Eager, Lazy and Hybrid Algorithms for Multi-Criteria 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 map a data instance to its appropriate class (or label). In associative classification the mapping is done through an association rule with the consequent restricted to the class attribute. Eager associative classification algorithms build a single rule set during the training phase, and this rule set is used to classify all test instances. Lazy algorithms, however, do not build a rule set during the training phase, the rule set generation is delayed until a test instance is given. The choice between eager and lazy algorithms is not simple. Using a single rule set to perform all the predictions may not take advantage of specific characteristics of a test instance. On the other hand, building a specific rule set for each test instance may incur in excessive computational efforts. In this paper we propose new eager and lazy algorithms for associative classification. Also, we present heuristics and caching mechanisms to alleviate computational costs during lazy classification, and a new hybrid algorithm which combines characteristics of eager and lazy approaches. Finally, we also propose a multi-criteria rule selection technique which is motivated by the intuition that a single criterion cannot give a global picture of the rule (i.e., rules with high confidence but low support). We performed a systematic evaluation of the proposed approaches using real data from an important application: spam detection.*

1. Introduction

Given a set of data instances and their class labels (i.e., the training data), the task of classification is to find a function which maps a new data instance (i.e., a test instance) to its corresponding class. Eager classification uses the training data to build a single rule set that is used to classify all test instances. On the other hand, lazy classification uses the training data to build a specific rule set for each test instance, and therefore, decisions are based on the properties of the particular test instance being evaluated. While eager algorithms build a single classifier that is good (on average) for all predictions, lazy algorithms take advantage of particular characteristics of a given test instance, and this may reflect in an improvement of the classification accuracy. This improvement in accuracy comes with the cost of building several different rule sets, one for each test instance, making the classification process slower. There are two alternatives to speedup the process. First, one can employ heuristics to reduce the size of the rule set, focusing only on the most important rules. Another alternative is to reduce work replication by caching some

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

already processed rules. After a few instances have been classified, commonly used rules already exist, and their computations do not need to be repeated. In this paper we focus on associative classification, a well-known classification technique based on a special type of association rule, where the consequent is restricted to the class attribute. We propose eager and lazy associative classification algorithms, and after discussing the advantages and the drawbacks of each approach, we also propose an hybrid (eager/lazy) algorithm.

The accuracy of associative classification algorithms (whether they are eager or lazy) is intrinsically dependent on the criterion used to choose the best rule, that will be used to perform the classification. The problem is that, usually, a single criterion cannot give a global picture of the rules (i.e., rules with high confidence and low support), turning the algorithms prone to misclassification. In this paper we also deal with multi-criteria classification, where the best rule is the one that has the highest values in all the criteria used. We present different rule selection criteria, and then we propose a multi-criteria rule selection approach that starts the generation of the rule set by the most general rules and keep generating more specific rules until the best rule is found.

The paper is organized as follows. In Section 2 we present necessary background and related work on associative classification. In Section 3 we discuss in more detail eager and lazy approaches and present the corresponding algorithms. In Section 4 we show the experimental results. Finally, in Section 5 we draw the conclusions.

2. Preliminaries

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

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. An itemset of size k, $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ is called a k-itemset. We say that $\mathcal{X}$ contains $\mathcal{Y}$ if $\mathcal{Y} \subseteq \mathcal{X}$.

DEFINITION 2. [TRAINING AND TEST INSTANCES] A data instance $\mathcal{T}_i$ is an itemset, where i is a natural number called the instance identifier or tid. A dataset $\mathcal{D}$ is a finite set of instances, and it is divided into two partitions, $\mathcal{D} = \mathcal{D}_{seen} \cup \mathcal{D}_{unseen}$, where $\mathcal{D}_{seen}$ is the set of training instances (i.e., the training data) and $\mathcal{D}_{unseen}$ is the set of test instances (i.e., the test data). If $\mathcal{T}_i$ is followed by the class attribute c, then $\mathcal{T}_i \in \mathcal{D}_{seen}$, otherwise $\mathcal{T}_i \in \mathcal{D}_{unseen}$. The support of an itemset $\mathcal{X}$ is the fraction of training instances that contain $\mathcal{X}$, given as, $\sigma(\mathcal{X}) = \frac{|\{\mathcal{T}_i \in \mathcal{D}_{seen} | \mathcal{X} \subseteq \mathcal{T}_i\}|}{|\mathcal{D}_{seen}|}$. The itemset $\mathcal{X}$ may appear too frequently in some classes, and too rarely in others. Weighted support is the support of $\mathcal{X}$ normalized by the support of each class, and it is given by $\gamma(\mathcal{X}) = \frac{\sigma(\mathcal{X} \cup c)}{\sigma(c)}$.

DEFINITION 3. [ASSOCIATION RULES] An association rule is a rule with the form $\mathcal{X} \xrightarrow{\theta,\phi,\pi} c$, where c is the class attribute and $\mathcal{X}$ is an itemset ($c \notin \mathcal{X}$). The confidence of the rule, denoted as θ, is given by $\frac{\sigma(\mathcal{X} \cup c)}{\sigma(\mathcal{X})}$. The exclusiveness (or weighted confidence) of the rule, denoted as ϕ, is given by $\frac{\gamma(\mathcal{X} \cup c)}{\gamma(\mathcal{X} \cup c) + \gamma(\mathcal{X} \cup \overline{c})}$. The higher the exclusiveness, the more strongly $\mathcal{X}$ is associated to class c. The conviction of the rule, denoted as π, is given by $\frac{\sigma(\mathcal{X}) \times \sigma(\overline{c})}{\sigma(\mathcal{X} \cup \overline{c})}$ and measures implication. It is directional and it is maximal ($\pi = \infty$) for perfect implications, and it properly indicates when the rule does not hold more than expected ($\pi = 1$). A rule $\mathcal{X} \xrightarrow{\theta,\phi,\pi} c$ matches a data instance $\mathcal{T}_i$ if $\mathcal{X} \subseteq \mathcal{T}_i$.

RELATED WORK – There are several association rule mining algorithms in the literature. The most well known are APRIORI [Agrawal and Srikant 1994], FP-GROWTH [Han et al. 2000], and ECLAT [Zaki et al. 1997]. There are also several algorithms for classification, such as ID3 [Quinlan 1986], C4.5 [Quinlan 1993], and CART [Breiman et al. 1984], but they are based on decision trees or Bayesian rules. Differently, our work is related to associative classification, which was introduced 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). These algorithms are all based on eager classification, but there are also lazy classification algorithms [Friedman et al. 1996, Zheng and Webb 2000] which are all based on decision trees and Bayesian rules, and not on association rules. Our work is novel, first because the proposed algorithms (including the eager algorithm) use a novel multi-criteria for rule selection approach, and also because there are no lazy or hybrid algorithms for associative classification in the literature.

THE ECLAT ALGORITHM – Some ideas behind the ECLAT algorithm are specially important for us, and we will also use these ideas in our classification algorithms. We start by discussing search space partitioning using equivalence classes, which is very useful for sake of enumerating itemsets with a particular item (i.e., the class attribute). For example, to generate the rule $\mathcal{X} \rightarrow c$ we first need to generate the itemset $\mathcal{X} \cup c$. We can easily constrain the enumeration of itemsets to only supersets that contain one class, c, by using the equivalence class partitioning scheme [Zaki et al. 1997].

Another important characteristic of the ECLAT algorithm is the vertical dataset layout approach to perform support counting, which is more appropriate to our lazy rule generation approach. Each itemset $\mathcal{X}$ has a tidset, denoted as $\mathcal{L}(\mathcal{X})$, which is the set of $tids$ in $\mathcal{D}_{seen}$ in which $\mathcal{X}$ has occurred. Then $\sigma(\mathcal{X})$ is given as $\frac{|\mathcal{L}(\mathcal{X})|}{|\mathcal{D}_{seen}|}$. Basically, we can count the support of a k-itemset $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ in two different ways. The 2-way intersection uses the tidsets of two (k-1)-subsets of $\mathcal{X}$ (say $\mathcal{Y}$ and $\mathcal{Z}$) to calculate $\mathcal{L}(\mathcal{X})$, which is given as $\mathcal{L}(\mathcal{Y}) \cap \mathcal{L}(\mathcal{Z})$. In some cases, however, we only have access to the tidsets of the 1-itemsets, and in these cases, we use the k-way intersection approach to calculate $\sigma(\mathcal{X})$, by calculating $\mathcal{L}(\mathcal{X})$ as $\mathcal{L}(x_1) \cap \mathcal{L}(x_2) \cap \ldots \cap \mathcal{L}(x_k)$.

3. Algorithms for Multi-Criteria Associative Classification

In this section we describe our proposed algorithms. First, we will explain our multi-criteria rule selection approach, which is used in all the proposed algorithms. Then, for each algorithm we will explain how it deals with the training and test data.

MULTI-CRITERIA RULE SELECTION – Our multi-criteria rule selection approach is based on three criteria: θ, ϕ and π. The best rule is the one with the highest values in all these three criteria. The rule selection always starts with the most general rules (rules of size 2), and if no best rule is found (there is no rule with the highest values in all criteria), then more specific rules (rules of size 3) are analyzed. This process continues analyzing more specific rules until the best rule is found or the maximum rule size is reached (in order to avoid overfitting). If no best rule is found, the rule with the highest conviction value is selected (In our experiments, conviction seems to be the best single criterion).

EAGER ALGORITHM (EC) – The algorithm starts by generating a set of frequent itemsets [1] from $\mathcal{D}_{seen}$. The itemset generation follows the ECLAT algorithm execution, which is based on equivalence classes, where the search space is partitioned and itemsets with the form $\mathcal{X} \cup c_i$ are generated for each class $c \in \{positive, negative\}$ [2]. The support computation is based on the 2-way intersection approach. After all frequent itemsets are found, EC generates a rule set $\mathcal{R}$. Then, for each test instance $\mathcal{T}_i$, EC retrieves all matching rules from $\mathcal{R}$ (i.e., rule with the form $\mathcal{X} \xrightarrow{\theta,\phi,\pi} c$ where $\mathcal{X} \subseteq \mathcal{T}_i$). From the retrieved rules, the best rule is selected according to our multi-criteria approach, and a class is finally chosen for the prediction. If there is no matching rule in $\mathcal{R}$, a default prediction (i.e., the most frequent class in $\mathcal{D}_{seen}$) is performed.

LAZY ALGORITHM (LC) – The algorithm starts by constructing the tidset of each item in $\mathcal{D}_{seen}$ (including the classes, which are treated as frequent items). Until this moment no rule has been generated. Now, for each test instance $\mathcal{T}_i = \{t_1, t_2, \ldots, t_n\}$, LC generates a rule set $\mathcal{R}$. To reduce the amount of computation for generating the rule set, the items $\{t_1, t_2, \ldots, t_n\} \in \mathcal{T}_i$ are sorted in according to their support values (in descending order), and the first v (where v is specified by the user) most frequent items are maintained. The algorithm keeps performing combinations between the v most frequent items and generating rules such as $t_1 \xrightarrow{\theta,\phi,\pi} c$, $t_2 \xrightarrow{\theta,\phi,\pi} c$ and $\{t_1 \cup t_2\} \xrightarrow{\theta,\phi,\pi} c$. If v is much smaller than n, few rules will be generated and the quality of the rule set will be reduced, since the amount of information about the test instance will be smaller. The support computation is based on the k-way intersection approach. Different test instances have different rule sets, but different rule sets may share common association rules. To avoid generating the same rule more than once when processing different test instances, LC uses a cache whose key is given by the items that compose the rule and the content is composed by θ, ϕ and π. From the generated rule set, the best rule is selected according to our multi-criteria approach, and class is finally chosen for the prediction. All rules generated match the test instance because the rules were generated based on the items that appear in the test instance, and so there is no need for a default prediction.

HYBRID ALGORITHM (HC) – As mentioned earlier, the EC algorithm suffers from the missing rule (or rule set coverage) problem when there is no matching rule in the rule set. In this case the solution adopted by EC is to use a default prediction. However, this solution is too restrictive and lacks diversity, since choosing a pre-determined class does not take into account any characteristic of the test instance. The HC algorithm works similarly to the EC algorithm, the only difference is that HC generates a customized rule set for each test instance that is not covered by the original rule set.

In order to evaluate our algorithms, after all test instances are classified four measures are 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.

[1] An itemset $\mathcal{X}$ is frequent if $\sigma(\mathcal{X}) \geq \sigma^{min}$, where σ^{min} is a threshold specified by the user.
[2] In this paper we focus on binary classification.

4. Experimental Results

In this section we present experimental results obtained from the application of our classification algorithms to a real-world problem: spam detection. We start with a description of the input parameters and evaluation metrics that will be used throughout the experiments. All experiments were run on a PENTIUM III 1GHz 512MB main memory.

During our experiments we varied the following input parameters: the minimum support (when analyzing the EC and HC algorithms), the cache size and the maximum number of items in a test instance (when analyzing the LC algorithm). We have also varied the size of the training data and the size of the test data. Some metrics were also defined with the goal of evaluating the effectiveness of the classification performed using our algorithms:

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. Improvement (Imp): It quantifies how much it is better using the classifier than not using it. Greater improvement values indicate better performance. For Imp $<$ 1, not using the classifier is better. Improvement is given by: $\frac{TT+FT}{TF+FT}$.

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.

DATASET AND PROBLEM DESCRIPTIONS – The problem is to classify if a given electronic message is spam or legitimate (a message is a data instance). In our case, the classification is based only on the content of the message (i.e., the words and symbols that compose the message). Spams are marked as positive cases, legitimate messages are negative cases. Spams correspond to 25% of the total number of messages [3]. All attributes are continuous (i.e., the same word/symbol may appear more than once in the message), and for each attribute we assume that its value ranges is discretized into 3 intervals, and then the intervals are mapped to consecutive positive integers.

RESULTS – We start the evaluation by comparing the EC and HC algorithms. Figure 1(a) shows precision, recall and accuracy numbers for these two algorithms. The hybrid algorithm performs better in all three metrics. This result shows that using a lazy procedure is better than using a pre-specified class for default prediction. Figure 1(b) shows the impact of the lazy procedure in the total execution time, for different training and testing configurations. As expected, EC is slightly faster than HC. Figure 1(c) shows

[3]This is the set of e-mail messages from the SPAMASSASSIN project, and it is largely accepted as the ultimate test collection for spam detection benchmark. It is publicly available at http://spamassassin.apache.org/publiccorpus/

how much of improvement the algorithm can achieve by second. The hybrid algorithm needs less time to achieve more improvement and performs better than the eager one.

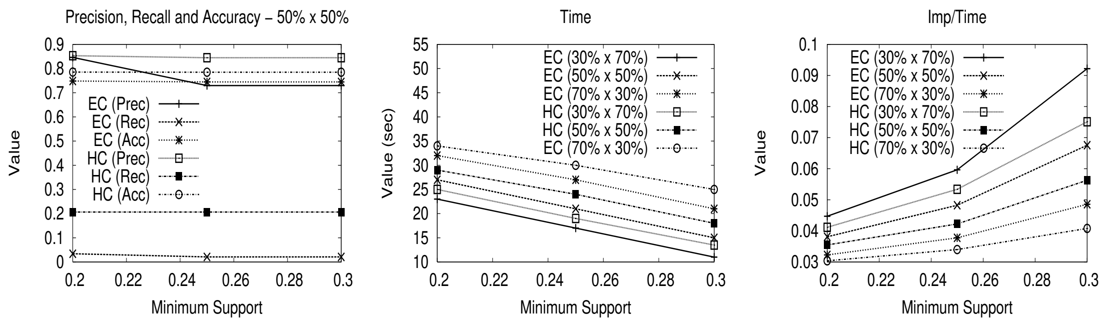

Figure 1. (a) Prec/Rec/Acc for EC and HC Algorithms, (b) Execution Time and (c) Improvement per second for EC and HC using Different Training Configurations.

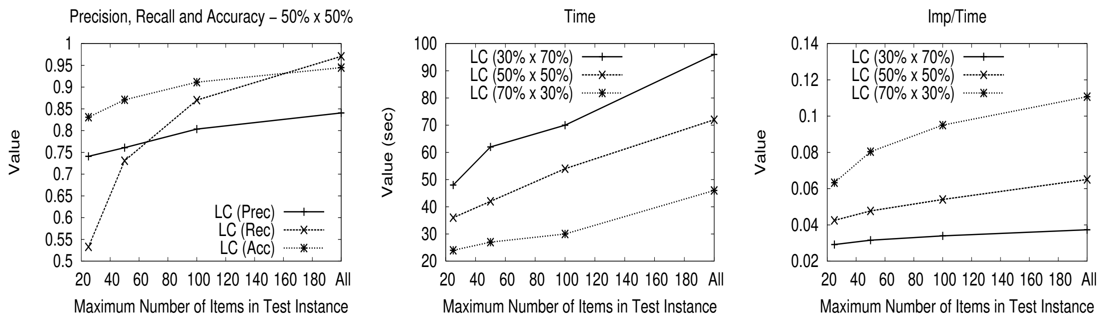

Figure 2. (a) Prec/Rec/Acc for LC Algorithm. (b) Execution Time and (c) Improvement per second, for LC using Different Training Configurations.

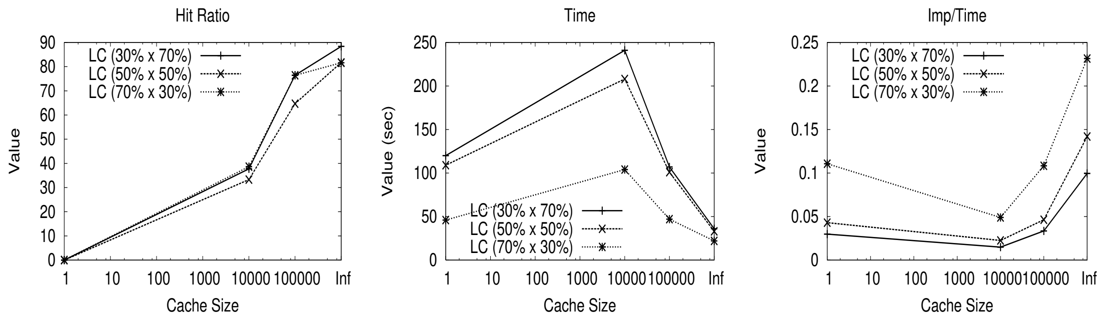

Figure 3. (a) Cache Effectiveness, (b) Execution Time, and (c) Improvement per second for LC, using Different Training Configurations.

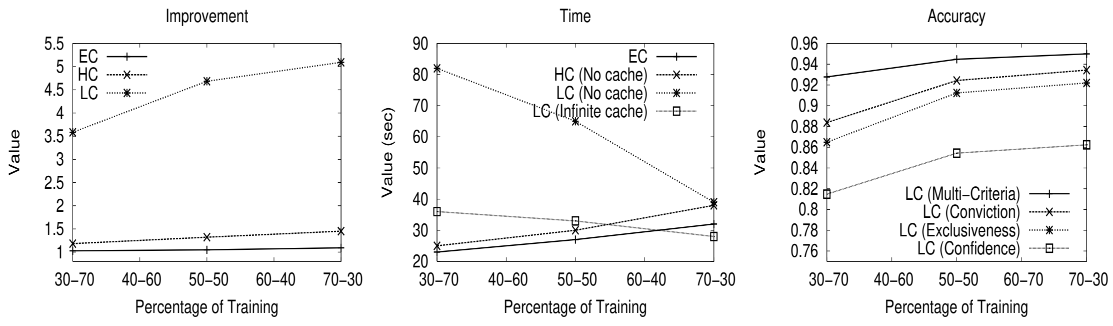

Figure 4. (a) Improvement and (b) Execution Time for the Different Algorithms. (c) Comparing Multi-Criteria against Single Criterion Approaches.

Next we analyze our proposed heuristic based on limiting the number of items in a given test instance. Figure 2(a) shows precision, recall and accuracy numbers by varying the maximum number of items in the test instances. All these numbers increase by increasing the number of items in the test instance, and this is expected since more information are available with more items being analyzed. Figure 2(b) shows the total time spent by varying the maximum number of items, and clearly the time increases linearly with the number of items. Also, we can observe that the time spent by LC increases with the number of test instances (i.e., 30% $\times$ 70%). Figure 3(c) the Improvement/Time trade-off for the LC algorithm and, by comparing with Figure 1(c), we can observe that LC generally performs better than both EC and HC algorithms.

Now we turn the attention to the analysis of the cache effectiveness. Using the cache only affects the execution time of LC, and it does not affect precision, recall, accuracy, and improvement. Figure 3(a) shows the hit ratio. The ratio goes from 0 to 90% by varying the cache size from 0 to ∞. Also from this figure we observed that the cache is more effective for larger training configurations (i.e., 70% $\times$ 30%). Figure 3(b) shows the impact of caching in execution time. From this figure we can see that larger caches are generally better, except for small (but greater than 0) cache sizes, because in these cases, there is a large overhead of cache replacement (since the replacement procedure is called many times). Figure 3(c) shows the Improvement/Time trade-off. We can compare this result with the ones depicted in Figures 1(c) and 2(c), and we can observe that caching is the best mechanism, since caching does not reduces the improvement.

Our last analysis compares all algorithms by varying training and testing configurations. Figure 4(a) shows improvements for the different algorithms. As we can see, the improvement increases with the percentage of training for all algorithms. LC is the one with highest improvement, showing that our lazy algorithm is the best one. Figure 4(b) shows the execution time for all algorithms. The time spent by EC and HC algorithms increases with the percentage of training instances, since the amount of data that has to be processed in order to generate the rule set also increases. Differently, the time spent by LC diminishes with the percentage of training instances, since (consequently) the amount of test instances is reduced (and less rule set are generated). Figure 4(c) shows the comparison between EC, HC and LC, in terms of accuracy numbers by applying different rule selection approaches. We compared out multi-criteria approach against single-criterion ones, for instance highest confidence, highest conviction and highest exclusiveness. Clearly, our multi-criteria approach performs much better than all other rule selection approach. The best single criterion used is conviction, since this metric quantifies both the negative and positive aspects of the rule (and not only the positive one, such as confidence does).

5. Conclusions and Future Work

In this paper we investigated techniques for rule generation and rule selection in associative classification. We propose three algorithms with different rule generation characteristics. EC is an eager algorithm, which generates a single rule set before any test instance is given. LC is a lazy algorithm which generates one rule set for each test instance. HC is a hybrid algorithm which combines characteristics from both EC and LC algorithms. We evaluated the three algorithms, and concluded that LC shows the best precision, recall and accuracy numbers. Also, LC demands more computation due to the fact that it has to generate a possibly large number of rule sets. We propose two techniques to alleviate this problem: one heuristic and a simple caching mechanism. Our heuristic is based on limiting the size of the test instances (i.e., it analyzes only the k most frequent items on a test instance), and consequently, the heuristic also affects the effectiveness of the algorithm. Our caching mechanism is simple, but extremely efficient, and makes LC faster than both EC and HC. Our rule selection approach is based on the observation that one single criterion is not sufficient to give the global picture (i.e., rules with high confidence and low support). We propose to use a multi-criteria approach based on three metrics: confidence, exclusiveness and conviction. The best rule is the one with highest value in all these three metrics. The search for the best rule starts with more general rules and keeps generating more specific ones. The multi-criteria approach performs better than all single-criterion ones.

We are currently investigating associative classification techniques that are more robust regarding changes in the underlying processes that generate the data (i.e., concept drifts and shifts). For example, in the spam detection application, the spammer periodically changes the spam generation procedure in order to escape from the spam detector. Our current work involves techniques that can capture trends and other properties in the rule set, and use these properties to make the classifier more robust.

References

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.

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

Friedman, J., Kohavi, R., and Yun, Y. (1996). Lazy decision trees. In *Proc of the Nat. Conf. on Artificial Intelligence*, pages 717–724, Menlo Park, California. AAAI Press.

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.

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*.

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., 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.

Zheng, Z. and Webb, G. (2000). Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84.