**University of Alberta**

# ASSOCIATIVE CLASSIFIERS: IMPROVEMENTS AND POTENTIAL

by

**Catalina Maria-Luiza Antonie**

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2008

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

*Your file Votre référence*
*ISBN: 978-0-494-46273-7*
*Our file Notre référence*
*ISBN: 978-0-494-46273-7*

# Abstract

Classification of objects in pre-defined classes is an important task in many applications. A classification model is learned from a set of objects with their classes. When a new object has to be classified the model is employed and one or more class labels are predicted for the new object. In this research we focus on associative classifiers, classification systems that use association rules in building their model. These classification systems learn patterns from data that are associated with the pre-defined classes. Associative classifier models have been recently proposed in the literature, thus their development is in an early stage. We investigate the current issues that influence their performance and propose several solutions to overcome them. We study the performance of our classification model on real-life applications and we show that associative classifiers are competitive classification systems.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

With the continuous growth of digital collections, the research field of Knowledge Discovery and Data Mining has become very important. There is great demand for the data mining algorithms when discovery of useful knowledge is needed. Data Mining algorithms can be used in various domains including market basket analysis, medical applications, business management, space exploration and financial data analysis.

Classification is one of the main tasks studied in data mining. A classifier is a system that assigns, or predicts, one or more class labels for a given object. Classification has multiple applications and has already been applied in many areas such as text categorization, medical analysis and space exploration. Besides decision trees [FB91, Qui93], Bayesian classifiers [Lew98], neural networks [HKP91, Bis95] and support vector machines [Vap95], associative classifiers started attracting attention in the last decade [LHM98, AZC01, LHP01, ZA02].

Associative classifiers are classification systems that build their rule-based model using association rule mining. An important advantage that these classification systems bring is that they are able to examine several features at a time using association rule mining, while other state-of-the-art methods, like decision trees or naïve Bayesian classifiers, consider that each feature is independent of one another. However, in real-life applications, the independence assumption is not necessarily true, and correlations and co-occurrence of features can be very important. In addition, the associative classifiers can handle a large number of features, while other classification systems do not work well for high dimensional data. The associative classification systems proved to be competitive with other techniques reported in the literature.

The associative classifiers are models that can be read, understood, modified by humans and thus can be manually enriched with domain knowledge. Given the readability of the associative classifiers, they are especially suited to applications where the model may assist domain experts in their decisions. The medical field is a good example where such applications may appear. Let us consider an example where a physician has to examine a patient. There is a considerable amount of information associated with the patient (e.g., personal data and medical tests). A classification system can assist the physician in this process. The system can predict if the patient is likely to have a certain disease or present incompatibility with some treatments. Considering the output of the classification model, the physician can make a better decision on the treatment recommended for this patient. Given the transparency of associative classifiers, a health practitioner can understand how the classification model reached its decision.

The classification systems based on association rules have three main phases. First, classification rules are generated. Then, redundant or less interesting rules are pruned. The last stage deals with selecting the rules for the classification process. In this dissertation we show that associative classifiers can be improved on all fronts: rule generation, rule pruning, and rule selection, to outperform other classifiers on real datasets.

## 1.1 Problem Definition

Many classification systems exist. Nevertheless, the research on classification is not concluded. Research in classification carries on and there are many reasons for this. There is no classification system that performs best over the entire range of real-life applications. Some of these systems take a long time to develop a model or to classify new objects. Many of the classifiers produce a model that is hard to understand. Another issue that may interfere with their performance is the class imbalance that characterizes many applications where some classes may have just a few representatives. Thus, while classification is a well studied subject, there are still many open problems.

The associative classifier is a new type of classification system that has recently been proposed. One reason that the associative classifier systems attract researchers is that they are built upon association rule mining, which is a well-known and well-studied problem in the Data Mining community. This allows the fast discovery of classification rules, even from very large datasets. Another attractive point for these classifiers is that association rule mining allows them to examine several features at a time. In addition, the model that the associative classifiers construct is a rule-based one, which makes it easy to understand and is readable to humans. This allows experts in the application at hand to read, modify and improve the classification system with relative ease. The first classification systems were rule-based models. The difference is that those rules were not created automatically from data, but created by experts who were very familiar with the data. Given that associative classifiers are rule-based systems, there is the opportunity of incorporating additional domain knowledge after the model was created. Suppose that some domain knowledge was not expressed in the data given for the training set (it was not available in the training phase or it was not available for mining due to privacy issues). Missing data can be incorporated in the associative classifiers at the classification stage by using expert-created rules. Moreover, transparent systems are more trusted by end users in certain domains, such as the medical field.

Although the associative classifiers started to attract attention, their development is at an early stage. Some of the existing classifiers have been tested only on small problems. Others have a very naïve way of classifying a new object once the model has been built. In addition, these classifiers will produce a large number of rules under certain conditions. These problems are the very focus of this dissertation.

## 1.2 Approach to the Problem

In this section, we describe the issues we tackle in this dissertation. In building an associative classification model that is accurate and human readable we have two main goals. The first objective is to improve the performance of the classifier. To achieve it, we incorporate new types of rules in the model and develop a better scoring scheme in the classification

phase. The second goal is to reduce the number of rules in the model, so that human interaction is possible. Although the rules in the associative classification model are readable, an overwhelming number of rules may make the system hard to deal with. That is why reducing the number of rules in the model represents an important objective for this work. In addition, fewer rules will make the classification stage faster. However, these two goals are antagonistic. By incorporating new types of rules we will likely increase the number of generated classification rules. On the other hand, by cutting down the number of rules, we may reduce the performance of the classifier.

### 1.2.1 Improving the Performance

In this section we present the solutions proposed in this dissertation for improving the performance of associative classifiers. Two of the proposed solutions deal with incorporating new types of association rules in the associative classifiers, while the third one tackles the use of rules in the classification stage.

#### Negative Association Rules

All the traditional association rule mining algorithms were developed to find positive associations between items. By positive associations we refer to associations between items existing in transactions (e.g., items bought). How about associations of the type: "customers that buy Coke *do not* buy Pepsi" or "customers that buy juice *do not* buy bottled water"? In addition to the positive associations, the negative associations can provide valuable information in devising marketing strategies or making a classification decision. Interestingly, very few researchers have focused on negative association rules due to the difficulty in discovering these rules. In Chapter 3 we show that absence of features can be a good discriminator in the case of associative classifiers.

#### Association Rules with Re-occurring Items

Most of the existing association-rule approaches consider that an item appears or does not appear in a certain transaction (i.e., they work only with binary transactions). They do not take into account the re-occurrence of items in transactions. However, there are applications where the re-occurrence of objects may play an important role. Let us consider that we have a collection of documents. In some documents the phrase "association rules" is repeated many times. Not surprisingly, by reading those documents we may find they are talking about techniques related to association rules. In the collection there are also some documents that rarely use the phrase "association rules" and they are describing data mining related issues. Another application where this type of rule may fit is image classification, where visual features may be repeating and their repetition could be a good classification indicator. In Chapter 4 we argue that by incorporating association rules with re-occurring items in the classification model, the classification accuracy is improved for some applications.

#### Use of Rules in the Classification Stage

Once developed, the purpose of a classification system is to classify new instances. In the case of associative classifiers, this step deals with using the rules to categorize a new example. To classify an instance, an associative classifier proceeds in three steps. First, it

determines which of its rules apply to the object. Then, it selects a subset of the applicable rules (possibly all of them) based on some measure of their "strength" or precedence. Finally, if it chooses more than one rule, it combines the class predictions of all the selected rules to produce a final classification. A good scoring scheme is essential to the performance of an associative classifier. Devising a scoring scheme is not an easy task given the factors that have to be taken into consideration (the number of rules that apply and their strength, combining their predictions, etc.). We investigate this problem in Chapter 5.

### 1.2.2 Reducing the Number of Rules

In this dissertation we argue that reducing the number of rules is important. Not only that the model becomes more readable, but other advantages may be reached with fewer association rules. The classification time is reduced using a smaller model. In addition, redundant rules that may hinder classifiers' performance are eliminated. We investigate the reduction of the rule redundancy in the model.

#### Closed and Maximal Itemsets

Association rule mining can lead to a very large number of generated rules. A considerable amount of redundancy is found in the typical frequent itemset collection. The redundancy in the frequent itemsets is translated into a very large set of rules discovered. To deal with the problem of redundancy, the closed and maximal patterns have been introduced. These types of patterns are a concise representation of frequent itemsets. Closed and maximal itemsets can be a few orders of magnitude smaller than the frequent itemsets. We investigate the effect of reducing rule redundancy on associative classifiers in Chapter 6.

## 1.3 Contributions of this Thesis

The contributions of this thesis are in two main directions. First, we focus on improving the performance of associative classifier by introducing new types of classification rules and by proposing a novel technique in the classification stage. Second, we investigate and propose solutions for reducing the number of classification rules. For this, we propose a pruning technique where rules that cause more errors than help in the classification process are eliminated. In addition, we investigate and discuss the redundancy reduction in the classification rule set. Figure 1.1 shows how the contributions of this thesis are incorporated in the general associative classification framework. The highlighted boxes represent areas where this thesis brings contributions.

### 1.3.1 Associative Classifiers Using Negative Association Rules

We introduce a new algorithm to generate positive and negative associations discovered in transactional data. The interestingness measure that our algorithm relies on is the correlation coefficient. We demonstrated the potential of strong positive and negative correlated rules in the classification context. The results of the classification show that a much smaller set of positive and negative association rules can perform similar to or outperform existing categorization systems. Our findings on associative classifiers using negative association rules are published in [AZ04a, AZ04b].

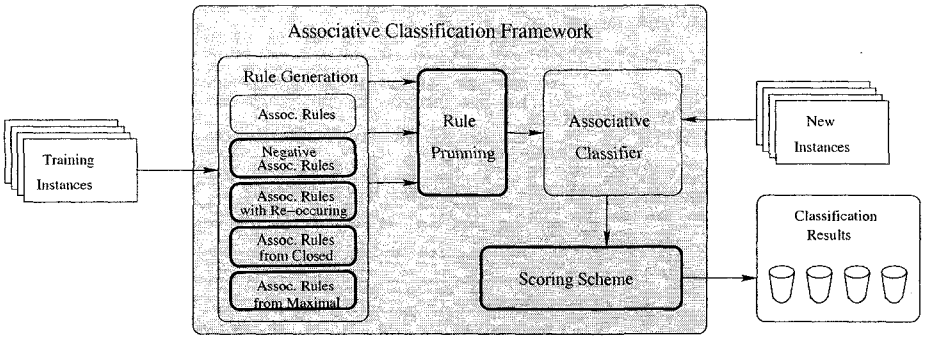


Figure 1.1: Associative Classification Framework

### 1.3.2 Associative Classifiers Using Association Rules with Re-occuring Items

We introduce the idea of combining associative classification and mining frequent itemsets with re-occurring items. We combine these two and we propose a new approach of associative classification with re-occuring items (ACRI). By comparing ACRI with existing associative classifiers we show that considering repetitions of observed features is beneficial. Moreover, the accuracy of ACRI is less sensitive to the parameters used in associative classifiers, while most associative classifiers are typically very sensitive to these parameters. Our results on combining associative classification and mining frequent itemsets with re-occurring items are published in [RSZA05].

### 1.3.3 Two Stage Architecture

Rule-based classifiers use predefined weighted voting schemes to combine the class predictions of the applicable rules. By contrast, the methods proposed in this thesis automatically learn the scoring scheme. We achieve this by developing a two-stage system, with a layer of feature definitions interposed between the output of the first learning model (i.e., the association rule-based model) and the input of the second. Our two stage classification system (2SARC) equals or outperforms state-of-the-art classifiers under rigorous statistical analysis. Our findings are published in [AZH06].

### 1.3.4 Associative Classifiers Using Association Rules from Closed and Maximal Itemsets

We investigate the performance of associative classifiers when the classification rules are generated from frequent, closed and maximal itemsets. We show that maximal itemsets substantially reduce the number of classification rules without jeopardizing the accuracy of the classifier. Our extensive analysis demonstrates that the performance remains stable and even improves for some applications. Our analysis using cost curves also provides recommendations on when it is appropriate to remove redundancy in frequent itemsets. Based on our thorough analysis we are confident that any investigation of associative classifiers should consider first and foremost classification rules generated from maximal patterns. This work was submitted to an international journal for review.

Overall, the contributions of this thesis are important because they advance the state-of-the-art in associative classifiers. We show new ways in which associative classifiers can be used and that the associative classifiers can be competitive classification systems. In addition, we highlight new directions on associative classifiers research. The problems investigated in this thesis and their proposed solutions constitute significant progress toward improving the quality of associative classifiers.

## 1.4 Outline

This thesis is organized as follows. Chapter 2 introduces the background notions necessary throughout this thesis and presents the related work relevant to our research. Chapter 3 presents the integration of negative association rules with associative classifiers. In Chapter 4 we propose and study combining associative classification and frequent itemsets with re-occurring items. Chapter 5 introduces our two stage architecture for the classification stage. In Chapter 6 we investigate the performance of associative classifiers when the classification rules are generated from frequent, closed and maximal itemsets. Chapter 7 concludes our dissertation, discusses several limitations and proposes future research directions.

# Chapter 2

# Background and Related Works

This chapter introduces the background notions necessary throughout this dissertation and presents the related work relevant to our research. As stated in the introduction the main scope of this dissertation is to advance the state-of-the-art of associative classifiers. These systems integrate association rules with the classification process. Details on these two main components are presented along with some related work in these domains.

## 2.1 Association Rules

Mining association rules from data is the process of finding interesting relationships or associations that exist between objects/items in a collection of data. The dataset has to be modeled in a transactional format and association rule mining is the data mining process that discovers the associations between items. Association rules have been extensively studied in the literature. The efficient discovery of such rules has been a major focus in the data mining research community, given their popularity in market basket analysis and other application domains.

A typical example of association rule mining application is market basket analysis. This process analyzes the behaviour of customers and discovers associations between items bought in the store (i.e., products placed in the customers' shopping baskets). The discovery of interesting patterns in this collection of data can lead to important marketing and strategic management decisions. For instance, if a customer buys bread, what is the probability that he/she will buy milk as well? Depending on the probability of such an association, marketing personnel can develop better planning of the shelf space in the store or they can base their discount strategies on such correlations found in data. Several efficient algorithms exist to discover this type of associations in the data. From the original *apriori* algorithm [AIS93] there have been a remarkable number of variants and improvements [BMUT97, Zak99, HPY00, GZ03, ZEH05]. All these algorithms discover the same associations in the data given the same parameters, their main contribution being the improvement in the efficiency of the process.

Formally, association rules are defined as follows. Let $\mathcal{I} = \{i_1, i_2, ...i_m\}$ be a set of items. Let $\mathcal{D}$ be a set of transactions, where each transaction $T$ is a set of items such that $T \subseteq \mathcal{I}$. Each transaction is associated with a unique identifier $TID$. A transaction $T$ is said to contain $X$, a set of items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form "$X \Rightarrow Y$", where $X \subseteq \mathcal{I}, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has a *support* $s$ in the transaction set $\mathcal{D}$ if $s\%$ of the transactions in $\mathcal{D}$ contain $X \cup Y$. In other

words, the support of the rule is the probability that $X$ and $Y$ hold together among all the possible presented cases. It is said that the rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with *confidence* $c$ if $c\%$ of transactions in $\mathcal{D}$ that contain $X$ also contain $Y$. In other words, the confidence of the rule is the conditional probability that the consequent $Y$ is true under the condition of the antecedent $X$. The problem of discovering all association rules from a set of transactions $\mathcal{D}$ consists of generating the rules that have a *support* and *confidence* greater than given thresholds. These rules are called *strong rules* and represent interesting patterns in data.

## 2.2 Classification

Classification is an important task in many applications. A classifier is a system that assigns, or predicts, one or more class labels for a given object. It can be created by using a learning algorithm to construct a model from a training set of objects whose classifications are known.

The main idea of the classification task is to discover interesting patterns in a training set of data that will be used later in classifying new objects. Classification has multiple applications and has already been applied in many areas such as text categorization, medical analysis and space exploration. Besides decision trees [FB91, Qui93], Bayesian classifier [Lew98], neural networks [HKP91, Bis95] and support vector machines [Vap95], classification based on association rule mining started attracting attention in recent years [LHM98, LHP01]. There are three steps in the classification process: the learning phase also known as training phase, the testing phase and the classification stage. In the training phase a classification model is developed based on a training set, where all the instances are given with their associated class label. The main purpose of the testing phase is to evaluate and eventually validate the model that was built in the previous step. Finally, the third phase uses the classification model to categorize new objects that have no class labels attached. Figure 2.1 shows an overview of the classification process.

### 2.2.1 Training and Testing Sets

Given a data collection for classification purposes, we have a training and a testing set. These datasets have associated to each example one or a set of classes. These sets (training and testing) are either generated thorough cross-validation or are given apriori with the application to solve. Cross-validation (or N-fold cross validation) is the process where the dataset is randomly split in N folds. In each fold the class distribution is approximately preserved as in the original dataset. Classification methods build a classification model from the training set. The testing set is typically used for evaluating the performance of the built model. It is very important that the objects in the testing set have not been used in any way in the training process. In the case of cross-validation the classification model is learned from N-1 folds and tested on the remaining fold. The most common value for N is 10. Extensive tests with several classification systems showed that this is the best value to get a good estimate about the performance of the classifier [WF05].

### 2.2.2 Single and Multiple Label Classification

Let us assume that we have to learn a classification model on a dataset with $m$ distinct classes $\{C_1, C_2...C_m\}$. The case when only one class has to be assigned to an object is

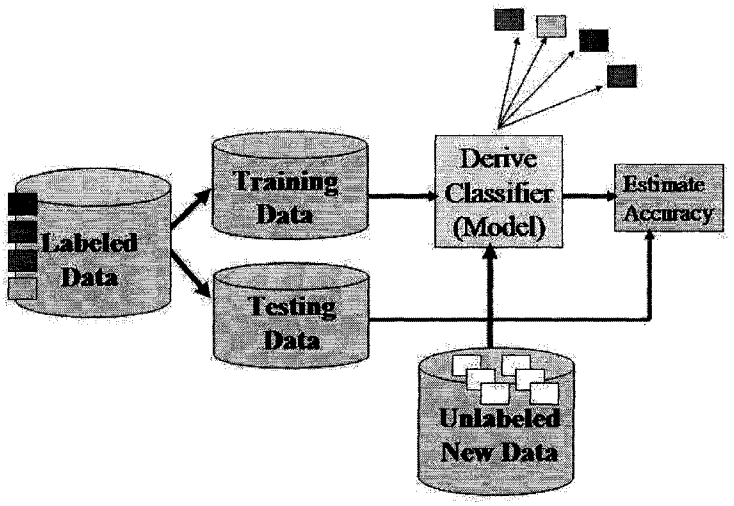


Figure 2.1: Overview of the classification process

called single-label classification, while the case where any number of classes from 1 to $m$ can be assigned to an object is called multi-label classification.

The single-label case is the standard classification task studied in machine learning, and a variety of test datasets and systems are available for comparison. The UCI datasets [BM98] contain tasks requiring single label classification and are detailed in Section 2.6.1. For instance, the *Diabetes* dataset from the UCI repository deals with predicting if a patient has diabetes. Each patient can belong to only one class and a new patient will be predicted as being either diabetic or not. The single-label classification is not necessarily a two-class problem (also called binary classification). The *Iris* dataset, also from the UCI repository, is a three-class problem, but the objects are predicted as belonging to only one class, as the classes are mutually exclusive.

In multiple label classification, an object can be assigned to several of the classes simultaneously, i.e. the classes are not mutually exclusive. The standard testbed for this task is classifying news articles into subject categories, where it is necessary for some news articles to be assigned to multiple categories. For example, an article about selling a sports franchise should be put into at least two categories, "sports" and "business". The Reuters dataset [Reu08] is a typical collection used for multi-label classification and is detailed in Section 2.6.1. A multi-label classification problem can be transformed into $m$ independent classification problems of binary single-label classification with categories $\{C_i, \neg C_i\}$ for $i = 1..m$.

### 2.2.3 Class Imbalance

Although a plethora of classification algorithms has been developed, there are several aspects that influence and hinder their performances when applied to real-life applications. One of these issues that may interfere with the classification process is *class imbalance*.

This problem arises when some classes have a large number of examples in comparison with others. Class imbalance is very common in real-life applications when the events of interest are usually in a small number. As an example, let us consider a fraud detection problem in credit card usage. Although it is of great importance to detect and understand the patterns of fraud occurrence, it is a difficult task for the classifications systems, because the class of interest (i.e. fraudulent credit card usage) is heavily under-represented. The collected data in fraud detection task can be overwhelming. However, the interesting examples are infrequent, thus called the minority class. Other applications with similar characteristics are oil spills and cancer detection. Most of the existing classification algorithms have been developed and tested on balanced class problems. That is why their performance suffers when rare classes exist among dominant classes. Although the class imbalance problem is not currently solved, nor completely understood, in the past few years there have been some methods proposed to deal with it, such as over-sampling, under-sampling or creating new systems to address this issue. For a survey of these methods see [JS02].

## 2.3 Associative Classifiers

Association rule-based classifiers have recently emerged as competitive classification systems [AZC01, LHP01, LHM98]. These classification systems consist of a set of association rules, with each rule predicting that an object belongs to a specific class if it has certain properties. The rules are discovered using an association rule mining algorithm [AIS93]. The association rule mining problem has been thoroughly studied in the data mining community [BMUT97, Zak99, GZ03, HPY00, ZEH05], thus there are several fast algorithms for discovering these types of rules. An attractive characteristic that associative classifiers possess is their readability (the model created is rule-based thus being easily read and interpreted by domain experts). However, the number of classification rules they generate is quite large.

The first reference to using association rules as classification rules is credited to [Bay97], while the first classifier using these association rules was CBA introduced in [LHM98] and later improved by CMAR [LHP01] and ARC [ZA02]. Given a training set modeled with transactions where each transaction contains all features of an object together with the class label of the object, we can constrain the mining process to generate association rules that always have as consequent a class label. In other words, the problem consists of finding the subset of strong association rules of the form $X \Rightarrow C$ where $C$ is a class label and $X$ is a conjunction of features.

The main steps in building an associative classifier from a training set are the following:

1. *Generate the set of association rules from the training set.* In this phase association rules of the form *set_of_features* $\Rightarrow$ *class_label* are discovered by using a mining algorithm. This step of the algorithm can be completed in two ways:

   - Using an association rule mining algorithm, generate all the strong association rules. Once these are generated, filter them so that only the rules of interest are kept (those that have as consequent a class label and their antecedent is composed of features other than class labels).
   - Modifying an association rule mining algorithm by imposing a constraint during rule generation. The constraint is that the association rules to have always as consequent a class label. In this way, a more efficient algorithm is employed

| Transaction ID | Age | Income | Credit (class label) |
|---|---|---|---|
| 1 | <25 | low | bad |
| 2 | 25..40 | low | bad |
| 3 | 25..40 | medium | good |
| 4 | >40 | low | good |
| 5 | <25 | high | good |
| 6 | 25..40 | medium | good |
| 7 | <25 | high | good |
| 8 | 25..40 | high | good |

Table 2.1: A transactional dataset

since less candidate items are generated. All candidate itemsets generated contain a class label and only the association rules with a class label on the right hand side are generated.

2. *Build the Classifier.* In the previous phase a large set of association rules can be generated, especially when a low support is given. Rule pruning techniques are employed to discover the best set of rules that can cover the training set and weed out those rules that may introduce errors or are overfitting in the classification stage. Different pruning strategies for associative classifiers have been proposed in the literature and they will be presented later on in this section together with the associative classifier with which they were proposed.

3. *Classification Phase.* At this level a system that can make a prediction for a new object is built. The challenge here is how to make use of the set of rules from the previous phase to give a good prediction. Using the rules to classify a new object means to have a good way of selecting one or more rules to participate in the prediction process.

Let us consider the following banking application to illustrate how an associative classifier works.

**Example Associative Classifier.** Let us consider the transactional dataset presented in Table 2.1. In the first stage of the associative classification a set of association rules are mined. The set of itemsets that were mined from the attributes in the transactional dataset are presented in Table 2.2. If we set the support threshold to 25% (2 transactions out of 8) and the confidence threshold to 50% we can observe that the rules numbered 1, 3, 5 and 7 should be eliminated. The association rules that remain after the support and confidence thresholds have been taken into consideration are the result of the first step in building the associative classifier.

The next step consists of applying pruning techniques to the remaining set. For the purpose of this example, no rules are pruned in this step. The associative classifier will consist of the rules listed in Table 2.3.

The last step represents the classification phase where new objects are classified. For our example, let the new object be represented by the transaction in Table 2.4. According to the rules forming the associative classifier (Table 2.3), two rules apply to the new object: *age*=25..40 $\Rightarrow$ *credit=good* with confidence 75% and *income=medium* $\Rightarrow$ *credit=good*

| 1-itemset | Support | Rule id | Association rules btw. the 1-itemset and a class label | Rule confidence |
|---|---|---|---|---|
| age<25 | 3 | 1 | $age < 25 \Rightarrow credit{=}bad$ | 33% |
| | | 2 | $age < 25 \Rightarrow credit{=}good$ | 67% |
| age=25..40 | 4 | 3 | $age{=}25..40 \Rightarrow credit{=}bad$ | 33% |
| | | 4 | $age{=}25..40 \Rightarrow credit{=}good$ | 75% |
| age>40 | 1 | 5 | $age > 40 \Rightarrow credit{=}good$ | 100% |
| income=low | 3 | 6 | $income{=}low \Rightarrow credit{=}bad$ | 67% |
| | | 7 | $income{=}low \Rightarrow credit{=}good$ | 33% |
| income=medium | 2 | 8 | $income{=}medium \Rightarrow credit{=}good$ | 100% |
| income=high | 3 | 9 | $income{=}high \Rightarrow credit{=}good$ | 100% |

Table 2.2: Mined 1-itemsets with their supports and possible associations with the class labels.

| Association rules | Rule confidence |
|---|---|
| $age < 25 \Rightarrow credit{=}good$ | 67% |
| $age{=}25..40 \Rightarrow credit{=}good$ | 75% |
| $income{=}low \Rightarrow credit{=}bad$ | 67% |
| $income{=}medium \Rightarrow credit{=}good$ | 100% |
| $income{=}high \Rightarrow credit{=}good$ | 100% |

Table 2.3: The associative classifier.

| Transaction ID | Age | Income |
|---|---|---|
| 1 | 25..40 | medium |

Table 2.4: New object to be classified.

with confidence 100%. Having two rules that apply and indicate a possible class for the new object, we need to combine their information and associate a class to the new object. As we shall see later, combining the rules' decisions is an important step for associative classifiers, with several strategies being proposed. For this example, we average for each class the confidence of the rules that applied to the new object, concluding that the new object should be considered under 'good' credit with 86% confidence.

### 2.3.1 CBA: Classification Based on Associations

Classification Based on Associations (CBA) [LHM98] is the first associative classifier proposed in the literature. It integrates classification and association rule mining to develop a classifier based on association rules. It adopts an apriori-like algorithm [AIS93] in the rule generation phase and then it employs some pruning techniques to reduce the set of association rules.

**Generate the Set of Rules.** Generating the association rules is the first phase of CBA. It consists of finding all CARs (class association rules) that satisfy minimum support and confidence requirements. CBA finds all the itemsets that exceed minimum support and they are of the following form: $< F, c >$, where $F$ is a set of items and $c$ is a class label. Each item that satisfies this format represents a rule: $R : F \rightarrow c$. If the confidence of the rule is greater that the minimum confidence the rule belongs to the CAR set.

**Build the Classifier.** The authors chose to employ a pruning technique in order to select the best representative association rules from the CAR set. This pruning technique is based on the database coverage. First, CBA sorts all the rules in CAR based on their rank where rank is defined as follows.

**Definition 1:** Given two rules $R_1$ and $R_2$, $R_1$ is higher ranked than $R_2$ if:

(1) $R_1$ has higher confidence than $R_2$;

(2) if the confidences are equal support($R_1$) must exceed support($R_2$);

(3) both confidences and supports are equal but $R_1$ has fewer attributes in the antecedent than $R_2$

Next, it selects the most representative and high ranked rules according to the database coverage explained below:

(1) **foreach** rule R in the set CAR=sort(CAR)
(2) go over dataset D and find those transactions that are covered by the rule R
(3) **if** R associates at least one transaction to its correct class
(4) select R
(5) remove those cases from D that were covered by R
(6) create a default rule (i.e., a rule that selects majority class in the remaining tuples in database)

The associative classifier consists of the set of rules that remains after database coverage was applied.

**Classification Phase.** In the classification phase, CBA searches the set of rules that represents the classifier starting with the highest ranked. It finds the first rule that matches the new object to be classified. When such a rule is found, its class label is associated with the new object. Otherwise, the default rule is applied.

### 2.3.2 CMAR: Classification based on Multiple Association Rules

Classification based on Multiple Association Rules (CMAR) [LHP01] was proposed two years after CBA. It outperforms CBA on the UCI datasets [BM98] on which both methods were tested. In the association rule mining phase it employs the FP-tree growth algorithm [HPY00]. It proposes a new structure for rule storage and employs pruning techniques for selecting a high quality set of rules. The classification process is based on multiple rule analysis (the decision is made based on the analysis performed on a set of rules).

**Generate the Set of Rules.** Based on the FP-tree growth algorithm in the rule generation phase, CMAR finds the complete set of rules in the form: $R : F \rightarrow c$, where $F$ is a set of attributes and $c$ is a class label. Only the rules that pass minimum support and minimum confidence thresholds are considered as candidates in building the classifier. The authors propose a new indexing structure for rule storage which is also used in the pruning phase.

**Build the Classifier.** The classifier consists of a high quality set of rules that results after the pruning techniques are applied on the initial candidate set. CMAR employs three pruning techniques: specific rule elimination, selecting positively correlated rules and database coverage.

The rules are sorted according to the same ordering definition as in CBA. Then, lower ranked specific rules are eliminated as follows.

**Definition 2** Given two rules $T_1 \Rightarrow C$ and $T_2 \Rightarrow C$ we say that the first rule is more general rule w.r.t. the second rule iff $T_1 \subseteq T_2$.

(1) sort the rules according to **Definition 1**
(2) **foreach** rule in the set of candidate rules
(3) find all those rules that are more specific (according to **Definition 2**)
(4) prune those that have lower confidence

The next pruning technique that is employed is selecting only those rules that have a positive correlation between their antecedent and the associated class label. This is done by using the chi-square test.

The last pruning technique is the database coverage. It is similar to the one used in CBA, except that for CMAR the authors introduced a coverage threshold $\delta$. Each transaction in the database has a count associated with it. The count increases when a rule covers the tuple that is associated with the count. The tuples are eliminated only when the count is greater than the coverage threshold. In CBA the tuples would be eliminated when the count equals 1.

**Classification Phase.** The classification phase in CMAR is based on multiple association rules. Consider a new tuple T to be classified. CMAR chooses the subset of rules that match T. It divides the rules according to the class label and evaluates these groups based

on chi-square analysis [LHP01]. Then, CMAR chooses the strongest group based on the chi-square measure and classifies T in its corresponding class. The pseudocode for this classification stage is given below.

(1) select the rules that match T and put them in prediction-set P
(2) **if** all the rules in P indicate the same class label
    attach that label to the new tuple
(3) **else**
(4)  divide P into subsets based on the class labels
(5)  in each group compute the weighted chi-square for each rule and sum them up
(6)  attach to T that class label that had the highest sum of weighted chi-square

### 2.3.3 ARC: Association Rule-based Classification

We introduced Association Rule-based Classification (ARC) in [AZC01, Ant02]. We proposed two algorithms, ARC-AC (AC=All Classes) and ARC-BC (BC=By Class).

**Generate the Set of Rules.** ARC algorithms generate all the constrained association rules in an apriori-like fashion. They discover all the rules of the form $R : F \rightarrow c$ that exceed the support and confidence thresholds. The difference between ARC-AC and ARC-BC algorithms is on what dataset the apriori-like algorithm is applied. In ARC-AC, the rules are generated from the entire training set, similar to CBA and CMAR. In ARC-BC, the training set is divided into subsets by class. The mining algorithm is applied on each of these subsets and the generated rules are merged to form the final set of rules. Thus the set of rules generated is different than the one obtained by ARC-AC. The reason for mining by class is that of class imbalance. In some applications, there are classes of interest that are very small compared to the entire collection of data. Classification rules for these small classes may not be discovered when all the classes are mined together, if their support is smaller that the given threshold.

**Build the Classifier.** Some pruning techniques are applied on the set of rules discovered in the first phase. Database coverage and removal of low ranked specialized rules were applied, similar to CBA and CMAR. We also studied the behaviour of conflicting rules. These are rules that have the same antecedent, but imply different classes. We found that removing these rules improved the accuracy of the classifier in the case of single-class classification.

**Classification Phase:** ARC algorithms classify new objects based on confidence. Consider a new tuple T to be classified. The algorithm chooses the subset S of rules that match T. This set is divided according to the class label. For each class the average confidence of the rules is computed, which will constitute the class score. The classifier classifies T in the class with the highest score. A pseudocode is given below.

(1) divide S in subsets by category: $S_1, S_2 ... S_n$
(2) **foreach** subset $S_1, S_2 ... S_n$
(3)  sum the confidences of rules and divide by the number of rules in $S_k$
(4)  $score_i = \frac{\sum r.conf}{\#rules}$

| | CBA | CMAR | ARC-AC | ARC-BC |
|---|---|---|---|---|
| Year of publication | 1998 | 2001 | 2001 | 2002 |
| Rule generation phase | Apriori-like on the entire set | FP-growth on the entire set | Apriori-like on the entire set | Apriori-like on subsets (by category) |
| Pruning phase | Database coverage | Database coverage and removal of low ranked specialized rules | Database coverage, removal of low ranked specialized rules and elimination of conflicting rules | Database coverage, removal of low ranked specialized rules and elimination of conflicting rules |
| Single-class classification | Apply only the first raked rule | Analyze a set of rules and decide the winner using a weighted chi-square | Analyze a set of rules and decide the winner using the average of the confidences | Analyze a set of rules and decide the winner using the average of the confidences |
| Multi-class classification | N/A | N/A | Use the dominance factor to allow multiple classes as prediction | Use the dominance factor to allow multiple classes as prediction |
| Reported experiments | UCI datasets | UCI datasets | UCI datasets, medical images and text documents | Medical images and text documents |

Table 2.5: Comparison between Associative Classifiers

(5) put the new object in the class that has the highest confidence score
(5) $o \leftarrow C_i$, with $score_i = max\{score_1..score_n\}$

When single-class classification is performed the new object is associated to the class that has the highest score. When multi-class categorization is employed the new object is assigned to k classes using the notion of *dominance factor* that was introduced in [ZA02]. *Dominance factor* represents a percentage that allows us to classify an object in all the classes whose score exceeds this percentage from the maximum score.

### 2.3.4 Comparison among Associative Classifiers

Table 2.5 compares the associative classifiers presented above to emphasize their differences.

A few other associative classifiers have been proposed in the literature. Harmony [WK05] is an associative classifier that generates rules in an instance-centric fashion. This approach assures that at least one high-confidence rule for each training instance is included in the final set of rules. The claim is that by employing an instance-centric approach more interesting rules are generated. A lazy associative classifier was introduced in [VJZ06]. The generation of classification rules is done at the time a new example has to be classified. To deal with the computational issue of the lazy associative classification some caching mechanisms are used. An extension to this system was proposed in [VJGZ07]. A multi-label associative classifier which deals with the issue of small disjuncts and takes into account the dependencies among class labels is proposed. The correlation among labels is considered by allowing the presence of multiple labels in the antecedent. To reduce the search space of the classification rules, the authors employ a heuristic called progressive label fo-

| |
|---|
| $t_1 \bigwedge t_3 \bigwedge t_4 \Rightarrow c_i$ |
| $\overline{t_2} \bigwedge t_5 \Rightarrow \overline{c_i}$ |
| $\overline{t_1} \bigwedge t_2 \bigwedge t_7 \Rightarrow c_j$ |
| $t_1 \bigwedge t_2 \bigwedge \overline{t_4} \Rightarrow \overline{c_j}$ |

Table 2.6: Examples of decision rules

cusing. In [VWMC+06] an application driven lazy associative classifier that incorporates link-based evidence is proposed for document classification.

## 2.4 Other Rule-based Classification Models

Another approach to building rule-based classifiers comes from the Machine Learning community. Over the years, several rule induction algorithms have been proposed in the literature. Most of these algorithms are covering algorithms (i.e., they generate rules that reliably cover a subset of the data). The algorithms differ from one another in the searching employed and in the measures used to account for the interestingness of the discovered rules. In addition, different pruning techniques are used to avoid overfitting. In the following we briefly discuss several algorithms.

### 2.4.1 Decision Rules

Rule induction methods have been extensively discussed in the literature [Qui93, WK91]. A decision rule classifier consists of a set of rules in disjunctive normal form (DNF). The antecedent of the rule is formed by the presence or absence of terms in documents, while the consequent says whether to classify the object under class $c_i$ or not. An example of such a set of rules is given in Table 2.6

The decision rules induction is done in a bottom-up fashion, while the induction in decision trees, as described below, is done in a top-down manner. When the building of a classifier for category $c_i$ starts, all the terms in each training document represents the antecedent of a rule. The consequent is either $c_i$, if the object is a positive example, or $\overline{c}_i$, if the object is a negative one. This process may lead to an overfitting set of rules. That is why, when a set of rules is induced, generalization and pruning criteria are employed. The heuristics for these two steps differ from one classifier to another. In the next paragraph we describe these two steps into more details.

**CN2 algorithm** CN2 [CN89] is a rule induction algorithm that performs a general to specific search. To reduce the risk of finding a suboptimal set of rules, CN2 performs a general to specific beam search (i.e., a search where the algorithm keeps at every step a set of best candidates, rather than a single one).

The induction of a set of rules starts with an empty set of rules. CN2 algorithm searches in the training set for the best pattern. Then, it eliminates from the training set the covered cases by this pattern and adds to the rule list a relationship between the pattern and the most common class covered by the pattern. It repeats this step until there are no more objects in the training set or no other pattern can be found. The patterns are discovered in the following way. Consider A as being the set of all attributes in the training set D.

Construct the candidate set to best rule by doing all the combinations between an element from best_set (best_set built using a heuristic) (rules that candidate to best_rule) and one from A. From the candidate set remove those tuples that are already in best_set or that are contradictory. Once the candidate set is built take each element one by one and if it is more informative than the existing best_rule, replace the best_rule with this element. When all the elements in candidate set were tested, return the best_rule.

### 2.4.2 RIPPER algorithm

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [Coh95] is built upon IREP (Incremental Reduced Error Pruning) algorithm [FW94]. RIPPER brings several modifications to IREP. The modifications keep the efficiency that IREP provided, while the performance of the classification model is improved. Following IREP's strategy, RIPPER splits the training set in two sets. One of them is used to grow the rules and the other to prune the rules. A rule is grown as follows: the algorithm starts with an empty rule and it repeatedly adds conditions that maximize the information gain criterion. Once the rule is grown, conditions are deleted to maximize a function during the pruning phase. When a rule has been discovered, all the examples that are covered by these rules are removed from the training set. The above process continues to learn rules for the remaining training set. In addition, RIPPER applies a rule optimization to refine the final set of rules. This refining is guided to minimize the error of the entire set of rules.

### 2.4.3 CPAR - a hybrid between associative classifiers and rule-based classifiers

CPAR [YH03] (Classification based on Predictive Association Rules) is a hybrid between associative classifiers and rule-based classifiers that use greedy techniques. It uses a greedy algorithm to search the space of attributes. The main difference is that it keeps all close-to-the-best attributes in rule generation, unlike rule-based methods which use only the best attribute. Thus, it creates more rules than traditional rule-based approaches, but much less than associative classifiers. In the classification stage the system uses the best $k$ rules that satisfy the new object. To make the classification decision CPAR uses the expected accuracy measure.

Although the rule induction algorithms generate rules in the same format as associative classifiers, they differ in several ways. First, the rule generation phase is completely different: rule-induction algorithms perform a search over the space to find optimal rules that distinguish between a set of positive examples and a set of negative ones (different search criteria are used to accomplish this step); associative classifiers search the space to find interesting associations between a set of conditions and a class label. In addition, the measures used are different: rule-learning systems use certain functions to be maximized (e.g., information gain), while associative classifiers discover associations that exceed a certain support and confidence threshold. Rule learning algorithms use post-pruning techniques to improve the set of rules. The associative classifiers may use pruning techniques as well. The difference is that while most rule learning algorithms remove conditions from the antecedent of a rule in this phase, the associative classifiers remove only entire rules from the set.

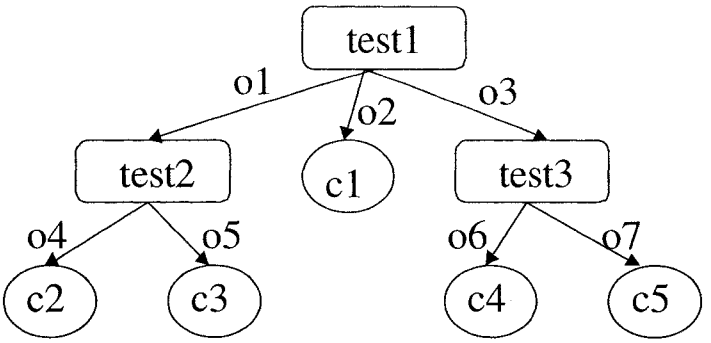


Figure 2.2: Decision Tree

## 2.5 Other Classification Systems

This section introduces some well-known classification system that are used and/or compared against throughout this dissertation.

### 2.5.1 Decision Trees

A decision tree is a tree structure where each internal node represents a test on an attribute, each branch is labeled by the outcome of the test and each leaf node represents a category. An example of such tree is given in Figure 2.2. Let **d** be a new instance to be classified. In order to classify it, all its attribute values $d_k$ are tested against the decision tree.

A procedure for building a decision tree is to apply a greedy algorithm in a top-down recursive divide-and-conquer manner. The basic algorithm is as follows:

(1) create a node representing the training set
(2) **if** all the samples belong to the same category
(3) attach this label to the node, and it becomes a leaf
(4) **else**
(5) using a heuristic search find the attribute that best separates the samples into classes, and place each class in a subtree

The algorithm is used recursively to form a decision tree for each partition. The recursive process stops when:

(a) all samples for a given node, belong to the same class

(b) the samples cannot be further divided (no attributes left)

(c) there are no samples for a certain branch. In this case a leaf is created with the majority class.

Entropy is a common measure for how informative a node is in the tree. When a new tuple is presented to the system for classification, the attributes of this tuple are tested against the decision tree. Starting with the root node the attributes are tested until a leaf node is reached. At this point the value of the leaf node is attached to the tested tuple as class. There are some standard packages for creating a decision tree when a training set is given. Among the most popular are ID3 [FB91], C4.5 [Qui93, CH98, CS99, Joa98] and C5 [LJ98].

### 2.5.2 Probabilistic Classifiers

Classifiers based on probabilistic models have been proposed in the literature and proved to perform well. Naïve-Bayes [Lew98] is a widely used probabilistic classifier.

Probabilistic classifiers estimate the probability of each class given the features of the new object to be classified. To determine these probabilities, Bayes theorem is used. Let **d** be a new instance to be classified. Its class is unknown. Let $c_i$ be the hypothesis that the object falls into category $c_i$.

Bayes theorem is given by the following formula:

$$P(c_i|\mathbf{d}) = \frac{P(c_i) * P(\mathbf{d}|c_i)}{P(\mathbf{d})} \tag{2.1}$$

The above equation states that the object **d** falls in class $c_i$ with probability $P(c_i|\mathbf{d})$. However, the computation of this probability is difficult. As a consequence, it is common to make an independence assumption that will make the computation easier. It is assumed that there are no dependence relationships among the object attributes. The independence assumption presumes that the attributes when considered as random variables are statistically independent. Thus, the key term in the right-hand side of Equation 2.1 becomes:

$$P(\mathbf{d}|c_i) = \prod P(d_k|c_i) \tag{2.2}$$

where $d_k$ is the kth attribute of object **d**.

Probabilistic classifiers that make use of this assumption and compute the probability according to equation 2.2 are called Naïve Bayes classifiers. Despite the fact that the assumption used is not entirely true in real applications Naïve Bayes classifiers are effective. More details and a thorough survey of Bayesian classifiers is given in [Lew98].

### 2.5.3 Example-based Classifiers

Example-based classification methods [DH73] do not build an actual classifier, but rather base their decision upon the training objects similar to the test ones. They are based on analogy, rather than learning. That is why these methods belong to the example-based class and they are called lazy learning systems.

The best known classification method belonging to this class is k-NN (k nearest neighbours) [CH67]. When a new object has to be classified using this method, k-NN finds the k training objects most similar to the new one. The similarity of two objects is usually defined in terms of Euclidean distance. In text categorization, often, the documents are represented in a vector model, the Euclidean distance being computed between each element of the vectors associated to documents (i.e. the weights associated with each term). The Euclidean distance between two points is given by the formula in Equation 2.3.

$$d(X, Y) = \sqrt{\sum (x_i - y_i)^2} \tag{2.3}$$

A decision is made on these k objects. If a large portion of the k nearest neighbors is classified in class $c_i$, then the new object falls in that class. The building of a k-NN classification system also involves finding the threshold k. This is usually done experimentally by using a validation set.

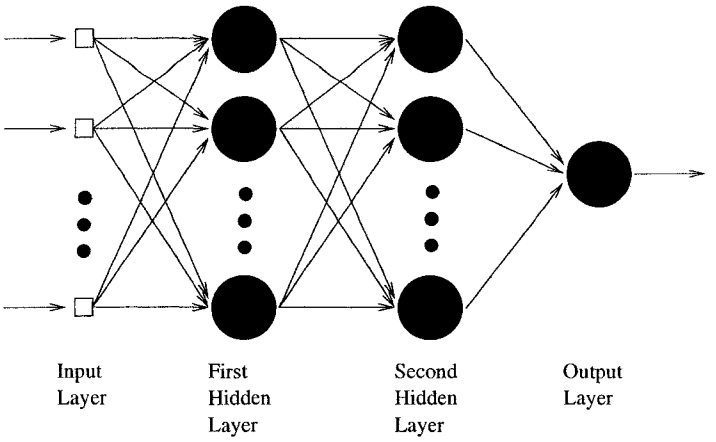


Figure 2.3: 3-Layer Neural Network

### 2.5.4 Neural Networks

Artificial neural network models [RHW86, HKP91, Bis95] have been studied for many years in the hope of achieving human-like performance in fields like speech and image understanding. The networks are composed of many non-linear computational elements operating in parallel and arranged in patterns reminiscent of biological neural networks. Computational elements or nodes are connected in several layers (input, hidden and output) via weights that are typically adapted during the training phase to achieve high performance. Instead of performing a set of instructions sequentially, neural network models explore simultaneously many hypotheses using parallel networks composed of many computational elements connected by links with variable weights. A typical n-layer neural network is described in Figure 2.3.

The weights of a neural network are set during the training phase. The attributes of a training sample are given to the input layer and the weights are automatically adjusted so that the output layer indicates the category assigned to the training object. This process is repeated for all training objects. The classifier is represented by the neural network with the weights adjusted during the training process. To classify a new object the attributes are assigned to the nodes in the input layer and the value given by the output layer indicates the class in which the object should fall.

Given a set of training examples to be fed into a neural network the goal is to adjust the weights attached to the nodes in the network in such a manner that makes almost all the tuples in the training data classified correctly.

Below are presented the basic steps in generating a trained neural network.

**Training a neural network:**

(1) initialize weights with random values
(2) feed the training examples into the network one by one
(3) **foreach** example
(4) compute the input to the network as a linear combination of all the inputs to the input layer
(5) compute the value for the output node using the activation function
(6) evaluate the error between the expected outcome and

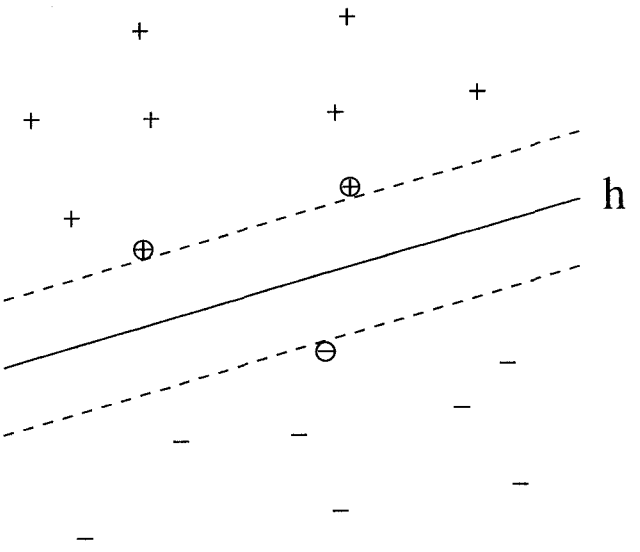


Figure 2.4: Support Vector Machine Classifier

the predicted value

(7) update the weights and the bias of the network

### 2.5.5 Support Vector Machines

The concept of support vector machines was introduced in 1995 by Vapnik [Vap95]. This method is based on the *Structural Risk Minimization* principle from computational learning theory. The main idea is to find in the space of data, the hyperplane $h$ that discriminates best between two classes. The samples that lie closest to the hyperplane (both positive and negative examples) are called support vectors. Once the hyperplane is determined, new objects can be classified by checking on which side of the hyperplane they lie. A graphical representation is given in Figure 2.4.

The problem is to find the $h$ with the lowest error. The upper bound of the error is given in Equation 2.4, where $n$ is the number of training examples and $d$ is the Vapnik-Chervonenkis dimension. The VC-dimension characterizes the complexity of the problem.

$$P(error(h)) \leq train\_error(h) + 2 * \sqrt{\frac{d * (ln\frac{2*n}{d} + 1) - ln\frac{\eta}{4}}{n}} \tag{2.4}$$

The idea is to find the hypothesis that minimizes equation 2.4. When the optimal hyperplane is found for each class, the classification phase is trivial. For each new object to be classified it is checked on which side of the hyperplane it falls, and that category is assigned to it. SVM is a typical approach where the multi-class problem is divided into disjoint binary categorization tasks. To classify a new object all binary classifiers are invoked and their decisions are combined to predict the new classes associated with the object to be classified.

### 2.5.6 Boosting and Bagging

The technique of combining the prediction of multiple classifiers to build a single classifier was studied by the researchers based on the idea that k classification systems can perform better than a single one. This general process is called voting. Such a classifier takes a learning method and a training set and builds multiple classifiers on different versions of the training set. These resulting classifiers are combined to generate the final one. Voting algorithms can be divided in two classes: boosting and bagging.

**Boosting.** The boosting method [Sch99] builds a classifier by using multiple previously generated classifiers. By using the same learning method it induces k classifiers. The difference among these systems is the set used as training. It differs from one classifier to another. The method selects N samples from the training set when creating a new classifier. However, the probability of choosing a sample is not the same for all classifiers. It depends on how often the sample was misclassified with the previous created classifiers. Thus the boosting method attempts to create at each stage better classifiers in order to increase the performances of the final classifier. Many methods employing boosting algorithms were presented in the literature [Qui96, Elk97]. They differ in the way the samples are selected when a new classifier is to be generated.

**Bagging.** The bagging method [Bre96], like boosting, takes as input a learning method and a training set of objects. The difference consists of how the samples are chosen for building the classifiers. The training sample has the same size for each classifier as the original training set. For each classifier, the algorithm makes a number of replacements in the training set by uniform probability random selection. This means that some samples could repeat in the training set, while others may not be present at all.

To classify a new object, all the classifiers built are invoked. The new object falls in the class that obtained the most votes.

## 2.6 Performance Evaluation

Learning predictive systems is an important problem with many applications. However, the learning and developing of automatic classification systems is not the only task to be considered for a new application. The other issue to be taken into account is the classifiers' performance evaluation. We first present some datasets used throughout this dissertation for evaluating the performance of the proposed techniques, followed by evaluating measures that allow us to compare our proposed systems with previously proposed traditional or state-of-the-art systems.

### 2.6.1 Datasets

This section presents an overview of the datasets that were used throughout this dissertation. First, the characteristics of the datasets from the UCI repository [BM98] are presented. Then, the Reuters dataset [Reu08] is discussed.

| dataset name | number of instances | number of attributes | number of numeric attributes | number of classes | percentage of instances in the majority class |
|---|---|---|---|---|---|
| annealing | 898 | 38 | 9 | 6 | 67.7 |
| australian | 690 | 14 | 6 | 2 | 55.5 |
| breast-w | 699 | 10 | 10 | 2 | 65.5 |
| cleve | 303 | 13 | 6 | 2 | 54.4 |
| crx | 690 | 15 | 6 | 2 | 55.5 |
| diabetes | 768 | 8 | 8 | 2 | 65.1 |
| german | 1000 | 20 | 7 | 2 | 70.0 |
| glass | 214 | 9 | 9 | 7 | 35.5 |
| heart | 270 | 13 | 6 | 2 | 55.5 |
| hepatitis | 155 | 19 | 6 | 2 | 79.4 |
| horse | 368 | 21 | 7 | 2 | 57.1 |
| iris | 150 | 4 | 4 | 3 | 33.3 |
| labor | 57 | 16 | 8 | 2 | 64.9 |
| led7 | 3200 | 7 | 7 | 10 | 10.6 |
| pima | 768 | 8 | 8 | 2 | 65.1 |
| tic-tac-toe | 958 | 9 | 0 | 2 | 65.3 |
| vehicle | 846 | 18 | 18 | 4 | 25.7 |
| waveform | 5000 | 21 | 21 | 3 | 33.3 |
| wine | 178 | 13 | 13 | 3 | 39.9 |
| zoo | 101 | 17 | 2 | 7 | 40.6 |

Table 2.7: Summary of the UCI datasets used throughout this dissertation

### UCI Datasets

Table 2.7 shows the characteristics of the UCI datasets [BM98] that we used in our evaluation throughout this dissertation. It describes each dataset in terms of the number of instances, the number of attributes, how many of these attributes are numerical, the number of classes and the class distribution.

On each UCI dataset we performed C4.5's shuffle utility [Qui93] for shuffling the datasets to ensure randomness in the training/testing splits. A 10-fold cross validation (defined in Section 2.2.1) was performed on each dataset and the reported results are averages over the 10 folds. In addition, we used the same discretization method for continuous attributes as in [LHM98] to have a fair comparison with other algorithms.

### Reuters Dataset

In the last decades tasks dealing with textual information embedded in documents gained an important status in research and industrial fields. The main reason behind this increase is the growing number of documents available in digital format. Since the first automated text categorization system was proposed in the literature in 1961 [Mar61] there have been many direct applications to text classification, such as automatic indexing to assist information retrieval systems, document classification, text filtering and document routing. These are just a few applications that are highly dependent on a good classification system. Nowadays with the expansion of the Web, text classification is useful for web page categorization, e-mail classification and filtering. Another application for text classification is word sense disambiguation, one of the most important problems in computational linguistics. For a thorough survey of the existing text classification systems see [Seb99].

We chose this application for our evaluation not only for its importance as discussed above, but because of its challenges as well. In this application we encounter a very high-

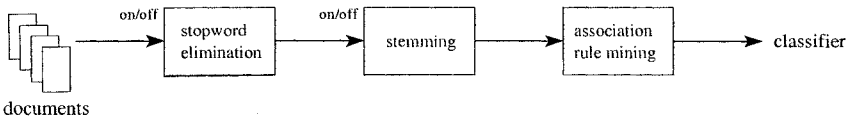


Figure 2.5: Associative Classifier

dimensional space due to the large number of words that appear in a text corpora. In addition, a very interesting and challenging problem is that this application requires multi-label classification. Each document in the collection may be classified into one or more classes. We are interested to see how our proposed approaches work in this context.

In order to objectively evaluate our algorithm vis-à-vis other approaches we used the Reuters-21578 text collection [Reu08] as benchmark, like other researchers in the field of automatic text categorization. Text collections for experiments are usually split into two parts: one part for training or building the classifier and a second part, for testing the effectiveness of the system.

There are many splits of the Reuters collection; we chose to use the *ModApte* version. This split leads to a corpus of 12,202 documents consisting of 9,603 training documents and 3,299 testing documents. There are 135 topics to which documents are assigned. However, only 93 of them have more than one document in the training set and 82 of the categories have fewer than 100 documents. We tested our classifier on the ten categories with the largest number of documents assigned to them in the training set.

A data cleaning phase is required to weed out those words that are of no interest in building the associative classifier. We consider stemming and stopwording as well as the transformation of documents into transactions as a pre-processing phase. Stopword removal is done according to the term frequency values and a given list of stopwords. Figure 2.5 illustrates the phases of the association-rule-based text categorizer construction with the optional stemming and stopwording modules. Stopwords removal considerably reduces the dimensions of the transactional database and thus significantly improves the rule extraction time (i.e. training time). Moreover, while we use a common stopword list in English [Fox92], too frequent terms that are associated to all categories can be automatically added as words to reject. Note that the stopword lists from any language can be used as well.

In our approach, we model text documents as transactions where items are words from the document as well as the categories to which the document belongs. On these documents we performed stopword elimination but no stemming. It is only after the terms are selected from the cleansed documents that the transactions are formed. The subsequent phase consists of discovering association rules from the set of cleansed transactions.

### 2.6.2 Evaluation Measures

The evaluation process is a very important step in building a good and reliable classification system. The measures and techniques used in the evaluation give us an insight into the reliability of our classification systems. These measures present us with an estimate of how well our classifier will perform and how it compares to other classification systems.

The most common measure used in classification evaluation is accuracy, which represents the percentage of the correct classifications out of the total number of classifications performed. Its definition is given in the following equation:

| | | actual class | |
|---|---|---|---|
| | | Yes | No |
| predicted class | Yes | TP | FP |
| | No | FN | TN |

Table 2.8: Contingency table for a two-class problem

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.5}$$

A two-class classification problem and its possible outcomes is presented in the contingency table shown in Table 2.8. The TP (true positives) and TN (true negatives) are correct classifications. A FP (false positive) occurs when the outcome is incorrectly predicted as "yes". A FN (false negative) occurs when the outcome is incorrectly predicted as "no" when it is actually "yes".

Accuracy is a good evaluation method in binary classification and balanced datasets. However, accuracy is not an appropriate measure in all classification cases. There are two well-known such cases. One of them is in the case of rare classes and the other one appears when multi-label classification is performed. In our experiments, we encounter both of these cases. This is why we need other evaluation measures. Some evaluation measures, as well as some used in our evaluation, can be defined in terms of precision and recall. Precision represents how many of the objects predicted as belonging to the "yes" class do belong to that class. Recall represents how many of the objects that belong to the "yes" class were predicted as such. The formulae for precision (P) and recall (R) are:

$$P = \frac{TP}{TP + FP} \tag{2.6}$$

$$R = \frac{TP}{TP + FN} \tag{2.7}$$

Another evaluation measure is the *break-even point* (Equation 2.8). Breakeven point (BEP) is the point at which precision equals recall and it is obtained as reported in [BEYTW01]:

$$BEP = \frac{R + P}{2} \tag{2.8}$$

F1 measure is another measure that combines precision and recall. F1 measure is a particular case of the $F_\beta$ measure derived from van Rijsbergen's E measure [vR79]. This measure is a harmonic average of precision (P) and recall (R) and combines them as in the following formula:

$$F_\beta = \frac{(\beta^2 + 1) \times P \times R}{R + \beta^2 \times P} \tag{2.9}$$

F1 measure is obtained when $\beta$ equals 1.

When dealing with multiple classes, as in the text classification application, there are two possible ways of averaging these measures, namely, *macro-average* and *micro-average*. In macro-averaging, one contingency table as in Table 2.8 per class is used, the performance measures are computed for each class and then averaged over all classes. In micro-averaging only one contingency table is used over all classes, the average of all the classes

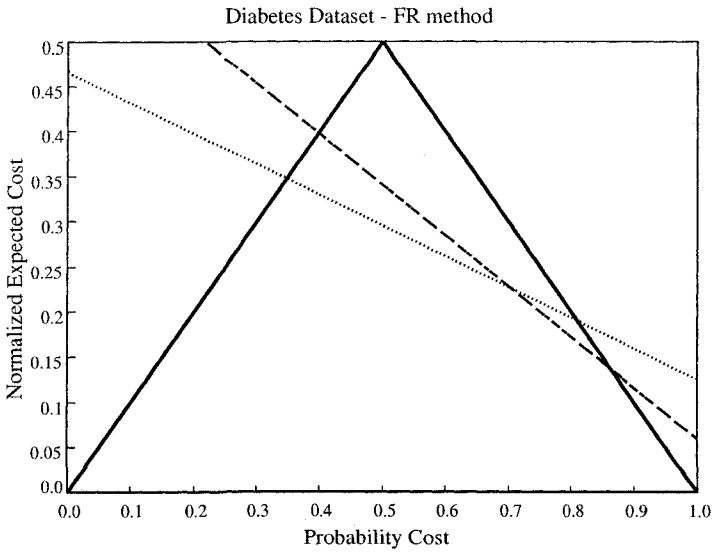


Figure 2.6: An example of a classifier evaluation with a cost curve

is computed for each cell and the performance measures are obtained therein. The macro-average weights equally all the classes, regardless of how many documents belong to each one of them. The micro-average weights equally all the documents, thus favouring the performance on common classes.

Predicting rare classes is an important issue in many domains as discussed in Section 2.2.3. A way of evaluating a classification system when rare classes are involved is by using a Receiver Operating Characteristic (ROC) curve [PF97]. An ROC curve is a plot of the true positive rate against the false positive rate (formulas given in Equation 2.11) for the different possible thresholds. The ROC curve records various combinations of false positive and false negative errors, thus evaluating a classification system on a broader range by computing the area under the curve (AUC). The larger the area under the curve, the more accurate the test.

Cost curves [DH06] are evaluation tools for classification systems that have been proposed as an alternative to ROC curves [PF97]. Their advantage is that in their visualization one can easily see the performance of a classifier over the entire range of class frequencies and costs. Each classifier is represented by a straight line in the cost space. The y-axis is the normalized expected cost (NEC) of a classifier and is between 0 and 1. The x-axis (PC(+)) is the fraction of the total cost of using a classifier that is due to positive examples.

The performance of a classifier is represented in the cost space by the following equation:

$$NEC = (1 - TPR - FPR) \times PC(+) + FPR \tag{2.10}$$

where TPR (true positive rate) and FPR (false positive rate) are defined as follows:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{2.11}$$

An example of a classifier evaluation with a cost curve is shown in Figure 2.6. A line in

the cost space shows how the performance of the system varies when the class distribution or cost change. Ideally, one wants to have a classifier represented by a horizontal line at the normalized expected cost of 0. Thus, the best classifier for a probability cost value is the one whose normalized expected cost is the lowest. Visually, the lowest line in the graph represents the best classifier (the lower the line the better the classification system). From the graphs we can also directly see the difference in performance between two classifiers, which is their vertical height difference at some $PC(+)$ value.

The measures presented in this section are used throughout this dissertation in the evaluation of our proposed solutions.

# Chapter 3

# Associative Classifiers Using Negative Association Rules

This chapter presents the integration of negative association rules with associative classifiers. Typical association rules consider only items enumerated in transactions. Such rules are referred to as *positive association rules. Negative association rules* also consider the same items, but in addition consider negated items (i.e. absent from transactions). Negative association rules are useful in market-basket analysis to identify products that conflict with each other or products that complement each other. Many other applications would benefit from negative association rules if it was not for the expensive process to discover them. Indeed, mining for such rules necessitates the examination of an exponentially large search space. Despite their usefulness, and while they are referred to in many publications, very few algorithms to mine them have been proposed to date. In this chapter we motivate the importance of negative association rules and propose a solution for mining these types of rules. First, an algorithm based on statistical correlation to discover classification rules with negations is proposed and discussed. Then, these types of classification rules are integrated with the associative classifier framework and the results are presented. The challenge here is in dealing with new types of classification rules and in integrating them in the classification process.

## 3.1 Discovering Negative Association Rules

All the traditional association rule mining algorithms were developed to find positive associations between items. By positive associations we refer to associations between items existing in transactions (i.e. items bought). What about associations of the type: "customers that buy Coke *do not* buy Pepsi" or "customers that buy juice *do not* buy bottled water"? In addition to the positive associations, the negative associations can provide valuable information in devising marketing strategies. Interestingly, very few studies have focused on negative association rules due to the difficulty in discovering these rules. Although some researchers have pointed out the importance of negative associations [BMS97], only a few groups of researchers [WZZ02, THC02, SON98] have proposed algorithms to mine these types of associations. This not only illustrates the novelty of negative association rules, but also the challenge in discovering them.

| | organic | ¬organic | $\sum_{row}$ |
|---|---|---|---|
| non-organic | 20 | 60 | 80 |
| ¬non-organic | 20 | 0 | 20 |
| $\sum_{col}$ | 40 | 60 | 100 |

Table 3.1: Example 1 Data

### 3.1.1 Motivation

This section introduces a motivating example for the importance of negative association rules.

**Example 1.** Suppose we have an example from the market basket data. In this example we want to study the purchase of organic versus non-organic vegetables in a grocery store. Table 3.1 gives us the data collected from 100 baskets in the store. In Table 3.1 "organic" means the basket contains organic vegetables and "¬ organic" means the basket does not contain organic vegetables. The same applies for non-organic. On this data, let us find the positive association rules in the "support-confidence" framework. The association rule "non-organic $\dashrightarrow$ organic" has 20% support and 25% confidence (support(non-organic $\wedge$ organic)/support(non-organic)). The association rule "organic $\rightarrow$ non-organic" has 20% support and 50% confidence (support(non-organic $\wedge$ organic)/support(organic)). The support is considered fairly high for both rules. Although we may reject the first rule on the confidence basis, the second rule seems a valid rule and may be considered in the data analysis. Now, let us compute the statistical correlation between the *non-organic* and *organic* items. A more elaborated discussion on the correlation measure is given in Section 3.1.3. The correlation coefficient between these two items is -0.61. This means that the two items are negatively correlated. This measure sheds a new light on the data analysis on these specific items. It shows that the rule "organic $\rightarrow$ non-organic" is misleading. The correlation measure brings new information that can help in devising better marketing strategies.

The example above illustrates some weaknesses in the "support-confidence" framework and the need for the discovery of stronger interesting rules. The interestingness of an association rule can be defined in terms of the measure associated with it, as well as in the form an association can be found.

Brin *et. al* [BMS97] mentioned for the first time the notion of negative relationships in the literature. Their model is chi-square based. They use the statistical test to verify the independence between two variables. To determine the nature (positive or negative) of the relationship, a correlation metric was used. In [SON98] the authors present a new idea to mine strong negative rules. They combine positive frequent itemsets with domain knowledge in the form of a taxonomy to mine negative associations. However, their algorithm is hard to generalize since it is domain dependant and requires a predefined taxonomy. Wu *et. al* [WZZ02] derived another algorithm for generating both positive and negative association rules. They add on top of the support-confidence framework another measure called *mininterest* for a better pruning of the frequent itemsets generated. In [THC02] the authors use only negative associations of the type $X \rightarrow \neg Y$ to substitute items in market basket analysis.

We define as *generalized negative association rule*, a rule that contains a negation of an item (i.e a rule for which its antecedent or its consequent can be formed by a conjunction of presence or absence of items). An example for such association would be as follows: $A \wedge \neg B \wedge \neg C \wedge D \rightarrow E \wedge \neg F$. To the best of our knowledge, there is no algorithm

that can determine all generalized negative association rules. Deriving such an algorithm is not an easy problem, since it is well known that the itemset generation in the association rule mining process is an expensive one. It would be necessary not only to consider all items in a transaction, but also all possible items absent from the transaction. There could be a considerable exponential growth in the candidate generation phase. This is especially true in datasets with highly correlated attributes. That is why it is not feasible to extend the attribute space by adding the negated attributes and use the existing association rule algorithms. We generate and use in the classification process a subset of the generalized negative association rules. We refer to them as *confined negative association rules.* A confined negative association rule is one of the follows: $\neg X \rightarrow Y$ or $X \rightarrow \neg Y$, where the entire antecedent or consequent is treated as an atomic entity and the entire entity is either negated or not. In this dissertation, we refer to the confined negative association rules simply as negative association rules, but they are in fact a subset of the generalized negative association rules.

### 3.1.2 Our Approach to Discovering Classification Rules with Negations

The most common framework in the association rules generation is the "support-confidence" one. Although these two parameters allow the pruning of many associations that are discovered in the data, there are cases when many uninteresting rules may be produced. In our approach we consider another framework that adds to the support-confidence some measures based on correlation analysis. The next section introduces the correlation coefficient, which we add to the support-confidence framework in this chapter and then we present our algorithm to discover confined negative association rules.

### 3.1.3 Correlation Coefficient

The correlation coefficient measures the strength of the linear relationship between a pair of variables. It is discussed in the context of association patterns in [TK00]. For two variables X and Y, the correlation coefficient is given by the following formula:

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \quad . \tag{3.1}$$

In Equation 3.1, $Cov(X,Y)$ represents the covariance of the two variables (represents the strength of the correlation between two or more sets of random variables) and $\sigma_X$ stands for the standard deviation of $X$. The range of values for $\rho$ is between -1 and +1. If the two variables are independent then $\rho$ equals 0. When $\rho = +1$ the variables considered are perfectly positive correlated. Similarly, when $\rho = -1$ the variables considered are perfectly negative correlated. A positive correlation is evidence of a general tendency that when the value of X increases/decreases so does the value of Y. A negative correlation occurs when for the increase/decrease in X's value we discover a decrease/increase in the value of Y.

Let X and Y be two binary variables. Table 3.2 summarizes the information about X and Y variables in a dataset in a 2x2 contingency table. The cells of this table represent the possible combinations of X and Y and give the frequency associated with each combination. N is the size of the dataset considered.

Given the values in the contingency table for binary variables, Pearson introduced the

| | Y | $\neg Y$ | $\sum_{row}$ |
|---|---|---|---|
| X | $f_{11}$ | $f_{10}$ | $f_{1+}$ |
| $\neg X$ | $f_{01}$ | $f_{00}$ | $f_{0+}$ |
| $\sum_{col}$ | $f_{+1}$ | $f_{+0}$ | N |

Table 3.2: 2x2 Contingency table for binary variables

$\phi$ *correlation coefficient* which is given in the equation 3.2:

$$\phi = \frac{f_{11}f_{00} - f_{10}f_{01}}{\sqrt{f_{+0}f_{+1}f_{1+}f_{0+}}} \quad . \tag{3.2}$$

We can transform this equation by replacing $f_{00}$, $f_{01}$, $f_{10}$, $f_{0+}$ and $f_{+0}$ as follows:

$$\phi = \frac{f_{11}(N - f_{10} - f_{01} - f_{11}) - f_{10}f_{01}}{\sqrt{f_{+0}f_{+1}f_{1+}f_{0+}}} \tag{3.3}$$

$$\phi = \frac{f_{11}N - f_{11}f_{10} - f_{11}f_{01} - f_{11}^2 - f_{10}f_{01}}{\sqrt{f_{+0}f_{+1}f_{1+}f_{0+}}} \tag{3.4}$$

$$\phi = \frac{f_{11}N - (f_{11} + f_{10})(f_{11} + f_{01})}{\sqrt{f_{+0}f_{+1}f_{1+}f_{0+}}} \tag{3.5}$$

$$\phi = \frac{Nf11 - f_{1+} * f_{f+1}}{\sqrt{f_{1+}(N - f_{1+})f_{+1}(N - f_{+1})}} \quad . \tag{3.6}$$

We make this transformation to use values that we already have computed during the mining process. The measure given in Equation 3.6 is the measure that we use in the association rule generation algorithm to discover positive and negative confined rules.

Cohen [Coh88] discusses about the correlation coefficient and its strength. In his book, he considers that a correlation of 0.5 is large, 0.3 is moderate, and 0.1 is small. The interpretation of this statement is that anything greater than 0.5 is large, 0.5-0.3 is moderate, 0.3-0.1 is small, and anything smaller than 0.1 is insubstantial, trivial, or otherwise not worth worrying about as described in [Hop02].

We use these arguments to introduce an automatic progressive thresholding process. We start by setting our correlation threshold to 0.5. If no strong correlated rules are found the threshold slides progressively to 0.4, 0.3 and so on until some rules are found with moderate correlations. This progressive process eliminates the need for manually adjusted thresholds. It is well known that the more parameters a user is given, the more difficult it becomes to tune the system. Association rule mining is certainly not immune to this phenomenon.

### 3.1.4 Our Algorithm

Traditionally, the process of mining for association rules has two phases: first, mining for frequent itemsets; and second, generating strong association rules from the discovered frequent itemsets. In our algorithm, we combine the two phases and generate the relevant rules on-the-fly while analyzing the correlations within each candidate itemset. This avoids evaluating item combinations redundantly. Indeed, for each generated candidate itemset, we compute all possible combinations of items to analyze their correlations. In the end, we keep only those rules generated from item combinations with strong correlation. The strength of the correlation is indicated by a correlation threshold, either given as input or by

default set to 0.5 (see above for the rationale). If the correlation between item combinations $X$ and $Y$ of an itemset $XY$, where $X$ and $Y$ are itemsets, is negative, negative association rules are generated when their confidence is high enough. The produced rules have either the antecedent or the consequent negated: ($\neg X \rightarrow Y$ and $X \rightarrow \neg Y$), even if the support is not higher than the support threshold. However, if the correlation is positive, a positive association rule with the classical support-confidence idea is generated. If the support is not adequate, a negative association rule that negates both the antecedent and the consequent is generated when its confidence and support are high.

The algorithm generates all positive and negative association rules that have a strong correlation. If no rule is found, either positive or negative, the correlation threshold is automatically lowered to ease the constraint on the strength of the correlation and the process is redone. Figure 3.1 gives the detailed pseudo-code for our algorithm.

Initially both sets of negative and positive association rules are set to empty (line 1). After generating all the frequent 1-itemsets (line 2) we iterate to generate all frequent k-itemsets, stored in $F_k$ (line 8). $F_k$ is verified from a set of candidate $C_k$ computed in line 4. The iteration from line 3 stops when no more frequent itemsets can be generated. Unlike the join made in the traditional *Apriori* algorithm to generate candidates at level $k$, instead of joining frequent $(k-1)$-itemsets, we join the frequent itemsets at level $k-1$ with the frequent 1-itemsets (line 4). Please note that this is not a natural join. This is because we want to extend the set of candidate itemsets and have the possibility to analyze the correlation of more item combinations. The rationale will be explained shortly. Every candidate itemset generated this way is on one hand tested for support (line 7), and on the other hand used to analyze possible correlations even if its support is below the minimum support (loop from line 9 to 22). Correlations for all possible pair combinations for each candidate itemset are computed. For an itemset $i$ and a pair combination $(X, Y)$ such that $i = X \cup Y$, the correlation coefficient is calculated (line 10). If the correlation is positive and strong enough, a positive association rule of the type $X \rightarrow Y$ is generated, if the $supp(X \cup Y)$ is above the minimum support threshold and the confidence of the rule is strong. Otherwise, if we still have a positive and strong correlation but the support is below the minimum support, a negative association rule of the type $\neg X \rightarrow \neg Y$ is generated if its confidence is above the minimum confidence threshold (lines 15-16). On the other hand, if the correlation test gives a strong negative correlation, association rules of the types $X \rightarrow \neg Y$ and $\neg X \rightarrow Y$ are generated and appended to the set of association rules if their confidence is adequate. The result is compiled by combining all discovered positive and negative association rules. Lines 26 onward illustrate the automatic progressive thresholding for the correlation coefficient. If no rules are generated at a given correlation level, the threshold is lowered by 0.1 (line 27) and the process is re-iterated.

## 3.2 Related Work in Negative Association Rule Mining

In this section, we discuss two algorithms [WZZ02, THC02] that generate some form of positive and negative association rules. We compare our approach with them in the experimental section.

**Algorithm** Positive and Negative Association Rules Generation
**Input** TD, $minsupp$, $minconf$, and $\rho_{min}$, respectively Transactional Database, minimum support, minimum confidence, and correlation threshold.
**Output** AR: Positive and Negative Association Rules.
**Method:**

(0) **if** $\rho_{min}$ is undefined **then** $\rho_{min}$ = 0.5
(1) $positiveAR \leftarrow \emptyset$; $negativeAR \leftarrow \emptyset$ /*positive and negative AR sets*/
(2) scan the database and find the set of frequent 1-itemsets ($F_1$)
(3) **for** ($k = 2, F_{k-1} \neq \emptyset, k++$){
(4) $C_k$= $F_{k-1} \bowtie F_1$
(5) **foreach** $i \in C_k$ {
(6) s=support(TD,$i$) /*support of item $i$ is computed*/
(7) **if** s$\geq$ $minsupp$ **then**
(8) $F_k \leftarrow F_k \cup \{i\}$ /*item i is added to $F_k$*/
(9) **foreach** X,Y ($i = X \cup Y$) {
(10) $\rho$=correlation(X,Y) /*correlation btw X and Y is computed*/
(11) **if** $\rho \geq \rho_{min}$ **then**
(12) **if** s$\geq$ $minsupp$ **then**
(13) **if** $confidence(X \rightarrow Y) \geq minconf$ **then**
(14) $positiveAR \leftarrow positiveAR \cup \{X \rightarrow Y\}$
(15) **else if** $confidence(\neg X \rightarrow \neg Y) \geq minconf$ and
$support(\neg X \neg Y) \geq minsupp$ **then**
(16) $negativeAR \leftarrow negativeAR \cup \{\neg X \rightarrow \neg Y\}$
(17) **if** $\rho \leq -\rho_{min}$ **then** /*$\rho < 0$ and $|\rho| \geq \rho_{min}$ */
(18) **if** $confidence(X \rightarrow \neg Y) \geq minconf$ **then**
(19) $negativeAR \leftarrow negativeAR \cup \{X \rightarrow \neg Y\}$
(20) **if** $confidence(\neg X \rightarrow Y) \geq minconf$ **then**
(21) $negativeAR \leftarrow negativeAR \cup \{\neg X \rightarrow Y\}$
(22) }
(23) }
(24) }
(25) $AR \leftarrow positiveAR \cup negativeAR$
(26) **if** $AR = \emptyset$ **then** {
(27) $\rho_{min} = \rho_{min} - 0.1$
(28) **if** $\rho_{min} \geq 0$ **then go to** step (3)
(29) }
(30) **return** AR

Figure 3.1: Discovering positive and negative confined association rules

### 3.2.1 Negative Association Rule Algorithms

The first algorithm we discuss is proposed by Wu *et. al* [WZZ02]. They add on top of the support-confidence framework another measure called *mininterest* (the argument is that a rule $A \rightarrow B$ is of interest only if $supp(A \cup B) - supp(A)supp(B) \geq mininterest$). The authors consider as itemsets of interest those itemsets that exceed minimum support and minimum interest thresholds. Although, [WZZ02] introduces the *"mininterest"* parameter, the authors do not discuss how to set it and what would be the impact on the results when changing this parameter. The approach differs from our algorithm in that we use the correlation coefficient as measure of interestingness, which was thoroughly studied in the statistics community. In addition, the value of our parameter is well defined and it is not as sensitive to the dataset as the *mininterest* parameter. In our algorithm (line 9) we compute the correlation coefficient for every pair X,Y of an itemset $i$ where $i = X \cup Y$. As described earlier, when such a pair is found correlated an association rule is generated from it. In [WZZ02], they compute the interest for every pair X,Y of the itemset $i$ where $i = X \cup Y$. However, they extract rules from itemset $i$ only if any expression $i = X \cup Y$ exceeds the minimum interest threshold. In addition, in our algorithm the candidate set $C_k$ is generated as a join between $F_{k-1}$ and $F_1$. In [WZZ02] the candidate set $C_k$ is generated as a union of two frequent itemsets in $F_i$ for $1 \leq i \leq k-1$. This turns out to be expensive. Since we all make the assumption that a k-itemset must have all its subsets in $F_{k-1}$ we prove in the next theorem that our join generates the same itemsets as in [WZZ02].

**Theorem** All candidate items $c \in C_k$ generated by $F_i \bowtie F_j, 1 \leq i, j \leq k-1$ for which $\exists t \in c$ such that $t \in F_{k-1}$, can be discovered by $F_{k-1} \bowtie F_1$ .

**Proof** Let us suppose $\exists c \in C_k$ such that $c \in F_i \bowtie F_j, 1 \leq i, j \leq k-1$ and $c \notin F_{k-1} \bowtie F_1$. Given the condition stated in theorem $\exists t \in c$ such that $t \in F_{k-1}$. Since $c \notin F_{k-1} \bowtie F_1$ and $t \in F_{k-1}$ it follows that $c - t \notin F_1$. This is false as $c - t$ is of length one and $c \in C_k$ was generated from frequent itemsets. Thus $\forall c \in C_k, c \in F_{k-1} \bowtie F_1$. Q.E.D

The second algorithm we discuss was proposed in [THC02]. The algorithm is named SRM (substitution rule mining) by the authors. We refer to it in the same way throughout the chapter. The authors develop an algorithm to discover negative associations of the type $X \rightarrow \neg Y$. These association rules can be used to discover which items are substitutes for others in market basket analysis. Their algorithm discovers first what they call *concrete items*, which are those itemsets that have a high chi-square value and exceed the expected support. Once these itemsets are discovered, they compute the correlation coefficient for each pair of them. From those pairs that are negatively correlated, they extract the desired rules (of the type $X \rightarrow \neg Y$, where $Y$ is considered as an atomic item). Although interesting for the substitution items application, SRM is limited in the kind of rules that it can discover.

Using the next example, we present some of the differences among the three algorithms.

**Example 2.** Let us consider the small transactional table with 10 transactions and 6 items shown in Table 3.3. To illustrate the challenges in mining negative association rules we create another transactional database where for each transaction, the complement of each missing item is appended to it. The new created dataset is shown in Table 3.4. This new database can be mined with the existing association rule mining algorithms. However, there are a few drawbacks of this naïve approach. In practice, the data collections are very large, thus adding all the complemented items to the original database requires a large

| TID | Items |
|---|---|
| 1 | A,C,D |
| 2 | B,C |
| 3 | C |
| 4 | A,B,F |
| 5 | A,C,D |
| 6 | E |
| 7 | B,F |
| 8 | B,C,F |
| 9 | A,B,E |
| 10 | A,D |

Table 3.3: TD

| TID | Items | Equivalent bit vector |
|---|---|---|
| 1 | $A, \neg B, C, D, \neg E, \neg F$ | (101100) |
| 2 | $\neg A, B, C, \neg D, \neg E, \neg F$ | (011000) |
| 3 | $\neg A, \neg B, C, \neg D, \neg E, \neg F$ | (001000) |
| 4 | $A, B, \neg C, \neg D, \neg E, F$ | (110001) |
| 5 | $A, \neg B, C, D. \neg E, \neg F$ | (101100) |
| 6 | $\neg A, \neg B, \neg C, \neg D, E, \neg F$ | (000010) |
| 7 | $\neg A, B, \neg C, \neg D, \neg E, F$ | (010001) |
| 8 | $\neg A, B, C, \neg D, \neg E, F$ | (011001) |
| 9 | $A, B, \neg C, \neg D, E, \neg F$ | (110010) |
| 10 | $A, \neg B, \neg C, D, \neg E, \neg F$ | (100100) |

Table 3.4: TD

| Correlation | Interest | Concrete |
|---|---|---|
| AD | AD | AD |
| BF | BF | BF |
| BD | BD | BD |
| CE | CE | |
| | DF | |

Table 3.5: 2-itemsets

| Correlation | Interest | Concrete |
|---|---|---|
| ACD | | ACD |
| ABC | ABC | |
| ABD | | ABD |
| BCD | | |

Table 3.6: 3-itemsets

storage space. Not only the storage space has to increase considerably, but the execution times as well, in particular when the number of unique items in the database is very large. In addition, many association rules would be generated, many of them being of no interest to the applications at hand.

Using a minimum support of 0.2, the following itemsets are discovered using the three discussed algorithms. For this example the *correlation coefficient* was set to 0.5, and the *minimum interest* to 0.07. In Table 3.5 and Table 3.6, the first column presents the results when our approach was used. The second column uses the algorithm from[WZZ02], while in the third one the results are obtained using the approach in [THC02]. In both tables the positive itemsets are separated by the negative ones by a double horizontal line. The positive itemsets are in the upper part of the tables. As it can be seen, for the 2-itemsets all three algorithms find the same positive ones. The differences occur for the negative itemsets. The itemset *DF* has a *minimum interest* of 0.09, but it has a *correlation* of only 0.42. That is why it is not found by our approach or by the SRM algorithm. The itemset *CE* is not found by SRM because their condition is that the itemset should have higher correlation than the minimum value. In our approach the condition is to be greater or equal. Since the itemset *CE* has a *correlation* of 0.5 it is discovered by our algorithm, but not by SRM.

In Table 3.6 there are differences for both, the positive and the negative ones. The algorithm that uses the *minimum interest* parameter discovers only the *ABC* itemset because it is the only one that has all the pairs X,Y of the item *ABC* where $ABC = X \cup Y$ above the parameter. Although all the other itemsets discovered by the other algorithms have at least two strong pairs they are not considered of interest. Our approach and SRM generate the same positive 3-itemset. The itemsets *BCD* and *ABC* are not discovered by SRM because none of its subsets of two items are generated as concrete during the process.

From the itemsets that were shown in Table 3.5 and Table 3.6 a set of association rules

| | | #rules | support | confidence | PS | Q | IS | J |
|---|---|---|---|---|---|---|---|---|
| Correlation | 0.3 | 285 | **0.23±0.03** | 0.77±0.19 | 0.12±0.03 | 0.84±0.26 | 0.75±0.10 | 0.58±0.15 |
| | 0.4 | 235 | **0.23±0.03** | 0.79±0.16 | 0.14±0.02 | 0.84±0.28 | 0.78±0.08 | 0.63±0.12 |
| | 0.5 | 219 | 0.22±0.02 | **0.80±0.15** | **0.14±0.01** | 0.85±0.29 | **0.79±0.07** | **0.65±0.11** |
| Interest | 0.05 | 295 | **0.23±0.03** | 0.76±0.20 | 0.12±0.03 | 0.82±0.27 | 0.74±0.10 | 0.57±0.15 |
| | 0.07 | 219 | **0.23±0.03** | 0.79±0.18 | 0.13±0.02 | **0.85±0.27** | 0.76±0.09 | 0.61±0.14 |
| | 0.09 | 93 | 0.23±0.04 | 0.79±0.17 | 0.13±0.02 | 0.84±0.30 | 0.77±0.08 | 0.62±0.11 |
| SRM | all | 297 | 0.22±0.03 | 0.76±0.20 | 0.12±0.03 | 0.82±0.27 | 0.73±0.10 | 0.57±0.15 |

Table 3.7: Results for rules of type $X \rightarrow Y$

can be generated. Here we show, some of the rules that were generated from the itemsets that were discovered by one algorithm, but not by others. From itemset CE, the association rule $\neg E \rightarrow C$ can be found with support 0.5 and confidence of 62%. This rule seems to be strong, but it is missed by the SRM algorithm. From itemset DF, which is discovered only by the *mininterest* algorithm, the association rules $\neg D \rightarrow F$ and $D \rightarrow \neg F$ can be discovered. However, both rules have support 0.3 and confidence of 42%. These rules could have been eliminated when the confidence threshold is set to 50%, thus our approach and SRM do not miss much by not generating them. In addition, our approach generates the 3-itemset BCD. From this itemset the rule $B \rightarrow \neg C \neg D$ is discovered and it has support of 0.2 and confidence of 60%.

### 3.2.2 Experimental Results

We conducted our experiments on a real dataset to study the behaviour of the compared algorithms. We use the Reuters-21578 text collection [Reu08] considering only the ten largest categories.

We compare the three algorithms discussed in the previous sections. For each algorithm a set of values for their main interestingness measure was used in the experiments. Our algorithm and SRM [THC02] had the correlation coefficient set to 0.5, 0.4 and 0.3. In [WZZ02] the authors used the value 0.07 in their examples. We used this value and two others in its vicinity (0.05, 0.07 and 0.09). Each algorithm was run to generate a set of association rules. Throughout this section we will refer to our algorithm as *Correlation*, to the approach proposed in [WZZ02] as *Interest* and to the algorithm introduced in [THC02] as *SRM*.

In [TKS02] a set of measures for evaluating the interestingness of a patterns are compared and discussed. These measures apply to a single rule and they measure the strength between the antecedent and the consequent. The measures are clustered with respect to their similarity. We chose to compute a few measures from different clusters on the generated rules to ensure the diversity of evaluation: support, confidence, Piatetsky-Shapiro measure (PS), Yule's Q (Q), cosine measure (IS) and the Jaccard measure (J). For more details on these measures for frequent patterns see [TKS02].

Tables 3.7, 3.8, 3.9 and 3.10 show the results obtained for the Reuters dataset. We conducted the experiments with 20% support threshold and no constraint on the confidence threshold. In each table a subset of the obtained rules are compared. Table 3.7 reports the results for rules of the type $X \rightarrow Y$, Table 3.8 for rules of the type $X \rightarrow \neg Y$, Table 3.9 for rules of the type $\neg X \rightarrow Y$ and Table 3.10 for rules of the type $\neg X \rightarrow \neg Y$. In each table the average of the measurement and the standard deviation are reported. The value in bold represents the best value for each measure.

| | | #rules | support | confidence | PS | Q | IS | J |
|---|---|---|---|---|---|---|---|---|
| Correlation | 0.3 | 118 | 0.25±0.04 | 0.99±0.02 | 0.10±0.02 | 0.50±0.47 | 0.67±0.04 | 0.46±0.06 |
| | 0.4 | 6 | 0.33±0.10 | **0.99±0.0** | 0.11±0.01 | **0.99±0.0** | **0.72±0.05** | **0.52±0.08** |
| | 0.5 | 2 | **0.34±0.13** | **0.99±0.0** | **0.11±0.0** | **0.99±0.0** | 0.72±0.07 | 0.52±0.09 |
| Interest | 0.05 | 10 | 0.27±0.04 | 0.95±0.05 | 0.07±0.01 | 0.75±0.29 | 0.60±0.04 | 0.38±0.05 |
| | 0.07 | 4 | 0.25±0.01 | 0.98±0.02 | 0.08±0.01 | 0.70±0.47 | 0.62±0.03 | 0.39±0.03 |
| | 0.09 | 1 | 0.25±0.0 | **0.99±0.0** | **0.11±0.0** | **0.99±0.0** | 0.67±0.0 | 0.45±0.0 |
| SRM | 0.3 | 116 | 0.19±0.09 | 0.74±0.37 | 0.10±0.01 | 0.49±0.47 | 0.67±0.04 | 0.46±0.06 |
| | 0.4 | 6 | 0.33±0.10 | **0.99±0.0** | 0.11±0.01 | **0.99±0.0** | **0.72±0.05** | **0.52±0.08** |
| | 0.5 | 2 | **0.34±0.13** | **0.99±0.0** | **0.11±0.0** | **0.99±0.0** | 0.72±0.07 | 0.52±0.09 |

Table 3.8: Results for rules of type $X \rightarrow \neg Y$

| | | #rules | support | confidence | PS | Q | IS | J |
|---|---|---|---|---|---|---|---|---|
| Correlation | 0.3 | 118 | 0.25±0.04 | 0.35±0.05 | 0.10±0.02 | 0.49±0.47 | 0.67±0.04 | 0.46±0.06 |
| | 0.4 | 6 | 0.33±0.10 | 0.49±0.08 | 0.11±0.01 | **0.99±0.0** | 0.72±0.05 | 0.52±0.08 |
| | 0.5 | 2 | 0.34±0.13 | 0.52±0.10 | **0.11±0.0** | **0.99±0.0** | 0.72±0.07 | 0.52±0.09 |
| Interest | 0.05 | 10 | 0.28±0.06 | 0.40±0.08 | 0.07±0.01 | 0.75±0.29 | 0.61±0.06 | 0.39±0.07 |
| | 0.07 | 4 | 0.34±0.06 | 0.46±0.09 | 0.08±0.01 | 0.70±0.47 | 0.67±0.06 | 0.45±0.08 |
| | 0.09 | 1 | **0.44±0.0** | **0.59±0.0** | **0.11±0.0** | **0.99±0.0** | **0.77±0.0** | **0.59±0.0** |

Table 3.9: Results for rules of type $\neg X \rightarrow Y$

Table 3.7 shows that for positive association rules our approach tends to generate a more interesting set of rules compared to the other methods. *SRM* generates the same set of positive rules for all parameters, since it produces the same set of association rules as the algorithms that use the support-confidence framework. Based on confidence, Piatetsky-Shapiro, cosine and Jaccard measures, our algorithm generates the most interesting set of rules. Based on support, both *Correlation* and *Interest* generate equally interesting rules, with *SRM* generating a slightly less interesting set of rules. Yule's Q measure indicates that *Interest* generates the most interesting rules, followed closely by *Correlation* with the same rule set interestingness, but a slightly higher standard deviation.

For rules of type $X \rightarrow \neg Y$ our approach and *SRM* perform best based on the interestingness measures, as shown in Table 3.8. They produce the same set of rules for correlation values of 0.5 and 0.4, the interestingness measures having the same values for the two algorithms. When the correlation threshold is lowered to 0.3 our algorithm finds better rules than *SRM* as expressed by the interestingness measures. When compared against *Interest*, *Correlation* generates equally interesting set of rules based on three of the measures. However, the support, cosine and Jaccard measure indicate that the rules generated by *Correlation* are more interesting by a large margin than those generated by *Interest*.

Table 3.9 and Table 3.10 compare our approach with *Interest* only, since *SRM* does not generate rules of the types $\neg X \rightarrow Y$ and $\neg X \rightarrow \neg Y$.

In Table 3.9 the symmetric rules of the ones in Table 3.8 are generated, since the confidence threshold is not applied during rule generation and the correlation and minimum interest are computed for the XY itemset. According to the interestingness measures, the best set of rules is generated for the *Interest* algorithm. However, the measures are calculated over one rule, which makes a conclusion more difficult. If we consider only the sets with more than 2 rules, the rule sets generated by *Correlation* are more interesting by a large margin according to five of the six measures.

For the rules of type $\neg X \rightarrow \neg Y$, *Interest* generates a smaller set of rules than *Correlation* with higher values on five of the six interestingness measures. Based on Yule's Q

| | | #rules | support | confidence | PS | Q | IS | J |
|---|---|---|---|---|---|---|---|---|
| Correlation | 0.3 | 2006 | 0.31±0.08 | 0.41±0.13 | 0.14±0.03 | 0.79±0.24 | 0.77±0.08 | 0.62±0.12 |
| | 0.4 | 1474 | 0.31±0.09 | 0.41±0.13 | 0.15±0.02 | 0.84±0.20 | 0.80±0.06 | 0.66±0.10 |
| | 0.5 | 1088 | 0.31±0.09 | 0.41±0.14 | 0.16±0.02 | **0.88±0.18** | 0.82±0.06 | 0.69±0.09 |
| Interest | 0.05 | 818 | 0.31±0.10 | 0.44±0.16 | 0.14±0.04 | 0.82±0.29 | 0.80±0.10 | 0.66±0.14 |
| | 0.07 | 148 | 0.49±0.09 | 0.67±0.13 | 0.16±0.05 | 0.81±0.39 | 0.87±0.08 | 0.77±0.11 |
| | 0.09 | 105 | **0.54±0.04** | **0.74±0.08** | **0.17±0.04** | 0.87±0.31 | **0.90±0.05** | **0.82±0.08** |

Table 3.10: Results for rules of type $\neg X \rightarrow \neg Y$

measure, *Correlation* generates a slightly more interesting set of rules.

Overall, none of the three algorithms generates sets of rules that are more interesting that those generated by the other methods based on all measures. *SRM* does not generate two of the four types of rules considered, limiting its applicability. Our algorithm generates more interesting sets of rules for rules of types $X \rightarrow Y$, $X \rightarrow \neg Y$ and $\neg X \rightarrow Y$, while interestingness measures suggest that *Interest* generated more interesting rule of type $\neg X \rightarrow \neg Y$.

### 3.2.3 Summary

In summary, we introduced a new algorithm to generate both positive and negative association rules. Our method adds to the support-confidence framework the correlation coefficient to generate stronger positive and negative rules. We compared our algorithm with other existing algorithms on a real dataset. We discussed their performances on a small example for a better illustration of the algorithms and we presented and analyzed experimental results for a text collection. The results prove that our algorithm can discover strong patterns. In addition, our method generates all types of confined rules, thus allowing it to be used in different applications where all these types of rules could be needed or just a subset of them.

## 3.3 Using Negative Association Rules in Associative Classifiers

### 3.3.1 Associative Classification

The set of rules that were generated as discussed in the previous section represent the actual classifier. This categorizer is used to predict to which classes new objects are attached. Given a new object, the classification process searches in this set of rules for those classes that are relevant to the object presented for classification. The set of positive and negative rules discovered as explained in the previous section are ordered by confidence and support. This sorted set of rules represents the associative classifier ARC-PAN (Association Rule Classification with Positive And Negative). This subsection discusses the approach for labeling new objects based on the set of association rules that forms the classifier.

In the algorithm (Classification of a new object), a set of applicable rules is selected within a confidence margin. The interval of selected rules is between the confidence of the first rule and this confidence minus the confidence margin as checked in line 7. The prediction process starts at line 10. The applicable set of rules is divided according to the classes in line 10. In lines 11-12 the groups are ordered according to the average confidence per class. In line 13 the classification is made by assigning to the new object the class that has the highest score.

**Algorithm** Classification of a new object
**Input** A new object to be classified $o$;
The associative classifier (ARC-PAN);
The confidence margin $\tau$;
**Output** Category attached to the new object
**Method:**

(1) $S \leftarrow \emptyset$ /*set of rules that match $o$*/
(2) **foreach** $r$ in ARC-PAN /*the sorted set of rules*/
(3) **if** ($r \subset o$) { count++ }
(4) **if** (count == 1)
(5) fr.conf $\leftarrow$ r.conf /*keep the first rule confidence*/
(6) $S \leftarrow S \cup r$
(7) **else if** (r.conf $>$ fr.conf-$\tau$)
(8) $S \leftarrow S \cup r$
(9) **else** break
(10) divide S in subsets by category: $S_1, S_2 ... S_n$
(11) **foreach** subset $S_1, S_2 ... S_n$
(12) $score_i = \frac{\sum r.conf}{\#rules}$
(13) $o \leftarrow C_i$, with $score_i = max\{score_1..score_n\}$

The association rules of the type $X \rightarrow C$ and $\neg X \rightarrow C$ can be treated in the same way. Both of them have a confidence attached and they have an association with the class label. These types of rules can be considered together and their confidence can be added to the C class total. However, the rules of the type $X \rightarrow \neg C$ have to be treated differently. We chose to subtract their confidences from the total confidence of their corresponding class.

### 3.3.2 Experimental Results

We tested our algorithm on some datasets from UCI ML Repository [BM98]. On each dataset we performed C4.5's shuffle utility [Qui93] for shuffling the datasets. The shuffle ensures a more accurate classification as observations become randomized. A 10-fold cross validation was performed on each dataset and the results are given as average of the errors obtained for each fold. In addition, to have a fair comparison with the other algorithms that we wanted to compare, we used the same discretization method for continuous attributes as in [LHM98]. The parameters for C4.5 were set to their default values. For all three association rule based methods the minimum support was set to 1% and the minimum confidence to 50%. In our approach the confidence margin was set at 10%.

The experimental results are shown in Table 3.11. The average error results for CBA and C4.5 are taken from [LHM98]. The last three columns give the results for our classification method. Column *rules+* presents the results when only strong positive rules are considered for classification. Column *rules+-* shows the results when positive rules and negative rules of the form $\neg X \rightarrow C$ are considered. Column *rulesall* lists the results when all rules as described in the previous sections are considered for categorization.

As presented in Table 3.11 the results for classification with the classifier based on the positive and negative rules (ARC-PAN) are encouraging. When all types of rules are used

| Datasets | c4.5 rules | CBA | | ARC-PAN | | |
|---|---|---|---|---|---|---|
| | | w/o prun | prun | rules+ | rules+- | rules_all |
| breast | 3.9 | 4.2 | 4.2 | 5.5 | 4.8 | **3.8** |
| diabetes | 27.6 | 24.7 | 25.3 | **23.3** | 25.4 | 25.1 |
| heart | 18.9 | 18.5 | 18.5 | 16.3 | 17.0 | **16.2** |
| iris | **5.5** | 7.1 | 7.1 | 6.6 | 6.6 | 6.0 |
| led7 | **26.5** | 27.8 | 27.8 | 28.7 | 28.7 | 28.9 |
| pima | 27.5 | 27.4 | 27.6 | 27.4 | 27.1 | **26.9** |

Table 3.11: Classification Results (Error rates)

| Datasets | strong rules | | correlated rules | |
|---|---|---|---|---|
| | #rules | error | #rules | error |
| breast | 17000 | 5.0 | 1000 | 5.5 |
| diabetes | 4000 | 21.8 | 40 | 23.3 |
| heart | 200000 | 24.7 | 80 | 16.3 |
| iris | 140 | 7.3 | 60 | 6.6 |
| led7 | 4000 | 34.3 | 500 | 28.7 |
| pima | 4000 | 22.0 | 50 | 27.4 |

Table 3.12: Comparison in number of rules

the classification accuracy increases on three datasets when compared with the state-of-the-art classifier C4.5 and with CBA. The first column under ARC-PAN shows that the classification accuracy can be improved as well with only the generation of positive association rules that are strongly correlated. We ran classification with the negative rules only as well, but the results decreased in this case. The results are not presented in the table.

Table 3.12 shows the drastic reduction in number of rules when the correlation measure is used to derive interesting rules. The second column in the table shows the approximate (averaged over 10-folds) number of rules derived in the "support-confidence" framework. As one can see from the table, when compared to the fourth column (where the number of rules discovered in the correlation framework are listed), there is a large decrease in the number of rules from one framework to the other. Moreover, as observed from the error results presented in the third and fifth columns, the error rate remains in the same range, or even decreases in some cases.

A small number of classification rules is very desirable. When a small set of strong classification rules is presented, the classification phase is faster, which can be important for some applications. Another advantage is that a small set becomes human readable. It is realistically feasible to read, edit and augment hundreds of rules, but thousands of rules is impractical. Because of the transparency of the associative classifier, manually updating some rules is favorable and practical in many applications.

### 3.3.3 Summary

In this chapter we introduced a new algorithm to generate positive and negative associations discovered in transactional data. The interestingness measure that our algorithm relies on is the correlation coefficient. We demonstrated the potential of strong positive and negative correlated rules in the classification context. The results of our experiments show that a

much smaller set of positive and negative association rules can perform similar or outperform existing categorization systems.

# Chapter 4

# Associative Classifiers Using Association Rules with Re-occuring Items

Given the binary nature of association rules, associative classifiers do not take into account repetition of features when categorizing. In this chapter, we enhance the idea of associative classifiers with associations with re-occurring items and show that this mixture produces a good model for classification when repetition of observed features is relevant in the data mining application at hand. We motivate the importance of re-occurrence, we discuss the rule generation in this context and we present a classifier based on this type of rule. This work was published in [RSZA05] in a collaboration with Rafal Rak and Wojciech Stach.

## 4.1 Introduction

One considerable limitation of associative classifiers is that they do not handle observations with repeated features. In other words, if a data object is described with repeated features, only the presence of the feature is considered, but not its repetition. However, in many applications such as medical image categorization or other multimedia classification problems, the repetition of the feature may carry more information than the existence of the feature itself [ZHZ00]. Also in text mining and information retrieval, it is widely recognized that the repetition of words is significant and symptomatic, hence the common use of TF/IDF (i.e. the frequency of a term in a document relative to the frequency of the term in a collection) for feature representation in text classification.

Associative classifiers use association rule mining to build a classification model. However, association rule mining typically considers binary transactions; transactions that indicate presence or absence of items. Binary transactions simply do not model repetitions. A few approaches to mining association rules with re-occurring items have been proposed, such as MaxOccur [ZHZ00], FP'-tree [ONL01] and WAR [WYY04]. Association rules with re-occurring items differ from quantitative association rules [SA96, AL99] as follows: association rules with re-occurring items are discovered from discrete attributes and the occurrences of the attributes are taken into account, while quantitative association rules are discovered from continuous attributes. Quantitative association rules discover associations among discrete intervals of the continuous attributes. The focus of this chapter is to devise a classifier that combines the idea of associative classification and association rules with reoc-

curring items. Our contributions presented in this chapter exploit, combine, and extend the ideas mentioned above, especially ARC-BC [ZA02] and MaxOccur [ZHZ00] algorithms. We also suggest new strategies to select rules for classification from the set of discovered association rules. Our hypothesis is that associative classifiers with recurrent items have more discriminatory power since they maintain and exploit more information about both objects and rules.

A delicate issue with associative classifiers is the use of a subtle parameter: support. Support is a difficult threshold to set, inherited from association rule mining. It indicates the proportion of the database transactions that support the presence of an item (or object). The support threshold is hard to tune in practice and its value depends on the application at hand. In the associative classification literature it has been commonly and arbitrarily set to 1%. However, the accuracy of the classifier can be very sensitive to this parameter. In the case of re-occurring items, there are two ways of calculating support: transaction-based support and object-based support [ZHZ00] (i.e. either the proportion of transactions or the proportion of objects that support the existence of an object in the database). Our experiments show that an associative classifier that considers re-occurrence of features is considerably less sensitive to the variation of support. This leads to more practical applications and eventually the possibility to automatically determine and tune this parameter.

## 4.2 Problem Statement and Related Work

Current associative classifiers use transactions in the form of $< \{i_1, i_2, ...i_n\}, c >$, where $i_k$ is an item in a transaction and $c$ is a class label. Our task is to combine the associative classification with the problem of recurrent items. More formally, it can be stated that our goal is to modify the original approach using transactions to the form of $< \{o_1 i_1, o_2 i_2, ...o_n i_n\}, c >$, where $o_k$ is the number of the occurrences of the item $i_k$ in the transaction.

Association rules have been recognized as a useful tool for finding interesting hidden patterns in transactional databases. Several different techniques have been introduced. However less research has been done considering transactions with re-occurrence of items. In [WYY04], the authors assign weights to items in transactions and introduce the WAR algorithm to mine the rules. This method has two steps: in the first step frequent itemsets are generated without considering weights and then weighted association rules (WARs) are derived from each of these itemsets. MaxOccur algorithm [ZHZ00] is an efficient Apriori-based method for discovering association rules with recurrent items. It reduces the search space by effective usage of joining and pruning techniques. The FP'-tree approach presented in [ONL01] extends the FP-tree design [HPY00] with a combination from the MaxOccur idea. For every distinct number of occurrences of a given item, a separate node is created. When a new transaction is inserted into the FP'-tree, it might increase support count for the different path(s) of the tree as well. This is based on the intersection between these two itemsets. Given the complete tree, the enumeration process to find frequent patterns is similar to that of the FP-tree approach [HPY00]. In spite of the existing work on generating association rules with re-occurring items, ACRI (the system presented in this chapter) is the first associative classifier using such rules. Recently, Rak at al. [RKR07] developed an application-specific associative classifier. The system is based on ACRI and it assigns MeSH keywords to articles in a multilabel classification task. Another work on generating and using association rules with re-occurring items is presented in [RKR08]. The authors propose a new algorithm for generation of classification rules that generates rules

with one or more class labels in the consequent and takes into account recurrent items. Although the paper's main contribution is in the rule generation phase, the authors also show and discuss a classification system based on these rules.

## 4.3 The Proposed Approach

Our approach, ACRI (Associative Classifier with Reoccurring Items), consists of two modules: rule generator and classifier. We decided to base our algorithm for mining associations with reoccurring items on Apriori-based MaxOccur. The building of the classification model follows the ARC-BC approach detailed in Section 2.3.3. The rationale is based on the efficiency of this method in the case of non-evenly distributed class labels. Indeed other associative classification methods are biased towards dominant classes in the case when rare classes exist. Running MaxOccur on transactions from each known class separately makes the core of our rule generator module. It mines the set of rules with reoccurring items from the training set. These rules associate a condition set with a class label such that the condition set may contain items preceded by a repetition counter. The classification process might be considered as plain matching of the rules in the model to the features of an object to classify. Different classification rules may match, thus the classifier module applies diverse strategies to select the appropriate rules to use. In addition, simple matching is sometimes not possible because there is no rule that has the antecedent contained in the feature set extracted from the object to classify. With other associative classifiers, a default rule is applied, either the rule with the highest confidence in the model or simply assigning the label of the dominant class. Our ACRI approach has a different strategy allowing partial matching or closest matching by modeling antecedents of rules and new objects in a vector space. The following elaborates on both modules.

### 4.3.1 Rule generator

This module is designed for finding all frequent rules of the form $< \{o_1i_1, o_2i_2, \ldots, o_ni_n\}, c >$ from a given set of transactions. The modules's general framework is based on ARC-BC [ZA02]: transactions are divided into N subsets - each for one given class (N is the number of classes); once rules are generated for each individual class, the rules are merged to form a classification model. The rule generator for each class $C_x$ is an Apriori-based algorithm for mining frequent itemsets that extends the original method by taking into account reoccurrences of items in a single transaction à la MaxOccur [ZHZ00]. In order to deal with reoccurrences, the support count was redefined. Typically, the support count is the number of transactions that contain an item. In our approach, a single transaction may increase the support of a given itemset by more than one. The formal definition of this approach is as follows. A transaction $T =< \{o_1i_1, o_2i_2, \ldots, o_ni_n\}, c >$ supports itemset $I = \{l_1i_1, l_2i_2, \ldots, l_ni_n\}$ if and only if $\forall i = 1..n \;\; l_1 \leq o_1 \wedge l_2 \leq o_2 \wedge \ldots \wedge l_n \leq o_n$. The number $t$ by which $T$ supports $I$ is calculated according to the formula: $t = min[\frac{o_i}{l_i}] \;\; \forall i = 1..n, l_i \neq 0 \wedge o_i \neq 0$.

### 4.3.2 The Classifier

This module labels new objects based on the set of mined rules obtained from the rule generator. An associative classifier is a rule-based classification system, which means that an object is labeled on the basis of a matched rule (or set of rules in case of multi-class

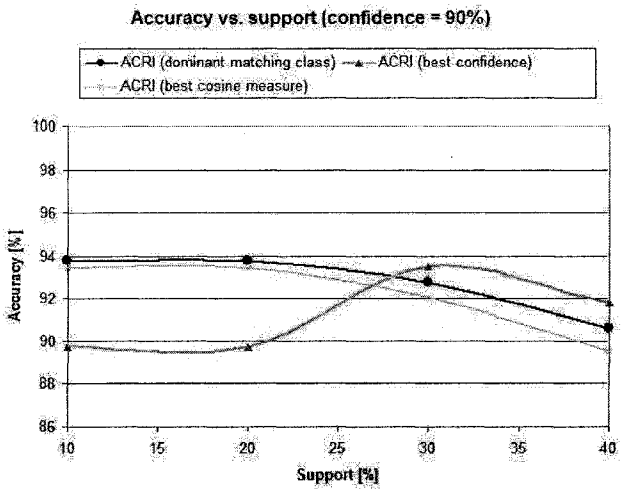


Figure 4.1: Accuracy of rule selection strategies versus support for the Mushroom dataset.

classification). This task is simple if there is an exact match between a rule and an object. The model, however, often does not include any rule that matches a given object exactly. In such a case, in order to make the classification, all rules are ranked according to a given strategy and the best one (or several) is matched to a given object. Rule ranking might be performed following different strategies, which associate each rule to a number that reflects its similarity to a given object. These strategies may be used either separately or in different combinations. We have tested *cosine measure, coverage, dominant matching class, support* and *confidence*. Let us consider the rule $< \{o_1 i_1, o_2 i_2, \ldots, o_n i_n\}, c >$ and the object to be classified $< l_1 i_1, l_2 i_2, \ldots, l_n i_n >$. The corresponding n-dimensional vectors can be denoted as $\overrightarrow{0} = [o_1, o_2, \ldots, o_n]$ and $\overrightarrow{l} = [l_1, l_2, \ldots, l_n]$. The *Cosine measure (CM)* assigns a similarity value that is equal to the angle between these two vectors, i.e. The smaller the CM value is, the smaller the angle, and the closer these vectors are in the n-dimensional space. *Coverage (CV)* assigns a similarity value that is equal to the ratio of the number of common items in the object and rule to the number of items in the rule (ignoring re-occurrences). In this case, the larger the CV ratio is, the more items are common for the rule and the object. CV=1 means that the rule is entirely contained (ignoring re-occurrences) in the object. With *Dominant matching class*, the class label is assigned to the object by choosing the one being the most frequent from the set of rules matching the new object. Notice that dominance can be counted by simply enumerating the matching rules per class or a weighted count using the respective confidences of the matching rules. The *support* and *confidence* are used to rank rules. They refer to the rule property only and do not depend on the classified object. Thus, they have to be used with other measures that prune the rule set.

## 4.4 Experiments

We tested ACRI on different datasets to evaluate the best rule selection strategy as well as compare ACRI with ARC-BC. Experiments were run on the Mushroom dataset from the UCI repository [BM98]. Our experimental evaluation shows that rule selection strategies have roughly similar performance in terms of accuracy. However, the accuracy varies with

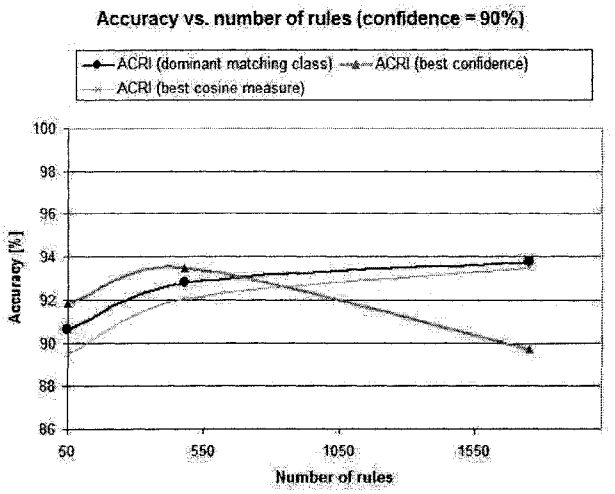


Figure 4.2: Accuracy of rule selection strategies versus number of rules for the Mushroom dataset.

the support threshold. The lower the support, the more rules are discovered allowing a better result using selection based on the cosine measure for example. Using the dominant matching class also did well, confirming the benefit of the dominance factor introduced in [ZA02]. We also observed that coverage (CV) gave better results when set to 1. Thus all results reported herein have CV set to 1. The other measures are comparable in performance and trend, except for best confidence. When the support threshold is high, fewer rules are discovered and confidence tends to provide better results while the cosine measure returns matches that have big angles separating them from the object to classify, hence the lower accuracy. Figure 4.1 shows the superiority of the rule selection strategy dominant matching class up to a support threshold of 25%, beyond which best confidence becomes a winning strategy. Figure 4.2 shows how the more rules are discovered the more effective in terms of accuracy the strategies dominant matching class and cosine measure becomes in comparison to best confidence approach. The number of rules is correlated with support.

We compare ACRI with ARC-BC using the Reuters-21578 text collection using the top 10 biggest topics [Reu08]. We produced several different sets of rules to be used in the classifier. For ARC-BC we varied the support threshold range from 10 to 30% with a step of 5%; and 15 to 65% support range with the same step for ACRI. The difference between the support thresholds lies in the definition of support for mining rules with recurrent items. A single document can support a set of words more than once. Therefore, if we consider support as the ratio of support count to the total number of transactions, as it was introduced in [ONL01], we may encounter support more than 100% for some itemsets. On the other hand, if we use the definition presented in [ZHZ00], i.e., the ratio of support count to the number of distinct items (words), the support will never reach 100%. In practice, the latter support definition requires setting very small thresholds to obtain reasonable results. Hence, we decided to use the first definition as it is more similar to the "classical" definition of support. It is important to notice that no matter which definition we choose, it eventually leads to setting the same support count with ARC-BC. For each support threshold we set three different confidence thresholds: 0, 35, and 70%. The latter threshold was used in

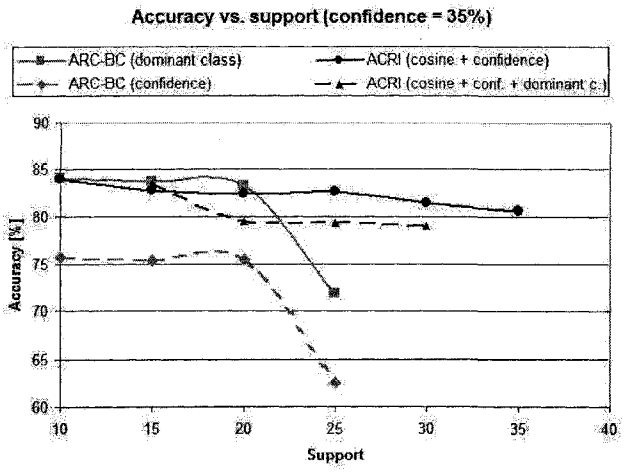


Figure 4.3: Accuracy of ACRI and ARC-BC with high support

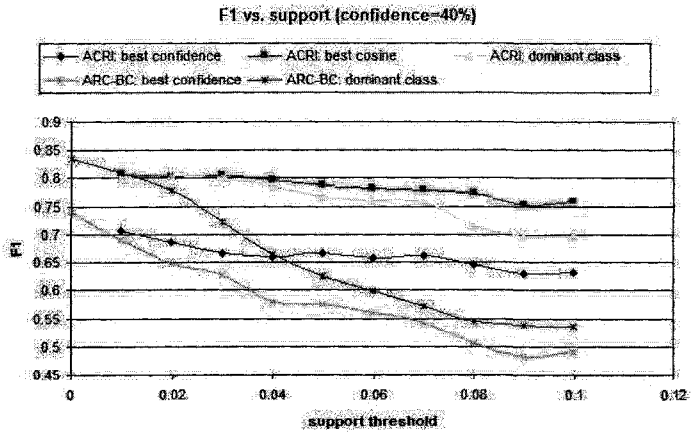


Figure 4.4: Effectiveness at low support

[ZA02] as minimum reasonable threshold for producing rules; the first one (no threshold) was introduced to observe the reaction of the classifier for dealing with a large number of rules; and the threshold of 35% is simply the middle value between the two others. For each single experiment we tried to keep the level of more then 98% of classified objects, which resulted in manipulating the coverage CV from 0.3 to 1. We discarded cases for which it was not possible to set CV to satisfy the minimum number of classified objects. More than 90% of the remaining results had CV = 1. We also performed experiments without specifying CV (using different methods of choosing applicable rules); however, they produced lower accuracy than those with CV $> 0.3$.

We used different classification techniques for choosing the most applicable rule matching the object. Best confidence and dominant matching class matching methods were utilized for both ARC-BC and ACRI approaches. Additionally, ACRI was tested with the cosine measure technique. For all reported experiments the coverage (CV) is set to 1. In other words, for a rule to be selected for classification, all features expressed in the an-

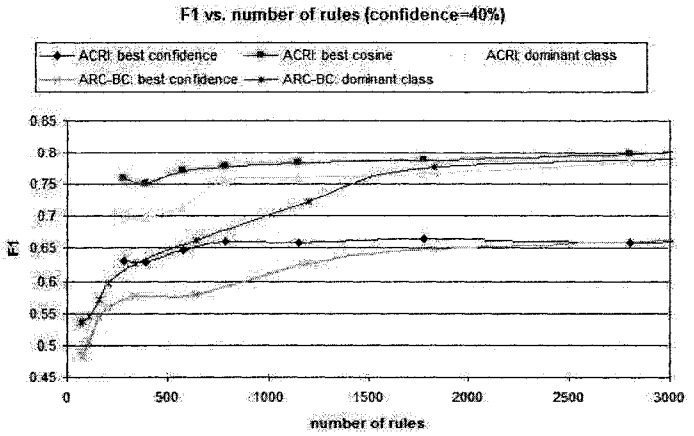


Figure 4.5: Effectiveness versus size of model

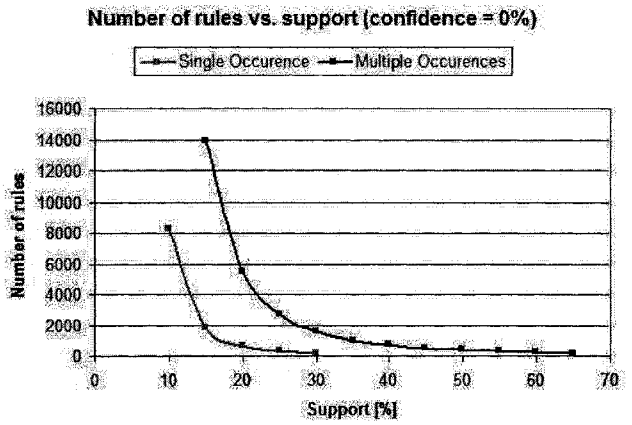


Figure 4.6: Number of rules with confidence > 0%

tecedent of the rule have to be observed in the new object to classify. We also performed tests with combination of matching techniques with different tolerance factors for each test. An example scenario in Figure 4.3, combines cosine measure, dominant matching class and best confidence: (1) choose top 20% of rules with the best cosine measure, then (2) choose 50% of the remaining rules with the highest confidence, and then (3) choose the rule based on the dominant class technique. We also did a battery of tests using relatively low supports. This significantly increases the number of classification rules. We varied the support between 0 and 0.1% and compared the harmonic average of precision and recall (F1 measure) for the same cases as before: best confidence and dominant matching class for both ARC-BC and ACRI approaches, and the cosine measure technique for ACRI (Figures 4.4 and 4.5).

Categorizing documents from the Reuters dataset was best performed when the confidence level of the rules was at the 35% threshold for both the ACRI and ARC-BC approaches. For the ARC-BC classifier, the best strategy was to use dominant factor, whereas in case of ACRI combination of cosine measure and confidence factors worked best. Figure

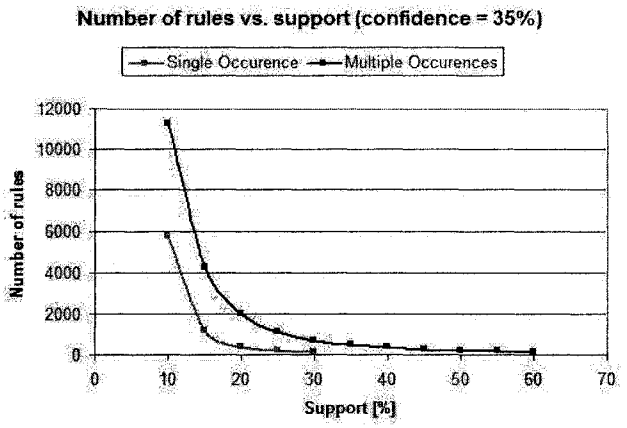


Figure 4.7: Number of rules with confidence > 35%

4.3 shows the relationship between support and accuracy for these approaches. Comparing the best-found results, ARC-BC slightly outperforms the ACRI using the dominant matching class strategy at the 20% support level. However, ARC-BC seems to be more sensitive to changes of the support threshold. The accuracy of ACRI virtually does not depend on the support threshold and is stable. In the case of ARC-BC the accuracy decreases significantly when this support is greater than 20%.

Figures 4.6 and 4.7 show the number of generated rules with and without recurrent items. Considering recurrences results in having more rules, this has its origin in the different support definition. The other interesting relationship is that by increasing the confidence threshold from 0% to 35%, the difference between the number of rules decreases more rapidly for ACRI.

Experiments using low support thresholds confirm the stability of ACRI with regard to support. When varying the support from 0% to 0.1% ARC-BC loses in precision and recall while ACRI remains relatively consistent or loses effectiveness at a slower pace. Figure 4.4 also shows that ACRI outperforms ARC-BC at these lower support thresholds. Using the cosine measure for selecting rules appears to be the best strategy. The cosine measure is also the best rule selection strategy when considering the number of rules discovered. In addition, the more rules that are available the more effective the cosine measure becomes at selecting the right discriminant rules.

Figure 4.8 shows the relationship between running time for the rule generator with and without considering recurrent items ('multiple support' refers to considering re-occurrences). The algorithm with recurrences is slower, since it has to search a larger space, yet the differences become smaller when increasing the support threshold.

The best results for ACR-BC were found in [ZA02] for confidence threshold greater than 70%. However, our experiments show that effectiveness is better on lower confidence for both ARC-BC and ACRI approaches. In other words, some classification rules with low confidence have more discriminant power and are selected by our rule selection strategies. This discrepancy with previous results may be explained by the use of the different method of counting support and confidence or/and by the fact that our classifier ACRI with re-occurring items and without re-occurrence consideration to simulate ARC-BC is using a different setup for rule selections.

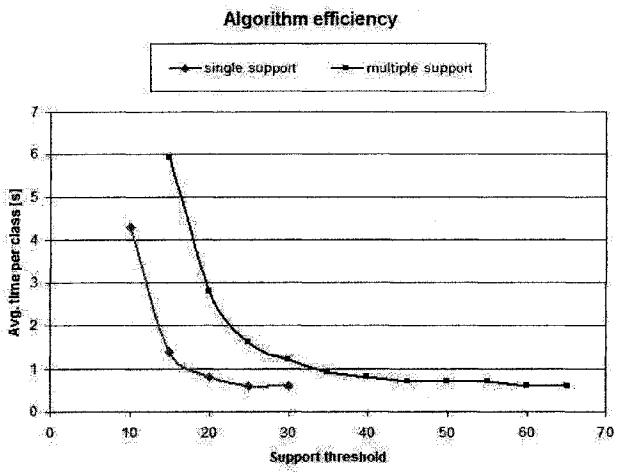


Figure 4.8: Algorithms CPU time efficiency

## 4.5 Summary

In this chapter we introduced the idea of combining associative classification and mining frequent itemsets with recurrent items. We combined these two and presented ACRI, a new approach to associative classification with recurrent items. We also suggest new strategies to select classification rules during the classification phase. In particular, using the cosine measure to estimate the similarity between the objects to classify and the available rules is found very effective for associative classifiers that consider re-occurrence. When comparing ACRI approach with ARC-BC we found that considering repetitions of observed features is beneficial. In particular in the case of text categorization, repetition of words has discriminant power and taking these repetitions into consideration can generate good classification rules. Our experiments also show that ACRI becomes more effective as the number of rules increases in particular with the cosine measure for rule selection. Moreover, the accuracy of ACRI seems to be less sensitive to the support threshold, while most associative classifiers are typically very sensitive to the support threshold which is very difficult to determine effectively in practice.

# Chapter 5

# Two Stage Architecture

The classification stage in associative classifiers is a very important step in the entire system. This is the step where the system uses the discovered classification rules to make new predictions. Several approaches have been considered with different systems. However, there are several limitations to these approaches. A detailed discussion of these schemes and their limitations will be presented in this chapter. Given the limitations of the existing scoring schemes, the focus of this chapter is to investigate this problem and develop a better scheme for rule usage. This chapter introduces our two stage architecture for the classification stage which represents the solution that we propose to address this problem.

## 5.1 Introduction

One deficiency of current associative classifiers is the selection and use of rules in the classification stage. Current systems assign classes to new objects based on the best rule applied or on some predefined scoring of multiple rules. In this chapter we propose a new technique where the system automatically learns how to select and use the rules. We achieve this by developing a two-stage classification model. First, we use association rule mining to discover classification rules. Second, we employ another learning algorithm to model the use of these rules in the prediction process. Our two-stage approach outperforms C4.5, RIPPER and the other associative classifiers (ARC, CBA, CMAR, CPAR) on the UCI datasets. The versatility of our method is also demonstrated by applying it to text classification, where it equals the performance of the best known systems for this task, SVMs.

To classify a given object, an associative classifier proceeds in three steps. First, it determines which of its rules apply to the object. Then it selects a subset of the applicable rules (possibly all of them) based on some measure of their "strength" or precedence. Finally, if it chooses more than one rule, it combines the class predictions of all the selected rules to produce a final classification.

For example, CBA [LHM98] classifies an object using only the highest ranking rule that applies to the object, where rank is defined by the rule's confidence on the training set. This method has two shortcomings. The first is that by basing its classification on only the highest ranking rule, CBA might be ignoring a large number of high ranking, applicable rules that might agree with each other and disagree with the highest ranking rule. Second, because each rule predicts just a single class, CBA is incapable of assigning a given object to multiple classes simultaneously, which is essential in some applications.

CMAR [LHP01] and ARC [AZ02] overcome CBA's shortcomings by selecting the $K$ highest ranking applicable rules, not just the first. The key issue in this case is how to combine the class predictions of the selected rules to produce a final classification? While both systems use a weighted voting scheme, they differ in the details of how weights are calculated. CMAR uses a chi-square weighting scheme, while ARC weighs classes based on the average confidence of the selected rules that predict that class. These systems trade part of their comprehensibility inherited from the association rules for improved performance. This trade-off is the result of using a weighting score on the rules.

In this chapter we also use a weighted voting scheme to combine the class predictions of the selected rules to produce a final classification, but instead of pre-defining the way in which weights are computed, we use a second learning algorithm to determine the weights. Therefore, in our system learning takes place in two stages. First, an associative classifier is learned using standard methods. Second, predefined features computed on the outputs of the rules in the learned associative classifier are used as the inputs to another learning system, which is trained (using a separate training set) to weigh the features appropriately to produce highly accurate classifications.

This two-stage system, with a layer of feature definitions interposed between the output of the first learned system and the input of the second, is the focus of this chapter. Further on, we investigate two types of features that can be used between the first and second learning stage. We evaluate the proposed approach through an extensive experimental study, followed by a statistical analysis of the results.

The performance of our two-stage system is evaluated experimentally on two distinct classification tasks, single label classification and multiple label classification. In single label classification, the task is to assign an object to exactly one of the classes. This is the standard classification task studied in machine learning, and a variety of test datasets and systems are available for comparison. We use twenty of the UCI datasets[BM98], and compare our approach to seven existing systems. Our two-stage approach outperforms C4.5, RIPPER and the other associative classifiers (ARC, CBA, CMAR, CPAR) on the UCI datasets. We prove that the performance increase is due to the automatically learned scoring scheme.

In multiple label classification, an object can be assigned to several of the classes simultaneously. The standard testbed for this task is classifying news articles into subject categories, where it is necessary for some news articles to be assigned to multiple categories. For example, an article about selling a sports franchise should be put into at least two categories, "sports" and "business". The best known algorithms for this text classification task are SVMs [Joa98]. Our experiments using the Reuters dataset establishes our two-stage system as the first classification method to equal the performance of SVMs on this task.

## 5.2 A Two-Stage Approach to Classification

Most of the classification techniques work as follows: given a set of examples with categories attached, a learning model is developed from a subset of the data (training set); the model created is then tested and validated on the remaining data (testing set).

Once developed, the purpose of a classification system is to classify new instances. In the case of associative classifiers, this step deals with using the rules to categorize a new object. To demonstrate the importance of the classification stage we show an example of

| |
|---|
| R1: D $\Rightarrow$ Class 1 - confidence 95% |
| R2: B $\wedge$ E $\Rightarrow$ Class 2 - confidence 90% |
| R3: A $\wedge$ D $\Rightarrow$ Class 1 - confidence 90% |
| R4: B $\wedge$ C $\Rightarrow$ Class 3 - confidence 90% |
| R5: A $\Rightarrow$ Class 1 - confidence 85% |
| R6: A $\wedge$ B $\Rightarrow$ Class 2 - confidence 85% |
| R7: B $\Rightarrow$ Class 2 - confidence 80% |
| R8: C $\Rightarrow$ Class 2 - confidence 80% |
| R9: A $\wedge$ C $\Rightarrow$ Class 3 - confidence 70% |

Table 5.1: Example of a rule-based model

| |
|---|
| R5: A $\Rightarrow$ Class 1 - confidence 85% |
| R6: A $\wedge$ B $\Rightarrow$ Class 2 - confidence 85% |
| R7: B $\Rightarrow$ Class 2 - confidence 80% |
| R8: C $\Rightarrow$ Class 2 - confidence 80% |
| R4: B $\wedge$ C $\Rightarrow$ Class 3 - confidence 90% |
| R9: A $\wedge$ C $\Rightarrow$ Class 3 - confidence 70% |

Table 5.2: Rules that apply to an object with features A, B, and C.

a rule-based model and the classification result when different classification strategies are used.

**Example 1.** Table 5.1 shows a hypothetical rule-based model that was generated from a training set using an association rule learning method. Each rule may have several measures attached, such as confidence in our example. Suppose we are given an object *O* to classify that has features A, B, and C. The subset of rules that apply to object *O* is shown in Table 5.2.

Associative classifiers use different strategies to classify a new instance based on the subset of rules that apply to it. In our example let us consider three different approaches for classifying object *O*:

- Approach 1: classification based on highest ranking rule. If we classify by the highest confidence ranked rule that applies we have to predict *Class 3* for object *O*, based on rule R4 with 90% confidence;
- Approach 2: classification based on average confidence. If we predict the class whose applicable rules have the highest average confidence, object *O* will be classified as *Class 1* (Average of 85% for Class 1; 81.6% for Class 2 and 80% for Class 3);
- Approach 3: classification based on number of applicable rules. If the new object is put into the class that has the largest number of applicable rules, object *O* will be classified as *Class 2*.

The preceding classification schemes are just simple examples to illustrate that there is not a unique way to combine the individual conclusions of a set of rules to create a final classification. The actual schemes used by existing systems are as follows. CBA [LHM98] classifies a new object with the class of the highest confidence-based ranked rule.

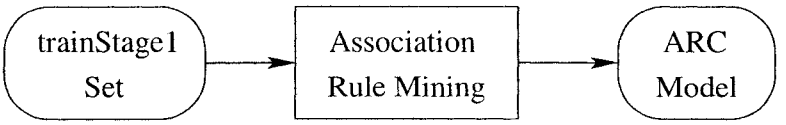


Figure 5.1: First stage of learning


Figure 5.2: Second stage of learning

In [LHP01], the authors use a weighted chi-square over the rules that apply and chooses the class with the highest score. In [AZ02], we base our prediction on a set of rules by using the average of the confidences. We also experimented with other measures such as cosine measure, Jaccard coefficient, etc. [RSZA05]. Some have also considered the size of the rule [CL04].

Bagging [Bre96] and boosting [Sch99] are two ensemble methods that achieve a better classification performance by combining the conclusion of multiple classifiers. These classifiers are generated by using the same learning method on different distributions of data. Bagging uses different replicas of the training set, while boosting uses the same training examples for all the classifiers but attaches different weights to them. The generated classifiers are combined by voting to obtain a final decision. In bagging, all classifiers have equal weight in the voting process. Boosting weighs each classifier's vote by its accuracy on the training set.

All these methods can be thought of as weighted voting schemes, where the weights for each rule, or class, are defined by specific scoring schemes. In this chapter we propose to automatically learn the scoring scheme for weighted voting after first learning the set of rules. Thus we propose a *two stage learning process*. The first stage is standard association rule mining. In this stage an association rule classification model (i.e., a set of classification rules) is generated. When applied to a given object, the first model produces as output a set of rules that apply to the object (possibly a subset selected from among all the rules that apply), along with each rule's conclusion and associated measures such as the rule's confidence. The second learning stage consists of a learning system whose input is a set of features derived from the first stage output.

Our method that learns to use the generated rules and selects the appropriate ones works as follows:

1. split the training set into two subsets: *trainStage1* and *trainStage2*; given that we have two learning stages in our technique we need to have enough examples for both stages; we split the initial training set into two disjoints subsets; this ensures the ability of our second stage to improve the performance of the overall model without overfitting by learning on the same set as the first stage;

2. generate classification rules from *trainStage1* using an association rule mining algorithm; we denote with *ARC Model* the discovered set of rules; Figure 5.1 shows this step of the algorithm;

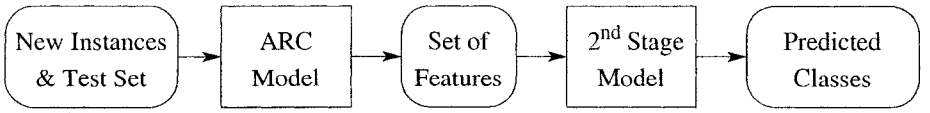


Figure 5.3: Evaluation and classification

3. for each instance in *trainStage2*, use *ARC Model* to generate a new set of features;

4. apply a new learner in this new feature space to learn how to use the rules in the prediction process; denote the model generated by the second learning algorithm as *2nd Stage Model*; Figure 5.2 shows this step of the algorithm;

5. classify the objects in the testing set using *ARC Model* and *2nd Stage Model* combined (this system is called 2SARC) (see Figure 5.3); when a new object has to be classified the *ARC Model* generates the features for the *2nd Stage Model*, which in turn makes the classification decision.

As one may see from Example 1, to score and make a decision for a new object is a difficult task. The second learning method in our technique learns a scoring scheme. The advantage of using this second step is twofold: it takes into account more information than using just one or few rules for prediction; treating the scoring scheme as a learning problem we create an automated process to learn the scheme from data (different scoring schemes are automatically generated for different applications and different datasets).

Given a classification problem that has $C$ classes to be learned, we use association rule mining in the first stage of our system to discover classification rules. Let us consider that our discovered ARC model consists of a set $R$ with N rules, ordered by their confidence and support (ordering is important when only a subset of rules is considered for classification). Each rule $R_i, i = 1..N$ has an associated support $\sigma_i$ and a confidence $C_i$. When a new instance $O$ is to be classified, a subset $R_O$ ($R_O \subset R$) of $K$ rules ($K \leq N$) applies $R_O = \bigcup R_i, i = 1..K$. A rule is applicable to the new instance if the antecedent of the rule matches the new object and its confidence is within the confidence margin [AZ02]. Naturally, a scoring scheme has to be used in order to make the classification decision. This scoring scheme has to be a function that represents the strength of the decision and it has to take into account all rules. The rules are characterized by support, confidence and their class. The scoring scheme has to produce a score for all the classes to be considered.

We have developed and tested two approaches for devising automatically learned scoring schemes. They differ in the type of features generated for the second stage. In our first approach, we use the rules directly, the feature space consisting of rules' characteristics. Each rule in our model applies or not to a new object and indicates its contribution with a certain confidence. Thus we learn a classification model from these characteristics. Given that the number of classification rules can be quite large for some applications (especially at low support thresholds), the rule-based feature space can become large. The large dimensionality of the feature space is a challenge for many learners. Thus, we introduce our second approach where instead of rule characteristics we use class-based aggregate measures as features. This approach reduces considerably the feature space since the number of features is a multiple of the number of classes in the application which is very small in general. The two approaches that we propose are as follows.

| R1 | | R2 | | R3 | | R4 | | R5 | | R6 | | R7 | | R8 | | R9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 95 | 0 | 90 | 0 | 90 | 0 | 90 | 1 | 85 | 1 | 85 | 1 | 80 | 1 | 80 | 1 | 70 | 1 |

Table 5.3: Sample rule features characterizing object $O$ when 2SARC-RF is used

| Class 1 | | Class 2 | | Class 3 | |
|---|---|---|---|---|---|
| avg conf | # rules | avg conf | # rules | avg conf | # rules |
| 85 | 1 | 81.6 | 3 | 80 | 2 |

Table 5.4: Sample class features characterizing object $O$ when 2SARC-CF is used

**2SARC-RF: Two Stage Classification with Rule-based Features.** In this approach the feature space is represented by the rules. Given our rule-based ARC model, we input into the second learning method the characteristics of each rule $R_i$ in $R$ with respect to the new object $O$. For each object $O$, a rule either applies or not. We introduce this information along with the rule's confidence into the model for the second stage.

For the rule model in Table 5.1 and an object $O$ with features $\{A, B, C\}$, the set of rule-based features that is generated is presented in Table 5.3. For each rule, the first value represents the rule's confidence and the second shows if the rule applied (value 1) or not (value 0) to object $O$. In its second stage our system learns how to use the rules given this set of features and the class for each object in *trainStage2*.

Although the architecture of 2SARC-RF may seem similar to stacking [Wol92] they differ as follows. In stacking level one is represented by one or more classification methods. Their classification results represent the input space for the second level classifier. In our approaches, the first level is a rule-based method that discovers classification rules. The input to the next level are features generated using the model built in the first level. The feature layer is what mostly distinguishes our technique from the stacking approach.

**2SARC-CF: Two Stage Classification with Class-based Features.** From the set of rules $R_O$ that apply to a new object $O$, M measures $m_j$, $j = 1..M$ can be computed over $R_O$ for each class $c_i, i = 1..C$: $\{m_1(c_1), ..., m_M(c_1), m_1(c_2), ...m_M(c_C)\}$. For instance, these measures could be the average confidence, the number of applicable rules and the maximum confidence. All these aggregations are done by class. At this stage each class would have several measures associated with it. All these measures are the input to the second learning method.

Considering Example 1, the features collected for object $O\{A, B, C\}$ are average confidence and number of rules per class as shown in Table 5.4. For each class a set of features is generated (e.g., Class1: average confidence is 85%, # rules supporting this class is 1). In its second stage our system learns how to use the rules given this set of features and the class for each object in *trainStage2* set.

The next section presents the evaluation of our proposed system with the two types of feature spaces used in our second stage.

## 5.3 Experiments

We evaluated our proposed system against other associative classifiers and rule-based classification methods on 20 UCI datasets [BM98]. In addition, we studied the performance of our system in the text classification context. This type of application is more challenging given that the feature space is very large and it requires multi-label classification (i.e., one or more class labels are attached to each object to be classified). UCI datasets are single-label classification problems. The next section describes the setup of our system followed by the experimental results.

### 5.3.1 Experimental Setup

The goal of 2SARC is to improve the classification stage for associative classifiers. Thus the output model of our first stage is a rule-based model generated with an association rule mining algorithm. Any association rule mining algorithm would produce the same classification rules given the same support and confidence thresholds. To automatically learn the scoring scheme we use another learner in the second stage. To determine the best architecture for our proposed system we experimented with three different learning algorithms in the second stage. Background on these classification systems are given in Section 2.5. The reasons behind our choices and the setup for our experiments are discussed below.

**Neural Network (NN):** Since we have to learn a numerical scoring scheme in the second stage of our system, the use of a neural network is a natural choice. Our neural network is a standard 3-layer feedforward network whose inputs are the set of features derived from the first stage output. The number of outputs equals the number of classes in the application, each output neuron corresponding to a class. The number of neurons in the hidden layer was the average number between the input and output neurons. A logistic function is used to compute the weights inside the network. The output of the function ranges from 0 to 1. When a single class has to be predicted for an object the output neuron (i.e., class) with the highest value is chosen as the winner. If multiple classes have to be predicted every class above a threshold is considered a winner. A drawback of neural networks is that they cannot handle well large numbers of inputs. For most applications association-based models generate a large number of rules, thus creating a very large feature space for 2SARC-RF. The 2SARC-RF technique used 10, 25, 50 or 100 rules per class. In our evaluation we use backpropagation learning for the neural network [Bor08b].

**k Nearest Neighbour (kNN):** Our second choice for the second stage learner is K Nearest Neighbour. Compared to Neural Networks, it is a fast learner and it can handle very well large feature spaces. We have the following measures to compute the inter-object distances: Manhattan distance, weighted Manhattan distance, Euclidean distance, weighted Euclidean distance, cosine measure and weighted cosine measure. To decide the class of a new object we use $k$ nearest neighbours with $k$ varying between 1 and 20. The 2SARC-RF technique used 2000 rules per class.

**Naïve Bayes (NB):** We chose to experiment with Naïve Bayes in the second stage of our system since it is a widely used learner which shows good performance for many applications. In our evaluation we used Borgelt's implementation [Bor08b].

| | 2SARC-CF | | | 2SARC-RF | | |
|---|---|---|---|---|---|---|
| dataset | kNN | NB | NN | kNN | NB | NN |
| anneal | 97.31 | 96.72 | **97.41** | **97.71** | 96.87 | 90.91 |
| australian | **87.67** | 86.80 | 87.35 | **87.27** | 86.43 | 80.74 |
| breast-w | **97.27** | 96.62 | 97.14 | **97.43** | 97.40 | 95.02 |
| cleve | **85.10** | 84.44 | 84.58 | **85.04** | 83.30 | 80.09 |
| crx | 87.24 | 86.45 | **87.29** | **86.60** | 86.19 | 80.54 |
| diabetes | **76.53** | 75.65 | **76.53** | **75.94** | 75.24 | 75.55 |
| german | 74.40 | 72.24 | **74.54** | 73.36 | **73.72** | 72.46 |
| glass | 71.19 | **71.28** | 70.81 | 69.03 | **69.25** | 68.64 |
| heart | **84.68** | 83.57 | 84.12 | **83.54** | 82.24 | 80.02 |
| hepatitis | 86.45 | 83.81 | **86.81** | **87.30** | 85.33 | 84.47 |
| horse | **84.68** | 84.56 | 84.50 | **84.18** | 82.23 | 81.22 |
| iris | **95.86** | 95.33 | 95.62 | **95.99** | 95.33 | 95.60 |
| labor | 92.19 | 89.40 | **93.57** | 88.67 | 87.93 | **90.02** |
| led7 | 72.73 | **73.37** | 72.57 | **73.94** | 73.76 | 73.64 |
| pima | **75.77** | 74.78 | 75.44 | **75.22** | 74.04 | 74.64 |
| tic-tac-toe | **99.59** | 97.56 | 99.57 | **100.00** | 98.55 | **100.00** |
| vehicle | **68.22** | 61.90 | 67.74 | **66.86** | 61.81 | 56.04 |
| waveform | 79.23 | 78.39 | **79.64** | **82.58** | 76.50 | 78.28 |
| wine | **96.75** | 95.30 | 96.74 | **97.07** | 94.83 | 95.80 |
| zoo | **93.11** | 88.76 | 91.73 | **97.87** | 94.34 | 95.93 |

Table 5.5: Accuracy on 20 UCI datasets for 2SARC-RF and 2SARC-CF with different learners in the second stage; results in bold represent the best accuracy for a dataset/classifier pair

### 5.3.2 Single Label Classification - UCI Datasets

We evaluated our system against other associative classifiers (CBA [LHM98], CMAR [LHP01]) on 20 UCI datasets [BM98]. In addition we compared our method with two rule-based classification methods (C4.5 rules [Qui93] and RIPPER [Coh95]), a boosting algorithm [Sch99] and a hybrid between rule-based methods and associative classifiers (CPAR) [YH03].

On each UCI dataset we performed C4.5's shuffle utility [Qui93] for shuffling the datasets. A 10-fold cross validation was performed on each dataset and the reported results are averages of the accuracies over the 10 folds. In addition, we used the same discretization method for continuous attributes as in [LHM98] to have a fair comparison with the other algorithms.

All classification methods should be evaluated on the same randomly generated folds to ensure a fair comparison of the methods. As the code for CPAR has the cross validation incorporated and it does not allow one to specify the folds we could not guarantee that CPAR was evaluated on the same folds as the other algorithms. CMAR code is not available. Thus the results presented in our table are the best results for CPAR [YH03] and CMAR [LHP01] as reported by their authors. All the other results were obtained in our study. For CBA we used the code provided by their authors, while for the other algorithms (C4.5, RIPPER, boosted RIPPER) we used their Weka [WF05] implementations (Weka version 3.4.8). The parameters for all the algorithms were set to their default values. Weka's default settings

| dataset | 2SARC-CF | 2SARC-RF | C4.5 | RIPPER | BooR | ARC | CBA | CMAR | CPAR |
|---|---|---|---|---|---|---|---|---|---|
| anneal | 97.31 | 97.71 | 89.87 | 94.66 | **99.33** | 97.11 | 97.91 | 97.30 | 98.40 |
| australian | **87.67** | 87.27 | 86.96 | 85.80 | 85.80 | 86.23 | 85.38 | 86.10 | 86.20 |
| breast-w | 97.27 | **97.43** | 94.71 | 95.28 | 95.85 | 96.42 | 96.28 | 96.40 | 96.00 |
| cleve | **85.10** | 85.04 | 80.52 | 80.19 | 83.51 | 82.16 | 82.83 | 82.20 | 81.50 |
| crx | **87.24** | 86.60 | 85.36 | 85.80 | 84.78 | 85.36 | 85.38 | 84.90 | 85.70 |
| diabetes | **76.53** | 75.94 | 74.21 | 74.34 | 73.95 | 74.22 | 74.45 | 75.80 | 75.10 |
| german | 74.40 | 73.36 | 71.60 | 71.60 | 71.90 | 73.20 | 73.50 | **74.90** | 73.40 |
| glass | 71.19 | 69.03 | 71.47 | 69.70 | 69.70 | 71.13 | 73.90 | 70.10 | **74.40** |
| heart | **84.68** | 83.54 | 80.74 | 81.85 | 83.33 | 80.74 | 81.87 | 82.20 | 82.60 |
| hepatitis | 86.45 | **87.30** | 79.25 | 76.04 | 80.08 | 81.13 | 81.82 | 80.50 | 79.40 |
| horse | 84.68 | 84.18 | **85.04** | 84.23 | 82.89 | 84.23 | 82.36 | 82.60 | 84.20 |
| iris | 95.86 | **95.99** | 94.00 | 94.00 | 94.67 | 95.33 | 94.67 | 94.00 | 94.70 |
| labor | **92.19** | 88.67 | 81.00 | 86.33 | 91.33 | 80.67 | 86.33 | 89.70 | 84.70 |
| led7 | 72.73 | 73.94 | **74.19** | 69.31 | 69.31 | 71.91 | 72.06 | 72.50 | 73.60 |
| pima | **75.77** | 75.22 | 73.70 | 73.19 | 73.84 | 74.87 | 72.90 | 75.10 | 73.80 |
| tic-tac-toe | 99.59 | **100.00** | 85.60 | 97.71 | 98.33 | 98.65 | 99.59 | 99.20 | 98.60 |
| vehicle | 68.22 | 66.86 | 67.04 | 64.31 | 68.67 | 65.02 | 68.92 | 68.80 | **69.50** |
| waveform | 79.23 | 82.58 | 75.08 | 75.04 | 78.92 | 78.28 | 79.68 | **83.20** | 80.90 |
| wine | 96.75 | **97.07** | 92.12 | 92.12 | 96.67 | 88.79 | 94.96 | 95.00 | 95.50 |
| zoo | 93.11 | **97.87** | 92.18 | 89.09 | 96.09 | 95.09 | 96.78 | 97.10 | 95.10 |
| average | **85.30** | 85.28 | 81.73 | 82.03 | 83.95 | 83.03 | 84.08 | 84.38 | 84.17 |

Table 5.6: Accuracy on 20 UCI datasets for several classification methods; results in bold represent the best accuracy on a dataset

follow the best parameter setup as proposed by their respective authors [Coh95, Qui93].

It is well-known that the support threshold plays a fundamental role in association rule discovery. Since associative classifiers are based on association rule mining they inherit the sensitivity to the support threshold. It is hard to know apriori the best support threshold and its value is usually set experimentally. In our evaluation, we ran our algorithms with support values of 1%, 5% and 10% for each dataset and we reported the best result. The minimum confidence was set to 50%. In our experiments we used a split ratio of 50/50 or 75/25 between *trainStage1* and *trainStage2* sets. To eliminate any bias caused by the split we ran our methods over 5 splits and averaged the results. To reduce the dimension of the feature space to a manageable size for the neural network, we experimented with using only 10, 20 or 30 best rules per class (for instance, if only the two best rules per class are used in Example 1 we have: R1, R3 for Class 1; R2 and R5 for Class 2; R8 and R9 for Class 3) to generate the feature space for the second stage.

The results for 2SARC-CF and 2SARC-RF with 3 different learners in the second stage are shown in Table 5.5. The results show that both 2SARC-CF and 2SARC-RF perform best when kNN is used in the second stage (12 wins and 17 wins respectively). This is the reason why we chose this combination when we compare our system with other classifiers.

Table 5.6 presents the accuracy of the following methods: C4.5, RIPPER, boosted RIPPER (booR), CBA, CMAR, CPAR, ARC and the results for the proposed techniques (2SARC-CF and 2SARC-RF). The standard deviation is not reported as our statistical analysis (presented next) uses non-parametric tests.

2SARC-CF obtains best overall performance for 7 datasets, followed by 2SARC-RF (6 datasets), CMAR, C4.5 and CPAR (2 datasets each) and boosting (1 datasets). Ripper, CBA and ARC are the algorithms that do not have any overall win.

On some datasets the differences in accuracy between the winner and the second best

| | wins | losses | ties |
|---|---|---|---|
| 2SARC-CF vs. ARC* | 19 | 1 | 0 |
| 2SARC-CF vs. 2SARC-RF | 11 | 9 | 0 |
| 2SARC-CF vs. c4.5* | 17 | 3 | 0 |
| 2SARC-CF vs. Ripper* | 20 | 0 | 0 |
| 2SARC-CF vs. BoostingR* | 17 | 0 | 0 |
| 2SARC-CF vs. CBA* | 14 | 5 | 1 |
| 2SARC-CF vs. CMAR* | 16 | 4 | 0 |
| 2SARC-CF vs. CPAR | 14 | 6 | 0 |

Table 5.7: Method 2SARC-CF compared to the rest of the algorithms on UCI datasets; (*) indicates statistically significant difference

| | wins | losses | ties |
|---|---|---|---|
| 2SARC-RF vs. ARC* | 18 | 2 | 0 |
| 2SARC-RF vs. 2SARC-CF | 9 | 11 | 0 |
| 2SARC-RF vs. c4.5* | 16 | 4 | 0 |
| 2SARC-RF vs. Ripper* | 18 | 2 | 0 |
| 2SARC-RF vs. BoostingR* | 16 | 4 | 0 |
| 2SARC-RF vs. CBA* | 16 | 4 | 0 |
| 2SARC-RF vs. CMAR* | 15 | 5 | 0 |
| 2SARC-RF vs. CPAR* | 15 | 5 | 0 |

Table 5.8: Method 2SARC-RF compared to the rest of the algorithms on UCI datasets; (*) indicates statistical significant difference

are quite small (e.g., german), while for other the improvement in the performance is large (e.g., hepatitis, heart, cleve).

2SARC-CF has the best overall performance, as it can be see from Table 5.6. Table 5.3.2 shows the count of wins, losses and ties for 2SARC-CF when compared to the rest of the methods, while the performance of 2SARC-RF in terms of wins, losses, ties is shown in Table 5.3.2. Table 5.3.2 shows that 2SARC-RF significantly outperforms all the other classification methods except 2SARC-CF. If we consider the win to loss ratio, the algorithm second in performance to 2SARC-CF is 2SARC-RF, followed by CPAR.

### Statistical Analysis

The results presented in Table 5.6 give some insight into the performance of the algorithms. However, those results do not provide enough support for drawing a strong conclusion in favour or against any of the studied methods. There is no overall dominance over the entire range of datasets.

To better understand the results of our techniques when compared to the other classification approaches we performed a statistical analysis of our results. In our experimental study we collected classification results for 9 classification methods on 20 datasets. In this type of experimental design a careful consideration has to be given to choosing the appropriate statistical tools. When a large number of comparisons is made (i.e., 180 in our design) the likelihood of finding significance by accident increases. The significance level has to be controlled so that it accounts for the multiple comparisons. This issue is known in statistics

| dataset | 2SARC-CF | 2SARC-RF | C4.5 | Ripper | BooR | ARC | CBA | CMAR | CPAR |
|---|---|---|---|---|---|---|---|---|---|
| anneal | 5 | 4 | 9 | 8 | 1 | 7 | 3 | 6 | 2 |
| australian | 1 | 2 | 3 | 7.5 | 7.5 | 4 | 9 | 6 | 5 |
| breast-w | 2 | 1 | 9 | 8 | 7 | 3 | 5 | 4 | 6 |
| cleve | 1 | 2 | 8 | 9 | 3 | 6 | 4 | 5 | 7 |
| crx | 1 | 2 | 6.5 | 3 | 9 | 6.5 | 5 | 8 | 4 |
| diabetes | 1 | 2 | 8 | 6 | 9 | 7 | 5 | 3 | 4 |
| german | 2 | 5 | 8.5 | 8.5 | 7 | 6 | 3 | 1 | 4 |
| glass | 4 | 9 | 3 | 7.5 | 7.5 | 5 | 2 | 6 | 1 |
| heart | 1 | 2 | 8.5 | 7 | 3 | 8.5 | 6 | 5 | 4 |
| hepatitis | 2 | 1 | 8 | 9 | 6 | 4 | 3 | 5 | 7 |
| horse | 2 | 6 | 1 | 3.5 | 7 | 3.5 | 9 | 8 | 5 |
| iris | 2 | 1 | 8 | 8 | 5.5 | 3 | 5.5 | 8 | 4 |
| labor | 1 | 4 | 8 | 5.5 | 2 | 9 | 5.5 | 3 | 7 |
| led7 | 4 | 2 | 1 | 8.5 | 8.5 | 7 | 6 | 5 | 3 |
| pima | 1 | 2 | 7 | 8 | 5 | 4 | 9 | 3 | 6 |
| tic-tac-toe | 2.5 | 1 | 9 | 8 | 7 | 5 | 2.5 | 4 | 6 |
| vehicle | 5 | 7 | 6 | 9 | 4 | 8 | 2 | 3 | 1 |
| waveform | 5 | 2 | 8 | 9 | 6 | 7 | 4 | 1 | 3 |
| wine | 2 | 1 | 7.5 | 7.5 | 3 | 9 | 6 | 5 | 4 |
| zoo | 7 | 1 | 8 | 9 | 4 | 6 | 3 | 2 | 5 |
| average rank | 2.575 | 2.85 | 6.75 | 7.475 | 5.6 | 5.925 | 4.875 | 4.55 | 4.4 |

Table 5.9: Ranking of the systems based on their accuracies

as controlling the family-wise error [Dem06].

Demsar discusses in [Dem06] the issue of multiple hypothesis testing and recommends the use of several statistical procedures for this problem. Following Demsar's recommendation, we first tested if there is any significant difference among the studied classification methods. Demsar recommends the use of Friedman test to compare several classifiers on multiple datasets.

Let us assume that we have $k$ algorithms to compare on N datasets. The Friedman test can be applied as follows:

- find $r_i^j$ - the rank of the algorithm $j$ on the $i^{th}$ dataset; (the computed ranks for UCI results are shown in Table 5.9, when ties occur the average rank is considered)
- compute the average rank R of algorithm $j$:

$$R_j = \frac{1}{N}\sum_i r_i^j \tag{5.1}$$

- the null hypothesis states that all algorithms have the same average rank;
- compute the Friedman statistic:

$$\chi_F^2 = \frac{12N}{k(k+1)}(\sum_j R_j^2 - \frac{k(k+1)^2}{4}) \tag{5.2}$$

- if $\chi_F^2$ exceeds the critical value (available in statistics tables) we can reject the null hypothesis otherwise we accept it;

- if the null hypothesis has been rejected a post-hoc test must be applied to determine the nature of the difference.

By applying Friedman test [Dem06] we concluded that there is a significant difference among the methods. Since the null hypothesis is rejected we have to proceed with further analysis to better understand the behaviour of the classification algorithms. We are interested in the performance of our proposed technique. Thus, we perform a series of Wilcoxon signed ranked tests between our best method (2SARC-CF) and the other classification methods. The Friedman and Wilcoxon tests are non-parametric tests, they do not make any assumptions about the distributions of the values.

For single-label classification results the Friedman test finds significant difference among the studied classification methods. The only strong conclusion that we can draw when paired Wilcoxon signed ranked tests are performed is that 2SARC-CF performs significantly better than all other classifiers except 2SARC-RF and CPAR. This is how the statistical evaluation was performed for the results presented in Tables and

**Further analysis**

In this chapter we have proposed a new scoring scheme for associative classifiers. We have argued that the new system performs better than existing associative classifiers due to the automatically learned scoring scheme. To further support this claim we performed another experiment where we try to tie the performance increase to the automatically learned scoring scheme, isolating it from other changes introduced by our new system. For that, we compare the performance of 2SARC-CF and 2SARC-RF against ARC (described in Section 2.3.3). There are two main differences between 2SARC and ARC. First, they use different scoring schemes, which is exactly what we are interested in comparing. Second, the classification rules are generated from different subsets of the training dataset. When running 2SARC and ARC over the entire training set, ARC discovers rules from the entire set (*trainStage1* + *trainStage2* sets), while 2SARC uses rules generated from *trainStage1* only. We are interested in understanding if 2SARC's performance improvement over ARC is due to the automatically learned scoring scheme or it is caused by the different training sets used for generating rules in 2SARC and ARC. To eliminate the difference in generated rules between 2SARC and ARC, we measured the performance of ARC using rules generated from *trainStage1* only. The experiment confirms that 2SARC's increase in performance comes from learning the scoring scheme. Over all UCI datasets 2SARC performed better than ARC when ARC generated rules from either *trainStage1* or the entire training set. On several datasets the difference in performance was significantly large, which shows that learning the scoring function from the data is beneficial to the classification process.

Our system has two stages and uses for training two subsets (*trainStage1* and *trainStage2*) of the original training set. Considering that we split randomly the original training set we need to understand how the split affects the interestingness of a rule. Our assumption is that the changes in the rule measures are small between different splits. To test this hypothesis, we performed the following experiment: we generated 100 random splits of the original training set into *trainStage1* and *trainStage2* sets with a fixed split ratio; we selected at random a rule generated by mining the original training set and we recorded its support and confidence when generated from *trainStage1* set of each of the 100 splits. A summary of the results is presented in Table 5.10. The results are presented as average confidence and standard deviation over 100 runs. Smaller datasets show bigger changes

| dataset size | dataset | avg conf | std |
|---|---|---|---|
| small(<200 transactions) | labor | 86.1 | ±1.52 |
| | hepatitis | 90.6 | ±2.95 |
| medium(200-1000 transactions) | cleve | 93.8 | ±0.85 |
| | pima | 94.5 | ±0.85 |
| large(>1000 transactions) | led7 | 65.1 | ±1.26 |
| | waveform | 69.4 | ±0.59 |

Table 5.10: Average confidence and standard deviation recorded over 100 split runs

in the rule confidence as indicated by the the standard deviation. This effect was expected since it is more difficult to preserve the initial distribution of data during the split with less examples in the original training set. In the case of larger datasets, the variation in the rule confidence is small, confirming that the the random split has little effect on the confidence of the generated rules. This result indicates that the randomness of the split has little effect on the quality of the generated rules.

### 5.3.3 Discussion

Rule-based classification systems classify a new instance based on a set of rules that apply to the new object. In previous works, the scoring schemes under which the system takes a classification decision are predefined. In this work we proposed a two-stage classification method. Our system (2SARC) learns automatically the scoring scheme in the second stage. In addition, we investigate two techniques. 2SARC-CF learns the scoring scheme from class features while 2SARC-RF learns it from rule features.

2SARC-CF performs best on most UCI datasets or it ranks very close to the best as Table 5.6 shows. There are only 2 datasets where the performance is much lower than the best (*waveform* and *zoo*). *zoo* is a very small datasets with 101 examples. Small datasets may hinder the performance of our system given that it needs enough examples to train the models for the two stages. *Waveform* is a dataset where all the initial attributes are numerical. Association rules are more suitable for nominal attributes, thus for this particular dataset the mining has to rely on the discretization process. The influence of the discretization process on the performance of association-rule based systems needs further study.

Our method has two stages, each stage employing a classification algorithm. It may appear at first that the training time is higher than for each algorithm applied alone or for other algorithms. This is not necessarily the case, as each stage uses only a partition of the data (according to the split ratio). In addition the k-nearest neighbour which we choose as the second learner has virtually no training time.

### 5.3.4 Multi Label Classification - Text Categorization

Associative classifiers are more suitable for categorical data rather than numerical given that these systems are based on association rules techniques. Association rules have been developed to discover associations/relationships among items/objects in categorical data. Therefore, text classification is a good application to study the performance of our method.

Most of the research in text categorization comes from the machine learning and information retrieval communities. Rocchio's algorithm [Hul94] is the classical method in information retrieval, being used in routing and filtering documents. Researchers tackled

| | 2SARC-CF | | 2SARC-RF | |
|---|---|---|---|---|
| category | NN | kNN | NN | kNN |
| acq | **93.7** | 84.3 | 81.5 | **93.8** |
| corn | **88.4** | 85.1 | **83.3** | 75.6 |
| crude | 83.0 | **87.6** | 83.5 | **88.6** |
| earn | **95.4** | 93.3 | 87.2 | **96.7** |
| grain | **91.1** | 90.4 | 87.9 | **91.9** |
| interest | 72.8 | **77.0** | 73.9 | **82.0** |
| moneyfx | **78.7** | 72.0 | 73.8 | **83.3** |
| ship | **83.1** | 75.6 | 77.3 | **81.0** |
| trade | **84.9** | **84.9** | 86.7 | **89.7** |
| wheat | **88.7** | 88.2 | **85.3** | **85.3** |
| weighted avg | **90.61** | 87.09 | 82.34 | **92.08** |
| macro avg | **85.98** | 83.84 | 82.04 | **86.79** |

Table 5.11: Accuracy on Reuters dataset for 2SARC-CF and 2SARC-RF with kNN and NN in the second stage

| category | # rec | Bayes | Rocchio | C4.5 | k-NN | SVM | ARC | 2SARC-CF | 2SARC-RF |
|---|---|---|---|---|---|---|---|---|---|
| acq | 719 | 91.5 | 92.1 | 85.3 | 92.0 | **95.2** | 89.9 | 93.7 | 93.8 |
| corn | 56 | 47.3 | 62.2 | 87.7 | 77.9 | 85.2 | 82.3 | **88.4** | 75.6 |
| crude | 189 | 81.0 | 81.5 | 75.5 | 85.7 | **88.7** | 77.0 | 83.0 | 88.6 |
| earn | 1087 | 95.9 | 96.1 | 96.1 | 97.3 | **98.4** | 89.2 | 95.4 | 96.7 |
| grain | 149 | 72.5 | 79.5 | 89.1 | 82.2 | 91.8 | 72.1 | 91.1 | **91.9** |
| interest | 131 | 58.0 | 72.5 | 49.1 | 74.0 | 75.4 | 70.1 | 72.8 | **82.0** |
| money-fx | 179 | 62.9 | 67.6 | 69.4 | 78.2 | 75.4 | 72.4 | 78.7 | **83.3** |
| ship | 89 | 78.7 | 83.1 | 80.9 | 79.2 | **86.6** | 73.2 | 83.1 | 81.0 |
| trade | 118 | 50.0 | 77.4 | 59.2 | 77.4 | 77.3 | 69.7 | 84.9 | **89.7** |
| wheat | 71 | 60.6 | 79.4 | 85.5 | 76.6 | 85.7 | 86.5 | **88.7** | 85.3 |
| weighted-avg | | 84.25 | 87.94 | 85.14 | 89.68 | **92.15** | 84.12 | 90.61 | 92.08 |
| macro-avg | | 65.21 | 79.14 | 77.78 | 82.05 | 86.01 | 78.24 | 85.98 | **86.79** |

Table 5.12: Precision/Recall-breakeven point on ten most populated Reuters categories for most known classifiers and our new methods

the text categorization problem in many ways. Classifiers based on probabilistic models have been proposed starting with the first presented in literature by Maron [Mar61] and continuing with Naïve-Bayes [Lew98] that proved to perform well. C4.5 is a well-known package whose core is making use of decision trees to build automatic classifiers [Qui93]. K-nearest neighbor (k-NN) is another technique used successfully in text categorization [Yan99]. In the last decade neural networks and support vector machines (SVM) were used in text categorization and they proved to be powerful tools [Joa98].

We used the Reuters-21578 text collection [Reu08] as benchmark. There are several splits of the Reuters collection; we chose to use the *ModApte* version. This split leads to a corpus of 12,202 documents consisting of 9,603 training documents and 3,299 testing documents, and is the most used split in the literature. We tested our classifiers on the ten categories with the largest number of documents assigned to them in the training set. On these documents we performed stopword elimination but no stemming.

Several measurements have been used in previous studies for text classification evaluation. In our evaluation we use the breakeven point (BEP) and F1 measure discussed in detail in Section 2.6.2. When dealing with multiple classes there are two possible ways of averaging these measures, namely, macro-average and micro-average. In the macro-averaging, one contingency table per class is used, the performance measures are computed on each of them and then averaged. In micro-averaging only one contingency table is used for all classes, a total of all the classes is computed for each cell and the performance measures are obtained therein. The macro-average weights equally all the classes, regardless of how many documents belong to a class. The micro-average weights equally all the documents, thus favoring the performance on common classes.

In our experiments, we ran our algorithms with support values of 10%, 15% and 20% and we reported the best result. The minimum confidence was set to 50% and the confidence margin was set at 10%. Our proposed technique used a split ratio of 80/20 between trainStage 1 and trainStage2 sets. Details on the neural network and k-nearest neighbour setup are given in Section 5.3.1.

In the first experiment, we evaluated the performance of our approaches with neural network and k-nearest neighbour classifiers in the second stage. The results are presented in Table 5.11 with the best performance for each algorithm presented in bold. In the case of 2SARC-CF the use of neural network in the second stage is most beneficial, while for 2SARC-RF k-nearest neighbour is the best choice. Thus for the rest of our evaluation we use these combinations.

Table 5.12 shows the performance for several well-known text categorizers and our algorithms. The evaluation is done using precision/recall break-even point. Results are presented by category for the ten most populous categories in Reuters collection. Except for our algorithms, the results are presented as reported in [Joa98, TWL02]. Given that the micro-average value for the other algorithms was computed on a different set of classes we used a weighted average scheme to approximate the micro-average. The formula for weighted average is:

$$WA = \frac{\sum_{i=1}^{N} \frac{test_i}{\sum_{j=1}^{N} test_j} \times BEP_i}{N} \tag{5.3}$$

where N stands for the number of classes, $test_i$ represents the number of test examples in class $i$ and $BEP_i$ is the breakeven point for class $i$.

Both SVM and 2SARC-RF perform best for 4 out of 10 classes in the Reuters collection. 2SARC-CF wins in 2 classes, while the other algorithms don't have any overall wins. SVM ranks first based on weighted-average and 2SARC-RF ranks first based on the macro-average.

Table 5.13 shows the count of wins, losses and ties over the 10 classes in Reuters for 2SARC-CF when compared to the other algorithms. 2SARC-CF loses on six categories to SVM and on seven categories to 2SARC-RF. When compared to the rest of the algorithms 2SARC-CF outperforms each one of them, winning in at least 7 out of the 10 classes.

Table 5.14 shows the count of wins, losses and ties over the 10 classes for 2SARC-RF when compared to the other algorithms. 2SARC-RF loses on six categories to SVM. When compared to the rest of the algorithms 2SARC-RF outperforms each one of them, winning in at least 7 out of the 10 classes.

| | wins | losses | ties |
|---|---|---|---|
| 2SARC-CF vs. Bayes* | 9 | 1 | 0 |
| 2SARC-CF vs. Rocchio* | 8 | 1 | 1 |
| 2SARC-CF vs. C4.5* | 9 | 1 | 0 |
| 2SARC-CF vs. kNN | 7 | 3 | 0 |
| 2SARC-CF vs. SVM | 4 | 6 | 0 |
| 2SARC-CF vs. ARC* | 10 | 0 | 0 |
| 2SARC-CF vs. 2SARC-RF | 3 | 7 | 0 |

Table 5.13: Method 2SARC-CF compared to the rest of the algorithms on Reuters collection; (*) indicates statistical significant difference

| | wins | losses | ties |
|---|---|---|---|
| 2SARC-RF vs. Bayes* | 10 | 0 | 0 |
| 2SARC-RF vs. Rocchio* | 9 | 1 | 0 |
| 2SARC-CF vs. C4.5* | 8 | 2 | 0 |
| 2SARC-RF vs. kNN* | 8 | 2 | 0 |
| 2SARC-RF vs. SVM | 4 | 6 | 0 |
| 2SARC-RF vs. ARC* | 8 | 2 | 0 |
| 2SARC-RF vs. 2SARC-CF | 7 | 3 | 0 |

Table 5.14: Method 2SARC-RF compared to the rest of the algorithms on Reuters collection; (*) indicates statistical significant difference

#### Statistical Analysis

We performed the same statistical analysis as discussed in Section 5.3.2 to the text classification results. Friedman's test indicates that the methods evaluated are not equal. When the Wilcoxon tests are applied to the results of text classification for pairwise comparisons between 2SARC and the rest of the algorithms, we can conclude the following: both 2SARC-CF and 2SARC-RF are significantly better than Bayes, Rocchio, C4.5, ARC at a significance level of 0.05. In addition 2SARC-RF is significantly better than kNN. When compared with SVM, the test can not reject the null hypothesis, thus it can be stated that 2SARC-RF and SVM perform statistically similar on the Reuters dataset.

## 5.4 Summary

Rule-based classifiers use predefined weighted voting schemes to combine the class predictions of the applicable rules. By contrast, the methods described in this chapter automatically learn the scoring scheme. We achieve this by developing a two-stage system, with a layer of feature definitions interposed between the output of the first learning model and the input of the second. Our two stage classification system (2SARC) shows a good performance both for UCI datasets and text classification, under rigorous statistical analysis.

# Chapter 6

# Associative Classifiers Using Association Rules from Closed and Maximal Itemsets

An attractive characteristic that associative classifiers possess is their readability. However, the number of classification rules discovered is quite large. In addition, these rules contain redundant information since classification rules are obtained from mined frequent itemsets and the latter are known to be repetitive. In this chapter we investigate the performance of associative classifiers when the classification rules are generated from frequent, closed and maximal itemsets. Closed and maximal itemsets are concise representations of frequent itemsets. We show that maximal itemsets substantially reduce the number of classification rules without jeopardizing the accuracy of the classifier. Our extensive analysis demonstrates that the performance remains stable and even improves in some cases. Our analysis using cost curves also provides recommendations on when it is appropriate to remove redundancy in frequent itemsets.

## 6.1 Introduction

Typically, associative classifiers generate classification rules from frequent patterns (i.e., all patterns that are seen frequently in the training data). Closed [PBTL99] and maximal [Bay98] patterns are compressed representations of all the frequent patterns. They have been proposed to substantially reduce the number of frequent patterns. This reduction is achieved by eliminating redundancy present in the frequent patterns set. Closed patterns are a lossless form of compression, as the frequent patterns and their respective supports can be reproduced from this representation. On the other hand, maximal patterns represent a lossy compression since the support measures of the frequent patterns have to be recomputed.

Several research studies [PCT+03] demonstrate the usefulness of closed and maximal itemsets in different applications. In the case of classification, the use of these patterns has not been thoroughly explored. Closed and maximal frequent patterns reduce the number of association rules, but it is not clear that they reduce the number of classification rules as well. We hypothesize that they do and probably even improve the performance of the classification when these types of classification rules are used.

In this chapter we investigate the performance of associative classifiers when the classification rules are generated from closed and maximal itemsets. Through our analysis we

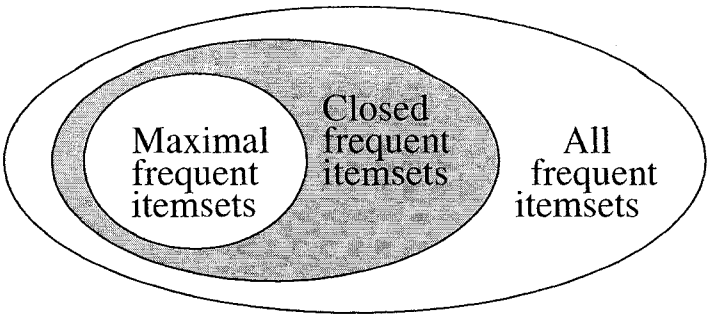


Figure 6.1: Frequent, closed and maximal itemsets

answer the following critical questions:

- How substantial is the reduction in the number of classification rules? It is known that closed and maximal representations reduce the number of patterns, but how does this translate to classification rules?
- What is the effect of rules extracted from closed and maximal itemsets on the classification performance? Are closed and maximal itemsets a good substitute for frequent itemsets in associative classifiers?

In the framework proposed in this chapter, we integrate several associative classifiers with classification rules generated from frequent, closed and maximal patterns. Our hypothesis is that the use of closed and maximal is advantageous to associative classifiers. The benefit is twofold: first, the set of classification rules generated from closed and maximal itemsets is smaller; second, the performance level of the classifier stays the same or it improves. We test our hypothesis with an extensive experimental study and we show that this hypothesis holds over a large range of applications.

## 6.2 Prerequisites

Formally, frequent pattern mining is defined as follows. Let $\mathcal{I} = \{i_1, i_2, ...i_m\}$ be a set of items. Let $\mathcal{D}$ be a set of transactions, where each transaction $T$ is a set of items such that $T \subseteq \mathcal{I}$. A transaction $T$ is said to contain $X$, a set of items in $\mathcal{I}$, if $X \subseteq T$. We define in the following the types of patterns that we study in this chapter.

**Definition 1. Frequent itemset:** An itemset $f \subseteq I$ is said to be frequent if its support $s$ (i.e., the percentage of transaction in $\mathcal{D}$ that contain $f$) is greater than or equal to a given minimum support threshold (denoted as $minsupp$ throughout the chapter).

**Definition 2. Frequent closed itemset:** A frequent itemset $c \subseteq I$ is said to be frequent closed if and only if there is no frequent itemset $c'$ such that $c \subseteq c'$ and the support of $c$ equals the support of $c'$.

**Definition 3. Maximal frequent itemset:** A frequent itemset $m \subseteq I$ is said to be maximal frequent if there is no other frequent itemset that is a superset of $m$.

Note that the set of maximal frequent patterns is included in the set of frequent closed patterns which is in turn included in the set of frequent patterns (Figure 6.1). Figure 6.2 presents a token example to illustrate the concepts with a transactional dataset of 5 transactions. The pattern lattice represents all possible itemsets. The pattern mining is done with an

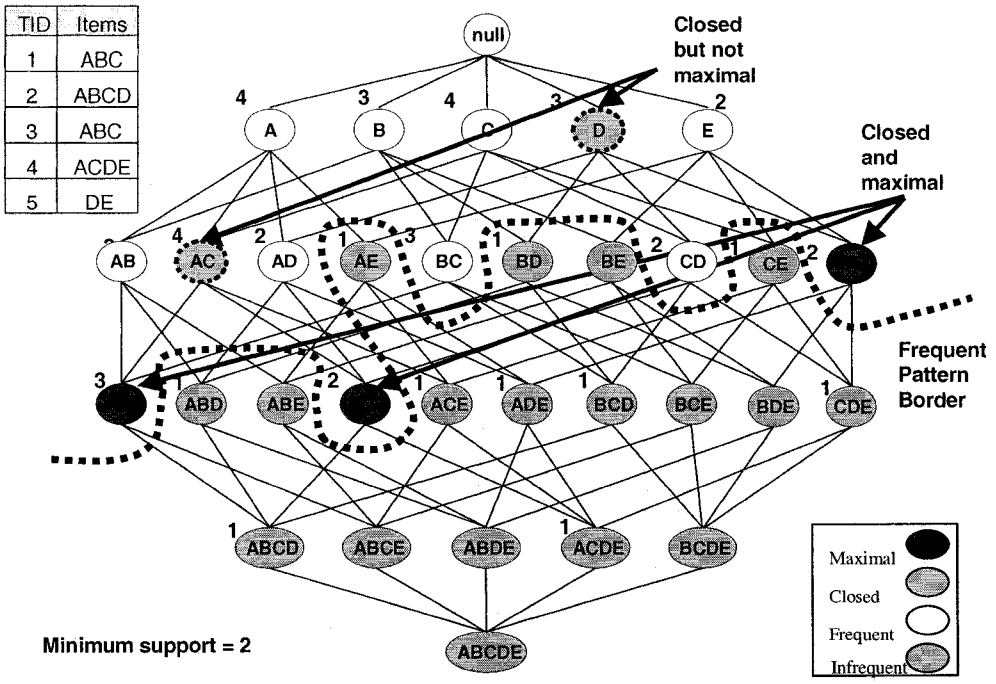


Figure 6.2: Frequent, closed and maximal patterns in the itemset lattice

absolute support threshold of 2. The number next to each itemset in Figure 6.1 represents its support. The border, shown as a dotted line, separates frequent patterns from non-frequent patterns (the frequent patterns are above the border). The set of frequent closed itemsets is: C={D, AC, DE, ABC, ACD} and the set of maximal frequent itemsets is: M={DE, ABC, ACD}. D is a frequent closed itemset because none of its supersets has the same support, although they are frequent. CD is neither frequent closed nor maximal frequent because its superset ACD is frequent and it has the same support. In the following we simply refer to these three types of itemsets as frequent, closed and maximal itemsets.

Associative classifiers have the advantage over other rule-based classification systems that they guarantee to find all interesting rules (in the support-confidence framework). However, this property also guarantees that the number of classification rules will be quite large. In this chapter, we focus our efforts on lowering the number of rules by mining closed and maximal itemsets. In addition to reducing the size of the system (i.e., the number of classification rules), redundant information is also eliminated. The next section gives a detailed description of our framework.

## 6.3 Integrating Associative Classifiers with Closed and Maximal Patterns

Our hypothesis is that the use of closed and maximal patterns is beneficial to associative classifier. The benefit is twofold: first, the set of classification rules generated from closed and maximal itemsets is smaller; second, the performance level of the classifier stays the same or it improves. In our framework we integrate several associative classifiers with

classification rules generated from frequent, closed and maximal patterns to investigate our hypothesis.

The classification stage is what most distinguishes one associative classifier from another. That is why we integrate in our framework several classification schemes. We investigate these systems when classification rules are generated from frequent, closed and maximal itemsets. The goal of our study is to find out the effect that closed and maximal patterns produce when integrated with associative classifiers.

The modules of our framework are as follows:

- *Discover frequent, closed and maximal itemsets.* Let F be the set of all frequent itemsets, F=$\{\bigcup$ f such that f $\subseteq$ I, and supp(f) $\geq$ *minsupp*$\}$. Let C be the set of all closed itemsets, C=$\{\bigcup$ c such that c $\subseteq$ I, it is closed and supp(c) $\geq$ *minsupp*$\}$. Let M be the set of all maximal itemsets, M=$\{\bigcup$ m such that m $\subseteq$ I, it is maximal and supp(c) $\geq$ *minsupp*$\}$.

- *Generate classification rules from the mined itemsets.* From one set of patterns (F, C or M) find all rules such that the consequent of the rule is a class label. Let R be the set of all classification rules, R=$\{\bigcup$ r such that r is of the form $X \rightarrow Y$, where Y is a class label and conf(r) $\geq$ *minconf*$\}$. Let us consider that $R_f$, $R_c$ and $R_m$ are the rule sets generated from frequent, closed and maximal itemsets. Order classification rules in $R_f$, $R_c$ and $R_m$ by confidence and support.

- *Classify a new object using the set of classification rules ($R_f$, $R_c$ or $R_m$).* The classification decision is made according to a scoring scheme. The scoring schemes that we investigate are as follows:

  1. Classify a new object according to the highest ranked rule that applies. This scheme is used in CBA [LHM98]. We denote this method as ***FR*** (first rule)
  2. Classify a new object based on average of the confidences. Let us assume that $k$ rules apply to a new object. Let $S$ be the set of $k$ rules. Divide $S$ in subsets by class label: $S_1, S_2 ... S_n$. For each subset $S_i$ sum the confidences of rules and divide by the number of rules in $S_i$, this is the score that is associated to class $i$. The object is classified in the class with the highest score [AZ02]. We denote this method as ***AvR*** (average the confidences of the rules that apply).
  3. Classify a new object using a two stage classification approach (see Chapter 5).
     - for each instance in the training set, use $R_f$, $R_c$ or $R_m$ to collect a set of features (class-features or rule-features as defined in Chapter 5);
     - apply a learning method in this new feature space to learn how to use the rules in the prediction process; denote the model generated by the second learning algorithm as *2SARC*;
     - classify the objects in the testing set using $R_f$, $R_c$ or $R_m$ and *2SARC* combined;

     We denote these systems as follows: ***2SARC-CF*** is the system that uses class features in the two stage classification approach; ***2SARC-RF*** is the system that uses rule features in the two stage classification approach.

| | supp=1% | | |
|---|---|---|---|
| **dataset** | **Frequent** (# rules) | **Closed** (% saved) | **Maximal** (% saved) |
| anneal | 10500.10 | 27.92 | 29.61 |
| australian | 19457.20 | 15.30 | 23.83 |
| breast-w | 3153.80 | 9.14 | 41.66 |
| cleve | 7898.80 | 11.11 | 26.80 |
| crx | 24353.30 | 16.84 | 22.31 |
| diabetes | 635.60 | 13.09 | 51.64 |
| german | 31688.40 | 14.71 | 24.38 |
| glass | 958.60 | 44.12 | 48.26 |
| heart | 2242.10 | 7.33 | 32.74 |
| hepatitis | 25074.90 | 14.96 | 18.51 |
| horse | 31909.10 | 20.66 | 27.52 |
| iris | 92.50 | 57.19 | 87.03 |
| labor | 4790.90 | 34.25 | 37.36 |
| led7 | 258.10 | 0.46 | 38.09 |
| pima | 577.20 | 13.60 | 53.59 |
| tic-tac-toe | 6362.80 | 5.04 | 37.07 |
| vehicle | 48465.50 | 16.26 | 17.71 |
| waveform | 34886.30 | 0.01 | 27.67 |
| wine | 17860.30 | 20.77 | 23.81 |
| zoo | 25084.20 | 17.67 | 17.67 |
| **Average** | 14812.49 | 18.02 | 34.36 |

Table 6.1: Number of classification rules generated from frequent itemsets at 1% support and the percentage of rules dropped when closed and maximal itemsets are used

## 6.4 Experimental Study

To test our hypothesis and to study our framework we performed an extensive experimental study in which we evaluated all four classification systems in our framework on several datasets. The details of the datasets and the evaluation techniques used are described in the following sections.

### 6.4.1 Datasets and Experimental Setup

We evaluated our framework on 20 UCI datasets [BM98]. In addition, we studied the performance of associative classifiers in the classification of several challenging microarray datasets. In our evaluation we used the following experimental setup. We used an apriori-like algorithm to mine frequent, closed and maximal itemsets [Bor08a]. We generated classification rules from these patterns and we integrated them with the associative classifiers discussed in Section 6.3. A k-nearest neighbour algorithm is used in the second stage of the 2SARC system. In Chapter 5 we evaluated 2SARC using neural networks, Naïve Bayes and k-nearest neighbour algorithms as second stage learners and we concluded that the k-nearest neighbour algorithm is the best choice.

For UCI datasets we set the support threshold to 1%, 5% and 10%. The confidence

| | supp=5% | | |
|---|---|---|---|
| **dataset** | **Frequent** (# rules) | **Closed** (% saved) | **Maximal** (% saved) |
| anneal | 2783.20 | 33.28 | 38.88 |
| australian | 4765.20 | 13.76 | 34.49 |
| breast-w | 894.90 | 2.56 | 49.46 |
| cleve | 2730.90 | 5.31 | 38.18 |
| crx | 6180.80 | 19.07 | 31.89 |
| diabetes | 233.90 | 9.79 | 66.61 |
| german | 5328.60 | 8.54 | 37.01 |
| glass | 321.00 | 48.69 | 58.60 |
| heart | 1074.20 | 3.84 | 47.77 |
| hepatitis | 10493.30 | 17.91 | 26.19 |
| horse | 2706.50 | 19.38 | 52.32 |
| iris | 65.50 | 54.81 | 88.55 |
| labor | 684.90 | 49.06 | 65.02 |
| led7 | 237.60 | 0.00 | 39.69 |
| pima | 232.70 | 10.10 | 67.98 |
| tic-tac-toe | 411.10 | 0.02 | 35.13 |
| vehicle | 9854.00 | 20.51 | 25.24 |
| waveform | 610.20 | 0.00 | 37.30 |
| wine | 5586.50 | 27.39 | 34.16 |
| zoo | 10963.30 | 22.38 | 22.38 |
| **Average** | 3307.92 | 18.32 | 44.84 |

Table 6.2: Number of classification rules generated from frequent itemsets at 5% support and the percentage of rules dropped when closed and maximal itemsets are used

threshold was set to 50%. On each UCI dataset we performed C4.5's shuffle utility [Qui93] for shuffling the datasets. A 10-fold cross validation was performed on each dataset and the reported results are averages over the 10 folds. In addition, we used the same discretization method for continuous attributes as in [LHM98]. The support threshold was harder to set for the microarray datasets. Across the set of datasets the support threshold ranged from 10% for *Prostate Cancer* dataset to 75% for *Lung Cancer* dataset. The confidence threshold was set to 50%. For each dataset we report the best result obtained under this parameter setting. We evaluated the performance of the classifiers based on accuracy and cost curves [DH06].

### 6.4.2 Results

This section presents the results that we obtained in our study. Our hypothesis has two components: first, we anticipate that the number of classification rules is significantly reduced when rules are obtained from closed and maximal itemsets; second, we expect to keep the same level or improve the performance of associative classifiers when the classification rules generated from closed and maximal patterns are used. The findings of our empirical studies clearly support our first hypothesis for the case of maximal frequent itemsets and in most cases of closed frequent itemsets. Indeed, in some cases the closed itemsets generated

| | supp=10% | | |
|---|---|---|---|
| **dataset** | **Frequent** (# rules) | **Closed** (% saved) | **Maximal** (% saved) |
| anneal | 1259.10 | 33.40 | 43.47 |
| australian | 1942.90 | 13.10 | 42.92 |
| breast-w | 484.80 | 1.98 | 51.20 |
| cleve | 1161.30 | 3.01 | 48.28 |
| crx | 2597.10 | 22.53 | 40.04 |
| diabetes | 94.90 | 6.74 | 67.44 |
| german | 1746.00 | 5.54 | 47.15 |
| glass | 199.20 | 50.10 | 62.00 |
| heart | 547.10 | 2.07 | 52.90 |
| hepatitis | 5208.40 | 16.69 | 29.81 |
| horse | 482.90 | 11.14 | 63.88 |
| iris | 50.20 | 53.19 | 91.04 |
| labor | 137.30 | 42.32 | 72.10 |
| led7 | NA | NA | NA |
| pima | 94.90 | 7.80 | 68.81 |
| tic-tac-toe | 104.40 | 0.00 | 21.26 |
| vehicle | 1781.70 | 23.72 | 36.53 |
| waveform | 47.80 | 0.00 | 21.97 |
| wine | 2490.30 | 32.29 | 40.06 |
| zoo | 5817.60 | 25.50 | 25.50 |
| **Average** | 1381.47 | 17.56 | 46.32 |

Table 6.3: Number of classification rules generated from frequent itemsets at 10% support and the percentage of rules dropped when closed and maximal itemsets are used

almost the same rules as those obtained from frequent patterns, but when maximal itemsets were used the number of classification rules always decreased. However, for our second hypothesis, the findings were not conclusive and further analysis (statistical and visual) with cost curves was necessary. In the remainder we will highlight these findings using different datasets and show how the rules were effectively reduced without putting the classification accuracy in jeopardy.

### UCI Datasets

Tables 6.1, 6.2 and 6.3 show the number of classification rules generated from frequent itemsets for 1%, 5% and 10% supports. Next to the number of rules, the percentage of rules reduced when they are obtained from closed and maximal is given. It can be observed from these tables that our first hypothesis is correct. The number of classification rules is substantially reduced when closed and maximal patterns are employed. In addition, it can be observed that the use of maximal patterns is the most beneficial.

When closed patterns mined at support 1% are used, the set of classification rules is reduced up to 57.19% (*iris* dataset). Under the same support condition, the set of classification rules generated from maximal patterns is up to 87.03% (*iris* dataset) smaller than the rules obtained from frequent patterns. Similar trends are observed for supports of 5% and

| dataset | Frequent | Closed | Maximal |
|---|---|---|---|
| anneal | 93.74 | 93.74 | 93.96 |
| australian | 85.53 | 85.53 | 85.53 |
| breast-w | 95.41 | 95.4 | 95.99 |
| cleve | 84.52 | 84.52 | 83.85 |
| crx | 85.37 | 85.37 | 85.37 |
| diabetes | 74.32 | 74.32 | 75.5 |
| german | 71.1 | 71.1 | 71.1 |
| glass | 72.02 | 72.95 | 72.95 |
| heart | 82.24 | 82.24 | 82.24 |
| hepatitis | 80.56 | 80.56 | 80.56 |
| horse | 84.25 | 83.98 | 85.05 |
| iris | 94.67 | 94.67 | 89.33 |
| labor | 86.34 | 86.34 | 91.66 |
| led7 | 71.81 | 71.81 | 71.81 |
| pima | 74.2 | 74.2 | 75.25 |
| tic-tac-toe | 97.92 | 97.92 | 98.02 |
| vehicle | 59.72 | 59.72 | 59.6 |
| waveform | 81.86 | 81.86 | 82.0 |
| wine | 91.03 | 91.59 | 89.37 |
| zoo | 86.18 | 86.18 | 86.18 |
| **Average** | 82.64 | 82.7 | 82.77 |

Table 6.4: Accuracy with FR scoring scheme

10%. For three of the datasets (*led7*, *tic-tac-toe* and *waveform*) the set of classification rules generated from closed patterns is less than 0.5% smaller than the set obtained from frequent patterns. However, the use of maximal patterns for these datasets is still advantageous as it reduces the set of rules between 21.26% (*tic-tac-toe* dataset) and 39.69% (*led7* dataset). Note that for *led7* dataset no classification rules are generated at 10% support. Although the reduction in number of rules is sometimes small for closed patterns, the maximal patterns produce a substantial reduction in most cases. For instance, when *heart* dataset is mined at support 10% closed patterns reduce the number of rules by only 2.07%, while the use of maximal itemsets lowers the number of rules by 52.9%. The averages provided in Tables 6.1, 6.2 and 6.3, shows that closed patterns reduce the number of rules by around 18% for all support thresholds, while the use of maximal patterns lowers the number of rules between 34.36% and 46.32% for the range of supports. Thus our hypothesis that the number of classification rules is substantially reduced by the use of closed and maximal patterns holds.

The second component of our hypothesis is about the level of performance of the classification systems. The results for the four associative classifiers investigated in our study are shown in Tables 6.4 to 6.7: Table 6.4 presents the accuracies of the ***FR*** method; Table 6.5 shows the accuracies of the ***AvR*** method; Table 6.6 presents the accuracies of the ***2SARC-CF*** method; and Table 6.7 presents the accuracies of the ***2SARC-RF*** method.

The results for ***FR*** method are presented in Table 6.4. When first rule scoring scheme ***FR*** is employed, the variation in the performance of the systems using closed and maximal patterns compared to the system built with frequent patterns is as follows: for closed

| dataset | Frequent | Closed | Maximal |
|---|---|---|---|
| anneal | 94.86 | 94.97 | 94.64 |
| australian | 87.27 | 87.26 | 87.12 |
| breast-w | 96.56 | 96 | 96.42 |
| cleve | 83.54 | 83.15 | 83.86 |
| crx | 86.38 | 86.55 | 86.53 |
| diabetes | 75.75 | 75.49 | 75.76 |
| german | 72 | 72 | 72 |
| glass | 70.62 | 71.08 | 71.54 |
| heart | 81.87 | 81.87 | 82.98 |
| hepatitis | 84.59 | 84.49 | 84.59 |
| horse | 84.22 | 84.49 | 84.25 |
| iris | 94.66 | 95.33 | 89.33 |
| labor | 88 | 88 | 90 |
| led7 | 71.53 | 71.53 | 71.58 |
| pima | 75.64 | 75.5 | 75.51 |
| tic-tac-toe | 93.85 | 93.85 | 93.75 |
| vehicle | 58.17 | 58.18 | 58.18 |
| waveform | 78.64 | 78.64 | 77.86 |
| wine | 96.09 | 93.8 | 92.69 |
| zoo | 92.09 | 91.09 | 91.09 |
| **Average** | 83.32 | 83.16 | 82.98 |

Table 6.5: Accuracy with AvR scoring scheme

patterns it ranges from -0.27% to 0.93%; for maximal itemsets it ranges from -5.36% to 5.32%. It can be observed that the biggest variation in accuracy occurs for *iris* for which the accuracy went down due to the fact the number of rules discovered was very small to start with, and *labor* for which the accuracy went up substantially. When *iris* and *labor* datasets are discarded, the range is much smaller (-1.66% to 1.18%). On average, both closed and maximal patterns improve by a small margin the classification performance.

The accuracy results for average rules scoring scheme ***AvR*** is shown in Table 6.5. The difference in performance for closed patterns when compared to frequent patterns ranges from -2.29% to 0.66%; for maximal itemsets it ranges from -5.33% to 1.11%. On average, both closed and maximal patterns slightly decrease the classification performance.

The performance of ***2SARC-CF*** system is presented in Table 6.6. The use of closed patterns decreases the performance by up to 1% and increases it by up to 1.66%; the variation in performance for maximal itemsets ranges from -3.34% to 3.33%. Note that, again, for maximal itemsets the range is much smaller (-1% to 0.58%) when the difference is computed without *iris* and *labor* datasets. On average, the performance increases slightly when closed patterns are used and decreases slightly when classification rules are generated from maximal patterns.

When ***2SARC-RF*** scoring scheme (results shown in Table 6.7) is considered the variation in the performance of the system for closed patterns ranges from -1.33% to 1.8%; for maximal itemsets it ranges from -3.99% to 3.67%. Note that for maximal itemsets the range is much smaller (-1.24% to 0.48%) when *iris* and *labor* datasets are discarded. On average, both closed and maximal patterns decrease insignificantly the classification performance.

| dataset | Frequent | Closed | Maximal |
|---|---|---|---|
| anneal | 98.45 | 98.01 | 98.01 |
| australian | 87.27 | 87.41 | 87.85 |
| breast-w | 97.28 | 97.29 | 97.28 |
| cleve | 84.84 | 84.5 | 84.18 |
| crx | 86.97 | 86.84 | 86.83 |
| diabetes | 77.05 | 77.18 | 76.14 |
| german | 74.9 | 74.7 | 74 |
| glass | 70.99 | 70.68 | 71.13 |
| heart | 85.2 | 84.83 | 84.46 |
| hepatitis | 87.08 | 86.42 | 86.41 |
| horse | 84.78 | 84.53 | 84.25 |
| iris | 95.34 | 95.33 | 92 |
| labor | 93.33 | 94.99 | 96.66 |
| led7 | 73.85 | 73.85 | 73.87 |
| pima | 75.51 | 75.11 | 74.86 |
| tic-tac-toe | 99.69 | 100 | 98.85 |
| vehicle | 66.91 | 68.21 | 67.38 |
| waveform | 79.7 | 79.8 | 79.06 |
| wine | 96.64 | 97.73 | 96.06 |
| zoo | 95.09 | 94.09 | 94.09 |
| **Average** | 85.54 | 85.58 | 85.17 |

Table 6.6: Accuracy for 2SARC-CF scoring scheme

These very small increases or decreases in accuracy do not seem to be significant, but a verification is required to support our hypothesis. We performed a series of Wilcoxon signed ranked tests between the classifier built from frequent patterns and the model built from maximal patterns. In addition, we tested the classifier built from frequent patterns versus the model built from closed patterns. There was indeed no statistically significant difference in the performance. The Wilcoxon test is a non-parametric test [Dem06], it does not make any assumptions about the distributions of the values. Based on average performance we can conclude that using maximal patterns is advantageous because the reduction in the number of rules is significant, while the improvement or the drop in the performance level is not statistically significant.

Table 6.8 shows on how many datasets the use of maximal patterns performs better (wins), as well as (ties) or worse (losses) than when frequent patterns are used.

The use of maximal patterns is more beneficial to ***FR*** and ***AvR*** systems. This is due to their naïve scoring schemes and thus reducing the redundancy in the rule set is beneficial. ***2SARC-CF*** and ***2SARC-RF*** use scoring schemes that are learned automatically from data and thus making the system less sensitive to the redundancy in the set of rules.

This empirical study shows that there is no significant effect on the accuracy of classifiers with different rule selection schemes when closed or maximal frequent patterns are used instead of all frequent itemsets. The gain, however, is in the reduction of classification rules. It remains to see whether this observation is still true with more challenging datasets such as microarray data which contain a relatively small set of samples.

| dataset | Frequent | Closed | Maximal |
|---|---|---|---|
| anneal | 99.01 | 99.01 | 99.12 |
| australian | 88.28 | 87.86 | 87.84 |
| breast-w | 97.43 | 97.29 | 97.42 |
| cleve | 84.19 | 83.52 | 84.21 |
| crx | 86.97 | 86.39 | 86.96 |
| diabetes | 75.49 | 75.36 | 74.33 |
| german | 73.6 | 73.8 | 73.8 |
| glass | 71.97 | 71.52 | 71.57 |
| heart | 85.94 | 85.94 | 85.94 |
| hepatitis | 85.84 | 87.64 | 85.8 |
| horse | 83.98 | 85.04 | 83.42 |
| iris | 95.99 | 96.65 | 92 |
| labor | 92.99 | 91.66 | 96.66 |
| led7 | 74.28 | 74.09 | 74.08 |
| pima | 74.86 | 74.33 | 73.82 |
| tic-tac-toe | 100 | 100 | 100 |
| vehicle | 71.99 | 71.15 | 70.92 |
| waveform | 80.8 | 80.88 | 81.28 |
| wine | 97.77 | 97.22 | 97.74 |
| zoo | 98.33 | 97.09 | 97.09 |
| **Average** | 85.98 | 85.82 | 85.7 |

Table 6.7: Accuracy for 2SARC-RF scoring scheme

| M vs. F | wins | losses | ties |
|---|---|---|---|
| FR | 9 | 4 | 7 |
| AvR | 8 | 9 | 3 |
| 2SARC-CF | 5 | 14 | 1 |
| 2SARC-RF | 5 | 13 | 2 |

Table 6.8: Maximal versus frequent on UCI datasets

### Microarray Datasets

Microarray data contains measurements of a large number of genes for a particular sample. Due to high acquisition costs, there is generally a small number of samples in microarray datasets. The large feature space (given by the number of genes) and the small number of samples make the construction of a good classifier difficult. We have access to microarray data for breast cancer, lung cancer, leukemia[1] and prostate cancer. A method for reducing the dimensionality of the feature space for these microarray data has been proposed in [AG07]: first, find biclusters in data (a bicluster represents a subset of genes that are similar for a subset of samples); second, transform the original data based on bicluster membership. This transformation reduces dramatically the feature space. In addition, the new features are binary, representing the membership in a bicluster. This new representation is highly suitable for association rule mining. Thus we investigated our framework on these

[1]Microarray data for leukemia is also referred as AML-ALL in the literature.

| dataset | Frequent (# rules) | Closed (% saved) | Maximal (% saved) |
|---|---|---|---|
| Breast Cancer | 11304 | 20.21 | 20.25 |
| Lung Cancer | 10139 | 22.6 | 22.6 |
| Leukemia | 40441.4 | 15.23 | 15.23 |
| Prostate Cancer | 95307.8 | 14.28 | 14.28 |

Table 6.9: Average number of classification rules generated from frequent itemsets and the percentage of rules dropped when closed and maximal itemsets are used

microarray datasets and the results are presented in Tables 6.9 and 6.10. All the datasets are obtained from [AG07] in this new representation. For each instance its membership to 30 bi-clusters is considered. All the results are average accuracies over 5-fold cross validation. It is relevant to notice that while the authors of [AG07] claim to have reached the best known classification results on these microarray datasets, our results using 2SARC with maximal patterns outperforms their classification results on Lung Cancer and Leukemia datasets (the best results reported in [AG07] for these datasets are as follows: Breast Cancer: 90.79, Lung Cancer: 96.13, Leukemia: 84.72 and Prostate Cancer: 86.77).

Table 6.9 shows the number of classification rules and their reduction when rules are generated from closed and maximal patterns. For all the microarray datasets the reduction in the number of rules is almost the same for closed and maximal, indicating yet again that using maximal patterns is indeed a winning strategy. The accuracy results for ***FR***, ***AvR*** and ***2SARC-RF*** methods (shown in Table 6.10) remain the same when closed and maximal patterns are used instead of frequent ones. The only variation occurs for ***2SARC-CF*** for *Lung Cancer* dataset: the performance insignificantly decreases for the approaches using closed and maximal patterns.

The results on microarray data and UCI datasets confirm our hypothesis, that the use of closed and maximal patterns maintains the level of performance while reducing the number of classification rules. Thus, based on our results so far, we can conclude that the use of maximal patterns is advantageous. However, we want to study when exactly is the use of maximal patterns more advantageous than closed patterns and vice-versa. We use cost curve analysis for this purpose.

### Cost Curve Analysis

In the previous sections we presented and discussed the accuracy obtained for all the studied methods. To gain a better insight into the examined framework we perform an analysis based on cost curves.

Cost curves [DH06] are evaluation tools for classification systems that have been proposed as an alternative to ROC curves [PF97]. Their advantage is that in their visualization one can easily see the performance of a classifier over the entire range of class frequencies and costs.

In our analysis we are interested to see under what conditions the use of closed and maximal patterns is advantageous to an associative classifier. Cost curves allow us to easily visualize and detect these conditions. Each classifier is represented by a straight line in the cost space. The y-axis (NEC) is the expected cost of a classifier normalized to be between 0 and 1. The x-axis (PC(+)) is the fraction of the total cost of using a classifier that is due to positive examples.

| dataset | Frequent | Closed | Maximal |
|---|---|---|---|
| FR scoring scheme | | | |
| Breast Cancer | 58.0 | 58.0 | 58.0 |
| Lung Cancer | 96.1 | 96.1 | 96.1 |
| Leukemia | 75.72 | 75.72 | 75.72 |
| Prostate Cancer | 79.82 | 79.82 | 79.82 |
| AvR scoring scheme | | | |
| Breast Cancer | 58.0 | 58.0 | 58.0 |
| Lung Cancer | 96.1 | 96.1 | 96.1 |
| Leukemia | 77.16 | 77.16 | 77.16 |
| Prostate Cancer | 84.34 | 83.6 | 83.64 |
| 2SARC-CF scoring scheme | | | |
| Breast Cancer | 81.58 | 81.58 | 81.58 |
| Lung Cancer | 97.24 | 96.7 | 96.7 |
| Leukemia | 92.88 | 92.88 | 92.88 |
| Prostate Cancer | 86.6 | 86.6 | 86.6 |
| 2SARC-RF scoring scheme | | | |
| Breast Cancer | 85.5 | 85.5 | 85.5 |
| Lung Cancer | 96.72 | 96.72 | 96.72 |
| Leukemia | 91.44 | 91.44 | 91.44 |
| Prostate Cancer | 86.6 | 86.6 | 86.6 |

Table 6.10: Accuracy on microarray datasets

The performance of a classifier is represented in the cost space by the following equation:

$$NEC = (1 - TPR - FPR) \times PC(+) + FPR \tag{6.1}$$

where TPR (true positive rate) and FPR (false positive rate) are defined as follows:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{6.2}$$

The true positives (TP) and true negatives (TN) are correct classifications, while false positives (FP) and false negatives (FN) are misclassifications. A false positive occurs when an object is incorrectly classified as positive, while a false negative occurs when a positive object is classified as negative.

The definition of the probability cost function ($PC(+)$) is given in Equation 6.3, where $p(+)$ is the probability of a given object being in the positive class and $C(+|-)$ is the cost incurred if an object in the negative class is misclassified as being in the positive class. Note that when there are no costs associated with the classification decision (as it is the case for our empirical study), $PC(+)$ reduces to $p(+)$.

$$PC(+) = \frac{p(+) \times C(+|-)}{p(+) \times C(-|+) + p(-) \times C(+|-)} \tag{6.3}$$

A line in the cost space shows how the performance of the system varies when the class distribution or cost change. Ideally, one wants to have a classifier represented by a hori-

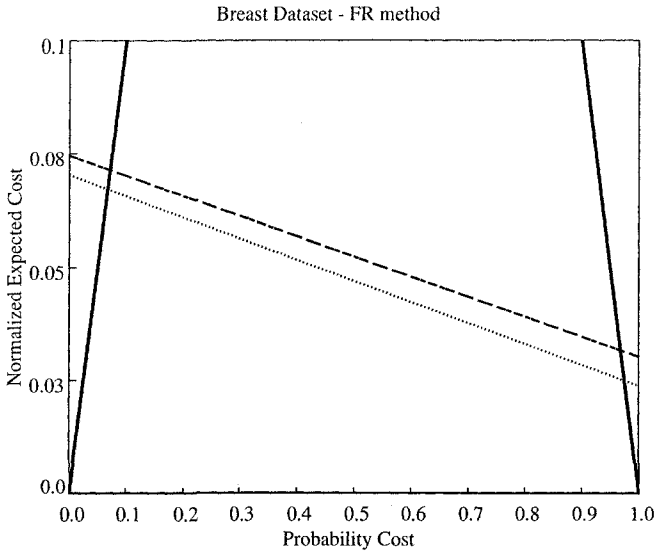


Figure 6.3: Cost curve performance for FR method on Breast-w dataset: frequent patterns - long-dashed line; closed patterns - dashed line; maximal patterns - dotted line; trivial classifiers - solid lines. Note that the dashed and long-dashed lines are overlapping.

zontal line at the normalized expected cost of 0. Thus, the best classifier for a probability cost value is the one whose normalized expected cost is the lowest. Visually, the lowest line in the graph represents the best classifier (the lower the line the better the classification system). From the graphs we can also directly see the difference in performance between two classifiers, which is their vertical height difference at some $PC(+)$ value.

Figures 6.3 to 6.6 show the performance of classification for several datasets. All the graphs show the performance of a classification method when classification rules are generated from frequent, closed and maximal patterns. Thus, three classifiers are compared in each graph. The classifier built from frequent patterns is represented by a long-dashed line. The one built from closed itemsets is shown with a dashed line. The dotted line corresponds to the classification system built from maximal patterns. The two solid lateral lines represent the trivial classifiers: the leftmost line represents the classifier that always predicts the negative class, while the rightmost line represents the classifier that always predicts positives. Note that cost curve evaluation can be done only for 2-class datasets.

Let us analyze the *Breast-w* dataset. As shown in Tables 6.1, 6.2 and 6.3 the use of closed patterns reduces only slightly (2%-9%) the set of classification rules, while the set of classification rules generated from maximal patterns is substantially smaller (40% to 50% smaller) than the one generated from all frequent itemsets. The accuracy for frequent and closed patterns is almost identical, while for maximal patterns it increases with 0.58%. Based on this information, one may conclude that the use of maximal patterns with ***FR*** method is the best choice. The cost curve shown in Figure 6.3 confirms this choice since the dotted line representing the maximal is indeed the lowest across the full range of possible $PC(+)$ values. Note that the dashed and long-dashed lines are overlapping, indicating that the classifier based on closed and frequent patterns have the same performance. Let us now look at the same dataset when ***2SARC-CF*** method is used. The reduction in number of rules is the same as with ***FR***. However, the performance in terms of accuracy is identical for

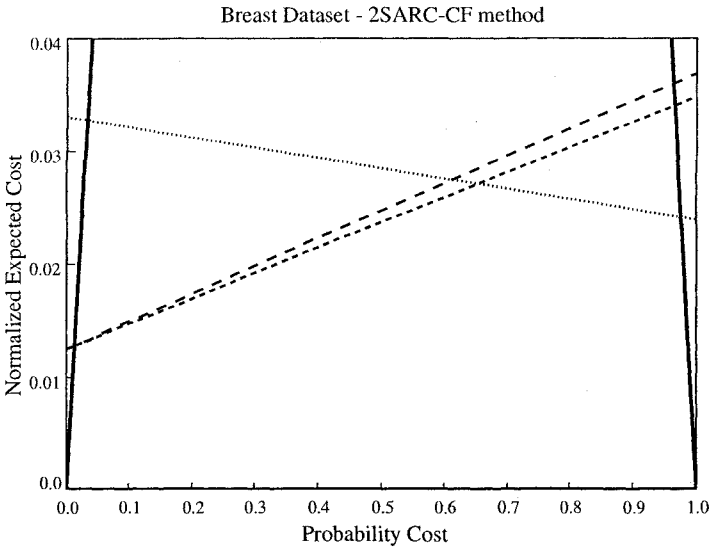


Figure 6.4: Cost curve performance for 2SARC-CF method on Breast-w dataset: frequent patterns - long-dashed line; closed patterns - dashed line; maximal patterns - dotted line; trivial classifiers - solid lines

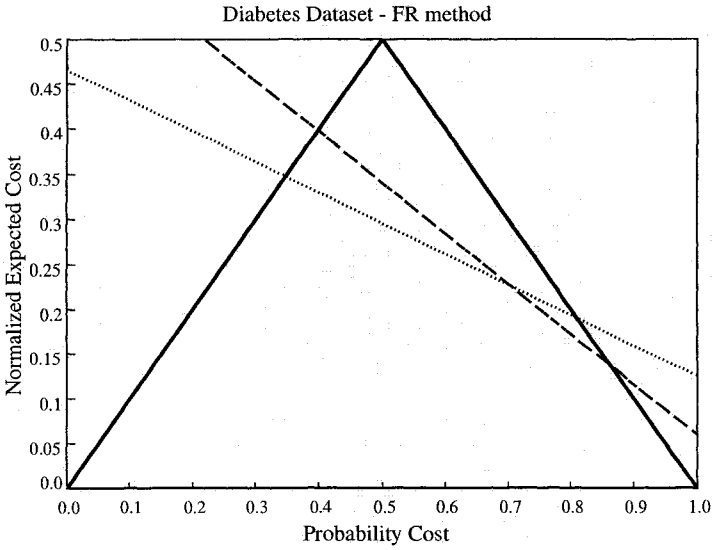


Figure 6.5: Cost curve performance for FR method on Diabetes dataset: frequent patterns - long-dashed line; closed patterns - dashed line; maximal patterns - dotted line; trivial classifiers - solid lines

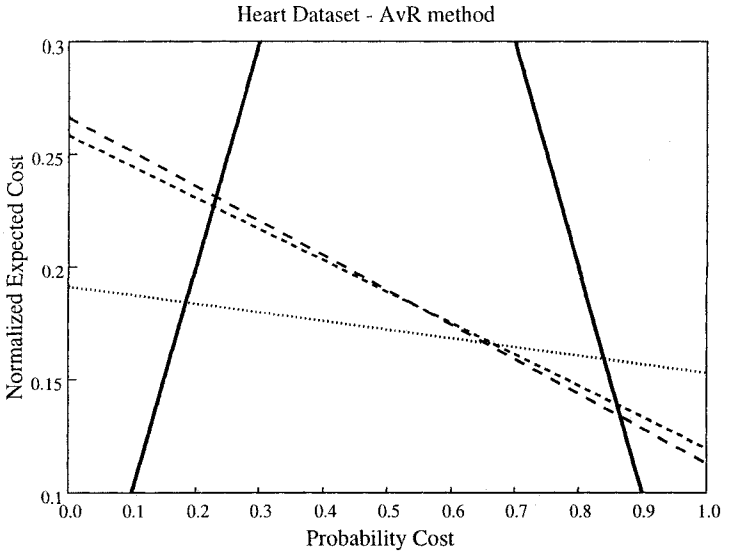


Figure 6.6: Cost curve performance for AvR method on Heart dataset: frequent patterns - long-dashed line; closed patterns - dashed line; maximal patterns - dotted line; trivial classifiers - solid lines

frequent, closed or maximal. Thus one may assume again, that the use of maximal patterns (they lead to the smallest set of classification rules) would be the best choice. However, the analysis with cost curves sheds new light on this choice (see Figure 6.4). It can be observed from the graph that maximal patterns should be used only for a probability cost higher than 0.65, while for a smaller probability, closed patterns should be preferred. The use of closed patterns reduce only slightly the set of classification rules. This would be an indication that also the performance should not vary too much between frequent and closed. This assumption is validated by both figures (Figure 6.3 and 6.4). In Figure 6.3 frequent and closed have exactly the same performance over the entire range of class frequencies, while in Figure 6.4 there is only a maximum difference of 0.0015 in the normalized expected cost.

*Diabetes* is another interesting dataset to be studied. The trend in rule reduction is similar to the *Breast-w* dataset, but the performance of the classifiers on this dataset is much poorer. The performance of ***FR*** method for *Diabetes* dataset is shown in Figure 6.5. Contrary to the classification systems in Figure 6.4, the classifiers do not perform well on the entire range of class frequencies. Indeed, the trivial classifier can outperform those classifiers. They should be used only on a smaller range of $PC(+)$ ([0.35-0.85]). Outside this interval trivial classifiers perform better. The use of maximal patterns is advantageous in the [0.35-0.7] range, while any of the frequent or closed patterns should be used in the [0.7-0.85] interval. Again, the classifiers built from frequent and closed patterns overlap in this graph.

Figure 6.6 presents the performance of ***AvR*** method for *Heart* dataset. ***AvR*** method performs better than the trivial classifiers in the [0.18-0.86] interval. Maximal patterns are more beneficial than frequent or closed patterns in the $PC(+)$ range [0.18-0.64]. Frequent patterns or even closed itemsets should be favored on the remaining interval of $PC(+)$.

In this section we have analyzed with cost curves several interesting cases. In general, our investigations using cost curves suggest that the use of maximal patterns leads to the

best classification performance over most of the probability ranges. In applications were the class distribution changes, one may want to use the cost curves to determine the best classifier.

## 6.5 Summary

In this chapter we investigated the performance of associative classifiers when the classification rules are generated from frequent, closed and maximal itemsets. We showed that maximal itemsets substantially reduce the number of classification rules without jeopardizing the accuracy of the classifier. Our extensive analysis demonstrates that the performance remains stable and even improves in some cases. Our analysis using cost curves also provides recommendations on when it is appropriate to remove redundancy in frequent itemsets. Based on our thorough analysis we are confident that any investigation of associative classifiers should consider first and foremost classification rules generated from maximal patterns.

# Chapter 7

# Conclusions

## 7.1 Summary

This dissertation has focused on improving associative classifiers in two directions. The first direction has dealt with improving the performance of associative classifiers by introducing new types of classification rules (Chapters 3 and 4) and by proposing a novel technique in the classification stage (Chapter 5). The second direction has dealt with solutions for reducing the number of classification rules (Chapter 6). In Chapter 6 we studied the redundancy reduction in the classification rule set through the use of maximal and closed patterns. In this dissertation we made contributions to all three stages of associative classifiers: rule generation in Chapters 3 and 4, rule pruning in Chapter 6 and rule selection in Chapter 5.

In Chapter 3, we introduced a new algorithm to generate both positive and negative association rules. Our method adds to the support-confidence framework the correlation coefficient to generate stronger positive and negative rules. We compared our algorithm with other existing algorithms on a real dataset. We discussed their performances on a small example for a better illustration of the algorithms and we presented and analyzed experimental results for a well-known text collection. The results proved that our algorithm can discover strong interesting patterns. In addition, our method generates all types of confined rules, thus allowing to be used in different applications where all these types of rules, or just a subset of them, could be needed. We demonstrated the potential of strong positive and negative correlated rules in the classification context. The results of the classification show that associative classifiers using a much smaller set of positive and negative association rules can perform similar or outperform existing classification systems.

Chapter 4 introduced the idea of combining associative classification and mining frequent itemsets with re-occurring items. We combined these two and presented ACRI, a new approach of associative classification with re-occurring items. We also suggest new strategies to select classification rules during the classification phase. In particular, using the cosine measure to estimate the similarity between objects to classify and available rules is very effective for associative classifiers that consider re-occurrence. When comparing ACRI approach with ARC-BC we found that considering repetitions of observed features is beneficial. In particular in the case of text categorization, repetition of words has discriminant power and taking these repetitions in consideration can generate good classification rules. Our experiments also showed that ACRI becomes more effective as the number of rules increases, in particular with our cosine measure for rule selection. Moreover, the accuracy of ACRI seems to be less sensitive to the support threshold, while most associative

classifiers are typically very sensitive to the support threshold, which is very difficult to determine effectively in practice.

In Chapter 5 we proposed a novel technique in the classification stage. Rule-based classifiers use predefined weighted voting schemes to combine the class predictions of the applicable rules. By contrast, the methods described in Chapter 5 automatically learn the scoring scheme. We achieve this by developing a two-stage system, with a layer of feature definitions interposed between the output of the first learning model and the input of the second. Our two stage classification system (2SARC) showed a good performance both for UCI datasets and text classification, under rigorous statistical analysis.

In Chapter 6 we investigated the performance of associative classifiers when the classification rules are generated from frequent, closed and maximal itemsets. We showed that maximal itemsets substantially reduce the number of classification rules without jeopardizing the accuracy of the classifier. Our extensive analysis demonstrates that the performance remains stable and even improves in some cases. Our analysis using cost curves also provides recommendations on when it is appropriate to remove redundancy in frequent itemsets. Based on our thorough analysis we are confident that any investigation of associative classifiers should consider first and foremost classification rules generated from maximal patterns.

## 7.2 Limitations and Future Directions

Although we address and present solutions to interesting problems for associative classifiers, there are some limitations in our solutions.

We introduced a new algorithm to generate confined positive and negative association rules. Our solution discovers a subset of all the possible positive and negative associations in the data. To mine all possible positive and negative associations in the data, it would be necessary not only to consider all items in a transaction, but also all possible items absent from the transaction. There could be a considerable exponential growth in the candidate generation phase. This is especially true in datasets with highly correlated attributes. That is why it is not feasible to extend the attribute space by adding the negated attributes and use the existing association rule mining algorithms. Although we presented a solution to this problem, generating negative association rules still remains an interesting and challenging open problem. A limitation of our solution is that it adds a new parameter based on the correlation measure to the rule mining phase. This increases the complexity of the system and adds an additional parameter to the associative classifier that has to be set and tuned for the application at hand.

We proposed and discussed the integration of rules with re-occurring items in the associative classifier framework. These types of rules can be very beneficial to the associative classifiers in some applications such as text categorization, image classification, etc. However, given that we have to take into account re-occurring items and to incorporate this information in the classification rules, it increases the complexity of the classification rule, thus making it more complicated to match rules to new objects. Some measures of similarity have to be used to calculate the rule matching.

We proposed a novel technique in the classification stage. We achieved this by developing a two-stage system, with a layer of feature definitions interposed between the output of the first learning model based on association rule mining and the input of the second learner. Although our solution increases the performance of the associative classifier, it reduces its

readability: it is more complicated to trace the categorization result to a small subset of rules that applied to the new object. The trade-off is between readability and performance. The complexity of the system is increased as well with the addition of the second level learner. In our experiments we found that the best choice for the second level learner is the k nearest neighbour algorithm. However, any classification system can be used and with the continuous research in this field better classification system will be developed. It is feasible that 2SARC could perform better with a different second stage classifier on some applications. Thus the user may choose to deal with selecting the most appropriate second stage learner for the application at hand. In applications with small training data, the performance of 2SARC is possibly hindered by the number of training examples, which are further reduced by the split of the training set into two subsets, one for each of the learning stages.

## 7.3 Final Word

This dissertation has focused on improving associative classifiers in two directions. The first direction was on improving the performance of associative classifiers by introducing new types of classification rules (negative association rules and association rules with re-occuring items) and by proposing a two stage architecture for the classification stage . The second direction was on finding solutions for reducing the number of classification rules. We investigated the redundancy reduction in the classification rule set through the use of maximal and closed patterns. We made new contributions in all the stages involved in building an associative classifier: rule generation, rule pruning and classification.

Overall, the contributions of this dissertation are important because they advance the state-of-the-art in associative classifiers. We showed new ways in which associative classifiers can be used and that the associative classifiers can be competitive classification systems. In addition, we highlighted new directions on associative classifiers research. The problems investigated in this dissertation and their proposed solutions constitute significant progress toward improving the quality of associative classifiers.

# Bibliography


[AG07] Nasimeh Asgarian and Rusell Greiner. Technical report, University of Alberta, 2007. http://www.cs.ualberta.ca/ greiner/R/RoBiC/.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of SIGMOD*, pages 207–216, 1993.

[AL99] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *KDD '99: Proceedings of ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 261–270, 1999.

[Ant02] Maria-Luiza Antonie. Categorizing digital documents by associating content features. Master's thesis, University of Alberta, 2002.

[AZ02] M.-L. Antonie and O. R. Zaïane. Text document categorization by term association. In *Proc. of ICDM*, pages 19–26, 2002.

[AZ04a] Maria-Luiza Antonie and Osmar R. Zaïane. An associative classifier based on positive and negative rules. In *9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD-04)*, pages 64–69, Paris, France, 2004.

[AZ04b] Maria-Luiza Antonie and Osmar R. Zaïane. Mining positive and negative association rules: An approach for confined rules. In *8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 04)*, pages 27–38, Pisa, Italy, 2004.

[AZC01] Maria-Luiza Antonie, Osmar R. Zaïane, and Alexandru Coman. Application of data mining techniques for medical image classification. In *Proc. of Int'l Workshop on Multimedia Data Mining (MDM/KDD)*, pages 94–101, 2001.

[AZH06] M.-L. Antonie, O. R. Zaïane, and R. Holte. Learning to use a learned model: A two-stage approach to classification. In *Proc. of ICDM*, 2006.

[Bay97] R. Bayardo. Brute-force mining of high-confidence classification rules. In *Proc. of SIGKDD*, pages 123–126, 1997.

[Bay98] Roberto J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 85–93, 1998.

[BEYTW01] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. On feature distributional clustering for text categorization. In *Proc. of SIGIR*, pages 146–153, New Orleans, US, 2001.

[Bis95] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, pages 265–276, Tucson, Arizona, May 1997.

[BMUT97] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of SIGMOD*, pages 255–264, 1997.

[Bor08a] C. Borgelt. Apriori software. http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html, 2008. Last accessed in August 2008.

[Bor08b] C. Borgelt. Backpropagation software. http://fuzzy.cs.uni-magdeburg.de/~borgelt/mlp.html, 2008. Last accessed in August 2008.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[CH67] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions in Information Theory*, IT-13(1):21–27, 1967.

[CH98] W.W. Cohen and H. Hirsch. Joins that generalize: text classification using Whirl. In *Proc. of SIGKDD*, pages 169–173, 1998.

[CL04] F. Coenen and P. H. Leng. An evaluation of approaches to classification rule selection. In *Proc. of ICDM*, pages 359–362, 2004.

[CN89] Peter Clark and Tim Niblett. The cn2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[Coh88] J. Cohen. *Statistical power analysis for the behavioral sciences (2nd ed.).* Lawrence Erlbaum, New Jersey, 1988.

[Coh95] W. Cohen. Fast effective rule induction. In *Proc. of ICML*, pages 115–123, 1995.

[CS99] W.W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141 –173, 1999.

[Dem06] J. Demsar. Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, 1973.

[DH06] Chris Drummond and Robert C. Holte. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130, 2006.

[Elk97] C. Elkan. Boosting and naive bayesian learning, 1997.

[FB91] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248, 1991.

[Fox92] C. Fox. *Information Retrieval: Data Structures and Algorithms*, chapter Lexical analysis and stoplists, pages 113–116. Prentice-Hall, 1992. ftp://sunsite.dcc.uchile.cl/pub/users/rbaeza/irbook/stopper/.

[FW94] J. Furnkranz and G. Widmer. Incremental reduced error pruning. In *Proc. of ICML*, pages 70–77, 1994.

[GZ03] B. Goethals and M.J. Zaki, editors. *FIMI'03: Workshop on Frequent Itemset Mining Implementations*, volume 90 of *CEUR Workshop Proceedings series*, 2003.

[HKP91] J.A. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Inc., Redwood City, CA, 1991.

[Hop02] Will Hopkins. A new view of statistics. http://www.sportsci.org/resource/stats/, 2002. Last accessed in August 2008.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of SIGMOD*, pages 1–12, 2000.

[Hul94] D. A. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proc. of SIGIR*, pages 282–289, 1994.

[Joa98] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of ECML*, pages 137–142, 1998.

[JS02] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, 6(5), 2002.

[Lew98] D. Lewis. Naïve (Bayes) at forty: The independence assumption in information retrieval. In *Proc. of ECML*, pages 4–15, 1998.

[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of SIGKDD*, pages 80–86, 1998.

[LHP01] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proc. of ICDM*, pages 369–376, 2001.

[LJ98] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[Mar61] M. Maron. Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8(3):404–417, 1961.

[ONL01] K.-L. Ong, W.-K. Ng, and E.-P. Lim. Multi-level rules with recurrent items using fp'-tree. In *Proc. of ICICS*, 2001.

[PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. *Lecture Notes in Computer Science*, 1540:398–416, 1999.

[PCT+03] Feng Pan, Gao Cong, Anthony K. H. Tung, Jiong Yang, and Mohammed J. Zaki. Carpenter: finding closed patterns in long biological datasets. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–642. ACM, 2003.

[PF97] Foster J. Provost and Tom Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Knowledge Discovery and Data Mining*, pages 43–48, 1997.

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

[Qui96] J. R. Quinlan. Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press, 1996.

[Reu08] The Reuters-21578 text categorization test collection, 2008. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 318–362, 1986.

[RKR07] R. Rak, L. A. Kurgan, and M. Reformat. Multilabel associative classification categorization of medline articles into mesh keywords. *IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine & Biology Society*, 26(2):47–55, 2007.

[RKR08] Rafal Rak, Lukasz Kurgan, and Marek Reformat. A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation. *Data Knowl. Eng.*, 64(1):171–197, 2008.

[RSZA05] R. Rak, W. Stach, O. R. Zaïane, and M.-L. Antonie. Considering re-occurring features in associative classifiers. In *Proc. of PAKDD*, pages 240–248, 2005.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, New York, NY, USA, 1996. ACM.

[Sch99] R. E. Schapire. Theoretical views of boosting. In *Proc. of European Conference on Computational Learning Theory (EuroCOLT)*, pages 1–10, 1999.

[Seb99] F. Sebastiani. Machine learning in automated text categorization. Technical Report IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999.

[SON98] A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. In *Proc. of ICDE*, pages 494–502, 1998.

[THC02] W.G. Teng, M.J. Hsieh, and M.S. Chen. On the mining of substitution rules for statistically dependent items. In *Proc. of ICDM*, pages 442–449, 2002.

[TK00] P.N. Tan and V. Kumar. Interestingness measures for association patterns: A perspective. In *Proc. of Workshop on Postprocessing in Machine Learning and Data Mining*, 2000.

[TKS02] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proc. of SIGKDD*, pages 32–41, 2002.

[TWL02] C. M. Tan, Y. F. Wang, and C. D. Lee. The use of bigrams to enhance text categorization. *Information Processing and Management*, 38(4):529–546, 2002.

[Vap95] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.

[VJGZ07] Adriano Veloso, Wagner Meira Jr., Marcos André Gonçalves, and Mohammed Javeed Zaki. Multi-label lazy associative classification. In *PKDD*, pages 605–612, 2007.

[VJZ06] Adriano Veloso, Wagner Meira Jr., and Mohammed J. Zaki. Lazy associative classification. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 645–654, 2006.

[vR79] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[VWMC+06] Adriano Veloso, Jr. Wagner Meira, Marco Cristo, Marcos Gonçalves, and Mohammed Zaki. Multi-evidence, multi-criteria, lazy associative document classification. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 218–227, 2006.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[WK91] S.M. Weiss and C.A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, 1991.

[WK05] J. Wang and G. Karypis. Harmony: Efficiently mining the best rules for classification. In *Proc. of SIAM*, 2005.

[Wol92] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[WYY04] W. Wang, J. Yang, and P. Yu. War: Weighted association rules for item intensities. *Knowledge and Information Systems*, vol. 6:203–229, 2004.

[WZZ02] X. Wu, C. Zhang, and S. Zhang. Mining both positive and negative association rules. In *Proc. of ICML*, pages 658–665, 2002.

[Yan99] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90, 1999.

[YH03] X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *Proc. of SDM*, 2003.

[ZA02] Osmar R. Zaïane and Maria-Luiza Antonie. Classifying text documents by associating terms with text categories. In *Proc. of Australasian Database Conference (ADC'02)*, pages 215–222, 2002.

[Zak99] M. J. Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency: Special Issue on Parallel Mechanisms for Data Mining*, 7(4):14–25, 1999.

[ZEH05] O. R. Zaïane and M. El-Hajj. Pattern lattice traversal by selective jumps. In *Proc. of SIGKDD*, pages 729–735, 2005.

[ZHZ00] Osmar R. Zaïane, J. Han, and H. Zhu. Mining recurrent items in multimedia with progressive resolution refinement. In *Proc. of ICDE*, pages 461–470, 2000.