PRM-BASED MULTI-RELATIONAL ASSOCIATION RULE MINING

by

Qidan Cheng

B.Sc., Simon Fraser University, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

© Qidan Cheng 2007
SIMON FRASER UNIVERSITY
Summer 2007

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.

APPROVAL

Name: Qidan Cheng

Degree: Master of Science

Title of thesis: PRM-based Multi-relational Association Rule Mining

Examining Committee: Dr. QianPing Gu
Chair

Dr. Martin Ester, Senior Supervisor

Dr. Oliver Schulte, Supervisor

Dr.Jian Pei, SFU Examiner

Date Approved: June 25, 2007

SFU SIMON FRASER UNIVERSITY
LIBRARY

Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <www.lib.sfu.ca> at: <http://ir.lib.sfu.ca/handle/1892/112>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada

Revised: Summer 2007

Abstract

One of the primary goals of data mining is to extract patterns from a large volume of data. Rules characterize patterns in a humanly comprehensible manner. In particular, association rules have become one of the central research topics in data mining. Association rule mining has previously been restricted to data in a single table. As most data-intensive applications employ a relational database for storage and retrieval, this thesis aims at mining association rules from a standard relational database.

Fundamentally different from previous works, the proposed algorithm is driven by the Probabilistic Relational Model (PRM) of a relational database rather than the minimum support restriction. Based on the conditional independence relationships inferred from the PRM structure, our algorithm removes a set of antecedent attributes that lead to the generation of redundant rules to improve the learning efficiency and effectiveness.

Keywords: multi-relational association rule; Probabilistic Relational Model; conditional independence

Acknowledgments

I would like to express my sincerely thanks to my senior supervisor, Professor Martin Ester, for his excellent supervision. His support, guidance and encouragement of the work made this thesis a reality. My gratitude also goes to Professor Oliver Schulte, who has gone through every word of my thesis and provided valuable feedback that has served to improve the quality of this thesis.

I am truly grateful to those who have served in my supervisory examining committees for their time and effort in reading my thesis and attending to my defence seminar.

Without the support from my family, I would not have been able to complete this thesis. My parents and my husband had taken care of all the housework when I was writing this thesis after work. My little boy, Roy Lee, tried his best to avoid interrupt me when I was working on my thesis. My dear family, I feel I am indebted to every one of you. I am so happy I can spend more time with you now.

Finally, I would like to express my thanks to all of my friends in SFU. They encouraged me and helped me when I met difficulties, and made my stay in this school memorable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The primary goal of data mining is to extract knowledge from a large volume of data. If-then rules are known to be one of the most expressive and humanly comprehensible representation of knowledge. In particular, association rules have become one of the central research topics in data mining, which has many important real life applications, such as market basket analysis, customer profiling and Genomic Data analysis.

Market basket analysis provides answers to the following questions: "if a customer purchases product A, how likely is he to purchase product B?" and "What product will a customer buy if he buys product A?". An association rule has the format of $A \rightarrow B$. The probability of co-occurrence of both A and B is called the *support* and the probability of the occurrence of B given the occurrence of A is called the *confidence*. Although association rule mining algorithms have been extensively studied for decades, most existing approaches focus on mining rules from data residing in a single table. Today most data-intensive applications employ relational databases, typically consisting of multiple tables, for data storage and retrieval. It would be advantageous to automatically learn association rules directly from multi-relational databases. This thesis focuses on finding association rules from multi-relational databases in a cost effective manner.

1.1 Motivation

Association rule mining is a process of finding relationships or associations among specific values of categorical variables in the data set. Most of the current association rule mining algorithms assume that all data reside, or can be made to reside, in a single table. However,

many industrial database applications make use of relational databases to store, manipulate and retrieve structured data. Multi-relational databases consist of a collection of tables. Records in each table only represent part of the information about individuals. Individuals need to be reconstructed by joining foreign key relations between tables. The single table assumption prevents the use of current mining techniques in some important domains, or requires significant effort to pre-process data. Rather than mining association rules from individuals that can be thought of as rows of attribute-value data in a single table, we need to develop algorithms for mining association rules directly from multi-relational databases. By mining rules directly from relational databases, we can make use of database indexing and query optimization techniques implemented in relational database management systems (RDBMS) to improve mining performance.

The major challenge in association rule mining is to improve efficiency. In transactional databases, the number of possible rules grows exponentially with the number of items. In relational databases, if the database contains n descriptive attributes and on average each attribute has m values, then the number of candidate rules grows exponentially with $m \cdot n$ as each pair of (attribute, value) corresponds to one item in a transactional database. In order to improve the performance, different algorithmic approaches have attempted to keep the search space manageable. Apriori-like algorithms make use of the anti-monotonic property of support. Another common approach to reduce search space is to define a set of constraints on the patterns, referred to as *declarative bias*. The Inductive Logic Programming (ILP) approach uses WARMODE declarations [6] as its declarative bias. The multi-relational data mining approach [15] uses selection graph as its declarative bias.

An associated problem in association rule mining in general is the overwhelmingly large number of rules typically generated. For example, Warmr [6] generated almost 100 rules for a tiny Prolog database (shown in Table 1.1) when both minimum support and minimum confidence are set to 0.1. If the data set size becomes larger, typical for an industrial relational database, more than 100,000 rules are selected. As a common observation, many of these rules are redundant. A rule is considered redundant if its information can be obtained from another more general rule. The general approach is to apply an additional redundancy elimination phase, as proposed in [19] [26][18]. It would be advantageous to push redundancy elimination into the rule generation step.

Relational databases usually come with larger data set and richer relational structure. Is it possible to find a novel algorithm to speed up the rule mining process while realizing the

Table 1.1: Prolog database with customer information

customer(allen).	parent(allen,bill).	buys(allen,wine).
customer(bill).	parent(allen,carol).	buys(bill,cola).
customer(carol).	parent(bill,zoe).	buys(bill,pizza).
customer(diana).	parent(carol,diana).	buys(diana,pizza).

goal of effectiveness in finding association rules? This thesis attempt to propose a solution in answering this question.

1.2 Our Approach

Our approach is inspired by the probabilistic relational model (PRM) [9]. PRMs extends Bayesian networks with the concepts of objects, their properties and relations between them. A PRM consists of two components: the qualitative dependency structure and the quantified parameters associated with it. The dependency structure encodes the dependency relationships among the attributes in the same table or in different tables of a database. As stated in the previous section, the most important consideration in rule mining is the efficiency of the algorithm and usefulness of the rules generated. The algorithm proposed in this thesis utilizes the PRM dependency structure learned from the database to prune out redundant rules in the candidate generation step. It can speed up the rule discovery process while effectively reducing redundancy.

In addition, our algorithm does not rely on the anti-monotonic property of support to reduce the search space. Apriori-based algorithms need a minimum support threshold, and find rules with supports exceeding the threshold. Minimum support threshold can help avoid evaluating too many candidates but may miss those rules whose supports are below the threshold. For example, rules about golden credit card user or unsuccessful loans in the financial database usually have very low support. If the minimum support threshold is large, these rules may not be discovered by Apriori-based algorithms. However these rules may be of great interest to the users. Our algorithm differentiates from the Apriori-based algorithms by using the conditional independence relationships to prune candidates other than minimum support threshold, thus our algorithm can efficiently find high confidence rules with very low minimum support.

1.3 Contributions

Main contributions in this thesis include:

- We investigate the current approaches for relational rule mining and analyze their limitations regarding the performance and the usefulness of the resulting rules.
- We explore the semantic information of PRM structure component and propose an efficient attribute-oriented candidate generation method based on the conditional independency relationships encoded in PRM.
- By pruning the attribute sets leading to the generation of redundant rules at the candidate generation step, the proposed PRM-based association rule mining algorithm drastically speeds up the mining process as well as effectively eliminates the redundant rules at an earlier stage.
- We show experimentally that our algorithm is efficient for generating non-redundant rules and the rule set is significantly smaller than the non-pruning algorithm. We also show that our algorithm efficiently finds all the high-confidence rules without minimum support limitation.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows:

- Chapter 2 explains the concept of association rules, Apriori algorithm, and current approaches of multi-relational association rule mining.
- Chapter 3 introduces the background knowledge of the probabilistic models, including *conditional independence*, *Baysian networks* and *Probabilistic Relational Model.*
- Chapter 4 gives a detailed description and explanation of our PRM-based relational data mining algorithm.
- Chapter 5 describes our experiment design, methodology and the experimental evaluation on a real world financial database.
- Chapter 6 concludes with a brief summary of this thesis and directions for future works.

Chapter 2

Related Works

Since its introduction in 1993 [22], the task of association rule mining has received a lot of attention in the literature. A broad variety of algorithms have been proposed to deal with computation-intensive association rule mining. Based on the data source these algorithms apply to, rule mining algorithms can be classified into different categories: mining association rules from relational database, transactional database, spatial database, temporal and time-series database and the world-wide web [27]. This thesis focuses on mining association rule from relational databases. As association rule mining originated in *market basket analysis* which aims at finding patterns from transactional databases, we start with a brief introduction of association rule mining algorithms for transactional databases and then move to existing association rule mining paradigms for relational databases.

2.1 Association Rule Mining Algorithms

Association rules find interesting associations or correlated relations from large amounts of data. The resulting rules can help users make intelligent business decisions. A typical application of association rule mining is *market basket analysis*. Given a large amount of transaction records of customer purchases, the association rule mining process finds associations between the different items that customers put into their shopping baskets. Sample results from association rule mining process include: "60% of customers who purchased milk also bought bread", or "40% of the customers who purchased apples also bought juice". This kind of information could be used by retailers to design their store layout or to decide which type of customers are likely to respond to certain offers.

A transaction record basically consists of a transaction id and a set of items. An example of a transactional database is shown in Table 2.1.

Table 2.1: Transactional Database Example

Transaction id	items
1	printer, laptop, scanner
2	desktop, printer, ink
3	printer, monitor
4	printer, scanner

The definition of an association rule, according to Agrawal [25], is as follows:

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of all items. Let D be a set of all transactions, where every transaction T is a set of items $T \subseteq I$. A set $X \subseteq I$ with $k = |X|$ is called a $k-itemset$. An association rule is of the form $X \rightarrow Y$ where $X \subset I$, $Y \subset I$, and $X \bigcap Y = \emptyset$. The rule $\{printer\} \rightarrow \{scanner\}$ is an example of association rule from the data in Table 2.1.

The support of itemset T is the probability of that set of items appearing in a transaction. The support of the association rule $X \rightarrow Y$, is the probability $\mathrm{P}(X \bigcup Y)$. The support is calculated by dividing the number of transactions containing all the items in both X and Y by the total number of transactions. In Table 2.1, the support of the rule $\{printer\} \rightarrow \{scanner\}$ is 1/2. An itemset whose support exceeds a pre-defined support threshold is called a frequent itemset.

The confidence of an association rule is defined as the conditional probability $P(Y \mid X)$. The confidence is calculated by dividing the number of transactions containing all the items in both X and Y by the total number of transactions containing all the items in X. The confidence of the example rule above is 2/3. The rule can be interpreted as "67% of the people buying a printer will also buy a scanner". A rule that satisfies the minimum support and confidence threshold requirement is called a strong rule.

Although the support-confidence framework can weed out a lot of uninteresting rules, it may also introduce many rules that are not interesting to the user or exclude some rules that may be interesting to the user but do not satisfy the minimum support requirement. Some alternative metrics have been proposed to measure the interestingness of a given rule. For example, *lift*, *correlation* and *collective strength* are often used to determine the interestingness of association patterns.

In brief, "*Association rule mining* is to find out association rules that satisfy the pre-defined minimum support and confidence from a given database" [2]. The problem is usually

decomposed into two subproblems:

1. Find all frequent itemsets: finding all the itemsets whose supports exceed the predefined minimum support.

2. Generate strong association rules from the frequent itemsets: finding all the rules that satisfy the minimum support and the minimum confidence requirement.

2.1.1 Apriori Algorithm

The best known algorithm for finding frequent itemsets is the Apriori algorithm. A key observation exploited in Apriori is that all subsets of a frequent itemset are also frequent. This property is called the anti-monotone property. The Apriori algorithm employs a level-wise search. It starts with finding the set of frequent 1-itemsets and denotes this set as L_1. Then L_1 is used to find L_2 and so on, until no more frequent k-itemsets can be found. The pseudocode of the Apriori algorithm [11] is presented in Algorithm 1.

Algorithm 1 Apriori

Input: Database D, minimum support threshold *min_sup*

```
1: L_1 = find_frequent_1 − itemsets(D);
2: for k=2; L_{k−1} ≠ ∅; k++ do
3:    C_k=aprori_gen(L_{k−1},min_sup)
4:    for each transaction t ∈ D do
5:       C_t=subset(C_k,t);
6:       for each candidate c ∈ C_t do
7:          c.count++;
8:       end for
9:       L_k={c ∈ C_k | c.count ≥ min_sup}
10:   end for
11: end for
12: return L = ⋃_k L_k
```

The apriori_gen procedure consists of join and prune actions. In the join step, L_{k-1} is joined with itself to generate all possible candidates. Then any candidate whose subset is not frequent is removed from the candidate sets in the prune step. The following example illustrates the apriori_gen procedure. Let $L_3 = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{1,3,5\},\{2,3,4\}\}$. In the join step, $C_4 = \{\{1,2,3,4\},\{1,3,4,5\}\}$. After the pruning step, only $\{1,2,3,4\}$ is a valid candidate. Because $\{1,4,5\}$ is not in L_3, the anti-monotone property allow us to safely exclude the itemsets $\{1,3,4,5\}$ from the candidate sets without performing a database scan.

After the frequent itemsets in the database have been discovered, association rule generation is straightforward. Based on the definition of confidence, the following procedure is used to find the strong association rules.

- Find all non-empty subsets for each frequent itemset l.
- If the confidence of the rule $s \rightarrow l$ is above the confidence threshold (where s is a subset of l), output the rule $s \rightarrow l$.

By using the anti-monotone property of support, Apriori avoids the effort of counting the candidate itemsets that are known to be infrequent, thus greatly reduces the computation cost. However, Apriori still has two drawbacks. First, it requires multiple database scans and second, too many itemset candidates need to be tested. In order to improve the efficiency, several new algorithms with modifications or improvements have been developed. A brief overview of Apriori variations is given in the next section.

2.1.2 Variations of Apriori

Based on the Apriori algorithm, two approaches have been developed to improve the efficiency. The first approach is to reduce the number of passes over the database or reduce the size of the database to be scanned in every pass. The second approach explores different kinds of pruning techniques to reduce the number of candidate itemsets. Apriori counts all candidates at the same level in one database scan. The key part is to look up the candidates in one transaction. **AprioriTID** [2] represents each transaction with the candidates it currently contains. After the first pass, the size of this encoding becomes much smaller than the database, thus saving much reading effort. However, the major drawback of AprioriTID is that it needs more memory. Experimental results show that Apriori outperforms AprioriTID in earlier passes while AprioriTID has better performance at later passes. In light of this observation, **AprioriHybrid** [2] was developed to take advantage of both methods. It uses Apriori in the earlier passes and then switches to AprioriTID in the later passes. **Partitioning** [24] reduces the database scans to two passes. It uses the divide-and-conquer approach by dividing the entire database into several partitions such that all the data and candidate itemsets can fit into memory. In the first phase, local minimum support ($minSup$) is calculated by multiplying global $minSup$ by the number of transactions in that partition. For each partition, it uses the local $minSup$ to conduct a frequent itemset search as usual.

The underlying observation for the Partitioning algorithm is that an itemset can not be frequent if it does not appear in any of the local frequent items. After the local frequent itemset search is completed, comes the second phase, in which the union of frequent itemset from all partitions are tested by performing the second database scan. Dynamic Itemset Counting (**DIC**) algorithm [3] requires fewer database scans than Apriori. DIC partitions the database into several intervals. While scanning the first interval, the 1-itemsets are generated and counted. At the end of the first interval, the potential frequent 2-itemsets are also generated, and so on. In general, at the end of the *kth* interval, the potential frequent $(k+1)$-itemsets are generated. By separating counting and candidate generation step, the candidates are generated at an earlier stage, so that less database scans are needed.

2.1.3 Mining Frequent Itemsets without Candidate Generation

Han et al. [12] proposed a frequent pattern-growth (**FP-growth**) method to discover frequent itemsets without candidate generation. The FP-growth method first derives a so-called FP tree, which is a highly-condensed representation of the transaction database. The FP tree is constructed by converting each frequent item into a node and each record into a path in the tree. The second step is to derive the support values of all frequent itemsets from the FP tree. By avoiding the candidate generation process and requiring fewer passes over the database, FP-growth is an order of magnitude faster than Apriori algorithm. Three features contribute to the efficiency of the FP-growth method. First, only frequent items are included in the FP tree, and irrelevant information is discarded. As a result of such improvement, no computational resources will be wasted. Second, only two database scans are required. Frequent patterns are generated by traversing the FP tree, constructing the conditional FP tree which contains patterns with specified suffix patterns. Third, the divide-and-conquer approach employed by FP-growth considerably reduces the size of the subsequent conditional FP-tree, and longer frequent patterns are generated by adding a suffix to the shorter frequent patterns.

2.2 Propositionalization Approach of Relational Association Rule Mining

Although association rule mining has been studied extensively since its introduction, most algorithms assume that the data is stored in a single table. Very few papers address the problem of mining association rules when data is stored in multiple tables. A naive approach is to transform a relational representation of a learning problem into a propositional (attribute-value) representation, then apply the traditional association rule mining algorithm on the flattened representation. This kind of representation transformation is known as *propositionalization*. A simple propositionalization of the relational database is obtained by a complete join over all the tables in the database. However, the naive propositionalization approach may not only cause loss of information, but may also fail to produce correct rules, or produce rules whose support and confidence do not correctly reflect the knowledge embedded in the data.

The following example shows that the propositionalization approach sometimes fails to produce the correct association rule. Suppose we have a simple database consisting of three tables: Student table with attributes $\{studentID(SID), intelligence, ranking\}$, Registration table with attributes $\{registrationID(RID), studentID(SID), courseID(CID), grade\}$ and Course table with attributes $\{courseID(CID), courseName(CN), courseRate\}$. The Entity-Relationship (ER) diagram is illustrated in Figure 2.1 and the sample tables are illustrated in Table 2.2, Table 2.3 and Table 2.4. Joined table is obtained by joining the three tables above and is presented in Table 2.5 below.

Figure 2.1: Entity-Relationship Diagram of Registration Database

The rule, "$intelligence = High \rightarrow ranking = High$", involves the attributes "*intelligence*" and "*ranking*" which belongs to the same entity table Student. If we calculate the support and the confidence based on the Joined table, then the support of this rule is 60% and the confidence is 75%. Another way is to calculate these measures based on the Student table only, in which case the support of the rule is 33% and the confidence is 50%. A question arises as to which result is the correct one. As this rule only involves the attributes in

Table 2.2: Student Table

SID	intelligence	ranking
1	High	High
2	High	Middle
3	Low	Low

Table 2.3: Course Table

CID	CN	rate
1	cmpt275	High
2	cmpt307	Middle
3	cmpt354	High

Table 2.4: Registration Table

RID	SID	CID	grade
1	1	1	High
2	1	2	High
3	1	3	High
4	2	1	High
5	3	1	Middle

Table 2.5: Joined Table

RID	SID	intelligence	ranking	CID	CN	rate	grade
1	1	High	High	1	cmpt275	High	High
2	1	High	High	2	cmpt307	Middle	High
3	1	High	High	3	cmpt354	High	High
4	2	High	Middle	1	cmpt275	High	High
5	3	Low	Low	1	cmpt275	High	Middle

the single entity table Student, we think the confidence and support value should only be calculated from Student table. However, if we use Apriori to mine association rules from the propositionalized table, then we would obtain a misleading result of the support and the confidence measures.

This example illustrates that applying Apriori algorithm directly on the propositionalized database may produce misleading results. In the next section, we will introduce and analyze two other approaches that address the relational association rule mining problem.

2.3 Current Approaches of Relational Association Rule Mining

Two types of database are commonly used for storing and representing structured data, namely Prolog database and relational database. Each tuple in the relational database can be viewed as a typed logical formula in the conjunctive normal form, which can be represented by a set of facts (or predicates) in the Prolog database. Based on what kind of structured data representation format they apply to, two frameworks have been developed towards the discovery of association rules from structured data: Inductive Logic Programming (ILP) framework and Multi-Relational Data Mining framework (MRDM). The ILP approach can be seen as learning from a set of predicates whereas the relational data mining approach can be seen as learning from a set of tables.

Essentially, the problem of association rule mining can be modelled as a search problem over all the possible patterns. Usually, a rule discovery algorithm needs to define the following components: *pattern language*, *score function*, *search strategy* and *declarative bias*. In order to mine interesting patterns from large databases efficiently, a way to express patterns, which is called *pattern language*, needs to be defined first. During the search process, a *score function* for measuring the interestingness of a given pattern is also required. Given the pattern language and the score function, most algorithms do not conduct an exhaustive search. Instead, some prior information is used to limit the search space. Such constraint on the search space is commonly referred to as *declarative bias*. Regarding the search mode, both ILP and MRDM employ a top-down *search strategy*: starting with the most general pattern and progressively considering more specific patterns. In this section, we study how ILP and MRDM frameworks approach the *pattern language choice*, *score function* and *declarative bias* aspects differently, and explore the limitations for each framework.

2.3.1 ILP Approach: Warmr

Inductive Logic Programming (ILP) is a research area formed at the intersection of Machine Learning and Logic Programming. ILP systems develop predicate descriptions from examples and background knowledge. Warmr [6] is an inductive logic programming algorithm designed to discover association rules over a limited set of conjunctive queries on relational databases. Warmr uses Prolog queries as its pattern language. Moreover, it discovers those queries that are "frequent" in a given Prolog (relational) database. Before we introduce the

Warmr algorithm, the notion of *query extension*, *frequent query* and *query subsumption* [6] need to be defined.

Definition 2.1 (Atomic formula) *An atomic formula over R is an expression of the form $R(X)$, where R is a relation name and X is a k-tuple of variables and constants, with k the arity of R.*

Definition 2.2 (Query Extension) *is an existentially quantified implication in the form of*
$l_1, ..., l_m \rightarrow l_{m+1}, ..., l_n$ with $1 \leq m < n$. This is a shortened notion of the following format: $l_1, ..., l_m \rightarrow l_1, ..., l_m, l_{m+1}, ..., l_n$ where each l_i is an atomic formula.

A query extension actually consists of two queries. The former corresponds to the premise of a logic statement, and the latter is the consequence. In the unshortened form, the consequence part is longer than the premise. That is where the "extension" comes from. An example query extension taken from [6] is as follows:

$$? - customer(X), parent(X, Y) \rightarrow buys(Y, cola)$$

This should be interpreted as "if a customer has a child, then that customer also has a child that buys cola". The complete form of this query extension is:

$$? - customer(X), parent(X, Y) \rightarrow customer(X), parent(X, Y), buys(Y, cola)$$

Definition 2.3 (Frequent Query) *The support of a query Q over an instance I of the database is the number of tuples in the answer of query Q. A query Q is frequent if the support is above a pre-defined threshold.*

Definition 2.4 (Query Subsumption) *If the tuples returned by query $Q\prime$, for every possible instance of a database, is always a subset of the tuples returned by query Q, then we say Q subsumes $Q\prime$.*

Pattern Language

Warmr uses Prolog as the representation of data and patterns. Specifically, Warmr uses *Query Extension* as its pattern language. Please refer to above for the definition and an example of *Query Extension.*

Score Function

Like Apriori, Warmr uses support and confidence as its evaluation measure. A sample Prolog database is given in Table 2.6 [6].

Table 2.6: Customer Database

customer(allen).	parent(allen,bill).	buys(allen,wine).
customer(bill).	parent(allen,carol).	buys(bill,cola).
customer(carol).	parent(bill,zoe).	buys(bill,pizza).
customer(diana).	parent(carol,diana).	buys(diana,pizza).

The support of the query, $?-customer(X), parent(X,Y), buys(Y, cola).$ is 25% because only one customer (allen), satisfies the query. We interpret it as "25% of customers are parents of children who buy cola". For the next query, $?-customer(X), parent(X,Y).$, the query result constains three customers, namely "allen", "bill" and "carol". The support of the query is 75%, meaning "75% of customers are parents". The confidence of the following query extension, $?-customer(X), parent(X,Y) \rightarrow buys(Y, cola)$, is 33%. The interpretation of this rule should be "If a customer has a child, then that customer also has a child that buys cola."

Similar to Apriori, Warmr is based on a two-phase architecture. The first phase generates all frequent patterns and the second phase generates all frequent and confident query extensions. This approach requires the minimum support parameter and pushes the minimum support requirement into the search by using the anti-monotone property of the support measurement: if a query is frequent, then all of the generalizations of this query are frequent. Only when the first phase is completed, does Warmr then proceed into the second phase to generate all the confidence rules. This strategy implies that the confidence parameter is totally ignored until frequent queries are screened for rules of high support.

Although this support-confidence framework benefits from the anti-monotone property of support measure by reducing the large search space that has to be explored, it also suffers from the following problem. In a transactional database, we assume that all the items in the dataset have the same nature, so a low support itemset is considered to be uninteresting by the user. However, this assumption does not always hold, especially in a relational database. For example, suppose our analysis target is the *client* table in a financial database. Then we can classify a client based on certain descriptive attributes, such as *district* (in which the client lives) and *gender*. Let us consider two refinements of hypothesis $client(x)$:

$$RuleA : client(x), district(x, district1) \quad \text{support } 20\%$$
$$RuleB : client(x), gender(x, female) \quad \text{support } 45\%$$

Both rules are specializations of $client(x)$ and the support of *rule A* is much lower than the support of $ruleB$. If there are 78 distinct district values but only 2 values for the gender attribute, then *rule A* is more likely to be unexpected by the user than rule B. Therefore, the min-support threshold should depend on the number of attribute values. In the relational database, the number of values for each attribute varies significantly. A uniform min-support threshold seems unreasonable. Some readers may argue that we can solve the problem by using the adaptive min-support parameters. But, in the worst case, this approach would end up using a different min-support parameter for every attribute in the database. Assuming this is achievable, setting proper min-support values is another challenge: a small threshold may lead to the generation of too many candidates, whereas a too large one may miss some rules with low coverage. For example, if only 4% of the clients have a gold credit card, and the min-support threshold is set to be 5%, then all the rules with respect to gold card users would be left out from the resulting rules.

Declarative Language Bias

In order to constrain the query language to a meaningful and useful pattern, a mechanism to reduce the huge search space is needed. As in the Warmr algorithm, Warmode is a declarative language bias formalism adapted for it. Warmode consists of three components as discussed briefly below.

- **Warmode key**: This is an atom which must be included in all queries. In other words, all possible queries must extend the original key atom and this key constraint determines what is counted.

- **Warmode mode**: In the Warmode framework, each variable argument of an atom is bound with mode-labels +, - and ±.

  +: The variable is input, which means it must be bound before the atom is called.

  -: The variable is output; it is only bound by the atom.

  ±: The variable can be an input or an output.

- **Warmode type**: This is a constraint on the sharing of variable names. When an atom is appended to the previous query, the arguments that share a variable name must have the same types, or at least one variable is untyped.

The Warmode key, type and mode restrict what is to be counted, how the query can be extended by adding more atoms and how the variables can be used in each atom extension. For example [6], with key and mode declarations:

$$key : parent(-,-)$$

$$atoms : parent(-p,-c), buys(+c, cola)$$

Then the query

$$?- parent(X,Y), buys(Y, cola).$$

is a valid query, but

$$?- parent(X,Y), buys(X, cola).$$

is not a valid query.

Warmode formalism may generate candidates that do not comply with the general levelwise framework. In a sample beer drinker database, one possible frequent query is of the format:

$$Q_1(x) : -likes(x,y), visits(x,z), serves(z,y)$$

The answer of Q_1 consists of all drinkers that visit at least one bar which serves at least one beer they like. As stated by Bart Goethals [10], if the Warmode declaration for this example is shown below:

$$Key := visits(-,-)$$

$$Atoms := \{likes(\pm, Duvel), likes(\pm, Trappist), serves(\pm, Duvel), serves(\pm, Trappist)\}$$

Then the admissible candidate queries are :

$$Q_1(x_1, x_2) : -visits(x_1, x_2), likes(x_1, Duvel)$$

$$Q_2(x_1, x_2) : -visits(x_1, x_2), likes(x_3, Duvel)$$

Both queries are single extensions of the key. So they are generated in the same iteration. Clearly, Q_2 is more general than Q_1. In this case, the candidate generation step of Warmr generates patterns of different levels in the search space. Furthermore, although Q_2 is syntactically a refinement of the key visits(-,-), it is not a semantical refinement of the key. The addition of the new predicate *likes* will not reduce the number of the answers of the key. This problem is due to the syntactical nature of the declarative language bias used in Warmr and its inability to capture the semantic information of the relational schema.

In summary, Warmr was built on the solid theoretical foundations of ILP. It used the notion of atom sets as a first order logic extension of itemsets in transactional databases. The incorporation of techniques from ILP allows generating more complex and more expressive rules and taking the background knowledge into account. But Warmr also has some shortcomings that are listed below.

1. Although Warmr provides a sound theoretical basis for multi-relational association rule mining, it does not seriously address the efficiency of computation. As a consequence, Warmr does not scale well with regards to the number of relations and the number of attributes in the database. Thus they are usually inefficient for databases with complex schemas.

2. The syntax-based nature of the Warmode declaration makes it hard to be accurately specified by a non-expert user.

3. The Warmr approach primarily is designed for relational data stored as Prolog assertions. Adapting it for data stored in relational databases is complicated because Prolog engines are not designed to support the relational databases. As such it can not incorporate the existing relational database management technologies (e.g. query optimization technique, concurrency control) into the mining process.

2.3.2 Multi-Relational Data Mining Approach: Selection Graph

Knobbe and Blockeel [16] proposed the MRDM approach to support mining on full relational databases. The proposed algorithm finds rules with a previously defined consequence in the form of $S \rightarrow T$. In this approach, the semantic information in the relational database is explored to prune the search space. Also, the multiplicity of associations among tables are

used in the mining process to avoid redundant computation. A so-called *selection graph* is used to express the pattern instead of SQL statements or first order logic expressions. This section contains the detailed description of MRDM approach for rule discovery.

Pattern language and declarative language bias

The possible patterns that can be discovered depend on the declarative language bias specified by each framework. The multi-relational data mining approach expresses patterns in a graphical language of selection graphs. The definition of selection graphs is from [16].

Definition 2.5 (Selection Graph) *A* selection graph *G is a pair (N, E), where N is a set of pairs (t, C) called selection node, t is a table in the data model, and C is a possibly empty set of conditions on attributes in t of type $(t.a\Theta c)$. Θ is one of the following operators, $=, \leq, \geq$. E is a set of triples (p, q, a) called selection edges, where p and q are selection nodes, and a is an association between $q.t$ and $p.t$ in the data model. The selection graph contains at least one node n_0 which corresponds to the target table t_0.*

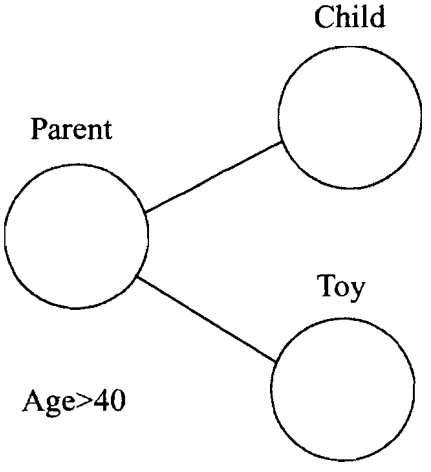


Figure 2.2: This selection graph represents the set of parents older than 40, who have at least one child, and bought one toy.

The selection graph is a representation of selected individuals in the relational database. A node in the selection graph selects a set of tuples in table t that satisfy the set of conditions in C. An edge in the selection graph specifies the association among tables. Selection graphs are more expressive and intuitive than SQL statements or Prolog expressions but they can

be easily translated into SQL or Prolog. The SQL translation of the above selection graph example is as follows:

```
SELECT DISTINCT Parent.name
FROM Parent, Child, Toy
WHERE Parent.name = Child.parentName
AND Parent.name = Toy.buyerName
AND Parent.age > 40
```

Selection graphs also provides a search space for multi-relational data mining algorithm to explore. The rule mining system Safarii, the implementation of MRDM rule discovery algorithm, explores the possible candidate pattern in a top-down fashion. Starting with a very general pattern, it progressively considers more complex and specific selection graphs. The refinement of a given selection graph is achieved by two possible operations as listed below [16].

1. Condition refinement: adding a condition to a selection node without changing the structure of the selection graph. Based on the type of the attributes, three operators $\leq, \geq$ or $=$ can be used.

2. Association refinement: adding an edge and a node to the graph. This operation specializes the originally selected individuals by requiring an association with instances in another table.

From the definition of the selection graph, conditions on each node are of the format: $t.a$ operator c, where t is a table in the data model and c is a constant. Adding a condition on a specific node means that only the objects that satisfy the condition in the corresponding table will be counted. Also the addition of edges are only allowed when it is consistent with the data model. That is, only when an association between two corresponding tables in the data model exists, can the association edge be added as a refinement of the original graph. The refinement specification guarantees that unnecessary and invalid patterns will not be generated and tested. In the beer drinker example, only the refinement Q_1 is valid since it is consistent with the data model. Q_2 is not a valid refinement as it is not the consequence of any of the two refinement operations.

Score function

Along the *score function* dimension, the safarii system offers a collection of rule evaluation measures proposed by Lavrac, Flach and Zupan [17]. These measures are selectable by users. The condensed list of the supported measurement options is as follows [15]:

$$support(S \rightarrow T) = P(ST)$$

$$coverage(S \rightarrow T) = P(S)$$

$$accuracy^1(S \rightarrow T) = P(T|S)$$

$$specificity(S \rightarrow T) = P(\neg S|\neg T)$$

$$sensitivity(S \rightarrow T) = P(S|T)$$

$$novelty^2(S \rightarrow T) = P(ST) - P(S)P(T)$$

Among the various score functions offered by the system, the author recommends that *accuracy* or *novelty* should be used in practice, because "they both express what is close to an expert's judgment of interesting" [17].

Safarii features a Client/Server architecture to separate the search process and the computation-intensive evaluation of candidate patterns in the database. This separation is achieved through a set of so-called multi-relational data mining primitives. The mining process never accesses the database directly, only through the use of a set of pre-defined primitives. The data primitives typically contain the statistical summaries for a range of similar candidate patterns. The implementation of data primitives allow the system to optimize the database access by combining several hypotheses testing into one query. The reduction of the database access in turn improves the performance of the rule discovery process.

Although the use of data models will reduce the search space to some extent, the number of hypotheses generated by MRDM is still too large when coping with industrial-sized databases. Computational complexity of a discovery process is strongly related to the number of hypotheses evaluated during the discovery process. The maximum number of possible

[1] also called confidence

[2] also called leverage

hypotheses can be calculated by multiplying the number of distinct attribute values in all attributes. This number is typically very large even for a moderate size data. Considering a single node in the selection graph, the number of possible condition refinements is $2^{m \cdot n}$ where m is the number of descriptive attributes in the entity table and n is the average number of values per attributes. With regards to the association refinement, typically any two tables from a relational database can be associated with each other, either directly or indirectly, through the foreign key chain. From the refinement operations of the selection graph, we can not see any reduction in the number of hypotheses. For example, in a dataset with 7 attributes, if all attributes have exactly 10 different values, then the MRDM would generate 10 million hypotheses. This is clearly an obstacle for the efficient association rule learning.

Based on our investigations of current approaches for mining interesting patterns from multi-relational databases, we can draw several conclusions:

- The propositionalization approach may fail to produce all interesting rules or may produce rules whose parameters do not reflect the real support and confidence measure of the rule.
- The support threshold favors patterns with a high coverage and discriminates patterns with lower coverage. This results in a large amount of high confidence rules with low coverage undiscovered.
- Rules among attributes of the same relation should be analyzed with respect to the set of individuals in the same relation. Rules among attributes of several tables should be analyzed with respect to all the relations involved.
- We need to find a solution to reduce the number of hypotheses in order to avoid the huge computational cost.

Chapter 3

Probabilistic Models

"Probabilistic models are a marriage between probability theory and graph theory. They are playing an increasingly important role in the design and analysis of machine learning algorithms." — Michael Jordan, 1998

Probabilistic graphical Models have been used successfully in many applications for representing statistical patterns in real world data. Among various graphical models developed so far, Bayesian Networks (BN) provide a very intuitive representation of the dependency relationships between different variables with a domain. Bayesian networks are designed to model attribute-based domains, so they are not well suited for modelling data residing in a relational database. Probabilistic Relational Model (PRM) extends the Bayesian Network framework to allow relational structure being fully represented and exploited. The conditional independence relationships encoded in the PRM graph can provide the guidance for searching interesting patterns from relational databases. In this chapter, we will briefly introduce the following topics: conditional independence, Bayesian Networks and Probabilistic Relational Models.

3.1 Conditional Independency

Suppose we have the following three attributes, each with its value domain shown in parentheses: $Intelligence(high, medium, low)$, $Ranking(high, medium, low)$, and $Grade(a, b, c)$. In the following example, we abbreviate each attribute by its first letter, capital letters for the attributes and lower case letters for a particular value of this attribute. We use $P(A)$ to denote a probability distribution over the possible values of attribute A, and $P(a)$ to denote

the probability of the event $A = a$.

A joint distribution specifies the probability of two or more co-occurring events. We can compute the probability of any instantiation of I, R and G given a joint distribution of attribute values. There are 27 entries in a complete joint distribution table for our example above, one for each possible combination of attribute values. A portion of the joint distribution table is shown in Table 3.1. This table only covers the joint probabilities when $G = b$.

Table 3.1: Partial joint distribution example

G	I	R	P(G, I, R)
b	high	high	0.118
b	high	low	0.059
b	high	medium	0.118
b	low	high	0.059
b	low	low	0.059
b	low	medium	0

In most cases, the distribution of one attribute is only influenced directly by certain attributes while indirectly influenced by others. For example, the effect of attribute I on R might be mediated by G. A formal assertion of this observation is called *conditional independence*. By definition, A and B (where A and B are two nodes in the graph corresponding to two attributes in the database) are independent, written as $A \perp B$, if for every value of A and B:

$$P(A, B) = P(A) \cdot P(B)$$

A and C are conditional independent given B, written $A \perp C \mid B$, if for every value of A, B and C:

$$P(A, C \mid B) = P(A \mid B) \cdot P(C \mid B)$$

or alternatively,

$$P(A \mid B) = P(A \mid B, C).$$

For the example above, if we know that the attribute R is only influenced directly by G, we can assert that for every combination of g, i, r, we have:

$$P(R = r|G = g, I = i) = P(R = r|G = g) \qquad .(1)$$

This assertion states that R is independent of I given G.

Conditional independency allows us to represent the joint probability with a much more compact factored form.

$$\begin{aligned} P(R, G, I) &= P(R|G, I) \cdot P(G|I) \cdot P(I) \\ &= P(R|G) \cdot P(G|I) \cdot P(I) \end{aligned}$$

The first equation follows from the chain rule and the second equation follows from Equation (1) above. Given the equation above, we can calculate the joint distribution using three smaller tables $P(R \mid G)$, $P(G \mid I)$ and $P(I)$. So the storage requirement for the factored representation is $9+9+3 = 21$ entries compared with 27 entries in the full joint distribution representation. The improved storage requirement does not seem quite impressive in this case, but if we have n nodes in a graph, the full joint distribution requires $O(2^n)$ space, and the factored form only requires $O(n2^k)$ space where k is the maximum in-degree of a node. If n is large, the amount of storage required can be significantly reduced.

3.2 Bayesian Networks

A Bayesian network is a graphical representation of a joint probability distribution, representing dependency and conditional independence relationships among the nodes in the graph. The underlying assumption is that only a few nodes in the graph affect each other directly. A simple Bayesian Network model for our *student* domain is illustrated in Figure 3.1.

A Bayesian network consists of two components. The first component is a directed acyclic graph where each node corresponds to one attribute and the edge denotes a direct dependence of an attribute on its parent. The structure of a Bayesian network encodes a set of conditional independence assumptions. Each node A_i is conditionally independent of its non-descendants given its parents. The second component consists of a set of local **C**onditional **P**robability **D**istribution (CPD), one for each node in the graph, denoted as $P(A_i|parent(A_i))$. CPD for a given node specifies the distribution over the values of A_i

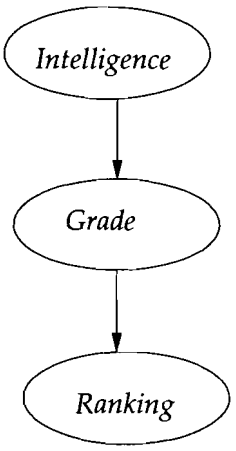


Figure 3.1: A simple Bayesian Network for student domain

given any possible assignment of values of its parents. These two components together allow us to construct the global joint distribution using the chain rule and the conditional independence relationship statements encoded in the structure component.

Table 3.2: CPD for Ranking node in student domain

	G=a	G=b	G=c
R=h	0.67	0.17	0.1
R=m	0.33	0.67	0.2
R=l	0	0.16	0.7

Consider the graph shown in Figure 3.1, in which *Intelligence*, *Grade* and *Ranking* are connected in a chain. There is a missing edge between I and R, meaning that I and R are conditionally independent given G, i.e.:

$$I \perp R \mid G$$

Moreover, we assert that there are no other conditional independencies characterizing this graph. Note that this assertion does not mean that no further conditional independencies can arise in any of the distributions in the family associated with this graph. For example, it is possible that there are some distributions of I and G, where $p(g \mid i) = p(i)$, so $G \perp I$. The key point is that edges that are present in a graph do not necessarily imply dependence, whereas edges that are missing do necessarily imply independence [14].

In essence, the issue comes down to a difference between universally quantified statements and existentially quantified statements, with respect to the family of distributions associated with a given graph. Asserted conditional independency *always* hold for these distributions. Non-asserted conditional independence *sometimes* fail to hold for the distributions associated with a given graph [14]. This statement is the key point of our algorithm design. We use conditional independencies rather than conditional dependencies for our pruning step in order to guarantee the correctness of our algorithm. The conditional independence inference method of the probabilistic models will be further discussed in the next section.

3.3 Bayes-Ball Algorithm: Determining Conditional Independence in Bayesian Network

We can derive a set of basic conditional independence statements from BNs as follows [14]: Defining an ordering I of the nodes in a graph G to be topological, if for every node X_i, the nodes in the parent set X_{π_i} appear before X_i in I. Let X_{v_i} denote the set of nodes that appear earlier than X_i in I, not including the parent nodes X_{π_i}, then $\{X_i \perp X_{v_i} \mid X_{\pi_i}\}$ for $i = 1, ..., n$. Informally, any node is independent of all its non-descendant nodes given its parents.

Apart from the basic conditional independence relations, the BN structure encodes many further conditional independence relations. The notion of *graph separation* provides a mechanism for inferring conditional independence relations implied by a network structure. In

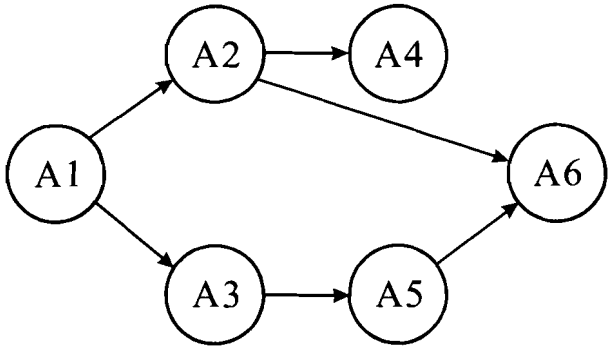


Figure 3.2: The nodes A_2 and A_3 separate A_1 from A_6

Figure 3.2, A_1 is independent of A_6 given A_2 and A_3. This is not one of the basic conditional

independencies for graph 4.2, but it can be inferred by the notion of *graph separation.* We say nodes A_2 and A_3 **d-separate** nodes A_1 and A_6. The 'd' in d-separation stands for *dependence.* If two variables are d-separated relative to a set of variables Z in a directed graph, then they are independent conditionally on Z in all probability distributions. However, naive testing of d-separation may be inefficient because of the large number of chains in acyclic directed graph.

The **Bayes-Ball algorithm** [23] is an efficient algorithm to assess the d-separation in the network. This algorithm can decide whether a conditional independence statement, $X_A \perp X_B \mid X_C$ is true, where X_A, X_B and X_C are any possible disjoint sets of nodes in the network. The complexity of the algorithm is $O(n+m)$ where n is the number of nodes and m is the number of edges, which is linear in the size of the graph. The Bayes-Ball algorithm starts by shading the nodes X_C, placing a ball at node X_A and let the ball bounce around the graph according to a set of rules. If the ball reach X_B, we assert that $X_A \perp X_B \mid X_C$ is not true, otherwise, we assert that $X_A \perp X_B \mid X_C$ is true [14].

Next we need to specify what happens when the ball arrives at node Y from node X destined to node Z (Figure 3.3). We specify these rules based on three canonical graphs.

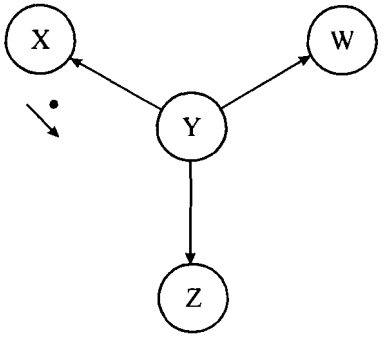
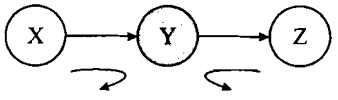


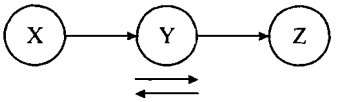
Figure 3.3: What happens when a ball originating from X arrives at Y while destined for Z?

- Case 1: (referring to Figure 3.4) The ball arrives at node Y from X and node Y is shaded. Clearly, if Y is given, X is independent with node Z, so we can not allow the ball to pass through node Y and reach the node Z. In this case, the ball should bounce back. On the other hand, if Y is not given, we can not assert that X is independent

with Z; it is possible that X is dependent with Z through Y, so we allow the ball to pass through Y and reach Z.



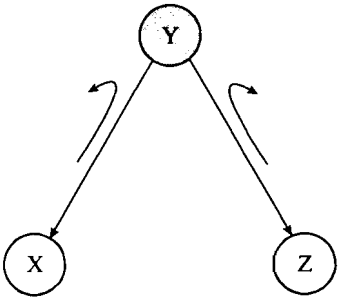
(a) when Y is shaded (given), then ball bounces back.



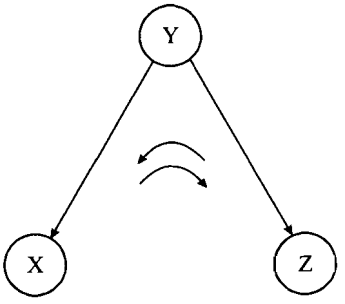
(b) when Y is unshaded, the ball passes through.

Figure 3.4: Case 1

- Case 2: (referring to Figure 3.5) In this canonical graph, similar considerations apply. When Y is known (shaded), then X is independent with Z and the ball must bounce back. When Y is unknown (unshaded), X is possibly dependent on Z through Y, so the ball passes through.



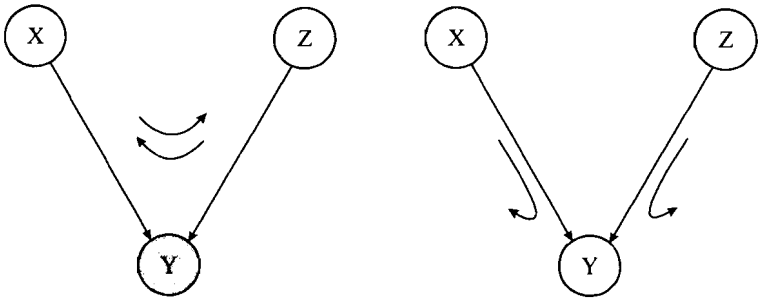
(a) when Y is shaded(given), then ball bounces back..



(b) when Y is unshaded, the ball passes through

Figure 3.5: Case 2

- Case 3: (referring to Figure 3.6) This graph is often referred to as a "v-structure". Here we simply reverse the rules. When node Y is given, we allow the ball to pass through, reflecting that we do not assert that X and Z are conditionally independent given Y. When node Y is unshaded, we require the ball to bounce back.



(a) when Y is shaded (given), then ball passes through.

(b) when Y is unshaded, the ball bounces back

Figure 3.6: Case 3

3.4 Probabilistic Relational Models

Probabilistic Relational Models [9] are an extension of Bayesian networks but incorporates a much richer relational structure. They are to Bayesian Networks as relational logic is to propositional logic. A PRM specifies a relational schema for the domain, a set of probabilistic dependencies between the attributes in the domain, as well as a probability distribution over the database.

3.4.1 Relational Schema

The relational schema of the PRM framework represents the structure of the database. The schema consists of a set of schema classes $\chi = \{X_1, X_2, ..., X_n\}$, where each schema class corresponds to a table in a database. For each class X, there is a set of *descriptive attributes* $A(X)$ which corresponds to columns in a database table. A schema class X_i also contains a set of *links* or *reference slots*, which corresponds to reference attributes in a database table. Figure 3.7 illustrates an example of a relational schema. Classes are shown as boxes which contain ovals representing descriptive attributes and round-cornered rectangles representing reference attributes. In this example, each object of the *Registration* class has a link called $StudentID$ referring to a *Student* object, and a link called $CourseID$ referring to a *Course* object.

A relational *skeleton* σ for a relational schema is defined as a set of instances for each class with the relationships that hold between them (that is, the value for each reference slot is specified). In other words, a relational skeleton is a partial specification of an instance of

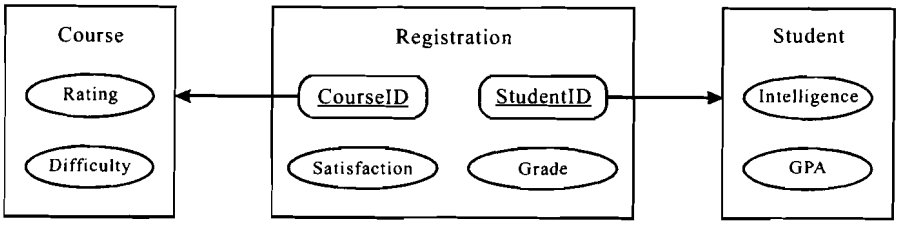


Figure 3.7: Relational schema example

the schema where the values of the attributes are not specified.

An *instantiation* I specifies a set of instances for each class $X \in \chi$ along with the values for each attributes and reference slot. The generic term *object* or *instance* refers to both entities and relationships.

3.4.2 Probabilistic Model for Attributes

A PRM specifies a probability distribution over attributes from all the classes. Similar to a Bayesian network, the underlying assumption is that each attribute is influenced directly by only a few others. So the probabilistic model for each attribute can be expressed by a function of these few influences. In addition, not all attributes from a schema will participate in a probabilistic model. Only the descriptive attributes with quantifiable values participate in the dependency model (for example, the *Rating* attribute in Figure 3.7). The reference attributes are non-probabilistic attributes which provide additional reference information about the instance (for example, the StudentID attribute in Figure 3.7).

There are two major differences between a PRM and a Bayesian network. First, a PRM defines the dependency structure between attributes at the class level. Any instance of a class implicitly inherit the dependency model for each attribute of that class. So, the dependency model applies universally on every object of a specific class. A Bayesian network lacks the concepts of "class" or "object", because it models a pre-specified set of random variables, whose relationships with each other are fixed in advance. Second, a PRM explicitly uses the relational structure of the model, which allows the attribute of an object to depend on attributes of related objects. A PRM consists of two components: the qualitative dependency structure and the quantitative parameters associated with it. The dependency structure specifies a set of parents $Pa(X.A)$ of each attribute $X.A$. The parents

are attributes that directly influence the attribute $X.A$. Two types of former parents are defined. The attribute $X.A$ can depend on another probabilistic attribute B of X; or $X.A$ can also depend on attributes of related objects $X.\tau.B$, where τ is a slot chain and B refers to an attribute in another table. The notion of slot chain is defined as follows [9]: Let $R(X_1, ..., X_k)$ be a relation. We can project R onto its i-th and j-th arguments to obtain a binary relation $\rho(x, y)$, which can be viewed as a slot of X_i. For any x in X_i, we let $x.\rho$ denote all the elements y in X_j such that $\rho(x, y)$ holds. We can concatenate slots to form longer slot chains $\tau = \rho_1, ..., \rho_m$. Intuitively, a slot chain is a function that maps the values of attribute $X.A$ to the values of attribute $Y.B$ residing in another table Y. If the slot chain is a multi-valued function, then an aggregation function will be used.

The formal definition of a PRM from [8] is given as follows.

Definition 3.1 (Probabilistic Relational Model) *A* Probabilistic Relational Model *(PRM)* $\prod$ *for a relational schema R is defined as follows. For each class* $X \in \chi$ *and each descriptive attribute* $A \in A(X)$*, we have:*

- *a set of parents* $Pa(X.A) = \{U_1, ..., U_l\}$ *where each* U_i *has the form* $X.B$ *or* $\gamma(X.\tau.B)$*, where* τ *is a slot chain and* γ *is an aggregate of* $X.\tau.B$*.*
- *a legal conditional probability distribution (CPD),* $P(X.A \mid Pa(X.A))$*.*

A PRM specifies the probability distribution over attributes from all the classes using the same underlying principles as a Bayesian network. We can derive a set of independency assertions of the descriptive attributes from the dependency structure of a PRM as from a Bayesian network. A set of basic conditional independency statements can be obtained from a PRM in the same way as from a Bayesian network: any node is independent of all its non-descendant nodes given its parents. Similarly, further conditional independence relations can also be inferred from the PRM structure using the Bayes-Ball algorithm introduced in Section 3.3.

Figure 3.8 shows the PRM structure of a simple university domain. In this PRM structure example, we can easily derive a set of conditional independence assertions: *Student.Ranking* is independent of *Student.Intelligence* given *Avg(Registration.Grade)*, *Student.Ranking* is independent of *Course.Difficulty* given *Avg(Registration.Grade)*.

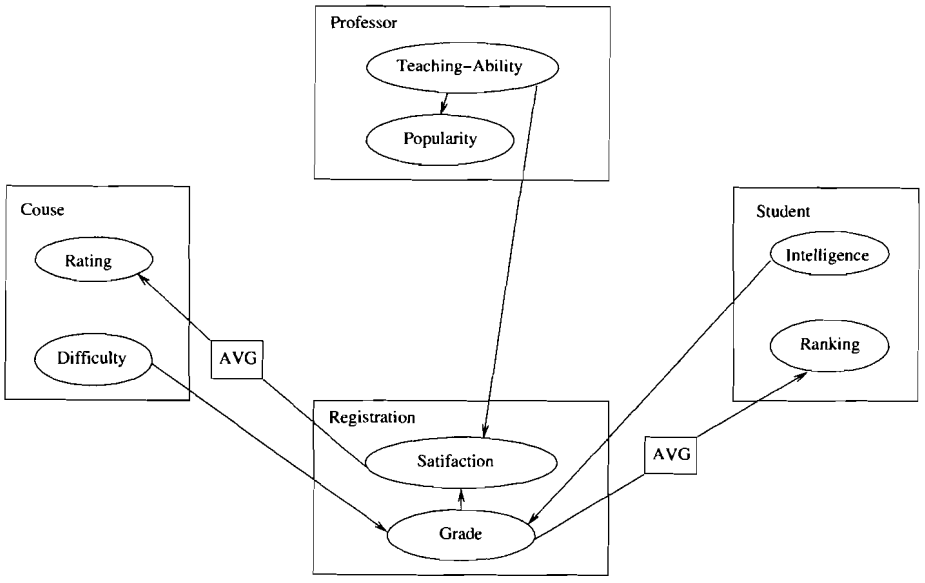


Figure 3.8: The PRM structure of simple university domain

3.4.3 PRM Semantics

PRMs define a distribution over possible instantiations of a database that are consistent with a relational skeleton. As with Bayesian networks, the joint distribution over different values of attributes can be represented by a much more compact factored form. The probability distribution over all instances can be computed as follows.

$$P(I \mid \sigma_r, S, \theta_s) = \prod_{x \in \sigma_r} \prod_{A \in A(x)} P(I_{x.A} \mid I_{Pa(X.A)}, \theta_s)$$
$$= \prod_{\chi_i \in \chi} \prod_{A \in A(\chi_i)} \prod_{x \in \sigma_r(\chi_i)} P(I_{x.A} \mid I_{Pa(X.A)}, \theta_s)$$

$I_{x.A}$ denotes the value of $x.A$ in the instantiation I, χ denotes the set of all classes in the schema and S denotes the probabilistic dependency structure having parameter θ_s. Informally, we can compute the joint distribution by taking the product, over all x.A, of the probability in the CPD of the specific value assigned by the instance to the attribute given the values assigned to its parents [8].

3.4.4 Learning PRMs

After we have reviewed the definition of PRM, its components and the semantics, we will briefly introduce the important task of automatic PRM structure learning from an existing database. Three aspects must be addressed before we start the learning process. First, we need to determine which dependency structures are legal. Second, we should specify how to evaluate the goodness of a given structure. Finally we have to decide the effective search strategy to find a good structure. This is known as the search-and-score algorithm. A structure is legal if the dependency between object attributes is acyclic. This is due to the restriction that PRM is a directed acyclic graph (no cycle is allowed). In order to guarantee that the generated structure is legal, the key point is to maintain a stratified class dependency graph and consider only structures whose dependency structures are consistent with the stratification. Specifically, whenever an edge is added into the graph, we can check whether the new edge will introduce a cycle in the graph. The checking step requires only $O(\mid V \mid + \mid E \mid)$ time where $\mid V \mid$ is the number of vertices and $\mid E \mid$ is the number of edges in the graph. After determining that a structure is legal, the next step is to evaluate the structure based on a score function. PRM learning adapts the Bayesian model selection methods for the framework. Let S denote a structure, I denote an instantiation of a database and σ denote the skeleton of a relational schema. The following is based on the Bayes rule:

$$P(S|I,\sigma) \propto P(I|S,\sigma)P(S|\sigma)$$

The goal is to maximize the posterior probability of a structure given an instantiation I and a skeleton σ. The posterior probability $P(S|I,\sigma)$ can be calculated from $P(S|\sigma)$ (the prior over the structure given the skeleton) and $P(I|S,\sigma)$ (the marginal likelihood). The third issue we need to address is the search strategy. PRM learning uses a greedy local search by considering candidates in the following order [7]:

- Consider only dependencies within the same table.
- Consider dependencies from neighboring tables, via relation chains.
- Consider dependencies from further tables, via relation chains.

Our implementation of PRM structure learning is adapted from the K2 algorithm [4] which is a greedy heuristic algorithm for finding Baysian Network structures. The algorithm

first assumes a node has no parent, and then incrementally adds a parent as its predecessor whose addition most increases the probability of the resulting structure until no improvement is possible or a pre-specified threshold on the maximum number of parents is reached.

Chapter 4

PRM-Based Association Rule Mining

Efficient pattern discovery from relational databases is challenged by the large size of databases. Inductive logical programming (ILP) techniques have had considerable success on a variety of multi-relational rule ming tasks. However, most ILP approaches are not scalable with regard to the number of relations and the number of attributes in the database. Thus they are usually inefficient for databases with complex schemas. The multi-relational data mining (MRDM) approach, on the other hand, improves its performance by using *data mining primitives*, but the number of hypotheses to be tested is quite large due to the lack of a candidate pruning strategy. Our algorithm can efficiently discover non-redundant, high-confidence rules with a pre-determined consequence attribute. Fundamentally different from previous works, the proposed algorithm is driven by the conditional independence relations between attributes rather than the minimum support restriction. This attribute-oriented pruning approach can efficiently remove a set of antecedent attributes that lead to the generation of redundant rules, therefore reduce the number of hypotheses to be evaluated during the rule mining process. In this chapter, we will describe our PRM-based Association Rule Mining algorithm (PARM) in detail.

4.1 Problem Definition

One major task of rule discovery is to extract interesting rules from large volumes of data. Often, interesting rules are buried under a large amount of trivial or redundant rules. In order to avoid this situation, two approaches can be used to reduce the rule size by eliminating trivial or redundant rules. The first approach is to utilize the subjective interests specified by users to eliminate uninteresting rules and effectively find valuable ones. In many applications, users are interested in discovering properties of a specific class of instances. We seize such situations by requiring users to specify an attribute as the consequence and we aim to find all such rules. The second approach is to simplify the resulting rule set by removing redundant rules. Most existing approaches eliminate the redundant rules by reorganizing and post-pruning the resulting rule set, as in [19], [18] and [26], which does not improve the efficiency of the learning process. It would be advantageous to push the redundant pruning phase into the mining process so as to discover the non-redundant rule set in a cost effective manner. In this section, we will start with some definitions, and then explain the single consequence non-redundant rule mining problem in more detail.

Definition 4.1 (Descriptive attribute of a relation) *Given a relation T with primary key K from database D, we define any attributes, denoted by $T.A$, that describe the properties of entities in T with quantifiable values as a descriptive attribute of the relation T.*

Definition 4.2 (Atomic formula) *Given a relation T from D, an atomic formula is a constraint on a descriptive attribute $T.A$. It is in the form $T.A\Theta c$. Θ is one of the following operators, $=, \leq, \geq$ and c is a constant.*

Definition 4.3 (Single consequence association rule) *Given a pre-defined target descriptive attribute $T.A$ of the target table T, a single consequence association rule is of the form $Q_1 \rightarrow Q_2$ where Q_1 is of the form $l_1, ...l_m$, Q_2 is of the form $l_1, ..., l_m, l_{m+1}$, l_i is an atomic formula and l_{m+1} is an atomic formula of $T.A$. In abbreviation, we write this rule as $l_1, ..., l_m \rightarrow l_{m+1}$.*

We say that a single consequence rule r $(X \rightarrow l)$ is more general than rule $r\prime$ $(X\prime \rightarrow l)$ if $X \subset X\prime$, denoted as $r \subset r\prime$. Conversely, $r\prime$ is more specific than r, denoted as $r\prime \supset r$.

Definition 4.4 (Non-redundant single consequence rule set) *Let R be the complete*

set of singel consequence association rules whose confidence is greater than or equal to minconf for a given database D and a pre-defined target attribute A. A set R_N is a non-redundant single consequence rule set w.r.t R if (1)$R_N \subseteq R$; (2)$\forall r\prime \in R - R_N$, $\exists r \in R_N$ such that $r \subset r\prime$ and $conf(r) \geq conf(r\prime)$; and (3) $\forall r \in R_N, !\exists r\prime \in R_N$ such that $r\prime \subset r$ and $conf(r\prime) \geq conf(r)$.

The non-redundant single consequence rule mining problem is formally defined as follows.

Definition 4.5 (Non-redundant single consequence rule mining problem) *Given a relational database D, the PRM dependency structure G of D, the minimal confidence threshold* minconf *and the consequence attribute A, find the set of all non-redundant rules having A as its single consequence attribute such that the confidence of each rule is greater than or equal to a pre-defined* minconf.

Our approach is aimed at finding a special subset of the complete association rule set without losing the completeness power of the latter. Let us imagine how a user makes predictions using a set of association rules. Given a set of association rules, the user selects a matching rule with the highest confidence from the rule set, and then uses the consequence of the rule as the prediction result. Assume that a complete association rule set contains the following two rules: $a \rightarrow c$ (confidence = 0.6) and $a, b \rightarrow c$ (confidence =0.5). Obviously, the user would pick the first rule over the second rule. Therefore, the second rule will never be selected in the above prediction procedure. We call the second rule a redundant rule because there exists a more general rule with a higher confidence. The proposed algorithm removes all the redundant rules from the association rule set and forms a new rule set, which can predict exactly the same results as the original association rule set. We define this subset as the non-redundant association rule set with regards to the complete association rule set.

It is clear that our definition of a non-redundant rule set is different from the general definition of a non-redundant rule set. In [19], a rule is considered redundant if its confidence is *close* to a more general rule. However, we define a rule as being redundant if its confidence is less than or equal to a more general rule. Therefore, the non-redundant rule set defined in [19] is larger than the non-redundant rule set defined by us. As explained above, our definition originates from how users will make predictions based on our rule set. If users make predictions by selecting the matching rules with the highest confidence value, then the specific rules with lower confidence values will never be selected during the prediction

procedure. However, our algorithm can be easily adjusted to find the non-redundant rule set defined in [19] by modifying the rule pruning procedure. In the rule pruning step, we remove a rule if there exists a subrule with a higher confidence. To adapt it to the non-redundant rule definition in [19], we can set a *minimum confidence difference* threshold to measure whether two confidence values are close enough, then remove a more specific rule only if there exists a more general rule with "close confidence".

The proposed approach is based on the PRM dependency structure of a relational database. Since only the descriptive attributes with quantifiable values can participate in the dependency model of the PRM, we mainly focus on finding a set of rules that explore the dependency relationships between the *descriptive attributes* in a relational database. The pattern language to be used is the conjunction of a set of atomic formulae, which represents the selection of instances that satisfy all the descriptive attributes constraints defined. The underlying assumption is that a set of constraints over the descriptive attributes can uniquely identify a set of instances from a database.

4.2 Comparison With the Classification Rules

Our problem setting presents some similarities with the classification rule discovery problem. Both problems aim at a pre-determined class attribute in the database and attempt to find regularities about the specified class. In order to distinguish the single literal consequence association rule discovery task and the classification task, we need to examine the functionalities of data mining first. In general, data mining tasks can be classified into two classes: predictive and descriptive. "Descriptive mining tasks characterize the general properties of the data in the database while predictive mining tasks make inference on the current data in order to predict unseen individuals" [11]. Based on the above definition, association rule mining can be classified as a descriptive task whereas classification falls into the predictive class. A predictive task is interested in global models that give a unique classification of each individual either in the database or yet unobserved. Usually no hard thresholds are involved in a traditional classification rule generation algorithm. A descriptive task, on the other hand, produces a local model that explains and increases understanding of current data. It is based on some hard thresholds, such as the minimum support and the minimum confidence. Our algorithm is a descriptive algorithm by nature. The rule set generated by our algorithm are independent with each other. They specify some knowledge about the

individuals covered by the antecedent of the rule, but not the remainder of the database. The rule set can not be treated as a classifier directly because of the following reasons:

- Not all individuals in the database are covered by at least one rule.
- Some individuals may be covered by multiple rules with different class labels.

Besides the differences mentioned above, there are other dissimilarities between a classification algorithm and an association rule algorithm. A classification algorithm usually generates a small set of rules while association rule sets are often very large. A classification rule mining algorithm normally uses a heuristic criterion and cannot guarantee finding the optimal rule set whereas an association rule mining algorithm generates all the rules satisfying the requirement.

Despite of the above differences, association rules and classification rules are closely related. It is possible to use association rule mining techniques to solve classification rule mining problems, but how to adapt our algorithm to make it suitable for the classification task is an interesting research topic for future.

4.3 Basic Idea

In this section, we will first briefly review the semantics of PRM dependency structure, and then introduce the basic idea of our proposed approach.

4.3.1 PRM Review

Our **P**RM-based **A**ssociation **R**ule **M**iner (PARM) algorithm for learning non-redundant single-consequence association rules is a PRM-based approach. As illustrated in Chapter 3, the dependency structure of a PRM encodes a set of dependency relationships between the descriptive attributes in a relational database with graphical representation. The PRM structure of a simple university database is shown in Figure 4.1.

From the PRM structure, we can derive a set of dependency assertions of the descriptive attributes. The dependency relations can either exist between attributes in the same table (*Registration.Satisfaction* depends on *Registration.Grade*), or between attributes in different tables (*Registration.Satisfaction* depends on *Professor.Teaching-Ability*). Intuitively, the dependency relationships can guide our search for interesting patterns from large databases.

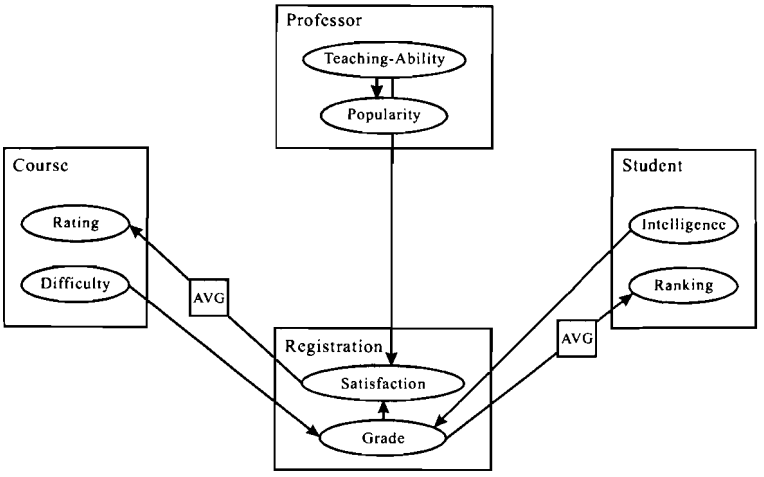


Figure 4.1: The PRM structure of university domain

Recall an important characteristic of the independency assertions: if two attributes are independent, the independency relationship applies for all values of the attributes. Based on this property, we propose an attribute-oriented pruning method to reduce the number of hypotheses. Antecedent attribute sets that lead to generating redundant rules are removed from the complete candidate sets. The attribute-oriented pruning allows us to save significant amount of computational effort for evaluating redundant rules. Other than the improved performance, the size of the discovered rule set is also reduced by means of redundancy elimination.

4.3.2 Derivable Candidate and Generating Candidate

An *attribute candidate* is a set of descriptive attributes that appear in the antecedent part of rules. Throughout this thesis, the term *candidate* will also be used in abbreviation for *attribute candidate*. Suppose a database consists of n descriptive attributes, the total number of possible attribute candidates is 2^{n-1} if the consequence attribute is excluded from the antecedent attribute set. We categorize all the attribute candidates into two classes: *derivable candidate* and *generating candidate*. In this section, starting with a brief overview of *conditional independence*, *conditional probability* and *confidence measurement*, we then introduce the definitions of *derivable candidate* and *generating candidate*.

In the original formulation of the association rule discovery problem, support and confidence are two interestingness measures. Support measures the joint probability of events

associated with a particular rule and confidence measures the conditional probability of events associated with the rule. If we formulate the support and confidence measure in the context of random variables and probability distribution, the interestingness of a rule $B = b \rightarrow A = a$ is usually defined as a function of $\mathrm{p}(A = a)$, $\mathrm{p}(B = b)$ and $\mathrm{p}(A = a \wedge B = b)$:

$$support(B = b \rightarrow A = a) = p(A = a \wedge B = b)$$

$$confidence(B = b \rightarrow A = a) = \frac{p(A = a \wedge B = b)}{p(B = b)} = p(A = a \mid B = b)$$

The structure of a PRM model gives us a set of conditional independence statements of the descriptive attributes in the model, but can we utilize conditional independence to reduce our search space? Suppose we have a single conclusion A, and we try to derive all the rules in the form of $X = x \rightarrow A = a$. If we have the rule

$$B = b \rightarrow A = a$$

$$conf(B = b \rightarrow A = a) = p(A = a \mid B = b) = p$$

and we also know $A \perp C \mid B$, then for every value a, b and c we have that

$$p(A = a \mid B = b) = p(A = a \mid B = b, C = c)$$

so,

$$\begin{aligned} conf(B = b, C = c \rightarrow A = a) &= p(A = a \mid B = b, C = c) \\ &= p(A = a \mid B = b) \\ &= conf(B = b \rightarrow A = a) \end{aligned}$$

This implies that the rule $B = b, C = c \rightarrow A = a$ is redundant in the sense that the confidence of a more general rule $B = b \rightarrow A = a$ is no less than its confidence. In other words, the confidence does not rise by adding knowledge about C. We can safely prune the rule $B = b, C = c \rightarrow A = a$ for every instantiation of A, B and C.

Another frequently used interestingness measure *lift* also presents this property:

$$lift(B = b \rightarrow A = a) = \frac{p(A = a \wedge B = b)}{p(B = b) \cdot p(A = a)} = \frac{p(A = a \mid B = b)}{p(A)}$$

Similarly, if we know $A \perp C \mid B$, we have:

$$\begin{aligned} lift(B = b, C = c \rightarrow A = a) &= \frac{p(A = a \mid B = b, C = c)}{p(A)} \\ &= \frac{p(A = a \mid B = b)}{p(A)} \\ &= lift(B = b \rightarrow A = a) \end{aligned}$$

Since the above equation applies to all values of A, B, C, we can safely remove the antecedent attribute set $\{B, C\}$ if the antecedent attribute set $\{B\}$ already exists. We call set $\{B, C\}$ a *derivable candidate.*

Definition 4.6 (derivable candidate) *Given a set of descriptive attributes S in database D and a consequence attribute A, a derivable candidate C_d is a set of descriptive attributes defined as $C_d \subseteq S - A$ such that $\exists x \in C_d$, $x \perp A \mid T$ where $T \subseteq (C_d - x)$.*

Definition 4.7 (generating candidate) *Given a set of descriptive attributes S in database D and a consequence attribute A, a generating candidate C_d is a set of descriptive attributes defined as $C_d \subseteq S - A$ such that $\exists x \in C_d$, $\forall T \subseteq (C_d - x)$, $x \perp A \mid T$ does not hold.*

Our goal is to remove derivable candidates from the complete candidate sets. The difference between the complete candidate set and the derivable candidate set is called the *generating candidate set.* Detecting derivable candidates involves checking whether $x \perp A \mid T$ where x is an element in set C_d and T is a subset of C_d. As introduced in section 3.3, the Bayes-Ball algorithm is efficient for determining whether a conditional independence statement, $X_A \perp X_B \mid X_C$ is true, where X_A, X_B and X_C are any possible disjoint sets of nodes in the network.

4.4 PRM-Based Association Rule Miner: PARM Algorithm

In this section, we introduce our PRM-Based Association Rule Miner (PARM) for generating the non-redundant single consequence association rule set. The PARM algorithm consists of three phases: the candidate generation phase, the rule generation phase and the rule pruning phase. The candidate generation phase first finds a set of generating candidates

by removing the derivable candidates from the complete candidate sets. An advantage of this step is to avoid generating candidates that lead to rules omitted by the non-redundant rule set. The rule generation phase then proceeds to evaluate the generating candidates selected in the candidate generation phase. More precisely, it calculates the confidences for all the rules that can be derived from the given candidate, and then reports rules whose confidence is greater than or equal to minimum confidence. As the last step, the rule pruning phase removes the redundant rules. Please note that in this section, the term "node" and "attribute" are used interchangeably.

The PARM algorithm enumerates all the subsets of the antecedent attribute set, where each subset is called an antecedent candidate. For each candidate, the PARM algorithm tests whether it is a derivable candidate or a generating candidate. The derivable candidates are discarded in the candidate generation step, and the generating candidates are kept for further evaluation. We want to remove as many derivable candidates as possible during the candidate generation phase. Recall that a derivable candidate must contain a node that is independent of the consequence node conditionally on an existing candidate. This means that the candidates tested earlier should "d-separate" as many other nodes as possible from the consequence node A. Based on the definition of "d-separation", only nodes that are in the path of any node A_n to A can "d-separate" A_n from A. Therefore, candidates consisting of closer nodes from A should be tested before the candidates consisting of farther nodes.

First, we calculate $d(A_i, A)$ for each node A_i (excluding the consequence node A) in the graph, where $d(A_i, A)$ is defined as the length of the shortest path from A_i to A. Then all the nodes are sorted in increasing order of $d(A_i, A)$. An enumeration tree is then constructed. Figure 4.2 illustrates the complete candidate enumeration tree for antecedent attribute set $\{A_1, A_2, A_3, A_4\}$, where A_1, A_2,..., A_4 are sorted in increasing order of $d(A_i, A)$.

PARM is a level-wise algorithm, which starts by considering all the candidates consisting of one attribute, then proceeds by considering all the candidates consisting of two attributes, and so on, until all the candidates have been processed. Any candidate in the enumeration tree satisfies the following conditions.

- The candidate has one more attribute A_i than its parent candidate.
- A_i has a larger distance from A than any attributes of the parent candidate.
- All the subsets of the candidate are in the previous levels.

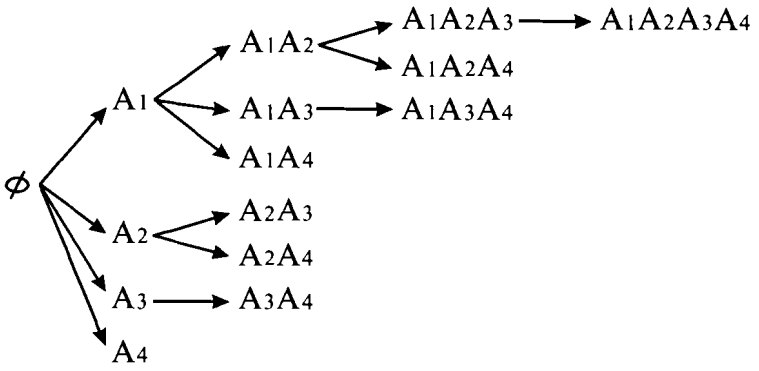


Figure 4.2: A candidate enumeration tree example

This processing order guarantees that the candidates consisting of closer (to the target node) nodes have been processed before farther nodes, which implies that the existing candidate is more likely to "d-separate" the newly added attribute A_i from A. It also ensures that the candidates generated at later iterations cannot be subsets of the candidates generated previously.

As illustrated in Algorithm 2, PARM performs the following operations at each level.

- The Candidate-gen method discovers a set of generating candidates.
- The Rule-gen method calculates the Local Confidence Table (LCT) for each generating candidate, and outputs the rules such that the confidence of each rule is great than or equal to *minconf*.
- The Rule-pruning method removes the rules omitted by the non-redundant rule set. For each rule generated at this level, if a more general rule with the same or higher confidence has already been discovered, then the new rule will be removed from the resulting rule set.

The Candidate-gen method, the Rule-gen method and the Rule-pruning method are explained in detail in Section 4.4.1, Section 4.4.2 and 4.4.3 respectively.

4.4.1 Candidate Generation

The Candidate-gen method works as follows. For each candidate X_n in level n, let N denote the additional node that is added to its parent $Parent(X_n)$. We test whether $N \perp$

Algorithm 2 PARM

Input: A relational database D, PRM G, minimum confidence threshold *minconf*, consequence attribute A.

Output: A set of non-redundant rules R with confidence that is greater than or equal to *minconf*.

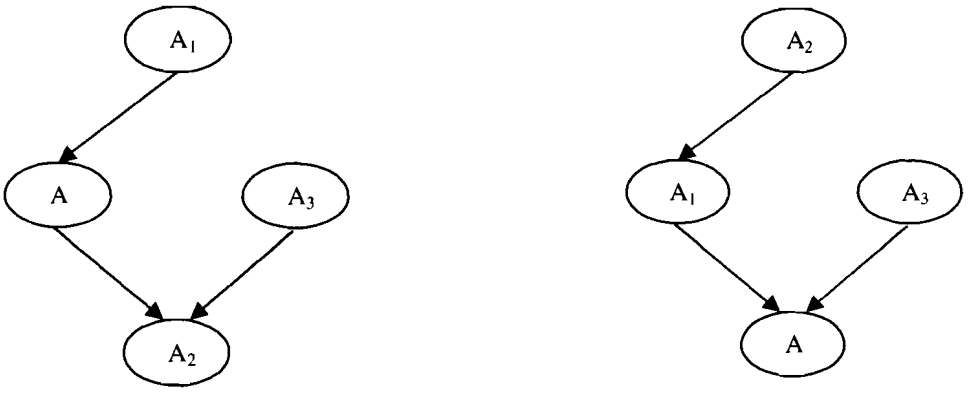
1: Initialize $R \leftarrow \emptyset$
2: Let $d(A_i, A)$ be the length of the shortest path from node A_i to A in G, find the $d(A_i, A)$, for every A_i where $A_i \in Vertices(G)$ and $A_i \neq A$
3: Sort all nodes in S in increasing order of $d(A_i, A)$
4: Construct the candidate enumeration tree
5: Initialize $n = 1$
6: $C_0 \leftarrow \emptyset$
7: **for** each level in the enumeration tree **do**
8: $P \leftarrow$ all the candidates at level n
9: $C_n \leftarrow$ Candidate-gen(P, A)
10: $R_n \leftarrow$ Rule-gen(C_n, A)
11: $N_n \leftarrow$ Rule-pruning(R_n, R)
12: $R \leftarrow R \cup N_n$
13: $n \leftarrow n + 1$
14: **end for**
15: return R

$A \mid Parent(X_n)$. If N is independent from A given $Parent(X_n)$, then the X_n is a derivable candidate. Otherwise, X_n is a generating candidate. The generating candidates will be passed to the Rule-gen method and the derivable candidates are discarded.

The method illustrated above implies an exhaustive testing over all the possible candidates, i.e. all subsets of the antecedent attribute set. Figure 4.3 shows two examples of the following cases, which demonstrates that an exhaustive testing is necessary.

- $A_3 \perp A \mid A_1$ does not guarantee that $A_3 \perp A \mid \{A_1, A_2\}$ is true. Therefore, although $\{A_1, A_3\}$ is a derivable candidate, testing $\{A_1, A_2, A_3\}$ is still necessary.
- $A_2 \perp A \mid A_1$ does not guarantee that $A_3 \perp A \mid \{A_1, A_2\}$ is true. Therefore, although $\{A_1, A_2\}$ is a derivable candidate, testing $\{A_1, A_2, A_3\}$ is necessary as well.

The algorithm of the Candidate-gen method is illustrated in Algorithm 3. The pruning step is achieved in line 4. If the new node is independent of the consequence node A given $Parent(X_n)$, then X_n is not included in the generating candidate set. We can safely prune this candidate because of the following reasons. First, $Parent(X_n)$ has been tested in



(a) $A_3 \perp A \mid A_1$ does not imply $A_3 \perp A \mid \{A_1, A_2\}$ (b) $A_2 \perp A \mid A_1$ does not imply $A_3 \perp A \mid \{A_1, A_2\}$

Figure 4.3: Examples where exhaustive testing is necessary

the last iteration. Either $Parent(X_n)$ or a subset of $Parent(X_n)$ has been included in the generating candidate set. Second, given that $Parent(X_n)$ or a subset of $Parent(X_n)$ already exists, the knowledge of attribute N would not help to predict the behavior of attribute A.

Algorithm 3 Candidate-gen(P, A)

Input: A set of candidates at level n S and the consequence attribute A.
Output: A set of generating candidates for the consequence node A at level n.

```
1: initialize C_n ← ∅
2: for each candidate X_n in S do
3:    N ← X_n − Parent(X_n)
4:    if N is not independent of A given Parent(X_n) then
5:       C_n ← C_n ∪ X_n
6:    end if
7: end for
8: return C_n
```

We will use the university database as a running example to illustrate the work flow of the Candidate-gen method. The PRM dependency graph of our university domain is shown in Figure 4.4.

Suppose that our consequence attribute is *Ranking* and the candidate node set S is $\{Grade, Intelligence, Satisfaction, Difficulty, Rating, Teaching\text{-}Ability, Popularity\}$. S is sorted by the increasing order of $d(N, A)$. At the first level, we check all the candidates consisting of a single attribute in increasing order of $d(A_i, A)$. For example, *Grade* will be

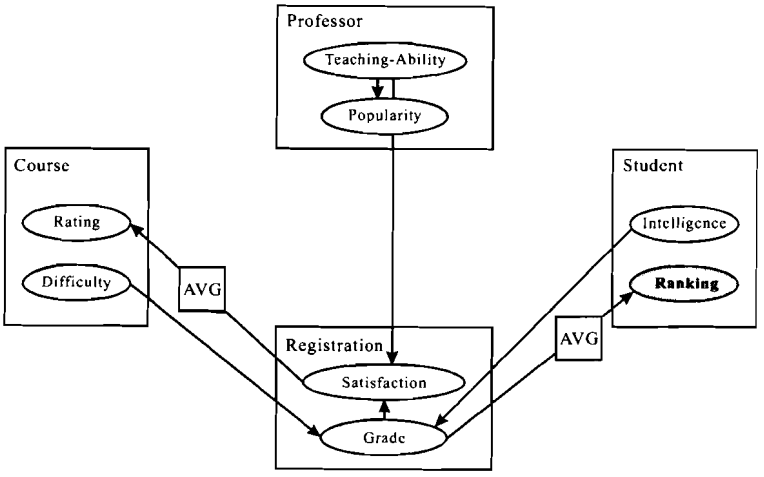


Figure 4.4: The PRM structure of university domain

checked first because it is the closest node, then *Intelligence* (the next closest node), until all the candidates consisting of a single attribute are processed.

In the next iteration, we will consider all the candidates consisting of two attributes. The child of the candidate $\{Grade\}$ is $\{Grade, Intelligence\}$. The Candidate-gen checks whether *Intelligence* is independent of *Ranking* given *Grade*. If true, $\{Intelligence, Grade\}$ is a derivable candidate. Otherwise, $\{Intelligence, Grade\}$ is a generating candidate. The next candidate to be checked is $\{Grade, Satisfaction\}$, and so on, until all the candidates at the second level of the enumeration tree have been processed.

Given that there are seven nodes in the university example, if no pruning step is carried out, the number of candidates would have been $2^7 = 128$. At the end of the Candidate-gen execution, the number of generating candidates is reduced to 28. Instead of the original 128 candidates, now only 28 candidates need to be evaluated by the Rule-gen method. In addition, the Candidate-gen method favors candidates of smaller size, and discards candidates of larger size. Considering that evaluating a candidate involves checking all the combinations of the (attribute, value) pairs, a larger set would generate more attribute value combinations to be tested in the rule generation phase. The computational effort is further reduced by pruning candidates of larger size.

The candidate generation method tests all the combinations of the nodes in the graph. The total number of generating candidates increases exponentially with the number of nodes to be processed. We can also reduce the number of generating candidates by limiting the

number of candidate nodes. One meaningful and condensed candidate node set is the Markov blanket of the consequence node. The Markov blanket of a node is interesting because it identifies all the nodes that shield off the node from the rest of the network. This means that the Markov blanket of a node is the only knowledge that is needed to predict the behavior of the consequence node.

Definition 4.8 (Markov Blanket) *A Markov blanket of a node A is a minimal set of nodes $MB(A)$ such that A is conditionally independent of any other node B given $MB(A)$. The Markov blanket for node A in the directed acyclic graph is defined as:*

$$MB(A) = parent(A) \cup child(A) \cup \{w : child(w) \cap child(A) \neq \emptyset\}$$

.

Essentially, the Markov blanket of A encompasses A's parents, children, and the parents of A's children but not A itself. From the definition of Markov Blanket, we know that all the nodes in MB(A) must have a greater impact on the value of consequence node A. So, instead of considering all the nodes in the graph, we can only consider those that are inside $MB(A)$, thus reducing the search space and the number of generating candidates. Our second method of candidate generation is to discover all the candidates in the form of $X \rightarrow A$, such that X is a subset of $MB(A)$. The Markov-Candidate-gen algorithm is illustrated in Algorithm 4.

Algorithm 4 Markov-Candidate-gen(A)

Input: A PRM G, the consequence node A.
Output: The power set of MB(A) C.
1: Find $MB(A)$
2: **for** every $Y \subseteq MB(A)$ **do**
3: $C \rightarrow C \cup \{Y\}$
4: **end for**
5: return C

We simply treat the powerset of the MB(A) as the candidate set. This method allows us to evaluate fewer candidates than the PARM approach. An obvious weakness of the Markov-Candidate-gen is its incapability to discover all the high confidence rules. $MB(A)$ shields off the node A from all the rest nodes in the network. But if any node from $MB(A)$ is missing, then the incomplete $MB(A)$ can not shield the node A from the rest of the network.

In this case, a node outside $MB(A)$ may be dependent on A. For example, $MB(Ranking)$ only contains one node *Grade* in the university domain. If the value of *Grade* is known, any other node is independent with *Ranking*. However, if the value of *Grade* is unknown, then *Ranking* may depend on the other nodes in the network, such as *Course.Difficulty* or *Student.Intelligence*.

The choice of the Markov-Candidate-gen method and the Candidate-gen method ultimately depends on the user's preference. If the dataset is large and the user prefers to find top rules efficiently, then the Markov approach is recommended. On the other hand, if the user is interested in finding all the high confidence rules, then the Candidate-gen method should be a better choice.

We can also utilize the Markov blanket to speed up the discovery of all generating candidates in candidate generation phase. When checking whether node N and A are independent given a set of nodes X, if X is a superset of the $MB(A)$, then we can assert that N is conditionally independent of A without running the Bayes-ball algorithm. Therefore, the candidate-gen method only needs to run the Bayes-Ball algorithm $2^n - 2^{n-m}$ times, where n is the total number of nodes and m is the number of nodes in $MB(A)$. In our university example, the $MB(ranking)$ contains only one node, namely *Grade*. For every candidate which is a superset of $\{Grade\}$, we can safely discard it without running the Bayes-Ball algorithm.

4.4.2 Rule Generation

A generating candidate discovered from the candidate generation phase is a antecedent attribute set that the consequence attribute may depend on. In the rule generation phase, our goal is to find the corresponding value of each attribute for a generating candidate. We need to calculate the local confidence tables (LCT) for the consequence attribute A by querying the database.

Definition 4.9 (Local Confidence Table) *Given a candidate (attribute set) X and the consequence node A, the Local Confidence Table defines the distribution over the node A given each combination of the candidate values: $P(A \mid X)$.*

The definition of LCT is similar to the conditional probability distribution (CPT) defined in the bayesian network. In a bayesian network, each node is associated with one CPT specifying the conditional probability of this node given its parents. In our case, each node

is associated with a number of generating candidates, and each generating candidate has its own LCT specifying the conditional probability of the node given the candidate. In the association rule context, the LCT of a given candidate is actually a collection of confidences of the rule $X = x \rightarrow A = a$ over the combination of antecedent attribute values.

Suppose we have the LCT for the node *Grade* given the candidate *Student.intelligence* as follows:

Table 4.1: LCT of the generating candidate $\{intelligence\}$

Intelligence	P($Grade = A|I$)	P($Grade = B|I$)	P($Grade = C|I$)
H	4/6	2/6	0
M	1/4	2/4	1/4
L	1/7	1/7	5/7

Since the *intelligence* node and the *grade* node are in two different tables, we need to define the conditional probability $P(Grade|Intelligence)$ first. If two attributes are dependent upon each other and they are in two different tables, then there must be a foreign key chain connecting these two tables. To compute the LCT of the candidate $\{student.intelligence\}$, we simply need to execute a foreign-key join of the table *Registration* and the table *Student*. The size of the joined table is exactly the size of *Registration* table based on referential integrity. Then we use "count" and "group by" queries to get the statistics in the LCT.

The support and confidence of the rule $Intelligence = i \rightarrow Grade = g$ is formulated using relational algebra notations, where σ denotes a *select* operation and $\bowtie$ denotes a *natural join* operation.

$$supp(R.Grade = g, S.Intelligence = i) = \frac{|\sigma_{R.Grade=g,S.Intelligence=i} R \bowtie S|}{|R \bowtie S|} = \frac{|\sigma_{R.Grade=g,S.Intelligence=i} R \bowtie S|}{|R|}$$

$$supp(S.Intelligence = i) = \frac{|\sigma_{S.Intelligence=i} R \bowtie S|}{|R|}$$

$$conf(S.Intelligence = i \rightarrow R.Grade = g) = P(R.Grade = g|S.Intelligence = i) = \frac{|\sigma_{R.Grade=g,S.Intelligence=i} R \bowtie S|}{|\sigma_{S.Intelligence=i} R \bowtie S|}$$

The relational algebra notations for calculating the confidence values can be easily translated into SQL queries. For example, the confidence calculation for the rule $S.Intelligence = i \rightarrow R.Grade = g$ consists of three steps. The first step is to join related tables. The second step is to query the support of $S.Intelligence = i$ and $S.Intelligence = i \bigwedge R.Grade = g$. The third step is to calculate the confidence based on the support values obtained from the second step. The SQL statement for querying the support of $S.Intelligence = i \bigwedge R.Grade = g$ is as follows:

```
SELECT COUNT (*)
FROM Registration R, Student S
WHERE R.studentID = S.ID
AND R.Grade = g
AND S.Intelligence =i
```

We can also speed up the calculation of LCT by using "count" and "group by" operators in the query. For example, we can obtain the count of all the attribute value combinations for the example candidate $\{S.Intelligence\}$ by executing the following SQL query:

```
SELECT S.Intelligence, R.Grade, COUNT (*)
FROM Registration R, Student S
WHERE R.studentID = S.ID
GROUP BY S.Intelligence, R.Grade
```

This query will return the count information for all the value combinations of $S.Intelligence$ and $R.Grade$ (Table 4.2), from which we can easily calculate all the entries in the LCT of candidate $\{S.Intelligence\}$ (Table 4.1).

Table 4.2: Query Result

	R.Grade	S.Intelligence	count
1	A	High	4
2	B	High	2
3	A	Middle	1
4	B	Middle	2
5	C	Middle	1
6	A	Low	1
7	B	Low	1
8	C	Low	5

From the LCT specified in Table 4.1, the Rule-gen method reports the following rules if the *minConf* is set to 50%.

$$S.Intelligence = high \rightarrow R.Grade = A$$

$$S.Intelligence = middle \rightarrow R.Grade = B$$

The detailed algorithm of Rule-gen is illustrated in Algorithm 5.

Algorithm 5 Rule-gen(C, A)

Input: A set of generating candidates C, the consequence attribute A.
Output: The rule set R that satisfies the *minConf* with the consequence attribute A.

```
1: Initialize R ← ∅
2: for each candidate c ∈ C do
3:    Compute the LCT of P(A|X)
4:    for all (x, a) such that P(A = a|X = x) ≥ minConf do
5:       r ← X = x → A = a
6:       R ← R∪ {r}
7:    end for
8: end for
9: return R
```

4.4.3 Rule Pruning

In this section, we describe the rule pruning method that removes redundant rules generated in the rule generation phase. In the candidate generation phase, a set of derivable candidates that lead to the generation of redundant rules have been discarded from our candidate set. However, the rules derived from the generating candidates may still contain the rules that

should be ignored by the non-redundant rule set. Assuming the consequence attribute is $S.Ranking$, both $\{S.Intelligence\}$ and $\{S.Intelligence, C.Difficulty\}$ are the generating candidates because $S.Intellegence$ cannot "d-separate" $C.Difficulty$ and $S.Ranking$. Thus, we have the two following rules:

R_1: $S.Intelligence = Low \rightarrow S.Ranking = Low$ $confidence = 0.6$
R_2: $S.Intelligence = Low, C.Difficulty = Low \rightarrow S.Ranking = Low$ $confidence = 0.5$

If R_1 is known, then R_2 should be removed because there exists a more general rule R_1 in the resulting rule set. The detailed algorithm of Rule-pruning is illustrated in Algorithm 6.

Algorithm 6 Rule-pruning(R_i, R)

Input: A rule set generated by the Rule_gen at level i R_i, a non-redundant rule set generated so far R.
Output: The non-redundant rule set N_i.

```
for each r ∈ R_i do
  if ∃r' ∈ R such that r' ⊂ r and conf(r') ≥ conf(r) then
    remove r from R_i
  end if
end for
N_i ← R_i
return N_i
```

We check the redundancy of a given rule based on an obvious observation: if rule r' is more general than rule r, then the antecedent attribute set of r' must be a subset of the antecedent attribute set of r. Therefore, the pruning proceeds as follows. First, we find the corresponding antecedent attribute set C of r, and then we iterate through all the generating candidates that have been evaluated so far. If generating candidate C' is a subset of C, then we look up the confidence of the rule with the same attribute values as r in the associated LCT of C'. If the confidence is greater than or equal to that of r, then r is removed. In order to improve the efficiency of this procedure, the *hash table* data structure is used for storing both the candidate set and the LCTs, which facilitates fast searching for the subsets of a given candidate and the confidence value for a specific rule.

Assuming that the PRM dependency structure accurately models the dependency relationships between the descriptive attributes for a given database, we claim that PARM generates the non-redundant association rule set correctly. We prove our claim based on

the definition of the non-redundant association rule set.

Firstly, we need to show that the generated rule set is a subset of the complete association rule set and satisfies the $minconf$ constraint. This is trivial because the rule generation phase computes the confidence of each rule and only reports rules with confidence greater than or equal to the $minconf$. Also, the rule set generated by PARM must be a subset of the complete rule set.

Secondly, we need to show that for every rule pruned by PARM, there exists a more general rule in the resulting rule set. In the candidate generation phase, we enumerate all the possible attribute combinations as our candidates. All the attribute value combinations for a given attribute set are also enumerated in the rule generation phase. This guarantees that all possible patterns defined by our pattern language have been considered by PARM. Now let us consider rules pruned by PARM. The candidate generation phase prunes a set of derivable candidates based on the conditional independence relationships encoded in the PRM dependency structure. From our analysis in Section 4.3, all the rules derived from a derivable candidate can be safely pruned because the confidence of such a rule equals the confidence of a rule derived from a generating candidate. This condition also holds for rules pruned by the rule pruning phase, because a rule is pruned only when PARM finds a more general rule from the existing rule set.

Thirdly, we need to show that for every rule r in the rule set R generated by PARM, there does not exist a rule $r\prime$ in R such that $r\prime$ is more general than r. When the rule pruning phase checks the validity of rule r, if there exists a more general rule in the existing rule set, then rule r is pruned from R. Therefore, it is guaranteed that no rule is more general than r in the rule set that is generated before r. Also, all the rules generated at later iterations cannot be more general than r because the candidates generated at later iterations cannot be a subset of the antecedent attribute set of r. This proves that for every rule r in R, there does not exist a rule that is more general than r.

In summary, PARM generates the non-redundant rule set correctly.

Chapter 5

Implementation and Experiments

To test the PARM algorithm, we implemented it in Java 1.4.2.07 under Microsoft Windows XP Professional Edition and ran it on a PC with an Intel Pentium 4, 1.4GHz CPU, 1GB main memory and a 20GB hard disk. We used the well known financial database from Discovery Challenge at PKDD'99 [1] as our dataset. A high level overview of the system architecture is shown in Figure 5.1.

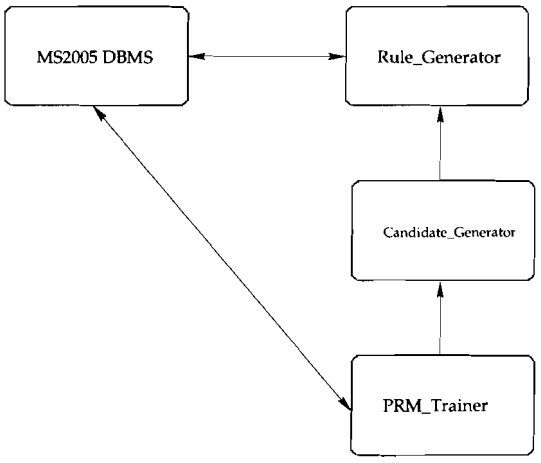


Figure 5.1: PARM System Architecture

5.1 Financial Database

The financial database comes from real world data generated and used by a Czech bank. It has 5369 clients and 4500 accounts. The data is stored in 8 tables as described below.

- The loan table describes a loan granted to a given account.
- The account table describes the static characteristics of an account.
- The client table describes the characteristics of a client.
- The disposition table relates a client to an account.
- The order table describes the characteristics of a payment order.
- The transaction table describes one transaction on an account.
- The credit card table describes a credit card issued to an account.
- The demographic data describes the demographic characteristics of a district.

For the purpose of validating our algorithm, only portions of the data will be used. One of the problems we would like to solve is to indicate whether or not the loan is successful before it is created. The data shows that about 11 percent of loans are unsuccessful. Here an unsuccessful loan means that either the loan was not paid when the contract was finished, or the client was in debt. Thus we choose the attribute "status" in the Loan table as one of our consequence attribute. This status attribute describes the payment status of the loan. The domain of status consists 4 values: 'A', 'B', 'C' and 'D'.

- Value A stands for a contract that was finished without any problems.
- Value B stands for a contract that was finished, but the loan was not paid.
- Value C stands for a running contract, OK so far.
- Value D stands for a running contract with a client in debt.

Another consequence attribute chosen for our experiment is the *loan.amount.* This attribute describes the total loan amount applied for by the client. The numerical values of this attribute are discretized into 3 categorical values: $\leq$ 80000, 80000-160000 and $\geq$ 160000.

5.1.1 Preliminary Data Analysis

During the preliminary investigation of our original financial database, some interesting features were found. These features will help us to better understand the dataset so as to make appropriate decisions at the pre-processing step.

- Each client can access only one *account_Id* while one account can be associated with one or two clients.
- If two clients share the same account, then one of them has the owner right to the account while the other has user privileges.
- There is at most one credit card issued for each account.
- There are a total of 682 loan contracts issued. 30% of them with the status value 'A'(loan finished without problem), 59% with the status 'C'(in progress, running OK), 4.5% with the status 'B' (loan contract finished, but loan not paid), 6.5% of them with status 'D' (in progress, but client is in debt).

The features listed above were obtained by querying the preliminary database using some simple SQL queries.

5.1.2 Attributes Selection

As not all the data in the databases are related to our mining task, in this step we select task related data from the original data source. The main task is to select task related tables and attributes. We pre-calculate some monthly-based attributes from the existing attributes. The following existing attributes, that may affect the value of *loan.status*, are selected: *loan.loan_id*, *loan.account_id*, *loan.amount*, *client.client_id*, *client.clientType*, *card.card_id*, *card.cardType*. In addition, four new attributes are computed based on the original data (see Figure 5.2 for the schema):

- *client.age* is derived from *client.birth_number*.
- *client.account_id* is obtained by joining the client and the disposition table. Since each client can access only one account, no potential information loss would be caused by joining these two tables.

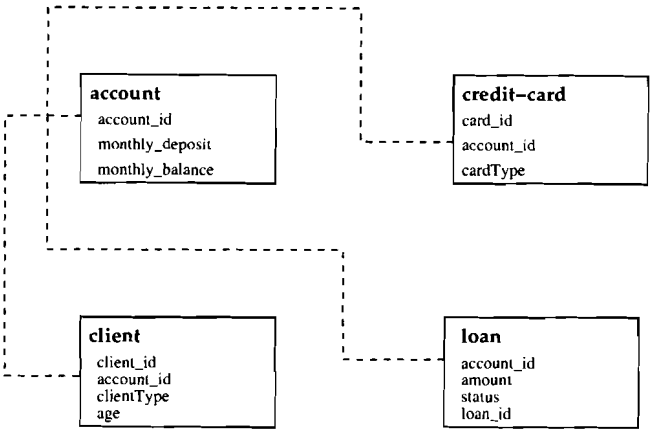


Figure 5.2: The schema of post-processed financial database

- Two new attributes were introduced for each account: *account.monthly_deposit* describes the average monthly deposit amount in 1998. This attribute is computed by aggregating the transaction.amount for this account with the transaction type 'deposit' within year 1998. *account.monthly_balance* is the difference between the average monthly deposit and average monthly withdrawal. A negative *monthly_balance* value indicates more withdrawals than deposits on a monthly basis.

5.1.3 Data Discretization

PARM algorithm is designed to handle nominal data only. This entails pre-processing all the numerical attributes into categorical equivalents based on fixed numeric thresholds (global discretization). Two commonly used global discretization methods include Equal Width Discretization (EWD) and Equal Frequency Discretization (EFD). EWD divides the number of values into k intervals of equal width and EFD divides the sorted values into k intervals so that each interval contains approximately the same number of values. Although these two methods may be deemed simplistic, they are commonly used in practice. In our experiment we use the EFD method to transform the continuous values of *account.monthly_deposit*, *account.monthly_balance*, *client.age* and *loan.amount* into categorical values.

5.1.4 PRM Structure

The input of the PARM algorithm contains the PRM structure of the target database. In order to experiment with PARM on the Financial database, we extend the Bayesian network

implementation BNJ[1] for implementing a PRM structure trainer. Our implementation is based on OpenJGraph which is a commonly used open source Java package for graphs[2]. The execution time of training the PRM is around 3 seconds for the financial database, which is ignorable in comparison with the execution time of PARM. The resulting PRM structure graph is illustrated in Figure 5.3

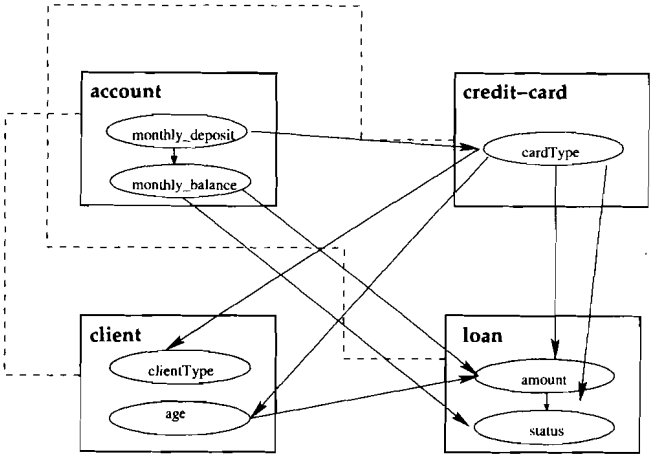


Figure 5.3: PRM structure of financial database

5.2 Experiments

We performed a number of experiments on the Financial database. Although this is a well known data set and a few analyses have been conducted on this data set [5] [20] [15], few results were comparable with ours due to the different rule format or different preprocessing strategy. Rather, we will show PARM's performance under various conditions. Apart from experimental comparison choices, an efficient implementation is needed in order to achieve better performance and result quality. The following decisions have been made in our implementation.

- Attributes with numeric values need to be discretized beforehand. We chose to use the EFD method to divide a range of continuous value into several intervals. A large number of intervals would generate a large number of candidates. Too many candidates

[1] Bayesian Network tools in Java, http://bnj.sourceforge.net/

[2] http://bnj.sourceforge.net

in turn would result in a performance degradation and many similar patterns, which would only differ slightly in a single attribute. Therefore, we chose to use 3-6 intervals for numeric attributes.

- Two consequence attributes were chosen in our experiments. One is *loan.status* and the other is *loan.amount*. In addition, we performed two separate sets of experiments on the loan status test. One is for successful loans which contain two target values ($loan.status = A$ and $loan.status = C$), and the other is for unsuccessful loans, which contain target values B and D. This decision was rooted from our initial investigation of existing data. Among 682 loan cases, only 11% were unsuccessful and 89% were successful. As the percentage of successful loans is approximately 9 times larger than that of the unsuccessful loans, a uniform confidence threshold will leave out many interesting rules of unsuccessful loans. In our experiment, two different threshold values were set: one for successful loan rules and the other for unsuccessful loan rule. The minimum confidence for successful loan rules is 0.5 and the minimum confidence for unsuccessful loan rules is 0.1.

- Confidence and lift are two rule measures supported by PARM:
  $confidence(A \rightarrow B) = P(A \mid B)$
  $lift(A \rightarrow B) = \frac{P(AB)}{P(A) \cdot P(B)}$

5.2.1 Experimental Results

We performed three sets of experimental evaluations on the financial datasets. Each targets one of the successful loans, the unsuccessful loans, and the loan amount. To show the improved performance of PARM, we compare PARM with a Non-Pruning PARM algorithm (NP-PARM). The Non-Pruning PARM simulates an association rule mining algorithm which does not perform PRM-based candidate pruning (i.e. assuming the PRM structure is a complete graph of all the descriptive attributes in the database). We also compare our Markov Blanket-based algorithm (MB) with PARM and NP-PARM for performance evaluation. Table 5.1, Table 5.2 and Table 5.3 depict the summary of our test results under two interestingness measures (confidence and lift) for successful loans, unsuccessful loans and loan amounts, respectively. In these tables, *Minimum Score* refers to the user-defined threshold for rules to be reported. *Best Score* is the score of the best rule. *Number of rules* refers to the number of rules generated. This number depends on the user-defined

minimum score. The tables additionally indicate the number of candidates (attribute set) to be evaluated. Finally, the running time in seconds is given.

Table 5.1: Financial database result on successful loans

	confidence			lift		
	NP-PARM	PARM	MB	NP-PARM	PARM	MB
best score	1	1	1	3.36	3.36	2.83
minimum score	0.5	0.5	0.5	1.5	1.5	1.5
number of rules	942	313	51	374	170	19
number of candidates	63	42	7	63	42	7
execution time	473	137	8	472	158	11

Table 5.2: Financial database result on unsuccessful loans

	confidence			lift		
	NP-PARM	PARM	MB	NP-PARM	PARM	MB
best score	1	1	1	28.33	28.33	18.88
minimum score	0.1	0.1	0.1	1.5	1.5	1.5
number of rules	219	126	10	349	242	19
number of candidates	63	42	7	63	42	7
execution time	433	69	7	432	70	11

Table 5.3: Financial database result on loan amount

	confidence			lift		
	NP-PARM	PARM	MB	NP-PARM	PARM	MB
best score	1	1	1	3.55	3.55	3.33
minimum score	0.5	0.5	0.5	1.5	1.5	1.5
number of rules	775	307	101	707	350	92
number of candidates	63	43	15	63	43	15
execution time	789	136	30	745	199	30

The results show that PARM achieved about a 40% reduction in the number of candidates, a 60% reduction in the number of rules and 70% reduction in the execution time compared with NP-PARM . The improved efficiency is due to the fact that PARM generates fewer candidates and accesses the database fewer times. Further more, the reduction in the execution time is much larger than the reduction in the number of candidates to be tested. This is because PARM avoids generating many long candidates which would take more time to evaluate. Assuming that each attribute has m distinct values on average, and

the cardinality of a candidate is n, then the number of rule hypotheses to be tested for this candidate is m^n. Therefore, the number of rule hypotheses is exponential in the cardinality of the candidate. All the derivable candidates (omitted by PARM) for the loan status are listed in Table 5.4. We can see that the cardinalities of the derivable candidates in general are quite large considering that the maximum cardinality of a candidate in the financial database is 6.

The MB-based algorithm generates only 24% and 35% of candidates compared with NP-PARM and PARM, respectively. Hence the execution time and the number of rules are only a small fraction of what is generated by the PARM family. But the rule set discovered by MB is only a subset of the rule set discovered by the PARM family and the completeness power is lost.

We perform another set of experiments to show the impact of the minimum confidence threshold on the size of rule sets and the performance. Figure 5.4 shows the sizes of rule sets generated by MB, PARM and NP-PARM for successful loans under different minimum confidence settings. The size difference between the PARM rule set and the NP-PARM rule set becomes more prominent as the minimum confidence decreases. This is because the length of rules becomes longer when the minimum confidence decreases, and longer rules are likely to be omitted by PARM algorithm than shorter rules. The same comparisons are conducted on the unsuccessful loans and loan amounts. The results are shown in Figure 5.5 and Figure 5.6, respectively.

Table 5.4: Derivable candidates for loan status

derivable candidates	cardinality
client.age, card.cardType, client.clientType	3
account.monthly_balance, client.age, card.cardType, account.monthly_deposit	4
loan.amount, account.monthly_balance, card.cardType, account.monthly_deposit, client.clientType	5
card.cardType, client.clientType	2
loan.amount, account.monthly_balance, card.cardType, account.monthly_deposit	4
loan.amount, account.monthly_balance, card.cardType, client.age	4
card.cardType, client.age, account.monthly_deposit, client.clientType	4
loan.amount, card.cardType, client.clientType	3
account.monthly_balance, card.cardType, account.monthly_deposit	3
account.monthly_balance, card.cardType, account.monthly_deposit, client.clientType	4
account.monthly_balance, card.cardType, client.age, account.monthly_deposit, client.clientType	5
card.cardType, account.monthly_deposit, client.clientType	3
loan.amount, card.cardType, account.monthly_deposit, client.clientType	4
loan.amount, account.monthly_balance, client.age, card.cardType, account.monthly_deposit	5
account.monthly_balance, card.cardType, client.clientType	3
loan.amount, account.monthly_balance, card.cardType, client.clientType	4
loan.amount, card.cardType, client.age, account.monthly_deposit, client.clientType	5
loan.amount, client.age, card.cardType, client.clientType	4
account.monthly_balance, client.age, card.cardType, client.clientType	4
loan.amount, account.monthly_balance, client.age, card.cardType, client.clientType	5
loan.amount, account.monthly_balance, card.cardType, client.age, account.monthly_deposit, client.clientType	6

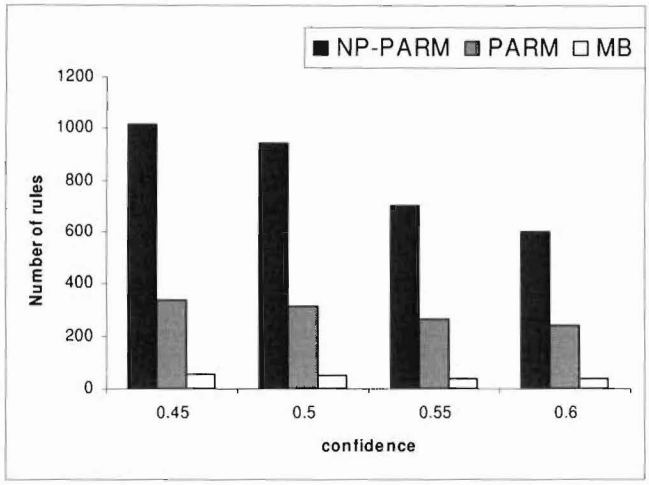


Figure 5.4: The comparison of sizes of rule sets for successful loans

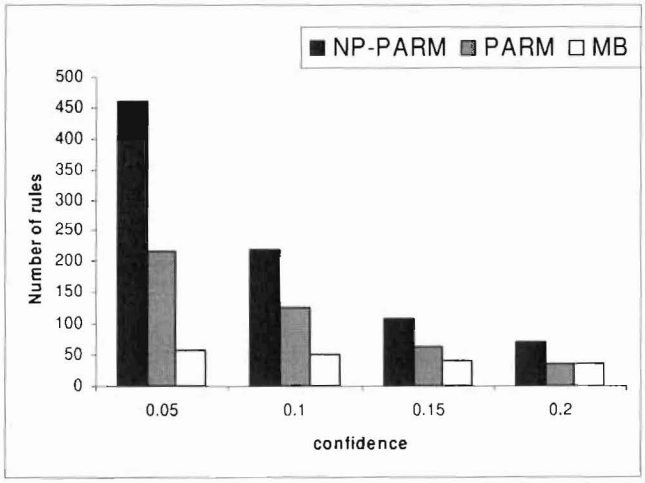


Figure 5.5: The comparison of sizes of rule sets for unsuccessful loans

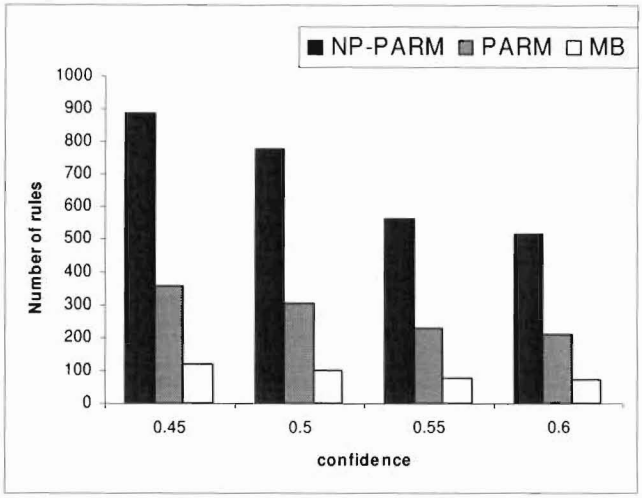


Figure 5.6: The comparison of sizes of rule sets for loan amount

Figure 5.7 shows the execution time comparison (in seconds) for successful loans. As expected, there is no significant performance difference among various minimum confidence settings. This is because our performance improvement is achieved by the candidate pruning. When the number of candidates is fixed, different minimum confidence thresholds would not cause significant change in performance. The same comparison for bad loans and loan amounts are shown in Figure 5.8 and Figure 5.9.

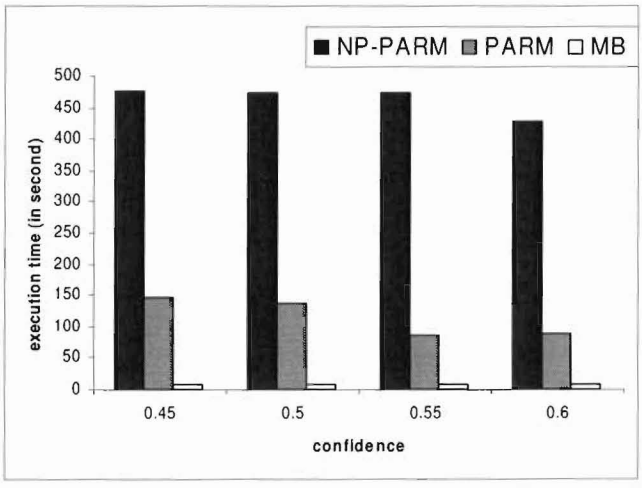


Figure 5.7: The comparison of execution time for successful loans

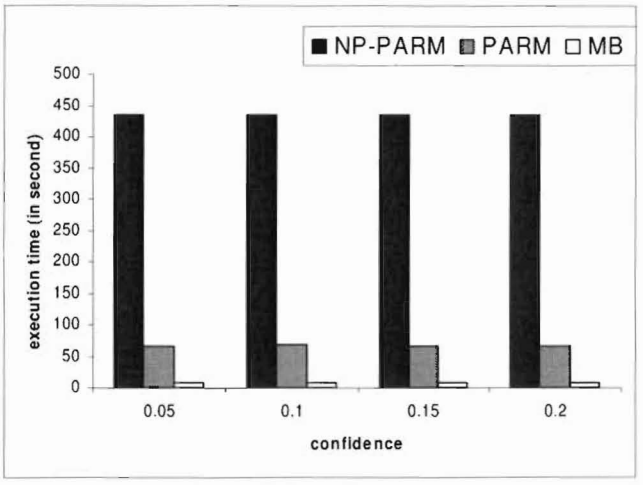


Figure 5.8: The comparison of execution time for unsuccessful loans

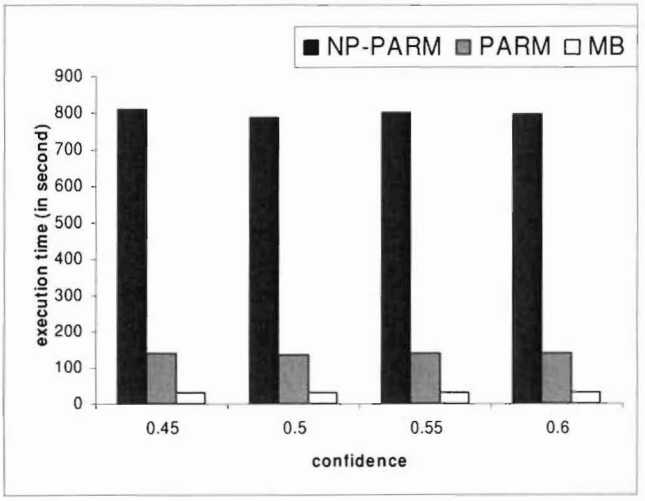


Figure 5.9: The comparison of execution time for loan amount under different minimum confidences

5.2.2 Support Distribution Analysis

Apriori-based association rule mining approaches use a uniform minimum support and minimum confidence during the mining process. This approach may miss those rules involves the minority class, which may be the class that we are interested in. In the financial database, the rule set with respect to the unsuccessful loans is a good example of this case. In order to find the low coverage rules, the Apriori-based algorithms have to set the minimum support very low, which may cause performance degradation. To illustrate this problem, we output both the support and the confidence of each rule for successful loans generated by PARM. Figure 5.10 shows the support and confidence distributions for the 313 discovered rules.

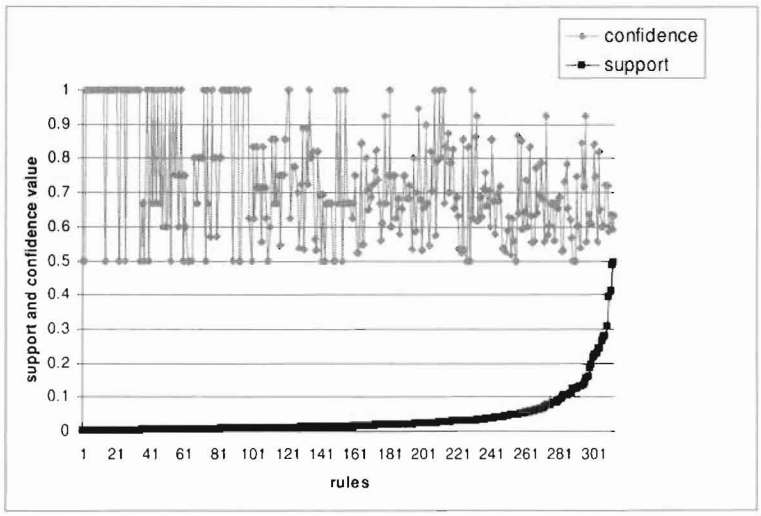


Figure 5.10: The support and confidence distribution for successful loan rules

The x-axis in Figure 5.10 represents the rule index and the y-axis represents the support and confidence values. All the rules are sorted by their support values. From the result shown above, we observed a skewed support distribution of the discovered rules. 313 rules in total were discovered by PARM when the minimum confidence was set to 50%. Among the 313 discovered rules, the support values ranged from 0.1209% to 49.4%. 43% of the rules had support less than 1% and approximately 90% of the rules had support less than 10%. If we set the minimum support to 10%, then 90% of the high confidence rules would be undiscovered. The skewed support distribution also makes choosing a proper minimum support threshold more challenging.

Another interesting observation is that the lower support rules tend to have higher confidence values whereas the higher support rules tend to have lower confidence values. A closer investigation shows that most of the high confidence rules are longer rules. Longer rules usually have lower supports due to the multiple rounds of the refinement. In this case, an improper minimum support threshold would not only miss rules exceeding the confidence threshold, but also would miss rules with top confidences.

5.2.3 Conceptual Comparison with Safarii

Safarii is a Data Mining package which provides an implementation of multi-relational rule discovery. Interestingly, Safarii also finds rules with a previously specified consequence without minimum support requirement, and it is used to perform a number of experiments on the same financial database for loan prediction. However, a quantitative comparison with Safarii is not possible because of two reasons. First, as we introduced in Chapter 2, Safarii uses selection graphs as its pattern language, which is different from our pattern language. Different pattern languages may discover different sets of patterns. Second, PARM and Safari pre-process the dataset differently and their pre-processing methods are not available to us. Rather, we list the relevant experimental results produced by Safarii and compare PARM to Safarii conceptually.

The basic algorithm of Safarii works as follows. It continues considering candidate rules in the order specified by the search strategy with the guide of the *selection graph* until all available candidates have been exhausted. The efficiency is achieved by using a Client/Server architecture to separate the search process and the candidate evaluation process through *data mining primitives*. "Data mining primitives are data structures of statistical information concerning some aspects of the contents of the database." [15]. Safarii implements a set of predefined primitives and the candidate evaluation is only through the use of primitives instead of the direct access to the database. One primitive can provide statistical information for evaluating multiple patterns, thus optimizing the database accessing time. PARM, on the other hand, focuses on pruning candidates algorithmically. Efficiency is achieved by eliminating derivable candidates based on the conditional independence relationships among the attributes.

Table 5.5 and Table 5.6 are the result summary of Safarii from [15]. The results were produced with a refinement-depth of 5 and the best-first search strategy. Since Safarii system does not perform candidate pruning, a large number of hypotheses need to be tested. As

Table 5.5: Financial database result on successful loans (Safarii)

	confidence	leverage
best score	0.989	0.0375
minimum score	0.889	0.02
number of rules	511	164
number of primitives	2001	2043
number of hypotheses	113236	173604
execution time	786	918

Table 5.6: Financial database result on unsuccessful loans (Safarii)

	confidence	leverage
best score	0.176	0.0219
minimum score	0.111	0.02
number of rules	489	8
number of primitives	1776	1897
number of hypotheses	114122	135771
execution time	1454	1844

shown in the results above, an average of above 120,000 hypotheses are generated in one test run. Another weakness of Safarii is the lack of redundant rule checking and eliminating step, so it tends to report interesting patterns in multiple copies. "Thus far the rules discovered have been treated as an independent rule set, with the disadvantage of having many similar rules" [15]. Multiple copies of logically equivalent rules require another post-pruning phase for redundancy elimination.

5.2.4 A Sample Run

In this section, we show PARM at work for the unsuccessful loan example used throughout this chapter. The PRM structure is presented in Figure 5.3 and the confidence threshold is set to 10%. Given these inputs, PARM reports 126 rules in total. Assuming that all attributes in the database can be uniquely identified without using table names, Table 5.7 lists the top 28 rules with confidence above 25%. The attribute names in the list are explained below:

- age: is the age of the client, and the value domain is $\{20-40, 40-60, >60\}$.
- amount: is the total loan amount the client applies for, and the value domain is $\{<80000, 80000-160000, >160000\}$.
- clientType: is the type of the client, and the value domain is $\{owner, user\}$.
- balance: is the abbreviation of *monthly_balance*, which stands for the difference between average monthly deposit and average monthly withdrawal of each account, and the value domain is $\{<-600, -600-0, 0-900, >900\}$.
- cardType: is the credit card type, and the value domain is {junior, classic, gold}
- deposit: is the abbreviation of *monthly_deposit*, which stands for the average monthly deposit amount of an account, and the value domain is $\{0-7000, 7000-20000, 20000-82000\}$.

We can interpret the first rule as follows. If a client is older than 60, has a loan with an amount of more than 160000, has an average monthly balance between 0 and 900, and has an average monthly deposit between 7000 and 20000, then his loan contract was finished with loan not fully paid. The second rule says: if a client has a loan with an amount of more than 160000, and his average monthly deposit is between 0 and 7000, then his loan contract is still running, but the client is in debt. From these two sample rules, we can see that the discovered rules are very useful and can provide guidance of risk analysis for future loan applications.

Table 5.7: All rules for unsuccessful loans with confidence above 0.25

rules	conf
$age => 60, amount => 160000, balance = 0 - 900, deposit = 7000 - 20000 \rightarrow status = B$	1
$amount => 160000, deposit = 0 - 7000 \rightarrow status = D$	1
$age = 20 - 40, amount => 160000, deposit = 0 - 7000 \rightarrow status = D$	1
$age => 60, amount => 160000, deposit = 0 - 7000 \rightarrow status = D$	1
$amount => 160000, balance = 0 - 900, deposit = 0 - 7000 \rightarrow status = D$	1
$amount => 160000, balance =< -600, deposit = 0 - 7000 \rightarrow status = D$	1
$age => 60, balance =< -600, deposit = 0 - 7000 \rightarrow status = D$	1
$age => 60, deposit = 0 - 7000 \rightarrow status = D$	1
$age = 20 - 40, clientType = owner, balance = 0 - 900, deposit = 0 - 7000 \rightarrow status = D$	0.5
$age => 60, amount = 80000 - 160000, balance => 900, deposit = 7000 - 20000 \rightarrow status = D$	0.5
$age => 60, amount =< 80000, balance = -600 - 0, deposit = 20000 - 82000 \rightarrow status = D$	0.5
$balance =< -600, deposit = 0 - 7000 \rightarrow status = D$	0.5
$age = 20 - 40, amount => 160000, cardType = gold \rightarrow status = D$	0.5
$age = 20 - 40, amount => 160000, balance = -600 - 0, deposit = 7000 - 20000 \rightarrow status = D$	0.33
$age = 40 - 60, amount => 160000, balance = -600 - 0, deposit = 7000 - 20000 \rightarrow status = D$	0.33
$age = 20 - 40, clientType = owner, deposit = 0 - 7000 \rightarrow status = D$	0.33
$age => 60, amount => 160000, cardType = classic \rightarrow status = D$	0.33
$amount => 160000, cardType = gold, balance => 900 \rightarrow status = D$	0.33
$age = 20 - 40, balance = 0 - 900, deposit = 0 - 7000 \rightarrow status = D$	0.33
$amount => 160000, balance = -600 - 0, deposit = 7000 - 20000 \rightarrow status = D$	0.3
$amount => 160000, balance = 0 - 900, deposit = 7000 - 20000 \rightarrow status = B$	0.27
$age => 60, clientType = owner, balance = -600 - 0, deposit = 20000 - 82000 \rightarrow status = D$	0.25
$age => 60, amount =< 80000, balance = -600 - 0, deposit = 7000 - 20000 \rightarrow status = B$	0.25
$age => 60, amount => 160000, balance =< -600 \rightarrow status = D$	0.25
$deposit = 0 - 7000 \rightarrow loan.status = D$	0.25
$age = 40 - 60, amount => 160000, clientType = owner, balance => 900, deposit = 7000 - 20000 \rightarrow status = B$	0.25
$age => 60, amount => 160000, clientType = owner, balance => 900, deposit = 20000 - 82000 \rightarrow status = D$	0.25
$age => 60, amount =< 80000, clientType = owner, balance => 900, deposit = 7000 - 20000 \rightarrow status = B$	0.25

Chapter 6

Conclusion

In this chapter, we conclude the thesis with an overall review and a general discussion on future works.

6.1 Summary

In this thesis, we focused on the association rule discovery in the context of relational database perspective of data mining. We started by investigating current approaches for discovering association rules from relational databases and revealed their limitations, such as the problem associated with the minimum support requirement. We explored the independency relationship presented by the PRM structure and delved into the correlation between the conditional independency and the confidence measure of the rules. Inspired by our findings, we developed the PARM algorithm for discovering single consequence attribute association rules from relational databases without a minimum support threshold. We consider this approach to be an interesting alternative because for some applications, minimum support is a constraint primarily required for the efficient computation purpose. The PARM algorithm adopts the attribute-oriented pruning approach based on the PRM structure, integrates database operations with the learning processes, and provides an efficient way of extracting non-redundant association rules.

The PARM algorithm is designed for learning single consequence attribute association rules from relational databases. Fundamentally different from the minimum support driven approach, the PARM algorithm prunes antecedent attribute sets based on the conditional

independency relationships. We have defined a new candidate set, namely generating candidate set, which excluded all the derivable candidates from the candidates set. In addition, a novel algorithm has been proposed to filter out the derivable candidates and create the generating candidates at the candidate generation step. Only the generating candidates need to be evaluated at the next step, thus saving the evaluation time for the derivable rules. By eliminating the derivable candidates, the final rule set contains a smaller number of more general and non-redundant rules. Our experimental results demonstrated that the rule set generated by PARM is significantly smaller than that generated by the non-pruning PARM, and the improved efficiency is the consequence of fewer candidates generated and fewer database accesses required by PARM.

6.2 Future Works

PARM finds a set of rules with a single fixed consequence attribute. The format of the discovered rule is similar to the rules from an associative rule-based classifier. In addition, the associative classification approach evaluates a rule by its coverage and accuracy measure. The coverage measurement of a rule R is the percentage of tuples that are covered by the rule antecedent and the accuracy of a rule R is the percentage of correctly classified tuple among all the tuples covered by the antecedent of the rule. In the context of association rule mining, the accuracy of a rule is actually the confidence, and the coverage is the support. As such, the output of the PARM algorithm is a perfect candidate as a rule base for the associative classification method under multi-relational settings. The remaining problem is how the rules generated by PARM should be analyzed and used for the associative classifier. This can be one of the future works of this thesis.

Although the PARM algorithm can efficiently learn a set of association rules with predefined consequence attribute based on the PRM structure, one major restriction inherited from the probabilistic model is the domain type restriction. So far all the numerical values have to be discretized in the preprocessing step before learning the PRM structure from the database. Our approach uses the simple Equal-Depth discretization method at the data preprocessing step. One possible improvement of dealing with the numerical attribute value is to adaptively partition a numerical attribute into a set of disjoint intervals with a suitable domain cardinality. For example, using the clustering technique for finding suitable intervals. Another solution is to incorporate the Hybrid Bayesian networks [21] technique

into our PRM structure learning step. Hybrid Bayesian network is an extension of the traditional BN learning that can deal with both numerical and categorical data. Either of the approaches mentioned above would be an interesting direction for future work.

The PARM algorithm requires at least one threshold of the interestingness measure (e.g. minimum confidence or minimum lift). However, in our financial database experiment, two different threshold values need to be set: one for successful loan rules and another one for unsuccessful loan rules. This is because in general, the unsuccessful loan rules have much lower confidence than the successful loan rules. It is clearly an obstacle for automatic rule generation. Hence, further research on the interestingness measures under the multi-relational setting and how to automatically choose the suitable thresholds remains a crucial step for multi-relational interesting rules mining.

The PARM algorithm presented in this thesis is efficient due to the use of the attribute-oriented pruning strategy. As an important source of efficiency is lost (the monotonic property of the minimum support), the efficiency of the algorithm highly relies on the PRM structure learned from the database. Specifically, the execution time of the PARM algorithm grows exponentially relative to the number of edges in the PRM network. If the resulting PRM structure contains many edges, the performance may not be satisfactory for industrial sized databases. Hence, incorporating other techniques, (i.e. sampling, parallel techniques or client-server architecture proposed by MRDM) into our current approach to speed up rule mining process is another interesting topic for future works.

Although rule generation has been studied for many years, incorporating and adapting current algorithms into the rich structured data mining is still a challenging problem and will draw a lot more attention in the coming years. The algorithm proposed in this thesis is an attempt to employ a wide spectrum of statistical analysis, machine learning and graph theories to solve a typical problem in data mining: multi-relational rule discovery. The problems outlined above suggest that more research needs to be conducted to find more efficient and effective ways of generating meaningful patterns from relational databases.

Bibliography

[1] Pkdd'99 discovery challenge guide to the financial data set. http://lisp.vse.cz/pkdd99/berka.htm.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499, 1994.

[3] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 255–264, 05 1997.

[4] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[5] D. Coufal, M. Holena, and A. Sochorova. Coping with discovery challenge by guha. In *Discovery Challenge. A Collaborative Effort in Knowledge Discovery from Databases*, 1999.

[6] L. Dehaspe and H. Toivonen. Discovery of relational association rules. In *Relational Data Mining*, pages 189–212. Springer-Verlag, 2001.

[7] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

[8] L. Getoor. *Learning Statistical Models from Relational Data.* PhD thesis, Stanford University, 2001.

[9] L. Getoor, Friedman.N., D. Koller, and A. Pfeffer. *Relational Data Mining chapter 13: Learning Probabilistic Relational Models.* Springer-Verlag, 2001.

[10] B. Goethals and J. V. Bussche. Relational association rules: getting warmer. *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery in Data Mining*, 2447:125–139, 2002.

[11] J. Han and M. Kamber. *Data Mining.* Morgan Kaufmann, 2001.

[12] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *2000 ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12, 05 2000.

[13] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[14] M. I. Jordan and C. M. Bishop. *An Introduction to Graphical Models.* 2000.

[15] A. J. Knobbe. *Multi-Relational Data Mining.* PhD thesis, 2004.

[16] A. J. Knobbe, H. Blockeel, A. P. J. M. Siebes, and D. M. G. van der Wallen. Multi-relational data mining. Technical report, Centrum voor Wiskunde en Informatica, 1999.

[17] N. Lavrac, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In *Ninth International Workshop on Inductive Logic Programming (ILP'99)*, pages 174–185, June 1999.

[18] B. Liu and W. Hsu. Post-analysis of learned rules. *AAAI/IAAI, Vol. 1*, pages 828–834, 1996.

[19] B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. *Knowledge Discovery and Data Mining*, pages 125–134, 1999.

[20] P. Miksovsky, F. Zelezny, O. Stepankova, and M. Pechoucek. Financial data challenge. *Workshop Notes on Discorvery Challenge*, 1:17–22, 1999.

[21] S. Monti and G. F.Cooper. Learning hybrid bayesian networks from data. Technical report, University of Pittsburgh, 1997.

[22] R.Agrawal, T.Imielinski, and A.Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (ACM SIGMOD '93), Washington,USA*, May 1993.

[23] R.Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, pages 480–487. Morgan Kaufmann, 1998.

[24] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in large databases. In *The VLDB Journal*, pages 432–444, 1995.

[25] T.Imielinski, R.Agrawal, and A.Swami. Mining associations between sets of items in massive databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,Washington D.C.*, pages 207–216, May 1993.

[26] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping of discovered association rules, 1995.

[27] Q. Zhao and S. S. Bhowmick. Association rule mining: A survey. Technical report, Nanyang Technological University, Singapore, 2003.