Mining Negative Rules in Large Databases using GRD

by

Dhananjay R. Thiruvady, BComp

Thesis

Submitted by Dhananjay R. Thiruvady

in partial fulfillment of the Requirements for the Degree of

Bachelor of Computer Science with Honours (1608)

in the School of Computer Science and Software Engineering at

Monash University

Monash University

November, 2003

Contents

List of Tables

List of Figures

Mining Negative Rules in Large Databases using GRD

Dhananjay R. Thiruvady, BCompSc(Hons)
Monash University, 2003

Supervisor: Professor Geoffrey I. Webb

Abstract

Association Rule Discovery is an approach to mining rules from data in a database. Often, the database is a database of transactions. The rules generated will consist of strong associations between items within the database. An initial support constraint is applied to items to generate Association Rules. Further constraints, such as confidence, can be applied to the rules to generate interesting rules.

Generalized Rule Discovery (GRD) is an alternative rule discovery method to association rule discovery. GRD and association rule discovery share several features. GRD allows the user to specify constraints to generate rules without the need to specify the support constraint. An additional feature of GRD is that it generates *n* rules (user specified) that maximize a search measure. GRD uses Optimized Pruning for Unordered Search (OPUS) algorithm as its search method, which is an effective method for searching large unordered search spaces (space of rules in rule discovery). Using the association rule discovery approach to mine negative rules has been given some attention. Similarly, the GRD approach can be used to mine negative rules. By applying the Tidsets/Diffsets technique it is possible mining negative rules effectively and efficiently.

Implementing negative rules using GRD provides interesting results. The search space to explore can be very large for positive rules. In addition to positive rules, searching for negative rules increases the search space exponentially. Therefore, the execution times for the new system on all datasets are longer than the original system. The solution derived from the new system always contained some proportion of negative rules. The solution for some datasets only contained negative rules, as a result the execution time for these datasets are a lot longer on the new system.

Mining Negative Rules in Large Databases using GRD

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Dhananjay R. Thiruvady
November 5, 2003

Acknowledgments

Undertaking the honors year at Monash University has been a wonderful experience. In doing so there have been several factors that contributed to a successful year.

I would like to acknowledge the help, support and patience that my honors mates and house mates have provided. My supervisor, Prof. Geoff Webb was supportive and has guided me through a challenging project. The time he has devoted to me through the course of my project is invaluable. A special thanks to Archana, for her unconditional support during a tough period in her life.

I would finally like to mention how much the support and love of my parents and brother have meant to me. Without them I would not have had the motivation and drive to achieve my goals.

Dhananjay R. Thiruvady

Monash University
November 2003

Chapter 1

Introduction

1.1 Rule Discovery and Classification Learning

The aim of Rule Discovery techniques is to identify rules between items of a database. Rule discovery approaches that have developed are Classification Rule Discovery and Association Rule Discovery.

Classification rules are rules developed from a database that attempt to classify the data within the database. Association rules contrast with classification rules by attempting to identify unexpected regularities between items of the database. Transactional data is usually the data from which association rules are developed. Generalized Rule Discovery is an alternative approach to developing association rules.

Classification Learning is the process of assigning data items from a database into groups known as classes. The classification process can be either unsupervised or supervised classification. In unsupervised classification, a sample from the database is used to develop a model for the data items. Classes along with attribute values are generated as a part of the classification process. Each data item from the sample belongs to a class.

From unsupervised classification a model is developed. In supervised classification, each data item from the database has a class associated with itself. From the known classification for the data items a description for each class is found [11]. Decision Trees and Decision Graphs [22] are commonly used to describe classification or a class structure.

Classification learning is different from rule discovery as it focuses on classifying data in a database by creating classes for the data or testing to see if the data for a given class structure fits the class structure. Rule discovery has a different objective as it tries to develop rules that describe the inter-relationship between the data items from the database. Rule discovery when applied to large datasets has proved to perform well by successfully identifying unexpected interesting rules [1].

The objective and aim of this research project is to identify negative correlations in large databases using the GRD approach and to determine if generating the negative rules are feasible. Some rules developed by the GRD system are likely to be spurious and will result in Type 1 error [5]. Type 1 error can be minimized using statistical tests [9, 4, 12]. Filters to remove spurious rules can be applied to remove rules that are statistically insignificant. Generating these rules is not computationally infeasible as eliminating them is a simple post-process. Testing rules for Type 1 error is beyond the scope of this research project.

1.2 Significance and Research Method

The GRD approach will be used to investigate mining negative rules in large databases. GRD is chosen for two main reasons. The first reason is that it allows users to generate rules from itemsets without the need to specify the initial support constraint. Several rules which are useful to a user may satisfy other constraints such as confidence. The second reason is that GRD also has the option to generate a specific number of rules. A search measure such as leverage is specified and the rules with highest leverage will be developed.

The GRD system is modified to generate negative rules. The changes made to the system are to modify the algorithm it uses and to modify the current rule data structure. The are several other changes made to the system which are not discussed in detail.

1.3 Structure of Thesis

The thesis is organized as follows. The Literature Review (Chapter 2) discusses the relevant rule discovery approaches, negative rules and database formats including tidsets and diffsets. The Literature Review is followed by a Research Proposal (Chapter 3) which discusses the aims and objectives of the research conducted as well as its significance.

Chapter 4, the Methodology Chapter, discusses the research methods used to implement the modified GRD system, GRDI. The next chapter, Analysis of Experiment Results (Chapter 5), looks at the implications of the results derived from experiments conducted on the datasets with both systems.

The conclusion of the thesis follows in Chapter 6 with a discussion of the limitations of the new system and the possibilities for future work.

1.4 Conclusion

The introduction provides a brief description of previous work related to the thesis topic. Some aspects of Rule Discovery and Classification Learning are discussed and compared. The significance of the research undertaken is discussed along with the methods that are going to be adopted to implement the new system. The GRD approach will be used to develop negative rules. An outline of the topics presented in the thesis is also provided.

Chapter 2

Literature Review

2.1 Introduction

Rule discovery involves searching through a space of rules to determine rules of interest to a user. The rules are usually developed from very large databases. From these rules interesting rules are chosen by a user.

Association rule discovery aims to find rules between frequent items in a dataset. From the dataset, frequent items (literals which satisfy a minimum support constraint) are used to generate rules. These rules (the rule set) can then be pruned by using further constraints defined by the user. The final set of rules developed are said to be interesting rules to the user of the system. Association rule discovery and the terminology associated with it are described in Section 2.2. Some implementations of association rule discovery systems are also discussed.

Generalized Rule Discovery is an alternative rule discovery approach. The rules in GRD are developed based on user defined constraints. The support constraint need not apply to rules which are generated by GRD. This allows for rules to be generated based on several possible constraints including minimum support. GRD also allows users to specify the number of rules to be generated. GRD optimizes the search measure of the rules generated. A discussion of GRD is presented in detail in Section 2.3.

Mining negative rules from databases has proved to be useful. Using association rule discovery to mine negative rules has been previously researched. Some of the ways in which negative rules are mined are presented in Section 4.4.

To mine negative rules the diffsets technique proves useful by providing the means for calculating complement sets for itemsets efficiently. Database representations including tidsets and diffsets are discussed in Section 4.5.

2.2 Association Rule Discovery

Association rule discovery aims to find rules with strong associations between items from the database. It focuses on detecting relationships between the items [29]. Mining association rules in large databases was first approached by Agrawal, Imielinski and Swami [2]. A database of transactions is the database from which rules are generated.

A rule is of the form $A \Rightarrow B$ where A is known as the antecedent and B is the consequent of the rule. Both A and B are *itemsets* from the database of transactions. An itemset can be a single item (Example: water) or a set of items (Example: water and chips). The rule implies that if an itemset A occurs in a transaction then itemset B is likely to occur in the same transaction of the database.

The search space of rules generated from the database can also be very large, therefore to mine association rules from the database, constraints are defined [20]. For example, 1000 items in the database have 2^{1000} possible combinations of itemsets which results in a large number of rules to explore. The *minimum support* constraint is used to limit the number of itemsets that can be considered for rules to be generated.

The *support* of an itemset is the frequency with which the itemset occurs in the database. For example, if 25 transactions out of 100 transactions (assuming that a set from the database consists of 100 transactions) contain Pepsi, then the support of Pepsi is 0.25. The itemsets which satisfy the minimum support constraint are *frequent itemsets*. From these itemsets the rules are developed. If the minimum support is defined as 0.2 by the user, then Pepsi is a frequent item in the previous example as support (Pepsi) $\geq$ minimum support.

For example, consider the rule pepsi $\Rightarrow$ chips.

- Support (pepsi) = 0.4, implies that 40 percent of all customer transactions contain pepsi.
- Support (pepsi $\Rightarrow$ chips) = 0.2 implies that 20 percent of all customer transactions contain pepsi and chips together. [24]

From the rule set that is developed the user can choose to apply further constraints. The result is several rules will be pruned form the space of rules. Confidence is usually the measure of interest for generating association rules in association rule discovery. The final set of rules are referred to as interesting rules to the user. Some measures of interest which the user can specify are:

1. Confidence (A $\Rightarrow$ B) = support (A $\Rightarrow$ B) $\div$ support (A)
2. Lift (A $\Rightarrow$ B) = support (A $\Rightarrow$ B) $\div$ (support (A) $\times$ support (B))
3. Leverage (A $\Rightarrow$ B) = support (A $\Rightarrow$ B) - (support (A) $\times$ support (B))

Discovering association rules is a three part process:

1. Search the data space to find all the itemsets (can be a single item) whose support is greater than the user specified minimum support. These itemsets are the frequent itemsets.

2. Generate interesting rules based on the frequent itemsets. A rule is said to be interesting if its confidence is above a user's specified *minimum confidence*.

3. Remove (*prune*) all rules which are not interesting from the rule set. [19]

Further constraints can be applied to generate rules specific to a user's needs after the three part process is complete. Therefore interesting rules satisfy the minimum support constraint and any additional constraint defined by the user in association rule discovery.

Generating frequent itemsets is the part which requires high computation and needs to be efficient. When a database contains thousands of transactions, computing frequent itemsets can take a lot of time. Therefore, most research related to association rule discovery has been conducted to improve the frequent itemsets generation process. Specifically, most attention has been given to improve part one of the three part process.

Zaki et al. [34] show that developing association rules using samples from the dataset improves the speed of the mining process. The samples which are relatively small can be stored in main memory. Therefore the I/O overheads of multiple scans of the dataset, which are usually high, are reduced.

There are several algorithms which have been developed to mine association rules quickly. Some of them are described in the following sections.

2.2.1 Apriori

The *Apriori algorithm* was proposed by Agrawal and Srikant [3]. The algorithm has proved to be an efficient algorithm for mining association rules and has become the standard algorithm used for association rule discovery. Apriori follows a two step process to generate rules:

- The first step is to find all frequent itemsets. The itemset frequency information (support) is maintained. This step will limit the number of itemsets which are considered for the antecedent and consequent of the rules.

- From these frequent itemsets, association rules are generated.

All the items in the database are tested for minimum support. The frequent 1-itemsets found, known as a seed set, can be used to construct a candidate set of itemsets [3]. Sets with k - 1 items which are frequent can be joined to construct candidate sets with k items. Then the candidate set (k itemset) is tested to see if it satisfies minimum support and if it does it becomes a seed for the next pass. This iterative process continues until no frequent itemsets are found.

Some variants of the Apriori approach have showed that very few passes through the database may be necessary to generate association rules [16], [24]. An efficient implementation of the Apriori algorithm by Borgelt is *Apriori* [7].

2.2.2 Other Algorithms and Search Methods

There have been several algorithms proposed to solve the task of generating frequent itemsets efficiently. They include [16], [35], [14].

The Partition Algorithm

Savasere, Omiecinski and Navathe [16] proposed an efficient algorithm for mining association rules. The algorithm is known as the Partition Algorithm and works as follows. The algorithm scans the database twice. In the first scan the algorithm identifies a set of the potentially frequent itemsets. The set generated is a superset of all possible frequent itemsets and possibly some which are not. The second scan is then used to measure the support of each frequent itemset.

The partition algorithm experimentally has proved to be more efficient than the Apriori algorithm for large databases.

A Cluster-based Approach

Zaki, Parthasarathy, Ogihara and Li [35] presented six different algorithms. The algorithms employ three main techniques.

- Cluster the itemsets using equivalence classes or maximal hypergraph cliques - this will obtain potential frequent itemsets.
- From each cluster sublattice the true frequent itemsets are obtained using bottom-up, top-down or hybrid lattice traversal.
- Two different database layouts are considered - Horizontal layout and the Vertical layout (Discussed in Section 2.5).

For small databases, Apriori outperforms some of the proposed algorithms, but as the database gets large all six algorithms outperform Apriori.

Direct Hashing and Pruning Algorithm

Park, Chen and Yu [14] have proposed a Direct Hashing and Pruning (DHP) algorithm. The algorithm has two main features: it speeds up the frequent itemset generation process and reduces the transaction database size. The algorithm uses the hashing technique to filter out itemsets that cannot be used for the next candidate set generation. All candidate itemsets are inserted into a Hash Table after pruning the search space of rules. Each bucket in the hash table contains a number to represent the number of itemsets in the bucket so far.

DHP trims each transaction to reduce the transactions size. In addition DHP also prunes transactions from the database. For large databases, DHP has proved to be faster than Apriori.

2.3 Generalized Rule Discovery

Generalized Rule Discovery (GRD) was developed by Webb and Zhang [30]. Webb [27] argues that for some applications a direct search may be more effective than the two part process of the Apriori algorithm. The algorithm presented maintains the data in memory from which the association rules can be generated using alternative constraints defined by a user. This algorithm was the basis for the GRD approach.

GRD's aims are very similar to that of association rule discovery. As with association rule discovery, GRD searches through the database to generates rules. The rules are generated based on constraints specified by a user. However, unlike association rule discovery, the initial support constraint to generate frequent itemsets is not used by GRD.

In some applications minimum support may not be a relevant criterion to generate rules. For example, if a user wanted interesting rules with the highest leverage, then with association rule discovery the minimum support constraint will first be applied to get the frequent itemsets. Some rules which have very high leverage may not be considered as itemsets within the rules may not be frequent. As a result several interesting rules may not be developed.

The advantage of GRD approach is that it develops rules based on alternative constraints defined by the user [30]. The rules can be generated based on minimum support, but it is not an essential criterion for generating rules. Other constraints to generate interesting rules include minimum confidence, minimum leverage, and minimum lift. An additional constraint the user can specify is a specific number to be generated, n rules. The user also specifies a *search measure* and the rules generated by GRD will maximize this search measure. For example, a user specifies that 100 rules are to be generated with leverage as the search measure. GRD will then generate the 100 rules with maximum leverage.

The GRD approach has been implemented in the GRD program. The GRD program and the search algorithm it implements are described below.

2.3.1 The GRD System

The GRD System was developed by Webb and Zhang [30]. The data fed into the GRD system includes a header file with information about the data and the data file with all the transactions. The user then specifies all the constraints, the number of rules to be generated, the number of cases in the database and the number of items for the antecedent. Then the rules from the database will be generated and displayed with their statistics for support, etc. An example output of the GRD program is available in Appendix B.1. The GRD system has proved to be successful by efficiently rules with the highest possible leverage [30].

> An example input: associations <header file> <data file> -number-of-cases=5822 -minimum-strength=0.8 -minimum-support=0.01 -max-number-of-associations=1000 -maximum-LHS-size=4 -search-by-measure=leverage.

Associations is the name of the executable file. The header file contains information about the data, and all the data is contained in the data file. The -number-of-cases=5822 are the total number of cases from the data file which will be searched to generate the rules. -minimum-strength=0.8 and -minimum-supports=0.01 are the constraints applied to generate the rules. -max-number-of-associations=1000 and -search-by-measure=leverage limits GRD to finding 1000 associations with highest leverage. -maximum-LHS-size=4 specifies that the antecedent can comprise one to four items only.

2.3.2 OPUS Search Algorithm

The search space can often be very complex. For such a search space, previous methods employed a Heuristic Search [15, 26]. However, this search method does not necessarily find its target. Heuristic algorithms may also introduce a bias.

The algorithm used by GRD is the Optimized Pruning for Unordered Search (OPUS) algorithm developed by Webb [26]. This algorithm can be used for classification rule discovery and was originally developed for that purpose. It is an algorithm that guarantees to find the target it seeks.

The OPUS algorithm is an efficient search method that prunes parts of the search space that will not result in interesting rules [28, 30]. Once an itemset is known not to be in the solution the search space is restructured (pruned). Restructuring the search space and pruning uninteresting rules allows very fast access to rules which satisfy the minimum constraints.

Consider the search space in Figure 2.1. If it is determined that (b) is not a frequent itemset then all the supersets of (b) can be pruned from the search space (see Figure 2.2). This pruning method almost halves the search space below the itemset in the search space.

```
    |–(a)
()–|–(b)—(a,b)
    |–(c)–|–(a,c)
          |–(b,c)
```

Figure 2.1: Sample search space

```
    |–(a)
()–|–(b)— X
    |–(c)–|–(a,c)
          |–X
```

Figure 2.2: Pruning all nodes containing itemset (b)

Previous algorithms [13] pruned the search space under a particular itemset only (see Figure 2.3). Toivonen, et al. [23] have explored pruning using *rule covers*.

A commercial system that implements a variant of the OPUS algorithm is Magnum OPUS [25]. It develops rules that need not be based on the frequent itemsets and therefore differs from association rule discovery systems. The OPUS_AR algorithm [26] is used to develop the rules. An example of an adaptation of the OPUS algorithm is the Brute algorithm [18].

2.4 Mining Negative Rules

The main interest in association rule discovery has been to mine rules with strong associations. Such associations are known as positive associations. Finding positive associations

```
    |–(a)
()–|–(b)— X
    |–(c)–|–(a,c)
          |–(b,c)
```

Figure 2.3: Pruning a branch of the search space

is useful to make predictions about the database. For example, if the database contains transactions at a supermarket, predictions through positive associations can be used by the manager at the supermarket to improve their sales.

Mining negative rules has been given some attention and has proved to be useful. Brin, Motwani and Silverstein [8] first talked about mining negative associations between two itemsets. Savasere, Omiecinski and Navathe [17] use the method of generating positive rules from which negative rules are mined. The result is that there are fewer but more interesting negative rules that are mined.

Negative association rules are associations rules between the antecedent and consequent of the rule. Either the antecedent or consequent or both have to be negated in order for the rule to be a negative rule.

Assume that A and B are Itemsets. B is a single Itemset. Then the rules to be mined can be of the form:

1. $A \Rightarrow B$ (A implies B, as used in association rules)
2. $A \Rightarrow \neg B$ (A implies not B)
3. $\neg A \Rightarrow B$ (not A implies B)
4. $\neg A \Rightarrow \neg B$ (not A implies not B) [31]

The rules above specify concrete relationships between each itemset compared to [17] who look at the rule $A \not\Rightarrow B$.

In the rules specified above, either the antecedent or the consequent or both of them are negated. Another possibility is to consider itemsets within the antecedent or the consequent being negated. However, the GRD approach limits the consequent to a single condition. The rules to be considered are listed below. The second and third rules listed will not be explored as there is more than one condition in the consequent.

1. $(\neg A \wedge B) \Rightarrow C$
2. $A \Rightarrow (\neg C \wedge D)$
3. $(\neg A \wedge B) \Rightarrow (\neg C \wedge D)$

A large number of negative rules can be generated from the database with thousands of transactions. Most of these rules may not be of interest to a user. Therefore constraints will have to be applied to negative rules as they are applied to positive rules in association rule discovery and generalized rule discovery. An example of useful negative rules: "60 percent of customers who buy Potato Chips do not buy bottled water" [17]. This information can be used by the manager of a store to improve the store's marketing strategy [17].

Transaction 1:	Itemset A	Itemset B	Itemset C	
Transaction 2:	Itemset A			Itemset D
Transaction 3:	Itemset A	Itemset B	Itemset C	Itemset D

Table 2.1: Horizontal Mining: each transaction is stored with items that occur in it

Itemset A:	Itemset B:	Itemset C:	Itemset D:
Transaction 1	Transaction 1	Transaction 1	
Transaction 2			Transaction 2
Transaction 3	Transaction 3	Transaction 3	Transaction 3

Table 2.2: Vertical Mining for a given Class

2.5 Tidsets and Diffsets

Some datasets that are used to mine rules from are very large. To be able to mine rules efficiently the data needs to be presented in a manner so that it can be analyzed quickly. Models for storing large amounts of data so that they can be efficiently read are discussed by Zaiane and Han [32] and Srivastava, et al. [21].

In a database of transactions, each transaction has items associated with it. The transactions can be stored Horizontally (see Table 2.1) or Vertically (see Table 2.2). Vertical mining has proved to be more efficient than Horizontal mining.

Vertical mining has outperformed horizontal mining as it supports fast frequency counting on tidsets [33]. This is because data that are not necessary are automatically pruned and transaction frequencies can be calculated quickly to satisfy the minimum support constraint.

The transaction set that belongs to an itemset is known as a *Tidset*. A reference tidset is defined for a set of items. Zaki and Gouda [33] define a reference tidset as a *class*. Please note this special use of the term *class* in the definitions, below. A class has a set of transactions associated with itself. For example, the class items may occur in say three transactions (1, 2, and 3) out of total number of transactions, say a hundred. Each itemset is stored with the transactions it is contained in for a particular class (Example: 1, and 3). The transaction sets for the itemsets in this example can be seen in Table 2.2, itemsets B and C occur in transactions 1 and 3.

The tidset for a class is known as a *prefix tidset* or a *superset*. The itemsets from within the class are tested to see if they satisfy a minimum support constraint. Those itemsets that do not are omitted as they are considered infrequent. The itemsets that are frequent will

Itemset A:	Itemset B:	Itemset C:	Itemset D:
			Transaction 1
	Transaction 2	Transaction 2	

Table 2.3: Diffsets for itemsets in Table 2.2

occur in most of the transactions from the class that they are part of. From Table 2.2 it can be seen that each itemset is in at least two out of three transactions.

Zaki and Gouda [33] proposed that each itemset should be stored with their *diffsets* rather than their tidsets in the class that the tidset appears in. A diffset is a set of transactions that an itemset does not occur in within a given class. Since the itemsets are frequent within the class, the size of the tidset for the itemset is likely to be large, that is most of the transactions in the class. Therefore, the size of the diffset for an itemset is much smaller. The same information is contained in both representations and the diffsets approach results in saving a lot of memory.

From Table 2.3 it can be seen that for the same given class in Table 2.2 the size of the diffset representation is a lot smaller than that of the tidset representation. GRD calculates the tidset for an itemset from the database. This information is stored in memory when developing association rules.

2.6 Conclusion

Techniques to mine rules from large databases include Association Rule Discovery and Generalized Rule Discovery. Both association rule discovery and GRD attempt to find rules with strong associations between items in the database.

The first step in association rule discovery is to search through the database and find frequent itemsets. The frequent itemsets are used to generate a space of rules. From the space of rules, rules which do not meet constraints specified by the user are pruned. The GRD approach is different from association rule discovery as it develops rules from itemsets based on alternative constraints defined by the user, not necessarily the frequency of an itemset. GRD also has the option to generate a particular number of rules that maximize a search measure.

Mining negative rules in databases has been given some attention using the association rule discovery approach. Negative rules are rules where the antecedent or consequent or both are negative. These rules can be mined from the database using the GRD approach. The GRD approach will allow a user to specify the number of rules and generate rules based on alternative constraints. Rules that are developed can then be assessed for usefulness.

All itemsets in transactional data are stored with their corresponding transaction sets (tidsets). A diffset is the tidset for the negation of an itemset. Using this information, the negation of itemsets can be calculated with very little additional computation. Similarly, the details for a negative rule, such as support and confidence, can be calculated quickly.

GRD currently calculates the tidsets for the itemset in the database. Using the GRD approach with the tidsets/diffsets technique, it is possible to mine positive rules and negative rules that satisfy user specified constraints effectively.

Chapter 3

Research Proposal

3.1 Significance of Research

Previous work related to mining negative rules in databases was done by Savasere, et al. [17] and Wu, et al. [31] using association rule discovery. In large databases, several itemsets may not satisfy the minimum support constraint. However, rules with these infrequent itemsets may prove to be interesting.

For example, an association between vodka and caviar may be of interest in market basket analysis. Since both vodka and caviar are infrequently bought they would not be discovered by an association rule discovery system. The rule Vodka $\Rightarrow$ Caviar might have high confidence and therefore be interesting to a user. Since GRD allows rules to be generated based on minimum constraints defined by the user, the user can define a minimum confidence value as the constraint. If Vodka $\Rightarrow$ Caviar satisfies the confidence constraint, then the GRD system generates this rule along with all other interesting rules which satisfy this constraint.

Using GRD to generate negative rules allows the user to view positive and negative associations between rules. Negative correlations within a database may be of interest to users. GRD also allows a user to generate a particular number of rules with the highest value of a specified search measure. The user can then search the rule space for interesting rules by defining further constraints.

3.2 Research Objective

The objective of this project is to develop negative rules using the GRD approach. The original GRD system was developed by Webb and Zhang [30]. GRD is chosen to implement the project as the minimum support constraint need not be applied to develop the rules. The aim of this project is to determine whether or not the solution developed with positive and negative rules is computationally tractable. The aim and objective are achieved by modifying the GRD system to generate negative rules and comparing the modified system with the original system. If a constraint on the number of rules is placed it is possible that a major number of rules generated are negative. The rules generated, positive and negative will be those rules with the highest value of the search measure.

3.3 Conclusion

GRD is the approach chosen to mine negative rules from large databases. Mining negative rules from databases using association rule discovery has been approached by Savasere, et al. [17] and Wu, et al. [31]. The reason GRD is chosen is that it allows a user to mine rules irrespective of the frequency of the itemsets within the dataset. An additional advantage is that only n rules that maximize a search measure are generated. This constraint allows the user to compare rules with the highest value of the search measure.

The research methods used to develop negative rules using GRD are described in Chapter 4 which follows next.

Chapter 4

Methodology

4.1 Introduction

The GRD approach will be used to develop negative rules from the database. Using association rule discovery to develop negative rules has been given some attention [8, 17, 31]. The main reasons for using the GRD approach are so that the minimum support constraint need not apply and the number of rules to be generated can be limited to a fixed number.

The core of the search for association rules lies with the OPUS search function. To developed association rules the OPUS algorithm is modified from the OPUS algorithm proposed by Webb [26]. To implement negative rules, there are further changes to be made to the OPUS algorithm. The OPUS algorithms are discussed in Chapter 2.

A rule's data structure consists of a left-hand-side (LHS) set and right-hand-side (RHS) set. A second LHS set is added to the rule's data structure which holds the set of negative antecedents for the rule. Since the RHS of a rule is limited to a single consequent, a flag within the rule is sufficient to indicate whether the consequent of a rule is negative or not. Section 4.3 addresses both rule data structures in detail.

To implement negative rules, the OPUS algorithm is modified and the rule's data structure changes. The GRD system calculates transaction sets for all available itemsets. Since a rule can include a negative set of antecedents, the tidsets for the negative items need to be calculated. This is done using the diffsets approach. The negative consequent does not require calculations for tidsets and diffsets as negative consequent sets are never needed. The calculations are explained in Section 4.4.

4.2 Modifying the GRD Algorithm

4.2.1 The GRD Algorithm

The GRD algorithm is a modified version of the OPUS search algorithm. OPUS aims at searching through the space of subsets of a dataset, whereas GRD aims to search through the space of pairs of antecedent conditions and consequent conditions. GRD performs the OPUS search for all potential antecedents and for each antecedent (could be a set of conditions) the set of consequent conditions are explored. The consequent conditions are limited to single condition to simplify the search.

User defined constraints determine which rules are part of the solution. The constraints include support, confidence, etc. and are used to prune the search space. GRD also allows the user to specify a particular number of rules which maximize a search measure. The solution will contain a maximum number of rules as specified by the user.

Webb and Zhang [30] define the Generalized Rule Discovery task or GRDtask by *4-tuple* *<A, C, D, M>* and rules are of the form $X \Rightarrow Y$.

- A: is the set of antecedent conditions, A is nonempty.
- C: is the set of consequent conditions, C is nonempty.
- D: is the set of records, where d $\in D$, conditions(d) are the set of conditions that are applied to d and conditions(d) $\subseteq$ A $\cup$ C. D is nonempty.
- M: is the set of constraints that result in the solution to the GRD task.
- X: is the set of antecedent conditions, X is nonempty.
- Y: is the set of consequent conditions, Y is nonempty.

The *solution* *<A, C, D, M>* $\rightarrow X \Rightarrow Y$ is a many-to-one function mapping a GRDtask to its solution, satisfying solution(*<A, C, D, M>*) = $\{X \Rightarrow Y \mid X \subseteq A \wedge Y \subseteq C \wedge X \Rightarrow Y$ satisfies all constraints in M with respect to $D\}$.

The GRD algorithm is a recursive function with three parameters. The three parameters are:

- CurrentLHS: the set of conditions that are currently considered for the antecedent of the rule. CurrentLHS is initialized to $\emptyset$.
- AvailableLHS: the set of conditions that may be added to the antecedent of the rule. AvailableLHS is initialized to A (antecedent conditions).
- AvailableRHS: the set of conditions that could be the consequent of the rule. AvailableRHS is initialized to C (consequent conditions).

A global variable, currentSolution consists of the set of rules that are part of the solution so far, which is initialized to 0. The GRD algorithm starts with a single condition in the antecedent. Further conditions are added to the antecedent as the algorithm tries to determine possible rules. From AvailableLHS, antecedent conditions are added to CurrentLHS to form NewLHS. Each available condition in AvalilableRHS is tested with NewLHS to determine whether the rule NewLHS $\Rightarrow$ c can be in the solution. At this stage a recursive call is made to the GRD function with NewLHS, NewAvailableLHS (created pruning AvailableLHS) and NewAvailableRHS (created pruning AvailableRHS) as the parameters.

The solution contains rules with the maximum value of the search measure. Line 13 of the algorithm in Figure 4.1 removes a rule if the number of rules exceeds MaxRules (specified by user). The rule that is removed is the rule which is MaxRules + 1. This will be the rule with the lowest value of the search measure. The pruning sections of the algorithm are omitted. To see the complete algorithm with pruning, refer to [30]. Positive aoociation rules are generated with this algorithm.

```
GRD (CurrentLHS, AvailableLHS, AvailableRHS)
1 :   SoFar = $\emptyset$
2 :   for all P in AvailableLHS do
3 :       NewLHS = CurrentLHS $\cup$ P
4 :       NewAvailableLHS = SoFar
5 :       If P in AvailableLHS then
6 :           NewAvailableRHS = AvailableRHS - P
7 :       else
8 :           NewAvailableRHS = AvailableRHS
9 :       end if
10:           for all Q in NewAvailableRHS do
11:           if insolution (NewLHS $\Rightarrow$ Q, <A, C, D, M $\wedge$ X $\Rightarrow$ Y $\in$
                currentSolution U {NewLHS $\Rightarrow$ Q}>) then
12:               add NewLHS $\Rightarrow$ Q to currentSolution
13:           remove from currentSolution any rule W $\Rightarrow$ Z: $\neg$ insolution (W $\Rightarrow$ Z,
                <A, C, D, M $\wedge$ X $\Rightarrow$ Y $\in$ currentSolution U {NewLHS $\Rightarrow$ Q}>)
14:           end if
15:       end for
16:       if NewAvailableLHS != 0 and NewAvailableRHS != 0 then
17:           GRD (NewLHS , NewAvailableLHS, NewAvailableRHS)
18:       end if
19:       SoFar = SoFar $\cup$ (P)
20:   end for
```

Figure 4.1: The GRD algorithm, [30]

```
GRDI (CurrentLHS, AvailableLHS, AvailableRHS, AvailNegLHS, AvailNegRHS)
1 :   SoFar = $\emptyset$
2 :   for all P in AvailableLHS do
3 :        CreateRules (P, AvailableRHS, AvailNegRHS, AvailableLHS,
               NewLHS, NewAvailableLHS, NewAvailableRHS, NewAvailNegRHS, SoFar)
4 :        if NewAvailableLHS != 0 and NewAvailableRHS != 0 then
5 :             GRD (NewLHS , NewAvailableLHS, NewAvailableRHS,
                    NewAvailNegLHS, NewAvailNegRHS)
6 :        end if
7 :        SoFar = SoFar $\cup$ (P)
8 :   end for
9 :   for P in AvailNegLHS do
10:        CreateRules (P, AvailableRHS, AvailNegRHS, AvailNegLHS,
               NewLHS, NewAvailNegLHS, NewAvailableRHS, NewAvailNegRHS, SoFar)
11:        if NewAvailNegLHS != 0 and NewAvailableRHS != 0 then
12:             GRD (NewLHS , NewAvailableLHS, NewAvailableRHS,
                    NewAvailNegLHS, NewAvailNegRHS)
13:        end if
14:        SoFar = SoFar $\cup$ (P)
15:   end for

CreateRules (P, AvailableRHS, AvailNegRHS, AvailableLHS, NewLHS,
NewAvailableLHS, NewAvailableRHS, NewAvailNegRHS, SoFar)
1 :        NewLHS = CurrentLHS $\cup$ P
2 :        NewAvailableLHS = SoFar
3 :        If P in AvailableLHS then
4 :             NewAvailableRHS = AvailableRHS - P
5 :             NewAvailNegRHS = AvailNegRHS - P
6 :        else
7 :             NewAvailableRHS = AvailableRHS
8 :             NewAvailNegRHS = AvailNegRHS
9 :        end if
10:        for all Q in NewAvailableRHS do
11:             if insolution (NewLHS $\Rightarrow$ Q, <A, C, D, M $\wedge$ X $\Rightarrow$ Y $\in$
                  currentSolution U {NewLHS $\Rightarrow$ Q}>) then
12:                  add NewLHS $\Rightarrow$ Q to currentSolution
13:             remove from currentSolution any rule W $\Rightarrow$ Z: $\neg$ insolution (W $\Rightarrow$ Z,
                  <A, C, D, M $\wedge$ X $\Rightarrow$ Y $\in$ currentSolution U {NewLHS $\Rightarrow$ Q}>)
14:             end if
15:        end for
16:        for all Q in NewAvailNegRHS do
17:             if insolution (NewLHS $\Rightarrow$ Q, <A, C, D, M $\wedge$ X $\Rightarrow$ Y $\in$
                  currentSolution U {NewLHS $\Rightarrow$ Q}>) then
18:                  add NewLHS $\Rightarrow$ Q to currentSolution
19:             remove from currentSolution any rule W $\Rightarrow$ Z: $\neg$ insolution (W $\Rightarrow$ Z,
20:             <A, C, D, M $\wedge$ X $\Rightarrow$ Y $\in$ currentSolution U {NewLHS $\Rightarrow$ Q}>)
21:             end if
22:        end for
```

Figure 4.2: The modified GRD algorithm and the CreateRules function

4.2.2 The Modified GRD Algorithm

The new algorithm to implement negative rules includes two additional input sets to the GRD function. The two sets are AvailNegLHS (negative antecedent set) and AvailNegRHS (negative consequent set).

Each condition in the positive antecedent set (AvailableLHS) is added to CurrentLHS to form NewLHS. For each condition in AvailableRHS, the rules of the form NewLHS $\Rightarrow$ c are explored and for each condition in AvailNegRHS, the rules NewLHS $\Rightarrow \neg$c are explored. Similarly, each condition from the negative antecedent (AvailNegLHS) set is tested with positive and negative consequent conditions from AvailableRHS and AvailNegRHS respectively. The algorithm to implement positive and negative rules is presented in Figure 4.2.

The additional procedure CreateRules can also be viewed in Figure 4.2. The parameters passed to CreateRules are listed below. Along with the parameters either (val) or (ref) are listed, which means that that particular parameter was called by value or reference respectively.

- P (val): is the current antecedent condition to be added to NewLHS.
- AvailableRHS (val): is the set of available conditions for the positive consequent.
- AvailNegRHS (val): is the set of available conditions for the negative consequent.
- AvailableLHS (val): is the set of available conditions for the antecedent. Depending on where the call was made from, AvailableLHS can be a positive set or a negative set.
- NewLHS (ref): is the new antecedent set which is the intersection of the current antecedent set, CurrentLHS, and the new antecedent condition, P.
- NewAvailableLHS (ref): is the new available antecedent set, which is set to SoFar.
- NewAvailNegLHS (ref): is the new available negative antecedent set, which is set to SoFar.
- NewAvailableRHS (ref): is the new available positive consequent set.
- NewAvailNegRHS (ref): is the new available negative consequent set.
- SoFar (val): is the set of antecedent conditions observed so far.

If A, B are antecedent conditions and C is a consequent condition, then the rules that will be explored are of the form:

1. A $\wedge$ B $\Rightarrow$ C
2. A $\wedge$ B $\Rightarrow \neg$C

3. A $\wedge$ ¬B $\Rightarrow$ C
4. A $\wedge$ ¬B $\Rightarrow$ ¬C
5. ¬A $\wedge$ B $\Rightarrow$ C
6. ¬A $\wedge$ B $\Rightarrow$ ¬C
7. ¬A $\wedge$ ¬B $\Rightarrow$ C
8. ¬A $\wedge$ ¬B $\Rightarrow$ ¬C

Rules that satisfy the constraints are added to currentSolution. The rule with the lowest value of the search measure is removed from the solution if the number of rules exceeds MaxRules.

4.3 Modifying the Rule Data Structure

4.3.1 The GRD Rule Data Structure

A rule that is to be added to currentSolution has a data structure as presented in Figure 4.3. The data structure includes a pointer to the next rule which effectively creates a rule list for currentSolution. The number of cases covered by the rule is the number of cases covered by the antecedent of the rule. This information is used to calculate the coverage (support of antecedent) of the rule, e.g. rule $\rightarrow$ coverage = rule $\rightarrow$ no_of_cases_covered $\div$ total_cases. The rule also has float values for strength (confidence), support, lift and leverage. All these statistics can be calculated using the support for the antecedent and the consequent of the rule. There are two sets included in the data structure, An_set is the set of positive antecedents and Con_set is a single set for the consequent. If A is the antecedent and C is the consequent then the statistics can be calculated as follows:

- Strength (A $\Rightarrow$ C) = support (A $\Rightarrow$ C) $\div$ support (A)
- Lift (A $\Rightarrow$ C) = support (A $\Rightarrow$ C) $\div$ (support (A) $\times$ support (C))
- Leverage (A $\Rightarrow$ C) = support (A $\Rightarrow$ C) - (support (A) $\times$ support (C))

4.3.2 The Modified Rule Data Structure

The modified data structure for a rule includes a negative antecedent set, negAn_set. The positive antecedent set is posAn_set labelled differently from the rule data structure in Figure 4.3. Negative consequents do not need an additional set as they can be stored in the same set for the positive consequents and the flag neg_con is used to determine whether

```
RULE_TYPE:

    RULE_TYPE *next;            /* Pointer to next rule in list */
    int no_of_cases_covered;    /* Cases covered by antecedent of rule */
    float coverage;             /* Coverage of rule (Support (antecedent)) */
    float strength;             /* Strength of rule (Confidence of rule) */
    float support;              /* Support of the rule */
    float lift;                 /* Lift of the rule */
    float leverage;             /* Leverage of the rule */
    set An_set;                 /* Set of antecedents */
    set Con_set;                /* Set of consequents */
```

Figure 4.3: The GRD rule data structure

the consequent set is negative or positive. The modified RULE_TYPE data structure is presented in Figure 4.4.

The new rule data structure has an additional negative set because rules of the form $A \wedge \neg B \Rightarrow C$ can be represented in the data structure by rule $\rightarrow$ posAn_set = $\{A\}$ and rule $\rightarrow$ negAn_set = $\{\neg B\}$. However, if the consequent is negative, then the negative consequent can be placed in the same location for the positive consequent. In the case of $A \wedge B \Rightarrow \neg C$ the rule data structure's consequent set will appear as rule $\rightarrow$ Con_set = $\{\neg C\}$.

4.4 Negative Rules Implementation

4.4.1 Calculations for Statistics and Negative Sets

The algorithm for the GRD system is modified to implement negative rules. The data structure is also modified to be able to store negative antecedent sets within a rule. Statistics within the rule data structure include coverage, support, strength, lift and leverage. To calculate these statistics the antecedent set size and the consequent set size are needed.

The consequent tidsets are stored in memory, therefore their sizes are easily attainable. The size for a negative consequent set is size (Superset) - size (Consequent Tidset). The tidsets for a negative consequent are not needed and therefore not calculated. Calculating the antecedent set sizes can also be done in this straightforward manner. However, to create NewLHS, negative antecedent sets are needed. The method by which negative sets are calculated is described described below.

GRD maintains the tidsets for positive items and the tidsets for negative items are required when a negative item is added to the antecedent. Calculating the tidset of negative items by

RULE_TYPE:

```
    RULE_TYPE *next;            /* Pointer to next rule in list */
    int no_of_cases_covered;    /* Cases covered by antecedent of rule */
    float coverage;             /* Coverage of rule (Support (antecedent)) */
    float strength;             /* Strength of rule (Confidence of rule) */
    float support;              /* Support of the rule */
    float lift;                 /* Lift of the rule */
    float leverage;             /* Leverage of the rule */
    set posAn_set;              /* Set of positive antecedents */
    set negAn_set;              /* Set of negative antecedents */
    set Con_set;                /* Set of consequents */
    int neg_con;                /* Negative Consequent Flag */
```

Figure 4.4: The modified rule data structure

negating the positive itemset could result in a lot of additional computation. An effective method of calculating these sets are by using diffsets.

If CurrentLHS contains an itemset and a new negative itemset is intersected with CurrentLHS to create NewLHS, then the diffsets technique proves useful. Instead of calculating the complement of the new set and intersecting the complement set with CurrentLHS, the difference between CurrentLHS and the new set provides the set for the intersection between the negative itemset and CurrentLHS. The difference set is stored in NewLHS. Using diffsets, the additional computational time is lower than if the complement set were calculated. Some calculations to efficiently calculate tidsets and support are shown later on in this section. The only occasion when complement sets are needed is when CurrentLHS is $\emptyset$ and an itemset needs to be added to CurrentLHS to form NewLHS. This is done by complementing the current tidset for an itemset and placing the complement set in CurrentLHS.

The following method is used to calculate the tidsets for negative itemsets and support of a rule. Similarly, strength, lift and leverage can be without using complement sets.

Assume the Reference tidset = T, Tidset for antecedent = A, Tidset for consequent = C, Tidset for antecedent and consequent = A $\wedge$ C.

1. For the rule A $\Rightarrow$ C, GRD computes:
    - Tidset (A)
    - Tidset (C)
    - Tidset (A $\wedge$ C)
    - Support (A $\Rightarrow$ C)

2. For the rule A $\Rightarrow$ ¬C, GRDI can compute:
    - Tidset (A)
    - Tidset (¬C) = Diffset (T, C) = T - Tidset (C)
    - Tidset (A $\wedge$ ¬C) = Diffset (A, C) = Tidset (A) - Tidset (A $\Rightarrow$ C)
    - Support (A $\wedge$ ¬C) = Support (A) - Support (A $\Rightarrow$ C)
3. For the rule ¬A $\Rightarrow$ C, GRDI can compute:
    - Tidset (¬A) = Diffset (T, A) = T - Tidset (A)
    - Tidset (C)
    - Tidset Tidset (¬A $\wedge$ C) = Diffset (C, A) = Tidset (C) - Tidset (A $\Rightarrow$ C)
    - Support (¬A $\wedge$ C) = Support (C) - Support (A $\Rightarrow$ C)
4. For the rule ¬A $\Rightarrow$ ¬C, GRDI can compute:
    - Tidset (¬A) = Diffset (T, A) = T - Tidset (A)
    - Tidset (¬C) = Diffset (T, C) = T - Tidset (C)
    - Tidset (¬A $\wedge$ ¬C) = Diffset (T, (A u C)) = T - (Tidset (A) + (Tidset (C) - Tidset (A $\Rightarrow$ C)))
    - Support (¬A $\wedge$ ¬C) = T - (Support (A) + (Support (C) - Support (A $\Rightarrow$ C)))
5. For the rule (A $\wedge$ ¬B) $\Rightarrow$ C, where either in the antecedent or consequent an itemset is negated, GRDI can compute:
    - Tidset (A)
    - Tidset (¬B) = Diffset (T, B) = T - Tidset (B)
    - Tidset (C)
    - Tidset (A $\wedge$ ¬B $\wedge$ C) = Diffset(A $\wedge$ C, B) = T - (Tidset(A $\wedge$ C) - Tidset(B))
    - Support (A $\wedge$ ¬B $\Rightarrow$ C) = (Support (A) - (Support (A $\wedge$ B)) $\wedge$ Support (C)

For each rule above, the calculation of the diffsets can be done through some combination of the calculations already done by the GRD system. Not calculating the complement sets results in relatively low additional computation.

An example output of positive and negative rules from both systems is presented in Appendix B. All the rule generated are not displayed.

4.4.2 Pruning the Search Space

Pruning in the new algorithm applies to the positive and negative sets as it applies to the positive set in the original GRD algorithm. Pruning conditions are defined in [30] based on whether they apply to the antecedent set or the consequent set. To cater for negative rules the pruning functions are modified accordingly. An example of a pruning condition is as follows:

> Consider GRDtask = <*A, C, D, M*>, for any P $\in$ AvailableLHS, P can be pruned from SoFar if Coverage(P) < minCoverage.

The full set of pruning conditions are available from [30]. The pruning condition above is extended for the negative antecedent set as follows:

> Consider GRDtask = <*A, C, D, M*>, for any P $\in$ AvailableLHS, P can be pruned from SoFar if Coverage(P) < minCoverage and for any R $\in$ AvailNegLHS, R can be pruned from SoFar if Coverage(R) < minCoverage.

Each pruning function for the antecedent and consequent is modified appropriately. This ensures that pruning conditions are applied to a negative set in the same manner as they are applied to positive set.

4.5 Conclusion

This chapter discussed the research method used to be able to implement negative rules within the GRD system. The GRD algorithm is modified by introducing two new sets, one for negative antecedents and the other for negative consequents. The new algorithm explores all possible conditions for the antecedent and for the consequent of a rule if the conditions are not pruned.

The data structure used for rules by the GRD system is modified so that negative antecedent sets can be stored within the structure. However, the negative consequents being single conditions can be stored in the same location as the positive consequents. A flag is introduced to the data structure to indicate whether a positive or negative consequent is being considered.

The only occasion when a complement set for an itemset needs to be calculated is when a new element is being added to the antecedent of a rule. On all other occasions the tidsets for negative sets are calculated using the diffset technique (positive itemset's tidset and its superset). The pruning functions within the GRD system are appropriately modified to prune negative conditions and positive conditions.

Chapter 5

Analysis of Experiment Results

5.1 Introduction

The modified GRD program is referred to as *GRDI (GRD new Implementation)*. Experiments were carried out on ten datasets with the modified GRD system. Most of the datasets used were the same datasets used for the comparison of the GRD system with Apriori in [30].

A short description of the datasets used follows in the next section. The records, values and attributes of each dataset are provided.

The experiments performed on the datasets provided interesting results. The experiments were run on the GRD system and the modified GRD system, GRDI. These results are discussed in Section 3. They are analyzed to identify where the additional computation comes from in the new system. The analysis includes a statistical test to identify correlations between execution times, leverage values of rules and rules evaluated.

5.2 Datasets

Most of the datasets used to test the systems were used for previous research [30]. There were ten datasets used for experiments. Nine out of the ten datasets are taken from the UCI Machine Learning and KDD repositories [6, 10]. The other dataset, ticdata2000 is a market-basket dataset used in research by Zheng, et al. [36] for Association Rule Discovery. Three sub ranges were created for numeric attributes. Each sub range approximately contained one third of the records.

The datasets varied from relatively small datasets to large ones based on the number of records they contained. The records, attributes and values of each dataset are listed in Table 5.1.

Data Files	*Records*	*Attributes*	*Values*
connect4	67,557	43	129
covtype	581,012	55	125
ipums.la.99	88,443	61	1883
letter-recognition	20,000	17	74
mush	8,124	23	127
pendigits	10,992	17	58
shuttle	58,000	10	34
soybean-large	307	36	119
splice junction	3,177	61	243
ticdata2000	5,822	86	709

Table 5.1: Datasets used for Experiments

5.3 Comparison of GRD and GRDI

GRD is the original system which develops positive association rules. GRDI (*GRD new Implementation*) is a modified GRD system which generates both negative and positive rules.

The number of itemsets to keep track of is important when developing rules from GRD. If the number of conditions available for the antecedent and consequent is 1000, then there are 2^{1000} possible combinations of itemsets that can be considered. Incorporating negative rules into the GRD system results in an increase in the number of combinations of itemsets to 2^{2000}. This amounts to an exponential increase in the number of itemsets considered and therefore a much greater number of rules explored.

The experiments were carried out on the same computer for both systems. The computer used was a Linux server, had a processor speed of 1.20 GHz and main memory of 256 MB RAM.

All the conditions in the datasets were allowed in the antecedent and the consequent of the rule for the experiments. The input for both programs were the same. An example input for an experiment to be performed on the shuttle dataset with GRD was:

```
associations ../data/shuttle.hdr ../data/shuttle.data -number-of-cases=58000 -
search-by-measure=leverage -nontrivial=0 -minimum-strength=0.8 -minimum-
support=0.01 -max-number-of-associations=1000 -maximum-LHS-size=4 -minimum-
lift=1.0
```

The same values were used with GRDI for the shuttle dataset. Header files contain information about the structure of the data describing the values of an attribute.

The constraints specified in this input and all other inputs were the same inputs used by Webb and Zhang [30] to conduct research when comparing Apriori with GRD. The constraints were specifically chosen so that the execution times of both programs on the datasets are not too long. For any new datasets used appropriate input measures were defined. For example, setting minimum strength to a lower value, 0.5, might result in a lot longer execution time compared to minimum strength at 0.8. Through investigation it was observed that the execution time was greater. GRD takes 5 seconds to execute at strength at 0.5 compared to 1 second when strength is set at 0.8.

In all experiments GRD and GRDI search for the 1000 rules (-max-number-of-associations=1000) with the highest value of the search measure, Leverage (search-by-measure=leverage). The maximum number of conditions available on the left-hand-side was 4 (-maximum-LHS-size=4) and both systems assume that only a single condition was available for the right-hand-side. This will simplify the search task.

Other constraints specified were minimum Strength (-minimum-strength=0.8), rules which have Strength higher than 0.8 were considered to potentially be added to the solution. Similarly, minimum Support (-minimum-support=0.01) and minimum Lift were defined. These constraints were specified so that the execution times of the programs on the datasets were not too long for experiment purposes.

5.3.1 Computational Time

The executions time for GRD and GRDI are presented in Table 5.2. Some of the observations from the results are:

1. GRD: Execution times for some large datasets (large number of records) are very short and some are very long. e.g. connect4 has 67,557 records and requires 20 seconds to develop rules, whereas ipums.la.99 has 88,443 records takes only 7 seconds.

2. GRDI: for most datasets GRDI's execution time is slightly greater than GRD, e.g. mush. However, some datasets require a lot longer execution times for GRDI than GRD, e.g. ticdata2002.

The reason for large increase in the computational time for some datasets (ipums.la.99) is primarily due to the increase in the size of the search space. If a small number of rules are negative then the execution time is only a little greater. If a majority of rules are negative (sometimes all) then the execution times are a lot greater. The increase in execution time is directly proportionate to the increase in size of the search space.

The Execution times for GRD and GRDI on all the datasets can be compared more easily with a line graph presented in Figure 5.1, created with Microsoft Excel 2000. A logarithmic scale along the x-axis is used to be able to view datasets with relatively low execution times.

Data Files	*Records*	*GRD*	*GRDI*	*Ratio*
connect4	67,557	20	106	5.30
covtype	581,012	835	1976	2.37
ipums.la.99	88,443	7	1634	233.43
letter-recognition	20,000	1	34	34.00
mush	8,124	1	8	8.00
pendigits	10,992	1	28	28.00
shuttle	58,000	1	11	11.00
soybean-large	307	1	4	4.00
splice junction	3,177	6	1872	312.00
ticdata2000	5,822	7	647	92.43

Table 5.2: Execution times of GRD and GRDI

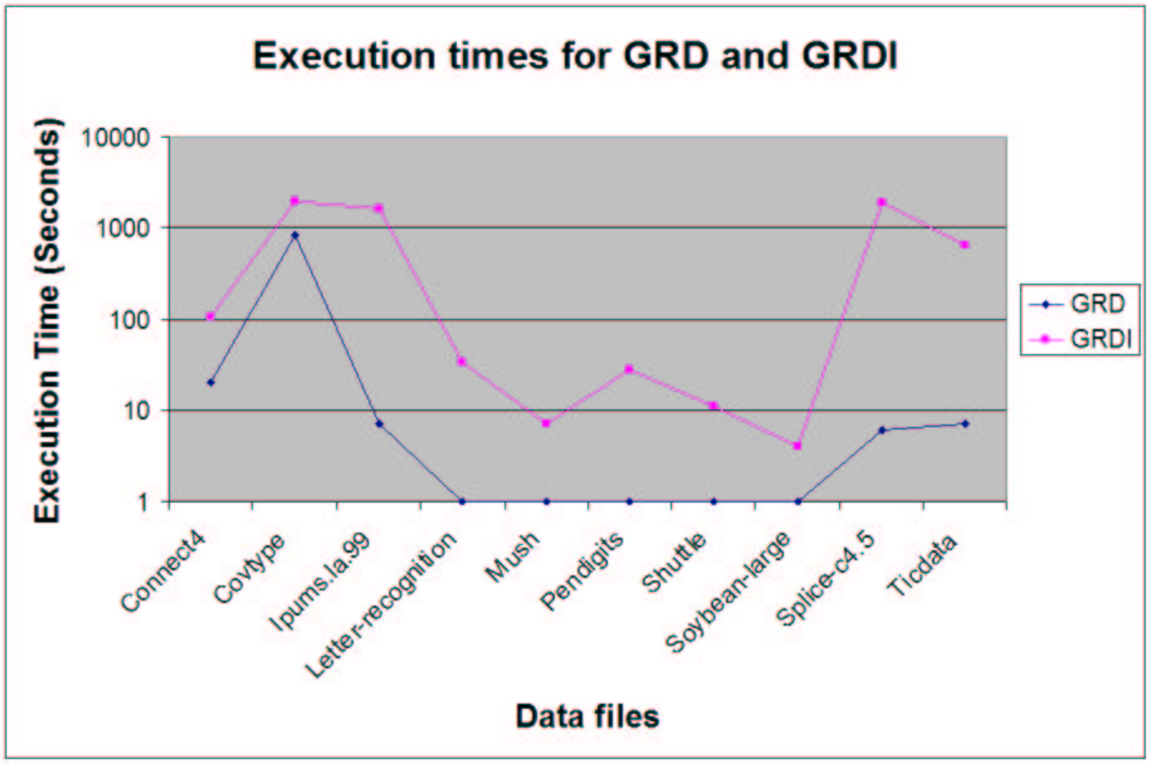

Figure 5.1: Comparison of Execution Times

	GRD			*GRDI*		
Data Files	*min. lev.*	*max. lev.*	*mean*	*min. lev.*	*max. lev.*	*mean*
connect4	0.1224	0.1227	0.1225	0.1688	0.1707	0.1698
covtype	0.1083	0.1743	0.1413	0.2459	0.2474	0.2467
ipums.la.99	0.2080	0.2484	0.2282	0.2499	0.2500	0.2500
letter-recognition	0.0455	0.1459	0.0957	0.1020	0.1499	0.1395
mush	0.1558	0.2109	0.1833	0.1994	0.4930	0.3390
pendigits	0.0615	0.1757	0.1186	0.1050	0.1832	0.1441
shuttle	0.0409	0.1599	0.1004	0.0911	0.2040	0.1766
soybean-large	0.2137	0.2359	0.2248	0.2286	0.6182	0.4324
splice junction	0.0404	0.1523	0.0963	0.1244	0.1733	0.1489
ticdata2000	0.1899	0.1922	0.1910	0.2184	0.5341	0.3763

Table 5.3: Comparison of Minimum and Maximum Leverage values

5.3.2 Analysis of Rules Generated

The execution times for several datasets were greater for GRDI compared to GRD. Comparing the leverage values for rules generated by both systems is indicative of the type of rule generated by both systems. Leverage is the difference between the joint frequency of the antecedent and consequent (Support (A $\Rightarrow$ C)) and the frequency if they were independent (Support (A) $\times$ Support (Y)).

The main observations about the rules are:

1. Both Systems: The leverage values for the rules from GRDI were greater than the values for the rules generated from GRD.
2. GRDI: Some proportion of all the rules in all datasets were negative rules. Some solutions consisted only of negative rules.

Consider the rules generated by GRD for connect4. The rules generated had a maximum leverage value of 0.1227. For GRDI, the minimum leverage for a rule was 0.1688, which shows that all 1000 rules generated by GRDI were negative rules. Table 5.3 lists the minimum and maximum leverage values for GRD and GRDI. The comparison of minimum leverage values of the rules generated by both systems shows that GRDI always contains negative rules in its solution. For the datasets in which GRDI's execution times were a lot longer than GRD, rules with much higher leverage were also generated. This is also true of the maximum leverage values. A comparison of the minimum and maximum leverage values are presented in a graph in Figure 5.2. An important observation is that for several datasets the minimum leverage value of GRDI is greater than the maximum leverage value of GRD. The information contained within the observation is that all the rules generated for those datasets are negative rules.

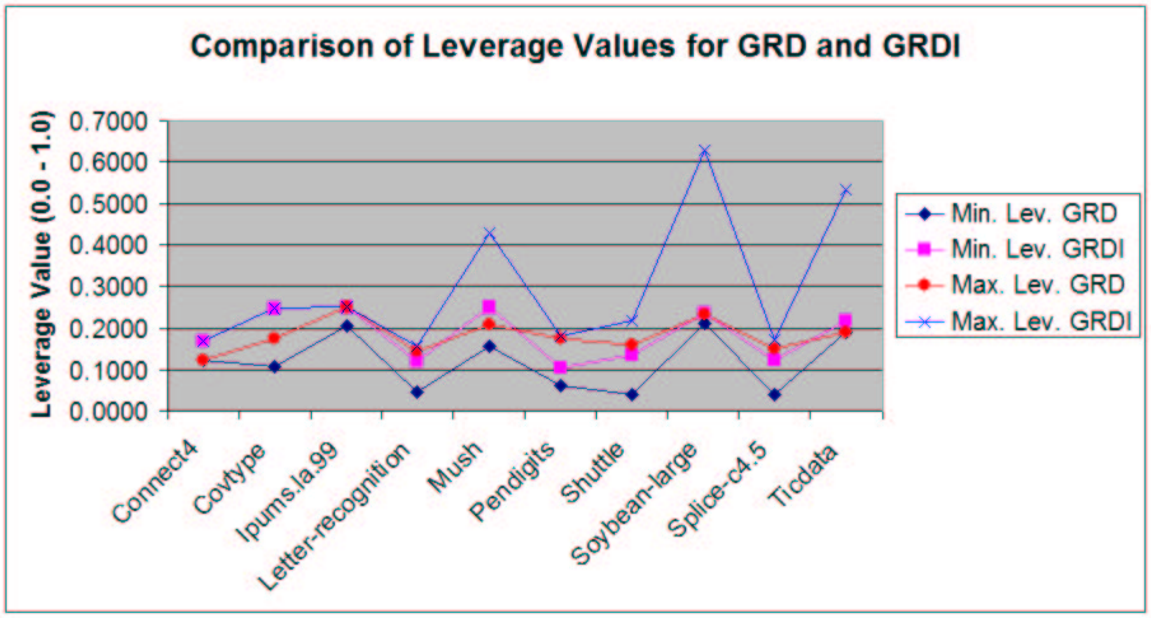

Figure 5.2: Comparison of Leverage of GRD and GRDI

5.3.3 Statistical Tests

Statistical tests are useful to determine whether or not correlations exist between Execution Times (see to Table 5.2), Leverage Values (see to Table 5.3)and the Number of Rules Evaluated by both systems. The number of rules evaluated and the number of data accesses are listed in Appendix C. The Coefficient of Correlation statistical test was applied to the results obtained from the experiments conducted. The interesting correlations are pointed out below.

1. Between the execution times for both systems, there exists a relatively strong positive correlation (0.558). The implication of this correlation is that when the execution time for one system is high, it is likely that the execution time for the other system will also be high.

2. There is a negative correlation between the execution times for both systems and the mean leverage values (see to Table 5.3) of rules generated by both systems (GRD: -0.0609, GRDI: -0.0986). The negative correlation is very weak however. These values imply that if a system is to take a long time to execute the leverage values are likely to be low. If it takes less time to execute the leverage values are likely to be high.

3. A positive correlation exists between the execution times for both systems and the number of rules evaluated. This correlation shows that more rules evaluated result in longer execution times.
    - GRD(Execution Time, Rules Evaluated): 0.5506
    - GRDI(Execution Time, Rules Evaluated): 0.5383

5.4 Conclusion

The modified GRD system is called GRDI (GRD new Implementation). To develop negative rules the number of itemsets considered increases exponentially which results in a larger number of rules to explore. Both systems GRD and GRDI are tested on several datasets to examine their execution times and the rules that they develop.

Ten datasets are used to test GRD and GRDI. Most of the datasets used are the same ones used for the comparison of Apriori and GRD in [30]. Nine out of the ten datasets are used from the UCI Machine Learning and KDD repositories, and the other one is the same datasets used for research by Zheng, et al. [36].

To compare GRD and GRDI the execution times for both systems are noted and observed. On some datasets GRDI requires a lot more computation time that GRD. Comparing the leverage values of the rules generated by GRD and GRDI shows that negative rules are almost always a part of the solution (high leverage compared to positive rules generated by GRD). This implies that the datasets used contained a lot more negative rules than positive rules. When execution time for GRDI is greater than that of GRD it is observed that a majority of rules or all the rules are negative in the solution.

The reason that GRDI takes a lot longer to execute on some datasets is because GRDI is searching through many more rules. The result is additional computation and the computation is a lot more if most of the rules to be generated are negative.

The Coefficient of Correlation statistical test shows that there is a positive correlation between the execution time of the two systems. There also exists a positive correlation between the execution times and the number of rules evaluated, but between the execution times and leverage values of a rule is a negative correlation.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

Mining negative rules in large databases has proved to be useful. A specific area is market basket analysis where rule discovery techniques can be used to improve sales for supermarkets. A rule of the form Chips $\Rightarrow$ (Soft Drink A) and Chips $\Rightarrow$ ($\neg$Soft Drink B) can be used by a manager of a supermarket to increase their sales. Using association rule discovery, mining negative rules has already been explored [17, 31].

The research objective of this project was to develop negative rules using the GRD approach. There are two main reasons for choosing to use GRD. The first reason is that the minimum support constraint is not an essential criterion for rule generation. Several rules which consist of infrequent itemsets could be interesting as the rules might have a high value of an alternative statistic. Such rules will not be generated using association rule discovery. A user has the freedom to specify alternative constraints to generate rules in GRD.

The second reason for choosing GRD is because the GRD method allows users to generate a specific number of rules that maximize a particular search measure. Constraints can still be applied to limit the space of rules that are searched. On most occasions, the rules with the highest value of the search measure will be the interesting rules. Therefore, generating n rules allows the user to look through a subset of the possible rules. As a result a lot of time can be saved by the user when analyzing the rules developed.

The diffsets technique is employed by the modified system to calculate tidsets for the negation of an itemset. Instead of calculating the complement set of an itemset, it is possible to generated the complement set using diffsets. For example, if A and $\neg$C are the antecedent and consequent of the rule respectively, then the tidset($\neg$C) is T - tidset(C) where T is the superset of C. Since GRD calculates tidsets for all itemsets, using diffsets results in a lower computational time.

The changes made to the GRD system are mainly to the GRD algorithm, the rule data structure and the pruning functions. The GRD algorithm is modified to iterate through a second antecedent set of negative items. Within this iteration a second consequent set of negative items is explored in addition to the positive consequent set.

The rule data structure is modified to include negative set for the antecedent, so that rules of the form $A \wedge \neg B \Rightarrow C$ can include within their data structure the set for A and the set for $\neg B$. The consequent, being a single condition does not require the rule to include an additional consequent set.

A comparison of GRD and GRDI is carried out. The results show that for several datasets GRDI took a lot longer to execute than GRD. The reason for this increase in execution time is because these particular datasets contained many more negative rules than positive rules. With the increased size of the search space, the execution times are bound to be longer for GRDI. The leverage values of rules generated show that the rules generated by the new system had higher leverage than rules developed by original system. This indicates that most rules generated for the datasets are negative rules. The research aim is to identify if the additional computation involved in developing negative rules is tractable. The outcome of the research undertaken is that the solution to the GRDtask (generating positive and negative rules) is computationally tractable.

Quantities for each item of the antecedent or consequent of a rule are currently handled by the system. Quantitative attributes will result in more interesting rules. For example, if a rule such as Tea $\Rightarrow \neg$Coffee exists then including specific quantities will generate a rule which is more useful. A resultant rule would appear as: 10 packets of Tea $\Rightarrow \neg$Coffee.

There are some limitations that were observed during the implementation of GRDI. There is also potential for future work regarding mining negative rules. Both these aspects are described in the following sections.

6.2 Limitations of the GRDI System

There following limitations of the GRDI system when generating negative rules were observed.

1. The conditions available for the consequent of a rule are limited to a single itemset. Even though all possible conditions are explored at the right-hand-side of a rule, including multiple conditions in the consequent of the rule may lead to interesting rules. However, the system is likely to be a lot more complex if multiple conditions for the consequent are allowed.

2. Rules of the form $\neg(A \wedge B) \Rightarrow C$ are not generated by the modified system. This is because of the need to repeat the loop for the negative antecedent set (see Figure 4.1), which would not be computationally efficient.

3. The execution times of GRDI could improve if negative tidsets are maintained in memory. This will however lead to large amounts of memory used by the program during execution.

4. If the constraint values are low, then there is a possibility of both positive and negative rules of the form Tea $\Rightarrow$ Coffee and Tea $\Rightarrow$ $\neg$Coffee appearing in the solution of the system. The information contained in the rules is not of much use to a user. A solution to this problem exists within the GRD approach. The user can choose to specify a fewer number of rules to be generated (Example: specify 500 rules instead of 1000). Since these rules are the rules with the highest value of the search measure, then it is likely that one of the rules mentioned above is going to be part of the solution and the other is not. Another solution to this problem is to raise the constraint values which will more likely result in one of the rules being eliminated.

5. Several rules generated by GRDI are likely to be spurious rules. There are efficient filters which can eliminate the rules which are statistically insignificant.The filters available require low execution times and can therefore be applied without worrying about additional computation.

6.3 Future Work

There is potential for future work in the area of negative rules developed using the GRD approach.

1. The rules developed by GRDI include both positive and negative rules. Assessing the usefulness of the negative rules developed will help users of the system. Eliminating spurious rules will also allow the user analyze the rules quickly.

2. If GRD is asked to generate 1000 rules from a retail store with the search measure as Lift, then those rules with the highest Lift value will be generated. Consider the rule Mobile Phone $\Rightarrow$ Mobile Phone Plan. It is possible that the first 500 rules generated are of the form Mobile Phone $\Rightarrow$ Mobile Phone Plan, between different mobile phones and mobile phone plans. The rules developed in this situation are not as interesting as expected.

   Grouping of association rules by clustering is discussed by Toivonen, et al. [23]. The motivation for this approach was to be able to view the large number or rules in manageable parts. This method provides a potential solution to such a problem.

   Another solution that is possible is to group rules of a particular category into a single variable to create a *Variable Hierarchy*. A user than has the capability of viewing rules which are independent of each other from different variables. The rules observed would be more interesting positive and negative rules.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database mining: A performance perspective. In Nick Cercone and Mas Tsuchiya, editors, *Special Issue on Learning and Discovery in Knowledge-Based Databases*, pages 914–925. Institute of Electrical and Electronics Engineers, Washington, U.S.A., 1993.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[4] Kamal Ali, Stefanos Manganaris, and Ramakrishnan Srikant. Partial classification using association rules. In *Knowledge Discovery and Data Mining*, pages 115–118, 1997.

[5] Gerald Keller Anthony Selvanathan, Saroja Selvanathan and Brian Warrack. *Australian Business Statistics*. Wadsworth Publishing Company, 2 edition, 2000.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/mlrepository.html, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[7] Christian Borgelt. Apriori (software). http://fuzzy.cs.uni-magdeburg.de/ borgelt/, Accessed 20/6/2003.

[8] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: generalizing association rules to correlations. In *Proc. of the ACM SIGMOD Conf.*, pages 265–276, 1997.

[9] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[10] S. Hettich and S. D. Bay. The UCI KDD archive. http://www.ics.uci.edu/, University of California, Irvine, Dept. of Information and Computer Sciences, 1999.

[11] Marcel Holsheimer and Arno P. J. M. Siebes. Data mining: The search for knowledge in databases. Technical Report CS-R9406, CWI, P.O. Box 94079, 1090 GB Amsterdam, 1994.

[12] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In Nabil R. Adam, Bharat K. Bhargava, and Yelena Yesha, editors, *Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401–407. ACM Press, 1994.

[13] Shinichi Morishita and Akihiro Nakaya. Parallel branch-and-bound graph search for correlated association rules. In *Large-Scale Parallel Data Mining*, pages 127–144, 1999.

[14] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash based algorithm for mining association rules. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 175–186, San Jose, California, 22–25 1995.

[15] Stuart Russell and Peter Norvig. *Artifcial Intelligence: A modern Approach.* Prentice Hall, 1995.

[16] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *The VLDB Journal*, pages 432–444, 1995.

[17] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. Mining for strong negative associations in a large database of customer transactions. In *ICDE*, pages 494–502, 1998.

[18] Richard B. Segal. *Machine Learning as Massive Search.* PhD thesis, University of Washington, 1997.

[19] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. *Future Generation Computer Systems*, 13(2–3):161–180, 1997.

[20] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 67–73. AAAI Press, 14–17 1997.

[21] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[22] Peter Tan and David Dowe. MML Inference of Decision Graphs with Multi-way Joins. In *Proc. 15th Australian Joint Conference on Artificial Intelligence*, pages 131–142. Springer-Verlag, 2002.

[23] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping of discovered association rules. In *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases.*, pages 47–52, 1995.

[24] Hannu Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *In Proc. 1996 Int. Conf. Very Large Data Bases*, pages 134–145. Morgan Kaufman, 09 1996.

[25] Geoffrey Webb, Shane Butler, and Doug Newlands. On detecting differences between groups. In *In Proc. of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 256–265. ACM, 2003.

[26] Geoffrey I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

[27] Geoffrey I Webb. Efficient search for association rules. In *Knowledge Discovery and Data Mining*, pages 99–107, 2000.

[28] Geoffrey I. Webb. Discovering associations with numeric variables. In *Knowledge Discovery and Data Mining*, pages 383–388, 2001.

[29] Geoffrey I Webb. *Association Rules*, chapter 2, pages 25–39. Lawrence Erlbaum Associates, 2003.

[30] Geoffrey I Webb and Songmao Zhang. Beyond association rules: Generalized rule discovery. In *Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, 14–17 2002.

[31] X. Wu, C. Zhang, and S. Zhang. Mining both positive and negative rules. *Machine Learning: proceedings of the nineteenth international conference*, pages 658–665, 2002.

[32] O. Zaiane and J. Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *In Proc. First Int. Conf. On Knowledge Discovery and Data Mining, Montreal, Canada, 1995.*, 1995.

[33] M. Zaki and K. Gouda. Fast vertical mining using diffsets. Technical Report 01-1, Rensselaer Polytechnic Institute, 2001.

[34] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Wei Li, and Mitsunori Ogihara. Evaluation of sampling for data mining of association rules. Technical Report TR617, University of Rochester, 1996.

[35] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. Technical Report TR651, The University of Rochestor, 1997.

[36] Zijian Zheng, Ron Kohavi, and Llew Mason. Real world performance of association rule algorithms. In Foster Provost and Ramakrishnan Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406, 2001.

Appendix A

Glossary of Terms

Term	*Explanation*
Antecedent	The set of conditions in the left-hand-side of a rule
Association Rules	Rules that indicate associations between itemsets of a database.
Class	The superset of a set of items.
Consequent	The set of conditions in the right-hand-side of a rule
Diffsets	The difference set between the superset and the transaction set.
Itemset	A conjunction of a set of items. *Note: could be a single item.*
Leverage	The difference between the joint frequency of the antecedent and consequent and the frequency of the the antecedent and consequent independently.
Lift	The joint frequency of the antecedent and consequent over the frequency of the antecedent and consequent independently.
Market Basket Analysis	Analysis of data obtained from a supermarket. Market basket data is represented as transactional data.
Negative Rules	Rules that consist of a negative antecedent or negative consequent or both.
OPUS	The Optimized Pruning for Unordered Search algorithm.
Search Measure	A measure which is used to search through the space rules. The rules generated maximize the search measure.
Support	The frequency of an itemset with respect to its superset.
Tidset	The transaction set of an item or itemset.

Appendix B

Example Outputs

B.1 Output from GRD for mush.data

```
DUAR - Deakin University Association Rules System.

Copyright (c) 1987-2000 Deakin University.

Fri Oct 31 13:07:44 2003

Arguments: ../data/mush.nam ../data/mush.data -number-of-cases=8124
-out-file=OutFiles/output-mushtest2.txt -search-by-measure=leverage -nontrivial=0
-minimum-strength=0.8 -minimum-support=0.01 -max-number-of-associations=10
-maximum-LHS-size=4 -minimum-lift=1.0

Maximum number of associations = 10
Maximum number of attributes on LHS = 4
Minimum coverage = 0.000
Minimum strength = 0.800
Minimum support = 0.010
Minimum lift = 1.000
Minimum leverage = 0.000
Trivial association rules are allowed Search by leverage


Association rules:

1.IF gill-spacing is c
     veil-color is w
     ring-type is p
THEN bruises? is t
[coverage= 0.405 (3288); strength= 0.937 ; support= 0.379 ; lift= 2.25 ;
leverage= 0.2109 (1713)]
```

```
2.IF gill-spacing is c
       veil-type is p
       veil-color is w
       ring-type is p
THEN bruises? is t
[coverage= 0.405 (3288); strength= 0.937 ; support= 0.379 ; lift= 2.25 ;
leverage= 0.2109 (1713)]
```

(All ten rules are not displayed)

```
10 association rules from 8124 cases

23 attributes and 127 attribute-value pairs

Execution time: 0 seconds (+ 0 input time)

Number of rules evaluated: 1794; Number of data accesses: 6642
```

B.2 Output from GRDI for mush.data

```
DUAR - Deakin University Association Rules System.
Copyright (c) 1987-2000 Deakin University.
Fri Oct 31 12:44:21 2003
Arguments: ../data/mush.nam ../data/mush.data -number-of-cases=8124
-out-file=OutFiles/output-mushtest2.txt -search-by-measure=leverage
-nontrivial=0 -minimum-strength=0.8 -minimum-support=0.01
-max-number-of-associations=10 -maximum-LHS-size=4 -minimum-lift=1.0
Maximum number of associations = 10
Maximum number of attributes on LHS = 4
Minimum coverage = 0.000
Minimum strength = 0.800
Minimum support = 0.010
Minimum lift = 1.000
Minimum leverage = 0.000
Trivial association rules are allowed Search by leverage
```

```
Association rules:

1.IF ¬stalk-color-above-ring is o
     ¬stalk-color-below-ring is o
THEN ¬population is v
[coverage= 0.976 (7932); strength= 0.953 ; support= 0.931 ; lift= 1.90 ;
leverage= 0.4397 (3572)]

2.IF ¬stalk-color-above-ring is o
     ¬stalk-color-below-ring is o
     veil-type is p
THEN ¬population is v
[coverage= 0.976 (7932); strength= 0.953 ; support= 0.931 ; lift= 1.90 ;
leverage= 0.4397 (3572)]
```

(All ten rules are not displayed)

```
10 association rules from 8124 cases

23 attributes and 127 attribute-value pairs

Execution time: 3 seconds (+ 1 input time)

Number of rules evaluated: 25109; Number of data accesses: 344481
```

Appendix C

Number of Data Accesses and Rules Evaluated

	Rules Evaluated		*Data Accesses*	
Data Files	*GRD*	*GRDI*	*GRD*	*GRDI*
connect4	46901	232080	316760	2336819
covtype	260367	149765	1248020	3859382
ipums.la.99	20012	1807053	93743	27433259
letter-recognition	75410	1455969	297256	5788870
mush	18031	122781	66382	968313
pendigits	113792	1135709	448557	6467684
shuttle	14928	78352	64162	455003
soybean-large	43144	278426	166292	3060069
splice junction	3080724	12251714	9460546	308737486
ticdata2000	189965	2711896	795346	104838282