Computational Intelligence, Volume 19, Number 3, 2003

MINING SHORT-RULE COVERS IN RELATIONAL DATABASES

CLAUDIO CARPINETO AND GIOVANNI ROMANO

Fondazione Ugo Bordoni, Via B. Castiglione 59, I-00142, Rome, Italy

An implication rule $Q \rightarrow R$ is a statement of the form "for all objects in the database, if an object has the attribute–value pairs Q then it has also the attribute–value pairs R." This simple type of rule is theoretically interesting, because it supports reasoning, similar to functional dependencies in database theory, and it may be of practical significance because the size of the set of implication rules that hold in a relation can remain substantially high even when mining real data and considering only most general covers; i.e., covers containing rules with unredundant right and left sizes. Motivated by these observations, we focus on the extraction of short-rule covers, which cannot be efficiently mined by standard rule miners. We present an algorithm driven by "negative examples" (i.e., satisfy Q but not R) to prune the rule-candidate lattice associated with each "positive example" (i.e., satisfies both Q and R). The algorithm scales up quite well with respect to the number of objects and it is particularly suitable for databases with attributes described by large domains. Furthermore, a perfect hash function ensures extraction of short-rule covers even from databases containing a large number of attributes.

Key words: knowledge discovery from databases, implication rules, data mining, hashing, computational complexity.

1. INTRODUCTION

Much recent research across a number of fields including databases, knowledge discovery, and artificial intelligence has been aimed at supporting data analysis and decision making through automatic mining of rules in relational databases. While differing in many respects, most rule miners share a common goal: to find all possible rules of a given type that can be extracted from data, usually subject to a set of user-specified constraints (Piatetski-Shapiro 1991; Agrawal, Imielinski, Swami 1993; Berry and Linoff 1997; Bayardo, Agrawal, and Gunopulos 1999; Carpineto, Romano, and d'Adamo 1999).

This contrasts with other methods for inducing rules from data, such as those developed in machine learning, which are primarily biased towards producing minimum subsets of classification rules (Clark and Niblett 1989; Cai, Cercone, and Han 1991; Quinlan 1993; Cohen 1995). Due to the use of domain-independent biases and heuristics, which may not agree with the user knowledge, these latter systems suffer from the understandability problem of the generated set of rules (Pazzani, Mani, and Shankle 1997). Furthermore, the use of classical induction algorithms such as decision trees to perform rule mining may easily result in the omission of equally plausible rules or valuable rules that are relatively rare (Riddle, Segal, and Etzioni 1994), while a straightforward adaptation of such algorithms, although possible, would suffer from serious inefficiency as well as redundancy problems (Schlimmer 1993; Oosthuizen 1994).

One of the most studied rule-mining task with completeness guarantees is the extraction of association rules from data (Agrawal et al. 1993; Agrawal and Srikant 1994; Agrawal et al. 1996; Park, Chen, and Yu 1997). Roughly, an association rule $Q \rightarrow R$ implies that "for a certain percentage of the objects (or records) in a database, if an object has properties Q then it has also properties R." More precisely, an association rule $Q \rightarrow R_{\mathrm{minsup,minconf}}$ holds if the percentage of objects that have both Q and R is greater than "minimum support," and the percentage of objects with Q, which have also R is greater than "minimum confidence."

Association rules have proved to be useful in a variety of tasks, including data analysis, prediction, and classification (Srikant, Vu, and Agrawal 1997; Liu, Hsu, and Ma 1998), but

Address correspondence to Claudio Carpineto at Fondazione Ugo Bordoni, Via B. Castiglione 59, I-00142, Rome, Italy; e-mail: carpinet@fub.it.

their use has been typically confined to application domains characterized by sparse data matrices (e.g., sales data), with the objects being usually described by a variable number of single-valued attributes. For relational data involving objects described by a fixed number of multivalued attributes, as found in many natural domains, the number of association rules may grow too large, easily in tens of thousands, to be mined efficiently or to be analyzed by the end user. In fact, the use of specialized algorithms for mining multidimensional association rules and the addition of constraints to prune or summarize the discovered set of association rules are two foci of much current research (Kamber, Han, and Chiang 1997; Beyer and Ramakrishnan 1999; Bayardo et al. 1999; Liu, Hsu, and Ma 1999; Shah et al. 1999; Liu et al. 2000; Pei, Han, and Lakshmanan 2001).

Another class of nonheuristic data regularities that have recently received some attention is represented by implication rules (IRs), sometimes called implications, or simply rules (Wille 1992; Godin and Missaoui 1994; Ziarko and Shan 1996; Carpineto et al. 1999). An IR $Q \rightarrow R$ is a statement of the form "for all objects in the database, if an object has Q then it has also R." Theoretically speaking, implications can be seen as a special case of association rules, with 100% confidence and support of at least one object. In fact, implications have different properties and cannot be efficiently extracted using classical association rule-mining algorithms based on the exploitation of the support threshold. Implications are also closely related to functional dependencies, which can be seen as an abstraction of the former rules (for a thorough discussion of the relationships between IRs and functional dependencies see (Carpineto et al. 1999)).

IRs are less flexible than association rules, because discovery of approximate dependencies is not permitted. On the other hand, they have the advantage of supporting reasoning, similar to functional dependencies in database theory, whereas association rules do not support inference axioms unless we make particular assumptions about the description of the data (Toivonen et al. 1995; Padmanabhan and Tuzhilin 2000). In particular, by analogy with work done in database theory, a definition of the most general cover for the set of IRs that hold in a relation can be introduced. This is an important property for rule-discovery systems, because it provides known and well-founded methods for finding compact representations of the set of rules generated.

Not only are IRs theoretically appealing, they have also a practical interest. The need for considering rules of extremely high confidence without regard for their support has been discussed by Cohen et al. (2000) and Wang et al. (2001). They point out that this kind of rules are a natural class of patterns in a number of applications including copy detection, text mining, and collaborative filtering. In such cases, it is likely that rules with low support and high confidence are interesting and provide new insights, whereas high-support rules are obvious and well known.

The focus of this paper is on IRs. We believe that a more widespread acceptance and utilization of this approach has been hindered so far by a shortage of theoretical and experimental evidence suggesting its utility and overall feasibility for practical data mining. The goal of this research is to contribute to fill this gap.

We first study the space complexity of the set of IRs that hold in a relation, showing that their number may be substantially high even when mined in real data. This result is somewhat unexpected, because IRs do not admit exceptions (i.e., the rules must hold for all the objects in the relation). We then argue that it is possible to achieve a significant and justified compactness gain by eliminating rule redundancy, similar to recent work on maximal item sets for association rules (Burdick, Calimlim, and Gehrke 2001; Gouda and Zaki 2001), although such a gain may still be insufficient to allow direct utilization of the generated rules by the end user. Furthermore, the available algorithms for computing an unredundant set of IRs may incur serious computational limitations.

These observations lead us to concentrate on the problem of reducing the set of rules generated, while increasing the overall efficiency of their generation. We are particularly interested in two features of the rule-mining process: to find a smaller but justified set of rules, without hurting the implication theory of unconstrained IRs, and to ensure the computational feasibility of their generation even for large databases.

For this purpose, we focus on the mining of *short* IRs, with the user specifying the maximum admissible length of the sought rules. Short rules preserve the implication theory, they perform well (Holte 1993; Riddle et al. 1994; Liu et al. 1998), and are easier to understand for the end user. Furthermore, with this restriction it is possible to perform data mining in large relational databases, where mining the whole set of rules would be computationally infeasible.

A major part of this paper is represented by the actual determination of the most general cover of the set of short IRs that hold in a relation. We present a novel and efficient algorithm in the context of lattice-space search, borrowing and extending ideas developed for concept induction in version spaces (Mitchell 1982; Mellish 1991; Carpineto 1992). The time complexity of the algorithm is roughly linear in the number of objects and it is nearly insensitive to the number of values per attribute in most practical situations, while its exponential growth with respect to the number of attributes can be controlled by limiting the length of the generated rules. We introduce a perfect hash function that ensures polynomial complexity, thus allowing extraction of short rules even from databases described by hundreds of attributes. These combined features allow our algorithm to deal with databases characterized by a large description space, i.e., containing many attributes and many values per attribute, which usually represents a difficult task for most current rule miners.

The rest of the paper is organized as follows. In the next section, we introduce most general IRs and discuss their utility. In Section 3, we analyze the space complexity of mining IRs theoretically and experimentally, which suggests focusing on short rules. Section 4 is devoted to the automatic determination of rule covers. A description of the algorithm for inferring unconstrained IRs is followed by the definition of a perfect hash function for efficient mining of short IRs and by a discussion of the complexity of the overall algorithm. Section 5 relates our research to existing methods for mining IRs and association rules, as well as to other brute-force approaches to knowledge discovery. Section 6 concludes the paper with a summary and some directions for future work.

2. RULES AND RULE COVERS

We assume that the data are represented by a relation. More precisely, given a set of objects (O), a set of attributes (A), and a set of attribute values (V), a relation is a quadruple (O, A, V, I), where I is a ternary relation between O, A, and V (i.e., $I \subseteq O \times A \times V$) such that $(o, a, v_1) \in I$ and $(o, a, v_2) \in I$ imply $v_1 = v_2$. Note that $(o, a, v) \in I$ reads: the object o has the value v for the attribute a; instead of writing $(o, a, v) \in I$ we can write $a(o) = v$. This is the usual relation employed in relational databases; our notation, borrowed from the field of formal concept analysis (Ganter and Wille 1997), has the advantage of expliciting taking into account both the presence of attributes and attribute values, thus facilitating analysis and comparison of rule-mining systems. IRs are defined in the following way.

Definition 1. An IR between two sets of attribute–value pairs is an expression $[(r_1, s_1), (r_2, s_2), \ldots, (r_h, s_h)] \rightarrow [(t_1, u_1), (t_2, u_2), \ldots, (t_k, u_k)]$, where $(r_x, s_x), (t_x, u_x) \subseteq (A \times V)$.

A relation (O, A, V, I) satisfies the IR $[(r_1, s_1), (r_2, s_2), \ldots, (r_h, s_h)] \rightarrow [(t_1, u_1), (t_2, u_2), \ldots, (t_k, u_k)]$ if $\forall\ o \in O$, $[r_1(o) = s_1 \wedge r_2(o) = s_2 \wedge \cdots\ r_h(o) = s_h] \Rightarrow [t_1(o) = u_1 \wedge t_2(o) = u_2 \wedge \cdots\ t_k(o) = u_k]$.

In other terms, an IR between two subsets of attribute–value pairs Q and R means that if a set of objects satisfies the attribute–value pairs contained in Q then it necessarily satisfies the attribute–value pairs contained in R. It should be noted that, following Definition 1, an IR may hold vacuously; i.e., when there is no object that supports it. Because rules that hold vacuously may be meaningless, or even misleading, for practical data mining, it seems useful to require that there should be at least one object containing all attribute–value pairs (r_x, s_x), (t_x, u_x) that describe each rule.

Owing to the nature of IRs, the inference system developed in database theory for functional dependencies (Maier 1983) holds also for IRs. The only caution is that the inferred rules should be supported by at least one object. By taking advantage of the logical implications between rules it is possible to reduce the representation of the set of IRs that hold in a relation. For this purpose, it is convenient to recall the following concepts, developed in the database theory. Given a set Σ of IRs, the closure Σ^+ is the set of rules implied by Σ by application of Armstrong's inference axioms (Maier 1983; page 48), i.e., reflexivity ($Q \rightarrow Q$), augmentation ($Q \rightarrow R$ implies $QZ \rightarrow R$), and pseudotransitivity ($Q \rightarrow R$ and $RZ \rightarrow W$ implies $QZ \rightarrow W$). Two sets Σ and Σ' are equivalent if they have the same closure. If Σ and Σ' are equivalent, then Σ' is a *cover* for Σ. Of course, it is interesting to find compact covers. We propose the following definition for the most general cover of the set of IRs that hold in a relation.

Definition 2. A cover Σ' is most general if:

(a) every right side of an IR in Σ' is a single attribute–value pair,
(b) for no $Q \rightarrow R$ in Σ' and proper subset S of Q is $\Sigma' - \{Q \rightarrow R\} \cup \{S \rightarrow R\}$ equivalent to Σ'.

In terms of Maier (1983), this is to say that we look for covers that are both right-reduced and left-reduced. Intuitively, condition (a) improves readibility and understandibility of the generated rules, while condition (b) guarantees that no attribute–value pair on any left side is redundant (i.e., left sides are maximally general). Although rule redundancy may be useful in certain situations, condition (b) allows the user to focus on those properties in the antecedent which are logically necessary to determine the consequent.

As an illustration, we refer to a simple database consisting of five objects described by five attributes with binary values (see Table 1). The set of implications of type lhs $\rightarrow r_1$ is

$$\{b_1 d_1, b_1 c_2, b_1 c_1 d_1, a_1 b_1 d_1, a_1 c_1 d_1, b_1 c_2 d_1, a_1 b_1 c_2, a_1 b_1 c_1 d_1, a_1 b_1 c_2 d_1\}.$$

The most general cover for this set of implications is given by

TABLE 1. An Example Database.

a_1	b_1	c_1	d_1	r_1
a_1	b_1	c_1	d_2	r_2
a_2	b_2	c_1	d_1	r_2
a_1	b_2	c_2	d_1	r_2
a_1	b_1	c_2	d_1	r_1

$$\{b_1d_1, b_1c_2, a_1c_1d_1\}.$$

If we consider the implications having a_2 as consequent, we get, for the whole set

$$\{b_2c_1, b_2c_1d_1, b_2c_1r_2, c_1d_1r_2, b_2c_1d_1r_2\},$$

and the most general cover is

$$\{b_2c_1, c_1d_1r_2\}.$$

Here we have considered just two attribute values (i.e., r_1 and a_2) for the sake of simplicity; the complete set of implications holding in the database can be obtained by collecting the implications holding for each possible attribute value.

Besides data mining, IRs are useful for other tasks such as database design and data warehousing. IRs cover can be efficiently transformed into functional dependency covers (Missaoui and Godin 1994; Carpineto et al. 1999), which are used for designing third normal form database relations (Maier 1983). Also, IRs allow for polynomial inference of embedded implications, (i.e., those implications, which hold for subsets of attributes), whereas the general case is exponential (Taouil and Bastide 2001). Embedded implications can be used for database exploration, as well as for data warehousing (Laurent et al. 1999).

IRs enjoy two useful monotonicity properties that will be used for their determination.

Proposition 1. If $Q \rightarrow R$ does not hold, then for any subset S of Q, $S \rightarrow R$ will not hold either.

Proposition 2. If $Q \rightarrow R$ holds, then for any superset S of Q, $S \rightarrow R$ will also hold, provided that $S \rightarrow R$ is satisfied by at least one object.

Note that these properties do not hold for association rules. A weaker condition than Proposition 2 holds for the case when we consider association rules without regard for their support and with a given minimum confidence (Wang et al. 2001).

Finding the most general cover guarantees the elimination of all redundant rules, but it does not rule out, per se, rules with a long left hand side. Because long rules are difficult to understand and tend to overfit the data, it may be convenient to consider only the shorter ones. In this way we might achieve a significant reduction of the number of admissible rules while arguably preserving the most informative ones. This view is supported by a number of results showing the good performance of short rules on prediction tasks (e.g., Holte 1993; Riddle et al. 1994; Liu et al. 1998). In addition, as we will see, the extraction of short rules allows us to perform data mining even in large databases, where mining the whole set of rules would be computationally infeasible.

Short rules are defined more precisely in the following way.

Definition 3. Let m the number of attributes and n the number of objects. Given an integer number q (length threshold), $1 \leq q \leq m - 1$, a short IR is an IR with $\|lhs\| \leq q$.

Note that for $q = m - 1$, we get just the definition of unconstrained IRs, and that the introduction of the threshold q does not adversely affect reasoning in that inference axioms, with obvious modifications, hold also for short IRs.

In the next section we study the computational space complexity of the set of IRs that hold in a relation.

3. SPACE COMPLEXITY OF IMPLICATION RULES

3.1. Effect of Main Parameters on Rule Size

We consider the three main parameters of the problem: the number of objects n, the number of attributes m, and the number of values per attribute v. We assume that numeric values have been discretized using some method (we will return to this in Section 3.3).

There are two theoretical upper bounds on the number of IRs that hold in a relation. One is related to the size of the rule-description space, the other to the number of objects. Assuming there are m attributes with v values each, the conjunctive language used to describe the rules contains $[(v+1)^{m-1}-1]mv$ possible distinct elements, regardless of the number of objects. On the other hand, with n objects it is possible to generate at most $(2^{m-1}-1)mn$ rules; i.e., when the n objects have no attribute–value pairs in common. The actual theoretical bound is the smaller one.

Besides considering the upper bounds, it is useful to see how the growth of each parameter affects the size of the set of IRs that hold in a relation.

As the number of objects grows, the number of IRs may decrease or it may increase. To explain this, consider that the introduction of a new object may result in new IRs holding between combinations of attribute–value pairs that had not been seen before; but it may also disconfirm IRs that held previously. The number of rules, in contrast, grows monotonically with respect to the number of attributes in the relation, because an expansion of the set of attributes describing the objects will not contradict any prior rule and, in addition, will usually form new rules. A similar situation holds for the number of values per attribute. Increasing the value of this parameter (for instance by reducing the interval length in a uniform discretization procedure for handling numeric attributes) results in a monotonic growth of the number of IRs, because formerly valid rules will not be affected by the change and new rules will typically appear.

These considerations provide some insights into the behavior of the size of the set of IRs that hold in a relation, but we are still in a position that does not allow us to understand the effect of parameter interaction, let alone to estimate the actual size in situations of interest. Let us study this problem for the simple case in which the objects are uniformly distributed over the object-description space.

3.2. Theoretical Space Complexity of Rule Covers

We would like some formula for the expected number of rules (NR) that hold in a database under the assumption of uniform distribution of the attribute values. For simplicity, we consider only the rules with $\|\text{rhs}\| = 1$, i.e., a right-reduced cover of the set of implications.

Ideally, the formula can be obtained by summing up the probabilities of occurrence of each possible rule. To make this computation feasible, it is convenient to consider the set of rules with a same rhs and with $\|\text{lhs}\| = k (1 \leq k \leq m-1)$, where each rule has the same probability of occurrence. There are $\binom{m-1}{k}v^k$ possible distinct rules of this kind with $\|\text{lhs}\| = k$. The probability that there is a rule with exactly i supporting objects $(1 \leq i \leq n)$ is given by the product of the probability that the first i objects support the rule (i.e., $[\frac{1}{v^{k+1}}]^i$) by the probability that the remaining $n-i$ objects neither support nor contradict the rule (i.e., $[1-\frac{1}{v^k}]^{n-i}$) by the number of possible ways to choose the supporting i objects among the n objects in the relation (i.e., $\binom{n}{i}$). Note that the second probability factor ensures that there is no other object supporting the rule and that the rule is not disconfirmed by some of the remaining objects.

As each rule must be supported, by definition, by *at least* one object, the expected number of rules with $\|\text{lhs}\| = k$ is

$$\binom{m-1}{k} v^k \sum_{i=1}^{n} \binom{n}{i} \left[\frac{1}{v^{k+1}}\right]^i \left[1 - \frac{1}{v^k}\right]^{n-i}. \tag{1}$$

To derive the complete formula we have to sum expression (1) over k and multiply the result by the number of possible right-reduced hand sides (mv):

$$\text{NR} = mv \sum_{k=1}^{m-1} \binom{m-1}{k} v^k \sum_{i=1}^{n} \binom{n}{i} \left[\frac{1}{v^{k+1}}\right]^i \left[1 - \frac{1}{v^k}\right]^{n-i}. \tag{2}$$

Formula (2) can be written, more compactly, as:

$$\text{NR} = mv \sum_{k=1}^{m-1} \binom{m-1}{k} v^k \left[\left(1 - \frac{1}{v^k} + \frac{1}{v^{k+1}}\right)^n - \left(1 - \frac{1}{v^k}\right)^n\right]. \tag{3}$$

Figure 1 (both scales are logarithmic) shows results plotted from formula (3) for four pairs of values for (m, v): (5, 5), (10, 2), (10, 5), (10, 10). For $n = 1$, $\text{NR} = m2^{m-1}$; i.e., for each attribute–value pair contained in the object it is possible to form rules with all possible combinations of the remaining $(m - 1)$ attribute–value pairs. For each fixed pair (m, v), as long as $n \ll v^m$, NR grows linearly with respect to n. For larger values of n, because previously unseen attribute–value pairs are encountered less frequently, NR decreases; in particular, as n grows to infinity, NR tends to zero because when all possible combinations of

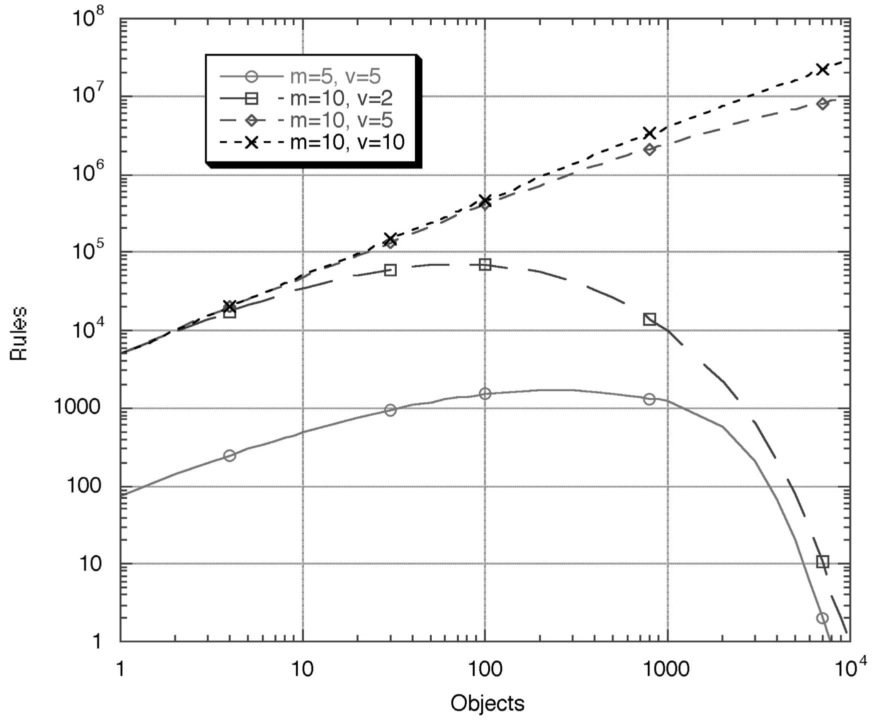

FIGURE 1. Theoretical size complexity of right-reduced covers under uniform distribution.

attribute–value pairs have been seen, there exist no more implications in the database. This behavior is apparent for the two lower curves in Figure 1, in which the object-description space is small (5^5 and 2^{10}, respectively).

Formula (2) shows that NR grows nearly exponentially with respect to the number of attributes, similar to the theoretical upper bound. Formula (2) also shows that NR grows monotonically with respect to v, from $(2^m - 1)m$, for $v = 1$ (i.e., all possible rules associated with one distinct object), to $(2^m - 1)mn$, as v tends to infinity (i.e., the theoretical upper bound holding when the objects have no attribute–value pairs in common). Finally, Figure 1 shows that, for a fixed value of m, as v grows the value of NR remains nearly stable, provided that $n \ll v^m$ (see the two upper curves in Figure 1).

It should be noted that although formula (2) has been derived for unconstrained IRs ($q = m - 1$), it can easily accomodate for a smaller length threshold. It is sufficient to let the index over the first summation vary in the range $1, \ldots, q$ ($1 \leq q \leq m - 1$).

We found experimental results consistent with the theoretical findings reported above. For several combinations of n, m, and v, for which a right-reduced cover of the set of implications could be easily built automatically, we created corresponding artificial data sets with random assignment of values, then computed the size of their right-reduced covers averaging the result of each data set over multiple runs. Comparisons between experimental results and formula (3) showed remarkable similarity across parameter combinations. To perform this computation, we used the algorithm that will be presented in Section 4; while the algorithm has been devised to find a most general cover of the set of IRs that hold in a relation, it can be easily adapted to generate a cover that is right-reduced but not left-reduced.

We have to emphasize that our results hold for random descriptor assignment. While, we have chosen this model because it is simple to analyze, we are aware that it may not be a good model for attribute values. In order to gain some insight into the behavior of the IRs complexity in more realistic domains, in the next section we experiment with a number of natural data sets. Furthermore, we are interested in estimating the size of the most general cover; i.e., a cover that is not only right-reduced but also left-reduced. Deriving a formula for the most general cover similar to formula (2) is probably too difficult, so we extend the experimental study to analyze also this issue

3.3. Experimental Space Complexity of Rule Covers

We used nine machine-learning benchmarks drawn from the UCI repository (Murphy and Aha 1995), whose main features are described in Table 2. As our method is suitable only for discrete data sets (at most, it can handle integer values with a small range), the values of numeric attributes were discretized into ten equal-length intervals; this choice represents a compromise between using a strong discretization (e.g., binary attributes), which would compress the diversity of the objects, and using a weak discretization, which would hide their similarity. For the missing values, we used the overall modal (most frequent) value for nominal attributes and Boolean features and used the overall mean for numeric attributes. Note, however, that a special treatment of missing values is not necessary in our approach; we could just ignore them.

We used also an artificially generated data set ("Random"), containing 300 objects and 10 attributes, with 10 randomly assigned values per attribute. After this treatment, the number of values per attribute in the considered data sets (not shown for space limitations) ranged from 2.9 to 10.2, with an average of 8.16.

For each data set we computed the size of its right-reduced cover and of its most general cover. The results are shown in Table 2 (all the programs tested in this paper have been implemented in Common Lisp on a SUN Ultra 2 equipped with 512 Mb of RAM).

TABLE 2. Experimental Size of Rule Covers.

Dataset	No. of objects	No. of attributes	Size of right-reduced cover	Size of most general cover	Size of short-rule cover $q = 1$	$q = 2$	$q = 3$
Abalone	4177	9	87209	6261 (7.1%)	88	1105	2794
Breast cancer	286	10	137835	7954 (5.7%)	30	724	2483
Breast cancer (W)	699	11	1157686	47432 (4.1%)	98	5902	34235
Bridges	108	13	1545023	10083 (0.6%)	158	2566	5165
Glass	214	11	409064	9357 (2.2%)	259	3241	4246
Liver disorder	345	7	10224	2934 (28.7%)	72	786	1552
Nursery	12960	9	0	0	0	0	0
Pima	768	9	202478	32952 (16.2%)	90	1922	10671
Random	300	10	1331017	150089 (11.2%)	0	6571	142690
Tic tac toe	958	10	136430	14760 (10.8%)	0	0	44

The most striking evidence is the huge number of implications, with three data sets scoring more than one million rules and 7 out of the 10 tested data sets yielding hundreds of thousands of rules. These results were somewhat unexpected, because one might believe that the number of IRs in a natural data set should be small. Our experiments show that this is not the case. This finding may represent an indication that, for many practical domains, it may be more useful to concentrate on how to reduce sets of simple rules instead of trying to mine more powerful or flexible types of rules, which are likely to unnecessarily further expand the set of rules to be pruned.

It should be noted that the observation about the high number of implications was confirmed by all data sets except for "Nursery," which yielded no rule at all. The anomalous behavior of Nursery can be explained by considering that it contains a large number of objects (12,960) with a relatively small description space (eight nominal attributes described by 3.6 values per attribute on average). Thus, its features approximate the theoretical behavior of the two lower curves in Figure 1.

The second main indication of our experiments is that when passing from right-reduced covers to the most general cover the size of the rule set shrinks sharply. In our experiments, the size of the most general cover was usually one and sometimes two orders of magnitude smaller; on average, it reduced to 8.7% of the size of the original redundant cover. This is a significant compactness gain entirely due to the implication theory.

However, the resulting rule set was often nmanageably large because it usually contained thousands of rules. The most critical factor for the size of rule covers, including the most general cover, seems to be the number of attributes and the number of values per attribute, as also suggested by earlier theoretical results. Mannila and Räihä (1994) show that for some relations over m attributes *all* the covers of *functional dependency* sets are of exponential size in m. Thus, for the same relations, all the covers of IR sets are *a fortiori* of exponential size in m. Our results suggest that this may be actually the case even for practical databases, unless the number of objects becomes larger than their description space.

Thus, the exponential growth of the size of most general covers with respect to the number of attributes may easily cause serious computational and usability problems even for data sets described by a relatively small number of attributes. Fortunately, the use of short rules may significantly alleviate this problem.

The third main result of our experiments is that the length threshold is an effective means to generate rule sets of practical significance even for very small values of the threshold. Indeed, Table 2 shows that for some data sets, even the number of rules with minimum length ($q = 1$) is relatively large. This may represent a futher indication that natural data sets often exhibit high regularities that can be mined using IRs.

We also considered how the number of rules of a fixed length varies as the length increases. It turned out that middle sets ($1 \leq q < m - 1$) usually contained many more rules than sets with short or long rules, somewhat similar to a normal distribution. Thus, the user may choose to generate manageable sets of rules at both ends of the length spectrum.

In the next section we describe an algorithm for computing the most general cover of the set of IRs that hold in a relation.

4. AN ALGORITHM FOR INDUCING A MOST GENERAL RULE COVER

For each attribute–value pair (a, v) present in the relation, we want to find the rule set LHS containing all possible maximally general rules of the form lhs $\rightarrow (a, v)$. To solve this task, we present an algorithm that has been inspired by earlier work on concept induction in version spaces, where the search space pruning is driven by examples and counterexamples (Mitchell 1982; Mellish 1991; Carpineto 1992).

The set of objects in the relation can be partitioned in two classes, i.e, the set of “positive” objects such that $a(o) = v$, and the set of “negative” objects such that $a(o) \neq v$. Given the definition of IRs, and provided that the constraint on the rule length is not violated, a positive object will support all rules of the form lhs $\rightarrow (a, v)$, where lhs is a subset of the attribute–value pairs describing the positive object; a negative object will inhibit all rules of the form lhs $\rightarrow (a, v)$ where lhs is a subset of the attribute–value pairs describing the negative object. For instance, assuming $(a, v) = r_1$, the (positive) object $a_1b_1c_1d_1r_1$ will support the rule $a_1b_1 \rightarrow d_1$, while the (negative) object $a_1b_1c_2d_2r_2$ will inhibit it.

The main loop of the algorithm iterates on the positive objects. The algorithm keeps and updates for each positive object the set of rules that can be theoretically generated from it; i.e., all the possible 2^{m-1} subsets (for the moment we consider the case $q = m - 1$) of the attribute–value pairs describing the object, except for (a, v). On an abstract level, for each positive object, the algorithm must first remove the rules that are contradicted by at least one negative object, then it must collect only the most general rules of the remaining ones, and finally it must update the set that contains the most general rules produced by all positive objects that have been examined until that point. These are three computationally expensive problems.

The key idea of the algorithm is to use an ordered structure for representing the candidate-rule space, over which the first two operations can be performed efficiently and in an integrated manner.

Let $O' = O - \{a, v\}$ the set of all attribute–value pairs describing the object except for (a, v), and consider the ordered set $\{P(O'); \geq^*\}$, formed by the power set of O' and by the standard set inclusion relation (i.e., $x \geq^* y$ if $x \subseteq y$). This ordered set is well formed with respect to the task at hand (see Propositions 1 and 2 in Section 2). In particular, if an element is ruled out by some negative object then all its greater ($\geq^*$) elements are also ruled out; conversely, if an element is the lhs of a valid rule then all its smaller elements are also valid rules.

Using $\{P(O'); \geq^*\}$ allows us to handle pruning due to negative objects in a straightforward manner. For each negative object the algorithm finds the intersection between the negative object and the current positive object and prunes all the elements that are greater than or equal to the intersection from the set $\{P(O'); \geq^*\}$. The second step involved in each iteration is the determination of the most general rules produced by the current positive object, which can be performed by a specific-to-general breadth-first search through the elements of $\{P(O'); \geq^*\}$ that have not been pruned.

In the final operation of each iteration, the algorithm must consistently add the rules found in the earlier step to the set that contains the maximally general rules produced by all positive objects that have been examined so far. It turns out that any rule produced by an object can be neither greater nor smaller (according to the ordering relation of $\{P(O'); \geq^*\}$) than any other most general rule produced by any earlier object, because (i) for each positive object the algorithm collects only the maximally general rules and (ii) the set of negative objects is the same for each positive object. Thus, we must simply add all rules produced by the current object that are not already contained in the cumulative rule set to the cumulative rule set itself.

A complete description of the algorithm is given in Table 3. The introduction of the threshold q $(1 \leq q \leq m - 1)$ is easily taken into account in the logic flow of the algorithm, but it requires careful implementation if we want to take advantage of the hypothetical reduction of the candidate-rule space, as will be discussed in Section 4.2.

It should also be noted that the same algorithm, with small modifications, can be used to find a cover that is right-reduced but not left-reduced. It is sufficient to turn off the statement which, for each candidate node, checks if all parents of the candidate node have been labelled as "del" ("pruning implied candidates" step in Table 3). This variant of the algorithm has been used in the experiments involving right-reduced covers.

TABLE 3. The Algorithm for Inferring a Most General Cover for Short Implication Rules.

```
Find-LHS
Input: a relation $(O, A, V, I)$, an attribute–value pair $(a^*, v^*)$ such that $(o, a^*, v^*) \in I$ for some $o \in O$,
       a length threshold $\beta$.
Output: a set LHS containing all maximally general rules of the form lhs $\rightarrow (a^*, v^*)$
begin
LHS : = Ø
$O_p$-set : = $\{o \in O \mid a^*(o) = v^*\}$;          /* set of positive objects */
$O_n$-set : = $\{o \in O \mid a^*(o) \bullet v^*\}$;          /* set of negative objects */
for each $O_p \in O_p$-set do          /* $O_p$ is a set of pairs $(a, v)$ */
  $O' = O_p - \{(a, v)\}$;
  $P_l(O')$: = $\{C \in \{P(O'); \geq^*\}$ such that $\|C\| \leq \beta\}$;          /* candidate-lhs set */
  for each $O_n \in O_n$-set do          /* pruning disconfirmed candidates */
    Int : = $O_p \cap O_n$;
    for each $C \in \{P_l(O'); \geq^*\}$ such that $C \geq^*$ Int do
      $C$.label : = "del"
    endfor
  endfor;
  for $C \in \{P_l(O'); \geq^*\}$ such that $C$.label $\bullet$ "del" do
    if all parents of $C$ are such that parent.label = "del" and /* pruning implied candidates */
      $C$ is not a member of LHS then          /* avoiding duplicate rules */
        LHS : = LHS $\cup \{C\}$
    endif
  endfor
endfor
return (LHS)
end
```

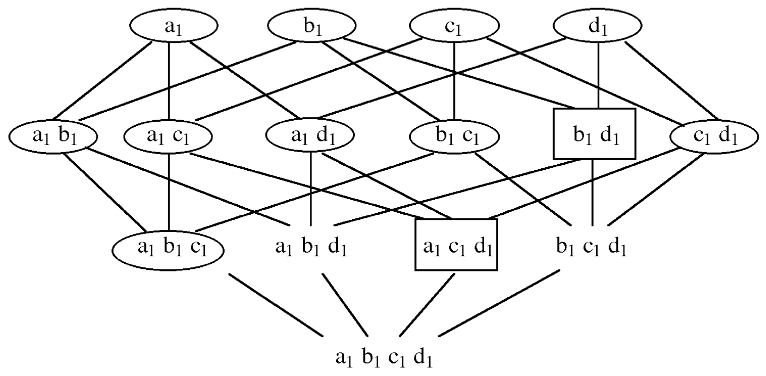

FIGURE 2. Ordered set of the lhs candidates for the rules lhs $\rightarrow r_1$ associated with the first (positive) object of the database shown in Table 1. Encircled elements are pruned by negative objects, boxed elements are the most general remaining candidates.

4.1. An Example

To illustrate the working of the algorithm, consider again the example database shown in Table 1. Let r_1 be the current rhs, let q = 4, and consider the iteration relative to the first positive object ($a_1b_1c_1d_1r_1$). The corresponding set $\{P(a_1, b_1, c_1, d_1); \geq^*\}$ is shown as a graph in Figure 2.

Examination of the first negative object ($a_1b_1c_1d_2r_2$) causes pruning of $a_1b_1c_1$ (intersection between $a_1b_1c_1d_2r_2$ and the current positive object) and of its ancestors (a_1b_1, a_1c_1, b_1c_1, a_1, b_1, and c_1) from the graph. Examination of the second negative object ($a_2b_2c_1d_1r_2$) causes pruning of element c_1d_1 and of its unpruned ancestors (d_1). Finally, the arrival of the third negative object ($a_1b_2c_2d_1r_2$) has the effect of pruning a_1d_1. Once all negative objects have been examined, the unpruned elements in the graph are visited and the most general of them are collected ($a_1c_1d_1$ and b_1d_1). The other positive object ($a_1b_1c_2d_1r_1$) produces two most general lhs candidates: b_1c_2 and b_1d_1, the second of which has already been generated. The final result is the LHS set containing the elements $a_1c_1d_1$, b_1d_1, and b_1c_2.

4.2. Implementation Issues

As one of the main objectives of our research is to make data mining from dense data computationally feasible, the actual complexity of the *Find-LHS* algorithm for mining short rules is of central importance. In this section we describe an efficient implementation based on tha use of a bitmap representation coupled with hashing.

By assigning an order to the attribute–value pairs present in $O' = O - \{a, v\}$, each element of $P(O')$ may be represented as a bit vector (BV), where BV[i], $I = 1, 2, \ldots, m - 1$, is equal to 1 or 0 depending on whether the attribute–value pair i is present or not in O'. From each BV it may then be computed a function that returns the address of an array cell that contains the information about the element of $P(O')$ corresponding to the BV, i.e., whether or not it has been marked by some negative object.

One simple solution to compute such a function is to treat the BV as a number and use it as an index of an array with 2^{m-1} cells, one for each element of $P(O')$. However, this approach is unsatisfying, because it allows us to deal only with databases described by a small number of attributes, regardless of the chosen value of the length threshold, whereas we want to be able to deal at least with small values of q even for large values of m.

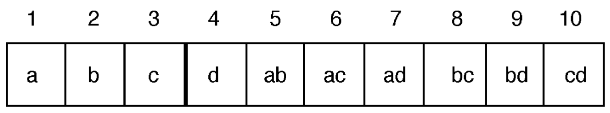

FIGURE 3. Storing of the lhs candidates of size ≤ 2 shown in Figure 2.

Alternatively, we can try to use an array that contains only the elements of $P(O')$ with $\|\text{lhs}\| \leq q$; i.e., the set of all rules of length q or smaller, denoted $P_q(O')$. One possible way to implement this strategy is the following. First of all, we order the elements of $P_q(O')$ by arranging them in q ascending cardinality partitions (sub-arrays), from 1 to q, with the elements within each partition sorted using a trie-like enumeration order. In other words, this inner enumeration consists of placing all attribute–value pairs contained in O' in a fixed order prior to the generation, and then generating the next element by replacing the lowest ranked pair in the current element with the next-ranked pair.

To illustrate, consider an example in which O' contains four elements, noted a, b, c, d (for simplicity we drop the attribute–value pair subscripts). Assume that $q = 2$. The storing of the elements of $P_2(O')$, assuming that the lexicographic order is such that a precedes b, which precedes c, which precedes d, is shown in Figure 3.

We now determine the function that maps the bit-vector representation of each element e of $P_q(O')$ on to the position of e in the reduced array. The problem may be decomposed into two parts. The first is to compute the address of the first cell of the sub-array relative to the elements of $P_q(O')$ with cardinality equal to that of e. This is given by $\sum_{i=1}^{k-1}\binom{m-1}{i} + 1$, where k is the number of bits equal to 1 in the BV corresponding to e. The second part of the problem is to compute the relative position of e into the relevant sub-array, which is determined by the number of elements that precede e given the trie-like enumeration order. In practice, this can be computed by summing up the lengths of the sequences of (sub-array) cells that must be skipped for each attribute not present in BV.

More specifically, assume that the attributes of BV are sorted in decreasing order, and suppose that there is a bit 0 in the ith position of BV. The length of the sequence of cells that must be skipped for such a bit is given by the number of combinations that can be formed with $m - 1 - i$ attributes (i.e., the attributes that have not been considered yet), chosen $\sum_{j=i}^{m-1} \text{BV}[j] - 1$ at a time (i.e., the number of bits equal to 1 in BV that follow the ith bit). To explain, consider that each cell that must be skipped can be seen as formed by two subsequences of attributes. The first contains all the attributes corresponding to the bit 1 of BV that have already been seen; the second subsequence contains as many attributes as $k - \sum_{j=1}^{i-1} \text{BV}[j] = \sum_{j=i}^{m-1} \text{BV}[j]$, chosen among all the attributes in the context that have not been considered yet.

The overall function is therefore

$$\sum_{i=1}^{k-1}\binom{m-1}{i} + 1 + \sum_{i=1}^{m-1}\left(\begin{matrix} m-1-i \\ \sum_{j=i}^{m-1} \text{BV}[j] - 1 \end{matrix}\right)(1 - \text{BV}[i]) \tag{4}$$

The final factor in formula (4) makes all the contributions of the bits of BV that are equal to 1 null. When $\sum_{j=i}^{m-1} \text{BV}[j] > 1$, the second term of formula (4) is equal to 0, because $\binom{x}{y} = 0$ for $y < 0$. Taken together, this implies that the attributes with a lower rank than the lowest-ranked attribute in BV (i.e., those after the last bit = 1 in BV) do not affect the computation of the position of the element corresponding to BV in the relevant sub-array. In fact, the index of the outer summation in the second term of formula (4) can be safely restricted to the last bit of BV = 1.

To illustrate the working of formula (4), suppose that the element of $P_2(O')$ of which we want to find the address into the array shown in Figure 3 is *bd*, whose BV is 0101. Thus, we have $m = 5$, $k = 2$. The first part in expression (4) returns 5, which is the address of the first element of the relevant sub-array. The second part in expression (4) is a summation with two terms. The contribution of the first bit 0 of BV (i.e., BV[1]) is $\binom{3}{1+1-1} = 3$, corresponding to the skipped sequence "*ab, ac, ad,*" while the contribution of the other bit 0 of BV (i.e., BV[3]) is $\binom{1}{0+1-1} = 1$, corresponding to the skipped cell "*bc*." Thus, the address of *bd*, as computed by formula (4), is 9 (see Figure 3).

The algorithm in Table 3 requires not only direct access to each element of $P(O')$, but also to its parents (according to $\geq^*$). The latter can be easily achieved by masking the bit 1 of the BV representation of the element itself and then by applying the same procedure described above to each of the newly generated BVs.

Thus, this implementation ensures perfect hashing for any value of q ($1 \leq q \leq m - 1$), because the size of the reduced array coincides with the size of $P_q(O')$; i.e.,

$$\sum_{i=1}^{q} \binom{m-1}{i}. \tag{5}$$

The only caution is that step "pruning" in Table 3, when $\|\text{int}\| > l$, requires the determination—via bit masking—of all groups with q elements that can be formed out of the attribute–value pairs present in int. For example, consider again the processing of the first positive object ($a_1b_1c_1d_1r_1$) in the example given in Section 4.1, but this time assume that q = 2. The intersection with the first negative object returns $a_1b_1c_1$ (BV = 1110), which is not contained in the search space, because it exceeds the length threshold. In fact, we must replace $a_1b_1c_1$ with all its more-general elements with cardinality = 2(a_1b_1, a_1c_1, and b_1c_1), and use each of them as a distinct intersection.

The final step of the Find-LHS algorithm requires the the update of the set LHS that contains the implications with their associated support factors. As the size of the set LHS may grow large, it is convenient to represent it as a trie-like structure. In this way, testing whether or not a candidate rule is contained in LHS (step "collection" in Table 3) requires time constant with respect to the size of the set.

We emphasize that by using this particular representation of the involved data structures, they will completely fit into most current main memories, even when considering moderate-sized rules and relations containing hundreds of attributes, which would otherwise be clearly unfeasible. For instance, using expression (5) with $m = 100$, the rule-candidate space contains 99, 4851, 156849, 3764376 elements for $q = 1$, $q = 2$, $q = 3$, $q = 4$, respectively. Expression (5) represents the actual space complexity of the Find-LHS algorithm; note that expression (5) is considerably smaller than m^l, and that, for any fixed q, m^q is polynomial in m.

4.3. Time Complexity of the Rule-Cover Finding Algorithm

In this section we analyze the time complexity of the Find-LHS algorithm. For simplicity, we first consider generation of all rules.

For each positive object the algorithm first checks all negative objects, pruning the disconfirmed rules from the candidate-rule space; then it selects the most general of the remaining rules, and finally it updates the output rule set. Because the first two steps require visiting distinct nodes and testing their parents, the time complexity involved is bounded by the maximum branching factor times the cardinality of the candidate-rule space. In fact, for each positive object the number of nodes visited is constant: $(m/2)2^{m-1}$, where $m/2$ is just the average branching factor of the elements in $(P(O'); \geq^*)$. The third step (i.e., update of the trie representing the output rule set) takes time proportional to gmv, where g is the number of maximally general rules associated with the current positive object. Thus, the time complexity for each positive object cannot exceed $\Theta((n_n + (m/4)2^m) + gmv)$, where n_n is the number of negative objects, and the time complexity of the Find-LHS algorithm is therefore $< \Theta(n_p((n_n + (m/4)2^m) + gmv))$, where n_p is the number of positive objects. In practice, $n \ll m2^m$ and $gv \ll 2^m$. As a consequence, in many practical applications, the complexity is nearly $\Theta(n_p m 2^m)$.

For the case when we are interested in short rules $(1 \leq q \leq m - 1)$, the adaptation is straightforward. Considering that, for any fixed q, the number of nodes visited is upper bounded by mq, and that the average branching factor of the elements in $(P_q(O'); \geq^*)$ is upper bounded by m, the complexity is less than $\Theta(n_p m^{q+1})$.

5. RELATED WORK

5.1. Implication Rules

Ziarko and Shan (1996) have presented an IR-finding method based on the computation of prime implicants of Boolean expressions. Their method is based on a particular representation of the input relation, called decision matrix, from which the rules are then extracted. Similar to our approach, the cover output by Ziarko and Shan contains only the most general rules. However, the complexity of the method, as also noted by the authors, seems to be prohibitively high even for databases of limited size; in fact, no evidence is reported in the paper that in some real domains such an approach would be feasible.

Another approach to discovery of IRs is based on a particular clustered representation of the input relation called concept lattice, from which the rules are then extracted (Godin and Missaoui 1994; Carpineto et al. 1999). Concept lattice should not be confused with the lattice shown in Figure 1 because in a concept lattice only some combinations of the attribute–value pairs describing the objects are admissible. The complexity of the concept lattice-based algorithm is not usually exponential in the number of attributes, as for the Find-LHS algorithm, but it grows at least as a cubic function of the number of objects, and it cannot be reduced by considering only short rules because we still need to construct the full concept lattice associated with the input database. An additional advantage of our method is that the IRs mined with the concept lattice-based approach are not maximally general because the cover generated satisfies a weaker property than condition (b) of Definition 2, namely, it guarantees the elimination of those rules that can be obtained from some other rule r in the same set by shifting some attribute–value pairs of r from the rhs to the lhs.

5.2. Association Rules

As IRs can be seen as a special case of association rules, one might be tempted to use association rule miners to mine also IRs. However, this would not work for a number of reasons.

Standard association rule miners including APRIORI (Agrawal et al. 1996) and its many variants are based on the empirical assumption that item sets of large size do not receive enough support from the data and therefore do not need to be generated. However, this behavior holds for transactional data but not for dense, relational data, where it has been observed a combinatorial explosion of frequent item sets (Bayardo 1997; Bayardo et al. 1999). For the case when the support threshold is very low, as with IRs, the situation gets even worse because one needs to mine also infrequent item sets. Because, there may be as many as $(v + 1)^m$ or $n(2^m - 1)$ distinct item sets in a relational table with n objects, m attributes, and v values per attribute, the association rule miners might thus become altogether inefficient.

Alternative approximate methods for generating high-confidence, free-support association rules have been recently proposed by Cohen et al. (2000). Such methods trade off accuracy and speed, thus favoring computational efficiency at the cost of generating incorrect rules. Furthermore, the rules generated may contain only rules of the form $Q \rightarrow R$, where Q and R are single attribute–value pairs (e.g., pairs of words that occur together in news articles).

In order to mine IRs it seems more convenient to focus on the implicit structure of the data, as with the Find-LHS algorithm. Given the absence of a minimum-support threshold, and consistent with the quest for most general rules, the pruning of the candidate-rule space is ensured by the well-formedness of the lattice search space with respect to the task at hand, which allows us to make inferences about the supersets and subsets of each candidate. Another advantage of our algorithm is that its complexity is nearly insensitive to the number of values per attribute for most databases of interest, whereas this parameter may represent a critical factor for the complexity of association rule miners, even when we are only interested in extracting short rules, because the number of item sets of size i is given by $\binom{v \cdot m}{i}$.

5.3. Hypothesis-Driven Rule Mining

There have been several early proposals for rule induction with a machine-learning focus that perform massive search through an ordered hypothesis space rather than greedy search driven by data (Carpineto 1992; Smyth and Goodman 1992; Schlimmer 1993; Riddle et al. 1994; Webb 1995). More recently, a similar search paradigm has been used in the data-mining field to find particular subsets of association rules such as maximal or closed sets (Bayardo 1998; Silverstein, Brin, and Motwani 1998; Pasquier et al. 1999; Zaki 2000; Burdick et al. 2001). These systems make use of a range of techniques, including lattice-based search, set-enumeration, hashing, and inference-driven pruning, which are also at the core of our approach.

The algorithm presented in this paper differs from previous work in the choice of the representation space being searched and in the pruning strategy. The rule-mining task is cast as an instance-driven search problem, to which the powerful methods developed for learning concepts from examples and counter-examples can be applied. The Find-LHS algorithm maintains an explicit representation of the search space associated with each "positive" object in the database, then updates it with the corresponding "negative objects," and finally merges the results. In this way, our algorithm does not incur the computational limitations associated with the representation of a prohibitively large search space such as the one containing all possibile attribute–value pairs, while still performing massive pruning due to the structuring of the data in each subspace.

Another distinguishing feature of our work is that we have proposed a perfect hash function that allows safe and efficient restriction to the portion of search space of interest (i.e., the one containing short rules), whereas this issue has not been usually addressed in other hypothesis driven approaches to rule induction. On the other hand, most former systems can handle approximate rules, which is not easy to accomodate in our framework.

6. CONCLUSION AND FUTURE WORK

We believe that the potentials of IRs for knowledge discovery have not been fully investigated. The aim of our research was to help fill this gap. We presented theoretical and experimental results suggesting that, although IRs do not admit exceptions, there may be a huge number of such rules that hold in a relation. Generating only maximally general rules can greatly reduce the number of valid rules, although this alone is not sufficient to guarantee their direct utilization by the end user. Motivated by these findings, we concentrated on the task of mining most general covers of short rules. We presented an efficient lattice-based search algorithm that performs inference-driven pruning, similar to earlier methods for concept induction in version spaces. The method scales up quite well with respect to the number of objects and it is particularly suitable for databases with attributes described by large domains due to its insensitivity to the number of values describing each attribute; furthermore, a perfect hash function ensures extraction of short-rule covers even from databases containing a large number of attributes. These features are not usually handled well by most standard rule miners.

In this paper, we did not address the utility issue. The next step of our research is an experimental evaluation of the predictive power of IRs in selected domains. Natural candidates are news articles, copy detection, and transactions, where the utility of rules with very low support has been demonstrated (Cohen et al. 2000; Wang et al. 2001). A direct utilization of IRs for class prediction on the “natural” data sets used throughout the paper seems more difficult, although the determination of such rules might be used as a preprocessing step of a larger learning system that employs additional rule selection mechanisms, similar to the approach reported in (Carpineto and Romano 1993).

A further important issue for future work is the identification of extensions to IRs that capture patterns in data not captured by standard IRs. Interesting examples of such extensions are represented by dependencies that hold in both the presence and absence of properties describing the data (Silverstein et al. 1998) and by approximate dependencies (Kivinen and Mannila 1994; Skowron and Polkowski 1997). Enriching IRs with the possibility of **Q1** handling negation and noise without loosing reasoning capabilities may significantly extend the theoretical scope of our method, although we should remark again that, in practice, it may be more useful to concentrate on how to reduce the number of IRs rather than generating rules with a richer description language.

One principled way to perform such a reduction would be to keep with the analogy with work done in database theory (Ullman 1988) by finding a *minimal* covers for the set of IRs, i.e., a most general cover which, in addition, contains dependencies that are not redundant by transitivity (formally, for no $Q \rightarrow R$ in Σ' is the set $\Sigma' - \{Q \rightarrow R\}$ equivalent to Σ'). This would result in more succint representations of the set of rules, although the user would probably find it more difficult to derive those rules that are logically implied by the minimal ones by transitivity. The set of holding IRs could be also pruned by evaluating their interestingness with probabilistic or heuristic techniques (Silberschatz and Tuzhilin 1996; Shah et al. 1999), or by using the user’s knowledge about the domain (Liu et al. 2000).

ACKNOWLEDGMENTS

The authors thank two anonymous reviewers for many useful comments and suggestions.

REFERENCES

AGRAWAL, R., T. IMIELINSKI, and A. SWAMI. 1993. Mining association rules between sets of items in large databases. *In* Proceedings of ACM SIGMOD, pp. 207–216. **Q2**

AGRAWAL, R., and R. SRIKANT. 1994. Fast algorithms for mining association rules. *In* Proceedings of the Twentieth VLDB Conference, Santiago, Chile, pp. 487–499.

AGRAWAL, R., H. MANNILA, R. SRIKANT, H. TOIVONEN, and A. I. VERKAMO. 1996. Fast discovery of association rules. *In* Advances in Knowledge Discovery and Data Mining, AAAI-Press, pp. 307–328. **Q3**

BAYARDO, R. 1997. Brute-force mining of high-confidence classification rules. *In* Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, pp. 123–126.

BAYARDO, R. 1998. Efficiently mining long patterns from databases. *In* Proceedings of the 1998 ACM SIGMOD Conference on Management of Data, pp. 85–93.

BAYARDO, R., R. AGRAWAL, and D. GUNOPULOS. 1999. Constraint-based rule mining in large, dense databases. *In* Proceedings of the Fifteenth International Conference on Data Engineering, pp. 188–197.

BERRY, M., and G. LINOFF. 1997. Data Mining Techniques for Marketing, Sales and Customer Support. John Wiley & Sons, Inc, New York. **Q4**

BEYER, K., and R. RAMAKRISHNAN. 1999. Bottom-up computation of sparse and iceberg CUBEs. SIGMOD Record, **28**(2):359–370.

BURDICK, D., M. CALIMLIM, and J. GEHRKE. 2001. MAFIA: A maximal frequent itemset algorithm for transactional databases. *In* Proceedings of the Seventeenth International Conference on Data Engineering, Heidelberg, Germany, pp. 443–452.

CAI, Y., N. CERCONE, and J. HAN. 1991. Learning in relational databases: An attribute-oriented approach. Computational Intelligence, **7:**119–132.

COHEN, E., M. DATAR, S. FUJIWARA, A. GIONIS, P. INDYK, R. MOTWANI, J. ULLMAN, and C. YANG. 2000. Finding interesting associations without support pruning. *In* Proceedings of the Sixteenth International Conference on Data Engineering, IEEE Computer Society, San Diego, CA, pp. 489–499.

CARPINETO, C. 1992. Trading off consistency and efficiency in version-space induction. *In* Proceedings of Ninth International Conference on Machine Learning, Aberdeen, Scotland, pp. 43–48.

CARPINETO, C., and G. ROMANO. 1993. GALOIS: An order-theoretic approach to conceptual clustering. *In* Proceedings of Tenth International Conference on Machine Learning, Amherst, Massachusetts, pp. 33–40.

CARPINETO, C., G. ROMANO, and P. D'ADAMO. 1999. Inferring dependencies from relations: A conceptual clustering approach. Computational Intelligence, **15**(4):415–441.

CLARK, P., and T. NIBLETT. 1989. The CN2 induction algorithm. Machine Learning, **3**(4):261–283.

COHEN, W. 1995. Fast effective rule induction. *In* Proceedings of the Twlefth International Conference on Machine Learning, Morgan Kaufmann, Tahoe City, CA, pp. 115–123.

GANTER, B., and R. WILLE. 1997. Formal Concept Analysis: Mathematical Foundations, Springer, Heidelberg.

GODIN, R., and R. MISSAOUI. 1994. An incremental concept formation approach for learning from databases. Theoretical Computer Science (special issue on formal methods in databases and software engineering), **133:**387–419.

GOUDA, K., and M. ZAKI. 2001. Efficiently mining maximal frequent itemsets. *In* Proceedings of the First IEEE International Conference on Data Mining, san Jose, CA, pp. 163–170.

HOLTE, R. 1993. Very simple classification rules perform well on most commonly used datasets. Machine Learning, **11**(1):63–90.

KAMBER, M., J. HAN, and J. Y. CHIANG. 1997. Metarule-guided mining of multi-dimensional association rules using data cubes. *In* Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, pp. 207–210.

KIVINEN, J., and H. MANNILA. 1994. Approximate inference of functional dependencies from relations. Theoretical Computer Science, **149:**129–149.

LAURENT, D., J. LECHTENBORGER, N. SPYRATOS, and G. VOSSEN. 1999. Complements for data warehouses. *In* Proceedings of the Fifteenth International Conference on Data Engineering, pp. 490–499.

LIU, B., W. HSU, and Y. MA. 1998. Integrating classification and association rule mining. *In* Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York, pp. 80–86.

LIU, B., W. HSU, and Y. MA. 1999. Pruning and summarizing the discovered associations. *In* Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, pp. 125–134.

LIU, B., W. HSU, S. CHEN, and Y. MA. 2000. Analyzing the subjective interestingness of association rules. IEEE Intelligent Systems, **15**(5). **Q5**

MAIER, D. 1983. The Theory of Relational Databases. Computer Science Press, Rockville, MD.

MANNILA, H., and K RÄIHÄ. 1994. Algorithms for inferring functional dependencies from relations. Data & Knowledge Engineering, **12**(1):83–99.

MITCHELL, T. 1982. Generalization as search. Artificial Intelligence, **18:**203–226.

MELLISH, C. 1991. The description identification problem. Artificial Intelligence, **52**(2):151–168.

MISSAOUI, R., and R. GODIN. 1994. Search for concepts and dependencies in databases. *In* Rough Sets, Fuzzy Sets and Knowledge Discovery. *Edited by* W. ZIARKO, Workshops in Computing series, Springer-Verlag, New York, pp. 16–23.

MURPHY, P., and D. AHA. 1995. UCI Repository of Machine Learning Databases (Machine-Readable Data Repository), University of California, Department of Information and Computer Science, Irvine, CA.

OOSTHUIZEN, D. 1994. The application of concept lattices to machine learning. Technical Report CSTR 94/01, University of Pretoria.

PADMANABHAN, B., and A. TUZHILIN. 2000. Small is beautiful: Discovering the minimal set of unexpected patterns. *In* Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000), Boston, MA, pp. 54–64.

PARK, J., M. CHEN, and P. YU. 1997. Using a hash-based method with transaction trimming for mining association rules. IEEE Transactions on Knowledge and Data Eineering, **9**(5):813–825.

PASQUIER, N., Y. BASTIDE, R. TAOUIL, and L. LAKHAL. 1999. Efficient mining of association rules using closed itemset lattices. Information Systems, **24**(1):25–46.

PAZZANI, M., S. MANI, and W. SHANKLE. 1997. Beyond concise and colorful: learning intelligible rules. *In* Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, pp. 235–238.

PEI, J., J. HAN, and L. LAKSHMANAN. 2001. Mining frequent item sets with convertible constraints. *In* Proceedings of the Seventeenth International Conference on Data Engineering, Heidelberg, Germany, pp. 433–442.

PIATETSKI-SHAPIRO, J. 1991. Discovery, analysis and presentation of strong rules. *In* Knowledge Discovery in Databases. AAAI Press, pp. 229–248. **Q6**

QUINLAN, R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

RIDDLE, P., R. SEGAL, and O. ETZIONI. 1994. Representation design and brute-force induction in a Boeing manufacturing domain. Applied Artificial Intelligence, **8:**125–147.

SCHLIMMER, J. 1993. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. *In* Proceedings of the Tenth International Machine Learnig Conference, Amherst, MA, pp. 284–290.

SHAH, D., L. LAKSHMANAN, K. RAMAMRITHAM, and S. SUDARSHAN. 1999. Interestingness and pruning of mined patterns. *In* Proceedings of the 1999 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD). Philadelphia.

SILBERSCHATZ, A., and A. TUZHILIN. 1996. What makes pattern interesting in knowledge discovery systems. IEEE Transactions on Knowledge and Data Eineering, **8**(6):970–974.

SILVERSTEIN, C., S. BRIN, and R. MOTWANI. 1998. Beyond market basket: generalizing association rules to dependence rules. Data Mining and Knowledge Discovery, **2**(1):39–68.

SKOWRON, A., and L. POLKOWSKI. 1997. Synthesis of decision systems from data tables. *In* Rough Sets and Data Mining. Analysis of Imprecise Data. Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 259–299.

SMYTH, P., and R. GOODMAN. 1992. An information theoretic approach to rule induction from databases. IEEE Transactions on Knowledge and Data Engineering, **4**(4):301–316.

SRIKANT, R., Q. VU, and R. AGRAWAL. 1997. Mining association rules with item constraints. *In* Proceedings of the Third International Conference on Knowledge Discovery and Data Mining. Newport Beach, CA, pp. 67–73.

TAOUIL, R., and Y. BASTIDE. 2001. Computing proper implications. *In* Proceedings of the ICCS-2001 International Workshop on Concept Lattices-Based Theory, Methods and Tools for Knowledge Discovery in Databases (CLKDD'01), Palo Alto, CA, pp. 49–61.

TOIVONEN, H., M. KLEMETTINEN, P. RONKAINEN, K. HAETOENEN, and H. MANNILA. 1995. Pruning and grouping of discovered association rules. *In* Working Notes of the MLnet Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases. Crete, Greece, pp. 47–52.

ULLMAN, J. 1988. Principles of Database and Knowledge-Base Systems, Vol. I. Computer Science Press. **Q7**

WANG, K., Y. HE, D. W. CHEUNG, and F. Y. L. CHIN. 2001. Mining confident rules without support requirement. *In* Proceedings of the Tenth International ACM Conference on Information and Knowledge Management (CIKM 2001). forthcoming.

WEBB, G. 1995. OPUS: An efficient admissible algorithm for unordered search. Journal of Artificial intelligence Research, **3:**431–465.

WILLE, R. 1992. Concept lattice and conceptual knowledge systems. Computer & Mathematics with Applications, **23**(6–9):493–515.

ZAKI, M. 2000. Generating non-redundant association rules. *In* Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, pp. 34–43.

ZIARKO, W., and N. SHAN. 1996. A method for computing all maximally general rules in attribute-value systems. Computational Intelligence, **12**(2):223–234.

QUERIES

Q1 Author: Please the change in year in Ref. Kivinen and Mannila.

Q2 Author: Please provide detail information about all the conference proceedings listed.

Q3 Author please provide place of publication.

Q4 Author: Please verify place of publication.

Q5 Author: Please provide page nos.

Q6 Author: Please provide place of publication.

Q7 Author: Please provide place of publication.