all $X \in C_k$
 $L_k = \{X \mid X \in C_k \text{ and } S(X) \geq \alpha\}$
 $L_{whole} = L_{whole} \cup L_k$
 $B_k = \{X \mid X \in (C_k - L_k); \forall x \in X, X - \{x\} \in L_{whole}; S(X) \geq \beta \text{ and}$

$$S(X) < \alpha$$

$$B'_{whole} = B'_{whole} \cup B_k$$

$$B''_{whole} = B''_{whole} \cup \{X \mid X \in (C_k - L_k); \forall x \in X, X - \{x\} \in L_{whole}; S(X) < \beta\}$$

$$k = k + 1;$$

End do

Figure 5.5: Modified Borders

Apart from the above mentioned algorithms several other works can be found in the literature [ATA99, LMDR04, KZY⁺05, LSNP07, HCK07, HLW08, TLJ08, SXG08, OLC08]. All these algorithms have their own strengths and weaknesses, and is capable of handling the incremental rule mining problem. However, from a careful study it was observed that most of these techniques suffer from the following disadvantages:

- A two phase association mining often can be found to be time and resource consuming in case of larger incremental databases.
- Due to conversion of the real-life data into market-basket domain, information loss occurs.
- Single objective function (i.e. based on only frequency of occurrence) based rule generation often can be found to be non-interesting.

5.3 Proposed Method

To address the issues mentioned in the previous section, a single phase incremental association mining technique has been reported here, which can extract the reduced set of interesting rules over the real-life dataset without transforming it into market basket domain. The new technique can be found to be significant in view of the following points:

Table 5.5: Symbols and Notations used

D, R_O	= Dataset
D'	= Updated Dataset
Δ^-	= set of deleted records
Δ^+, R_N	= set of newly added records
$\sigma_x, S(x)$	= support of x
$NBd(L^x), B_x$	= Border sets from dataset x
α	= Minimum support
δ_x^+	= Support of x in Δ^+

- During extraction of the rules, it evaluates the rules based on not only the support count, but also on the other measures like comprehensibility and interestingness.
- It does not require to transform the dataset into market basket domain.
- It avoids the frequent itemset generation phase, rather it generates the rules directly.

5.3.1 MORM in Incremental Databases

This algorithm extracts the association rules from a continuous valued dataset, using genetic algorithm. It is free from the difficulties mentioned in the previous section. During extraction of the rules, generally predictive accuracy or confidence of a rule is used to evaluate the rules. And for this, the support of different sets of items are needed. The above mentioned algorithms concentrate on the efficient extraction of the itemsets that meets the minimum threshold requirements. However, the proposed approach evaluates the rules based on three different measures, namely confidence, comprehensibility and

interestingness. To evaluate these measures, the expressions used in Chapter 3 are used.

The algorithm is capable of generating the rules directly without deriving the frequent itemsets. During the extraction of the rules, it needs to scan the old dataset only once. The algorithm is based on multi-objective genetic algorithm and here, every candidate rule is represented as a chromosome. Using the floating point encoding scheme, the values of attributes are encoded within the chromosome. Attribute names are not needed to encode, as positions of the values within the chromosome are sufficient to get attribute names. Two additional bits are needed for every attribute to represent the involvement of the attribute within the rule. If these two bits are 00, then the attribute next to these two bits appear in the antecedent part and if it is 11 then the attribute appear in the consequent part. And if these two bits contain either 01 or 10, then the attribute is not involved in either part of the rule. Another bit is used to represent the relational operator involved with the attribute. A continuous valued attribute may be involved with either \leq or \geq operator and a categorical valued attribute may be involved with either $=$ or \neq operator. If this bit is 0, then the attribute is involved with \leq or $=$ depending on its type. Some predefined number of bits are used to encode the attribute values. If the database contains n attributes in it, and m bits used to encode every attribute, then finally chromosome length will become $(n \times (m+3))$. For example, the rule $(A \leq 20 \& F \geq 28) \Rightarrow (B \geq 15 \& E \leq 30)$ from a database having attributes **ABCDEF**, will be encoded in the binary string **000101001110111110- - - -01- - - -1101111000111100** if 5 bits are used to encode every attribute. Since the attributes C and D are not involved in the rule, value of them will not play any role in representing the rule, so the bit position reserved for these two attributes may contain 0 or 1 and are marked with a hyphen(-) in the given bit string.

Rules are extracted from the static part, i.e. the old database, using the MORM technique discussed in Chapter 3. These rules are stored and can be used by the decision support system until the database is updated. Due to the addition of the new records to the database, some new rules may come into existence. To extract these rules, if it is there, the algorithm given in Figure 5.6 can be used.

Algorithm MORMI

Input: Database D , incremental part D' , old rules R_O , number of generations G .

Output: A set of non-dominated rules, R_C , from the complete database.

1. Load D' to memory.
2. Generate population P of N chromosomes randomly.
3. $gen = 0$.
4. Decode(p_i), $\forall i, p_i \in P$.
5. Find $SUP(A)$, $SUP(C)$ and $SUP(R)$ from D' , ($\forall R, R \in P$ and $R = A \Rightarrow C$).
6. Calculate *confidence*, *comprehensibility* and *interestingness*.
7. Rank(P_i), $\forall i, p_i \in P$.
8. Fitness(p_i) = $|P| - \text{Rank}(p_i)$, $\forall i, p_i \in P$.
9. $\forall i, p_i \in P$, if Rank(p_i)=1 then Maintain_Elite(p_i).
10. Based on Fitness(p_i) select chromosomes for next generation.
11. Perform multi-point crossover, then mutation to get new population O .
12. Replace population P by O .
13. if $gen++ < G$ then go to Step 4.
14. Find $SUP(A)$, $SUP(C)$ and $SUP(R)$ by scanning $D \cup D'$, ($\forall R, R \in P$ or $R \in R_O$).

15. $R_C = P \cup R_O$.
16. Calculate *confidence*, *comprehensibility* and *interestingness*, $\forall R, R \in R_C$
17. return Q

Figure 5.6: MORMI

The rules generated by the algorithm were finally evaluated by scanning the whole dataset. The algorithm requires at the most one pass of the whole dataset while generating rules. The different objectives of the rules reflect their existence within the whole dataset.

Following lemma is a result of our previous discussion which establishes the efficiency of the algorithm.

Lemma1: For generating the rules that are valid for the whole dataset, MORMI requires at the most one pass of the whole dataset.

The above algorithm can be found to be advantageous in view of the following points.

- The algorithm does not require the data in market basket form , it can work on the original continuous valued dataset.
- No separate frequent itemset generation phase is needed; it can produce the rules directly.
- User parameters like minimum support and minimum confidence are not required here and hence they cannot affect the execution time of rule generation process.
- Needs only one scanning of the whole dataset to produce the correct rules.
- A reduced ruleset will be generated and is controlled by the population size of the genetic algorithm.

5.3.2 Implementation and Results

The algorithm was implemented on a computer with Core2Duo 2.5G Hz processor and 3 GB RAM. During the execution of the algorithm for mining the rules crossover probability of 0.8 and mutation probability of 0.002 were used. 5-point crossover operator was used with the population size 40. 12 bits were used to encode each attribute. The algorithm was tested with several synthetic as well as standard databases. A few of the extracted rules from some standard datasets available at UCI Machine Learning Repository are discussed below.

A. Wisconsin Diagnostic Breast Cancer Database (WDBC)

This dataset contains 569 instances and 32 attributes. First and second are sample code number and class attribute respectively. All other 30 attributes are real valued attributes. Out of these 569 instances, 357 are benign and 212 are of malignant types.

Out of these first 500 instances were treated as the old dataset where 305 and 195 instances were benign and malignant respectively. The remaining 69 instances were considered as the incremental part where 52 instances are benign and 17 are malignant. The rules discovered from this incremental dataset when compared with the rules of the static part, it was observed that 7 new rules are coming up. And then for the other rules also when compared peer to peer, it is found that the values involved in the rules are differing slightly. Due to which the objectives of the generated rules have been changed. This is due to the reason that the database size in terms of records, and frequency count of the rules have an affect on the objective measures. Some rules from this dataset is given in Table 5.6. When these rules

were observed, it was found that in 84 instances the first rule is satisfied and out of 84 instances 66 (78.57%) are of malignant type. Similarly, second rule is satisfied by 56 instances out of which 52 (92.86%) instances are Malignant; and out of 248 instances that satisfy the fourth rule, 236 (95.16%) instances are of benign type.

When this set of rules were compared with the set of rules discovered by the algorithm presented in Section 3.5.2, it was found that both the sets are identical. From this fact it is established that this incremental approach of rule mining is capable of extracting the rules from the database if the whole database is used for extraction also. Similar observation was made for the next two databases also.

A significant amount of time was saved by MORMI to extract these rules from the updated database. The time taken to derive the rules from the static part and the complete dataset by multi objective approach along with time taken to derive the rules by the incremental approach are reported in Table 5.9.

B. Wisconsin Breast Cancer Database (WBC)

This dataset contains 699 instances and 11 attributes. First and last are sample code number and class attribute respectively. All other attributes are real valued attributes. Out of these 699 instances, 458 are benign type and 241 are of malignant type.

The first 600 instances are considered as the static dataset and the remaining 99 as the incremental part. 380 and 220 instances of the static part are benign and malignant type respectively, whereas 78 benign and 21

Table 5.6: Incremental rules from WDBC database

Mean perimeter ≤ 183.023 & Worst area ≤ 3924.124	\Rightarrow	Mean texture ≤ 30.579 & Standard error perimeter ≤ 15.730 & Worst compactness ≤ 0.944 & Worst symmetry ≥ 0.338
Mean radius ≥ 7.029 & Standard error perimeter ≤ 10.94 & Worst symmetry ≥ 361.012	\Rightarrow	Mean area ≥ 249.27 & Standard error smoothness ≤ 0.016
Mean radius ≤ 12.895 & Standard error radius ≤ 0.8346 & Worst area ≤ 901.53	\Rightarrow	Mean concave points ≤ 0.1152 & Standard error perimeter ≤ 6.217 & & Standard error fractal dimension ≥ 0.001

malignant instances are there in the incremental part. No additional rules were extracted due to the incremental part for this database. Reason for not finding new rules may be due to the reason that this database contains less number of attributes, that resulted in a smaller search space for the solutions. But the measures of some of the rules from the static part were changed due to the same reason explained in the discussion of the previous database. But here also the final rules match with those rules extracted by the algorithm described in Section 3.5.2.

Some rules from this dataset are given in Table 5.7. When these rules were observed we have found that in 168 instances the first rule is satisfied and out of these 168 instances 122 (72.62%) are of malignant type. Similarly

second rule is satisfied by 119 instances out of which 109 (91.6%) instances are malignant. And out of 63 instances that satisfy the third rule, 52 (82.54%) instances are of benign type.

Similar to the previous dataset, considerable amount of time was saved for this dataset also, when the algorithm MORMI was used to derive the rules. Table 5.9 presents the time required to extract the rules from the static part and complete dataset by the multi objective rule mining approach and the time required by MORMI.

Table 5.7: Incremental rules from WBC database

Uniformity of Cell Shape ≤ 8.248	\Rightarrow	Marginal Adhesion ≥ 2.708 & Mitoses ≤ 5.730
Single Epithelial Cell Size ≤ 9.888	\Rightarrow	Marginal Adhesion ≥ 4.571 & Mitoses ≤ 5.207
Bland Chromatin ≤ 8.842	\Rightarrow	Clump Thickness ≥ 0.972 & Normal Nucleoli ≥ 4.774 & Mitoses ≤ 5.642

C. Wisconsin Prognostic Breast Cancer (WPBC)

The dataset contains 198 instances and 34 attributes. First and second are sample code number and class attribute, respectively. All other 32 attributes are real valued attributes. Out of these 198 instances, 151 non-recurring, 47 recurring type.

Table 5.8: Incremental rules from WPBC database

Standard error texture ≤ 3.275 & Standard error smoothness ≤ 0.026	\Rightarrow	Mean fractal dimension ≥ 0.093 & Standard error compactness ≤ 0.092 & Worst smoothness ≥ 0.108 & Worst compactness ≥ 0.246 & Worst fractal dimension ≥ 0.068
Standard error symmetry ≤ 0.056 & Worst perimeter ≤ 214.958	\Rightarrow	Mean area ≥ 547.904 & Standard error texture ≤ 3.486 & Worst smoothness ≥ 0.152 & Worst fractal dimension ≥ 0.069
Time ≤ 112.585 & Mean perimeter ≤ 181.616 & Standard error texture ≤ 3.275 & Standard error smoothness ≤ 0.025 & Mean fractal dimension ≤ 0.082	\Rightarrow	Standard error compactness ≤ 0.094 & Worst compactness ≥ 0.264 & Worst fractal dimension ≥ 0.080

Out of the 150 instances, which contains 112 non-recurring and 38 recurring type, are considered as static part. Remaining 48 instances were considered as incremental part that contains 39 non-recurring and 9 recurring instances. After the extraction of the rules from the incremental part, 6 rules were newly found. Table 5.8 presents some rules derived from this dataset. When these rules were observed we have found that in 143 instances the first rule is satisfied and out of these 143 instances 107 (74.83%) are of non-recurring type. Similarly second rule is satisfied by 45 instances out of which 33 (73.33%) instances are non-recurring. And out of 105 instances

that satisfy the third rule, 80 (76.19%) instances are of non-recurring type. Due to the same reason described in the previous databases, objective measures of the rules from the static part and the complete database are differing slightly. But finally for this database also same rules were extracted if the algorithm in Section 3.5.2 is used over the complete database.

Table 5.9 reports the timing information to derive the rules from this dataset also. It can be observed that, a significant amount of time is saved to derive the rules when MORMI is used.

Table 5.9: Comparison of Static and Incremental MORM

Dataset	MORM on Static part(ns)	MORM on Complete(ns)	MORMI (ns)	Time saved (ns)
WDBC	8,923,647,500	10,708,377,000	1,806,146,254	8,902,230,746
WBC	7,235,610,857	8,441,546,000	1,222,818,235	7,218,727,765
WPBC	3,230,902,500	4,307,870,000	1,085,583,240	3,222,286,760
All timing information in this table are average of 15 runs of the program.				

5.4 Discussion

From the above discussion it was observed that this algorithm can extract the association rules from an incremental dataset with a single pass of the whole database. It uses the new incremental part of the dataset several times to discover the association rules. Similar experiments were carried out over several synthetic databases and the results were satisfactory. These synthetic databases had different number of records as well as different number of attributes. From those experiments it was observed that, if the dataset

contains a large number of attributes then probability of finding new rules becomes higher. The size of the incremental part also has some influence over the generated rules. Some new rules also may come into existence for a database with larger increments with smaller number of attributes. The objective measures confidence and interestingness of the rules, that have the influence of frequency within the dataset, differs due to the increment of the dataset. However an interesting observation made over these experiments was that same rules were extracted by the algorithm described in Section 3.5.2, when applied on the complete database.

Chapter 6

Conclusions and Future works

6.1 Conclusions

In this dissertation a study of different issues of association rule mining is presented. After a careful study of the different association rule mining algorithms it was found that all of them treat the problem as a single objective one, where finally the confidence of the rules are maximized. All of those algorithms are based on the approach pioneered by Agrawal et. al. [AIS93]. Generation of the rules are done in two phases namely *frequent itemset generation* and *rule generation*. First phase being the most crucial one, the existing algorithms have given more importance on it. A very few algorithms were found that attended the second phase. In the chapter 2, an efficient algorithm to attend the rule generation phase was presented. From the various experiments it has been found that the new algorithm works faster than all the other existing algorithms.

Though the association rule mining was handled as single objective prob-

lem by the existing algorithms, from the study of the association rules it was found that it should be treated as a multi-objective problem. Considering *comprehensibility* and *interestingness* as two other objectives along with *confidence*, association rule mining problem can be handled as multi-objective problem. After testing several expressions finally, two expressions to quantify comprehensibility and interestingness were formulated. In Chapter 3, two algorithms based on Pareto genetic algorithm are presented to extract the association rules from the database. In literature, Pareto genetic algorithm was found to be a better technique to handle the multi-objective problem. Out of the two algorithms presented in the Chapter 3, second one has been found to be more efficient.

During the study of the association rule mining algorithms, it was observed that the performance of the algorithms are affected by the dimensionality of the database. But almost all databases contain some irrelevant attributes (dimensions) in it. Several works have been found in the literature to attend this issue of data mining, commonly known as *dimensionality reduction*. Need of a dimensionality reduction technique suitable for the association rule mining technique has lead to the work presented in Chapter 4. Here two efficient algorithms for dimensionality reduction, having their own strengths are presented. Depending on the decision maker's need any of the algorithms can be used to reduce the size of the database in terms of attributes.

Mining association rules over an incremental database is yet another challenging problem of data mining. Several works based on the approach due to Agrawal et.al., can be found in the literature. Incremental rule mining being an allied area of association rule mining, a need of an algorithm to extract

the rules considering the multiple objectives was felt. This need has lead us to the development of the multi-objective incremental rule mining algorithm presented in the Chapter 5. From the various experiments it has been found that the proposed algorithm is efficient to extract the association rules using multiple objectives.

From the study and various experiments carried out during this dissertation work it is found that the association rule mining problem should be handled as a multi-objective problem rather than single objective one. To attend different issues related to the multi-objective association rule mining some efficient techniques are developed and presented here.

6.2 Future works

In this dissertation the association rule mining is treated as a multi-objective problem. And several techniques to attend different issues of association rule mining are presented here. Still there are some works left unattended and some works may need more tuning.

- Comprehensibility of the association rule is used as a measure of the rules. But it is difficult to quantify a subjective measure like comprehensibility. After considering a number of expressions, it was found that the expression used in this dissertation gives a better representation of the comprehensibility of association rules. It is not ensured that this is the best expression to quantify it. So this expression needs some more attention.

- Similarly expression used for the other subjective measure, i.e. interestingness also needs some attention.
- Most of all, the algorithms presented here are tested over some continuous(numeric) valued databases only. But, by nature, association rule mining is not restricted to continuous(numeric) valued databases. So the testing of the algorithms over categorical(nominal) valued database as well as mixed valued database is left to be done.

Bibliography

- [AD92] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of Ninth National Conference On Artificial Intelligence*, pages 547–552, Cambridge, Massachusetts, 1992. MIT press.
- [AFLM99] Y. Aumann, R. Feldman, O. Lipshtat, and H. Manilla. Borders : An efficient algorithm for association generation in dynamic databases. *Journal of Intelligent Information System*, 12(1):61–73, April 1999.
- [AIS93] R. Agrawal, T. Imeilinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceeding of ACM SIGMOD Conference on Management of data*, pages 207–216, Washington D.C., May 1993.
- [AP95] R. Agrawal and G. Psaila. Active data mining. In *proceedings of the 1st international Conference on KDDM*, pages 3–8, Montreal, August 1995.
- [AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conference on Very Large Databases*, pages 487–499, Santiago, Chile, Sept. 1994.

- [ATA99] N. F. Ayan, A. U. Tansel, and M.E. Arkun. An efficient algorithm to update large itemsets with early pruning. *Knowledge Discovery and Data Mining*, pages 287–291, 1999.
- [BFM98] P. Bradely, U. Fayyad, and O. Mangasarian. Data mining: Overview and optimization opportunities. research report MST-TR-98-04, Microsoft, january 1998.
- [BL97] M. J. Berry and G. Linoff. *Data mining techniques for marketing, sales and customer support*. John Wiley and Sons, 1997.
- [BMTU97] S. Brin, R. Motwani, D. Tsur, and J. Ullman. Dynamic itemset counting and implication rules for market basket data. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, USA, 1997.
- [Car93] C. Cardie. Using decision trees to improve case based learning. In *Proceedings of Tenth International Conference on Machine Learning*, pages 25–32, Amherst, MA, 1993. Morgan Kaufmann.
- [CC61] A. Charnes and W.W. Cooper. *Management models and industrial applications of linear programming*. Wiley, New York, 1961.
- [CCF55] A. Charnes, W.W. Cooper, and R Ferguson. Optimal estimation of executive compensation by linear programming. *Management Science*, 1(138-151), 1955.
- [CHNW96] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. of 12th International Conference on Data Engineering*, pages 106–114, New Orleans, Louisiana., 1996.

- [CHY96] M. S. Chen, J. Han, and P. S. Yu. Data mining an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.
- [CKN08] J. Cheng, Y. Ke, and W. Ng. A survey on algorithms for mining frequent itemsets over data streams. *Knowledge and Information Systems*, 16(1):1–27, July 2008.
- [CLK97] D. W. Cheung, S. D. Lee, and B. Kao. A general incremental technique for maintaining discovered association rules. In *Proceedings of the 5th International Conference on Database System for Advanced Applications*, pages 185–194, Melbourne, Australia., 1997.
- [CLV07] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.
- [Coe96] Carlos Artemio Coello Coello. *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design*. PhD thesis, Department of Computer Science, Tulane University, New Orleans, Louisiana, April 1996.
- [Coe99] C. A. C. Coello. A comprehensive survey of evolutionary-based multi-objective optimization technique. *Knowledge and Information Systems*, 1:269–308, 1999.
- [CPS00] K. J. Cios, W. Pedrycz, and R. W. Swiniarski. *Data mining methods for knowledge discovery*. Kluwer Academic Publishers, 2000.

- [CTL09] C-J. Chu, S. V. Tseng, and T. Liang. Efficient mining of temporal emerging itemsets from data streams. *Expert Syst. Appl.*, 36(1):885–893, 2009.
- [DB04] A. Das and D. K. Bhattacharyya. Feature selection using frequency count. In *proceedings of 12th international conference on Advanced Computing and Communications (ADCOM2004)*, pages 620–624, Ahmedabad, India, December 2004.
- [DB05] A. Das and D. K. Bhattacharyya. Rule mining for dynamic databases. *AJIS*, 13(1):19–39, 2005.
- [Deb01] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.
- [DL97] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1:131–156, 1997.
- [Doa92] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical report, University of California, Department of Computer Science, 1992.
- [ES91] L. J. Eshelman and J. D. Schaffer. Preventing premature convergence in genetic algorithms by preventing incest. In *Proceedings of the 4th. Int. Conf. on GA*, pages 115–122. Morgan Kaufmann, 1991.
- [ES02] C. I. Ezeife and Y. Su. Mining incremental association rules with generalized FP Tree. In *Proceedings of 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, pages 147–160. Calgary, Canada, May 2002.

- [FC07] T-H. Fan and K-F. Cheng. Tests and variables selection on regression analysis for massive datasets. *Data & Knowledge Engineering*, 63(3):811–819, December 2007.
- [FF93] C. M. Fonseca and P. L. Fleming. Genetic algorithms for multi-objective optimization formulation, discussion and generalization. In S. Forrest, editor, *Proc. of 5th International Conference on Genetic Algorithm*, pages 416–423, San Mateo, CA, 1993. Morgan Kaufmann.
- [FF95] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multi-objective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
- [FL07] T-C. Fu and C-L. Lui. Agent-oriented network intrusion detection system using data mining approaches. *International Journal of Agent-Oriented Software Engineering*, 1(2):158–174, July 2007.
- [Fla76] R.B. Flavell. A new goal programming formulation. *The International Journal of Management Science, Omega*, 4(6):731–732, 1976.
- [FLFG00] M. V. Fidelis, H. S. Lopes, A. A. Freitas, and Ponta Grossa. Discovering comprehensible classification rules with a genetic algorithm. In *Proc. of Congress on Evolutionary Computation*, pages 805–810, 2000.
- [FPSM91] W. Frawley, G. Piatasky-Saprio, and C. Matheus. *Knowledge discovery in data bases: an overview*. AAAI/MIT Press, 1991.

- [Fre01] A. A. Freitas. Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review*, 16(3):177–199, November 2001.
- [Fre02] A. A. Freitas. *Data mining and knowledge discovery with evolutionary algorithms*. Springer-Verlag, 2002.
- [GAMH06] S. Guirguis, K. M. Ahmed, N. M. E. Makky, and A. M. Hafez. Mining the future: Predicting itemsets support of association rules mining. In *Proceedings of Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 474–478, 2006.
- [GE08] S. Gunal and R. Edizkan. Subspace based feature selection for pattern recognition. *Information Sciences: an International Journal*, 178(19):3716–3726, October 2008.
- [GGR00] V. Ganti, J. Gehrke, and R. Ramakrishnan. DEMON: mining and monitoring evolving data. In *IEEE Transactions on Knowledge and Data Engineering*, pages 439–448, 2000.
- [GN04] A. Ghosh and B. Nath. Multi-objective rule mining using genetic algorithm. *Information Sciences, Elsevier*, 163(1-3):123–133, 2004.
- [Gol89] David E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.
- [GS93] D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13(2-3):229–257, Nov/Dec 1993.

- [HCK07] J-P. Huang, S-J. Chen, and H-C. Kuo. An efficient incremental mining algorithm-qs. *Intelligent Data Analysis*, 11(3):265–278, August 2007.
- [HCX08] J-J. Huang, Y-Z. Cai, and X-M. Xu. A parameterless feature ranking algorithm based on MI. *Neurocomputing*, 71(9):1656–1668, March 2008.
- [HLS+07] Z. Huang, J. Li, H. Su, G. S. Watts, and H. Chen. Large-scale regulatory network analysis from microarray data: modified bayesian network learning and association rule mining. *Decision Support Systems*, 43(4):1207–1225, August 2007.
- [HLW08] T-P. Hong, C-W. Lin, and Y-L. Wu. Incrementally fast updated frequent pattern trees. *Expert Systems with Applications: An International Journal*, 34(4):2424–2435, May 2008.
- [Hol75] John H Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
- [HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candiadate generation. In *Proc. 2000 ACM-SIGMOD Int. Conference on Management of data*, pages 1–12, Dallas, Texas, USA, May 2000.
- [HS95] M. Houtsma and A. Swami. Set-oriented mining of association rules in relational databases. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 25–33, March 1995.
- [Ign76] J.P. Ignizio. *Goal programming and extensions*. Lexington Books, Lexington, 1976.

- [JT02] D.F. Jones and M Tamiz. *Goal programming in the period 1990-2000*. in Multiple Criteria Optimization: State of the art annotated bibliographic surveys, M. Ehrgott and X. Gandibleux(eds). Kluwer, 2002.
- [KCN08] Y. Ke, J. Cheng, and W. Ng. Correlated pattern mining in quantitative databases. *ACM Transactions on Database Systems (TODS)*, 33(3):1–45, August 2008.
- [KP94] R. R. Kohavi and K. Peger. Irrelevant features and subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.
- [KR92] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of Tenth National Conference on Artificial Intelligence*, pages 129–134. MIT Press,, 1992.
- [KS95] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: Over fitting and dynamic search space topology. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, pages 192–197. Morgan Kaufman, 1995.
- [KS96] D. Koller and M. Sahami. Towards optimal feature selection. In *Proceedings of 13th International Conference on Machine Learning*, pages 284–292, 1996.
- [KZY+05] B. Kao, M. Zhang, C-L. Yip, D. W. Cheung, and Usama Fayyad. Efficient algorithms for mining and incremental update of maximal frequent sequences. *Data Mining and Knowledge Discovery*, 10(2):87–116, March 2005.

- [LK98] D. I. Lin and Z. M. Kedem. Pincer-search: an efficient algorithm for discovering the maximal frequent set. In *Proceedings of 6th European Conference on Extending Database Technology*, pages 105–119, March 1998.
- [LMDR04] J. Li, T. Manoukian, G. Dong, and K. Ramamohanarao. Incremental maintenance on the border of the space of emerging patterns. *Data Mining and Knowledge Discovery*, 9(1):89–116, July 2004.
- [LS96] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of International Conference on Machine Learning*, pages 319–327, 1996.
- [LSNP07] P-A. Laur, J-E. Symphor, R. Nock, and P. Poncelet. Statistical supports for mining sequential patterns and improving the incremental update process on data streams. *Intelligent Data Analysis*, 11(1):29–47, January 2007.
- [NF77] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature selection. *IEEE Transaction on Computers*, C-26(9):917–922, September 1977.
- [NFL99] E. Noda, A. A. Freitas, and H. S. Lopes. Discovering interesting prediction rules with a genetic algorithm. In *Proceedings of the Congress on Evolutionary Computation*, pages 1322–1329, 1999.
- [NLWF05] S-C. Ngan, T. Lam, R. C-W. Wong, and A. W-C Fu. Mining n-most interesting itemsets without support threshold by the cofitree. *International Journal of Business Intelligence and Data Mining*, 1(1):88–106, July 2005.

- [OLC08] J-C. Ou, C-H. Lee, and M-S. Chen. Efficient algorithms for incremental web log mining with dynamic thresholds. *The International Journal on Very Large Data Bases*, 17(4):827–845, July 2008.
- [Oli93] J. R. Oliver. Discovering individual decision rules: an application of genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 216–222, Urbana-Champaign, 1993.
- [Par96] V. Pareto. *Cours d’Economie Politique.*, volume I and II. F. Rouge, Lausanne:, 1896.
- [PCY95] J.S. Park, M. S. Chen, and P.S. Yu. An effective hash based algorithm for mining association rules. In *Proceedings of ACM SIGMOD*, pages 175–186, May 1995.
- [Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Matco, California, 1993.
- [RFJT08] M. X. Ribeiro, M. R. P. Ferreira, C. Traina. Jr., and A. J. M. Traina. Data pre-processing: a new algorithm for feature selection and data discretization. In *Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*, pages 252–257, Paris, France, October 2008.
- [SAD⁺93] M. Stomebraker, R. Agrawal, U. Dayal, E.J. Neuhold, and A. Reuter. The dbms research at crossroads. In *Proc. of the VDLB Conference*, pages 688–692, Dublin, August 1993.
- [SB04] K. Srikumar and B. Bhasker. Efficiently mining maximal frequent sets in dense databases for discovering association rules. *Intelligent Data Analysis*, 8(2):171–182, April 2004.

- [SD93] N. Srinivas and K. Deb. Multiobjective optimization using non-dominated sorting in genetic algorithms. Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1993.
- [SD94] N. Srinivas and K. Deb. Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
- [SDN90] J. Shieinvald, B. Dom, and W. Niblack. A modelling approach to feature selection. In *Proceedings of Tenth International Conference on Pattern Recognition*, pages 535–539, June 1990.
- [SDZ07] A. Salappa, M. Doumpos, and C. Zopounidis. Feature selection algorithms in classification problems: an experimental evaluation. *Optimization Methods and Software*, 22(1):199–212, February 2007.
- [Sha85] J. D. Shaffer. Multiple objective optimization with vector evaluated genetic algorithm. In *First International Conference on Genetic Algorithm*, pages 93–100, 1985.
- [SL08] S. J. Shin and W. S. Lee. On-line generation association rules over data streams. *Information and Software Technology*, 50(6):569–578, May 2008.
- [SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st conf. on Very Large Databases*,, pages 432–443, Zurich, Switzerland, September 1995.

- [Sri96] R. Srikant. *Fast algorithms for mining association rules and sequential patterns*. Phd thesis, University of Wisconsin, Madison, 1996.
- [SXG08] G. Shaw, Y. Xu, and S. Geva. Deriving non-redundant approximate association rules from hierarchical datasets. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 26–30, Napa Valley, California, USA, October 2008.
- [TLJ08] M-C. Tseng, W-Y. Lin, and R. Jeng. Incremental maintenance of generalized association rules under taxonomy evolution. *Journal of Information Science*, 34(2):174–195, April 2008.
- [XL07] Y. Xu and Y. Li. Generating concise association rules,. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 6–10, Lisbon, Portugal, November 2007.
- [Yun07] Unil Yun. Efficient mining of weighted interesting patterns with a strong weight and/or support affinity. *Information Sciences: an International Journal*, 177(17):3477–3499, September 2007.
- [ZDT00] E. Zitzler, K. Deb, and L. Thiele. Comparison of multi-objective evolutionary algorithms: empirical results. *Evolutionary Computation*, 8(3):173–195, June 2000.
- [ZKCY07] M. Zhang, B. Kao, D. W. Cheung, and K. Y. Yip. Mining periodic patterns with gap requirement from sequences. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(2):7–13, August 2007.

List of publications

- A. Ghosh, B. Nath, **Multi-objective rule mining using genetic algorithms**, Information Sciences, vol 163, No 1-3, pp 123-133, Elsevier Inc. 2004.
- B. Nath, D.K. Bhattacharyya, A. Ghosh, **Faster Generation of Association Rules**, IJITKM, vol 1, No 2, pp 267-279, Serials Publications, India, July-December 2008.
- B Nath and D K Bhattacharyya, **Partition Based Association Rule Mining in Cancer Databases Using Multi-objective Genetic Algorithms** in the Proc. of BIOT'2005, Colorado, 2005. PP 85-90.
- B Nath, D K Bhattacharyya and A Ghosh, **Partition Based Association Rule Mining Using Multi-objective Genetic Algorithm**, in the Proc of ADCOM'2005, December, 2005.
- B Nath, D K Bhattacharyya and A Ghosh, **Mining for Association Rules using Multi-objective Genetic Algorithms: A partition Based Approach**, in the Proc. of NCTAC, 2007, Tezpur, pp. 119-124.
- B Nath, D K Bhattacharyya and A Ghosh, **Dimentionality Reduction using Multi-objective Genetic Algorithm** in the Proc. of NCTAC, 2007, Tezpur, pp. 155-161.
- B Nath, D K Bhattacharyya and A Ghosh, **Frequency count based filter for Dimensionality Reduction**, in the Proc. of ADCOM07 (IEEE), pp 377-381, 2007, Guwahati, India.

- S. Das, B. Nath, **Dimesionality Reduction using Association Rule Mining**, in the Proc. of ICIIS 2008, pp 1-6, December 8-10, 2008, Kharagpur, India.
- B Nath, D K Bhattacharyya and A Ghosh, **Discovering Association Rules from Incremental Datasets**, IJDMMM, Inderscience publishers, (Communicated).
- B Nath, D K Bhattacharyya and A Ghosh, **Dimensionality Reduction for Association Rule Mining**, IJKDB, IGI Global, (Communicated)