On Support Thresholds in Associative Classification

Elena Baralis
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi, 10129, Torino, Italy
elena.baralis@polito.it

Silvia Chiusano
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi, 10129, Torino, Italy
silvia.chiusano@polito.it

Paolo Garza
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi, 10129, Torino, Italy
paolo.garza@polito.it

ABSTRACT

Associative classification is a well-known technique for structured data classification. Most previous works on associative classification use support based pruning for rule extraction, and usually set the threshold value to 1%. This threshold allows rule extraction to be tractable and on the average yields a good accuracy. We believe that this threshold may be not accurate in some cases, since the class distribution in the dataset is not taken into account. In this paper we investigate the effect of support threshold on classification accuracy. Lower support thresholds are often unfeasible with current extraction algorithms, or may cause the generation of a huge rule set. To observe the effect of varying the support threshold, we first propose a compact form to encode a complete rule set. We then develop a new classifier, named L^3_G, based on the compact form. Taking advantage of the compact form, the classifier can be built also with rather low support rules. We ran a variety of experiments with different support thresholds on datasets from the UCI machine learning database repository. The experiments showed that the optimal accuracy is obtained for variable threshold values, sometime lower than 1%.

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 profitably exploited to perform classification. 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 [18]. Differently from decision trees, association rules consider the simultaneous correspondence of values of different attributes, hence allowing to achieve better accuracy.

Different approaches have been proposed for associative classification such as CAEP [10], CMAR [12], CBA [13] and ADT [19]. Unfortunately, in large or highly correlated datasets, the rule extraction algorithms have to deal with the combinatorial explosion of the solution set. This makes (i) frequently unfeasible (or at least time consuming) the rule extraction process, (ii) difficult to optimally exploit the generated rules, and (iii) extremely complex for a human to analyze the rules.

To tackle this problem, various techniques have been proposed to prune the rule base obtained by association mining. These techniques discard either rules that are redundant from a functional point of view, or that achieve less accuracy in classification according to parameters such as support, confidence, and chi-square test [12, 13, 19]. On the other hand, recently a novel classification algorithm named L^3 (Live and Let Live) [2] has been proposed. L^3 is based on the idea that most pruning techniques may discard also useful knowledge together with low quality rules. Hence, L^3 performs lazy pruning, i.e., a very reduced amount of pruning, by eliminating only "harmful" rules (rules that only produce wrong classification results in the training data).

In this paper we address the issue of "use and abuse" of pruning techniques by considering pruning based on support constraints. A common position in previous approaches in the area of associative classification is pruning rules with support below 1%. This threshold represents a sort of trade off which allows achieving good accuracy in many datasets, coupled with limiting the complexity of the rule extraction process and avoiding the generation of a huge number of rules. In this paper we ran experiments with variable support thresholds. These experiments showed that the 1% support threshold may sometime be far from optimal.

Some works [14] already pointed out that this reference threshold may be not enough accurate in some cases, since the class distribution in the dataset is not taken into account. Hence, rules for classes with low frequency are pruned, even when the confidence of these rules is very high. As an alternative, it has been proposed to use multiple support thresholds [14] in order to limit the number of extracted rules for frequent classes, but at the same time not to hide infrequent classes.

We believe that, in the context of classification rules, a direction not yet well investigated to cope with this problem is the representation of rules in a compact form. Differently from associative classification, for association rule mining several methods have been proposed to tackle with the generation of a huge number of itemsets. Some methods mine maximal frequent itemsets [4], but many approaches have been proposed to mine frequent closed itemsets [3, 15, 17, 21]. Closed itemsets allow the generation of all the frequent itemsets with their frequency. As an alternative to closed

SAC'04, March 14-17, 2004, Nicosia, Cyprus

itemsets, δ-free itemsets [6] and disjunctive rules [8] have been proposed as a condensed representation of the itemsets in a dataset. In [16, 20], closed itemsets have been used to compute a set of association rules from which all association rules can be generated. In associative classification, a form to encode the relevant knowledge in a classification rule set is proposed in [11]. The idea of using δ-free sets for classification purposes has been introduced in [9].

The present paper proposes an approach to encode a complete rule set. The proposed compact form relies on the notion of closed itemset for rule set compression, and is based on the concept of essential rule presented in [11]. The representation proposed in this paper avoids information loss and allows regenerating the complete rule set. Experiments show that the size of the compact form is significantly smaller than the complete rule set.

We then propose the novel classifier L^3_G, which is based on the compact representation of the rule set. L^3_G incorporates the idea of lazy pruning presented in [2], and extends it by providing a wider selection of rules obtained by allowing lower support thresholds. By means of the compact rule set representation, L^3_G allows us to analyze the effect of the support threshold on the achieved accuracy.

The paper is organized as follows. Section 2 introduces the basic definitions and notations in associative classification. Section 3 characterizes the compact representation. Section 4 presents L^3_G describing how the compact form is used to build the classifier. Section 5 presents a set of experiments to evaluate the effect of varying support thresholds on the accuracy of L^3_G and to validate the compression effectiveness of the proposed representation. Finally, Section 6 draws conclusions.

2. PROBLEM STATEMENT

The dataset $\mathcal{D}$ is represented as a relation R, whose schema is given by k distinct attributes $A_1 \dots A_k$ 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 remainder of the paper. The domain of items will be denoted as $\mathcal{I}$ in the following. 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. In this way, all attributes are treated uniformly.

A classifier is a function from $A_1, \dots, A_n$ 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.

In the following we will denote as $\mathcal{R}$ an arbitrary rule set extracted, with a given support threshold, from a dataset $\mathcal{D}$.

3. COMPACT RULE SET REPRESENTATION

In [11] a compact form to encode all the classification knowledge available in a given rule set is proposed. This form was based on the concept of essential rule. However, from this compact form it was not possible to derive the original rule set, which is needed to adopt the lazy pruning approach of L^3. In this paper, we extend the concepts presented in [11] to develop a compact form which encodes both the classification knowledge available in a given rule set, and all the rules in it. The following definition states when a rule is essential.

DEFINITION 1. *(Essential Rule). Let $r_i : X \rightarrow c_i$ be an arbitrary rule in $\mathcal{R}$. r_i is essential if $\nexists r_j \in \mathcal{R}$, $r_j : Y \rightarrow c_j$, such that (i) $c_i = c_j$, (ii) $Y \subset X$, (ii) $sup(r_i) = sup(r_j)$, and (iii) $conf(r_i) = conf(r_j)$.*

Intuitively, an essential rule is the most general rule among rules with equal support and confidence. A rule which is not essential is said to be a *specialization* of an essential rule.

Essential rules have been characterized [11] by means of the theory of closed itemsets. In the following we briefly summarize the main concepts on closed itemsets. More details can be found in [11]. A *closed* itemset [15, 21] is the maximal set of items common to a set of transactions. It is a compact representation of all the itemsets which are i) included in it and ii) included in its same transactions. A dataset can be encoded by means of the whole set of its closed itemsets.

Closed itemsets have been characterized by means of the Galois connection and the Galois closure operator [15, 21]. Denoting with γ the Galois closure operator, the *closure* of an itemset X is the itemset $\gamma(X)$. X and $\gamma(X)$ are in the same transactions and thus have equal support. X is a closed itemset if $\gamma(X) = X$. The generator [16] of a closed itemset X is an itemset G such that $\gamma(G) = X$, and there is not an itemset G' such that $G' \subset G$ and $\gamma(G') = X$. A closed itemset can have more generators.

As shown in [11], the Galois closure operator γ, which was formerly introduced for datasets of unlabeled transactions, can be applied also to the labeled transaction dataset $\mathcal{D}$. The knowledge of the closed itemset generators allows the characterization of essential rules.

The following theorem states some properties of classification rules based on the Galois closure operator. The theorem states that the rules have equal value of support, confidence, and chi-square (χ^2). χ^2 pruning 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 [7] and is also used in [12] for pruning purposes.

THEOREM 1. *Let $r_i : X \rightarrow c$ and $r_j : Y \rightarrow c$ be two arbitrary rules in $\mathcal{D}$. If $\gamma(X) = \gamma(Y)$, then (i) $sup(r_i) = sup(r_j)$, (ii) $conf(r_i) = conf(r_j)$, (iii) $\chi^2(r_i) = \chi^2(r_j)$, and (iv) r_i and r_j cover the same training data.*

Note that Theorem 1 states a sufficient but not necessary condition. Based on Theorem 1, the next theorem states when a rule is essential.

THEOREM 2. *Let $r : X \rightarrow c$ be an arbitrary rule in $\mathcal{R}$. r is an essential rule iff X is a generator itemset.*

Based on the two theorems above, we developed the compact form presented in this paper. In the compact form, essential rules are explicitly represented, while specialistic rules are summarized by means of an appropriate encoding. The compact form consists of a set of elements named *compact rules*. Each compact rule includes an essential rule and the encoding of the rules that are specializations of it, which will be denoted as *tail* of the compact rule in the following.

DEFINITION 2. *(Compact rule) Let $r : G \rightarrow c$ be an arbitrary essential rule in $\mathcal{R}$. Let $T \subseteq \mathcal{I}$ be an arbitrary itemset such that $T \subseteq (\gamma(G) - G)$. Then, $r_c : \{G, T\} \rightarrow c$ is a compact rule in $\mathcal{R}$.*

A compact rule $r_c : \{G, T\} \rightarrow c$ encodes an essential rule plus $2^{|T|} - 1$ specialistic rules. An arbitrary rule $r : X \rightarrow c$ encoded by r_c has the following characteristics: (i) r is labeled with the same class as r_c and (ii) X includes all items in G and a subset of zero (i.e., r is an essential rule), or more items in T(i.e., r is a specialistic rule). Based on Definition 2, given two arbitrary rules $r_i : X \rightarrow c$ and $r_j : Y \rightarrow c$ in r_c, it is $\gamma(X) = \gamma(Y)$. Hence, r_i and r_j satisfy the properties in Theorem 1. The set of all compact rules satisfying a given support threshold represents the complete set of classification rules satisfying the same support threshold without information loss. Furthermore, the definition of the compact rule tail (Definition 2) allows to avoid redundancy.

4. ASSOCIATIVE CLASSIFICATION

The compact rule representation described in the former section allows an effective representation of a large number of classification rules. We exploit this representation to enhance the ability of the classification algorithm L^3 (Live and Let Live) [2] to generate a classifier containing a large number of rules. L^3 first introduced the concept of lazy pruning, which was based on the observation that most previous approaches [10, 12, 13, 19] when performing pruning may go too far and discard also useful knowledge. Lazy pruning discards from the classifier exclusively the rules that only classify wrongly the training cases. L^3 covers the training cases to detect these rules by means of a database coverage technique.

Lazy pruning separates unpruned rules in two groups:

- *Level I*, providing a high level model of each class, which includes rules that have already correctly classified at least one training case.
- *Level II*, allowing to increase the accuracy of the classifier by capturing "special" cases which are not covered by Level I. This level includes all rules that have not been used during the training phase, but may become useful later.

Classification of a new test case is performed by means of a two steps classification algorithm. Initially, rules in Level I are considered. If no rule in Level I matches the case, then rules in Level II are considered.

This section focuses on the description of the new classification algorithm L^3_G, which uses the compact rule representation described in Section 3. In particular, in Section 4.1 we describe how a compact rule set is exploited to generate the L^3_G classifier, while in Section 4.2 we describe how L^3_G predicts the class label for new test cases.

4.1 Pruning and Generation of the Classifier

Before pruning, a global order is imposed on the compact rules. The compact rules are first sorted on descending confidence, next on descending support, then on decreasing length (number of items in the body) of the longest rule encoded in each compact rule, and finally lexicographically on items. As already observed in [2], this approach is slightly different from previous approaches [12, 13], in which rules are ordered by increasing length, and gives a higher rank in the ordering to more specific rules (rules with a larger number of items in the body) over generic rules. This choice allows us to prevent misclassification by shorter rules. The same global order will be used to order rules in the two levels.

We have introduced in L^3_G rule pruning based on the chi-square test (χ^2). 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 purposes. 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 rule extraction step. Recall that by Theorem 1 all rules included in a compact rule have the same χ^2 value. Hence, pruning of a compact rule r_c is equivalent to pruning all its instances.

L^3_G performs lazy pruning of the classification rule set by using the database coverage approach. However, database coverage should take into account the compact rule representation used in L^3_G. In particular, pruning of the compact rule set does not require to extract all rules encoded in it. For each compact rule r_c, it is sufficient to consider the longest rule encoded in it.

Consider an arbitrary compact rule $r_c : \{G, T\} \rightarrow c$. Assume to extract the rules encoded in r_c and let $\mathcal{R}_c = \{r_1, ..., r_n\}$ be the resulting rule set. Based on Definition 2 and Theorem 1, the rules in $\mathcal{R}_c$ have same support and confidence, and cover the same training cases. Rules in $\mathcal{R}_c$ are now sorted on decreasing length. Let r_i be the longest rule in r_c. Then the body of r_i includes both items in G and T. The following three cases are given:

1. r_i correctly classifies at least one training case. In this case, r_i is included in Level I and all cases covered by it are deleted from the training set. By Theorem 1, the rules in $\mathcal{R}_c$ cover the same training cases. Hence, the remaining rules in $\mathcal{R}_c$ will no longer match any training case. According to the lazy pruning approach, these rules belong to Level II. To exploit the compact form, we include in Level II rule r_c.
2. r_i does not match any training case. Recall that by Theorem 1 the rules in $\mathcal{R}_c$ cover exactly the same training cases. Thus, also all other rules in $\mathcal{R}_c$ will not match any training case[1]. Hence, r_c is included in Level II.

[1]This applies also for rules which are shorter than r_i.

3. r_i classifies only wrongly. In this case r_i is pruned. Again by Theorem 1 all other rules in r_c will classify only the same cases and assign an incorrect class label. In this case r_c is pruned.

With this approach, Level I is the same[2] in L^3_G and L^3. Note that Level I of L^3_G does not contain any compact rule. Level II of L^3_G, instead, only contains compact rules. It includes almost the same rules as L^3. The slight difference between L^3_G and L^3 is due to a more conservative pruning approach in L^3_G. In particular, in case 3, L^3_G prunes all the rules encoded in r_c. However, in L^3 some of these rules may be included in Level II, because they end up not matching any case, due to the interleaving with other longer rules with same support and confidence, but not belonging to $\mathcal{R}_c$. We have performed several experiments, which have shown that these rules do not affect the accuracy of the classifier.

4.2 Performing Classification

Classification of a new test case d is performed in two steps. First Level I is considered. The algorithm selects the first rule (i.e., the rule with highest rank) matching case d and assigns to d the class label of the selected rule.

If no rule in Level I matches the test case, then rules in Level II are considered. Recall that Level II only contains compact rules. However, the matching process does not require to extract all rules from a compact rule. A compact rule $r_c : \{G, T\} \rightarrow c$ matches a case d if $G \subseteq d$. Let r_c be the compact rule with highest rank and matching the new case d. The algorithm now builds a set of candidate compact rules $\mathcal{R}$ including r_c and all compact rules matching d and having the same value of support and confidence as r_c. The longest rule generated from a compact rule in set $\mathcal{R}$ and matching case d classifies d. For a given compact rule r_c, the longest rule in it matching d is the rule including in the body G and the largest subset T' of T such that $(G \cup T') \subseteq d$.

5. EXPERIMENTAL RESULTS

We performed a variety of experiments, whose aim was twofold. On the one hand, we wanted to analyze the effect of the support threshold on the classification accuracy. The extraction of classification rules in a compact form allows us to reach rather low support threshold, that were not always feasible with former extraction techniques. On the other hand, we performed an evaluation of the compression achievable by means of the proposed compact representation.

The experiments have been performed using 26 datasets downloaded from UCI Machine Learning Repository [5]. To run the experiments, a 10 fold cross validation test has been used to compute the accuracy of the classifier. To discretize continuous attributes we have adopted the technique used in [13]. All the experiments have been performed on a 1000Mhz Pentium III PC with 1.5G main memory, running RedHat Linux 7.2.

By using L^3_G we ran a number of experiments with different support thresholds, and we report in Table 1 the highest accuracy and the value of the threshold used to achieve it. For comparison, in Table 1 we also report the accuracy obtained by using the 1% support threshold, both for L^3_G and CMAR [12], and the accuracy obtained by the L^3 algorithm [2], which uses a different support threshold for each class. For CMAR, we use the standard values for all parameters suggested in [12]. In both L^3_G and CMAR the χ^2 test has been applied with the standard threshold value (i.e., 3.84). For L^3_G, we do not enforce the confidence constraint. Table 1 also shows the characteristics of the datasets in terms of number of attributes, class labels, and transactions (columns 2, 3 and 4 respectively).

Based on Table 1, we observe that the selection of an appropriate support threshold allows L^3_G to improve the accuracy in 13 cases out of 26 with respect to using the standard threshold 1%. The appropriate support value varies depending on the data distribution that characterizes the dataset, and is sometime lower than 1% and sometime higher. For some datasets (e.g., Anneal) the highest accuracy is achieved for absolute support 1. This value seems not to cause overfitting.

To further explore the effect of support on accuracy, we plotted the accuracy for different thresholds in four datasets. Figures 1(a) and 1(b) show that for Anneal and Hypo the accuracy increases for support values lower than 1%. Instead, in Auto (Figure 1(c)) the accuracy shows a singular peak for a very specific support value (2.2%). Finally, Sonar (Figure 1(d)) shows a rather steady accuracy for $minsup \geq 1\%$.

This analysis clearly shows that accuracy can heavily depend on the selection of an appropriate support threshold value. Previous approaches in associative classification used a uniform value 1%, which usually allowed rule extraction to be tractable and on average yielded a good accuracy. Lower support thresholds have not been explored because of both the untractability of the rule extraction step, and the huge size of the generated rule set. In this case, the compact form exploited by L^3_G may allow a more thorough analysis of the appropriate support threshold value, by enabling the extraction of rules with very low support thresholds.

Comparing the results on Table 1, we also observe that L^3_G improves the accuracy with respect to using multiple support thresholds with L^3 in 13 cases (in 9 cases L^3_G uses a support threshold different from 1%), and with respect to CMAR in 16 cases (in 9 cases L^3_G uses a support threshold different from 1%).

Tables 2 analyzes the compression rate of the compact representation. We characterize the compact rule sets generated by using the support thresholds which allow L^3_G to achieve the highest accuracy. Tables 2 reports the number of compact rules, the size of the file storing them, the total number of rules encoded in the compact representation, and the compression factor, computed as $(\#compact\ rules)/(\#rules)$. A low value of the compression factor shows that the compact rule set is significantly smaller than the complete rule set. The compression factor is significant in the majority of the examples. For some datasets, such as Auto, Hypo, Iono, Sick, and Sonar, the number of rules encoded by the compact representation is very large, up to 10^{10}. Extraction of such very large rule sets may become untractable with existing approaches. Lower support thresholds may be unfeasible also for L^3_G. The compression factor is usually not high in small or not dense datasets (e.g., Diabetes, Led7, Tic-Tac, and Waveform), where most of the rules are essential.

The structure of the two levels in L^3_G is similar to L^3 as reported in [2].

[2]When the χ^2 threshold is not enforced in L^3_G and a single support threshold is used for rule pruning.

Dataset	a	c	r	CMAR	L^3	L^3_G	L^3_G			Δ_{acc}		
						sup=1%	sup %	abs sup	acc	L^3_G-CMAR	L^3_G-L^3	L^3_G-L^3_G(1%)
ANNEAL	38	6	898	97.3	96.1	96.32	0.1	1	97.77	0.5	1.7	1.45
AUSTRAL	14	2	690	86.1	86.2	85.65	1.3	9	86.09	0	-0.1	0.44
AUTO	25	7	205	78.1	77.6	80.97	2.2	4	81.46	3.4	3.9	0.49
BREAST	10	2	699	96.4	96.0	96.28	1.3	9	96.34	-0.1	0.4	0.06
CLEVE	13	2	303	82.2	85.2	81.85	1.0	3	81.85	-0.4	-3.3	0
CRX	15	2	690	84.9	85.8	84.64	2.0	14	85.65	0.8	-0.1	1.01
DIABETES	8	2	768	75.8	78.4	79.17	0.3	2	79.30	3.5	0.9	0.13
GERMAN	20	2	1000	74.9	72.5	74.80	1.0	10	74.80	-0.1	2.3	0
GLASS	9	7	214	70.1	77.6	77.10	1.0	2	77.10	7.0	-0.5	0
HEART	13	2	270	82.2	83.0	84.44	1.0	3	84.44	2.2	1.5	0
HEPATI	19	2	155	80.5	86.5	85.81	1.0	2	85.81	5.3	-0.6	0
HORSE	22	2	368	82.6	83.2	83.69	1.3	5	83.70	1.1	0.5	0.01
HYPO	25	2	3163	98.4	95.2	97.70	0.4	13	98.70	0.3	3.5	1.00
IONO	34	2	351	91.5	92.9	91.73	1.0	4	91.73	0.2	-1.2	0
IRIS	4	3	150	94.0	93.3	93.33	1.0	2	93.33	-0.7	0	0
LABOR	16	2	57	89.7	93.0	91.23	1.0	1	91.23	1.5	-1.8	0
LED7	7	10	3200	72.5	72.0	72.00	0.7	22	72.06	-0.4	0.1	0.06
LYMPH	18	4	148	83.1	84.5	83.78	2.0	3	84.46	1.4	0	0.68
PIMA	8	2	768	75.1	78.0	78.39	1.0	8	78.39	3.3	0.4	0
SICK	29	2	2800	97.5	93.9	94.54	1.0	28	94.54	-3.0	0.7	0
SONAR	60	2	208	79.4	80.8	79.33	1.0	2	79.33	-0.1	-1.4	0
TIC-TAC	9	2	958	99.2	99.5	98.96	0.5	5	99.48	0.3	0	0.52
VEHICLE	18	4	846	68.8	68.6	72.93	1.1	9	73.29	4.5	4.7	0.36
WAVEFORM	21	3	5000	83.2	82.5	82.64	0.8	40	82.96	-0.2	0.4	0.32
WINE	13	3	178	95.0	97.2	97.19	1.0	2	97.19	2.2	0	0
ZOO	16	7	101	97.1	93.1	93.07	1.0	1	93.07	-4.0	0	0
Average				85.22	85.85	86.06			86.31	1.09	0.46	0.25

Table 1: Accuracy of L^3_G, L^3, and CMAR

6. CONCLUSIONS AND FUTURE WORK

The contribution of this paper is twofold. On the one hand, we propose a new compact form to encode a classification rule set. On the other hand, we propose a new classification algorithm based on the proposed compact form, which allows us to thoroughly analyze the effect of the support threshold on classification accuracy.

Our compact form relies on the notion of the Galois closure operator and is based on the concept of *essential rules* and *tail* of essential rules to encode all the rules. The compact form avoids information loss and allows regenerating all the encoded rules. Experiments show that the proposed form is significantly more compact than the complete rule set.

We have then presented a new classifier named L^3_G, which exploits the compact representation for classification. Using L^3_G we ran experiments to study the effect of the support threshold on classification accuracy. By taking advantage of the compact representation, L^3_G can reach rather low threshold values also on dense datasets. Experiments show that the optimal accuracy is achieved with varying support threshold values and that the standard support threshold value 1% is sometime far from optimal.

Acknowledgments

We wish to acknowledge Luigi Palermo and Chiara Vicari for implementing the code and performing the experiments.

7. REFERENCES

[1] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. *In SIGMOD'93.*

[2] E. Baralis and P. Garza. A lazy approach to pruning classification rules. *In ICDM'02.*

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, December 2000.

[4] R. J. Bayardo. Efficiently mining long patterns from databases. *In ACM SIGMOD'99.*

[5] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[6] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery journal, 7(1), pp. 5-22, Kluwer Academics Publishers*, May 2003.

[7] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing associations rules to correlations. *In ACM SIGMOD'97.*

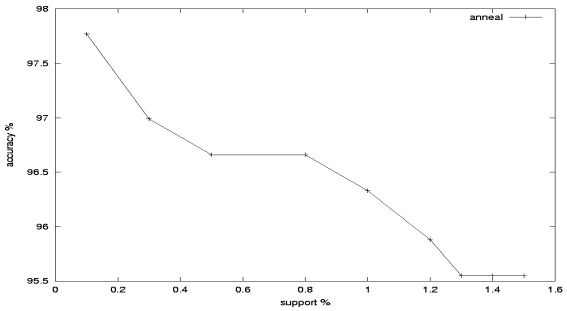
[8] A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. *In PODS'01.*

[9] B. Cremilleux and J.-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. *In ES'02.*

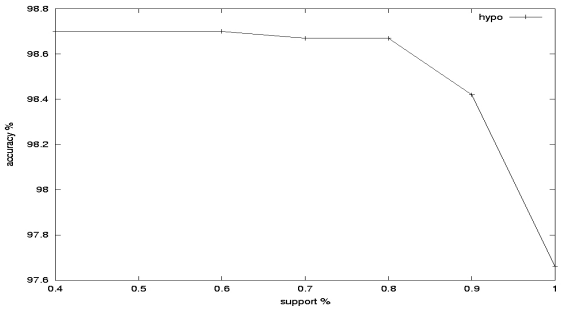
[10] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by aggregating emerging patterns. *In Int. Conf. on Discovery Science, 1999.*

[11] E.Baralis and S. Chiusano. Minimal non redundant classification rule sets. *IEEE ICDM Workshop on Foundations of Data Mining and Discovery, 2002.*

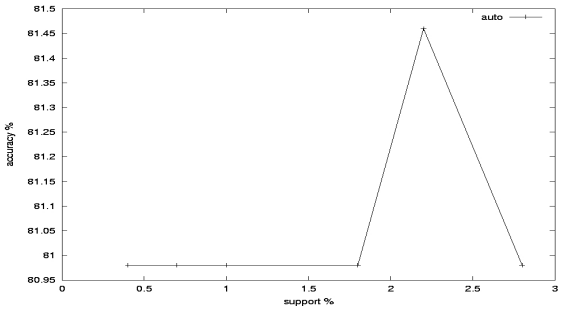
[12] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. *In ICDM'01.*



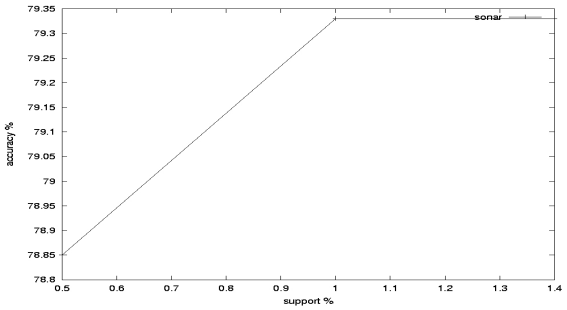
(a) Anneal



(b) Hypo



(c) Auto



(d) Sonar

Figure 1: Effect of support threshold on accuracy

Datasets	Sup %	Compact rules Number	MB	Rules	Compr. factor %
ANNEAL	0.1	7169	0.8	642778	1.12
AUSTRAL	1.3	60314	6	137744	43.79
AUTO	2.2	42399	5	18501580	0.23
BREAST	1.3	4098	0.4	5795	70.72
CLEVE	1.0	10234	1	17831	57.39
CRX	2.0	62972	6.3	149303	42.18
DIABETES	0.3	505	0.04	625	80.80
GERMAN	1.0	50462	5	96449	52.32
GLASS	1.0	603	0.1	1436	41.99
HEART	1.0	2706	0.3	3738	72.39
HEPATI	1.0	16333	1.7	689351	2.37
HORSE	1.3	23364	2.5	124726	18.73
HYPO	0.4	651435	89	53066952	1.23
IONO	1.0	1648002	249	34294845440	0.005
IRIS	1.0	68	0.01	109	62.39
LABOR	1.0	159	0.02	350	45.43
LED7	0.7	2262	0.2	2265	99.87
LYMPH	2.0	13379	1.4	410788	3.26
PIMA	1.0	481	0.04	581	82.79
SICK	1.0	5671477	803	341838208	1.66
SONAR	1.0	356741	41	10054523	3.55
TIC-TAC	0.5	6082	0.6	6249	97.33
VEHICLE	1.1	191620	21	3243734	5.91
WAVEFORM	0.8	95641	8.9	95695	99.94
WINE	1.0	12502	1.2	125073	10.00
ZOO	1.0	13666	1.5	1515369	0.90
Average					38.39

Table 2: Number of compact rules and rules

[13] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. *In KDD'98*.

[14] B. Liu, Y. Ma, and K. Wong. Improving an association rule based classifier. *In PKDD'00*.

[15] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemsets lattice. *In Information Systems*, 1999.

[16] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed itemsets discovery of small covers for association rules. *In Networking and Information Systems*, June 2001.

[17] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. *In ACM SIGMOD DMKD'00*.

[18] J. Quinlan. *C4.5: program for classification learning*. Morgan Kaufmann, 1992.

[19] K. Wang, S. Zhou, and Y. He. Growing decision trees on support-less association rules. *In KDD'00*.

[20] M. Zaki. Generating non-redundant association rules. *In KDD'00*.

[21] M. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. *In SIAM'02*.