Majority Classification by Means of Association Rules

Elena Baralis and Paolo Garza

Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
{baralis,garza}@polito.it

Abstract. Associative classification is a well-known technique for structured data classification. Most previous work on associative classification based the assignment of the class label on a single classification rule. In this work we propose the assignment of the class label based on simple majority voting among a group of rules matching the test case.
We propose a new algorithm,L^3_M, which is based on previously proposed algorithm L^3. L^3 performed a reduced amount of pruning, coupled with a two step classification process. L^3_M combines this approach with the use of multiple rules for data classification. The use of multiple rules, both during database coverage and classification, yields an improved accuracy.

1 Introduction

Association rules [1] describe the co-occurrence among data items in a large amount of collected data. Recently, association rules have been also considered a valuable tool for classification purposes. Classification rule mining is the discovery of a rule set in the training database to form a model of the data, the classifier. The classifier is then used to classify appropriately new data for which the class label is unknown [12]. Differently from decision trees, association rules consider the simultaneous correspondence of values of different attributes, hence allowing to achieve better accuracy [2,4,8,9,14].

Most recent approaches to associative classification (e.g., CAEP [4], CBA [9], ADT [14], and L^3 [2]) use a single classification rule to assign the class label to new data whose label is unknown. A different approach, based on the use of multiple association rules to perform classification of new data has been proposed in CMAR [8], where it has been shown that this technique yields an increase in the accuracy of the classifier. We believe that this technique can be applied orthogonally to almost any type of classifier. Hence, in this paper we propose L^3_M, a new algorithm which incorporates multiple rule classification into L^3, a levelwise classifier previously proposed in [2].

L^3 was based on the observation that most previous approaches, when performing pruning to reduce the size of the rule base obtained from association rule mining, may go too far and discard also useful knowledge. We extend this

N. Lavrač et al. (Eds.): PKDD 2003, LNAI 2838, pp. 35–46, 2003.

idea to considering multiple rules to perform classification of new data. In this paper, we propose L^3_M, a new classification algorithm that combines the lazy pruning approach of L^3, which has been shown to yield accurate classification results, with a rule assignment technique that selects the class label basing its decision on a group of eligible rules which are drawn either from the first or the second level of the classifier.

The paper is organized as follows. Section 2 introduces the problem of associative classification. In Section 3 we present the classification algorithm L^3_M, by describing both the generation of the two levels of the classifier and the classification of test data by means of majority voting applied to its two levels. Section 4 provides experimental results which validate the L^3_M approach. Finally, in Section 5 we discuss the main differences between our approach and previous work on associative classification, and Section 6 draws conclusions.

2 Associative Classification

The database is represented as a relation R, whose schema is given by k distinct attributes $A_1 \ldots A_k$ and a class attribute C. Each tuple in R can be described as a collection of pairs *(attribute, integer value)*, plus a class label (a value belonging to the domain of class attribute C). Each pair *(attribute, integer value)* will be called *item* in the reminder of the paper. A training case is a tuple in relation R, where the class label is known, while a test case is a tuple in R where the class label is unknown.

The attributes may have either a categorical or a continuous domain. For categorical attributes, all values in the domain are mapped to consecutive positive integers. In the case of continuous attributes, the value range is discretized into intervals, and the intervals are also mapped into consecutive positive integers[1]. In this way, all attributes are treated uniformly.

A classifier is a function from $A_1, \ldots, A_n$ to $\mathcal{C}$, that allows the assignment of a class label to a test case. Given a collection of training cases, the classification task is the generation of a classifier able to predict the class label for test cases with high accuracy.

Association rules [1] are rules in the form $X \rightarrow Y$. When using them for classification purposes, X is a set of items, while Y is a class label. A case d is said to match a collection of items X when $X \subseteq d$. The quality of an association rule is measured by two parameters, its support, given by the number of cases matching $X \cup Y$ over the number of cases in the database, and its confidence given by the the number of cases matching $X \cup Y$ over the number of cases matching X. Hence, the classification task can be reduced to the generation of the most appropriate set of association rules for the classifier. Our approach to such task is described in the next section.

[1] The problem of discretization has been widely dealt with in the machine learning community (see, e.g., [5]) and will not be discussed further in this paper.

3 Majority Classification

In this paper, we introduce the use of multiple association rules to perform classification of structured data, in the levelwise classifier L^3 [2]. In L^3, a lazy pruning technique is proposed, which only discards "harmful" rules, i.e., rules that only misclassify training cases. Lazy pruning is coupled with a two levels classification approach. Rules that would be discarded by currently used pruning techniques are included in the second level of the classifier and used only when first level rules are not able to classify a test case.

Majority selection of the class label requires (a) selecting a group of good quality rules matching the case to be classified, and (b) assigning the appropriate class with simple majority voting among selected rules. To obtain a good quality rule set in step (a), a wide selection of rules from which to extract matching rules should be available. In Section 3.1 we describe how association rules are extracted, while in section 3.2 we discuss how the rules that form the model of the classifier are selected. Finally, in Section 3.3 the majority classification technique is presented.

3.1 Association Rule Extraction

Analogously to L^3, in L^3_M abundance of classification rules allows a wider choice of rules both for rule selection when the classifier is generated, and for new case classification. Hence, during the rule extraction phase, the support threshold should be set to zero. Only the confidence threshold should be used to select good quality rules. Unfortunately, no rule mining algorithm extracting rules only with a confidence threshold is currently available[2].

In L^3_M, the extraction of classification rules is performed by means of an adaptation of the well-known FP-growth algorithm [7], which only extracts association rules with a class label in the head. Analogously to [8], we also perform pruning based on χ^2 (see below) during the rule extraction process.

3.2 Pruning Techniques and Classifier Generation

In L^3_M two pruning techniques are applied: χ^2 pruning and lazy pruning. χ^2 is a statistical test widely used to analyze the dependence between two variables. The use of χ^2 as a quality index for association rules is proposed for the first time in [11] and is also used in [8] for pruning purposes. This type of pruning was not performed in L^3. However, we performed a large number of experiments, which have shown that rules which do not match the χ^2 threshold are usually useless for classification purpose. Since the use of χ^2 test heavily reduces the size of the rule set, it may significantly increase the efficiency of the following steps without deteriorating the informative content (quality) of the rule set after pruning. We perform χ^2 pruning during the classification rule extraction step.

[2] Some attempt in this direction has been proposed in [13], but its scalability is unclear.

Even with χ^2 pruning, if a low minimim support threshold is used, a huge rule set may be generated during the extraction phase. However, most of these rules may be useful [2]. The second pruning technique used in L^3_M is the lazy pruning technique proposed in L^3.

Before performing lazy pruning, a global order is imposed on the rule base. Rules are first sorted on descending confidence, next on descending support, then on descending length (number of items in the body of the rule), and finally lexicographically on items. The only significant difference with respect to most previous work ([8], [9]) is rule sorting on descending length. Most previous approaches prefer short rules over long rules. The reason for our choice is to give a higher rank in the ordering to more specific rules (rules with a larger number of items in the body) over generic rules, which may lead to misclassification. Note that, since shorter rules are not pruned, they can be considered anyway.

The idea behind lazy pruning [2] is to discard from the classifier only the rules that do not correctly classify any training case, i.e., the rules that only negatively contribute to the classification of training cases. To this end, after rule sorting, we cover the training cases to detect "harmful" rules (see Figure 1), using a database coverage technique. However, to allow a wider selection of rules for majority classification, a different approach is taken in the generation of the classifier levels. In L^3_M a training document is removed from the data set when it is covered by δ rules, while in L^3 each training case is removed as soon as is covered by one rule. Hence, by setting $\delta = 1$ the lazy pruning performed by L^3_M degenerates in that of L^3.

Lines 1-26 of the pseudocode in Figure 1 show our approach. The first rule r in the sort order is used to classify each case d still in *data* (lines 3-11). Each case d covered by r is included in the set $r.dataClassified$, and the counter $d.covered$ is increased. When d is covered by δ rules ($d.covered = \delta$), d is removed from *data* (line 9). The appropriate counter of r is increased (lines 6-7), depending on the correctness of the label.

After all cases in *data* have been considered, r is checked. If rule r only classified training cases wrongly (lines 12-18), then r is discarded, and the counter of each case classified by r is decreased by one. Cases included in $d.covered$ and removed before (line 9), because covered by δ rules, are included again in *data* (line 15).

The loop (lines 2-20) is repeated for the next rule in the order, considering the cases still in *data*. The loop ends when either the data set or the rule set are empty. The remaining rules are divided in two groups (lines 21-26), which will form the two levels of the classifier:

Level I which includes rules that have already correctly classified at least one training case,

Level II which includes rules that have not been used during the training phase, but may become useful later.

Rules in each level are ordered following the global order described above.

Rules in level I provide a high level model of each class. Rules in level II, instead, allow us to increase the accuracy of the classifier by capturing "special"

```
Procedure generateClassifier(rules,data,δ)
1. r = first rule of rules;
2. while (data not empty) and (r not NULL) {
3.    for each d in data {
4.      if r matches d {
5.         r.dataClassified = r.dataClassified ∪ d;
6.         if (d.class==r.class)   r.right++;
7.         else   r.wrong++;
8.         d.matched++;
9.         if (d.matched==δ) delete d from data;
10.     }
11.   }
12.   if r.wrong>0 and r.right==0 {
13.      delete r from rules;
14.      for each d in r.dataClassified {
15.        if (d.matched==δ)   data=data ∪ d;
16.        d.matched- -;
17.      }
18.   }
19.   r=next rule from rules;
20.}
21. for each r in rules {
22.    if r.right>0
23.       levelI = levelI ∪ r;
24.    else
25.       levelII = levelII ∪ r;
26. }
```

Fig. 1. L^3_M classifier generation

cases which are not covered by rules in the first level. Even if the levels are used similarly to L^3, their size may be significantly different. In particular, the use of δ generally increases the size of the first level, compared to the first level of L^3. Hence, L^3_M is characterized by a first level which is "more fat" that that of L^3. In Section 4 it is shown that this technique may provide a higher accuracy than L^3, but the model includes more rules and is hence somewhat less readable as a high level description of the classifier. However, we note that the readability of the classifier generated by L^3_M is still better than that of non-associative classifiers (e.g., Naive-Bayes [6]).

3.3 Classification

Majority classification is performed by considering multiple classification rules to assign the class label to a test case. The first step is the selection of a group of rules matching the given test case. When rules in the group yield different class labels, a simple majority voting technique is used to assign the class label. The size of the rule group (i.e., the maximum number of rules used to classify new

cases) may vary, depending on the number of rules matching the new case. It is limited by an upper bound, defined by the parameter max_rules.

This technique is combined with the two levels of the L^3 classifier. To build a rule group, rules in level I of the classifier are first considered. If no rule in this level matches the test case, then rules in level II are considered. Hence, rules in a rule group are never selected from both levels.

When a new case is to be classified, the first level is considered. The algorithm selects (at most) the first max_rules rules in the first level matching the case. When at least one rule matches the case, the matching process stops either at the end of the first level, or when the upper limit max_rules is reached.

Selected rules are divided in sets, one for each class label. Then, simple majority voting takes place. The rule set with the largest cardinality assigns the class label to the new case. A different approach, based on the evaluation of a weight for each rule set using χ^2 (denoted as $\chi^2 - max$), is proposed in CMAR [8]. We performed a wide set of experiments which showed that the average accuracy obtained by using this method is slightly lower than that given by the simple majority technique described above, differently from what is reported in [8]. The difference between our results and those reported in [8] may be due to the elimination of redundand rules applied in CMAR and not in L^3_M.

If no rule in the first level matches the test case, then rules in level II are considered. Both matching process and label assignment are repeated analogously for this level.

We observe that the use of $\delta > 1$ during the lazy pruning phase is necessary when using the simple majority technique described before to classify new cases. Indeed, if δ is set to one, only few rules are included in the first level, which becomes very thin. In this case, just a couple of rules may be available in the first level for matching and majority voting, and the selection of the appropriate class label may degenerate to the case of single rule classification.

L^3_M usually contains a large number of rules. In particular, the first level of the classifier contains a limited number of rules, which during the training phase covered some training cases. The cardinality of the first level is comparable to the size of the rule set in CMAR, while most previous approaches, including L^3 first level, were characterized by a smaller rule set. As shown in Section 4, this level performs the "heavy duty" classification of most test cases and provides a general model of each class. By contrast, level II of the classifier usually contains a large number of rules which are seldom used. These rules allow the classification of some more cases, which cannot be covered by rules in the first level.

Since level I usually contains about 10^2-10^3 rules, it can easily fit in main memory. Thus, the main classification task can be performed efficiently. Level II, in our experiments, included around 10^5-10^6 rules. Rules were organized in a compact list, sorted as described in Section 3.2. Level II of L^3 could generally be loaded in main memory as well. Of course, if the number of rules in the second level further increases (e.g., because the support threshold is further lowered to capture more rules with high confidence), efficient access may become difficult.

4 Experimental Results

In this section we describe the experiments to measure accuracy and classification efficiency for L^3_M. We compared L^3_M with the classification algorithms CBA [9], CMAR [8], C4.5 [12], and with its previous version with single rule classification L^3 [2]. The differences between our approach and the above algorithms is further discussed in Section 5. A large set of experiments has been performed, using 26 data sets downloaded from UCI Machine Learning Repository [3]. The experiments show that L^3_M achieves a larger average accuracy (+0.47% over the best previous, i.e., L^3), and has best accuracy on 10 data sets over 26.

For classification rule extraction the mininum support threshold has been set to 1%, a standard value used by previous associative classifiers. For 5 data sets (auto,hypo,iono,sick,sonar) the mininum support threshold has been set to 5%, to limit the number of generated rules. The confidence constraint has not been enforced, i.e., $minconf$=0. We have adopted the same technique used by CBA to discretize continuous attributes. A 10 fold cross validation test has been used to compute the accuracy of the classifier. All the experiments have been performed on a 1000Mhz Pentium III PC with 1.5G main memory, running RedHat Linux 7.2.

Recall from Section 3 that the performance of L^3_M depends on the values of two parameters: δ and max_rules. δ is used during the training phase and sets the maximal number of rules that can match a document. max_rules is used during the classification phase and sets an upper bound on the size of the selected rule group before voting. A huge amount of experiments has been performed, using different values for the parameters δ and max_rules. Unfortunately, it has not been possible to find overall optimal values for the parameters. However, we have devised values, denoted as default values, that yield a good average result, and are sufficiently appropriate for every data set considered in the experiments. The default values are $\delta = 9$, $max\_rules = 9$. These values may be used for "normal" classification usage, for any data set.

We report in Figure 2 the variation of accuracy with varying δ for different values of max_rules. For many data distributions, of which data set TicTac is representative, accuracy tends to be stable after a given threshold for parameter values. Hence, default values and optimal values tend to be very close. A different behavior is shown by data set Cleve, for which high accuracy is associated with very specific values of the parameters (e.g., optimal values are $\delta = 2$, $max\_rules = 9$). For these exceptional cases, optimal values can only be computed by running a vast number of experiments in which values of the parameters are varied and average accuracy is evaluated on a ten fold. The value pair that yields the best accuracy is finally selected. This technique requires fine tuning for a specific data set and should be used only when very high accuracy is needed.

Table 1 compares the accuracy of L^3_M with the accuracy of L^3, C4.5, CBA and CMAR, obtained using standard values for all the parameters. In particular, the columns of Table 1 are: (1) name of data set, (2) number of attributes, (3) number of classes, (4) number of cases (records), (5) accuracy of C4.5, (6) accuracy of

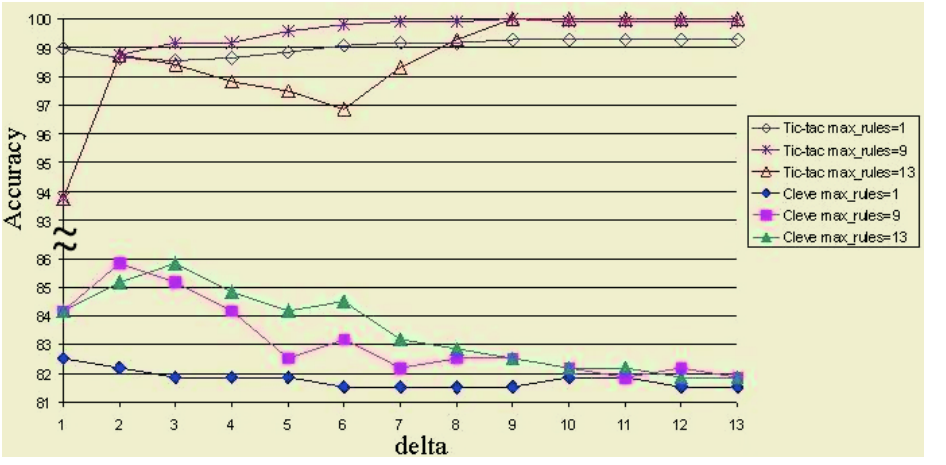

Fig. 2. Variation of accuracy with varying δ

CBA, (7) accuracy of CMAR, (8) accuracy of L^3, (9) accuracy of L^3_M with default values ($\delta = 9$, $max\_rules = 9$, identical for all data sets), (10) accuracy obtained using only the first level of L^3_M (always with default parameters), (11) improvement in accuracy given by the second level of L^3_M, and (12) accuracy of L^3_M with optimal values (different values of parameters for each data set).

L^3_M, with default values, has best average accuracy (+0.47% with respect to L^3) and best accuracy on 10 of the 26 UCI data sets. Only for 7 data sets the accuracy achieved by L^3_M is lower than that achieved by L^3, while for 15 data sets the accuracy is larger. Hence, the use of majority voting can improve the approach proposed by L^3.

We ran experiments to separate the contribution in accuracy improvement due to the use of multiple rules during the classification phase, and to the second level. In particular, we compared the accuracy obtained by only using rules in level I of L^3_M with the accuracy obtained by using both levels. The results of the experiments are reported in Table 1. The related columns of Table 1 are: (10) accuracy of L^3_M using only rules in the first level, (11) difference between L^3_M with both levels (column (9)) and L^3_M with only first level (column (10)). By considering only rules in the first level, L^3_M achieves best accuracy on 9 of the UCI data sets, and has average accuracy higher than L^3 (+0.25%). This result shows the significant effect due to multiple rule usage in the first level.

The effect of the second level in L^3_M is definitely less relevant than in L^3. We observe an increase in accuracy given by the second level only in 8 data sets, and the average increase in accuracy is +0.22%. The increase given by the second level of L^3 is more relevant [2]. In particular, for 20 data sets the second level is useful, and an average accuracy increase of +1.67% is given by the use of the second level. These results highlight that the second level is very useful when δ is set to 1 (L^3) and the first level is very thin. However, its contribution is less

Table 1. Comparison of L^3_M accuracy with respect to previous algorithms

Name	A	C	R	C4.5	CBA	CMAR	L^3	L^3_M default values	Only I level	Δ_{acc}	L^3_M optimal values
Anneal	38	6	898	94.8	**97.9**	97.3	96.2	96.4	96.4	0.00	96.4
Austral	14	2	690	84.7	84.9	**86.1**	85.7	**86.1**	86.1	0.00	**86.4**
Auto(*)	25	7	205	**80.1**	78.3	78.1	81.5	78.5	76.6	1.90	81.5
Breast	10	2	699	95.0	96.3	96.4	95.9	**96.6**	96.6	0.00	**96.7**
Cleve	13	2	303	78.2	82.8	**82.2**	82.5	82.5	82.5	0.00	**86.4**
Crx	15	2	690	84.9	84.7	84.9	84.4	**85.5**	85.1	0.40	**85.9**
Diabetes	8	2	768	74.2	76.7	74.5	75.8	**78.6**	78.6	0.00	**79.0**
German	20	2	1000	72.3	73.4	**74.9**	73.8	74.5	74.5	0.00	74.7
Glass	9	7	214	68.7	73.9	70.1	**76.6**	75.7	75.7	0.00	**76.6**
Heart	13	2	270	80.8	81.9	82.2	**84.4**	83.3	83.3	0.00	**84.4**
Hepatic	19	2	155	80.6	81.8	80.5	**81.9**	**81.9**	81.3	0.60	**83.2**
Horse	22	2	368	82.6	82.1	82.6	**82.9**	82.1	81.8	0.30	**83.2**
Hypo(*)	25	2	3163	**99.2**	98.9	98.4	95.2	97.5	97.5	0.00	97.5
Iono(*)	34	2	351	90.0	92.3	91.5	**93.2**	92.8	92.0	0.80	**93.2**
Iris	4	3	150	**95.3**	94.7	94.0	93.3	93.3	93.3	0.00	94.0
Labor	16	2	57	79.3	86.3	89.7	91.2	**96.5**	96.5	0.00	**96.5**
Led7	7	10	3200	**73.5**	71.9	72.5	72.0	72.4	72.4	0.00	72.8
Lymph	18	4	148	73.5	77.8	83.1	**85.1**	84.5	83.8	0.70	**85.1**
Pima	8	2	768	75.5	72.9	75.1	**78.4**	78.0	78.0	0.00	**79.2**
Sick(*)	29	2	2800	**98.5**	97.0	97.5	94.7	94.7	94.7	0.00	94.7
Sonar(*)	60	2	208	70.2	77.5	79.4	78.9	**81.7**	81.7	0.00	**81.7**
Tic-tac	9	2	958	99.4	99.6	99.2	98.4	**100.0**	100.0	0.00	**100.0**
Vehicle	18	4	846	72.6	68.7	68.8	73.1	**73.2**	73.0	0.20	**73.2**
Waveform	21	3	5000	78.1	80.0	**83.2**	82.1	82.8	82.8	0.00	82.8
Wine	13	3	178	92.7	95.0	95.0	98.3	**98.9**	98.3	0.60	**98.9**
Zoo	16	7	101	92.2	96.8	**97.1**	95.1	97.0	97.0	0.00	97.0
Average				83.34	84.69	85.22	85.88	**86.35**	86.13	0.22	**86.84**

(*)*Minimum support threshold 5%*

relevant for $\delta > 1$, when the first level is already rich enough to allow a good coverage of test cases. The second level remains always useful to capture special cases and allows a further increase in accuracy.

In column (12) of Table 1 are reported the accuracy results obtained by using optimal values of the δ and *max_rules* parameters for each data set. The effect of fine tuning the parameters values is significant, since it yields an increase of about +1% compared to L^3, and 0.49% with respect to L^3_M with default values. Furthermore, the classifier shows best accuracy on 17 data sets.

Table 2 allows us to compare the structure and usage of the two levels for L^3 and L^3_M(with default values for the parameters). In order to analyze only the effect of multiple rule selection on the size of the two levels, L^3 has been modified to incorporate χ^2 pruning. This allows us to observe the difference in

level size between L^3_M and L^3 due exclusively to the level assignment technique based on multiple rule selection.

In Table 2 the comparison of the number of rules in the first level of L^3 (column (3)) and of L^3_M (column (4)) shows that the first level of L^3_M is about an order of magnitude larger. We performed other experiments, not reported here, using different values for δ, which showed that the first level of L^3_M is approximately δ times larger than the first level of L^3. The only exception to this rule is data set Wine, where the size of the first level in L^3_M and L^3 is comparable. In this case, the second level becomes more useful, since rules in the first level are not enough to cover all test cases. We can conclude that the first level of L^3_M trades a reduced readability in favor of an increased accuracy and the value of δ allows to fine tune the tradeoff between these two features.

In Table 2 is also reported the number of rules in the second level for L^3 (column (5)) and L^3_M (column (6)). We observe that the second level of L^3_M is usually slightly smaller than that of L^3. This may be due to two different effects. (1) The first level of L^3_M is larger and contains some rules that would have been assigned to the second level of L^3. This effect is particularly evident in the case of data set Iris, where the total number of rules is rather small. In this case, most rules migrate from the second level to the first level, leaving an almost empty second level. (2) The multiple matching technique used for generating the classifier causes L^3_M to analyze more rules, which L^3 did not consider at all. If these rules make only mistakes, they are pruned by L^3_M, but not by L^3 (L^3 considers them unused and assigns them directly to the second level).

We also analyzed the performance of L^3_M during the classification of test data. The classification time is not affected by the use of the second level, because it is used rarely (see column(8) of Table 2). The average time for classifying a new case is about 1ms, and is comparable to that reported for L^3. With respect to memory usage, since the size of both levels is not dramatically different for L^3_M and L^3, the same considerations already reported in [2] hold also for L^3_M.

5 Previous Related Work

CMAR [8] is the first associative classification algorithm where multiple rules are used to classify new cases. CMAR proposes a suite of different pruning techniques: pruning of specialistic rules, use of the χ^2 coefficient, and database coverage. In L^3_M pruning based on the χ^2 coefficient is adopted, but specialistic rules are not pruned. Our database coverage technique is more tolerant, since it allows more rules to cover the same training case. This effect depends on the value of the δ parameter, discussed in Section 4. A similar parameter is available in CMAR (denoted as δ), but its suggested value allows a lower number of rules during the selection step. Hence, in CMAR useful rules may be pruned, thus reducing the overall accuracy of the classifier. This problem has been denoted as overpruning in [2]. Furthermore, we use simple majority voting to assign the final class label to a test case, while in CMAR a more complex weighting technique

Table 2. Usage of the two levels

Name	R	Rules I level L^3	Rules I level L^3_M	Rules II level L^3	Rules II level L^3_M	Use of I level L^3_M	Use of II level L^3_M
Anneal	898	38	358	169802	168851	99.44	0.56
Austral	690	152	1458	171638	159165	100.00	0.00
Breast	699	51	516	6241	5407	100.00	0.00
Cleve	303	74	724	16481	14676	100.00	0.00
Crx	690	159	1422	341382	322675	99.57	0.43
Diabetes	768	65	360	466	180	100.00	0.00
German	1000	291	2420	62359	57786	100.00	0.00
Glass	214	30	274	1385	1047	100.00	0.00
Heart	270	56	506	3449	2725	100.00	0.00
Hepatic	155	31	313	185453	184757	98.71	1.29
Horse	368	97	888	179345	177803	99.73	0.27
Iris	150	8	82	88	13	100.00	0.00
Labor	57	13	85	209	119	100.00	0.00
Led7	3200	75	318	1159	980	100.00	0.00
Lymph	148	40	302	1442098	1441055	95.95	4.05
Pima	768	64	362	472	174	100.00	0.00
Tic-tac	958	28	599	3258	2566	100.00	0.00
Vehicle	846	180	1433	2408341	2406231	99.53	0.47
Wine	178	8	11	122249	122116	99.40	0.60
Zoo	101	10	72	1515389	1515288	100.00	0.00
Average						99.63	0.37

based on χ^2 is proposed. Experiments show that our technique is both simpler and more effective.

The L^3_M algorithm derives its two level approach from the L^3 algorithm proposed in [2] and enhances L^3 with the introduction of classification based on multiple rules. However, introducing majority voting requires a larger first level, which may reduce the readability of the model with respect to L^3. Hence, the selection of an appropriate value for the δ parameter allows the fine tuning of the richness of the first level. We observe that L^3 can be seen as a degenerate case of L^3_M, when both δ and max_rules parameters are set to 1.

Associative classification has been first proposed in CBA [9]. CBA, based on the Apriori algorithm, extracts only a limited number of association rules (max 80000). Furthermore, it applies a database coverage pruning technique that significantly reduces the number of rules in the classifier, thus losing relevant knowledge. A new version of the algorithm has been presented [10], in which the use of multiple supports is proposed, together with a combination of C4.5 and Naive-Bayes classifiers. Unfortunately, none of these techniques addresses the overpruning problem described in [2].

ADT [14] is a different classification algorithm based on association rules, combined with decision tree pruning techniques. All rules with a confidence

greater or equal to a given threshold are extracted and more specific rules are pruned. A decision tree is created based on the remaining association rules, on which classical decision tree pruning techniques are applied. Analogously to other algorithms, the classifier is composed by a small number of rules and prone to the overpruning problem.

6 Conclusions

In this paper we have described L^3_M, an associative classifier which combines levelwise classification with majority voting. This approach is a natural extension of the concept of exploiting rule abundance for associative classification, initially proposed in [2]. In [2] rule abundance was only pursued when selecting rules to form the classifier by performing lazy pruning. With L^3_M we extend the same concept to the classification phase, by considering multiple rules for label assignment. Experiments show that the adopted approach allows a good increase in accuracy with respect to previous approaches. The main disadvantage of this approach is the (slightly) reduced readability of the first level of the classifier, which should provide a general model of classes.

References

1. R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. *In SIGMOD'93 , Washington DC*, May 1993.
2. E. Baralis and P. Garza. A lazy approach to pruning classification rules. *In ICDM'02, Maebashi, Japan*, December 2002.
3. C. Blake and C. Merz. UCI repository of machine learning databases, 1998.
4. G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by aggregating emerging patterns. *In Int. Conf. on Discovery Science, Tokyo, Japan*, Dec. 1999.
5. U. Fayyad and K. Irani. Multi-interval discretization of continuos-valued attributes for classification learning. *In IJCAI'93*, 1993.
6. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning, 29:131-163*, 1997.
7. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *In SIGMOD'00, Dallas, TX*, May 2000.
8. W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. *In ICDM'01, San Jose, CA*, November 2001.
9. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. *In KDD'98, New York, NY*, August 1998.
10. B. Liu, Y. Ma, and K. Wong. Improving an association rule based classifier. *In PKDD'00, Lyon, France*, Sept. 2000.
11. R. Motwani, S. Brin, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlation. *ACM SIGMOD'97 , Tucson, Arizona*, May 1997.
12. J. Quinlan. *C4.5: program for classification learning*. Morgan Kaufmann, 1992.
13. K. Wang, Y. He, D. W. Cheung, and F. Y. L. Chin. Mining confident rules without support requirement. *In CIKM'01, Atlanta, GA*, November 2001.
14. K. Wang, S. Zhou, and Y. He. Growing decision trees on support-less association rules. *In KDD'00, Boston, MA*, August 2000.