Lazy Associative Classification*

Adriano Veloso[a], Wagner Meira Jr.[a], Mohammed J. Zaki[b]
[a] Computer Science Dept, Federal University of Minas Gerais, Brazil
[b] Computer Science Dept, Rensselaer Polytechnic Institute, Troy, USA
{adrianov,meira}@dcc.ufmg.br, zaki@cs.rpi.edu

Abstract

Decision tree classifiers perform a greedy search for rules by heuristically selecting the most promising features. Such greedy (local) search may discard important rules. Associative classifiers, on the other hand, perform a global search for rules satisfying some quality constraints (i.e., minimum support). This global search, however, may generate a large number of rules. Further, many of these rules may be useless during classification, and worst, important rules may never be mined. Lazy (non-eager) associative classification overcomes this problem by focusing on the features of the given test instance, increasing the chance of generating more rules that are useful for classifying the test instance. In this paper we assess the performance of lazy associative classification. First we demonstrate that an associative classifier performs no worse than the corresponding decision tree classifier. Also we demonstrate that lazy classifiers outperform the corresponding eager ones. Our claims are empirically confirmed by an extensive set of experimental results. We show that our proposed lazy associative classifier is responsible for an error rate reduction of approximately 10% when compared against its eager counterpart, and for a reduction of 20% when compared against a decision tree classifier. A simple caching mechanism makes lazy associative classification fast, and thus improvements in the execution time are also observed.

1 Introduction

The classification problem is defined as follows. We have an input data set called the *training data* which consists of a set of multi-attribute records along with a special variable called the *class*. This class variable draws its value from a discrete set of classes. The training data is used to construct a model which relates the feature variables (or attribute values) in the training data to the class variable. The *test instances* for the classification problem consist of a set of records for which only the feature variables are known while the class value is unknown. The training model is used to predict the class variable for such test instances.

Classification is a well-studied problem (see [12, 20] for excellent overviews) and several models have been proposed over the years, which include neural networks [17], statistical models like linear/quadratic discriminants [14], decision trees [2, 19], and genetic algorithms [11]. Among these models, decision trees are particularly suited for data mining. Decision trees can be constructed relatively fast compared to other methods. Another advantage is that decision tree models are simple and easy to understand [19].

As an alternative to decision trees, associative classifiers have been proposed [8,16,18]. These methods first mine association rules from the training data, and then build a classifier using these rules. This classifier produces good results and yields improved accuracy over decision trees [18].

Decision trees perform a greedy search for rules by heuristically selecting the most promising features. They start with an empty concept description, and gradually add restrictions to it until there is not enough evidence to continue, or perfect discrimination is achieved. Such greedy (local) search may prune important rules. Associative classifiers, on the other hand, perform a global search for rules satisfying some quality constraints. This global search, however, may generate a large number of rules, and many of the generated rules may be useless during classification (i.e., they are not used to classify any test instance).

In this paper we propose a novel lazy associative classifier, in which the computation is performed on a demand-driven basis. We place our associative classifier within an information gain framework that allows us to compare it to decision tree classifiers. Our method can overcome the large rule-set problem of traditional (eager) associative classifiers, by focusing on the features that actually occur within the test instance while generating the rules. We show that the proposed lazy classifier outperforms its eager counterpart, since in the lazy approach only the "useful" portion of the training data is mined for generating the rules ap-

*This research was sponsored by UOL (www.uol.com.br) through its UOL Bolsa Pesquisa program, process number 20060519184000a.

plicable to the test instance. Due to this local focus, the lazy classifier can better classify a test instance, for which a global, eager rule-set may not work that well. Simple caching mechanisms are used to avoid work replication during lazy associative classification. First we demonstrate that associative classifiers perform no worse than decision tree classifiers. Then we show that lazy classifiers outperform the corresponding eager classifiers. Our claims are empirically confirmed by an extensive set of experimental results. Timings are also showed in order to evaluate different classifiers with respect to computational complexity.

2 Related Work

Most existing work on associative classification relies on developing new algorithms to improve the overall classification accuracy. CBA [18] generates a single rule-set and ranks the rules according to their confidence/support values. Then it selects the best rule to be applied to each test instance. Enhancements to CBA were proposed in [8, 16, 21, 23]. HARMONY [21] uses an instance-centric rule-generation approach in the sense that it assures the inclusion of at least one rule for each training instance in the final rule-set. CMAR [16] uses multiple rules (instead of a single best one) to perform the classification. CPAR [23] adopts a greedy technique to generate a smaller rule-sets. CAEP [8] explores the concept of *emerging patterns*, and, as a result, it usually predicts accurately all classes, even if their populations are unbalanced. It has been empirically shown that these associative classifiers usually perform better than decision trees. However, there is a lack of studies showing the theoretical implications of associative classifiers. Therefore, there is little intuition regarding the actual reasons behind the better performance of associative classifiers when compared to decision trees.

Rule induction classifiers, such as RISE [6], RIPPER [3], and SLIPPER [4], use greedy heuristics which are driven by global metrics. RISE performs a complete overfiting by considering each instance as a rule, and then it generalizes the rules. RIPPER and SLIPPER extend the "overfit and prune" paradigm, that is, they start with a large rule-set and prune it using several heuristics. Further, the SLIPPER algorithm also associates a probability with each rule, weighting the contribution of the rule during classification.

Most work on lazy classification [1] was based on nearest neighbor algorithms [5]. The problem of *small disjuncts* was first noted in [13], where it was showed that existing classifiers create models that are good for large disjuncts but are far from ideal for small disjuncts (which correctly classifies only few training instances). It is hard to assess the accuracy of small disjuncts because they cover few instances, yet removing all of them is unjustified since many of them may be significant and the overall accuracy would degrade.

Play	Outlook	Temperature	Humidity	Windy
yes	rainy	cool	normal	false
no	rainy	cool	normal	true
yes	overcast	hot	high	false
no	sunny	mild	high	false
yes	rainy	cool	normal	false
yes	sunny	cool	normal	false
yes	rainy	cool	normal	false
yes	sunny	hot	normal	false
yes	overcast	mild	high	true
no	sunny	mild	high	true
?(yes)	sunny	cool	high	false

Figure 1. Training and Test Instances.

A lazy decision tree was proposed in [10] and it was shown that the lazy approach is superior than the corresponding eager one (i.e., C4.5). Despite all the improvements obtained by using lazy algorithms, we are not aware of any proposals of lazy associative classification algorithms, as well as an assessment that demonstrates why they perform better than both decision trees and eager associative classifiers.

3 Eager Associative Classifiers

In this section we describe eager associative classifiers, and demonstrate why they perform better than decision trees. We start by discussing how decision rules may be generated from decision trees. Then we describe associative classifiers that are based on information gain, so that we may compare them regarding the rules that are generated by each approach.

3.1 Decision Trees and Decision Rules

Given any subset of training instances S, let s_i denote the number of instances with class c_i, and let $|S| = \sum_i s_i$ be the total number of training instances. Then $p_i = \frac{s_i}{|S|}$ denotes the probability of class c_i in S. The entropy of S is then given as $E(S) = \sum_i p_i \log p_i$. For any partition of S into m subsets S_i, with $S = \cup_{i=1}^{m} S_i$, the resulting split entropy is given as $E(\{S_i\}) = \sum_{i=1}^{m} \frac{|S_i|}{|S|} E(S_i)$. The information gain for the split is then given as $I(S, \{S_i\}) = E(S) - E(\{S_i\})$.

A decision tree is built using a greedy, recursive splitting strategy, where the best split is chosen at each internal node according to the information gain criterion. The splitting at a node stops when all instances are from a single class or if the size of the node falls below a minimum support threshold, called $minsup$. Figure 1 shows an example of training data, and Figure 2 shows the corresponding decision tree.

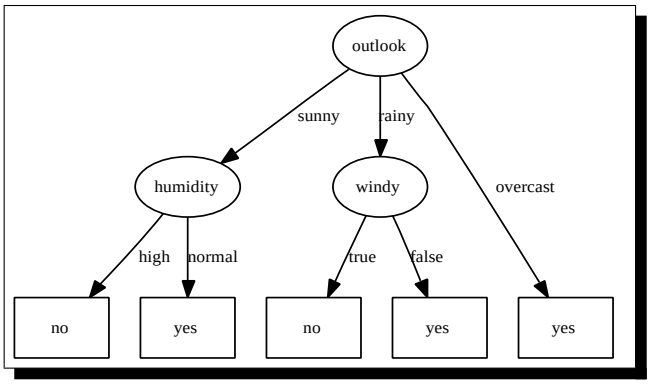

Figure 2. Decision Tree Classifier.

The last row of the table in Figure 1 shows one test instance which is recognized by the decision tree in Figure 2. The decision tree can be considered as a set of disjoint decision rules, with one rule per leaf. In that way, a decision tree can be simulated by a set of decision rules. In this case, the information gain for each decision rule is calculated in the same way as it is calculated for each path of the decision tree. Thus, a decision rule has the same value of information gain of its corresponding path in the decision tree[1].

3.2 Entropy-based Associative Classifier

We denote as *class association rules (CARs)* [18] those association rules of the form $\mathcal{X} \rightarrow c$, where the antecedent ($\mathcal{X}$) is composed of feature variables and the consequent (c) is just a class. CARs may be generated by a slightly modified association rule mining algorithm. Each itemset must contain a class and the rule generation also follows a template in which the consequent is just a class. CARs are essentially decision rules, and as in the case of decision trees, CARs are ranked in decreasing order of information gain. Finally, during the testing phase, the associative classifier simply checks whether each CAR matches the test instance; the class associated with the first match is chosen. Note that, seen in the light of CARs, a decision tree is simply a greedy search for CARs, using a level-wise search algorithm, that only expands the current best rule with other features. On the other hand, an eager associative classifier mines *all* possible CARs with a given $minsup$. It is also interesting to note that sorting the final rule-set on information gain, and using the best CAR for classification, is also a greedy strategy. While the greedy approach has its limitations, eager associative classifiers are not limited by the *prefix problem* of decision rules, that is, once the best feature is chosen at each node, all nodes under that subtree must contain it.

[1] A single decision rule may correspond to multiple paths in the decision tree, depending on the order in which the items in the rule are considered.

let $\mathcal{D}$ be the set of all n training instances
let $\mathcal{T}$ be the set of all m test instances
1. let $\mathcal{C}^e$ be the set of all rules $\{\mathcal{X} \rightarrow c\}$ mined from $\mathcal{D}$
2. sort $\mathcal{C}^e$ according to information gain
3. for each $t_i \in \mathcal{T}$ do
4. pick the first rule $\{\mathcal{X} \rightarrow c\} \in \mathcal{C}^e | \mathcal{X} \subseteq t_i$
5. predict class c

Figure 3. Eager Associative Classifier.

Figure 3 shows the basic steps of the eager associative classifier. In the initial step, the algorithm mines all frequent CARs, and sort them in descending order of information gain. Then, for each test instance t_i, the first CAR matching t_i is used to predict the class. Figure 4 shows an associative classifier built from our example set of training instances in Figure 1, using the algorithm showed in Figure 3. Three CARs match the test instance of our example (last row of Table 1):
1. {windy=false and temperature=cool→play=yes}
2. {outlook=sunny and humidity=high→play=no}
3. {outlook=sunny and temperature=cool→play=yes}

Rule {windy=false and temperature=cool→ play=yes} would be selected, since it is the best ranked CAR. By applying this CAR, the test instance will be correctly classified. Intuitively, associative classifiers perform better than decision trees because associative classifiers allow several CARs to cover the same partition of the training data. In our example, the test case is recognized by only one rule in the decision tree, while the same test case is recognized by three CARs in the associative classifier. Selecting the proper CAR to be applied is an issue in associative classification.

Next we present a theoretical discussion about the performance of decision trees and eager associative classifiers.

Theorem 1 *The rules derived from a decision tree are a subset of the CARs mined using an eager associative classifier based on information gain.*

Proof 1 *Let $maxE$ be the maximum entropy of all decision tree rules. Select a set $\mathcal{C}^e$ from all CARs such that their entropy is at most $maxE$. It is clear that the decision tree rules are a subset of $\mathcal{C}^e$.* □

Theorem 1 states that, for a given $minsup$, CARs contain (at least) all information of the corresponding decision tree. Since each decision tree rule may be seem as a CAR, and since all possible CARs were enumerated, then the decision tree can be built by choosing the proper CARs.

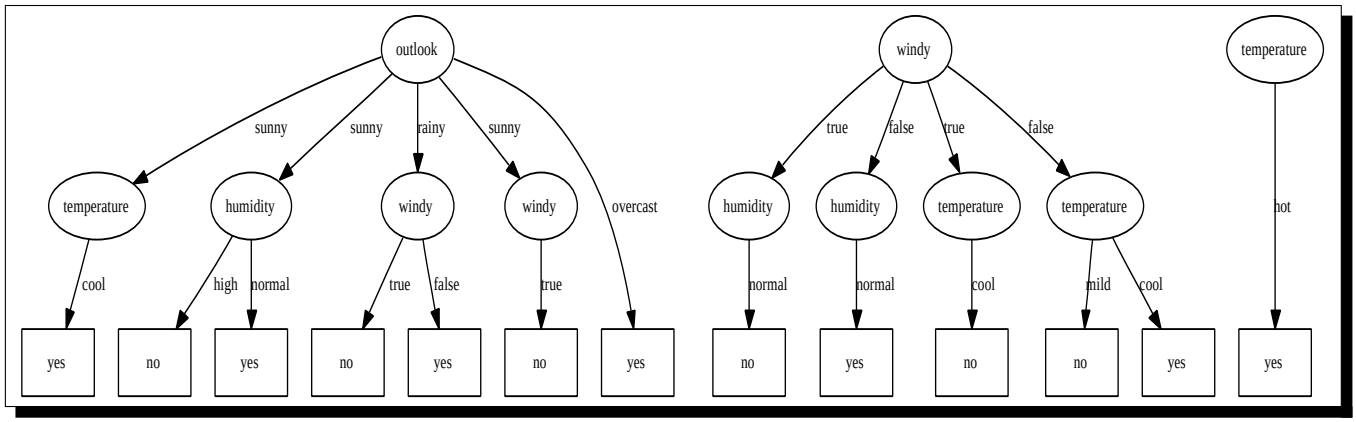

Figure 4. Associative Classifier

Theorem 2 *CARs perform no worse than decision tree rules, according to the information gain principle.*

Proof 2 *Given an instance to be classified, and, without loss of generality, a decision tree with just pure leaves, the decision tree predicts class c for that instance. We analyze two scenarios: first, just one CAR matches the instance, and second, more than one CAR matches. When just one CAR matches, it is the same as the decision tree rule, since the set of CARs subsumes the set of decision rules. In this case, the associative classifier and the decision tree make the same prediction. When more than one CAR matches an instance, the prediction may be either the same class (say c) as the matching decision rule or another class. If the associative classifier predicts c then the two approaches are equivalent. In case a class other than c is predicted, by definition, the best matching CAR provides a better information gain than the decision rule, and thus, according to the information gain principle, the CAR will make a better prediction.* □

Theorem 2 states that the additional CARs of the associative classifier that are not in the decision tree, cannot degrade the classification accuracy. This is because an additional CAR is only used if it is better than all decision rules (according to the information gain principle).

However, eager associative classifiers generate a large number of CARs, most of which are useless during classification. For instance, from the set of 13 CARs showed in Figure 4, only 3 match the test instance (the remaining 10 CARs are useless). Next, we present a lazy classifier and compare it to the eager version described in this section.

4 Lazy Associative Classifier

Unlike the eager associative classifier that extracts a set of ranked CARs from the training data, the lazy associative classifier induces CARs specific to each test instance. The lazy approach *projects* the training data, $\mathcal{D}$, only on those features in the test instance, $\mathcal{A}$. From this projected training data, $\mathcal{D}_\mathcal{A}$, the CARs are induced and ranked, and the best CAR is used. From the set of all training instances, $\mathcal{D}$, only the instances sharing at least one feature with the test instance $\mathcal{A}$ are used to form $\mathcal{D}_\mathcal{A}$. Then, a rule-set $\mathcal{C}^l_\mathcal{A}$ is generated from $\mathcal{D}_\mathcal{A}$. Since $\mathcal{D}_\mathcal{A}$ contains only features in $\mathcal{A}$, all CARs generated from $\mathcal{D}_\mathcal{A}$ must match $\mathcal{A}$. The lazy associative classifier is presented in Figure 5.

let $\mathcal{D}$ be the set of all n training instances
let $\mathcal{T}$ be the set of all m test instances
1. for each $t_i \in \mathcal{T}$ do
2. let $\mathcal{D}_{t_i}$ be the projection of $\mathcal{D}$ on features only from t_i
3. let $\mathcal{C}^l_{t_i}$ be the set of all rules $\{\mathcal{X} \rightarrow c\}$ mined from $\mathcal{D}_{t_i}$
4. sort $\mathcal{C}^l_{t_i}$ according to information gain
5. pick the first rule $\{\mathcal{X} \rightarrow c\} \in \mathcal{C}^l_{t_i}$, and predict class c

Figure 5. Lazy Associative Classifier

Now we demonstrate that the lazy associative classifier produces better results than its eager counterpart. Given a test instance $\mathcal{A}$, and a set of CARs $\mathcal{C}$, we denote by $\mathcal{C}_\mathcal{A}$ those CARs $\{\mathcal{X} \rightarrow c\}$ in $\mathcal{C}$ where $\mathcal{X} \subseteq \mathcal{A}$.

Theorem 3 *Let $\mathcal{A}$ be the set of features in a given test instance. Let $\mathcal{C}^e_\mathcal{A}$ be the set of CARs obtained from the eager associative classifier induced by $\mathcal{A}$, and $\mathcal{C}^l_\mathcal{A}$ be the set of CARs obtained from the lazy associative classifier induced by $\mathcal{A}$. For a given $minsup$, we have $\mathcal{C}^e_\mathcal{A} \subseteq \mathcal{C}^l_\mathcal{A}$.*

Proof 3 *By definition, both $\mathcal{C}^e_\mathcal{A}$ and $\mathcal{C}^l_\mathcal{A}$ are composed of CARs $\{\mathcal{X} \rightarrow c\}$ in which $\mathcal{X} \subseteq \mathcal{A}$, that is, all CARs contain only features in $\mathcal{A}$. Also the training instances matching $\mathcal{A}$ (i.e., the projected training data) are a subset of the set of*

Play	Outlook	Temperature	Humidity	Windy
no	–	–	–	true
yes	overcast	hot	–	–
yes	–	hot	–	–
yes	overcast	–	–	true
no	–	–	–	true
?(yes)	overcast	hot	low	true

Figure 6. Projected Training Data.

all training instances (i.e., $\mathcal{D}_\mathcal{A} \subseteq \mathcal{D}$). Thus, for a given $minsup$, if a rule $\{\mathcal{X} \rightarrow c\}$ is frequent in $\mathcal{D}$, then it must also be frequent in $\mathcal{D}_\mathcal{A}$. Since $\mathcal{C}^l_\mathcal{A}$ is generated from $\mathcal{D}_\mathcal{A}$ and $\mathcal{C}^e_\mathcal{A}$ is generated from $\mathcal{D}$ (and $\mathcal{D}_\mathcal{A} \subseteq \mathcal{D}$), $\mathcal{C}^e_\mathcal{A} \subseteq \mathcal{C}^l_\mathcal{A}$. □

The next example illustrates Theorem 3. Figure 6 shows the training data given in Figure 1 (i.e., $\mathcal{D}$), projected by the features in the test instance (i.e., $\mathcal{A}$) showed in the last row in Figure 6. The projected training data (i.e., $\mathcal{D}_\mathcal{A}$) is composed of the 5 instances showed in the figure. Suppose $minsup$ is set to 40%. In this case, the set of CARs, $\mathcal{C}^e$, found by the eager classifier is composed of the two CARs:
1. {windy=false and humidity=normal→play=yes}
2. {windy=false and temperature=cool→play=yes}

None of the two CARs matches the test instance, and thus, $\mathcal{C}^e_\mathcal{A}$=$\emptyset$. On the other hand, the projected training data has less instances ($\mathcal{D}_\mathcal{A} \subseteq \mathcal{D}$), and therefore, CARs not frequent in $\mathcal{D}$ may be frequent in $\mathcal{D}_\mathcal{A}$. This is because a frequent CAR must occur at least 4 times in $\mathcal{D}$ (since $|\mathcal{D}|$=10), but only 2 times in $\mathcal{D}_\mathcal{A}$ (since $|\mathcal{D}_\mathcal{A}|$=5). The lazy classifier found two CARs in $\mathcal{D}_\mathcal{A}$:
1. {outlook=overcast→play=yes}
2. {temperature=hot→play=yes}

These lazy CARs not only predict the correct class, but also are simpler than the eager CARs. Next we discuss how the lazy CARs perform when compared to the eager CARs.

Theorem 4 *Lazy CARs perform no worse than eager CARs, according to the information gain prnciple.*

Proof 4 *Theorem 3 showed that, for a given $minsup$, $\mathcal{C}^e_\mathcal{A} \subseteq \mathcal{C}^l_\mathcal{A}$. Let $\mathcal{R}^e$ be the best rule in $\mathcal{C}^e_\mathcal{A}$ (according to the information gain principle), and let $\mathcal{R}^l$ be the best rule in $\mathcal{C}^l_\mathcal{A}$. Two scenarios have to be considered when determining a class for the test instance $\mathcal{A}$. In the first scenario, $\mathcal{R}^l$ is identical to $\mathcal{R}^e$; in this case the same class is predicted by both eager and lazy classifiers. In the second scenario, $\mathcal{R}^l$ is better than $\mathcal{R}^e$, and thus $\mathcal{R}^l$ must provide a better prediction.* □

Theorem 4 states that the CARs added by the lazy classifier do not degrade the classification accuracy. This is because an additional lazy CAR is only used if it is better than all eager CARs (according to the information gain principle). Intuitively, lazy classifiers perform better than eager classifiers because of two characteristics:

- Missing CARs: Eager classifiers search for CARs in a large search space, which is induced by all features of the training data. While this strategy generates a large rule-set, CARs that are important to some specific test instances may be missed (this is particularly true for skewed/unbalanced distributions). Lazy classifiers, on the other hand, are context-sensitive and focus the search for CARs in a much smaller search space, which is induced by the features of the test instance.

- Highly Disjunctive Spaces: Eager classifiers generate CARs before the test instance is even known. In this case, the difficulty for the classifier is in anticipating all the different directions in which it should attempt to generalize its training examples (i.e., what CARs must be generated). For this reason, eager classifiers often combine small disjuncts in order to generate more general predictions (more general CARs should be applicable to more test instances). This can reduce classification performance in highly disjunctive spaces, where single disjuncts may be important to classify specific instances. Lazy classifiers, on the other hand, generalize their training examples exactly as needed to cover the test instance. Thus, lazy classifiers are often most appropriate when the search space is complex, and there are myriad ways to generalize a case.

The aforementioned discussion shows an intuitive concept, that is, the more CARs are generated, the better is the classifier. However, the same concept also leads to overfitting, reducing the generalization and affecting the classification accuracy. In fact, overfiting and high sensitivity to irrelevant features are shortcomings of lazy classifiers. A natural solution is to identify and discard the irrelevant features. Thus, feature selection methods have been proposed [7]. As the experimental results show in the next section, we have no evidence that our lazy classifiers were seriously affected by overfitting. We explain these results because just the best and more general CARs are used.

Another disadvantage is that lazy classifiers typically require more work to classify all test instances. However, simple caching mechanisms are very effective to decrease this workload. The basic idea is that different test instances may induce different rule-sets, but different rule-sets may share common CARs. In this case, memorization or caching of CARs is very effective in reducing work replication.

Our cache is a pool of entries, and it stores CARs of the form $\{\mathcal{X} \rightarrow c\}$. Each entry has the form <key,data>, where key=$\{\mathcal{X}, c\}$ and $data$={support,confidence,info gain}. A given CAR has only one entry in the cache, and

our implementation stores all cached CARs in main memory. Before generating a CAR, the algorithm first checks whether this CAR is already in the cache. If an entry is found, the CAR in the cache entry is used. Otherwise, the CAR is processed and then it is inserted into the cache.

The cache size is limited, and when the cache is full, some CARs are discarded to make room for others. Since it is impossible to predict how far in the future a specific CAR will be used, we choose the LFU (Least Frequently Used) heuristic (which counts how often a CAR is used, and those that are used least are discarded first), in order to improve cache efficiency. Rule caching is extremely effective in reducing the computation time for lazy classification. In fact, caching can make lazy classification faster than eager classification, as we will show in the next section.

5 Experimental Evaluation

In this section we show the experimental results for the evaluation of the proposed classifiers in terms of classification effectiveness and computational performance. Our evaluation is based on a comparison against C4.5 [19] and LazyDT [10] decision tree classifiers. We also compare our numbers to some results from other associative classifiers, such as CPAR [23], CMAR [16] and HARMONY [21], and to some results from rule induction classifiers, such as RISE [6], RIPPER [3], and SLIPPER [4]. We used 26 datasets from the UCI Machine Learning Repository to compare the effectiveness of the classifiers[2].

In all the experiments we used 10-fold cross-validation and the final results of each experiment represent the average of the ten runs. We quantify the classification effectiveness of the classifiers through the conventional error rate (the percentage of test instances incorrectly classified). We used the entropy method [9] to discretize continuous attributes. In the experiments we set minimum confidence to 50% (for confidence based classifiers) and $minsup$ to 1%. All other parameters were tuned according to [10,16,18,21]. The experiments were performed on a Linux-based PC with a Intel Pentium III 1.0 GHz processor and 1.0 GByte RAM.

5.1 Decision Trees and Eager Classifiers

We start our analysis by comparing the effectiveness of C4.5 decision trees and eager classifiers. Table 1 shows the error rates obtained by each classifier and the error reduction relative to C4.5. As can be seen, EAC always outperforms C4.5, which is not true for CBA, CMAR, and HARMONY. Further, CBA performs better when the dataset is sparse, that is, there is usually a large number of features and each instance contains just few of them. On average, EAC shows to be slightly better than CBA. CMAR performs better than EAC because it uses multiple CARs to classify a test instance, while EAC uses only the best ranked CAR.

Some datasets deserve special discussion. The hepatitis dataset, for instance, has two very unbalanced classes. EAC is able to reduce 20.8% of the errors in this dataset. We investigated the reason for such an improvement, and we observed that the effectiveness of C4.5 varies heavily with each class. For the most frequent class, C4.5 and EAC present a similar result. However, for the less frequent class, C4.5 performs poorly. Similar results were observed in the labor and sonar datasets. The gains provided by associative classification do not come exclusively from its capacity of performing equally accurately in all classes when they are unbalanced, since EAC performed much better than C4.5 in the waveform dataset, which has very well balanced classes. C4.5 was superior than CBA in the horse dataset. Investigating the reason for that, we observed that CBA generates a very large number of 100% confident rules (homogeneous partitions), so that breaking ties and selecting a single best rule becomes hard and prone to error.

5.2 Eager and Lazy Associative Classifiers

We continue our analysis by comparing the effectiveness of eager (EAC) and lazy classifiers (LAC). Table 1 shows their corresponding error rates. For very small datasets eager and lazy classifiers perform similarly, since the CARs that were generated by both classifiers were essentially the same for the parameters used. For instance, the result obtained with labor and zoo datasets were exactly the same. Also, we can observe that lazy classifiers perform better when the dataset is sparse (i.e., auto, pima, diabetes, german, and wine datasets). The error reduction in these datasets range from 13.9% to 52.7%. This result is expected, since the small disjuncts problem is more likely to happen in sparse datasets. Further, we can also notice that the lazy classifiers always outperform the corresponding eager ones, except for the ionosphere dataset. We believe that, for this dataset, the lazy classifiers have overfitted the data.

Figure 7 shows the rule-set utilization for eager and lazy classifiers. A CAR is useful if it is used to classify a test instance. Utilization is given by the number of useful CARs divided by the total number of distinct CARs that were generated. Ideally, only the best CAR matching each test instance should be generated. This ideal case corresponds to a total rule-set utilization (100%), that is, all generated CARs are used to classify at least one test instance. Since the number of test instances is constant, the rule-set utilization depends on the number of distinct CARs that are generated. Increasing $minsup$ reduces the number of CARs, and therefore, it increases rule-set utilization. However, the chance of having no CAR matching a test instance will also increase

[2]We used the implementations in the MLC++ library [15].

Dataset	Decision Tree Classifiers		Eager Associative Classifiers				Lazy Associative Classifiers		Error Reduction over C4.5(%)	
	C4.5 inf. gain	LazyDT inf. gain	EAC inf. gain	CBA conf	CMAR conf	HARMONY conf	LAC inf.gain	LAC conf	EAC inf. gain	LAC inf. gain
anneal	6.5	4.2	3.9	3.6	**2.7**	8.5	3.5	3.6	40.0	46.1
australian	13.5	15.2	13.0	13.4	13.9	-	**12.7**	13.4	3.7	6.2
auto	29.2	24.7	26.5	27.2	**21.9**	39.0	22.2	22.9	9.2	24.0
breast	3.9	5.1	3.6	4.2	3.6	7.6	**3.2**	4.1	7.7	17.9
cleve	18.2	17.2	16.1	16.7	17.8	-	**15.6**	16.7	11.5	14.3
crx	15.9	16.9	15.0	**14.1**	15.1	-	14.2	**14.1**	5.7	10.7
diabetes	27.6	24.9	24.6	25.3	24.2	-	**19.7**	21.3	10.9	28.6
german	29.5	26.1	27.2	26.5	25.1	-	23.4	**22.0**	7.8	20.7
glass	27.5	26.5	26.8	27.4	29.9	50.2	**26.0**	27.4	2.5	5.4
heart	18.9	17.7	18.1	18.5	17.8	43.5	**16.9**	17.2	4.2	10.6
hepatitis	22.6	20.3	17.9	**15.1**	19.5	22.0	17.1	**15.1**	20.8	24.3
horse	16.3	17.2	15.4	18.7	17.4	17.5	**14.5**	17.2	5.5	11.0
hypo	1.2	1.2	1.2	1.7	1.6	-	**1.0**	1.2	0.0	16.7
ionosphere	8.0	8.0	**7.6**	8.2	8.5	8.0	7.8	8.5	5.0	2.5
iris	5.3	5.3	4.9	7.1	6.0	6.7	**3.2**	4.6	7.5	39.6
labor	21.0	20.4	19.1	17.0	**10.3**	-	19.1	17.0	9.0	9.0
led7	26.5	26.5	24.2	27.8	27.5	25.4	**22.1**	25.5	8.7	16.6
lymph	21.0	20.1	20.2	19.6	**16.9**	-	19.1	18.2	3.8	9.0
pima	27.5	25.9	27.5	27.6	24.9	27.6	22.0	**21.3**	0.0	20.0
sick	2.1	2.1	2.1	2.7	2.5	-	**2.0**	**2.0**	0.0	4.8
sonar	27.8	24.6	22.9	21.7	20.6	-	20.5	**19.6**	17.6	26.3
tic-tac-toe	0.6	0.6	0.6	**0.0**	0.8	7.7	0.6	**0.0**	0.0	0.0
vehicle	33.6	31.8	30.8	31.3	31.2	-	**28.9**	30.0	8.3	14.0
waveform	24.6	22.7	21.3	20.6	**16.8**	19.5	19.3	18.9	13.4	21.5
wine	7.9	7.9	7.2	8.4	5.0	8.1	**3.4**	3.9	8.9	57.0
zoo	7.8	7.8	6.6	5.4	**2.9**	7.0	6.5	5.4	15.4	16.7
Average	17.1	16.2	15.5	15.8	14.8	19.9	**14.0**	14.3	8.7	18.2

Table 1. Error Rates for Decision Trees and Associative (Eager and Lazy) Classifiers.

(i.e., the missing rule problem). On the other hand, lowering $minsup$ obviously reduces the percentage of CARs that are used, since the number of CARs will increase. As we can see, reducing $minsup$ always reduces the percentage of CARs that are used. Further, lazy classifiers always provide a better rule-set utilization. This is because lazy classifiers focus only on the features of each test instance during rule generation. Differently, eager classifiers use all features within the training data, and therefore, the search space for CARs is augmented. The rule-set utilization of lazy and eager classifiers approaches for dense datasets, since in this case the search space for CARs tends to be similar.

5.3 Rule Induction and Associative Classifiers

We also compared the proposed eager and lazy associative classifiers against RISE, RIPPER and SLIPPER, using results reported in [3,4,6]. Table 2 shows the relative performance for each classifier (i.e., the accuracy of one classifier divided by the accuracy of the other classifier), when compared to eager associative classifiers. Each number in this table indicates how many times EAC is superior than the corresponding adversary RISE, RIPPER or SLIPPER (in terms of accuracy). The SLIPPER algorithm won in 6 of the 26 datasets (and lost in 11), showing to be most competitive rule induction classifier. The RIPPER classifier won in 4 datasets and the RISE classifier won in only one dataset.

Table 2 also shows the relative performance of rule induction classifiers when compared to the lazy associative classifiers. RISE and RIPPER lost in almost all datasets, and SLIPPER was the most competitive one. Compared to LAC (inf. gain), SLIPPER won in one dataset (and matched in 6), and compared to LAC (confidence), SLIPPER won in 4 datasets (and matched in 2).

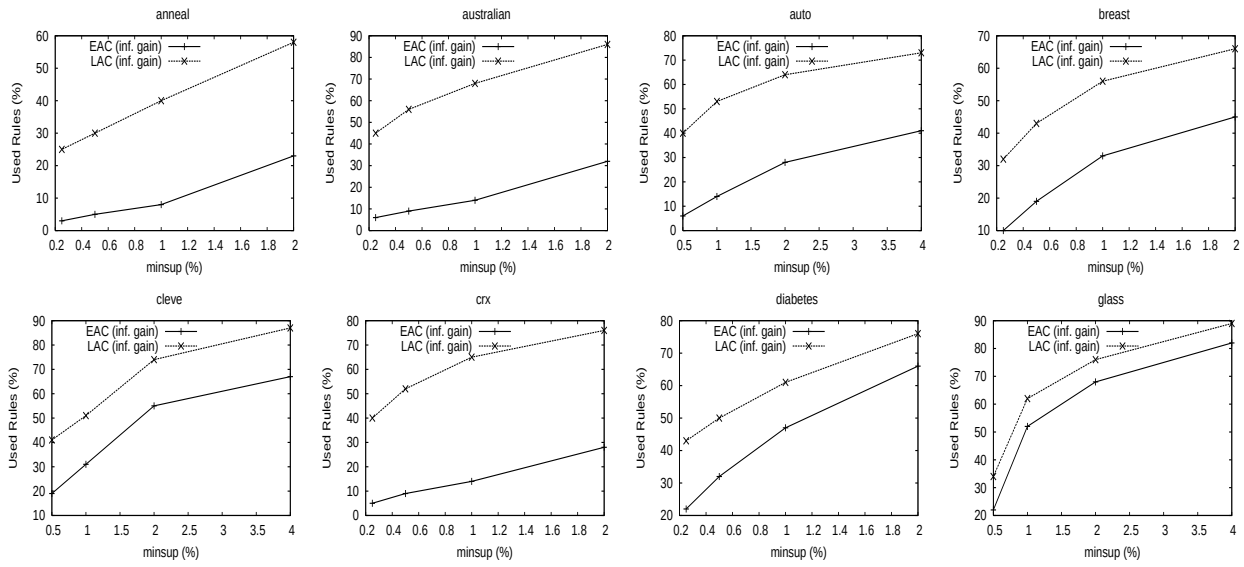

Figure 7. Rule-Set Utilization.

5.4 Overfitting and Underfitting

We also analyze the sensitivity of the classifiers regarding the complexity of the models they induce. The model complexity has an intuitive relation with the size of the rules (or the height of the tree) that constitute the model. The longer the rules within the model, the more complex it is. By using complex rules, the classifier tends to overfit the data, hurting the classifier accuracy. On the other hand, by using simple rules, the classifier will underfit the data also hurting the accuracy. Thus, the choice of the complexity of the rules is a trade-off between underfitting and overfitting, and may vary according to the dataset being used. Figure 8 shows the error rates associated with different classifiers for some of the datasets by varying the complexity of the model. The datasets anneal, cleve, crx, glass, led7, sick and zoo benefit from more complex models. These datasets are more dense and need more complex rules. We notice that the effectiveness of both eager and lazy classifiers are similar for these datasets. On the other hand, for datasets such as iris, auto, lymph and diabetes there is a demand for simpler rules. We observed that the lazy classifier is able to provide improvements in the majority of the cases.

5.5 Execution Times

Table 3 shows the execution times obtained by C4.5, RIPPER, EAC, and LAC classifiers. In all executions, $minsup$ was set to 1%, and the cache size is set to 10,000 CARs. All times correspond to the total time spent (in seconds) using 10-fold cross-validation. In general, C4.5 is the worst performer. RIPPER and EAC are very competitive, but RIPPER shows to be slightly superior. For some datasets (such as hypo, sick and waveform), EAC and RIPPER generate a very large number of CARs. Although LAC implements extra work, namely feature projection and rule generation for every test instance, it usually generates much less distinct CARs than EAC and RIPPER, and thus, LAC is the best performer on average.

6 Conclusions and Future Work

Decision tree classifiers perform a greedy search which may discard relevant information. Based on this observation we present an assessment of associative classification. The generated CARs are ranked based on their information gain, so that we can compare the performance of decision trees and associative classifiers. We present evidence regarding the superiority of associative classifiers. However, it is well known that no classifier can outperform others in all settings [22], and thus there may be certain specific situations where decision trees outperform associative classifiers.

We also propose improvements to associative classification by introducing a novel lazy classifier. The lazy classifier searches a larger hypothesis space than the corresponding eager classifier, because it uses many different local models to form its implicit global approximation to the target function. Eager classifiers commit at training time to a single global approximation. An important feature of the proposed lazy classifier is its ability to deal with the *small disjuncts* problem. Based on this observation, we present evidence showing that a lazy associative classifier outperforms the corresponding eager one. Our claims were con-

Dataset	Relative Performance EAC(inf.gain)			Relative Performance LAC(inf. gain)			Relative Performance LAC(confidence)		
	RISE	RIPPER	SLIPPER	RISE	RIPPER	SLIPPER	RISE	RIPPER	SLIPPER
anneal	1.01	1.00	1.00	1.01	1.00	1.01	1.01	1.00	1.01
australian	1.03	1.00	0.99	1.04	1.00	1.00	1.03	0.99	0.99
auto	1.04	1.00	0.98	1.10	1.06	1.04	1.09	1.06	1.03
breast	1.02	1.01	1.01	1.03	1.02	1.02	1.02	1.00	1.01
cleve	1.03	1.02	1.00	1.03	1.02	1.01	1.02	1.01	0.99
crx	1.00	1.00	1.00	1.01	1.01	1.01	1.01	1.01	1.01
diabetes	1.03	1.01	1.00	1.10	1.08	1.06	1.08	1.05	1.04
german	1.03	1.04	1.02	1.08	1.10	1.08	1.10	1.12	1.10
glass	1.08	1.06	1.02	1.10	1.07	1.03	1.07	1.05	1.02
heart	1.07	1.01	1.01	1.09	1.03	1.02	1.09	1.03	1.02
hepatitis	1.09	1.07	1.05	1.10	1.08	1.06	1.12	1.11	1.09
horse	1.00	0.99	0.99	1.02	1.01	1.00	0.98	0.98	0.97
hypo	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
ionosphere	1.03	1.01	1.00	1.03	1.01	1.00	1.02	1.00	0.99
iris	1.02	1.02	1.00	1.04	1.03	1.02	1.02	1.02	1.01
labor	0.99	0.96	0.95	0.99	0.96	0.95	1.01	0.99	0.98
led7	1.06	1.09	1.03	1.08	1.12	1.06	1.04	1.07	1.01
lymph	1.06	1.01	1.00	1.07	1.02	1.01	1.08	1.04	1.02
pima	1.04	0.99	0.96	1.12	1.07	1.04	1.13	1.08	1.05
sick	1.01	1.00	1.00	1.01	1.00	1.00	1.01	1.00	1.00
sonar	1.02	0.98	1.01	1.05	1.01	1.00	1.06	1.03	1.01
tic-tac-toe	1.02	1.02	1.01	1.01	1.01	1.01	1.02	1.02	1.02
vehicle	1.05	1.10	1.01	1.08	1.13	1.04	1.06	1.12	1.03
waveform	1.05	1.04	1.02	1.08	1.06	1.04	1.08	1.07	1.05
wine	1.01	1.01	0.98	1.05	1.05	1.02	1.04	1.05	1.02
zoo	1.07	1.06	1.02	1.07	1.06	1.02	1.08	1.07	1.03
Win/Tie/Lost	22/3/1	16/6/4	11/9/6	24/1/1	21/4/1	19/6/1	24/1/1	18/4/4	20/2/4

Table 2. Performance of Associative Classifiers relatively to RISE, RIPPER and SLIPPER.

firmed by empirical comparisons to C4.5 and LazyDT decision tree classifier, using datasets from the UCI data repository. We also compared the proposed classifiers against other three associative classifiers and three rule induction classifiers and outperformed them in most of the cases.

So far, our classifiers use only the best CAR for sake of classification. In the future we will combine simple and complex CARs in order to enhance classification. Finally, we will explore more realistic application scenarios.

References

[1] D. Aha. Lazy learning. *Artificial Intelligence Review*, 11:1–5, 1997.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees. *Wadsworth Intl.*, 1984.

[3] W. Cohen. Fast effective rule induction. In *In'l Conf. on Machine Learning*, 1995.

[4] W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proc. of the national conference on Artificial intelligence*, pages 335–342, Menlo Park, CA, USA, 1999.

[5] B. Dasarathy. *Nearest Neighbor Pattern Classification Techniques*. IEEE Computer Society Press, 1990.

[6] P. Domingos. Rule induction and instance-based learning: A unified approach. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, 1995.

[7] P. Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997.

[8] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by aggregating emerging patterns. In *Proc. Intl. Conf. on Discovery Science*, pages 30–49, 1999.

[9] U. Fayyad and K. Irani. Multi interval discretization of continuous-valued attributes for classification learning. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, pages 1022–1027, 1993.

[10] J. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. In *Proc. of the Conf. on Art. Intelligence*, pages 717–724, 1996.

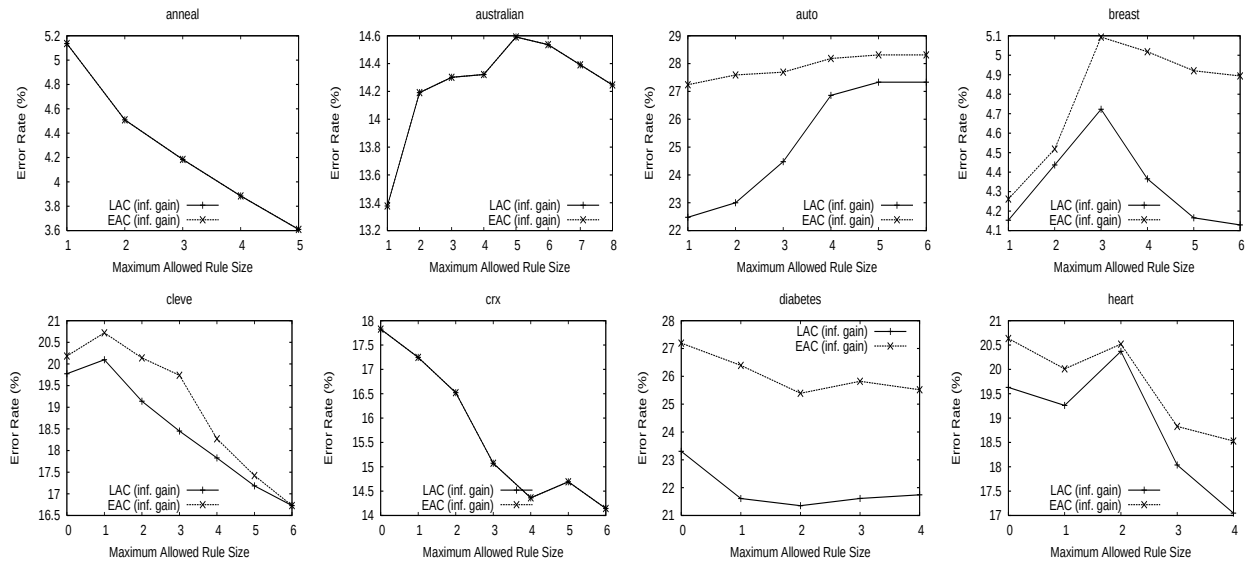

Figure 8. Rule Size and Error Rates.

[11] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Morgan Kaufmann, 1989.

[12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[13] R. Holte, L. Acker, and B. Porter. Concept learning and the problem of small disjuncts. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, pages 813–818, 1989.

[14] M. James. *Classificaton Algorithms*. Wiley, 1985.

[15] R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 234–245, 1996.

[16] W. Li, J. Han, and J. Pei. Efficient classification based on multiple class-association rules. In *Proc. of the Intl. Conf. on Data Mining*, pages 369–376, 2001.

[17] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(22), 1987.

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

[19] J. Quinlan. C4.5: Programs for machine learning. *Morgan Kaufmann*, 1993.

[20] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[21] J. Wang and G. Karypis. HARMONY:efficiently mining the best rules for classification. In *Proc. of the Intl. Conf. on Data Mining*. SIAM, 2005.

[22] D. Wolpert. The relationship between PAC, the statistical physics framework, the bayesian framework, and the VC framework. *The Mathematics of Generalization*, 1995.

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

Dataset	C4.5 inf. gain	RIPPER	EAC inf. gain	LAC inf. gain
anneal	24.2	19.8	18.7	**14.5**
australian	9.1	**5.4**	7.0	6.1
auto	4.4	**2.9**	4.4	4.2
breast	1.6	1.1	**0.9**	1.9
cleve	5.9	**4.5**	5.9	6.8
crx	9.3	5.6	**4.7**	5.6
diabetes	0.8	0.9	**0.3**	1.1
german	14.4	6.6	6.3	**4.6**
glass	1.8	**0.8**	1.1	1.7
heart	4.1	**3.8**	4.7	3.9
hepatitis	3.8	**2.7**	**2.7**	4.2
horse	9.7	**5.7**	6.4	6.2
hypo	69.9	42.7	48.9	**34.2**
ionosphere	13.2	7.2	5.9	**3.6**
iris	5.5	**1.8**	2.3	2.7
labor	1.3	**1.1**	1.2	1.2
led7	3.6	**1.9**	2.4	2.9
lymph	11.5	**4.2**	4.8	**4.2**
pima	5.4	2.2	**1.9**	2.5
sick	65.8	32.5	38.8	**27.6**
sonar	12.1	**5.5**	7.3	6.1
tic-tac-toe	2.8	**1.4**	2.0	1.8
vehicle	14.4	8.6	8.2	**6.5**
waveform	19.8	12.8	16.6	**11.7**
wine	9.2	4.5	**3.3**	6.8
zoo	7.1	**3.2**	4.7	6.6
Average	12.6	7.2	8.0	**6.9**

Table 3. Execution Times (seconds).