Prepublication draft, to appear in *Data Mining and Knowledge Discovery*, **10**(1), 2005, pp 39-79.

K-Optimal Rule Discovery

Geoffrey I. Webb (webb@infotech.monash.edu.au)
School of Computer Science and Software Engineering,
PO Box 75, Monash University,
Melbourne, Victoria 3800 Australia
Telephone +61 3 9905 3296 Facsimile +61 3 99055146

Songmao Zhang (szhang@nlm.nih.gov)
US National Library of Medicine (LHC/CgSB)
National Institutes of Health
8600 Rockville Pike, MS 43 (Bldg 38A, B1N28T)
Bethesda, MD 20894 USA

Abstract. *K*-optimal rule discovery finds the *k* rules that optimize a user-specified measure of rule value with respect to a set of sample data and user-specified constraints. This approach avoids many limitations of the frequent itemset approach of association rule discovery. This paper presents a scalable algorithm applicable to a wide range of *k*-optimal rule discovery tasks and demonstrates its efficiency.

Keywords: Exploratory Rule Discovery, Association Rules, Classification Rules, Rule Search, Search Space Pruning

1. Introduction

Association rule discovery (Agrawal and Srikant, 1994; Agrawal et al., 1996) is an enduring and popular data mining technology. It differs from conventional machine learning techniques by finding all rules that satisfy some set of constraints, rather than finding a single model that bests fits the sample data. This proves useful in many data mining contexts as it empowers the user to select between the many potential models that the data may support. However, association rule discovery is based on the application of a minimum support constraint. This constraint is used to prune the search space and make computation feasible. This can be limiting, as support is often not directly related to the potential value of the rule. One example of this is the so called *vodka and caviar problem* (Cohen et al., 2000). Ketel vodka and Beluga caviar are low sales-volume products and hence a correlation between purchases of them is likely to have low support. Nonetheless, a strong correlation between them may be of considerable interest as they are high profit items and hence the affinity is likely to represent substantial benefit. Where support is not directly related to the potential value of a rule, the application of a minimum support constraint carries a risk that the most valuable rules will not be discovered. A further limitation of the use of minimum support as the primary constraint on the rules to be discovered is that it is often not possible to predict a minimum support threshold that will result in a useful number of rules. Set the threshold too high and very few rules will be produced. There is often a narrow range of minimum support values below which the number of rules produced becomes extraordinarily large and unmanageable (Zheng et al., 2001).

The OPUS_AR algorithm (Webb, 2000) presented an alternative approach to such exploratory rule discovery that we call *k-optimal rule discovery*. Under this approach the user specifies a rule value measure λ, a set of constraints $\mathcal{M}$ and the number of rules to be discovered, k. The system then returns the k rules that optimize λ within constraints $\mathcal{M}$. This extends previous techniques that have sought the single rule that optimizes a value measure for a pre-specified consequent (Webb, 1995; Bayardo and Agrawal, 1999). In contrast, the new algorithm finds multiple rules and allows any condition in the role of consequent. This paper provides a formal definition of the k-optimal rule discovery task, presents a refined version of the OPUS_AR algorithm, proves the correctness of the algorithm, presents a number of pruning mechanisms and proves their correctness, and evaluates the efficiency of the algorithm and pruning mechanisms with respect to a number of widely studied rule discovery tasks.

2. Constraint-Satisfaction Rule Discovery

We define both association rule discovery and k-optimal rule discovery in terms of a generic description of rule discovery under constraints, defining what we call a *constraint satisfaction rule discovery task* (*CStask*). A *record* is an entity to which predicates called *conditions* apply. For notational convenience we treat a record d as denoting the set of conditions that apply to d. A database $\mathcal{D}$ is a nonempty collection of records. For any set of conditions S and database $\mathcal{D}$,

$$\text{coverset}(S, \mathcal{D}) = \{d \in \mathcal{D} \mid d \supseteq S\}, \text{and} \tag{1}$$

$$\text{cover}(S, \mathcal{D}) = \frac{|\,\text{coverset}(S, \mathcal{D})|}{|\mathcal{D}|}. \tag{2}$$

Where the database is implied by the context we abbreviate the former to coverset(S) and the latter to cover(S). A *CStask* is a 3-tuple $\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle$, where

$\mathcal{C}$: is a nonempty set of conditions;

$\mathcal{D}$: is a database of records $r \subseteq C$;

$\mathcal{M}$: is a set of constraints on the rules that form the solution for the *CStask*;

A *rule* is a pair of sets of conditions denoted by $X \rightarrow Y$, where

X: is a nonempty set of conditions called the *antecedent*; and

Y: is a nonempty set of conditions called the *consequent*.

The predicate satisfies($X \rightarrow Y, \mathcal{D}, \mathcal{M}$) is true of all and only rules $X \rightarrow Y$ that satisfy all constraints in $\mathcal{M}$ with respect to $\mathcal{D}$. CSsolution : $\{\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle\} \rightarrow \{\{X \rightarrow Y\}\}$ is a many-to-one function from *CStask*s to their solutions, satisfying

$$\begin{aligned}&\text{CSsolution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle)\\ &\quad = \{X \rightarrow Y \mid X \subseteq \mathcal{C} \wedge Y \subseteq \mathcal{C} \wedge \text{satisfies}(X \rightarrow Y, \mathcal{D}, \mathcal{M})\}.\end{aligned}$$

Constraint-satisfaction rule discovery provides a generic formal framework in which it is possible to describe, analyze and contrast a wide range of exploratory rule discovery techniques.

Note that the feasibility of computing a solution to a constraint satisfaction rule discovery task will depend critically upon the number of rules that satisfy $\mathcal{M}$ with respect to $\mathcal{D}$. For example, a given set of constraints may fail to exclude any rules with respect to a given $\mathcal{D}$, in which case the number of rules and hence time complexity will be exponential with respect to the number of conditions in $\mathcal{C}$.

3. Association Rule Discovery

Association rule discovery (Agrawal and Srikant, 1994; Agrawal et al., 1996) can be described as a form of *CStask*. This approach to rule discovery overcomes the computational complexity of an unconstrained *CStask* by imposing a constraint on the minimum allowed value for *support*, where

$$\text{support}(X \rightarrow Y, \mathcal{D}) = \text{cover}(X \cup Y, \mathcal{D}).$$

Early association rule discovery used an additional constraint on the minimum value for *confidence*, defined in Section 4.1, but subsequent work has substituted other auxiliary constraints such as *lift*, also defined in Section 4.1.

Most association rule discovery algorithms utilize the frequent itemset strategy as exemplified by the Apriori algorithm (Agrawal et al., 1993). The frequent itemset strategy first discovers all *frequent itemsets* $\{I \subseteq \mathcal{C} \mid \text{support}(I, \mathcal{D}) \geq min\_support\}$, those sets of conditions whose support exceeds a user defined threshold *min_support*. Association rules are then generated from these frequent itemsets. This approach is very efficient if there are relatively few frequent itemsets. It is, however, subject to a number of limitations.

1. There may be no natural lower bound on support. Associations with support lower than the nominated *min_support* will not be discovered. Infrequent itemsets may actually be especially interesting for some applications. As illustrated by the vodka and caviar problem, in many applications high value transactions are likely to be both relatively infrequent and of great interest.

2. Even if there is a natural lower bound on support, the analyst may not be able to identify it. If *min_support* is set too high then important associations will be overlooked. If it is set too low then processing may become infeasible. There may be no means of determining, even after an analysis has been completed, whether important associations have been overlooked due to the lower bound on support being set too high.

3. Even when a relevant minimum support can be specified, the number of frequent itemsets may be too large for computation to be feasible. Many datasets are infeasible to process using the frequent itemset approach with sensible specifications of minimum support (Bayardo, 1998).

4. The frequent itemset approach does not readily admit to techniques for improving efficiency by using constraints on the properties of rules that cannot be derived directly from the properties only of the antecedent, the consequent, or the union of the antecedent and consequent. Thus, they can readily benefit from a constraint on support (which depends solely on the frequency of the union of the antecedent and consequent) but cannot readily benefit from a constraint on confidence (which relates to the relationship between the support of the antecedent and of the union of the antecedent and the consequent). Where such constraints can be specified, potential efficiencies are lost.

An extension of the frequent itemset approach allows *min_support* to vary depending upon the items that an itemset contains (Liu et al., 1999). While this introduces greater flexibility to the frequent itemset strategy, it does not resolve any of the four issues identified above.

Most research in association rule discovery has sought to improve the efficiency of the frequent itemset discovery process (Agarwal et al., 2000; Han et al., 2000; Savasere et al., 1995; Toivonen, 1996, for example). This has not addressed any of the above problems, except the closed itemset approaches (Pasquier et al., 1999; Pei et al., 2000; Zaki, 2000), that reduce the number of itemsets required, alleviating the problems of point 3, but not addressing 1, 2 or 4.

Note that auxiliary constraints, such as confidence and lift, typically reduce the number of rules produced, but have little impact on computational efficiency. This is due to the use of the frequent itemset strategy. Under this strategy it is the minimum support constraint that has the primary impact on the computational efficiency of the task.

4. K-optimal rule discovery

Webb (2000) outlines an alternative approach to rule discovery. Under this approach the user specifies a rule value measure λ, a set of constraints $\mathcal{M}$ and the number of rules to be discovered k. The system then returns the k rules that optimize λ with respect to the database $\mathcal{D}$ within constraints $\mathcal{M}$.

A k-optimal rule discovery task (*KOtask*) is a 5-tuple $\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$ where

$\mathcal{C}$: is a nonempty set of conditions;

$\mathcal{D}$: is a database of records $r \subseteq C$;

$\mathcal{M}$: is a set of constraints on the rules that form the solution for the task;

$\lambda : \{X \rightarrow Y\} \times \{\mathcal{D}\} \rightarrow \mathbb{R}$ is a function from rules and databases to real values and defines an *value* measure such that the greater $\lambda(X \rightarrow Y, \mathcal{D})$ the greater the (potential) value to the user of $X \rightarrow Y$ given the database; and

k: is an integer denoting the number of rules in the solution for the task.

solution : $\{\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle\} \rightarrow \{\{X \rightarrow Y\}\}$ is a many-to-many function mapping a *KOtask* to its solutions, satisfying

$\forall s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$

$$\begin{aligned}
&s \subseteq \text{CSsolution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle) \\
&\quad \wedge |s| \leq k; \\
&\quad \wedge \neg \exists r \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle) : |r| < |s| \\
&\quad \wedge \neg \exists X \rightarrow Y \in s, X' \rightarrow Y' \in (\text{CSsolution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle) - s) : \\
&\qquad\qquad \lambda(X \rightarrow Y, \mathcal{D}) < \lambda(X' \rightarrow Y', \mathcal{D}).
\end{aligned}$$

Note, it follows from this definition that if the CSsolution contains k or more rules then all solutions to the *KOtask* contain k rules. If the CSsolution contains fewer than k rules then there is only one solution to the *KOtask* and it is identical to the CSsolution. There may only be multiple solutions for a given *KOtask* if there are multiple rules with the same value for λ, each of which may take the k^{th} place in a solution for the task.

It is important to distinguish k-optimal rule discovery from Bayardo and Agrawal's (1999) system that finds all rules that are *sc-optimal*. Any rule that optimizes any of a wide range of common value measures including lift and leverage is sc-optimal. Finding all such rules may be useful if a user does not have any idea of what is an appropriate measure of rule value. In contrast, k-optimal rule discovery finds the k rules that optimize a specific measure of rule value. For many measures the highest rating such rule will be sc-optimal, but the second and subsequent rule may not be sc-optimal. If the user knows that they are interested in rules that maximize some specific measure such as lift or leverage, finding rules that maximize all of a continuum of other measures of rule value will only serve to obscure the rules of interest that are discovered.

4.1. Interestingness measures

Rule value measures are central to the enterprize of k-optimal rule discovery. Such measures will often not actually directly capture the true value of a rule, as this will often depend upon many factors that are difficult to formalize and incorporate in a computational process. Rather, what we for convenience refer to as *rule value measures* will usually be measures of potential value.

We define four such measures with respect to a rule $X \rightarrow Y$:

$$\text{coverage}(X \rightarrow Y, \mathcal{D}) = \text{cover}(X, \mathcal{D}),$$

$$\text{support}(X \rightarrow Y, \mathcal{D}) = \text{cover}(X \cup Y, \mathcal{D}),$$

$$\text{confidence}(X \rightarrow Y, \mathcal{D}) = \frac{\text{support}(X \rightarrow Y, \mathcal{D})}{\text{coverage}(X \rightarrow Y, \mathcal{D})},$$

$$\begin{aligned}&\text{leverage}(X \rightarrow Y, \mathcal{D})\\ &\quad = \text{support}(X \rightarrow Y, \mathcal{D}) - \text{cover}(X, \mathcal{D}) \times \text{cover}(Y, \mathcal{D}).\end{aligned}$$

Piatetsky-Shapiro (1991) argues that many measures of rule value are based on the difference between the observed joint frequency of the antecedent and consequent, support$(X \rightarrow Y)$, and the frequency that would be expected if the two were independent, cover$(X) \times$ cover(Y). He asserts that the simplest such measure is one that we call *leverage*, as defined above. This measure has also been called *interest*. Note that leverage can also be expressed as

$$\begin{aligned}&\text{leverage}(X \rightarrow Y, \mathcal{D})\\ &\quad = \text{cover}(X, \mathcal{D}) \times (\text{confidence}(X \rightarrow Y, \mathcal{D}) - \text{cover}(Y, \mathcal{D}))\,.\end{aligned}$$

Expressed in this form, it has also been called *weighted relative accuracy* (Todorovski et al., 2000).

Leverage is of interest because it measures the number of additional records that an interaction involves above and beyond those that should be expected if one assumes independence. This directly represents the volume of an effect and hence will often directly relate to the ultimate measure of interest to the user such as the magnitude of the profit associated with the interaction between the antecedent and consequent. This contrasts with the traditional association rule measure

$$\text{lift}(X \rightarrow Y) = \frac{\text{support}(X \rightarrow Y)}{\text{cover}(X) \times \text{cover}(Y)}$$

which is the ratio of the observed frequency with which the consequent occurs in the context of the antecedent over that expected if the two were independent. A rule with high lift may be of little interest because it has low coverage and hence applies in very few circumstances. This might be provided as justification for the application of minimum support constraints in conjunction with the lift measure, but this is at best a crude approximate fix to the problem. It results in a step function with the very undesirable property that the addition of one more record in support of a rule with very high lift can transform it from being of no interest whatsoever, because it fails to meet a minimum support criterion, to being of very high interest, because it now meets that criterion and has high lift. In contrast to the use of support and lift as measures of interest, the use of leverage ensures both that rules with very low support will receive low values and that there are no artificial threshold values at which extreme changes in the objective function occur. For these reasons we use leverage as the value measure λ for the purposes of this paper. Note, however, that the algorithm we present is

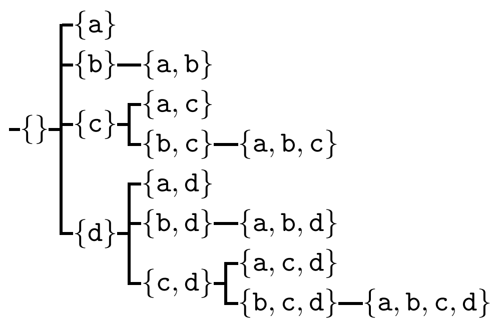

Figure 1. A fixed-structure search space

independent of this measure and that many of the auxiliary techniques proposed extend directly to other measures of rule value.

5. The OPUS Search Algorithm

We propose an algorithm for solving a useful class of k-optimal rule discovery tasks based on the OPUS (Webb, 1995) search algorithm. OPUS provides efficient search for subset selection, such as selecting a subset of available conditions that optimizes a specified measure. It was developed for classification rule discovery. Prior to the development of OPUS, classification rule discovery algorithms that performed complete search ordered the available conditions and then conducted a systematic search over the ordering in such a manner as to guarantee that each subset was investigated once only, as illustrated in Figure 1.

Critical to the efficiency of such search is the ability to identify and prune sections of the search space that cannot contain solutions to the search task. This is usually achieved by identifying subsets that cannot appear in a solution. For example, it might be determined that no superset of $\{b\}$ can be a solution in the search space illustrated in Figure 1. Under previous search algorithms (Clearwater and Provost, 1990; Morishita and Nakaya, 2000; Provost et al., 1999; Rymon, 1992; Segal and Etzioni, 1994), the search space below such a node was pruned, as illustrated in Figure 2. In this example, pruning removes one subset from the search space.

This contrasts with the pruning that would occur if all supersets of the pruned subset were removed from the search space, as illustrated in Figure 3. This optimal pruning almost halves the search space below the parent node.

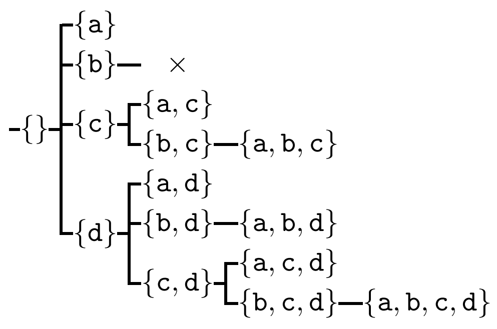

Figure 2. Pruning a branch from a fixed-structure search space

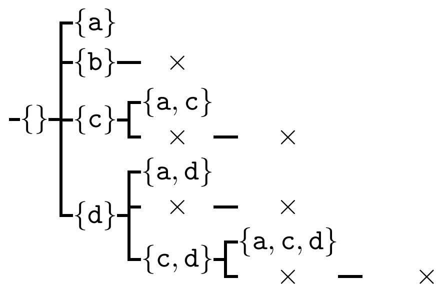

Figure 3. Pruning all nodes containing a single condition from a fixed-structure search space

OPUS achieves the pruning illustrated in Figure 3 by maintaining a set of available items at each node in the search space. When adding an item i to the current subset s results in a subset $s \cup \{i\}$ that can be pruned from the search space, i is simply removed from the set of available items at s that is propagated below s. As supersets of $s \cup \{i\}$ below s can only be explored after $s \cup \{i\}$, this simple mechanism with negligible computational overheads guarantees that no superset of a pruned subset will be generated in the search space below the parent of the pruned node. This greatly expands the scope of a pruning operation from that achieved by previous algorithms which only extended to the space directly beneath the pruned node. Further pruning can be achieved by reordering the search space (Webb, 1995). However, this proves to be infeasible in k-optimal rule discovery search as there is a large amount of information associated with each node in the search space (specifically, the set of records covered by the current antecedent) and it is more efficient to utilize this information when it is first calculated than to either store it for later use or recalculate it at a later stage as would be required if the nodes were reordered before expansion.

Note that nodes are pruned only when they need not be explored during the search. Nodes may need to be explored even when they are not candidate solutions because candidate solutions might occur deeper in the search space below those nodes. Pruning is applied only when it is possible to guarantee that the search space below a node cannot contain any rule in the solution to the discovery task. In consequence, OPUS does not require either monotonicity or anti-monotonicity from its objective function. It requires only that the value of the objective function can be bounded so that branch and bound techniques can exclude sections of the search space from exploration.

It is interesting to observe that while the rule discovery systems Max-Miner and Dense-Miner have been described as using SE-Tree search (Bayardo, 1998; Bayardo et al., 2000), they are perhaps more accurately described as using OPUS search, as they use both the propagation of a set of available items and search space reordering, two strategies that distinguish OPUS search from SE-Tree search.

6. The KORD Algorithm

The KORD algorithm extends the OPUS search algorithm to k-optimal rule discovery (Webb, 2000). OPUS supports search through spaces of subsets. Previous rule discovery tasks to which OPUS and related algorithms have been applied (Bayardo, 1998; Bayardo et al., 2000; Webb, 1995) have required that the consequent of each of the rules discovered be a fixed 'class' value. Hence it has been necessary only to explore the space of subsets of conditions that may form an antecedent for that consequent. In contrast, the k-optimal rule discovery task requires search through the space of pairs $\langle a \subseteq \mathcal{C}, c \in \mathcal{C} \rangle$, where a is the antecedent and c the consequent of rule. KORD achieves this by performing OPUS search through the space of antecedents, maintaining at each node a set of potential consequents, each of which is explored at each node.

KORD relies upon there being a measure of rule value and an upper limit k on the number of rules to be returned. These are used to prune the search space. This contrasts with the frequent itemset approach that uses a constraint on minimum support to prune the search space. KORD can also perform pruning by utilizing additional constraints such as the traditional association rule discovery constraints on support and confidence or lift. The solution to a KOtask is the set of rules that optimize the measure of value within those that satisfy all the constraints. Note that it may not be apparent when a rule is encountered whether or not it is in the solution. For example, if we are seeking the 100 rules with the highest leverage, we may not know the cutoff value for leverage until the search has been completed.

To manage this problem, KORD is restricted to KOtasks $\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$ that satisfy the following property:

$$\begin{aligned} &\forall R \subseteq \{W \rightarrow Z \mid W \subseteq \mathcal{C} \land Z \subseteq \mathcal{C}\}, \\ &\quad \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle) \cap R \\ &\qquad \subseteq \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in R, \lambda, k \rangle) \end{aligned} \tag{3}$$

This condition states that for any subset R of the search space over which the KOtask is performed, the set of rules from R that are contained in the solution to the KOtask must be a subset of the solution to an otherwise identical KOtask for which the solution is constrained to rules in R. This allows an incremental search to be performed. In this search, rules in the search space are considered one at a time. Let S represent the set of rules in the search space explored so far. A record *currentSolution* is maintained such that $\mathit{currentSolution} = \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in S, \lambda, k \rangle)$. For each new rule r considered during the search, the algorithm need only update *currentSolution* by

– adding r if $r \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in (S \cup \{r\}), \lambda, k \rangle)$, and

– removing any $z \in \mathit{currentSolution}$ if $z \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in (S \cup \{r\}), \lambda, k \rangle)$.

Appendix A presents a proof that the update strategy for *currentSolution* is sufficient and necessary to ensure that on termination $\mathit{currentSolution} = \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$.

This simple constraint, (3), greatly simplifies the search task. The constraint is not unduly restrictive as it is satisfied by all λ and $\mathcal{M}$ that we have considered to date. Note in particular that it allows monotone and anti-monotone constraints, as well as constraints that are neither. Our algorithm uses branch-and-bound techniques, and hence relies on neither monotonicity nor anti-monotonicity in its constraints. The objective function, leverage, used in our experiments is non-monotone. The addition of a new condition to an antecedent can raise, lower or leave unaffected the leverage of a rule.

We limit the rules discovered to rules with a single condition in the consequent. This is because

– in our experience users have primarily been interested in rules with consequents containing a single condition,

- for many applications the introduction of rules with multiple condition consequents greatly increases the number of rules discovered for any given minimum value of a rule value measure, and
- restricting the search space to single element consequents greatly decreases the amount of computation required for rule discovery.

Table I displays the algorithm that results from extending the OPUS search algorithm (Webb, 1995) to obtain efficient search for this rule discovery task. The algorithm is presented as a recursive procedure with three arguments:

CurrentLHS: the set of conditions in the antecedent of the rule currently being considered.

AvailableLHS: the set of conditions that may be added to the antecedent of rules to be explored below this point

AvailableRHS: the set of conditions that may appear on the consequent of a rule in the search space at this point and below

The algorithm also maintains a global variable *currentSolution*, the solution to the KOtask constrained to the rules in the search space considered so far.

To solve KOtask $\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$, *currentSolution* is initialized to $\emptyset$ and the initial call to KORD is made with CurrentLHS=$\emptyset$, AvailableLHS=$\mathcal{C}$, and AvailableRHS=$\mathcal{C}$.

The KORD algorithm is a search procedure that starts with rules with one condition in the antecedent and searches through successive rules formed by adding conditions to the antecedent. It loops through each condition in AvailableLHS and adds it to CurrentLHS to form the NewLHS. For the NewLHS, it loops through each condition c in AvailableRHS to check whether NewLHS$\rightarrow\{c\}$ might be in the solution. After the AvailableRHS loop, the procedure is recursively called with the arguments NewLHS, NewAvailableLHS and NewAvailableRHS. The two latter arguments are formed by removing the pruned conditions from AvailableLHS and AvailableRHS, respectively.

Pruning rules seek to identify areas of the search space that cannot contain a solution. This is represented within the algorithm by the use of a predicate insolution($a\rightarrow c, T$) that is true iff the rule $a\rightarrow c$ is in the solution to the KOtask, T. The predicate proven(X) is true iff pruning rules provided to the algorithm prove the proposition X. The use of this predicate allows us to abstract the algorithm from the sets of pruning rules that might be used to provide efficient search for a given set of constraints. The efficiency of KORD will depend critically on the power of the pruning rules with which it is provided.

Table I. The *k*-optimal rule discovery algorithm

```
Algorithm: KORD(CurrentLHS,AvailableLHS,AvailableRHS,$\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)

 1: SoFar := $\emptyset$
 2: for all P in AvailableLHS do
 3:   if $\neg$proven($\forall x \subseteq$ AvailableLHS, $y \in$ AvailableRHS :
      $\neg$insolution($x \cup$ CurrentLHS $\cup$ {P}$\rightarrow y$, $\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)) then
 4:     NewLHS := CurrentLHS $\cup$ {P}
 5:     NewAvailableLHS := SoFar
 6:     if $\neg$proven($\forall x \subseteq$ NewAvailableLHS : $\forall y \in$ AvailableRHS :
        $\neg$insolution($x \cup$ NewLHS$\rightarrow y$, $\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)) then
 7:       NewAvailableRHS := AvailableRHS - P
 8:       if $\neg$proven($\forall y \in$ NewAvailableRHS :
          $\neg$insolution(NewLHS$\rightarrow y$, $\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)) then
 9:         for all Q in NewAvailableRHS do
10:           if proven($\forall x \subseteq$ NewAvailableLHS :
              $\neg$insolution($x \cup$ NewLHS$\rightarrow$Q, $\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)) then
11:             NewAvailableRHS := NewAvailableRHS - Q
12:           else
13:             if $\neg$proven($\neg$insolution(NewLHS$\rightarrow$Q, $\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)) then
14:               if insolution(NewLHS$\rightarrow$Q, $\langle\mathcal{C},\mathcal{D},\mathcal{M} \wedge X\rightarrow Y \in$
                  currentSolution $\cup$ {NewLHS$\rightarrow$Q}, $\lambda,k\rangle$) then
15:                 add NewLHS $\rightarrow$ Q to currentSolution
16:                 remove from currentSolution any rule
                    $W\rightarrow Z$ : $\neg$insolution($W\rightarrow Z$, $\langle\mathcal{C},\mathcal{D},\mathcal{M} \wedge X\rightarrow Y \in$
                    currentSolution $\cup$ {NewLHS$\rightarrow$Q}, $\lambda,k\rangle$)
17:                 tune the settings of the constraints
18:               end if
19:               if proven($\forall x \subseteq$ NewAvailableLHS :
                  $\neg$insolution($x \cup$ NewLHS$\rightarrow$Q, $\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)) then
20:                 NewAvailableRHS := NewAvailableRHS - Q
21:               end if
22:             end if
23:           end if
24:         end for
25:       end if
26:     end if
27:     if NewAvailableLHS $\neq \emptyset$ and NewAvailableRHS $\neq \emptyset$ then
28:       KORD(NewLHS,NewAvailableLHS,NewAvailableRHS,$\langle\mathcal{C},\mathcal{D},\mathcal{M},\lambda,k\rangle$)
29:     end if
30:   end if
31:   SoFar := SoFar $\cup$ {P}
32: end for
```

As KORD traverses the space of rules, the *minLeverage* constraint is increased dynamically so that it is always the k^{th} leverage value of all the rules searched so far. Line 15 records the rules satisfying the current constraints. Whenever adding a rule to $currentSolution$ causes the number of rules in $currentSolution$ to exceed k, line 16 removes the unqualified rule whose leverage value ranks $(k+1)$. When the search is finished, $currentSolution$ is the solution to the KOtask.

Appendix B proves the correctness of the KORD$'$ algorithm, a variant of KORD with all pruning removed. As KORD$'$ is correct, KORD will also be correct so long as all the pruning rules are correct.

The worst case complexity of KORD is exponential on the number of conditions in $\mathcal{C}$ as in the worst case no pruning is performed and the algorithm explores the entire search space. The efficiency of KORD depends critically, then, upon the efficacy of the pruning rules. The pruning rules depend upon, k, the rule value measure λ, and the constraints $\mathcal{M}$. In the following section, we examine a typical class of KO tasks, which requires the specification of a value measure and set of constraints. This enables us to provide examples of some pruning rules that this combination of measure and constraints allow. We prove the correctness of the pruning rules. We then present experimental evidence as to the effectiveness of the KORD algorithm for this example task.

7. An Example KO Task

For the purposes of this example we use leverage as the value measure for the reasons outlined in Section 4.1. We restrict the set of constraints $\mathcal{M}$ to user specified constraints on the coverage, support, confidence and size of the antecedent of a rule, together with the constraint that the consequent contain a single condition only. $\mathcal{M}$ can thus be described by

maxLHSsize denoting the maximum number of conditions allowed on the antecedent of rule

minCoverage denoting the minimum coverage,

minSupport denoting the minimum support, and

minConfidence denoting the minimum confidence.

In consequence, for this example, $CSsolution : \{\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle\} \rightarrow \{\{X \rightarrow Y\}\}$ is a many-to-one function mapping a $CStask$ to its solution, satisfying

$$\text{CSsolution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle)$$

$$\begin{aligned} = \{X{\rightarrow}Y \mid {} & X \subseteq \mathcal{C} \\ & \wedge Y \in \mathcal{C} \\ & \wedge 1 \leqslant |X| \leqslant \mathit{maxLHSsize} \\ & \wedge \text{coverage}(X{\rightarrow}Y) \geqslant \mathit{minCoverage} \\ & \wedge \text{support}(X{\rightarrow}Y) \geqslant \mathit{minSupport} \\ & \wedge \text{confidence}(X{\rightarrow}Y) \geqslant \mathit{minConfidence}\}. \end{aligned}$$

solution : $\{\langle\mathcal{C},\mathcal{D},\mathcal{M},\text{leverage},k\rangle\} \rightarrow \{\{X{\rightarrow}Y\}\}$ is a many-to-many function mapping a KOtask to its solutions, satisfying

$$\begin{aligned} & \forall s \in \text{solution}(\langle\mathcal{C},\mathcal{D},\mathcal{M},\text{leverage},k\rangle) \\ & \quad s \subseteq \text{CSsolution}(\langle\mathcal{C},\mathcal{D},\mathcal{M}\rangle) \\ & \qquad \wedge |s| \leq k; \\ & \qquad \wedge \neg\exists r \in \text{solution}(\langle\mathcal{C},\mathcal{D},\mathcal{M},\text{leverage},k\rangle) : |r| < |s| \\ & \qquad \wedge \neg\exists X{\rightarrow}Y \in s, X'{\rightarrow}Y' \in \text{CSsolution}(\langle\mathcal{C},\mathcal{D},\mathcal{M}\rangle) - s : \\ & \qquad\qquad \text{leverage}(X{\rightarrow}Y,\mathcal{D}) < \text{leverage}(X'{\rightarrow}Y',\mathcal{D}). \end{aligned}$$

In the following sections we provide an example walk through of the KORD algorithm for a simple example of such a task and present pruning rules that support tasks of this type.

8. An example walk-through

We illustrate the algorithm with a simple *KOtask* of the type defined in Section 7, where the conditions $\mathcal{C} = \{a, b, c\}$ and the dataset $\mathcal{D} = \{\{a, b\}, \{a, b\}, \{a, c\}, \{c\}\}$. There are no constraints $\mathcal{M}$, λ = leverage and $k = 2$. We assume no pruning techniques are available. In consequence, the predicate proven always returns false.

Top level call:
$CurrentLHS{=}\{\}, AvailableLHS{=}\{a, b, c\}, AvailableRHS{=}\{a, b, c\}$
Line 1: $SoFar{=}\{\}$
Loop at line 2:

- $P{=}a$

  $NewLHS{=}\{a\}$

  $NewAvailableLHS{=}\{\}$

  $NewAvailableRHS{=}\{b, c\}$

Loop at line 9:

- $Q{=}b$

  $CurrentSolution$ does not yet contain k rules so any rule that satisfies the constraints M will be added.

  $CurrentSolution{=}\{\{a\}{\rightarrow}\{b\}\lambda{=}0.125\}$

- $Q{=}c$

  $CurrentSolution$ still does not yet contain k rules so the new rule is also added.

  $CurrentSolution{=}\{\{a\}{\rightarrow}\{b\}\ \lambda{=}0.125, \{a\}{\rightarrow}\{c\}\ \lambda{=}{-}0.125\}$

line 31: $SoFar{=}\{a\}$

– $P{=}b$

$NewLHS{=}\{b\}$

$NewAvailableLHS{=}\{a\}$

$NewAvailableRHS{=}\{a, c\}$

Loop at line 9:

- $Q{=}a$

  The value of the new rule $\{b\}{\rightarrow}\{a\}\ \lambda{=}0.125$ is greater than that of the lowest value rule in $CurrentSolution$, so it replaces that rule.

  $CurrentSolution{=}\{\{a\}{\rightarrow}\{b\}\ \lambda{=}0.125, \{b\}{\rightarrow}\{a\}\ \lambda{=}0.125\}$

- $Q{=}c$

  The value of $\{b\}{\rightarrow}\{c\}\ \lambda{=}{-}0.250$ is lower than that of the two rules in $CurrentSolution$, so $CurrentSolution$ is unaltered.

Recursive call at line 28:

$CurrentLHS{=}\{b\}$

$AvailableLHS{=}\{a\}$

$AvailableRHS{=}\{a, c\}$

Line 1: $SoFar = \{\}$

Loop at line 2:

- P=a

  $NewLHS$=$\{a, b\}$

  $NewAvailableLHS$=$\{\}$

  $NewAvailableRHS$=$\{c\}$

  Loop at line 9:

  * Q=c

    The value of $\{a, b\}\rightarrow\{c\}$ λ=-0.250 is lower than that of the two rules in *CurrentSolution*, so *CurrentSolution* is unaltered.

  line 31: $SoFar$=$\{a\}$

line 31: $SoFar$=$\{a, b\}$

– P=c

  $NewLHS$=$\{c\}$

  $NewAvailableLHS$=$\{a, b\}$

  $NewAvailableRHS$=$\{a, b\}$

  Loop at line 9:

  - Q=a

    The value of $\{c\}\rightarrow\{a\}$ λ=-0.125 is lower than that of the two rules in *CurrentSolution*, so *CurrentSolution* is unaltered.

  - Q=b

    The value of $\{c\}\rightarrow\{b\}$ λ=-0.250 is lower than that of the two rules in *CurrentSolution*, so *CurrentSolution* is unaltered.

  Recursive call at line 28:

    $CurrentLHS$=$\{c\}$

    $AvailableLHS$=$\{a, b\}$

    $AvailableRHS$=$\{a, b\}$

    Line 1: $SoFar = \{\}$

    Loop at line 2:

- $P{=}a$
  $NewLHS{=}\{a,c\}$
  $NewAvailableLHS{=}\{\}$
  $NewAvailableRHS{=}\{b\}$
  Loop at line 9:
  - $Q{=}b$
    The value of $\{a,c\}{\rightarrow}\{b\}\,\lambda{=}{-}0.125$ is lower than that of the two rules in $CurrentSolution$, so $CurrentSolution$ is unaltered.

  line 31: $SoFar{=}\{a\}$
- $P{=}b$
  $NewLHS{=}\{b,c\}$
  $NewAvailableLHS{=}\{a\}$
  $NewAvailableRHS{=}\{a\}$
  Loop at line 9:
  - $Q{=}a$
    The value of $\{b,c\}{\rightarrow}\{a\}\,\lambda{=}0.000$ is lower than that of the two rules in $CurrentSolution$, so $CurrentSolution$ is unaltered.

  line 31: $SoFar{=}\{a,b\}$
  Recursive call at line 28:

    $CurrentLHS{=}\{b,c\}$
    $AvailableLHS{=}\{a\}$
    $AvailableRHS{=}\{a\}$
    Line 1: $SoFar = \{\}$
    Loop at line 2:
    - $P{=}a$
      $NewLHS{=}\{a,b,c\}$
      $NewAvailableLHS{=}\{\}$
      $NewAvailableRHS{=}\{\}$
      Loop at line 9 iterates over an empty set and so generates no new rules.
      line 31: $SoFar{=}\{a\}$

line 31: $SoFar=\{a, b, c\}$

9. Pruning and related techniques

The example walk-through of the algorithm provides a good illustration of the potential for pruning. The final solution was found after considering only the first three rules. If the system could have determined at this point that the remaining search space could not contain any rules that could be added to the solution, then the search could have been terminated early. A number of pruning rules for the class of KOtasks defined in Section 7 are presented in Appendix C, together with proofs of their correctness.

To illustrate how these rules are applied, we consider the application of pruning rule 7 at line 10 of the KORD algorithm. This pruning rule calculates an upper bound on the possible value of leverage for any rule with Q as the consequent and with the current antecedent $NewLHS$ or a specialization thereof that can be reached by adding to $NewLHS$ conditions in $NewAvailableLHS$. A loose upper bound on leverage is provided by calculating the leverage of a hypothetical rule that has both coverage and support equal to the support of $NewLHS \rightarrow Q$. When applied in the above example when $NewLHS = \{c\}$ and $Q = a$, the resulting optimistic value is 0.031. This is less than the lowest value in the solution so far, indicating that no specialization of $\{c\} \rightarrow \{a\}$ can be in the solution. As a result, a can be immediately deleted from $NewAvailableRHS$ (line 11). The same applies when $Q = b$. As a result, when line 27 is executed $NewAvailableRHS$ is empty and the recursive call on line 28 is not made, saving exploration of the rules $\{a, c\} \rightarrow \{b\}$ and $\{b, c\} \rightarrow \{a\}$.

9.1. Reducing data accesses

A major computational overhead in rule discovery is accessing the dataset to evaluate the required statistics with respect to each antecedent, consequent, and antecedent-consequent pair. One of the key factors in the success of the Apriori algorithm is its success in minimizing the number of such data accesses that are required. In addition to pruning regions of the search space, another important technique used in KORD to reduce compute time is to save data access. Data access is required to evaluate the cover and support of a rule or a set of conditions. However, data access can be avoided when these values can be derived from other evidence or it can be determined that the values will not satisfy the search constraints. Different saving rules can be adopted at different stages during the search process. Whereas the pruning rules save data access by discarding the region of the search space below a node, the saving rules save data access for a node without removing its branch.

In order to evaluate the number of records covered by set of conditions, the dataset is normally accessed by KORD at least once for each antecedent ($NewLHS$) and once for the union of the antecedent and consequent. Techniques for saving such data accesses are directed at avoiding the need to perform one or the other of these computations. Two such techniques are presented in Appendix D.

9.2. Dynamic constraint update

Although the constraints *minCoverage*, *minSupport*, and *minConfidence* are initialized by the user, during search it may be possible to derive tighter constraints on these statistics than those initial values. Such tighter constraints can be used to prune more of the search space or save the evaluation of rules. Based on Theorems 10, 11 and 12 presented in Appendix D, whenever a new rule is added to $currentSolution$ at line 15 of the algorithm, all the constraints can be updated at line 17 according to the rules listed below.

1. If $minSupport < minCoverage \times minConfidence$, then $minSupport = minCoverage \times minConfidence$.

2. If $minCoverage < minSupport$, then $minCoverage = minSupport$.

3. If $minSupport < minLeverage$, then $minSupport = minLeverage$.

10. Proof-of-concept experiments

We investigate the computational efficiency of KORD search for rules that optimize leverage. Experiments are performed on ten large datasets, nine from the UCI ML and KDD repositories (Blake and Merz, 2001; Bay, 2001) and one market-basket dataset used in previous association rule discovery research (Kohavi et al., 2000; Zheng et al., 2001). These datasets are listed in Table II. In our experiments for all the datasets, all the conditions available are allowed both in the antecedent and consequent of rules. Numeric attributes were discretized into three sub-ranges, each containing as close as possible to one third of the records.

It might be thought that traditional association rule techniques could be applied to this problem by first finding all frequent itemsets

Table II. Datasets for experiments

name	**records**	**attributes**	**values**
BMS-WebView-1	59,602	497	497
connect-4	67,557	43	129
covtype	581,012	55	125
ipums.la.99	88,443	61	1,883
kddcup98	52,256	480	4,244
mush	8,124	23	127
pendigits	10,992	17	58
shuttle	58,000	10	34
splice junction	3,177	61	243
ticdata2000	5,822	86	709

and then generating all rules from those itemsets, sorting them on leverage, and discarding all but the top n. We evaluate the feasibility of this approach by applying Borgelt's (2000) efficient implementation of Apriori. We do not include recent alternatives to Apriori in this study as it has been argued elsewhere (Zheng et al., 2001) that they suffer the same performance degradation as Apriori on tasks for which minimum support is set to a level that results in excessively large numbers of frequent itemsets, as is the case for the current task with respect to many real world datasets.

A difficulty with applying the frequent itemset approach to discovering the k rules with highest leverage is that there does not appear to be any way to determine apriori what minimum support level to employ. How does one ensure that the minimum support constraint applied does not prevent the generation of one of the k rules with the highest leverage? However, it follows from the definitions of leverage and support that leverage$(X \rightarrow Y) \leq$ support$(X \rightarrow Y)$. Hence, if rules R are derived from the set of itemsets $\{i \mid \text{support}(i) \geq kth\}$, where *kth* is the k^{th} highest leverage a rule in R, then it follows that the k rules with highest leverage derived from those itemsets are the k rules with highest leverage that would be derived without a minimum support constraint. Using this insight, an iterative process could repeatedly generate frequent itemsets at gradually lowering values of *minSupport* and generate rules R therefrom until leverage$(R_k) \geq$ *minSupport*, where leverage(R_k) is the k^{th} highest value of leverage of a rule in R. To give an indication of a lower bound on the overheads of such an approach, we use Apriori with *minSupport* set to the minimum value for leverage of the 1000 rules with highest leverage as discovered by KORD.

It might be thought that the earlier OPUS (Webb, 1995), Max-Miner (Bayardo, 1998) and Dense-Miner (Bayardo et al., 2000) algorithms, upon which KORD builds, should also be comparators against which KORD is evaluated. However, this is not feasible, as those algorithms require that the consequent be restricted to a single pre-specified value, and hence are not capable of performing the k-optimal rule discovery task. It is, indeed, this very limitation that KORD has been designed to overcome. It might be thought that standard OPUS search could simply be repeated once for each possible consequent. Indeed, our first attempt to tackle the k-optimal rule discovery task pursued exactly this strategy. This proves extremely inefficient, however, as much time can be spent finding the k-optimal rules with respect to a consequent for which the most interesting rule does not appear in the solution to the *KOtask*.

In all the experiments, KORD seeks the top 1000 rules on leverage within the constraints of the maximum number of conditions in the antecedent of a rule is 4 and the maximum number of conditions in the consequent of a rule is 1. The same maximum antecedent and consequent size used for KORD were also used for Apriori. The experiments were performed on a Linux server with dual 933MHz CPUs, 1.5Gb RAM, and 4Gb virtual memory.

Table III shows the efficiency of KORD and Apriori on the ten large datasets. For KORD this table presents the compute time in hours, minutes, and seconds; the number of rules evaluated; and the minimum leverage for a rule in the top 1000 rules on leverage. For Apriori this table lists the compute time for frequent itemset generation in hours, minutes and seconds, and the number of itemsets generated.

Comparing CPU times we can see that on every dataset other than BMS-WebView-1, for which compute times are extremely small, KORD requires less compute time than Apriori. For kddcup98 Apriori runs out of memory when generating itemsets. This supports previous analyses of the inefficiency of the frequent itemset strategy for dense datasets (Bayardo, 1998). It also suggests that KORD provides a more widely applicable approach to the k-optimal rule discovery problem than Apriori when the user seeks to discover a limited number of rules that optimize a value measure.

11. Computational complexity and scalability

The worst-case complexity of KORD is exponential on the number of conditions that are available for inclusion in the antecedent and consequent, as in the worst case no pruning is possible and the entire

Table III. Efficiency of KORD and Apriori on real world datasets

	KORD			**Apriori**	
	time	**rules**	**min.**	**time**	**itemsets**
Datasets	**h:m:s**	**evaluated**	**leverage**	**h:m:s**	**generated**
BMS-WebView-1	0:0:4	91,244	0.0019	0:0:1	150,136
connect-4	0:0:55	94,682	0.1279	1:0:2	2,360,136
covtype	0:8:3	155,489	0.2212	30:2:55	6,070,334
ipums.la.99	0:0:29	104,582	0.2424	1:30:6	2,649,154
kddcup98	1:31:30	6,259,666	0.2431	Not enough memory	
mush	0:0:1	12,892	0.1558	0:0:4	47,867
pendigits	0:0:1	50,733	0.0698	0:0:3	25,108
shuttle	0:0:1	10,029	0.0504	0:0:4	6,451
splice junction	0:0:6	2,160,784	0.0454	0:0:38	2,084,694
ticdata2000	0:0:33	187,247	0.1899	0:38:47	13,216,656

rule space must be explored. If there are k conditions that may appear in an antecedent then the space of possible antecedents is 2^k. If any one of these k conditions may also appear in the consequent, then the space of possible rules is of order $O(2^{k+1})$, despite a restriction that a condition may not appear in both the antecedent and conclusion of the same rule. The efficiency of the algorithm depends critically, then, upon the effectiveness of the pruning mechanisms for a given task. As the experiments above have demonstrated, for a number of larger real-world datasets, efficient search by leverage is a reality.

This raises the question, however, of how well will the algorithm scale to even larger datasets. A fundamental limitation of the current algorithm is that it requires that all data be retained in memory during search, as it requires frequent assessment of relevant statistics. With current hardware, this restricts application of the algorithm to datasets of size measured in gigabytes. It is conceivable, however, that a variant of the algorithm could be created that uses the strategy developed for Dense-Miner (Bayardo et al., 2000), performing breadth-first search with two stages at each level, the first generating candidate rules and the second evaluating them all in a single pass through the data.

In theory, the computation required by KORD should increase linearly with respect to the size of the data, as the size of the data affect directly only the process of counting the statistics used to evaluate a rule. To assess this, we took the largest of our datasets, covtype, and formed sample datasets containing the first 25%, 50%, 75% and 100% of the data. We also formed larger datasets by appending each of

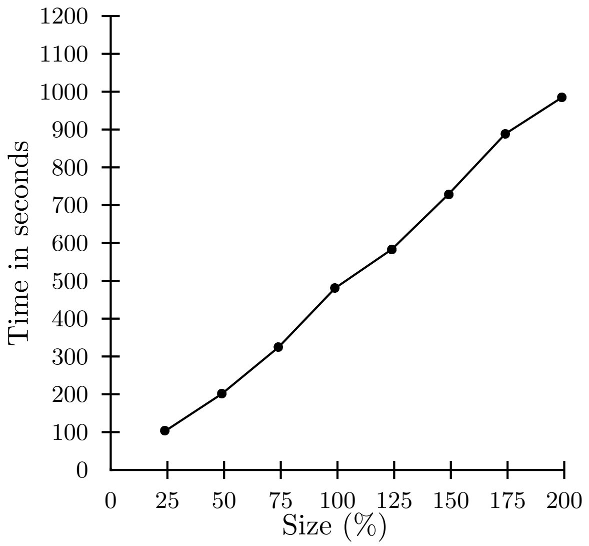

Figure 4. Compute time vs data set size for variants of the covtype dataset

these sample datasets to the original, resulting in four further datasets containing 125%, 150%, 175% and 200% of the original amount of data, albeit with some records duplicated. Figure 4 plots the size of these datasets against the compute time required to perform the task described in Section 10.

Inspection of the resulting graph reveals that the points 25%, 50%, 75% and 100% appear to form a curve that it slightly super-linear. However, it is clear that this effect is not due solely to the direct increased costs of processing the additional data, as the processing time for 200%, in which all data points are duplicated, is almost exactly twice that of 100%. Rather, due to the characteristics of this data, as the number of data points increases up to 100%, there are more combinations of conditions that achieve sufficient support to make them credible candidates during search. Hence the number of rules explored increases. Figure 5 plots the number of rules explored at each data set size. As can be seen, there is an increase in the number of rules up until the stage when 100% of the data is used. When data elements are simply duplicated, however, no more rules need to be considered. In this case, the increase in execution time reflects only an increase in the time taken to evaluate the statistics for each rule that is considered.

To assess whether the number of rules evaluated can always be expected to increase as the number of unique data points increases, we

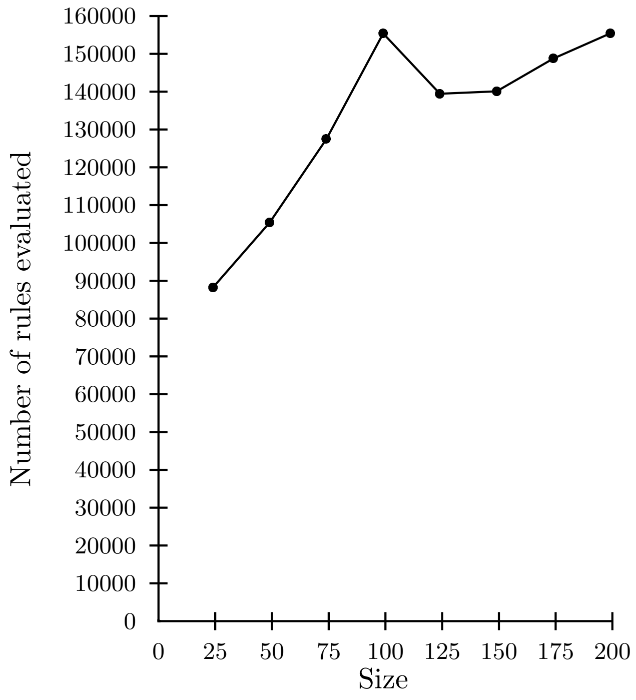

Figure 5. Number of rules evaluated vs data set size for variants of the covtype dataset

repeated the experiment with the next largest data set, ipums.la.99. As the effect of duplicating data elements does not appear to need further investigation, we used only data sets containing 25%, 50%, 75% and 100% of the data. The compute time and number of rules evaluated are plotted in Figures 6 and 7. It is evident that there is no simple relationship between data set size and the number of rules that need to be explored. As a result, the amount of time taken varies at different rates over different increases in data size. Processing all the data takes 29 seconds which is less than 4 times the 8 seconds taken to process 25% of the data. We can expect the increase on average to be linear on the number of records, but relationships within the data may mean that additional records result in the need to explore either more or fewer potential rules, resulting in concomitant increases or decreases in computation.

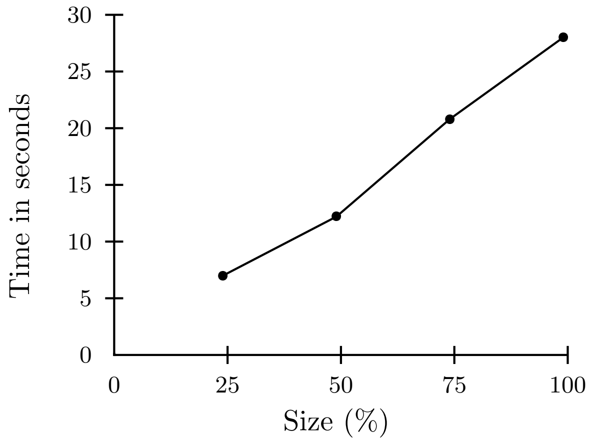

Figure 6. Compute time vs data set size for variants of the ipums.la.99 dataset

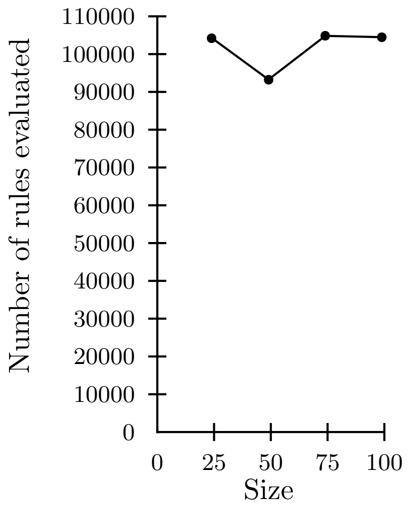

Figure 7. Number of rules evaluated vs data set size for variants of the ipums.la.99 dataset

12. Conclusions

Traditional machine learning seeks a single model that describes the data. In contrast, exploratory rule discovery finds multiple potentially interesting models, empowering the user to select between those models using criteria that may be difficult to articulate in a form that could be utilized by a knowledge discovery system.

Association rule discovery is the most widely used exploratory rule discovery technique. However, its support-confidence framework limits its applicability. In particular, where minimum support is not a relevant criterion for selecting rules, the frequent itemset strategies that have been developed for association rule discovery risk failing to identify interesting rules.

We have presented a new framework for rule discovery that we call *k-optimal rule discovery*, together with a scalable algorithm for tackling k-optimal rule discovery tasks. The computational complexity of the algorithm is linear with respect to data set size, although different sets of data imply different search spaces which may require greater or lesser computation to explore. We have demonstrated that KORD can efficiently solve an interesting class of k-optimal rule discovery task, finding the 1000 rules with the highest leverage. We have further demonstrated that it is difficult to tackle this task using the frequent itemset framework pioneered within the association rule discovery paradigm.

An important benefit of the k-optimal rule discovery framework is the ability to impose and gain computational benefit from contraints on the maximum number of rules to discover. A common complaint levelled against association rule discovery is that the user cannot control how many rules are returned. The number of rules returned is controlled indirectly by the minimum support constraint. Small changes in this constraint can transform a problem for which very few rules are discovered to one for which millions of rules are discovered. In the k-optimal rules discovery framework, the user places an upper limit k on the number of rules to discover. This constraint ensures that the user is only presented with the most interesting rules according to the specified measure of rule value. It also speeds up the rule discovery task by allowing pruning of areas of the search space that cannot contain one of the k most valuable rules.

This work demonstrates the efficiency of the KORD algorithm for one type of useful rule discovery task. It falls to future research to identify further sets of interesting constraints for k-optimal rule discovery and develop pruning rules that will facilitate their efficient solution.

Acknowledgements

This research has been supported by the Australian Research Council under grant DP0450219. We are grateful to Mike Pazzani for suggesting the term *k-optimal rule discovery.*

Appendix

A. Proof of sufficiency and necessity of update strategy

The following theorems prove that the update strategy for *currentSolution* is sufficient and necessary to ensure that on termination $currentSolution = \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$.

Theorem 1 (Sufficiency of currentSolution). *Suppose KOtask* $= \langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any* $S \subseteq \{X \to Y \mid X \subseteq \mathcal{C} \wedge Y \subseteq \mathcal{C}\}$, *let* $currentSolution = \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S, \lambda, k\rangle)$. *For any* $s \in \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap S$, $s \in currentSolution$ *holds.*

This follows directly from (3).

Theorem 2 (Necessity of currentSolution). *Suppose KOtask* $= \langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any* $S \subseteq \{X \to Y \mid X \subseteq \mathcal{C} \wedge Y \subseteq \mathcal{C}\}$, *let* $currentSolution = \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S, \lambda, k\rangle)$. *For any* $s \in currentSolution$, $s \in \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap S$ *holds.*

Proof. Assume there exists a rule s_1 satisfies $s_1 \in currentSolution$ but $s_1 \notin \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap S$. So $s_1 \notin \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$ or $s_1 \notin S$ holds. If $s_1 \notin \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$, then $s_1 \notin \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S, \lambda, k\rangle)$, contradicting $s_1 \in currentSolution$; if $s_1 \notin S$, obviously $s_1 \notin \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S, \lambda, k\rangle)$, contradicting $s_1 \in currentSolution$. Therefore s_1 does not exist. □

Due to the following theorem, it is not necessary to keep track of S. Rather, it is sufficient to update *currentSolution* as each rule is added to S.

Theorem 3 (Sufficiency of addition to currentSolution). *Suppose KOtask* $= \langle\mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any* $S \subseteq \{X \to Y \mid X \subseteq \mathcal{C} \wedge Y \subseteq \mathcal{C}\}$ *and rule* $r \notin S$, *let* $currentSolution = \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S, \lambda, k\rangle)$ *and* $currentSolution' = \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S \cup \{r\}, \lambda, k\rangle)$. *The following holds.*

$$\begin{aligned}&\text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in currentSolution \cup \{r\}, \lambda, k\rangle)\\ &\quad \subseteq currentSolution'\end{aligned} \tag{4}$$

Proof. Assume there exists a rule s_1 that satisfies

$$s_1 \in \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in currentSolution \cup \{r\}, \lambda, k\rangle) \tag{5}$$

but $s_1 \notin currentSolution'$ which is

$$s_1 \notin \text{solution}(\langle\mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \to Y \in S \cup \{r\}, \lambda, k\rangle) \tag{6}$$

According to the theorem of sufficiency of *currentSolution*, the following holds.

$$\begin{aligned}&\text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (S \cup \{r\}) \\ &\quad \subseteq \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in S \cup \{r\}, \lambda, k\rangle)\end{aligned} \tag{7}$$

From (6) and (7), we obtain

$$s_1 \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (S \cup \{r\}) \tag{8}$$

From (5) we obtain $s_1 \in \mathit{currentSolution} \cup \{r\}$. Let us first assume $s_1 \in \mathit{currentSolution}$, which is $s_1 \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in S, \lambda, k\rangle)$. According to the theorem of necessity of *currentSolution*, $s_1 \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap S$ holds. Thus $s_1 \in S$. So $s_1 \in S \cup \{r\}$. So $s_1 \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (S \cup \{r\})$ holds, contradicting (8).

Now let us assume $s_1 \in \{r\}$. This means $s_1 \in S \cup \{r\}$. From (5) and according to the theorem of necessity of *currentSolution*, $s_1 \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (\mathit{currentSolution} \cup \{r\})$ holds. Thus $s_1 \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$. From $s_1 \in S \cup \{r\}$, we obtain $s_1 \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (S \cup \{r\})$ holds, contradicting (8). Thus s_1 does not exist. It follows that (4) holds. □

Actually, the necessity of addition to *currentSolution* also holds, as shown in the following theorem.

Theorem 4 (Necessity of addition to currentSolution). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any* $S \subseteq \{X{\rightarrow}Y \mid X \subseteq \mathcal{C} \wedge Y \subseteq \mathcal{C}\}$ *and rule* $r \notin S$, *let currentSolution* $= \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in S, \lambda, k\rangle)$ *and currentSolution*$'$ $= \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in S \cup \{r\}, \lambda, k\rangle)$. *The following holds.*

$$\begin{aligned}&\mathit{currentSolution}' \\ &\quad \subseteq \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in \mathit{currentSolution} \cup \{r\}, \lambda, k\rangle)\end{aligned} \tag{9}$$

Proof. For any $s \in \mathit{currentSolution}'$, which means $s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in S \cup \{r\}, \lambda, k\rangle)$, according to the theorem of necessity of *currentSolution*, $s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (S \cup \{r\})$ holds. Thus

$$s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \tag{10}$$

and $s \in S \cup \{r\}$ holds. Let us first assume $s \in S$. From (10), $s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap S$. According to the theorem of sufficiency of *currentSolution*, $s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X{\rightarrow}Y \in S, \lambda, k\rangle) = \mathit{currentSolution}$. Thus $s \in \mathit{currentSolution} \cup \{r\}$. Now let us assume $s \in \{r\}$. Obviously $s \in \mathit{currentSolution} \cup \{r\}$ holds. Therefore we obtain

$$s \in \mathit{currentSolution} \cup \{r\} \tag{11}$$

Table IV. The k-optimal rule discovery algorithm without pruning

```
Algorithm: KORD′(CurrentLHS,AvailableLHS,AvailableRHS)

SoFar := $\emptyset$
for all P in AvailableLHS do
  NewLHS := CurrentLHS $\cup$ {P}
  NewAvailableLHS := SoFar
  NewAvailableRHS := AvailableRHS - P
  for all Q in NewAvailableRHS do
    if insolution(NewLHS$\rightarrow$Q, $\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \rightarrow Y \in$
    currentSolution $\cup$ {NewLHS$\rightarrow$Q}, $\lambda, k\rangle$) then
      add NewLHS $\rightarrow$ Q to currentSolution
      remove from currentSolution any rule
      $W \rightarrow Z$ : $\neg$insolution($W \rightarrow Z, \langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \rightarrow Y \in$
      currentSolution $\cup$ {NewLHS$\rightarrow$Q}, $\lambda, k\rangle$)
    end if
  end for
  if NewAvailableLHS $\neq \emptyset$ and NewAvailableRHS $\neq \emptyset$ then
    KORD′(NewLHS,NewAvailableLHS,NewAvailableRHS)
  end if
  SoFar := SoFar $\cup$ {P}
end for
```

From (10) and (11), $s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle) \cap (\mathit{currentSolution} \cup \{r\})$. According to the theorem of sufficiency of *currentSolution*, we obtain $s \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \rightarrow Y \in \mathit{currentSolution} \cup \{r\}, \lambda, k\rangle)$. Thus (9) holds. □

B. Correctness of the KORD algorithm without pruning

By removing all the pruning in KORD, we get the KORD′ algorithm shown in Table IV. The correctness of KORD′ is proven by its uniqueness and completeness. We first prove the completeness of KORD′ in the following theorem.

Theorem 5 (Completeness). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any CurrentLHS, AvailableLHS* $\subseteq \mathcal{C}$ *and AvailableRHS* $\subseteq \mathcal{C}$ *satisfying AvailableLHS* $\neq \emptyset$, *AvailableRHS* $\neq \emptyset$, *CurrentLHS* $\cap$ *AvailableLHS* $= \emptyset$ *and CurrentLHS* $\cap$ *AvailableRHS* $= \emptyset$, *let AvailableLHS* $= \{P_1, \cdots, P_n\}$ *where* $n \geqslant 1$ *and* $P_1, \cdots, P_n \in \mathcal{C}$, *let* $SEARCHED(CurrentLHS, \{P_1, \cdots, P_m\}, AvailableRHS)$ *denote* *the set of rules KORD′(CurrentLHS, AvailableLHS, AvailableRHS) has searched after the loop from P_1 to P_m at line 2 where $1 \leqslant m \leqslant n$, the following holds.*

$$\begin{aligned} &SEARCHED(CurrentLHS, \{P_1, \cdots, P_m\}, AvailableRHS) \\ &\quad = \{CurrentLHS \cup L \rightarrow \{Q\} \mid L \neq \emptyset \\ &\qquad \wedge L \subseteq \{P_1, \cdots, P_m\} \\ &\qquad \wedge Q \in AvailableRHS \\ &\qquad \wedge Q \notin L\} \end{aligned} \tag{12}$$

Proof. The proof is by induction on m. For the basis, it is readily seen that the theorem is true when $m = 1$, that is,

$$\begin{aligned} &SEARCHED(CurrentLHS, \{P_1\}, AvailableRHS) \\ &\quad = \{CurrentLHS \cup \{P_1\} \rightarrow \{Q\} \mid Q \in AvailableRHS \wedge Q \neq P_1\} \end{aligned} \tag{13}$$

For the inductive hypothesis, assume that the theorem is true for $1 \leqslant m \leqslant k$. We will prove that the theorem is true for $m = k + 1$. When KORD′ loops at line 2 until P_{k+1},

$$SEARCHED(CurrentLHS, \{P_1, \cdots, P_k, P_{k+1}\}, AvailableRHS)$$

is composed of two parts, one is

$$SEARCHED(CurrentLHS, \{P_1, \cdots, P_k\}, AvailableRHS)$$

, and the other is the set of rules KORD′ searches from lines 3 to 15 for P_{k+1}. For P_{k+1}, from lines 3 to 11, KORD′ searches

$$\{CurrentLHS \cup \{P_{k+1}\} \rightarrow \{Q\} | Q \in AvailableRHS \wedge Q \neq P_{k+1}\}$$

and at line 13,

$$KORD'(CurrentLHS \cup \{P_{k+1}\}, \{P_1, \cdots, P_k\}, AvailableRHS)$$

is called since at this point

$$SoFar = \{P_1, \cdots, P_k\}.$$

Applying the inductive hypothesis, we obtain:

$$\begin{aligned} &SEARCHED(CurrentLHS, \{P_1, \cdots, P_k, P_{k+1}\}, AvailableRHS) \\ &\quad = SEARCHED(CurrentLHS, \{P_1, \cdots, P_k\}, AvailableRHS) \\ &\qquad \cup \{CurrentLHS \cup \{P_{k+1}\} \rightarrow \{Q\} \mid Q \in AvailableRHS \\ &\qquad\qquad \wedge\ Q \neq P_{k+1}\} \end{aligned}$$

$$
\begin{aligned}
&\cup\, SEARCHED(\,CurrentLHS \cup \{P_{k+1}\},\\
&\qquad\qquad \{P_1, \cdots, P_k\},\\
&\qquad\qquad AvailableRHS)\\
&= SEARCHED(CurrentLHS, \{P_1, \cdots, P_k\}, AvailableRHS)\\
&\quad \cup \{CurrentLHS \cup \{P_{k+1}\} \rightarrow \{Q\} \mid Q \in AvailableRHS\\
&\qquad\qquad\qquad\qquad\qquad\qquad \wedge Q \neq P_{k+1}\}\\
&\quad \cup \{\, CurrentLHS \cup \{P_{k+1}\} \cup L_1 \rightarrow \{Q\} \mid\\
&\qquad\quad L_1 \neq \emptyset\\
&\qquad\quad \wedge\ L_1 \subseteq \{P_1, \cdots, P_k\}\\
&\qquad\quad \wedge\ Q \in AvailableRHS\\
&\qquad\quad \wedge\ Q \notin L_1 \cup \{P_{k+1}\}\}\\
&= SEARCHED(CurrentLHS, \{P_1, \cdots, P_k\}, AvailableRHS)\\
&\quad \cup \{\, CurrentLHS \cup \{P_{k+1}\} \cup L_1 \rightarrow \{Q\} \mid\\
&\qquad\quad L_1 \subseteq \{P_1, \cdots, P_k\}\\
&\qquad\quad \wedge\ Q \in AvailableRHS\\
&\qquad\quad \wedge\ Q \notin L_1 \cup \{P_{k+1}\}\}\\
&= \{CurrentLHS \cup L \rightarrow \{Q\} \mid L \neq \emptyset\\
&\qquad\qquad\qquad\qquad\qquad \wedge L \subseteq \{P_1, \cdots, P_k\}\\
&\qquad\qquad\qquad\qquad\qquad \wedge Q \in AvailableRHS\\
&\qquad\qquad\qquad\qquad\qquad \wedge Q \notin L\}\\
&\quad \cup \{\, CurrentLHS \cup \{P_{k+1}\} \cup L_1 \rightarrow \{Q\} \mid\\
&\qquad\quad L_1 \subseteq \{P_1, \cdots, P_k\}\\
&\qquad\quad \wedge\ Q \in AvailableRHS\\
&\qquad\quad \wedge\ Q \notin L_1 \cup \{P_{k+1}\}\}\\
&= \{CurrentLHS \cup L \rightarrow \{Q\} \mid L \neq \emptyset\\
&\qquad\qquad\qquad\qquad\qquad \wedge L \subseteq \{P_1, \cdots, P_k, P_{k+1}\}\\
&\qquad\qquad\qquad\qquad\qquad \wedge Q \in AvailableRHS\\
&\qquad\qquad\qquad\qquad\qquad \wedge Q \notin L\}
\end{aligned}
$$

proving the theorem. □

For any $KOtask = \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$, according to the theorem of sufficiency of addition to *currentSolution*, the variable *currentSolution* at line 8 in KORD′ always records solution($\langle \mathcal{C}, \mathcal{D}, \mathcal{M} \wedge X \rightarrow Y \in S \cup \{r\}, \lambda, k \rangle$) where S is the search space considered so far and r is the current rule being considered. According to its completeness, when it finishes, KORD′ has searched all the rules, thus what the variable *currentSolution* records now is solution($\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$). This proves the uniqueness of KORD′.

C. Pruning rules

C.1. Properties of the rules in the solution of KOtask

To facilitate the analysis of the pruning rules, we analyze some necessary properties of KOtasks and their solutions. Before presenting the theorems, we give the following lemma.

Observation 1 (Subset cover). *Given KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, for any* $S_1, S_2 \subseteq \mathcal{C}$*, if* $S_1 \subseteq S_2$*, then* $\mathrm{coverset}(S_2) \subseteq \mathrm{coverset}(S_1)$*, and hence,* $\mathrm{cover}(S_2) \leqslant \mathrm{cover}(S_1)$ *holds.*

Proof. For any $d \in \mathrm{coverset}(S_2)$, from the definition of coverset (1), $S_2 \subseteq d$ holds. Since $S_1 \subseteq S_2$, $S_1 \subseteq d$ holds. Hence $d \in \mathrm{coverset}(S_1)$. So $\mathrm{coverset}(S_2) \subseteq \mathrm{coverset}(S_1)$ holds. □

Observation 2 (Upward closure of coverage). *Given KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, for any rule* $X \rightarrow Y$*, if there exists* $X_1 \subseteq X$ *satisfying* $\mathrm{cover}(X_1) < minCoverage$*,* $X \rightarrow Y \notin \mathrm{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$*.*

Proof. According to the definition of KOtask, $X_1 \subseteq X$ and Observation 1, the following holds.

$$\mathrm{coverage}(X \rightarrow Y) = \mathrm{cover}(X) \leqslant \mathrm{cover}(X_1) < minCoverage$$

Hence $X \rightarrow Y \notin \mathrm{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$. □

Observation 3 (Upward closure of support). *Given KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, for any rule* $X \rightarrow Y$*, if there exists* $Z \subseteq X \cup Y$ *satisfying* $\mathrm{cover}(Z) < minSupport$*,* $X \rightarrow Y \notin \mathrm{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$*.*

Proof. According to the definition of KOtask, $Z \subseteq X \cup Y$ and Observation 1, the following holds.

$$\mathrm{support}(X \rightarrow Y) = \mathrm{cover}(X \cup Y) \leqslant \mathrm{cover}(Z) < minSupport$$

Hence $X \rightarrow Y \notin \mathrm{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$. □

Theorem 6 (Lower bound on leverage). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*. For any rule* $X \rightarrow Y$*, if*

$$\mathrm{cover}(Y) > 1 - \frac{minLeverage}{\mathrm{cover}(X)}$$

or

$$\mathrm{cover}(X) > 1 - \frac{minLeverage}{\mathrm{cover}(Y)}$$

then

$$X \rightarrow Y \notin \mathrm{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle).$$

Proof. Let us first prove the theorem when $\text{cover}(Y) > 1 - \frac{minLeverage}{\text{cover}(X)}$. According to the definition of KOtask, we obtain:

$$\begin{aligned}\text{leverage}(X \rightarrow Y) &= \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y) \\ &= \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y)\end{aligned}$$

From

$$\text{cover}(Y) > 1 - \frac{minLeverage}{\text{cover}(X)}$$

we obtain

$$\text{cover}(X) \times \text{cover}(Y) > \text{cover}(X) - minLeverage.$$

Thus $\text{leverage}(X \rightarrow Y)$ satisfies:

$$\begin{aligned}\text{leverage}(X \rightarrow Y) &< \text{cover}(X \cup Y) - (\text{cover}(X) - minLeverage) \\ &= \text{cover}(X \cup Y) - \text{cover}(X) + minLeverage.\end{aligned}$$

From Observation 1 we obtain $\text{cover}(X \cup Y) \leqslant \text{cover}(X)$. Thus the following holds.

$$\text{leverage}(X \rightarrow Y) < minLeverage$$

Hence $X \rightarrow Y \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$. Similarly the theorem is proved when

$$\text{cover}(X) > 1 - \frac{minLeverage}{\text{cover}(Y)}.$$

$\square$

Theorem 7 (Auxiliary lower bound on leverage). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$. *For any rule* $X \rightarrow Y$, *if* $\text{cover}(X) \times (1 - \text{cover}(X)) < minLeverage$, $X \rightarrow Y \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$.

Proof. Firstly let us assume

$$\text{cover}(X) \leqslant \text{cover}(Y).$$

It follows that

$$1 - \text{cover}(X) \geqslant 1 - \text{cover}(Y).$$

According to the definition of KOtask and Observation 1, we obtain:

$$\begin{aligned}\text{leverage}(X \rightarrow Y) &= \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y) \\ &= \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y) \\ &\leqslant \text{cover}(X) - \text{cover}(X) \times \text{cover}(Y) \\ &= \text{cover}(X) \times (1 - \text{cover}(Y)) \\ &\leqslant \text{cover}(X) \times (1 - \text{cover}(X)) \\ &< minLeverage\end{aligned}$$

Secondly let us assume $cover(X) > \text{cover}(Y)$. Thus $\frac{\text{cover}(Y)}{\text{cover}(X)} < 1$ holds. According to the definition of KOtask and Observation 1, we obtain:

$$\begin{aligned}
\text{leverage}(X \to Y) &= \text{support}(X \to Y) - \text{cover}(X) \times \text{cover}(Y) \\
&= \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y) \\
&\leqslant \text{cover}(Y) - \text{cover}(X) \times \text{cover}(Y) \\
&= \text{cover}(Y) \times (1 - \text{cover}(X)) \\
&< \text{cover}(Y) \times \frac{minLeverage}{\text{cover}(X)} \\
&< minLeverage
\end{aligned}$$

So for both cases $leverage(X \to Y) < minLeverage$ holds. Hence $X \to Y \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$. □

Theorem 8 (Lower bound on confidence). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$. *For any rule* $X \to Y$, *if* $\frac{\text{cover}(Y)}{\text{cover}(X)} < minConfidence$, $X \to Y \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$.

Proof. According to the definition of KOtask and Observation 1, we obtain:

$$\text{confidence}(X \to Y) = \frac{\text{support}(X \to Y)}{\text{coverage}(X \to Y)} = \frac{\text{cover}(X \cup Y)}{\text{cover}(X)} \leqslant \frac{\text{cover}(Y)}{\text{cover}(X)}$$

From

$$\frac{\text{cover}(Y)}{\text{cover}(X)} < minConfidence$$

it follows that

$$\text{confidence}(X \to Y) < minConfidence.$$

Hence, $X \to Y \notin \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$. □

Theorem 9 (Full cover). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$. *For any rule* $X \to Y$, *if* $\text{cover}(X) = 1$ *or* $\text{cover}(Y) = 1$, $\text{leverage}(X \to Y) = 0$ *holds.*

Proof. When $\text{cover}(X) = 1$, $\text{cover}(X \cup Y) = \text{cover}(Y)$ holds. Thus,

$$\begin{aligned}
\text{leverage}(X \to Y) &= \text{support}(X \to Y) - \text{cover}(X) \times \text{cover}(Y) \\
&= \text{cover}(Y) - \text{cover}(Y) \\
&= 0
\end{aligned}$$

Similarly the theorem is proved when $\text{cover}(Y) = 1$. □

Theorem 10 (Coverage for rules in the solution). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any rule* $X \to Y \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$, $\text{coverage}(X \to Y) \geqslant \textit{minSupport}$ *holds.*

Proof. According to the definition of KOtask, Observation 1 and that $X \to Y$ is in the solution, we obtain:

$$\begin{aligned}\text{coverage}(X \to Y) &= \text{cover}(X) \geqslant \text{cover}(X \cup Y)\\ &= \text{support}(X \to Y)\\ &\geqslant \textit{minSupport}\end{aligned}$$

□

Theorem 11 (Support for rules in the solution). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any rule* $X \to Y \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$, $\text{support}(X \to Y) \geqslant \textit{minCoverage} \times \textit{minConfidence}$ *holds.*

Proof. According to the definition of KOtask and that $X \to Y$ is in the solution, we obtain:

$$\begin{aligned}\text{support}(X \to Y) &= \text{coverage}(X \to Y) \times \text{confidence}(X \to Y)\\ &\geqslant \textit{minCoverage} \times \textit{minConfidence}\end{aligned}$$

□

Theorem 12 (Bound on support from leverage). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle$. *For any rule* $X \to Y \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$, $\text{support}(X \to Y) \geqslant \textit{minLeverage}$.

Proof. According to the definition of KOtask and that $X \to Y$ is in the solution, we obtain:

$$\begin{aligned}\text{support}(X \to Y) &= \text{leverage}(X \to Y) + \text{cover}(X) \times \text{cover}(Y)\\ &\geqslant \text{leverage}(X \to Y)\\ &\geqslant \textit{minLeverage}\end{aligned}$$

□

C.2. Relations between rules in KOtask

We investigate the relations between two rules when they share the same coverage value under some condition. We start with the following lemma.

Lemma 1 (Union cover). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$. *For any nonempty* $S_1, S_2, S_3 \subseteq \mathcal{C}$ *satisfying* $S_1 \cap S_2 = \emptyset$, $S_2 \cap S_3 = \emptyset$ *and* $S_1 \cap S_3 = \emptyset$, *if*

$$\text{cover}(S_1) = \text{cover}(S_1 \cup S_2) \tag{14}$$

the following holds.

$$\text{cover}(S_1 \cup S_3) = \text{cover}(S_1 \cup S_2 \cup S_3) \tag{15}$$

Proof. From (14) and the definition of KOtask, we obtain:

$$|\,\text{coverset}(S_1)| = |\,\text{coverset}(S_1 \cup S_2)| \tag{16}$$

From Observation 1 we obtain:

$$\text{coverset}(S_1) \supseteq \text{coverset}(S_1 \cup S_2) \tag{17}$$

From (16) and (17), we obtain:

$$\text{coverset}(S_1) = \text{coverset}(S_1 \cup S_2) \tag{18}$$

For any $d \in \mathcal{D} \wedge S_1 \cup S_3 \subseteq d$, $S_1 \subseteq d$ and $S_3 \subseteq d$ hold. From $S_1 \subseteq d$ and (18), we obtain $S_1 \cup S_2 \subseteq d$. And since $S_3 \subseteq d$, $S_1 \cup S_2 \cup S_3 \subseteq d$ holds. Hence we obtain:

$$\text{coverset}(S_1 \cup S_3) \subseteq \text{coverset}(S_1 \cup S_2 \cup S_3) \tag{19}$$

From Observation 1 we obtain:

$$\text{coverset}(S_1 \cup S_3) \supseteq \text{coverset}(S_1 \cup S_2 \cup S_3) \tag{20}$$

From (19) and (20), coverset$(S_1 \cup S_3)$ = coverset$(S_1 \cup S_2 \cup S_3)$ holds, proving (15). □

Theorem 13 (Upper bound on leverage from confidence). *Suppose KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$. *For any rule* $X \rightarrow Y$, *if* confidence$(X \rightarrow Y) = 1$, *for any* $X_1 \subset \mathcal{C}$ *satisfying* $X_1 \cap X = \emptyset$ *and* $X_1 \cap Y = \emptyset$, *the following holds.*

$$\text{leverage}(X \cup X_1 \rightarrow Y) \leqslant \text{leverage}(X \rightarrow Y)$$

Proof. From confidence$(X \rightarrow Y) = 1$ and the definition of KOtask, we obtain:

$$\text{cover}(X) = \mathit{cover}(X \cup Y) \tag{21}$$

According to the definition of KOtask and (21), we obtain:

$$\begin{aligned}
\text{leverage}(X \rightarrow Y) &= \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y) && (22)\\
&= \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y) && (23)\\
&= \text{cover}(X) - \text{cover}(X) \times \text{cover}(Y) && (24)\\
&= \text{cover}(X) \times (1 - \text{cover}(Y)) && (25)
\end{aligned}$$

From (21) and Lemma 1, we obtain:

$$\text{cover}(X \cup X_1) = \mathit{cover}(X \cup X_1 \cup Y) \tag{26}$$

According to the definition of KOtask and (26), we obtain:

$$\begin{aligned} &\text{leverage}(X \cup X_1 \to Y) && (27)\\ &\quad = \text{support}(X \cup X_1 \to Y) - \text{cover}(X \cup X_1) \times \text{cover}(Y) && (28)\\ &\quad = \text{cover}(X \cup X_1 \cup Y) - \text{cover}(X \cup X_1) \times \text{cover}(Y) && (29)\\ &\quad = \text{cover}(X \cup X_1) - \text{cover}(X \cup X_1) \times \text{cover}(Y) && (30)\\ &\quad = \text{cover}(X \cup X_1) \times (1 - \text{cover}(Y)) && (31) \end{aligned}$$

From Observation 1 we obtain $\text{cover}(X \cup X_1) \leqslant \text{cover}(X)$. Thus, from (25) and (31), the following holds.

$$\text{leverage}(X \cup X_1 \to Y) \leqslant \text{leverage}(X \to Y)$$

□

Theorem 14 (Unproductive specializations). *Suppose KOtask = $\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$. For any rule $X \to Y$ and $X \cup X_1 \to Y$ where $X \cap X_1 = \emptyset$, if*

$$\text{coverage}(X \to Y) = \text{coverage}(X \cup X_1 \to Y) \tag{32}$$

the following holds.

$$\text{support}(X \to Y) = \text{support}(X \cup X_1 \to Y) \tag{33}$$

$$\text{confidence}(X \to Y) = \text{confidence}(X \cup X_1 \to Y) \tag{34}$$

$$\text{leverage}(X \to Y) = \text{leverage}(X \cup X_1 \to Y) \tag{35}$$

Proof. According to (32) which is $\text{cover}(X) = \text{cover}(X \cup X_1)$, and from Lemma 1, we obtain:

$$\text{cover}(X \cup Y) = \text{cover}(X \cup X_1 \cup Y)$$

proving (33). From (32) and (33), (34) and (35) follow. □

C.3. Pruning the condition before it is added to the antecedent

This pruning rule, applied at line 3 in the KORD algorithm, prunes a condition in *AvailableLHS* before it is added to *CurrentLHS*.

Pruning 1. *In KORD for KOtask = $\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$, for any condition $P \in AvailableLHS$, if* $\text{cover}(\{P\}) < minCoverage$*, P can be pruned from $SoFar$.*

Given the upward closure of coverage, there does not exist any rule $X \rightarrow Y \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$ such that $P \in X$, thus P can be pruned from $SoFar$ so that P will not go into any $NewAvailableLHS$.

C.4. Pruning the new antecedent

This pruning rule, applied at line 6 in KORD, is used to prune the new antecedent $NewLHS$ which is formed by the union $CurrentLHS \cup \{P\}$ where $P \in AvailableLHS$.

Pruning 2. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, for* $NewLHS = CurrentLHS \cup \{P\}$ *where* $P \in AvailableLHS$*, if* $\text{cover}(NewLHS) < minCoverage$*,* P *can be pruned from* $SoFar$*.*

Given the upward closure of coverage, there does not exist any rule $X \rightarrow Y \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$ such that $NewLHS \subseteq X$, thus P can be pruned from $SoFar$ so that P will not go into any $NewAvailableLHS$ ready to be added to the antecedent containing $CurrentLHS$.

C.5. Pruning the consequent condition before the evaluation of rule

Pruning rules applied at line 10 in KORD are used to prune the consequent condition before the evaluation of a rule. We give three pruning rules.

Pruning 3. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}) \rangle$*, for any condition* $Q \in NewAvailableRHS$*, if* $\text{cover}(\{Q\}) < minSupport$*, then* Q *can be pruned from* $NewAvailableRHS$*.*

Given the upward closure for support, if $\text{cover}(\{Q\}) < minSupport$ then $\forall A \subseteq \mathcal{C}, \text{support}(A \rightarrow Q) < minSupport$, therefore Q can be pruned.

The second pruning rule functions according to the current lower bound on *minLeverage* before the evaluation of the rule. Note that the lower bound on *minLeverage* is the leverage of the k^{th} rule that satisfies the other criteria of those found so far, ordered from highest to lowest value on leverage.

Pruning 4. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, for the current* $NewLHS$*, for any condition* $Q \in NewAvailableRHS$*, if* $\text{cover}(\{Q\}) > 1 - \frac{minLeverage}{\text{cover}(NewLHS)}$*, then* Q *can be pruned from* $NewAvailableRHS$*.*

According to Theorem 6, if $\text{cover}(\{Q\}) > 1 - \frac{minLeverage}{\text{cover}(NewLHS)}$ then $NewLHS \rightarrow Q$ cannot be in the solution, therefore Q can be pruned.

The third pruning rule is for any consequent condition that covers the whole dataset.

Pruning 5. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle\rangle$*, for any condition* $Q \in \mathit{NewAvailableRHS}$*, if* $\mathrm{cover}(\{Q\}) = 1$ *and* $\mathit{minLeverage} > 0$*, then* Q *can be pruned from* $\mathit{NewAvailableRHS}$*.*

According to Theorem 9, if $\mathrm{cover}(\{Q\}) = 1$ then $\forall A \subseteq \mathcal{C}, \mathrm{leverage}(\mathit{NewLHS} \rightarrow Q) = 0$. When *minLeverage* > 0, such a rule cannot be in the solution. Therefore Q can be pruned.

C.6. Pruning the consequent condition after the evaluation of rule

Pruning rules applied at line 19 in KORD are used to prune the consequent condition after the evaluation of the current rule. We give three such pruning rules.

Pruning 6. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle\rangle$*, after the evaluation of the current rule* $\mathit{NewLHS} \rightarrow Q$*, if* $\mathrm{confidence}(\mathit{NewLHS} \rightarrow Q) = 1$ *and* $\mathrm{leverage}(\mathit{NewLHS} \rightarrow Q) < \mathit{minLeverage}$*,* Q *can be pruned from* $\mathit{NewAvailableRHS}$*.*

According to Theorem 13, if $\mathrm{confidence}(\mathit{NewLHS} \rightarrow Q) = 1$ then no rule with Q as the consequent in the search space below the current node can have higher leverage than $\mathit{NewLHS} \rightarrow Q$. Therefore if $\mathrm{leverage}(\mathit{NewLHS} \rightarrow Q) < \mathit{minLeverage}$, none of these rules can be in the solution. Hence Q can be pruned from $\mathit{NewAvailableRHS}$.

The second pruning rule utilizes *optConfidence*, an upper bound on the value of the confidence for a rule with Q as the consequent in the search space below the current node.

Pruning 7. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, after the evaluation of the current rule* $\mathit{NewLHS} \rightarrow Q$ *where* $\mathit{NewLHS} = \mathit{CurrentLHS} \cup \{P\}$*,* $P \in \mathit{AvailableLHS}$*, optConfidence is computed by:*

$$\mathit{optConfidence} = \frac{\mathrm{support}(\mathit{NewLHS} \rightarrow Q)}{\mathit{min\_cover}}$$

where

$$\mathit{min\_cover} = \max(\mathit{minCoverage}, \mathit{lower}) \quad (36)$$

$$\mathit{lower} = \mathrm{cover}(\mathit{NewLHS}) - \mathit{max\_spec} \times \mathit{max\_reduce} \quad (37)$$

$$\mathit{max\_spec} = \min(\mathit{maxLHSsize} - |\mathit{NewLHS}|, |\mathit{NewAvailableLHS}|)$$

$$\mathit{max\_reduce} = \max_{S \in \mathit{NewAvailableLHS}} \mathrm{reduce}(\mathit{CurrentLHS}, \{S\})$$

$$\mathrm{reduce}(X, Y) = (\mathrm{cover}(X) - \mathrm{cover}(X \cup Y))$$

If optConfidence $<$ minConfidence, Q can be pruned from $NewAvailableRHS$. Let

$$\textit{optLeverage} = \text{support}(NewLHS \rightarrow Q) - \textit{min\_cover} \times \text{cover}(\{Q\})$$

If optLeverage $<$ minLeverage, Q can be pruned from $NewAvailableRHS$.

Proof. Assume when *optConfidence* $<$ *minConfidence*, Q cannot be pruned from $NewAvailableRHS$. So there exists a rule $NewLHS \cup L \rightarrow Q \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k\rangle)$ where $L \subseteq NewAvailableLHS$ and $L \neq \emptyset$. Let $L = \{L_1\} \cup \cdots \cup \{L_l\}$ where $L_1, \cdots, L_l \in NewAvailableLHS, 1 \leqslant l \leqslant |NewAvailableLHS|$. Since only *maxLHSsize* of conditions are allowed on antecedent of rules in the solution, $l \leqslant \textit{maxLHSsize} - |NewLHS|$ holds. Thus we obtain:

$$\begin{aligned} l &\leqslant \min(\textit{maxLHSsize} - |NewLHS|, |NewAvailableLHS|) \\ &= \textit{max\_spec} \end{aligned} \tag{38}$$

cover($NewLHS \cup L$) is the number of records covered by $NewLHS$ minus the number of records not covered by $NewLHS \cup L$, divided by $|\mathcal{D}|$. Thus the following holds.

$$\begin{aligned} &\text{cover}(NewLHS \cup L) \\ &= \text{cover}(NewLHS) - \\ &\quad \frac{|\{d \,|\, d \in \mathcal{D} \wedge NewLHS \subset d \wedge NewLHS \cup L \not\subset d\}|}{|\mathcal{D}|} \end{aligned} \tag{39}$$

For any $d_1 \in \{d \,|\, d \in \mathcal{D} \wedge NewLHS \subset d \wedge NewLHS \cup L \not\subset d\}$, since $NewLHS = CurrentLHS \cup \{P\}$ and $NewLHS \subset d_1$, $CurrentLHS \subset d_1$ holds. From $NewLHS \cup L \not\subset d_1$, we obtain $L \not\subset d_1$. Thus $CurrentLHS \cup L \not\subset d_1$ holds. Hence,

$$d_1 \in \{d \,|\, d \in \mathcal{D} \wedge CurrentLHS \subset d \wedge CurrentLHS \cup L \not\subset d\}$$

Thus the following holds.

$$\begin{aligned} &\{d \,|\, d \in \mathcal{D} \wedge NewLHS \subset d \wedge NewLHS \cup L \not\subset d\} \\ &\quad \subseteq \{d \,|\, d \in \mathcal{D} \wedge CurrentLHS \subset d \wedge CurrentLHS \cup L \not\subset d\} \end{aligned} \tag{40}$$

According to the definition of *max_reduce* in (37), for any $1 \leqslant i \leqslant l$,

$$\begin{aligned} &\textit{max\_reduce} > \\ &\quad \frac{|\{d \,|\, d \in \mathcal{D} \wedge CurrentLHS \subset d \wedge CurrentLHS \cup \{L_i\} \not\subset d\}|}{|\mathcal{D}|} \end{aligned} \tag{41}$$

Since $L = \{L_1\} \cup \cdots \cup \{L_l\}$, and from (38), (40) and (41), we obtain:

$$\begin{aligned}
&\frac{|\{d|d \in \mathcal{D} \wedge NewLHS \subset d \wedge NewLHS \cup L \not\subset d\}|}{|\mathcal{D}|} \\
&\leqslant \frac{|\{d|d \in \mathcal{D} \wedge CurrentLHS \subset d \wedge CurrentLHS \cup L \not\subset d\}|}{|\mathcal{D}|} \\
&\leqslant \frac{\sum_{i=1}^{l} |\{d|d \in \mathcal{D} \wedge CurrentLHS \subset d \wedge CurrentLHS \cup \{L_i\} \not\subset d\}|}{|\mathcal{D}|} \\
&\leqslant max\_spec \times max\_reduce \qquad (42)
\end{aligned}$$

From (39) and (42), we obtain:

$$\text{cover}(NewLHS \cup L) \geqslant \text{cover}(NewLHS) - max\_spec \times max\_reduce \qquad (43)$$

Since $NewLHS \cup L \rightarrow Q \in \text{solution}(\langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle)$, $\text{cover}(NewLHS \cup L) \geqslant minCoverage$ holds. From (43) and (37), we obtain:

$$\text{cover}(NewLHS \cup L) \geqslant min\_cover$$

Therefore, according to Observation 1 confidence$(NewLHS \cup L \rightarrow Q)$ satisfies:

$$\begin{aligned}
\text{confidence}(NewLHS \cup L \rightarrow Q) &= \frac{\text{cover}(NewLHS \cup L \cup \{Q\})}{\text{cover}(NewLHS \cup L)} \\
&\leqslant \frac{\text{cover}(NewLHS \cup \{Q\})}{\text{cover}(NewLHS \cup L)} \\
&= \frac{\text{support}(NewLHS \rightarrow Q)}{\text{cover}(NewLHS \cup L)} \\
&\leqslant \frac{\text{support}(NewLHS \rightarrow Q)}{min\_cover} \\
&= optConfidence \\
&< minConfidence
\end{aligned}$$

This contradicts the proposition that $NewLHS \cup L \rightarrow Q$ is in the solution. Therefore Q can be pruned from $NewAvailableRHS$. Accordingly, leverage$(NewLHS \cup L \rightarrow Q)$ satisfies:

$$\begin{aligned}
&\text{leverage}(NewLHS \cup L \rightarrow Q) \\
&= \text{support}(NewLHS \cup L \rightarrow Q) - \text{cover}(NewLHS \cup L) \times \text{cover}(\{Q\}) \\
&= \text{cover}(NewLHS \cup L \cup \{Q\}) - \text{cover}(NewLHS \cup L) \times \text{cover}(\{Q\}) \\
&\leqslant \text{cover}(NewLHS \cup \{Q\}) - \text{cover}(NewLHS \cup L) \times \text{cover}(\{Q\})
\end{aligned}$$

$\leqslant \text{cover}(NewLHS \cup \{Q\}) - min\_cover \times \text{cover}(\{Q\})$
$= \text{support}(NewLHS \rightarrow Q) - min\_cover \times \text{cover}(\{Q\})$
$= optLeverage$
$< minLeverage$

This contradicts the proposition that $NewLHS \cup L \rightarrow Q$ is in the solution. Therefore Q can be pruned from $NewAvailableRHS$. □

Inherent in the selection of pruning rules is a trade-off between the amount of computation required to identify opportunities to prune and the amount of computation saved by applying pruning. The *optConfidence* measure requires little computation to evaluate, but provides a very loose upper bound on confidence, and hence is less effective at pruning than a tighter bound would be.

The third pruning rule utilizes a tighter bound on confidence, *optConfidence′* which requires a one-step lookahead to compute. This requires greater computation than *optConfidence*, and hence is evaluated only after other pruning rules have failed to prune a node in the search space.

Pruning 8. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, after the evaluation of the current rule* $NewLHS \rightarrow Q$*, optConfidence′ is computed by:*

$$optConfidence' = \frac{\text{support}(NewLHS \rightarrow Q)}{min\_cover'}$$

where min_cover' *is identical to* min_cover *(36), except that* $NewLHS$ *is used in place of* $CurrentLHS$ *in the definition of* max_reduce*. If optConfidence′ < minConfidence, Q can be pruned from* $NewAvailableRHS$*. Let*

$$optLeverage' = \text{support}(NewLHS \rightarrow Q) - min\_cover' \times \text{cover}(\{Q\})$$

If optLeverage′ < minLeverage, Q can be pruned from $NewAvailableRHS$*.*

The proof for this pruning rule mirrors that of the previous rule except that $NewLHS$ is used in place of $CurrentLHS$.

Although it requires considerable additional computation to evaluate $\text{cover}(NewLHS \cup \{S\})$ where $S \in NewAvailableLHS$ for min_cover', pruning by *optConfidence′* and *optLeverage′* can still improve the efficiency of the search.

D. Saving data access

D.1. Saving data access by identifying unqualified antecedents

Line 8 in KORD is for saving data access for the rules with $NewLHS$ as antecedent. We give two saving rules. The first is based on the theorem of minimum leverage for the antecedent of rules in the solution.

Saving 1. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, if* $|NewLHS| = maxLHSsize$ *and* $\text{cover}(NewLHS) \times (1 - \text{cover}(NewLHS)) < minLeverage$*, for any* $Q \in NewAvailableRHS$*, there is no need to access data to evaluate* $NewLHS \rightarrow Q$*, as it is not in the solution.*

Please note that although such a $NewLHS$ is not qualified to be the antecedent for a rule in the solution, it cannot be pruned since some of its supersets might have leverage larger than *minLeverage*. Saving the data access for all rules with $NewLHS$ as the antecedent might prevent application of pruning rules due to the absence of information about $\text{cover}(NewLHS \cup Q)$ where $Q \in NewAvailableRHS$. For this reason, we add the limitation to the above saving rule that $|NewLHS| = maxLHSsize$, which is the maximum search depth and below which no pruning will be performed. Saving data access at this stage cannot slow down the overall efficiency, as $\text{cover}(NewLHS \cup Q)$ is not used for pruning.

The second saving rule is for a $NewLHS$ that covers the whole dataset.

Saving 2. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, if* $|NewLHS| = maxLHSsize$*,* $\text{cover}(NewLHS) = 1$ *and* $minLeverage > 0$*, for any* $Q \in NewAvailableRHS$*, there is no need to access data to evaluate* $NewLHS \rightarrow Q$*, as it is not in the solution.*

According to Theorem 9, the leverage value of any rule with such $NewLHS$ as the antecedent is 0. When $minLeverage > 0$, such a rule cannot be in the solution. However, such a $NewLHS$ cannot be pruned since some of its supersets may not cover the whole dataset anymore and thus can make the rule have leverage larger than 0.

D.2. Saving data access by identifying unqualified rules

Line 13 in KORD is for saving data access for the current rule $NewLHS \rightarrow Q$. We give two saving rules. The first is based on Theorem 8.

Saving 3. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle\rangle$*, for the current rule* $NewLHS{\rightarrow}Q$*, if* $|NewLHS| = maxLHSsize$ *and* $\frac{\text{cover}(Q)}{\text{cover}(NewLHS)} < minConfidence$*, there is no need to access data to evaluate* $NewLHS{\rightarrow}Q$*, as it is not in the solution.*

The reason that the saving is adopted instead of pruning under this situation is in the branch below the current $NewLHS{\rightarrow}Q$, some of the supersets of $NewLHS$ with lower values of coverage might make the rule have confidence larger than *minConfidence*. While saving data access, it is no longer possible to perform pruning based on the results of the data access. In consequence, the overall efficiency might be slowed down accordingly. Due to this, the constraint $|NewLHS| = maxLHSsize$ is added to the above saving rule to ensure that it is applied only at the maximum search depth where no pruning is necessary.

The next saving rule is based on Theorem 6.

Saving 4. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle\rangle$*, for the current rule* $NewLHS{\rightarrow}Q$*, if* $|NewLHS| = maxLHSsize$ *and* $\text{cover}(NewLHS) > 1 - \frac{minLeverage}{\text{cover}(\{Q\})}$*, there is no need to access data to evaluate* $NewLHS{\rightarrow}Q$*, as it is not in the solution.*

Any rule as described above has a value for leverage less than *minLeverage*. However no pruning should be performed here as some of the supersets of $NewLHS$ might have lower cover than $(1 - \frac{minLeverage}{\text{cover}(\{Q\})})$ and thus have leverage larger than *minLeverage*.

D.3. Saving data access by identifying generalizations with identical statistics

During the evaluation of the current rule $NewLHS{\rightarrow}Q$, where $NewLHS = CurrentLHS \cup \{P\}$, $P \in AvailableLHS$, line 14 in KORD adopts a data access saving rule utilizing the relationship between $CurrentLHS$ and P. It is based on the theorem of relation from coverage.

Saving 5. *In KORD for KOtask* $= \langle \mathcal{C}, \mathcal{D}, \mathcal{M}, \lambda, k \rangle$*, for the current rule* $NewLHS{\rightarrow}Q$ *where* $NewLHS = CurrentLHS \cup \{P\}$*,* $P \in AvailableLHS$*, if* $|NewLHS| = maxLHSsize$*, the number of rules in curentSolution is less than* $|\text{coverset}(NewLHS)|$*, and* $\text{cover}(CurrentLHS) = cover(NewLHS)$*, instead of accessing data to evaluate* $NewLHS{\rightarrow}Q$*, check if* $CurrentLHS{\rightarrow}Q$ *exists in currentSolution, and if yes, copy all the statistic values of* $CurrentLHS{\rightarrow}Q$ *to* $NewLHS{\rightarrow}Q$*, otherwise,* $NewLHS{\rightarrow}Q$ *is not in the solution.*

Since $CurrentLHS \rightarrow Q$ is investigated before $NewLHS \rightarrow Q$ in KORD, and they share the same statistic values, $NewLHS \rightarrow Q$ will be in $currentSolution$ if and only if $CurrentLHS \rightarrow Q$ is in $currentSolution$. Due to the same reasons as in the above subsection, we add $|NewLHS| = maxLHSsize$ in the saving rule to make sure that application of the saving rule cannot slow down the overall efficiency. If the number of rules in $currrentSolution$ is greater than $|\,\text{coverset}(NewLHS)|$, searching these rules might be less efficient than accessing $\text{coverset}(NewLHS)$ to compute $\text{cover}(NewLHS \cup Q)$.

References

Agarwal, R. C., C. C. Aggarwal, and V. V. V. Prasad: 2000, 'Depth First Generation of Long Patterns'. In: *Proceedingsof the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2000).* Boston, MA, pp. 108–118, ACM.

Agrawal, R., T. Imielinski, and A. Swami: 1993, 'Mining Associations between Sets of Items in Massive Databases'. In: *Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data*. Washington, DC, pp. 207–216.

Agrawal, R., H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo: 1996, 'Fast Discovery of Association Rules'. In: U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.): *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA.: AAAI Press, pp. 307–328.

Agrawal, R. and R. Srikant: 1994, 'Fast Algorithms for Mining Association Rules'. In: *Proceedings of the 20th International Conference on Very Large Databases*. Santiago, Chile, pp. 487–499.

Bay, S. D.: 2001, 'The UCI KDD Archive'. [http://kdd.ics.uci.edu] Irvine, CA: University of California, Department of Information and Computer Science.

Bayardo, Roberto J., J.: 1998, 'Efficiently Mining Long Patterns from Databases'. In: *Proceedings of the 1998 ACM-SIGMOD International Conference on Management of Data*. pp. 85–93.

Bayardo, Roberto J., J. and R. Agrawal: 1999, 'Mining the Most Interesting Rules'. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 145–154.

Bayardo, Roberto J., J., R. Agrawal, and D. Gunopulos: 2000, 'Constraint-Based Rule Mining in Large, Dense Databases'. *Data Mining and Knowledge Discovery* **4**(2/3), 217–240.

Blake, C. and C. J. Merz: 2001, 'UCI Repository of Machine Learning Databases'. [Machine-readable data repository]. University of California, Department of Information and Computer Science, Irvine, CA.

Borgelt, C.: 2000, 'Apriori'. (Computer Software) `http://fuzzy.cs.Uni-Magdeburg.de/~borgelt/`.

Buchanan, B. G. and E. A. Feigenbaum: 1978, 'DENDRAL and Meta-DENDRAL: Their Applications Dimension'. *Artificial Intelligence* **11**, 5–24.

Clark, P. and T. Niblett: 1989, 'The CN2 Induction Algorithm'. *Machine Learning* **3**, 261–284.

Clearwater, S. H. and F. J. Provost: 1990, 'RL4: A Tool for Knowledge-based Induction'. In: *Proceedings of Second Intl. IEEE Conf. on Tools for AI*. Los Alamitos, CA, pp. 24–30, IEEE Computer Society Press.

Cohen, E., M. Datar, S. Fujiwara, A. Gionis, R. Indyk, P. Motwani, J. Ullman, and C. Yang: 2000, 'Finding Interesting Associations without Support Pruning'. In: *Proceedings International Conference on Data Engineering*.

Cohen, W. W.: 1995, 'Fast Effective Rule Induction'. In: *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann.

Han, J., J. Pei, and Y. Yin: 2000, 'Mining Frequent Patterns without Candidate Generation'. In: *Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*. Dallas, TX, pp. 1–12.

Kohavi, R., C. Brodley, B. Frasca, L. Mason, and Z. Zheng: 2000, 'KDD-Cup 2000 organizers' report: Peeling the onion'. *SIGKDD Explorations* **2**(2), 86–98.

Liu, B., W. Hsu, and Y. Ma: 1998, 'Integrating Classification and Association Rule Mining'. In: *Proceedings Knowledge Discovery and Data Mining (KDD-98)*. pp. 80–86.

Liu, B., W. Hsu, and Y. Ma: 1999, 'Mining Association Rules with Multiple Minimum Supports'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*. San Diego, CA, pp. 337–341.

Michalski, R. S.: 1977, 'Synthesis of Optimal and Quasi-optimal Variable-valued Logic Formulas'. In: *Proceedings of the 1975 International Symposium on Multiple Valued Logic*. Bloomington, Indiana, pp. 76–87, Reidel.

Morishita, S. and A. Nakaya: 2000, 'Parallel Branch-and-Bound Graph Search for Correlated Association Rules'. In: *Proceedings of the ACM SIGKDD Workshop on Large-Scale Parallel KDD Systems*, Vol. LNAI 1759. Berlin: Springer, pp. 127–144.

Pasquier, N., Y. Bastide, R. Taouil, and L. Lakhal: 1999, 'Discovering frequent closed itemsets for association rules'. In: *Proceedings of the Seventh International Conference on Database Theory (ICDT'99)*. Jerusalem, Israel, pp. 398–416.

Pei, J., J. Han, and R. Mao: 2000, 'CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets'. In: *Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery (DMKD'00)*. Dallas, TX, pp. 21–30.

Piatetsky-Shapiro, G.: 1991, 'Discovery, Analysis, and Presentation of Strong Rules'. In: G. Piatetsky-Shapiro and J. Frawley (eds.): *Knowledge Discovery in Databases*. AAAI/MIT Press, pp. 229–248.

Provost, F., J. Aronis, and B. Buchanan: 1999, 'Rule-Space Search for Knowledge-Based Discovery'. CIIO Working Paper IS 99-012, Stern School of Business, New York University, , NY, NY 10012.

Rymon, R.: 1992, 'Search through Systematic Set Enumeration'. In: *Proceedings KR-92*. Cambridge, MA, pp. 268–275.

Savasere, A., E. Omiecinski, and S. Navathe: 1995, 'An Efficient Algorithm for Mining Association Rules in Large Databases'. In: *Proceedings of the 21st International Conference on Very Large Data Bases*. pp. 432–444, Morgan Kaufmann.

Segal, R. and O. Etzioni: 1994, 'Learning Decision Lists Using Homogeneous Rules'. In: *AAAI-94*. Seattle, WA, AAAI press.

Todorovski, L., P. Flach, and N. Lavrac: 2000, 'Predictive Performance of Weighted Relative Accuracy'. In: D. A. Zighed, J. Komorowski, and J. Zytkow (eds.): *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*. pp. 255–264, Springer-Verlag.

Toivonen, H.: 1996, ‘Sampling Large Databases for Association Rules’. In: *Proceedings of the 22nd International Conference on Very Large Data Bases*. pp. 134–145, Morgan Kaufmann.

Webb, G. I.: 1995, ‘OPUS: An Efficient Admissible Algorithm for Unordered Search’. *Journal of Artificial Intelligence Research* **3**, 431–465.

Webb, G. I.: 2000, ‘Efficient search for association rules’. In: *The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, MA, pp. 99–107, The Association for Computing Machinery.

Zaki, M. J.: 2000, ‘Generating Non-Redundant Association Rules’. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2000)*. Boston, MA, pp. 34–43, ACM.

Zheng, Z., R. Kohavi, and L. Mason: 2001, ‘Real World Performance of Association Rule Algorithms’. In: *KDD-2001: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*. San Francisco, pp. 401–406, ACM.