Spring 2010

Associative pattern mining for supervised learning

Harpreet Singh

ASSOCIATIVE PATTERN MINING FOR SUPERVISED LEARNING

by

Harpreet Singh, B.Tech

A Dissertation Presented in the Partial fulfillment
of the Requirement for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2010

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

03/31/2010
Date

We hereby recommend that the dissertation prepared under our supervision by Harpreet Singh

entitled ASSOCIATIVE PATTERN MINING FOR SUPERVISED LEARNING

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

[illegible]
Supervisor of Dissertation Research

[illegible]
Head of Department

COMPUTATIONAL ANALYSIS AND MODELING
Department

Recommendation concurred in:

[illegible]

[illegible]

[illegible]

Jinko Kanno 3/31/10

Advisory Committee

Approved:

[illegible]
Director of Graduate Studies

[illegible]
Dean of the College

Approved:

[illegible]
Dean of the Graduate School

GS Form 13a
(6/07)

ABSTRACT

The internet era has revolutionized computational sciences and automated data collection techniques, made large amounts of previously inaccessible data available and, consequently, broadened the scope of exploratory computing research. As a result, data mining, which is still an emerging field of research, has gained importance because of its ability to analyze and discover previously unknown, hidden, and useful knowledge from these large amounts of data. One aspect of data mining, known as frequent pattern mining, has recently gained importance due to its ability to find associative relationships among the parts of data, thereby aiding a type of supervised learning known as "associative learning".

The purpose of this dissertation is two-fold: to develop and demonstrate supervised associative learning in non-temporal data for multi-class classification and to develop a new frequent pattern mining algorithm for time varying (temporal) data which alleviates the current issues in analyzing this data for knowledge discovery. In order to use associative relationships for classification, we have to algorithmically learn their discriminatory power. While it is well known that multiple sets of features work better for classification, we claim that the isomorphic relationships among the features work even better and, therefore, can be used as higher order features. To validate this claim, we exploit these relationships as input features for classification instead of using the underlying raw features. The next part of this dissertation focuses on building a new classifier using associative relationships as a basis for the multi-class classification problem. Most of the existing associative classifiers represent the instances from a class in a row-based format wherein one row represents features of one instance and extract association rules from the entire dataset. The rules formed in this way are known as "class constrained rules," as they have class labels on the right side of the rules. We argue that this class constrained representation schema lacks important information that is necessary for multi-class classification. Further, most existing works use either the intra-class or inter-class importance of the association rules, both of which sets of techniques offer empirical benefits. We hypothesize that both intra-class and inter-class variations are important for fast and accurate multi-class classification. We also present a novel weighted association rule-based classification mechanism that uses frequent relationships among raw features from an instance as the basis for classifying the instance into one of the many classes. The relationships are weighted according to both their *intra-class* and *inter-class* importance.

The final part of this dissertation concentrates on mining time varying data. This problem is known as "inter-transaction association rule mining" in the data-mining field. Most of the existing work transforms the time varying data into a static format and then use multiple scans over the new data to extract patterns. We present a unique index-based algorithmic framework for inter-transaction association rule mining. Our proposed technique requires only one scan of the original database. Further, the proposed technique can also provide the location information of each extracted pattern. We use mathematical induction to prove that the new representation scheme captures all underlying frequent relationships.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author [illegible]

Date 05/12/2010

GS Form 14
(5/03)

DEDICATON

To

My Parents

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLDGEMENTS

First and foremost I would like to thank my dissertation advisor, Dr. Sumeet Dua, for his direction, support, patience, and encouragement throughout my graduate work. Not only is he an excellent researcher, but he is also a great mentor. It was his vision and insight that led me to the field of data mining and helped me through many difficult times. This dissertation would not have been possible without his constant and unwavering support.

I would like to extend my thanks to the honorable committee members, Dr. Weizhong Dai, Dr. Katie Evans, Dr. Jinko Kanno, and Dr. Abdelkader Baggag, for their valuable discussions, patience, and cooperation for improving the content of this dissertation.

I would further like to thank my colleagues and friends at the Data Mining and Research Laboratory, Pradeep Chowriappa, Sheetal Saini, Michael Dessauer, Alan Alex, and Xian Du for being supportive of my work and always willing to provide insights. They have all helped in making our lab an enjoyable and productive workplace. I would like to extend my thanks to my colleague Brandy McKnight for proofreading my dissertation and providing valuable insight for improving the presentation. I would also like to thank my friends outside the lab, Dheeraj Sharma, Gursimran Singh Walia, Ravipal Singh Randhawa, Munish Kumar, and Gurjot Singh, for their support and encouragement.

Finally, I would like to thank my parents, Badan Singh and Paramjit Kaur, my sister, Gurpreet Kaur, and my brother, Manpreet Singh, for their unconditional love and support throughout my dissertation work.

CHAPTER 1

INTRODUCTION

1.1 Data Mining and Why it is Important

The growth of multi-dimensional datasets has exploded since the beginning of the internet era. Collecting and sharing large amounts of data has become easy due to the ability of the internet to house large data repositories and the ease of sharing such repositories. Multi-national companies, supermarkets, and scientific research fields routinely collect hundreds and thousands of giga bytes (GB) and tera bytes (TB) of data. However, unless we analyze this massive amount of data and put it to good use, it is nothing more than an ineffectual storage place. Such situations in which researchers have an abundance of data but not enough data analysis tools for learning from that data, are aptly labeled as "*data rich but information poor*" scenarios [1]. The sheer size of this data presents daunting challenges since most of the existing data analysis methods are not adept at handling such high dimensionality data alone.

Our quest to find meaningful information from data is not new. For centuries, humans have manually examined data to generate significantly useful patterns. However, with the amount of data growing by a factor of 10 every five years [2], humans can no longer produce meaningful results in adequate time. Data mining, also known as knowledge discovery in databases (KDD), provides a solution to this problem in the form of sophisticated algorithmic tools which can replace humans and are more reliable and faster. According to one of the many definitions, data mining is "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [3]. Currently, data mining tools are used in a wide range of domains, for example in the market place to help organizations make better decisions and develop more efficient business strategies; in the scientific field to help uncover important data patterns by providing exploratory data analysis and predictive and descriptive data modeling; and in the medical filed to help technicians make better decisions about patient diagnoses and information retrieval.

1.2 The KDD Process

As with the many uses of data mining, the exact definition of the KDD process varies from person to person. While some people may treat data mining synonymously with KDD, others think of data mining as a separate and intricate step in the KDD process [1]. The work of Fayyad et al. [4] clearly distinguishes data mining from KDD wherein data mining, as a step in the KDD process, provides results, which are then transformed into useful information. Knowledge discovery is an iterative and data driven approach, which consists of the following steps [1]:

1. **Data Cleaning**: Raw data captured from different sources cannot always be used for analysis due to the inherent noise and inconsistencies present. Data might be missing due to faulty data capturing mechanisms or due to withholding of information for privacy concerns, (e.g. a social security number or an address). These inconsistencies need to be fixed before any other operation can be performed on the data.

2. **Data Selection**: Not all of the data available is useful, as the knowledge mining procedure is usually task driven. Therefore, only the data relevant to the task needs to be selected from complete data.
3. **Data Transformation/Preprocessing**: The data needs to be transformed into a format that is compatible with the knowledge mining tools used and to offer better insights on the underlying data distribution. Discretization, binning, and normalization are examples of the transformation and preprocessing operations.
4. **Data Mining**: Data mining is the most important step in KDD. In this step, intelligent pattern mining methods are used to generate relationships, trends, patterns and anomalies present in the data.
5. **Pattern Evaluation**: Only some of the underlying patterns extracted from the data are interesting and useful. Moreover, the data mining tools can discover not all interesting patterns. Hence, it is important to use an interestingness measure to interpret and evaluate the patterns and to narrow the extracted information to usable patterns.
6. **Knowledge Presentation:** The results are validated using information from the problem domain and presented using visualization and knowledge representation techniques to the user.

One of the most significant topics of scientific computing research in the field of data mining due to its practical and theoretical importance is *frequent pattern mining*. Two key reasons for the interest in frequent pattern mining are its ability to provide key insight by summarizing the data and its ability to serve as a preprocessing tool for other data mining tasks like classification ([5], [6], [7], and [8]), clustering ([9], [10], and [11]), and event prediction ([12], [13], and [14]). Figure 1.1 shows the complete KDD process.

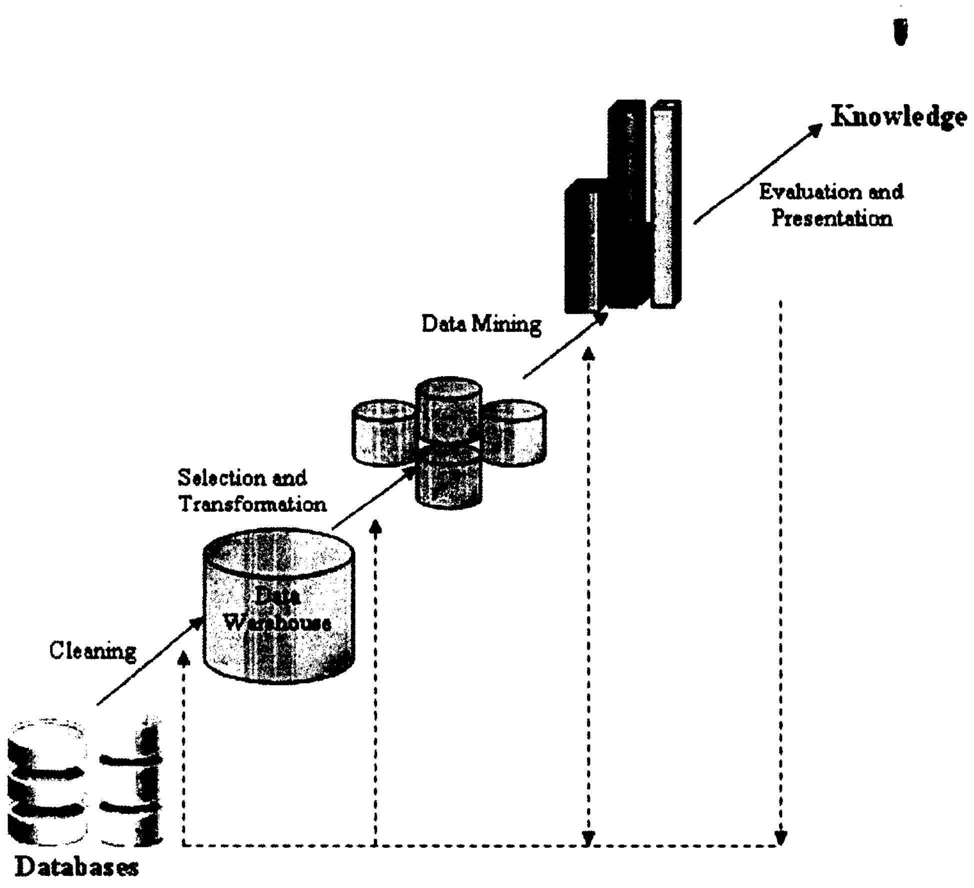


Figure 1.1 Steps in the KDD Process

1.3 Frequent Pattern Mining

Depending on the type of data available and the type of patterns to be mined, frequent pattern mining can be segregated either as frequent itemset mining or as temporal pattern mining.

1.3.1 Frequent Itemset Mining

The problem of frequent itemset mining, also called association rule mining, was first introduced by Agarwal et al. [15] for mining associative relationships from large-scale databases, typically "market-basket" databases. Their algorithm called "Apriori" used the strong rule interestingness measures put forward by Piatetsky Shapiro [16] as their basis for itemset mining. The motivation for this step was to formulate a way to link one type of behavior to another in order to understand common trends in user purchasing habits, which could then be used to maximize profit. One of the classical examples of association rules is the "beer and diaper" problem. The example states that "when men bought diapers on Thursdays and Saturdays, they also tended to buy beer" [17]. While such a pattern may not be implicit to a human observer and such patterns are difficult, if not impossible, to perceive, the pattern shows how data mining can extract hidden patterns from the data. We formally define the frequent itemset mining problem as follows.

Let $I = \{i_1, i_2, ..., i_M\}$ be a set of items and D be a database consisting of transactions $T_1, T_2, ..., T_N$. A subset $A \subseteq I$ is called an *itemset* and an itemset with k items is known as a *k-itemset*. Each transaction $T_i = \{i, P\}$ is a tuple such that $1 \leq i \leq N$ and P is a set of items such that $P \subseteq I$. An association rule can then be represented as $X \Rightarrow Y$. Here, the left-hand side of the rule is known as an *antecedent*, and the right-hand side is known as a *precedent*. The following three constraints are required for a rule to be called an association rule:

1) $\exists T_i \mid X \cup Y \subseteq P, P \in T_i$,

2) $X \subseteq I, Y \subseteq I$, and

3) $X \cap Y = \phi$.

Two measures of rule interestingness are associated with each rule: *support* and *confidence*. Support gives the percentage/probability of transactions in D that contain both X and Y. This percentage can also be represented by probability, $P(X \cup Y)$.

$$Support(X \Rightarrow Y) = P(X \cup Y) =$$

$$\frac{|T_i \mid \{X \cup Y\} \subseteq T_i|}{|D|} = \frac{\text{Number of Transactions with } \{X \cup Y\}}{\text{Total Number of Transactions}}.$$

The confidence of a rule gives the conditional probability of the occurrence of precedent of the rule, given the antecedent has already occurred, $P(Y \mid X)$. For example:

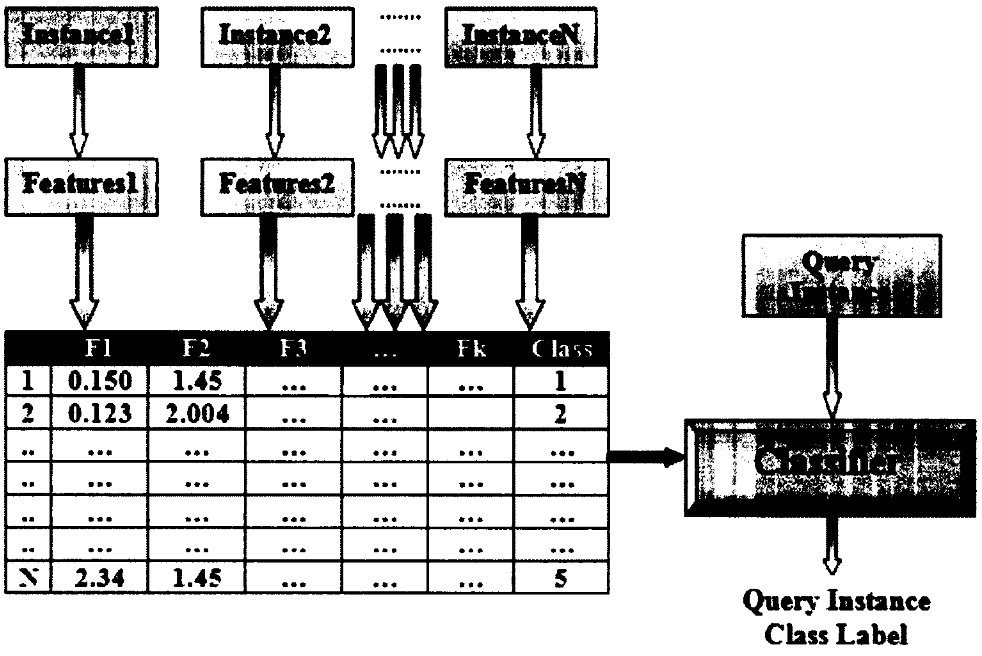
$$Confidence(X \Rightarrow Y) = P(Y \mid X) = \frac{|T_i \mid \{X \cup Y\} \subseteq T_i|}{|T_i \mid X \subseteq T_i|} = \frac{\text{Number of Transactions with } \{X \cup Y\}}{\text{Number of Transactions having X}}$$

An association rule is called a *frequent* association rule if the support and confidence of a rule are greater than a user specified minimum support and confidence. Since the items in the rule are present in the same transaction, we call these rules *intra-transaction* rules. The apriori algorithm proposed by Agarwal et al. is a commonly used frequent association rule-mining algorithm [15]. Since these types of rule mining algorithms generate rules whereby individual items happen at the same time, associating items in the same transaction, they are also known as *intra-transaction* association rules.

1.3.2 Associative Classification

Both supervised and unsupervised learning techniques play an important role in the data mining process. Supervised learning methods are generally more abundant and successful in application, primarily because their distinctive learning component tries to extract, interpret and employ the natural behavior of data distribution [18]. Classification is a supervised learning method and the primary goal of any classification algorithm is to group data into categories based on similarity or dissimilarity measures. A classifier usually extracts some knowledge from the training data to build a classification model, which is then used to categorize/classify previously unknown instances.

The classification problem can be defined as follows: given C, a set of class labels and training data $\{(T_1,c_1),(T_2,c_2),....,(T_N,c_N)\}$, where $T_i, \forall 1 \le i \le N$ represents the feature set of i^{th} instance, N= total number of instances, and $c_i \in C$ represents a class label of the i^{th} instance, build a classifier $\delta : T \to C$ which maps a query instance T_k to its class label $c_i \in C$. Figure 1.2 shows a typical classification mechanism.



	F1	F2	F3	...	Fk	Class
1	0.150	1.45	...	...	...	1
2	0.123	2.004	...	...		2
..	...	...	...	...	...	...
..	...	...	...	...	...	...
..	...	...	...	...	...	...
..	...	...	...	...	...	...
N	2.34	1.45	...	...	...	5

Figure 1.2 A Normal Classification Mechanism

The type of model used for classification and the representation of knowledge gathered from training data depends upon the classifier used. Classification using associative patterns as the base of knowledge is known as "*associative classification*."

Associative classifiers generally connect features of the data instances based on their co-occurrences, regulatory behavior or other interdependencies. These "connections" are then transformed into classifier rules, which are then employed for supervised learning purposes. It must be noted that this process of discovering associations is not aimed at reducing the dimensionality of the data and can at times leads to a gain or loss in dimensionality. This change in dimensionality results from the fact that the values of each feature are usually converted to a set of categorical values, which create pseudo instances of that feature. While the associations between these features will be fewer than the number of these instances, they can be more or less in number than the original set of features.

One such area where associative classifiers are a better fit than regular classifiers is toxicity analysis [19]. It is not considered a viable option to measure the in-vivo response of each chemical in some experiments. As a result, scientists usually resort to the classification models to predict/classify the in-vivo response of a chemical based on its in-vitro response or the chemical compound structure. While regular classifiers can perform this task efficiently, they still lack the functionality to provide the actual causal rules such as which chemical compound combinations lead to what type of in-vivo response and by what probability. Associative relationships on the other hand easily capture relationships like a low dosage of compound A and compound B will lead to a high in-vivo response 80% of the time or a high in-vitro response of chemical K combined with the low dosage of compound A will result in a low in-vivo response almost 95% of the time. These types of rules could be more helpful to researchers in predicting the performance of new chemicals.

Since the performance of a classifier depends heavily on the type of data examined, no single classifier can be considered the "best" for all situations. Therefore, researchers are always searching for ways to improve the existing classifiers and build new ones for more robust and scalable results. Associative classifiers have recently become more important because of the ease with which their results can be interpreted and because of the high levels of accuracy. Figure 1.3 shows a typical association-rule-based classification mechanism.

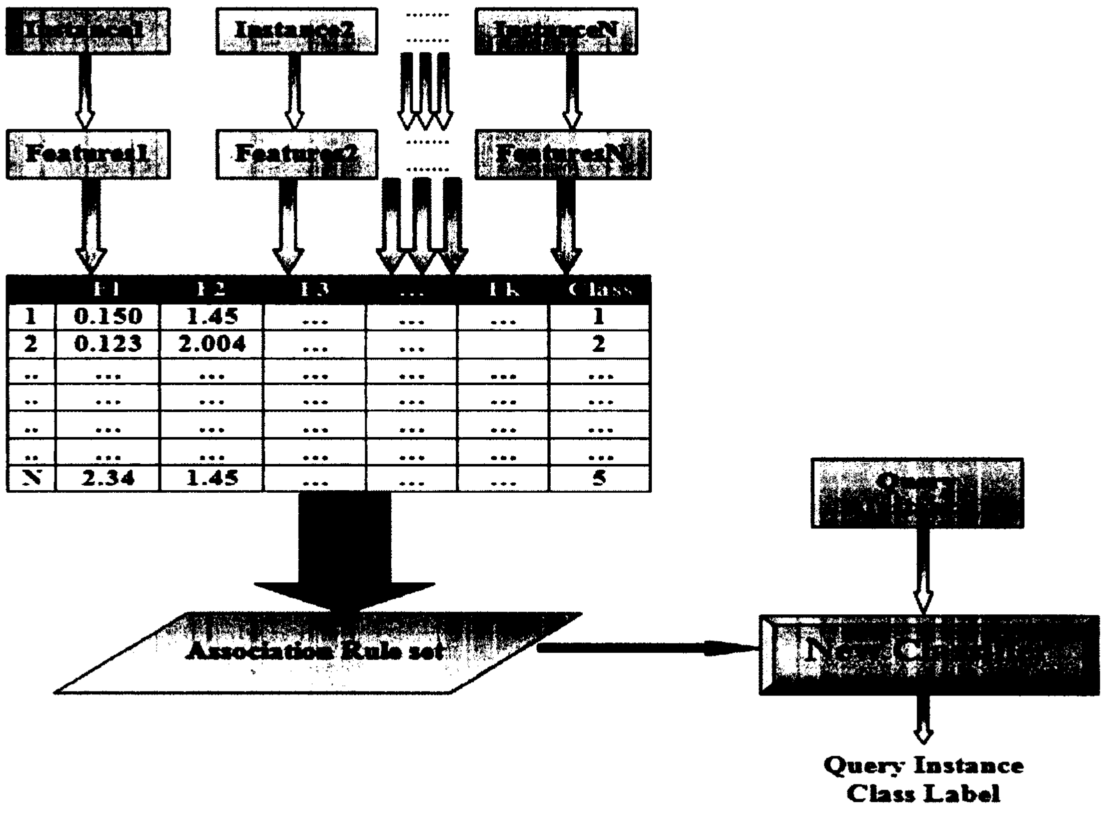


Figure 1.3 Associative Classification Mechanism

Despite the efforts of numerous researchers, some questions remain. For instance, researchers have yet to determine the best type of data representation to use for mining, what support and confidence to use, whether rule weighting or rule pruning should be used, whether class-based rules should be used, and how to accurately perform the weighting when dealing with weighted rules. We address these issues by developing a new weighted rule-based classification algorithm. We used the new rule-based classification algorithm with datasets from different domains to test its robustness and scalability.

1.3.3 Temporal Data Mining

Most frequent-itemset mining approaches find patterns that occur at the same time (e.g. when a customer buys bread and butter, he/she will also buy jelly). As explained in Section 1.3.1, these types of rules are called *intra-transaction* association rules. However, in reality, not all the data is static. For example, meteorological and stock market data are dynamic, and are also known as temporal data, (i.e. data that changes with time). Regular frequent itemset mining algorithms cannot work in these scenarios as the patterns of importance here have a time difference. One example of such a pattern is if Microsoft stock goes up on the first day, and Intel stock goes up on the second day, then the Apple stock will go down on the third day. These types of patterns, which have time information associated with them, are temporal patterns. Notice that the pattern is still a combination of frequent items from the database, but now the items belong to different transactions. Throughout the rest of the dissertation, we will call these types of patterns "*inter-transaction patterns*" or "*inter-transaction association rules*", as the scope of the pattern covers multiple transactions but each item only lasts for a single transaction. In this work, we present a novel Transaction ID based windowless inter-transaction rule-mining algorithm. The proposed algorithm not only addresses most of the issues (discussed later in detail) that plague current inter-transaction rule mining algorithms, but also provides the location information of every generated rule. Such location information is required to confirm the accuracy of the rules.

1.4 Dissertation Outline

In Chapter 2, we provide background information and related literature for associative classification, inter-transaction association rule mining, and temporal pattern mining. In Chapter 3, we show how association rules can replace raw features as higher-order and aid in classification. In Chapter 4, we explain our proposed weighted rule based associative classification mechanism. In Chapter 5, we provide details about our proposed inter-transaction rule-mining algorithm. Finally, we provide our conclusions and future directions in Chapter 6.

CHAPTER 2

BACKGROUND AND RELATED RESEARCH

2.1 Rule Based Learning

Rule based learning methods generally use inductive or derivative if – then type decision rules to build learning models. These decision-based models are then used for predicting the class labels of test instances. Rule-based classification is not new to the scientific community. One of the earliest decision tree-based rule mining algorithms, ID3 [20], was developed in 1986 by J. R. Quinlan. ID3 was the precursor to early rule-based classifiers C4.5 [5] and C5.0/See5 [21]. Other rule-based classifiers were FOIL [22] and RIPPER [23]. Interest in the field of rule based classification has steadily grown over the years.

Classical rule based algorithms like ID3 and C4.5 build a decision tree to predict the label of a target instance based on several input variables. Every node of the tree consists of questions regarding one of the variables of the instance, and leaves consist of the classification labels. ID3 and C4.5 use information gain to decide which variable to use as test variable for a node. In order to classify a new instance, one starts at the root node and follows all the test nodes one by one depending upon the answers. The root node reached by finishing all the tests gives the predicted class labels for test instance. While rule based methods have some advantages like simplicity to interpret and understand and ability to handle both numerical and categorical data, they have some disadvantages as well such as high memory overhead resulting from data storage and overfitting resulting from complex decision tree models. These disadvantages limit their use for many applications.

2.2 Associative Classification

Associative classification is special type of rule based learning which uses association rules for building classification models. The earliest associative classifier, developed by Liu et al. [6], is known as CBA (classification based on associations). CBA was designed to integrate association rule mining with classification. Liu et al. employed the classical apriori algorithm for rule generation and concentrated on generating only a special subset of all possible association rules, which they called class association rules (CARs). CAR is such a rule that has feature values on the LHS of the rule and only a class label on the RHS of the rule: $f_1, f_2, .. f_k \Rightarrow c_1$. They divided the approach into three steps: (1) data discretization, (2) CAR generation, and (3) classifier building using CAR's.

Apriori generates all the association rules that satisfy the user defined minimum support and confidence. Given m rules, in order to build the best classifier having only a subset of these rules and minimum error, one would require an evaluation of all possible subsets (2^m). Such an evaluation may be computationally expensive. Hence, Liu et al. developed a heuristic classifier, which ranked the rules according to their support and confidence and by how well each rule classified the instances in the training data. Only those rules that classified at least one training instance correctly were kept. For classification, a test instance was matched sequentially with each rule's LHS, starting from the highest ranked rule, and the instance was classified to the class of the first rule it matched. If a test instance did not match any rule, then it was classified to a default class (the majority class of the training data after rule ranking). The results showed that CBA outperformed the most common rule-based algorithm at that time, C4.5.

Dong et al. introduced CAEP (Classification by Aggregating Emerging Patterns) to use emerging patterns (EPs), itemsets which contain support that changes from class to class, for classification [24]. They divided the algorithm into two parts: finding emerging patterns and building a classifier using pattern aggregation and scoring. CAEP uses the border differential procedure to generate emerging patterns for a class *C*, satisfying some user defined minimum support and growth rate thresholds [25]. The growth rate thresholds are a ratio of instances in class *C* having the pattern to instances of other classes with the pattern. For classifying an instance, the differentiating power of all EPs of *C* that occur in the instance are aggregated and then normalized with some base score from training instances to generate an aggregated differentiating score for each class. The instance is classified to a class with the largest normalized base score. Because the scores are normalized, CAEP can handle imbalances in the training data very well. Experimental results showed that CAEP outperformed both CBA and C4.5 in most test cases.

Li et al. proposed CMAR (classification based on multiple association rules) [8]. As the name suggests, CMAR is unlike CBA in that it requires multiple rules for classification, thereby alleviating the classification bias caused by single rule classification. Another important difference between CMAR and CBA is that CMAR uses an FP-growth property to generate class distribution-based associated FP-Tree instead of apriori to mine rules [26]. CMAR uses a compact prefix-tree-based novel data structure called a CR-Tree to store and retrieve association rules for classification. Rule pruning was performed based on confidence, correlation, and database coverage. A new technique called weighted χ^2 was used to find strong association rules by considering both conditional support and class distribution. For classifying a new test instance, first, all the rules matching for this test instance were retrieved from the CR-Tree, and then the class label of these rules was analyzed. If there was no conflict in the class label, (i.e. all the rules had the same class label), the test instance was assigned the class label. However, if one or more rule had differing class labels, the rules were grouped according to their class labels, the effects of each group were compared using weighted χ^2, and the instance was classified to the strongest group. Experiments using 26 UCI machine learning datasets showed that CMAR outperformed both CBA and C4.5 in average accuracy, efficiency, and scalability.

Yin et al. proposed CPAR (classification based on predictive association rules). CPAR adopts a greedy approach to generate a smaller set of rules and the combined advantages of both associative classifiers and traditional rule-based classifiers [27]. CPAR uses the rule generation ideas from FOIL [22]. In FOIL, the information gained by adding a literal to the rule is measured using foil gain. For extracting predictive association rules, CPAR uses a PRM (predictive rule-mining) algorithm, which is a modified version of FOIL. For multi-class classification rule pruning in FOIL, an accurately predicted training instance is removed from the dataset, but in CPAR only its weight is reduced. Due to this methodological difference, CPAR usually generates more predictive rules than FOIL. In the experiment, Yin et al. avoided rule redundancy by considering a set of already-generated rules. They then used CPAR to generate more rules since the classification method considers all the close-to-best literals instead of using only the best literals, as FOIL does. Once all the rules were formed, their prediction power was evaluated using the *Laplace expected error estimate* [28]. For classification of a test instance, all the matching rules for the test instance and the best *k*-rules for each class were found, the average expected accuracy of each class was calculated, and the instance was classified to the class with the highest accuracy. The experiments were performed on 26 UCI machine learning datasets with the best five-rules, and CPAR outperformed CBA, C4.5, and CMAR in most cases.

2.3 Inter-Transaction Pattern Mining

Since temporal data varies with time, it has time information attached to it. Mining such time varying data is a highly complex task due to the added dimensionality and complexity of time. Most existing classical pattern mining algorithms are inadequate to handle this type of data as they only deal with the co-occurrence of values at a time. However, mining time varying data results in a time dependent causal pattern for which one event triggers the response to another. For example, if the temperature drops in hour-one and the humidity increases in hour-two, then there will be rain in hour-three. In terms of data mining, these types of patterns are called "inter-transaction patterns" or "inter-transaction association rules". Significant research effort has been devoted to this task in the past decade. There are two major types of data formats, horizontal data-format or vertical data-format. The market basket data is usually captured and saved in the horizontal data format in which each row has a unique identifier called a "Transaction ID" (TID), and the corresponding values in the transaction represent the events that occur in this transaction. The vertical database, on the other hand, is an event-based representation for which each row represents an event having its own IDList that contains the TIDs of all the transactions where an event happens. Figure 2.1 shows the example of horizontal data format and vertical data format.

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

(a)

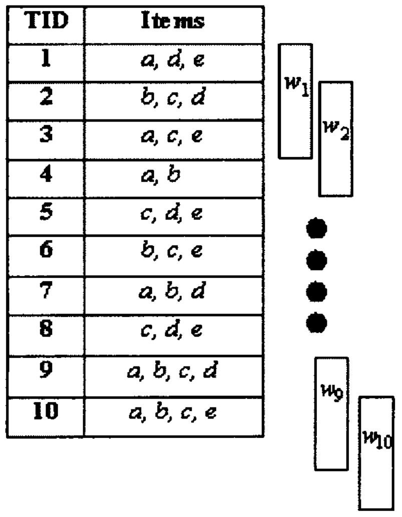
Item	IDList
a	1,3,4,5,7,9,10
b	2,4,6,7,9,10
c	2,3,5,6,8,9,10
d	1,2,5,7,8,9
e	1,3,5,6,8,10

(b)

Figure 2.1 Data Formats: (a) Horizontal Data Format (b) Vertical Data Format

Most of the existing inter-transaction association rule mining algorithms can be divided based on these two data types. In this section first, we explain the horizontal data-based algorithms (E-Apriori, EH-Apriori, FITI, EFP-Tree, and MMIT). Then, we discuss the vertical data-based algorithms (PROWL, ITPMine).

The concept of inter-transaction association rules was first introduced by Lu et al. in [29]. Two algorithms *E-Apriori* and *EH-Apriori* were proposed to extract inter-transaction association rules from stock market data. Both of these techniques were based on the apriori algorithm. As a preprocessing or data preparation step, each item in the dataset was appended with a TID. Then, a sliding window mechanism was used to transform a set of simple transactions into mega-transactions. The items in mega-transactions, called mega-items or extended items, address connected items from the original database. Figure 2.2 explains this mega-transaction creation scheme.



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

(a)

TID	Items
1	*a*(0),*d*(0),*e*(0),*b*(1),*c*(1),*d*(1),*a*(2),*c*(2),*e*(2)
2	*b*(0),*c*(0),*d*(0),*a*(1),*c*(1),*e*(1), *a*(2),*b*(2),*c*(2)
3	*a*(0),*c*(0),*e*(0),*a*(1),*b*(1),*c*(1),*c*(2),*d*(2),*e*(2)
4	*a*(0),*b*(0),*c*(0),*c*(1),*d*(1),*e*(1),*b*(2),*c*(2),*e*(2)
5	*c*(0),*d*(0),*e*(0),*b*(1),*c*(1),*e*(1),*a*(2),*b*(2),*d*(2)
6	*b*(0),*c*(0),*e*(0),*a*(1),*b*(1),*d*(1),*c*(2),*d*(2),*e*(2)
7	*a*(0),*b*(0),*d*(0),*c*(1),*d*(1),*e*(1),*a*(2),*b*(2),*c*(2), *d*(2)
8	*c*(0),*d*(0),*e*(0),*a*(1),*b*(1),*c*(1), *d*(1),*a*(2),*b*(2),*c*(2),*e*(2)
9	*a*(0),*b*(0),*c*(0),*d*(0),*a*(1),*b*(1),*c*(1),*e*(1)
10	*a*(0),*b*(0),*c*(0),*e*(0)

(b)

Figure 2.2 (a) Original Database with Sliding Windows (w=3), (b) Mega-Transaction Database after the Sliding Window Operation

In this way, an inter-transaction rule-mining problem was reduced to an intra-transaction rule-mining problem. Then, a regular apriori algorithm was used to generate association rules between these extended items. Experiments were performed on the stock data from the Singapore Stock Exchange (SES), and the technique captured fluctuation patterns between stocks of various companies correctly.

FITI (First Intra Then Inter) builds upon the mega-transaction format put forward by the EH-Apriori algorithm [30]. As the name suggests, the algorithm first makes frequent intra-transaction itemsets and then generates frequent inter-transaction itemsets from these intra-transaction items. The technique is divided into three phases: Phases I, II, and III. In Phase I, frequent inter-transaction itemsets are mined and then stored in a unique data structure. In Phase II, the original database is transformed into a new one using the data structure from Phase I. Finally, in Phase III, inter-transaction itemsets are generated from the transformed database. The motivation for using this approach is that any frequent inter-transaction itemset should contain only the frequent intra-transaction itemsets. To meet this goal, the algorithm first generates frequent intra-transaction itemsets using the apriori algorithm. Then, each of these itemsets is given a unique number called an ID, and these itemsets are stored in a new data structure called FILT (Frequent-Itemset Linked Table). The FILT data structure consists of a hash table with nodes connected by four types of links: lookup links, generator and extension links, subset links, and descendent links. After FILT formation, the original database is transformed into a number of new ID encoded databases known as FIT (Frequent-Itemset Table). Each FIT table only represents the ID encoded frequent itemsets of one level. Hence, the number of FIT tables $\{F_1, F_2, \ldots, F_{\max_k}\}$ is equal to the maximum size of the intra-transaction itemset discovered in Phase I. After this transformation, *intra-transaction join* and *inter-transaction joins* are used to combine the ID encoded itemsets for generating candidate inter-transaction itemsets. The support of these candidates is calculated using a hash-tree-based mechanism and making multiple passes over the database. Rule generation stops when no more candidates can be generated. Experimental results over synthetic data and stock market data show that FITI outperforms EH-Apriori in CPU time consumption.

Luhr et al. proposed EFP-Tree, which used the FP-tree-based approach for rule mining [31]. As in EH-Apriori, they used the sliding window concept to transform the dataset into the mega-transaction format as a data preparation step by appending the time stamp/transaction ID with the items. Then, they used a modification of the FP-growth property, called EFP-Growth (extended FP-growth), for itemset and rule generation. They performed three passes over the data. The frequency of single items was calculated in the first pass by scanning the extended mega-transaction itemset database once. In the second pass, Luhr et al. built the *intra-transaction* FP-tree and calculated the conditional frequencies of inter-transaction items. Finally, in the third pass, they built the inter-transaction EFP-Tree. Frequent itemsets and inter-transaction rules were generated from this EFP-Tree. Experimental results on both synthetic and real world data showed that EFP outperformed FITI in terms of memory usage and execution time, in most cases.

Wang et al. proposed MMIT (matrix mining inter-transaction), which was based on the concept of co-occurrence matrix [32]. Their technique is divided into five phases. In Phase I, they scanned the original transaction database using the sliding window format to generate extended items. These extended items were frequent-one level inter-transaction itemsets, which were sorted. Only 1-frequent inter-transaction itemsets were selected. In Phase II, the authors created a co-occurrence matrix of 1-frequent itemsets [33]. In Phase III, the transaction database was scanned once again with the sliding window format, and the frequent inter-transaction itemset information was stored in the co-occurrence matrix. Inter-transaction itemset mining for level k $(k \geq 2)$ was generated in Phase IV. Finally, in Phase V, strong inter-transaction association rules were generated from the frequent inter-transaction itemsets generated in Phase IV. Experimental results on synthetic data showed that their technique outperformed FITI in terms on execution time.

To the best of our knowledge, PROWL (PROjected Window List) was the first algorithm which deviated from this norm and used the vertical data format (Figure 2.2

(b)) [34]. For PROWL, the original database is scanned once, and each item is then associated with the IDs of the transactions where the item occurs, thereby making an ID-list for each item in the database. The length of an ID-list gives the support of the item. Frequent 1-itemsets are then found using the support constraints, and their ID lists are kept for higher-level itemset generation. For example, let there be three frequent 1-items $\{e_1,e_2,e_3\}$ with IDLists of $\{IDlist(e_1),IDlist(e_2),IDlist(e_3)\}$. For each frequent-1 item, a Projected Window List (PWL), $\{PWL(e_1)\ PWL(e_2),PWL(e_3)\}$ is created by adding one to the existing IDs of the IDList. Then, candidate 2-itemsets of the form e_1 followed by e_2 one time point later $\{e_1(0)e_2(1)\}$ are created by intersecting $PWL(e_1)$ with $IDlist(e_2)$. The intersecting IDs form the IDList of 2-itemset $\{e_1(0)e_2(1)\}$, represented by $IDlist\{e_1,e_2\}$. Using the same formula, we can make $IDlist\{e_1,e_3\}, IDlist\{e_2,e_3\}$, $IDlist\{e_2,e_1\}, IDlist\{e_3,e_1\}$, and $IDlist\{e_3,e_2\}$. Further, $PWL(e_1)$ is also the IDlist of $\{e_1,*\}$ $(IDlist\{e_1,*\})$, where * indicates that the item for this time stamp is undefined. The "*" notation is used to allow a mismatch at a position. It means that the user does not care about the item present at this position. This $IDlist\{e_1,*\}$ is then used to create itemsets where the second item following the first happens two time points later instead of one, for e.g. e_1 followed by e_2 two time points later $(e_1(0),e_2(2))$, or e_1 followed by e_3 two time points later $(e_1(0),e_3(2))$. The IDLists are represented as $IDlist\{e_1,*,e_2\}$ and $IDlist\{e_1,*,e_3\}$. Similarly, the itemsets of the form $(e_1(0),e_2(1),e_3(3))$ are then formed by generating $PWL\{e_1,e_2\}$ and intersecting it with $IDlist\{e_3\}$. In the same way, the depth first search (DFS) mechanism of PWL generation is repeated until the maxspan level has been reached for all events in the database. Unlike the vertical data format-based techniques, PROWL does not generate all frequent-k itemsets at the same time, since the itemsets generated by a maximum of m time points will only be generated by PWLs generated at maxspan level m. For example, frequent-2 itemsets separated by three time points $\{(e_1(0)e_2(3)),(e_1(0)e_3(3))\,\text{etc.}\}$ will be generated by the PWLs at maxspan level three only. For more information, we advise the user to refer to [34].

ITPmine, which is also a DFS-based mechanism, is the only other algorithm, which we are aware of that uses the vertical data format to find inter-transaction association rules [35]. ITPMine scans the entire database once using the sliding window mechanism. In one pass, it adds the IDs (called dats for ITPmine) for all the 1-itemsets/ events. During this first scan of the database, ITPmine also uses the EH-Apriori type hash-based approach to hash all the possible 2-itemsets in the sliding window into a hash tree. During this process, it hashes the candidate 2-itemsets along with the support counting for 1-itemsets during one sliding window scan over the database. Once the ID lists (called Dat lists for ITPMine) of the 1-events are created, ITPmine finds the support of 1-itemsets by counting the length of each Dat list. Frequent-1 itemsets, (i.e. the itemsets with a length/support higher than the user defined minimum support are kept), and others are deleted. Candidate 2-itemsets are generated by joining each frequent 1-itemset with another and adding the time stamp. The support of these candidate itemsets is found by hashing the itemsets and checking the support at the hash address. If the support is higher than a user defined minimum support, then the itemset is added to the new itemsets and their corresponding dats are found in the dat list. The new itemsets are divided into joinable and extendable groups. Then, the DFS mechanism is used for each frequent 2-itemset to generate all possible k-itemsets by joining each frequent 2-itemset with itemsets from its joinable group and using the pruning mechanisms. The procedure is repeated for each frequent 2-itemset. For further reading, please refer [35].

2.4 Major Contribution of the Dissertation

In this dissertation, we concentrate on two components of data mining: associative learning for classification and temporal pattern mining. Our goals are to:

- Show the discriminatory power of associative relationships for supervised learning,
- Develop an intra-class/inter-class weighted rule-based classification system for the classification of multidimensional datasets, and
- Develop a framework for capturing point-based and event-based frequent temporal patterns from multivariate time series data.

The overarching aim of the first part of the dissertation is to build a new associative classifier that uses intra-class and inter-class-based weighted rules for classification. We deviate from the common row/instance-based representation for associative classifiers and introduce an alternate data representation whereby each instance is treated separately for association rule mining. Since we have introduced a new representation, we first validate that such a representation can provide more information than the commonly used row/instance representation. Therefore, we present a novel data transformation schema that transforms the association rules into higher order features for classification in Chapter 3. The detailed experiments in this chapter validate our hypothesis that the new representation scheme provides better classification results.

The insights obtained from results in Chapter 3 motivated us to further explore this new representation and determine how the rules extracted can be weighted according to inter-class and intra-class importance. Consequently, we present a novel associative classification mechanism based on this data representation in Chapter 4. The proposed algorithm uses the intra-class and inter-class variations to weight the association rules. These weighted rules, along with the rule cardinality, form the basis of the new classifier, which we named WAR-BC (Weighted Association Rule Based Classifier).

Chapter 5 contains the second part of this dissertation: (frequent pattern mining for time varying data). Most existing works transform the time varying data using a sliding window approach to a static format and then use existing frequent pattern mining algorithms (with small modifications) for static data to extract underlying patterns. These types of techniques have several limitations such as the requirement of new dataset generation every time a sliding window size is changed, increased dimensionality, and lack of location information to check the accuracy of rules. More detail is provided in Chapter 5. We show that the underlying problem is not the existing work, but the data format with which the previous researchers have worked. We present a new, windowless item index-based frequent pattern mining algorithm which not only extracts the frequent patterns in less time than existing work, but which also provides the location information for each rule generated from the database. We use mathematical induction to prove that the proposed algorithm extracts all the underlying patterns from the dataset.

2.5 Conclusion

This dissertation work is a compilation of three works that have already been published and one work that is currently in progress:

1. S. Dua, H. Singh, and H. W. Thompson, "Associative Classification of Mammograms using Weighted Rules based Classification," *Elsevier Expert Systems with Applications,* volume 36, Issue 5, 2009.
2. H. Singh, S. Dua, and H. W. Thompson, "Weighted Rule based Algorithmic Tool for Image Classification," *NSF EPScor Research Infrastructure Improvement Annual Meeting*, 11 May 2009.
3. S. Dua and H. Singh, "Biomedical Image Classification using Association Rule Mining," *24th Annual Houston Conference on Biomedical Engineering Research*, Houston Society for Engineering in Medicine and Biology, 2007.
4. H. Singh and S. Dua, "DMITAR: Difference Matrix based Inter-Transaction Association Rule Mining," *IEEE Transactions on Knowledge and Data Engineering* (in Progress).

Chapters 3 and 4 are based on the first three publications. Chapter 5 is based on the fourth publication.

CHAPTER 3

ASSOCIATIVE RELATIONS AS HIGHER ORDER FEATURES

While a great deal of effort has been put into building new associative classifiers which use class constrained rules, little has been done in terms of using regular association rules as higher order features for classification with existing classifiers. We believe that associative relationships existing in data are discriminatory and provide valuable information for learning. The first part of our research focuses on justifying our claim that association rules can be treated as higher order features and employing them as classifier inputs instead of raw features will yield better results. In Chapter 3, we attempt to answer two questions:

1. How can non-class specific association rules be uniquely represented as features?
2. Do these higher order features have better discriminatory power than raw features?

Most frequent pattern-based algorithms represent each instance as a row, and the combination of these individual instances, stacked one beneath the other, are used as market basket data to extract class-constrained association rules. In market basket data, one instance equals to one row representation, which is the main limitation for non-class association rules. Therefore, in order to extract such non-class association rules, it becomes imperative that the data representation be such that each instance is represented as an individual market basket database. Hence, the dataset choice becomes an important factor for such a technique.

In lieu of the above factors, we have used image and image datasets for our analysis. The number of images and image datasets are increasing rapidly due to the ease and availability of such datasets over internet. Because of the increased size, it has become difficult for humans to organize, store, and manage these enormous datasets efficiently. Hence, automated image classification techniques are needed to solve this issue. The advent of computational technology in the field of medicine has seen a meteoric rise in the amount of medical image data collected on a daily basis. Brain scans, MRI's, diabetic retinopathy images, CAT scans, and mammograms are some of the examples of medical images. Mammograms are widely used by doctors to detect and evaluate breast cancer. Since the etiology of cancer is unknown, it cannot be prevented. However, regular mammogram screenings can help reduce the mortality rate of breast cancer through early detection, which leads to earlier treatment. In most cases, more than one radiologist analyzes a mammogram before a diagnosis or treatment decision is made, and recent studies have shown that error in diagnosis can be reduced by almost 10-15% by multiple readings [36].

Moreover, these multiple readings further cause a significant bottleneck for healthcare. First, there are significant delays since many healthcare facilities do not have enough radiologists. Second, most healthcare facilities avoid employing multiple radiologists since it is cost prohibitive to do so. Finally, many insurance companies do not pay for multiple radiologists. Therefore, an automated system which could provide a valuable diagnosis to the physician and replace the second radiologist is highly sought.

We present an association-rule-based automated classification system in this dissertation. In Chapter 4, we present a new classifier built upon these association rules. In this chapter, we focus on the usability of association rules as higher order features. We propose a simple rule transformation schema, which we use to transform the association rules extracted from data into a new feature space. This feature space is then used as input for classification.

3.1 Problem Definition

Let D be a training database, $I = \{I_1, I_2, ..., I_M\}$ be a set of images, C be a class attribute $C = \{c_1, c_2, ..., c_N\}$ and $F = \{f_1, f_2, ..., f_k\}$ be a set of features. Let $f_i = \{f_{i1}, f_{i2}, .., f_{ik}\}$ be set of features extracted for each image $I_i \mid I_i \in I, \forall 1 \leq i \leq M$. Then an image I_i in the dataset can be represented as $\{f_i, c_j\} \mid i \leq M, j \leq N$, where $c_j \in C$ is the class label of image *i*. Assuming R_i is the set of rules extracted for image *i*, the rule representation of an image I_i is $\{R_i, c_j\}$. Then, higher order based classification is used to build mapping to transform the low order rule representation R_i into a higher order feature representation $R_i^{'}$ using the transformation $T : \{R_i, c_j\} \rightarrow \{R_i^{'}, c_j\}$. A multi-class classifier can then be used for classification.

3.2 Dataset

Data has to be in the market basket (transaction) format for association rule extraction. With this information in mind, we selected the digital mammogram database as our training and test set.

Mammogram Dataset: We used the MIAS (Mammogram Image Analysis Society) dataset for our analysis [37]. The dataset is commonly used for medical image classification paradigms and contains 322 mammogram images. The images are divided into three major disease categories: normal, benign, and malignant. There are 208 images in class normal, 63 in class benign, and 51 in class malignant. Each image in the dataset is of 1024x1024 resolution. The original dataset was digitized at 50-micron pixel edge, but was then reduced to 200 micron pixel edge. It was clipped and padded with 0's (black background) to ensure that the size of all images were the same. The position of the nipple on the breast could be either on the left or on the right, which makes it even harder to classify this dataset.

3.3 Proposed Methodology

Our methodology can be divided into five parts:

I. Data preprocessing,

II. Feature extraction,

III. Data preparation,

IV. Association rule mining, and

V. Rule transformation.

Once the rules are transformed into a feature format, we use two classifiers: FKNN and SVM for classification comparison.

3.3.1 Data Preprocessing

Medical images present in the mammogram database are noisy, and some labels that are present are not useful. These labels need to be removed, so they do not confuse the classifier with redundant and non-informative rules. Further, the images are large

(1024x1024), and most of the area consists of a black background that provides no useful information and can be omitted. Here, we present a method, based on the connected component theory, to remove the labels from the image and crop the big image into a smaller one. Figure 3.1 shows the complete preprocessing procedure.

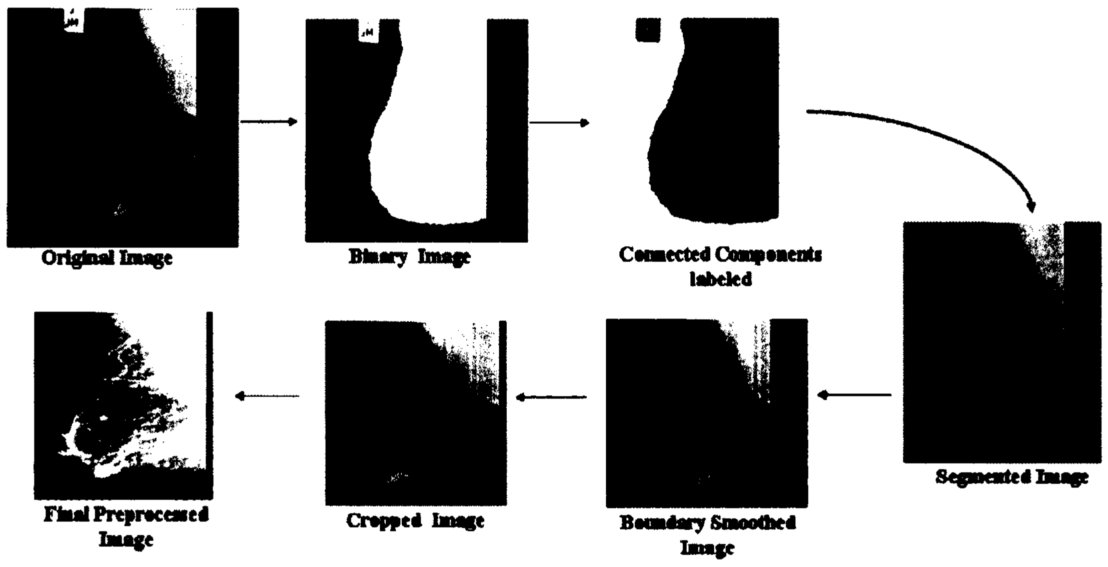


Figure 3.1 Data Preprocessing for Label and Noise Removal

First, every image is changed into its binary form, and then connected components are formed from this format, providing us with all the connected components present in the image. Presumably, the connected component with the largest area is the breast part of the mammogram. This area is extracted from the original image. Once the area has been extracted, the next step is to smooth the breast boundaries. The segmented image is scanned line-by-line, and the starting and ending points of each segment on the line are taken. The same points are taken on the unsegmented image. The image is read, left from the starting point and right from the ending point, and two new boundary points (cut points) are marked on the unsegmented image, when five consecutive pixels are below threshold value β. After careful consideration of the images, we define the β value as 20. We make a new cut point when five consecutive pixels have an intensity of less than 20 on the left and right sides of the starting and ending points. Every pixel to the left of the first cut point and to the right of the second cut point on this line is set to zero. The same procedure is applied to each following line of the segmented image, and a new boundary is formed for it. Finally, once the borders have been smoothed, the image is cropped to within 10 pixels of the border tip in both directions. The data preprocessing algorithm is explained in Figure 3.2.

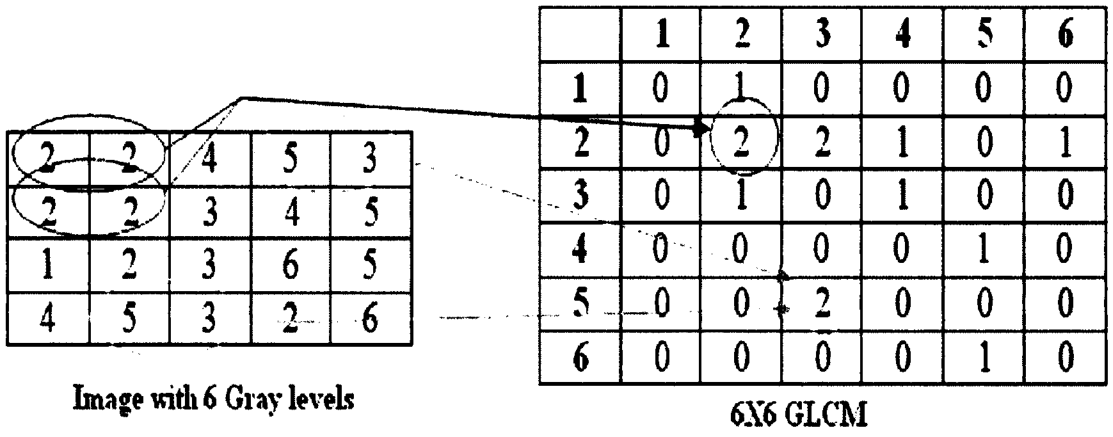
```
Algorithm  Border Smoothing and Image Cropping (BSI-Crop)

Input  Segmented part of the mammogram, starting pixel and ending pixel points on
each line of the segment, original image, number of rows N, intensity threshold $\alpha = 10$,
border threshold $\beta = 20$

Output  Cropped segmented and border smoothened image

Method
(1) For every line (row) j in the segmented image $\forall j < N$
(2)       Scan the line
(3)       Start_pixel ← starting pixel position of the segment
(4)       End_pixel ← ending pixel position of the segment
(5)       Scan the same line on original image
(6)       Read left from Start_pixel
(7)       New_Start_point(j) ← pixel position when five consecutive pixels have intensity
(8)                                   $< \alpha$
(9)       Read right from End_pixel
(10)      New_End_point(j) ← pixel position when five consecutive pixels have intensity
(11)                                  $< \alpha$
(12)      Change every pixel value to the left of New_Start_point(j) and to the right of
(13)      New_End_point(j) to zero
(14) End For
(15) Tip_Border_left ← minimum (New_Start_point (1-N))
(16) Tip_Border_Right ← maximum (New_End_point (1-N))
(17) Left_border ← Tip_Border_left - $\beta$
(18) Right_border ← Tip_Border_right + $\beta$
(19) New_cropped_segmented_image ← crop the border smoothened image into
(20)                                   Left_border and Right_border limits
```

Figure 3.2 Data Preprocessing Algorithm

3.3.2 Feature Extraction

We use grid-based image segmentation to extract low-level texture features from the images. Initially, each image is divided into *n x n* non-overlapping segments, and then low-level features are extracted from each of these segments. The size of the block, (i.e. n x n), is chosen so that approximately 1000 – 1500 sub-blocks of the image are obtained. Our aim is to segment the image into smaller blocks and capture local relationships among image features. Once the image has been segmented, we extract eight texture features from each segment. Therefore, each segment represents a feature vector of length eight.

In order to differentiate one feature vector from another, each vector is given a unique ID. In our case, this ID is the number of the segments from which the features have been extracted, e.g. *TID 1 (f1, f2, f3……..f8) and TID 2 (f1, f2, f3……..f8)*. The Haralick texture features first introduced by Haralick et. al. used the co-occurrence matrix formed from the intensity value of pixels in the image to calculate 14 texture features for an image [38]. Here, we use eight of those features. Texture is an important aspect of an image and has several definitions, namely the frequency of change and arrangement of tones of color in an image and the statistical distribution of spatial relationships between gray-level properties of pixels in an image. When distribution of texture changes slightly with distance in an image, the texture is called *coarse* texture, and when the distribution changes rapidly with distance, it is called a *fine* texture. Information regarding the change in texture can be captured in a co-occurrence matrix. When the image is a gray-level image, it is called a *gray-level co-occurrence matrix* (GLCM). This GLCM matrix was used by Haralick in to find statistical texture features. Since these features are based on the intensity profile of an image, and since mammogram images are also intensity-based images, GLCM is a perfect fit for our problem. Figure 3.3 shows the calculation of a 6x6 co-occurrence matrix.



2	2	4	5	3
2	2	3	4	5
1	2	3	6	5
4	5	3	2	6

	1	2	3	4	5	6
1	0	1	0	0	0	0
2	0	2	2	1	0	1
3	0	1	0	1	0	0
4	0	0	0	0	1	0
5	0	0	2	0	0	0
6	0	0	0	0	1	0

Figure 3.3 Gray Level Co-Occurrence Matrix for an Image with 6 Gray Level Values

The eight features that we use in our method are:

1) Energy $\sum_{i=0}^{n}\sum_{j=0}^{n}\{p(i,j)\}^2$

2) Contrast $\sum_{i=0}^{n}\sum_{j=0}^{n}(i-j)^2 p(i,j)$

3) Local Homogeneity $\sum_{i=0}^{n}\sum_{j=0}^{n}\frac{p(i,j)}{1+(i-j)^2}$

4) Correlation $\sum_{i=0}^{n}\sum_{j=0}^{n}((ij)p(i,j)-\mu_x\mu_y)/\sigma_x\sigma_y$

5) Entropy $-\sum_{i=0}^{n}\sum_{j=0}^{n}p(i,j)\log p(i,j)$

6) Cluster Shade $\sum_{i=0}^{n}\sum_{j=0}^{n}(i-M_x+j-M_y)^3 p(i,j)$

where $M_x=\sum_{i=0}^{n}\sum_{j=0}^{n}ip(i,j)$ and $M_y=\sum_{i=0}^{n}\sum_{j=0}^{n}jp(i,j)$

7) Information Measure of Correlation HXY-HXY1/ max{HX,HY}

where HXY= Entropy $P_x = \sum_{j=0}^{n} p(i,j)$, $P_y = \sum_{i=0}^{n} p(i,j)$

$$HX = -\sum_{i=0}^{n} P_x(i) \log P_x(i) \quad , \quad HY = -\sum_{j=0}^{n} P_y(j) \log P_y(j),$$

$HXY1 = -\sum_{i=0}^{n}\sum_{j=0}^{n} P(i,j) \log\{P_x(i) P_y(j)\}$, and

8) Maximum Probability $\max_{i,j} P(i,j)$

Here, *P(i,j)* is an entry in the co-occurrence/spatial dependence matrix with row number *i* and column number *j*. μ_x and μ_y are the means for rows and columns, respectively, and σ_x, and σ_y are the corresponding standard deviation. Four possible angular nearest-neighbor distances can be used to calculate the co-occurrence matrix. These angles are $0°, 45°, 90°, 135°$. Figure 3.4 represents these four directions.

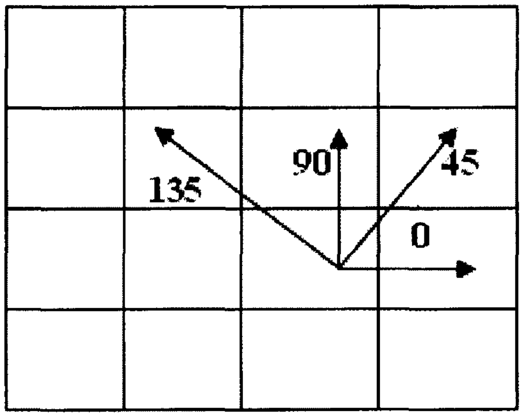


Figure 3.4 Different Directions of Co-Occurrence Matrix

The values of all the features are calculated in these four directions, and the average values are represented as the value of a feature. We used the 1-nearest neighbor distance approach to calculate the co-occurrence matrix. The algorithm for segmentation and feature extraction is presented in Figure 3.5.

Algorithm

SEgmentation and **F**eature **Ex**traction (**SE-FEX**) divides the image into different non-overlapping segments, extracts features from these segments and arranges them in a Transactional Database

Input Preprocessed images $I_1, I_2, \ldots\ldots, I_N$. Segment size **nxn**, set of discrete values a feature can take $\{v_1, v_2, \ldots\ldots, v_k\}$, number of Haralick texture features H

Output Images $I_1, I_2, \ldots\ldots, I_N$ in transaction database format where each transaction is a vector representing features extracted from each segment

Method:

```
(1) For every Image $I_j(1\ldots r, 1\ldots c)$ $\forall j \in (1..N)$, $r$ shows number of rows and $C$ number of
(2)     columns
(3)     number of segments ($N_s$) ← (r*c)/(n*n)
(4)     For every segment $S_l$ $\forall l \in (1..N_s)$
(5)         $S_l(I_j[F_h]) \leftarrow v_t \forall t \le k, h \le H, j \le N, l \le N_s$ extract features from the
(6)         segment $S_l$
(7)     Endfor
(8) Endfor
```

Figure 3.5 Algorithm for Segmentation and Feature Extraction

3.3.3 Data Preparation

Once feature extraction has been performed, we need to modify the data so that it can be used for association rule mining. During the feature extraction phase, some features give "not a number," or (NaN), values because of noise in the images. This problem is observed often in medical images, as, despite segmentation, most of the background is black in these images. As a result, some statistical formulas provide the NaN value. We observe that the corresponding features for the same segment do not have a discriminatory value (most of them are 0 and 1). As a result, we have decided to remove those transactions or segments, as they do not provide us any information and will not have any part in the classification procedure. Once data cleaning has been performed, the next step is to normalize the data.

Since different features cover different values and the values differ drastically from each other, it is better to normalize these values to a comparable base. We use the *z-score* normalization procedure to address this issue. In z-*score* normalization, the values of attribute A are normalized based on the mean and standard deviation of A. *Z-score* normalization maps a value v of A to v' using the formula:

$$v' = \frac{v - \overline{A}}{\sigma_A}, \qquad \text{(Equation 3.1)}$$

where $\overline{A}$ is the mean and σ_A is the standard deviation of the attribute. To provide an accurate representation of the mean and standard deviation of one attribute (feature), we combine the training data for each attribute to make a combined matrix. Then, we find the mean and standard deviation from this combined matrix for each attribute. This mean and standard deviation is used to normalize the data. Figure 3.6 shows the results of data preparation mechanism.

Seg ID	F1	F2	..	..	..	Fm
S1	.345	2.44	....	....	39.9	56.7
S2	1	0	Nan	0	NaN	1
S3	1	0	Nan	0	Nan	1
S4	.456	5.67	...	...	...	675
...	...	...	...	...	...	...
Sn	.896	12.8	...	...	34.8	457

(a)

Seg ID	F1	F2	...	...	...	Fm
S1	2.32	0.244	....	....	1.229	1.44
S4	2.447	1.447	...	...	...	1.768
...	...	...	...	...	...	...
Sn	-.8070	2.180	...	...	1.234	.897

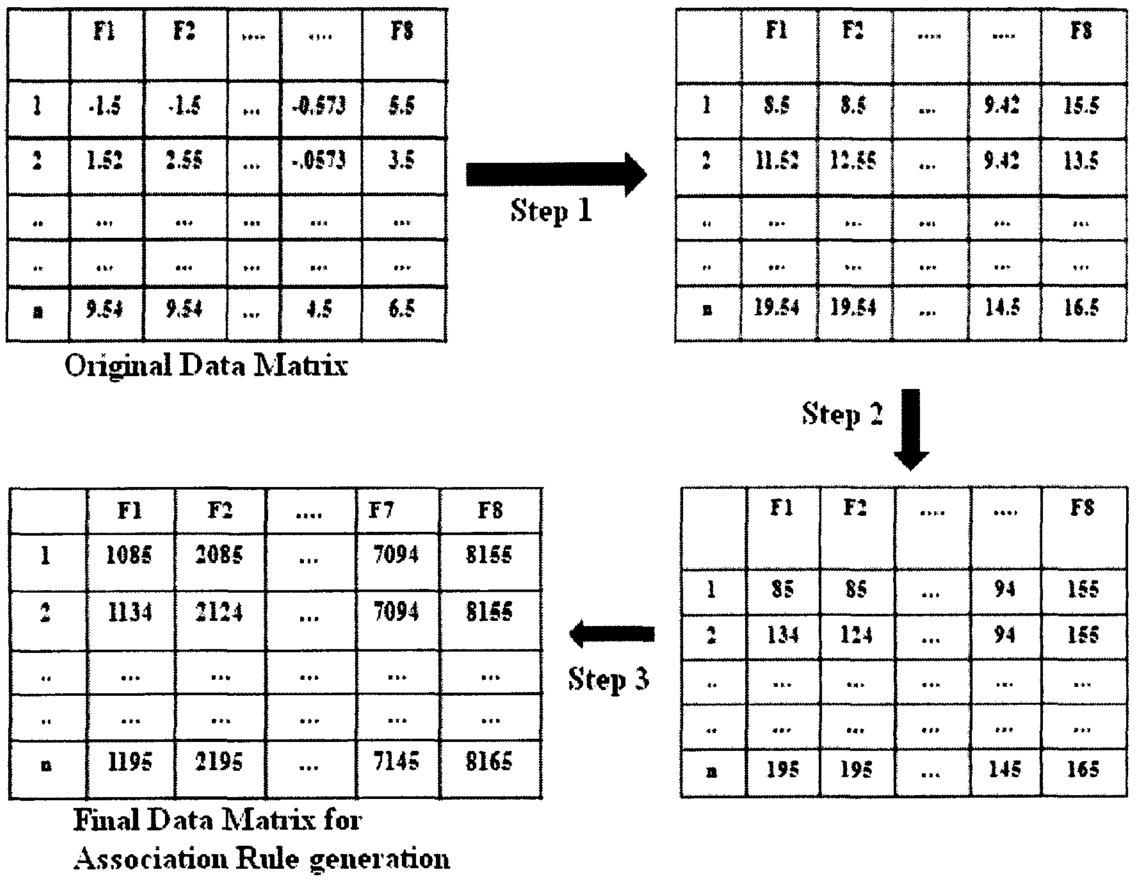
(b)

Figure 3.6 Data Preparation: (a) Original Image Data Matrix,
(b) Data Matrix after Data Removal and Normalization

The features extracted using this technique have continuous values, and, hence, it is necessary to discretize them so that association rules can be extracted from among them. In our approach, we partition each feature range into ten intervals. The value that occurs in each interval is replaced by the median of that interval; for example, a value falling in the interval 0 – 0.1, like 0.0345, is replaced by 0.05. Again, we use the combined training data for all the attributes to make the interval list for each attribute. We find the minimum and maximum from the combined data and then make the interval range starting at the minimum and ending at the maximum, with ten intervals between them. Once the interval range is formed for every attribute, the data in each image is discretized using this interval range.

3.3.4 Association Rule Mining

In our approach, each image can be seen as a dataset consisting of transactions that map to the individual feature vectors of segments. Items refer to individual feature values. The number of transactions in a database will be equal to the number of segments in the image. Since all the features have been normalized in a base range, it is necessary to give a specific ID to values in each feature, so that value -0.1567 ofthe first feature can be distinguished from value -0.1567 of the second and third features. To solve this problem, we prepare the data further using a three step procedure (shown in Figure 3.7).



Original Data Matrix

	F1	F2	....	....	F8
1	-1.5	-1.5	...	-0.573	5.5
2	1.52	2.55	...	-.0573	3.5
..	...	...	...	...	...
..	...	...	...	...	...
n	9.54	9.54	...	4.5	6.5

	F1	F2	....	....	F8
1	8.5	8.5	...	9.42	15.5
2	11.52	12.55	...	9.42	13.5
..	...	...	...	...	...
..	...	...	...	...	...
n	19.54	19.54	...	14.5	16.5

	F1	F2	....	....	F8
1	85	85	...	94	155
2	134	124	...	94	155
..	...	...	...	...	...
..	...	...	...	...	...
n	195	195	...	145	165

	F1	F2	....	F7	F8
1	1085	2085	...	7094	8155
2	1134	2124	...	7094	8155
..	...	...	...	...	...
..	...	...	...	...	...
n	1195	2195	...	7145	8165

Final Data Matrix for Association Rule generation

Figure 3.7 Data Preparation for Association Rule Development

1. First, we make all the values positive by adding 10 to every value in the matrix.
2. Second, we multiply each value in the matrix by 10 and take the floor value of each attribute. This step is performed to make each value in the matrix distinct in at least two points of precision.

3. Finally, the product of feature index *i* and 1000 is added to each value to make a value in one feature different from the same value in another feature so that it will be possible to find association rules. The following formula is used for this purpose:

$$F(i, j) = (1000 * j) + F(i, j) \qquad \text{(Equation 3.2)}$$

$$i = 1: n \quad ; \quad j = 1: \ 11.$$

In the Equation 3.2, *F (i, j)* represents the value of feature j in row i.

We then use an apriori-based association rule-mining algorithm. The support and confidence for rule mining are fixed depending upon the data. The common norm is to keep the support low and confidence high in order to extract strong representative rules. In our technique, an association rule is represented in the form:

$$1134 \wedge 2124 \Rightarrow 7094 \wedge 8074, \text{ Support= } 5\%, \text{ Confidence= } 0.9835.$$

The rule implies that if the first feature has a value of 134, and the second feature has a value of 124, then the seventh feature will have a value of 94, and the eleventh feature will have a value of 74 with 5% support and 98.35% confidence. Similar types of rules are then formed for every image in the dataset.

3.3.5 Classifier Training and Classification

For raw features, we take all the segments from an image and combine them to generate one row vector for each image. Once we generate the row vectors, each image is represented by only one row vector, where the length of the row is equal to the product of the number of segments in an image multiplied by eight. If an image has *n* segments, then the total number of columns (or length of feature vector) for that image will be *n* x 8 (because we extract eight features from each segment). Class labels are included for training data and excluded for testing data.

While the data transformation schema is simple for raw features, it is not straightforward for association rules because different images will have different rules, as will different classes of images. We, therefore, introduce a new transformation scheme based on the global rule presence. Figure 3.8 shows a graphical representation of this procedure.

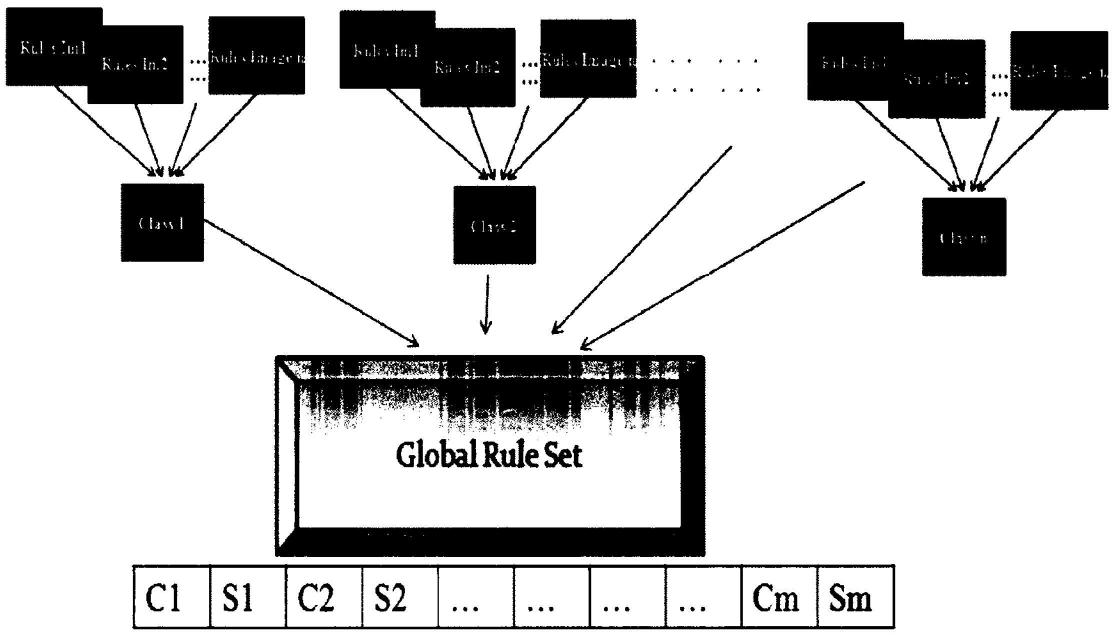
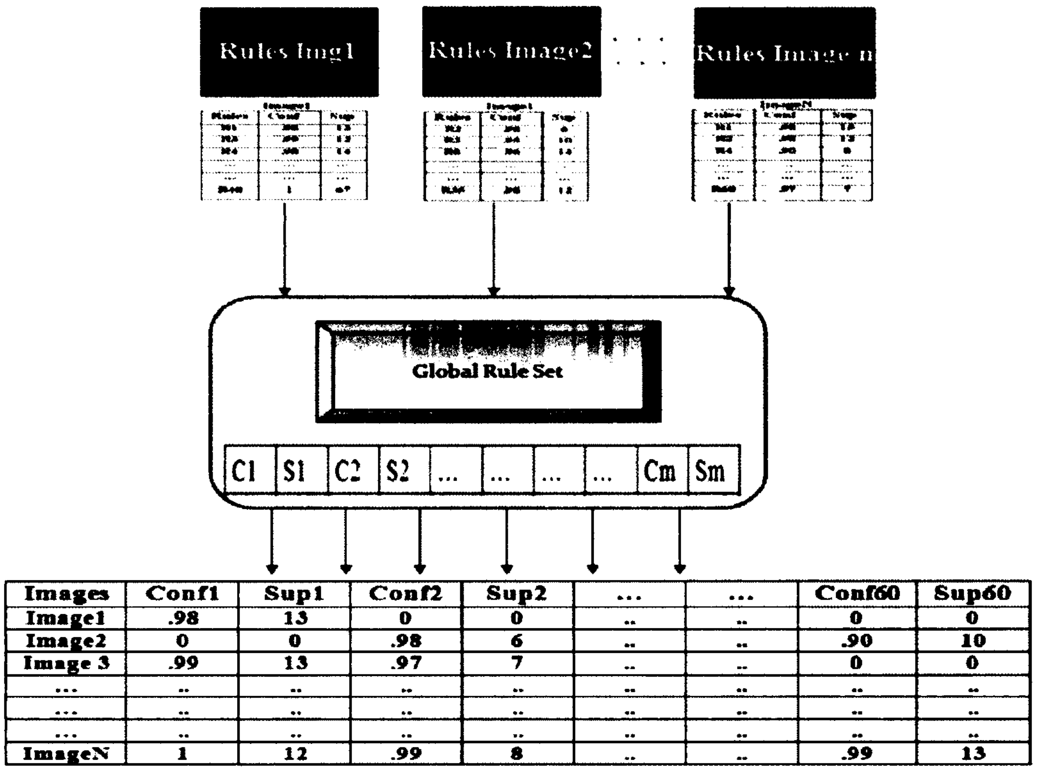


Figure 3.8 Images Rules to Global Rule Set Generation

Once the rules have been generated for each file/image, we combine the rules from objects of the same class into one set to form *class-level* rule sets. Class-level association rule sets are combined to form an aggregate *global* rule set over a complete database. Unique rules from this aggregate rule set are selected and arranged in a data matrix. Each row in the data matrix represents an image, and two consecutive columns represent the support and confidence of a single rule.

Since not all the images have every rule present, for an object/row that does not have a particular rule from the aggregate rule set, the corresponding columns are set to zero. The columns containing the confidence and support for a rule R_j can be located by the function (i-1)*2+1. For example, the confidence and support for a rule R_{40} for all the images is found in columns 79 and 80, respectively. As with the raw features, the class labels are included for training and excluded for testing. Once the transformation is complete, we have the vector-based data representation for both raw features and association rules. Figure 3.9 shows this feature vector to data matrix transformation.



Images	Conf1	Sup1	Conf2	Sup2	...	...	Conf60	Sup60
Image1	.98	13	0	0	..	..	0	0
Image2	0	0	.98	6	..	..	.90	10
Image 3	.99	13	.97	7	..	..	0	0
...	..	..	..	..	..	..	..	..
...	..	..	..	..	..	..	..	..
...	..	..	..	..	..	..	..	..
ImageN	1	12	.99	8	..	..	.99	13

Figure 3.9 Association to Data Matrix Transformation

This vector-based data representation is provided as input to multi-class classifiers for query image classification. These individual vectors are arranged in a data-matrix format where each row represents the vector based feature representation for an image. The classifiers used in our experiments are FKNN and SVM, which are described below.

FKNN (Fuzzy K-Nearest Neighbors) [40]: FKNN is an extension of the commonly used KNN (K-Nearest Neighbor) algorithm in which the memberships are fuzzy (some probabilities) and not crisp. The disadvantage of using a definitive mechanism is that it treats all the samples in a class equally. In other words, members who are at the boundary are given equal importance to the members who are in the center of the instance pool. Fuzzy theory solves this problem by replacing the crisp "is member/ not member" mechanism with probabilities like "how close are you to other instances in this class and other classes." FKNN uses a "Bayes like" normalizing factor to provide memberships to different instances.

SVM (Support Vector machines) [41]: SVM is a commonly used classification and regression method. An SVM classifies instances by constructing a hyperplane in a multi-dimensional space. It views the classification problem as an optimization problem where it constructs an optimum hyperplane which separates the training instances into two classes. The shape of the hyperplane is generated using one of the many kernel functions: linear, radial basis, or polynomial, for example. SVMs are known as "maximum margin classifiers," because they minimize the empirical classification error by maximizing the separating margin between instances. Although SVMs are used primarily for binary classification, they can be modified to work with multiple classes using the "one-vs.-one" or "one-vs.-all" strategy. We used the one-vs.-all strategy for our classification purposes.

3.4 Experimental Results and Evaluation

For our experiments, we used 90% of the data from each class for training and 10% of the data from each class for testing. For extracting association rules from each image, the support was kept low, 4%, and the confidence was kept high, 90%. Five-fold repetitions were performed in order to normalize any bias in the data. The results for both FKNN and SVM show a significant increase in accuracy when association rules are used as input instead of raw features. The results for both FKNN and SVM using raw features and association rules as input are presented in Figure 3.10 and Figure 3.11, respectively.

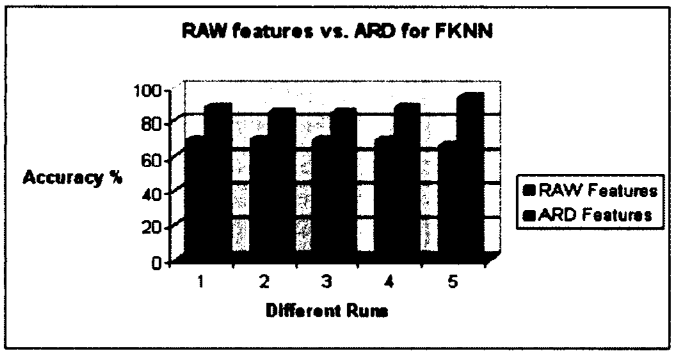


Figure 3.10 Comparison of Results for FKNN Classifier

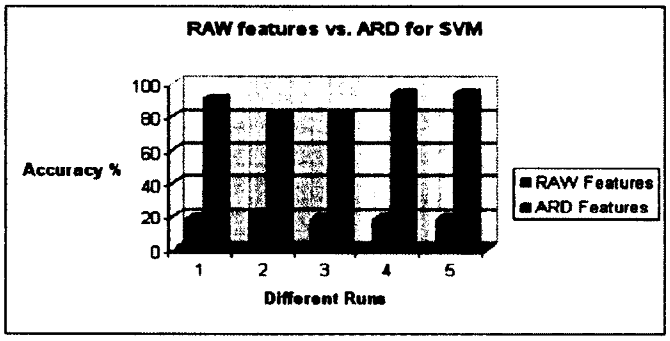


Figure 3.11 Comparison of Results for SVM Classifier

Although the increase in accuracy varies for both classifiers, it still proves that associative relationships have more discriminatory power than non-associative relationships. These results validate our hypothesis that association rules can be used successfully as higher order features for classification.

3.5 Conclusions

Associative rule mining has gained importance in the field of supervised learning due to the ability of associative rules to provide detailed information regarding the underlying data and to provide better and more accurate results. In this work, we have provided an innovative higher order data representation based on association rules. A simple transformation mechanism was used to represent the association rules as higher order features. These features were then used with SVM and F-KNN for the classification of mammograms. The comparative evaluation of results with raw features and higher order features shows that both the classifiers performed significantly better with higher order features. This discovery validated our hypothesis that associative relationships can indeed be represented as higher order features, and they capture more information about the underlying data than raw features represented as higher order features do.

While there is significant improvement in accuracy using this simple transformation schema, it still only uses the support and confidence of the rules. The intra-class and inter-class presence of association rules can provide even more information. This observation has led us to focus on building a novel associative classifier that uses the intra-class and inter-class importance of rules. In Chapter 4, we present this framework.

CHAPTER 4

WAR-BC: WEIGHTED RULE BASED ASSOCIATIVE CLASSIFICATION

4.1 Introduction

In Chapter 3, we introduced a novel association-rule-based data representation schema which provided us with valuable insight into the associative classification process. In this chapter, we present a model for building a new associative classifier that utilizes the inter-class and intra-class similarities and dissimilarities of the data as basis to weight the association rules and then use these rules to perform classification.

4.2 Why a New Algorithm is Necessary

Classification using frequent patterns, otherwise known as "associative classification" (AC), has recently gained importance due to its ability to provide better results and to summarize the underlying information of data. While it is a well-known fact that a combination of features usually provides more knowledge than a single feature, not all patterns are useful. Hence, it is important that we generate models that will only use the informative patterns. In order to generate a good model, we must determine what pattern is useful, and how to measure usefulness. While most of the ACs use only a handful of patterns, we argue that deleting/pruning patterns causes problems in multiclass classification scenarios. One of the biggest problems with current AC algorithms is class-constrained rules, (i.e. keeping the class on the right hand side of the rule). Most AC algorithms push the class label into the rule generation module, thereby generating only “class association rules” (CARs). We argue that this limitation is a major component of multi-class classification problems, especially when the difference in instances from heterogeneous classes is very low, (e.g. images of sky and airplane). Let us view this problem through the following example. Consider a rule $f_1, f_2,..f_k \Rightarrow c_i$. This rule could be present in both classes c_1 and c_2, and, hence, could cause confusion for the classifier. In most ACs, this rule will be deleted to avoid confusion. If we examine this rule more carefully, we can see that *frequent* itemset $f_1, f_2,..f_k$ might be present in more than one class. However, a rule $f_1, f_2 \Rightarrow f_k$ or $f_1 \Rightarrow f_2, f_k$ might be present in only one of the two classes. Hence, this rule could be used as an important discriminator for classification. In addition, even if the same rule is present in more than one class, it might be present with a different support and confidence. This support and confidence information could again be used for classification purposes. Some other problems with associative classifiers like rule ranking, incremental learning, rule overlapping, and multi-label rule classification, were put forward by Thabtah [42]. We address all of these issues in our proposed research.

We propose a novel weighted rule-based associative classification algorithm, WAR-BC, and address most of those issues. The class constraint issue stems from the restrictions in data representation, so we propose to use a different type of representation where each instance is treated as a separate transaction database. Then, regular association rules are formed followed by classifier training. We used two datasets from different domains to evaluate our method. Since these datasets are from different domains, some data preprocessing and feature extraction modules are different for both of them.

4.3 Problem Definition

Let D be a training database, $I = \{I_1, I_2, ..., I_M\}$ be a set of images, and C be a class attribute $C = \{c_1, c_2, ..., c_N\}$. Let $f_i = \{f_{i1}, f_{i2}, ..., f_{ik}\}$ be a set of features extracted for each image $I_i \mid I_i \in I, \forall 1 \le i \le M$. Then, every image in a dataset can now be represented as $\{f_i, c_j\}$, where $c_j \in C$ is the class label of image *i*. Let $R_i = \{r_{i1}, r_{i2}, ..., r_{ip}\} \forall p \ge 1$ be the set of rules extracted for image *i*. The weight of a rule is a positive real number representing the importance of a rule. Let r_intra_{i1} and r_inter_{i1} represent the intra-class and inter-class weights of the rule r_{i1}; then the weighted rule representation of an image *i* is $\{RWeight_i, c_j\}$. Then the associative classification problem is to learn a classification model based on the weighted rule space $F : \{RWeight_i\} \rightarrow y$ where y is a class label.

4.4 Datasets

Mammogram Dataset: This dataset is the same MIAS mammogram dataset that we used for our experiments in Chapter 3.

Corel Image Dataset [43]: The Corel image dataset is a commonly used scenic image dataset for multi-class image classification. We only use a subset of the 2000 image dataset. The dataset consists of 20 classes of 100 images. The images are saved in the JPEG format, and each image is of either 384x256 or 256x384 resolution. The 20 image categories used from the Corel database are Africa, Beach, Buildings, Buses, Dinosaurs, Elephants, Flowers, Horses, Mountains, Food, Dogs, Lizards, Models, Sunsets, Cars, Waterfalls, Antiques, Ships, Skiing, and Deserts. The major reason for using this dataset was the availability of benchmark results for comparison [44]. Figure 4.1 shows a representative image from each of the 20 classes.

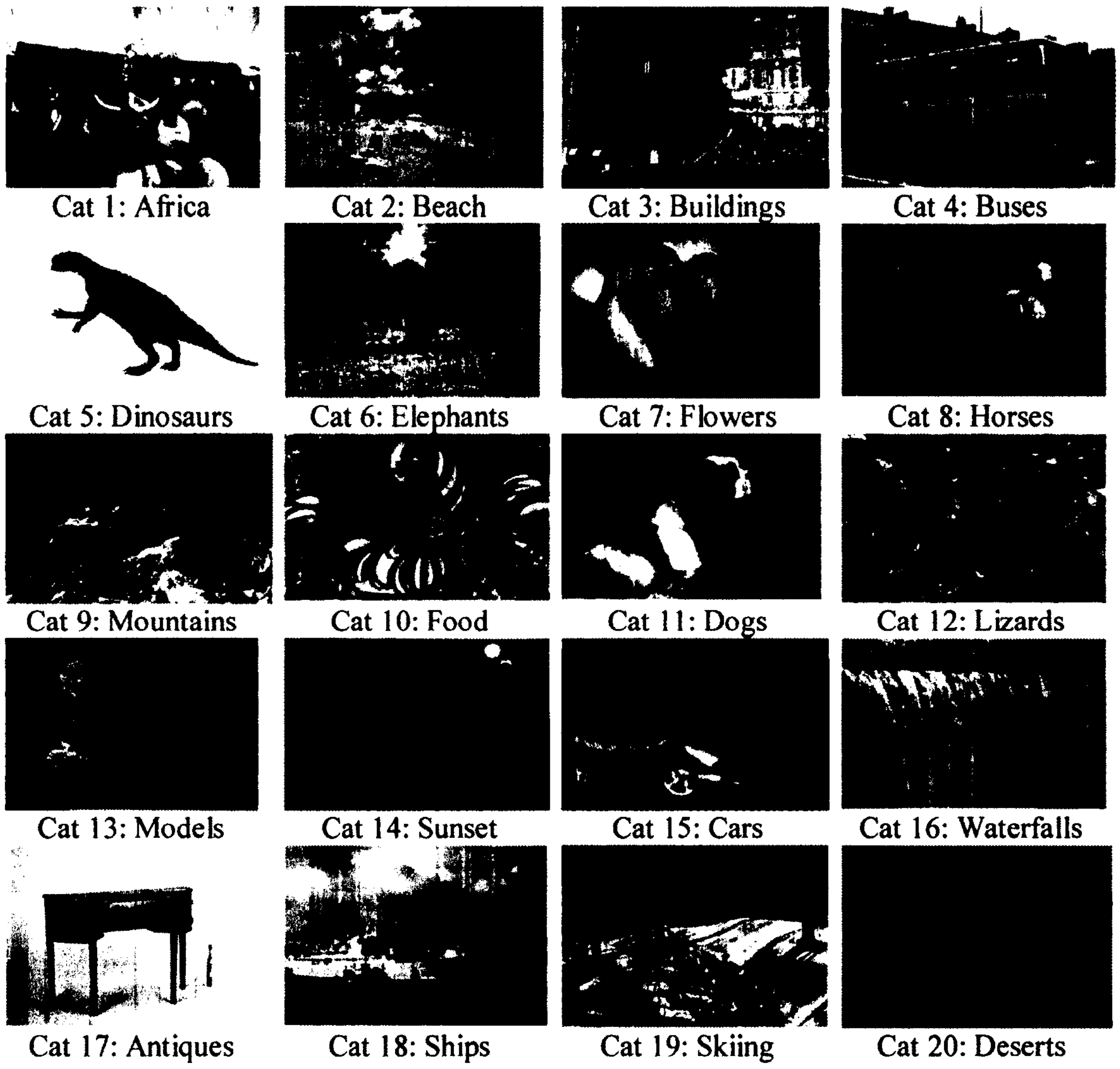


Figure 4.1 Representative Images from the 20 COREL Image Categories [43]

4.5 Proposed Methodology

Since we have used three datasets to evaluate WAR-BC, some steps of the methodology are different for all of the datasets, particularly feature extraction and data preparation. We will explain these steps, which are different for our datasets, separately. The methodology can be divided into five phases:

I. Data preprocessing,

II. Feature extraction,

III. Data preparation,

IV. Association rule mining, and

V. Classifier training and classification.

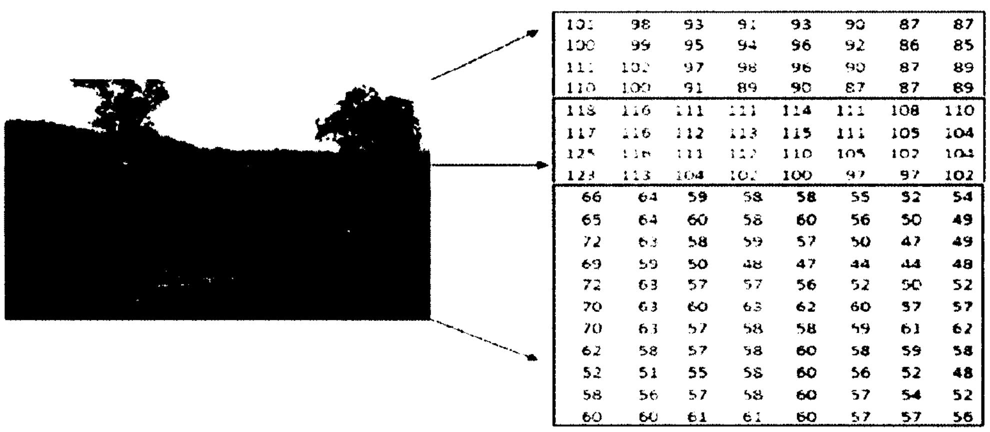
4.5.1 Data Preprocessing

As explained in Chapter 3, because mammograms contain labels that are intentionally added to them by technicians for reference, they are noisy. Hence, they need to be cleaned before any features can be extracted from them. For brevity, we do not explain the preprocessing step here, as it is the same as described in Section 3.3.1. The Corel image dataset does not require any preprocessing, as it is significantly noise free.

4.5.2 Feature Extraction

Mammogram Dataset: The feature extraction procedure for a mammogram dataset is the same as described in Section 3.3.2.

Corel Image Dataset: We use the same grid-based segmentation as is used for mammograms to extract low-level wavelet-based features from the images. Additionally, since these images are colored, we also extract color features from them. Initially, each image is divided into *n x n* non-overlapping segments, and then low-level features are extracted from each of these segments. The size of the block, (i.e. n x n), is chosen so that approximately 2000 – 3000 sub-blocks of the image are obtained. For our analysis, we use n=5. The scenic images from Corel stock data contain a great deal more information and objects of interest than the mammograms, which contain only one object in an image, the breast. This high level of content is one reason we keep the block size small. Corel stock images are 3-D colored images, so we use both the color and wavelet information for classification. A colored image is an image where each pixel is represented by a combination of three values- one value for red, one value for green, and one value for blue. These values are why colored images are also sometimes referred to as RGB images. In most platforms, a colored image is stored as a 3-D matrix, where the first matrix represents the values of red color for individual pixels, the second matrix represents the green values for the same pixels, and the third matrix represents the blue values. The true color of each pixel is represented by the combination of each of these three RGB values for that pixel. Figure 4.2 shows a colored image and its matrix representation.



101	98	93	91	93	90	87	87
100	99	95	94	96	92	86	85
111	102	97	98	96	90	87	89
110	100	91	89	90	87	87	89
118	116	111	111	114	111	108	110
117	116	112	113	115	111	105	104
125	116	111	112	110	105	102	104
123	113	104	102	100	97	97	102
66	64	59	58	58	55	52	54
65	64	60	58	60	56	50	49
72	63	58	59	57	50	47	49
69	59	50	48	47	44	44	48
72	63	57	57	56	52	50	52
70	63	60	63	62	60	57	57
70	63	57	58	58	59	61	62
62	58	57	58	60	58	59	58
52	51	55	58	60	56	52	48
58	56	57	58	60	57	54	52
60	60	61	61	60	57	57	56

Figure 4.2 A Colored Image and Its Matrix Representation

For colored images, we extract six features, three color moments, and three wavelet moments. The three color features are the first order moment (average) of the red, green, and blue values of the pixels in the current segment. The three wavelet moments represent the energy, which is the square root of the second order moment of wavelet coefficients, in high frequency bands of the wavelet transform [44]. These features are extracted using the following method:

1) Take an individual non-overlapping segment, and transform this colored segment into its grayscale counterpart using color-to-grayscale conversion;
2) Perform one level wavelet transform on this segment using Daubechies-4 wavelet as the parent wavelet;
3) The transformation provides coefficients in four frequency bands: LL, LH, HL, and HH. The three wavelet features are then extracted by taking the square root of the second order moment (variance) of the coefficients from HL, LH, and HH bands separately.

These three wavelet features are combined with the mean (average) values of the red, green, and blue values of the segment resulting in a feature vector of length $k=6$. Each vector is given a unique segment ID, which, in our case, is the number of the segment from which the features were extracted, e.g. *TID 1 (f1, f2, f3.....fk)* and *TID 2(f1, f2, f3....fk)*, where k=6, since there are six features for any given segment.

4.5.3 Data Preparation

We use the same data preparation step for mammograms that we explained in Section 3.3.3. Since there is no single object of interest in the Corel stock images, the features extracted from these images do not give any NaN values. As a result, we do not need to delete any rows from the database. However, as with the mammogram dataset, we use the Z-score normalization and data discretization mechanism. For Corel images, we again discretize the values into ten bins.

4.5.4 Association Rule Mining

Association rule mining is similar for both the mammogram dataset and the Corel image dataset. Each image is viewed as a dataset consisting of transactions that map to the individual feature vectors of segments. The length of the feature vector varies depending on the type of images used (eight for mammograms and six for Corel images). Items refer to individual feature values. The number of transactions in a database is equal to the number of segments of the image. While the number of transactions for each mammogram may differ depending upon the size of the breast part and the black background, the number of segments for a Corel image dataset is the same, since the total resolution of the Corel images is the same (384x256 = 256x386=98304). The feature values are changed using the data transformation mechanism (Equation 3.1) explained in Section 3.3.4, so that similar values from different features are represented by unique IDs. For a quick review, the formula for Equation 3.2 is provided below for transformation.

$$NewF\,(i, j) = (1000 * j) + F\,(i, j) \qquad \text{(Equation 3.2)}$$

$$i = 1 : n \quad ; \quad j = 1 : \ 11.$$

In Equation 3.2, $F\ (i, j)$ represents the value of feature j in row i. An apriori algorithm with minimum support and confidence is used to generate association rules of the form

$1004 \wedge 2104 \Rightarrow 7090 \wedge 8070$ Support= 7%, Confidence= 90%.

4.5.5 Classifier Training and Classification (WAR-BC)

The procedure is similar for both mammogram and scenic images. We will explain the differences between them. First, a fixed percentage of the data is selected from each class for the training phase. The data chosen depends upon the user-defined percentage.

After association rule generation for the training images, the rules from images in each class are combined into a class-level association rule set. Each class will have its own rule sets. The size of this class level rule set varies for each class depending upon the depth of information present in images of a class and the rules generated for them. Further, the size is also affected by the imbalance in data for different classes. The mammogram data is also affected by an imbalance in data, as we usually have more normal data than cancerous data. For an individual class-level rule set, the *frequency* of each rule for that class (the intra-class weight of a rule) is calculated. The frequency information for mammograms is the percentage of training images per class that have the rule present in them. The frequency information for Corel images is the number of images in a class that have that rule. Since the classes in the Corel images dataset are balanced, we do not choose a percentage for Corel images.

However, we can work with percentages in the Corel images. The rules are ranked using this intra-class frequency measure to find the most important to the least important rule. Now, it is possible that one rule is present in the images of other classes, possibly because we generate all rules and not just predictive rules from data and because there are classes that might be similar, (e.g. sky and airplane, and ocean and ship, etc). Therefore, to solve this issue, we assign another frequency weight to a rule based on its presence across multiple classes. We call this the "inter-class weight" of a rule. To calculate this weight, we combine the class-level rule sets from all the classes into a global rule set which has only unique rules from all the classes. Associated with each rule is the frequency of the rule in each class. We introduce two rule measures, *horizontal* weight and *vertical* weight, for all the rules in the database. Horizontal weight evaluates the inter-class importance of the rule and, hence, is calculated based on the *frequency* of a rule across the classes. After calculating the frequency of a rule in all the classes in the database, the individual class frequencies are divided by the sum of all the frequencies of that rule over the classes, resulting in the relative frequency measure of a rule for each class.

The relative frequency of a rule for a class is known as its horizontal weight for that class. For example, suppose there are 1000 unique rules present across all classes. In a class for which the rule is not present in any of its images, the rule is given a horizontal weight of zero. For instance, then, if rule R_j is present in 20% of training images in Class 1, 60% in Class 2, 30% in Class 3, and 40% in Class 5, and it is not present in any other class of the dataset, then the frequency of R_j, in Class 1 is 0.20, in Class 2 is 0.60, in Class 3 is 0.30, and in Class 5 is 0.40. The relative frequency/horizontal weight of R_j for Class 1 is 0.20/(0.20+0.60+0.30+0.40)=0.133, for Class 2 is 0.60/1.5=0.40, for Class 3 is 0.30/1.5 =0.2, and for Class 5 is 0.40/1.5= 0.267.

While horizontal weighting uncovers the inter-class importance of a rule, the vertical weight finds the intra-class importance of each rule. Vertical weight is performed separately for each class. For calculating vertical weights, the individual class-level rule sets are first sorted according to the decreasing order of the confidence value of a rule. However, it is common that more than one rule will have the same confidence. Therefore, the rules are sorted further according to the decreasing order of their support within each confidence value. After this rearrangement, the rule with the highest confidence/support pair gets the highest rank (1^{st}) and the highest weight. The highest weight is equal to the number of rules present in a class. For example, if there are 200 rules in a class, then the highest ranked rule (first rule) will be assigned a weight of 200. The weight of second ranked rule is one less, then the first ranked rule, which, in our example, will be 199. The weight of the third ranked rule is one less than second ranked rule and two less than the first ranked rule. The procedure is used to rank all the rules in the class-level rule set until the last ranked rule, which is assigned a weight of 1. These weights are then normalized in the 0-1 range. The pseudocode for rule weighting is shown in Figure 4.3.

```
Algorithm  Rule Weighting (R-Weight) is used to provide Horizontal and Vertical
           weights to every rule present in the training database

Input   Number of classes $C$, combined list of training rules for each class $L_C$ ;
           Number of rules in each class (j) $C_j$, Total number of rules $N$

Output  Horizontal and Vertical weight matrices of rules

Method:

  (1) For every rule $R_iC_j$ $\forall i \leq N, j \leq C$
  (2)     $frequency(R_iC_j)$ ← percentage of images in $C_j$ having $R_iC_j$
  (3)     $Horizontal\_weight(R_iC_j) \leftarrow frequency(R_iC_j) / \sum_{j=1}^{C} R_iC_j$
  (4) End For // Horizontal weighting complete
  (5) For every class $j$ $\forall j \leq C$
  (6)     $Rank\_rules_j$ ← Sort rules according to confidence and then support in each
  (7)                  Confidence
  (8)     For every rule $R_iC_j$ $\forall i \leq C_j, j \leq C$
  (9)         $Vertical\_weight(R_iC_j) \leftarrow (C_j - Rank\_rules(R_iC_j))$
  (10)    End For
  (11)    $Normalized\_weight(R_iC_j)$ ← normalize vertical weights in the range 0-1
  (12) End For
```

Figure 4.3 Algorithm for Rule Weighting

Finally, we also use the cardinality, (i.e. the number of items in a rule), of the rule. The total weight of the rule is then represented as a sum of its horizontal and vertical weight multiplied by its *cardinality*. The weighting notations are explained herein.

Let us assume C = the total number of classes and N = the total number of global rules,

CD_i = Cardinality of rule $R_i, \forall i \leq N$;

C_j = Rules in class $j, \forall j \leq C$;

$H_j R_i$ = Horizontal weight of rule R_i for class $j, \forall (i \leq N, j \leq C)$;

$V_j R_i$ = Vertical weight of rule R_i for class $j, \forall (i \leq C_j, j \leq C)$;

Q = Number of rules from query image which match with global set of rules (N).

Then, the weight of a rule R_i for class C_j is defined by the formula:

$$W_j R_i = (H_j R_i + V_j R_i) \times CD_i .$$

Once the classifier is trained, the next step is to classify previously unseen instances. For every new test instance, association rules are generated using the same support and confidence as was used for training instances. Then, an individual rule from a query image is taken and is matched with the global rule set to find its horizontal weight. A match is defined as the matching of all the items in a rule body, both on the left and right hand side of the rule. This matching is different from the CAR matching, for which only the LHS was used because the predictor or class label was always on the RHS. The same rule is then matched with the different class-level rule sets to find its vertical weight for each class. Next, the horizontal and vertical weights of a rule are added and multiplied by the cardinality of the rule. The resultant product is called the *score* of the rule. The same procedure is repeated for every rule in the query image. Finally, the scores of the matching rules are added on a class-by-class basis, and a cumulative sum is calculated for each class. Finally, the query image is classified to the class with the highest cumulative sum. Figure 4.4 shows the algorithmic process.

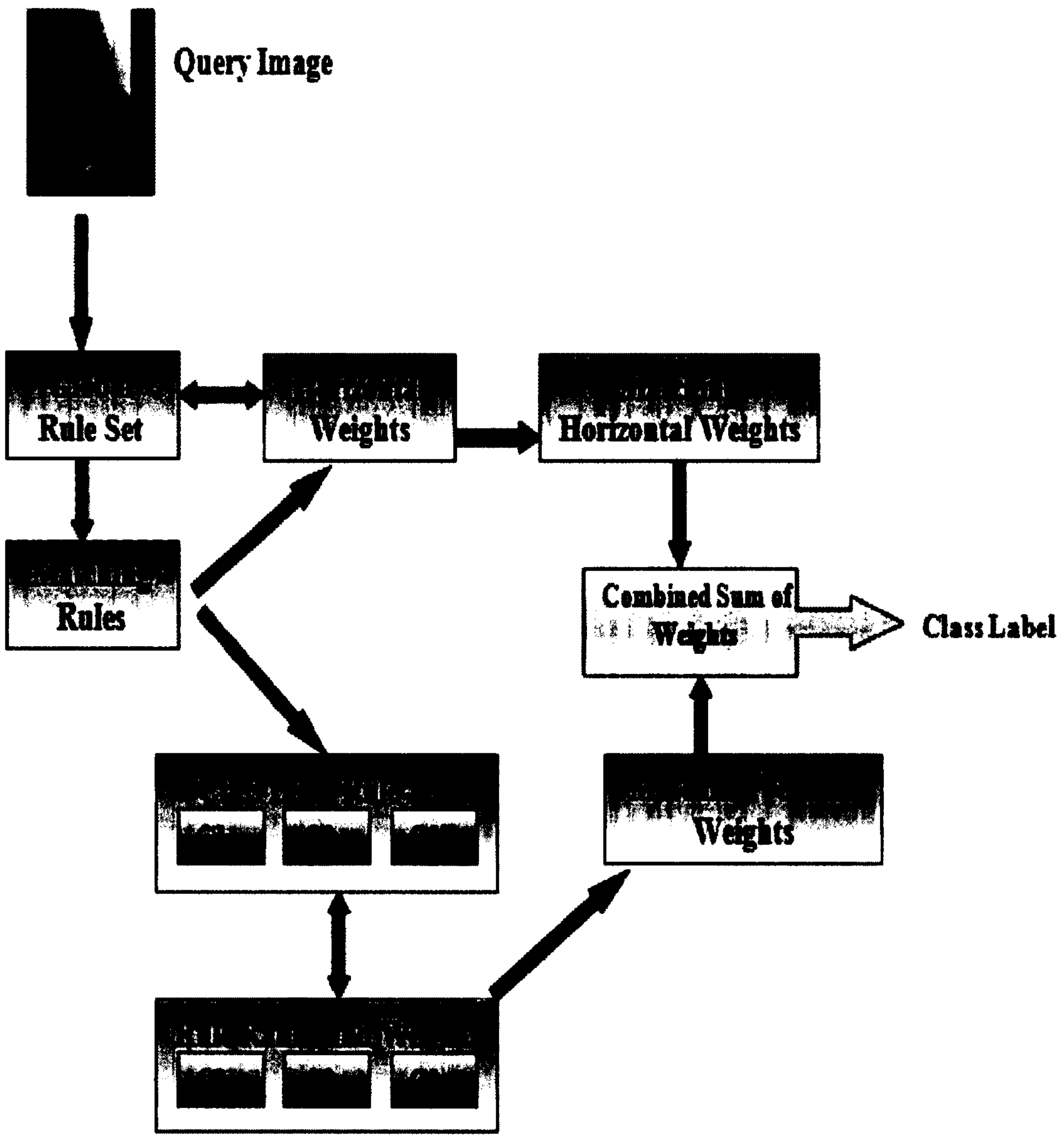


Figure 4.4 Query Mechanism for the Algorithm

The sum of all the rules for class C_j is calculated using the formula defined below:

$$Tot_j = \sum_{i=1}^{Q} (W_j R_i) .$$

Then, the output label (predicted class) can be decided using the formula:

$$ClassLabel \Leftarrow \arg\max_{j=1}^{C}(Tot_j) .$$

If N_i = the number of query images from class *I* and Q_i = the number of images in class *i* correctly predicted by the classifier, then the accuracy for class *i* can be defined by: $CAcc_i = \frac{Q_i}{N_i}$,and the total accuracy is given by: $TotAccuracy = \frac{\sum_{i=1}^{C} Q_i}{\sum_{i=1}^{C} N_i}$

4.6 Results

Since we use a score-based classification mechanism, each query image will have its scores for every class in the database. Hence, we can have both "crisp" classification where only one class is predicted per query image, or "fuzzy" classification where the query image is classified to the top *k* classes based on the top k scores. We explain the results for each datasets separately.

4.6.1 Mammogram Dataset Results

In order to make an accurate comparison, we compare the results of the proposed technique WAR-BC with six existing techniques from literature. Antonie et al.used two techniques for mammogram classification: a three layer (input, hidden, and output) back propagation neural network (BPNN) and an association-rule-based classifier (ARC-AC)

[45]. Each layer in BPNN had different nodes with 69 nodes in input layer, 10 nodes in hidden layer, and 1 node in output layer. The single node in output layer was responsible for the classification of a query image.

The authors developed ARC-AC as an association-rule-based classification algorithm. The rules generated were class constrained rules (class label on the RHS of the rule) extracted from the entire mammogram dataset. Initial support was set at 10% and an initial confidence was set at 0%. Depending on the classification accuracy over training data, the confidence was increased in the tuning phase. ARC-BC, proposed by the same authors, improved on ARC-AC [46]. The association rules in ARC-BC were formed separately for each class with different support and confidence values, unlike ARC-AC wherein the same support and confidence were used for rule generation from the entire dataset. Yun et al. proposed the Joining Associative Classifier (JAC), which they used to combine rough set theory with association rule mining to build a hybrid classifier [47]. For further reading about these works please refer to [45], [46], and [47].

Apart from the existing benchmark results present in the literature, we also implement two extra classifiers for comparison: F-KNN [40] and PNC2 [48]. The details of the F-KNN classifier haven been presented in Chapter 3. We perform ten-fold repetition using F-KNN with the same images as used for WAR-BC. We also use PNC2, a hierarchical agglomerative clustering tool that generates "if – then" type rules directly from the data for use with classification. Initially, the input data matrix provided to PNC2 for rule generation was the same as the input data matrix provided for F-KNN. While the model training time for one run of the system with this input matrix is very large (25 hours on a single processor AMD opteron 2.39 GHz Machine), the testing accuracy is only 53.13%.

In an attempt to boost the model training runtime for 10-fold repetition, we take four consecutive segments (in row-major format) for each data matrix and average the features to extract the derived aggregated value. These average features values are afterwards used as input for rule generation and classification. This modification reduces the data size and time for model learning significantly without compromising the training accuracy. For further information about the working of FKNN and PNC2, we recommend [40] and [48].

4.6.1.1 Associative Classification

All of the existing benchmark techniques have used 90% data for training and 10% for testing. In order to make accurate comparisons, we use the same training/testing split. For association rule generation using WAR-BC, the support value is kept low, at 4%, and the confidence level is kept high, at 90%. The same support and confidence values are used for all the classes to generate rules separately.

The MIAS mammogram dataset is highly unbalanced with more normal cases than benign and malignant. Our rule-weighting schema easily handles this multiple class imbalance. This non-sensitivity to class imbalance is an important improvement over existing association-rule-based techniques that are sensitive to unbalanced data like ARC-AC. Further, we use true association rules, rather than class-constrained rules like those used by ARC-AC and ARC-BC, in which the class is kept on the right-hand side of the rule.

We perform ten-fold repetition to compare the results of [45], [46], [47], [40], and [48] with WAR-BC. The comparative results of these six techniques with our proposed technique are presented in Table 4.1.

Table 4.1 Accuracy Comparison of Existing Techniques with WAR-BC

	BP NN	ARC-AC	JAC	ARC-BC	F-KNN	PNC2	WAR-BC
1st	96.87	67.64	69.342	80	59.37	53.13	93.75
2nd	90.62	79.41	86.373	93.33	46.87	56.25	90.62
3rd	90.62	67.64	77.586	86.67	56.25	62.5	93.75
4th	78.125	61.76	72.912	76.67	71.87	62.5	84.37
5th	81.25	64.7	78.224	70	53.12	59.37	93.75
6th	84.375	64.7	77.055	76.67	75	20	81.25
7th	65.625	64.7	77.691	83.33	65.25	68.75	90.62
8th	75	64.7	73.752	76.67	56.25	20	90.62
9th	56.25	67.64	82.123	76.67	53.12	68.75	87.5
10th	93.75	88.23	79.819	83.33	59.37	12.5	90.62
Avg. Acc	**81.2485**	**69.112**	**77.4877**	**80.334**	**59.647**	**48.375**	**89.685**

As can be seen from Table 4.1, WAR-BC outperforms all six comparative techniques in terms of accuracy over the same training/testing data. The results for BP NN, ARC-AC, ARC-BC, JAC, F-KNN, and PNC2 are based on 90% training and 10% testing data. The average accuracy of our technique for the same training/testing data is almost 20% higher than ARC-AC, 10% higher than ARC-BC, 8% higher than BP NN, 12% higher than JAC, 30% higher than F-KNN, and 40% higher than PNC2. These results show that WAR-BC is superior to existing techniques. Additionally, we also experimented with two splits of data, namely 70% training and 30% testing, and 80% training and 20% testing. The average accuracy achieved for these splits is higher than the accuracy compared to the existing six techniques. These results can be seen in Figure 4.5. (a). Further, we experimented with class constrained rules (WAR-CCBC), (i.e. rules with class label on the RHS). These results are seen in Figure 4.5. (b). As can be seen, the accuracy with WAR-CCBC is still higher than all the other techniques.

WAR-BC		
	70%	80%
1st	93.81	96.87
2nd	92.7	93.75
3rd	90.72	96.87
4th	61.85	71.87
5th	90.62	93.75
6th	65.97	73.47
7th	69.07	75
8th	84.53	84.37
9th	84.37	82.81
10th	93.75	93.75
Avg. Acc	**82.739**	**86.251**

(a)

WAR-CCRBC	
1st	96.87
2nd	90.62
3rd	96.87
4th	75
5th	93.75
6th	75
7th	78.12
8th	87.5
9th	90.62
10th	93.75
Avg. Acc	**87.81**

(b)

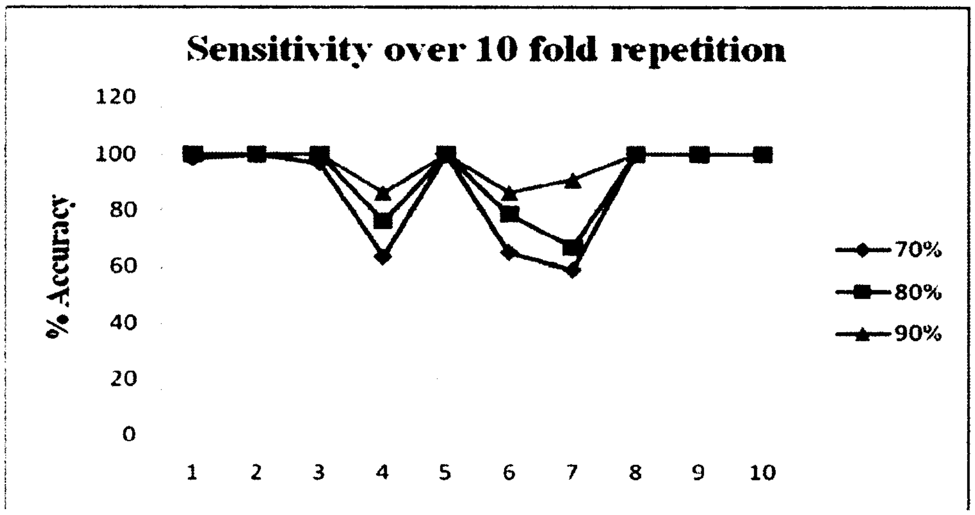
Figure 4.5 (a) Average Accuracy for 70% and 80% Training Data over 10-Fold Repetition, (b) Average Accuracy for 90% Training Data using Class Constrained Rules

In order to assess the accuracy for each class, we also calculate the precision and sensitivity. Following formulas are used for calculating Precision and Sensitivity values:

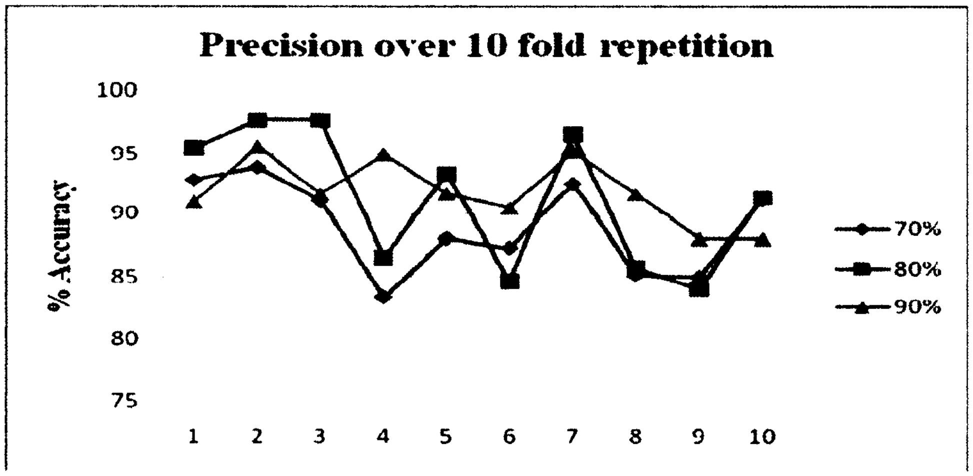
$$Precision = \frac{TP}{TP + FP} \qquad \text{(Equation 4.1)}$$

$$Sensitivity/Recall = \frac{TP}{TP + FN} \qquad \text{(Equation 4.2)}$$

Since precision and sensitivity is calculated for only binary cases, we keep the data from class normal as such, but combine the data from classes benign and malignant to form a new class, labeled abnormal. Hence, the precision and sensitivity is calculated for normal versus abnormal mammogram classification. In this case, TP = images which are normal and are labeled normal by the classifier, FP = images which are abnormal, but are labeled normal, TN = images which are abnormal and are labeled abnormal, and FN = images which are normal, but are labeled abnormal. Figure 4.6. (a)-(b) shows the precision and sensitivity graphs using WAR-BC for three percentage splits of data (70/30, 80/20, and 90/10) over tenfold repetition.



(a)



(b)

Figure 4.6 (a) Sensitivity over 10 Runs (b) Precision over 10 Runs

As can be seen from the graphs, the precision and sensitivity values for our algorithm are high for all pairs of classification. For 90% training data, the average precision is 91.83%, and the average sensitivity is 96.36%. A confusion matrix for the best-case scenario is shown in Figure 4.7. We can see from the results that all normal images are classified correctly into class normal and only one image each from the benign and malignant classes is misclassified into the normal class

	Normal	Benign	Malign
Normal	22	0	0
Benign	1	5	0
Malign	1	0	3

Figure 4.7 Confusion Matrix for the Best Case Scenario

4.6.1.2 Classification Based on Mammogram Density

During experimentation, we note that misclassification indicates a problem with the image, rather than with the classifier. After a close inspection, we find that the mammogram density is an important factor affecting this result. This revelation motivates us to perform classification separately for different mammogram density classes. Three mammogram density classes: fatty, glandular, and dense, are used for the MIAS mammogram dataset.

In this set of experiments, WAR-BC is used to classify normal vs. benign vs. malignant cases separately, for each of the three tissue densities. Of the 322 images in the dataset, 108 belong to class fatty, 101 belong to class glandular, and 112 belong to class dense. These images are divided based on abnormality. For class fatty, there are 67 normal, 23 benign, and 18 malignant images; for class glandular, there are 65 normal, 20 benign, and 16 malignant images, and for class dense, there are 76 normal, 20 benign, and 16 malignant images. Training and testing of WAR-BC is performed separately for each density class with three percentage splits of data (70/30, 80/20, and 90/10). The accuracy in the class fatty for 70/30 data pair is 77%, for 80/20 is 86.84%, and for 90/10 is 95%. Average accuracy for class glandular for the same data split is 85%, 84.38%, and 88%. Moreover, the accuracy for dense class is 84.23%, 87.7%, and 86.6% for the same data split. Figure 4.8 shows the average accuracy for different density classes.

90%	1st run	2nd run	3rd run	4th run	5th run	Average
Fatty	100	100	100	75	100	**95**
Glandular	100	80	80	90	90	**88**
Dense	91.66	83.33	83.33	83.33	91.66	**86.662**

(a)

80%	1st run	2nd run	3rd run	4th run	5th run	Average
Fatty	95.65	95.65	95.65	69	78.26	**86.842**
Glandular	100	77.27	77.27	81	86.36	**84.38**
Dense	95.65	86.95	86.95	82	86.95	**87.7**

(b)

70%	1st run	2nd run	3rd run	4th run	5th run	Average
Fatty	91.17	82.35	79.14	67.64	64.7	**77**
Glandular	100	81.25	81.25	75	87.5	**85**
Dense	96.96	78.78	78.78	87.87	78.78	**84.234**

(c)

Figure 4.8 Density-Based Classification over Five Runs

4.6.1.3 Classification Using Region of Interest Information

In the previous set of experiments, we ignored the region of interest (ROI) information provided with mammograms. The ROI provides us with the location of the micricalcification on the mammogram. The *x* and *y* coordinates and the radius (in pixels) of the abnormality on the mammogram are provided for some abnormal (benign and malignant) mammograms. If the calcifications are scattered over the entire mammogram, then information for the ROI is omitted. From the images, which contain ROI information, we segment the ROI and formulate rules for the remaining portions of the image. Rules for images from one class are combined to form a class rule set (for example, R2). These rules are matched with the existing rule set for the same class (R1) where rules from the entire image (including ROI) are formulated. Horizontal weighting for rule set R1 is performed in the same way as earlier but the vertical weighting is modified.

The rules in R1, which did not match with the rules in R2 (for example, *R_NotCommon*), are given the highest weight in vertical weighting as these rules are assumed to come from the ROI since they are not present in the new rules set R2. Rule ranking is performed separately on this set of rules first. Then, the rules that match both set R1 and R2 and for which the support and confidence are different for set R1 and R2 are selected (say *R_CDiffSupConf*). These rules are important as a part of these rules lies in ROI (evident from the decrease in support and confidence for rules set R2). Rule ranking is performed on them separately, but the ranks are lower than the ranks for *R_NotCommon*. Finally, the remaining rules are taken and ranks are given to them. Therefore, the highest ranks are given to *R_NotCommon*, the next highest weights are given to *R_CDiffSupConf*, and the lowest weights are given to rules that are common for both R1 and R2, and their support and confidence does not change. The class level and overall accuracies for 10 runs can be seen in Figure 4.9 and Figure 4.10 respectively.

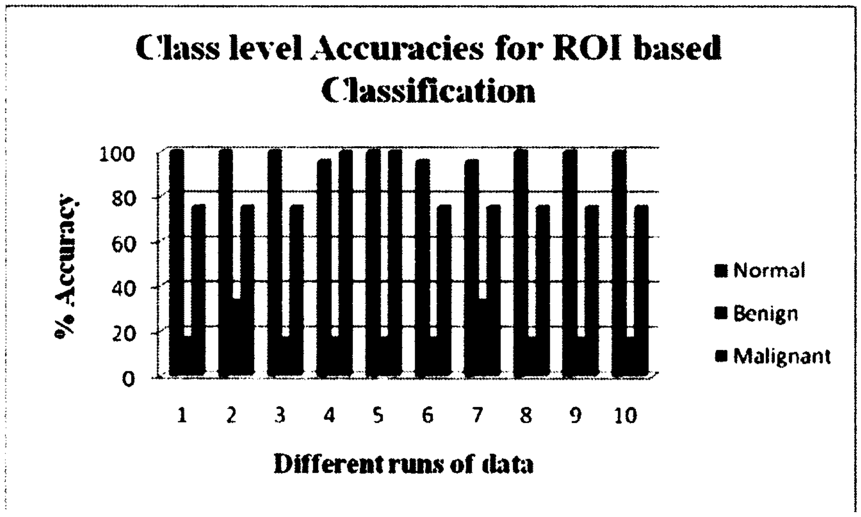


Figure 4.9 Class-Level Accuracies for WAR-BC over 10 Runs

	ROI-BC
1st	81.25
2nd	84.37
3rd	81.25
4th	81.25
5th	84.37
6th	78.12
7th	81.25
8th	81.25
9th	81.25
10th	81.25
Avg. Acc	**81.561**

Figure 4.10 Classification Accuracy for Regions-Based Classifier

From the results in Figure 4.10, it is evident that incorporating the ROI information into the classifier decreases the accuracy. After a close inspection we see that there is only one class, Benign, which brings down the overall accuracy. Hence, we conclude that the current set of features cannot be used to capture the information present in the ROI accurately.

4.6.2 Scenic Dataset Results

For the COREL dataset images, we compare the results of WAR-BC with four existing works: MILES [44], DD-SVM [49], MI-SVM [50], and k-means SVM [51]. In order to make an accurate comparison, we use the same amount of training/testing data (50/50) as was used for MILES.

4.6.2.1 Associative Classification

Two sets of experiments were performed in MILES: one with 1000 image and a 10-class image dataset, and another with 2000 images and 20-class image dataset. Initially, we experiment with three segment sizes and two combinations of support and confidence values. The corresponding classification results are shown in Table 4.2.

Table 4.2 Classification Accuracy for Different Combinations of Segment Size and Support and Confidence

Sup/Conf combination→ Segment Size↓	Sup=0.01%, Conf=90%	Sup=0.005%, Conf=90%
4x4	78.80%	80.00%
5x5	80.60%	84.00%
7x7	79.00%	81.80%

The segment sizes used in Table 4.2 are 4x4, 5x5, and 7x7, and the support confidence combination for each of these segment sizes are support= 0.01% and confidence= 90% , and support= 0.005% and confidence= 90%. Since window segment 5x5 and the (support, confidence) combination of (0.005%, 90%) provide the best classification accuracy, we use this combination for the rest of our experiments. The experiments presented in the rest of the dissertation are based on this combination of segment size, and support and confidence unless otherwise specified. For the remainder of the dissertation, we will call the experiments using the 1000 image dataset COREL-I and the experiments using the 2000 image dataset COREL-II. Figure 4.11 shows the results for WAR-BC with COREL-I.

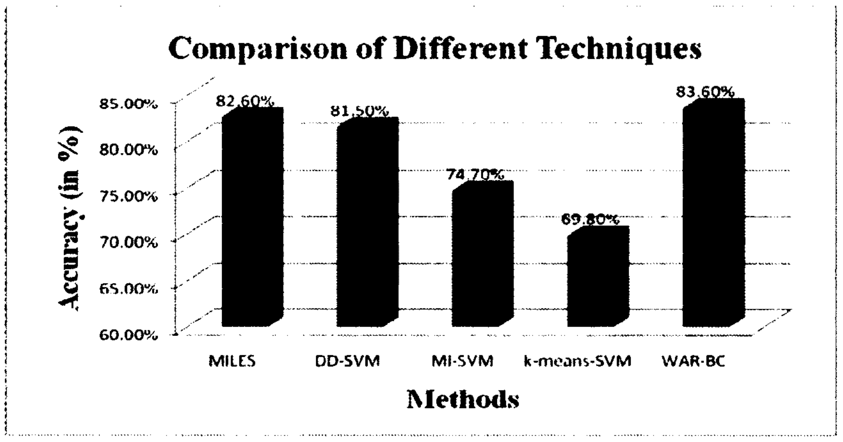


Figure 4.11 Comparison of Different Techniques with WAR-BC

From the results, we can see that the accuracy of WAR-BC is better than the accuracy for the existing techniques. The accuracy is only slightly better using WAR-BC, but we use less features (six) compared to nine (as the six used by WAR-BC and three region-based features) used by MILES, DD-SVM, MI-SVM and k-means SVM. We also investigate the confusion matrix (averaged over five runs) of COREL-I classification using WAR-BC to discover the individual performance of each class. Each row in the matrix represents the percentage of images in a category classified to one of the ten categories. The number on the diagonal shows the percentage of images that were accurately classified and the number in adjacent columns represents the percentage of images in a category that were misclassified into another category. Table 4.3 shows the confusion matrix for COREL-I.

Table 4.3 Confusion Matrix for COREL-I

%	Africa	Beach	Buildings	Buses	Dinosaurs	Elephants	Flowers	Horses	Mountains	Food
Africa	86.4	0.4	2	0.8	0	4	0	1.2	2.8	2.4
Beach	2.4	73.2	1.2	1.2	0	2.4	0.8	0	18.8	0
Buildings	10.4	3.6	68	8	0	4	1.2	0	4	0.8
Buses	0	2	0	92.4	0	2	0	0	2.4	1.2
Dinosaurs	0.8	0	0.4	0	93.6	1.2	0	0	1.2	2.8
Elephants	7.6	2	2	0	0	79.2	0	0.8	7.2	1.2
Flowers	1.6	1.6	0	0	0	0.8	95.2	0	0	0.8
Horses	10.4	0.8	0.8	0.8	0	4	0.8	80.8	0.8	0.8
Mountains	0	16.8	0.4	0	0	2	1.2	0.8	78.8	0
Food	7.2	0.4	0	0.8	0	0.8	0	0	2	88.8

After a close inspection of the confusion matrix, we find that a major misclassification occurs between two classes: Beach and Mountains, 16.8% of images from class Mountains are misclassified to class Beach while 18.8% of images from class Beach are classified to class Mountains. These errors are in-line with the errors reported in [44] and [49]. The possible explanation for this misclassification is that most of the images in these two classes have similar semantic regions like sky, ocean, etc.

Further, class Africa is a major culprit for misclassification. As can be seen, two classes (buildings and horses) have major misclassification with this class. One reason for this behavior is that there is a great deal of intra-class variance in class Africa. This behavior results in a high number of rules generated for this class. Hence, this class absorbs a number of rules during classification. Figure 4.12 shows the classification accuracy of each of the ten classes over five runs.

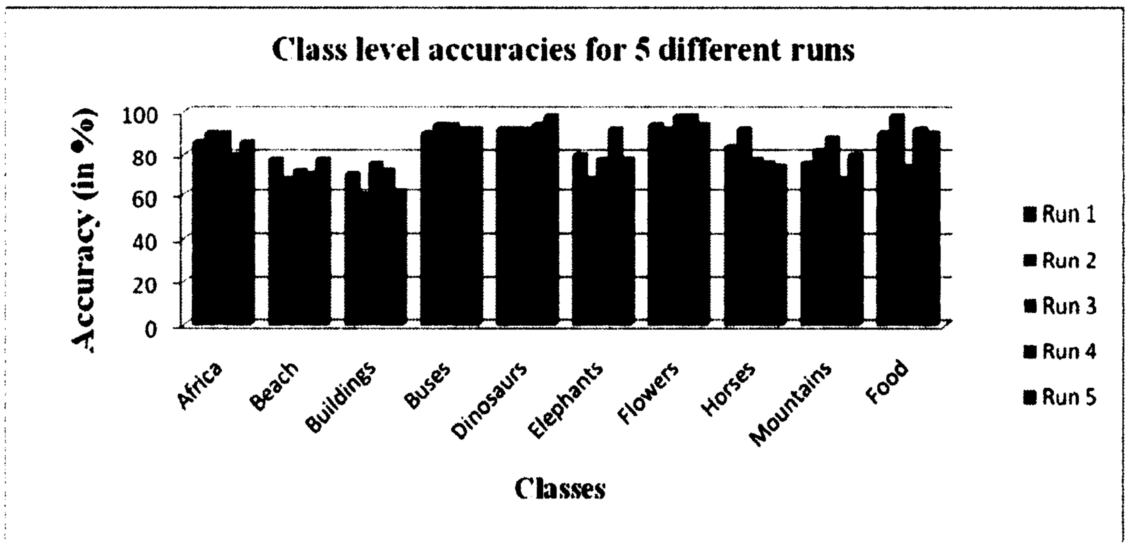


Figure 4.12 Class Level Accuracies of COREL-I over 5 Runs

As can be seen from the graph, the classes with the highest accuracy (Africa, Buses, Dinosaurs, and Flowers) are also the most stable, as the classification rates vary little for these classes. Similarly, the classes with the lowest accuracy are those that have varying classification rates over different runs. This observation validates our hypothesis that the intra-class variance among the images present in these classes results in low classification accuracy. Figure 4.13 shows the classification results for COREL-II dataset.

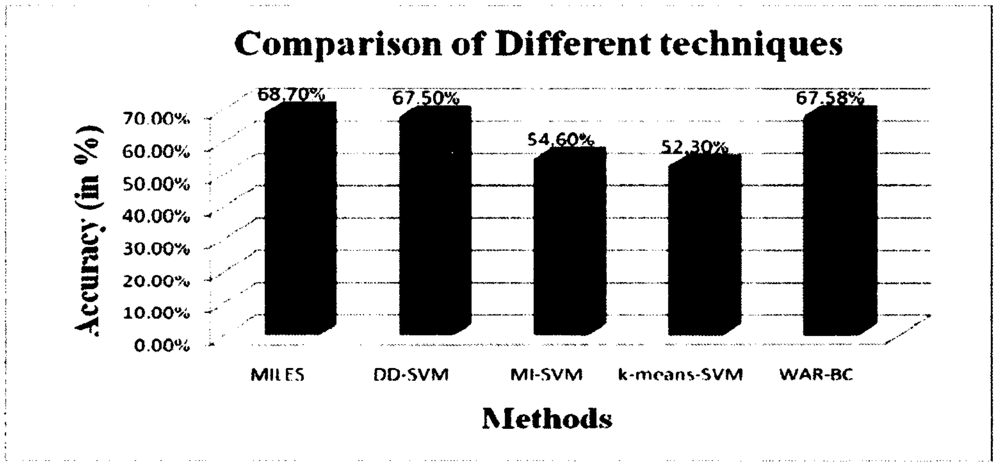


Figure 4.13 Classification Accuracy for COREL-II Dataset

Once again, we see that the classification accuracy for WAR-BC is comparable to the best technique, MILES. As expected, the classification accuracy drops significantly when the number of classes increases, as many of the new classes share some semantic information with the existing ones. Figures 4.14 and 4.15 show the class-level accuracies for COREL-II over five runs.

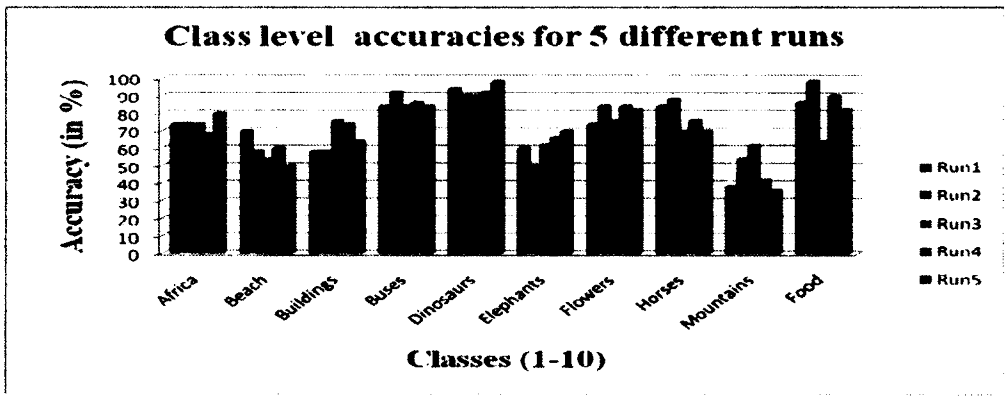


Figure 4.14 Class-Level Accuracies for COREL-II over Five Runs (Classes 1-10)

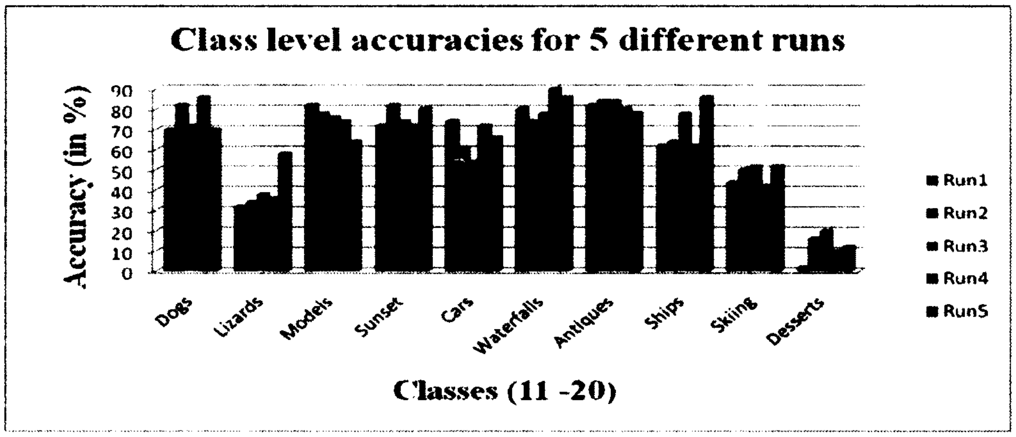


Figure 4.15 Class-Level Accuracies for COREL-II over Five Runs (Classes 11-20)

Again, we see that the classes with the highest average accuracy over different runs are also the most stable classes. It can be seen from the individual class-level accuracies that there are a few classes: Beach, Mountains, Elephants, Lizards, and Desserts, which bring down the overall accuracy. Of these, Class Desert performs the worst. After careful evaluation of the images in this class, we find a high intra-class variation among images. Figure 4.16 shows examples of the images from class Desserts, and we can see that many images share semantic regions with other classes.

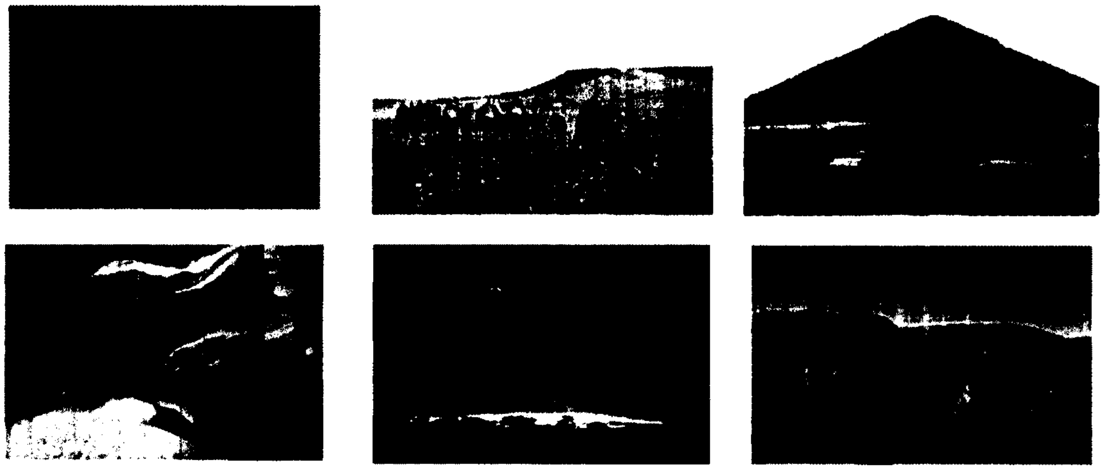


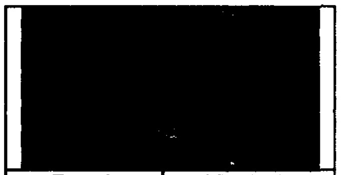
Figure 4.16 Sample Images of Class Desert from COREL Dataset [43]

This semantic similarity is one reason for high misclassification rate for Class Desserts. The output of WAR-BC is a classification score for each class. Figure 4.17 shows example query images that have been classified correctly into their respective classes. The first column shows the class label and the second column shows the corresponding score generated by WAR-BC for that class.

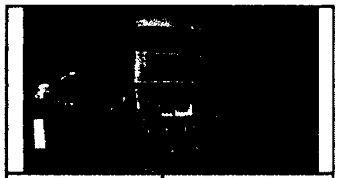
Africa	**4419.495**
Food	3359.502
Buses	3072.004

Africa	**3661.853**
Food	3630.555
Buses	2611.479

Beach	**4601.331**
Mountains	3705.558
Buses	3347.623

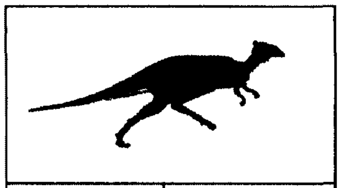


Beach	**2706.122**
Mountains	2676.830
Elephants	2577.635

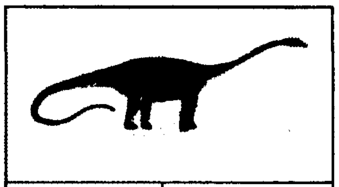


Buses	**2906.740**
Beach	2863.032
Buildings	2769.600

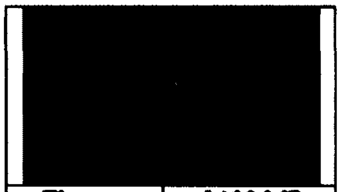
Buses	**2663.406**
Buildings	1960.447
Mountains	1842.509



Dinosaurs	**1349.739**
Food	1145.882
Africa	954.710



Dinosaurs	**1785.943**
Food	1152.08
Africa	1051.148



Flowers	**3466.067**
Africa	2143.862
Food	1906.092

Flowers	**3805.264**
Africa	3766.070
Elephants	3138.711

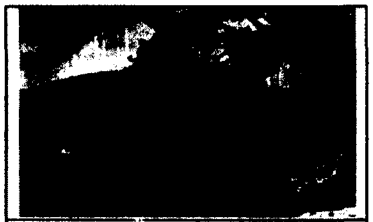
Food	**3494.866**
Africa	2816.378
Flowers	2225.199

Food	**2299.883**
Africa	1842.104
Flowers	1677.349

Figure 4.17 Examples of Accurately Classified Query Images

The class label assigned in this set of experiments is the class with the highest score (shown in bold). All the images in Figure 4.17 were classified accurately to their base classes. Figure 4.18 shows some images that were misclassified. The class label assigned to these query images is the top most class label. The correct class label for each of these images is shown in bold.

Mountains	2506.099
Beach	**2226.481**
Buildings	1275.873

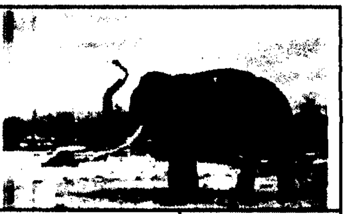


Mountains	3227.081
Beach	**3211.962**
Buildings	2651.056

Buses	2671.305
Buildings	**2539.044**
Beach	2059.232

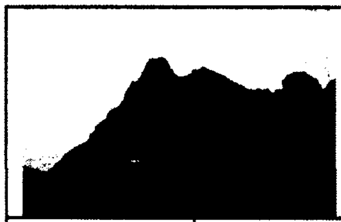
Buses	3891.169
Africa	3844.424
Buildings	**3076.835**

Africa	2037.983
Beach	1867.822
Elephants	**1831.629**



Africa	3611.602
Elephants	**3593.019**
Mountains	3207.599

Beach	3962.903
Mountains	**3548.362**
Buses	3361.031



Beach	1625.409
Mountains	**1321.558**
Africa	952.053

Beach	3177.609
Mountains	**3098.840**
Elephants	2667.880

Figure 4.18 Examples of Inaccurately Classified Query Images

It can be seen from the misclassified query images of classes Mountain and Beach that these images do share some semantic regions such as sky, water, and sand. This semantic similarity, as explained earlier in the section, is the major reason for misclassification. Further, we also see that the true class label of most misclassified images appears in the top three scores provided here. This observation leads us to focus on another set of experiments explained in the next section.

4.6.2.2 Classification With Top-k Classes (WAR-BCk)

The experiments performed in the last section assign a query image with a label of the class with the highest score (recall the classification formula $ClassLabel \Leftarrow \arg\max_{j=1}^{C}(Tot_j))$. This type of classification can be called crisp classification, as only one class is assigned to a query image. However, as discussed in the last section, many images share some semantic regions. This semantic similarity leads to similarity in the type of rules for these classes, which in turn, leads to an increase in the classification score for such similar classes. This increased classification score is one reason for misclassification. For example, an image from class Mountains might be categorized to class Skiing because they both contain a mountainous region or the sky.

As a result, providing one label to an image may not be adequate, and crisp classification in such cases might not be always correct. Therefore, we perform another set of experiments in which we assign multiple labels to the query images instead of assigning only the single class label with the highest score. We call this type of classification "classification with top-k classes." The scores of WAR-BC for each class are sorted in the decreasing order, and the class with the top-k highest scores is assigned as possible class labels. We use the same training and testing splits as used in the last section for both COREL-I and COREL-II. We perform experiments with top-3 (WAR-

BC3) and top-5 classes (WAR-BC5). For the COREL-I dataset, the average classification accuracy increases to 96% with WAR-BC3 and to 99.16% with WAR-BC5. Figure 4.19 and Figure 4.20 show the classification accuracy of WAR-BC compared with WAR-BC3 and WAR-BC5 over five splits of data for the COREL-I dataset and the COREL-II dataset, respectively.

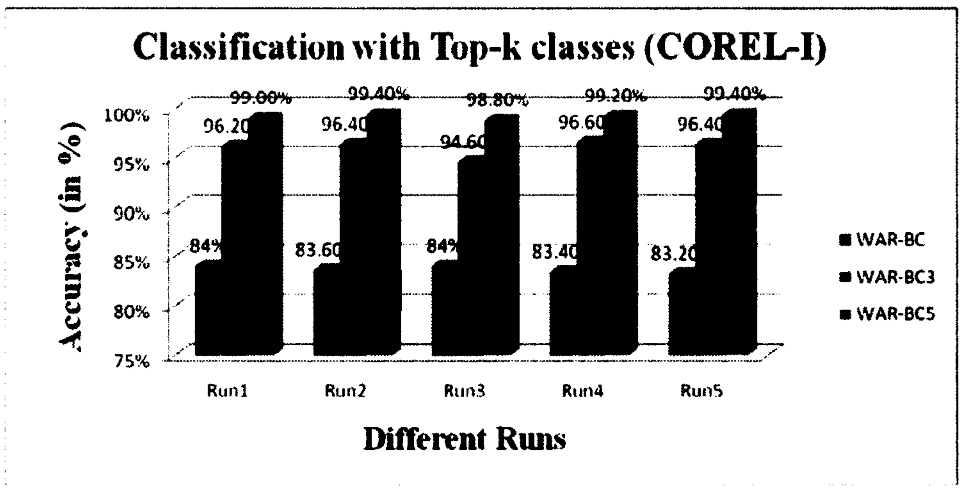


Figure 4.19 Classification with Top-3 and Top-5 Classes over Five Runs

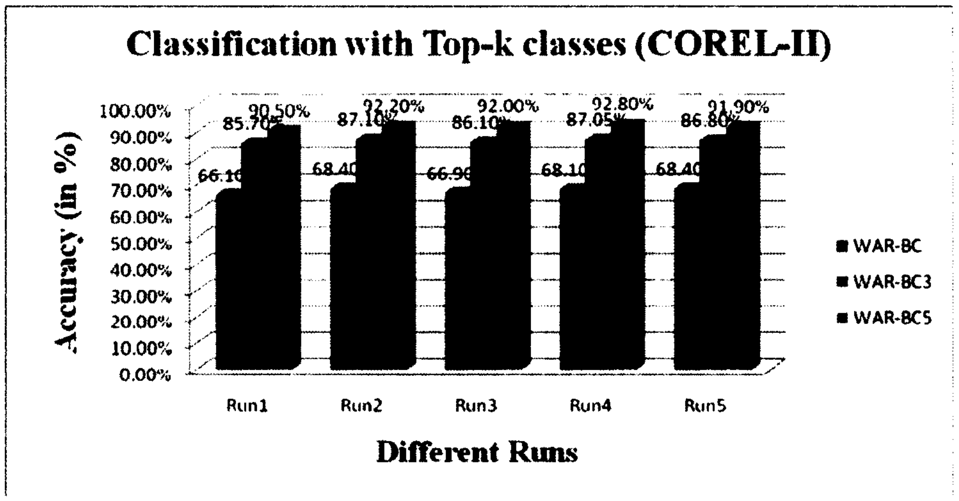
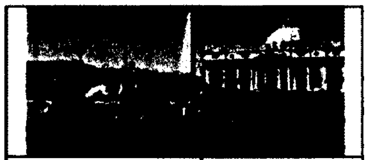


Figure 4.20 Classification with Top-3 and Top-5 Classes for COREL-II

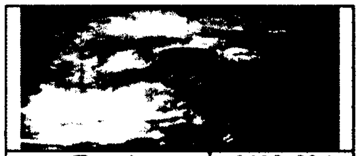
It is evident from the results that once again our algorithm is stable for all the runs of data, as the accuracy does not deviate much from the average classification accuracy for the COREL-I and COREL-II datasets. As expected, the classification accuracy increases for both the top-3 and the top-5 classes. This increased accuracy is due to images, which would be misclassified using only the top class label, as they share semantic regions, but the real class label might appear in the top-k classes. Figure 4.21 shows the images that were misclassified using only the top label but were classified accurately using the top-3 labels.



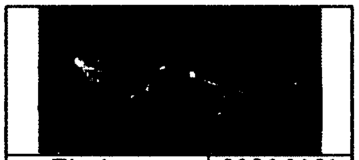
Buses	2847.1364
Buildings	**2643.9307**
Ships	2309.0274

Dogs	5202.328
Models	4334.845
Africa	**4279.832**

Waterfalls	2007.246
Mountains	1958.146
Dogs	**1634.928**



Beach	6413.524
Waterfalls	5798.765
Dogs	**5671.081**



Elephants	2056.0151
Cars	2019.1409
Lizards	**1991.1185**

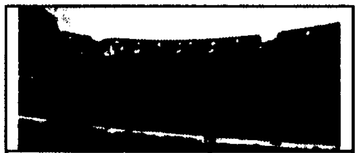
Dogs	3237.9402
Lizards	**3165.9201**
Africa	2905.8142

Mountains	2808.0851
Waterfalls	2741.0494
Beach	**2648.7578**

Mountains	2498.893
Waterfalls	2467.8077
Beach	**2417.1565**

Dogs	4934.5877
Elephants	**4914.0357**
Africa	4821.3894

Ships	2909.7916
Desserts	**2792.5467**
Skiing	2683.8771



Waterfalls	2971.9063
Buildings	**2928.0153**
Africa	2787.6748

Beach	3995.6789
Waterfalls	3522.1906
Mountains	**3452.372**

Figure 4.21 Correctly Classified Images with Top-3 Labels

Figure 4.22 shows the images, which were misclassified using both the top class label and top-3 class labels, but were classified accurately using the top-5 classes.

Mountains	3274.9542
Waterfalls	3204.5366
Beach	3159.9684
Africa	**3081.0648**
Cars	3052.7322

Buses	4052.4303
Cars	3952.8409
Antiques	3746.1479
Buildings	**3642.0979**
Beach	3626.6663

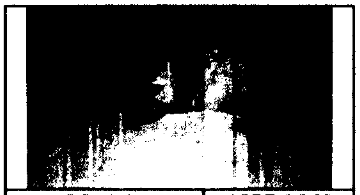
Dogs	2241.9605
Desserts	2137.4992
Buildings	2099.4626
Elephants	**2093.2415**
Waterfalls	1914.9168

Beach	4538.4324
Waterfalls	4468.7564
Ships	4414.1495
Mountains	**4143.224**
Buildings	3978.7322

Elephants	2938.7076
Cars	2841.0189
Africa	2820.702
Buildings	**2818.9413**
Waterfalls	2634.3732

Waterfalls	4039.5236
Beach	3917.8572
Ships	3490.2595
Elephants	3337.4867
Mountains	**3307.907**

Waterfalls	1668.6634
Beach	1559.1128
Dogs	1257.0841
Mountains	**1252.8287**
Cars	1106.884



Models	4577.9568
Flowers	4469.7799
Waterfalls	4374.9349
Dogs	**4374.9325**
Desserts	4370.7741

Dogs	3250.2518
Horse	2974.9747
Elephants	2715.7592
Lizards	**2634.9352**
Antiques	2596.4805

Cars	3385.1634
Food	3149.8028
Sunset	3139.0264
Lizards	**3071.0322**
Desserts	2856.6584

Buildings	1621.8011
Elephants	1338.7745
Horses	1308.8916
Beach	**1244.4448**
Africa	1231.5664

Ships	3407.3938
Beach	3247.0554
Skiing	3234.8804
Dogs	3037.1283
Desserts	**2989.5472**

Figure 4.22 Correctly Classified Images with Top-5 Labels

4.7 How WAR-BC Addresses the Issues Faced by Associative Classifiers

Thabtah introduced interesting limitations that plague state of the art AC algorithms. Our proposed solution addresses most of these issues [42]:

Multi-Label Rules Classifier: Most existing classifiers are single label classifiers as they only provide the most obvious class correlated to the data and ignore the other classes correlated with these rules. This problem stems from the data representation and the type of rules (CARs) used by the AC algorithms. We address this problem by finding non-class constrained rules and then associating the frequency of these rules in each class. WAR-BC calculates scores for each class and classifies the testing instance to the class with the highest score. However, we can extend this single label classification to multi-label classification, in which the top-k labels are used for classification.

Rule Ranking: Rule ranking, also known as rule sorting according to some defined criteria, plays an important role in classifying AC algorithms. Most algorithms take into account only the support, confidence, and cardinality of the rules. This low threshold of information is not considered a sound reasoning, as these values are not a proper representation of the importance of a rule for some classes. WAR-BC addresses this issue by using the support, confidence, and cardinality information and the inter-class and intra-class based frequency measures of the rules.

Incremental Learning: Due to their inherent reliability for using class-constrained rules, most of the existing AC algorithms mine the training datasets as a whole to find associations. Adding a new data instance (e.g. a new image) means mining the dataset again, wasting time and resources. We address this issue by representing each instance as a separate dataset instead of the class constrained representation. Therefore, if a new data instance is added to the dataset, then we only need to extract the rules for this instance and update the vertical weight for the class to which the new instance belongs.

Rules Overlapping: Many classic rule-based classification algorithms build the classifier heuristically so that once a rule is evaluated during training, all the training objects covered by it are discarded. This technique is not a good method of training, as objects might be associated with more than one rule (especially true in case of multi-label classification). In our methodology, we keep all the data and we weigh the rules according to their intra-class and inter-class presence.

4.8 Conclusions

The method of using association rules for classification has intrigued researchers in data mining, due to the ability of the rules to portray inherent capacity to capture causal relationships among underlying data. The past decade has seen a sharp increase in the applicability of these associative relationships for supervised learning tasks like classification. Despite the inherent success of the myriad of associative classification algorithms developed, there is still a significant room for improvement as most of the algorithms suffer several limitations, some of which we have discussed in this chapter. Some of the features, such as multi-label classification, incremental learning, and intra-class and inter-class pattern weighting are necessities for all classifiers, whether they are a regular classifier or an associative classifier. Therefore, it becomes imperative that any new classifier developed should be able to address them

In this work, we have proposed a novel associative classification framework which uses the intra-class and inter-class rule presence to weight the rules. The proposed algorithm uses non-class constrained association rules, which boosts classification accuracy. Exhaustive experimentation and comparative evaluation with existing works over two dataset from completely different domains show that our algorithm is far superior to the existing works for most of the cases. We also show that our algorithm can perform both crisp and fuzzy classification without any major modifications. In respect to image classification, fuzzy classification does make sense as there is no benchmark for image semantics and many image types share semantic regions. Further, as explained in Section 4.7 the proposed algorithm is able to address most of the issues that plague associative classifiers.

CHAPTER 5

TEMPORAL PATTERN MINING

5.1 Introduction

The frequent itemset mining problem was first introduced almost two decades ago. Since that time, a significant amount of research has focused on developing better algorithms. One addition to frequent itemset mining has been the advent of inter-transaction itemset mining, which is only a few years old. Researchers are still looking for better algorithms for mining inter-transaction itemsets. Unlike the intra-transaction itemset mining whereby the most extensively used methods are Apriori and FP-Growth, there is no consensus on what the best algorithm for inter-transaction itemset mining is. As explained in Section 1.3.1, most of the frequent-itemset mining approaches find a pattern based on time of occurrence (i.e. events that happen at the same time), and that is why these types of patterns/rules are known as *intra-transaction* patterns/association rules. Inter-transaction pattern/rule mining generates temporal/time-stamped patterns from time series data. Meteorological data and stock market data are two examples of dynamic or temporal data, data that changes with time.

5.2 Problem Definition

In Section 5.2, we formally introduce the problem of inter-transaction association rule mining.

Definition 5.1 Let $E = \{e_1, e_2, ..., e_N\}$ be the set of items and *I* be a dimensional attribute in which *Dom(I)* represents the domain of *I*. A transaction database *D* consists of a set of transactions in the form of (t, S), where $t \in Dom(I)$ and $S \subseteq E$.

The dimensional attribute *I* describes the properties associated with items, time and space, for example. Depending on the application used, a dimensional attribute can be divided into *m* equal parts. For instance, time can be divided into hours, days, weeks, or months.

An association is said to span across *k* intervals if it associates items that are *k* intervals apart. Since finding all such associations is laborious and since rules separated by large intervals might not be interesting, a window called maxspan (w) is introduced to look at future intervals. Therefore, at any given time, there are exactly *w* transactions in the maxspan window. Only rules that are separated by a maximum of *w* intervals are then mined.

Definition 5.2 Given *E* and window *W* with *w* transactions $\{t_0, t_1, ..., t_{w-1}\}$, the set of extended items E' is defined as $E' = \{e_i(j)\} \mid e_i \in t_j, t_j \subseteq E, \forall 1 \le i \le N, 0 \le j \le w-1$. Therefore, given *w* and *E*, the complete set of extended items will be $E' = \{e_1(0), e_1(1), ..., e_1(w-1), e_2(0), ..., e_2(w-1), ..., e_N(w-1)\}$.

Definition 5.3 Given E', an inter-transaction association rule is an implication of the form $A \Rightarrow B$, where

1. $A \subseteq E', B \subseteq E'$, 2. $\exists e_i(0) \in A, 1 \le i \le N$,

3. $\exists e_i(j) \in B, 1 \le i \le N, j \ne 0$, and 4. $A \cap B = \phi$.

For example an inter-transaction association rule could be of the form $a(0), b(1), c(2)$ or $a(0), b(1), c(1)$.

Definition 5.4 If M is the number of transactions in database ($|D| = M$), $A \Rightarrow B$ is the inter-transaction association rule, Σ_{AB} is the set of sliding windows W that contain $A \cup B$, and Σ_A is the set of windows which contain only A, the support and confidence of the inter-transaction association rule $A \Rightarrow B$ is defined as:

$Support\ (A \Rightarrow B) = \frac{|\Sigma_{AB}|}{M}$ (Equation 5.1) $Confidence(A \Rightarrow B) = \frac{|\Sigma_{AB}|}{|\Sigma_A|}$ (Equation 5.2)

Inter-transaction itemsets, which have a support higher than the user defined minimum support, are known as frequent inter-transaction itemsets. Given the minimum support and confidence, the inter-transaction association rule-mining problem is to find all such frequent association rules in the database.

5.3 Issues with Current Techniques and Proposed Solution

In Chapter 2, we explained the current state of the art methods for inter-transaction association rule mining algorithms. Below we present some limitations of these algorithms and propose our own novel solution. One of the biggest problems that current inter-transaction rule mining methods face is the sliding window-based mega-transaction database format for inter-transaction rule mining. Below we list some of the problems caused by this format.

1) The dimensionality of the new mega-transaction-based database will increase with windows size, (e.g. assuming ten dimensions in the original database and the window size of five, in the worst-case scenario there would be 10x5 = 50 new

attributes/dimensions for the mega-transaction database). Anyone who has used association rule mining knows that the execution time will skyrocket with the increase in the number of dimensions.

2) The optimum overlap for the sliding window is time consuming to obtain. In order to make sure that one does not miss any inter-transaction rules and for accurate *support* and *confidence* calculations, the window should be slid at a maximum of only one time point/transaction per overlap. This restraint further adds to the execution time.
3) Which window size is the best? Can the window size be changed later? These issues bring another overhead for the algorithm, because, if at any moment the user feels like he wants to change the window size, for instance, in order to find rules covering ten days between them instead of five days, the mega-transaction database has to be rebuilt from scratch, and rule mining has to be performed again.
4) Finally, the existing methods lack the functionality of providing the location specific information for the rules. This information could be used to check the accuracy of the rules extracted. Further, this information could also tell the user whether a rule is localized to a certain part of the dataset or whether it reoccurs after short intervals.

PROWL was the first algorithm to our knowledge which did not use this mega transaction format. In PROWL, the PWL (projected window list) generation method extends the technique from single events to multiple events at a time point by generating multiple PWLs. The method stores the IDs of every event in a list, called the time list or the window list, in which each ID represents the occurrence time slot of an event. Then, for each frequent 1-event, a PWL is generated by adding one to the pattern time list. This PWL is then mined for frequent events, and the new frequent inter-transaction event is formed by the current pattern and the frequent events from the new PWL.

While PROWL does not use the mega-transaction format, and it can provide the exact location of patterns, it has some limitations. The PWL method for generating itemsets is unidirectional. For two itemsets e_1 and e_2, the PWLs of e_1 can only give itemsets of the form e_1 followed by e_2. In order to find itemsets of the form e_2 followed by e_1, the PWL procedure has to be repeated with e_2. Since PROWL uses the DFS method to generate all the itemsets between e_1 and e_2, we need to generate all the PWLs up to the maxspan level for both e_1 and e_2. Therefore, although an itemset of the form $e_1(0)e_2(1), e_1(0)e_2(2)$ may not exist, we need to generate $IDlist\{e_1,*\}$ and $IDlist\{e_1,*,*\}$ in order to generate itemsets of the form $e_1(0)e_2(3)$. This phantom IDlist generation becomes an even bigger issue as the maxspan level and number of items increase. Additionally, PROWL lacks the ability to generate inter-transaction itemsets in which more than one event is present at the same time, (e.g. $e_1(0)e_2(0)e_3(1)$ or $e_1(0)e_2(1)e_3(1)$). These types of itemsets are formed by the intra-transaction join in FITI. This issue arises because PROWL only allows one item to be present in the PWL of a base event, while making an intersection of the IDs. This issue becomes more severe when the maxspan size is less than the number of events in the database. For example, if the number of events is ten and the maxspan is five, PROWL can only associate a maximum of five events at once for making inter-transaction itemsets. Finally, PROWL cannot generate patterns of varying lengths, (i.e. temporal patterns).

While ITPMine does not suffer from the same problems that PROWL does, it has limitations. First, since ITPMine uses the sliding window format to get the dats for 1-itemsets and hashing candidate 2-itemsets, any change in the maxspan means that the database will have to be scanned again and, at the very least, additional itemsets will have to be hashed again. Second, unlike FITI, which uses the intra-transaction rule mining to find frequent intra-transaction itemsets, ITPMine does not have any prior knowledge of the frequent 1-itemsets while hashing the candidate 2-itemsets during the first scan. This hashing mechanism is more similar to EH-Apriori than FITI. Therefore, ITPMine could make combinations of 1-itemsets that might not be frequent. This possibility adds to the computation time. Finally, the hashing function used for ITPMine $h = ((u * (maxspan + 1) + w) * N1 + v) \bmod Hashsize$ is misleading. In many cases, more than one itemset is hashed to a similar hash address. We will explain this problem using the following example: Let there be ten items, each having a unique ID from 1 to 10, maxspan=5; let u and v represent two items, and let w represent the lag or time point difference between the first and second item. Then, using the Hashsize formula for FITI the Hashsize will be $5 \times 10^2 = 500$.Using the hashing function, the hash address for $u(0)v(2)$ with u=9 and v=6 will be $h = ((9 * (5+1) + 2) * 10 + 6) \bmod 500 = 566 \bmod 500$ $= 66$, and the hash address for $u(0)v(0)$ with u=1 and v=6 will be $h = ((1 * (5+1) + 2) * 10 + 6) \bmod 500 = 66 \bmod 500 = 66$. Given these addresses, both itemsets 9(0)6(2) and 1(0)6(0) will be hashed to the same address of 66. We can see that, although the hashing function used by ITPMine is similar to FITI, there is one major difference between the systems. FITI does not allow the lag or the time difference between two items to be zero while hashing, (i.e. $w \neq 0$, intra-transaction itemsets of the form a(0)b(0), a(0)c(0), etc.).

However, ITPMine does allow a zero time difference between two items, written as $0 \leq w \leq maxspan$. This small difference in the conditions leads to problems since more than one itemset shares the same hash address.

The success of PROWL and ITPMine leads us to believe that an inter-transaction rule-mining algorithm will benefit by using a vertical data format rather than a horizontal one. The limitations of current techniques described earlier have motivated us to focus on an improved inter-transaction rule-mining algorithm. In this research, we propose DMITAR (Difference Matrix-based Inter-Transaction Association Rule Miner). While most inter-transaction algorithms use the horizontal data format (transaction data), PROWL uses the vertical data format (IDs of events). In the proposed algorithm, we use the vertical data format of PROWL and develop a windowless, ID based itemset representation, which alleviates all the issues explained for PROWL and other window based approaches described earlier in this section. Our proposed algorithm uses the ID information to generate difference lists for two events, which are then used to extract inter-transaction itemsets.

5.4 Datasets Used

We have finalized three multi-variate time series datasets, stock market dataset, weather dataset, and synthetic dataset, to evaluate our proposed method.

Stock Data [52]: We gather the closing price of eight stocks (Google, Yahoo, Apple, Microsoft, IBM, HP, Dell, and Intel) for the 2008-2009 year from the Yahoo Finance website. Our dataset includes data from 253 trading days in the 2008-2009 year. Hence, the data matrix is 253x8 (rows x columns), with rows representing the trading days and columns representing one of the eight stocks.

Weather Data [53]: The U.S. Department of Commerce National Climactic Data Center provides free, unedited, daily, and historical data for more than 700 locations in the U.S. We have finalized the use of the data for New Orleans for the past two years (January 2007 to January 2009) observed at the Louis Armstrong International Airport. There are several variables for this data, but we only used the six most informative ones: average temperature (in Fahrenheit), dewpoint, wetbulb, station pressure, sea level pressure, and average wind speed (in mph).

Synthetic Data [54]: Finally, we use synthetic data for our evaluation. For this purpose, we use the CRU weather generator. The CRU weather generator is a stochastic weather data generator, which uses the Markov chain model to generate 30 years of daily data for 11 UK sites. This model does not take Leap years into account, and every year has 365 days. We will use multiple year data for the "Heathrow" site. For more information regarding CRU the readers, please see the CRU documentation [55]. CRU provides several weather variables, but we use the six most informative ones: maximum temperature (degrees C), vapor pressure (hecta pascals), relative humidity (%), wind speed (ms^{-1}), sunshine (hrs.), and reference potential evapotranspiration ($mm day^{-1}$).

5.5 Proposed Methodology

Our proposed inter-transaction association rule-mining algorithm is divided into three parts: Frequent-1 itemset generation, Frequent-2 itemset generation, and Frequent-k itemset generation (k >2).

5.5.1 Frequent 1-Itemset Generation

In order to form the 1-itemsets, the original database is scanned once, and an IDList is formed for each event. The entries in an IDList represent the IDs of the rows in which the 1-event happens. For example, the IDList of event $\{a\}$ will list all the rows where $\{a\}$ occurs.

Definition 5.6 Give a set of items $E = \{e_1, e_2, ..., e_N\}$ and a transaction database D in the form of (T_i, S_i), where $|D| = M$, $1 \le T_i \le M$, and $S \subseteq E$. The IDList of an event e_j, where $j \le N$ can be defined as $IDList(e_j) = \{T_i \mid 1 \le T_i \le N, e_j \in S_i\}$.

Using the above definition and the example database of Figure 5.1, we can find the IDLists for {a, b, c, d, e}.

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

Figure 5.1 Example of a Temporal Database

The respective IDLists are:

$IDList(a) = \{1,3,4,7,9,10\}, IDList(b) = \{2,4,6,7,9,10\}, IDList(c) = \{2,3,5,6,8,9,10\}$, $IDList(d) = \{1,2,5,6,7,8,9\}$, and $IDList(e) = \{1,3,5,6,8,10\}$. Now, the support of each itemset is the length of the IDList.

Definition 5.7 Given a 1-itemset e_j and its $IDList\{e_j\}$, we define the support of e_j as the length of its *IDList*, $Sup(e_j) = |IDList\{e_j\}|$.

Therefore, the support of *a, b, c, d,* and *e* are 6, 6, 7, 7, and 6, respectively. Let the minimum support *(minsup)* = 2. The frequent 1-itemsets can be found by using the following formula: e_j is frequent, if $Sup(e_j) \geq min\_sup$. Using this formula, we discover that the frequent 1-itemsets are *{a, b, c, d,* and *e}*. For clarity, we call the 1-itemsets *atomic itemsets* and k-itemsets $(k \geq 2)$ *composite itemsets* in the rest of the paper. A *k*-itemset is composed of *k* atomic itemsets; for example, a composite itemset a(0)b(1)c(2) consists of the atomic itemsets a, b, and c. For the remainder of the paper, we use the following terminology regarding the IDLists: $HIDList =$ height of IDList = number of rows in the IDList, and $|IDList\{e_j\}| =$ length of the IDList= number of columns in the IDList. At any given level *k* of frequent itemsets, the height of the IDList =*k*. The *k*th row represents the IDs of the transaction where the *k*th event happens. $Set\{Freq-k\}$ stores the set of all the itemsets of level *k*, and $List\{Freq-k\}$ store their corresponding IDLists. Once the itemsets of level *k* and their corresponding IDLists are stored, we can generate inter-transaction frequent *k*-itemsets $(k \geq 2)$ from these intra-transaction frequent 1-itemsets.

5.5.2 Frequent 2-Itemset generation

Since inter-transaction patterns are temporal patterns, there are only three possible relations between two items (say *a* and *b*): *a* happens before *b*, *a* happens with *b*, or *a* happens after *b*. We introduce a new data structure called the DIFFerence Matrix (DIFFMat) which captures and stores information about all these possible relations. Using the frequent 1-itemset ID lists, we generate the difference matrices for all pairs of frequent 1-itemsets. This pairing of frequent 1-itemsets is similar to the Apriori principle of 2-itemset generation. However, unlike PROWL and ITPMine, the DIFFMat mechanism is bidirectional (i.e. one DIFFMat is sufficient to generate all *a* followed by *b* and *b* followed by *a* patterns). The IDs of one itemset form the column indices/labels, and the IDs of the other itemset form the row indices/labels. Then, each cell in a matrix (q,r) is calculated by taking the difference of the current row label *(Rlabel(q))* from the current column label *(Clabel(r))*.

Defnition 5.8. Given $Set\{Freq-1\}$ and $List\{Freq-1\}$, the difference matrix for two frequent *atomic* items e_i and e_j is represented as $DFMat\{e_i,e_j\}$. A cell (q,r) in the difference matrix represents the value for *q*th row and *r*th column where:

1. $DFMat\{e_i,e_j\}(q,r) = IDlist\{e_j\}(r) - IDlist\{e_i\}(q)$,
2. $1 \le i \le I, 1 \le j \le I, i \ne j$, and
3. $q \le m \mid m = |IDlist\{e_i\}|$, and $r \le n \mid n = |IDlist\{e_j\}|$.

Here, $IDlist\{e_k\}(l)$ represents the l^{th} value in the IDList of atomic item e_k. Therefore, a DIFFMat between two items e_i and e_j will be an *m* row and *n* column matrix, where *m* is the number of IDs in the IDList of e_i and *n* is the number of IDs in IDList of e_j. Using the above definition, we generate the difference matrix for all possible pairs of frequent 1-itemsets. Since, as stated earlier, a single DIFFMat is sufficient to find all possible relations between two items, then, if $DFMat\{e_i,e_j\}$ has been formed, there is no need to form $DFMat\{e_j,e_i\}$. We explain the reason for this scenario later in this section. Therefore, if there are N different frequent atomic items in the database $e_1, e_2, ..., e_N\}$, the number of difference matrices formed will be $N(N-1)/2$. Hence, the number of difference matrices formed from the frequent 1-itemsets from Section 5.5.1 is $5(5-1)/2 = 10$. The difference matrices are $DFMat\{a,b\}, DFMat\{a,c\}, DFMat\{a,d\}$, $DFMat\{a,e\}, DFMat\{b,c\}, DFMat\{b,d\}, DFMat\{b,e\}, DFMat\{c,d\}, DFMat\{c,e\}$, and $DFMat\{d,e\}$. Four of the ten difference matrices are represented in Figures 5.2 – 5.5.

Labels b→ a↓	**2**	**4**	**6**	**7**	**9**	**10**
1	1	3	5	6	8	9
3	-1	1	3	4	6	7
4	-2	0	2	3	5	6
7	-5	-3	-1	0	2	3
9	-7	-5	-3	-2	0	1
10	-8	-6	-4	-3	-1	0

Figure 5.2 Difference Matrix $DIFFMat\{a,b\}$

Labels c→ a↓	**2**	**3**	**5**	**6**	**8**	**9**	**10**
1	1	2	4	5	7	8	9
3	-1	0	2	3	5	6	7
4	-2	-1	1	2	4	5	6
7	-5	-4	-2	-1	1	2	3
9	-7	-6	-4	-3	-1	0	1
10	-8	-7	-5	-4	-2	-1	0

Figure 5.3 Difference Matrix $DIFFMat\{a,c\}$

Labels c→ b↓	**2**	**3**	**5**	**6**	**8**	**9**	**10**
2	0	1	3	4	6	7	8
4	-2	-1	1	2	4	5	6
6	-4	-3	-1	0	2	3	4
7	-5	-4	-2	-1	1	2	3
9	-7	-6	-4	-3	-1	0	1
10	-8	-7	-5	-4	-2	-1	0

Figure 5.4 Difference Matrix $DIFFMat\{b,c\}$

Labels d→ c↓	**1**	**2**	**5**	**7**	**8**	**9**
2	-1	0	3	5	6	7
3	-2	-1	2	4	5	6
5	-4	-3	0	2	3	4
6	-5	-4	-1	1	2	3
8	-7	-6	-3	-1	0	1
9	-8	-7	-4	-2	-1	0
10	-9	-8	-5	-3	-2	-1

Figure 5.5 Difference Matrix $DIFFMat\{c,d\}$

Three types of values are present in every DIFFMat: positive, negative, and zero. From this point forward, we refer to these types of values as lags (l). Positive values from $DFMat\{a,b\}$ indicate that a happens, and then b happens; negative values indicate that first b happens, then a happens, and zero indicates that a and b happen at the same time. Each difference value/lag represents the exact time difference between the occurrences of two events. From each DIFFmat, we find the different lags and their support (number of times the lag appears in the DIFFMat). For example, in Figure 5.2 ($DFMat\{a,b\}$), we find that the different lags are $l_1 = 0, l_2 = 1, l_3 = 2, l_4 = 3, l_5 = 4, l_6 = 5, l_7 = 6, l_8 = 7, l_9 = 8, l_{10} = 9, l_{11} = -1, l_{12} = -2,$ $l_{13} = -3, l_{14} = -4,$ $l_{15} = -5, l_{16} = -6, l_{17} = -7, \text{and}\, l_{18} = -8$, and their corresponding support are $|l_1| = 4, |l_2| = 2, |l_3| = 2, |l_4| = 4, |l_5| = 1, |l_6| = 2, |l_7| = 3, |l_8| = 1, |l_9| = 1,$ $|l_{10}| = 1, |l_{11}| = 3, |l_{12}| = 2,$ $|l_{13}| = 3, |l_{14}| = 1, |l_{15}| = 2, |l_{16}| = 1, |l_{17}| = 1, \text{and}$ $|l_{18}| = 1$, respectively.

Once we have the lags, we can find out what each of these lags represents. Lag $l_1 = 0$ represents an occurrence of a followed by b with a lag of zero. In other words, a and b happen at the same time (intra-transaction itemset, $a(0)b(0)$). The pattern repeats itself four times in the database (support of this pattern is four, $|l_1| = 4$). Lags from l_2 to l_{10} represent positive values. Therefore, these lags represent itemsets of the form "when a happens, then b happens l_k time points later $(a(0)b(l_k))$". Lag l_2 indicates that when a happens, then b happens one time point later $(a(0)b(1))$. This pattern repeats twice in the database ($|l_2| = 2$). Similarly l_3 indicates that when a happens, then b happens two time points later $(a(0)b(2))$, and l_{10} indicates that when a happens, then b happens nine time points later $(a(0)b(9))$. Finally, the lags from l_{11} to l_{18} represent negative values. These lags tell us that b has happened before a or that when b happens, a happens after a lag of l_k time points $(b(0)a(l_k))$. Lag l_{11} indicates that when b happens, then a happens one time point later $(b(0)a(1))$. This pattern repeats three times in database ($|l_{11}| = 3$). Similarly, l_{12} indicates that when b happens, then a happens two time points later $(b(0)a(2))$; l_{18} indicates that when b happens, then a happens eight time points later $(b(0)a(8))$. As can be seen, both the a followed by b and b followed by a patterns can be formed from a single difference matrix $DFMat\{a,b\}$.Therefore, we do not need to make another matrix $DFMat\{b,a\}$. In the same way, we can find the 2-itemsets from all the difference matrices.

Definition 5.9 Given a difference matrix $DFMat\{e_i,e_j\}$, let Σ_{lag} represent all the unique lags in the difference matrix, then a 2-itemset is defined as:

1. $e_i(0)e_j(l_k), \forall l_k \in \Sigma_{lag}$ if $l_k \geq 0$ and
2. $e_j(0)e_i(abs(l_k)), \forall l_k \in \Sigma_{lag}$ if $l_k < 0$.

Therefore, using the above definition, the possible 2-itemsets for $DFMat\{a,b\}$ are: $a(0)b(0), a(0)b(1), a(0)b(2), a(0)b(3), a(0)b(4), a(0)b(5), a(0)b(6), a(0)b(7), a(0)b(8),$ $a(0)b(9), b(0)a(1), b(0)a(2), b(0)a(3), b(0)a(4), b(0)a(5), b(0)a(6), b(0)a(7),$ and $b(0)a(8)$. As can be seen in the equations above, we can find all the possible 2-itemsets from this combination. The algorithm for frequent 2-itemset generation is explained in Figure 5.6

```
Algorithm Frequent 2-itemset generation

Input   $Set\{Freq-1\}$, $List\{Freq-1\}$, $min\_sup$

Output Frequent 2-itemset ( $Set\{Freq-2\}$ )
          and their IDLists ( $List\{Freq-2\}$ )

Methodology
(1) For u=1 to $|Set\{Freq-1\}|-1$
(2) // $|Set\{Freq-1\}|-1$ since DFMat's are bidirectional
(3)    For v=u+1 to $|Set\{Freq-1\}|$
(4)      // let $e_i = Set\{Freq-1\}(u)$ and
(5)      // let $e_j = Set\{Freq-1\}(v)$
(6)      $DF_{Mat}\{e_i,e_j\} \leftarrow$ difference matrix of $e_i$ and $e_j$
(7)       $\Sigma_{Lag} = unique(DF_{Mat}\{e_i,e_j\})$
(8)       For k=1: $|\Sigma_{Lag}|$
(9)          $l_k = \Sigma_{Lag}(k)$
(10)         If $|l_k| <$ minsup
(11)           Go to next $l_k$
(12)         else
(13)           If $l_k = 0$
(14)             $Set\{Freq-2\} \leftarrow e_i(0)e_j(0)$
(15)           else if $l_k > 0$
(16)             $Set\{Freq-2\} \leftarrow e_i(0)e_j(l_k)$
(17)           else
(18)             $Set\{Freq-2\} \leftarrow e_j(0)e_i(abs(l_k))$
(19)           end // end of if $l_k > 0$ loop
(20)         end // end of If $|l_k| <$ minsup loop
(21)    end // end of for v loop
(22)    end // end of for u loop
```

Figure 5.6 Algorithm for Frequent 2-Itemset Generation

In most cases, a user is interested only in patterns that happen in a certain time interval (the Maxspan window). Our technique provides the flexibility of using such a maxspan by simply keeping a threshold $-Maxspan \le l_k \le Maxspan$ on the lag l_k. For example, if the Maxspan equals 5, then only the lag values from -5 to 5 will be used for making 2-itemsets. Therefore, $a(0)b(6)$ to $a(0)b(9)$ and $b(0)a(6)$ to $b(0)a(8)$ will not be formed.

One important addition that our technique provides is that the Maxspan can be changed without remaking DIFFMats, as the DIFFMats can be adjusted to a changing Maxspan by changing the threshold on l_k. In most of the existing work, a change in Maxspan means the entire process of itemset generation has to be redone. The frequent 2-itemsets can be formed by simply finding the lags that have a higher support than *minsup*. A 2-itemset $e_i(0)e_j(l_k)$ is frequent, if $|l_k| \ge \min\sup$. Thus, the frequent 2-itemsets using minsup=2 defined in Section 5.5.1. are $a(0)b(0), a(0)b(1)$, $a(0)b(2), a(0)b(3), a(0)b(5), a(0)b(6)$, $b(0)a(1), b(0)a(2), b(0)a(3)$, and $b(0)a(5)$. Once we have the itemsets, we need to find their IDLists. Since these are 2-itemsets, the number of rows for the IDList $(HIDList)$ will be two. The first row identifies the IDs of the transactions where the first event happens, and the second row identifies the IDs of the transactions where the second event happens. This information can be extracted easily from the DIFFMat of the events.

Definition 5.10 Given $DFMat\{e_i, e_j\}$ and $e_i(0)e_j(l_k)$, let $RLabel$ and $CLabel$ represent the row labels and column labels of $DFMat\{e_i, e_j\}$, respectively. Then IDList of $e_i(0)e_j(l_k)$ is formed as follows:

1. $IDlist\{e_i(0)e_j(l_k)\}(1,u) = RLabel(q), 1 \le u \le |l_k|$,

2. $IDlist\{e_i(0)e_j(l_k)\}(2,u) = CLabel(r), 1 \le u \le |l_k|$,

Such that $DFMat\{e_i, e_j\}(q,r) = l_k$, and $RLabel(q) \in IDlist(e_i), CLabel(r) \in IDlist(e_j)$.

We explain DIFFMat using $DFMat\{a,b\}$. The row and column labels are $RLabel = \{1,3,4,7,9,10\}$ and $CLabel = \{2,4,6,7,9,10\}$. Therefore, $IDlist\{a(0),b(1)\}(1,1) = 1$ $IDlist\{a(0)b(1)\}(1,2) = 3$, $IDlist\{a(0)b(1)\}(1,3) = 9$, $IDlist\{a(0)b(1)\}(2,1) = 2$, $IDlist\{a(0)b(1)\}(2,2) = 4$, and $IDlist\{a(0)b(1)\}(2,3) = 10$. The first row provides us with the IDs where $a(0)$ happens, and the second row provides us with the IDs where $b(1)$ happens. Here, the lag $l_k = 1$, and we see that IDs in the second row are exactly one time point away. Therefore, we see that not only can we generate the inter-transaction 2-itemsets using a difference matrix format; we can also find the exact location of these itemsets. Figure 5.7 (a) – (d) represents the IDLists of 2-itemsets $a(0)b(1), a(0)c(1), b(0)c(1)$, and $c(0)d(1)$, respectively.

$a(0)$	**1,3,9**
$b(1)$	**2,4,10**

(a)

$a(0)$	**1,4,7,9**
$c(1)$	**2,5,8,10**

(b)

$b(0)$	**2,6,9**
$a(1)$	**3,7,10**

(c)

$c(0)$	**2,3,6,8,9**
$a(1)$	**3,4,7,9,10**

(d)

Figure 5.7 IDList for Different 1-Itemset Combinations (a) $IDlist\{a(0),b(1)\}$, (b) $IDlist\{a(0),c(1)\}$, (c) $IDlist\{b(0),a(1)\}$, (d) $IDlist\{c(0),a(1)\}$

We can check the accuracy of the 2-itemsets by checking the IDLists in the original database. For example, $IDlist\{a(0)b(1)\}$ shows that when $a(0)$ happens in transactions 2, 3, and 9 in the database, then b also happens one time point later in transactions 3, 4, and 10, respectively. This correlation proves that the itemsets formed using DMITAR are indeed correct inter-transaction 2-itemsets. The support of an itemset, as explained in Section 5.5.1, is the length of its IDList. Therefore, using the association rule definitions described in Section 5. 2, we can find the inter-transaction association rules. Again using $a(0)b(1)$ as an example, we know that the support of $a(0)b(1) = |IDlist\{a(0)b(1)\}| = 3$ and the support of $a(0) = 6$. Therefore, the support and confidence of an inter-transaction association rule $a(0) \rightarrow b(1)$ are 3 and 3/6=0.50, respectively. Similarly, we can find all the possible inter-transaction association rules. Now that we have the 2-itemsets, we will explain how they can be extended to higher order itemsets.

5.5.3 Frequent k-Itemset Generation (k>2)

We now examine how to generate inter-transaction frequent-k itemsets (k>2). A common procedure for generating level k-itemsets from (k-1)-itemsets is to use the cross-join operation on (k-1)-itemsets. Given two frequent (k-1)-itemsets $\alpha = \{u_0(0), u_1(l_1), \ldots\ldots, u_{k-1}(l_{k-1})\}, \forall u_i \in E$ and $\beta = \{v_0(0), v_1(p_1), \ldots\ldots, v_{k-1}(p_{k-1})\}, \forall v_i \in E$, if the first k-1 items of both α and β are similar, and $u_{k-1}(l_{k-1}) > v_{k-1}(p_{k-1})$, then these two itemsets can be joined to form a new k-itemset represented as $\{u_0(0), u_1(l_1), \ldots\ldots, u_{k-1}(l_{k-1}), v_{k-1}(p_{k-1})\}$. For example, $a(0)b(1)$ can be joined to $a(0)c(1)$ to form the new candidate itemset $a(0)b(1)c(1)$. As explained by Kam and Fu [56], if a 2-itemset BC is not frequent, then any itemset for which this itemset forms the tail (ABC, ADBC, etc.) is not frequent. Keeping the above property in mind, we argue that since inter-transaction rules are directional temporal patterns, the only condition necessary is the condition on the last two atomic items $u_{k-1}(l_{k-1})$ and $v_{k-1}(p_{k-1})$. Therefore, if $u_{k-1}(0)v_{k-1}(p_{k-1} - l_{k-1})$ is not frequent, then $\{u_0(0), u_1(l_1), \ldots\ldots, u_{k-1}(l_{k-1}), v_{k-1}(p_{k-1})\}$ is not frequent. As a result, frequent k-itemsets can be generated by simply making the 2-itemset combinations of the last atomic item of the (k-1)-itemset with the existing frequent 1-itemsets. This combination reduces the number of candidates by a huge number, as the regular join operation usually finds many candidates. Further, since we already have the prior knowledge of the existing frequent 2-itemsets, we can remove infrequent 2-itemsets. Now, we will explain how we generate candidate k-itemsets from (k-1)-itemsets.

Given a frequent (k-1)-itemset, the first step is to check the frequent 2-itemsets and find those itemsets where the kth item is the first item. The second item from these 2-itemsets form the joinable candidate 1-items for this kth item.

Definition 5.11 Given $\alpha = \{e_0(0), e_1(l_1), \ldots\ldots, e_{k-1}(l_{k-1})\}$, the joinable candidate group for e_{k-1} is defined as $Cand(e_{k-1}(l_{k-1}) = \{e_j\}$, where

1. $e_{k-1} \neq e_j, \forall e_{k-1} \in E, e_j \in E$ and
2. $\exists l_j \mid e_{k-1}(0)e_i(l_j) \in Set(Freq-2)$.

For instance, given a frequent 2-itemset $a(0)b(1)$, the candidate group for b is $Cand(b(1)) = \{c,d,e\}$. Now that we have the candidates, the 2-itemset formation procedure is repeated to create $DFMat\{b(1),c\}, DFMat\{b(1),d\}$ and $DFMat\{b(1),e\}$. We already know the IDList for {c,d,e}, and the IDs for $b(1)$ are the IDs in the second row of $IDlist\{a(0)b(1)\}$. As a result, $IDlist\{b(1)\} = \{2,4,10\}$. While making the DIFFMats for frequent-k itemset generation (k>2), an extra constraint is added so that the IDs of *k*th item always form the row labels, and the IDs of its candidates form the column labels. The rest of the difference matrix formation procedure is the same as described in Section

5.5.2. *DFMat*$\{b(1),c\}$, *DIFFMat*$\{c(1),b\}$, *DIFFMat*$\{a(1),b\}$, and *DIFFMat*$\{c(1),d\}$ are shown in Figures 5.8 –5.11, respectively.

Labels c→ b↓	**2**	**3**	**5**	**6**	**8**	**9**	**10**
2	0	1	3	4	6	7	8
4	-2	-1	1	2	4	5	6
10	-8	-7	-5	-4	-2	-1	0

Figure 5.8 Difference Matrix *DFMat*$\{b(1),c\}$ for *k*-Itemset Generation

Labels b→ c↓	**2**	**4**	**6**	**7**	**9**	**10**
2	0	2	4	5	7	8
5	-3	-1	1	2	4	5
8	-6	-4	-2	-1	1	2
10	-8	-6	-4	-3	-1	0

Figure 5.9 Difference Matrix *DIFFMat*$\{c(1),b\}$ for *k*-Itemset Generation

Labels b→ a↓	**2**	**4**	**6**	**7**	**9**	**10**
3	-1	1	3	4	6	7
4	-2	0	2	3	5	6
7	-5	-3	-1	0	2	3
9	-7	-5	-3	-2	0	1
10	-8	-6	-4	-3	-1	0

Figure 5.10 Difference Matrix *DIFFMat*$\{a(1),b\}$ for *k*-Itemset Generation

Labels d→ c↓	**1**	**2**	**5**	**7**	**8**	**9**
2	-1	0	3	5	6	7
5	-4	-3	0	2	3	4
8	-7	-6	-3	-1	0	1
10	-9	-8	-5	-3	-2	-1

Figure 5.11 Difference Matrix *DIFFMat*$\{c(1),d\}$ for *k*-Itemset Generation

Once we have the DIFFMats, we can find the 2-itemsets using the following definition.

Definition 5.12 Given $DFMat\{e_i(l_{k-1}),e_j\}$, let Σ_{lag} represent the different lags in the matrix. Then, a 2-itemset is defined as $e_i(0)e_j(l_p)$, where

1. $l_p \geq 0, \forall l_p \in \Sigma_{lag}$ and
2. $e_i > e_j$ if $l_{k-1} = l_p$.

Notice that the first constraint on lags in Definition 5.12 is different from the constraints in Definition 5.9. This difference is because in Definition 5.12, we only consider $l_p \geq 0$ to reduce redundancy. Take, for example, a 3-itemset $a(0)b(1)c(2)$. If both the positive and negative lags are taken for Definition 5.12, then the 2-itemset combo $b(1)c(2)$ can be formed from $DFMat\{b,c\}$ and $DFMat\{c,b\}$. Therefore, to avoid such redundancy during k-itemset generation, only $l_p \geq 0$ is used. If $a(0)b(1)c(2)$ is indeed a frequent 3-itemset, then it will be formed by extending the 2-itemset $a(0)b(1)$ and not $a(0)c(2)$. The second constraint ensures that redundant itemsets of the form $a(0)b(1)c(1)$ and $a(0)c(1)b(1)$ are not created. Both of these itemsets can be read as "if a happens, then b and c happen one time point later." Therefore, using Definition 5.12, the 2-itemsets generated for $DFMat\{b(1),c\}$ are: $b(0)c(0), b(0)c(1), b(0)c(2)$, $b(0)c(3), b(0)c(4), b(0)c(5), b(0)c(6)$, $b(0)c(7)$, and $b(0)c(8)$. Their corresponding supports are 2,2,1,1,2,1,2,1, and 1, respectively.

Therefore, the frequent 2-itemsets are $b(0)c(0)$, $b(0)c(1), b(0)c(4)$, and $b(0)c(6)$ with their corresponding supports 2, 2, 2, and 2, respectively. The corresponding IDLists using Definition 5.10 in $(Rlabel, Clabel)$ format are $IDlist\{b(0)c(0)\} = \{(2,2)(10,10)\}$,

$IDlist\{b(0)c(1)\} = \{(2,3), (4,5)\}$, $IDlist\{b(0)c(4)\} = \{(2,6)(4,8)\}$, $IDlist\{b(0)c(6)\} = \{(2,8)(4,10)\}$. Now that we have the frequent 2-itemset combinations of the *k*th item of (k-1)-itemset, we can extend the existing (*k*-1) itemset using the following definition.

Definition 5.13 Let $\alpha = \{e_0(0), e_1(l_1), ..., e_{k-1}(l_{k-1})\}$ be a (k-1)-itemset, and let $2Set\{e_{k-1}(l_{k-1})\}$ represent the new 2-itemsets formed using $DFMat\{e_{k-1}(l_{k-1}), e_j\}, \forall e_j \in Cand(e_{k-1}(l_{k-1}))$. Therefore $\beta = \{e_{k-1}(0), e_j(l_p)\}$, such that $\beta \in 2Set\{e_{k-1}(l_{k-1})\}$ can be joined with α to form a new *k*-itemset δ as follows:

$$\delta = \{e_0(0), e_1(l_1), ..., e_{k-1}(l_{k-1}), e_j(l_p + l_{k-1})\}.$$

Using the Definition 5.13 and $a(0)b(1)$ as an example, we can see that the new 3-itemsets formed by joining $a(0)b(1)$ with $b(0)c(0)$, $b(0)c(1)$, $b(0)c(4)$, and $b(0)c(6)$ are: $a(0)b(1)c(1)$, $a(0)b(1)c(2)$, $a(0)b(1)c(5)$, and $a(0)b(1)c(7)$. Similarly, we can combine *b* with the remaining items in its candidate set $Cand(b(1))$. The same procedure can then be repeated to extend all the existing (k-1)-itemsets.

Further, Maxspan can be used if the user is only interested in itemsets that span across a certain number of transactions. The extra itemsets from newly formed *k*-itemsets can be pruned by keeping only those itemsets in which $l_k \leq Maxspan$. Therefore, the 3-itemsets for Maxspan =5 are $a(0)b(1)c(1)$, $a(0)b(1)c(2)$, and $a(0)b(1)c(5)$. $a(0)b(1)c(7)$ is pruned, since $l_k = 7 > Maxspan$.

Figure 5.12 presents the algorithms for frequent k-itemset generation (no maxspan is used in this algorithm).

```
Algorithm Frequent k-itemset generation

Input    $Set\{Freq-1\}$, $List\{Freq-1\}$, $minsup$,
         $Set\{Freq-2\}$, $List\{Freq-2\}$, $Set\{Freq-(k-1)\}$,
         $List\{Freq-(k-1)\}$

Output Frequent k-itemset ($Set\{Freq-k\}$)
         and their IDLists ($List\{Freq-k\}$)

Methodology
(1) For u =1 to $|Set\{Freq-(k-1)\}|$ // Loop 1
(2)   // let $\alpha = \{e_0(0), e_1(l_1), \ldots\ldots, e_{k-1}(l_{k-1})\}$ represent the current k-1 itemset
(3)   Find $Cand(e_{k-1}(l_{k-1}))$
(4)   For each value $e_j$ in $Cand(e_{k-1}(l_{k-1}))$ // Loop 2
(5)       Find $DF_{Mat}\{e_{k-1}(l_{k-1}), e_j\}$
(6)       $\Sigma_{lag} = unique(DF_{Mat}\{e_{k-1}(l_{k-1}), e_j\})$
(7)       For p=1: $|\Sigma_{lag}|$ // Loop 3
(8)         $l_p = \Sigma_{lag}(p)$
(9)         If $|l_p| <$ minsup
(10)          Go to next $l_p$
(11)        else
(12)          If $l_p \geq 0$
(13)            $2Set\{e_{k-1}(l_{k-1})\} \leftarrow e_{k-1}(0)e_j(l_p)$
(14)          end // end of if $l_p \geq 0$ loop
(15)             end // end of If $|l_p| <$ minsup loop
(16)          end // end of Loop 3
(17)      For each $e_{k-1}(0)e_j(l_p) \in 2Set\{e_{k-1}(l_{k-1})\}$
(18)    $Set\{Freq-k\}$
(19)          $\leftarrow \{e_0(0), e_1(l_1), \ldots\ldots, e_{k-1}(l_{k-1}), e_j(l_p + l_{k-1})\}$
(20)      end // end of for $e_{k-1}(0)e_j(l_p)$ loop
(21)   end // end of Loop 2
(22) end // end of Loop 1
```

Figure 5.12 Algorithm for Frequent-*k* Itemset Generation

Once we have the itemsets, we need to find the IDList of the newly generated itemsets. Then, given a k-itemset, IDList{k-1}, the 2-itemset candidate formed by the kth item, and IDList{2-cand}, we intersect the IDs from the kth row of IDList{k-1} with the IDs from the first row of IDList{2-cand}and find the indexes of the matching IDs. Once we find the indexes of the matching IDs, we discard those IDs that do not match so that only those that match are kept in the IDList of new k-itemset. Using $a(0)b(1)$ as (k-1)-itemset and $b(0)c(0)$ as one of the candidate 2-itemsets for $b(1)$, we find that the matching indexes using the second row of $IDlist\{a(0)b(1)\}$ and the first row of $IDlist\{b(0)c(0)\}$ are {1,3} (the first and third index of $IDlist\{a(0)b(1)\}$ match the indexes in $IDlist\{a(0)b(1)\}$). Therefore, we only keep those IDs that have these indexes for all the rows in $IDlist\{a(0)b(1)\}$. The last row in $IDlist\{a(0)b(1)c(1)\}$ will be filled using the indexes from the second row of $IDlist\{b(0)c(0)\}$. Therefore, the resulting IDList will have the following IDs $(Rlabel, Clabel)$ format: $IDlist\{a(0)b(1)c(1)\} = \{(1,9),(2,10),(2,10)\}$. The IDLists for $a(0)b(1)c(1), a(0)b(1)c(5), a(0)c(1)b(2)$, $a(0)c(1)b(3), a(0)c(1)d(1)$, and $a(0)c(1)d(4)$ can be seen in Figure 5.13 (a) – (f).

$a(0)$	**1, 9**
$b(1)$	**2,10**
$c(1)$	**2,10**

(a)

$a(0)$	**1, 3**
$b(1)$	**2,4**
$c(5)$	**6,8**

(b)

$a(0)$	**4,7**
$c(1)$	**5,8**
$b(2)$	**6,9**

(c)

$a(0)$	**1,4,7**
$c(1)$	**2,5,8**
$b(3)$	**4,6,9**

(d)

$a(0)$	**1,4,7**
$c(1)$	**2,5,8**
$d(1)$	**2,5,8**

(e)

$a(0)$	**1,4**
$c(1)$	**2,5**
$d(4)$	**5,7**

(f)

Figure 5.13 IDLists of Different Combinations: (a) $IDlist\{a(0),b(1)c(1)\}$, (b) $IDlist\{a(0),b(1)c(5)\}$, (c) $IDlist\{a(0),c(1)b(2)\}$, (d) $IDlist\{a(0),c(1)b(3)\}$, (e) $IDlist\{a(0),c(1)d(1)\}$, (f) $IDlist\{a(0),c(1)d(4)\}$

5.6 Completion of DMITAR (An Induction-Based Proof)

Now that we have explained the methodology, we will prove that DMITAR does not miss any itemsets during itemset generation. If $Set\{Freq-n\}$ is the set of frequent inter-transaction itemsets present in the database and $Set\{Freq-n)'$ is the frequent itemsets generated by DMITAR, then, to prove the completeness of DMITAR, we must first prove that $Set\{Freq-n\} = Set\{Freq-n\}'$. Given (*min_sup*), let Arg_n represent the argument that $Set\{Freq-n\}' = Set\{Freq-n\}$.

Since $Set\{Freq-1)'$ is the set of unique items present in the database with support>*min_sup*, Arg_1 is true. Let $IDList\{Freq_1\}$ be the corresponding IDLists of frequent 1-itemsets. Let us assume Arg_k is true. We need to prove that Arg_n, where n=k+1 is true. We will prove the equation by contradiction. Let us assume that $Set\{Freq-(k+1)\}' \neq Set\{Freq-(k+1)\}$. This equation implies that there exists an itemsets $\alpha = \{e_0(0), e_1(l_1), \ldots\ldots, e_{k-2}(l_{k-2}), e_{k-1}(l_{k-1})\}$ such that $\alpha \in Set\{Freq-(k+1)\}$ (the set of complete itemsets in the database), but $\alpha \notin Set\{Freq-(k+1)\}'$ (the set of itemsets generated by DMITAR). We already know that α was formed by combining $e_{k-2}(l_{k-2})$ with e_{k-1} to make a 2-itmeset. Therefore, for α to be a frequent k+1 itemset, $e_{k-2}(l_{k-2}), e_{k-1}(l_{k-1})$ should be a frequent 2-itemset. Now we already know from Arg_k that $e_0(0), e_1(l_1), \ldots\ldots, e_{k-2}(l_{k-2})$ is true, therefore α can only not be frequent if $e_{k-2}(l_{k-2}), e_{k-1}(l_{k-1}) \notin Freq_2'$. This logic implies that there exists and index K such that $IDList\{e_{k-2}(l_{k-2})e_{k-1}(l_{k-1})\}(1,K) \notin IDList\{e_{k-2}(l_{k-2})\}$ or $IDList\{e_{k-2}(l_{k-2})e_{k-1}(l_{k-1})\}(2,K) \notin IDList\{e_{k-1}(l_{k-1})\}$. However, this assumption contradicts the statement that Arg_1 is true. Hence, $e_{k-2}(l_{k-2}), e_{k-1}(l_{k-1}) \in Freq_2^{'}$. The above assumption indicates that $Set\{Freq-(k+1)\}^{'} = Set\{Freq-(k+1)\}$, and thus if Arg_k is true, then Arg_{k+1} is also true. Therefore, the argument $Set\{Freq-n\}^{'}$ $= Set\{Freq-n\}$ is true by induction. Since $Set\{Freq-n\}^{'} = Set\{Freq-n\}$, we know that DMITAR generates all possible frequent inter-transaction itemsets.

5.7 Results

As explained in Section 5.3, we used three different datasets: stock price data, weather data, and synthetic data to evaluate our algorithm. In order to convert the time series data into an event based data representation a threshold is used. If the difference between the current value of a variable and its future value is higher than or equal to the threshold, it is considered as a positive change, otherwise it is considered a negative change.

Stock Data: We use $\delta = 0$ as a threshold on the change of a stock price for transforming the original database into its binary counterpart. If the relative change in the closing price of a stock for the current day $\geq \delta$, then the stock value is changed to +1. Otherwise, it is changed to -1. Therefore, given *N* stock, there are a total of *N**2 events in the database. A positive change in stock S is represented by S1, while a negative change is represented by S2.

Weather Data: Again, we use $\delta = 0$ as a threshold for the relative change in all the six variables giving a total of 6*2 = 12 total events. The positive change in a variable (V) was represented by V1, and negative change was represented by V2.

Synthetic Data: Once again $\delta = 0$ is used as a threshold for the relative change in all the six variables giving a total of 6*2 = 12 total events. A positive change in a variable (V) is represented by V1, and negative change is represented by V2.

5.7.1 Comparative Evaluation of DMITAR With Existing Methods

We implement the DMITAR algorithm using Matlab R2007b on an AMD Opteron 150 Pc with 2.36 GHz Processor and 1 GB RAM. For comparison, we also implement FITI, PROWL, and ITPMine in Matlab and in the same processing environment we used for the DMITAR experiments. For comparative evaluation, we only take a subset of the complete data for the three datasets: the first 50 days and the first five stocks for the stock database, three month data (April 2009 to June 2009) and the six variables for the weather database, and one year (365 days) data and the six variables for the synthetic database. We only compared the performance of DMITAR for frequent itemset generation, since, as explained in [30], frequent itemset mining takes most of the execution time. The maxspan is fixed at five days, and the support is varied from 14% to 24%. Tables 5.1, 5.2, and 5.3 show the execution time (in seconds) for FITI, ITPMine, PROWL, and DMITAR over the stock, weather, and synthetic databases. The graphs showing the relative performance of DMITAR with existing techniques are presented in Figures 5.14 - 5.16. As is evident from the comparative evaluation, DMITAR takes less execution time on all three datasets than FITI and ITPMine.

Table 5.1 Comparison of DMITAR with Existing Techniques over Stock Dataset

Stock Database				
Support	**FITI**	**ITPMine**	**PROWL**	**DMITAR**
14%	6424.7s	132.39s	3.03s	5.556s
16%	2348.9s	67.14s	2.14s	4.015s
18%	861.92s	34.62s	1.55s	2.89s
20%	334.51s	18.89s	1.12s	2.07s
22%	143.84s	10.87s	0.87s	1.45s
24%	63.62s	7.15s	0.671s	1.04s

Table 5.2 Comparison of DMITAR with Existing Techniques over Weather Dataset

Weather Database				
Support	**FITI**	**ITPMine**	**PROWL**	**DMITAR**
14%	36362.6s	893.1094s	5.843s	19.8281s
16%	11913.04s	378.2188s	3.8906s	13.4375s
18%	4116s	170.3438s	2.75s	9.1406s
20%	1507s	86.5781s	2.14s	6.203s
22%	859.2813s	63.3438s	1.7969s	5.7656s
24%	378.5313s	36.1875s	1.4375s	3.5625s

Table 5.3 Comparison of DMITAR with Existing Techniques over Synthetic Dataset

Synthetic Dataset				
Support	**FITI**	**ITPMine**	**PROWL**	**DMITAR**
14%	1651.6s	199.843s	3.1406s	17.015s
16%	574.32s	119.32s	2.0938s	10.875s
18%	416.109s	95.31s	1.6094s	7.39s
20%	370.25s	83.31s	1.453s	5.8438s
22%	265.78s	66.3438s	1.3594s	4.75s
24%	133.96s	43.0781s	0.9219s	3.5781s

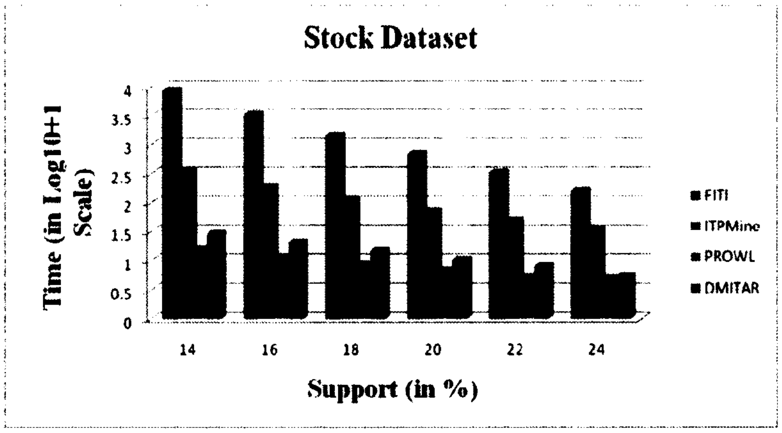


Figure 5.14 Time Comparison of DMITAR with Other Methods for Stock Dataset

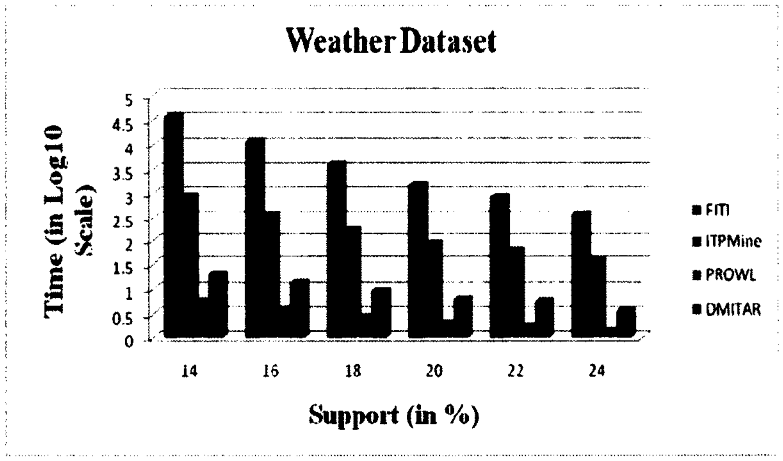


Figure 5.15. Time Comparison of DMITAR with Other Methods for Weather Dataset

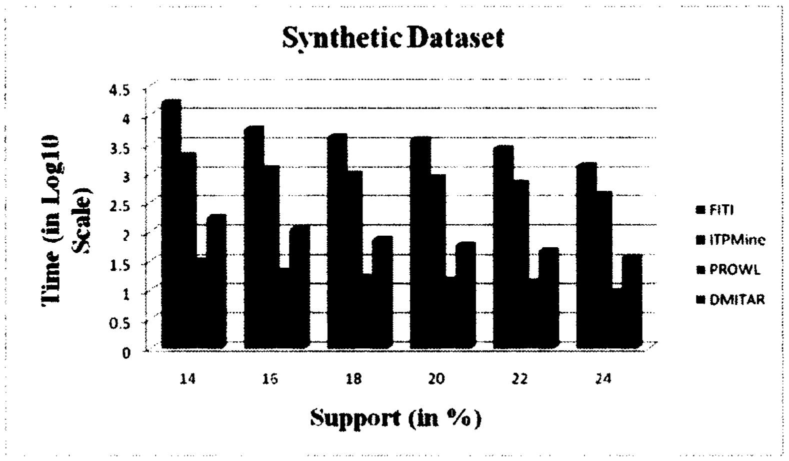
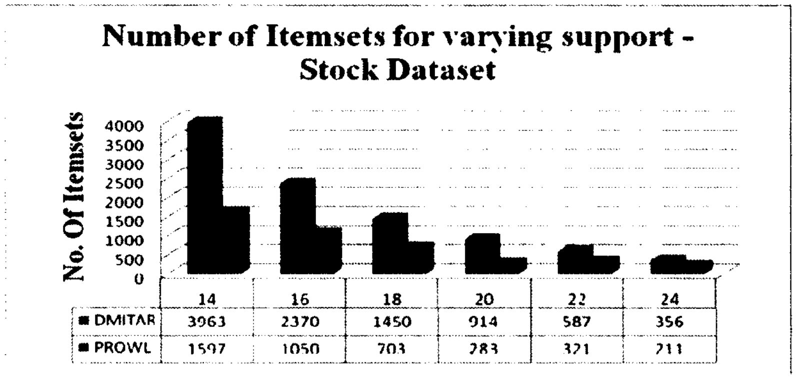


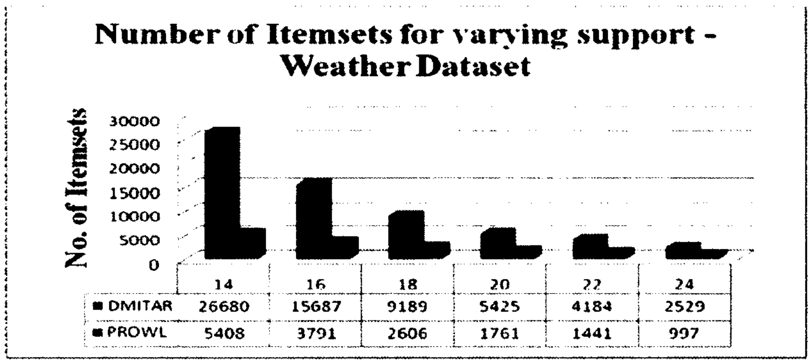
Figure 5.16 Time Comparison of DMITAR with Other Methods for Synthetic Dataset

The only algorithm that performs faster than DMITAR is PROWL. But as we have already discussed it in Section 5.3, PROWL does not generate all the inter-transaction itemsets. Figure 5.17 (a) – (c) shows the number of itemsets generated by DMITAR and PROWL.



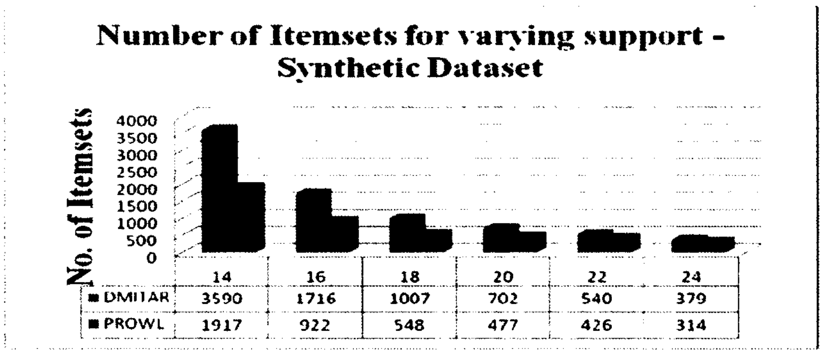
	14	16	18	20	22	24
■ DMITAR	3963	2370	1450	914	587	356
■ PROWL	1597	1050	703	283	321	211

(a)



	14	16	18	20	22	24
■ DMITAR	26680	15687	9189	5425	4184	2529
■ PROWL	5408	3791	2606	1761	1441	997

(b)



	14	16	18	20	22	24
■ DMITAR	3590	1716	1007	702	540	379
■ PROWL	1917	922	548	477	426	314

(c)

Figure 5.17 Comparison of Total Number of Itemsets Generated by DMITAR and PROWL: (a) Stock Dataset, (b) Weather Dataset, (c) Synthetic Dataset

It can be seen from the graphs that the difference between the actual number of itemsets present in the database (the ones generated by DMITAR, FITI, and ITPMine) and the ones generated by PROWL increases with the decrease in support. This difference in the number of itemsets generated will be even higher if the Maxspan size is smaller than the number of variables used (explained in Section 5.3). Therefore, we can see that although PROWL is the fastest of the four algorithms, it is still inefficient in capturing all the itemsets. Hence, we only make comparisons of DMITAR with FITI and ITPMine in the rest of the experiments. On a close inspection of Tables 5.1, 5.2, and 5.3, we see that the time difference between FITI compared with ITPMine and DMITAR is very large. We will discuss the time difference in execution in more detail here. Due to the previously explained inefficiencies of PROWL, comparison between it and our system is not required and, hence, is avoided here.

First, we compare DMITAR with FITI. FITI is a horizontal data-based sliding window technique. It is iterative and joins itemsets to first generate candidates and then scan the database repetitively to count the support of these candidates. Every time a transaction is read for support counting in FITI, the algorithm has to match all the candidate inter-transaction itemsets with at least *w* consecutive transactions. This multi-scan repetitive nature adds to the execution time. ITPMine and DMITAR, on the other hand, are both single pass-based schemes and, as a result, take far less time to execute.

As can be seen from the graphs, DMITAR performs faster than ITPMine. There are several reasons for this time difference. First, as explained in Section 4, ITPMine does not have a mechanism to find frequent 1-itemsets while hashing candidate 2-itemsets. As a result, it hashes 2-itemsets that have subsets that might not satisfy the downward closure property of association rules. Second, unlike DMITAR, ITPMine is a DFS based unidirectional technique. Therefore, during itemset formation for α, only itemsets of the form "α followed by β" are formed. The procedure has to be repeated to generate itemsets of the form "β followed by α." Finally, DMITAR avoids unnecessary candidates during k-itemset generation by using the constraints that, if $u_{k-1}(0)v_{k-1}(p_{k-1}-l_{k-1})$ is not frequent, then $\delta = u_0(0), u_1(l_1), \ldots\ldots, u_{k-1}(l_{k-1}), v_{k-1}(p_{k-1})$ is not frequent either. ITPMine also uses the same constraint on the last two items of the *k*-itemset.

In this step, there is one major difference between the methods. ITPMine first makes the joinable group for $\alpha = u_0(0), u_1(l_1), \ldots\ldots, u_{k-2}(l_{k-2}), u_{k-1}(l_{k-1})$ $(joinable(\alpha))$, such that the first *k*-1 items of all the itemsets in $(joinable(\alpha))$ are the same as the first *k*-1 items of α $(u_0(0), u_1(l_1), \ldots\ldots, u_{k-2}(l_{k-2}))$, and the *k*th item for each itemset in $joinable(\alpha)$ is greater than the kth item in α. Then, before making $\{u_0(0), u_1(l_1), \ldots\ldots, u_{k-1}(l_{k-1}), v_{k-1}(p_{k-1})\}$ using α and an itemsets from $joinable(\alpha)$, it hashes the 2-itemset $u_{k-1}(0)v_{k-1}(p_{k-1}-l_{k-1})$ to see if it is frequent. If so, only then is $\{u_0(0), u_1(l_1), \ldots\ldots, u_{k-1}(l_{k-1}), v_{k-1}(p_{k-1})\}$ generated as a candidate. As a result, ITPMine takes more time than DMITAR because, regardless of the fact that some 2-itemsets might not be frequent (a fact that can be verified from the set of frequent 2-itemsets), it still has to hash all the 2-itemsets formed using $joinable(\alpha)$ to find their support. For example, suppose there are 100 itemsets in $joinable(\alpha)$ and only 20 of those would form frequent 2-itemsets. In such a case, ITPMine still makes 2-itemsets with all these 100 itemsets to find their support from the hash table. DMITAR, on the other hand, uses the information from frequent 2-itemsets.

Thereby, it only creates combinations with known frequent itemsets. Further, the fact that $u_{k-1}(0)v_{k-1}(p_{k-1}-l_{k-1})$ is frequent does not guarantee that $\{u_0(0),u_1(l_1),.....,$ $u_{k-1}(l_{k-1}),v_{k-1}(p_{k-1})\}$ is also frequent. ITPMine still has to match the IDs in the IDList of $\{u_0(0),u_1(l_1),.....,u_{k-2}(l_{k-2}),u_{k-1}(l_{k-1})\}$ and $u_0(0),u_1(l_1),.....,u_{k-2}(l_{k-2})v_{k-1}(p_{k-1})$ to find the matching IDs and then check its support to see if it is frequent. The DIFFMats formed for DMITAR capture all this information in just one difference matrix, thereby creating all the possible higher order itemsets.

Once the frequent itemsets have been generated, we can very easily generate the frequent association rules from these itemsets using a threshold on minconf and Definitions 5.3 and 5.4. Tables 5.4, 5.5, and 5.6 show the top five rules extracted (Sup = 20%, and Conf = 60%) from the Stock, Weather, and Synthetic datasets, respectively.

Table 5.4 Top Five Rules from the Stock Dataset

Yahoo $\Uparrow$, IBM $\Uparrow$ (0), Google $\Downarrow$ (1) $\Rightarrow$ Microsoft $\Downarrow$ (2) *Support* = 22%, *Confidence* = 100%
Google $\Downarrow$, Microsoft $\Uparrow$ (1) $\Rightarrow$ Apple $\Downarrow$ (1) *Support* = 22%, *Confidence* = 91.67%
Google $\Downarrow$, IBM $\Downarrow$ (1) $\Rightarrow$ Apple $\Downarrow$ (3) *Support* = 22%, *Confidence* = 91.67%
Yahoo $\Uparrow$, Microsoft $\Downarrow$ (1), IBM $\Downarrow$ (0) $\Rightarrow$ Google $\Downarrow$ (2) *Support* = 20%, *Confidence* = 90.91%
Apple $\Downarrow$, IBM $\Uparrow$ (0) $\Rightarrow$ Microsoft $\Downarrow$ (3) *Support* = 20%, *Confidence* = 90.91%

Table 5.5 Top Five Rules from the Weather Dataset

Dewpoint ⇑, St. Pressure ⇓ (0), Wetbulb ⇑ (1) ⇒ Avg. Temp ⇑ (3)
Support = 20%, *Confidence* = 100%
Avg. Temp ⇑, St. Pressure ⇑ (1), Sea Pressure ⇑ (1) ⇒ Wetbulb ⇑ (1)
Support = 23.33%, *Confidence* = 100%
Sea Pressure ⇓, St. Pressure ⇑ (3) ⇒ Wetbulb ⇑ (1)
Support = 22.22%, *Confidence* = 100%
St. Pressure ⇓, Sea Pressure ⇑ (2) ⇒ Wetbulb ⇑ (2)
Support = 26.66%, *Confidence* = 100%
Avg. WindSpeed ⇓, Sea Pressure ⇑ (1) ⇒ Wetbulb ⇑ (1)
Support = 24.44%, *Confidence* = 100%

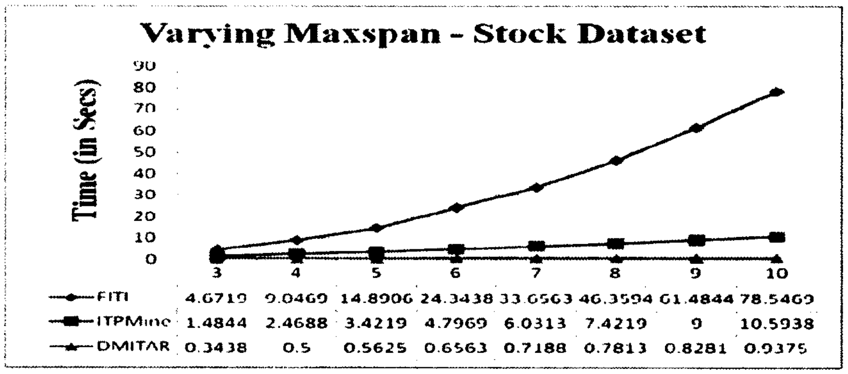
Table 5.6 Top Five Rules from the Synthetic Dataset

Max. Temp ⇓, Vap. Pressure ⇓ (0), Rel Humidity ⇓ (0) ⇒ Pot. EvaTransp. ⇓ (1)
Support = 15.06%, *Confidence* = 78.57%
Max. Temp ⇑, Vap. Pressure ⇓ (1), Rel Humidity ⇓ (0) ⇒ Pot. EvaTransp. ⇓ (1)
Support = 13.97%, *Confidence* = 76.12%
Rel Humidity ⇓, Vap. Pressure ⇓ (2) ⇒ Pot. EvaTransp. ⇓ (1)
Support = 18.08%, *Confidence* = 79.52%
Max. Temp ⇓, Rel Humidity ⇓ (0) ⇒ Pot. EvaTransp. ⇓ (1)
Support = 15.34%, *Confidence* = 76.71%
WindSpeed ⇑, Vap. Pressure ⇓ (2) ⇒ Pot. EvaTransp. ⇓ (1)
Support = 17.26%, *Confidence* = 75.90%

The rules are represented in the format: Attribute name ⇑ / ⇓ (lag) where ⇑ represents a positive change in the attribute value (e.g. stock going up or temperature going up), ⇓ represents a negative change in the attribute value (e.g. stock going down or temperature going down), and lag represents the time difference of the current attribute occurrence from the occurrence of the last attribute (e.g. two days later or two hours later). There is no lag value after the first attribute value since the occurrence of this attribute leads to the occurrence of other attributes. Therefore, by default, it can be assumed that such a value occurs at the first time point (e.g. the first day). A Lag value of zero (0) means that the attribute values happen at the same time. The first rule from the Stock dataset can be read as "If Yahoo's stock price rises, IBM's stock price rises the same day (lag is zero), and Google's stock price falls one day later, then Microsoft's stock price will fall two days later." This rule has a 22% support and 100% confidence. Similarly, the first rule from the Weather dataset can be read as "If dew point goes up and station pressure goes down in the same day and wetbulb increases one day later, then the average temperature will go up three days later with 20% Support and 100% Confidence. All the remaining rules can be interpreted in the same way.

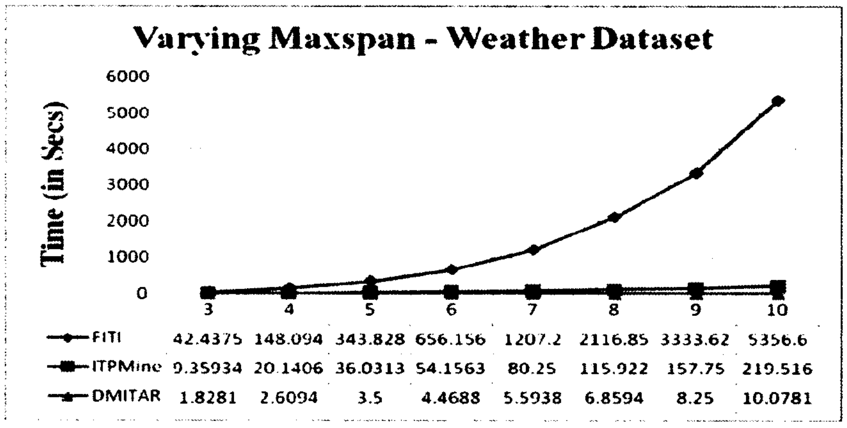
5.7.2 Execution Time with Changing Window (Maxspan) Size

To evaluate the effect of changing maxspan, we varied the maxspan value from three to ten. The support was fixed at 24% for all three datasets. The amount of data used here for the stock, weather, and synthetic datasets were the same as those used in Section 5.7.1. As can be seen from the graphs in Figure 5.18 (a) – (c), the execution time for FITI increases rapidly for all three datasets while the time for ITPMine and DMITAR increases linearly. This pattern can be attributed to the increase in maxspan, which results in a rapid increase in the number of candidate itemsets for FITI.



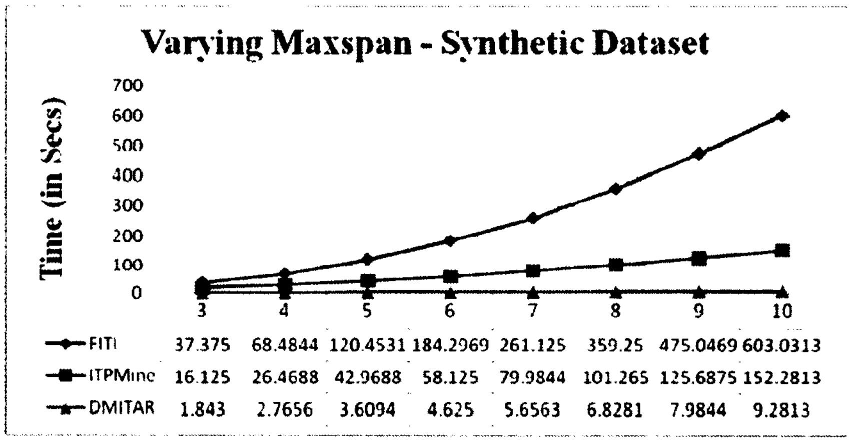
	3	4	5	6	7	8	9	10
FITI	4.6719	9.0469	14.8906	24.3438	33.6563	46.3594	61.4844	78.5469
ITPMine	1.4844	2.4688	3.4219	4.7969	6.0313	7.4219	9	10.5938
DMITAR	0.3438	0.5	0.5625	0.6563	0.7188	0.7813	0.8281	0.9375

(a)



	3	4	5	6	7	8	9	10
FITI	42.4375	148.094	343.828	656.156	1207.2	2116.85	3333.62	5356.6
ITPMine	9.35934	20.1406	36.0313	54.1563	80.25	115.922	157.75	219.516
DMITAR	1.8281	2.6094	3.5	4.4688	5.5938	6.8594	8.25	10.0781

(b)



	3	4	5	6	7	8	9	10
FITI	37.375	68.4844	120.4531	184.2969	261.125	359.25	475.0469	603.0313
ITPMine	16.125	26.4688	42.9688	58.125	79.9844	101.265	125.6875	152.2813
DMITAR	1.843	2.7656	3.6094	4.625	5.6563	6.8281	7.9844	9.2813

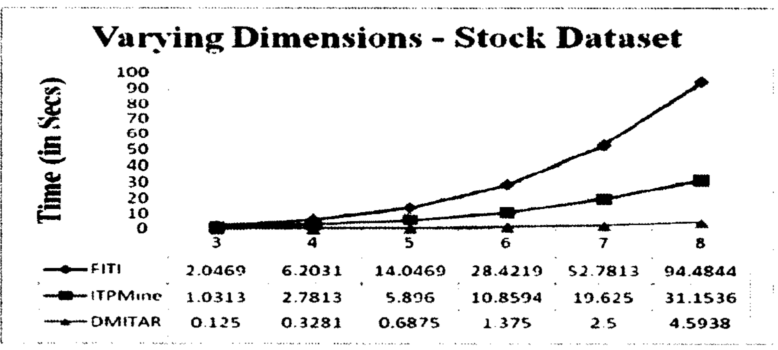
(c)

Figure 5.18 Execution Time Comparison of DMITAR with FITI and ITPMine for Varying Maxspan Size (3 to 10): (a) Stock Dataset, (b) Weather Dataset, (c) Synthetic Dataset

As a result, the time required to count the support of the itemsets also increases. Alternatively, both ITPMine and DMITAR use a single scan over the database and avoid unnecessary candidate generation; therefore, their time increases only linearly. However, for the same support DMITAR far outperforms ITPMine with an increase in maxspan.

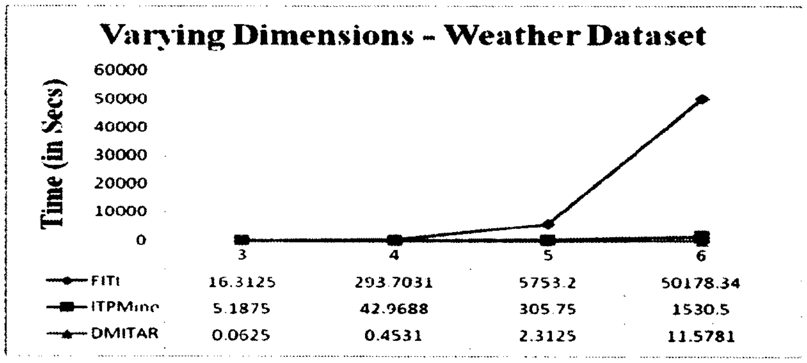
5.7.3 Execution Time with Varying Number of Dimensions

Next, we examine the effect of changing the size of the transactions in the database. Since the maximum number of columns in our stock dataset is eight (eight stocks), and, in the weather and synthetic datasets is six, we increase the transaction size from three to eight for stock data and from two to six for synthetic and weather data. For these experiments, the support measure is fixed at 20%, and the maxspan is fixed at five intervals. For stock data, we use all of the 253 day trading data; for weather data, we use twelve month data (July 2008 to June 2009), and, for synthetic data, we use the data for one year (365 days). The resulting execution time is presented in Figure 5.19 (a) – (c).



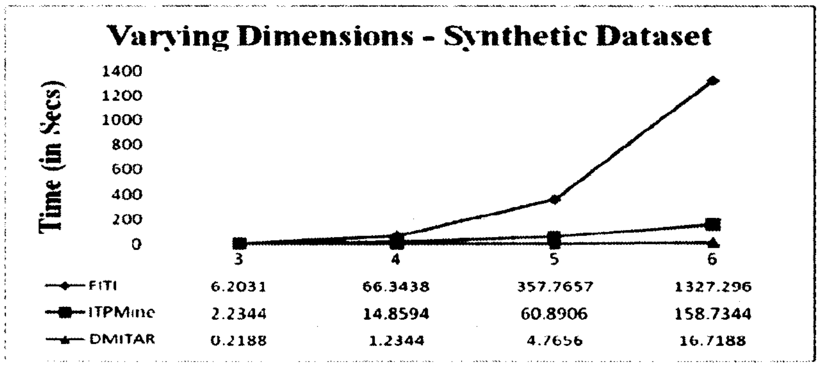
	3	4	5	6	7	8
FITI	2.0469	6.2031	14.0469	28.4219	52.7813	94.4844
ITPMine	1.0313	2.7813	5.896	10.8594	19.625	31.1536
DMITAR	0.125	0.3281	0.6875	1.375	2.5	4.5938

(a)



	3	4	5	6
FITI	16.3125	293.7031	5753.2	50178.34
ITPMine	5.1875	42.9688	305.75	1530.5
DMITAR	0.0625	0.4531	2.3125	11.5781

(b)



	3	4	5	6
FITI	6.2031	66.3438	357.7657	1327.296
ITPMine	2.2344	14.8594	60.8906	158.7344
DMITAR	0.2188	1.2344	4.7656	16.7188

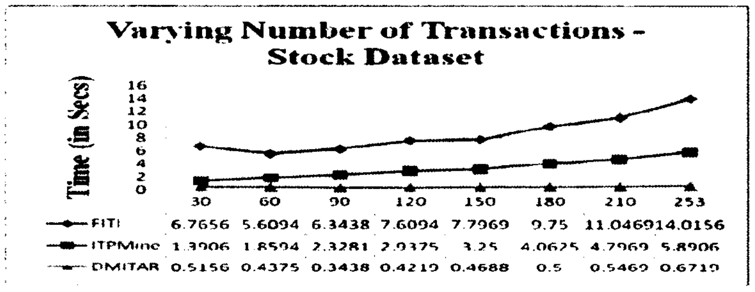
(c)

Figure 5.19. Execution Time Comparison of DMITAR with FITI and ITPMine for Varying Number of Dimensions (a) Stock Dataset, (b) Weather Dataset, (c) Synthetic Dataset

The number of candidate itemsets increased exponentially as the transaction size increased. Since the execution time of DMITAR was dependent on the number of frequent itemsets, as expected, the time increased exponentially. FITI and ITPMine exhibited a similar behavior, as well. However, once again DMITAR was faster than the other three techniques.

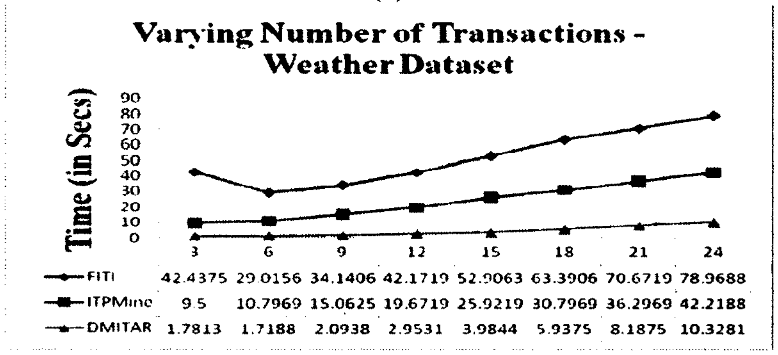
5.7.4 Execution Time with Varying Number of Transactions

Finally, we investigate the effect of the varying the number of transactions in the database on the execution time of DMITAR. For the stock dataset, we vary the number of transactions from 100 to 253 (trading days). For the weather data, we vary the number of transactions from 180 (6 months data) to 703 (24 month data), and for the synthetic dataset, we vary the number of transactions from 365 (one year data) to 2555 (seven year data). For these experiments, the support and maxspan are fixed at 20% and 5, respectively. For stock data, all eight stocks were used, and for weather and synthetic data, all six variables used in Section 5.7.1 were used. The comparative results of DMITAR with FITI and ITPMine are presented in Figures 5.20 (a) – (c).



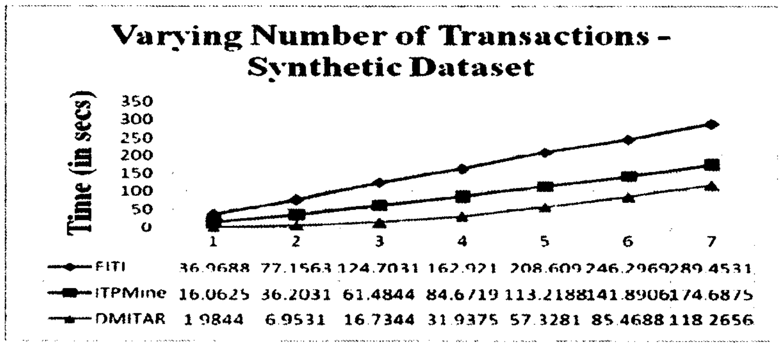
	30	60	90	120	150	180	210	253
FITI	6.7656	5.6094	6.3438	7.6094	7.7969	9.75	11.0469	14.0156
ITPMine	1.3906	1.8594	2.3281	2.9375	3.25	4.0625	4.7969	5.8906
DMITAR	0.5156	0.4375	0.3438	0.4219	0.4688	0.5	0.5469	0.6719

(a)



	3	6	9	12	15	18	21	24
FITI	42.4375	29.0156	34.1406	42.1719	52.9063	63.3906	70.6719	78.9688
ITPMine	9.5	10.7969	15.0625	19.6719	25.9219	30.7969	36.2969	42.2188
DMITAR	1.7813	1.7188	2.0938	2.9531	3.9844	5.9375	8.1875	10.3281

(b)



	1	2	3	4	5	6	7
FITI	36.9688	77.1563	124.7031	162.921	208.609	246.2969	289.4531
ITPMine	16.0625	36.2031	61.4844	84.6719	113.2188	141.8906	174.6875
DMITAR	1.9844	6.9531	16.7344	31.9375	57.3281	85.4688	118.2656

(c)

Figure 5.20 Execution Time Comparison of DMITAR with FITI and ITPMine for Varying Number of Transactions (a) Stock Dataset, (b) Weather Dataset, (c) Synthetic Dataset

From the graphs, we see that the execution time increases almost linearly for all three techniques (FITI, ITPMine, and DMITAR) over the three datasets. However, once again DMITAR outperforms both FITI and ITPMine on all three datasets.

5.8 Conclusions

The unprecedented generation and availability of large temporal databases and the ability of association rules to discover embedded useful knowledge therein has lead to inter-transaction and intra-transaction rule discovery efforts. However, such efforts have focused primarily on abstracting the data space so that classical apriori principle-based algorithms can be applied to them. As such, most inter-transaction association rule mining algorithms use a sliding overlapping window concept to convert a transaction database into a mega transaction database, thereby abstracting the problem to an intra-transaction rule-mining problem. This database transformation scheme is subject to amplified dimensionality, supervised determination of the fixed sliding window size, and determination of an optimal window overlap for extracting the rules. Moreover, the rules discovered from such schemas lack temporal location characterization and are difficult to apply practically. This lack of characterization leads us to state that the problem has not been defined comprehensively in the existing literature, which has led to sparseness and voids in the proposed solutions.

We have taken a fresh look at the problem of inter-transaction rule discovery in this work. We present a new algorithmic framework, DMITAR, based on the differences of the index of frequent items. DMITAR eliminates the need for a sliding window scheme and the creation of a mega-transaction, hence, constraining data dimensionality further improves its performance. Additionally, the algorithm provides the location of every frequent inter-transaction rule generated from the database. We have performed exhaustive experimentation with three temporal datasets from different domains. Experimental results demonstrate that the technique is comparatively superior, and is robust in resolving key issues faced by existing inter-transaction association rule mining algorithms.

CHAPTER 6

CONCLUSIONS

6.1 Contributions to Associative Classification

The research presented in this dissertation and related published results addresses the problem of higher order knowledge representation and provides a novel perspective for the associative classification problem. In Chapter 3, we present the association rule-based higher-order data representation framework. By viewing every instance of a class as a separate transaction database, we discover non-class constrained isomorphic patterns. Experimental results showed that this new representation captures more information than raw feature based representation. The research and insight provided by this work motivated us to develop a new associative classifier. Earlier associative classification algorithms focus on finding perfect discriminatory patterns from data, which is a difficult task because many times, such perfect patterns are only present for one type of class instance or do not exist at all. Discarding patterns that are common to some classes is not the best method, as the importance of patterns varies from class to class. Therefore, it becomes imperative to weight the patterns according to their importance in their own class (*intra-class*) and their importance in other classes (*inter-class*). In Chapter 4, we present a novel weighted rule-based associative classifier called WAR-BC. The objective of this work is to develop a non-class constrained rule-based associative classifier that would incorporate the inter-class and intra-class importance of the rules in rule weighting. Our proposed method alleviates most of the issues that plague current associative classifiers and open the door for further innovations. While the current work is complete, there is still room for improvement. It is well-known that certain feature types are more important for different domain of images. One immediate line of improvement would be to use multiple feature types for images and then perform ensemble-based classification [57]. Incorporating this knowledge could increase the classification of the algorithm. Another line of research would be to extend the current associative classification framework by incorporating classifier delegation [58], whereby a separate classifier (associative or another) would be used to train those instances which are deemed difficult (i.e. those images which the classifier is not able to categorize during training). These explorations will enhance our understanding of the classification process in general.

6.2 Contribution to Temporal Pattern Mining and Future Directions

The definition for temporal pattern mining builds a strong foundation for the establishment of the proposed research and future associated endeavors. Most existing algorithms, with the exception of PROWL and ITPMine (as referenced earlier), use the horizontal data format to solve the inter-transaction rule-mining problem. We have shown that this representation causes various problems. Using the success built by the vertical format of PROWL and ITPMine, we developed a new framework which solves all the issues faced by techniques using the horizontal data format and the issues faced by PROWL and ITPMine. The results from this research provide better insight into the inter-transaction pattern-mining problem. The technique addresses key limitations of the existing methods, especially those stemming from an increase in the dimensionality of the data space, multiple passes over databases, and lack of location-specific rule information. The applications of such algorithms include, but are not limited to, the analysis of financial databases, gene expression time series, growth patterns in toxicoinformatics, and hydrology, as well as hurricane prediction and modeling and cybersecurity applications. The elucidation of the design and implementation of a robust, scalable, and efficient algorithm for the discovery of inter-transaction rules is complete, yet there are several opportunities for enhancement. We list some of them here:

1. Currently, DMITAR only generates one-dimensional inter-transaction rules, as there is only one varying dimension, (i.e. time). Adding more dimensions, such as space (distance), to the data for generating multi-dimensional inter-transaction rules would create interesting challenges for ID encoding-based difference matrix formulation [59].

2. Along with prediction, time series classification is a key application for time-series analysis algorithms. The application of DMITAR to such domains as EEG signal classification, target tracking, and cancer-cell mutation could be important additions to the algorithm.

Future efforts will entail addressing these issues and performing characterization with increased and heterogeneous data workloads.

REFERENCES

[1] Han J., Kamber M., Data Mining Concepts and Techniques, *San Francisco*, Morgan Kaufman Publishers, 2001.

[2] Gantz J., "An Updated Forecast of Worldwide Information Growth through 2011," IDC White Paper, 2008.

[3] Frawley W.J., Piatetsky-Shapiro, G., and Matheus, C., "Knowledge Discovery in Databases: An Overview," *In Knowledge Discovery in Databases*, AAAI Press/MIT Press, Cambridge MA, 1991, pp. 1-30.

[4] Fayyad, U.M., Piatetsky-Shapiro, G., and Smyth, P., "From Data Mining To Knowledge Discovery: An Overview," *In Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, Menlo Park, CA, pp. 1-34, 1996.

[5] Quinlan J.R., "Programs for Machine Learning," Morgan Kaufman, 1993.

[6] Liu B., Hsu W., Ma Y., "Integrating Classification and Association Rule Mining," *In KDD 98*, New York, NY, Aug 1998.

[7] Dong G., Zhang X., Wong L., Li J., "CAEP: Classification by Aggregating Emerging Patterns," *In DS' 99* (LNCIS 1721), Japan, Dec. 1999.

[8] Lei W., Han J., Pei J., "CMAR: Accurate and Efficient Classification based on Multiple Class-Association Rules," *IEEE ICDM*, Nov. 2001.

[9] Chen C-L., Tseng F.S.C., Liang T., "Hierarchical Document Clustering using Fuzzy Association Rules," *3rd International Conference on Innovative Computing Information and Control*, 2008.

[10] Lent B., Swami A., Widom J., "Clustering Association Rules", *13th International Conference on Data Engineering,* 1997.

[11] Gupta G., Strehl A., Ghosh J., "Distance Based Clustering of Association Rules," *In Proceedings of ANNIE*, 1999.

[12] Yaik O.B., Yong C.H., Haron F., "Time Series Prediction using Adaptive Association Rules," *First International Conference on Distributed Frameworks for Multimedia Applications*, 2005.

[13] Kamei Y., Monden A., Morisaki S., Matsumoto K-I., "A Hybrid Faulty Model Prediction using Association Rule Mining and Regression Analysis," *Proceedings of Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 279-281,2008.

[14] Wu H-C., Huang S-H S., "Masquerade Detection using Command prediction and Association Rule Mining," *International Conference on Advanced Information Networking and Applications*, 2009.

[15] Agrawal R., Imielinski T., Swami A. "Mining Association Rules between Sets of Items in Large Databases," *SIGMOD Conference*, pp. 207-216, 1993.

[16] Piatetsky-Shapiro, G. *Discovery, Analysis, and Presentation of Strong Rules*, In G. Piatetsky-Shapiro & W. J. Frawley, eds, '*Knowledge Discovery in Databases*,' AAAI/MIT Press: Cambridge, MA, 1991.

[17] Power D. J. (2002, November 10). *DSS BiWeekly News Letter* [Online]. Available: http://www.dssresources.com/newsletters/66.php.

[18] Wright P. (1998, January) *Knowledge Discovery in Databases: Tools and Techniques* [Online]. Available: http://www.acm.org/crossroads/xrds5-2/kdd.html.

[19] Yang, C., Richard A. M., Cross K. P., "The Art of Data Mining the Minefields of Toxicity Databases to Link Chemistry to Biology." *Current Computer-Aided Drug Design*.Vol. 2, No. 2, pp.135-150, 2006.

[20] Quinlan, J. R., "Induction of Decision Trees," Machine Learning 1, pp. 81-106, March 1986.

[21] Rulequest Research See5 Software (2009, November). *See5: An Informal Tutorial* [Online]. Available: http://www.rulequest.com/see5-win.html.

[22] Quinlan, J. R., C-Jones R. M., "FOIL: A Midtern Report," *In proceedings on the European Conference on Machine Learning*, pp. 3-20, 1993.

[23] Cohen W., "Fast effective rule induction", *In Proceedings of 12th international conference on Machine learning*, pp. 115-123, 1995.

[24] Dong G., Zhang X., Wong L., Li J., "CAEP: Classification by Aggregating Emerging Patterns," *In Proceedings of International Conference on Discovery Science*, pp 30–49, 1999.

[25] Dong G., Li. J. "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," *In Proceedings of the Fifth ACM SIGKDD Conference*, pp. 43-52, 1999.

[26] Han J., Pei J., Yin Y., Mao R., "Mining Frequent Patterns without Candidate Generation," *Data Mining and Knowledge Discovery* , pp. 53-87, 2004.

[27] Yin X., Han J., "CPAR: Classification Based on Predictive Association Rules," *In Proceedings of SDM*, 2003.

[28] Clark P., Boswell R., "Rule Induction with CN2: Some Recent Improvements," *In Proceedings of European Working Session on Learning*, pp. 151, 1991.

[29] Lu H., Han J., Feng L., "Stock Movement Prediction and n-Dimensional Inter-Transaction Association Rules," *In Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pp. 297-301, 1999.

[30] Tung A. K. H., Lu H., Han J., Feng L., "Efficient Mining of InterTransaction Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, pp. 43-56, 2003.

[31] Luhr S., West G., Venkatesh S., "Recognition of Emergent Human Behavior in a Smart Home: A Data Mining Approach," *Pervasize and Mobile Computing* 3(2), pp. 95-116, 2007.

[32] Wang Z. Z. H-W, Huang G-C., "A New Algorithm Based on Matrix for Mining Inter-Transaction Association Rules," *IEEE International Conference on Wireless communications, Networking, and Mobile Computing*, pp. 6717-6720, 2007.

[33] Farlex Free Dictinoray (2010, March) *Cooccurence Matrix.*[Online] Available: http://encyclopedia2.thefreedictionary.com/cooccurrence+matrix.

[34] Huang K-Y., Chang C-H., Lin K-Z., "PROWL: An Efficient Frequent Continuity Mining Algorithm on Event Sequences," *In Proceedings of International Conference on Data Warehousing and Knowledge Discovery*, Lecture Notes in Computer Science, pp. 351-360, 2004.

[35] A. J. T. Lee, C-S. Wang, "An efficient algorithm for mining frequent inter-transaction patterns," *Elsevier International Journal of Information Sciences*, pp. 3453-3476, 2007

[36] Weiss M. (2009, April). *Mammogram Technique and Types*. [Online]. Available: http://www.breastcancer.org/symptoms/testing/types/mammograms/types.jsp

[37] MIAS Database, *The PCCV Project*: Benchmarking Vision Systems, http://peipa.essex.ac.uk/info/mias.html.

[38] Haralick R. M., Shanmugan K., Dinstein I., "Textural features for image classification," *IEEE Transactions on System Man and Cybernetics*, pp. 610-621, 1973

[39] Matlab Image Processing Toolbox.

[40] Keller, J.M., Gray, M.R., and Givens, J.A. (1985). "A fuzzy k-nearest neighbor algorithm," *IEEE Transaction on Systems, Man and Cybernetics,* 15 (4), 580

[41] Burges C. J. C., "A Tutorial on Support Vector Machines for Pattern Recognition," Data mining and Knowledge Discovery, Vol. 2, No. 2, pp. 121-167, 1998

[42] Thabtah F. "Challenges and Interesting Research Directions in Associative Classification," *ICDMW*, 2006.

[43] Wang J.Z. (2004, January) *Download Database for Research Comparison* [Online] Available: http://wang.ist.psu.edu/docs/related.shtml.

[44] Chen Y., Bi J., Wang J.Z., "MILES: Multiple Instance Learning via Embedded Instance Selection," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, Vol. 28, Issue 12, pp 1931-1947, 2006.

[45] Antonie, M.L., Zaiane, O.R., and Coman, A. (2001). "Application of Data mining Techniques for Medical Image Classification," *MDM/KDD*. (pp 94-101).

[46] Antonie, M.L., Zaiane, O.R., and Coman, A. (2003). "Associative classifiers for medical images," *LNICS 2797, MMCD*, Berlin/Heidelberg: Springer. (pp. 68-83)

[47] Yun, J., Zhanhuai L., Yong, W., Longbo, Z. (2005). "Joining associative classifier for medical images," Fifth international Conference on Hybrid Intelligent Systems, pp. 367-372, 2005.

[48] Haendel L. (January, 2004). The PNC2 Cluster Algorithm [Online]. Available: http://www.newty.de/pnc2/PNC2.html

[49] Y. Chen, J. Z. Wang, "Image Categorization by Learning and Reasoning with Regions," *Journal of Machine Learning research*, Vol. 5, pp. 913-939, 2004

[50] Andrews S., Tsochantaridis I., Hoffman T., "Support Vector Machines for Multiple-Instance Learning," *Advances in Neural Information Processing Systems 15*, pp. 561-568, 2003

[51] Csurka G., Bray C., Dance C., Fan L., "Visual Categorization with Bags of Keypoints," *Proc. ECCV '04 Workshop Statistical Learning in Computer Vision*, pp. 59-74, 2004

[52] Stock Database (2009, May). *Yahoo Finance: Yahoo Corporation*. [Online]. Available: http://finance.yahoo.com/.

[53] Weather Database. (2009, May)., Unedited Climatological Data [online] Available: http://cdo.ncdc.noaa.gov/ulcd/ULCD.

[54] Synthetic Climate Data. (2009, May) *Climactic Research Unit*. [Online]. Available: http://www.cru.uea.ac.uk/cru/projects/betwixt/cruwg_daily/.

[55] Jones P. (2004, February). BETWIXT Technical Briefing Note 1: The CRU Daily Weather Generator. [Online]. Available: http://www.cru.uea.ac.uk/projects/betwixt/documents /BETWIXT_TBN_1_v22.pdf.

[56] Kam P.S., Fu A.W.C., "Discovering Temporal Patterns for interval-Based Events," *Proc. Second Int'l Conf. Data Warehousing and Knowledge Discovery*, 2000.

[57] Dietterich T. G., "Ensemble Methods in machine Learning," *In Proceedings of the first international workshop on Multiple Classifier Systems, LNCS*, Vol. 1857, pp. 1-15, 2001

[58] Ferri C., Flach P., Hernandez J., "Delegating Classifiers," *21st International Conference on Machine learning*, pp. 27-36, 2004

[59] Q. Li, L. Feng, A. Wong, "From intra-transaction to generalized inter-transaction: Landscaping multidimensional contexts in association rule mining", *Elsevier Information Sciences, Vol 172, No. 3-4, 2005.*