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

October, 2003
# Contents

List of Tables ................................................................. 4

List of Figures ................................................................. 5

Abstract .................................................................................. 6

1 Introduction ........................................................................... 7

2 Literature Review ............................................................... 8
   2.1 Introduction .................................................................... 8
   2.2 Association Rule Discovery .............................................. 8
      2.2.1 Apriori ............................................................... 10
      2.2.2 Other algorithms and search methods ....................... 10
   2.3 Generalized Rule discovery ............................................ 12
      2.3.1 The GRD program ............................................... 12
      2.3.2 OPUS search algorithm ...................................... 13
   2.4 Mining Negative Rules .................................................. 14
   2.5 Tidsets and diffsets ...................................................... 15
   2.6 Conclusion ................................................................. 17

3 Research Proposal .............................................................. 18
   3.1 Significance of research ............................................... 18
   3.2 Research Objective ..................................................... 18
   3.3 Conclusion ................................................................. 19
CONTENTS

4 Methodology ................................................................. 20
  4.1 Introduction .......................................................... 20
  4.2 Modifying the GRD algorithm ....................................... 21
    4.2.1 The GRD Algorithm ............................................. 21
    4.2.2 The modified GRD algorithm .................................. 23
  4.3 Modifying the Rule Data Structure ................................. 25
    4.3.1 The GRD Rule Data Structure ................................. 25
    4.3.2 The modified Rule Data Structure ............................. 25
  4.4 Implementing Negative Rules in the GRD system ................... 27
    4.4.1 Calculating Tidsets and Support ............................. 27
    4.4.2 Pruning ....................................................... 28
  4.5 Conclusion ........................................................... 28

5 Analysis of Experiment Results ........................................ 30
  5.1 Introduction .......................................................... 30
  5.2 Datasets ............................................................... 30
  5.3 Comparison of GRD and GRDI ....................................... 31
    5.3.1 Computational Time ........................................... 32
    5.3.2 Rules Generated ............................................... 32
  5.4 Conclusion ........................................................... 34

6 Conclusions and Future Work .......................................... 36
  6.1 Conclusion ........................................................... 36
  6.2 Limitations of the GRDI system .................................... 37
  6.3 Future Work .......................................................... 38

References ................................................................. 39

Appendix A Glossary of Terms ........................................... 42

Appendix B Example Output: Output from connect4.data .............. 43

Appendix C An example of a modified Pruning Function .............. 44
List of Tables

2.1 Horizontal Mining: each transaction is stored with items that occur in it . . 16
2.2 Vertical Mining for a given Class . . . . . . . . . . . . . . . . . . . . . . . 16
2.3 Diffsets for itemsets in Table 2.2 . . . . . . . . . . . . . . . . . . . . . . . . 17

5.1 Datasets used for Experiments . . . . . . . . . . . . . . . . . . . . . . . . . 31
5.2 Comparison of Execution Times . . . . . . . . . . . . . . . . . . . . . . . . . 33
5.3 Comparison Minimum and Maximum Leverage values . . . . . . . . . . . . 34
List of Figures

2.1 Sample search space .................................................. 13
2.2 Pruning all nodes containing itemset (b) .......................... 14
2.3 Pruning a branch of the search space ............................... 14

4.1 The GRD algorithm, [21] .................................................. 22
4.2 The modified GRD algorithm including the CreateRules function ........................................... 24
4.3 A sample rule data structure ............................................ 26
4.4 The modified rule data structure ....................................... 26

5.1 Execution times of GRD and GRDI ................................. 33
5.2 Execution times of GRD and GRDI ................................. 35
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 be based on strong associations between items within the training data. Strong associations are defined based on the minimum support constraint. Generalized Rule Discovery (GRD) is an alternate 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, whereas a minimum support has to be specified in association rule discovery. The search method used by GRD, Optimized Pruning for Unordered Search (OPUS) algorithm for an unordered search, 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 and applying the Tidsets/Diffsets approach mining negative rules can be done effectively and efficiently. The rules generated by the GRD program can then be assessed for usefulness.
Chapter 1

Introduction

Classification learning is the process of assigning data items from a dataset into groups known as classes. The classification process can be either unsupervised or supervised classification. In unsupervised classification, a sample from the training data 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 training data has a class associated with itself. From the known classification for the data items a description for each class is found [7]. Decision trees 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 dataset 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 a dataset. With very large datasets rule discovery has proved to perform well [1].

The primary aim of this project is to find negative rules in large databases using the GRD approach. The negative rules will then be tested to see if they are of potential interest or not.

Some rules developed by GRD may be spurious and will result in Type 1 error. Type 1 error can be minimized using statistical tests, however, this will not be an aim of this project.
Chapter 2

Literature Review

2.1 Introduction

Rule discovery involves searching through a space of rules to determine rules of interest to a user. Very large datasets are often used to develop the rules. The rules developed are the space of rules from which interesting rules are chosen.

Association rule discovery aims to find rules between frequent items in a dataset. From the dataset, frequent items (literals which satisfy minimum support) are used to generate rules. These rules (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.

Generalized rule discovery (GRD) is an alternative to association rule discovery. The rules in GRD are developed based on user defined constraints. The support constraint need not be applied 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 will optimize the rules generated based on constraints specified by the user.

2.2 Association Rule Discovery

The aim of association rule discovery is to find rules with strong associations between items from the training data. It focuses on detecting relationships between the items [20]. Mining association rules in large databases was first approached by Agrawal, Imielinski and Swami [2]. A database of transactions is the training data 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 training data.

For example: consider the rule water ⇒ chips

- Support (water) = 0.4, implies that 40 percent of all customer transactions contain water.
- Support (water ⇒ chips) = 0.2 implies that 20 percent of all customer transactions contain water and chips together.
- Confidence (water ⇒ chips) = 0.8, implies that 80 percent of customers who bought water also bought chips.

The search space of rules generated from the training data can also be very large, therefore to mine association rules from the training data, constraints have to be defined. For example, 1000 items in the training data 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 training data. For example, if 25 transactions out of 100 transactions (assuming that a set from the training data includes 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. From the rule set that is developed the user can choose to apply further constraints. As a result several rules will be pruned from the space of rules. The final set of rules are referred to as interesting rules to the user.

Some measures of interest are:

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

Confidence is usually the measure of interest for generating association rules in association rule discovery.

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. [14]

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 the training data has 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 of the attention has been given to improve part one of the three part process.

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. An itemset is frequent if it satisfies the minimum support constraint. The itemset frequency information (support) is also 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 training data 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 large itemsets are found.

Some variants of the Apriori approach have showed that very few passes through the training data may be necessary to generate association rules [11], [16].

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 [11], [24], [9].
CHAPTER 2. LITERATURE REVIEW

The Partition Algorithm

Savasere, Omiecinski and Navathe [11] proposed an efficient algorithm for mining association rules. The algorithm is known as the Partition Algorithm and works as follows. The algorithm scans the training data 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 training data.

A Cluster-based Approach

Zaki, Parthasarathy, Ogihara and Li [24] 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 training data, Apriori outperforms some of the proposed algorithms, but as the training data gets large all six algorithms outperform Apriori.

Direct Hashing and Pruning Algorithm

Park, Chen and Yu [9] 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 (training data) 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 training data, DHP has proved to be faster than Apriori.
2.3 Generalized Rule discovery

Generalized Rule Discovery (GRD) was developed by Webb and Zhang [21]. Webb [19] 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 minimum constraints. 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 training data and 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 generated.

The advantage of GRD approach the is that it generates rules based on alternative constraints defined by a user [21]. The rules can be generated based on minimum support of the itemsets, 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, \( n \) rules to be generated. The user also has to specify 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 program

The GRD program was developed by Webb and Zhang [21]. The training data fed into the GRD program 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 training data and the number of items for the antecedent. Then the rules from the training data will be generated and displayed with the interest statistics. The GRD program has proved to be successful when attempting to find rules with the highest possible leverage [21].
Figure 2.1: Sample search space

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 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-support=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 algorithm used by GRD is the Optimized Pruning for Unordered Search (OPUS) algorithm developed by [18]. 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 search space can often be very complex. For such a search space, a heuristic search is usually employed [10], [18]. However, the search does not necessarily find its target. Heuristic algorithms may also introduce a bias.

The OPUS algorithm is an efficient search method that prunes parts of the search space that will not result in interesting rules. Once an infrequent itemset has been discovered 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.

For example 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. Previous algorithms [8] pruned the search space under a particular itemset only (see Figure 2.3).
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 is useful to make predictions about the training data. For example, if the training data 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 [5] first talked about mining negative associations between two itemsets. Savasere, Omiecinski and Navathe [12] 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.

Magnum OPUS [17] is a rule-discovery application. It develops rules that need not be based on the frequent itemsets and therefore differs from association rule discovery. The OPUS-AR algorithm [18] is used to develop the rules. The Brute [13] algorithm is an example of an adaptation of the OPUS algorithm.
CHAPTER 2. LITERATURE REVIEW

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) [22]

The rules above specify concrete relationships between each itemset compared to [12] who look at the rule \( A \neg \Rightarrow B \).

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. Therefore, the second and third rules listed will not be explored. Then the rules to be considered are of the form:

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

A large number of negative rules can be generated from training data 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” [12]. This information can be used by the manager of a store to improve the store’s marketing strategy [12].

2.5 Tidsets and diffsets

In a database of transactions, each transaction has items associated with it. The transactions can be stored Horizontally (see 2.1) or Vertically (see 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 [23]. 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 [23] define the reference tidset as a class.
Table 2.1: Horizontal Mining: each transaction is stored with items that occur in it

| 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 2.2: Vertical Mining for a given Class

| 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 |

The 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, itemsets B and C occur in transactions 1 and 3.

The tidset for a class is known as a prefix tidset. 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 occur in most of the transactions from the class that they are part of. From Table 2 it can be seen that each itemset is in at least two out of three transactions.

Zaki and Gouda [23] 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 Figure 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 training data. This information is stored in memory when developing association rules.
Table 2.3: Diffsets for itemsets in Table 2.2

<table>
<thead>
<tr>
<th>Itemset A:</th>
<th>Itemset B:</th>
<th>Itemset C:</th>
<th>Itemset D:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction 1</td>
<td>Transaction 2</td>
<td>Transaction 2</td>
<td></td>
</tr>
</tbody>
</table>

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 training data.

The first step in Association rule discovery is to search through the training data 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 training data using the GRD approach. The GRD approach will allow the 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 (tid-sets). 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 training data. 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 [12] and [22] 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 confidence as a 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. The negative correlations 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 GRD. GRD is chosen to implement the project as the minimum support constraint need not be applied to develop the rules. The original GRD system was developed by [21]. The aim of this project is to modify the GRD system to generate negative rules along with positive rules. 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 [12] and [22]. 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 Methodology chapter which follows next.
Chapter 4

Methodology

4.1 Introduction

The GRD approach will be used to generate negative rules from the training data. Using association rule discovery to generate negative rules has been given some attention [5], [12] and [22]. The main reasons for using the GRD approach are so that the minimum support constraint does not have to be specified 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 develop association rules the OPUS algorithm is modified from the OPUS algorithm proposed by [18]. To implement negative rules, there are further changes to be made to the OPUS algorithm. The OPUS algorithms are discussed in Chapter 2.

A rules data structure consists of a left-hand-side (LHS) set and right-hand-side (RHS) set. A second LHS set (negAn_Set) is added to the rules 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 needs to be modified and the rules data structure needs to change. The GRD system calculates transaction sets (tidsets) for all available itemsets. Since the rule data structure includes a negative set of antecedents, the tidsets for the negative antecedent needs to be calculated. This is done using the diffsets approach. The negative consequent does not require calculations for diffsets. 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 is aimed 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.

User defined constraints determine which rules are part of the solution. The constraints include support, confidence, etc. and they are used to prune the search space. GRD also allows the user to specify n rules which maximize a particular search measure.

The Generalized Rule Discovery task or GRD task is defined 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 \(d \in D\), \(\text{conditions}(d)\) are the set of conditions that are applied to \(d\) and \(\text{conditions}(d)\) \(\subseteq\) \(A \cup C\). \(D\) 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 and \(X \Rightarrow Y\) satisfies all constraints in \(M\) with respect to \(D\).

The GRD algorithm is a recursive function with three parameters. The three parameter 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).
CHAPTER 4. METHODOLOGY

GRD (CurrentLHS, AvailableLHS, AvailableRHS)
1: SoFar = ∅
2: for all P in AvailableLHS do
3: NewLHS = CurrentLHS ∪ 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 ⇒ Q, <A, C, D, M ∧ X ⇒ Y ∈ currentSolution U {NewLHS ⇒ Q}>) then
12: add NewLHS ⇒ Q to currentSolution
13: remove from currentSolution any rule W ⇒ Z: ¬ insolution (W ⇒ Z, <A, C, D, M ∧ X ⇒ Y ∈ currentSolution U {NewLHS ⇒ 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 ∪ (P)
20: end for

Figure 4.1: The GRD algorithm, [21]

A global variable, currentSolution consists of the set of rules that are consist 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 AvailableRHS is tested with NewLHS to determine whether the rule NewLHS ⇒ c can be in the solution. At this stage a recursive call is made to the GRD function 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 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 GRD algorithm is presented in 4.2.1. The pruning sections of the algorithm have been omitted. To see the complete algorithm with pruning, refer to Appendix D. Positive association rules are generated with this algorithm.
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 ⇒ c are explored and for each condition in AvailNegRHS, the rules NewLHS ⇒ ¬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 can be seen in Figure 4.2.2.

The additional procedure CreateRules can also be viewed in Figure 4.2.2. 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): The new available antecedent set, which is SoFar.
- NewAvailableRHS (ref): The new available positive consequent set.
- NewAvailNegRHS (ref): The new available negative consequent set.
- SoFar (val): 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. \( A \land B \Rightarrow C \)
2. \( A \land B \Rightarrow \neg C \)
3. \( A \land \neg B \Rightarrow C \)
GRDI (CurrentLHS, AvailableLHS, AvailableRHS, AvailNegLHS, AvailNegRHS)
1: SoFar = ∅
2: for all P in AvailableLHS do
3:  CreateRules (P, AvailableRHS, AvailNegRHS, AvailableLHS,
5:  if NewAvailableLHS != 0 and NewAvailableRHS != 0 then
6:    GRD (NewLHS, NewAvailableLHS, NewAvailableRHS,
7:      NewAvailNegLHS, NewAvailNegRHS)
8:  end if
9:  SoFar = SoFar ∪ (P)
10: end for
11: for P in AvailNegLHS do
12:  CreateRules (P, AvailableRHS, AvailNegRHS, AvailNegLHS,
14:  if NewAvailNegLHS != 0 and NewAvailableRHS != 0 then
15:    GRD (NewLHS, NewAvailableLHS, NewAvailableRHS,
16:      NewAvailNegLHS, NewAvailNegRHS)
17:  end if
18:  SoFar = SoFar ∪ (P)
19: end for

1: NewLHS = CurrentLHS ∪ 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 ⇒ Q, <A, C, D, M ∧ X ⇒ Y > ∈ currentSolution U {NewLHS ⇒ Q}) then
12:    add NewLHS ⇒ Q to currentSolution
13:    remove from currentSolution any rule W ⇒ Z: ¬ insolution (W ⇒ Z, <A, C, D, M ∧ X ⇒ Y > ∈ currentSolution U {NewLHS ⇒ Q})
14: end if
15: end for
16: for all Q in NewAvailNegRHS do
17:  if insolution (NewLHS ⇒ Q, <A, C, D, M ∧ X ⇒ Y > ∈ currentSolution U {NewLHS ⇒ Q}) then
18:    add NewLHS ⇒ Q to currentSolution
19:    remove from currentSolution any rule W ⇒ Z: ¬ insolution (W ⇒ Z, <A, C, D, M ∧ X ⇒ Y > ∈ currentSolution U {NewLHS ⇒ Q})
20: end if
21: end for

Figure 4.2: The modified GRD algorithm including the CreateRules function
4. A ∧ ¬B ⇒ ¬C
5. ¬A ∧ B ⇒ C
6. ¬A ∧ B ⇒ ¬C
7. ¬A ∧ ¬B ⇒ C
8. ¬A ∧ ¬B ⇒ ¬C

The rules that satisfy the constraints are added to the solution. 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 of the GRD algorithm has a data structure as presented in Figure refGRDRuleStruct. 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. \( \text{coverage} = \frac{\text{no\_of\_cases\_covered}}{\text{total\_cases}} \). The rule also has float values for strength (confidence), support, lift, and leverage. The GRD rule data structure is presented in Figure 4.3.1. 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.

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 \( posAn\_set \) labeled differently from the rule data structure in Figure 4.3.1. 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 the consequent set is negative or positive. The modified RULE_TYPE data structure is presented in Figure 4.3.2.

The new rule data structure has a negative set because rules of the form \( A \land \neg B \Rightarrow C \) can be represented in the data structure by rule \( \to posAn\_set = \{A\} \) and rule \( \to negAn\_set = \{-B\} \). 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 \( A \land B \Rightarrow \neg C \) the rule data structure’s consequent set will appear as rule \( \to Con\_set = \{-C\} \).
Figure 4.3: A sample rule data structure

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 4.4: The modified rule data structure

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 */
4.4 Implementing Negative Rules in the GRD system

4.4.1 Calculating Tidsets and Support

The algorithm for the GRD system is modified to implement the negative rules. The data structure is also modified to be able to store the negative antecedent set of within a rule.

The GRD system calculates tidsets for the itemsets. To calculate negative sets for itemsets the diffsets technique is used. The only place where the diffsets are needed is when CurrentLHS is 0 and an itemset needs to be added to CurrentLHS to form NewLHS. This is done by complementing the current tidset for an itemset and intersecting the complement set with CurrentLHS. Calculations are shown later on in this section.

To calculate the support, strength, lift and leverage of a rule the size of the antecedent tidset is needed. In this situation the diffsets for an itemset need not be calculated which requires extra computation.

The following method is used to calculate the tidsets for negative itemsets and the support of a rule. Similarly, strength, lift and leverage can be calculated.

Assume the Reference tidset = T, Tidset for antecedent = A, Tidset for consequent = C, Tidset for antecedent and consequent = A \(\land\) C.

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

2. For the rule A \(\Rightarrow\) \(\neg\)C, GRDI can compute:
   - Tidset (A)
   - Tidset (\(\neg\)C) = Diffset (T, C) = T - Tidset (C)
   - Tidset (A \(\land\) \(\neg\)C) = Diffset (A, C) = Tidset (A) - Tidset (A \(\Rightarrow\) C)
   - Support (A \(\land\) \(\neg\)C) = Support (A) - Support (A \(\Rightarrow\) C)

3. For the rule \(\neg\)A \(\Rightarrow\) C, GRDI can compute:
   - Tidset (\(\neg\)A) = Diffset (T, A) = T - Tidset (A)
   - Tidset (C)
   - Tidset Tidset (\(\neg\)A \(\land\) C) = Diffset (C, A) = Tidset (C) - Tidset (A \(\Rightarrow\) C)
   - Support (\(\neg\)A \(\land\) C) = Support (C) - Support (A \(\Rightarrow\) C)

4. For the rule \(\neg\)A \(\Rightarrow\) \(\neg\)C, GRDI can compute:
CHAPTER 4. METHODOLOGY

- \( \text{Tidset (} \neg A \text{)} = \text{Diffset (} T, A \text{)} = T - \text{Tidset (} A \text{)} \)
- \( \text{Tidset (} \neg C \text{)} = \text{Diffset (} T, C \text{)} = T - \text{Tidset (} C \text{)} \)
- \( \text{Tidset (} \neg A \land \neg C \text{)} = \text{Diffset (} T, (A \lor C) \text{)} = T - (\text{Tidset (} A \text{)} + (\text{Tidset (} C \text{)} - \text{Tidset (} A \Rightarrow C \text{)})) \)
- \( \text{Support (} \neg A \land \neg C \text{)} = T - (\text{Support (} A \text{)} + (\text{Support (} C \text{)} - \text{Support (} A \Rightarrow C \text{)})) \)

5. For the rule \((A \land \neg B) \Rightarrow C\), where either in the antecedent or consequent an itemset is negated, GRDI can compute:

- \( \text{Tidset (} A \text{)} \)
- \( \text{Tidset (} \neg B \text{)} = \text{Diffset (} T, B \text{)} = T - \text{Tidset (} B \text{)} \)
- \( \text{Tidset (} C \text{)} \)
- \( \text{Tidset (} A \land \neg B \land C \text{)} = \text{Diffset}(A \land C, B) = T - (\text{Tidset}(A \land C) - \text{Tidset}(B)) \)
- \( \text{Support (} A \land \neg B \Rightarrow C \text{)} = (\text{Support (} A \text{)} - (\text{Support (} A \land B \text{)}) \land \text{Support (} C \text{)}) \)

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 diffsets results in low additional computation.

4.4.2 Pruning

Pruning applies to positive sets as it applies to negative sets. The pruning functions within the GRD system are modified to also be able to prune negative sets. Depending on the whether the pruning function applies to the antecedent set or consequent set, the pruning functions are modified accordingly.

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 time when a diffset 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 a positive itemset’s tidset and its superset. The pruning functions within the GRD system are appropriately modified to prune negative conditions in addition to positive conditions.
Chapter 5

Analysis of Experiment Results

5.1 Introduction

The modified GRD program is referred to as GRDI. Experiments are carried out on ten datasets with the modified GRD system. Most of the datasets are the same datasets used for the comparison of the GRD system with Apriori in [21].

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 with the new system.

5.2 Datasets

There are ten datasets used for experiments. Nine out of the ten datasets are taken from the UCI Machine Learning and KDD repositories [4], [6]. The other dataset, ticdata2000 is a market-basket dataset used in research by [25] 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 small datasets to relatively large ones based on the number of records they contained. The records, attributes and values of each dataset are listed in Table 5.1.
Table 5.1: Datasets used for Experiments

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

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 be generated 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. When negative rules are incorporated into the GRD system, the number of combinations of itemsets increase to $2^{2000}$. This results in an exponential increase in the number of itemsets considered and therefore a much greater number of rules explored.

The experiments are carried out on the same computer for both systems. The computer used for the experiments was a Linux server, had a processor speed of 1.20 GHz and 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 the programs is the same. An example input for an experiment to be performed on the shuttle dataset is:

```
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 are used for the second program for the shuttle dataset. The 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 are the same inputs used by [21] 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 example, setting minimum strength to a lower value, 0.5, might result in a lot longer execution time. Through investigation it can be seen that the setting the minimum strength to 0.5 results in a lot more execution time.

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 is 4 (-maximum-LHS-size=4) and both systems assume that only a single condition is available for the right-hand-side. This will simplify the search task.

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

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 does not take a lot more execution time than GRD, e.g. mush. However, some datasets require a lot longer execution times for GRDI than GRD, e.g. ticdata.

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.

5.3.2 Rules Generated

The execution times for several datasets were long for GRDI compared to GRD. Comparing the leverage values for the rules generated by both systems indicates the type of rule generated by both systems. Leverage is the difference between the joint frequency of the antecedent and consequent (Support (A ⇒ C)) and the frequency if they were independent (Support (A) × Support (Y)).
Table 5.2: Comparison of Execution Times

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

Figure 5.1: Execution times of GRD and GRDI
Table 5.3: Comparison Minimum and Maximum Leverage values

<table>
<thead>
<tr>
<th>Data Files</th>
<th>GRD min. lev.</th>
<th>GRD max. lev.</th>
<th>GRDI min. lev.</th>
<th>GRDI max. lev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect4</td>
<td>0.1224</td>
<td>0.1227</td>
<td>0.1688</td>
<td>0.1707</td>
</tr>
<tr>
<td>covtype</td>
<td>0.1083</td>
<td>0.1743</td>
<td>0.2459</td>
<td>0.2474</td>
</tr>
<tr>
<td>ipums.la.99</td>
<td>0.2080</td>
<td>0.2484</td>
<td>0.2499</td>
<td>0.2500</td>
</tr>
<tr>
<td>letter-recognition</td>
<td>0.0455</td>
<td>0.1459</td>
<td>0.1020</td>
<td>0.1499</td>
</tr>
<tr>
<td>mush</td>
<td>0.1558</td>
<td>0.2109</td>
<td>0.1994</td>
<td>0.4930</td>
</tr>
<tr>
<td>pendigits</td>
<td>0.0615</td>
<td>0.1757</td>
<td>0.1050</td>
<td>0.1832</td>
</tr>
<tr>
<td>shuttle</td>
<td>0.0409</td>
<td>0.1599</td>
<td>0.0911</td>
<td>0.2040</td>
</tr>
<tr>
<td>soybean-large</td>
<td>0.2137</td>
<td>0.2359</td>
<td>0.2286</td>
<td>0.6182</td>
</tr>
<tr>
<td>splice junction</td>
<td>0.0404</td>
<td>0.1523</td>
<td>0.1244</td>
<td>0.1733</td>
</tr>
<tr>
<td>ticdata2000</td>
<td>0.1899</td>
<td>0.1922</td>
<td>0.2184</td>
<td>0.5341</td>
</tr>
</tbody>
</table>

For example, the positive rules generated by GRD for connect4 had a maximum leverage value of 0.1227. With 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 was 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 Figure 5.2. It can be seen that for several datasets the minimum leverage value of GRDI is greater than the maximum leverage value of GRD. The information contained here is that all the rules generated for those datasets are negative rules.

### 5.4 Conclusion

The modified GRD system that is developed is referred to as GRDI (GRD new Implementation). To develop negative rules the number of itemsets considered increases exponentially which results in much large 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 [21]. 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 [25].

To compare GRD and GRDI the execution times for both systems are noted and observed. It can be seen that for some datasets GRDI requires a lot more computation 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). When GRDI took a lot longer to execute a dataset it can be seen that a majority of rules generated are negative or all the rules in the solution are negative.

Even though GRDI requires a lot longer to execute on some datasets, GRDI is searching through a space many more rules than GRD does. This can result in little additional computation, and appears as though it is a lot more if most of the rules to be generated are negative.
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. Mining negative rules has already been explored using association rule discovery [12], [22].

The GRD approach was chosen to implement negative rules. 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 another alternative constraint. Such rules will not be generated using association rule discovery. The user is allowed to specify alternative constraints to generate rules in GRD.

The second reason for choosing GRD is because the GRD approach allows users to generate a specific number of rules that maximize a particular search measure. Constraints can still be applied to limit the search. 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 is saved for the user of the system.

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 this set using diffsets. For example, if A and \( \neg C \) are the antecedent and consequent of the rule then the tidset(\( \neg C \)) is \( T \) - tidset(C) where \( T \) is the superset. Since GRD calculates tidsets for all itemsets, using diffsets results in a lot of execution time being saved.

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 along with the positive consequent set.

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

A comparison of the modified system is made with GRD. 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 were bound to be longer for GRDI. The leverage values of the rules generated show that the rules with the new system had much higher leverage for those datasets that took longer to execute. This indicates that most rules generated for these datasets are negative rules.

There are some limitations that were observed during the implementation of GRDI. The are a few possibilities of future work with mining negative rules. Both these aspects are described in the following sections.

### 6.2 Limitations of the GRDI system

There are some limitations of the GRDI system when generating negative rules.

1. The conditions are limited to a single itemset for the consequent of a rule. Even though all possible conditions are explored as the consequent of a rule, including multiple conditions in the consequent of the rule may lead to very interesting rules.

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

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

4. If the constraint values are low, then there is a possibility of both rules \( \text{Tea} \Rightarrow \text{Coffee} \) and \( \text{Tea} \Rightarrow \neg \text{Coffee} \) appearing in the solution of the system. The information contained in the rules is not of much use in this situation. However, GRD allows the user to specify a particular number of rules to be generated. These rules are the rules with the highest value of the search measure, therefore, it is likely that one of the rules mentioned above is going to be part of the solution and the other is not.
6.3 Future Work

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

1. Quantities for each item of the antecedent or consequent can be considered. For example, if a rule such as Tea ⇒ ¬Coffee exists then including specific quantities will generate a rule which is more useful. A resulting rule generated could appear as: 10 packets of Tea ⇒ ¬Coffee.

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. If all rules of a particular category are grouped into a single variable then more rules can be developed. The new rules developed are more interesting.
References


REFERENCES


Appendix A

Glossary of Terms
Appendix B

Example Output: Output from connect4.data
Appendix C

An example of a modified Pruning Function