Mining Optimal Class Association Rule Set

Jiuyong Li, Hong Shen and Rodney Topor

School of Computing and Information Technology
Griffith University
Nathan Qld 4111 Australia
Email: {jiuyong,hong,rwt}@cit.gu.edu.au

Abstract. We define an optimal class association rule set to be the minimum rule set with the same prediction power of the complete class association rule set. Using this rule set instead of the complete class association rule set we can avoid redundant computation that would otherwise be required for mining predictive association rules and hence improve the efficiency of the mining process significantly. We present an efficient algorithm for mining the optimal class association rule set using an upward closure property of pruning weak rules before they are actually generated. We have implemented the algorithm and our experimental results show that our algorithm generates the optimal class association rule set, whose size is smaller than $\frac{1}{17}$ of the complete class association rule set on average, in significantly less time than generating the complete class association rule set. Our proposed criterion has been shown very effective for pruning weak rules in dense databases.

1 Introduction

1.1 Mining predictive association rules

The goal of association rule mining is to find all rules satisfying some basic requirement, such as the minimum support and the minimum confidence. It was initially proposed to solve market basket problem in transaction databases, and has then been extended to solve many other problems such as classification problem. A set of association rules for the purpose of classification is called *predictive association rule set*. Usually, predictive association rules are based on attribute value (relational) databases, where the consequences of rules are pre-specified categories. Clearly, an attribute value database can be mapped to a transaction database when an attribute and attribute value pair is considered as an item. After having mapped an attribute value database into a transaction database, a class association rule set is a subset of association rules with the specified targets (classes) as their consequences. Generally, mining predictive association rules undergoes the following two steps.

1. Find all class association rules from a database, and then
2. Prune and organize the found class association rules and return a sequence of predictive association rules.

In this paper, we focus on the first step. There are two problems in finding all class association rules.

- It may be hard to find the all class association rule set in dense databases due to the huge number of class association rules. For example, many databases support more than $80,000$ class association rules as in [12].
- Too many class association rules will reduce the overall efficiency of mining predictive association rule set. This is because the set of found class association rules is the input of the second step processing whose efficiency is mainly determined by the number of input rules.

To avoid the above problems, it is therefore necessary to find a small subset but with the same prediction accuracy of the complete class association rule set, so that this subset can replace the complete class association rule set. Our proposed *o*ptimal class association rule set is the smallest subset with the same prediction power, which will be formally defined in Section 2, of the complete class association rule set. We present an efficient algorithm to generate the optimal class association rule set that takes the advantage of an upward closure property to prune those complex rules that have lower accuracy than their simple form rules have before they are actually generated in dense databases. Our algorithm avoids redundant computation of mining the complete class association rule set from dense databases and improves efficiency of the mining process significantly.

1.2 Related work

Mining association rules [1] is a central task of data mining and has shown applications in various areas [7, 3, 12]. Currently most algorithms for mining association rules are based on Apriori [2], and used the so-calle "downward closure" property which states that all subsets of a frequent itemset must be frequent. Example of these algorithms can be found in [10, 14, 17]. A symmetric expression of downward closure property is *upward closure* property — all supersets of an infrequent itemset must be infrequent. We will use this property throughout the paper.

Finding classification rules has been an important research focus in the machine learning community [18, 8]. Mining classification rules can be viewed as a special form of mining association rules, since a set of association rules with prespecified targets can be used for classification. Techniques for mining association rules have already been applied to mining classification rules [3, 12]. Particularly, results in [12] are very encouraging, since it can build more accurate classifiers than those from C4.5 [18]. However, the algorithm in [12] is not very efficient since it uses Apriori-like algorithm to generate the class association rules, which may be very large when the minimum support is small. In this paper will show that we can use a much smaller class association rule set to replace this set while not losing accuracy (prediction power).

Generally speaking, class association rule set is a type of target-constraint association rules. Constraint rule sets [5] and optimal rule sets [4] belong to this type. Problems with these rule sets are that they either exclude some useful predictive association rules, or contain many redundant rules that are of no use for prediction. Moreover, algorithms for mining these rule sets handle only one target at one time (building one enumeration tree), so they cannot be efficiently used for mining class association rules that are on multiple targets, especially when the number of targets is large. Our optimal class association rule set differs from these rule sets at that it is minimal in size and keeps high prediction accuracy. We propose an algorithm that finds this rule set with respect to all targets at once.

In this paper we only address the first step of mining predictive association rules. Related work on pruning and organizing the found class association rules can be referred to [9, 13, 16].

1.3 Contributions

Contributions in this paper are the following.

1. We propose the concept of optimal class association rule set for predictive association rule mining. It is the minimum subset of complete class association rule set with the same prediction power as the complete class rule set, and can be used as a substitute of the complete class association rule set.
2. We present an efficient algorithm for mining the optimal class association rule set. This algorithm is different from Apriori at that 1) it uses an additional upward closure property for forward pruning weak rules (pruning before they are generated), and 2) it integrates frequent sets mining and rule finding together.
   Unlike the existing constraint and optimal rule mining algorithms, our algorithm finds strong (optimal) rules with all possible targets at one time.

2 Optimal class association rule set

Given attribute-value database D with n attribute domains. A record of D is a n-tuple. For the convenience of description, we consider a record as a set of attribute and value pairs, denoted by T. A *pattern* is a subset of a record. We say a pattern is a *k-pattern* if it contains k attribute and value pairs. An *implication* in database D is $A \Rightarrow c$, where A is a pattern, called *antecedent*, and c is an attribute value, named *consequence*. Exactly, the consequence is an attribute and value pair, but in class association rule mining, the target attribute is usually specified, so we can use its value directly without confusing. The *support* of pattern A is defined to be the ratio of the number of records containing A to the number of all records in D, denoted by $sup(A)$. The support of implication $A \Rightarrow c$ is defined to be the ratio of the number of records containing both A and c to the number of all records in D, denoted by $sup(A \Rightarrow c)$. The *confidence* of the implication $A \Rightarrow c$ is defined to be the ratio of $sup(A \Rightarrow c)$ to $sup(A)$, represented by $conf(A \Rightarrow c)$.

A *class association rule* is defined to be an implication with a pre-specified target (a value of target attribute) as its consequence and its support and confidence are above given thresholds from a database respectively. Given a target attribute, minimum support σ and minimum confidence ψ, a *complete class association rule set* is a set of all class association rules, denoted by $R_c(\sigma, \psi)$.

Our goal in this section is to find the minimum subset of the complete class association rule set that has the same prediction power as the complete class association rule set.

To begin with, let us have a look at how a rule makes prediction. Given a rule r, we use $cond(r)$ to represent its antecedent (conditions), and $cons(r)$ to denote its consequence. Given a record T in a database D, we say rule r can make prediction on T if $cond(r) \subset t$, denoted by $r(T) \rightarrow cons(r)$. If $cons(r)$ is the category (target attribute value) of record T, then this is a correct prediction. Otherwise, a wrong prediction.

Then we consider the accuracy of a prediction. We begin by defining the accuracy of a rule. Confidence is not the accuracy of a rule, or more precisely, not the prediction accuracy of a rule, but the sample accuracy, since it is obtained from the sampling (training) data. Suppose that all instances in a database are independent of one another. Statistical theory supports the following assertion [15]: $acc_t(r) = acc_s \pm z_N \sqrt{\frac{acc_s(1-acc_s)}{n}}$, where acc_t is the true (prediction) accuracy, acc_s is the accuracy over sampling data, n is the number of sample data ($n \geq 30$), and z_N is a constant relating to confidence interval. For example, $z_N = 1.96$ if confidence interval is 95%. We use pessimistic estimation as the prediction accuracy of a rule. That is $acc(r) = conf(r) - z_N \sqrt{\frac{conf(r)(1-conf(r))}{|cov(r)|}}$, where $cov(r)$ is the covered set of rule r that is defined in the next section. If $n < 30$, then we use Laplace accuracy instead [8], that is $acc(r) = \frac{sup(r)*|D|+1}{|cov(r)|+p}$, where p is the number of target attribute values (classes).

After we have obtained the prediction accuracy of a rule, we can estimate the accuracy of a prediction as follows: the accuracy of a prediction equals to the prediction accuracy of the rule making such prediction, denoted by $acc(r(T) \rightarrow c)$.

In the following part, we will discuss a prediction made by a rule set, and how to compare the prediction power of two rule sets.

Given a rule set R and an input T, there may be more than one rule in R that can make prediction, such as, $r_1(T) \rightarrow c_1, r_2(T) \rightarrow c_2, \ldots$. We say that the prediction made by R is the same as the prediction made by r if r is the rule with the highest prediction accuracy of all r_i where $cond(r_i) \subset t$. The accuracy of such prediction equals to the accuracy of rule r. In case if there are more than one rule with the same highest prediction accuracy, we choose the one with the highest support among them. When the predicting rules have the same accuracy and support, then we choose the one with the shortest antecedent. If there is no prediction made by R, then we say the rule set gives arbitrary prediction with accuracy zero.

To compare prediction power of two rule sets, we define

Definition 1. *Prediction power*
Given rule sets R_1 and R_2 from database D, we say that R_2 has at least the same power as R_1 iff, for all possible input, both R_1 and R_2 give the same prediction and prediction accuracy of R_2 is at least the same as that of R_1.

It is clear that not all rule sets are comparable in their prediction power. Suppose that rule set R_2 has more power than rule set R_1. Then for all input T, if there is rule $r_1 \in R_1$ giving prediction c with accuracy κ_1, then there must be another rule $r_2 \in R_2$ so that $r_2(T) \to c$ with accuracy $\kappa_2 \geq \kappa_1$.

We represent that rule set R_2 has at least the same power as rule set R_1 by $R_2 \geq R_1$. It is clear that R_2 has the same power as R_1 iff $R_2 \geq R_1$ and $R_1 \geq R_2$.

Now, we can define our optimal class association rule set.

Given two rules r_1 and r_2, we say that r_2 is *stronger* than r_1 iff $r_2 \subset r_1 \wedge acc(r_2) > acc(r_1)$, denoted by $r_2 > r_1$. Specifically, we mean $cond(r_2) \subset cond(r_1)$ and $cons(r_2) = cons(r_1)$ when we say $r_2 \subset r_1$. Given a rule set R, we say a rule in R is *strong* if there is no other rule in R that is stronger than it. Otherwise, the rule is *weak*. Thus, we have the definition for optimal class association rule set.

Definition 2. *Optimal class association rule set*
Rule set R_o is optimal for class association over database D iff (1) $\forall r \in R_o, \nexists r' \in R_o$ such that $r < r'$ and (2) $\forall r' \in R_c - R_o, \exists r \in R_o$ such that $r > r'$.

It is not hard to prove that the optimal class association rule set is unique at given minimum support and minimum confidence from a database. Let $R_o(\sigma, \psi)$ stand for the optimal class association rule set on database D at given minimum support σ and minimum confidence ψ. Then $R_o(\sigma, \psi)$ contains all strong rules from the complete class association rule set $R_c(\sigma, \psi)$.

Finally, we consider the prediction power of the optimal class association rule set we are concerned with.

Theorem 1. *The optimal class association rule set is the minimum subset of rules with the same prediction power as the complete class association rule set.*

Proof. For simplicity, let R_c stand for $R_c(\sigma, \psi)$ and R_o for $R_o(\sigma, \psi)$.

First, from the previous definitions we have that $R_c \geq R_o$ and $R_o \geq R_c$, so the optimal class association rule set has the same prediction power as the complete class association rule set has.

Secondly, we prove the minimum property of optimal class association rule set. Suppose that we leave out rule r from the optimal class association rule set R_o, $R'_o = R_o - r$, and R'_o has the same prediction power as R_c has. From the definition, we know that there is no rule being stronger than rule r, so $R_o \geq R'_o$, but $R'_o \ngeq R_o$. As a result, R'_o cannot be the same prediction power as R_c is, leading to contradiction. Hence, R_o is the minimum rule set with the property of same prediction power as the complete class association rule set has.

The fact that the optimal class association rule set has the same prediction power as the complete class association rule set is because it contains all strong rules. Even though the class association rule set is usually much larger than the optimal class association rule set, it contains many weak rules that cannot provide more prediction power than their strong rules do. In other words, the optimal class association rule set is totally equivalent to the complete class association rule set in terms of prediction power. Thus, it is not necessary to keep a rule set that is larger than the optimal class association rule set, and we can find all predictive association rules from the optimal class association rule set.

In the next section, we will present an efficient algorithm to mine the optimal class association rule set.

3 Mining algorithm

A straightforward method to obtain the optimal class association rule set R_o is to first generate the complete class association rule set R_c and then prune all weak rules from it. Clearly mining complete class association rule set R_c is very expensive and almost impossible when the minimum support is low. In this section, we present an efficient algorithm that can find the optimal class association rule set directly without generating R_c first.

Most efficient association rule mining algorithms use the upward closure property of infrequency of pattern: if a pattern is infrequent, so are all its super patterns. If we can find a similar property for weak rules, then we can avoid generating many weak rules, hence making the algorithm more efficient. In the following we will discuss an upward closure property for pruning weak rules

.

Let us begin with some definitions. We say that r_1 is a *general rule* of r_2 or r_2 is a *specific rule* of r_1 if $cond(r_1) \subset cond(r_2) \wedge cons(r_1) = cons(r_2)$. We define the *covered set* of rule r to be the set of records containing antecedent of the rule, denoted by $cov(r)$. Similarly, covered set of a pattern A is defined to be the set of records containing the pattern, denoted by $cov(A)$. It is clear that the covered set of a specific rule is a subset of the covered set of its general rule.

Suppose that X and Y are two patterns in database D, and XY is the abbreviation of $X \cup Y$. We have the following two properties of covered set.

Property 1. $cov(X) \subseteq cov(Y)$ iff $sup(X) = sup(XY)$.

Property 2. $cov(X) \subseteq cov(Y)$ iff $Y \subset X$.

Now we discuss an upward closure property for pruning weak rules. Given database D and a target value c in target attribute C, we have

Lemma 1. *If $cov(X\neg c) \subseteq cov(Y\neg c)$, then $XY \Rightarrow c$ and all its specific rules must be weak.*

Proof. We rewrite the confidence of rule $A \Rightarrow c$ as $\frac{sup(Ac)}{sup(Ac)+sup(A\neg c)}$. We know that function $f(u) = \frac{u}{u+v}$ is monotonically increasing with u when v is a constant. Noticing $sup(Xc) \geq sup(XYc)$ and $sup(X\neg c) = sup(XY\neg c)$, we have

$conf(X \Rightarrow c) \geq conf(XY \Rightarrow c)$. Using relation $|cov(X \Rightarrow c)| \geq |cov(XY \Rightarrow c)|$, we have $acc(X \Rightarrow c) \geq acc(XY \Rightarrow c)$. As a result, $X \Rightarrow c \geq XY \Rightarrow c$

Since $cov(XZ\neg c) \subseteq cov(YZ\neg c)$ for all Z, we have $XZ \Rightarrow c > XYZ \Rightarrow c$ for all Z.

Consequently, $XY \Rightarrow c$ and all its specific rules are weak.

We can perceive the lemma as follows: adding a pattern to the conditions of a rule is to make the rule more precise (with less negative examples), and we shall omit the pattern that fails to do so.

Corollary 1. *If $cov(X) \subseteq cov(Y)$, then $XY \Rightarrow c$ and all its specific rule must be weak for all $c \in C$.*

We can understand the corollary in the following way: we cannot combine a super concept with a sub concept as the antecedent of a rule to make the rule more precise.

Lemma 1 and Corollary 1 are very helpful for searching strong rules, since we can remove a set of weak rules as soon as we find that one satisfies the above Lemma and Corollary. Hence, the searching space for strong rules is reduced.

To find those patterns satisfying Lemma 1 and Corollary 1 efficiently, we need to use properties 1 and 2. Property 1 enables us to find subset relation by comparing supports of two patterns. This is very convenient and easy to implement since we always have support information. By Property 2, we can always find that the covered set of a pattern (e.g. X) is a subset covered set of its $|X| - 1$ cardinality subpattern. So, we only need to compare the support of a k-pattern with that of its $(k-1)$-subpatterns in order to decide whether the k-pattern should be removed.

Since both Lemma and Corollary state upward closure property of weak rules, we can have an efficient algorithm to prune them.

Basic idea of the proposed algorithm

We use a level-wise algorithm to mine the optimal class association rule set. We search strong rules from antecedent of 1-pattern to antecedent of k-pattern level by level. In each level, we select strong rules and prune weak rules. The efficiency of the proposed algorithm is based on fact that a number of weak rules are removed once satisfaction of the Lemma or the Corollary is found. Hence, searching space is reduced after each level's pruning. The number of phases of reading a database is bounded by the length of the longest rule in the optimal class association rule set.

Storage structure

A prefix tree, or enumerate tree [5] is used as the storage structure. A prefix tree is an ordered and unbalanced tree, where each node is labeled by an element in a sorted base set, B, representing a set $S \subset B$ containing all labels from the root to the node. Since set S is unique in a prefix tree, we can use it as the identity of a node.

We use an extended prefix tree, named *candidate tree* in our algorithm. The base set here contains all attribute and value pairs and they are sorted in the order of their first references. A node in a candidate tree store a pattern A that is the identity of the node, a potential target set Z, and a supset of possible attribute and value pair sets $\mathcal{Q}$. Pattern A is the antecedents of a possible rule. The potential target set Z is a set of values of target attribute that may be consequences of A. For each target (e.g. z_j) in Z, there is a set of possible attribute and value pairs which may be conjunct with A to form more accurate rules, $Q_j \in \mathcal{Q}$.

Our algorithm is given as follows. One distinction between this algorithm and other prefix tree based algorithms is that our algorithm finds all class association rules with respect to all consequences from one candidate tree rather than many candidate trees.

Algorithm: Optimal Class Association Rule Set Miner

Input: Database D with specified target attribute C, minimum support σ and minimum confidence ψ.

Output: Optimal class association rule set R.

```
Set optimal class association rule set $R = \emptyset$
Count support of 1-patterns
Initiate candidate tree $T$
Select strong rules from $T$ and include them in $R$
Generate new candidates as leaves of $T$
While (new candidate set is non-empty)
    Count support of the new candidates
    Prune the new candidate set
    Select strong rules from $T$ and include them in $R$
    Generate new candidates as leaves of $T$
Return rule set $R$
```

In the following, we present and explain two unique functions in the proposed algorithm.

Function: Candidate Generating

This function generates candidates for strong rules. Let n_i denote a node of the candidate tree, A_i be the pattern of node n_i, $Z(A_i)$ be the potential target set of A_i, and $Q_q(A_i)$ be a set of potential attribute value pairs of A_i with respect to target z_q. We use $\mathcal{P}^p(A_k)$ to denote the set of all p-subsets of A_k.

```
for each node $n_i$ at the $p$-th layer
    for each sibling node $n_i$ and $n_j$ ($n_j$ is after $n_i$)
        generate a new candidate $n_k$ as a son of $n_i$ such that      // combining
            $A_k = A_i \cup A_j$
            $Z(A_k) = Z(A_i) \cap Z(A_j)$
        $Q_q(A_k) = Q_q(A_i) \cap Q_q(A_j)$ for all $z_q \in Z(A_k)$
    for each $z \in Z(A_k)$                                   // testing
        if $\exists A \in \mathcal{P}^p(A_k)$ such that $sup(A \cup z) \leq \sigma$
            then $Z(A_k) = Z(A_k) - z$
if $Z_k = \emptyset$ then remove node $n_k$
```

We generate the $(p+1)$-layer candidates from the p layer in the candidate tree. First, we combine a pair of sibling nodes and insert their combination as a new node in the next layer. We initiate the new node with the union of the two nodes. Next, if any of its p-subpatterns cannot get enough support with any of the possible targets (consequences), then we remove the target from the target set. When there is no possible target left, remove the new candidate.

Function: Pruning

This function prunes weak rules and infrequent candidates in the $(p+1)$-th layer of candidate tree. Let T_{p+1} be the $(p+1)$-layer of the candidate tree.

```
for each $n_i \in T_{p+1}$
    for each $A \in \mathcal{P}^p(A_i)$            //$A$ is a $p$-subpattern of $A_i$
    if $sup(A) = sup(A_i)$ then remove node $n_i$          //Corollary 1
        else for each $z_j \in Z(A_i)$
            if $sup(A_i \cup z_j) < \sigma$ then $Z(A_i) = Z(A_i) - z_j$
             // minimum support requirement
                else if $sup(A \cup \neg z_j) = sup(A_i \cup \neg z_j)$ then $Z(A_i) = Z(A_i) - z_j$
                 // Lemma 1
        if $Z(A) = \emptyset$ then remove node $n_i$
```

This is the most important part of the algorithm, as it dominates the efficiency of the algorithm. We prune a leaf from two aspects, frequent rule requirement and strong rule requirement. Let us consider a candidate n_i in the $(p+1)$-th layer of tree. To examine satisfaction of Corollary 1, we test support of pattern A_i stored in the leaf with the support of its subpatterns by Property 1. There may be many such subpatterns when size of A_i is large. However, we only need to compare its p-subpatterns since upward closure property. Hence, the number of such comparisons is bounded by $p+1$. Once we find that the support of A_i equals to the support of any of its p subpattern A, we remove the leaf from the candidate tree. So all its super patterns will not be generated in all deeper layers. In this way, the number of removed weak rules may increase at an exponential rate. Examination of satisfaction of Lemma 1 is in the similar way, but it is with respect to a particular target. That is, we only remove a target from the potential target set in the leaf. Pruning those infrequent patterns is the same as that in other association rule mining algorithms. In our experiments, we will show the efficiency of weak rule pruning in dense databases.

4 Experiment

We have implemented the proposed algorithms and evaluated them on 6 real world databases from UCL ML Repository [6]. For those databases having continuous attributes, we use Discretizer in [11] to discretize them.

We have mined the complete class association rule sets and the optimal class association rule set of all testing databases with the minimum confidence of 0.5 and the minimum support of 0.1. Here support is specified as *local support* that is defined to be the ratio of the support of a rule to the support of the rule's consequence, since significance of a rule depends much on how much proportion of occurrences of its consequence it accounts for. We generate the complete class association rule set by the same algorithm without weak rule pruning and strong rule selecting. We restrict the maximum layer of candidate trees to 4 because of the observation that too specific rules (with many conditions) usually have very limited prediction power in practice. In fact, the proposed algorithm performs more efficiently when there is no such restriction, and this is clear from the second part of our experiment. We do so in order to present competitive results, since rule length constraint is an effective way to avoid combinatorial explosion. Similar constraints have been used in practice, for example, [12] restricts the maximum size of the found rule set.

The comparisons of rule set size and time to generate between the complete class association rule set and optimal class association rule set are listed in Figure 1. It is easy to see that the size of a optimal class association rule set is much smaller than that of the corresponding complete rule set, on the average less than $\frac{1}{17}$ of that. Because the optimal class association rule set has the same prediction power as the complete class association rule set has, so this rule set size reduction is very impressive. Similarly, the time for generating rules is much shorter as well. We have obtained more than $\frac{3}{4}$ reduction of mining time on average. Moreover, using a smaller optimal class association rule set instead of a lager complete class association rule set as the input for finding predictive association rules, we will have more efficiency improvement for other data mining tasks too.

The core of our proposed algorithm is to prune weak rules. To demonstrate the efficiency of pruning stated in Lemma 1 and Corollary 1 on dense databases, we have illustrated the number of nodes in each layer of the candidate trees of two databases in Figure 2. In this experiment, we lift the restriction of maximum number of layers. We can see that the tree nodes explode at a sharp exponential rate without weak rule pruning. In contrast, tree nodes increase slowly with weak rule pruning, reach a low maximum quickly, and then decrease gradually. When a pruning tree (weak rule pruning) stops growing, its corresponding unpruned tree just passes its maximum. In the deep tree level, after 4 in our case, the nodes being pruned are more than 99%. This shows how much redundancy we have eliminated. In our experiment, more than 95% time is used for such redundant computing when there is no maximum layer restriction. Considering that how much time it will take if we compute strong rules after obtaining all class association rules, we can see how effective our proposed weak rule pruning criterion is. Besides, from this detailed illustration of candidate tree growing

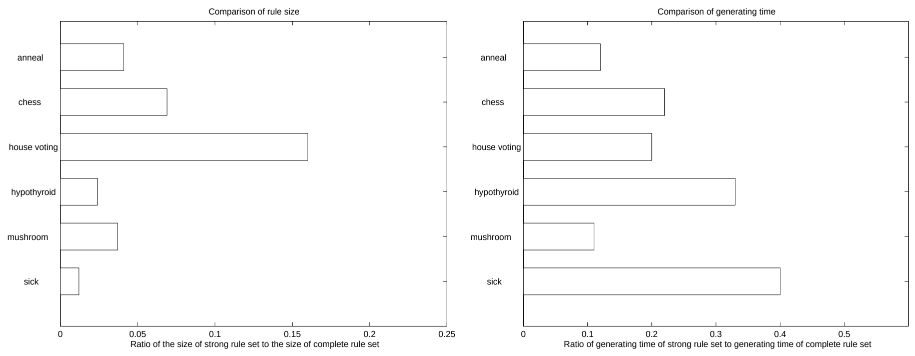

Fig. 1. Overall comparisons of rule size and generating time between R_o and R_c (in the ratio of R_o to R_c)

without length restriction, we can understand that the proposed algorithm will perform more efficiently when there is no maximum layer number restriction in comparison with mining the complete class association sets.

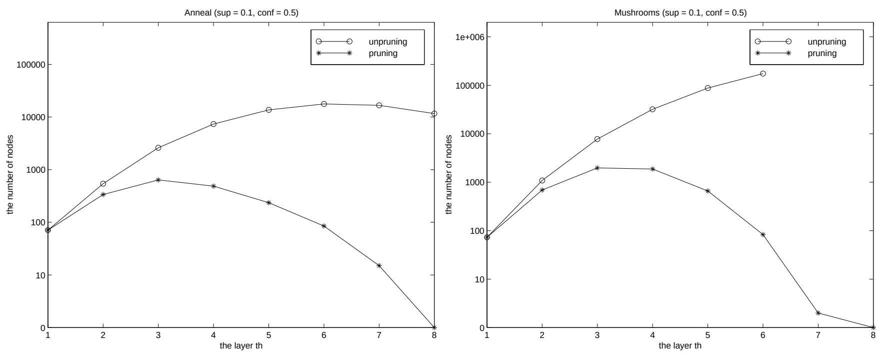

Fig. 2. Comparison of the number of candidates before and after weak rule pruning

5 Conclusion

In the paper, we studied an important problem of efficiently mining predictive association rules. We defined the optimal class association rule set, which preserves all prediction power of the complete class association rule set and hence can be used as a replacement of the complete class association rule set for finding predictive association rules. We developed a criterion to prune weak rules before they are actually generated, and presented an efficient algorithm to mine the optimal class association rule set. Our algorithm avoids redundant computation required in mining the complete class association rule set, and hence improves efficiency of the mining process significantly. We implemented the proposed algorithm and evaluated it on some real world databases. Our experimental results show that the optimal class association rule set has a much smaller size and requires much less time to generate than the complete class association rule set. It was also shown that the proposed criterion is very effective for pruning weak rules in dense databases.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive database s. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, 1993.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Fayyad U. and et al, editors, Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.
3. Kamal Ali, Stefanos Manganaris, and Ramakrishnan Srikant. Partial classification using association rules. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 115. AAAI Press, 1997.
4. Roberto Bayardo and Rakesh Agrawal. Mining the most interesting rules. In Surajit Chaudhuri and David Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, N.Y., August 15–18 1999. ACM Press.
5. Roberto Bayardo, Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense database. In *Proc. of the 15th Int'l Conf. on Data Engineering*, pages 188–197, 1999.
6. E. Keogh C. Blake and C. J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
7. Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):265.
8. P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Y. Kodratoff, editor, Machine Learning - EWSL-91*, 1991.
9. H. Toivonene M. Klemettinen P RonKainen K Hatonen and H Mannila. Pruning and grouping discovered association rules. Technical report, Department of Computer Science, University of Helsinki, Finland (`ftp.cs.helsinki.fi/pub/Reports/by_Project/PMDM/`), 1998.
10. M. Houtsma and A. Swami. Set-oriented mining of association rules in relational databases. In *11th Intl. Conf. data Engineering*, 1995.
11. Ron Kohavi, Dan Sommerfield, and James Dougherty. Data mining using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 234–245. IEEE Computer Society Press, 1996. Received the best paper award.
12. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *SIGKDD 98*, pages 80 – 86, 1998.
13. Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *SIGKDD 99*, 1999.
14. H. Mannila, H. Toivonen, and I. Verkamo. Efficient algorithms for discovering association rules. In *AAAI Wkshp. Knowledge Discovery in Databases*, July 1994.
15. Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
16. Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, volume 27,2 of *ACM SIGMOD Record*, pages 13–24, New York, June1–4 1998. ACM Press.
17. J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *ACM SIGMOD Intl. Conf. Management of Data*, May 1995.
18. J. R. Quinlan. *C4. 5: Programs for Machine Learning.* MK, San Mateo, CA, 1993.