Improving an Association Rule Based Classifier

Bing Liu, Yiming Ma, and Ching Kian Wong

School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543

{liub, maym, wongck}@comp.nus.edu.sg

Abstract. Existing classification algorithms in machine learning mainly use heuristic search to find a subset of regularities 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. Although the whole set of rules may not be used directly for accurate classification, effective 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 C4.5 by 19%, and improve performance on 29 datasets. Similar good results are also achieved against RIPPER, LB and a Naïve-Bayes classifier.

1 Introduction

Building effective classification systems is one of the central tasks of data mining. Past research has produced many techniques and systems (e.g., C4.5 [10], and RIPPER [3]). The existing techniques are, however, largely based on heuristic/greedy search. They aim to find only a *subset* of the regularities that exist in data to form a classifier.

In the past few years, the database community studied the problem of rule mining extensively under the name of association rule mining [1]. 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) constraints.

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 [7], LB [8] and CAEP [4]. The major strength of such systems is that they are able to use the most accurate rules for classification. This explains their good results in general. However, they also have some weaknesses, inherited from association rule mining.

- Traditional association rule mining uses only a single minsup in rule generation, which is inadequate for unbalanced class distribution (this will be clear later).
- 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 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 system called msCBA. Experiments on a set of 34 benchmark problems show that on average msCBA achieves lower error rate than CBA, C4.5 (tree and rules), and a Naïve-Bayse classifier (NB), LB and RIPPER (CAEP is not available for comparison).

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 msCBA are not accurate. The decision tree method is a clear choice because decision trees often go very deep, i.e., using many conditions. We then propose a technique to combine msCBA with the decision tree method as in C4.5. The basic idea is to use the rules of msCBA 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 composite method results in remarkably accurate classifiers.

2 Association Rule Mining for Classification

Association rule mining is stated as follows [1]: 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, $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$. The rule $X \rightarrow Y$ 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 the 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*). An efficient algorithm for mining association rules is the Apriori algorithm [1].

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. Each data record is also labeled with a class. The table data can be converted to transaction data as follows: We first discretize each continuous attribute into intervals (see e.g., [5] [6] on discretization algorithms). 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*.

For classification, we only need to generate rules of the form $X \rightarrow c_i$, where c_i is a possible class. We call such rules the *class association rules* (CARs). It is easy to modify the Apriori algorithm to generate CARs. We will not discuss it here (see [7]).

3 Classifier Building in CBA

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

Definition: Given two rules, r_i and r_j, $r_i \succ r_j$ (also called r_i precedes r_j or r_i has a higher precedence than r_j), if the confidence of r_i is greater than that of r_j, or if their confidences are the same, but the support of r_i is greater than that of r_j, or if 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. A CBA classifier is of the form:

$$\langle r_1, r_2, \ldots, r_n, \mathit{default\_class} \rangle. \tag{1}$$

where $r_i \in R$, $r_a \succ r_b$ if $b > a$. In classifying an unseen case, the first rule that satisfies the case classifies it. If no rule applies, the default class is used. A simple version of the algorithm for building such a classifier is given in Figure 1. [7] presents an efficient implementation of the algorithm. It makes at most two passes over the data.

```
R = sort(R);              /* according the precedence ≻ */
for each rule r ∈ R in sequence do
   if there are still cases in D AND r classifies at least one case 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

4 Improving CBA

4.1 Using Multiple Minimum Class Support

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 classic 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. If we set the minsup value too high, we may not find sufficient rules of infrequent classes. If we set the minsup value too low, we will find many useless and over-fitting rules for frequent classes. To solve the problems, msCBA adopts the following (*multiple minimum class supports*):

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:

$$\text{minsup}_i = \text{t\_minsup} \times \text{freqDistr}(c_i). \tag{2}$$

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 over-fitting rules for frequent classes.

4.2 Seeking Help From Other Techniques

As we mentioned earlier, for many 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. The aim is to combine msCBA with a method that is able to find long rules. Clearly, the decision tree method is a natural choice because decision trees often go very deep, i.e., using many conditions. In our implementation, we also include the Naïve-Bayes method (NB) as NB comes free from msCBA (the probabilities needed by NB are all contained in the 1-condition rules of msCBA).

The proposed combination method is based on the competition of different classifiers on different segments of the training data. 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 msCBA, *T* be the decision tree built by C4.5, and *N* be the Naïve-Bayse classifier. We use the rules in *A* to segment the data. For the set of training examples covered by a rule r_i in *A*, 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.

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. $Error_i$, $Error_{Ti}$ and $Error_{Ni}$ are initialized to 0. From line 9-11, 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.

```
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_Ti = Error_Ti + 1 end
6        if  N classifies e wrongly then Error_Ni = Error_Ni + 1 end
7  endfor
8  for each rule r_i (X → c_j) in R do      /*X is the set of conditions */
9        if Error_i ≤ Error_Ti and Error_i ≤ Error_Ni then  keep r_i
10       elseif Error_Ti ≤ Error_Ni then use X → (use T) to replace r_i
11       else use X → (use N) to replace r_i
12 endfor
```

Figure 2. The combination algorithm

5 Experiments

We now compare the classifiers built by msCBA, CBA, C4.5 (tree and rules, Release 8), RIPPER, NB, LB, and various combinations of msCBA, C4.5 and NB. The evaluations are done on 34 datasets from UCI ML Repository [9]. We also used Boosted C4.5 (the code is obtained from Zijian Zheng [11]) in our comparison. We ran all the systems using their default settings. We could not compare with existing classifier combination methods as we were unable to obtain the systems.

In all the experiments with msCBA, *minconf* is set to 50%. For *t_minsup*, 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%.

			1	2	3	4	5	6	7	8	9	10	11
			msCBA	CBA	C4.5 tree	C4.5 rules	RIPPER	NB	LB	C4.5 + NB	msCBA + NB	msCBA + C4.5	msCBA + C4.5 + NB
			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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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	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
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
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
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
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
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
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
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
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
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
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
	Average		*14.9*	*15.5*	*16.0*	*15.6*	*15.8*	*16.9*	*15.3*	*15.5*	*14.2*	*13.5*	*13.3*
			won-lost-tied: msCBA+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*	

			12	13	14	15	16	17	18	19	20	21	22	23
			msCBA + C4.5 + NB Vs											
			msCBA	CBA	C4.5 tree	C4.5 rules	RIPPER	NB	LB	C4.5 + NB	msCBA + NB	msCBA + C4.5	Boosted C4.5	msCBA + Boosted C4.5
			ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	ratio	err (%)	err (%)
1	anneal	CV-10	1.319	.769	.369	.533	.606	1.045	.778	.365	1.319	1.259	4.3	1.4
2	australian	CV-10	.972	1.059	.959	.927	.932	1.011	1.053	.971	1.000	1.000	15.9	13.5
3	auto	CV-10	.928	.679	1.049	.928	.776	.575	.656	.971	1.000	.882	15.1	18.5
4	breast-w	CV-10	.659	.581	.436	.488	.604	1.004	.897	.437	.775	.811	3.1	2.2
5	cleve	CV-10	1.023	1.047	.813	.802	.829	1.021	1.021	.802	1.039	.999	20.5	18.2
6	crx	CV-10	.981	1.016	.955	.948	.978	.979	1.110	.951	.989	1.009	15.7	13.6
7	diabetes	CV-10	.864	.871	.844	.854	.871	.904	.904	.858	.885	.889	29.4	25.2
8	german	CV-10	.940	.940	.877	.899	.896	1.012	1.008	.871	.976	.987	28.5	24.8
9	glass	CV-10	1.130	1.076	.970	.942	.842	1.003	.958	1.034	1.111	1.000	24.7	28.0
10	heart	CV-10	.940	.920	.781	.886	.867	.939	.938	.807	.940	.979	20.7	20.0
11	hepatitis	CV-10	.858	1.074	.891	.836	.926	1.082	1.038	.892	.926	1.000	17.5	18.2
12	horse	CV-10	.972	.914	1.163	.983	1.163	.828	.828	1.309	.984	1.000	18.7	16.0
13	hypo	CV-10	.870	.512	1.243	1.088	1.145	.565	.540	.916	.813	.926	1.1	0.9
14	ionosphere	CV-10	1.073	1.007	.787	.826	.725	.690	.937	.763	1.000	1.000	6.8	8.3
15	iris	CV-10	.506	.377	.570	.570	.503	.446	.503	.575	.506	1.000	5.3	2.7
16	labor	CV-10	.876	.706	.538	.580	.727	.857	.973	.632	.878	1.000	8.3	8.3
17	led7	CV-10	.930	.940	.857	.986	.848	.979	.981	1.016	.979	.967	26.2	25.1
18	lymph	CV-10	.795	.896	.738	.663	.843	.720	.893	.759	.795	1.000	18.3	16.9
19	pima	CV-10	.812	.797	.853	.898	.837	.898	.890	.866	.845	.971	27.3	24.1
20	sick	CV-10	.664	.689	1.691	1.240	.964	.473	.628	1.632	.626	1.000	1.3	1.7
21	sonar	CV-10	.960	.995	.760	.724	.775	.937	.900	.775	.960	.940	20.2	23.0
22	tic-tac-toe	CV-10	.500	2.000	.014	.333	.083	.007	.006	.014	1.000	1.000	3.3	0.2
23	vehicle	CV-10	1.106	1.095	1.203	1.251	1.092	.856	1.124	1.209	1.107	1.151	24.3	29.0
24	waveform21	CV-10	.871	.858	.775	.807	.864	.914	1.009	.767	.886	.998	18.2	16.0
25	wine	CV-10	1.000	.595	.686	.686	.590	.526	2.994	.639	1.000	.998	4.0	4.5
26	zoo	CV-10	.909	.539	.373	.373	.265	.212	.500	1.455	1.000	1.000	0.0	2.9
27	Adult	test	.833	.963	.950	.984	.887	.877	.974	.989	.992	.933	16.2	16.2
28	Chess	test	.142	.149	.560	.248	.149	.022	.039	.596	.130	1.000	0.3	0.2
29	DNA	test	.771	.514	1.086	1.148	.960	1.202	1.094	1.191	1.057	.960	5.3	6.2
30	Letter	test	.459	.467	1.120	1.006	.905	.550	.856	1.122	.563	.987	5.2	8.9
31	Satimage	test	.966	.890	.966	.953	.937	.786	1.048	.979	.976	1.007	10.3	12.3
32	Segment	test	.631	.559	.628	.571	.484	.605	.676	.660	.829	1.000	3.1	4.2
33	Soybean Big	test	.941	.941	.669	.731	.762	1.143	.800	.667	1.067	1.000	6.1	7.0
34	Waveform40	test	.954	.951	.784	.761	.866	1.068	1.086	.795	1.007	1.000	20.9	23.1
	Average		*0.86*	*0.83*	*0.82*	*0.81*	*0.78*	*0.79*	*0.90*	*0.86*	*0.91*	*0.99*	*13.1*	*13.0*
			*** 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*		

Table 1: Experiment Results

* 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 msCBA+C4.5+NB vs the other methods after the largest and smallest values are removed.

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. All the composite methods involving C4.5 uses C4.5 tree.

Error rate comparison: For each dataset, column 1-11 show the error rates of msCBA, CBA, C4.5 tree, C4.5rules, RIPPER, NB, LB, C4.5+NB (C4.5 tree combinedwith NB), msCBA+NB, msCBA+C4.5 and msCBA+C4.5+NB respectively. Column 12-21 show the ratios of the error rate of msCBA+C4.5+NB vs. the other methods. From the table we see that on average the error rate of msCBA is lower than every other individual method. It is also clear that over the 34 datasets, the composite methods are superior to individual methods. msCBA+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 msCBA by 14%, and its won-lost-tied record against msCBA is 28-5-1. Similar good results are also achieved against CBA, RIPPER, NB and LB.

msCBA+C4.5 and msCBA+C4.5+NB have similar performances. This confirms our intuition that msCBA's weakness is overcome by deep trees of C4.5. Column 22 and 23 show the error rates of boosted C4.5 and msCBA+boostedC4.5. We see that msCBA+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 msCBA+C4.5+NB is also among the best.

6 Conclusion

This paper aims to improve an exhaustive search based classification system CBA. It first identified two weaknesses of the system, and it then proposed two new techniques to deal with the problems. The techniques produce markedly better classifiers.

References

1. Agrawal, R. & Srikant, R. 1994. Fast algorithms for mining association rules. VLDB-94.
2. Chan, P. & Stolfo, J. S. 1993. Experiments on multistrategy learning by meta-learning. Proc. Second Intl. Conf. Info. Know. Manag., 314-323.
3. Cohen, W. 1995. Fast effective rule induction. ICML-95.
4. Dong, G., Zhang, X. Wong, L. Li, J. 1999. CAEP: classification by aggregating emerging patterns. Discovery-Science-99.
5. Fayyad, U. & Irani, K. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. IJCAI-93, 1022-1027.
6. Kohavi, R., John, G., Long, R., Manley, D., & Pfleger, K. 1994. MLC++: a machine-learning library in C++. Tools with artificial intelligence, 740-743.
7. Liu, B., Hsu, W. & Ma, Y. 1998. Integrating classification and association rule mining. KDD-98.
8. Meretkis, D. & Wuthrich, B. 1999. Extending naïve bayes classifiers using long itemsets. KDD-99.
9. Merz, C. J. & Murphy, P. 1996. UCI repository of machine learning database. [http://www.cs.uci.edu/~mlearn].
10. Quinlan, J. R. 1992. C4.5: program for machine learning. Morgan Kaufmann.
11. Zheng, Z. and Webb, G. 1999. Stochastic attribute selection committees with multiple boosting: Learning more accurate and more stable classifier committees. *PAKDD-99.*