DECISION MAKING BASED ON ASSOCIATION RULES

by

Senqiang Zhou
B.Eng., Xi'an Jiaotong University, 1995
M.Sc., National University of Singapore, 2000

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

In the
School
of
Computing Science

SIMON FRASER UNIVERSITY

Fall 2006

APPROVAL

Name:	**Senqiang Zhou**
Degree:	**Doctor of Philosophy**
Title of Thesis:	**Decision Making Based on Association Rules**

Examining Committee:

Chair: **Dr. Wo-Shun Luk**
Professor, School of Computing Science

Dr. Ke Wang
Senior Supervisor
Professor, School of Computing Science

Dr. Martin Ester
Supervisor
Associate Professor, School of Computing Science

Dr. Binay Bhattacharya
Internal Examiner
Professor, School of Computing Science

Dr. Hui Xiong
External Examiner
Assistant Professor, Management Science and
Information Systems Department
Rutgers, the State University of New Jersey

Date Defended/Approved: Oct. 23rd, 2006

ABSTRACT

Data is being accumulated in a fast speed for many application domains, like finance and biology. Utilizing the huge volume of data to help make correct decisions is important for a company/organization to survive in this competitive world. Many general algorithms have been proposed in building decision-making systems. However, it is difficult to apply them to real-world domains without major changes due to different application natures (e.g. different goals, different data characteristics, etc). In this thesis, we study the problem of building decision-making systems using association rules for real-life applications. Unlike many existing algorithms that only touch the performance issue, we also focus on improving the interpretability of systems, which is very important in helping users understand how decisions are made (by the system). Association rules are easy to interpret and, thus, help us achieve this purpose.

There are two major contributions in this thesis. First, we propose a common framework which can serve as the guideline for building decision-making systems. The design goal of this framework is to build *both* understandable and effective systems. To help make the system understandable, the framework uses association rules as the basic elements. Also it provides the flexibility for the users to prune the system using the domain knowledge. Such pruning is very important to keep the system small and, thus, understandable. To help make the system effective, it emphasizes pushing the application goal down to the rule searching phase (the first step of system building). As our second contribution, we propose a collection of algorithms for several real-life applications by following the guidelines in the framework. All proposed algorithms share the themes in framework; however, each of them is unique and is specially designed to meet the distinct challenges of its application domain. Experiments show the effectiveness of these algorithms.

Keywords: association rule; classification; actionability; pessimistic estimation

Subject Terms: data mining

To my family

ACKNOWLEDGEMENTS

First, my gratitude goes to my senior supervisor, Dr. Wang Ke, for his guidance, generous support and comments to my research work. It is certainly my luck and pleasure to work with a supervisor so knowledgeable and amiable. I am very thankful to my supervisor, Dr. Martin Ester, for his insightful comments and advice. I really appreciate his help in improving the quality of my thesis. My gratitude and appreciation also go to Dr. Binay Bhattacharya and Dr. Hui Xiong for serving as examiners of my thesis.

I'm also grateful to all the nice people I met here. I thank my friends for their help. It is their encouragement and friendship that make my days in SFU shining and memorable. In particular, I want to express my appreciation to people in the Intelligence Database Lab who provide help and feedback on my research.

Foremost I would like to express my thanks to my parents and my wife, for the love they have been giving me all along in my life. Without them, I would not be complete. At last, I would say thanks to my daughter who shows me that it is the simple things in life that bring the most joy.

TABLE OF CONTENTS

Approval ii

Abstract iii

Dedication iv

Acknowledgements v

Table of Contents vi

List of Figures ix

List of Tables x

CHAPTER 1 Introduction 1

1.1 Decision-Making Systems 2
1.2 General Learning Algorithms 3
1.3 Motivation 4
1.4 Contributions 5
1.5 Organization of the Thesis 8

CHAPTER 2 The General Framework 9

2.1 Common Properties 9
2.2 Why Association Rules? 10
2.3 The General Framework 11
2.3.1 The Input 12
2.3.2 Generating Interesting Rules 12
2.3.3 Building the Initial Model M_I 13
2.3.4 Pruning the Initial Model M_I 14
2.4 Summary 15

CHAPTER 3 Growing Decision Trees On Association Rules 16

3.1 Introduction 16
3.2 Related Work 18
3.3 Mining Confident Rules 19
3.4 Building the Initial Classifier 24
3.5 Pruning the Classifier 27
3.5.1 Removing Redundant Rules 27
3.5.2 Building ADT 28
3.5.3 Pruning ADT 30
3.6 Experiments 34
3.6.1 Accuracy 37
3.6.2 Size 40
3.7 Conclusion 40

CHAPTER 4 Mining customer value: from association rules to direct marketing 41
4.1 Motivation 41
4.2 Introduction 43
4.2.1 Task Definition 44
4.3 Related Work 44
4.4 Generating FARs (Focused Association Rules) 46
4.5 Building the Initial Model 51
4.6 Pruning the Model 53
4.7 Validation 57
4.7.1 Sum of Actual Profit 58
4.7.2 Profit Lift 60
4.7.3 Classification 61
4.8 Conclusion 62

CHAPTER 5 Profit Mining: from patterns to actions 63
5.1 Introduction 63
5.2 Problem Definition 64
5.3 Related Work 65
5.4 Our Approach 67
5.5 Generating Rules 69
5.6 Building the Initial Recommender 71
5.7 Optimizing the Recommender 71
5.8 Evaluation 73
5.8.1 The Methodology 73
5.8.2 Results on Synthetic Data Sets 74
5.8.3 Results on Real Life Data Set 79
5.9 Conclusion 80

CHAPTER 6 Localization Site Prediction for Membrane Proteins by Integrating Rule and SVM Classification 82
6.1 Introduction 83
6.2 Algorithm Overview 86
6.2.1 Background 86
6.2.2 Our Approach 88
6.3 SVM Phase 89
6.4 Rule Phase 90
6.4.1 Generating Rules 91
6.4.2 Pruning Redundant Features 92
6.4.3 Pruning Insignificant Features 92
6.4.4 Pruning Redundant Rules 94
6.4.5 Pruning Insignificant Rules 95
6.5 Stealing Phase 96
6.6 Experiments 97
6.6.1 Performance and Significance 99
6.6.2 Interpretability 101
6.6.3 Pruning Effectiveness 102

6.7 Related Work 105
6.8 Conclusion 106

CHAPTER 7 Discovering catalog matchings on the web 108

7.1 Introduction 108
7.2 Problem Definition 111
7.2.1 Matching Rules 112
7.2.2 Catalog Matching 114
7.3 Algorithm Overview 114
7.4 Generating Matching Rules from Flat Catalogs 116
7.4.1 Flat-Head 116
7.4.2 Flat-Body 118
7.5 Generating Matching Rules from Hierarchical Catalogs 119
7.5.1 Hierarchical-Head 119
7.5.2 Hierarchical-Body 120
7.6 Evaluation 121
7.6.1 Evaluation Criteria 122
7.6.2 Performance 126
7.6.3 Effect of Terms 128
7.6.4 Simplicity 129
7.7 Conclusion 131

CHAPTER 8 Conclusion 132

8.1 Summary of the Thesis 132
8.2 Future Research Directions 134

Bibliography 136

LIST OF FIGURES

Figure 2-1 The general framework of building decision-making systems......................11
Figure 3-1 The unpruned ADT..29
Figure 3-2 The pruned ADT..32
Figure 4-1 Left: before pruning Right: after pruning..57
Figure 4-2 The distribution of donation 58
Figure 4-3 The accumulative profit lift curve 61
Figure 5-1 *H* and *MOA*(*H*) 69
Figure 5-2 The results for synthetic dataset I..76
Figure 5-3 The results for real life data set 77
Figure 6-1 The five primary localization sites in a Gram-Negative bacterial cell 84
Figure 6-2 A linear SVM in a two-dimensional space..87
Figure 6-3 Feature coefficients for the sample data..93
Figure 6-4 Selecting the prefix...97
Figure 6-5 CPU time (in seconds) for building classifiers..105
Figure 7-1 Job catalogs 109
Figure 7-2 Example hierarchy...112
Figure 7-3 The overview of catalog mapping algorithm...116
Figure 7-4 Performance comparison. Term-allowance=0% 127
Figure 7-5 Performance of the Rules approach against term-allowance 129
Figure 7-6 Reuters: H_I=H_a, term-allowance=0%..130
Figure 7-7 Tvshow: H_I=Yahoo!, term-allowance=0% 130
Figure 7-8 Industry: H_I=Canada, term-allowance=0%...131

LIST OF TABLES

Table 3-1 The sample database 22
Table 3-2 The overview of mining confident rules 23
Table 3-3 The sample data sets for a small example 26
Table 3-4 The initial classifier 27
Table 3-5 Classification on testing cases 33
Table 3-6 21 Data sets used in experiments 35
Table 3-7 Classification accuracy 36
Table 3-8 Win vs. Loss 37
Table 3-9 Size of classifiers 38
Table 3-10 Size of ADTs at different stages 39
Table 4-1 Algorithm of generating focused association rules (FARs) 48
Table 4-2 The sample database 49
Table 4-3 The D_r after applying the maximum support 50
Table 4-4 Count of items 50
Table 4-5 The FARs generated with minimum support and maximum support of 40% 51
Table 4-6 Coverage and rank of rules 53
Table 4-7 *Estimated(r)* before and after pruning 56
Table 4-8 Comparison with published results 59
Table 4-9 The confusion matrix 61
Table 6-1 A sample data set 94
Table 6-2 Data statistics 99
Table 6-3 Precision/Recall (%) on D(test) 100
Table 6-4 Comparison on interpretability 102
Table 6-5 Effectiveness of feature pruning 103
Table 6-6 Effectiveness of rule pruning 104
Table 7-1 Example data set 112
Table 7-2 The matching set 117
Table 7-3 Planted categories for the Reuters domain 124
Table 7-4 Statistics for three data sets 126

CHAPTER 1
INTRODUCTION

Nowadays data is accumulated in a fast speed thanks to the advanced data collection and storage technologies. However, more data does not necessarily lead to more information. The gap between data and decision is large. A survey[1] conducted by BusinessWeek showed that many companies are incapable of retrieving useful information from a huge amount of data to make sound decisions. On the other hand, making correct decisions is extremely important for a company to survive in this competitive world. Therefore, the task of using the accumulated data to increase the chance of making correct decision is becoming more and more urgent.

In data mining community, various technologies have been proposed to help users manage and analyse their data. Decision-making system is such an approach that reveals useful patterns from underlying data and assists user to make decisions using these patterns. In this thesis, we propose a general framework served as guidelines for building such systems. Moreover, we use the framework to solve the challenges in several real life applications. It would be ideal if we can have one algorithm that can be applied to all applications. However, the existence of such algorithm is virtually impossible due to the diverse nature of real-life applications. Instead, for each application we propose a unique algorithm which addresses the distinct challenges in that application. Though these algorithms are different, they do share some common properties identified in the framework.

A good decision-making system should at least satisfy two requirements: effective in making decisions and easy to interpret. So far the majority of decision-making algorithms only

[1] http://www.businessobjects.com/news/press/press2004/20040720_businessweek_research_comp.asp

focus on improving the effectiveness but not the interpretability. The algorithms we propose in this thesis aim at both targets.

The rest of this chapter is organized as follows. First, we give a brief explanation of decision-making systems and an introduction on some general decision-making algorithms. After that, we present the motivation and contribution of our work. Section 1.5 gives the organization of entire thesis.

1.1 Decision-Making Systems

We start with an example in explaining decision-making systems. Let's take a look at direct marketing problem which refers to a process of identifying and mailing to potential "valuable" customers. In particular, given a new customer, we need to make a decision if a (product promotion) mail should be sent to him. If a mail is sent and the customer responds, certain profit is generated. If the customer does not reply the mail, we lose the mail postage. The goal is to generate the profit as much as possible for a group of customers. Certainly the historical information would help us make such decisions. Due to the large amount of past data, it becomes virtually impossible to explore them manually. Hence, it is important to build a model (decision-making system) from the historical information to automatically give recommendations (sending or not sending a mail) on new customers.

From the previous example it is clear that a decision-making system serves the purpose of guiding users' actions to achieve an optimal goal by making effective use of the vast array of information available to the company (organization). To build such system, it requires a collection of historical data, a user-defined application goal, and optionally, domain knowledge. A good decision-making system should at least satisfy two primary requirements:

- The decisions should be reliable in the sense that users should have certain confidence to count on these suggestions.

- The system build should be easy to interpret so that users can understand how a decision is made.

For the first requirement, different applications often have different interpretations on "reliability". For example, in classification it means high accuracy while in direct marketing it means high profit. For the second one, it requires the final system is compact and presented in an easy-to-understand way. In this thesis, we use various methods to make sure the system we build for every application satisfies these two requirements.

1.2 General Learning Algorithms

Quite many algorithms have been proposed to build decision-making systems, in which most of them focus on classification, a sub-area of decision-making. In this section we give a very brief introduction on several popular algorithms that are referenced in this thesis. More details will be given in the following chapters when they are actually used and/or compared.

Association rule-based algorithms. Motivated by market-basket analysis, association rule [AIS93, AS94] is an important and active area of data mining research. It finds all the rules above the user specified thresholds in the form of $X \rightarrow Y$, where X and Y are set of items. Using association rules for decision-making generally has the following two main steps. In step 1, it mines a set of rules in the form of $X \rightarrow C_i$, where X is a set of items (or set of attribute-value pairs for table data) and C_i is a possible decision. In the second step, given an example t whose decision needs to be determined, the algorithm selects a rule (based on certain criterion) among all the rules that match t. (A rule $X \rightarrow C_i$ *matches* t if $X \subseteq t$.) The decision of the selected rule is taken as the decision for t. [DL99, LHM98, LMW00] shows some algorithms in this category.

Decision trees. They are perhaps the most widely studied inductive learning models in the machine learning and data mining community. Some representative algorithms are: CART [BFOS84], ID3 [Qui86], CN2 [CN89], C4.5 [Qui93]. Generally speaking, a decision tree is a tree in which each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes and class distribution. . The basic strategy of building a decision tree is "divide and conquer" [Qui93]: A set of cases *T* is refined into subsets of cases that are, or seem to be heading towards, single-class collections of cases.

Support vector machines (**SVMs**) [Vap95] recently demonstrated superior performance gains and robustness of SVM in many applications over traditional methods. One striking property of SVMs is the ability to produce the unique global minimum of the error function [Bur98]. Also it builds the model independent of the dimensionality of the feature space [Joa98b] because SVMs measure the complexity of hypotheses based on the margin with which they separate the data, not the number of features. However, the SVM model comes with a major defect: It involves thousands of features in a single kernel function, making it impossible to see a simple relationship between the prediction and features that trigger it.

Naïve Bayes classifiers [FGG97]. They probably are the most popular statistical classifiers that encode probabilistic relationships between variables of interest [DH73]. They can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. These classifiers learn from the data the conditional probability $P(A_i=a_i|C)$ of each attribute $A_i=a_i$ given the class label C. During classification, given a testing case $t=\{A_1=a_1,\dots, A_n=a_n\}$, it finds the class C that maximize the posterior probability $\Pi_k P(A_k=a_k|C)P(C)$. Here it makes the assumption that all attributes are independent, which might not be always true. Quite a few extensions have been developed, most of which aim at relaxing its strong independence assumption.

1.3 Motivation

The learning algorithms introduced in the previous section are designed for general-purpose tasks and do not always work effectively in real-life applications.

1. Those algorithms do not give guidelines on how to integrate domain-specific requirements into the process of system construction, which is very much needed by users. Identifying the common properties from various applications and summarizing them into an easy-to-follow framework would greatly speed up the system construction.

2. Most algorithms target at achieving high accuracy while many applications have their own application goals. For example, in direct marketing [WZYY03] the goal is to produce high profit instead of high accuracy. Such requirement makes it difficult to apply those algorithms directly without major changes.

3. Most algorithms are not easy to be adapted to accommodate different data characteristics in different applications. For example, decision tree algorithms [Qui93] perform poorly on high dimension data. Such behaviour puts serious restrictions on their usages.

4. Many algorithms do not consider interpretability as one of their goals. For example, SVM [Vap95] makes use of high dimension kernels for classification, which is a black box to users. In real life, users often want to know how these decisions are made. Having insight look into a model could help users understand what happens and, thus, take appropriate actions in advance.

Apparently, we need a new approach to solve these issues.

1.4 Contributions

We have two major contributions in this thesis. First, we propose a general framework which targets at solving the issues mentioned in the previous section. The framework is flexible and easy to incorporate domain-specific application goals. To address issues 3 and 4, the framework uses association rules as basic elements for decision-making. Rules are relatively easy to interpret and there are tons of algorithms in mining rules from various types of data sets. In addition, the framework reflects two important factors in building decision-making systems: pushing application goal down to the very beginning of system construction and optimizing the system on future data.

The second contribution is the studies of 5 challenging real-life applications. In these applications, we build effective and understandable decision-making systems by making use of the general framework. On the other hand, each application has its own unique challenges and requires specially designed algorithm to handle them. In the following, we summarize the contributions for each individual application:

- In CHAPTER 3 we study a classical decision-making problem: classification. We propose an algorithm that combines the advantages of association rules and decision trees. It leverages the hierarchical organization of decision trees to prune the richness of association rules and make the model easy to understand. We also propose an efficient approach to generate confident association rules that do not require a minimum support threshold. Such characteristic is very useful since in many cases it is difficult to determine an appropriate threshold.
- We study another challenging application in CHAPTER 4: direct marketing. Data set in this domain generally is very large, and has high dimension and skewed distribution. Most association rule mining algorithms do not scale up well in such situation. Making the thing even worse, most rules found (especially non-respondent rules) are redundant and not interesting to users. To address this issue, we propose an algorithm to mine *focused association rules* which are most likely to increase the profit return. Another challenge in direct marketing is the inverse correlation between the probability that a customer responds and the dollar amount generated by a response. Traditional algorithms no longer work in this case since they don't consider

such correlation. We present a solution to this problem based on a creative use of association rules.

- The number of online stores increases fast nowadays. So does the importance of online product recommendation systems. We discuss this interesting problem in CHAPTER 5. In a recommendation system, the target is to maximize the profit generated from the recommended products (thus we call the problem "profit mining"). Unlike the direct marketing problem, the recommendation here is two-dimension: recommending "right" items at "right" prices. It puts a unique challenge in model construction. Moreover, products naturally are categorized and related to each other. Using such domain knowledge for model construction will help improve the performance of the system. We propose profit patterns to address these issues. Such patterns consider both confidence (i.e. how likely a customer would like to buy the product) and profit (i.e. how much money the seller can get). Also it explores the product categorization systems and uses them to improve the recommendation. Experiments results show that the algorithm based on profit patterns has better performance than its competitors.

- CHAPTER 6 studies an interesting biological application: protein localization prediction. Given a protein, users like to know its position in a cell (e.g. at inner membrane or outer membrane) and the patterns determining its position. So it requires the decision model be both accurate and interpretable. Unlike previous applications where we build systems solely based on rules, this time we try a new approach: combining high-accuracy SVM model with easy-to-understand rule-based model. The idea is: The rule part captures the major patterns and presents them to users. And the SVM part does predictions for the minor patterns that cannot be captured by the rule part. We have two main challenges. First, the accuracy of the combined system should not be worse than that of applying SVM model alone. Second, the rule part should be small and really capture the major patterns. The algorithm discussed in that chapter addresses these issues.

- An interesting web application "catalog mapping" is discussed in CHAPTER 7. Given a catalog H_1 which is a hierarchical organization of categories, find the definitions for its categories (i.e. decide the meanings for its categories) in terms of the categories from another catalog H_2. The mappings are expressed in rules to make them understandable. Such catalog mapping is useful for many problems, like catalog integration. Unlike most existing algorithms that only study 1-to-1 correspondence, our algorithm finds complex mappings and, at the same time, makes few assumptions on existing catalogs.

1.5 Organization of the Thesis

We organize this thesis as follows. In CHAPTER 2 we present the general framework of building a decision-making system. From CHAPTER 3 to CHAPTER 7 we study five interesting applications: classification, direct marketing, profit mining, protein localization prediction and web catalog mapping. In each application, we discuss its unique challenges and the solution. The thesis concludes in CHAPTER 8. Some future directions are also discussed in this chapter.

CHAPTER 2
THE GENERAL FRAMEWORK

The construction of decision-making systems relies on many factors and users often have difficulties in applying algorithms built for domain *A* into domain *B*. For example, neural network might be good for statisticians but it would disappoint a financial institute director due to its difficulties to be interpreted. However, algorithms in various domains could share some common properties. Identifying such properties can help users develop algorithms in their own domains. In this chapter, we discuss these properties and propose a general framework based on them.

2.1 Common Properties

Due to various natures of applications, it is very difficult, if not impossible, to invent a universal algorithm that can be directly applied into different domains. On the other hand, we do believe that different applications share some common properties. Identifying these properties can help establish guidelines for building decision-making systems.

1. **Building interpretable systems.** Building an interpretable system is very much rewarding since it can greatly help users understand how decisions are made and thus make appropriate actions. Association rules are easy to understand and, thus, are good candidates as basic elements for decision-making systems. To make the system more interpretable, pruning it to a small size often is required. Moreover, pruning can also remove the over-fitting rules which are generated due to the noise in training data.

2. **Pushing the application goal down to the very begining of system construction**. Often we want to build a system that maximizes the application goal

(e.g. maximizing profit for direct marketing). To achieve that, we need to integrate the goal into the system construction. Many algorithms consider such knowledge only after the system is built. [Dom99] is such an example. It assumes that each misclassification is associated with a cost and tries to build a system that minimizes the overall cost on misclassifications. So it proposes a general method to make classifiers cost-sensitive. However, the cost of misclassification is only considered after the basic classifiers are built. Application goal (in this case, minimizing the cost) is not considered during the construction of basic classifier. As a consequence, there is no guarantee that the basic classifier is optimum. To maximize the application goal, we should push down the goal to the very beginning of system construction.

2.2 Why Association Rules?

When building decision-making systems, a fundmental issue a user faces is: How to choose the basic structure of the system? It could be a decision tree, a neural network or an SVM. In this thesis, we use association rules due to the following reasons. However, it does not imply that association rules could replace other algorithms because each of them has its own advantages. Actually in one application (protein localization) we integrate association rules with SVM to achieve optimal results.

- Compared to other competitors (like neural network or SVM), association rules are naturally easy to understand. Such property helps us build interpretable decision-making systems.

- Rule generation algorithms have been studied for a long time. The abundance of such algorithms makes it relatively easy to select (and modify if necessary) one to an application.

- Association rules are independent to each other. It helps improve the interpretability of the decision-making system since users need not to consider the interactions among rules. Moreover, it also makes the system pruning easier since pruning one rule won't affect other rules.

2.3 The General Framework

Figure 2-1 The general framework of building decision-making systems

```
Inputs:     Training data $D$; Application goal

Output:     Decision-making system $M$

Algorithm:

    1. Generate interesting association rules from $D$ with the consideration of application
       goal; try to keep the number of rules minimal;

    2. Build an initial model $M_I$ from those rules with the consideration of application goal;
       prompt the rules that benefit the system most;

    3. Prune $M_I$ with the consideration of application goal; remove the negative rules and
       optimize the system on future data;

    4. Return the final model $M$;
```

Figure 2-1 shows the general framework of using association rules to build a decision-making system. It has three major steps: generating interesting rules, building the initial model, and pruning the initial model. In the following sections, we will present details for each step. First, we have a brief introduction on the inputs of framework.

2.3.1 The Input

The input of the general framework is the training data *D* collected by users and the application goal. Each record *t* in *D* at least has two parts: the description of the record and the decision made based on the description. The exact data format varies across applications. For example, in supermarket applications the description of a record could be a set of items; while in financial domains the description could be a collection of attribute-value pairs. In many applications, the training data itself is pretty challenging due to its huge size, high dimension or skewed distribution. And there is no general cure for it. In this thesis, for each application we propose an innovative algorithm to handle these issues.

Besides the training data, many applications also have their unique application goals. Integrating them into system construction is critical for the success of a system.

2.3.2 Generating Interesting Rules

This step generates a set of rules which forms the foundation of a decision-making system. Each rule has the form of *X*→*C*, where *X* is a description and *C* is a decision. A rule is often associated with some interestingness measurements which reflect the likeliness of using it for decision-making. Consider a sample rule for document classification: *{process, thread, deadlock}*→*operating-system*, *conf=70%*. Here, *{process, thread, deadlock}* is the rule description, "*operating-system*" is the decision, and "*conf=70%*" is the interestingness measurement. This rule can be explained as: If a document contains the keywords *process*, *thread* and *deadlock*, we classify it into the topic *operating-system*, with *70%* confidence.

Often the rule generation space is huge for many applications. To save time and help build interpretable systems we must control the number of rules generated. A good way to do that is to set rule interestingness thresholds and only generate those rules with interestingness not less than the thresholds. For example, in document classification we can control the number of rules generated by setting the minimum support and minimum confidence. However, many times it is difficult for users to select an appropriate value for a threshold. For example, if the minimum support is too low, too many rules are generated. If it is too high, then many valuable rules could be left out. It would be good if users are not required to speciy such thresholds. We explore such possibility in several algorithms.

A big challenge in this step is to integrate application goal into rule mining. Doing that is critical for the success of a system. General rule mining algorithms (like Apriori [AS94) do not address this issue. We propose innovative approaches to address this issue in several applications.

2.3.3 Building the Initial Model M_I

With a set of rules generated in the previous step, now the problem is: How to organize these rules so they can work together as a decision-making system? Basically, we need to solve two issues:

- Given a case *t*, identify the rules that are eligible to make decisions on *t*
- If multiple such rules exist, select the most appropriate one(s)

For the first issue, we need to identify the rules which are "relevant" to the given case *t*. Intuitively we can define a rule *r* is relevant to case *t* if *r*'s description matches the description of *t*'s. And only those rules that are relevant to *t* are eligible to make decisions on *t*. The exact "match" definition is application dependent.

Example 2-1 Suppose the description of document *t* is: {*thread, process, deadlock*}. And we have two rules: r_1={*thread, deadlock*}→*OS*, r_2={*deadlock, table*}→*database*. We regard rule r_1 is relevant to document *t* since the description of r_1 matches (a subset in this case) *t*. But r_2 does not match *t*.

For the second question, it is equivalent to identify the rule(s) which can maximize user's application goal. If an interestingness measurement is defined based on user's goal, we could just simply select the rule(s) with the highest interestingness values. Based on the previous discussions, we have the following *MIF* principle for building the initial model.

Definition 2-1 (The MIF Principle). Given a case *t*, we only select relevant rule(s) to make decision. If there are choices, the rule(s) of higher *interestingness* has the priority. This is called the most-interesting-first principle, or simply *MIF principle.* □

The MIF principle turns a set of rules into a decision-making system which can be used to recommend decisions for new cases. However, it is just a general guideline and users need to materialize it when they actually apply it.

Given a testing case, the MIF principle allows to use multiple rules for prediction. However, it becomes a problem when those rules give different predictions. One solution to solve the conflict is to weight the predictions based on the priority of those rules. However, it is not easy to assign the weights appropriately. To avoid this problem, in this thesis we only use one rule (the rule with the highest rank) for prediction. We believe such rule has highest possibility (compared to other matched rules) to maximize user's application goal. Another advantage of this approachs is to avoid specifying the threshold for the number of rules to be used for prediction.

2.3.4 Pruning the Initial Model M_I

By organizing individual rules into a system in Step 2 we have new opportunities to optimize the system and prune rules due to the introduction of correlation among rules that does not exist in Step 1. That's why we have an extra step for refining the model. The actual optimizing/pruning strategy is application dependent. However, one important thing we should do in this step is to optimize the system on future data.

In the previous two steps we build a model based on *historical* (training) data. However, this model is intended to be used on *future* data. Typically the training data contains noise, which could pollute the patterns (rules) mined. Such patterns should be removed so they won't hurt the system performance on future data.

2.4 Summary

In this chapter, we introduce a general domain-independent framework which can be served as guidelines to build decision-making systems. Unlike most algorithms that only focus on performance, the framework emphasizes on both performance and interpretability. More importantly, it pushes the application goal down to the very beginning of system construction.

In the following chapters we will study several challenging applications using the framework as guidelines. Each application has its own unique challenges and needs innovative approaches to solve them. The framework helps and guides users to find such approaches.

CHAPTER 3
GROWING DECISION TREES ON ASSOCIATION RULES

This chapter studies a well-known decision-making problem: classification. Leveraging association rules for classification is an active research area in data mining. The richness of association rules gives this approach an edge over heuristically guided rule search. However, association rules suffer from two major issues. First, the minimum support requirement (i.e. requiring all the generated rules having support above certain threshold) for mining association rules often compromises the confidence requirement for classification rules. Second, association rules are not mutually exclusive, and an ad-hoc handling of rule interaction often diminishes the classification structure. To deal with the first problem, we abandon the support requirement and employ all association rules above a minimum confidence for classification. To deal with the second problem, we build an ADT (association based decision tree) from association rules and prune over-fitting rules on an accuracy-driven basis like the decision tree induction. By combining the richness of association rules with the accuracy-driven pruning of the classic decision tree induction, ADT outperforms other classifiers in both accuracy and classifier size. A paper based on this chapter was accepted by SIGKDD 2000 conference [WZH00].

3.1 Introduction

In this chapter we introduce an innovative approach for classification: association based decision tree (ADT). It takes the advantages from both association rules and decision trees by organizing association rules into a decision-tree-like structure.

In a decision tree, rules are organized into a generalization tree where the cases covered by a parent are covered exclusively by the child (specific) rules. Such exclusive coverage of the training cases enables a systematic, accuracy-based bottom-up pruning of over-fitting rules. However, decision tree evaluates one attribute at a time. Such heuristic-based local search could diminish the typical structure that several attributes collectively determine the class. On the other hand, the association mining searches globally for rules according to the joint predictiveness of several attributes disregarding the interaction of other rules. The richness of rules gives this approach the potential of finding the true classification structure in the data. Unfortunately, this strength becomes a weakness when pruning over-fitting rules because rules are inter-related by covering common cases and by a non-tree generalization hierarchy.

To prune over-fitting association rules, we organize rules into a generalization tree and leveraging the bottom-up pruning of decision tree. The result is called *ADT* (*association based decision tree*). Thus, the association rules used for building the classifier are selected from the full set of rules and collectively ensure the maximum accuracy with respect to the bottom-up pruning allowed. In other words, ADT has the advantage of combining the richness of association rules and the accuracy-driven pruning of decision tree induction. Our experiments on 21 benchmark data sets show that ADT leads other classifiers by an average of 2.4% in accuracy and by an average of 15% in classifier size.

Another major contribution in this chapter is the *confident rule* mining algorithm. Several algorithms have been proposed to use association rules for classification [AMS97, LHM98, MW99, WZL99]. Those approaches generate association rules by specifying minimum support and minimum confidence thresholds. Often, a good threshold is unknown in advance (if it exists at all) because different rules may have different requirements [WHH00]. In such cases, a viable pruning approach should be accuracy-driven, rather than threshold-driven. In ADT, we abandon the ad-hoc minimum support requirement and employ association rules satisfying only the minimum confidence, called *confident rules*, to build a classifier. To find all confident rules without exhaustive enumeration, we propose a confidence-based pruning by exploiting a certain monotonicity of confidence so that general rules are examined only if some specific rules are confident.

This chapter is organized as follows. In Section 3.2 we introduce the related work. From Section 3.3 to 3.5 we present the algorithm of building ADT by following the guidelines in the general framework. We evaluate the effectiveness of our approach in Section 3.6. Section 3.7 concludes this chapter.

3.2 Related Work

Classification has been a subject of research in machine learning, statistics, pattern recognition, neural networks and other areas for several decades. Association rule mining was first studied in [AIS93], and subsequently in a number of papers, e.g., [AS94, Bay98, HF95, PCY95, SA95, SON95]. These two problems are largely studied in isolation until association rules are used for classification recently. We focus on the work on classification based on association rules.

Association rules are used for partial classification in [AMS97] and for covering training cases in [Bay97]. Both works are not concerned with building a classifier. [LHM98] uses association rules to build a classifier and prunes specific rules using both the minimum support and the pessimistic estimation. In particular, [LHM98] uses confidence and support to compute the observed error of rules. This computation ignores the interaction among rules by repeatedly covering a training case using all matching rules, thus, fails to model the reality that each case is classified by one rule. [DZWL99, MW99] combines several association rules to classify a new case. Thus, this approach partially addresses the "low support" issue of classification rules because a combined rule could have a lower support. The rule pruning in [DZWL99, MW99] is essentially threshold-based. In [WZL99], multi-level association rules are used to build hierarchical classifiers where both the class space and the feature space are organized into taxonomy or an *isa*-hierarchy. All the above approaches crucially rely on a minimum support to prune specific rules. We abandon the support requirement and deal with over-fitting by an accuracy-driven pruning. Finding association rules between two values without a support requirement is studied in [CDF+00]. But such rules are too general for classification. Several researchers, e.g., [Sch93, MP94], have tried to build classifiers by extensive search. None of them, however, uses association-mining techniques.

3.3 Mining Confident Rules

For classification application often the goal is to maximize the prediction accuracy. Hence, rules with high confidences are preferred. As pointed out in the framework, we should consider this domain constraint when mining association rules. In this section, we propose an algorithm to mine confident rules. Rules with higher confidences are more preferred. The algorithm does not require users to select the minimum support, which is often difficult to determine in many cases.

We assume that the database is represented by a relational table T over m non-class attributes $A_1, \ldots, A_m$ and one class attribute C. A case in the database has the form $<a_1, \ldots, a_m, c>$, where a_i are values of A_i and c is a class of C. A rule, or a k-rule, has the form $A_{i1}=a_{i1} \wedge \ldots \wedge A_{ik}=a_{ik} \rightarrow C=c$, with each attribute occurring at most once. By prefixing a value with its attribute, we can omit attributes and write a rule as $a_{i1}, \ldots, a_{ik} \rightarrow c$, or simply $x \rightarrow c$, where x denotes one or more values. We say a case t and a rule $x \rightarrow c$ match if t contains all the values in x. A rule $x, a_i \rightarrow c$ is a *A_i-specialization* of $x \rightarrow c$ if a_i is a value of A_i. $|T|$ denotes the number of cases in T, and $num(x)$ denotes the number of cases in T that contain all the values in x. The *support* of rule $x \rightarrow c$, denoted $sup(x \rightarrow c)$, is $num(x, c)/|T|$, where $num(x, c)$ denotes the number of cases in T that contain both x and c. The confidence of rule $x \rightarrow c$, denoted $conf(x \rightarrow c)$, is $num(x, c)/num(x)$.Given a minimum confidence *minconf*, a rule is *confident* if $conf(x \rightarrow c) \geq minconf$.

Definition 3-1 (Mining confident rules) The problem of *mining confident rules* is to find all confident rules for a given minimum confidence. □

Since mining confident rules does not require a minimum support, the support-based pruning in [AIS93] is not applicable. What we need is a confidence-based pruning that pushes the confidence requirement to prune unpromising rules as early as possible. This change turns out to be drastic. On the one hand, confidence no longer enjoys the downward closure as enjoyed by support-based pruning [AIS93]: *Age=young→Buy=yes* and *Gender=M→Buy=yes* could have lower confidence than *Age=young, Gender=M→Buy=yes*. Consequently, even though shorter rules are not confident, longer rules still need to be examined. On the other hand, confidence does not enjoy the upward closure either because *Age=old, Gender=F→Buy=yes* could have lower confidence than *Age=old→Buy=yes* or *Gender=F→Buy=yes*. Thus, a straightforward pruning based on the downward or upward closure does not work.

To motivate the confidence-based pruning, consider the following rules:

r1: *Age=young→Buy=yes*

r2: *Age=young, Gender=M→Buy=yes*

r3: *Age=young, Gender=F→Buy=yes*

Suppose that *r1* has confidence of 0.60, that is, 60% of young people buy the Internet service. We can infer that some of *r2* and *r3* has at least confidence of 0.60. The key observation is that, since the two conditions *Gender=M* and *Gender=F* are mutually exclusive, if one condition impacts confidence negatively, the other condition must impact confidence positively. We can exploit this property to prune *r1* if none of *r2* and *r3* is confident. We call this property the existential upward closure because it assures that some specialized rule of a given confident rule must be confident.

Theorem 3-1 (The existential upward closure) Consider any attribute A_i not occurring in rule $x \rightarrow c$. (i) Some A_i-specialization of $x \rightarrow c$ has at least the confidence of $conf(x \rightarrow c)$. (ii) If $x \rightarrow c$ is confident, so is some A_i-specialization of $x \rightarrow c$. □

Proof: (ii) follows immediately from (i). So we prove (i) only. Suppose that for every value a_i of A_i, $conf(x, a_i \rightarrow c) < conf(x \rightarrow c)$. That is, $num(x, a_i, c)/num(x, a_i) < conf(x \rightarrow c)$, or $num(x, a_i, c) < num(x, a_i) * conf(x \rightarrow c)$, where $num(x, a_i, c)$ denotes the number of cases in T that contain x, a_i *and* c. Summing up over all a_i on both sides, we have

$$\Sigma a_i num(x, a_i, c) < \Sigma a_i num(x, a_i) * conf(x \rightarrow c), \text{ or,}$$

$$\Sigma a_i num(x, a_i, c) / \Sigma a_i num(x, a_i) < conf(x \rightarrow c).$$

Since each training case contains at most one value of A_i, $\Sigma a_i \ num(x, a_i, c) = num(x, c)$ and $\Sigma a_i \ num(x, a_i) = num(x)$. Therefore, $num(x, c)/num(x) < conf(x \rightarrow c)$. But this contradicts the definition of confidence. (i) is proved. □

The existential upward closure suggests the following level-wise rule generation. Assume that all confident k-rules have been generated (starting with $k=m$, the number of non-class attributes). We generate a candidate (k-1)-rule $x \rightarrow c$ only if for every attribute A_i not occurring in $x \rightarrow c$, some A_i-specialization of $x \rightarrow c$ is confident. We can implement this candidate generation in relational algebra, thus, in SQL supported by any database system. In particular, let R_k be the set of confident k-rules and let $R_k(X, C)$ be the set of rules in R_k with attributes X on the left-hand side. Similarly, let C_k and $C_k(X, C)$ be the corresponding notations for generated candidate rules. We represent rules $x \rightarrow c$ as tuples $<x, c>$ and represent $R_k(X, C)$ and $C_k(X, C)$ as relational tables over attributes X and C. Theorem 3-1 (ii) can be restated as: A (k-1)-rule $<x, c>$ is in $C_{k-1}(X, C)$, where x is a vector of values over X, only if $<x, c>$ is in the projection $\pi_{X,C} R_k(X, A_i, C)$ for every non-class attribute A_i not occurring in X. This gives rise to the following computation of $C_{k-1}(X,C)$.

Corollary 3-1 Let $C_{k-1}(X,C) = \bigcap_{A_i} \pi_{X,C} R_k(X,A_i,C)$, where $\pi_{X,C} R_k(X,A_i,C)$ denotes the projection onto the attributes X and C, and A_i ranges over all non-class attributes not in X. Then $C_k(X, C) \supseteq R_k(X, C)$. □

The corollary above implies two results. First, it suffices to examine only confident k-rules, not other rules, in order to generate a superset of confident (k-1)-rules; thus, examining this superset is sufficient to find all confident rules. Second, this generation can be implemented by a relational expression over the set of confident k-rules, i.e., R_k, therefore, can be immediately implemented on top of a commercial database system. The actual confident rules in $C_{k-1}(X,C)$ can be found by computing the confidence of candidates in C_{k-1} in one pass of the database T.

Table 3-1 The sample database

A_1	A_2	A_3	C
7	0	0	1
7	0	1	1
5	2	0	0
5	0	1	0
5	2	1	1

Example 3-1 Consider a sample database in Table 3-1. Suppose *minconf*=80%. We see that:

$R_2(A_1, A_2, C)$={<7,0,1>,<5,0,0>}

$R_2(A_1, A_3, C)$={<7,0,1>,<7,1,1>,<5,0,0>}

$R_2(A_2, A_3, C)$={<0,0,1>,<2,0,0>,<2,1,1>}

C_1 can be computed as follows:

$$C_1(A_1, C) = \pi_{A_1,C} R_2(A_1, A_2, C) \bigcap \pi_{A_1,C} R_2(A_1, A_3, C) = \{<7,1>,<5,0>\}$$

$$C_1(A_2, C) = \pi_{A_2,C} R_2(A_1, A_2, C) \bigcap \pi_{A_2,C} R_2(A_2, A_3, C) = \{<0,1>\}$$

$$C_1(A_3, C) = \pi_{A_3,C} R_2(A_1, A_3, C) \bigcap \pi_{A_3,C} R_2(A_3, A_3, C) = \{<0,1>,<1,1>,<0,0>\}$$

By finding the confidence of candidates in C_1, we have $R_1(A_1,C)$={<7,1>}, $R_1(A_2,C)=R_1(A_3,C)=\varnothing$.

□

Table 3-2 The overview of mining confident rules

```
Input: Table $T$ over $A_1, ..., A_m$, $C$ and $minconf$;
Output: All confident rules;
Algorithm:
k=m;
$C_k(A_1, ..., A_m, C)=T$;
while $k \geq 1$ and $C_k$ is not empty do
    Compute the confidence of candidates in $C_k$ in one pass of $T$;
    $R_k$=all confident candidates in $C_k$;
    Generate $C_{k-1}$ from $R_k$ (based on Corollary 3-1);
    k--;
Return all $R_k$;
```

The table above gives the algorithm for finding confident rules. The seed C_{m} is initialised to the set of training cases in the database T (line 2). In iteration k, we compute the confidence of candidates in C_k in one pass of T (line 4), collect confident k-rules (line 5) from the candidates, and generate C_{k-1} (line 6). To compute the confidence of candidates in C_k, we borrow the computation of itemset support in [AS94]: First, candidates $x \rightarrow c$ in C_k are stored in the hash-tree as they are generated. Then, we scan the cases in T, and for each case t, we update *num*(x) and *num*(x, c) for all the matching candidates x→c such that $x \sqsubseteq t$ using the subset function of the hash-tree, exactly as in [AS94]. In particular, if t has class c, we increment both *num*(x) and *num*(x, c); otherwise, we increment *num*(x). At the end of the database scan, the confidence of a candidate $x \rightarrow c$ is *num*(x, c)/ *num*(x).

3.4 Building the Initial Classifier

Let C denote the set of confident rules plus the default rule, denoted $\varnothing \rightarrow c$, where c is the majority class in data T. Note that every case matches the default rule. In classification the goal is to achieve high accuracy. Hence, we prefer the rules with high confidence for prediction. The following definition ranks rules based on their confidences.

Definition 3-2 (The ranking of confident rules). Consider two rules r and r'. We say that r is ranked higher than r', written as $r \prec_R r'$, if the following condition holds (in that order):

- *conf*(r)>*conf*(r'),
- or if *conf*(r)=*conf*(r'), but *sup*(r)>*sup*(r'),
- or if *sup*(r)=*sup*(r'), but *size*(r)<*size*(r'),
- or if *size*(r)=*size*(r'), but r precedes r' in the lexicographical order of rules.

where *conf*(r) denotes the confidence of rule r, *sup*(r) denotes the support of rule r, and *size*(r) denotes the number of attributes in the left-hand-side of rule r. □

In words, $\prec_R$ ranks rules by confidence, followed by support, followed by size, followed by the lexicographical order of rules, in that order. The following principle governs the preference of rules, which favours the most predictive rule among all applicable ones. It is an application of MIF principle (in the general framework) where the interestingness is measured by confidence.

Definition 3-3 (The MCF principle) If there are choices, the rule of the highest rank has the priority. This is called the *most-confident-first principle*, or simple *MCF principle.* □

The MCF principle serves two roles in the construction of a classifier. First, it turns the set of rules into a classifier by covering a case using the rule that matches the case and has the highest possible rank. This rule is called the *covering rule* of the case. The term *initial classifier* refers to this classifier. The second role of the MCF principle is resolving the interaction among rules, by partitioning the covered cases according to covering rules. Each training case belongs to the partition corresponding to its covering rule. The significance of this partitioning is that if we compute the "observed error" of a rule using the training cases covered by the rule, we do not have the problem of repeatedly considering a training case for several rules. This is clearly a more truthful error estimation of the prediction model where each new case is predicted using one rule. With the observed error, we can then estimate the "predictive error" of a classifier and determine the over-fitting rules to be pruned.

Example 3-2 Consider the training set *T* in Table 3-3 (left). Suppose we find 8 confident rules as shown in Table 3-4. r_9 is the default rule. (Ignore the last three columns in Table 3-4 at this time.)

Table 3-3 The sample data sets for a small example

TID	A_1	A_2	A_3	*C*
t1	0	0	1	0
t2	0	1	2	0
t3	0	2	0	1
t4	1	0	1	1
t5	2	0	0	0
t6	1	1	0	1
t7	2	0	0	1
t8	0	2	1	0
t9	2	0	2	1
t10	2	2	0	0

The training set

TID	A_1	A_2	A_3	*C*
t11	0	2	0	0
t12	1	1	1	1
t13	0	1	1	0
t14	1	2	1	1

The testing set

Consider *t11* in the testing set. The matching rules of *t11* are r_4, r_8 and r_9. r_4 is the covering rule of *t11* because it is ranked higher than the other two rules. Thus, *t11* is classified into class 1 incorrectly. Similarly, the covering rule of *t12, t13, t14* is r_2, r_1, r_2, respectively, all classified correctly. ◫

Table 3-4 The initial classifier

Rule ID	Rule	Conf.	Sup.	Covered Training Case	N (#case covered)	E (#case covered wrongly)
r_1	$A_1=0, A_3=1 \rightarrow 0$	1.00	0.20	*t1, t8*	2	0
r_2	$A_1=1 \rightarrow 1$	1.00	0.20	*t4, t6*	2	0
r_3	$A_1=0, A_2=1 \rightarrow 0$	1.00	0.10	*t2*	1	0
r_4	$A_1=0, A_3=0 \rightarrow 1$	1.00	0.10	*t3*	1	0
r_6	$A_1=2, A_2=2 \rightarrow 0$	1.00	0.10	*t10*	1	0
r_7	$A_1=2, A_3=2 \rightarrow 1$	1.00	0.10	*t9*	1	0
r_8	$A_1=0 \rightarrow 0$	0.75	0.30	*none*	0	0
r_9	$\varnothing \rightarrow 0$	0.50	1.00	*t5, t7*	2	1

3.5 Pruning the Classifier

In this section we introduce the algorithm ADT to prune the classifier. As pointed out in the general framework, the purpose of pruning is to optimize the system on future data and improve the interpretability by reducing its size. First, we remove the redundant rules.

3.5.1 Removing Redundant Rules

Let *lhs*(*r*) denote the left-hand-side of rule *r*. we say that rule *r* is more general than rule *r'*, or *r'* is more special than *r*, written as r p$_G$ r', if *lhs*(*r*) ⊆*lhs* (*r'*). We observe that, under the MCF principle, rules that are more specific but do not offer higher confidence will never be used. Such rules can be removed without affecting classification.

Definition 3-4 (Redundant rules) A rule r is *redundant* if there is some rule r' that is more general and ranked higher than r, that is, $r' \prec_G r$ and $r' \prec_R r$. □

A redundant rule r will have no turn to cover any case because some general rule r' matches whatever cases r matches and has a higher rank than r. From now on, we assume that redundant rules are removed from the classifier.

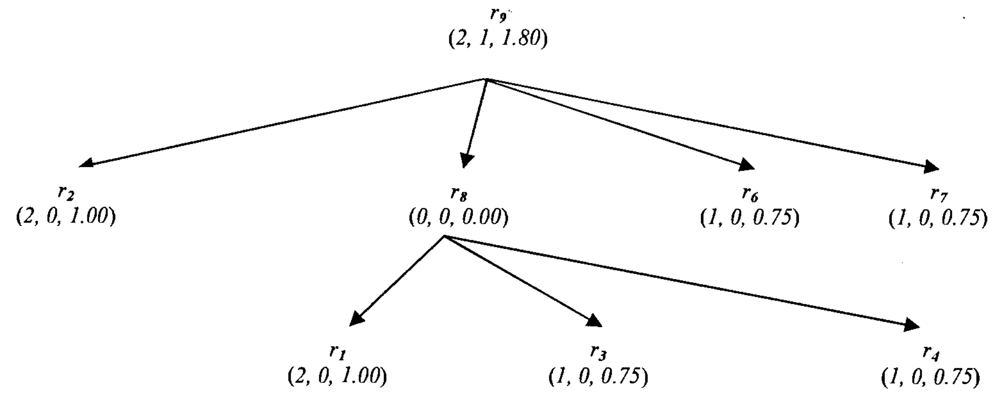
3.5.2 Building ADT

Now we consider pruning non-redundant rules. The idea is to use the error observed in the training cases to estimate the error on new cases, and to prune rules if it helps to reduce the estimated error. We borrow the error estimation of decision tree [Qui93]. In decision tree, the training cases covered by a parent rule (general rule) are refined by child rules (special rules) using the values of the splitting attribute chosen at the parent. If child rules are over-fitting, measured by an increase of the estimated error, it is natural to "cancel" such refinement by pruning the child rules. However, the decision tree method is not directly applicable to our initial classifier. This is because unlike decision tree, $\prec_G$ is a lattice of rules, rather than a tree of rules, where an association rule may have several general rules. If a special rule is pruned, more than one general rule can be used to cover the cases covered by the special rule. Therefore, to adopt the decision tree pruning, we need to convert the lattice into a tree by resolving the conflict of parenthood. The significance of having a tree structure is that a parent rule will "act on behalf of" its child rules if the latter are pruned.

Consider a rule r. If r is pruned, we need to choose a general rule to cover the cases originally covered by r. Again, the MCF principle helps us to make this choice: The highest ranked general rule, say r', should be chosen. In this sense, r' acts on behalf of r in case that r is pruned. This immediately converts the lattice of association rules into a tree of such "acts on behalf of" relationships between rules.

Definition 3-5 (ADT) Consider a non-default rule *r*. The parent of *r* is the rule that is more general than *r* and has the highest possible rank. The ADT (association based decision tree) for a rule set S_R contains a node for each rule in S_R and contains an edge from a non-default rule to its parent. □

Figure 3-1 The unpruned ADT



The root of ADT is the default rule, which is the only rule having no parent. Figure 3-1 shows the ADT for the initial classifier (ignore the tuple associated with each node at this time) in Table 3-4. In the following, the terms "node" and "rule" are interchangeable.

Corollary 3-2. Consider a child rule *r* and its parent *r'* in the ADT. (i) $r \prec_R r'$ (ii) There is no rule *r''* other than *r* and *r'* such that $r' \mathrm{p}_G\, r'' \mathrm{p}_G\, r$.

Proof: Notice that $r' \prec_G r$. If $r \prec_R r'$ does not hold, $r' \prec_R r$ must hold. Then $r' \prec_G r$ implies that *r* is redundant, contradicting that all redundant rules are removed. This proves (i). To see (ii), suppose that there is a rule *r''* such that $r' \mathrm{p}_G\, r'' \mathrm{p}_G\, r$. Since *r'* has the highest rank among all general rules of *r*, $r' \prec_R r''$. This implies that *r''* is redundant, a contradiction again. □

From (i), rules at lower levels (i.e., children) have higher rank than those at higher levels (i.e., parents). Thus, the MCF principle will select the lowest matching rule to be the covering rule of a case. From (ii), the parent is the most conservative generalization of a child because no general rule comes in between. Such a dense generalization order helps avoid over-pruning "good" general rules.

3.5.3 Pruning ADT

The key step of pruning ADT is to estimate the error on new cases. We adopt the *pessimistic estimation* for pruning decision tree [Qui93]. The idea is to regard misclassifying the training cases covered by a rule r as the binomial distribution. This distribution is then used to estimate the error on the whole population covered by r. In particular, if $N(r)$ training cases are covered by r, $E(r)$ of them incorrectly, this is regarded as observing $E(r)$ events in $N(r)$ trials in the binomial distribution. For a given confidence level CF, the upper limit on the probability of error over the entire population is written $U_{CF}(E(r), N(r))$. Then the upper limit is (pessimistically) taken as the estimated error rate of r on the whole population. The exact computation of $U_{CF}(E(r), N(r))$ is less important and can be found in the C4.5 code [Qui93]. And a theoretical account can be found in [CP34]. The idea is that a smaller sample size N is penalized by a larger upper limit $U_{CF}(E(r), N(r))$ to guarantee the specified confidence level CF. The default value of CF in C4.5 is 25%.

Thus, if r is used to classify a set of new cases of the same size as the training cases covered by r, the estimated error of r is $N(r)*U_{CF}(E(r), N(r))$, and we have the CF% confidence that the actual error is within this upper limit.

The pruning of ADT proceeds in the bottom-up order by considering all child nodes before considering a parent node. Suppose that we are currently considering node v. Let *tree*(v) denote the subtree rooted by v. If v is a non-leaf node, we check whether pruning *tree*(v) can reduce the predicted error of the classifier. Here, pruning *tree*(v) means making v a leaf node, which will cover all the cases covered by the rules within *tree*(v). If pruning is worthwhile, i.e., reducing the predicted error, we perform the pruning immediately; otherwise, we do nothing at v. With such pruning we can both optimize the tree on future data (by reducing the overall estimated error) and improve the interpretability (by reducing the number of rules).

Let *Tree_Err*(v) and *Leaf_Err*(v) denote the predicted error before and after pruning the subtree rooted by v, respectively. These errors are computed by

$$Leaf\_Err(v) = N' * U_{CF}(E', N')$$

$$Tree\_Err(v) = \Sigma_u[\, N(u) * U_{CF}(E(u), N(u))] \qquad \textbf{(3-1)}$$

where u ranges over all nodes within *tree*(v). N' is the number of training cases covered by v as a leaf node, which is equal to the total number of training cases covered by the rules within *tree*(v). E' is the number of cases incorrectly covered by v as a leaf node.

If $Leaf\_Err(v) \leq Tree\_Err(v)$, we prune *tree*($v$) into leaf node v and update $E(v)$ and $N(v)$ to E' and N'. If $Leaf\_Err(v) > Tree\_Err(v)$, nothing is done at node v. The above consideration is repeated until the root of ADT is considered.

The final step is to remove any remaining rule that covers many cases incorrectly. The *merit* of a rule r is defined by

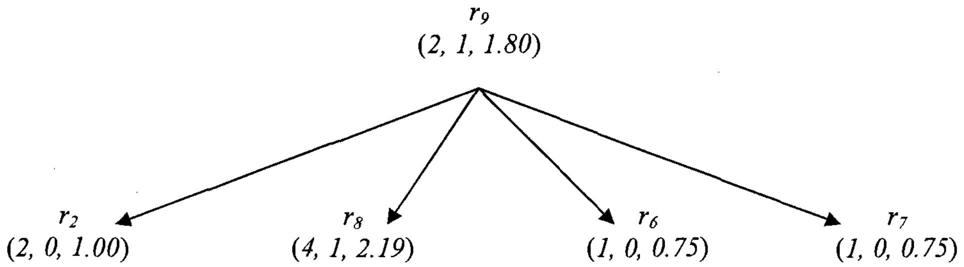
$$merit(r) = (N(r) - E(r))/N(r) \qquad \text{if } N(r)>0$$

$$merit(r) = 0 \qquad \text{if } N(r)=0 \qquad \textbf{(3-2)}$$

Thus, a small merit means many cases are covered incorrectly. If the merit of a rule r is below a user-specified threshold, denoted by *minmeri*, we can prune r from ADT. Unlike confidence, the notion of merit is based on the non-repeated covering of training cases. A rule may have a high confidence but a small merit because many cases contributing to the high confidence could be covered by other rules. Pruning such rules often improves the accuracy of classifier.

Example 3-3 Consider pruning the ADT in Figure 3-1. Suppose that we like to have 25% confidence that the actual error rate does not exceed the estimated error rate. Then CF=25%. See Table 3-4 for the training cases covered by each rule. Associated with each node v in Figure 3-1 is a tuple ($N(v)$, $E(v)$, $Err(v)$). $Err(v)= N(v)*U_{CF}(E(v), N(v))$ denotes the estimated error of v.

Figure 3-2 The pruned ADT



First, we consider non-leaf node r_8. The estimated error of $tree(r_8)$ is the total error of r_1, r_3, r_4, and r_8. So $Tree\_Err(r_8)$=0.75*2+1.00=2.50. The estimated error of r_8 as a leaf node, $Leaf\_Err(r_8)$, is computed by assuming that it covers all cases covered by $tree(r_8)$. From Table 3-4, these cases are *t1*, *t2*, *t3*, *t8*, of which three are correctly classified into class 0, and one incorrectly into class 1. So, N'=4, E'=1, and $Leaf\_Err(r_8)=N'*U_{CF}(E', N')$=2.19. Since $Leaf\_Err(r_8)<Tree\_Err(r_8)$, $tree(r_8)$ is pruned into leaf node r_8.

Next, we consider non-leaf node r_9. $Tree\_Err(r_9)$=1+0.75+0.75+2.19+1.80=6.49 and $Leaf\_Err(r_9)=N'*U_{CF}(E', N')$=6.54, where E'=5 and N'=10. Therefore, there is no pruning at r_9. The final ADT is shown in Figure 3-2.

Table 3-5 shows the result of classifying the testing cases in Table 3-1 using both unpruned and pruned ADT. The pruning improves the accuracy from 75% to 100%. In the unpruned ADT, the error is caused by r_4 on the case *t11*. From the training set we can see that the classification structure mainly consists of r_2 and r_8. r_4, generated by the noise case *t3*, is an overly specialized rule of r_8. Without the pruning, r_4 takes over r_8 to cover *t11*, thereby, producing an error. □

Table 3-5 Classification on testing cases

	Pruned ADT		**Unpruned ADT**	
Testing case	**Covering Rule**	**Error**	**Covering Rule**	**Error**
t11	r_8	0	r_4	1
t12	r_2	0	r_2	0
t13	r_8	0	r_1	0
t14	r_2	0	r_2	0

It is helpful to compare ADT with decision tree. Decision tree is both a rule generator and a pruning method. At each step, decision tree selects the attribute that best splits the classes in the current partition of data, evaluated by the information gain. Such a one-attribute-at-a-time top-down data splitting induces a tree structure where a root-to-leaf path corresponds to a rule. Consequently, the rules in decision tree follow a tree-structured sharing of features. In comparison, ADT is built from rules produced elsewhere and its purpose is to prune over-fitting rules. Though we have considered mainly association rules, ADT can be built using rules produced by any rule generator. This decoupling of rule generating from rule pruning eliminates the unnatural sharing of features in decision tree, and combines the richness of externally generated rules with the systematic pruning of decision tree induction.

In particular, by leveraging association rules we are able to evaluate rules several-attribute-at-a-time, rather than one-attribute-at-a-time. This change is highly desirable for capturing co-occurred features. Unlike decision tree that explicitly represents rules themselves, ADT represents the "acting" relationship of rules and hides rules within nodes. Thus, ADT does not impose an actual structure on rules.

3.6 Experiments

We evaluate ADT performance on 21 datasets from UCI Repository [MM96] as shown in Table 3-6. The columns "#Tuple", "#Attribute" and "#Class" denote the number of tuples, the number of attributes, and the number of different classes for each data set.

We compare the performance of ADT with five other methods, i.e., C4.5 [Qui93], NB [DH73], TAN [FGG97], CBA [LHM98] and LB [MW99]. C4.5 is frequently used as the benchmark in the classification paradigm. NB is a Naïve Bayes classifier which shows reasonable accuracy in many cases. TAN is an extension of NB and outperforms many Bayesian Network classifiers. CBA is a classification based on association rules, like ours. LB is a hybrid of NB and association rule approach, by extending NB from itemsets of length 1 to length *k*.

For all methods, the parameters are set to their default values as suggested in the literature. For example, CBA uses the minimum support of 0.5% plus the pruning option; TAN uses the smoothing factor of 5, etc. For ADT, *minconf* is 50% and *minmeri* is 10%.

Table 3-6 21 Data sets used in experiments

Data set	#Tuple	#Attribute	#Class
Australia	690	14	2
Balance	625	4	3
Bridges	108	13	6
Car	1625	6	4
CRX	691	15	2
Diabetes	768	8	2
Flare	323	10	3
Glass	214	10	7
Iris	150	4	3
Monks-1	432	6	2
Monks-2	432	6	2
Monks-3	432	6	2
New-Thyr	215	5	3
Nursery	12415	8	5
Page-Blo	5473	10	5
Post-Ope	90	8	3
Shuttle-S	15	6	2
Tic-Tac-Toe	958	9	2
Voting	435	16	2
Wine	178	13	3
Zoo	101	17	7

Our study focuses on the accuracy and the size of classifiers obtained as the average of the 5-fold cross validation. If a data set is already pre-partitioned into the training set and testing set, we combine them before applying the 5-fold cross validation. Since all methods, except C4.5, deal with discrete attributes, continuous attributes are discretized using entropy discretization as implemented in the MLC++ system [KJL+94].

Table 3-7 Classification accuracy

Data set	C4.5-con	C4.5-dis	TAN	NB	LB	CBA	ADT	Avg.
Australia	0.857	0.865	0.845	0.859	0.867*	0.849	0.855	0.857
Balance	0.560	0.560	0.640	0.778	0.778	0.683	0.797*	0.685
Bridges	0.657	0.670		0.632	0.632	0.671	0.690*	0.565
Car	0.917	0.917	0.943*	0.856	0.886	0.938	0.921●	0.911
CRX	0.857*	0.854		0.851	0.857	0.842	0.852●	0.730
Diabetes	0.737	0.737	0.740	0.750*	0.736	0.729	0.739●	0.738
Flare	0.823	0.826	0.826	0.804	0.823	0.801	0.830*	0.819
Glass	0.681	0.690	0.681	0.690	0.690	0.710	0.714*	0.694
Iris	0.933	0.940*	0.920	0.927	0.927	0.920	0.920	0.927
Monks-1	0.978	0.978	1.000*	0.746	1.000*	1.000*	1.000*	0.958
Monks-2	0.616	0.616	0.622	0.627	0.627	0.763*	0.730●	0.657
Monks-3	0.989*	0.989*	0.987	0.964	0.965	0.971	0.989*	0.979
New-Thyr	0.916	0.935	0.940	0.944	0.944	0.944	0.953*	0.939
Nursery	0.965	0.965	0.935	0.903	0.946	0.981	0.983*	0.954
Page-Blo	0.969*	0.966	0.954	0.932	0.961	0.954	0.952	0.955
Post-Ope	0.689	0.711	0.689	0.667	0.667	0.533	0.712*	0.667
Shuttle-S	0.997	0.996	0.998*	0.992	0.997	0.997	0.997●	0.996
Tic-Tac-Toe	0.841	0.841	0.743	0.702	0.689	0.991*	0.974●	0.826
Voting	0.966*	0.966*		0.905	0.931	0.940	0.949●	0.808
Wine	0.897	0.874	0.989*	0.989*	0.989*	0.920	0.931	0.941
Zoo	0.950*	0.950*	0.940	0.950*	0.950*	0,940	0.940	0.946
Average	0.847	0.850	0.855	0.832	0.851	0.861	**0.877***	0.853

3.6.1 Accuracy

Table 3-7 shows the average accuracy on 5-fold cross-validation of seven different classifiers. The standard deviations of accuracy across different folds are small so we do not show them here. "C4.5-con" stands for C4.5 without attribute discretization, and "C4.5-dis" stands for C4.5 with attribute discretization. The blanks in the column for TAN indicate that TAN is not applicable to those data sets with missing values. For each data set (indicated by a row), the most accurate classifier(s) is marked with '*'. The last row is the average accuracy of a classifier over all data sets, and the last column is the average accuracy of each data set over all classifiers.

We can see that no classifier is uniformly superior across all data sets. However, ADT performs better in several ways. First, ADT scores the highest average accuracy, i.e., 0.877 in bold face. This is 3.0% and 2.7% higher than that of "C4.5-dis" and "C4.5-con", and is 1.6% higher than the second best, CBA. Second, ADT scores the most number of '*', i.e., the highest accuracy. Third, shown in the last two columns, for the 13 data sets for which ADT does not score '*', ADT is above the average for 8 of them, marked with '•', and is close to the average for the other 5.

In Table 3-8, the first row shows the average win of ADT for the data sets for which ADT is the best, and the second row shows the average loss of ADT to the best classifier for the data sets for which ADT is not the best. The comparison of the two rows shows that the win is always more substantial than the loss.

Table 3-8 Win vs. Loss

Loser **ADT's win**	C4.5-con 0.055	C4.5-dis 0.046	TAN 0.138	NB 0.037	LB 0.028	CBA 0.051
Winner **ADT's loss**	C4.5-con 0.009	C4.5-dis 0.020	TAN 0.008	NB 0.034	LB 0.012	CBA 0.026

Table 3-9 Size of classifiers

Data set	C4.5-con	C4.5-dis	CBA	ADT	Avg.
Australia	34.4	23.2*	129.2	43.8	57.6
Balance	35.0	35.0	117.2	22.2*	52.3
Bridges	30.4*	30.4*	34.8	48.8	36.0
Car	164.2	164.2	126.8*	194.4	162.4
CRX	29.2	20.8*	127.2	49.2	56.6
Diabetes	41.0	16.0*	38.8	16.0*	27.9
Flare	2.6	5.2	28.8	1.0*	9.4
Glass	42.2	27.0	25.8*	34.6	32.4
Iris	7.8	4.0*	5.6	4.6	5.5
Monks-1	44.4	44.4	22.0	15.4*	31.5
Monks-2	21.8*	21.8*	117.8	111.8	68.3
Monks-3	19.0	19.0	16.6	4.2	14.7
New-Thyr	11.4	12.0	11.4	11.2*	11.5
Nursery	467.4	467.4	411.6	269.8*	404.0
Page-Blo	75.4*	128.8	202.0	146.2	138.1
Post-Ope	1.6	1.0*	23.4	1.0*	6.7
Shuttle-S	27.8*	41.0	55.4	27.8*	38.0
Tic-Tac-Toe	119.8	119.8	26.8*	26.8*	73.3
Voting	10.2*	10.2*	31.8	25.2	19.3
Wine	10.6	12.8	10.2*	15.0	12.1
Zoo	17.8	17.8	8.6*	15.2	14.8
Average	57.8	58.1	74.8	**51.6**	60.6

Table 3-10 Size of ADTs at different stages

Data set	Confident Rules	Unpruned ADT	Pruned ADT	Final Classifier
Australia	$2.27*10^{6}$	$9.67*10^{3}$	$3.04*10^{3}$	$4.38*10^{1}$
Balance	$1.15*10^{3}$	$3.98*10^{2}$	$3.28*10^{1}$	$2.22*10^{1}$
Bridges	$9.80*10^{4}$	$7.64*10^{2}$	$6.72*10^{2}$	$4.88*10^{1}$
Car	$7.62*10^{3}$	$4.71*10^{2}$	$3.99*10^{2}$	$1.94*10^{2}$
CRX	$4.67*10^{6}$	$1.19*10^{4}$	$4.30*10^{3}$	$4.92*10^{1}$
Diabetes	$2.67*10^{3}$	$1.85*10^{2}$	$4.46*10^{1}$	$1.60*10^{1}$
Flare	$9.02*10^{3}$	$1.75*10^{2}$	$1.00*10^{0}$	$1.00*10^{0}$
Glass	$2.23*10^{3}$	$2.71*10^{2}$	$2.22*10^{2}$	$3.46*10^{1}$
Iris	$1.04*10^{2}$	$2.46*10^{1}$	$1.94*10^{1}$	$4.60*10^{0}$
Monks-1	$2.93*10^{3}$	$3.97*10^{2}$	$2.76*10^{2}$	$1.54*10^{1}$
Monks-2	$3.14*10^{3}$	$4.76*10^{2}$	$2.12*10^{2}$	$1.12*10^{2}$
Monks-3	$2.88*10^{3}$	$2.94*10^{2}$	$1.52*10^{2}$	$4.20*10^{0}$
New-Thyr	$4.53*10^{2}$	$6.22*10^{1}$	$4.70*10^{1}$	$1.12*10^{1}$
Nursery	$1.06*10^{5}$	$9.03*10^{3}$	$6.75*10^{3}$	$2.70*10^{2}$
Page-Blo	$3.28*10^{5}$	$5.89*10^{3}$	$3.05*10^{3}$	$1.46*10^{2}$
Post-Ope	$6.73*10^{3}$	$3.21*10^{2}$	$1.00*10^{0}$	$1.00*10^{0}$
Shuttle-S	$2.93*10^{4}$	$1.05*10^{3}$	$8.08*10^{2}$	$2.78*10^{1}$
Tic-Tac-Toe	$1.06*10^{5}$	$5.07*10^{3}$	$3.94*10^{3}$	$2.68*10^{1}$
Voting	$4.60*10^{6}$	$6.11*10^{3}$	$4.32*10^{3}$	$2.52*10^{1}$
Wine	$4.29*10^{5}$	$1.51*10^{3}$	$1.43*10^{3}$	$1.50*10^{1}$
Zoo	$1.81*10^{6}$	$1.55*10^{3}$	$1.55*10^{3}$	$1.52*10^{1}$

3.6.2 Size

Table 3-9 shows the size of classifiers in the number of rules. The size information is not available for LB, NB, and TAN. ADT produces the smallest classifier for many data sets. Interestingly, there seems to be a strong correlation between the datasets on which ADT produces the most accurate classifier and the datasets on which ADT produces the smallest classifier. Table 3-10 shows the size of ADT at each stage of the construction. “Confident Rules” stands for the total number of confident rules generated; “Unpruned ADT” represents the number of rules in the initial classifier with redundant rules removed; “Pruned ADT” is the size of the pruned ADT before applying *minmeri*; “Final ADT” shows the size of the final classifier. The comparison of these stages shows that every stage of pruning is effective in reducing the classifier size.

3.7 Conclusion

In this chapter we propose a novel algorithm for the classification problem by integrating two techniques together: association rule and decision tree induction. Association rules are rich, but lacking of a systematic method of pruning over-fitting rules for classification. Decision tree induction, on the other hand, has an accuracy-driven pruning, but imposes restrictive structures on rules. The comparison motivates our work of combining the two approaches for building better classifiers. We propose a method to build decision trees from association rules, i.e., ADT. The advantage of ADT is the preservation of the strength of both approaches, i.e., the richness of rules and the systematic accuracy-based pruning. To give ADT the full pruning power, we use all confident association rules without a support requirement. A confidence-based mining is proposed for finding all such rules. Experiments have shown that the proposed ADT not only builds more accurate classifiers, but also does this by finding more truthful structures, as indicated by the smaller size of classifiers.

CHAPTER 4
MINING CUSTOMER VALUE: FROM ASSOCIATION RULES TO DIRECT MARKETING

Direct marketing is a modern business activity with an aim to maximize the profit generated from marketing to a selected group of customers. A key to direct marketing is to select a right subset of customers so as to maximize the profit return while minimizing the cost. Achieving this goal is difficult due to the *extremely imbalanced data and the inverse correlation between the probability that a customer responds and the dollar amount generated by a response.* Traditional probability based approaches cannot solve this problem. In this chapter, we present a solution based on a creative use of association rules. A chief advantage is the completeness of association rule search and the focus on promising customers by pushing the recorded customer value. A paper based on the algorithm proposed in this chapter was accepted by DMKD international journal [WZYY05].

4.1 Motivation

Direct marketing makes it possible to offer goods or services or transmit messages to a specific, targeted segment of the population by mail, telephone, email or other direct means. However, building decision-making systems for direct marketing is a challenging task due to the following reasons.

1. The high dimensionality and the scare target population present a significant challenge for extracting the features of the "respond" class. For example, the KDD-CUP-98 dataset [KDD98-data] used in our experiment contains 191,779 records about individuals contacted in the 1997 mailing campaign. Each record is

described by 479 non-target variables and two target variables indicating the "respond" or "not-respond" classes and the actual donation in dollars. About 5% of records are "respond" records and the rest are "not-respond" records. Since any subset of variables can be a feature for distinguishing the "respond" class from "not-respond" class, searching for such features is similar to searching for a needle from a haystack The "one attribute at a time" gain criterion [Qui93] does not search for correlated variables as features.

2. Quoted from [KDD98-result], "there is often an inverse correlation between the likelihood to respond and the dollar amount of the gift". It means that there are many "small customers" making small purchases and few "big customers" making big purchases. A pure probability based ranking tends to favour "small customers" because of higher likelihood to respond, and ignore "big customers". Previous researches addressed this issue in two steps: obtain the probability estimation from a standard classification model such as decision tree [LL98, MS96], bagging [Dom99] and smoothing [ZE01], and re-rank the probability based ranking by taking into account the customer value [MS96, ZE01]. The disadvantage of this approach is that the customer value is ignored in the first step.

To address challenge 1, we propose the notion of *focused association rules* to focus on the features that are typical of the "respond" class and not typical of the "not-respond" class. A focused association rule makes use of only items that have higher frequency and correlation in the "respond" class. The search space is determined by "respond" records and items that occur infrequently in the "not-respond" records. This prunes all "not-respond" records (to deal with the scarcity of the target class) and all items that occur frequently in the "not-respond" class (to deal with the high dimensionality).

In the presence of Challenge 2, innovative solutions are needed because statistically insignificant rules could generate a significant profit. Our approach is to push the customer value into the model building/pruning so that the estimated profit over the whole population is maximized.

4.2 Introduction

In direct marketing, typically, what available is a *historical* database containing information about previous mailing campaigns, including whether a customer responded and the dollar amount collected if responded. The task is to build a model to predict *current* customers who are likely to respond. The goal is to maximize the sum of net profit, Σ(*dollar amount - mailing cost*), over the contacted customers. We choose the KDD-CUP-98 dataset [KDD98-data] as the case study. This dataset was collected from the result of the 1997 Paralysed Veterans of America fundraising mailing campaign and only 5% of records are responders. Thus, simply classifying all customers into non-responders would give 95% accuracy, but this does not generate profit.

In this chapter, we propose a novel approach to address the above issues. First, we exploit association rules [AIS93, AS94} of the form $X \rightarrow$*respond* to extract features for responders, where X is a set of items that is correlated with the "respond" class. We select a small subset of association rules to identify potential customers in the current campaign. We address two key issues, namely, push the customer value in selecting association rules, and maximize profitability over the current customers (instead of historical ones). On the challenging KDD-CUP-98 task, which has 5% responders and 95% non-responders, this method generates 41% more profit than the winner of the competition and 35% more profit than the best known result after the competition, and the average profit per mail is 3.3 times that of the winner. This method identifies correctly 57.7% of responders and 78% of non-responders, thus, also provides a competitive solution to the cost sensitive classification.

4.2.1 Task Definition

Historical records are stored in a relational table of m non-target variables $A_1, \ldots, A_m$ and two target variables *Class* and *V*. *Class* takes one of the “respond” and “not-respond” classes as the value. *V* represents a continuous donation amount. Given a set of records of this format, our task is to build a model for predicting the donation profit over current customers represented by the validation set in the KDD-CUP-98 dataset. Precisely, we want to maximize $\Sigma_u(V_u\text{-}\$0.68)$, where u ranges over the current customers who are predicted to have a donation greater than the mailing cost \$0.68. An implicit assumption is that current customers follow the same class and donation distribution as that of historical records. Since the donation amount V for a current customer is not known until the customer responds, the algorithm is evaluated using a holdout subset from the historical data, i.e., the validation set.

In the following sections we first examine the related work. Next, we present our algorithms by following the steps in the general framework. Section 4.7 gives the experiment results on KDD-CUP-98 dataset. In Section 0 we conclude the chapter.

4.3 Related Work

In direct marketing, a principled method is ranking customers by the estimated probability to respond and selecting some top portion of the ranked list [LL98, MS96]. For example, if the top 5% of the ranked list contains 30% of all responders, the lift model gives the lift of 30/5=6. A significant drawback of this approach is that the actual customer value, e.g., the donation amount in the example of fundraising, is ignored in the ranking, or it requires a uniform customer value for all customers. As pointed out in [KDD98-result] for the KDD-CUP-98 task, there is an inverse correlation between the likelihood to buy (or donate) and the dollar amount to spend (or donate). This inverse correlation reflects the general trend that the more dollar amount is involved, the more cautious the buyer (or donor) is in making a purchase (or donation) decision. As a result, a probability based ranking tends to rank down, rather than rank up, the valuable customers.

The realization that a cost-sensitive treatment is required in applications like direct marketing has led to a substantial amount of research. [Dom99] proposed the MetaCost framework for adapting accuracy-based classification to cost-sensitive learning by incorporating a *cost matrix* $C(i, j)$ for misclassifying true class j into class i. [ZE01] examined a more general case where the benefit $B(i, j, x)$ depends not only on the classes involved but also on the individual customers x. For a given customer x, the "optimal prediction" is the class i that leads to the minimum expected cost [Dom99]

$$\sum_j P(j \mid x) C(i, j)$$

or the maximum expected benefit [ZE01]

$$\sum_j P(j \mid x) B(i, j, x)$$

Both methods require to estimate the conditional class probability $P(j|x)$. In this phase, since only the frequency information about x, not the customer value of x, is examined, valuable customers, who tend to be infrequent because of the "inverse correlation", are likely to be ignored. The customer value is factored only at the end via the factor $B(i, j, x)$.

The motivation of association rules in the market basket analysis has led to several attempts to extend and apply such rules in business environments. [SON98] considers negative association rules that tell what items a customer will not likely buy given that he/she buys a certain set of other items. [TKS00] considers indirect association rules where the association of two items is conditioned on the presence of some set of other items. Such associations are purely count or occurrence based and have no direct relationships with the "inverse correlation" considered here that addresses profit. We focus on *using* association rules based on customer value, whereas these works focus on *finding* association rules based on count information. This distinction is substantial because association rules themselves do not tell how to maximize an objective function, especially in the presence of the "inverse correlation". Our work differs from the product recommendation in [WZH02] and item selection in [BSVW99, WS02] in that we identify valuable customers instead of items or products.

In the rest of this chapter, the following terms are interchangeable: customer and record, responder and "respond" record, non-responder and "not-respond" record.

4.4 Generating FARs (Focused Association Rules)

Obviously, general association rule mining algorithms (like Apriori) won't work well due to the high dimension and huge data size. In addition, mining confident rules (discussed in CHAPTER 3) no longer works here since the application goal is not accuracy. Actually we need to find the rules/patterns that have the potential to bring high profit. In this section we introduce focused association rules to solve these issues.

As a necessary data pre-processing step, we discretize continuous non-target variables using the MLC discretization utility[1] before generating any rules. After discretization, each value a_{i_j} is either a categorical value or an interval. We are interested in "respond" rules of the form

$$A_{i_1} = a_{i_1}, \ldots, A_{i_k} = a_{i_k} \rightarrow respond$$

that are potentially useful for discriminating responders from non-responders. Despite many efficient algorithms for mining association rules (see [AIS93, AMS+96, AS94], for example), we encountered a significant difficulty in this step: To find "respond" rules we have to set the

[1] http://www.sgi.com/tech/mlc

minimum support well below 5%, i.e., the percentage of "respond" records in the dataset; however, with 481 variables and 95% records in the "not-respond" class, the number of "not-respond" rules satisfying the minimum support is so large that finding "respond" rules is similar to searching a needle from a haystack. Sampling techniques cannot reduce the "width" of records that is the real curse behind the long running time. We consider a simple but efficient solution to this problem by focusing on items that occur *frequently* in "*respond*" records but occur *infrequently* in "*not-respond*" records.

Let D_r be the set of "respond" records and let D_n be the set of "not-respond" records. We have the following definition:

Definition 4-1 (Focused association rules) The *support* of item $A_i=a_i$ in D_r or D_n is the percentage of the records in D_r or D_n that contain $A_i=a_i$. The *support* of a rule in D_r or D_n is the percentage of the records in D_r or D_n that contain all the items in the rule. Given a minimum support for D_r and a maximum support for D_n, an item $A_i=a_i$ is *focused* if its support in D_n is not more than the maximum support and its support in D_r is not less than the minimum support. A "respond" rule is a *focused association rule* (*FAR*) if it contains only focused items and its support in D_r is not less than the minimum support. □

In words, a FAR occurs frequently in D_r (as per the minimum support) but none of its items occurs frequently in D_n (as per the maximum support). Notice that FARs exclude the "respond" rules that as a whole do not occur frequently in D_n but some of its items does. This "incompleteness" trades for the data reduction achieved by pruning all non-focused items. For the KDD-CUP-98 dataset, this prunes all "not-respond" records, which accounts for 95% of the dataset, and all items that occur frequently in D_n, which accounts for 40%-60% of all items. Our experiments show that the notion of FARs works exactly towards this goal.

Table 4-1 Algorithm of generating focused association rules (FARs)

```
Input:      $D_r$, $D_n$, the minimum support for $D_r$ and the maximum support for $D_n$

Output:     FARs

Algorithm:

/* Compute the support in $D_n$ for items in $D_r$ */
for all tuple $t$ in $D_r$ do
    for all item in $t$ do
        Create a counter for the item if not yet created;
    end for
end for
for all tuple $t$ in $D_n$ do
    for all item in $t$ do
        Increment the counter for the item if found;
    end for
end for
/* Remove the items from $D_r$ whose support in $D_n$ exceeds the maximum support */
for all tuple $t$ in $D_r$ do
    Remove the items from $t$ whose support in $D_n$ exceeds the maximum support;
end for
Find "respond" rules above the minimum support in $D_r$ such as in [AIS93];
```

Algorithm shown in Table 4-1 finds FARs for given minimum support in D_r and maximum support in D_n. First, it computes the support in D_n for the items in D_r (line 1-11) and removes those items from D_r for which this support exceeds the maximum support (line 12-15). Then, it applies any association rule mining algorithm such as [AIS93] to the updated D_r to find "respond" rules above the minimum support (line 16). This association rule mining is expensive, but is applied to only "respond" records and only items whose support in D_n is not more than the maximum support. After finding the FARs, we add to the rule set the (only) "not-respond" rule of the form

$$\emptyset \rightarrow \textit{not-respond}$$

This rule, called the *default rule*, is used only if a customer matches no FAR.

Table 4-2 The sample database

D_r				
TID	***A***	***B***	***C***	***V***
p1	a1	b1	c3	$30.68
p2	a1	b2	c3	$50.68
p3	a1	b2	c1	$40.68
p4	a2	b2	c2	$20.68
p5	a2	b1	c3	$20.68
D_n				
TID	***A***	***B***	***C***	***V***
n1	a1	b1	c1	$0.00
n2	a2	b1	c3	$0.00
n3	a2	b2	c1	$0.00
n4	a2	b1	c3	$0.00
n5	a3	b2	c1	$0.00

Table 4-3 The D_r after applying the maximum support

D_r				
TID	***A***	***B***	***C***	***V***
p1	a1	/	c3	\$30.68
p2	a1	b2	c3	\$50.68
p3	a1	b2	/	\$40.68
p4	/	b2	c2	\$20.68
p5	/	/	c3	\$20.68

Table 4-4 Count of items

Item	Count in D_n	Count in D_r
a1	1	3
a2*	3	2
b1*	3	2
b2	2	3
c1*	3	1
c2	0	1
c3	2	3

Example 4-1 Consider a small sample database in Table 4-2. There are 10 records, 5 in D_r and 5 in D_n. Each record has 3 attributes *A, B, C* and donation *V*. Suppose that both minimum support for D_r and maximum support for D_n are 40%.

Table 4-4 shows the support count for each item in D_r. The items exceeding the maximum support in D_n (i.e., occur in more than 2 records in D_n) are marked with "*".

Table 4-3 shows the D_r with such items removed. Table 4-5 shows the FARs found from D_r, plus the default rule. □

Table 4-5 The FARs generated with minimum support and maximum support of 40%

RID	Rule	Support in D_r
r1	*∅→not-respond*	5/5=100%
r2	*A=a1→respond*	3/5=60%
r3	*B=b2→respond*	3/5=60%
r4	*C=c3→respond*	3/5=60%
r5	*A=a1, B=b2→respond*	2/5=40%
r6	*A=a1, C=c3→respond*	2/5=40%

In the rest of this chapter, a "rule" refers to either a FAR or the default rule; $sup(r)$ denotes the support of rule r in $D_r \cup D_n$, i.e., the percentage of all records containing both sides of the rule, $lhs(r)$ denotes the set of items on the left-hand side of rule r, $|\ lhs(r)|$ denotes the number of items in $lhs(r)$. We say that a rule r *matches* a record t, or vice versa, if t contains all the items in $lhs(r)$. We say that a rule r is more *general* than a rule r' if $lhs(r) \subseteq lhs(r')$.

4.5 Building the Initial Model

In direct marketing, to maximize the profit generated, we prefer the rule that matches the customer and has the largest observed profit on the learning set. We should consider this domain knowledge when constructing the system. Let $profit(r, t)$ denote the profit generated by the prediction of r on a learning record t. The observed profit of r is defined as:

$$O\_avg(r) = \sum_t profit(r,t)/P \qquad \textbf{(4-1)}$$

where t is a learning record that matches r and P is the number of such records. A large $O\_avg(r)$ means that the customers (in the learning set) matched by r make a large donation on average.

$profit(r, t)$ can be calculated as follow:

$$profit(r,t) = \begin{cases} V\text{-}0.68 & \text{if } t \text{ (as a respondent) is classified as a respondent} \\ -0.68 & \text{if } t \text{ (as a non-respondent) is classified as a respondent} \\ 0 & \text{otherwise} \end{cases}$$

To maximize the profit on a current customer, we prefer the matching rule of the largest possible *O_avg*. We give the total rule ranking definition below.

Definition 4-2 (Ranking rules) Consider two rules r and r'. We say that r is ranked higher than r', written as $r \prec_R r'$, if the following condition holds (in that order):

- (Average profit) $O\_avg\,(r) > O\_avg\,(r')$, or
- (Generality) if $O\_avg\,(r)=O\_avg\,(r')$, but $sup(r)>sup(r')$, or
- (Simplicity) if $sup(r)=sup(r')$, but $|lhs(r)|<|lhs(r')|$, or
- if $|lhs(r)|=|lhs(r')|$, but r precedes r' in the lexicographical order of rules.

Given a record t, a rule r is the covering rule of t, or r covers t, if r matches t and has the highest possible rank. □

Similar to the ADT algorithm, we remove the redundant rules before we continue on the next step.

Example 4-2 Continue with Example 4-1. Rules are ranked by *O_avg* in Table 4-6. For example, $r2$ matches 4 records $p1$, $p2$, $p3$ and $n1$. $O\_avg(r2)=\Sigma_t profit(r2,\ t)/4$=(\$30+\$50+\$40-

$0.68)/4=$29.83. *O_avg* for other rules is similarly computed. *p2* is matched by all 6 rules and is covered by *r5*, the matching rule of highest rank. Similarly, the covering rules of other records can be determined. □

Table 4-6 Coverage and rank of rules

RID	Records matched	Records covered	*O_avg*	Ranking
r5	*p2, p3*	*p2, p3*	$45.00	1st
r6	*p1, p2*	*p1*	$40.00	2nd
r2	*p1, p2, p3, n1*	*n1*	$29.83	3rd
r3	*p2, p3, p4, n3, n5*	*p4, n3, n5*	$21.73	4th
r4	*p1, p2, p5, n2, n4*	*p5, n2, n4*	$19.73	5th
r1	*p1-p5, n1-n5*	∅	$0.00	6th

4.6 Pruning the Model

The above rule ranking criterion favours specific rules that match a small number of customers of high profit. In the classic classification problem, such rules are pruned due to statistical insignificance. In the presence of inverse correlation between the likelihood to respond and the dollar amount generated by a response, extra care should be taken because valuable customers do not show up very often and pruning their rules could lead to the loss of significant profit. To address this issue, we propose pruning rules on the basis of increasing the estimated profit over the whole population. Below, we describe this new pruning method.

First, we explain how to estimate the profit of a rule *r* over the whole population; then, we give a method for pruning rules based on this estimation. The profit of *r* (over the whole population) can be estimated in two steps. First, we estimate the "hits" of r over the whole population. Second, we compute the profit of the estimated hits using the observations in the learning set. Similar to what we did for ADT, we borrow the pessimistic estimation [CP34, Qui93] for estimating the "hits" of r.

Let *Cover*(r) denote the set of learning records covered by a respond rule r. Let $N(r)$ denote the number of records in *Cover*(r), $E(r)$ of which do not match the class in r. $E(r)/N(r)$ is the observed error rate of r on the learning sample. Given a confidence level *CF*, we estimate the "hits" is $N(r)*(1-U_{CF}(E(r), N(r)))$, and the number of "misses" is $N(r)*U_{CF}(E(r), N(r))$. The average profit per hit in *Cover*(r) is

$$avg_h(r) = \sum_t (V - 0.68)/(N - E) \qquad \textbf{(4-2)}$$

where t is a "respond" record in *Cover*(r), V is the donation amount in t. The average profit per miss in *Cover*(r) is the cost of mailing to a non-responder, i.e., 0.68. We extend these averages to the above estimated hits and misses.

Definition 4-3 (Estimated profit) Assume that r covers N learning records, E incorrectly. The *estimated profit* of r is

$$Estimated(r) = \begin{cases} N*(1-U_{CF}(E(r), N(r)))*avg_h(r) - N*U_{CF}(E(r), N(r))*0.68 & \text{if } r \text{ is a respond rule} \\ 0 & \text{if } r \text{ is the default rule} \end{cases}$$

The *estimated average profit* of r, denoted $E\_avg(r)$, is $Estimate(r)/N$. The *estimated profit* of a model is $\Sigma_r Estimated(r)$ over all rules r (for $|D_r|+|D_n|$ customers randomly chosen from the whole population). □

Notice the difference between *O_avg*(r) and *E_avg*(r). *O_avg*(r) is the average profit *observed* for the learning records that are *matched* by r. The matching rule of the largest *O_avg*(r) is the covering rule of a given record. *E_avg*(r) is the average profit *estimated* for the records in the whole population that are *covered* by r. We use *E_avg*(r) to estimate the profit generated by each prediction of r over the whole population. *E_avg*(r) depends on *O_avg*(r) to define the notion of covering rules.

To prune over-fitting rules to maximize Σ_r*Estimated*(r), we organize rules into a decision tree, like what we did in CHAPTER 3 for ADT. However, different rule pruning criterion is required. This time we use estimated profit. Let *Estimated*(r) denote the estimated profit for a node r, *Estimated_tree*(r) denote the estimated profit for the subtree rooted at r, and *Estimated_leaf*(r) denote the estimated profit after pruning the tree at r. *Estimated_tree*(r) is Σ_u*Estimated*(u) over all nodes u within the subtree at r. If *Estimated_tree*(r)≤*Estimated_leaf*(r), it prunes the subtree at r by making r a new leaf node in the covering tree and removing the rules in the subtree. If *Estimated_tree*(r)>*Estimated_leaf*(r), it does nothing at r. The nodes outside the subtree at r are not considered because their estimated profit remains unchanged. Essentially, the bottom-up pruning has the effect of cutting off some lower portion of the tree to maximize Σ_r*Estimated*(r) over remaining rules r.

Example 4-3 In this example we show how to prune the tree using the estimated profit criterion. First we build the tree for Example 4-2. Consider rule *r5* for example. *r1, r2* and *r3* are more general than *r5*, but *r2* has the highest rank among them. So, *r2* is the parent of *r5*. In this way, we build the tree on the left of Figure 4-1.

Table 4-7 shows *Estimated*(r) before and after the pruning at r. For example, *r5* covers correctly *p2* and *p3*, so N=2 and E=0. The estimated number of misses is $2*U_{CF}(0,2)=2*0.50=1.00$, and the estimated number of hits is $2*(1-U_{CF}(0,2))=1.00$.

$avg_h(r5)$=[(50.68-0.68)+(40.68-0.68)]/(2-0)=\$45.00. From Definition 4-3, *Estimated*($r5$)=1.00* $avg_h(r5)$-1.00*0.68=\$44.32.

After examining nodes *r5* and *r6*, the bottom-up pruning examines the node *r2*. *Estimated_tree*(*r2*)=*Estimated*(*r2*)+*Estimated*(*r5*)+*Estimated*(*r6*)=-0.68+44.32+6.99=\$50.63. Pruning the subtree at *r2* makes *r2* cover *p1, p2, p3* and *n1*, and *N*=4 and *E*=1. In this case, the estimated number of misses is 4*U_{CF}(1,4)=4*0.55=2.20, the estimated number of hits is 4*(1-U_{CF}(1,4))=4*0.45=1.80, and $avg_h(r2)$=[(50.68-0.68)+(40.68-0.68)+(30.68-0.68)]/(4-1)=\$40.00. Following Definition 4-3,

Estimated_leaf(*r2*)=1.80*40.00-0.68*2.20=\$70.50.

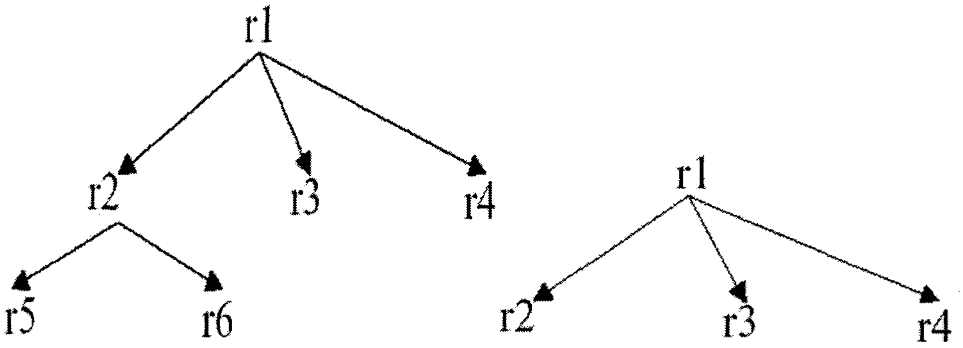
Since *Estimated_tree*(*r2*)≤*Estimated_leaf*(*r2*), the subtree at *r2* is pruned.

Using the same approach we examine nodes *r2, r3, r4*, and *r1*.The final pruned tree is shown on the right of Figure 4-1. □

Table 4-7 ***Estimated(r)*** **before and after pruning**

RID	**Before pruning**			**After pruning**		
	Cover(r)	***(E, N)***	***Estimated(r)***	***Cover(r)***	***(E, N)***	***Estimated(r)***
r5	*p2, p3*	(0, 2)	\$44.32	*N/A*	*N/A*	*N/A*
r6	*p1*	(0, 1)	\$6.99	*N/A*	*N/A*	*N/A*
r2	*n1*	(1, 1)	-\$0.68	*p1, p2, p3, n1*	(1, 4)	\$70.50
r3	*p4, n3, n5*	(2, 3)	\$2.10	*p4, n3, n5*	(2, 3)	\$2.10
r4	*p5, n2, n4*	(2, 3)	\$2.10	*p5, n2, n4*	(2, 3)	\$2.10
r1	∅	(0, 0)	\$0.00	∅	(0, 0)	\$0.00

Figure 4-1 Left: before pruning Right: after pruning



4.7 Validation

In this section, we validate the proposed method using the standard split of the KDD98-learning-set (95,412 records, 4,843 "responders") and KDD98-validation-set (96,367 records, 4,873 "responders") used by the KDD competition [KDD98-data]. The KDD98-learning-set is used for learning a model. In our method, we split the KDD98-learning-set randomly into 70% for the building set (66,788 records, 3,390 "respond" records) and 30% for the testing set (28,624 records, 1,453 "respond" records). The testing set is used for tuning the minimum and maximum support in our method, not for evaluation purpose. The evaluation is performed using the standard KDD98-validation-set, which is held out from the learning phase of all algorithms. The competition criterion is the *sum of actual profit* on the KDD98-validation-set, defined as $\sum_t(V-0.68)$ for all validation records *t* predicted to have a positive profit, where *V* is the donation amount in *t*.

We compare our method with three categories of published results. The first includes the top five results from the KDD-CUP-98 competition. As pointed out by [KDD98-result], these contestants used state-of-the-arts techniques such as 2-stage, multiple strategies, combined boosting and bagging. The second category includes the results produced by the MetaCost technique [Dom99]. The third category includes the results produced by the direct cost-sensitive decision-making [ZE01]. The results from the latter two categories are taken from [ZE01], which implemented MetaCost and direct cost-sensitive decision-making using advanced techniques for probability estimation and donation estimation, including multiple linear regression, C4.5, Naïve Bayes classifier, smoothing, curtailment, binning, averaging, and Heckman procedure. Interested readers are referred to [ZE01] for more details.

Figure 4-2 The distribution of donation

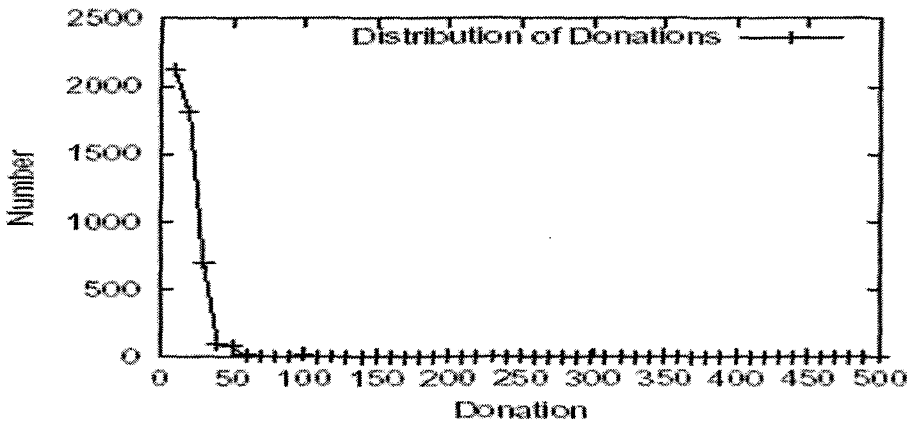


Figure 4-2 shows the distribution of donation amount for "respond" records in validation set. There is a clear inverse correlation between the probability that a customer responds and the dollar amount generated by a response.

4.7.1 Sum of Actual Profit

The summary of comparison is shown in Table 4-8 based on the KDD98-validation-set. The first row (in bold face) is our result. Next come the three categories of published results: the top five contestants of the KDD-CUP-98 as reported in [KDD98-result], five algorithms of MetaCost and five algorithms of direct cost-sensitive decision-making as reported in [ZE01].

Table 4-8 Comparison with published results

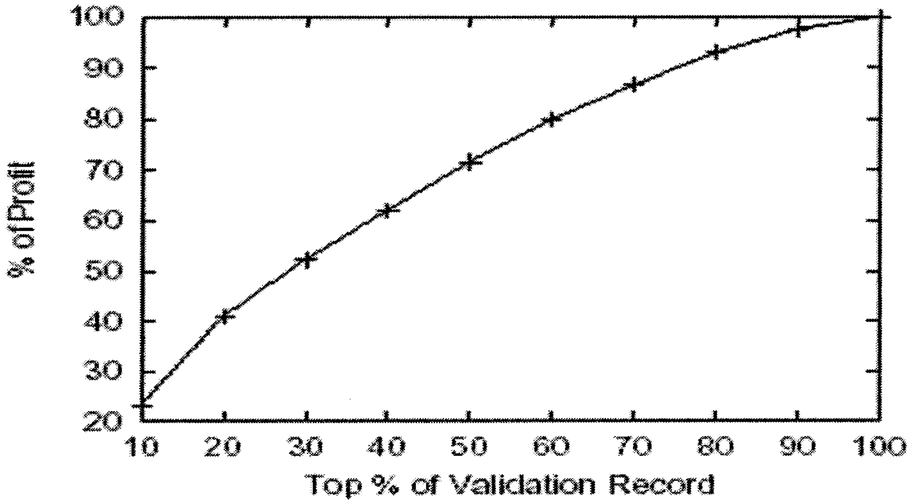
Category	**Algorithm**	**Sum of actual profit**	**#Mailed**	**Average profit**
	Our algorithm	**$20,693**	**23,437**	**$0.88**
KDD-CUP-98 results	GainSmarts (the winner)	$14,712.24	56,330	$0.26
	SAS (#2)	$14,662.43	55,838	$0.26
	Quadstone (#3)	$13,954.47	57,836	$0.24
	ARIAI (#4)	$13,824.77	55,650	$0.25
	Amdocs (#5)	$13,794.24	51,906	$0.27
MetaCost	Smoothed C4.5 (sm)	$12,835	N/A	N/A
	C4.5 with curtailment (cur)	$11,283	N/A	N/A
	Binned naïve Bayes (bin)	$14,113	N/A	N/A
	Average(sm, cur)	$13,284	N/A	N/A
	Average(sm, cur, bin)	$13,515	N/A	N/A
Direct Cost-Sensitive	Smoothed C4.5 (sm)	$14,321	N/A	N/A
	C4.5 with curtailment (cur)	$14,161	N/A	N/A
	Binned naïve Bayes (bin)	$15,094	N/A	N/A
	Average(sm, cur)	$14,879	N/A	N/A
	Average(sm, cur, bin)	$15,329	N/A	N/A
	Max possible profit	$72,776	4,873	$14.93
	Mail to everyone	$10,548	96,367	$0.11

Our method generated the sum of actual profit of $20,693. This is 41% more than the KDD-CUP-98 winner ($14,712.24), 47% more than the best profit of MetaCost ($14,113), and 35% more than the best profit of direct cost-sensitive decision-making ($15,329). According to the analysis in [ZE01], a minimum difference of $1,090 is required to be statistically significant. Our performance gain far exceeds this requirement. Our average profit per mail is $0.88. This is 3.38 times that of the KDD-CUP-98 winner, and 8 times that of the Mail to Everyone Solution. Compared to the KDD-CUP-98 winner, we generated 41% more profit by predicting less than a half number of contacts. [ZE01] did not report the number of mailed, so we cannot compute their average profit. These higher total profit and average profit suggest that the proposed method is highly successful in focusing on valuable customers. This success is credited to the novel feature extraction based on the global search of association rule mining, and the profit estimation that pushes the customer value as the first class information.

4.7.2 Profit Lift

We extend the concept of "lift" in the literature [LL98, MS96] to evaluate the "profit lift" of our result. In the *cumulative lift curve* [LL98, MS96], validation records are ranked by the estimated probability of belonging to the "respond" class, and each point (x, y) on the curve represents that the top x percent of the ranked list contains y percent of all actual responders. In the cumulative profit lift curve, each point (x, y) represents that the top x percent of the ranked list generates y percent of the total profit. Thus, the cumulative lift curve is a special case of the cumulative profit lift curve when every responder generates the same profit. Figure 4-3 shows the cumulative profit lift curve of our result. For example, the top 20% of the ranked list generates 42% of the total actual profit, giving the profit lift of 2.1. The bend toward the upper-left corner suggests that our method ranks valuable customers toward the top of the list.

Figure 4-3 The accumulative profit lift curve



4.7.3 Classification

Table 4-9 shows the confusion matrix for the KDD98-validation-set. 2,813 of the 4,873 responders are predicted as responders (i.e., contacted), and 71,389 of the 91,494 non-responders are predicted as non-responders (i.e., not contacted), giving the "hit rate" of 57.7% on responders and 78.0% on non-responders. In other words, the hit rate for responders is more than 10 times the percentage of responders in the data (i.e., 5%). This strongly suggests that our method has achieved the goal of identifying valuable customers.

Table 4-9 The confusion matrix

	Not contacted	**Contacted**
Non-responder	71,389	20,105
Responder	2,060	2,813

4.8 Conclusion

In this chapter we study the direct marketing problem which becomes increasingly important in retail, banking, and insurance industries. One challenge in direct marketing is the inverse correlation between the likelihood to buy and the dollar amount to spend, which implies that the traditional probability based ranking will rank valuable customers low rather than high. Another challenge is the extremely high dimensionality and extremely low proportion of the target class. In such cases, finding rules to distinguish the target class from non-target classes is similar to finding a needle from a haystack.

To solve the first challenge, we push the customer value as the first class information. Our approach is to estimate directly the profit generated on a customer without estimating the class probability. For the second challenge, we only mine "focused rules" on respondents only. It reduces the rule searching space by discarding the items that are not so "unique" to respondents. The evaluation on the well known, large, and challenging KDD-CUP-98 task shows the effectiveness of our algorithm.

CHAPTER 5
PROFIT MINING: FROM PATTERNS TO ACTIONS

A major obstacle in data mining applications is the gap between the statistic-based pattern extraction and the value-based decision-making. "Profit mining" aims to reduce this gap. In profit mining, given a set of past transactions and pre-determined target items, we like to build a model for recommending target items and promotion strategies to new customers, with the goal of maximizing profit. Though this problem is studied in the context of retailing environment, the concept and techniques are applicable to other applications under a general notion of "utility".

There are several unique challenges in profit mining. First, we need to recommend both products and their prices, which are much more difficult than just predicting a class. And for most products they have different prices at different times, which makes the problem even more challenging. Second, products often have relationships among themselves. Making use of this domain knowledge for recommendation is a non-trivial task. We study these challenges and propose solutions in this chapter. Also we evaluate the effectiveness of our approach using both real and synthetic data sets. A paper based on the algorithm proposed in this chapter was accepted by EDBT 2002 conference [WZH02].

5.1 Introduction

It is a very complicated issue whether a customer buys a recommended item. Consideration includes items stocked, prices or promotions, competitors' offers, recommendation by friends or customers, psychological issues, conveniences, etc. For on-line retailing, it also depends on security consideration. It is unrealistic to model all such factors into a single system. In this chapter, we focus on one type of information available in most retailing applications, namely past transactions. The belief is that shopping behaviors in the past may shed some light on what customers like. We try to use patterns of such behaviors to recommend items and prices.

Consider an on-line store that is promoting a set of *target items*. At the cashier counter, the store likes to recommend one target item and a promotion strategy (such as a price) to the customer based on *non-target items* purchased. The challenge is determining an item interesting to the customer at a price affordable to the customer and profitable to the store. We call this problem *profit mining*.

Most statistics-based rule mining, such as association rules [AS94], considers a rule as "interesting" if it passes certain statistical tests such as support/confidence. To an enterprise, however, it remains unclear how such rules can be used to maximize a given business object. For example, knowing "*Perfume→Lipstick*" and "*Perfume→Diamond*", a store manager still cannot tell which of Lipstick and Diamond, and what price should be recommended to a customer who buys Perfume. Simply recommending the most profitable item, say Diamond, or the most likely item, say Lipstick, does not maximize the profit because there is often an inverse correlation between the likelihood to buy and the dollar amount to spend. This inverse correlation reflects the general trend that the more dollar amount is involved, the more cautious the buyer is when making a purchase decision.

5.2 Problem Definition

In profit mining, we are given a collection of past transactions, target items and non-target items, and promotion codes containing the pricing and cost information of items. A transaction contains one target sale of the form *<I, P, Q>*, for some target item *I*, and several non-target sales of the form *<I', P, Q>*, for non-target items *I'*. The presence of *<I, P, Q>* (or *<I', P, Q>)* in a transaction conveys that *I* (or *I'*) was sold in the quantity of *Q* under the promotion code *P*. Profit mining is to build a model, called *recommender*, that recommends a pair of target item *I* and promotion code *P* to future customers whenever they buy non-target items. A successful recommendation generates (*Price*(*P*)-*Cost*(*P*))**Q profi*t, where *Price*(*P*) and *Cost*(*P*) are the price and cost represented by *P*, and *Q* is the quantity sold because of the recommendation. The benefit goal is to maximize the total profit of target items on future customers.

Example 5-1 Suppose that a target item *2%_Milk* has four promotion codes (not necessarily offered at the same time): ($3.2/4-pack, $2), ($3.0/4-pack, $1.8), ($1.2/pack, $0.5) and ($1/pack, $0.5), where the first element denotes the price and the second element denotes the cost. Let *P* denote ($3.2/4-pack, $2). A sale <*Egg*, *P*, 5> generates of 5*($3.2-$2)=$6 profit on the target item. Note that the price, cost and quantity in a sale refer to the same packing (e.g., 4-pack).

Some (descriptive) items, such as *Gender=Male*, do not have a natural notion of promotion code. For such items, we set *Price*(*P*) and Q to 1 and *Cost*(*P*) to 0, and the notion of profit becomes the notion of support.

5.3 Related Work

Profit maximization is different from the "hit" maximization as in classic classification because each hit may generate different profit. Several approaches were proposed to make classification *cost-sensitive*. [Dom99] proposed a general method that can serve as a wrapper to make a traditional classifies cost-sensitive. [ZE01] extended the error metric by allowing the cost to be example dependent. [PAZ02] introduced a method to make sequential cost-sensitive decisions, and the goal is to maximize the total benefit over a period of time. These approaches assume a given error metric for each type of misclassification, which is not available in profit mining.

Profit mining is related in motivation to *actionability* (or *utility*) of patterns: A pattern is interesting in the sense that the user can act upon it to her advantage [ST96]. Recently, there were several works applying association rules to address business related problems. [BSVW99, WFW03, WS02] studied the problem of selecting a given number of items for stocking. The goal is to maximize the profit generated by selected items or customers. These works present one important step beyond association rue mining, i.e., addressing the issue of converting a set of individual rules into a single actionable model for recommending actions in a given scenario.

There were several attempts to generalize association rules to capture more semantics [CYS03, YHB04]. Instead of a uniform weight associated with each occurrence of an item, these works associate a general weight with an item and mine all itemsets that pass some threshold on the aggregated weight of items in an itemset. Like association rule mining, these works did not address the issue of converting a set of rules or itemsets into a model for recommending actions.

Collaborative filtering [RV97] makes recommendation by aggregating the "opinions" (such as rating about movies) of several "advisors" who share the taste with the customer. Built on this technology, many large commerce web sites help their customers to find products. The goal is to maximize the hit rate of recommendation. For items of varied profit, maximizing profit is quite different from maximizing hit rate. Also, collaborative filtering relies on carefully selected "item endorsements" for similarity computation, and a good set of "advisors" to offer opinions. Such data are not easy to obtain. The ability of recommending prices, in addition to items, is another major difference between profit mining and other recommender systems.

Another application where data mining is heavily used for business targets is *direct marketing*. See [LL98, MS96, WZYY03], for example. The problem is to identify buyers using data collected from previous campaigns, where the product to be promoted is usually fixed and the best guess is about who are likely to buy. The profit mining, on the other hand, is to guess the best item and price for a given customer. Interestingly, these two problems are closely related to each other. We can model the direct marketing problem as profit mining problem by including customer demographic data as part of her transactions and including a special target item NULL representing no recommendation. Now, each recommendation of a non-NULL item (and price) corresponds to identifying a buyer of the item. This modeling is more general than the traditional direct marketing in that it can identify buyers for more than one type of item and promotion strategies.

5.4 Our Approach

Our first consideration is that recommendation often depends on some categories (or concepts) of items. The categorization of items can be specified by a concept hierarchy [HF95, SA95].

Concept hierarchy. A *concept hierarchy*, denoted *H*, is a rooted, directed acyclic graph, with each leaf node representing an item and a non-leaf node representing a concept. For example, assume that an item *Flake_Chicken* belongs to categories *Chicken, Meat, Food, ANY.* If a customer bought *Flake_Chicken,* obviously the customer also "bought" *Chicken, Meat, Food, Any.* For non-target items, such generalization allows us to search for the best category that captures certain recommendations. We do not consider categories for target items because it does not make sense to recommend a concept and a price (such as *Appliance* for $100).

The key to profit mining is to recommend "right" items and "right" prices. If the price is too high, the customer will go away without generating any profit; if the price is too low or if the item is not profitable, the profit will not be maximized. To maximize profit, it is important to recognize that paying a higher price does not imply that the customer will not pay a lower price; rather, it is because no lower price was available at the transaction time. This behaviour is called *shopping on unavailability.* Taking into account this behaviour in rule extraction will bring new opportunities for increasing the profit.

Mining on availability - MOA. If a customer is willing to buy an item under some promotion code, we assume that the customer will buy the item under a more favourable promotion code. This assumption is called the *mining on availability,* or simply *MOA.* To incorporate the knowledge of MOA into search, we treat a more favourable promotion code P as a "concept" of a less favourable one P'. The effect is that a sale under P' implies a sale under P. This can be done by extending the concept hierarchy H as follows.

Definition 5-1 (MOA(H)) For each item I, let $(\prec, I)$ denote the hierarchy of pairs $<I, P>$ induced by $\prec$ on the promotion codes P for I, with I being added as the root. $MOA(H)$ is the hierarchy obtained by making each leaf node I in H as the root of the hierarchy $(\prec, I)$. □

A transaction can be generalized by generalizing its sales using $MOA(H)$ as defined below.

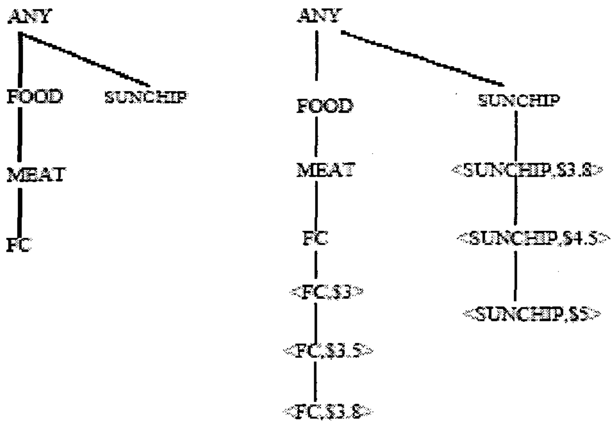
Definition 5-2 (Generalized sales) In $MOA(H)$, (i) every parent node is a *generalized sale* of every child node; (ii) every node of the form $<I, P>$ is a *generalized sale* of a sale of the form $<I, P, Q>$; (iii) "is a generalized sale of" is transitive. A set of generalized sales $G=\{g_1,..., g_k\}$ *matches* a set of sales $S=\{s_1, ..., s_p\}$ if each g_i is a generalized sale of some s_j. □

(i) generalizes a sale using concepts and favourable promotion codes. (ii) generalizes a sale by ignoring the quantity of the sale. A generalized sale has one of the forms $<I, P>$, or I, or C, where P is a promotion code, I is an item, C is a concept. For a target item I, we consider only generalized sales of the form $<I, P>$ because only this form represents our recommendation of a pair of target item and promotion code. Note that a generalized sale of the form $<I, P>$ contains the packing quantity defined by the promotion code P. The quantity of individual sales will be factored in the profit of rules.

Example 5-2 Consider a non-target item *Flaked_Chicken*, abbreviated as *FC*, and a target item *Sunchip*. Figure 5-1(a) shows the concept hierarchy *H*. Suppose that *FC* has three promotion codes: \$3, \$3.5, and \$3.8. *Sunchip* has three promotion codes: \$3.8, \$4.5, and \$5. For simplicity, we omit the cost and assume that the packing quantity for all promotion codes is 1. Figure 5-1 (b) shows *MOA(H)*. <*FC*, \$3.8> and its ancestors are generalized sales of sales <*FC*, \$3.8, Q>. <*FC*,

\$3> and its ancestors are generalized sales of sales <*FC*, \$3, Q>, or <*FC*, \$3.5, Q>, or <*FC*, \$3.8, Q>. Similar generalization exists for target item *Sunchip.*

Figure 5-1 ***H*** **and** ***MOA*(*H*)**


(a) ***H*** **(b)** ***MOA*(*H*)**

With the definition of *MOA*(*H*), a *rule* has the form $\{g_1, ..., g_k\} \rightarrow <I, P>$, where $g_1, ..., g_k$ are generalized non-target sales such that no g_i is a generalized sale of other g_j, and $<I, P>$ is a generalized target sale. Consider a customer represented by a set of non-target sales $\{s_1, ..., s_p\}$. A rule $\{g_1, ..., g_k\} \rightarrow <I, P>$ *matches* the customer if $\{g_1, ..., g_k\}$ generalizes $\{s_1, ..., s_p\}$.

5.5 Generating Rules

In profit mining the application goal is to maximize the profit by selling customers the recommended products. So we prefer the rules that can capture “customer intention” well. The confidence no longer serves this purpose and we need a new measurement here.

Suppose that a rule r: $\{g_1, ..., g_k\} \rightarrow <I, P>$ matches a given transaction t: $\{s_1, ..., s_p, <I_t, P_t, Q_t>\}$, where $<I_t, P_t, Q_t>$ is the target sale. If $<I, P>$ generalizes $<I_t, P_t, Q_t>$, that is, $I=I_t$ and $P \prec P_t$, then r has captured the intention of t. In this case, we credit the worth of r by the profit of r generated on t. To estimate this profit, we regard t as a future customer and determine the quantity Q the customer will buy under the more favourable promotion code P. The *generated profit* of r on t is defined as

$$p(r, t) = \begin{cases} (Price(P)\text{-}Cost(P))*Q, & \text{if } <I, P> \text{ generalizes } <I_t, P_t, Q_t> \\ 0, & \text{otherwise} \end{cases}$$

A conservative of estimation of the actual purchase quantity Q for the more favourable promotion code P under *MOA* is to keep the original quantity Q_t unchanged, thus, saving money. This estimation does not increase the spending at a favourable promotion for a customer. A more greedy estimation could associate the increase of spending with the relative favourability of P over P_t and the uncertainty of customer behaviours. We will consider such estimation in our experiments.

Definition 5-3 (Recommendation profit) Consider a rule r: $G \rightarrow g$. The *rule profit* of r, denoted as $Prof_{ru}(r)$, is defined as $\Sigma_t p(r, t)$, where t is a transaction matched by r. The *recommendation profit* of r, denoted as $Prof_{re}(r)$, is defined as $Prof_{ru}(r)/N$, where N is the number of transactions matched by r. □

The recommendation profit is on a per-recommendation basis and factors in both the hit rate (i.e., confidence) and the profit of the recommended item. It is possible that a rule of high recommendation profit matches only a small number of transactions that have large profit. Determining whether such rules should be used is a tricky issue and will be examined later.

To find rules of minimum worth, the user can specify minimum thresholds on these measures. The minimum support could be specified to take advantage of the support-based pruning [AS94]. If all target items have non-negative profit, a similar pruning is possible for rule profit and the minimum support can be replaced by the minimum rule profit. We follow [SA95, HF95] to find association rules, with *MOA(H)* being the input concept hierarchy.

In the rest of discussion, let R denotes the set of rules generated as above, plus the *default rule* $\varnothing \rightarrow g$, where g is the generalized target sale that maximizes $Prof_{re}(\varnothing \rightarrow g)$. Adding the default rule ensures that any set of non-target sales has at least one matching rule in R.

5.6 Building the Initial Recommender

A key for making recommendation is to select a recommendation rule from R for a given customer. Our selection criterion is maximizing the recommendation profit of the selected rule, as stated below.

Definition 5-4 (Rule ranking) *For any two rules r and r', we say that r is ranked higher than r'*

- (Recommendation profit) if $Prof_{re}(r) > Prof_{re}(r')$, or
- (Generality) if $Prof_{re}(r) = Prof_{re}(r')$, but $Supp(r) > Supp(r')$, or
- (Simplicity) if $Supp(r) = Supp(r')$, but $|lhs(r)| < |lhs(r')|$, or
- (Totality of order) if $|lhs(r)| = |lhs(r')|$, but r is generated before r'.

Given a set B of non-target sales, a rule r in R is the recommendation rule for B if r matches B and has highest possible rank. We also say that recommendation rule r covers B. □

If a rule is more special and ranked lower than some other rule in R, this rule will never be used as a recommendation rule because some general rule of higher rank will cover whatever it matches. From now on, we assume all such rules are removed from R.

With ranking criterion being established, we can turn a set of rules into a recommender like we did in previous chapters.

5.7 Optimizing the Recommender

So far we have not dealt with the over-fitting rules yet because a high recommendation profit does not imply a high support. It does not work to simply remove rules of low support because high-profit items typically have a low support. Our approach is to prune rules on the basis of increasing the projected profit on future customers. Suppose that we know how to estimate the projected profit of a rule r using the given transactions covered by r, denoted by *Cover*(r). We can prune one rule at a time if doing so increases the projected profit of the recommender, defined as the sum of the projected profit of all rules in the recommender.

Similar to what we did in direct marketing, we can build a decision-tree-like structure in which each node is a rule, and a rule r is the parent of rule r' if r is more general than r' and has the highest possible rank. If a rule r is pruned, the parent of r will cover the transactions covered by r.

Consider the current non-leaf node r. Let *Tree_Prof*(r) denote the projected profit of the subtree at r. Let *Leaf_Prof*(r) denote the projected profit of r as a leaf node. The estimation of these profits will be explained shortly. If *Leaf_Prof*(r)≥*Tree_Prof*(r), we prune the subtree at r immediately; otherwise, we do nothing at r.

Now we sketch the idea of estimating the projected profit of a rule r, denoted $Prof_{pr}(r)$. We estimate $Prof_{pr}(r)$ by X*Y. X is the (estimated) number of "hits" of r, i.e., number of acceptances of the recommendation, in a random population of N=|*Cover*(r)| customers that are covered by r. Y is the observed average profit per hit.

We can compute X using the *pessimistic estimation* borrowed from [Qui93]. Suppose that $E(r)$ of $N(r)$ transactions covered by r are not hit. For a given confidence level CF, the upper limit of the probability of non-hit in the entire population is estimated by $U_{CF}(E(r), N(r))$ as computed in [Qui93]. Then, X=N*(1-$U_{CF}(E(r), N(r))$). Y is estimated by

$$\frac{\sum_{t \in Cover(r)} p(r,t)}{num\, of\, hits\, in\, Cover(r)}$$

Recall that $p(r, t)$ is the generated profit of r on transaction t. *Tree_Prof*(r) is computed as the sum of $Prof_{pr}(u)$ over all nodes u in the subtree at r. This sum can be computed incrementally in the bottom-up traversal of the tree. *Leaf_Prof*(r) is computed as $Prof_{pr}(r)$ by assuming that r covers all the transactions covered by the subtree at r.

5.8 Evaluation

In this section we like to validate two claims: The refined recommender is profitable, and incorporating profit and MOA into model building is essential for achieving this profitability.

5.8.1 The Methodology

We perform 5 runs on each dataset using the 5-fold cross-validation. The average result of the 5 runs is reported. We define the *gain* of a recommender as the ratio of generated profit over the recorded profit in the validating transactions:

$$\sum_t p(r, t) / \sum_t \text{(the recorded profit in } t\text{)}$$

where $p(r, t)$ is the generated profit of the recommendation rule r on a validating transaction t. The higher the gain is, the more profitable is the recommender.

Let *PROF+MOA* represent the recommender that makes use of both profit and MOA in model construction. We compare *PROF+MOA* with:

- *PROF-MOA*: the recommender without MOA. This comparison will reveal the effectiveness of MOA.
- *CONF+MOA*: the recommender using the binary profit: $p(r, t)$=1 if the recommendation is a hit; otherwise, $p(r, t)$=0. Thus, the model building ignores the profit and relies on the hit rate (i.e., confidence) for ranking and pruning rules. This comparison will reveal the effectiveness of profit-based model building.

- *CONF-MOA*: *CONF+MOA* without MOA.

- *kNN*: the *k-nearest neighbour* classifier [YP97]. Given a set of non-target sales, *kNN* selects *k* transactions (the *k* nearest neighbours), for some fixed integer *k*, that are most similar to the given non-target sales and recommends the pair of target item and promotion code most "voted" by these transactions. We used the *kNN* that is tailored to sparse data, as in [YP97] for classifying text documents, and we applied MOA to tell whether a recommendation is a hit. These modifications substantially increase the hit rate and profit.

- *MPI*: the *most profitable item* approach, which simply recommends the pair of target item and promotion code that has generated most profit in past transactions.

5.8.2 Results on Synthetic Data Sets

5.8.2.1 Datasets

The synthetic datasets were generated by the IBM synthetic data generator[1], but modified to have price and cost for each item in a transaction. First, we apply the IBM generator to generate a set of transactions, with the number of transactions 100K and the number of items 1000, and default settings for other parameters. For simplicity, each item has single cost and single packing for all promotion codes. In this case, we use "price" for "promotion code". The cost of item I (here I is an integer) is denoted by $Cost(I)$. For item I, we generate the cost $Cost(I)=c/I$, where c is the maximum cost of a single item, and m prices $P_j=(1+j*\delta)*Cost(I)$, $j=1, \ldots, m$. We use $m=4$ and $\delta=10\%$. Thus, the profit of item I at its price P_j is $j*\delta*Cost(I)$. Each item in a transaction is mapped to a non-target sale by randomly selecting one price from the m prices of the item. For simplicity, all sales have unit quantity.

[1] http://www.almaden.ibm.com/software/quest/Resources/index.shtml

We consider two distributions for generating the target sale in each transaction. In dataset I, we consider two target items with cost of \$2 and \$10 respectively. Many important decision-makings such as direct marketing are in the form of two-target recommendation. We model the sales distribution of the two target items using the Zipf law[1]: The target item of cost \$2 occurs five times as frequently in the dataset as the target item of cost \$10. The price generation and selection for target items are similar to those for non-target items. In dataset II, there are 10 target items, numbered from 1 to 10. The cost of target item *I* is *Cost*(*I*)=10**I*. Unlike dataset I, the frequency of target items follows the normal distribution[2]: Most customers buy target items with the cost around the mean. In the following we only discuss the results obtained on dataset I. Similar results are observed on dataset II as well. Figure 5-2(e) show the profit distribution of target sales in dataset I.

5.8.2.2 Results

Figure 5-2 shows the results on dataset I. Figure 5-2 (a) shows the gain of the six recommenders (for *kNN*, *k*=5 gives the best result) with two obvious trends: *PROF+MOA* performs significantly better than other recommenders, and the recommenders with MOA perform significantly better than their counterparts without MOA. This clearly demonstrates the effectiveness of incorporating profit and MOA into the search of recommenders. *PROF+MOA* achieves 76% gain at minimum support 0.1%. This gain is encouraging because the MOA adopted is conservative in profit estimation.

[1] See http://www.nslij-genetics.org/wli/zipf/, for example

[2] See http://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm, for example

Figure 5-2 The results for synthetic dataset I

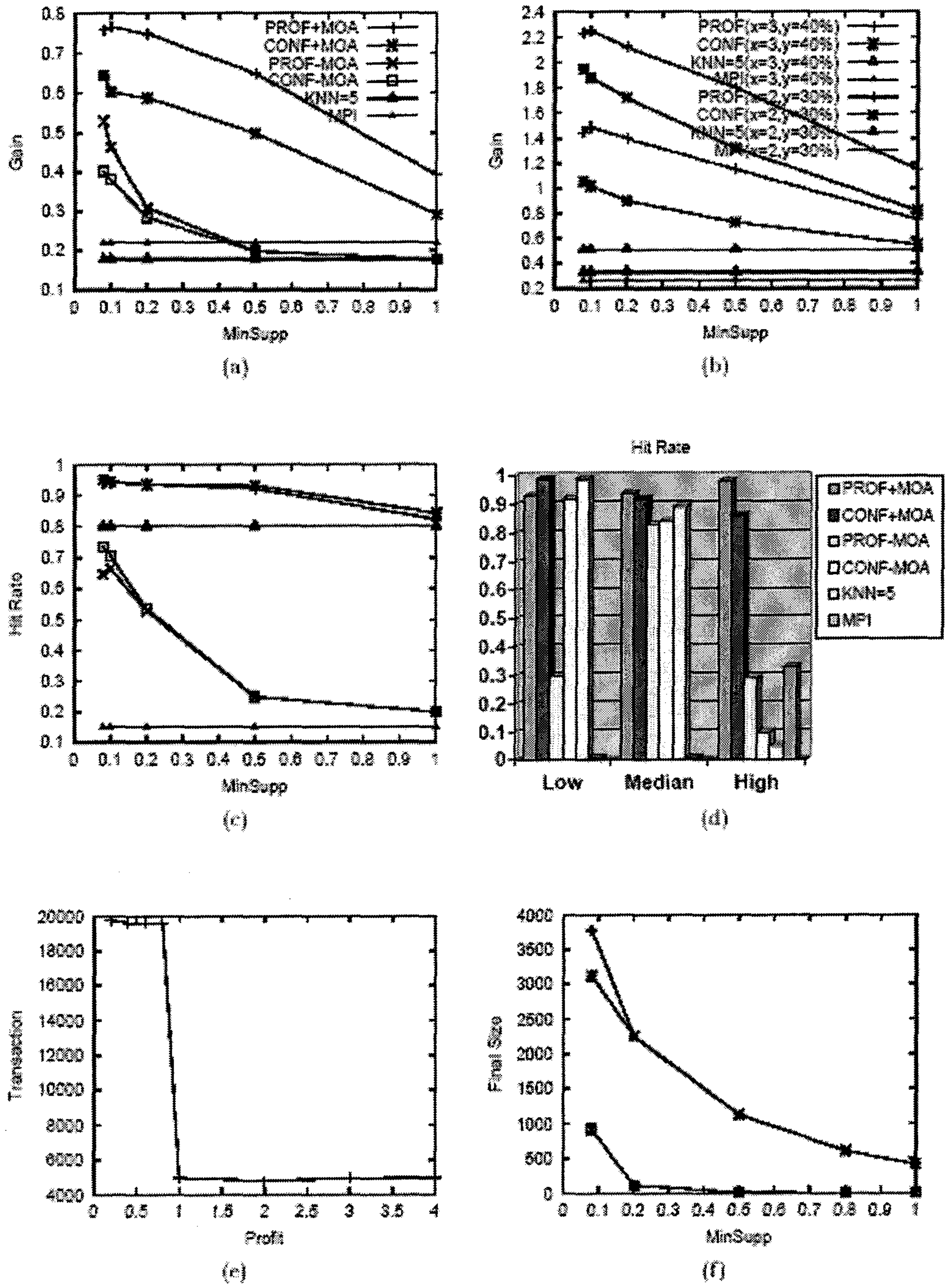
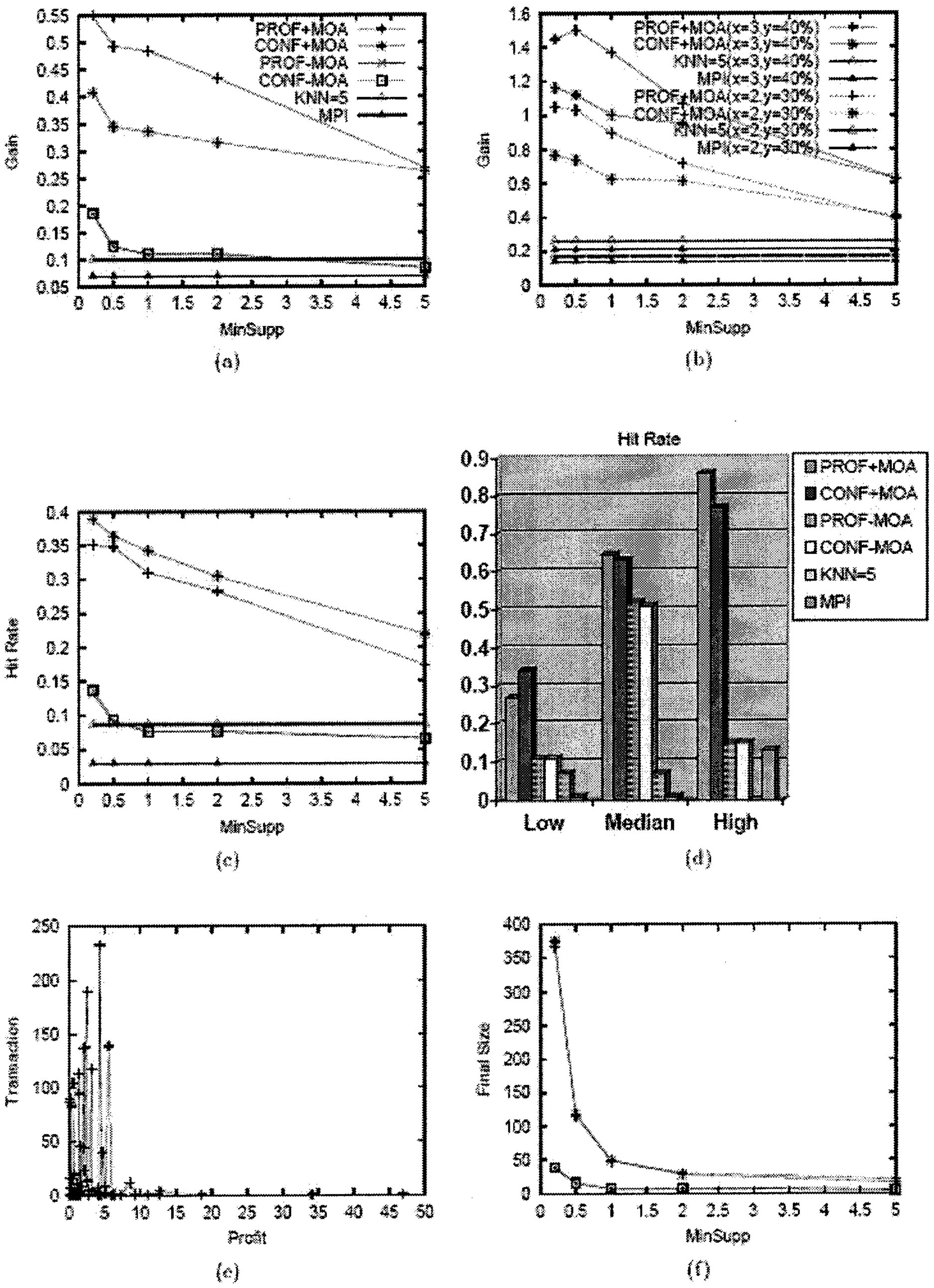


Figure 5-3 The results for real life data set



Interestingly, the curve for *PROF-MOA* shows that profit-based mining is not effective without MOA and the curves for CONF + MOA shows that MOA is not effective either without profit-based mining.

To model that a customer buys and spends more at a more favourable price, for each validating transaction, we compare the recommended price P_p with the recorded price P_q of the target item. Recall that $P_j=(1+j*\delta)*Cost(I)$, $j=1, \ldots, 4$, for item I. If q-p=1 or q-p=2, that is, the recommended price P_p is 1 or 2 step lower than the recorded price P_q, we assume that the customer doubles the purchase quantity in the transaction with the probability of 30%. We denote this setting by (x=2, y=30%). If q-p=3 or q-p=4, we assume that the customer triples the purchase quantity in the transaction with the probability of 40%. We denote this setting by (x=3, y=40%). Figure 5-2 (b) shows the gain of all recommenders using MOA with the purchase quantity determined by (x=2, y=30%) and (x=3, y=40%). With this more realistic shopping behaviour, the gain for all recommenders increases. *PROF+MOA* with the setting (x=3, y=40%), denoted *PROF*(x=3, y=40%), achieves the encouraging gain of 2.23 (at minimum support of 0.1%)!

Figure 5-2 (c), which uses the legend in Figure 5-2 (a), shows the hit rate of recommenders. *PROF+MOA* and *CONF+MOA* achieve the hit rate of 95%. For minimum support of 0.08%, Figure 5-2 (d) shows the hit rate at different profit ranges. "Low", "Medium", and "High" represent the low, middle, and high 1/3 of the maximum profit of a single recommendation. The legend from top to bottom corresponds to left to right in the bar chart. For example, *kNN* has nearly 100% hit rate at the "Low" range, but less than 10% at the "High" range. *CONF+MOA* and *CONF-MOA* also have a similar trend. In contrast, *PROF+MOA* is "profit smart" in maintaining a high hit rate in a high profit range. Though *MPI* picks up the hit rate in a high profit range, the hit rate is still too low compared to *PROF+MOA*. *PROF-MOA* is unstable for this dataset.

Figure 5-2 (f), which uses the legend in Figure 5-2 (a), shows the number of rules in recommenders. *kNN* and *MPI* have no model, so no curve is shown. The number of rules prior to the model-pruning phase (not shown here) is typically several hundreds times the final number. This shows that the pruning method proposed effectively improves the interpretability of recommenders. MOA generally increases the size due to additional rules for alternative prices. Not surprisingly, the minimum support has a major impact of the size. The execution time is dominated by the step of generating association rules. In our experiments, we adopted the multi-level association rule mining whose performance has been studied elsewhere [SA95, HF95]. The time for constructing the pruning tree from generated association rules and for the bottom-up traversal is insignificant.

We also conducted the experiment on data set II. This dataset has 40 item/price pairs for recommendation because each target item has 4 prices. Therefore, the random hit rate is 1/40, which is more challenging than dataset I. Despite the difference in cost distribution and a lower hit rate, the results are consistent with those of dataset I, that is, support the effectiveness of profit-based mining and MOA.

We also modified *kNN* to recommend the item/price of the most profit in the *k* nearest neighbours. This is a post-processing approach because the profit is considered only after the *k* nearest neighbours are determined. For dataset I, the gain increases by about 2%, and for dataset II, the gain decreases by about 5% (not shown in the figure). Thus, the post-processing does not improve much.

5.8.3 Results on Real Life Data Set

5.8.3.1 Dataset

This dataset comes from a retail chain store which sells goods and drugs across Canada. The raw data contains the information about items (i.e., price, cost, category, etc.) and the information about sales (i.e., transaction id, item, price, quantity, date of shopping, etc.) in several tables. Items are organized into a six-level category hierarchy. For our experiments, we extracted a dataset by specifying the category of target items and treating all items not in the category as non-target items. Transactions containing either no target item or no non-target item are discarded. If a transaction contains more than one target sale, we use the target sale of most profit. We report the result on the dataset with target items specified by the path:

ANY/General_Health/Pharmacy/Health_and_Beauty_Aids

There are 32 target item/price pairs, 1898 transactions containing these target items, and 3477 non-target items. Figure 5-3 (e) shows the profit distribution of target sales, which is quite different from those of synthetic datasets.

5.8.3.2 Results

The gain in Figure 5-3 (a) and (b) show that *PROF+MOA* is a clear winner over other recommenders. Again, Figure 5-3 (d) (for minimum support of 0.2%) shows that *PROF+MOA* picks up the hit rate quickly in a high profit range. The *kNN* modified to recommend the most profitable item/price (instead of most voted) only increases the gain by about 1% (not shown in the figure). This experiment further confirms the effectiveness of using profit and MOA in building recommenders. Figure 5-3 (f) shows that all recommenders for this real life dataset are much smaller than those for the synthetic datasets. This is due to the increased data sparsity in the case of more non-target items.

In summary, the experiments on both synthetic and real life datasets confirm our goals set at the beginning of the section.

5.9 Conclusion

In this chapter we study a profit-based decision-making application called *profit mining*. The goal of profit mining is to construct a recommender that recommends target items and promotion codes on the basis of maximizing the profit of target sales on future customers. We address several unique issues in profit mining: pruning specific rules on a profit-sensitive basis and dealing with the behaviour of shopping on unavailability. Experiments on a wide range of data characteristics show very encouraging results. This economic orientation and actionability will contribute to wider and faster deployment of data mining technologies in real life applications.

CHAPTER 6

LOCALIZATION SITE PREDICTION FOR MEMBRANE PROTEINS BY INTEGRATING RULE AND SVM CLASSIFICATION

In this chapter we study a localization prediction problem for membrane proteins in biology domain. Identifying a protein's location in a bacterial cell is of primary research interests for antibiotic and vaccine drug design. Biologists often have two basic requirements on the models built for localization prediction. First, prediction of a target localization site must have a *high precision in order to be useful to biologists. Achieving such a precision is made harder* by the fact that target sequences are often much fewer than background sequences. Second, the rationale of prediction should be understandable to biologists for taking actions. Meeting all these requirements presents a significant challenge.

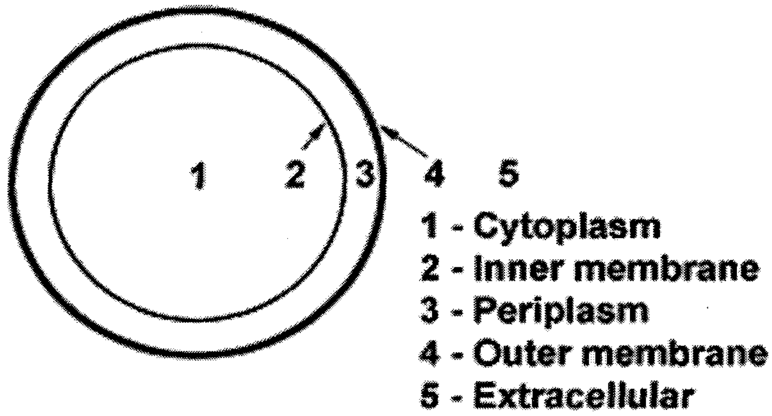
Recent research shows that the support vector machine (SVM) models [Vap95] achieve high precision in localization prediction. However, the kernel function of a SVM model could involve many features and is not easy for users to understand, therefore, does not address the second requirement. We address both requirements by integrating the SVM model with a rule-based model, where the understandable rules capture "major structures" and the elaborated SVM model captures "subtle structures". Importantly, the integrated model preserves the precision/recall performance of SVM and, at the same time, exposes major structures in a form understandable to human users. The purpose of the algorithm proposed in this chapter is not improving the precision/recall of SVM, but is *manifesting* the rationale of an SVM classifier through *partitioning* the classification between rules and the SVM classifier, and *preserving* the precision/recall of SVM. Unlike previous applications where we build pure rule based systems, in this chapter we build a hybrid decision-making system which consists of two components: rule-part and SVM-part. When constructing the rule-part, we still follow the ideas in the general framework. A paper based on the algorithm in this chapter was accepted by IEEE TKDE journal [ZW05].

6.1 Introduction

In the last decade, biologists have accumulated a huge amount of protein sequences. Each protein is composed of a linear sequence of amino acid residues. Since proteins play critical roles in determining the structures and functions of all living organisms [Str95], classifying these sequences into corresponding functional families is an important task for biologists.

One of the most important protein classification problems is to predict the subcellular localization of proteins [EB98]. For proper functioning, a protein has to be transported to the correct intra- or extra-cellular compartments in a soluble form, or attached to a membrane that surrounds the cell; hence the subcellular localization of a protein plays a key role with regard to its functions. Figure 6-1 shows the 5 primary localization sites for a family of disease-causing bacteria, collectively known as Gram-Negative bacteria. The ability to identify the localization site from the sequence information alone would allow researchers to quickly prioritise a list of proteins for potential drug and vaccine targets [SCW+03].

Figure 6-1 The five primary localization sites in a Gram-Negative bacterial cell



The above problem can be summarized as predicting the localization site for a protein from its amino acid sequence with the following requirements.

High precision. The precision of predicting the target localization site must be "very high", in most cases at least 90% or even 95%, while the recall is as high as possible. Whenever a protein is predicted to be located at the target site, the biologist wants to be fairly sure that the prediction is indeed correct [SCW+03]. Achieving high precision is made harder by the fact that the target examples are often much fewer than the examples in the contrasting class.

Interpretable models. Relevant patterns that summarize what triggers the prediction in a concise form are useful for biologists to perform further analysis and devise actions. To address the issue in a domain-independent manner, the interpretability refers to the *syntax simplicity* of the model such as the number and size of rules, not anything that requires domain knowledge.

High dimensionality. With any subsequence of amino acids being potentially a feature, it is common to have tens or even hundreds of thousands of features that are necessarily needed for high precision. Typically, combinations of features must be used to achieve high precision because general rules tend to include sequences in the contrasting class. Searching such combinations in a high dimensionality requires pruning a large portion of search space.

Meeting all these requirements presents a significant challenge because an inherently high dimensional problem requires a complex model that is hard to understand. Equally challenging is pruning a large portion of search space without degrading the performance of the final model. Finally, too much dependency on user-specified thresholds would introduce uncertainty to the robustness of the model, and an approach that minimizes this dependency is preferred.

Recently, SVMs demonstrate superior performance gains and robustness in many applications over traditional methods[1]. However, the SVM model comes with a major defect: It involves thousands of features in a single kernel function, making it impossible to see a simple relationship between the prediction and features that trigger it. A rule-based model such as ID3 and C4.5, on the other hand, presents the logic of prediction in the user-friendly rule format, but has inferior performance and often involves too many rules on high dimensional problems. To address the above requirements, an innovative solution is needed.

In this chapter, we integrate the SVM model with the rule-based model. The idea is to partition the classification so that each model captures the type of structures they are good at. The rule-based model captures "major structures" shared by many sequences in a small number of rules, and the SVM model captures "subtle structures" representing special case patterns and more complex relationships that do not have a concise description. The integrated model, called *rule-SVM (RSVM),* places the rules at the top and the SVM at the bottom: The SVM classifier is applied only if there is no matching rule. For this reason, we say that the rules *steal* classification from the SVM.

[1] http://www.clopinet.com/isabelle/Projects/SVM/applist.html

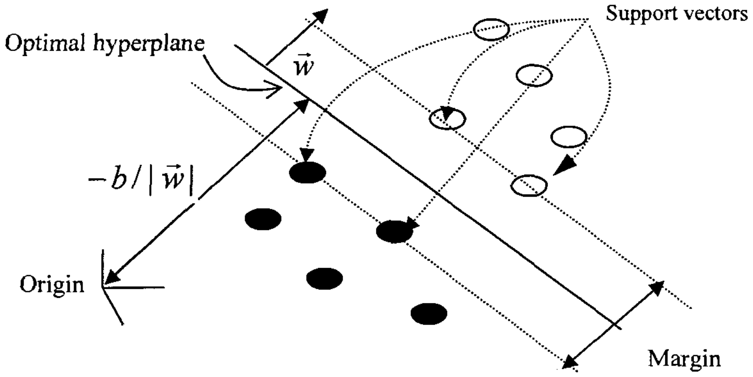
6.2 Algorithm Overview

The algorithm must address two key issues. First, the rules used must preserve the precision of SVM, which is typically higher than that of a rule-based classifier. Only high quality rules can be used. Second, the rules used are not to replace the SVM entirely, but to replace only the portion of classification that can be accurately represented by simple rules; the other remaining classification is still performed by SVM.

6.2.1 Background

A sequence is a string of items chosen from a fixed alphabet. For protein sequences, the alphabet consists of the 20 amino acid residues. A labelled sequence is associated with one of two classes: "positive" (+1) or "negative" (-1). A labelled sequence is positive/negative if its class is. Given a collection of labelled sequences D, D(train) and D(test) denote the split of the *training set* and the *testing set.* A classifier is built using D(train) and is evaluated by the precision and recall of classification on the testing sequences in D(test). The *precision* refers to the percentage of positive sequences among those that are classified as positive. The *recall* refers to the percentage of the sequences classified as positive among those that are indeed positive. A classifier is over-fitting if it is only accurate on D(train) but not on D(test). To avoid over-fitting, a classifier should use structures that are statistically significant, therefore, likely repeating in the whole population.

Figure 6-2 A linear SVM in a two-dimensional space



SVMs are based on the *Structural Risk Minimization* principle [Vap95] from computational learning theory. The idea is finding a hyperplane that separates the positive examples from negative ones in the training set while maximizing the total distance of *support vectors* from the hyperplane, where support vectors are the examples (positive and negative) that have the shortest distance to the hyperplane. Figure 6-2 illustrates the maximum margin separating hyperplane and the support vectors in the two dimensional space. The norm vector $\vec{w}$ represents the direction of the separating hyperplane, which is determined through a set of support vectors. For an SVM with a linear kernel, a new sequence *d* is classified by the sign of the following decision function:

$$f(d) = \sum_{i=1}^{n} w_i * x_i + b \qquad \textbf{(6-1)}$$

where d=<x_1, x_2 ..., x_n>, $\vec{w}$=<w_1,w_2 ...,w_n> and w_i is the weight for the *ith* feature. *d* is classified as positive if *f(d)>0*, and as negative otherwise. In this case, finding the SVM classifier is determining the weight w_i and the bias *b*. We consider SVMs with a linear kernel function in this chapter. Our previous studies show that the linear kernel function achieves better or similar results on outer membrane proteins [SCW+03]. For the non-linearly separable case, we can first transform the problem into a linearly separable problem [Bur98] and apply the method presented in this chapter.

6.2.2 Our Approach

We first map each sequence into a data point in a multi-dimensional space. Each dimension, also called a *feature,* is defined by a *frequent segment*, i.e., some consecutive items that occur in at least some minimum fraction of sequences specified by the *minimum support.* We find frequent segments only from positive sequences since we are interested in predicting the positive class. Compared to the spectrum kernel [LEN02] that uses features of a fixed length, our approach allows features of flexible length. Suppose that we have n features. We map a sequence to a 1/0 vector $<x_1, x_2 ..., x_n>$ in the feature space: If the *ith* feature occurs in the sequence, $x_i=1$, otherwise, x_i=0.

As in most cases, we consider only rules that predict the positive class (but our work can be extended to rules for two classes). A *rule* is a set of features. A rule *matches* a sequence (or vice versa) if the sequence contains all the features in the rule. Given a set of sequences, the *support* of a rule is the percentage of matched sequences among all the sequences, and the *confidence* of a rule is the percentage of positive sequences among all matched sequences. A rule classifies a matched sequence as the positive class.

We are interested in a classification model that has the performance of SVM classifiers but expresses "major structures" in the form of rules. We propose such a model called RSVM.

Definition 6-1 Let M be an SVM classifier. Let $r_1, r_2 ... r_m$ be a set of rules. A *RSVM* classifier has the form:

$$r_1, r_2 ..., r_m, M.$$

R(RSVM) denotes the set of rules $r_1, r_2 ..., r_m$. R(RSVM) *steals* the classification from M as follows: if a sequence matches some rule r_i, classify the sequence as the positive class, otherwise, classify the sequence by M. A RSVM is required to satisfy three properties:

- *Performance*: R(RSVM) has a precision similar to that of M on D(test).
- *Significance*: R(RSVM) steals a large portion of classification from M. In other words, R(RSVM) shares a significant portion of recall.
- *Interpretability:* R(RSVM) contains a small number of simple rules. □

The intention is that the rule portion R(RSVM) captures simple and major structures, whereas M captures subtle structures that do not have simple rule representation. The performance requirement ensures that R(RSVM) preserves the precision of the SVM (therefore, RSVM preserves the precision of the SVM). Under this condition, rules are preferred to the SVM. The significance and interpretability requirements ensure that the rules play active roles in manifesting a significant portion of classification in a simple and understandable form. Our objective is to capture simple structures by rules whenever they exist.

We find a RSVM classifier in three phases. *SVM phase* maps training sequences to the feature space and builds the SVM classifier M using the standard software. *Rule phase* generates a set of high performance rules preserving the precision of M. *Stealing phase* determines the partition of classification between the rules and M. We explain each phase in details.

6.3 SVM Phase

First, we find a set of frequent segments as the feature space. Frequent segments are mined from the positive sequences in D(train). To count the support of segments, we implemented the generalized suffix tree (GST) [WCM+94].

To avoid losing useful features, a small minimum support should be used. We used the minimum support of 1% in all our experiments and it worked fine. SVM is quite robust in dealing with the high dimensionality and ignoring insignificant features (by assigning a small weight). In addition, our pruning strategies will prune insignificant features before the rule generation. Therefore, a small minimum support does not necessarily blow up the rule generation, but helps include potentially useful features.

Next, we map the sequences in D(train) to data points in the feature space, as described in Section 6.2.2, and apply the standard SVM software to produce the SVM classifier M, in particular, the weights w_i and the bias b. We use the *SVM-light*[1] implementation of SVMs in [Joa98a].

6.4 Rule Phase

This phase generates statistically significant rules that preserve the precision of M. Before generating rules, we split D(train) into two disjoint parts randomly: D(build) and D(prune). We use D(build) for the rule generation and use D(prune) for making pruning decisions. We use the split of 4/5 vs. 1/5.

Since the application goal here is to build a mixed classifier which has at least the accuracy given by SVMs, we should only include the rules that have at least the same precision as the SVM classifier M. Such requirement needs to be accommodated during rule mining.

A major challenge for rule generation in this case is the large search space due to tens of thousands of features. A limitation with most rule generation algorithms such as [AIS93] and [AS94] is the heavy dependency on user-specified thresholds (e.g., minimum support) to prune rules. It is often difficult for the user to decide appropriate values for such thresholds, while this decision significantly impacts the performance of resulting classifiers. Note that we use minimum

[1] http://svmlight.joachims.org

support in the SVM phase for feature generation, not rule generation, where the suffix tree algorithm is very efficient, and generated features are subject to the weighting by the SVM model and further pruning by our method.

Definition 6-2 A rule *preserves* the SVM M if the precision of the rule is not less than the precision of M on the sequences in D(build) that match the rule. □

If a preserving rule is "statistically significant", it will preserve the SVM precision over unseen sequences. We will consider a measure of statistical significance shortly. Unlike other works such as [AIS93], which require users to specify thresholds in advance, our algorithm determins the statistical significance automatically.

6.4.1 Generating Rules

The task here is to find all statistically significant, preserving rules. To reduce the database scan, we search rules in a level-wise manner starting from shortest rules. For details of level-wise rule generation please refer to [AMS+96]. Here we briefly explain how it is used to generate significant rules. Let R_1 be the set of all size-1 rules $\{f_i\}$ over the features for which M has a non-zero weight w_i. Next, we extend every rule $\{f_i\}$ in R_1 by adding one feature f_j in R_1 to generate all size-2 statistically significant rules $\{f_i, f_j\}$, where $f_i \neq f_j$, denoted R_2. If a rule is not statistically significant (see Section 6.4.5 for the definition of rule significance), we will not include it in R_2 and not extend it further because any extended rule is not statistically significant. In general, at the *kth* iteration, we generate the size-(k+1) statistically significant rules, denoted R_{k+1}, using two rules $\{f_1, \ldots, f_{k-1}, f_k\}$ and $\{f_1, \ldots, f_{k-1}, f_{k+1}\}$ in R_k. One scan of D(build) would check the statistical significance of generated rules. We continue this process until no statistically significant rule is generated. Then, we scan D(build) once to filter out all rules that do not preserve M. This rule generation can be expensive if the number of features is large.

Below, we consider several strategies to prune the search space. The first two strategies are aimed at pruning features before the rule generation. The last two strategies are aimed at pruning rules during the rule generation.

6.4.2 Pruning Redundant Features

Our first observation is that if several features occur exactly in the same set of sequences in D(build), the rule generation is not able to distinguish them and we can remove all such features except one before the rule generation. A special case is that such features have sub-string/super-string relationships (e.g., "bc", "abc", "bcd", "abcd").

Strategy 1: For all features that occur exactly in the same set of sequences in D(build), we keep only one of them for the rule generation. □

6.4.3 Pruning Insignificant Features

If a feature does not have a significant contribution, it can be pruned before the rule generation. Consider Equation (6-1). The further the weight w_i is from zero, the more influential the *ith* feature f_i is on the decision value $f(x)$. Therefore, we can sort the features f_i having non-zero w_i according to the influence $|w_i|$ into a list F and concentrate on the features in some prefix of F. (Other ranking criteria such as information gain can be used instead. But we believe that w_i is preferred because such weights are determined in the presence of all features.) To determine this prefix, for each prefix F', we consider the simplified SVM, denoted M', based on only the features in the prefix F'. M' is obtained from M by setting $w_i=0$ for all features f_i not in F'. Let E' be the error of M' on D(prune).

Strategy 2: Select the shortest prefix F' of F with the minimum E'. □

In other words, *F'* is selected to minimize the error of the simplified SVM on D(prune). The use of D(prune) instead of D(build) is to avoid the over-fitting that tends to select the full list *F*.

Figure 6-3 Feature coefficients for the sample data

f_i	f_1	f_4	f_2	f_3	f_6
w_i	0.636	-0.581	0.554	0.554	0.118
f_i	f_7	f_5	f_8	f_9	b
w_i	0.118	0.036	0.018	0.018	-1.091

Example 6-1 We use D(train), split into D(build) and D(prune), in Table 6-1 to show Strategy 2. Ignore the last column at this moment. For simplicity, we show the features (i.e., f_1, f_2, ..., f_9) contained in each sequence instead of actual amino acids. Applying the software *SVM-light* to D(build), we get the SVM classifier M described by the weights w_i and the bias b in Figure 6-3, sorted by $|w_i|$. Consider the prefix $F'=<f_1>$. All the sequences in D(prune) are predicted as negative by the simplified SVM M' because $w_1+b<0$. So $E'=2$. Similarly, we can compute the error for all other prefixes. The shortest prefix with the minimum error is $<f_1, f_4, f_2, f_3>$, which has 1 error on *d11*. The remaining five features, i.e., f_5-f_9, are pruned from D(build) and D(prune). The classes predicted by this simplified SVM are listed in the column "Predicted class", which happens to be exactly the same as that of the original M based on all 9 features. □

Table 6-1 A sample data set

ID	Examples	Class	Predicted class
D(build)			
d1	f_1, f_2, f_3, f_5, f_8	+1	+1
d2	f_1, f_2, f_3	+1	+1
d3	f_1, f_3, f_6, f_7	+1	+1
d4	f_1, f_2, f_4, f_8, f_9	+1	-1
d5	f_1, f_4, f_5	-1	-1
d6	f_2, f_4, f_6	-1	-1
d7	f_3, f_4, f_7	-1	-1
d8	f_1, f_4, f_8, f_9	-1	-1
d9	f_3, f_4, f_6	-1	-1
d10	f_2, f_8	-1	-1
D(prune)			
d11	f_1, f_3, f_4, f_6	+1	-1
d12	f_1, f_2, f_3, f_4, f_5	+1	+1
d13	f_2, f_3, f_4, f_6	-1	-1
d14	f_4, f_5, f_6, f_7	-1	-1
d15	f_2, f_3, f_4, f_7, f_8	-1	-1

6.4.4 Pruning Redundant Rules

Consider two rules $r_1=\{f_1, f_2 \ldots f_k\}$ and $r_2=\{f_1, f_2 \ldots f_k, f_{k+1}\}$, where r_2 is obtained by extending r_1 with the feature f_{k+1}. If f_{k+1} is a sub-string of some feature in r_1, e.g., r_1={“*abcd*”} and r_2={"*abcd*", "*ab*"}, the two rules will match exactly the same set of sequences, and we can keep the shorter rule r_1 and prune the longer rule r_2. Now consider the case that f_{k+1} is a super-string of some feature in rule r_1, say f_1 without loss of generality. From the above discussion, we only need to consider $\{f_2, \ldots f_k, f_{k+1}\}$ instead of r_2. We summarize these observations in the next strategy.

Strategy 3: If a feature f_{k+1} has sub-string or super-string relationship with some feature in rule r_1, stop extending r_1 with f_{k+1}. □

6.4.5 Pruning Insignificant Rules

Now we consider how to tell if a rule is statistically significant. A statistically significant rule should be accurate on the *whole population*, in addition to D(build). Like what we did in previous chapters, we use the *pessimistic error estimation* [Qui93] to estimate the error rate of r_1 on the whole population. Suppose that a rule r_1 matches N_1 sequences in D(build), among which E_1 are classified wrongly. Given a confidence level *CF*, we take $U_{CF}(E_1,N_1)$ as the estimated error rate of r_1 on the whole population.

Suppose that we extend the rule $r_1=\{f_1, f_2 \ldots f_k\}$ with E_1/N_1 to the rule $r_2=\{f_1, f_2 \ldots f_k, f_{k+1}\}$ with E_2/N_2. Note that $N_2 \leq N_1$. If $U_{CF}(E_1,N_1) \leq U_{CF}(E_2,N_2)$, we regard the rule r_2 as over-fitting, i.e., statistically insignificant, because it does not decrease the error rate on the whole population. Once r_2 is over-fitting, so is any rule obtained by extending r_2.

Strategy 4: If r_2 is an extension of r_1 such that $U_{CF}(E_1,N_1) \leq U_{CF}(E_2,N_2)$, stop generating r_2 and any extension of r_2 (because r_2 is statistically insignificant). □

Example 6-2 Let *CF* be *25%*. Suppose that the features f_1, f_2, f_3, f_4 represent frequent segments "a", "b", "c", "ab", respectively. The rule $r_1=\{f_1\}$ matches 6 examples in D(build) in Table 6-1, with 2 being negative. So $N_1=6$, $E_1=2$, and the upper limit $U_{CF}(E_1,N_1)$ is estimated to be *0.56*. We extend r_1 into $r_2=\{f_1, f_2\}$. Note that f_2 has no sub-string or super-string relationship with f_1. r_2

matches 3 examples in D(build), all being positive. So $N_2=3$, $E_2=0$, and $U_{CF}(E_2,N_2)$ is estimated to be 0.37. Since $0.37 < 0.56$, according to Strategy 4, rule r_2 is kept. Now consider extending r_2 to $r_3=\{f_1, f_2, f_3\}$. r_3 matches 2 examples in D(build), both being positive. So $N_3=2$, $E_3=0$, and $U_{CF}(E_3,N_3)$ is estimated to be 0.50. Since $0.50 > 0.37$, rule r_3 is dropped and no further rule is extended from r_3. □

6.5 Stealing Phase

Let $R=\{r_1, r_2 \dots r_k\}$ be the set of statistically significant, preserving rules found in the rule phase. Let us assume that the rules $r_1, r_2 \dots r_k$ are sorted by the confidence on D(build), and in case of tie, sorted by support. Now we can turn the rule list into a classifier: To classify a sequence, the first matching rule in the list, if there is one, is applied because of higher confidence. Under this preference, rules towards the end of the list tend to classify fewer examples, therefore, have less contribution. To reduce the classifier size, we only select a prefix of R for building the RSVM classifier.

Consider a prefix R' of R. Let $E(R')$ be the error of R' on the matching sequences in D(prune), and let $E(M,R')$ be the error of M on such sequences. $E(M,R')-E(R')$ measures the (possibly negative) performance gain of replacing M with R' over such sequences. To ensure that as many sequences as possible are classified by rules, we select the longest prefix R' that maximizes $E(M,R')-E(R')$. Note that the selected prefix has a non-negative performance gain $E(M,R')-E(R')$ because the empty prefix gives the zero performance gain. We then remove all the rules in the selected R' that classify no sequence in D(prune). Finally, we put R' on top of the SVM to construct the RSVM classifier:

$$R', M.$$

Example 6-3 Continue on Example 6-2. Suppose that two rules, $r_1=\{f_1, f_3\}$ and $r_2=\{f_2, f_3\}$, are found in the rule phase. They have the same confidence in D(build), but the first rule has higher support. So, *R* in the sorted order is (r_1,r_2). Figure 6-4 shows the classification of the examples in D(prune) by each prefix *R'*. The prefix $R'=(r_1)$ is selected because it is the longest prefix that maximizes *E(M,R')-E(R')*. The RSVM classifier is:

$$r_1, M.$$

On D(prune), r_1 correctly classifies two positive examples (i.e., *d11* and *d12*) and classifies no negative example as positive. In comparison, M incorrectly classifies one of the two positive examples (i.e., *d11*) as negative. Therefore, the use of r_1 has actually improved the performance of the SVM classifier on D(prune). In terms of interpretability, r_1 presents a more understandable structure of positive sequences than the SVM kernel function that involves 9 features. □

Figure 6-4 Selecting the prefix

Prefix *R'*	The empty prefix	r_1	r_1, r_2
Examples classified in D(prune)	none	*d11, d12*	*d11, d12, d13, d15*
E(R')	0	0	2
E(M,R')	0	1	1
E(M,R')-E(R')	0	1	-1

6.6 Experiments

The purpose of experiments is to evaluate the properties of RSVM classifiers and the effectiveness of various pruning strategies. We use the SVM-light[1] implementation [Joa98a] of SVMs, and compare the interpretability of RSVMs against the C4.5 classifier[2] [Qui93], which is widely considered as accurate and understandable classifiers. For fair comparison, we choose the

[1] http://svmlight.joachims.org

[2] http://www.rulequest.com/Personal/

rule option C4.5 classifier that has fewer rules than the tree option. We use default settings in both systems and conduct experiments on a PC with 2.4G CPU and 1GB main memory.

The evaluation was conducted using two groups of membrane: Gram-Negative bacteria[1] and Gram-Positive bacteria[2]. All proteins included in these data sets have been experimentally verified for their localization sites. Each group has several primary localization sites. One data set can be created by taking each primary localization site as the positive class and taking the remaining sites as the negative class. We chose the 5 data sets on which SVMs have at least 90% precision and 30% recall. The feature set was mined with the minimum support of 1% or 2 positive sequences, whichever is larger, and features of length less than 3 were discarded because they tended to occur in every sequence. Table 6-2 describes the data sets based on the average of the 5-fold cross validation. For example, the data set named "Neg-Inner" is from Gram-Negative bacteria and has "Inner membrane" as the positive class.

For comparison purpose, we considered several competing classifiers: SVM, Rule-alone, C4.5-prune and C4.5-all. SVM is the standard SVM classifier. Rule-alone is the rule list produced in the rule phase but cut off by minimizing the error on D(prune), with the negative class being the default class. Rule-alone serves the baseline for our rules without integration with SVMs. C4.5-all is the standard C4.5 classifier (of the rule option). C4.5-prune is the standard C4.5 classifier built using the feature set produced after pruning redundant and insignificant features as described in Section 6.4. We also followed [Joa98b] and built the C4.5 classifier using the top $p\%$ features (ranked by information gain), where $p=\{1, 5, 10, 20, 50\}$. The results are either much worse or very close to C4.5-prune, so are not included here.

[1] http://www.psort.org/dataset/datasetv1.html, Version 1.1. This is the version available at the time of experiments

[2] http://www.psort.org/dataset/, Version 2.0.

Table 6-2 Data statistics

Data sets	# Seq.	# Pos. seq.	Seq. length	# Features	# Features per seq.
Neg-Inner	1572	292	410	34828	3817
Neg-Outer	1572	377	559	42079	3507
Neg-Extra	1572	191	469	115786	7904
Pos-Cellwall	576	61	1059	87727	9449
Pos-Extra	576	183	451	67381	3096

6.6.1 Performance and Significance

Table 6-3 shows the precision/recall (in percentage) on D(test). RSVM preserves the precision and recall of SVM quite well. This is because the rule portion R(RSVM) has a precision comparable to the precision of SVM. Consequently, RSVM outperforms the C4.5 classifiers by a similar margin as SVM does. Rule-alone has a (slightly) higher precision than RSVM, i.e., 3%, because it was selected to minimize the error on the sequences it matches. However, this slight advantage is at the heavy expense of a much lower recall, i.e., 18% compared to 69% of RSVM. RSVM has at least as much recall as the SVM. This is a consequence of using positive rules only in RSVM: If a sequence is predicted as positive by the SVM, it will be predicted as positive by either R(RSVM) or the (same) SVM in the RSVM. Typically, the recall of RSVM is several percentage points higher than that of the SVM because additional structures were captured by rules.

The significance of the RSVM is measured by the portion of classification performed by rules, i.e., the recall of R(RSVM). This is shown under the R(RSVM) column in Table 6-3. The larger this recall is, the more classification is stolen by the rules and the more effective the rules are. Note that these rules are constrained to preserve the precision of SVM, so simply including more rules in R(RSVM) does not help. On average, the recall of R(RSVM) is 30%, compared to the 69% recall of the RSVM. This means that about 43% of the classification (of the positive class) done by the RSVM was performed by rules, therefore, was manifested to the human user. As we will show shortly, these rules are quite compact and are understandable to the human user.

Table 6-3 Precision/Recall (%) on D(test)

Data sets	**RSVM**		**SVM**	**Rule-alone**	**C4.5-prune**	**C4.5-all**
	R(RSVM)	**RSVM**				
Neg-Inner	97/41	98/88	99/85	98/17	61/51	60/43
Neg-Outer	100/22	98/81	98/80	100/15	70/72	69/68
Neg-Extra	96/23	93/44	94/42	100/17	60/49	58/45
Pos-Cellwall	91/37	89/68	91/67	90/19	57/46	50/43
Pos-Extra	93/28	92/68	96/54	98/23	55/48	50/41
Average	**95/30**	**94/69**	**95/65**	**97/18**	**60/53**	**57/48**

Compared to the C4.5 classifiers, R(RSVM) is more than 36% more accurate in precision. This huge gain makes the rules of the RSVM more useful to the biologist, who wants to be damn sure that any prediction about the target localization is correct. Though the C4.5 classifiers have a larger recall, their quality is much less trusted, because of the significantly lower precision (i.e., 36% lower). Compared to Rule-alone, R(RSVM) is preferred due to the much higher recall (i.e., 12% higher) with only slightly lower precision (i.e., 2% lower).

We also compare with publicly available software tools TMHMM[1] and Phobius[2] that are primarily used to identify the presence and location of transmembrane helices in a protein. The

[1] http://www.cbs.dtu.dk/services/TMHMM

[2] http://phobius.cgb.ki.se/index.html

presence of transmembrane helices indicates inner membrane proteins (also called the cytoplasmic membrane), and 3 or more transmembrane helices is a more reliable indication [GSW+03]. Based on this property, TMHMM and Phobius produce Precision/Recall of 98/83 and 99/82 on the testing data of our 5-fold cross validation. While the precision is similar to that of RSVM and SVM, the recall is 3% to 6% lower. If we require only 2 or more transmembrane helices, these numbers are 94/91 and 95/88, and if we require only 1 or more transmembrane helices, these numbers are 66/96 and 87/96. Note that this method cannot predict the other localization sites where proteins do not necessarily contain transmembrane helices.

6.6.2 Interpretability

Table 6-4 compares *x*/*y*/*z* in R(RSVM), Rule-alone and C4.5-prune, where *x* is the number of rules, *y* is the average rule length, and *z* is the average feature length (C4.5-all has more rules than C4.5-prune, so is not included). The column "# Non-zero weight features in SVM" contains the number (and percentage) of non-zero weight features in the kernel function of the SVM classifier. R(RSVM) and Rule-alone have a rather small number of rules, i.e., 21 and 14 respectively, with short rules (i.e., average of 2.2 features per rule) and simple features (i.e., average of 5.5 amino acids per feature). Rule-alone has fewer rules than R(RSVM), but it comes with a much lower recall (see the above discussion). C4.5-prune uses much more rules, i.e., 50, and the features in these rules are much longer, i.e., the average of 25.8 amino acids per feature, than those in R(RSVM) and Rule-alone. These features were chosen by C4.5 because of high confidence in D(train), therefore, high information gain. But since these features have very low support, they did not perform well on D(test), which explains why C4.5-prune has a low precision (see Table 6-3). Our rule generation prunes rules containing such features due to statistical insignificance. The SVM classifier has tens of thousands of features in its kernel function even after removing all zero weight features. A complexity of this scale would bury any useful and simple structures that the biologist could use for further analysis and actions.

Table 6-4 Comparison on interpretability

Data sets	**R(RSVM)**	**Rule-alone**	**C4.5-prune**	**# Non-zero weight features in SVM**
Neg-Inner	31 / 2.0 / 4.0	22 / 2.0 / 4.0	61 / 2.1 /	15176 (43.57%)
Neg-Outer	15 / 2.6 /	12 / 2.7 /	63 / 2.1 /	39895 (94.80%)
Neg-Extra	12 / 2.1 / 4.1	11 / 2.1 / 4.1	40 / 2.2 /	17902 (15.46%)
Pos-	13 / 2.0 / 4.1	8 / 2.0 / 4.1	15 / 2.2 /	13345 (15.21%)
Pos-Extra	33 / 2.1 / 4.0	17 / 2.3 / 4.0	72 / 2.1 /	26591 (39.46%)
Average	**21 / 2.2 / 5.4**	**14 / 2.2 / 5.5**	**50 / 2.1 /**	**22581 (41.7%)**

6.6.3 Pruning Effectiveness

As shown in Table 6-2, there are tens and even hundreds of thousands of features, and each sequence contains more than 3000 features. Mining rules from such high dimensional data is extremely expensive and must rely on strong pruning strategies to reduce the search space. The column "Features kept" in Table 6-5 shows the percentages of features remaining at different stages, with respect to the initial number of features. The first number is the percentage of features after pruning redundant features (Strategy 1). The second number is the percentage of features after pruning those with zero weight in the SVM model. The third number is the percentage of features after pruning insignificant features (Strategy 2). Roughly speaking, almost 2/3 of features are redundant, 1/3 of non-redundant features have zero weight, and 1/3 of the remaining non-zero weight features are further pruned due to insignificance. As a result, the rules of R(RSVM) are searched using no more than 11% of the features that are used for training SVM. The column "Features per seq." in Table 6-5 shows that, by feature pruning (Strategy 1 and 2), the average number of features contained in a sequence is reduced to 2.1% of the number of features in a sequence before the pruning. This significantly reduces the data size, the search space, and the rules generated.

Table 6-5 Effectiveness of feature pruning

Data sets	Features kept (%)	Features per seq. (%)
Neg-Inner	45.0 / 28.1 / 20.0	2.5
Neg-Outer	35.2 / 13.4 / 8.5	2.0
Neg-Extra	28.7 / 6.3 / 4.1	1.5
Pos-Cellwall	17.0 / 10.6 / 7.0	2.0
Pos-Extra	30.6 / 19.3 / 13.0	2.1
Average	**31.3 / 15.5 / 10.5**	**2.0**

With only 10% (of the features in Table 6-2) remaining after the feature pruning, the number of features ranges from 10^3 to 10^4. Without any rule pruning, the number of size-k rules is 10^{3*k} to 10^{4*k}. The maximum *k* for the rules in our RSVMs is 3. This amounts to the search space of 10^9 to 10^{12} rules if no rule pruning is done.

Figure 6-5 compares the average CPU time (seconds) for building RSVM and C4.5 classifiers. The time for generating the feature set is the same for all algorithms and is not included. For RSVM, the time includes building the SVM model, rule generation and stealing phase. More than 70% of the time was spent on the rule phase. For this reason, the time for Rule-alone (not shown) is similar to the time for RSVM. For the C4.5 classifiers, the time includes building the decision tree and rule pruning. RSVM is more efficient than C4.5-prune. C4.5-all is too slow due to the high dimensionality of data.

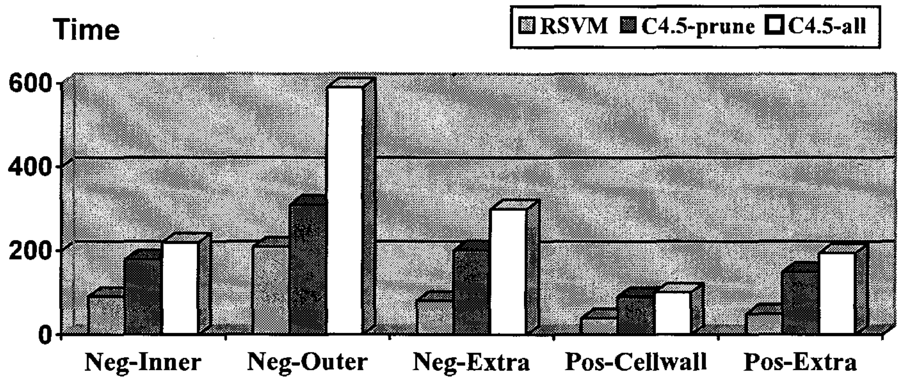
Table 6-6 presents the number of rules generated at different phases in our algorithm. Compared to the above search space without rule pruning, the number of rules generated (denoted "#Significant" for statistically significant rules) is significantly reduced. Among the rules generated, about 1% to 10% are preserving rules (denoted "#Preserving"), and only about 0.01% is included in the final RSVM (denoted "#Final").

Figure 6-5 compares the average CPU time (seconds) for building RSVM and C4.5 classifiers. The time for generating the feature set is the same for all algorithms and is not included. For RSVM, the time includes building the SVM model, rule generation and stealing phase. More than 70% of the time was spent on the rule phase. For this reason, the time for Rule-alone (not shown) is similar to the time for RSVM. For the C4.5 classifiers, the time includes building the decision tree and rule pruning. RSVM is more efficient than C4.5-prune. C4.5-all is too slow due to the high dimensionality of data.

Table 6-6 Effectiveness of rule pruning

Data sets	**#Significant**	**#Preserving**	**#Final**
Neg-Inner	$6.0*10^6$	$5.6*10^4$	31
Neg-Outer	$8.7*10^6$	$1.9*10^5$	15
Neg-Extra	$1.1*10^6$	$1.2*10^4$	12
Pos-Cellwall	$5.1*10^6$	$8.6*10^4$	13
Pos-Extra	$3.4*10^6$	$2.5*10^4$	33

Figure 6-5 CPU time (in seconds) for building classifiers



6.7 Related Work

Many algorithms have been proposed on membrane protein localization problem, including neural network [DFU+98, JMF+01, RH98], Markov chain model [Yua99], hidden Markov model [MFKC02], and SVM [HS01, Ver02]. In all these works, the prediction is a black box because there was no attempt to make the prediction understandable. On the other hand, traditional rule-based classifiers, such as C4.5 and ID3, are relatively easy to understand but they perform poorly on high dimensional problems such as the one considered here, compared to the SVM model. Many algorithms were proposed that try to combine the strengths of these two approaches.

Extracting understandable rules has been intensively studied for neural network classification [ADT95]. The decomposition method focuses on extracting rules at the level of individual components of neural networks, such as clustering the hidden unit activation, searching for weighted links that caused hidden or output units to be active. The learning-based method extracts rules by using the neural network to generate examples for a rule-based method. The situation is similar in the case of SVMs. All these works attempt to replace the neural network or SVM with the rules extracted, which tends to produce too many rules and unmatched performance. We emphasize preserving the performance of SVM, by employing only high quality rules and replacing only part of the SVM classification.

The hybrid decision tree [ZC02] builds an upper portion of the standard decision tree and embeds neural networks into some leaves to accomplish the remaining prediction. The classification at a leaf node represented by a neural network is still a black box. The perceptron decision tree generalizes the standard one-attribute split at each internal node by a general split represented by a hyperplane. See [BCW00] for examples. Each conjunct in the body of a rule is a multivariate linear inequality. Though perceptron decision trees have demonstrated good results for some real world problems, they tend to over-fit the data by involving many variables in a split, due to the increased flexibility. Rules generated by such splits are less interpretable.

Recently, association rules have been used for classification for high dimensional transactional data [AMS97, LHM98, WZH00] and have shown promising results on outer membrane localization prediction [SCW+03]. However, this approach has several limitations: It depends on a carefully chosen minimum support; the performance is not as good as SVM; and the number of rules used is large, therefore, not easily understandable.

6.8 Conclusion

Motivated by applications in antibiotic and vaccine drug design, we examine the subcellular protein localization problem for disease-causing bacteria in this chapter. This problem has several demanding and conflicting requirements: high precision of prediction, interpretability of models, and high dimensionality of data. Our approach is integrating the precision-driven SVM model with the interpretable rule-based model, with each doing what they are best at. The SVM model focuses on classification involving subtle structures, whereas the rule-based model focuses on main structures that can be represented by concise rules. The integrated model, called RSVM, *preserves* the performance of the SVM model and *exposes* simple structures in understandable rules. The experiments on real subcellular protein localization tasks have demonstrated the effectiveness of RSVMs.

CHAPTER 7
DISCOVERING CATALOG MATCHINGS ON THE WEB

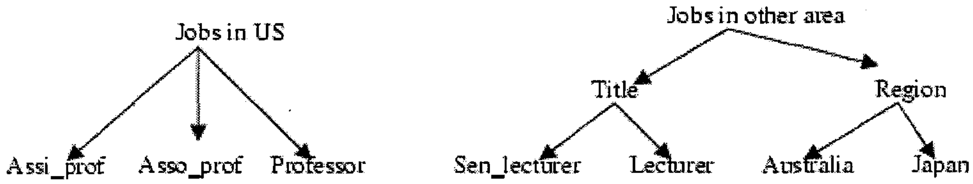
In this chapter we study an interesting decision-making problem which establishes the mapping between two different catalogs. Unlike previous applications where we mainly discuss how to build decision-making systems from a *given* data set, in this chapter we focus on how to *generate* the appropriate data set (from given data) so we can apply rule-based decision-making algorithms on it.

The most successful paradigm for making the mass of information on the Internet comprehensible is by organizing them into catalogs, i.e., categories (or topics) of hierarchical specificity. Due to the distributed nature, there are explosive numbers of catalogs even for applications in the same domain. It makes the information exchange and comparison difficult and slows down the business negotiation. In this chapter, we address this problem by learning the concept of a given category in one catalog in terms of the categories in the other catalog. Such characterization tells how the documents in a given category in one catalog are categorized in another catalog, even without knowing what these documents are. We express the learning results in interpretable rules so they are easy to understand. We evaluate this approach using real world data sets and the results are promising.

7.1 Introduction

An essential requirement for information interchange in electronic markets is the following *catalog matching*: given two distributed catalogs (H_1, DB_1) and (H_2, DB_2) in the same domain, where DB_i is a collection of text documents categorized according to the categories in H_i, characterize a given category in H_1 in terms of the categories in H_2. The characterization tells whenever a document falls into certain categories in H_2, it falls into the given category in H_1. The two catalogs are distributed in that they were created independently, and are in the same domain in that they addressed a similar application. The following example illustrates some of the points.

Figure 7-1 Job catalogs



Example 7-1 Consider two job catalogs (H_1, DB_1) and (H_2, DB_2), one for North America and one for other regions, shown in Figure 7-1. Suppose that a catalog matching for the "Asso_prof" category in H_1 could be

Sen_lecturer, Australia →Asso_prof

where "Sen_lecturer" and "Australia" are categories from H_2. This matching says that a job opening under *both* the "Sen_lecturer" and "Australia" categories in H_2 would be under the "Asso_prof" category in H_1. Knowing this matching, an employer in North America may consider offering an associate professor position to a senior lecturer applicant from Australia. □

The major challenge for catalog matching comes from the distributed nature of sources where the catalogs were created independently and catalog matching is an after-thought. They do not necessarily share same categories or documents (*collection diversity*); different terms may be used for semantically similar categories, and the same term may be used for semantically dissimilar categories (*naming diversity*); pair wise or structural correspondence rarely exists (*structural diversity*).

Our approach. A key issue is how to model the notion of "same domain" under the "distributed assumption". A recent study by He and Chang [HC03] suggests that sources of the same domain tend to have converging vocabularies. Though their study is for schema matching in the context of "deep web", it is applicable to catalog matching because the distributed nature remains the same. This observation leads us to hypothesize the following:

Hypothesis 7-1 For two catalogs (H_1, DB_1) and (H_2, DB_2) in the same domain, there exists a common underlying document distribution from which DB_1 and DB_2 are drawn. □

The essence of this hypothesis is that a model M_A learned about a category A from (H_1, DB_1) could be applied to the documents in (H_2, DB_2) to determine their A or $\neg A$ category. Therefore, for each document $<C_2, D_2>$ in (H_2, DB_2), where D_2 contains terms and C_2 is the set of H_2-categories for D_2, we create a new example $<A, C_2 \cup D_2>$ or $<\neg A, C_2 \cup D_2>$, stating that a document $<C_2, D_2>$ will have the A or $\neg A$ category in (H_1, DB_1). By ignoring D_2, we have a set of examples $<A, C_2>$ or $<\neg A, C_2>$ that tell what H_2-categories co-occur with the A or $\neg A$ category. Then we can extract and refine rules of the form $x_1, \ldots, x_k \rightarrow A$ for expressing the matching for A, where x_i's are categories or absence of categories in H_2.

This approach has several major differences from existing works.

- The two sources do not have to share common documents, or similar structures and category names. In contrast, [Cha00, MWJ99, NM00] uses a variety of heuristics, such as common tokens, to match ontology elements, and [DMDH02, ITH01, LG01] assumes that a category is always mapped to a single (i.e., the most similar) category in the other catalog, i.e., pair wise correspondence. For example, these approaches cannot find the matching in Example 7-1.
- It expresses the matching at the category level, not the term level. Rules of the form $x_1, \ldots, x_k \rightarrow A$ tell the correspondence between the categories x_i's in H_2 and the category A in H_1 for *all* documents. Such matching is at the summary level, therefore, easier to understand. In contrast, catalog integration [AS01, OF01, SM01, ZL04] and

cross-training [SCG03] perform the matching "one document at a time" because the target category A of a document in (H_2, DB_2) depends on both terms and categories of the document.

- It deals with the language, search, and accuracy of matching by principled machine learning techniques. A set of rules $x_1, \ldots, x_k \rightarrow A$ for A expresses general Boolean relationships between x_i's and A and can be searched as classification rules for A. Machine learning was used in [DMDH02], but only pair wise correspondence was considered where the language and search of matching were not a major issue.
- Catalog matching is semantically different from schema matching [RB01]. In [DLD+04, HCH04], complex semantic correspondences between schemas are studied. For example, the attribute "*author*" in one database could correspond to the attribute group {"*first name*", "*last name*"} in another database. Quite differently, catalog matching is about the correspondence of the topics of documents categorized: "*Sen_lecturer*", "*Australia*"→ "*Asso_prof*" represents the documents summarized in the "*Asso_prof*" category are summarized in both "*Sen_lecturer*" and "*Australia*" categories in another source. Schema information often is available for schema matching, including data type, relationships, constraints, schema structure [RB01]. However, such information is not available for catalog matching.

7.2 Problem Definition

We first define some concepts. A document is a bag of *terms*. A *catalog* is a collection of documents *DB* categorized into categories of a hierarchical structure. A child category represents a specialization (e.g., Sedan) of a parent category (e.g., Car). The *hierarchical categorization* implies that if a document belongs to a category, it also belongs to all ancestor categories. In this chapter, a category starts with the upper case and a term starts with the lower case.

Example 7-2 (Running example) Figure 7-2 and Table 7-1 show two catalogs (H_1, DB_1) and (H_2, DB_2). Document D_{11} contains the terms *corn, farm, rice* and belongs to the category *Grain*, therefore, *Plant*. Document D_{22} belongs to the categories *Vegetable* and *Dairy*. □

Figure 7-2 Example hierarchy

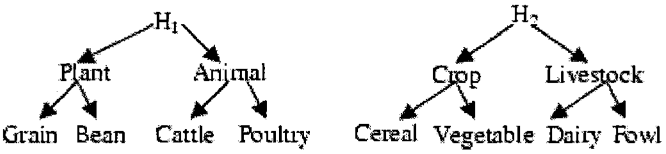


Table 7-1 Example data set

Terms	Categories
(H_1, DB_1)	
D_{11}: *corn, farm, rice*	C_{11}: *Grain,Plant*
D_{12}: *farm, pea, soybean*	C_{12}: *Bean, Plant*
D_{13}: *beef, milk, rice*	C_{13}: *Cattle, Grain, Animal, Plant*
D_{14}: *chicken, egg*	C_{14}: *Poultry, Animal*
(H_2, DB_2)	
D_{21}: *farm, rice, wheat*	C_{21}: *Cereal, Crop*
D_{22}: *milk, pea, potato*	C_{22}: *Vegetable, Dairy, Crop, Livestock*
D_{23}: *farm, milk*	C_{23}: *Dairy, Livestock*
D_{24}: *egg, hatchery*	C_{24}: *Fowl, Livestock*

7.2.1 Matching Rules

Consider two catalogs (H_1, DB_1) and (H_2, DB_2). A matching rule for a category A in H_1 has the form,

$$x_1, \ldots, x_k \rightarrow A$$

where each conjunct x_i is a category, or term, or absence of such, from (H_2, DB_2). We read this matching rule as: If a document fits the description $x_1, \ldots, x_k$ in (H_2, DB_2), it would belong to the category A in (H_1, DB_1). A single matching rule expresses A as $\wedge$ (intersection) and $\neg$ (differentiation) of the sets of documents containing certain terms or belonging to certain categories in H_2. A set of matching rules for A expresses A as $\vee$ (union) of several such descriptions. Thus, matching rules are substantial generalization of the pair wise correspondence in [DMDH02].

For hierarchical catalogs, categories are not independent of each other and there is a further requirement on matching rules. If a category x_i is an ancestor of a category x_j, x_i and x_j cannot co-occur in a matching rule, neither can $\neg x_i$ and $\neg x_j$. However, x_i and $\neg x_j$ can co-occur in a matching rule. Matching rules extracted by C4.5 [Qui93] automatically ensure this property.

We are primarily interested in matching rules in which x_i's are either categories or absence of categories, but not terms. Such rules are more explicit and understandable to the human user because they express the matching by the categories that summarize the documents, not the detailed terms in the documents. However, there are cases where terms are necessary for accurate matching. For example, the "Camry" category in one catalog cannot be accurately described by any combination of the categories "Sedan", "SUV", and "Truck/Mini Van" in another catalog. Since "Camry" corresponds to a subclass of "Sedan" that has the medium size, terms such as "medsize" in some documents under "Sedan" would help: "Sedan, medsize→Camry". However, we hypothesize that most catalog matching can be captured at the category level or nearly category level, as stated below.

Hypothesis 7-2 For two catalogs in the same domain, only matching rules involving no or few terms are needed. □

Hypothesis 7-2 does not mean that terms can be ignored in the search of catalog matching. To the contrary, our approach heavily depends on documents' terms to learn a model for determining the external categories in the other catalog. However, once such categories are determined, Hypothesis 7-2 allows us to search for matching rules by ignoring most terms. We will experimentally study the extent to which Hypothesis 7-1 and Hypothesis 7-2 hold.

7.2.2 Catalog Matching

Motivated by Hypothesis 7-2, we define the problem of catalog matching as follows. Consider two catalogs (H_1, DB_1) and (H_2, DB_2). The *catalog matching* for a category A in H_1 is a set of matching rules for A learnt from a data set that contains no more than a user-specified σ percentage of the terms in DB_2. σ is called *term-allowance*. We say that a catalog matching is at the *category level* if the term-allowance is zero, i.e., the matching rules involve only categories.

A catalog matching serves two purposes. The first purpose is to predict the A or $\neg A$ category for documents whose categories in H_2 are known, therefore, to integrate the documents in (H_2, DB_2) into (H_1, DB_1). If the catalog matching is at the category level, this prediction or integration depends on only the categories, not the content, of the documents. The second purpose is to characterize the A category in terms of the categories in H_2, in order to understand the relationships between the categories in the two catalogs.

7.3 Algorithm Overview

To find the catalog matching for A in H_1, our approach works as follows. From (H_1, DB_1) in which each document belongs to either A or $\neg A$, we learn a model, called the *head-generating* model, to determine A or $\neg A$ for each document $<C_2, D_2>$ in (H_2, DB_2), where C_2 is the H_2-categories of document D_2. Hypothesis 7-1 implies that such cross-source application is valid. This would create examples of the form $<A, C_2 \cup D_2>$ or $<\neg A, C_2 \cup D_2>$, stating that D_2

categorized as C_2 in (H_2, DB_2) would be categorized as A or $\neg A$ in (H_1, DB_1). This set of examples, called the *matching set* for A, provides a training data for learning matching rules x_1, ..., $x_k \rightarrow A$, with the descriptors x_i coming from $C_2 \cup D_2$. To satisfy the term-allowance constraint, we remove all but the top σ percentage of terms from the matching set. A standard feature selection can be used for this purpose.

Alternatively, the matching set for A can be obtained by generating the H_2-categories for each document <A, D_1> or <$\neg A$, D_1> in (H_1, DB_1). In particular, we learn a model for each category B in H_2 using the training set (H_2, DB_2). The collection of these models is called the *body-generating* model. We then apply these models to each document <A, D_1> or <$\neg A$, D_1> in (H_1, DB_1) to determine their H_2-categories, say C_2. The matching set consists of the documents <A, $C_2 \cup D_1$> or <$\neg A$, $C_2 \cup D_1$>. The terms in D_1 but not in DB_2 are removed because the descriptors in a matching rule come from (H_2, DB_2).

Figure 7-3 summarizes the above approach in four steps. Step 1 learns either a head-generating model or a body-generating model. Step 4 extracts a catalog matching by learning a rule-based classifier from the matching set. Both steps can be done by applying standard learning algorithm. Step 3 removes all but the top σ percentage of terms and is done by a standard feature selection. Below, we focus on the key step of generating the matching set, which involves Step 1 and Step 2.

We first consider flat H_1 and H_2 to present the main ideas, and then extend them to hierarchical H_1 and H_2 where the impact of hierarchical categories is examined. In each case, we consider head-generating model and body-generating model for generating the matching set.

Figure 7-3 The overview of catalog mapping algorithm

```
Input:      ($H_1$, $DB_1$) and ($H_2$, $DB_2$), the target category $A$ in $H_1$, and term-allowance $\sigma$
Output:     the catalog matching for $A$
Algorithm
    1. Learn the (head- or body-) generating model;
    2. Apply the generating model to generate the matching set for $A$;
    3. Remove all but the top $\sigma$ percentage of terms from the matching set;
    4. Learn the catalog matching for $A$ using the matching set;
```

7.4 Generating Matching Rules from Flat Catalogs

This section assumes that the hierarchies H_1 and H_2 are flat. We consider two alternative implementations.

7.4.1 Flat-Head

In the first algorithm, called *Flat-Head*, Step 1 learns the head-generating model. The training set contains a positive example for each document in (H_1, DB_1) that belongs to A, and a negative example for each remaining document in (H_1, DB_1). Let M_A denote this model. Step 2 applies M_A to the documents in (H_2, DB_2) to generate the A or $\neg A$ category for them. The example below explains.

Table 7-2 The matching set

Class	**Attributes**									
	egg	*milk*	*pea*	*rice*	*Cereal*	*Vegetable*	*Dairy*	*Fowl*	*Crop*	*Livestock*
Grain	0	0	0	1	1	0	0	0	1	0
¬Grain	0	1	1	0	0	1	1	0	1	1
¬Grain	0	1	0	0	0	0	1	0	0	1
¬Grain	1	0	0	0	0	0	0	1	0	1

Example 7-3 Consider (H_1, DB_1) and (H_2, DB_2) in Table 7-1, but ignore the hierarchies above the table. Suppose that we want to find the catalog matching for *Grain* in H_1. To learn the head-generating model M_{Grain}, the positive examples are D_{11}, D_{13} because they belong to *Grain*, and the negative examples are D_{12}, D_{14} because they belong to *¬Grain*. Suppose that M_{Grain} has only one rule: *rice→Grain*. Note that this is a regular classification rule, not a matching rule. Finding classification rules from a given training set is a standard classification problem.

Next, we apply M_{Grain} to the documents in (H_2, DB_2) to generate the matching set for *Grain*: If a document contains *rice*, label it as *Grain*, otherwise, as *¬Grain*. This produces the matching set:

<Grain, $C_{21}\cup D_{21}$>, <*¬Grain*, $C_{22}\cup D_{22}$>,

<*¬Grain*, $C_{23}\cup D_{23}$>, <*¬Grain*, $C_{24}\cup D_{24}$>.

C_{2j} denotes the categories for D_{2j} as given in Table 7-1. Suppose that the terms *farm, hatchery, potato*, and *wheat* are pruned for satisfaction of the term-allowance constraint. Table 7-2 shows the matching set in the form of a relational table with *Grain* and *¬Grain* being as the class labels and the remaining terms and categories from (H_2, DB_2) being the features. Now, a standard rule-based learner can be applied to extract the matching rules for *Grain* from this matching set. □

7.4.2 Flat-Body

In the second algorithm, called *Flat-Body*, we produce the matching set by generating the H_2-categories for the documents in (H_1, DB_1). First, we learn a body-generating model from (H_2, DB_2), consisting of one model M_B for each H_2-category B. The training set for M_B contains the documents belonging to B as positive examples and the documents belonging to $\neg B$ as negative examples. Then, for each document $<A, D_1>$ or $<\neg A, D_1>$ in (H_1, DB_1), we generate its H_2-categories by applying every M_B to D_1. Let C_2 be the set of H_2-categories B which M_B categorizes D_1 as belonging to. Before adding $<A, C_2 \cup D_1>$ or $<\neg A, C_2 \cup D_1>$ to the matching set, we remove all terms in D_1 that do not occur in DB_2. Step 3 and Step 4 remain the same as in Flat-Head.

Example 7-4 Suppose that the body-generating model is:

$M_{Crop}=\{farm \rightarrow Crop\}$, $M_{Livestock}=\{milk \rightarrow Livestock\}$,

$M_{Cereal}=\{rice \rightarrow Cereal\}$, $M_{Vegetable}= \{pea \rightarrow Vegetable\}$,

$M_{Dairy}=\{milk \rightarrow Dairy\}$, $M_{Fowl}=\{egg \rightarrow Fowl\}$.

To produce the matching set for *Grain*, we apply these models to the documents in (H_1, DB_1). Consider D_{11} for example. Since D_{11} contains terms *corn, farm* and *rice*, rules M_{Crop} and M_{Cereal} match it. Hence, D_{11} is assigned H_2-categories *Crop* and *Cereal*, creating the example:

$$<Grain, \{Cereal, Crop\} \cup D_{11}>$$

Similarly, we create other examples in the matching set for *Grain*:

$<\neg Grain, \{Vegetable, Crop\} \cup D_{12}>$, $<Grain, \{Dairy, Livestock, Cereal\} \cup D_{13}>$,

$<\neg Grain, \{Fowl\} \cup D_{14}>$. □

In Flat-Body, it is possible that the body-generating model M_B may generate "conflict" categories B for a document in (H_1, DB_1). Consider D_{14}. $M_{Livestock}$ classifies it as $\neg$*Livestock.* However, M_{Fowl} classifies it as *Fowl,* which is a child of *Livestock.* This issue will be addressed later when we discuss Hierarchical-Body approach.

7.5 Generating Matching Rules from Hierarchical Catalogs

Now, we consider hierarchical H_1 and H_2. In this case, a category B (i.e., those at a lower level) can get so specific that few documents belong to B and most documents belong to $\neg B$. Many learning algorithms get into trouble when faced with such imbalanced data [Pro00]. In particular, extracting rules for B will be difficult at Step 1 and 4. We address this issue below.

7.5.1 Hierarchical-Head

Our observation is that if we have a model to test whether a document belongs to the parent of B, say B', we do not need to include those documents that fail the test when learning the model for B. This makes sense because we can classify a document into B or $\neg B$ by first applying the model for B', if succeeded, further applying the model for B. This approach excludes all documents not belonging to B' from the training set for B, thereby, making the $B/\neg B$ distribution more balanced. Similarly, to learn a model for B', we could consider only those documents that belong to the parent of B'; and so on.

This observation leads to a new algorithm, called *Hierarchical-Head.* Consider the path from the root of H_1 to A,

$$A_0, A_1, \ldots, A_k,$$

where A_0 is the root and A_k=A. We build the head-generating model for A as a sequence of models,

$$M_{A1}, \ldots, M_{Ak},$$

where M_{Ai} is a model for A_i, learned using only the documents that belong to the parent of A_i. To generate the matching set for A, for each document $<C_2, D_2>$ in (H_2, DB_2), we apply these models *hierarchically*: D_2 is initially categorized as A_0, and we apply M_{Ai} to D_2 only if D_2 has been categorized as A_{i-1} by M_{Ai-1}. If D_2 is eventually categorized as A, we add $<A, C_2 \cup D_2>$ to the matching set; if D_2 is categorized as the parent of A but not A, we add $<\neg A, C_2 \cup D_2>$. Note that all the documents not categorized as the parent of A are not included in the matching set for A.

Example 7-5 The head-generating model for *Grain* consists of two models, M_{Plant} and M_{Grain}. M_{Plant} is built using positive examples $\{D_{11}, D_{12}, D_{13}\}$ and negative example $\{D_{14}\}$. Suppose $M_{Plant}=\{farm \rightarrow Plant\}$. To build M_{Grain}, we consider only the documents belonging to *Plant*: D_{11} and D_{13} are the positive examples, and D_{12} is the negative example. Suppose $M_{Grain}=\{rice \rightarrow Grain\}$.

The matching set for *Grain* is generated as follows. First, we apply M_{Plant} to all documents in (H_2, DB_2), categorizing D_{21} and D_{23} as *Plant*. Then, we apply M_{Grain} to (only) these documents, categorizing D_{21} as *Grain*, and D_{23} as $\neg$*Grain*. So, the matching set for *Grain* consists of

$<Grain, C_{21} \cup D_{21}>$, $<\neg Grain, C_{23} \cup D_{23}>$.

This data set is more balanced compared to the matching set in Example 7-3. □

7.5.2 Hierarchical-Body

In *Hierarchical-Body*, we create the matching set for A using a body-generating model. Like in Flat-Body, the body-generating model consists of one model M_B for each category B in H_2. Unlike in Flat-Body, M_B is learned using only the documents in (H_2, DB_2) belonging to the *parent* of B, as in Section 7.5.1. To produce the matching set for A, we consider only the documents D_1 in (H_1, DB_1) that belong to the parent of A and determine its H_2-categories, denoted C_2 using the body-generating model. Particularly, for each category B in H_2, we apply the models

M_{Bi} on the path to B "hierarchically" as in Section 7.5.1. C_2 is the set of categories B which D_1 is categorized as belonging to.

Example 7-6 In general, the body-generating model for Flat-Body is not the same as for Hierarchical-Body. For simplicity, we borrow the body-generating model in Example 7-4 to illustrate how to apply it in Hierarchical-Body. To produce the matching set for *Grain*, we apply these models to the documents in (H_1, DB_1) that belong to the parent of *Grain*, namely, D_{11}, D_{12}, D_{13}, to determine their H_2-categories. Consider D_{11} for example. First, M_{Crop} categorizes D_{11} as *Crop*, then, M_{Cereal} categorizes D_{11} as *Cereal* and $M_{Vegetable}$ categorizes D_{11} as $\neg$*Vegetable*. $M_{Livestock}$ categorizes D_{11} as $\neg$*Livestock*. We do not further apply M_{Dairy} and M_{Fowl} to D_{11}. So, *Crop* and *Cereal* are the H_2-categories for D_{11}. Similarly, we can generate the H_2-categories for D_{12} and D_{13}. This gives the following matching set for *Grain*:

<*Grain*, {*Cereal, Crop* }$\cup D_{11}$>, <$\neg$*Grain*, {*Vegetable, Crop*}$\cup D_{12}$>,

<*Grain*, {*Dairy, Livestock*}$\cup D_{13}$>. □

7.6 Evaluation

This section evaluates the three approaches, Flat-Head, Hierarchical-Head, Hierarchical-Body, in terms of the effectiveness of discovering catalog matching, and makes recommendations. Flat-Body is not considered because it may produce conflict categories as discussed earlier. We used *SVM-light*[1] in Step 1 for learning the generating model due to its good performance for handling text documents. In Step 4, we used C4.5-rules[2] since it also has rule generation and rule pruning phases. The default settings were used for both tools.

[1] http://svmlight.joachims.org/

[2] http://www.rulequest.com/Personal/c4.5r8.tar.gz

7.6.1 Evaluation Criteria

We evaluate catalog matching by accuracy and simplicity. The accuracy is measured by the recall and precision of predicting the given category A in H_1 for the documents in H_2. *Recall* refers to the percentage of documents that are predicted as belonging to A among those that actually belong to A. *Precision* refers to the percentage of documents that actually belong to A among those that are predicted as belonging to A. Both precision and recall are computed on an independent testing data. For a catalog matching to be useful, precision and recall must be above some specified minimum thresholds. We set both thresholds at 70%. *Coverage* refers to the percentage of the categories A in H_1 whose catalog matching have precision and recall above the specified thresholds. *Simplicity* is measured by the number and length of matching rules, and the percentage of terms used.

We consider two other competing methods.

The Meta-Learner method. This method was proposed in [DMDH02]. It combines the predictions of several base learners via weighted sum by assigning learner weights to base learners. Same as in [DMDH02], we used content learner (with 0.6 learner weight) and name learner (with 0.4 learner weight) as base learners. The core idea for each base learner is the same: Given a category A in H_1, it finds the "most similar" category B in H_2 for A, i.e., B maximizes some notion of similarity between A and B. We consider the Jaccard similarity:

$$\textit{Jaccard-sim}(A, B)=P(A\wedge B)/P(A\vee B),$$

where a term like $P(A\wedge B)$ is the probability that a randomly chosen document belongs to both A and B. We implemented the method described in [DMDH02] for estimating these probabilities using Naïve Bayes classifier. This approach considers only 1-to-1 matching or pair wise correspondence.

The 1-rule method. This is a degenerated version of our method by restricting the length of C4.5 rules to 1 at Step 4. In particular, if B is the attribute selected at the root node of the decision tree, the catalog matching for A will have exactly 1 matching rule $B \rightarrow A$. This method also produces 1-to-1 matching. However, unlike the Meta-learner method, it can exploit the hierarchy structure as described in Section 7.5. Comparison with this method would reveal the benefit from considering more general matching rules.

We conducted three experiments, each on a different domain. In the first experiment, we planted some known matching in the data and verify if we can find them. In the second and third experiments, we aimed to find whatever matching in the data sets, which are not known to us. To evaluate a catalog matching, a testing set is chosen so that it contains categories from *both* catalogs. Since matching rules are to predict the H_1-categories for the documents in (H_2, DB_2), we choose the testing data from the documents in (H_2, DB_2) whose H_1-categories are also known. We will explain this choice for each domain.

Domain 1: Reuters. This is a benchmark for text categorization with one flat catalog[1]. Using this catalog, we created two catalogs, (H_a, DB_a) and (H_b, DB_b), such that (H_b, DB_b) resembles the original catalog and (H_a, DB_a) resembles a new catalog obtained by applying set-operations to original categories. First, we randomly split the document collection into two disjoint sets of equal size, DB_a and DB_b. We created new categories for DB_a, as shown in Table 7-3, by applying set-operations to DB_a. For example, the documents for the new category $C_{11}=Grain \wedge Wheat$ were obtained as the intersection of *Grain*'s documents and *Wheat*'s documents in DB_a. All documents in DB_a that do not have any new category are removed. Let H_a denote the set of new categories for DB_a and let H_b be the set of original categories for DB_b. Since both H_a and H_b are flat, the Flat and Hierarchical implementations in Section 7.4 and 7.5 coincide.

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 7-3 Planted categories for the Reuters domain

	Created categories	#Document
Differentiation categories	C_{D1}=Money-fx∧¬Interest	243
	C_{D2}=Grain∧¬Wheat	126
	C_{D3}=Grain∧¬Corn	153
	C_{D4}=Ship∧¬Crude	87
	C_{D5}=Crude∧¬Net-gas	217
Intersection categories	C_{I1}=Grain∧Wheat	62
	C_{I2}=Money-fx∧Dlr	41
	C_{I3}=Grain∧Corn	38
	C_{I4}=Earn∧Acq	31
	C_{I5}=Ship∧Crude	29
Union categories	C_{U1}=Earn∨Acq	2416
	C_{U2}=Money-fx∨Acq	1138
	C_{U3}=Interest∨Wheat	320
	C_{U4}=Ship∨Wheat	243
	C_{U5}=Money-suppley∨Gnp	128
	C_{U6}=Coffee∨Gnp	121
	C_{U7}=Rubber∨Meal-feed	40
	C_{U8}=Barley∨Gas	47
	C_{U9}=Alum∨Iron-steel	52
	C_{U10}=Cotton∨Rice	52

This experiment was evaluated by 5-fold cross validation. First, the original data set in the Reuters source was split into the training set and the testing set in five different ways. For each way of splitting, two catalogs (H_a, DB_a) and (H_b, DB_b) were created using the training set as described above and were used to find the catalog matching. Every document in the testing set already has the H_b-categories. We determine their H_a-categories according to the definition in Table 7-3. If a testing document does not have any H_a-category, we discard it. The remaining testing documents have categories from both H_a and H_b.

Domain 2: Tvshow. This domain has one catalog from Yahoo![1] and one catalog from Lycos[2]. Each document is text description for an object such as a PDF file to JPEG file, with URLs linking the document to the described object. We consider only categories with at least 20 documents. To create the testing set, we consider the *mirror documents* in the two catalogs. A document in Yahoo! and a document in Lycos are mirrors of each other if their URLs link to the same object, i.e., if they describe the same object. If two documents are mirrors of each other, we consider they inherit each other's categories. Therefore, a mirror document has the categories from both Yahoo! and Lycos. We reserved all mirror documents in H_2 as the testing set and used the remaining documents as the training set.

Domain 3: Industry. In this domain, we used two different industry classification systems, one for the United Nation[3], and one for Canada[4]. Each system organizes the industry products/activities into a hierarchy catalog. In the Canada catalog, each category has a text description and example activities, and we create a single document using the description and example activities. In the UN catalog, each category has several explanatory notes, and we create one document for each explanatory note. We considered only categories at the top two levels by

[1] http://dir.yahoo.com/Entertainment/Television_Shows/

[2] http://dir.lycos.com/Arts/Television/Programs This URL is no longer valid and Lycos does not provide a replacement for this catalog.

[3] http://unstats.un.org/unsd/cr/registry/regcst.asp?Cl=17&Lg=1

[4] http://statcan.ca/english/Subjects/Standard/naics/2002/naics02-menu.htm

assigning the documents at a lower level to the ancestor categories at the top two levels. We remove the categories whose documents are less than 10. We randomly select 1/5 documents from H_2 as the testing set and manually determine their H_1-categories.

As usual, we removed stop-words and performed stemming as in Information Retrieval. We also removed words appearing in less than two documents, and documents having less than 5 words (for the industry domain, this value is 2 due to many short documents). Table 7-4 summarizes the statistics of all data sets. For Reuters, these are the average of 5-fold cross validation. The average fan out is calculated over internal nodes only. "#Training" is the number of training documents in a catalog, and "#Testing" is the number of testing documents when that catalog is taken as H_2.

Table 7-4 Statistics for three data sets

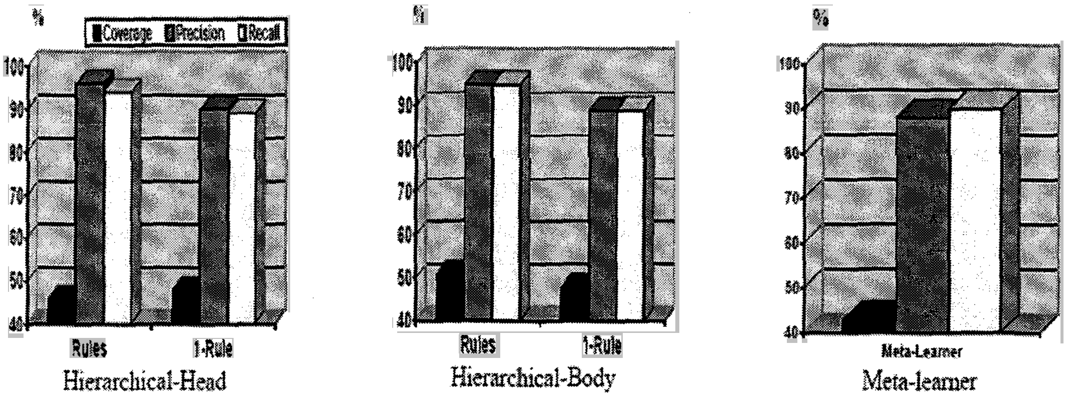
Catalogs		#Categories	Avg. Fanout	Depth	#Training	#Testing	Avg. Doc. Length	#Terms
Reuters	H_a	201	N/A	1	3469	1826	37.6	6774
	H_b	21	N/A	1	3652	1826	39.2	6996
Tvshow	Yahoo!	37	6.0	4	7953	420	8.8	6075
	Lycos	31	5.0	4	10550	558	11.3	5793
Industry	Canada	58	2.0	3	1482	370	18.3	5010
	UN	48	2.7	3	1793	448	4.6	3236

7.6.2 Performance

Figure 7-4 shows the average coverage, precision and recall over the three domains, where each domain takes one of the two catalogs as H_1 in turn. The term-allowance is set at zero.

Rules, *1-Rule* and *Meta-learner* refer to our method, 1-rule method, and Meta-learner method, respectively. *Rules* and *1-Rule* have two implementations: Hierarchical-Head and Hierarchical-Body. (Flat-Head will be examined below.) The minimum thresholds for precision and recall are 70%.

Figure 7-4 Performance comparison. Term-allowance=0%



Hierarchical-Head

Hierarchical-Body

Meta-learner

Rules and *1-Rule* have higher coverage than *Meta-learner*. This is because *Meta-learner* does not exploit the hierarchical structure. We observed that *Rules* has much higher coverage than *1-Rule* and *Meta-learner* on the Reuters domain (not shown), up to 70%. In fact, the catalog matching planted in Table 7-3 are not 1-to-1 matching. Hence, *Meta-learner* and *1-Rule* do not work well on Reuters domain. On the Tvshow and Industry domains, all three methods have similar coverage since most catalog matches are 1-to-1 correspondences. On precision and recall, *Rules* always performed better due to its capability of capturing matching general than 1-to-1 correspondences. *1-Rule* has a similar precision and recall to those of *Meta-learner*, but has much higher coverage than that of *Meta-learner*.

One of our objectives is to study which of Hierarchical-Head and Hierarchical-Body is more effective. Figure 7-4 shows that the precision and recall are similar in both cases, but Hierarchical-Body has a higher coverage consistently for all the term-allowances examined. With

Hierarchical-Body, the matching set is generated from the documents in (H_1, DB_1) where the target category A is located. This ensures that the matching set has some positive examples for A. With Hierarchical-Head, on the other hand, the target category A (or $\neg A$) is assigned to the documents in (H_2, DB_2), in which case it is possible that no or very few documents in (H_2, DB_2) are assigned with the category A and most documents are assigned the category $\neg A$. This is more so if A is at a lower level. Consequently, Hierarchical-Head more frequently fails to find a good catalog matching for A.

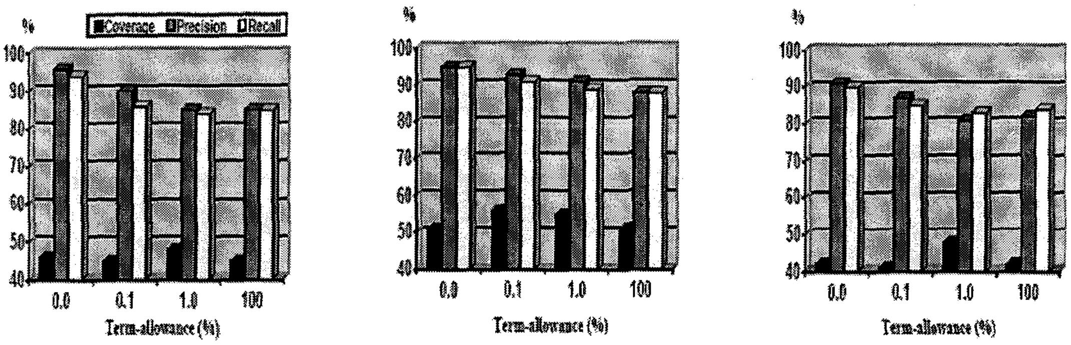
7.6.3 Effect of Terms

Another objective is to study whether terms have a major impact on catalog matching. We tested a range of term-allowances from zero to 100%, as shown in Figure 7-5. One striking trend is that a non-zero term-allowance does not increase, in fact, decreases, precision and recall in all cases. Coverage may increase a bit for up to 1% term-allowance; however, a larger term-allowance beyond 1.0% does not further increase coverage. When the term-allowance is 100%, i.e., no term is removed, the coverage is less than that for the zero term-allowance. This study confirms Hypothesis 7-2 that categories are the primary information needed for catalog matching and terms do not have a major effect. This is good news in that matchings in terms of mostly categories are more understandable to the human user.

This experiment also verified that Hierarchical-Head and Hierarchical-Body performed better than Flat-Head, demonstrating that the proposed method of handling hierarchical structures is effective in dealing with imbalanced class distribution. Indeed, the lack of positive examples in the case of Flat-Head leads to the frequent failure of finding matching rules, therefore, a low coverage.

In summary, the experimental study suggests the following: The proposed catalog matching for A can be highly accurate if the target category A is sufficiently general (i.e., has sufficient examples); Hierarchical-Body is preferred and the proposed handling of hierarchical categories is effective; matching rules are able to capture complex relationships; and catalog matching at the category level is as accurate as the catalog matching where terms are freely allowed.

Figure 7-5 Performance of the Rules approach against term-allowance



Hierarchical-Head Hierarchical-Body Flat-Head

7.6.4 Simplicity

The final objective is to study whether the proposed catalog matching is simple enough to be understood human users. A catalog matching is simple if it involves a small number of short matching rules $x_1, \ldots, x_k \rightarrow A$, with the descriptions $x_1, \ldots, x_k$ mostly given by categories instead of terms. The study in Section 7.6.3 has shown that x_i can indeed be mostly categories without losing the power of capturing matches. Therefore, we focus on the case of zero term-allowance.

We report the study for Reuters with H_l=H_a, Tvshow with H_l=Yahoo! and Industry with H_l=Canada. The results are similar in the other three cases where the other catalog in each domain is taken as H_l. Figure 7-6, Figure 7-7 and Figure 7-8 plot the number of categories A in H_l whose catalog matching is found (i.e., y-axis) vs. the number/average length of matching rules in the catalog matching (i.e., x-axis). Only the categories with precision and recall above 70% are included. For example, for Flat-Head in Figure 7-7, there are 3 categories (y-axis) in which each category has 3 mapping rules (x-axis). We see most categories have no more than 5 matching rules, and most matching rules have length no more than 3. Catalog matching of this scale is easy to understand for human users. These findings confirm that catalog matching can be both accurate and simple.

Figure 7-6 Reuters: H_I=H_a, term-allowance=0%

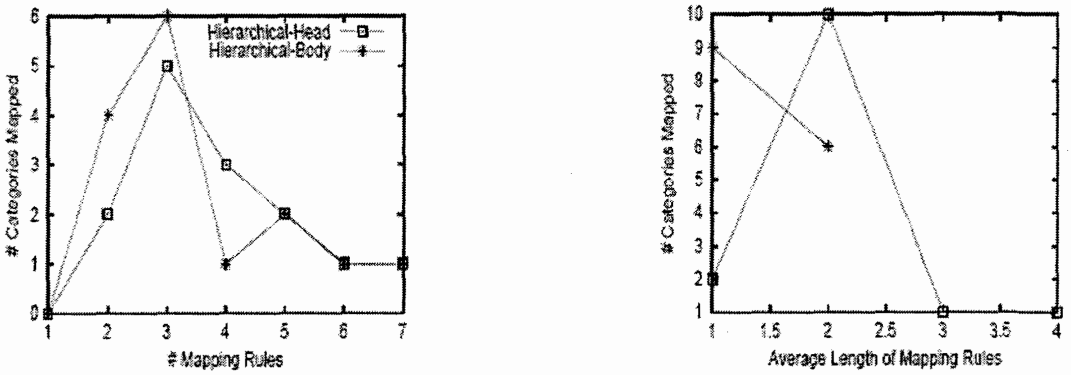


Figure 7-7 Tvshow: H_I=Yahoo!, term-allowance=0%

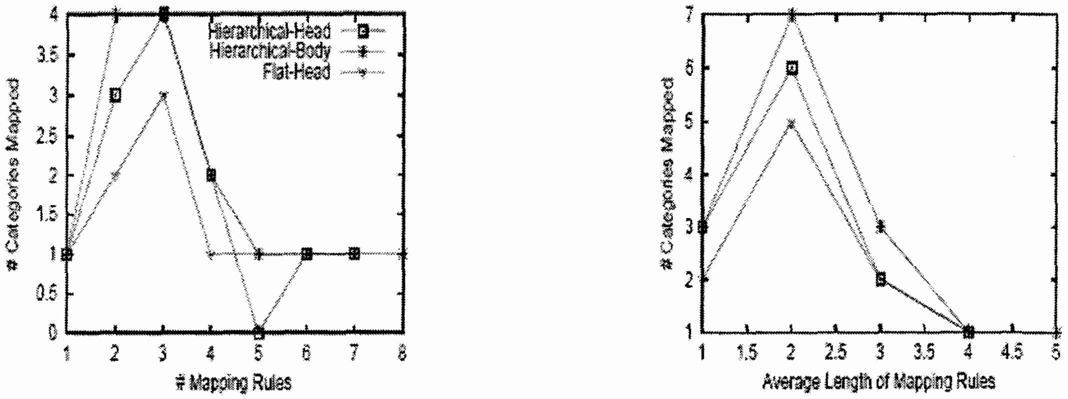
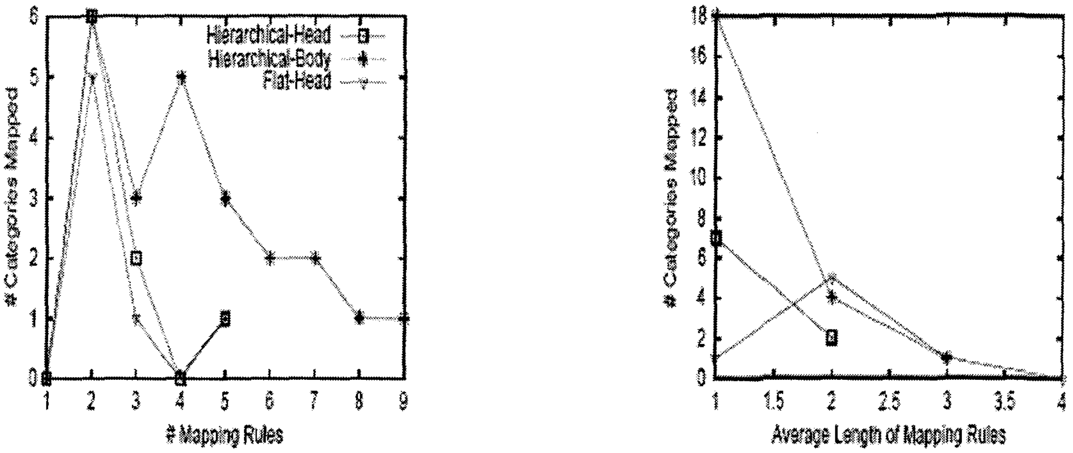


Figure 7-8 Industry: H_I=Canada, term-allowance=0%



7.7 Conclusion

In this chapter we use rules to capture category matching which can help share information across applications on the web. We examine two important issues: capturing the matches that are more than 1-to-1 correspondences and expressing the matches explicitly at the category level so that they can be understood without examining underlying data instances. We present several algorithms for finding such matches and evaluate their feasibility in terms of accuracy and simplicity. The evaluation on real life domains shows that accurate and understandable matches at the category level are possible and can be found by the proposed algorithms.

CHAPTER 8
CONCLUSION

Now we are in information era and the data accumulation rate increases fast. Mining and using knowledge from this huge amount of data become more and more important for this fast-changing world. This thesis focuses on using such data to help people making decisions. In particular, we use association rules to build interpretable decision-making systems which aim to maximize users' goals.

8.1 Summary of the Thesis

We study several important decision-making systems based on association rules across a wide range of application domains. Each domain has its own unique challenges and we propose novel algorithms to address them. Though these algorithms are different from each other, they follow the same general framework and share common themes. First, these algorithms use association rules as basic elements for system construction. Second, all the algorithms emphasize on pushing application goal down into the very beginning of model construction. Third, the algorithms take both interpretability and effectiveness as the essential requirements for decision-making systems. CHAPTER 2 gives the details on these common properties. In the following we briefly summarize the contributions we made on the studied application domains:

- In CHAPTER 3 we propose an algorithm to build decision tree from association rules, i.e., ADT. Decision trees and rule-based algorithms are two types of well-studied approaches fro classification. Association rules are rich, but lacking of a systematic method to prune over-fitting rules for classification. Decision tree induction, on the other hand, has an accuracy-driven pruning, but imposes restrictive structures on rules. The comparison motivates our work of combining the two approaches for building better classifiers. To optimize classifier's performance and improve its interpretability, we leverage the pessimistic bottom-up pruning from decision tree. The experiments results have shown that the proposed ADT algorithm not only builds more accurate classifiers, but also does this by finding more truthful structures, as indicated by the smaller size of classifiers

- We study the direct marketing problem in CHAPTER 4. In particular, we solve two major challenges. First, the classical rule-ranking criterion based on statistic probability no longer works here due to the inverse correlation between the likelihood to buy and the dollar amount to spend. To handle this issue, we propose a new criterion which combines both probability and profit information for rule ranking. The second challenge is that the historical data in direct marketing often have extremely high dimensionality and extremely low proportion of the target class. To attack this problem, we mine "focused rules" on respondents only, which dramatically reduces the rule searching space. The evaluation on the well known, large, and challenging KDD-CUP-98 task shows the effectiveness of our algorithm.
- CHAPTER 5 presents our approach in solving an interesting problem: profit mining. It is more challenge than direct marketing problem in the sense that it requires to recommend both "right" price and "right" product. We handle this issue by exploring the customers' behaviour of shopping on unavailability. Another challenge which makes this problem more interesting is how to make use of the relations among different products (e.g., TV and DVD player both are home electronics.). We propose MOA (mining on availability) technique which utilizes such product relations in rule mining and ranking. The experiments show the effectiveness of our algorithm.

- CHAPTER 6 introduces an algorithm on combining rules and SVM into one integrated classifier for the protein localization problem. The rule part exposes the main patterns which can be easily interpreted by human. The SVM part, which has shown high accuracy in many other applications, is responsible for classification involving subtle structures. This integrated model preserves the performance of the SVM and exposes simple structures in understandable rules. The algorithm proposed can be extended to integrate other classification algorithms (like neural network) as well. The experimental results on real subcellular protein localization tasks are quite promising.
- We study the catalog matching problem in CHAPTER 7. In many web-related applications catalog matching is important since it can help exchange information and speed up business decision process. To make the model understandable, we expressed catalog matching as a set of rules at the category level in which each rule defines a category in one catalog using categories from the other catalog. To make the model accurate, we also capture the matches that are more than 1-to-1 correspondences. Several variations of algorithms are studied in that chapter. The evaluation on real life domains showed that accurate and understandable matches at the category level are possible and can be found by the proposed algorithms.

8.2 Future Research Directions

With the success of using rules in many real life applications, it is worthwhile to extend its usage in other domains. Some of them are listed here.

- *Interactive model construction.* So far for each application we build "best" models in background and users have little control on the process. However, in many cases users want to guide the process. In addition, often users have some special requests on the model. For example, users may prefer to sacrifice the accuracy a little bit to have a much smaller model. Ideally, users can have control on every step and the algorithm will give the (estimated) feedback on user's decision (e.g., the estimated accuracy if user chooses to prune some rules).

- *XML document classification.* Nowadays more and more documents on web adopt the eXtensible Markup Language (XML). Thus, there arises the need to develop new techniques to classify these documents. Since XML documents are plain text, text classification algorithms can be applied here as well. But these algorithms cannot make use of the structures of XML documents. [FW02, ZA03] introduces some algorithms which assume that the presence of a particular kind of structural pattern in an XML document is related to its likelihood of belonging to a particular class (category). How to mine rules and rank them while considering both content and structure of XML documents is a big challenge when using rules for classification.
- *Spatial data prediction.* In many areas, we have collected a large quantity of spatial data, like maps, biomedical data and images. Building decision-making models on these data sets has unique challenges. First, the data itself is more complex than a typical business transactional database. Also it is very huge (e.g. terabytes or peta-bytes). Hence, the richness of rules could be a curse of rule mining. We need more efficient algorithms and data structures here. Second, spatial data tends to be highly auto-correlated (e.g. people with the same background tend to live in the same neighbourhood), which is un-true in classical statistics analysis. Integrating such correlation in rule mining and ranking is another challenge yet to be solved.

BIBLIOGRAPHY

[ADT95] R. Andrews, J. Diederich and A. Tickle. A survey and critique of techniques of extracting rules from trained artificial neural networks. In *Journal of Knowledge-Base Systems*, 8(6), pages 373-389, 1995.

[AIS93] R. Agrawal, T. Imilienski and A. Swami. Mining association rules between sets of items in large datasets. In *Proc.of 1993 ACM-SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 207–216, Washington, D.C., May, 1993.

[AMS97] K. Ali, S. Manganaris and R. Srikant. Partial classification using association rules. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD '97)*, pages 115-118, Newport Beach, CA, August, 1997.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo. Fast discovery of association rules. In *Advances in knowledge discovery and data mining*, pages 307-328, AAAI/MIT Press, 1996.

[AS01] R. Agrawal and R. Srikant. On integrating catalogs. In *Proc. of the Tenth International Conference on World Wide Web*, pages 603-612, Hong Kong, May, 2001.

[AS94] R. Agrawal and R. Srikant. Fast algorithm for mining association rules in large databases. In *Proc. of the 20th International Conference on Very Large Data Bases(VLDB'94),* pages 487-499, Santiago de Chile, Chile, September, 1994.

[Bay97] R.J. Bayardo. Brute-force mining of high-confidence classification rules. In *Proceeding of the third International Conference of Knowledge Discovery and Data Mining*, pages 123-126, Newport Beach, CA, August, 1997.

[Bay98] R.J. Bayardo. Efficient mining long patterns from databases. In *Proc. of 1998 ACM- SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 85-93, Seattle, WA, June, 1998.

[BCW00] K. P. Bennett, N. Cristianni and D. Wu. Enlarging the margins in perceptron decision trees. *Machine Learning*, Vol. 41(3), pages 295-313, December, 2000.

[BFOS84] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone. *Classification and regression trees*. Wadsworth, Belmont, CA, 1984.

[BSVW99] T. Brijs, G. Swinnen, K. Vanhoof and G. Wets. Using association rules for product assortment decisions: a case study. In *Proc. of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'99)* pages 254-260, San Diego, CA, August, 1999.

[Bur98] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121-167. Kluwer Academic Publishers, Hingham, MA, USA, June, 1998.

[CDF+00] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman and C. Yang. Finding interesting associations without support pruning. In *Proc. of 16th International Conference on Data Engineering (ICDE'2000),* pages 489-499, San Diego, CA, Feb. 2000.

[Cha00] H. Chalupsky. Ontomorph: a translation system for symbolic knowledge. In *Proc of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 471–482, Breckenridge, Colorado, April, 2000.

[CN89] P. Clark and T. Niblett . The CN2 induction algorithm. *Machine Learning Journal 3*(4), pages 261–283, 1989.

[CP34] C.J. Clopper and E.S. Pearson. The use of confidence or Fiducial limits illustrated in the case of the binomial. *Biometrika*, 26:4, 404-413. December, 1934. (Also available from http://www.jstor.org/journals/bio.html)

[CYS03] R. Chan, Q. Yang and Y. Shen. Mining high utility itemsets. In *Proc. of 2003 IEEE International Conference on Data Mining (ICDM'03)*, pages 19-26, Melbourne, FL, November, 2003.

[DFU+98] K. Diederichs, J. Freigang, S. Umhau, K. Zeth and J. Breed. Prediction by a neural network of outer membrane b-strand topology. *Protein Science,* 7:2413-2420, 1998.

[DH73] R. Duda and P. Hart. *Pattern classification and Scene analysis*. New York, John Wiley & Sons, 1937.

[DL99] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proc. of 1999 International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 43-52, San Diego, CA, August, 1999.

[DLD+04] R. Dhamankar, Y. Lee, A. Doan, A. Halevy and P. Domingos. iMAP: discovering complex mappings between database schemas. In *Proc. of International Conference on Management of Data (SIGMOD'04),* pages 383-394, Paris, France, June, 2004.

[DMDH02] A. Doan, J. Madhavan, P. Domingos and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Eleventh International Conference on World Wide Web,* pages 662-673, Honolulu, HA, May, 2002.

[Dom99] P. Domingos. Metacost: A general method for making classifiers cost sensitive. In *Proc. of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 155–164, San Diego, CA, USA, August, 1999.

[DZWL99] G. Dong, X. Zhang, L. Wong and J. Li. CAEP: Classification by aggregating emerging patterns. In *Proc. of Second International Conference on Discovery Science*, pages 30-42, Tokyo, Japan, December, 1999

[EB98] F. Eisenhaber and P. Bork. Wanted: subcellular localization of proteins based on sequences. *Trends in Cell Biology,* 8:169-170, 1998.

[FGG97] N. Friedman, D. Geiger and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131-163, 1997.

[FW02] N. Fuhr and G. Weikum. Classification and Intelligent Search on Information XML. *Bulletin of the IEEE Technical Committee on Data Engineering,* 25(1), 2002.

[GSW+03] J.L. Gardy, C. Spencer, K. Wang, M. Ester, G.E. Tusnady, I. Simon, S. Hua, K. deFays, C. Lambert, K. Nakai and F.S.L. Brinkman. PSORT-B: improving protein subcellular localization prediction for Gram-negative bacteria. *Nucleic Acids Research,* 31(13):3613-3617, 2003.

[HC03] B. He and K. C. Chang. Statistical schema matching across web query interfaces. In *Proc. of 2003 AC- SIGMOD International Conference on Management of Data (SIGMOD'03),* pages 217-228, San Diego, CA, June, 2003.

[HCH04] B. He, K. C. Chang and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *Proc. of the Tenth International Conference on Knowledge Discovery and Data Mining (KDD'04),* pages148-157, Seattle, WA, August, 2004.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. of the 21st International Conference on Very Large Data Base(VLDB'95),* pages 420-431, Zurich, Switzerland, September, 1995.

[HS01] S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics,* 17(8):721-728, 2001.

[ITH01] R. Ichise, H. Takeda and S. Honiden. Rule induction for concept hierarchy alignment. In *Proc. of the Second Workshop on OntologyLearning at the 17th International Conference on AI (IJCAI'01),* Seattle, WA, August, 2001.

[JMF+01] I. Jacoboni, P. Martelli, P. Fariselli, V. De Pinto and R. Casadio. Prediction of the transmembrane regions of β-barrel membrane proteins with a neural network-based predictor. *Protein Science,* 10, pages 779-787, 2001.

[Joa98a] T. Joachims. Making large-scale SVM learning practical. *Advances in Kernel Methods – Support Vector Learning.* MIT Press, 1998.

[Joa98b] T. Joachims. Text categorization with support vector machines: learning with many relevant features. *In Proceedings of the European Conference on Machine Learning.* pages 137-142, Springer, 1998.

[KDD98-data] KDD98 (1998a). The kdd-cup-98 dataset. In *http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html.*

[KDD98-result] KDD98 (1998b). The kdd-cup-98 result. In *http://www.kdnuggets.com/meetings/kdd98/kdd-cup-98.html.*

[KJL+94] R. Kohavi, G. John, R. Long, D. Manley and K. Pfleger. *MLC++: a machine learning library in C++.* Tools with artificial Intelligence, pages 740-743, 1994.

[LEN02] C. Leslie, E. Eskin and W. S. Nobel. The spectrum kernel: a string kernel for SVM protein classification. In *Proc. of Pacific Symposuim on Biocomputing,* pages 564-575, Lihue, HA, January, 2002.

[LG01] M. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proc. of the Fouteenth International Florida Artificial Intelligence Research Society Conference*, pages 305-309, Key west, FL, May, 2001.

[LHM98] B. Liu, W. Hsu and Y. Ma. Integrating classification and association rule mining. In *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 80-86, New York, NY, August, 1998.

[LL98] C. Ling and C. Li (1998). Data mining for direct marketing: problems and solutions. In *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 73-79, New York, NY, August, 1998.

[LMW00] B. Liu, Y. Ma and C. Wong. Improving an association rule based classifier. In *Principles of Data Mining and Knowledge Discovery*, pages 504–509, 2000.

[MFKC02] P. Martelli, P. Fariseli, A. Krogh and R. Casadio. A sequence-profile-based HMM for predicting and discriminating β barrel membrane proteins. *Bioinformatics,* 18(1), S46-S53, 2002.

[MM96] C.J. Merz and P. Murphy. *UCI repository of machine learning databases,* 1996 (http://www.cs.uci.edu/ ˜mlearn/MLRepository.html).

[MP94] P. Murphy and M. Pazzani. Exploring the decision forest: an empirical investigation of Occam's razor in decision tree induction. *Journal of AI Research,* 1:257-275, 1994.

[MS96] B. Masand and G. P. Shapiro. A comparison of approaches for maximizing business payoff of prediction models. In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 195-201, Portland, OR, August, 1996.

[MW99] D. Meretakis and B. Wuthrich. Extending naive bayes classifiers using long itemsets. In *Proc. of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD'99),* pages165-174, San Diego, CA, August, 1999.

[MWJ99] P. Mitra, G. Wiederhold and J. Jannink. Semi-automatic integration of knowledge sources. In *Proc. of Fusion 1999*, pages 572-581, Sunnyvale, USA, July, 1999.

[NM00] N. Noy and M. Musen. PROMPT: algorithm and tool for automated ontology merging and alignment. In *Proc. of the 17th National Conference on Artificial Intellegence(AAAI/IAAI'00)*, pages 450–455, Austin, TX, August, 2000.

[OF01] B. Omelayenko and D. Fensel. An analysis of B2B catalogue integration problems. In *Proc. of the 3rd International Conference on Enterprise Information Systems (ICEIS'01)*, pages 945-952, Setubal, Portugal, July, 2001.

[PAZ02] E. Pednault, N. Abe and B. Zadrozny. Sequential cost-sensitive decision making with reinforcement learning. In *Proc. of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD '02),* pages 259-268, Edmonton, Canada, July, 2002.

[PCY95] J.S. Part, M.S. Chen and P.S. Yu. An efficient hash-based algorithm for mining association rules. In *Proc. of 1995 ACM-SIGMOD International Conference on Management of Data (SIGMOD'95),* pages 175-186, San Jose, CA, May, 1995.

[Pro00] F. Provost. Machine learning from imbalanced data sets 101. In *AAAI Workshop on Learning from Imbalanced Data Sets. Invited paper for the AAAI Workshop on Imbalanced Data Sets,* 2000.

[Qui86] J. Quinlan. Induction of decision tree. In *Machine Learning,* pages 81-106, 1986.

[Qui93] J. Quinlan. C4.5: *Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[RB01] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases,* 10(4):334–350, 2001.

[RH98] A. Reinhardt and T. Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research,* 26(9):2230-2236, 1998.

[RV97] P. Resnick and H.R. Varian, Eds. CACM special issue on recommender systems. *Communications of the ACM,* 40(3):56-58, 1997.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21st International Conference on Very large Data Bases (VLDB'95),* pages 407-419, Zurich, Switzerland, September 1995.

[SCG03] S. Sarawagi, S. Chakrabarti and S. Godbole. Crosstraining: learning probabilistic mappings between topics. In *Proc. of the Ninth International Conference on Knowledge Discovery and Data Mining (KDD '03),* Washing DC, USA, August, 2003.

[Sch93] J. Schlimmer. Efficiently inducing determinations: a complete and systematic search algorithm that uses optimal pruning. International Conference on Machine Learning, pages 284-290, Amherst, MA, June, 1993.

[SCW+03] R. She, F. Chen, K. Wang, M. Ester, J.L. Gardy and F. Brinkman. Frequent subsequence-based prediction of outer membrane proteins. In *Proc. of the Ninth International Conference on Knowledge Discovery and Data Mining (KDD '03),* Washing DC, USA, August, 2003.

[SM01] G. Stumme and A. Madche. Fca-merge: Bottom-up merging of ontologies. In *Proc. of the 17th International Joint Conference on Artificial Intelligence,* Seattle, USA, August, 2001.

[SON95] A. Savasere, E. Omiecinski and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21st International Conference on Very large Data Bases (VLDB'95),* pages 432-444, Zurich, Switzerland, September, 1995.

[SON98] A. Savasere, E. Omiecinski and S. Navathe . Mining for strong negative associations in a large database of customer transactions. In *Proc. of the Fourteenth International Conference on DataEngineering (ICDE'98),* pages 494-502, Orlando, FL, February, 1998.

[ST96] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems? *IEEE Transactions on Knowledge and Data Engineering,* 8(6):970-974, 1996.

[Str95] L. Stryer. *Biochemistry. 4th Edition.* W.H. Freeman, NY, 1995.

[TKS00] P. N. Tan, V. Kumar and J. Srivastav. Indirect association: mining higher order dependencies in data. In *Proc. of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00),* pages 632-637, Lyon, France, September, 2000.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, NY, 1995.

[Ver02] J.P. Vert. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. In *Proc. of the Pacific Symposium on Biocomputing,* pages 649-660, Lihue, HA, January, 2002.

[WCM+94] J. Wang, G. Chirn, T. Marr, B.Shapiro, D. Shasha and K. Zhang. Combinatorial pattern discovery for scientific data: some preliminary results. In *Proc. of 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD'94),* pages 115-125, Minneapolis, MN, May, 1994.

[WFW03] R. Wong, A. Fu and K. Wang. MPIS: Maximal-profit item selection with cross-selling considerations. In *Proc. of* 2003 *IEEE International Conference on Data Mining (ICDM),* pages 371-378, Melbourne, FL, November, 2003.

[WHH00] K. Wang, Y. He and J. Han. Mining frequent itemsets using support constraints. In *Proc. of the 26th International Conference on Very large Data Bases (VLDB'00),* pages 43-52, Cairo, Egypt, September, 2000.

[WS02] K. Wang and M. Y. Su. Item selection by hub-authority profit ranking. In *Proc. of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 652–657, Edmonton, Canada, July, 2002.

[WZH00] K. Wang, S. Zhou and Y. He. Growing decision tree on support-less association rules. . In *Proc. of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD'00*), pages 265-269, Boston, MA, August, 2000.

[WZH02] K. Wang, S. Zhou and J. Han. Profit mining: from patterns to actions. In *Proc. of the Eighth International Conference on Extending Database Technology (EDBT'02),* pages 70-87, Prague, Czech Republic, March, 2002.

[WZL99] K. Wang, S. Zhou and S.C. Liew. Building hierarchical classifiers using class proximity. In *Proc. of the 25th International Conference on Very large Data Bases (VLDB'99),* pages 363-374, Edinburgh, UK, September, 1999.

[WZYY03] K. Wang, S. Zhou, J.M.S. Yeung and Q. Yang. Mining customer value: from association rules to direct marketing. In *Proc. of 19th International Conference on Data Engineering* (*ICDE'03),* pages 738-740, Bangalore, India, March, 2003.

[WZYY05] K. Wang, S. Zhou, J.M.S. Yeung and Q. Yang. Mining customer value: from association rules to direct marketing. In *Journal of Data Mining and Knowledge Discovery,* 11(1):57-80, July, 2005.

[YHB04] H. Yao, H.J. Hamilton and C.J. Butz. A foundational approach for mining itemset utilities from databases. In *Proc. of 2004 SIAM International Conference on Data Mining (SDM'04),* pages 482-486, Lake Buena, FL, April, 2004.

[YP97] Y. Yang and J.O. Pederson. A comparative study on feature selection in text categorization. In *Proc. of 14th International Conference on Machine Learning(ICML'97)*, pages 412-420, Nashville, TN, July, 1997.

[Yua99] Z. Yuan. Prediction of protein in subcellular locations using Markov chain models. *FEBS Letter*, 451:23-25, 1999.

[ZA03] M. J. Zaki and C. Aggarwal. XRULES: An effective structural classifier for XML data. In *Proc. of the 9th International Conference on Knowledge Discovery and Data Mining(KDD'03)*, pages 316-325, Washington, DC, August, 2003.

[ZC02] Z. Zhou and Z. Chen. Hybrid decision tree. *Knowledge-based systems*, 15(8): 515-528, Elsevier, 2002.

[ZE01] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proc. of the seventh international conference on Knowledge discovery and data mining (KDD'01)*, pages 204-213, San Francisco, CA, August, 2001.

[ZL04] D. Zhang and W. Lee. Web taxonomy integration using support vector machines. In *Proc. of the 13th International Conference on World Wide Web(WWW'04)*, pages 472-481,New York, NY, May, 2004.

[ZW05] S. Zhou and K. Wang. Localization Site Prediction for Membrane Proteins by Integrating Rule and SVM Classification. In *IEEE Transactions on Knowledge and Data Engineering,* 17(12):1694-1705, December, 2005.