Rule Generation and Rule Selection Techniques for Cost-Sensitive Associative Classification*

Adriano Veloso[1] **, Wagner Meira Jr**[1]

[1]Computer Science Department – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

{adrianov, meira}@dcc.ufmg.br

Abstract. *Classification aims to assign a data object to its appropriate class, what is traditionally performed through a small dataset model such as decision tree. Associative classification is a novel strategy for performing this task where the model is composed of a particular set of association rules, in which the consequent of each rule (i.e., its right-hand-side) is restricted to the classification class attribute. Rule generation and rule selection are two major issues in associative classification. Rule generation aims to find a set of association rules that better describe the entire dataset, while rule selection aims to select, for a particular case, the best rule among all rules generated. Rule generation and rule selection techniques dramatically affect the effectiveness of the classifier. In this paper we propose new techniques for rule generation and rule selection. In our proposed technique, rules are generated based on the concept of maximal frequent class itemsets (increasing the size of the rule pattern), and then selected based on their informative value and on the cost that an error imply (possibly reducing misclassifications). We validate our techniques using two important real world problems: spam detection and protein homology detection. Further, we compare our techniques against other existing ones, ranging from well known naïve-Bayes to domain-specific classifiers. Experimental results show that our techniques are able to achieve a significant improvement of 30% in the effectiveness of the classification.*

1. Introduction

Classification is a widely used data mining technique that consists of generating a dataset model (i.e., a decision tree or a rule set) which can then be used to assign a data object (or record) to its appropriate class. Associative classification, which is the subject of this paper, represents the dataset model by a rule set. Classification generally involves two phases, training and test. In the training phase the rule set is generated from the training data, where each rule associates a pattern to a class. In the test phase the generated rule set is used to decide the class that a test data record belongs to. Traditionally, greedy search techniques such as decision trees [Breiman et al. 1984, Quinlan 1986], rule learning [Liu et al. 1998, Li et al. 2001, Yin and Han 2003], and naïve-Bayes [Domingos and Pazzani 2000], are used to generate the rule set. These greedy techniques generate small rule sets because during the rule generation the overlapping between training data records tends to be minimized. However, small rule sets have some disadvantages:

*This research was sponsored by UOL (www.uol.com.br) through its UOL Bolsa Pesquisa program, process number 200503312358.

1. Rule set utilization: Small rule sets are very sensitive to missing values in the test data, that is, values that occur in the training data but not in the test data. As a result, some generated rules are useless during the test phase.
2. Coverage: Small rule sets rely significantly on the default decision, that is, when there is no rule with a pattern that cover a test data record.

Different from greedy techniques (which may only achieve global optimality if the problem has a optimal substructure), association rule mining searches globally for all rules that satisfy some minimum support and minimum confidence constraints [Agrawal et al. 1993], thus, it is able to achieve global optimality. For this reason, integrating classification and association rule mining is thus a suitable alternative for generating larger rule sets and to overcome the aforementioned problems. This integration is called *associative classification*, and it may be done by focusing on a particular subset of all possible association rules: the rules where the consequent (or right-hand-side) are restricted to the classification class attribute (*class association rules*). Rule generation and rule selection are two major issues in associative classification. Rule generation aims to find a set of association rules that better describe the entire dataset, while rule selection aims to select, for sake of classifying a given sample, the best rule among all rules generated. Recent studies [Liu et al. 1998, Li et al. 2001, Yin and Han 2003] showed that associative classification is generally more accurate than traditional classification techniques. However, current algorithms for associative classification still suffer from some problems:

1. Large rule sets: These algorithms generate very large rule sets during the training phase, and a large fraction of the generated rules may be useless in the test phase.
2. Small patterns: These algorithms generate rules composed of small patterns (or itemsets), that is, the itemset implicit in the rule has only few attributes. Rules derived from small itemsets do not work well because they do not provide enough information regarding the data object (record) to be classified. Small patterns tend to be very abstract, while long patterns tend to be very specific.
3. Overfitting and biased classification: When using association rules to classify a data object, it is possible that more than one rule can be applied. These algorithms select the best rules based on confidence, but high-confidence rules are very sensitive to noise in the training data, possibly incurring in biased rules.
4. Cost-Sensitive classification: These algorithms are not cost-sensitive, that is, they are unable to decide, during the test phase, if it is better to classify or not a data object based on the cost that a misclassification may imply.

In this paper we propose new techniques for rule generation and rule selection in associative classification. During the training phase only a representative set of rules derived from *maximal frequent class itemsets* is generated, making possible the generation of rules containing more items (or attributes) and, consequently, resulting in more informative rules. The rule set generated from those *maximal frequent class rules* may be orders of magnitude smaller than the rule set that would be generated from the complete set of class association rules, but the effectiveness of our technique is guaranteed by the self-containment property, that is, if a maximal frequent class rule does not apply to a given sample, the rule can be broken into sub-rules, until a sub-rule can be applied to that sample (or the default classification is used). Sub-rules are generated on-demand

during the test phase, maximizing the rule set utilization (i.e., the number of useless rules is reduced). If, during the test phase, more than one rule applies to the same sample, the best rule among those is selected based first on its size (how informative the rule is) and then on its confidence (how strong the rule is), reducing biased classifications. Further, our rule selection technique is able to account for the cost that a classification error may imply, and, as a consequence, to reduce the number of misclassifications.

The remaining of the paper is organized as follows. In the next section we will provide necessary background on classification and association rule mining problems, and then we will discuss existing related work. In Section 3 we will present our techniques for rule generation and rule selection. In Section 4 we will present experimental results by applying our techniques to real-world problems, such as spam detection and protein homology detection. Finally, in Section 5 we will present our conclusions and future work possibilities.

2. Background and Related Work

In this section we provide preliminary concepts of association rule mining and associative classification. Further, we also discuss some related work.

2.1. Association Rule Mining

DEFINITION 1. [ITEMSETS] For any set $\mathcal{X}$, its size is the number of elements in $\mathcal{X}$. Let $\mathcal{I}$ denote the set of n natural numbers $\{1, 2, \ldots, n\}$. Each $x \in \mathcal{I}$ is called an item. A non-empty subset of $\mathcal{I}$ is called an itemset. The power set of $\mathcal{I}$, denoted by $\mathcal{P}(\mathcal{I})$, is the set of all possible subsets of $\mathcal{I}$. An itemset of size k, $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ is called a k-itemset (for convenience we drop set notation and denote $\mathcal{X}$ as $x_1x_2 \ldots x_k$). For $\mathcal{X}$, $\mathcal{Y} \in \mathcal{P}(\mathcal{I})$ we say that $\mathcal{X}$ contains $\mathcal{Y}$ if $\mathcal{Y} \subseteq \mathcal{X}$. A set (of itemsets) $\mathcal{C} \subseteq \mathcal{P}(\mathcal{I})$ is called an itemset collection.

DEFINITION 2. [TRANSACTIONS] A transaction $\mathcal{T}_i$ is an itemset, where i is a natural number called the transaction identifier or tid. A transaction database $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$, is a finite set of transactions, with size $\mid \mathcal{D} \mid$. The support of an itemset $\mathcal{X}$ in $\mathcal{D}$ is the fraction of transactions in $\mathcal{D}$ that contain $\mathcal{X}$, given as, $\sigma(\mathcal{X}, \mathcal{D}) = \frac{|\{\mathcal{T}_i \in \mathcal{D} | \mathcal{X} \subseteq \mathcal{T}_i\}|}{|\mathcal{D}|}$.

DEFINITION 3. [FREQUENT AND MAXIMAL FREQUENT ITEMSETS] An itemset $\mathcal{X}$ is frequent if $\sigma(\mathcal{X}, \mathcal{D}) \geq \sigma^{min}$, where σ^{min} is a user-specified minimum-support threshold, with $0 < \sigma^{min} \leq 1$. A collection of frequent itemsets is denoted as $\mathcal{F}(\sigma^{min}, \mathcal{D})$. A frequent itemset $\mathcal{X} \in \mathcal{F}(\sigma^{min}, \mathcal{D})$ is $maximal$ if it has no frequent superset. A collection of maximal frequent itemsets is denoted as $\mathcal{MF}(\sigma^{min}, \mathcal{D})$.

LEMMA 1. [Agrawal and Srikant 1994] Any subset of a frequent itemset is frequent: $\mathcal{X} \in \mathcal{F}(\sigma^{min}, \mathcal{D})$ and $\mathcal{Y} \subseteq \mathcal{X}$ implies $\mathcal{Y} \in \mathcal{F}(\sigma^{min}, \mathcal{D})$. Thus, by definition a frequent itemset must be subset of at least one maximal frequent itemset. ■

LEMMA 2. [Gouda and Zaki 2001] $\mathcal{MF}(\sigma^{min}, \mathcal{D})$ is the smallest collection of itemsets from which $\mathcal{F}(\sigma^{min}, \mathcal{D})$ can be inferred. ■

PROBLEM 1. [MINING (MAXIMAL) FREQUENT ITEMSETS] Given σ^{min} and a transaction database $\mathcal{D}$, the problem of mining frequent itemsets is to find $\mathcal{F}(\sigma^{min}, \mathcal{D})$. Similarly, the problem of mining maximal frequent itemsets is to find $\mathcal{MF}(\sigma^{min}, \mathcal{D})$.

EXAMPLE 1. Let us consider the example in Figure 1, where $\mathcal{I}$ = {1,2,3,4,5} and the figure shows $\mathcal{D}$ and $\mathcal{P}(\mathcal{I})$. Suppose σ^{min} = 0.4 (40%). $\mathcal{F}(0.4, \mathcal{D})$ is composed of the shaded and bold itemsets, while $\mathcal{MF}(0.4, \mathcal{D})$ is composed only of the bold itemsets. Note that $\mid \mathcal{MF}(0.4, \mathcal{D}) \mid$ is much smaller than $\mid \mathcal{F}(0.4, \mathcal{D}) \mid$. A naive approach to find $\mathcal{F}(0.4, \mathcal{D})$ is to first compute $\sigma(\mathcal{X}, \mathcal{D})$ for each $\mathcal{X} \in \mathcal{P}(\mathcal{I})$, and then return only those that $\sigma(\mathcal{X}, \mathcal{D}) \geq$ 0.4. This approach is inappropriate because:

- If $\mid \mathcal{I} \mid$ (the dimension) is high, then $\mid \mathcal{P}(I) \mid$ is huge (i.e., $2^{|\mathcal{I}|}$).
- If $\mid \mathcal{D} \mid$ (the size) is large, then computing $\sigma(\mathcal{X}, \mathcal{D})$ for all $\mathcal{X} \in \mathcal{P}(\mathcal{I})$ is infeasible.

By applying lemma 1 we can significantly improve the search for $\mathcal{F}(0.4, \mathcal{D})$. In Figure 1, once we know that the itemset {34} is not frequent, we do not need to generate any of its supersets, since they must also be not frequent. This simple pruning trick was first used in [Agrawal and Srikant 1994] and it greatly reduces the number of candidate itemsets generated. Several different searches can be applied by using this pruning trick.

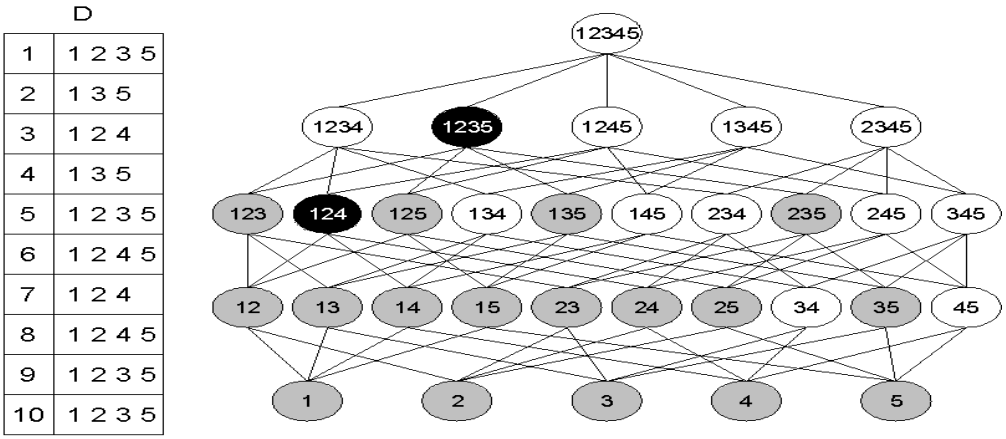

Figure 1. Frequent Itemset Mining Example.

ALGORITHM 1. [MINING (MAXIMAL) FREQUENT ITEMSETS] The algorithm was presented in [Gouda and Zaki 2001]. It employs a backtrack search to find $\mathcal{F}(\sigma^{min}, \mathcal{D})$ and $\mathcal{MF}(\sigma^{min}, \mathcal{D})$. A solution is represented as an itemset $\mathcal{X} = \{x_0, x_1, \ldots\}$. Each item x_j is chosen from a finite *possible set*, $\mathcal{P}_j$. Initially $\mathcal{X}$ is empty; it is extended one item at a time, as the search proceeds. The size of $\mathcal{X}$ is the same as the depth of the corresponding itemset in the search tree. Given a k-candidate itemset $\mathcal{X}_k = \{x_0, x_1, \ldots, x_{k-1}\}$, the possible values for the next item x_k comes from a subset $\mathcal{V}_k \subseteq \mathcal{P}_k$ called the *combine set*. Each iteration tries extending $\mathcal{X}_k$ with every item x in $\mathcal{V}_k$. An extension is valid if the resulting itemset $\mathcal{X}_{k+1}$ is frequent. If $\mathcal{X}_{k+1}$ is frequent and it is no subset of any already known maximal frequent itemset, then $\mathcal{X}_{k+1}$ is a maximal frequent itemset. The next step is to extract the new possible set of extensions, $\mathcal{P}_{k+1}$, which consists only of elements in $\mathcal{V}_k$ that follow x. The new combine set, $\mathcal{V}_{k+1}$ consists of those elements in the possible

set that produce a frequent itemset when used to extend $\mathcal{X}_{k+1}$. Any item not in the combine set refers to a pruned subtree. The algorithm performs a depth-first traversal of the search space. When the search finishes, $\mathcal{F}(\sigma^{min}, \mathcal{D})$ and $\mathcal{MF}(\sigma^{min}, \mathcal{D})$ were found. The computation of $\sigma(\mathcal{X}, \mathcal{D})$ is based on the associativity of subsets. Let $\mathcal{L}_{\mathcal{D}}(\mathcal{X})$ be the $tidset$ of $\mathcal{X}$ in $\mathcal{D}$ (the set of $tids$ in $\mathcal{D}$ in which $\mathcal{X}$ has occurred), and thus, $\mid \mathcal{L}_{\mathcal{D}}(\mathcal{X}) \mid = \pi(\mathcal{X}, \mathcal{D})$. According to [Zaki et al. 1997], $\mathcal{L}_{\mathcal{D}}(\mathcal{X})$ can be obtained by intersecting the tidsets of two subsets of $\mathcal{X}$. For example, in Figure 1, $\mathcal{L}_{\mathcal{D}}(123)$ = {1, 5, 9, 10} and $\mathcal{L}_{\mathcal{D}}(125)$ = {1, 5, 8, 9, 10}. Consequently, $\mathcal{L}_{\mathcal{D}}(1235) = \mathcal{L}_{\mathcal{D}}(123) \cap \mathcal{L}_{\mathcal{D}}(125)$ = {1, 5, 9, 10}. $\mid \mathcal{L}_{\mathcal{D}}(1235) \mid = \pi(1235, \mathcal{D})$ = 4, and thus $\sigma(1235, \mathcal{D})$ = 0.4.

EXAMPLE 2. Figure 2 shows how the algorithm works. We used sequence numbers above each itemset to facilitate the understanding of the backtrack search. First the algorithm process the items, and then it starts a depth-first search for frequent itemsets. For example, for sequence number s = 13, the itemset being processed is $\{124\}$. Therefore, the depth is k = 3, and $\mathcal{P}_3 = \mathcal{V}_3 = \{5\}$. When s = 15, the algorithm visits the itemset $\{124\}$ again, but now $\mathcal{V}_3 = \emptyset$ and consequently $\{124\} \in \mathcal{MF}(0.4, \mathcal{D})$. Although an itemset $\mathcal{X}$ can be visited more than once, $\sigma(\mathcal{X}, \mathcal{D})$ is computed only in the first visit.

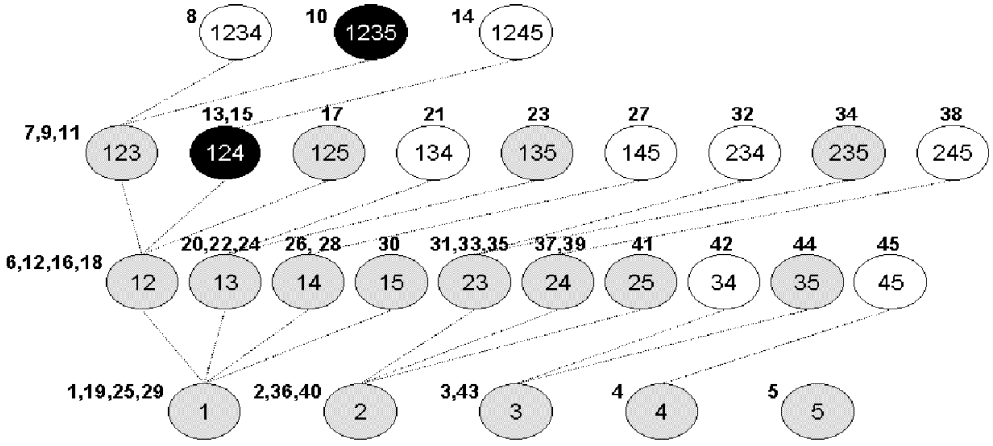

Figure 2. Basic Algorithm for (Maximal) Frequent Itemset Mining.

DEFINITION 4. [ASSOCIATION RULES] An association rule is a rule with the form $\mathcal{X} \xrightarrow{\theta} \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are two itemsets with $\mathcal{X} \cap \mathcal{Y} = \emptyset$, and θ is the confidence of the rule, which is given by: $\frac{\sigma(\mathcal{X} \cup \mathcal{Y}, \mathcal{D})}{\sigma(\mathcal{X}, \mathcal{D})}$.

PROBLEM 2. [GENERATING ASSOCIATION RULES] Given a minimum confidence threshold θ^{min} and $\mathcal{F}(\sigma^{min}, \mathcal{D})$, the problem of generating association rules is to find $\mathcal{R}(\theta^{min}, \mathcal{F}(\sigma^{min}, \mathcal{D}))$.

2.2. Associative Classification

DEFINITION 5. [DATA OBJECTS, TRAINING AND TEST DATA] A data object $\mathcal{O}$ is a transaction in $\mathcal{D}$, with $\mathcal{D} = \mathcal{D}_{seen} \cup \mathcal{D}_{unseen}$, where $\mathcal{D}_{seen}$ is the training data and $\mathcal{D}_{unseen}$ is the test data. If $\mathcal{O}$ is followed by the class attribute c (with $c \in$ {positive, negative}), then $\mathcal{O} \in \mathcal{D}_{seen}$, otherwise $\mathcal{O} \in \mathcal{D}_{unseen}$.

DEFINITION 6. [(MAXIMAL) FREQUENT CLASS ITEMSETS] A frequent itemset $\mathcal{X} \in \mathcal{F}(\sigma^{min}, \mathcal{D}_{seen})$ is a class itemset if $c \in \mathcal{X}$, where c is the class attribute. Similarly, a maximal frequent itemset $\mathcal{Y} \in \mathcal{MF}(\sigma^{min}, \mathcal{D}_{seen})$ is a maximal class itemset if $c \in \mathcal{Y}$. The set of all frequent class itemsets in $\mathcal{D}_{seen}$ is denoted as $\mathcal{F}(c, \sigma^{min}, \mathcal{D}_{seen})$ (i.e., it is composed of frequent itemsets that contain the attribute c), and similarly, the set of all maximal frequent class itemsets in $\mathcal{D}_{seen}$ is denoted as $\mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen})$.

ALGORITHM 2. [MINING (MAXIMAL) FREQUENT CLASS ITEMSETS] Algorithm 1 may be easily modified for mining frequent class itemsets. All that need to be done is to focus the search on itemsets that contain the class attribute c, that is an itemset $\mathcal{X} = \{x_0, x_1, \ldots\}$ is only a valid solution if $c \in \mathcal{X}$. As in Algorithm 1, each item x_j is chosen from a finite *possible set*, $\mathcal{P}_j$. Initially $\mathcal{X} = c$; it is extended one item at a time, as the search proceeds. Any generated itemset that does not contain c is pruned. The process continues untill all frequent class itemsets are found.

ALGORITHM 3. [MINING ONLY MAXIMAL FREQUENT CLASS ITEMSETS] Algorithm 2 can be easily modified for mining only maximal frequent class itemsets. All that need to be done is to introduce a maximality pruning [Gouda and Zaki 2001], that is, a subtree can be pruned if an itemset is subset of an already known maximal frequent itemset.

DEFINITION 7. [(MAXIMAL) CLASS ASSOCIATION RULES] A class association rule is a rule with the form $\{\mathcal{X}-c\} \xrightarrow{\theta} c$, where c is the class attribute and $\mathcal{X} \in \mathcal{F}(c, \sigma^{min}, \mathcal{D}_{seen})$. Similarly, a maximal class association rule is a rule with the form $\{\mathcal{Y} - c\} \xrightarrow{\theta} c$, where c is the class attribute and $\mathcal{Y} \in \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen})$.

PROBLEM 3. [GENERATING (MAXIMAL) CLASS ASSOCIATION RULES] The problem of generating frequent class association rules is to find $\mathcal{R}(\theta^{min}, \mathcal{F}(c, \sigma^{min}, \mathcal{D}_{seen}))$. Similarly, the problem of generating maximal frequent class association rules is to find $\mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$. Maximal frequent class association rules have the self-containment property, that is, a frequent class association rule in $\mathcal{R}(\theta^{min}, \mathcal{F}(c, \sigma^{min}, \mathcal{D}_{seen}))$ is a sub-rule of at least one maximal frequent class association rule in $\mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$ (this property is due to Lemma 2).

2.3. Related Work

Our work is clearly different from traditional classification algorithms, such as ID3 [Quinlan 1986], C4.5 [Quinlan 1993], and CART [Breiman et al. 1984], which only produce small rule sets. Our proposed work is related to associative classification techniques, which was introduced by Liu et al. in [Liu et al. 1998] with the CBA classification system. Enhancements were proposed in [Yin and Han 2003] where best rules are selected based on the *Ocam's Razor* principle, and in [Li et al. 2001] where the classification is based on multiple best rules (rather than on one single best rule). Another interesting work on associative classification is presented in [Baralis and Chiusano 2004], where a technique based on closed frequent itemsets [Zaki and Hsiao 2002] is proposed. This technique generates rule sets composed of non-redundant rules. Three characteristics

make our work different from the aforementioned ones: (1) our proposed rule generation technique is based on maximal frequent class rules, (2) our rule selection technique is based on how much specific the rule is (the size of the rule), and (3) our classification is cost-sensitive (a classification error has a cost, and different errors have different costs).

There are several algorithms for association rule mining. Algorithm 1 and Algorithm 2 presented above are related to APRIORI [Agrawal and Srikant 1994], ECLAT [Zaki et al. 1997] and FP-Growth [Han et al. 2000]. Algorithm 3 presented above is closely related to GENMAX [Gouda and Zaki 2001] and MAXMINER [Bayardo 1998], however Algorithm 3 performs a search focused just on the maximal frequent class itemsets (and not on all maximal frequent itemsets).

3. Cost-Sensitive Associative Classification based on Maximal Frequent Class Itemsets

In this section we present our techniques for rule generation and selection. During rule generation a rule set composed of rules of the form $\mathcal{X} \xrightarrow{\theta} c$ (where $\mathcal{X}$ is a pattern, or itemset, in $\mathcal{D}_{seen}$ and c is the class attribute) is computed. During rule selection for a given data object $\mathcal{O}$, a subset of rules matching $\mathcal{O}$ is extracted and the best rule among them is selected. We start by presenting a rule generation method based on maximal frequent class itemsets, and then we present a cost-sensitive rule selection method to avoid misclassification.

3.1. Rule Generation

Rule generation aims to find a set of representative rules that can accurately describe/summarize the training data ($\mathcal{D}_{seen}$). The quality of the rules generated is crucial for classification purposes. In this context, a rule $\mathcal{X} \xrightarrow{\theta} c$ has some desirable characteristics:

1. Frequency: A rule must occur a significant number of times in $\mathcal{D}_{seen}$, otherwise the rule will be probably useless in the test phase since the rule pattern (i.e., the antecedent $\mathcal{X}$) may not occur in $\mathcal{D}_{unseen}$ (i.e., the probability of occurrence of a rule pattern in $\mathcal{D}_{unseen}$ is related to its frequency in $\mathcal{D}_{seen}$, represented by $\sigma(\mathcal{X}, \mathcal{D}_{seen})$).
2. Statistical strength: A rule must have a high confidence value (i.e., θ), otherwise the application of the rule will probably incur in misclassification during the test phase.
3. Specificity: A rule must carry sufficient information about the specific data object being analyzed. The size of a rule can quantify how much information it carries. Very general rules may result in misclassification during the test phase.

A useful rule must have all the three characteristics. For example, rules with high statistical strength but low frequency are possibly due to noise in the training data, and they are not desirable rules. Algorithm 2, described in Section 2, can be used to find $\mathcal{F}(c, \sigma^{min}, \mathcal{D}_{seen})$, and consequently, to find the rule set $\mathcal{R}(\theta^{min}, \mathcal{F}(c, \sigma^{min}, \mathcal{D}_{seen}))$. If the parameters σ^{min} and θ^{min} are sufficiently high, the generated rules will have enough frequency and statistical strength (confidence). However, Algorithm 2 also generates too much general rules without sufficient specificity. This is due to the fact that Algorithm 2 generates small-size class association rules. We overcome this problem by making use of

the maximal frequent class itemsets, that is, our rule generation technique uses Algorithm 3 to find $\mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$, and this choice has two main advantages:

1. Rule specificity: The space for all frequent class itemsets is often huge. Practical computational constraints place severe limits on the subspace that can be explored, and consequently, small-size frequent class itemsets are generated. However, if the search is focused only on the maximal frequent class itemsets, more rules with longer sizes may be generated, and the generated rules will carry much more information about the data objects.
2. Condensed representation: Maximal frequent class rules are self-contained, that is they uniquely represent all frequent class rules, that is, smaller (yet informative enough) sub-rules can be generated from those maximal rules.

Thus, by using Algorithm 3 and finding $\mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$, our rule generation technique can generate more interesting rules with sufficient frequency, statistical strength and rule specificity.

3.2. Cost-Sensitive Rule Selection

Once the rule set $\mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$ is found, the test phase can begin. During this phase, each data object $\mathcal{O} \in \mathcal{D}_{unseen}$ is submitted for classification. For each data object submitted there is a best rule to be applied. Rule selection aims to find such best rule. For each submitted data object $\mathcal{O}$, our rule selection technique performs the following steps:

1. Step 1: search the longest rule $\mathcal{X} \xrightarrow{\theta} c \in \mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$, where $\mathcal{X} \subseteq \mathcal{O}$. Go to Step 2.
2. Step 2: if one or more rules with $\mathcal{X} \subseteq \mathcal{O}$ were found, choose the one with highest θ. Classify $\mathcal{O}$ as positive or negative, according to c (the consequent of the chosen rule) and then return.
3. Step 3: otherwise, if no rule with $\mathcal{X} \subseteq \mathcal{O}$ was found then break rules of size k into sub-rules of size $k-1$. If there is no more rules to be broken then classify $\mathcal{O}$ as negative (i.e., the default decision) and return, otherwise go to Step 2.

Sub-rules of the form $\mathcal{X} \xrightarrow{\theta} c$ (where $\mathcal{X} = \{x_1, x_2, ..., x_k\}$) are generated based on the k-intersection approach, that is, $\theta = \frac{|\mathcal{L}(x_1,\mathcal{D}_{seen}) \cap \mathcal{L}(x_2,\mathcal{D}_{seen}) \cap ... \cap \mathcal{L}(x_k,\mathcal{D}_{seen}) \cap \mathcal{L}(c,\mathcal{D}_{seen})|}{|\mathcal{L}(x_1,\mathcal{D}_{seen}) \cap \mathcal{L}(x_2,\mathcal{D}_{seen}) \cap ... \cap \mathcal{L}(x_k,\mathcal{D}_{seen})|}$. For this purpose, we maintain the tidsets of frequent 1-itemsets during the test phase. At the end of the test phase (all data objects in the test data were submitted for classification) four metrics will be obtained:

- TT: number of positive cases that were correctly classified as positive.
- TF: number of negative cases that were wrongly classified as positive.
- FT: number of positive cases that were wrongly classified as negative.
- FF: number of negative cases that were correctly classified as negative.

It is easy to see that $TT + TF + FT + FF = \mid \mathcal{D}_{unseen} \mid$, that is, the total number of data objects in $\mathcal{D}_{unseen}$. These metrics are used to evaluate the effectiveness of the classification.

The technique presented above tends to find long and strong rules. However, this technique is not cost-sensitive, that is, it assumes that classification errors have the same

cost. There are two types of misclassification: classify a positive case as negative, or classify a negative case as positive, respectively represented as TF and FT. Classification errors may have different costs, for example, in spam detection the majority of the users can tolerate a certain number of spams classified as legitimate messages, but they do not tolerate legitimate messages classified as spams. In this case, a TF error is much more costly than a FT error. In that way, we have three possibilities:

- Correctly classifying a data object (i.e., TT or FF) has cost 0.
- Wrongly classifying a positive data object as negative (i.e., FT) has cost 1.
- Wrongly classifying a negative data object as positive (i.e., TF) has cost λ.

The cost factor λ is specified during the test phase, and given two rules $\mathcal{X} \xrightarrow{\theta_p} (c = positive)$ and $\mathcal{Y} \xrightarrow{\theta_n} (c = negative)$, a data object is only classified as positive if $\theta_p \geq \lambda \times \theta_n$, otherwise it is classified as negative. Thus, we modified our rule selection technique, so that it can be cost-sensitive. Our cost-sensitive rule selection performs the following steps:

1. Step 1: search the longest rule $\mathcal{X} \xrightarrow{\theta_p} (c = positive) \in \mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$, where $\mathcal{X} \subseteq \mathcal{O}$. Go to Step 2.
2. Step 2: if one or more positive rules with $\mathcal{X} \subseteq \mathcal{O}$ were found, choose the one with highest θ_p. Go to Step 4.
3. Step 3: otherwise, if no positive rule with $\mathcal{X} \subseteq \mathcal{O}$ was found then break rules of size k into sub-rules of size $k-1$. If there is no more rules to be broken then go to Step 4, otherwise go to Step 2.
4. Step 4: search the longest rule $\mathcal{Y} \xrightarrow{\theta_n} (c = negative) \in \mathcal{R}(\theta^{min}, \mathcal{MF}(c, \sigma^{min}, \mathcal{D}_{seen}))$, where $\mathcal{Y} \subseteq \mathcal{O}$.
5. Step 5: if one or more negative rules with $\mathcal{Y} \subseteq \mathcal{O}$ were found, choose the one with highest θ_n. Go to Step 7.
6. Step 6: otherwise, if no negative rule with $\mathcal{Y} \subseteq \mathcal{O}$ was found then break rules of size k into sub-rules of size $k-1$. If there is no more rules to be broken then go to Step 7, otherwise go to Step 5.
7. Step 7: if there is no positive and no negative rules, classify $\mathcal{O}$ as negative and return. Otherwise, if there is no negative rules then $\theta_n = 1 - \theta_p$. Otherwise, if there is no positive rules then $\theta_p = 1 - \theta_n$. Go to Step 8.
8. Step 8: if $\theta_p \geq \lambda \times \theta_n$ then classify $\mathcal{O}$ as positive, otherwise classifify $\mathcal{O}$ as negative.

The basic idea behind our cost-sensitive rule selection is to first select the best positive and negative rules that match the appraised data object, and then quantify how much better (or worse) is the selected positive rule in relation to the selected negative one. Then, to be classified as positive, the appraised data object must have a positive rule λ times stronger (i.e., $\theta_p \geq \lambda \times \theta_n$) than the negative one.

4. Experimental Results

In this section we present experimental results obtained from the application of our classification techniques to two important real-world problems: spam detection and protein homology detection. We start with a description of the input parameters and evaluation metrics that will be used throughout the experiments. Then, for each application mentioned above, we will present the corresponding results.

4.1. Parameters and Evaluation Metrics

During our experiments we basically varied two input parameters: the minimum support (σ^{min}), and the error cost factor (λ). Values for both σ^{min} and λ are dependent of the application scenario. We have also defined some metrics with the goal of evaluating the effectiveness of the classification performed using our techniques:

1. Accuracy (Acc): It is the total number of correct classified cases (positive and negative) divided by the total number of cases, given by: $\frac{TT+FF}{TT+TF+FT+FF}$.
2. Precision (Prec): It is the number of correct classified positive cases divided by the total number of positive classified cases, given by $\frac{TT}{TT+TF}$.
3. Recall (Rec): It is the number of correct classified positive cases divided by the total number of real positive cases, given by $\frac{TT}{TT+FT}$.
4. Root Mean Squared Error (RMSE): It is the usual definition of squared error divided by the total number of cases to make the metric independent of dataset size. It is given by: $\sqrt{\frac{\sum_1^n (T-P)^2}{n}}$, where $n = TT + TF + FT + FF$, T is the class (0 or 1), and P is the confidence in the classification (values in $[0, 1]$).
5. Total Cost Ratio (TCR): Given a cost factor λ, TCR expresses how much times it is better using the classifier than not using it. Greater TCR values indicate better performance. For TCR < 1, not using the classifier is better. TCR is given by: $\frac{TT+FT}{\lambda \times TF+FT}$.

4.2. Spam Detection

Spams, or more formally unsolicited commercial electronic messages, are extremely annoying to most users, as they waste their time and extend dial-up connections. They also waste bandwidth, and often expose minors to unsuitable content by advertising pornographic sites. Without appropriate counter-measures spams can eventually undermine the usability of electronic messages. Technical counter-measures include the development of spam filters, which can automatically detect a spam message. Classification algorithms are usually the core of spam filters.

4.2.1. Dataset and Problem Descriptions

The problem is to classify if a given electronic message is spam or legitimate (a message is a data object). In our case, the classification is based only on the content of the message (i.e., the words and symbols that compose the message). The training data is composed of 2,893 messages (legitimate and spams). Spams are marked as positive cases, legitimate messages are negative cases. Spams correspond to 16% of the total number of messages [1]. There are both categorical and continuous attributes in this dataset. For a categorical attribute we assume that all possible values are mapped to a set of consecutive positive integers. For a continuous attribute we assume that its value ranges is discretized into 3 intervals, and then (similarly to a categorical attribute) the intervals are also mapped to consecutive positive integers. Spam and legitimate messages were randomly divided into ten equal sized sets of messages. Then, for each of the ten sets the remaining nine

[1] This dataset is called LINGSPAM. It is a set of e-mail messages from the linguist list, and it is largely accepted as the ultimate test collection for spam detection and allows for comparison between different approaches. It is publicly available at http://listserv.linguistlist.org/archives/linguist.html.

sets were used as training data before the set was classified. This is the common testing approach used for small datasets [Kohavi 1995]. The original dataset contains more than 65,000 items (different words or symbols), and therefore, we needed to apply an attribute selection procedure. We applied 3 different attribute selection procedures, based on the information gain of each attribute. We selected the 5%, 2% and 1% most significant attributes, resulting in 3 new different datasets, called LS5, LS2, and LS1.

4.2.2. Results

In this section we present results obtained from the utilization of our classification techniques for spam detection. Figures 3 and 4 show precision and recall numbers for the different datasets employed, LS5, LS2 and LS1. A general trend we observed is that recall values are reduced with increasing values of σ^{min} (because there are less rules for higher values of σ^{min}) and with increasing values of λ. Precision is not affected significantly by σ^{min} variations, but it decreases with lower values of λ. The best result overall (precision=0.97 / recall=0.92) was obtained with the LS1 dataset and employing σ^{min}=0.06 and λ=1. Figure 5 shows accuracy numbers. Clearly, the accuracy decreases as we increase λ. This is against expectations, since, as observed earlier, precision shows an opposite trend. However, the number of false-negatives (i.e., FT) increases significantly for higher values of λ, reducing the accuracy. This increase of false-positive cases does not affect precision (according to its definition). The best accuracy (0.985) was obtained with the LS1 dataset and employing σ^{min}=0.06 and λ=1. Figure 6 shows TCR values when varying σ^{min} and λ. We observed that TCR values are always reduced with increasing values of λ. We also observed that TCR values increase for lower values of σ^{min}, since, for these cases, higher values of precision are obtained.

Table 1 shows a comparison between the results obtained by our associative classification technique (T1), and the results obtained by two other techniques (T2 and T3) presented in [Sakkis et al. 2003]. This table shows three observed values (coming from T1, T2 and T3) for each of the analyzed metrics. The technique T2 employs a (memory-based) classification method based on automata, and the technique T3 is a traditional naïve-Bayes classifier. Our classification technique (T1) is almost always the best one, especially in terms of TCR, reaching a value of 9.36, that is, using our classification technique with σ^{min}=0.06 and λ=1, we achieved results almost 10 times better (in terms of cost) than not using any classification. This is the best result reported in the literature for the LINGSPAM benchmark.

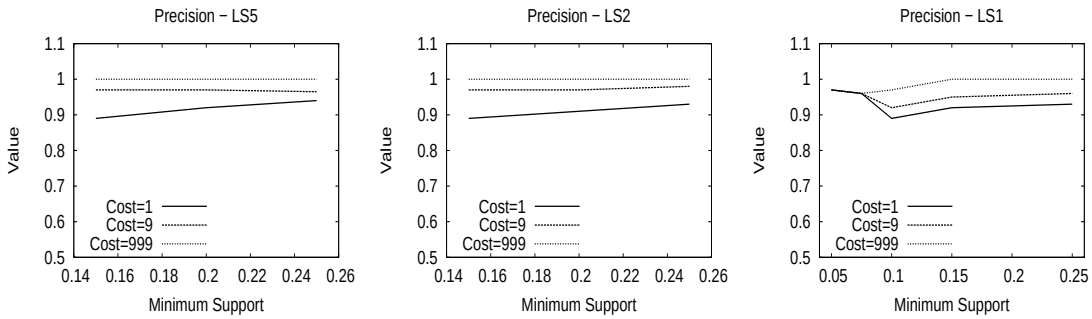

Figure 3. Precision Values for LS5, LS2 and LS1.

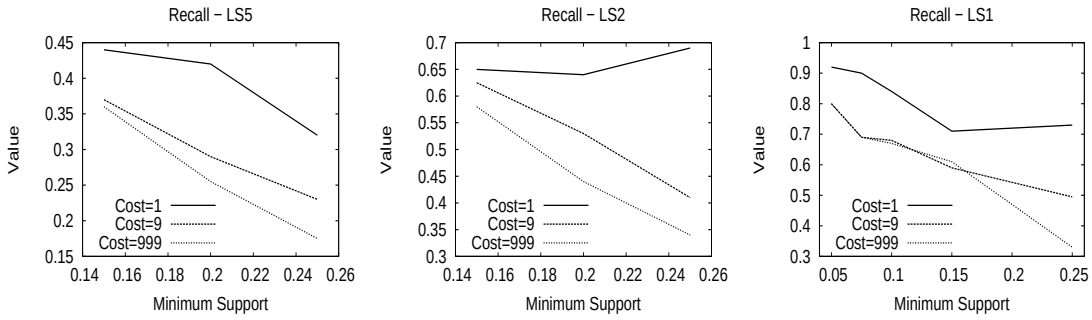

Figure 4. Recall Values for LS5**,** LS2 **and** LS1**.**

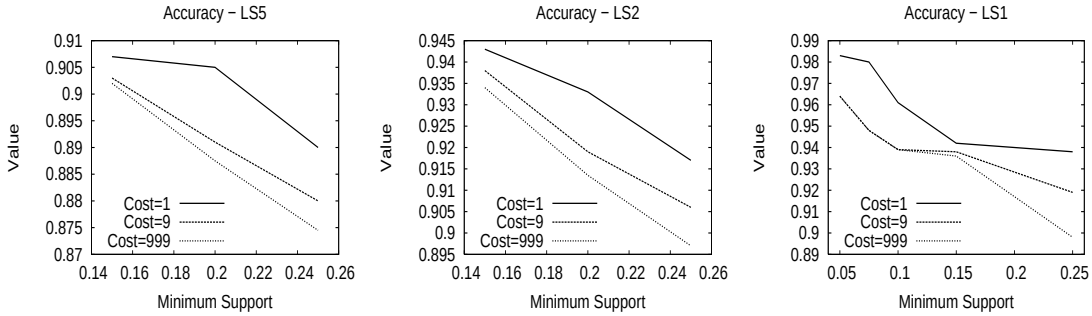

Figure 5. Accuracy Values for LS5**,** LS2 **and** LS1**.**

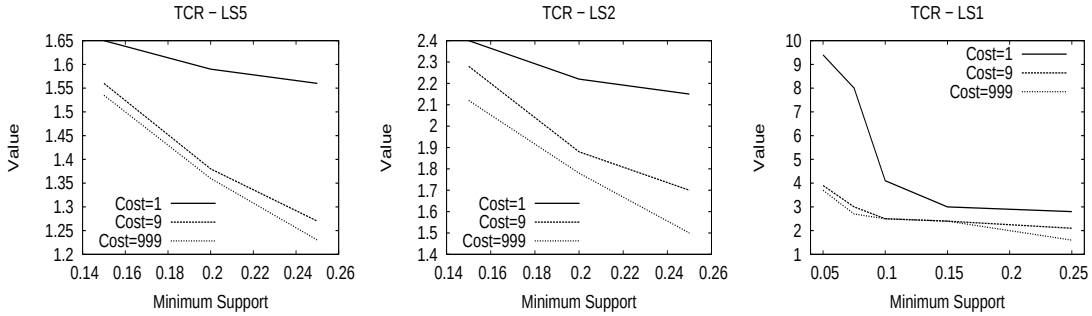

Figure 6. TCR Values for LS5**,** LS2 **and** LS1**.**

Table 1. Comparison against other Classification Techniques.

λ	Rec(T1)	Rec(T2)	Rec(T3)	Prec(T1)	Prec(T2)	Prec(T3)	TCR(T1)	TCR(T2)	TCR(T3)
1	**0.92**	0.89	0.82	0.97	0.97	**0.99**	**9.36**	7.18	5.41
9	**0.83**	0.82	0.78	**0.99**	**0.99**	**0.99**	**3.96**	3.64	3.82
999	**0.81**	0.60	0.64	0.99	**1.00**	**1.00**	**3.61**	2.49	2.86

4.3. Protein Homology Detection

Proteins play a fundamental role in disease; therefore they are of great interest to fields such as pharmacology and genomics. It is well known that proteins have a strong structure-function relationship, that is, the biochemical function that the protein performs is strongly related to its structure. This is the motivation of protein homology detection.

4.3.1. Dataset and Problem Descriptions

The problem is to classify if a protein is homologous to a given native sequence (the native sequence is a data object). The data is grouped into blocks around each native sequence. The training data is composed of 153 native sequences, and the test data is composed of 150 native sequences. For each native sequence, there is a set of approximately 1,000 protein sequences for which homology detection is needed. Homologous sequences are marked as positive cases, non-homologous sequences (also called "decoys") are negative cases [2]. The classification is based on 74 attributes that measure match [3].

There are both categorical and continuous attributes in this dataset. For a categorical attribute we assume that all possible values are mapped to a set of consecutive positive integers. For a continuous attribute we assume that its value ranges is discretized into 6 intervals, and then (similarly to a categorical attribute) the intervals are also mapped to consecutive positive integers.

4.3.2. Results

In this section we present results obtained from the utilization of our classification techniques for protein homology detection. Figure 7(a) and 7(b) shows precision and recall numbers obtained. As observed in the spam detection application, recall values decrease as we increase σ^{min} and λ. Differently from the observed in the spam detection application, precision is significantly affected by σ^{min} variations, and it decreases as we decrease λ. Figure 7(c) shows accuracy numbers. As occurred in the spam detection application, accuracy is reduced with increasing values of λ. The best result overall (precision=0.95 / recall=0.47) was obtained by employing σ^{min}=0.005 and λ=1.

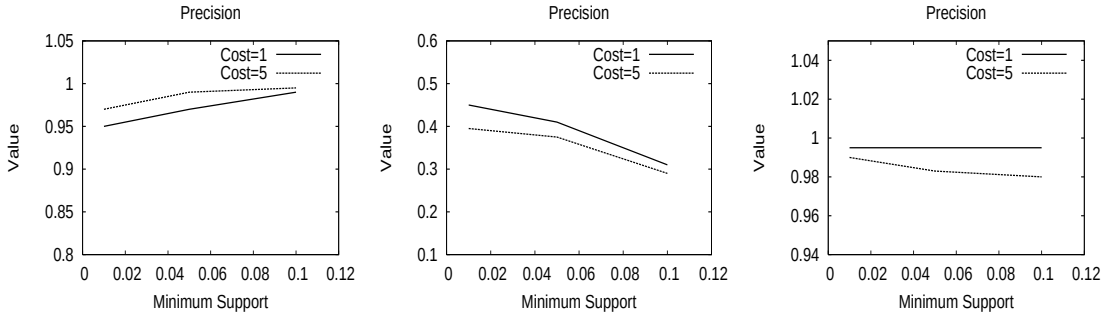

Figure 7. (a) Precision Values on the left, (b) Recall Values on the Middle, and (c) Accuracy Values on the Right.

Table 2 shows a comparison between the results obtained by our associative classification technique (T1), and the results obtained by the best 16 of the 102 algorithms

[2]This dataset was obtained from the KDD Cup 2004 task of protein homology detection. It is publicly available at http://kodiak.cs.cornell.edu/kddcup/datasets.html.

[3]If the reader is not familiar with protein matching and structure prediction, it might be helpful to think of this as a WWW search engine problem. There are 153 queries in the training data, and 150 queries in the test data. For each query there are about 1,000 returned documents, only a few of the documents are correct matches for each query. The goal is to classify which of the 1,000 documents best match the query.

that were submitted to the KDD Cup 2004 contest [Caruana et al. 2004]. In the table, the algorithms are ranked according to the observed RMSE value. Our technique (T1) is ranked at the eighth position, showing a RMSE value of 0.03832, when σ^{min}=0.005 and λ=1. Although our algorithm is not the best one, we believe that its performance was quite satisfactory, since all other algorithms listed in Table 2 were specifically developed to perform well in the provided dataset. Thus, we believe that our algorithm is more generic than those listed in the table.

Table 2. Comparison against other Classification Techniques.

Rank	Classifier	RMSE	Rank	Classifier	RMSE
1	CT.AC.CN	0.03501	9	Weka	0.03833
2	Rong Pan	0.03541	10	S575	0.03838
3	S587	0.03692	11	S541	0.03847
4	Mario Zille r	0.03766	12	S513	0.03848
5	MEDai/AI Insight	0.03805	13	S539	0.03850
6	S560	0.03826	14	S540	0.03850
7	S591	0.03830	15	PG445 UniDo *	0.03878
8	T1	0.03832	16	S561	0.03900

5. Conclusions and Future Work

In this paper we presented an algorithm for associative classification, which is a classification technique based on association rules. The novel components of the proposed algorithm are new techniques for rule generation and rule selection. Our rule generation technique is based on maximal frequent class rules, what reduces the complexity of the search for association rules, and make possible the generation of longer (and more informative) rules. Our rule selection technique is based on how informative and strong a rule is, and on a cost-sensitive mechanism that reduces misclassification. Experimental results on real-world applications indicates that the proposed techniques make our algorithm better than other generic algorithms (such as automata and naïve-Bayes based algorithms), and comparable to application-specific (and possibly dataset-specific) algorithms (such as the presented in the KDD Cup 2004 contest). As part of ongoing and future work we are evaluating the possibility of applying temporal information, such as sequential patterns [Zaki 2001], to improve the effectiveness of our algorithm.

References

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. of the Int. Conf. on Management of Data, SIGMOD*, pages 207–216, Washington, USA. ACM.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. of the Int. Conf. on Very Large Databases*, pages 487–499, SanTiago, Chile.

Baralis, E. and Chiusano, S. (2004). Essential classification rule sets. *ACM Trans. Database Systems*, 29(4):635–674.

Bayardo, R. (1998). Efficiently mining long patterns from databases. In *Proc. of the Int. Conf. on Management of Data, SIGMOD*. ACM.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classication and regression trees. *Wadsworth International Group*.

Caruana, R., Joachims, T., and Backstrom, L. (2004). KDD-Cup 2004: results and analysis. *SIGKDD Explor. Newsl.*, 6(2):95–108.

Domingos, P. and Pazzani, M. (2000). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130.

Gouda, K. and Zaki, M. (2001). Efficiently mining maximal frequent itemsets. In *Proc. of the Int. Conf. on Data Mining, ICDM*, pages 163–170, San Jose, USA. IEEE.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proc. of the Int. Conf. on Management of Data, SIGMOD*, pages 1–12. ACM.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Intl. Joint Conf. on Artificial Inteligence*, pages 223–228.

Li, W., Han, J., and Pei, J. (2001). CMAR: Efficient classification based on multiple class-association rules. In *Proc. of the Int. Conf. on Data Mining*, pages 369–376.

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. (1993). C4.5: Programs for machine learning. *Morgan Kaufmann*.

Sakkis, G., Androutsopoulos, I., and Paliouras, G. (2003). A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6:49–73.

Yin, X. and Han, J. (2003). CPAR: Classification based on predictive association rules. In *Proc. of the Int. Conf. on Data Mining, SDM*. SIAM.

Zaki, M. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1/2):31–60.

Zaki, M. and Hsiao, C. (2002). Charm: An efficient algorithm for closed itemset mining. In *Proc. of the Int. Conf. on Data Mining, SDM*, Arlington, USA. SIAM.

Zaki, M., Parthasarathy, S., Ogihara, M., and Li, W. (1997). New algorithms for fast discovery of association rules. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*, pages 283–290.