Knowledge-Based Systems 15 (2002) 399–405

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

# Mining the optimal class association rule set

Jiuyong Li[a,*], Hong Shen[b], Rodney Topor[c]

[a]*Dept. of Mathematics and Computing, University of Southern Queensland, Toowoomba, QLD 4350, Australia*
[b]*Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Patsunokuchi, Ishikawa, 923-1292, Japan*
[c]*School of Computing and Information Technology, Griffith University, Nathan, Qld 4111, Australia*

Received 2 April 2001; accepted 22 November 2001

**Abstract**

We define an optimal class association rule set to be the minimum rule set with the same predictive 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 1/17 of the complete class association rule set on average, in significantly less rime than generating the complete class association rule set. Our proposed criterion has been shown very effective for pruning weak rules in dense databases. 

*Keywords:* Data mining; Association rule mining; Class association rule set

## 1. Introduction

### 1.1. 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 relational databases, and the consequences of rules are in pre-specified column, class attribute. Clearly, a relational database can be mapped to a transaction database when an attribute and attribute value pair is considered as an item. After having mapped a relational database into a transaction database, a *class association rule* set is a subset of association rules with the specified classes as their consequences, and a predictive association rule set is a small subset of class association rule set. 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 Ref. [1].
- 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 of a class association rule set that makes predictions as accurately as the class association rule set does, so that this subset can replace the class association rule set. Our proposed optimal class association rule set is the smallest subset with the same predictive power, which will be formally defined in Section 2, of the complete class association rule set. We present an efficient algorithm to directly generate the optimal class association rule set by taking the advantage of upward closure properties of weak

* Corresponding author.
*E-mail addresses:* jiuyong@usq.edu.au (J. Li), shen@jaist.ac.jp (H. Shen), rwt@cit.gu.edu.au (R. Topor).

0950-7051/02/$ - see front matter 
PII: S0950-7051(02)00024-2

rules that will be discussed in this paper. A clear usefulness of the work is that we can obtain predictive association rules from the optimal class association rule set instead the class association rule set that may not be available because of its expensive computational cost from dense databases.

### 1.2. Related work

Mining association rules [2] is a central task of data mining and has shown applications in various areas [1,3,4]. Currently most algorithms for mining association rules are based on Apriori [5], and used the so-called 'downward closure' property which states that all subsets of a frequent itemset must be frequent. Example of these algorithms can be found in Refs. [6–8]. A symmetric expression of downward closure property is *upward closure* property—all supersets of an infrequent itemset must be infrequent. We will use this term throughout the paper.

Finding classification rules has been an important research focus in the machine learning community [9,10]. Mining classification rules can be viewed as a special form of mining association rules, since a set of association rules with pre-specified consequences (classes) can be used for classification. Techniques for mining association rules have already been applied to mining classification rules [1,4]. Particularly, results in Ref. [1] are very encouraging, since it can build more accurate classifiers than those from C4.5 [9]. However, the algorithm in Ref. [1] 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. This paper will show that we can use a much smaller class association rule set to replace this set while not losing accuracy (predictive power).

Generally speaking, class association rule set is a type of target-constraint association rules. Constraint rule sets [11] and interesting rule sets [12] 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 classes, especially when the number of classes is large. Our optimal class association rule set differs from these rule sets at that it is minimal in size and keeps all predictive power. We propose an algorithm that finds this rule set with respect to all classes 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 Ref. [13–15].

### 1.3. Contributions

Contributions in this paper are the following.

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 predictive power as the complete class rule set, and can be used as a substitute of the complete class association rule set.

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 interesting rule mining algorithms, our algorithm finds strong (optimal) rules with all possible classes at one time.

## 2. Optimal class association rule set

Given a relational 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 a class (a value of pre-specified class attribute), named *consequence*. Exactly, the consequence is an attribute and value pair, but in class association rule mining, the class 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 $\mathrm{conf}(A \Rightarrow c)$.

A class association rule is defined to be an implication with a class as its consequence and its support and confidence are above the given thresholds from a database, respectively. Given a class attribute, the minimum support $\sigma$ and the minimum confidence $\psi$, 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 predictive 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 $cond(r) \subseteq t$, denoted by $r(T) \rightarrow cons(r)$. If $cons(r)$ is the class 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 predictive 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 [16]: $\mathrm{acc}_t(r) = \mathrm{acc}_s \pm z_N\sqrt{\mathrm{acc}_s(1-\mathrm{acc}_s)/n}$, where $\mathrm{acc}_t$ is the true (predictive) accuracy, $\mathrm{acc}_s$ 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 predictive accuracy of a rule. That is $\mathrm{acc}(r) = \mathrm{conf}(r) - z_N\sqrt{\mathrm{conf}(r)(1-\mathrm{conf}(r))/|\mathrm{cov}(r)|}$, where $\mathrm{cov}(r)$ is the covered set of rule $r$ that is defined in Section 3. We note that $\sqrt{\mathrm{conf}(r)(1-\mathrm{conf}(r))/n}$ is symmetry about $\mathrm{conf}(r) = 0.5$. Hence, we expect that the minimum confidence is at least 0.5 by using this definition. This minimum confidence is reasonably low in most applications. If $n < 30$, then we use Laplace accuracy instead [10], that is $\mathrm{acc}(r) = \mathrm{sup}(r) * |D| + 1/|\mathrm{cov}(r)| + p$, where $p$ is the number of classes.

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

In the following part, we will discuss predictions made by a rule set, and how to compare the predictive 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 predictive accuracy of all $r_i$ where $cond(r_i) \subseteq t$. 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 predictive 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 the accuracy of zero.

To compare predictive power of two rule sets, we define

**Definition 1** (Predictive 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 predictive 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 predictive 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 $\kappa_1$, then there must be another rule $r_2 \in R_2$ so that $r_2(T) \rightarrow c$ 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 $r_2 \subset r_1 \wedge \mathrm{acc}(r_2) \geq \mathrm{acc}(r_1)$, 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 (maximal) *strong* if there is no other rule in $R$ that is stronger than it is. Otherwise, the rule is *weak*. Thus, we call the set of all strong rules as optimal class association rule set. More specifically,

**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 $\sigma$ and minimum confidence $\psi$. Then $R_o(\sigma, \psi)$ contains all strong rules from the complete class association rule set $R_c(\sigma, \psi)$.

Finally, we consider the predictive 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 predictive 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, it is clear that $R_c \geq R_o$. Since $R_o$ consists of all strong rules, for any input, $R_o$ give the same prediction as $R_c$ does with at least the same predictive accuracy. Hence, $R_o \geq R_c$. As a result, the optimal class association rule set has the same predictive 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 $R_o$, $R'_o = R_o \backslash r$, and $R'_o$ has the same predictive power as $R_c$ has. From the definition, we know that there is no rule being stronger than rule $r$, so for a record that is fitted best by rule $r$, $R'_o$ cannot give the prediction on it as accurately as $R_o$ does. As a result, $R'_o$ cannot have the same predictive power as $R_c$, leading to contradiction. Hence, $R_o$ is the minimum rule set with the property of with the same predictive power as the complete class association rule set. □

The fact that the optimal class association rule set has the same predictive 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 predictive 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 predictive 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 Section 3, 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 upward closure properties for pruning weak rules

Let us begin with some definitions. We say that $r_1$ is a *sub rule* of $r_2$ if $cond(r_1) \subset cond(r_2) \wedge cons(r_1) = cons(r_2)$. On the opposite, $r_2$ is a *supper rule* of $r_1$. We define the *covered set* of rule $r$ to be the set of records containing antecedent of the rule, denoted by $\text{cov}(r)$. Covered set of a pattern $A$ is defined to be the set of records containing the pattern, denoted by $\text{cov}(A)$. It is clear that the covered set of a supper rule is a subset of the covered set of its sub 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.** $\text{cov}(X) \subseteq \text{cov}(Y)$ iff $\text{sup}(X) = \text{sup}(XY)$.

**Property 2.** $\text{cov}(X) \subseteq \text{cov}(Y)$ if $Y \subseteq X$.

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

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

**Proof.** We rewrite the confidence of rule $A \Rightarrow c$ as $\text{sup}(Ac)/\text{sup}(Ac) + \text{sup}(A \neg c)$. We know that function $f(u) = u/u + v$ is monotonically increasing with $u$ when $v$ is a constant. Since $\text{cov}(X \neg c) \subseteq \text{cov}(Y \neg c)$, we have $\text{sup}(X \neg c) = \text{sup}(XY \neg c)$. Noticing $\text{sup}(Xc) \geq \text{sup}(XYc)$, we obtain $\text{conf}(X \Rightarrow c) \geq \text{conf}(XY \Rightarrow c)$. Using relation $|\text{cov}(X \Rightarrow c)| \geq |\text{cov}(XY \Rightarrow c)|$, we have $\text{acc}(X \Rightarrow c) \geq \text{acc}(XY \Rightarrow c)$. As a result, $X \Rightarrow c > XY \Rightarrow c$.
Since $\text{cov}(XZ \neg c) \subseteq \text{cov}(YZ \neg c)$ for all $Z$ if $\text{cov}(X \neg c) \subseteq \text{cov}(Y \neg c)$, we have $XZ \Rightarrow c > XYZ \Rightarrow c$ for all $Z$.
Consequently, $XY \Rightarrow c$ and all its supper 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 $\text{cov}(X) \subseteq \text{cov}(Y)$, then $XY \Rightarrow c$ *and all its supper rules must be weak for all* $c \in C$.

**Proof.** This can be proved by noticing that $\text{cov}(X \neg c) \subseteq \text{cov}(Y \neg c)$ for all $c \in C$ if $\text{cov}(X) \subseteq \text{cov}(Y)$. □

We can understand the corollary in the following way: we cannot combine a super concept with a sub concept to make the sub concept 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, the covered set of a pattern (e.g. $X$) is a subset of that of its $|X| - 1$ cardinality sub pattern. So, we can always compare the support of a $k$-pattern with that of its $(k-1)$-sub patterns in order to decide whether the $k$-pattern should be removed.

Since both Lemma 1 and Corollary 1 state upward closure property of weak rules, we can have an efficient algorithm to find all strong rules.

### *3.1. 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 the 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.

### *3.2. Storage structure*

In this proposed algorithm, we use an extended prefix tree $\{V, E\}$, called *candidate tree*. $V = [r, v_1, v_2, \ldots, v_n]$ is a sorted set where $r$ is the root and $v_i < v_j$ if $i < j$. For all $\{v_i, v_j\} \in E$, there is $v_i < v_j$.

In our algorithm, $V$ is a set of all attribute and value pairs, and sorted by their first references. A node of the candidate tree consists of $\{A, Z, Q\}$. $A$ is a set of attribute and value pairs in the path from the root to the node, and is the antecedent of a possible rule. Since $A$ is unique in a candidate tree, we use it as identity of the node. The potential target set $Z$ is a set of classes that may be consequences of $A$. $Q$ a supset of possible attribute and value pair sets, for each class (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, $Q_j \in Q$.

Our algorithm is given as follows.

*Algorithm: optimal class association rule set miner*

Input: database $D$ with class attribute $C$, the minimum support $\sigma$ and the minimum confidence $\psi$.

Output: optimal class association rule set $R$.
Set optimal class association rule set $R = \emptyset$
Count support of 1-patterns
Initialize 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 $n_i$ denotes 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 $\mathrm{P}^p(A_k)$ 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 $n_j(v_{n_j} > v_{n_i})$
    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 $\exists A \in \mathrm{P}^p(A_k)$ such that $\sup(A \cup z) \le \sigma$
        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 initialize the new node by manipulating information from the two nodes. Next, we prune unqualified candidate. If any of its $p$-sub patterns cannot get enough support with any of the possible targets (classes), then we remove the class 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 $A \in \mathrm{P}^p(A_i)$ // $A$ is a $p$-sub pattern 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 $Z(A_i) = Z(A_i) - z_j$
        // the minimum support requirement
          else if $\sup(A \cup \neg z_j) = \sup(A_j \cup \neg z_j)$
            then $Z(A_i) = Z(A_i) \backslash z_j$
// 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 sub patterns by Property 1. There may be many such sub patterns when size of $A_i$ is large. However, we only need to compare its $p$-sub patterns since upward closure property of weak rules. 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 $p$ sub pattern $A$, 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 (class). That is, we only remove a target (class) 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 how effective the weak rule pruning is in dense databases.

## 4. Experimental results

We have implemented the proposed algorithms and evaluated them on six real world databases from UCI ML

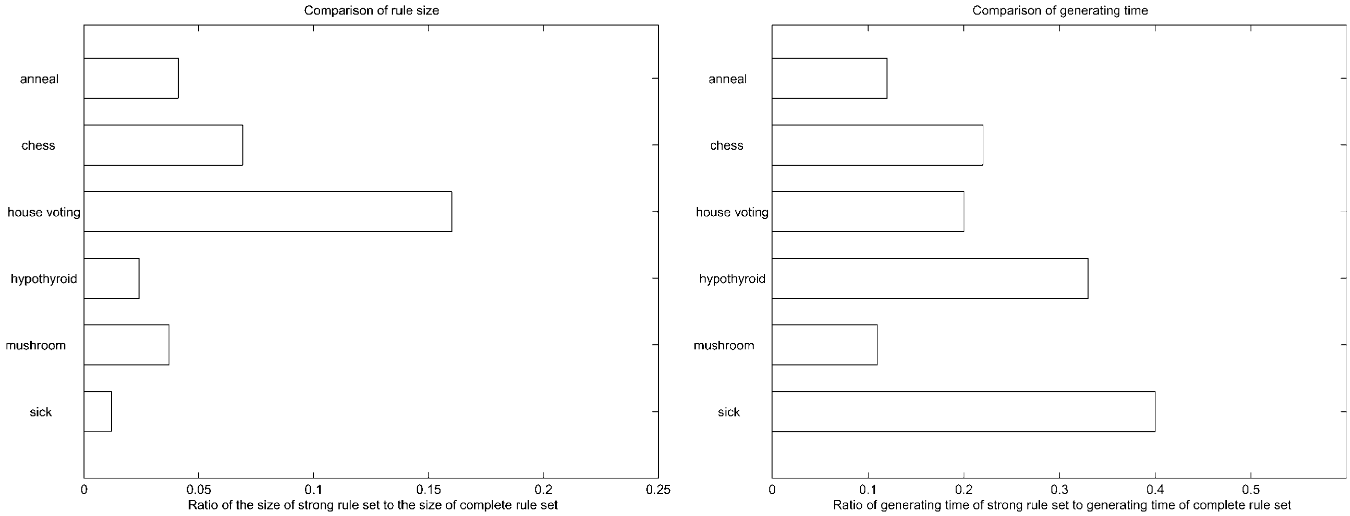


Fig. 1. Overall comparisons of rule size and generating time between $R_o$ and $R_c$ (in the ratio of $R_o$ to $R_c$).

repository [17]. For those databases having continuous attributes, we use discretizer in Ref. [18] to discretize them.

We have mined the complete class association rule set 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 the 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 four because of the observation that too specific rules (with many conditions) usually have very limited predictive 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, Ref. [1] 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 Fig. 1. It is easy to see that the size of an optimal class association rule set is much smaller than that of the corresponding complete rule set, on the average less than 1/17 of that. Because the optimal class association rule set has the same predictive 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 3/4 reduction of mining time on average. Moreover, using a smaller optimal class association rule set instead of a larger complete class association rule set as

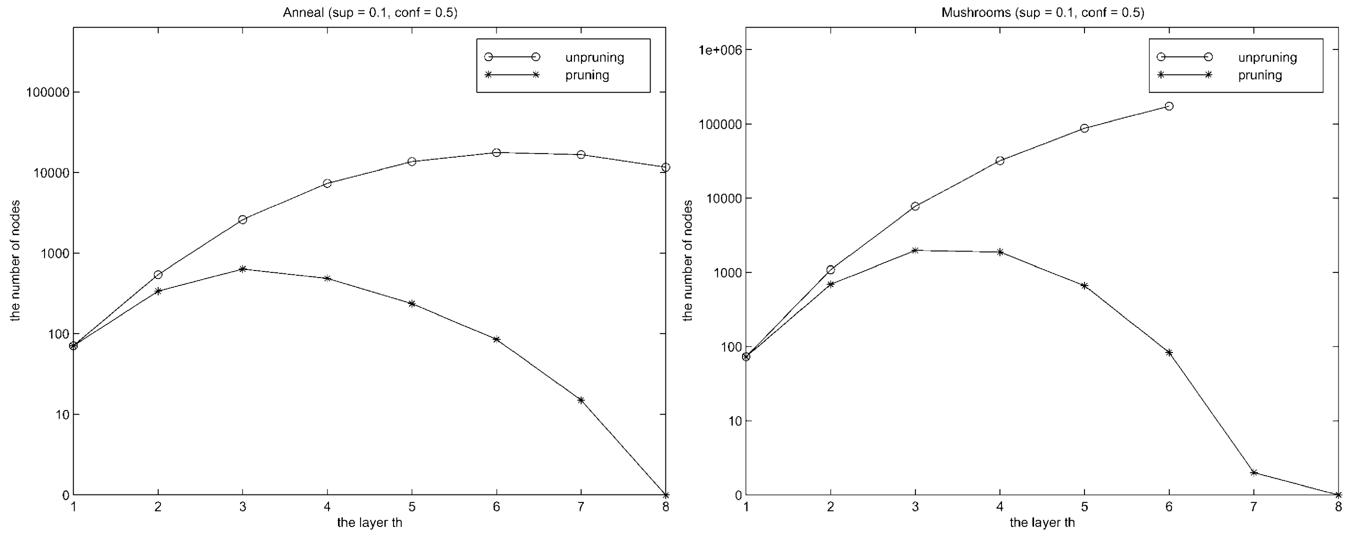


Fig. 2. Comparison of the number of candidates before and after weak rule pruning.

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 Fig. 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 un-pruned tree just passes its maximum. In the deep tree level, after four 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 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.

## 5. Conclusion

In the paper, we studied the problem of efficiently mining optimal class association rules. We defined the optimal class association rule set, which preserves all predictive 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 much 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] B. Liu, W. Hsu, Y. Ma, Integrating classification and association rule mining, SIGKDD 98, 1998, pp. 80–86.

[2] R. Agrawal, T. Imielinski, A. Swami, Mining associations between sets of items in massive databases, Proceedings of the ACM SIGMOD International Conference on Management of Data, 1993.

[3] S. Brin, R. Motwani, C. Silverstein, Beyond market baskets: generalizing association rules to correlations, SIGMOD Record (ACM Special Interest Group on Management of Data) 26 (2) (1997) 265.

[4] K. Ali, S. Manganaris, R. Srikant, Partial classification using association rules, in: D. Heckerman, H. Mannila, D. Pregibon, R. Uthurusamy (Eds.), Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), AAAI Press, Menlo Park, CA, 1997, p. 115.

[5] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo, Fast discovery of association rules, in: U. Fayyad (Ed.), Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, 1996.

[6] M. Houtsma, A. Swami, Set-oriented mining of association rules in relational databases, 11th International Conference Data Engineering, 1995.

[7] H. Mannila, H. Toivonen, I. Verkamo, Efficient algorithms for discovering association rules, AAAI Workshop Knowledge Discovery in Databases, July, 1994.

[8] J.S. Park, M. Chen, P.S. Yu, An effective hash based algorithm for mining association rules, ACM SIGMOD International Conference Management of Data, May, 1995.

[9] J.R. Quinlan, C4. 5: Programs for Machine Learning, MK, San Mateo, CA, 1993.

[10] P. Clark, R. Boswell, Rule induction with CN2: Some recent improvements, in: Y. Kodratoff (Ed.), Machine Learning—EWSL-91, 1991.

[11] R. Bayardo, R. Agrawal, D. Gunopulos, Constraint-based rule mining in large, dense database, Proceedings of the 15th International Conference on Data Engineering, 1999, pp. 188–197.

[12] R. Bayardo, R. Agrawal, Mining the most interesting rules, in: S. Chaudhuri, D. Madigan (Eds.), Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 1999, pp. 145–154, August 15–18.

[13] H. Toivonene, M. Klemettinen, P. RonKainen, K. Hatonen, H. Mannila, Pruning and grouping discovered association rules, Technical report, Department of Computer Science, University of Helsinki, Finland, 1998 (ftp.cs.helsinki.fi/pub/Reports/by_Project/PMDM/).

[14] B. Liu, W. Hsu, Y. Ma, Pruning and summarizing the discovered associations, SIGKDD 99, 1999.

[15] T.Ng. Raymond, L.V.S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained associations rules, Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98), ACM SIGMOD Record, Vol. 27(2), ACM Press, New York, 1998, pp. 13–24, June 1–4.

[16] M.T. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.

[17] E. Keogh, C. Blake, C.J. Merz, UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[18] R. Kohavi, D. Sommerfield, J. Dougherty, Data mining using MLC++: a machine learning library in C++, Tools with Artificial Intelligence, IEEE Computer Society Press, Silver Spring, MD, 1996, pp. 234–245, Received the best paper award.