Learning to Use a Learned Model: A Two-Stage Approach to Classification*

Maria-Luiza Antonie
Dept. of Computing Science
University of Alberta, Canada
luiza@cs.ualberta.ca

Osmar R. Zaïane
Dept. of Computing Science
University of Alberta, Canada
zaiane@cs.ualberta.ca

Robert C. Holte
Dept. of Computing Science
University of Alberta, Canada
holte@cs.ualberta.ca

Abstract

Association rule-based classifiers have recently emerged as competitive classification systems. However, there are still deficiencies that hinder their performance. One deficiency is the use of rules in the classification stage. Current systems assign classes to new objects based on the best rule applied or on some predefined scoring of multiple rules. In this paper we propose a new technique where the system automatically learns how to use the rules. We achieve this by developing a two-stage classification model. First, we use association rule mining to discover classification rules. Second, we employ another learning algorithm to learn how to use these rules in the prediction process. Our two-stage approach outperforms C4.5 and RIPPER on the UCI datasets in our study, and outperforms other rule-learning methods on more than half the datasets. The versatility of our method is also demonstrated by applying it to text classification, where it equals the performance of the best known systems for this task, SVMs.

1 Introduction

Classification is an important task in many applications. A classifier is a system that assigns, or predicts, one or more class labels for a given object. One way to create a classifier is to use a learning algorithm to construct a classifier from a training set of objects whose classification(s) is known.

Associative classifiers [2, 18, 19] consist of a set of rules, with each rule predicting that an object belongs to a specific class if it has certain properties. The rules are discovered using association rule mining algorithms [1]. The association rule mining problem has been thoroughly studied in the data mining community [13], thus there are several fast algorithms for discovering these types of rules.

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

For example, CBA [19] classifies an object using only the highest ranking rule that applies to the object, where rank is defined by the rule's confidence on the training set. This method has two shortcomings. The first is that by basing its classification on only the highest ranking rule, CBA might be ignoring a large number of high ranking, applicable rules that might agree with each other and disagree with the highest ranking rule. Secondly, because each rule predicts just a single class, CBA is incapable of assigning a given object to multiple classes simultaneously, which is essential in some applications.

CMAR [18] and ARC [2] overcome CBA's shortcomings by selecting the K highest ranking applicable rules, not just the first. The key issue now is, how to combine the class predictions of the selected rules to produce a final classification? Both systems use a weighted voting scheme, they differ in the details of how weights are calculated. CMAR uses a chi-square weighting scheme, while ARC weighs classes based on the average confidence of the selected rules that predict that class. These systems trade part of their comprehensibility inherited from the association rules for improved performance. This tradeoff is the results of using a weighting score on the rules.

In this paper we also use a weighted voting scheme to combine the class predictions of the selected rules to produce a final classification, but instead of pre-defining the way in which weights are computed, we use a second learning algorithm to determine the weights. Learning in our system therefore takes place in two stages. First, an associative classifier is learned using standard methods. Second, predefined features computed on the outputs of the rules in the learned associative classifier are used as the inputs to a neural network, which is trained, using a separate training

*This work was partially funded by Alberta Ingenuity Fund, Informatics Circle of Research Excellence (iCORE) and Natural Sciences and Engineering Research Council of Canada (NSERC).

set, to weigh the features appropriately to produce highly accurate classifications.

This two-stage system, with a layer of feature definitions interposed between the output of the first learned system and the input of the second, is the paper's main contribution. Another significant contribution of the paper is the use of a statistical analysis of the experimental data that is more rigorous than previous studies.

The performance of our two-stage system is evaluated experimentally on two distinct classification tasks, single label classification and multiple label classification. In single label classification, the task is to assign an object to exactly one of the classes. This is the standard classification task studied in machine learning, and a variety of test datasets and systems are available for comparison. We use twenty of the UCI datasets[5], and compare our approach to seven existing systems. Our approach outperforms C4.5 and RIPPER on all the datasets in our study, and outperforms the other rule-learning methods on about half the datasets.

In multiple label classification, an object can be assigned to several of the classes simultaneously. The standard testbed for this task is classifying news articles into subject categories, where it is necessary for some news articles to be assigned to multiple categories. For example, an article about selling a sports franchise should be put into at least two categories, "sports" and "business". The best known algorithms for this text classification task are SVMs [16]. Our experiments using the Reuters dataset establishes our two-stage system as the first classification method to equal the performance of SVMs on this task.

2 Prerequisites

A classifier is built by applying a learning method to a training set of objects. This model is further used to predict the labels to new incoming objects. With all the effort in this domain there is still place for improvement and a great deal of attention is paid to develop highly accurate classifiers.

The use of association rule mining for building classification models is very new. Recent studies have proposed the use of association rules in building effective classifiers. These classification systems discover the strongest association rules in the dataset (associating attribute values to class labels) and use them to build a categorizer.

2.1 Association Rule Mining

Mining association rules from data is the process of finding interesting relationships or associations that exist between objects in the collection to be mined.

Association rule mining is a data mining task that discovers relationships between items in a transactional database. Association rules have been extensively studied in the literature. The efficient discovery of such rules has been a major focus in the data mining research community, given their popularity in market basket analysis. From the original *apriori* algorithm [1] there have been a remarkable number of variants and improvements [13].

Formally, association rules are defined as follows: Let $\mathcal{I} = \{i_1, i_2, ...i_m\}$ be a set of items. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. A transaction T is said to contain X, a set of items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form "$X \Rightarrow Y$", where $X \subseteq \mathcal{I}, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has a *support* s in the transaction set $\mathcal{D}$ if $s\%$ of the transactions in $\mathcal{D}$ contain $X \cup Y$. In other words, the support of the rule is the probability that X and Y hold together among all the possible presented cases. It is said that the rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with *confidence* c if $c\%$ of transactions in $\mathcal{D}$ that contain X also contain Y. In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedent X. The problem of discovering all association rules from a set of transactions $\mathcal{D}$ consists of generating the rules that have a *support* and *confidence* greater than given thresholds. These rules are called *strong rules* and represent interesting patterns in data.

2.2 Associative Classifiers

The first reference to using association rules as classification rules is credited to [4], while the first classifier using these association rules was CBA introduced in [19] and later improved in CMAR [18], and ARC-AC [2] and ARC-BC [2]. The idea is relatively simple. Given a training set modelled with transactions where each transaction contains all features of an object in addition to the class label of the object, we can constrain the mining process to generate association rules that always have as consequent a class label. In other words, the problem consists of finding the subset of strong association rules of the form $X \Rightarrow C$ where C is a class label and X is a conjunction of features.

The main steps in building an associative classifier when a training set is given are the following:

1. *Modeling the data into transactions.*
2. *Generating the set of association rules from the training set:* In this phase association rules of the form *set_of_features* $\Rightarrow$ *class_label* are discovered by using a mining algorithm. This step of the algorithm can be completed in two ways:
    - Generating all the strong association rules using an existing association rule mining algorithm. Once these rules are generated, filter them so that only the rules of interest are kept (those that have

as consequent a class label and their antecedent is composed of features other than class labels).

- Modifying an association rule mining algorithm by imposing a constraint: the consequent of an association rule must be a class label. In this way, a more efficient algorithm is employed since less candidate items are generated. All candidate itemsets generated contain a class label and only the association rules with a class label on the right hand side are generated.

CBA and ARC use an apriori-like fashion [1] to generate the classification rules. CMAR uses an FP-growth approach [14] for rule generation. All methods impose a constraint on the rule form such that only classification rules are generated.

3. *Pruning the set of discovered rules:* In the previous phase a large set of association rules can be generated, especially when a low support is given. Pruning techniques are used to discover the best subset of rules that can cover the training set. This phase is employed to weed out those rules that may introduce errors or are overfitting in the classification stage. Pruning based on database coverage (i.e., checking the extent of exposure of rules in the database) has been used by all associative classifiers [2, 18, 19]. CMAR proposed reducing the number of rules by removing low ranked specialized rules. For other pruning techniques see [29].
4. *Classification phase:* At this level a system that can make a prediction for a new object is built. The challenge here is how to make use of the set of rules from the previous phase to give a good prediction. Using the rules to classify a new object means to have a good way of selecting one or more rules to participate in the prediction process. CBA classifies a new object with the class associated to the first rule that is applicable, which is the rule with the highest confidence. In [18], the CMAR's authors argue that making the decision based on a single rule leads to poor results. They claim that a decision based on a set of rules is more appropriate since more information is available. In our previous work [2], we also show that decision making based on a set of rules has more potential than single-rule based classification. Not only that the decisions are better when multiple rules are taken into consideration, but it also allows us to perform multi-label classification.

We consider that associative classifiers have two stages where more research is required to fully take advantage of their capabilities. First, the number of rules generated by an associative classifiers is very large. Better pruning techniques have to be studied and devised to select a small set of quality rules. Associative classifiers have the advantage over other rule-based classification systems that they guarantee to find all interesting rules (in the support-confidence framework). However, more research is required to weed out those rules that are not useful or even detrimental in the classification process. Second, the rule selection and their scoring in the classification process follows either some naïve techniques or use predefined schemes for scoring a new instance to be classified. However, a good selection strategy is fundamental for good accuracy.

In this paper, we focus our efforts on improving the classification phase by automatically learning to use the rules in the model. Next section gives a detailed description of our proposed technique.

3 Our Two-Stage Approach to Classification

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

Once developed, the purpose of a classification system is to classify new instances. In the case of associative classifiers, this step deals with using the rules to categorize a new object. The next example will describe a rule-base model. It uses different classification strategies.

Example 1. Table 1 shows a hypothetical rule-based model that was generated from a given training set using an association rule learning method. Suppose we are given an object O to classify that has features A, B, and C. The subset of rules applicable to O is shown in Table 2.

Table 1. Example of a rule-based model

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

Different methods can be used to classify new instances when a set of rules apply. These rules have measures attached, such as confidence in this example. Let us consider now different approaches for prediction:

- if we classify by the highest ranked rule that applies we have to predict *Class 3* for O, based on rule R8;
- if we predict the class whose applicable rules have the highest average confidence, object O will be classified

Table 2. Applicable rules to object O{A,B,C}

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

as *Class 1* (Average of 85% for Class 1; 81.6% for Class 2 and 80% for Class 3);

- if we classifiy by the largest number of applicable rules per class, object *O* will be classified as *Class 2*.

The preceding prediction schemes are just simple examples to illustrate that there is not a unique way to combine the individual conclusions of a set of rules to create a final classification. The actual schemes used by existing systems are as follows. CBA [19] classifies a new object with the class of the highest confidence-based ranked rule. In [18], the authors use a weighted chi-square over the rules that apply and chooses the class with the highest score. In our previous work [2], we base our prediction on a set of rules by using the average of the confidences. We also experimented with other measures such as cosine measure, Jaccard coefficient, etc. [22]. Some have also experimented with the size of the rule [8]. Bagging [7] and boosting [24] are two ensemble methods that achieve a better classification performance by combing the conclusion of multiple classifiers. These classifiers are generated by using the same learning method on different distributions of data. Bagging uses different replicas of the training set, while boosting uses the same training examples for all the classifiers but attaches different weights to them. The classifiers generated are combined by voting to obtain a final decision. In bagging, all classifiers have equal weight in the voting process. Boosting weighs each classifier's vote by its accuracy on the training set.

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

Given a data collection for classification purposes, we have a training and a testing set. These sets are either generated (cross-validation) or are given apriori with the application to solve. The associative classification methods described before learn a set of classification rules from the training set and then uses them in different fashions (depending on the method employed) to classify the testing set.

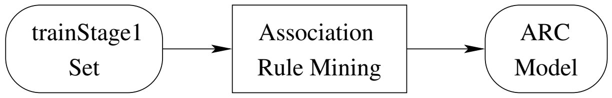

Figure 1. First stage of learning

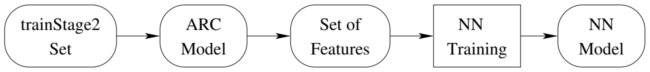

Figure 2. Second stage of learning

Our method to learn to use the generated rules and select the appropriate ones works as follows:

1. split the training set into two subsets: *trainStage1* and *trainStage2*; given that we have two learning stages in our technique we need to have enough examples for both stages; we split the initial training set in two disjoints subsets; this ensures the ability of our second stage to improve the performance of the overall model without overfitting by learning on the same set as the first stage;
2. use *trainStage1* to extract the classification rules using an association rule mining algorithm; we denote with *ARC Model* the discovered set of rules; Figure 1 shows this step of the algorithm;
3. for each instance in *trainStage2*, use *ARC Model* to collect a set of features (class-based or rule-based);
4. apply a neural network learning (or another learning method) in this new feature space to learn how to use the rules in the prediction process; denote the model generated by the second learning algorithm as *NN Model*; Figure 2 shows this step of the algorithm;
5. classify the objects in the testing set using *ARC Model* and *NN Model* combined (see Figure 3); when a new object has to be classified *ARC Model* generates the features for *NN Model*, which in turn makes the classification decision;

As one may see from Example 1, to score and make a decision for a new object is a difficult task. The second

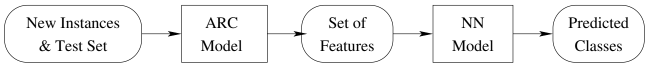

Figure 3. Evaluation and classification

learning method in our technique acts as a scoring scheme. The advantage of using this intermediary step is twofold: it takes into account more information than using just one or few rules for prediction; treating the scoring scheme as a learning problem we create an automated process to learn it from data (different scoring schemes are automatically generated for different applications).

Given that there is a numerical scoring scheme that we have to learn, we chose as the second learning algorithm a neural network. The network consists of a standard 3-layer, feedforward neural network whose inputs are the set of features derived from the first stage output. The number of outputs equals the number of classes in the application, each output neuron corresponding to a class. A logistic function is used to compute the weights inside the network. The output of the function ranges from 0 to 1. When a single class has to be predicted for an object the output neuron (i.e., class) with the highest value is chosen as the winner. If multiple classes have to be predicted every class above a threshold is considered a winner.

There are two approaches that we developed and tested for our two stage associative classification system (2SARC). They differ in the feature space generated and are detailed in the following sections.

3.1 2SARC1: Class-based Features

Given a classification problem that has n classes to be learned, we use association rules mining to discover classification rules. Let us consider that our discovered model consists of N rules, ordered by their confidence and support. Each rule R_i has an associated support σ_i and a confidence c_i. When a new instance is to be classified, a subset of K rules applies $R = \bigcup R_i$, where $i \subset [1..K]$. A rule is applicable to a new instance if the antecedent of the rule matches the new object and its confidence is within the confidence margin [2]. Naturally, a scoring scheme has to be used in order to make the classification decision. This scoring scheme has to be a function that represents the strength of the decision and it has to take into account all applicable rules. The rules are characterized by support, confidence and their class. The scoring scheme has to produce a score for all the classes to be considered.

From the set of rules that apply to the new object, M measures m_j, $j \in [1..M]$ can be computed for each class c_i that appeared in R $(m_1(c_1), ..., m_M(c_1), m_1(c_2), ...m_M(c_n))$. For instance, these measures could be the average confidence, number of rules applied and maximum confidence. All these aggregations are done by class. At this stage each class will have several measures associated. All these measures will be the input to the second learning method, which is a neural network in our case.

Table 3. 2SARC1: class features for object O

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

Table 4. 2SARC2: rule features for object O

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

Considering Example 1, the features collected for object O are average confidence and number of rules by class as shown in Table 3. For each class a set of features is generated (e.g., Class1: average confidence is 85%, # rules supporting this class is 1). These class-based features are generated for the objects in the trainStage2 set. Along with the class label these features represent now the input to the second learning stage.

3.2 2SARC2: Rule-based Features

In our previous approach, we collected aggregated measures per class. These measures represented a new feature space. From this space a scoring scheme is learned.

In our second approach we learn how to use the rules in the model instead of learning how to use the measures. The feature space in this case is represented by the rules. Given our rule-based ARC model, we input into the second learning method the characteristics of the rules in the ARC model. For each new object, a rule either applies or not. We introduce this information along with the rule's confidence into the model for the second stage.

For the example in Table 1, given object O, we can generate a set of rule-based features as presented in Table 4. The second learning algorithm has to learn how to use the rules given these type of features.

Generally, association-based models generate a large number of rules, thus creating for 2SARC2 a very large feature space. To reduce the dimension of the feature set we experimented with choosing only k best rules per class (e.g., k=2 for our example, R1, R3 for Class 1; R2 and R5 for Class 2; R8 and R9 for Class 3).

Although the architecture of our proposed system may seem similar to stacking [26] they differ as follows. In stacking level one is represented by one or more classification methods. Their classification results represent the input space for the second level classifier. In our approaches, the first level is a rule-based method that discovers classification rules. The input to the next level are features generated using the model built in the first level. This feature layer is what distinguishes our techniques from stacking.

4 Experiments

First, we evaluated our method against other associative classifiers and rule-based classification methods on 20 UCI datasets [5]. Second, we studied the performance of our system in the text classification context. This type of application is more challenging given that the feature space is very large and it requires multi-label classification (i.e., one or more class labels are attached to each object to be classified). UCI datasets are single-label classification problems.

4.1 Single Label Classification

We evaluated our algorithm against other associative classifiers (see Section 2.2) on several UCI datasets [5]. In addition we compare our method with two rule-based classification methods (C4.5 rules [21] and RIPPER [9]), a boosting algorithm [24] and a hybrid between rule-based methods and associative classifiers (CPAR) [28].

Rule-based classification approaches have been developed for decades in the machine learning community due to their readability when compared to other classifiers. These algorithms use greedy techniques in the rule generation process. C4.5 searches the space for the best attribute according to the heuristic used, divides the search space according to the values of the attribute and then continues the process recursively in those subspaces. Rules are generated following the paths that cover the feature space. RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [9] is built upon IREP (Incremental Reduced Error Pruning) algorithm [12]. Following IREP's strategy, RIPPER splits the training set in two sets. One of them is used to grow the rules and the other to prune the rules. The algorithm starts with an empty rule and it repeatedly adds conditions that maximize the information gain criterion. Once the rule is grown, conditions are deleted to maximize a function during pruning phase. When a rule has been discovered, all the examples that are covered by this rule are removed from the training set. The above process continues to learn rules for the remaining training set.

CPAR [28] (Classification based on Predictive Association Rules) is a hybrid between associative classifiers and rule-based classifiers that use greedy techniques. It uses a greedy algorithm to search the space of attributes.The main difference is that it keeps all close-to-the-best attributes in rule generation, unlike rule-based methods which use only the best attribute.

On each UCI dataset we performed C4.5's shuffle utility [21]. A 10-fold cross validation was performed on each dataset and the reported results are averages of the accuracies over the 10 folds. In addition, we used the same discretization method for continuous attributes as in [19] to have a fair comparison with the other algorithms.

All classification methods should be evaluated on the same randomly generated folds to ensure a fair comparison of the methods. As the code for CPAR has the cross validation incorporated and it does not allow one to specify the folds we could not guarantee that CPAR was evaluated on the same folds as the other algorithms. CMAR code is not available. Thus the results presented in our table are the best results for CPAR and CMAR as reported by their authors. All the other results were obtained in our study. For CBA we used the code provided by their authors, while for the other algorithms (C4.5, RIPPER, boosted RIPPER) we used their Weka [25] implementations (Weka version 3.4.8). The parameters for all the algorithms were set to their default values. Weka's default settings follow the best parameter setup as proposed by their respective authors [9, 21].

It is well-known that the support threshold plays a fundamental role in association rule discovery. Since associative classifiers are based on association rule mining they inherited the sensitivity to the support threshold. It is hard to know apriori the best support threshold and its value is usually set experimentally. In our evaluation, we ran our algorithms with support values of 1%, 5% and 10% for each dataset and we reported the best result. The minimum confidence was set to 50% and the confidence margin was set at 10%. Our proposed technique used a split ratio of 50/50 or 75/25 between trainStage1 and trainStage2 sets. 2SARC2 technique used 10, 20 or 30 rules per class. The algorithm used for the second stage was a backpropagation algorithm with 1 hidden layer. For the backpropagation algorithm we used Borgelt's implementation [6]. The number of neurons in the hidden layer was the average number between the input and output neurons.

Table 5 presents the accuracy of the following methods: C4.5, RIPPER, boosted RIPPER, CBA, CMAR, CPAR, ARC and the results for the two techniques proposed in this paper (2SARC1 and 2SARC2). Along with the accuracy result, the name of the dataset, the number of records in the datasets and the number of classes are given. The standard deviation is not reported as our statistical analysis uses non-parametric tests.

2SARC1 obtains best overall performance for 5 datasets, followed by CMAR (4 datasets) and C4.5 and boosting (3 datasets each). 2SARC2 performs best on 2 datasets. Ripper and CBA are the algorithms that do not have any overall win. On some datasets the differences in accuracy between the winner and the second best are quite small (e.g., austra), while for other the improvement in the performance is large (e.g., hepati, heart). Some datasets are very small (e.g., labor, zoo), which unfavours our algorithms given that we need enough examples for the two learning stages.

2SARC1 has the best overall performance, as it can be see fromTable 5. Table 6 shows the count of wins, losses and ties for 2SARC1 when compared to the rest of the meth-

Table 5. Accuracy on 20 UCI datasets for several classification methods

dataset	# rec	#cls	ARC	2SARC1	2SARC2	C4.5	Ripper	BooR	CBA	CMAR	CPAR
anneal	898	6	97.11	98.01	97.58	89.87	94.66	**99.33**	97.91	97.30	98.40
austra	690	2	86.23	86.84	81.90	**86.96**	85.80	85.80	85.38	86.10	86.20
breast	699	2	**96.42**	96.13	95.85	94.71	95.28	95.85	96.28	96.40	96.00
cleve	303	2	82.16	83.16	80.15	80.52	80.19	**83.51**	82.83	82.20	81.50
crx	690	2	85.36	**85.97**	84.51	85.36	85.80	84.78	85.38	84.90	85.70
diabetes	768	2	74.22	**76.27**	75.63	74.21	74.34	73.95	74.45	75.80	75.10
german	1000	2	73.20	72.50	73.05	71.60	71.60	71.90	73.50	**74.90**	73.40
glass	214	6	71.13	72.47	71.05	71.47	69.70	69.70	73.90	70.10	**74.40**
heart	270	2	80.74	**84.09**	80.39	80.74	81.85	83.33	81.87	82.20	82.60
hepati	155	2	81.13	**85.16**	80.21	79.25	76.04	80.08	81.82	80.50	79.40
horse	368	2	84.23	82.63	81.80	**85.04**	84.23	82.89	82.36	82.60	84.20
iris	150	3	95.33	94.67	**96.00**	94.00	94.00	94.67	94.67	94.00	94.70
labor	57	2	80.67	86.33	86.33	81.00	86.33	**91.33**	86.33	89.70	84.70
led7	3200	10	71.91	72.91	73.66	**74.19**	69.31	69.31	72.06	72.50	73.60
pima	768	2	74.87	74.34	74.86	73.70	73.19	73.84	72.90	**75.10**	73.80
tic-tac	958	2	98.65	**100.00**	99.95	85.60	97.71	98.33	99.59	99.20	98.60
vehicle	846	4	65.02	67.49	64.90	67.04	64.31	68.67	68.92	68.80	**69.50**
waveform	5000	3	78.28	75.72	77.62	75.08	75.04	78.92	79.68	**83.20**	80.90
wine	178	3	88.79	95.50	**97.18**	92.12	92.12	96.67	94.96	95.00	95.50
zoo	101	7	95.09	91.18	93.18	92.18	89.09	96.09	96.78	**97.10**	95.10

Table 6. Method 2SARC1 compared to the rest of the algorithms on UCI datasets; (*) indicates statistical significant difference

	wins	losses	ties
2SARC1 vs. ARC	13	7	0
2SARC1 vs. 2SARC2	12	7	1
2SARC1 vs. C4.5*	16	4	0
2SARC1 vs. Ripper*	18	1	1
2SARC1 vs. boostingR	11	8	1
2SARC1 vs. CBA	12	6	2
2SARC1 vs. CMAR	13	7	0
2SARC1 vs. CPAR	10	8	2

ods. If we consider the win to loss ratio, the algorithm second in performance to 2SARC1 is CPAR, followed by boostingR and 2SARC2.

4.1.1 Statistical Analysis

The results presented in Table 5 give some insight in the performance of the algorithms. However, those results do not provide enough support for drawing a strong conclusion in favour or against any of the studied methods. There is no overall dominance over the entire range of datasets.

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

Demsar discusses in [11] the issue of multiple hypothesis testing and recommends the use of several statistical procedures for this problem. Following Demsar's recommendation, we first tested if there is any significant difference among the classification methods studied. Demsar recommends the use of Friedman test to compare several classifiers on multiple datasets.

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

- find r_i^j - the rank of the algorithm j on the i^{th} dataset;
- compute the average rank R of alg. j: $R_j = \frac{1}{N}\sum_i r_i^j$
- the null hypothesis states that all algorithms have the same average rank;
- compute the Friedman statistic:

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

- if chi_F^2 exceeds the critical value we reject the null hypothesis, otherwise we accept it;

- when the null hypothesis is rejected a post-hoc test is used to determine the nature of the difference.

By applying Friedman test [11] we concluded that there is a significant difference among the methods. Since the null hypothesis is rejected we have to proceed with further analysis to better understand the behaviour of the classification algorithms. We are interested in the performance of our proposed technique. Thus, we perform a series of Wilcoxon signed ranked tests between our best method (2SARC1) and the other classification methods. Friedman test and Wilcoxon tests are non-parametric tests, they do not make any assumptions about the distributions of the values.

For single-label classification results Friedman test finds significant difference among the classification methods studied. The only strong conclusion that we can draw when paired Wilcoxon signed ranked tests are performed is that 2SARC1 performs significantly better than C4.5 and Ripper. The mixture of wins and losses when compared to the other algorithms (see Table 6) makes it impossible to single out one of the algorithms as the best one.

4.2 Multi Label Classification

Associative classifiers are more suitable for categorical data, rather than numerical. Therefore, text classification is a good application to study the performance of our method.

Most of the research in text categorization comes from the machine learning and information retrieval communities. Rocchio's algorithm [15] is the classical method in information retrieval, being used in routing and filtering documents. Researchers tackled the text categorization problem in many ways. Classifiers based on probabilistic models have been proposed starting with the first presented in literature by Maron [20] and continuing with naïve-Bayes [17] that proved to perform well. ID3 and C4.5 are well-known packages whose cores are making use of decision trees to build automatic classifiers [10]. K-nearest neighbor (k-NN) is another technique used successfully in text categorization [27]. Another method to construct a text categorization system is by an inductive rule learning method. This type of classifiers is represented by a set of rules in disjunctive normal form that best cover the training set [3]. In the last decade neural networks and support vector machines (SVM) were used in text categorization and they proved to be powerful tools [16].

We used the *ModApte* version of Reuters-21578 text collection [23] as benchmark. This split leads to a corpus of 12,202 documents consisting of 9,603 training documents and 3,299 testing documents, and is the most used split in the literature. We tested our classifiers on the ten most populated categories with the largest number of documents assigned to them in the training set. On these documents we performed stopword elimination but no stemming.

For evaluating the effectiveness of our system we used F1 measure and precision/recall breakeven point [2]. To report the performance over multiple classes we used micro-average and weighted average.

In our experiments, we ran our algorithms with support values of 10%, 15% and 20% and we reported the best result. The minimum confidence was set to 50% and the confidence margin was set at 10%. Our proposed technique used a split ratio of 80/20 between trainStage 1 and trainStage2 sets. 2SARC2 technique used 10, 25, 50 or 100 rules per class. The algorithm used for the second stage was a backpropagation algorithm with 1 hidden layer. The number of neurons in the hidden layer was the average number between the input and output neurons.

Table 7 shows the performance for several well-known text categorizers and our algorithms. The evaluation is done using precision/recall break-even point. Results are presented by category for the ten most populous categories in Reuters collection. Except for our algorithms, the results are presented as reported in [16]. Given that the micro-average value for the other algorithms was computed on a different set of classes we used a weighted average scheme to approximate the micro-average. The formula for weighted average is:

$$WA = \frac{\sum_{i=1}^{N} \frac{test_i}{\sum_{j=1}^{N} test_j} \times BEP_i}{N} \quad (2)$$

where N stands for the number of classes and $test_j$ represents the number of test examples in class j.

SVM performs best for 6 out of 10 classes in the Reuters collection. 2SARC1 wins in 3 classes, while 2SARC2 gets the best breakeven point for *trade* class. SVM ranks first based on weighted-average and macro-average, followed closely by 2SARC1. *Corn* and *wheat* are two categories that are difficult to be learned by most classifiers due to their unique characteristics in this collection; both of them highly overlap with *grain* category. Winning overall in this categories is a good indication for our algorithm that the new automated scoring is able to pick up data characteristics that would have been missed by a static scoring scheme.

Table 8 shows the count of wins, losses and ties over the 10 classes in Reuters for 2SARC1 when compared to the other algorithms. 2SARC1 losses on 6 categories to SVM. 2SARC1 outperforms the rest of the algorithms winning in at least 7 out of the 10 classes.

4.2.1 Statistical Analysis

We performed the same statistical analysis as discussed in Section 4.1.1 to the text classification results. Friedman's test indicates that the methods evaluated are not equal. When the Wilcoxon tests are applied to the results of text classification for pairwise comparisons between 2SARC1

Table 7. Precision/Recall-breakeven point on ten most populated Reuters categories

category	# rec	Bayes	Rocchio	C4.5	k-NN	SVM	ARC	2SARC1	2SARC2
acq	719	91.5	92.1	85.3	92.0	**95.2**	89.9	93.7	81.5
corn	56	47.3	62.2	87.7	77.9	85.2	82.3	**88.4**	83.3
crude	189	81.0	81.5	75.5	85.7	**88.7**	77.0	83.0	83.5
earn	1087	95.9	96.1	96.1	97.3	**98.4**	89.2	95.4	87.2
grain	149	72.5	79.5	89.1	82.2	**91.8**	72.1	91.1	87.9
interest	131	58.0	72.5	49.1	74.0	**75.4**	70.1	72.8	73.9
money-fx	179	62.9	67.6	69.4	78.2	75.4	72.4	**78.7**	73.8
ship	89	78.7	83.1	80.9	79.2	**86.6**	73.2	83.1	77.3
trade	118	50.0	77.4	59.2	77.4	77.3	69.7	84.9	**86.7**
wheat	71	60.6	79.4	85.5	76.6	85.7	86.5	**88.7**	85.3
weighted-avg		84.25	87.94	85.14	89.68	**92.15**	84.12	90.61	83.56
macro-avg		65.21	79.14	77.78	82.05	**86.01**	78.24	**86.0**	82.0

Table 8. Method 2SARC1 compared to the rest of the algorithms on Reuters collection; (*) indicates statistical significant difference

	wins	losses	ties
2SARC1 vs. Bayes*	9	1	0
2SARC1 vs. Rocchio*	8	1	1
2SARC1 vs. C4.5*	9	1	0
2SARC1 vs. kNN	7	3	0
2SARC1 vs. SVM	4	6	0
2SARC1 vs. ARC*	10	0	0
2SARC1 vs. 2SARC2	7	3	0

and the rest of the algorithms, we can conclude the following: 2SARC1 is significantly better than Bayes, Rocchio, C4.5, ARC at a significance level of 0.05. When compared with SVM, the test can not reject the null hypothesis, thus it can be stated that 2SARC1 and SVM perform statistically similar on the Reuters dataset.

5 Discussion

Rule-based classification systems classify a new instance based on a set of rules that apply to this new object. In previous works, the scoring schemes under which the system takes a classification decision are predefined. In this paper we proposed a two-stage classification method. Our system (2SARC) learns automatically the scoring scheme in the second stage. In addition, we investigate two techniques. 2SARC1 learns the scoring scheme from class features while 2SARC2 learns it from rule features.

The proposed system is versatile, it shows good performance over a large and different set of applications. We tested our method for single-label and multiple-label classification, using both small and large datasets. In addition, association rule mining is a mature domain with fast algorithms that can handle large dimensionality, thus making the classification rule discovery fast and reliable.

2SARC1 performs best on most UCI datasets or it ranks very close to the best as Table 5 shows. There are only 3 datasets where the performance is much lower than the best (*labor*, *waveform*, *zoo*). *Labor* and *zoo* are very small datasets with 57 and, respectively, 101 examples. Small datasets may hinder the performance of our system given that it needs enough examples to train the models for the two stages. *Waveform* is a dataset where all the initial attributes are numerical. Association rules are more suitable for nominal attributes, thus for this particular dataset the mining has to rely on the discretization process. The influence of the discretization process on the performance of association-rule based systems needs further study.

On Reuters dataset 2SARC1 has good overall performance, and excellent results for three categories (*corn*, *wheat* and *money-fx*). *Corn* and *wheat* are two categories that are highly overlapped with *grain*. They are hard to learn by other classifiers (including the state of the art SVM) due to this characteristic. On top of the overlapping property they are also two small classes, thus adding the imbalance issue to the classification process. By automatically learning the scoring scheme our system can pick up on existing relations between classes that a pre-defined scoring scheme would miss (especially if they solved the multi-label problem as binary subtasks).

Our method has two stages, each stage employing a classification algorithm. It may appear that the training time is higher than for each algorithm applied alone or for other algorithms. This is not necessarily the case, as each stage uses only a partition of the data.

2SARC2 shows lower performance than 2SARC1. This may be due to the limit that we enforce on the number of rules to be used as features. This constraint is used in the second stage as neural networks do not handle a large number of inputs well. A direction to be investigated further is the use of other classification methods in the second stage that would handle better the large dimensionality. Incorporating all the rules discovered could be of benefit to 2SARC2 as seen for the *iris* dataset where 2SARC2 performs best. The set of rules is very small for this dataset thus making possible the use of all rules in the second stage.

6 Conclusions

Rule-based classifiers use predefined weighted voting schemes to combine the class predictions of the applicable rules. By contrast, the methods described in this paper automatically learn the scoring scheme. We achieve this by developing a two-stage system, with a layer of feature definitions interposed between the output of the first learning model and the input of the second. Our two stage classification with class-based features (2SARC1) shows a good performance both for UCI datasets and text classification, under rigorous statistical analysis. The most accurate text classifier, particularly when multi-label classification is involved, is by far SVM. Our two stage associative classifier 2SARC1 is the first to equal its performance on the commonly used and challenging Reuters testset.

References

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

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

[3] C. Apte, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

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

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

[6] C. Borgelt. Backpropagation software. http://fuzzy.cs.uni-magdeburg.de/∼borgelt/mlp.html.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[23] The Reuters-21578 text categorization test collection. http://www.research.att.com/˜lewis/reuters21578.html.

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

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

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

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

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

[29] O. R. Zaïane and M.-L. Antonie. On pruning and tuning rules for associative classifiers. In *Proc. of Int'l Conf. on Knowledge-Based Intelligence Information & Engineering Systems (KES'05)*, pages 966–973, 2005.