Chapter # 1

Classification Using Association Rules: Weaknesses and Enhancements

Bing Liu, Yiming Ma, and Ching-Kian Wong

Key words: Classification, association rules, combining multiple models

Abstract: Existing classification and rule learning algorithms in machine learning mainly use heuristic/greedy search to find a subset of regularities (e.g., a decision tree or a set of rules) in data for classification. In the past few years, extensive research was done in the database community on learning rules using exhaustive search under the name of association rule mining. The objective there is to find all rules in data that satisfy the user-specified minimum support and minimum confidence. Although the whole set of rules may not be used directly for accurate classification, effective and efficient classifiers have been built using the rules. This paper aims to improve such an exhaustive search based classification system CBA (*Classification Based on Associations*). The main strength of this system is that it is able to use the most accurate rules for classification. However, it also has weaknesses. This paper proposes two new techniques to deal with these weaknesses. This results in remarkably accurate classifiers. Experiments on a set of 34 benchmark datasets show that on average the new techniques reduce the error of CBA by 17% and is superior to CBA on 26 of the 34 datasets. They reduce the error of the decision tree classifier C4.5 by 19%, and improve performance on 29 datasets. Similar good results are also achieved against the existing classification systems, RIPPER, LB and a Naïve-Bayes classifier.

1. INTRODUCTION

Building effective classification systems is one of the central tasks of data mining and machine learning. Past research has produced many techniques (e.g. decision trees [Q92], rule learning [CN89, C95] and Naïve-Bayes classification [DH73]) and systems (e.g., C4.5 [Q92], CN2 [CN89], and RIPPER [C95]). The existing techniques are, however, largely based on heuristic/greedy search. They aim to find only a *subset* of the regularities (e.g., a decision tree or a set of rules) that exist in data to form a classifier.

In the past few years, the database community studied the problem of rule learning extensively under the name of association rule mining [AS94]. The study there is focused on using exhaustive search to find all rules in data that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf). Many efficient algorithms are reported. The most important one is the Apriori algorithm in [AS94].

Clearly, both the computational complexity and the number of rules produced grow exponentially for association rule mining. minsup holds the key for the success of the model. In the next section, we will give an overview of association rule mining.

Although the complete set of rules may not be directly used for accurate classification, effective and efficient classifiers have been built using the rules, e.g., CBA [LHM98], LB [MW99] and CAEP [DZWL99], GAC [LL00], and ADT [WZH00]. The major strength of such systems is that they are able to use the most accurate rules for classification because their rule learners aim to find all rules. This explains their good performance in general. However, they also have some weaknesses, which they inherit from association rule mining.

1. Traditional association rule mining uses only a single minsup in rule generation, which is inadequate for unbalanced class distribution (this will be clear later) [LHM99].
2. Classification data often contains a huge number of rules, which may cause combinatorial explosion. For many datasets, the rule generator is unable to generate rules with many conditions, while such rules may be important for accurate classification.

This paper aims to improve the CBA system (*Classification Based on Associations*) by dealing directly with the above two problems. It tackles the first problem by using *multiple class minsups* in rule generation (i.e., each class is assigned a different minsup), rather than using only a single minsup as in CBA. This results in a new version of the system called CBA(2). CBA(2) is included as a new function (using multiple minimum class supports) in the CBA data mining system, which can be downloaded from http://www.comp.nnus.edu.sg/~dm2.

Experiments on a set of 34 benchmark problems from UCI machine learning repository [MM96] show that on average CBA(2) achieves lower error rate than CBA, C4.5 (tree and rules), the Naïve-Bayes classifier (NB), LB and RIPPER (CAEP, GAC and ADT are not available for comparison). CBA(2) is also efficient and scales well on large datasets, which is a key feature of association rule mining [AS94].

The second problem is more difficult to deal with directly as it is caused by exponential growth of the number of rules. We deal with it indirectly. We try to find another classification technique that is able to help when some rules from CBA(2) are not accurate. The decision tree method [Q92] is a clear choice because decision trees often go very deep, i.e., using many conditions. We then propose a technique to combine CBA(2) with the decision tree method as in C4.5, and the Naïve-Bayes method (NB). NB is included because it comes free (the probability estimates required by NB can be obtained from the rules of CBA(2) with no extra computation) and it also performs very well on many datasets [DP97].

The basic idea of our combination technique is to use the rules of CBA(2) to segment the training data and then select the classifier that has the lowest error rate on each segment to classify the future cases falling into the segment. This approach is proposed because we believe that different segments of the data may have different characteristics, and are thus more suitable for different classifiers. It is also well known that different classifiers have different strengths.

The resulting composite method performs remarkably well. It reduces the error of CBA(2) by 14% and is superior to CBA(2) on 28 of the 34 datasets. It reduces the error of C4.5 tree (or rules) by 18% (or 19%), and improves performance on 27 (or 29) datasets. Similar good results are also achieved against CBA, RIPPER, NB, and LB. The composite method's results are comparable with those of C4.5 with boosting [FS96]. The method is also fast because the decision tree and the NB methods are very efficient. They do not add much to the running time of CBA(2).

2. RELATED WORK

Our CBA(2) system is different from existing learning or classification systems, such as C4.5 [Q92], CN2 [CN89], and RIPPER [C95]. These systems all use heuristic search to learn a subset of the regularities in data to build a classifier. CBA(2) (or CBA)'s rule learner is based on exhaustive search and aims to find all rules. Its rule selection technique in classifier building is related to the traditional *covering method* [M80], which is used in, e.g., IREP [FW94] and RIPPER [C95]. The covering method works as follows: A rule set is built in a greedy fashion, one rule at a time. After a rule is found, all examples covered by the rule are removed. This process repeats until there are no training examples left. The key difference between the method in CBA(2) and the covering method is in the rules that they use. CBA(2) learns all rules from the entire training data using exhaustive search, while each rule in the covering method is learnt using a heuristic method from the remaining data after the examples covered by previous rules are deleted. Although in SLIPPER [CS99], the covered examples are not deleted, they are assigned lower weights in the subsequent rule learning using boosting. Rules learnt from these existing methods may not reflect the true regularities of the original data. Further, since they all use heuristic methods to find rules, many high quality rules may not be found.

Several researchers, e.g., Rymon [R96], Webb [W93], and Murphy and Pazzani [MP94] have tried to build classifiers by performing extensive search. None of them uses the association mining technique to find all rules with minsup and minconf constraints.

In data mining, Bayardo [B97] uses an association rule miner to generate high confident rules (greater than 90%). Ali, Manganaris & Srikant [AMS97] uses an association rule miner to find rules that can describe individual classes. Both works are not concerned with building classifiers.

LB [MW99], CAEP [DZWL99], GAC [LL00] and ADT [WZH00] are some other existing classification systems using association rules apart from CBA. LB and CAEP are based on rule aggregation rather than rule selection as in CBA and CBA(2). They also do not combine with other methods. GAC uses a RDBMS system to help build classifiers efficiently. ADT studies using association rules to build decision trees.

Regarding combining multiple learning methods, Quinlan [Q94] reports a composite method for predict continuous values (rather than discrete classes, which is our task) using instance-based and model-based methods.

Kohavi [K96] reports a hybrid system of NB and decision tree. The algorithm is similar to the decision tree-building algorithm, except that the leaf nodes created are NB classifiers instead of nodes predicting a single class. It uses a validation set to determine when NB will form a leaf. Our method is different. We do not integrate different techniques at the algorithm level, but only compare the error rates of individual classifiers on the training data to decide when each classifier should be used.

Littlestone & Warmuth [LW89] proposed several weighted majority algorithms for combining different classifiers. [CS93] adapts their methods and learns the weights using a validation set. It also has a technique that learns an arbiter to arbitrate among predictions generated by different classifiers. However, the evaluation was only done on a few datasets. Similar work in this direction includes stacked generation [W92], and combining multiple rule sets using Bayesian utility theory [AP96]. Our method is simpler. When to use which classifier is determined using the training data. No combination or arbitration is done at testing time.

3. ASSOCIATION RULE MINING FOR CLASSIFICATION

Association rule mining was first proposed to find all rules in a basket data (also called *transaction data*) to analyze how items purchased by customers in a shop are related (one data record per customer transaction). The model is as follows [AS94]:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D be a set of transactions (the dataset), where each transaction d (a data record) is a set of items such that $d \subseteq I$. An *association rule* is an implication of the form, $X \to Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \to Y$ holds in the transaction set D with *confidence c* if $c\%$ of transactions in D that support X also support Y. The rule has *support s* in D if $s\%$ of transactions in D contains $X \cup Y$.

Given a set of transactions D (the dataset), the problem of mining association rules is to discover all rules that have support and confidence greater than the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*). Association rule mining does not have a fixed target. That is, any item can appear on the right-hand-side or the left-hand-side of a rule.

One of the efficient algorithms for mining association rules is the Apriori algorithm given in [AS94]. It generates all rules in two steps:

1. Find all the frequent itemsets that satisfy minsup.
2. Generate all the association rules that satisfy minconf using the frequent itemsets.

An *itemset* is a set of items. A *frequent* itemset is an itemset that has support above minsup. Mining of frequent itemsets is done in a level-wise fashion. Let k-itemset denote an itemset of k items. At level 1, all frequent 1-itemsets are found. At level 2, all frequent 2-itemsets are found and so on. If an itemset is not frequent at level k-1, it is discarded as any addition of items to the set cannot be frequent (this is called the *downward closure property*). At level k, all potentially frequent itemsets (*candidate itemsets*) are generated from frequent itemsets at level k-1. To determine which of the candidate itemsets are actually frequent, the algorithm makes one pass over the data to count their supports. After all frequent itemsets are found, it generates rules, which is relatively simple.

Clearly, both the computational complexity and the number of rules produced grow exponentially for association rule mining. In the worst case, the number of itemsets is 2^n, where n is the total number of items in the data. For example, in a supermarket situation, n is the number of items sold in the supermarket, which can be thousands or more. minsup makes association rule mining practical. However, for classification, we often need to set minsup very low. This can cause combinatorial explosion.

Mining association rules for classification: The Apriori algorithm finds association rules in a transaction data of items. A classification dataset, however, is normally in the form of a relational table, which is described by a set of distinct attributes (discrete and continuous). Each data record (or example) is also labeled with a class. The table form data can be converted to transaction data as follows: As association rule mining does not handle continuous values but only individual items, we first discretize each continuous attribute into intervals. After discretization, we can transform each data record to a set of (*attribute*, *value*) pairs and a class label, which is in the transaction form. A (*attribute*, *value*) pair is an *item*. Discretization of continuous attributes will not be discussed in this paper, as there are many existing algorithms for the purpose (e.g., [FI93]; [DKS95]).

To generate all rules for classification, we also need to make some modifications to the Apriori algorithm because a dataset for classification has a fixed target, the class attribute.

Thus, we only need to generate those rules of the form $X \rightarrow c_i$, where c_i is a possible class. We call such association rules the *class association rules* (CARs).

It is easy to modify the Apriori algorithm to generate CARs. We will not discuss it here (see [LHM98] for details). For the purpose of classification, rule pruning may be performed to remove those overfitting rules, which are rules with many conditions and covering only a few data records.

Pruning the rules: In CBA, pruning is done using the pessimistic error based method in C4.5. It prunes a rule r as follows: If rule r's pessimistic error rate is higher than the pessimistic error rate of rule r^- (obtained by deleting one condition from the conditions of r), then rule r is pruned. See [Q92] for the computation of the method.

4. CLASSIFIER BUILDING IN CBA

After all rules (CARs) are found, a classifier is built using the rules. Clearly, there are many possible methods to build a classifier from the rules. In CBA, a set of high confidence rules is selected from CARs to form a classifier (this method is also used in CBA(2)). The selection of rules is based on a total order defined on the rules.

Definition: Given two rules, r_i and r_j, $r_i \phi r_j$ (also called r_i precedes r_j or r_i has a higher precedence than r_j) if

1. the confidence of r_i is greater than that of r_j, or
2. their confidences are the same, but the support of r_i is greater than that of r_j, or
3. both the confidences and supports of r_i and r_j are the same, but r_i is generated earlier than r_j.

Let R be the set of CARs, and D the training data. The basic idea of the classifier-building algorithm in CBA is to choose a set of high precedence rules in R to cover D. This method is related to the traditional covering method. However, there is a major difference, as we will see later in the related work. A CBA classifier is of the form:

$$\langle r_1, r_2, \ldots, r_n, default\_class \rangle$$

where $r_i \in R$, $r_a \phi r_b$ if $b > a$. In classifying an unseen case, the first rule that satisfies the case classifies it. If no rule applies to the case, it takes the default class (*default_class*). A simple version of the algorithm for building such a classifier is given in Figure 1.

```
R = sort(R); /* according the precedence ϕ */
for each rule r ∈ R in sequence do
   if there are still training examples in D AND r classifies
         at least one example correctly then
      delete all training examples covered by r from D;
      add r to the classifier
   end
end
add the majority class as the default class to the classifier.
```

Figure 1. A simple classifier-building algorithm

This algorithm can be easily implemented by making one pass through the training data for each selected rule. However, this is extremely inefficient for large datasets. [LHM98] presents an efficient algorithm. It makes at most two passes through the data.

5. IMPROVING CBA

5.1 Using Multiple Minimum Class Supports

The most important parameter in association rule mining is the minsup. It controls how many rules and what kinds of rules are generated. The CBA system follows the original association rule model and uses a single minsup in its rule generation. We argue that this is inadequate for mining of CARs because many practical classification datasets have uneven class frequency distributions. Using a single minsup will result in one of the following two problems:

1. If we set the minsup value too high, we may not find sufficient rules of infrequent classes.
2. If we set the minsup value too low, we will find many useless and overfitting rules for frequent classes.

More details about the problems can be found in [LHM99]. Let us use an example to illustrate. Suppose a dataset has 2 classes, *Y* and *N*, with *freqDistr*(*Y*) = 98% and *freqDistr*(*N*) = 2%. If we set minsup = 3%, we will not find any rule of class *N*. To solve the problem, we need to lower down the minsup. Suppose we set minsup = 0.2%. Then, we may find many overfitting rules for class *Y* because minsup = 0.2% is too low for class *Y*. To solve the two problems, CBA(2) adopts the following (*multiple minimum class supports*):

$\mathbf{minsup_i}$: For each class c_i, a different *minimum class support* is assigned. The user only gives a total minsup, denoted by *t_minsup*, which is distributed to each class according to their class distributions as follows:

$$\mathrm{minsup_i} = \mathrm{t\_minsup} \times \mathrm{freqDistr(c_i)}$$

The formula gives frequent classes higher minsups and infrequent classes lower minsups. This ensures that we will generate sufficient rules for infrequent classes and will not produce too many overfitting rules for frequent classes.

Regarding minconf, it has less impact on the classifier quality as long as it is not set too high since we always choose the most confident rules. Experiments on the 34 datasets show that on average CBA(2)'s error rate is lower than that of CBA, C4.5 (tree & rules), RIPPER, NB, and LB.

5.2 Seeking Help From Other Techniques

As we mentioned earlier, for many highly correlated datasets the rule generator is unable to generate rules with many conditions (i.e., long rules) due to combinatorial explosion. When such long rules are important for classification, our classifiers suffer. Here, we propose a combination technique to reduce the effect of the problem. The aim is to combine CBA(2) with a method that is able to find long rules. Clearly, the decision tree method [Q92] is a natural choice because decision trees often go very deep, i.e., using many conditions. Experiment results show that this strategy is extremely effective. In our implementation, we also include the Naïve-Bayes method (NB) as NB comes free from CBA(2) (the probabilities needed by NB are all contained in the 1-condition rules of CBA(2)).

The decision tree and NB methods will not be discussed here, as they are well known [Q92] [DH73, DP97]. Both methods are also very efficient.

The proposed combination method is based on the competition of different classifiers on different segments of the training data. For easy understanding, the algorithm below is presented as a combination of three classifiers. In fact, the framework can be used to combine any number of classifiers (see the experiments section). The key idea is to use one classifier to segment the training data, and then choose the best classifier to classify each segment.

Let A be the classifier built by CBA(2), T be the decision tree built by C4.5, and N be the Naïve-Bayes classifier. We use the rules in A from CBA(2) to segment the data [1]. For the set of training examples covered by a rule r_i in A (*default_class* is treated as a rule with no condition), we choose the classifier that has the lowest error on the set of examples to replace r_i. That is, if r_i has the lowest error, we keep r_i. If T has the lowest error, we use T to replace r_i. If r_i is replaced by T, then in testing when a test case satisfies the conditions of r_i, it is classified by T instead of r_i. The same applies to N. The algorithm is given in Figure 2.

1 construct the three classifiers, A, T, N;
2 **for** each training example e **do**
3   find the first rule r_i in A that covers e
4   **if** r_i classifies e wrongly **then** $Error_i = Error_i + 1$ **end**
5   **if** T classifies e wrongly **then** $Error_{i,T} = Error_{i,T} + 1$ **end**
6   **if** N classifies e wrongly **then** $Error_{i,N} = Error_{i,N} + 1$ **end**
7 **endfor**
8 **for** each rule r_i ($X \rightarrow c_j$) in R **do**   /*X is the set of conditions */
9   **if** $Error_i \leq Error_{i,T}$ and $Error_i \leq Error_{i,N}$ **then**
10   keep r_i
11   **elseif** $Error_{i,T} \leq Error_{i,N}$ **then** use $X \rightarrow$ (use T) to replace r_i
12   **else** use $X \rightarrow$ (use N) to replace r_i
13   **end**
14 **endfor**

Figure 2. The combination algorithm

From line 3-6, we compute the number of errors made by r_i, T, and N on the training examples covered by each r_i. $Error_i$, $Error_{i,T}$ and $Error_{i,N}$ are initialized to 0. From line 8-14, we use T (or N) to replace r_i if T (or N) results in fewer errors on the training examples covered by r_i. $X \rightarrow$ (use T) means that in testing if a test case satisfies X (the conditions of r_i), T will be used to classify the case.

The algorithm is $O(|A|M)$, where $|A|$ is the number of rules in A, and M is the number of training examples. M dominates the computation as it is normally much larger than $|A|$.

6. EXPERIMENTS

We now compare the classifiers built by CBA(2), CBA, C4.5 (tree and rules) (Release 8), RIPPER, NB, LB, and various combinations of CBA(2), C4.5 and NB. RIPPER is a rule based classifier, while LB is a Bayesian classifier using association rules. The evaluations are done on 34 datasets from UCI ML Repository [MM96]. We also used Boosted C4.5 (the code is

[1] We also tried to use other classifiers, e.g., C4.5, NB and RIPPER to segment the data, but the results are not satisfactory. We believe the reason is that these systems do not have the obvious problem of short-rules of CBA(2).

obtained from Zijian Zheng [ZW99]) in our comparison. C4.5, RIPPER and LB are obtained from their authors. We ran the systems using their default settings. We could not compare our system with existing classifier combination methods (see the Related Work section) as we were unable to obtain the systems.

In all the experiments with CBA(2), *minconf* is set to 50%. For *t_minsup*, it is more complex. *t_minsup* has a strong effect on the quality of the classifier produced. In general, lower *t_minsup* gives more accurate classifiers. However, for certain datasets, higher *t_minsup*s are better. From our experience, once *t_minsup* is lowered to 1-2%, the classifier built is already very accurate. In the experiment results reported below, we set *t_minsup* to 1%.

Since classification data often contains a huge number of rules, which may cause combinatorial explosion, we set a limit of 80,000 on the total number of rules. 24 of the 34 datasets used cannot be completed within this limit. However, the classifiers constructed are already quite accurate. In fact, when the limit reaches 60,000 (we tried different limits), the accuracy of the resulting classifiers starts to stabilize.

Discretization of continuous attributes is done using the entropy method in [FI93]. The code is taken from *MLC++* [KJLMP94]. No discretization is applied to the data when running C4.5, boosted C4.5 and RIPPER. In all experiments, CBA, C4.5, RIPPER, LB, and Boosted C4.5 parameters have their default values. Experiment results are shown in Table 1. The error rates on the first 26 datasets are obtained from 10-fold cross-validation, while on the last 8 datasets they are obtained from the test sets provided in UCI Repository (which are commonly used in evaluation). All the composite methods involving C4.5 uses C4.5 tree due to its efficiency.

Table 1: Experiment Results

			1	2	3	4	5	6	7	8	9	10	11	12	13
			CBA(2)	**CBA**	**C4.5 tree**	**C4.5 rules**	**RIPPER**	**NB**	**LB**	**C4.5 + NB**	**CBA(2) + NB**	**CBA(2) + C4.5**	**CBA(2) + C4.5 + NB**	**Boosted C4.5**	**CBA(2)+ Boosted C4.5**
			err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)	err (%)
1	anneal	CV-10	2.1	3.6	7.5	5.2	4.6	2.7	3.6	7.6	2.1	2.2	2.8	4.3	1.4
2	australian	CV-10	14.6	13.4	14.8	15.3	15.2	14.0	13.5	14.6	14.2	14.2	14.2	15.9	13.5
3	auto	CV-10	19.9	27.2	17.6	19.9	23.8	32.1	28.1	19.0	18.5	20.9	18.5	15.1	18.5
4	breast-w	CV-10	3.7	4.2	5.6	5.0	4.0	2.4	2.7	5.6	3.2	3.0	2.4	3.1	2.2
5	cleve	CV-10	17.1	16.7	21.5	21.8	21.1	17.1	17.1	21.8	16.8	17.5	17.5	20.5	18.2
6	crx	CV-10	14.6	14.1	15.0	15.1	14.6	14.6	12.9	15.1	14.5	14.2	14.3	15.7	13.6
7	diabetes	CV-10	25.5	25.3	26.1	25.8	25.3	24.4	24.4	25.7	24.9	24.8	22.0	29.4	25.2
8	german	CV-10	26.5	26.5	28.4	27.7	27.8	24.6	24.7	28.6	25.5	25.2	24.9	28.5	24.8
9	glass	CV-10	26.1	27.4	30.4	31.3	35.0	29.4	30.8	28.5	26.5	29.5	29.5	24.7	28.0
10	heart	CV-10	18.1	18.5	21.8	19.2	19.6	18.1	18.2	21.1	18.1	17.4	17.0	20.7	20.0
11	hepatitis	CV-10	18.9	15.1	18.2	19.4	17.5	15.0	15.6	18.2	17.5	16.2	16.2	17.5	18.2
12	horse	CV-10	17.6	18.7	14.7	17.4	14.7	20.6	20.7	13.1	17.4	17.1	17.1	18.7	16.0
13	hypo	CV-10	1.0	1.7	0.7	0.8	0.8	1.5	1.6	1.0	1.1	0.9	0.9	1.1	0.9
14	ionosphere	CV-10	7.7	8.2	10.5	10.0	11.4	12.0	8.8	10.8	8.3	8.3	8.3	6.8	8.3
15	iris	CV-10	5.3	7.1	4.7	4.7	5.3	6.0	5.3	4.7	5.3	2.7	2.7	5.3	2.7
16	labor	CV-10	13.7	17.0	22.3	20.7	16.5	14.0	12.3	19.0	13.7	12.0	12.0	8.3	8.3
17	led7	CV-10	28.1	27.8	30.5	26.5	30.8	26.7	26.6	25.7	26.7	27.0	26.1	26.2	25.1
18	lymph	CV-10	22.1	19.6	23.8	26.5	20.8	24.4	19.7	23.2	22.1	17.6	17.6	18.3	16.9
19	pima	CV-10	27.1	27.6	25.8	24.5	26.3	24.5	24.7	25.4	26.1	22.7	22.0	27.3	24.1
20	sick	CV-10	2.8	2.7	1.1	1.5	1.9	3.9	3.0	1.1	3.0	1.9	1.9	1.3	1.7
21	sonar	CV-10	22.5	21.7	28.4	29.8	27.9	23.0	24.0	27.9	22.5	23.0	21.6	20.2	23.0
22	tic-tac-toe	CV-10	0.4	0.1	13.8	0.6	2.4	30.1	32.1	14.5	0.2	0.2	0.2	3.3	0.2
23	vehicle	CV-10	31.0	31.3	28.5	27.4	31.4	40.1	30.5	28.4	31.0	29.8	34.3	24.3	29.0
24	waveform21	CV-10	20.3	20.6	22.8	21.9	20.5	19.3	17.5	23.0	20.0	17.7	17.7	18.2	16.0
25	wine	CV-10	5.0	8.4	7.3	7.3	8.5	9.5	1.7	7.8	5.0	5.0	5.0	4.0	4.5
26	zoo	CV-10	3.2	5.4	7.8	7.8	11.0	13.7	5.8	2.0	2.9	2.9	2.9	0.0	2.9
27	Adult	test	16.7	14.4	14.6	14.1	15.6	15.8	14.2	14.0	14.0	14.9	13.9	16.2	16.2
28	Chess	test	2.0	1.9	0.5	1.1	1.9	12.9	7.2	0.5	2.2	0.3	0.3	0.3	0.2
29	DNA	test	10.3	15.4	7.3	6.9	8.3	6.6	7.3	6.7	7.5	8.3	7.9	5.3	6.2
30	Letter	test	30.0	29.5	12.3	13.7	15.2	25.0	16.1	12.3	24.5	14.0	13.8	5.2	8.9
31	Satimage	test	14.6	15.9	14.6	14.8	15.1	18.0	13.5	14.4	14.5	14.0	14.1	10.3	12.3
32	Segment	test	6.0	6.8	6.0	6.6	7.8	6.2	5.6	5.7	4.6	3.8	3.8	3.1	4.2
33	Soybean Big	test	7.5	7.5	10.5	9.6	9.2	6.1	8.8	10.5	6.6	7.0	7.0	6.1	7.0
34	Waveform40	test	24.3	24.4	29.6	30.5	26.8	21.7	21.4	29.2	23.1	23.2	23.2	20.9	23.1
	Average		*14.9*	*15.5*	*16.0*	*15.6*	*15.8*	*16.9*	*15.3*	*15.5*	*14.2*	*13.5*	*13.3*	*13.3*	*13.0*
			won-lost-tied: CBA(2)+C4.5+NB vs the other methods												
			28-5-1	*26-8-0*	*27-7-0*	*29-5-0*	*31-3-0*	*24-10-0*	*23-11-0*	*26-8-0*	*21-7-6*	*14-4-16*			

Error rate comparison: For each dataset, columns 1-11 in Table 1 show the error rates of CBA(2), CBA, C4.5 tree, C4.5 rules, RIPPER, NB, LB, C4.5+NB (C4.5 tree combined with NB), CBA(2)+NB, CBA(2)+C4.5 and CBA(2)+C4.5+NB respectively. From the table we see that on average (last row) the error rate of CBA(2) is lower than every other individual method. Its won-lost-tied record against CBA is 19-13-2, and against C4.5 tree (or rules) is 20-12-2 (or 22-11-1). It also performs better than RIPPER and NB, and has a similar performance as LB.

It is clear that over the 34 datasets, the composite methods are in general superior to individual methods. CBA(2)+C4.5+NB gives the lowest error rate on average. It reduces the error of C4.5 tree (or rules) by 18% (or 19%) on average, and its won-lost-tied record against C4.5 tree (or rules) is 27-7-0 (or 29-5-0). It reduces the error of CBA(2) by 14%, and its won-lost-tied record against CBA(2) is 28-5-1. Similar good results are also achieved against CBA, RIPPER, NB and LB (see Table 1).

When the composite methods are compared, we see that the performance of CBA(2)+C4.5 is almost the same as CBA(2)+C4.5+NB. They are both better than CBA(2)+NB, which is better than C4.5+NB. These results suggest that CBA(2) plays the most important role in error reduction.

The fact that CBA(2)+C4.5 and CBA(2)+C4.5+NB perform almost equally well confirm our intuition that CBA(2)'s weakness is overcome by deep trees of C4.5. The *letter* dataset is a good example. CBA(2)'s error rate on the dataset is very high (30%) because it cannot generate rules with more than 3 conditions. The C4.5 tree, however, goes very deep using more than 10 conditions. When CBA(2) is combined with C4.5, C4.5's deep tree helps greatly. The error rate drops drastically to 14%.

Columns 12 and 13 give the error rates of boosted C4.5 and CBA(2)+boostedC4.5. We see that CBA(2)+C4.5+NB’s results are comparable to boosted C4.5, and its won-lost-tied record against boosted C4.5 is 18-15-1. Since boosted C4.5 is regarded as one of the best classifiers, we can say that CBA(2)+C4.5+NB is also among the best. CBA(2)+boostedC4.5 does not makes much improvement.

Table 2 (columns 1-10) shows the ratios of the error rate of CBA(2)+C4.5+NB vs. the other methods. We again can see that the combination method is superior.

Execution times: All the experiments are run on Sun Sparc I with 512MB of memory. Columns 11-12 in Table 2 show the execution times with each dataset. The average training time of CBA(2) in each fold of the 10-fold cross-validation is 13.33 seconds, while the classifier combining time for CBA(2)+C4.5+NB is only 2.21 seconds on average over the 34 datasets. In both cases, the datasets reside on disk. The execution times are reasonable. With a fixed in-memory rule limit, CBA(2) scales linearly with the size of the dataset, which is a feature of association rule mining.

7. CONCLUSION

This paper aims to improve an exhaustive search based classification system CBA. It first identified two problems or weaknesses of the system, i.e., single minsup, and not being able to generate long rules for many datasets. It then proposed two new techniques to deal with the problems. The first problem is dealt with by using multiple minimum class supports, while the second problem is dealt with by combining it with other classification methods, the decision tree method being particularly effective. The new combined system produces markedly better classifiers. On average over a set of 34 datasets, it outperforms CBA, C4.5, RIPPER, NB, and LB substantially, and has similar performances as boosted C4.5.

Table 2: Ratio of Error Rates, and Execution Times

			1	2	3	4	5	6	7	8	9	10	11	12
			CBA(2) + C4.5 + NB Vs										**Time**	
			CBA(2)	**CBA**	**C4.5 tree**	**C4.5 rules**	**RIPPER**	**NB**	**LB**	**C4.5 + NB**	**CBA(2) + NB**	**CBA(2) + C4.5**	**CBA(2)**	**Classifier Combining**
			ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	(sec)	(sec)
1	anneal	CV-10	1.319	.769	.369	.533	.606	1.045	.778	.365	1.319	1.259	8.52	0.11
2	australian	CV-10	.972	1.059	.959	.927	.932	1.011	1.053	.971	1.000	1.000	5.98	0.11
3	auto	CV-10	.928	.679	1.049	.928	.776	.575	.656	.971	1.000	.882	4.73	0.05
4	breast-w	CV-10	.659	.581	.436	.488	.604	1.004	.897	.437	.775	.811	0.66	0.01
5	cleve	CV-10	1.023	1.047	.813	.802	.829	1.021	1.021	.802	1.039	.999	3.80	0.01
6	crx	CV-10	.981	1.016	.955	.948	.978	.979	1.110	.951	.989	1.009	6.36	0.11
7	diabetes	CV-10	.864	.871	.844	.854	.871	.904	.904	.858	.885	.889	0.10	0.05
8	german	CV-10	.940	.940	.877	.899	.896	1.012	1.008	.871	.976	.987	6.63	0.22
9	glass	CV-10	1.130	1.076	.970	.942	.842	1.003	.958	1.034	1.111	1.000	0.17	0.05
10	heart	CV-10	.940	.920	.781	.886	.867	.939	.938	.807	.940	.979	0.88	0.01
11	hepatitis	CV-10	.858	1.074	.891	.836	.926	1.082	1.038	.892	.926	1.000	5.21	0.01
12	horse	CV-10	.972	.914	1.163	.983	1.163	.828	.828	1.309	.984	1.000	5.33	0.05
13	hypo	CV-10	.870	.512	1.243	1.088	1.145	.565	.540	.916	.813	.926	27.36	0.33
14	ionosphere	CV-10	1.073	1.007	.787	.826	.725	.690	.937	.763	1.000	1.000	4.11	0.06
15	iris	CV-10	.506	.377	.570	.570	.503	.446	.503	.575	.506	1.000	0.02	0.01
16	labor	CV-10	.876	.706	.538	.580	.727	.857	.973	.632	.878	1.000	0.11	0.01
17	led7	CV-10	.930	.940	.857	.986	.848	.979	.981	1.016	.979	.967	2.36	0.77
18	lymph	CV-10	.795	.896	.738	.663	.843	.720	.893	.759	.795	1.000	3.84	0.05
19	pima	CV-10	.812	.797	.853	.898	.837	.898	.890	.866	.845	.971	0.16	0.06
20	sick	CV-10	.664	.689	1.691	1.240	.964	.473	.628	1.632	.626	1.000	24.46	0.38
21	sonar	CV-10	.960	.995	.760	.724	.775	.937	.900	.775	.960	.940	6.05	0.01
22	tic-tac-toe	CV-10	.500	2.000	.014	.333	.083	.007	.006	.014	1.000	1.000	2.13	0.01
23	vehicle	CV-10	1.106	1.095	1.203	1.251	1.092	.856	1.124	1.209	1.107	1.151	8.23	0.22
24	waveform21	CV-10	.871	.858	.775	.807	.864	.914	1.009	.767	.886	.998	12.61	3.08
25	wine	CV-10	1.000	.595	.686	.686	.590	.526	2.994	.639	1.000	.998	2.58	0.01
26	zoo	CV-10	.909	.539	.373	.373	.265	.212	.500	1.455	1.000	1.000	5.64	0.01
27	Adult	test	.833	.963	.950	.984	.887	.877	.974	.989	.992	.933	94.49	33.34
28	Chess	test	.142	.149	.560	.248	.149	.022	.039	.596	.130	1.000	34.46	0.33
29	DNA	test	.771	.514	1.086	1.148	.960	1.202	1.094	1.191	1.057	.960	89.34	2.30
30	Letter	test	.459	.467	1.120	1.006	.905	.550	.856	1.122	.563	.987	44.13	29.61
31	Satimage	test	.966	.890	.966	.953	.937	.786	1.048	.979	.976	1.007	18.61	2.74
32	Segment	test	.631	.559	.628	.571	.484	.605	.676	.660	.829	1.000	9.38	0.38
33	Soybean Big	test	.941	.941	.669	.731	.762	1.143	.800	.667	1.067	1.000	11.30	0.44
34	Waveform40	test	.954	.951	.784	.761	.866	1.068	1.086	.795	1.007	1.000	3.61	0.06
	Average		*0.86*	*0.83*	*0.82*	*0.81*	*0.78*	*0.79*	*0.90*	*0.86*	*0.91*	*0.99*	*13.33*	*2.21*
			*** average ratios after the largest and smallest ratios are removed**											
			0.86	*0.82*	*0.82*	*0.81*	*0.79*	*0.80*	*0.86*	*0.86*	*0.92*	*0.99*		

* One problem with average error ratios is that when the actural error rates are very small, ratios tend to have extreme values. Here, we recompute the average ratios of CBA(2)+C4.5+NB vs the other methods after the largest and smallest values are removed.

REFERENCES

[AS97] R. Agrawal, and R. Srikant. Fast Algorithms for Mining Association Rules. *In Proceedings of VLDB-94*, 1994.

[AMS97] K. Ali, S. Manganaris and R. Srikant. Partial Classification Using Association Rules. *In Proceedings of KDD-97*, 115-118, 1997.

[AP96] K. Ali, and Pazzani M. Error Reduction through Learning Multiple Descriptions. *Machine Learning*, 24:3, 1996.

[B97] Bayardo, R. J. Brute-force mining of high-confidence classification rules. *In Proceedings of KDD-97*, 1997.

[CS93] P. Chan, and J. S. Stolfo. Experiments on multistrategy learning by meta-learning. *Proc. Second Intl. Conf. Info. Know. Manag.*, 314-323, 1993.

[CN89] P. Clark, and T. Niblitt. The CN2 Induction Algorithm. *Machine Learning* 3(1), 1989.

[C95] W. Cohen. Fast Effective Rule Induction. *In Proceedings of ICML-95*, 1995.

[CS99] W. Cohen, and Y. Singer. A Simple, Fast, and Effective Rule Learner. *In Proceedings*

of AAAI-99, 1999.
[DP97] P. Domingos, and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29, 1997.
[DZWL99] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by Aggregating Emerging Patterns. *In Proceedings of Discovery-Science-99*, 1999.
[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. *In Proceedings of ICML-95*, 1995.
[DH73] R. Duda, and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
[FI93] U. Fayyad, and K. Irani. Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. *In Proceedings of IJCAI-93*, 1022-1027, 1993.
[FS96] Y. Freund, and R. Schapire. Experiments with a New Boosting Algorithm. *In Proceedings of ICML-96*, 1996.
[FW94] J. Furnkranz, and G. Widmer. Incremental Reduced Error Pruning. *ICML-94*, 1994.
[K96] R. Kohavi. Scaling up the Accuracy of Naïve-Bayes Classifiers: A Decision-tree Hybrid. *In Proceedings of KDD-96*, 1996.
[KJLMP94] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A Machine-learning Library in C++. *Tools with artificial intelligence*, 740-743, 1994.
[LW89] N. Littlestone, and M. Warmuth. *The weighted majority algorithm*. Tech. report, UCSC-CRL-89-16: UC. Santa Cruz, 1989.
[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. *In Proceedings of KDD-98*, 1998.
[LHM99] B. Liu, W. Hsu, and Y. Ma. Mining Association Rules with Multiple Minimum Supports. *In Proceedings of KDD-99*, 1999.
[LMW00]B. Liu, Y. Ma and C-K. Wong. Improving an Exhaustive Search Based Rule Learner *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*, 2000.
[LL00] H. Lu, and H-Y. Liu. Decision Tables: Scalable Classification Exploring RDBMS Capabilities. *VLDB-2000*, 2000.
[MW99] D. Meretkis, and B. Wuthrich. Extending Naïve Bayes Classifiers Using Long Itemsets. *In Proceedings of KDD-99*, 1999.
[MM96] C. J. Merz, and P. Murphy. UCI Repository of Machine Learning Database. [http://www.cs.uci.edu/~mlearn], 1996.
[M80] R. Michalski. Pattern Recognition as Rule-guided Induction Inference. *IEEE action On Pattern Analysis and Machine Intelligence* 2, 349-361, 1980.
[MP94] P. Murphy and M. Pazzani. Exploring the Decision Forest: an Empirical Investigation of Occam's Razor in Decision Tree Induction. *J. of AI Research* 1:257-275, 1994.
[Q92] J. R. Quinlan. *C4.5: Program for Machine Learning*. Morgan Kaufmann, 1992.
[Q94] J. R. Quinlan. Combining Instance-based and Model-Based Learning. *In Proceedings of ICML-94*, 1994.
[R96] R. Rymon. SE-tree Outperforms Decision Trees in Noisy Domains. *In Proceedings of KDD-96*, 331-336, 1996.
[WZH00] K. Wang, S. Zhou, and Y. He. Growing Decision Trees on Support-less Association Rules. *In Proceedings of KDD-2000*, 2000.
[W93] G. Webb. Systematic Search for Categorical Attribute-value Data-driven Machine Learning. In Proceedings of *Australian conference on Artificial Intelligence*, 1993.
[W92] D. Wolpert. Stacked Generalization. *Neural networks*, 5:241-259, 1992.
[ZW99] Z. Zheng and G. Webb. Stochastic Attribute Selection Committees with Multiple Boosting: Learning More Accurate and More Stable Classifier Committees. *In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99),* 1999.