Rule-space Search for Knowledge-based Discovery

Authors: Foster Provost
Information Systems Department
Stern School of Business
New York University
44 W. 4th Street, Room 9-71
New York, NY 10012-1126
fprovost@stern.nyu.edu
212-998-0806
212-995-4228 (fax)

John M. Aronis and Bruce G. Buchanan
Computer Science Department, University of Pittsburgh
Pittsburgh, PA 15260
aronis@cs.pitt.edu, 412-624-9185
buchanan@cs.pitt.edu, 412-624-9183

Abstract:

Because the knowledge discovery process is ill-defined, iterative, and requires intense interaction, algorithm *flexibility* is crucial. In this paper, we present a straighforward, heuristic generate-and-test search algorithm for knowledge discovery. An analysis of the literature shows that this basic algorithm underlies many of the systems that have had practical success in data mining and knowledge discovery over the past twenty years. We argue that this search algorithm has persevered because it is flexible and well behaved as background knowledge is introduced in various forms—exactly what is needed to support the ill-defined knowledge discovery process. We illustrate this by showing how the basic algorithm can incorporate background knowledge implicitly, via a variety of "interestingness" criteria. We then show that the same basic algorithm applies to an extended representation including explicit background knowledge. We discuss the tradeoff between efficiency and expressiveness, and show how to speed up mining in the presence of explicit background knowledge. We conclude that this rule-space search algorithm is a good choice for supporting research into the rest of the knowledge discovery process, and argue that it sets the stage well for increased involvement of information systems researchers.

Contact author: Foster Provost

Some results contained in this paper appeared previously in two poster papers at KDD conferences:

- Aronis, J., F. Provost, and B. Buchanan, "Exploiting Background Knowledge in Automated Discovery." In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 355-358.
- Aronis, J. and F. Provost, "Increasing the Efficiency of Inductive Learning with Breadth-first Marker Propagation." In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), pp. 119-122.

Rule-space Search for Knowledge-based Discovery

Abstract:

Because the knowledge discovery process is ill-defined, iterative, and requires intense interaction, algorithm *flexibility* is crucial. In this paper, we present a straighforward, heuristic generate-and-test search algorithm for knowledge discovery. An analysis of the literature shows that this basic algorithm underlies many of the systems that have had practical success in data mining and knowledge discovery over the past twenty years. We argue that this search algorithm has persevered because it is flexible and well behaved as background knowledge is introduced in various forms—exactly what is needed to support the ill-defined knowledge discovery process. We illustrate this by showing how the basic algorithm can incorporate background knowledge implicitly, via a variety of "interestingness" criteria. We then show that the same basic algorithm applies to an extended representation including explicit background knowledge. We discuss the tradeoff between efficiency and expressiveness, and show how to speed up mining in the presence of explicit background knowledge. We conclude that this rule-space search algorithm is a good choice for supporting research into the rest of the knowledge discovery process, and argue that it sets the stage well for increased involvement of information systems researchers.

1. Introduction

The time has come for increased attention by information systems researchers to the field of Knowledge Discovery and Data Mining (KDD). Knowledge discovery systems have been used effectively in the sciences (Kohavi & Provost, 1998) and are seeing increasing use in business. However, knowledge discovery requires more than algorithmic advances. The knowledge discovery process is iterative, involving human interaction at several stages. Discovery of knowledge requires expert users to inject the process with background knowledge.

Many KDD researchers believe that the answer is to build new, more powerful data mining algorithms.[1] In contrast, we believe that simple search algorithms are sufficient both for effective discovery and for the study of the rest of the knowledge discovery process. In fact, as we argue in this paper, the flexibility afforded by simple algorithms facilitates the injection of domain knowledge in many ways.

There are two modes of use for knowledge discovery systems. First, they are used increasingly in the construction of information systems. We have used knowledge discovery methods for building systems for fraud detection, network diagnosis, medical diagnosis, and others; other practitioners have used them in many other application areas where relevant patterns (e.g., business rules) can be extracted from available data. Second, knowledge discovery systems are an important class of information system in their own right. Expert analysts use these systems to study business or scientific data, in order to discover previously unknown patterns. In this mode, a knowledge discovery system typically is used as an informed hypothesis generator to seed further analysis. For the empirical demonstrations in this paper, we have chosen data from two actual domains of knowledge discovery application: one from each mode. In the first, the goal is to discover rules that will be used for building a fraud detection system. In the second, knowledge discovery supports public health researchers directly. Details on the two domains can be found in Appendix B.

Knowledge discovery is an iterative process with intensive interaction required. The automated search for interesting patterns is a small but essential part of the cycle. For open-ended discovery tasks, those which do not immediately fit into a standard category such as classification or regression, much of the iterative process focuses on refining and utilizing domain knowledge. The prevailing approach to open-ended knowledge discovery is to use background knowledge to define and refine explicit criteria that define "interesting" patterns (Piatetsky-Shapiro, 1991; Silberschatz & Tuzhilin, 1996). Experimenting with different interestingness criteria is essential for such open-ended efforts to be successful, but it

1. Out of 248 papers submitted to the 2000 ACM SIGKDD International Conference on KDD, 173 were in the new algorithm category.

```
$new\_beam \leftarrow \{null\_pattern\}$
**while** $new\_beam \neq \emptyset$
    $beam \leftarrow new\_beam$;
    $new\_beam \leftarrow \emptyset$;
    **for** $p \in beam$
        $S \leftarrow$ all patterns obtained by the application of a single specialization operator to $p$; [SPECIALIZATION STEP]
        **for** $s \in S$
            **if** $s$ satisfies interestingness criteria, add $s$ to $interesting\_patterns$;
            **else if** $s$ satisfies pruning criteria, then discard $s$; [PRUNING STEP]
            **else** add $s$ to $new\_beam$;
        **end for**
    **end for**
**end while**
```

Figure 1: The heuristic generate-and-test rule-space search algorithm

can be difficult and time consuming. It is important that data mining algorithms be fast and flexible in order to facilitate such experimentation. A second approach to the incorporation of background knowledge, which has received little treatment relative to the others, is to represent the knowledge explicitly and declaratively. A significant research challenge is to develop algorithms that can mine data effectively and efficiently in the presence of such explicitly represented knowledge.

In this paper, we examine the use of the simple, heuristic generate-and-test (GAT) search algorithm shown in figure 1 for the task of finding rules that satisfy specified interestingness criteria. We discuss the algorithm in detail below; briefly, the algorithm starts with the empty pattern, and generates successively specialized patterns by applying specialization operators (the SPECIALIZATION STEP). The new patterns then are tested against the data: those that satisfy certain criteria are saved; those that meet certain pruning criteria are discarded (the PRUNING STEP), and the rest get specialized in the next iteration.

This basic algorithm underlies many discovery algorithms that have had practical success over the past twenty years. We are interested in why this algorithm has persevered, and indeed has been reinvented repeatedly in slightly different forms, for slightly different tasks. The contention of this paper is that the algorithm is flexible when it comes to the incorporation of background knowledge—a key to success in the iterative knowledge discovery process.

A convincing argument for an algorithm's flexibility is necessarily multi-faceted. Therefore, we present a variety of evidence. First we present a historical perspective on this algorithm's wide use. Then we delve into the basic algorithm, and show that many different interestingness criteria can be used to incorporate background knowledge, including those commonly used in other data mining work. We note that speed also is important for flexibility as knowledge changes during the discovery process, so (in Appendix A) we provide a brief complexity analysis showing that with a certain type of pruning, this algorithm's worst-case run-time complexity is linear in all the relevant parameters (few existing data mining algorithms have linear worst-case run-time complexity). We then move beyond simple numeric interestingness criteria for incorporating background knowledge. We discuss and demonstrate how the straightforward nature of the search provides the flexibility to introduce complex, explicitly represented background knowledge, without making fundamental changes to the algorithm. We then address the important issue of expressiveness versus efficiency, and introduce techniques that yield efficiency improvements for data mining with hierarchical background knowledge. These techniques improve upon the state of the art of using hierarchical background knowledge efficiently. Finally, because flexibility is most beneficial when algorithm behavior changes smoothly, in Appendix C we provide experimental evidence that this simple search behaves well as interestingness criteria are changed.

This paper makes several contributions to the research literature. First, we have written the paper in part to provide a tutorial overview of generate-and-test rule mining, the understanding of which is a prerequisite for information systems researchers to advance the state of the art. Second, we believe our central argument for why this algorithm has persevered (because of its flexibility) is novel, interesting, and well supported. Moreover, we hope that it will help information systems researchers, who want to study other portions of the knowledge discovery process, to choose an easy-to-code but effective data mining algorithm. Third, we present a novel technique for mining in the presence of relational background knowledge (breadth-first marker propagation) that advances the state of the art, and we demonstrate its effectiveness with empirical comparisons. Fourth, in Appendix C, we present additional new empirical results showing that the algorithm is well behaved as interestingness criteria are varied; flexibility is most advantageous if it results in smooth changes to algorithm behavior.

2. Generate-and-test rule mining: background

Generate-and-test rule mining belongs to the class of data mining algorithms known as "rule-learning" algorithms, which search for conditional sentences (rules) that are interesting based on user defined "interestingness" criteria. Data mining algorithms, generally, search through a space of patterns using a set of data for the evaluation of interestingness. Typically, patterns are matched against the data and various statistics are computed. Data records are often called *instances* or *examples*, and the set of instances used during the mining is called the *training set*. For example, consider a training database of records describing customer accounts, some of which have been defrauded. A rule-learning algorithm can be used to find transaction patterns indicative of fraud. For example, the following rule says that a cellular telephone call placed from cell-site number 123 in the evening is particularly likely to be fraudulent. In the next section we will describe in more detail the statistics typically computed to evaluate the interestingness of such a rule.

- $Cellsite = 123$ & $Evening \rightarrow Fraud$

Conditional sentences, or rules, are one of the most common representations used in data mining. Why mine rules? Why not some other form of knowledge? The conventional answer is, "because they are comprehensible to users." Many researchers and practitioners have found that rules are easy to understand for a wide variety of users, many of whom value comprehensibility even higher than predictive performance. For example, for knowledge discovery for building electronic commerce systems, Ansari et al. rank rules as the most comprehensible knowledge representation for their users (Ansari, Kohavi, Mason, & Zheng, 2000). Discussions of the comprehensibility advantages of rules can be found in works by Quinlan (Quinlan, 1993) and by Fürnkranz (Fürnkranz, 1999).

A second reason for choosing rules is their modularity. Not only does modularity contribute to comprehensibility, it also allows flexibility in how the rules are used. They can be formed into rule sets, along with rules learned with other algorithms as well as rules generated manually. Or they can be treated as individual "knowledge nuggets" for various purposes, such as seeding subsequent investigations.

It is important to mention at the outset that the data mining problem we consider, namely, discovering rules that satisfy explicit interestingness criteria, is different from the notion of rule learning typically considered in the field of Machine Learning. In Machine Learning, rule learning typically has been used once the problem has been formulated as a classification problem—the predictive assignment of instances into discrete classes. Specifically, the standard rule-learning problem is to find a small set of rules that gives high classification accuracy. This is an instance of the most common approach to the incorporation of background knowledge for data mining: place the burden of "knowledge engineering" on the problem formulation. It often is possible to pose the discovery problem as a task for which there are well-understood methods. With this approach, the necessary background knowledge is implicit, and its incorporation is ad hoc. Confusion may arise because some machine learning-style rule learners use generate-and-test (GAT) rule-space search combined with methods to select a small, accurate subset of the rules as a classifier. This is not the focus of this paper.

More generally, heuristic GAT rule-space search is an efficient procedure to produce a set of rules that satisfy certain criteria. These rules are then processed, manually or automatically, depending on the task. Classifier-building programs use the rules for one purpose. Human data miners may process the rules for another, perhaps manually interacting with domain experts and experimenting with different interestingness criteria (Provost & Aronis, 1996).

Human/computer collaborative discovery projects can use rule-space search as a basic tool (Lee, Buchanan, & Aronis, 1998). Rule-space search has even been used to automate other parts of the knowledge discovery process, for example for feature construction from complex data (Fawcett & Provost, 1997). These are just a few examples, taken from work in which we have been involved. They begin to illustrate the algorithm's flexibility.

The notion of rule learning as search was discussed explicitly early by Simon and Lea (Simon & Lea, 1973). Mitchell (Mitchell, 1982) provides a detailed overview of early uses of search for rule induction, culminating in the definition of *version spaces* (described below). Probably the first successful application of rule-space search for knowledge discovery was in the Meta-DENDRAL program (Buchanan, Feigenbaum, & Lederberg, 1971; Buchanan & Feigenbaum, 1978; Buchanan & Mitchell, 1978), which performed what would now be called "data mining" for scientific discovery in Chemistry. Meta-DENDRAL used chemistry-specific knowledge for pruning, and not only rediscovered known, published rules of mass spectrometry, but also made novel discoveries that were published in the chemistry literature.

Analyzing the literature describing a wide variety of successful knowledge discovery programs, one finds that many have as their basic, underlying procedures, minor variants of the algorithm of figure 1. In presenting the OPUS algorithms, Webb (Webb, 1995) provides a detailed specification of GAT algorithms' rule space and a description of admissible search pruning. He also provides clever methods for dynamic search-space reordering that yield impressive speedups. The ITRULE algorithm (Smyth & Goodman, 1992) conducts a GAT rule-space search with branch-and-bound pruning, using an information-theory-based measure of interestingness. The Brute programs (Riddle, Segal, & Etzioni, 1994; Segal & Etzioni, 1994) perform (highly optimized) GAT rule-space searches using Laplace-corrected confidence and complexity bounds to define interestingness. Weiss, et al., (Weiss, Galen, & Tadepalli, 1990) describe a GAT rule-learning search program (PVM) and several pruning heuristics for maximizing predictive value. The RL programs (Clearwater & Provost, 1990; Provost & Buchanan, 1995; Fawcett & Provost, 1997), perform GAT rule-space search, and have been used with a variety of interestingness and pruning criteria, such as (defined below) various forms of *confidence*, *support*, *complexity*, w-beam, and domain knowledge constraints.

It is important to note that the basic rule-space search applies whether or not one considers the patterns in the space to be rules. Both Rymon (Rymon, 1993) and Webb (Webb, 1995) show that this search space, more generally, is a structured enumeration of all possible unordered sets of elements from the description language. Rymon (Rymon, 1993) presents the SE-tree (Set-Enumeration-tree) algorithms, and relates rule-space search to decision tree learning. Schlimmer (Schlimmer, 1993) shows the application of the same basic algorithm to the mining of determinations, a different sort of pattern. Oates et al. (Oates, Schmill, & Cohen, 1999) show the application of the algorithm to various statistical dependency-mining tasks, including finding dependencies in multi-variate time series. Recently, Webb has demonstrated GAT search for association rules, and shows how it actually can outperform the standard association-rule algorithms (Webb, 2000). Meta-DENDRAL itself, more generally, considered the enumeration of complex graph structures using the same search-space structure, and below we consider the enumeration of patterns from a relational semantic network of background knowledge.

Much published work on GAT rule-space search stems from applications work. We believe that this is in part because GAT search is both flexible enough to be applicable in a wide variety of scenarios, and effective enough to succeed. We are most familiar with RL, which has been applied to problems including identification of human developmental toxicity (Gomez, Lee, & Mattison, 1993, 1994), trigger design for high-energy physics systems (Clearwater & Stern, 1991), sensitivity analysis in high-energy physics (Clearwater & Lee, 1993), building systems for diagnosing telecommunication networks (Danyluk & Provost, 1993), inducing rules for biological macro-molecule crystallization (Hennessy, Gopalakrishnan, Buchanan, Rosenberg, & Subramanian, 1994), analyzing large quantities of data on infant mortality (Provost & Aronis, 1996), predicting pneumonia outcome (Cooper & al., 1997), fraud detection (Fawcett & Provost, 1997), and predicting chemical carcinogenicity (Lee et al., 1998). A commonality that can be seen across the cited papers is that in each of these applications the simplicity of GAT search gave it the flexibility to be biased (favorably) by the interestingness criteria and background knowledge of the particular application.

One concern that arises when using data mining programs for knowledge discovery is that they produce too many "interesting" patterns. With GAT search, the absolute size of the resultant rule sets varies with the interestingness criteria. The rule sets may seem quite large when presentation to human experts is the next step in the knowledge discovery process. Very large rule sets can be reduced by strengthening the interestingness criteria (see Appendix C). However, it may be that presentation to humans is not the next stage in the KDD process. One may have a semantic filter of domain-knowledge constraints through which rules are to be passed (Lee et al., 1998), an automated (Fawcett & Provost, 1997) or semi-automated (Adomavicius & Tuzhilin, 2001) method of selecting out particularly interesting rules, or an automation of a larger part of the data mining cycle (Provost & Buchanan, 1995). The common criticism that "data mining programs produce too many uninteresting rules" is primarily a result of weak specifications of interestingness; domain-specific knowledge must be an input to the knowledge discovery process.[2] The simplicity of GAT search allows background knowledge to be introduced flexibly and processed efficiently.

3. Heuristic, rule-space search

We now describe in more detail the GAT search procedure. First we describe the algorithm, focusing on its flexibility to allow different interestingness/pruning criteria. Then we discuss specific criteria that are commonly used. In Appendix A we present a brief complexity

2. To our knowledge, the question remains open as to when to use domain-specific knowledge to aid search versus when to use it to filter the resulting rule set.

analysis of the algorithm, showing that it is a linear-time algorithm (and therefore it can be fast).

3.1 The basic, heuristic rule-space search algorithm

Let us now look in more detail at the general-to-specific heuristic GAT search algorithm shown above in figure 1. The space of possible conjunctive rules is structured as a tree rooted at the most general rule (the *null* rule), that with no conditions, which covers all data trivially. The search operators specialize rules. In its most basic form, each successive level of the search specializes the rules with a single additional condition (there are other ways of specializing rules, described below). For example, a GAT search for fraud detection rules for cellular telephony (Fawcett & Provost, 1997) starts with the null rule.

R0: $\{\} \rightarrow Fraud$

This rule then is specialized by applying operators that add various conjuncts.

R1: $Cellsite = 123 \rightarrow Fraud$

R2: $Cellsite = 456 \rightarrow Fraud$

R3: $Cellsite = 789 \rightarrow Fraud$

R4: $Evening \rightarrow Fraud$

R5: $Night \rightarrow Fraud$

R6: $Wee\text{-}hours \rightarrow Fraud$

The search proceeds, adding complexity to each hypothesized rule. For example, rule R1 is specialized with the remaining applicable operators:

R1.1: $Cellsite = 123 \;\&\; Evening \rightarrow Fraud$

R1.2: $Cellsite = 123 \;\&\; Night \rightarrow Fraud$

R1.3: $Cellsite = 123 \;\&\; Wee\text{-}hours \rightarrow Fraud$

Thus the search generates the space of all possible rules in a general-to-specific order. It gathers statistics on each as it proceeds, and uses the statistics, inter alia, to decide which rules to keep and which paths through the search space to prune. This algorithm is a *beam search*. Beam-search aficionados may note that we use the more general notion of beam search (Bisiani, 1987) which has regained popularity recently (Zhang, 1998). A beam search simply is a search in which "heuristic rules are used to discard nonpromising alternatives" (Bisiani, 1987).

The more restrictive notion of beam search, made popular possibly because of its description in certain text books, we will call a w-beam search. It works as follows. Consider a breadth-first search of the rule space. At each level of the search, each node is given a score based on a numeric interestingness criterion—which is intended to indicate the interestingness of this path in the search space, presumably because it is likely to lead eventually to an interesting rule. Instead of expanding all nodes, the w-beam search only expands the best w nodes (throwing away the rest). Thus, by varying w, a user can vary the search between a greedy search at one extreme ($w = 1$) and a complete search at the other ($w = \infty$, or no heuristic pruning). It should be noted that for large spaces, without other pruning criteria, the latter option may be impracticable. We consider w-beam pruning as one type of pruning in the more general beam search.

Much of the power of the rule-space search algorithm comes from search-space pruning. As an example, consider a problem where rules are not interesting if they cover fewer than a predetermined number of examples (variations of this heuristic appear in many data mining algorithms, possibly beginning with Meta-DENDRAL). If a given rule does not exceed the threshold, neither will any of its specializations, and these all can be eliminated from further consideration.

Admissible pruning of the rule space, such as in this example, is guaranteed never to eliminate interesting rules from consideration, and can yield tremendous search efficiencies

(Clearwater & Provost, 1990; Segal & Etzioni, 1994; Webb, 1995). On the other hand, *heuristic* pruning methods can result in even greater efficiency gains, but may discard potentially interesting rules.

It is important to acknowledge a difference of opinion on the issue of admissible pruning versus heuristic pruning. Consider the previous example pruning criterion. If an algorithm prunes a search path because the user specified that "rules covering fewer than 10% of the examples are not interesting," is it admissible pruning or heuristic pruning? There are convincing arguments for both answers, depending on whether one takes an algorithmic perspective or a user perspective. From the algorithmic perspective, the pruning seems admissible. Based on the input specification of interestingness, no rule in the pruned portion of the space is interesting. On the other hand, from the user perspective, the pruning may or may not be "admissible." Did the user choose 10% based on some fundamental restriction in the domain, or because it would speed up the algorithm, or because it would limit the number of rules found? It may be that from the user perspective the choice was "heuristic." For our purpose—analyzing the rule-space search algorithm—the distinction is irrelevant. The algorithm will prune the same whether a user chose the threshold based on semantic considerations or pragmatic ones.

Given a set of criteria (based on statistical, syntactic, or semantic factors) the set of rules that satisfy these criteria is known as the *version space* (Mitchell, 1982; Hirsh, 1989), one of the fundamental notions of machine learning. Mitchell describes how the set of most general rules in the version space (the *G-set*) and the set of most specific rules in the version space (the *S-set*) together are sufficient to describe the entire version space. Hirsh relaxes Mitchell's requirement of the rules' strict consistency with the training data. In fact, the notion of version spaces arose from early work on rule-space search (Buchanan & Mitchell, 1978).

Because the rules in the entire version space are logically entailed by the G-set, and any given rule in the G-set can subsume many different satisfactory specializations in the version space, we have found that in many domains the G-set contains the most interesting rules. For instance, if the rule A_1 &...& $A_n \rightarrow C$ is in the G-set, and it has specializations A_1 &...& A_n& $B_1 \rightarrow C$ ...A_1 &...& A_n& $B_k \rightarrow C$ in the version space, only the rule A_1 &...& $A_n \rightarrow C$ should be presented to the user. The rule-learning search algorithm in figure 1 stops with the G-set. Modifying the algorithm to select differently from the version space (e.g., to select the S-set) is straightforward.[3]

3.2 Explicitly represented interestingness criteria

In most cases, data miners incorporate background knowledge during the problem formulation stage of the knowledge discovery process. For example, in order to mine knowledge useful for constructing a fraud-detection system, a problem formulator may use time stamps on calls to create new variables describing the volume of calls during business hours, in the evening, and during the night, thereby incorporating background knowledge of legitimate and fraudulent calling patterns. Such *implicit* incorporation of domain knowledge is not our topic in this paper, but we note that it has two major drawbacks. First, often there are multiple related discovery tasks in a problem domain. If domain knowledge were explicitly represented, it could be transferred between tasks easily. Otherwise, problem engineering must be repeated for each subtask. Second, and more important, changing or augmenting

3. Yet another flexibility for specifying interestingness, which we will not treat in this paper.

the domain knowledge is burdensome—the problem must be reformulated, often requiring reprocessing large amounts of data.

After problem formulation, the second most common method for inserting problem-specific domain knowledge into the knowledge discovery process is to define explicit interestingness (and uninterestingness) criteria. The heuristic GAT search algorithm is flexible in the face of different discovery problems, because the basic algorithm is independent of the interestingness/pruning criteria.[4] Various criteria can be included to guide and restrict the search. It has been observed that many interestingness criteria can be expressed as functions of matching counts (Piatetsky-Shapiro, 1991; Silberschatz & Tuzhilin, 1996). In the PRUNING STAGE the algorithm in figure 1 works with criteria based on matching counts, based on pattern syntax, or based on semantic domain knowledge. A wide variety of interestingness criteria have been investigated; see for example the recent work of Padmanabhan and Tuzhilin on "unexpectedness" (Padmanabhan & Tuzhilin, 1999).

Recall that the data-mining algorithm searches through the space of rules, matching them against a set of instances, typically generated from historic data. Consider a rule $A_1 \mathbin{\&} \ldots \mathbin{\&} A_n \rightarrow C$. Its *confidence* is an estimate of the probability $p(C|A_1 \mathbin{\&} \ldots \mathbin{\&} A_n)$—the probability that the rule is correct if it fires. A simple frequency-based estimate is

$N(A_1 \mathbin{\&} \ldots \mathbin{\&} A_n \mathbin{\&} C)/N(A_1 \mathbin{\&} \ldots \mathbin{\&} A_n)$,

where $N(\cdot)$ is a count of the number of instances satisfying its argument. Since we are often interested in finding small rules, it is wise to use a statistical correction to this frequency-based estimate to reduce the number of spurious rules. Programs that mine conjunctive rules often use either Yates' correction (Quinlan, 1987) or the Laplace correction (Segal & Etzioni, 1994) to obtain a better estimate of confidence. A common way to use confidence

4. This is in contrast to most other data mining algorithms, whose design hinges on taking advantage of certain interestingness criteria, like accuracy or support.

as an interestingness criterion is to specify a confidence threshold below which rules are not interesting.

Support, also called *coverage*, is an estimate of $p(A_1 \ \& \ldots \& \ A_n \ \& \ C)$ (the probability that the pattern will occur). Often it is stated in absolute terms, "the rule covers 50 instances," rather than relative terms, "the rule covers 5% of the instances." A closely related notion of support, also called *positive coverage*, is the percentage of C that the rule covers, or $p(A_1 \ \& \ldots \& \ A_n|C)$. We limit our empirical analysis below to this latter variant of support, because we have found it to express users' interests better; the performance of the algorithm differs minimally among variants. A threshold on support is a very useful interestingness criterion, in part because it allows the fewer high-coverage rules to be presented to the user first, which facilitates the refinement of interestingness criteria for pruning or for sifting through the many more, lower-coverage rules.

In many applications, large sets of rules and rules with many conditions actually may detract from the process of discovering knowledge. Therefore, as ways of defining interestingness, we also investigate restrictions on the size, or *complexity*, of the rules and of the rule set.

To demonstrate, in Appendix C we experiment with three domain-independent criteria for search pruning: (i) prune all but the highest confidence rules at each level of the search, (ii) prune patterns with low support (below a threshold), and (iii) prune patterns with high complexity (above a threshold). We chose these three because we have found them useful, but more importantly because they have a long history of use in knowledge discovery and machine learning. Confidence and support have become popular recently as the basis of association rule algorithms (Agrawal, Imielinski, & Swami, 1993). The results show that each can provide powerful search-space pruning. The majority of the interesting patterns can be found quickly, and the behavior of the search algorithm changes smoothly and predictably as the criteria are varied.

4. Exploiting explicit, structured background knowledge

Some problem-specific knowledge can not be represented effectively as interestingness criteria. However, it is straightforward to augment GAT rule-space search with explicit structured background knowledge. To demonstrate, we consider background knowledge represented as inheritance networks with role links and a limited form of non-monotonic inheritance.[5] Augmenting the search with such structured knowledge extends the ability of the program to make discoveries by using the semantics of the features describing the data items. Because of the simplicity of GAT search, the only modification required of the basic algorithm is to add new search operators.

4.1 Representing background knowledge explicitly

Domain knowledge can take on a rich, structured form, including various taxonomies, categories, and relationships between concepts. To automate discovery using these forms of domain knowledge we must represent and reason about classes and relationships, and be able to bring the knowledge to bear on the discovery process. *Inheritance networks* are an efficient way to implement this kind of reasoning, because they can represent class structure and complex relational knowledge, yet can be navigated efficiently (Fahlman, 1979).

One of the applications we present below involves the analysis of data on exposures to poisonous plants (see Appendix B for more details). Figure 2 illustrates how some knowledge about plant families and their properties can be represented using standard inheritance

5. There are, of course, other types of explicitly represented background knowledge. For example, GAT rule-space search also has been shown to be able to incorporate explicit semantically based constraints on the form of the learned rules (Lee et al., 1998).

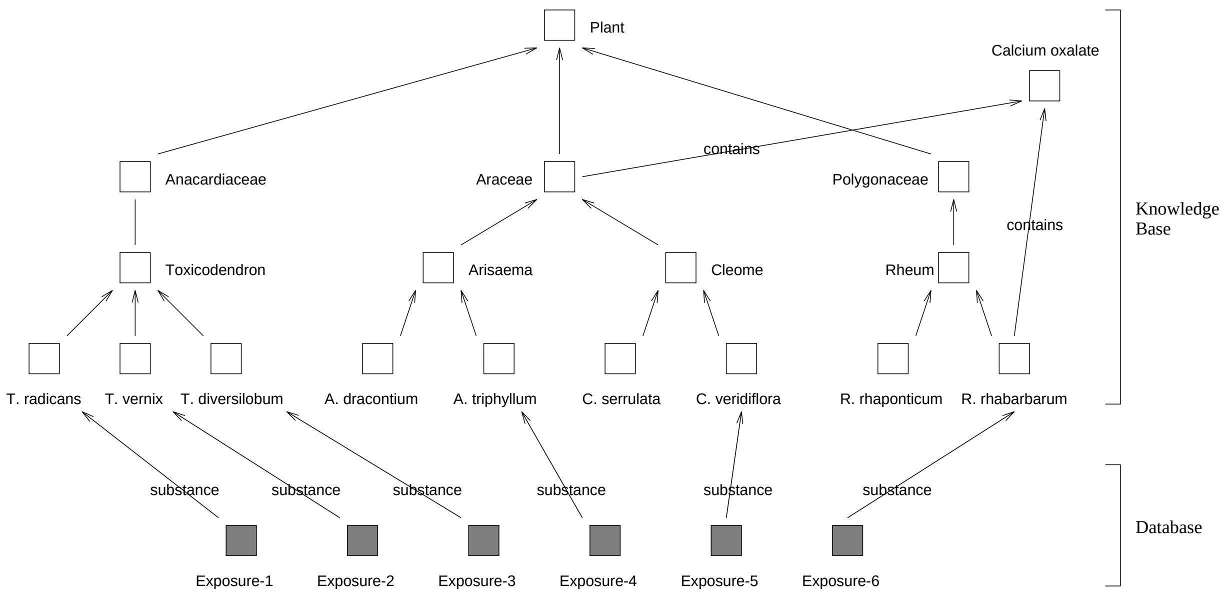


Figure 2: Linking Data to Botanical Knowledge.

network notation. A few records from the database of potentially toxic plant exposures and a small part of a botanical knowledge base are shown. Unlabeled arrows are *ISA links*, which can be interpreted as set inclusion. Thus, the link T. radicans $\rightarrow$ Toxicodendron means that every plant in the species T. radicans is also in the genus Toxicodendron. The link Toxicodendron $\rightarrow$ Anacardiaceae means that the genus Toxicodendron is a subset of the family Anacardiaceae. The *role link* Araceae $\xrightarrow{\text{contains}}$ Calcium-oxalate means that plants in the Araceae family contain calcium oxalate. Since calcium oxalate is present throughout the Araceae family we put the link at the family level, and let lower nodes *inherit* it. Calcium oxalate is specific to R. rhabarbarum (within its family), so the contains link is put directly on that species' node. This structured knowledge is not in the primary database; it was compiled from other databases.

Nodes and links can be used to form predicates. For instance, Toxicodendron(x) is true of everything in the genus Toxicodendron. Roles represent relations and can be multivalued; an exposure can have more than one substance link. We can use predicates to characterize sets of data items in terms of the knowledge base. For instance, Toxicodendron(substance(x)) characterizes the exposures 1-3. The more complicated predicate Calcium-oxalate(contains(substance(x))) characterizes exposures 4-5.

We note several advantages of this representation. First, inheritance networks provide a natural way to represent domain knowledge. For instance, our system allows a limited form of nonmonotonic inheritance to represent and reason about default and incomplete information. Second, since the representation does not duplicate domain knowledge for each database record there can be a huge gain in both time and space efficiency. Third, inheritance networks are sufficient to represent multi-table relational databases, in which much existing background information is stored, with role composition representing joins between tables. Finally, using inheritance networks connects automated discovery to work in knowledge representation—which then can be drawn upon to help with the human/computer interface.

4.2 An Illustrative Example

Consider the network in figure 3. Six examples of Datura exposures are shown, connected to a database of geographical and climate knowledge. Datura exposures normally occur in August-October; here we are interested in characterizing an anomalous subset of toxic exposures that occur in May. As above, the rule-space search starts with general predicates and specializes them. The user defines criteria with which the system will judge a discovery to be interesting. For this example, we use the simple criterion: an interesting rule is one that covers all of the May exposures, and none of the others.[6]

6. Of course, in practice we use more complex interestingness criteria, similar to those described above.

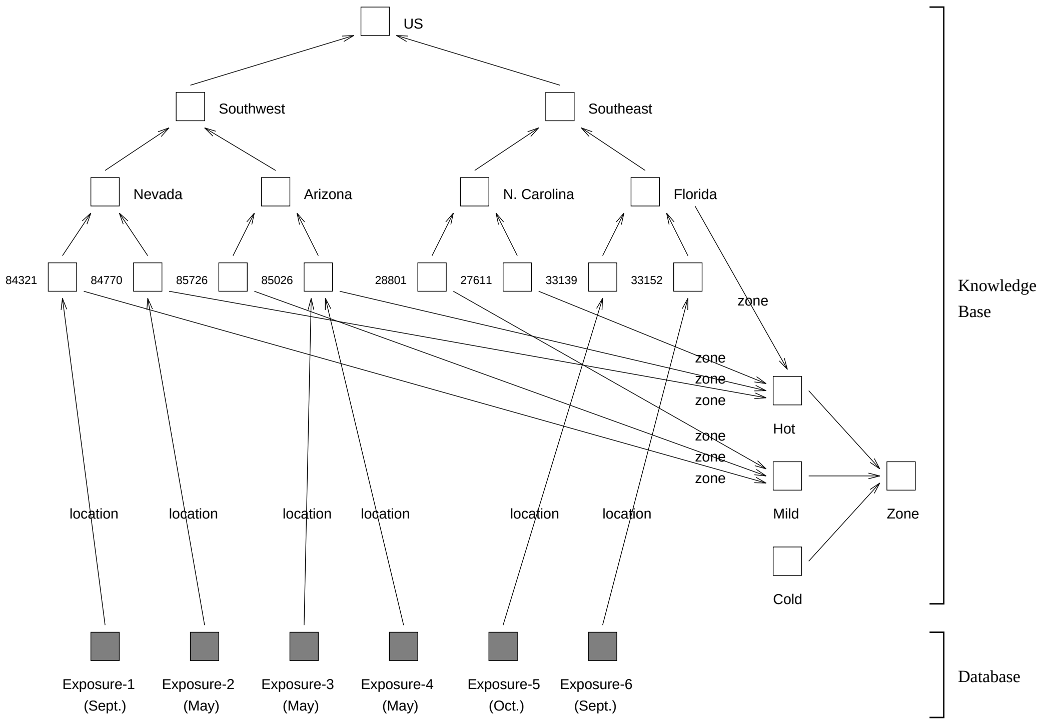


Figure 3: Characterizing May Datura Exposures.

The search starts by generating the general predicate US(location(x)). Since testing reveals that this is an overly general characterization, its *specializations* are generated from relationships in the network.

Southeast(location(x))

Southwest(location(x))

The first predicate fails to cover any members of the concept class in the database, so the system prunes it (and, implicitly, all of its specializations). The second correctly excludes some of the complement of the concept class, while still covering the incidents we are interested in categorizing, so the system retains it. However, this predicate still covers part of the complement. When the system continues the specialization of this path, it will examine:

Nevada(location(x))

$\mathsf{Arizona(location(x))}$

Neither of these have adequate coverage—they reject items in the concept class—so the system prunes them.

The hierarchy of locations provides no additional specializations, but the system can use the zone link. Additional specializations of the predicate already found, include:

$\mathsf{Southwest(location(x))}$ & $\mathsf{AnyZone(zone(location(x)))}$

Again, the additional predicate is vacuous, so it is specialized to create the three hypotheses:

$\mathsf{Southwest(location(x))}$ & $\mathsf{Hot(zone(location(x)))}$

$\mathsf{Southwest(location(x))}$ & $\mathsf{Mild(zone(location(x)))}$

$\mathsf{Southwest(location(x))}$ & $\mathsf{Cold(zone(location(x)))}$

Checking each of these verifies that the first characterizes the May incidents perfectly, so it is retained as a characterization that satisfies the system's criteria for an interesting discovery.

4.3 Rule-space search with background knowledge

The algorithm presented in figure 1 is sufficient for mining in this, more complex representation. Recall that at each stage of the search, the currently most promising rules are *specialized*. In the basic search described above, specialization took place by adding new conditions to the left-hand side of a rule. Another type of specialization operator restricts one of the rule's existing conditions. More specifically, we can instantiate the rule-space search with the following *specialization* operators:

1. *Add a Predicate.* The rule $\ldots \mathsf{P}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots \rightarrow \mathsf{C}(\mathsf{x})$ can be specialized to the rule $\ldots \mathsf{P}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots$ & $\mathsf{T}(\mathsf{x}) \rightarrow \mathsf{C}(\mathsf{x})$.

2. *Specialize a Predicate.* Given a rule of the form $\ldots \mathsf{P}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots \rightarrow \mathsf{C}(\mathsf{x})$ and ISA links $\mathsf{P_1} \rightarrow \mathsf{P}, \ldots, \mathsf{P_n} \rightarrow \mathsf{P}$ in the network, the rules $\ldots \mathsf{P_1}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots \rightarrow \mathsf{C}(\mathsf{x})$ through $\ldots \mathsf{P_n}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots \rightarrow \mathsf{C}(\mathsf{x})$ are specializations.

3. *Restrict a Role.* If the node P has f role values which are restricted to P', the rule $\ldots \mathsf{P}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots \rightarrow \mathsf{C}(\mathsf{x})$ specializes to $\ldots \mathsf{P}(\mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x}))$ & $\mathsf{P}'(\mathsf{f_{n+1}} \mathsf{f_n} \ldots \mathsf{f_1}(\mathsf{x})) \ldots \rightarrow \mathsf{C}(\mathsf{x})$.

The first operator—*Add a Predicate*—allows us to add additional predicates to a rule. This allows us to form rules with several conjuncts. The second operator—*Specialize a Predicate*—searches downward through a network identifying classes of the concept. It is important to note that in some cases there will be several different classifications of items. In botany, for example, there are different hierarchies based on different approaches to classification. The search algorithm explores all of these, specializing predicates according to each hierarchy and using the interestingness criteria to guide the search down paths that make meaningful distinctions in the current context. The third operator—*Restrict a Role*—selects a set of items based on relations to other parts of the knowledge base (as when the AnyZone predicate was added to the Southwest rule, above). Notice that the third operator is recursive, and we can restrict the predicate $\mathsf{P}(\mathsf{x})$ to $\mathsf{P}'(\mathsf{f}(\mathsf{x}))$, $\mathsf{P}''(\mathsf{gf}(\mathsf{x}))$, etc. Thus, we can talk about concepts such as "the average annual rainfall of the location of the exposure."

Membership in interesting classes may be determined by exceptional information, so it is important to incorporate and use some form of nonmonotonic information. We use a simple form of default inheritance that allows role values to be overridden by more specific information. Consider the diagram in figure 4. The items in the concept, marked by "+", are characterized by the predicate $\mathsf{Q_2(f(X))}$. This includes every item in $\mathsf{P_2}$, which all have f's that default to $\mathsf{Q_2}$, as well as $\mathsf{I_3}$, which has an exceptional f value.

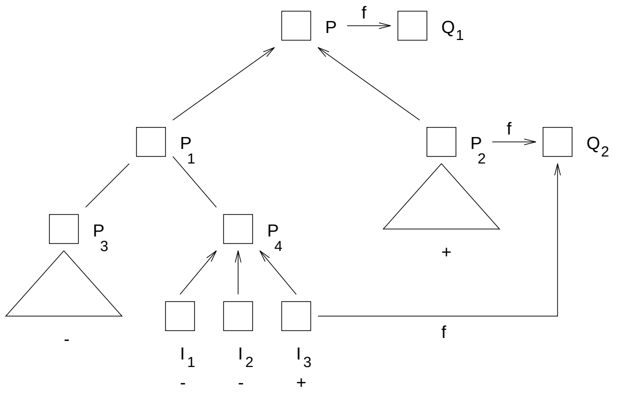


Figure 4: A Relation with an Exception.

We refer to the instantiation of the rule-space search, using the extended representation, as KBRL (for Knowledge-Based Rule Learner).

5. Expressiveness and efficiency

Our argument for the flexibility of rule-space search for "knowledge-based" discovery now must go deeper, because there is a critical tradeoff of efficiency for expressiveness. We have argued so far that the simplicity of the basic algorithm gives us the flexibility to increase the expressiveness of the representation—and still use the same basic mining algorithm. However, we must be careful not to gloss over the fact that the increase in expressiveness brings with it a decrease in efficiency. In the context of the iterative knowledge discovery process, for a mining system to be truly flexible it must be responsive. Obviously, a simple search of a very complex pattern space would take a long time. An easy counterargument is that empirical flexibility studies (see Appendix C) show that pruning criteria can be quite effective at productive search reduction. However, this argument seems too easy; it is not sufficient justification for us simply to ignore the issue of search efficiency in the context of our more expressive representation. In other work on learning with relational representations, even the fastest algorithms are several orders of magnitude slower than non-relational (using a propositional-logic representation) systems (Provost & Kolluri, 1999).

For the incorporation of relational background knowledge in KBRL, we purposely chose efficiency over expressiveness when it came to decisions about particularly expensive constructs. For example, we did not allow n-ary and recursive relational terms. However, even so, because of the generate-and-test nature of the rule-learning search algorithm, as the rule space becomes larger the efficiency of the algorithm may suffer in order to maintain efficacy. This is true not only for relational background knowledge, but even for standard propositional representations. As a simple example, a large data set may have thousands of values for a location field (e.g., zipcode). Unfortunately, the basic rule-learning search algorithm (and indeed most existing machine learning-style rule-learning algorithms) are prohibitively inefficient when it comes to large value sets. As described in the previous section, one may also want to group these specific locations based on knowledge, e.g., zipcode $\rightarrow$ city $\rightarrow$ state. Existing algorithms (including KBRL) also are relatively inefficient when it comes to even the most basic hierarchical background knowledge. Therefore, because we believe that speed is an essential component of flexibility, we now introduce an algorithmic technique that increases efficiency tremendously for these difficult problems.

5.1 Breadth-first marker propagation (BFMP)

In most implementations of GAT rule-space search, each pattern is "matched" against the database in order to gather the statistics needed to compute its interestingness score (if the pattern is not pruned for some other reason). This matching dominates the processing of GAT algorithms, especially for tasks with both large data sets and large description languages.

It is our claim that this central matching operation should be replaced with *breadth-first marker propagation (*BFMP*)* when description languages have either of two (particularly problematic) characteristics: large value sets or hierarchical background knowledge. As described in detail below, BFMP uses the data and the background knowledge to define a data structure that eliminates substantial inefficiencies of matching. For clarity, we will refer to the GAT rule-learning system that uses breadth-first marker propagation, instead of matching, as BFMP-RL.

Several prior approaches implement or discuss the use of hierarchical background knowledge for propositional learning algorithms. The RL rule learning system (Clearwater & Provost, 1990) extended the standard feature-vector-based rule-space search by allowing the possible values of attributes to be structured in ISA hierarchies. Núñez (Núñez, 1991) describes how ISA hierarchies can be used for decision-tree learning, and Quinlan (Quinlan, 1993) lists support of tree-structured attributes as a "desirable extension" to the well-known decision-tree learner, C4.5. Specifically, Quinlan describes a scheme for encoding taxonomic information into flat attribute-value tables that standard inductive learning programs can use. The state-of-the-art method also comes from decision-tree learning, as described by Almuallim, et al. (Almuallim, Akiba, & Kaneda, 1995); they use ISA hierarchies directly, and show their technique to be more efficient than the techniques suggested by Quinlan. We will compare BFMP to this last technique, which we call the *AAK-direct* approach.

We now describe breadth-first marker propagation, and compare its efficiency analytically with standard approaches. Later we give an empirical demonstration on very large versions of our two applications, augmented with hierarchical background knowledge.

5.2 BFMP technical details

As described in detail above, the fundamental operation of rule-space search is to specialize a hypothesis and to count the matches of the resulting specializations against the training database. These counts are used to compute the statistics-based interestingness measures. Matching also is the central operation of other rule-learning programs and of decision-tree learning programs. Breadth-first marker propagation replaces this generate-and-match method with a single operation to generate counts for all of a rule's specializations in one pass through the data. This approach is applicable to hierarchically structured attribute value sets, as well as to standard, flat attribute value sets.

Instead of viewing data items as vectors of attribute-value pairs to be matched against, consider them to be vectors of bidirectional pointers into the value space. Given that you want to specialize a k-conjunct hypothesis R (e.g., a rule or a decision-tree branch), breadth-first marker propagation generates counts of matches for all possible specializations as follows. The data structure VALUESET contains the set of attribute values with non-zero counts, which will be used as indices to retrieve the counts.

1. For each conjunct of R, mark the corresponding value with a *conjunct mark* (which we will denote &).

2. Following pointers, propagate these marks to the training instances, tallying how many marks accumulate on each instance.

3. For those instances with k conjunct marks, i.e., those that satisfy all k conjuncts of R, mark the instance with its class (e.g., $+$ or $-$).

4. Now, for each instance, and for each attribute, propagate the instance's class mark to the attribute value present in the instance. At each attribute value, keep a running

tally of the number of marks of each type. Add to VALUESET a pointer to each value marked.

5. For hierarchies of values, propagate *tallies of marks* in a breadth-first fashion from the leaves of the hierarchy to the root. Parent tallies are the sums of the corresponding child tallies. Add to VALUESET a pointer to each value visited.

Shape	**Color**	**Class**
Square	Red	$-$
Square	Blue	$+$
Triangle	Blue	$+$
Round	Blue	$-$

Figure 5: A Simple Data Mining Problem.

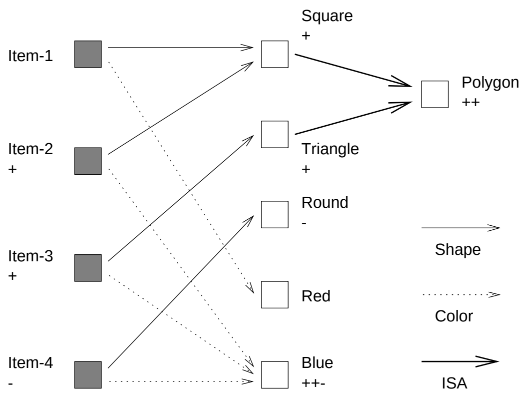


Figure 6: Network Representation of Simple Problem.

We will illustrate the algorithm on a simple problem. Consider the database given in figure 5, corresponding to the network of pointers shown in figure 6. Suppose the learner wants to specialize the hypothesis color=*blue* $\rightarrow$ +. We first mark blue with &, then move that marker down links onto items 2, 3, and 4. Since each of these items now has one & marker, corresponding to the single conjunct of the current hypothessis, we mark each item with its class (+ or $-$). Then, these markers are moved forward across links and tallied on each node. (This is the state the diagram illustrates.) Notice that the node Polygon accumulates two + markers and no − markers, indicating a perfect match of the positive examples. Note that we consider only purely hierarchical background knowledge (no role relations). Extending BFMP to the full KBRL representation is an open problem.

5.3 Complexity Analysis of Breadth-First Marker Propagation

Our purpose in introducing BFMP is to improve the efficiency of mining large data sets described by large description languages, in particular, those including hierarchical background knowledge. We first consider the complexity of hypothesis specialization in the case without ISA hierarchies. Assuming that there are e examples, a attributes, and (on average) v values for each attribute, even very efficient inductive algorithms based on matching require $O(e)$ matches for each of $O(av)$ potential specializations of each hypothesis for a time complexity of $O(eav)$, as described, for example, by Domingos (Domingos, 1996).

Now consider a learner that uses breadth-first marker propagation to replace matching. After walking through the examples once, each of the possible specializations will have class counts tallying all the examples that match it. The counts can be retrieved by walking through VALUESET, which (with no value hierarchies) can have no more than ae elements. The overall time complexity, $O(ae)$, is independent of the number of values. Thus, marker propagation should scale better for problems with large sets of values.

Now consider the case where attributes can have hierarchical, tree-structured values. As discussed above, the state of the art in efficient learning with value hierarchies comes from decision-tree learning (Almuallim et al., 1995). Almuallim et al. show the AAK-direct approach to be more efficient than other methods. It differs from our breadth-first marker propagation technique in that it walks each attribute value up the ISA hierarchy individually.

With ISA hierarchies of depth d, computing counts for e examples and a attributes takes time $O(ead)$.

Because breadth-first marker propagation combines counts at each level and propagates *tallies*, the process takes $O(ea + s)$ time, where s is the total number of values visited. It is clear that the set of values visited by breadth-first marker propagation is the same as the set values visited by the AAK-direct approach. However, breadth-first marker propagation visits each value only once. Thus, in the worst case, where no two examples share a value, and no two values share intermediate tree nodes as ancestors, marker propagation is equivalent to the AAK-direct approach. In any non-degenerate case, where there exists at least one place in the visited ISA hierarchy where its branching factor is greater than one, BFMP will be more efficient than the AAK-direct approach.

Moreover, for very large data sets, breadth-first marker propagation introduces efficiency benefits that are not apparent from the complexity analysis alone. Consider, again, the hypothesis specialization step. For a attributes and v values, matching methods typically make av passes through the set of e data items. Even more savvy programs, e.g., C4.5 (Quinlan, 1993), make a passes through the set of e examples. Breadth-first marker propagation performs only one pass through the data, performing a operations on each item. This introduces a huge savings in disk accesses if the data set does not fit in main memory. Mining disk-resident data is beyond the scope of this paper, but has been treated by other researchers (e.g., see work on learning decision trees from disk-resident data (Mehta, Agrawal, & Rissanen, 1996; Shafer, Agrawal, & Mehta, 1996)).

5.4 Results: marker propagation increases efficiency

To demonstrate the effect of breadth-first marker propagation, we replaced matching with BFMP in the rule-space search algorithm (BFMP-RL). As above, we use a w-beam search and a rule-complexity limit to restrict the search space of rules, and use confidence with the correction described by (Quinlan, 1987) to evaluate rule interestingness. The algorithm accepts a rule if its confidence is above a user-defined threshold.

Our first analytical result shows that, even without hierarchical background knowledge, breadth-first marker propagation is more efficient than conventional matching as the number of attribute values grows. To test this, we synthesized a sequence of problems consisting of 10,000 training examples with 10 attributes and an increasing number of values randomly assigned to these attributes (Aronis & Provost, 1997).

Figure 7 compares BFMP-RL's run time with that of rule-space search using standard matching (MATCHING-RL). Note that for these and the following experiments, the different systems performed identical searches and produced identical rule sets. As predicated analytically, the run time with breadth-first marker propagation remains nearly constant as the number of values increases, while the run time with matching increases linearly.

Our second analytical result predicts that breadth-first marker propagation will be more efficient than prior approaches when dealing with deep ISA hierarchies. Figure 8 shows the effect of increasing the depth of ISA hierarchies on breadth-first marker propagation compared with a version of RL using the AAK-direct approach, the current state-of-the-art method. Again, the empirical results support the analytical results: breadth-first marker propagation is strikingly more efficient for deep ISA hierarchies.

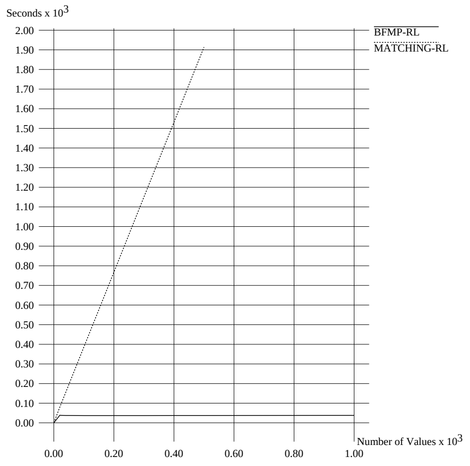


Figure 7: BFMP-RL vs. MATCHING-RL with Increasing Number of Values.

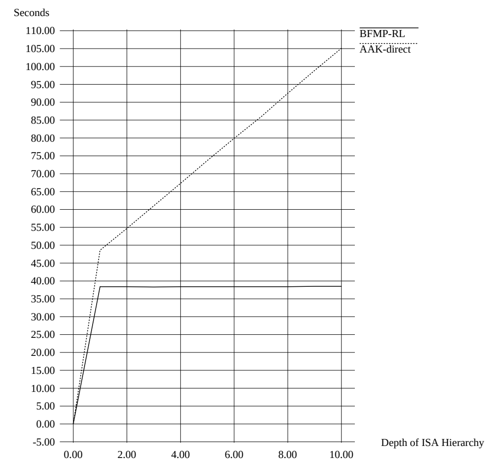


Figure 8: BFMP-RL versus AAK-direct with Increasing ISA Depth.

To demonstrate further that breadth-first marker propagation helps us to deal effectively with the efficiency/expressiveness tradeoff, we ran BFMP-RL on three real-world data sets (see Appendix B) with *one million* instances each, linked to large ISA hierarchies of background knowledge. The background knowledge for the fraud problem comprised a hierarchy of 1400 geographic locations of particular telephone numbers arranged in an ISA hierarchy of depth three. The background knowledge for the poisonous plant exposure data was a subset of the background knowledge described above (since here we do not deal with role relations), describing geographic regions (1014 distinct areas), climate types (55 types), and botanical classifications (2400 individual species, genera, and families) (Krenzelok, Jacobsen, & Aronis, 1995a). The infant mortality data were linked to background knowledge describing geographic hierarchies.

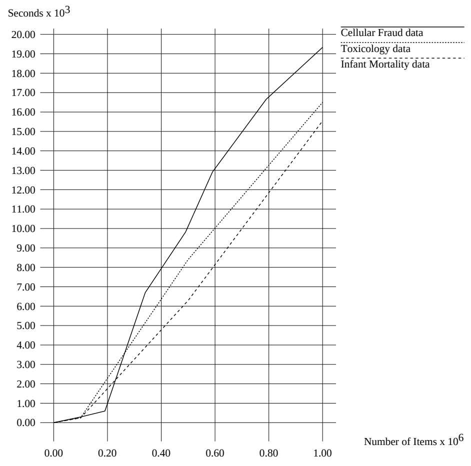


Figure 9: BFMP-RL with 10,000 to 1,000,000 Items.

Figure 9 shows the effect on BFMP-RL of increasing the number of data items for these three real-world data sets up to 1,000,000 items. BFMP-RL searched for rules with complexity of up to 5 conjuncts, using a *max_expanded* of 50. These interestingness criteria were chosen to create a difficult problem for standard rule-space search: using these criteria MATCHING-RL took nearly two hours to learn with only 100,000 examples and no ISA hierarchies. Furthermore, it is practically impossible to run MATCHING-RL on our workstation[7] with many more than 100,000 items due to memory-management thrashing. However, on 100,000 cellular fraud examples BFMP-RL performed a relatively thorough search of the rule space,

7. These tests were performed on a DECstation 5000 with 64Mbytes of memory.

defined by 23 attributes with 18,000 total values in an ISA hierarchy of depth 3, in less than five minutes.

In summary, introducing background knowledge raises questions about efficiency (Provost & Kolluri, 1999). We have shown that breadth-first marker propagation is an efficient alternative to existing approaches when data contain hierarchically structured values, and that even without such structures the technique is an efficient replacement for matching. The BFMP results are restricted to ISA-hierarchical background knowledge, and do not apply directly to knowledge with role links (e.g., as used in KBRL). In principle, the use of marker propagation provides a means to learn with more complex networks of background knowledge and with multitable databases. Generally, doing so is an open problem; some approaches have been described elsewhere (Aronis & Provost, 1994; Aronis, Kolluri, Provost, & Buchanan, 1997).

6. Conclusion

For many years we have been studying the use of data mining systems to help with real-world knowledge discovery. Although there are very many different data mining algorithms, again and again we have found a straightforward GAT rule-space search algorithm to be a better choice than other, more clever algorithms. By reviewing other published work we find that we are not alone. Especially in work stemming from real-world applications, many data mining programs have at their core this straightforward GAT search.

We have tried to provide support for our contention that the simplicity of this straightforward technique gives it flexibility. We believe that this is why it perseveres. Flexibility is essential in real-world discovery problems, because by the very nature of exploration and discovery, one does not know exactly what will be found. Furthermore, for "knowledge" discovery, it is critical that the algorithms can be biased by prior domain knowledge. We have shown that, indeed, GAT search can be so biased, in a variety of ways. Moreover, it is surprisingly efficient even with large data sets linked to explicit, structured background knowledge.

One of our duties as researchers on the border between the science and the application of knowledge discovery technologies is to identify important areas where more research is needed (Provost & Kohavi, 1998). We believe that the knowledge discovery process has not been paid enough attention by knowledge discovery researchers from computer science, who concentrate almost solely on algorithmic issues.

A potential problem facing research into non-algorithmic questions of knowledge discovery is "which of the many algorithms should we use" to study the rest of the process. We hope that we have argued convincingly that the rule-space search algorithm often is a "good enough" choice. Indeed, we believe that because every problem has its own idiosyncracies and unique knowledge to incorporate, a simple, flexible algorithm is the best choice. The GAT rule-space search algorithm can be implemented quickly and easily within a very wide variety of knowledge discovery applications, as attested to by the the wide variety of prior knowledge discovery work that, when observed closely, uses this simple algorithm.

As a final note, the knowledge discovery *process* lies at the interface between the human user and the discovery system, and does not lend itself to the neat technical results demanded by computer science research. Knowledge discovery and data mining systems are just beginning to become viable for use by non-researchers. However, as should be evident from our application descriptions, the two-way transfer of knowledge that characterizes the knowledge discovery process is ad hoc. We hope that more information systems researchers, who have considerable experience studying this type of problem, will see the knowledge discovery process as an exciting, wide-open area where many important research contributions can be made. We believe that because of its flexibility, the simple GAT search algorithm is a useful vehicle for such research.

7. Acknowledgements

This research was supported in part by National Science Foundation grants BES-9315428, IRI-9412549, and National Institutes of Health grant 2-P41-RR06009 (administered through the Pittsburgh Supercomputing Center).

References

Adomavicius, G., & Tuzhilin, A. (2001). Expert-driven validation of rule-based user models in personalization applications. *Data Mining and Knowledge Discovery*, *5*. To appear.

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207–216.

Almuallim, H., Akiba, Y., & Kaneda, S. (1995). On handling tree-structure attributes in decision tree learning. In *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann.

Ansari, S., Kohavi, R., Mason, L., & Zheng, Z. (2000). Integrating e-commerce and data mining: Architecture and challenges. Tech. rep., Blue Martini Software. Available: `http://http://xxx.lanl.gov/abs/cs.LG/0007026`.

Aronis, J., Kolluri, V., Provost, F., & Buchanan, B. (1997). The WoRLD: Knowledge discovery from multiple distributed databases.. In *Proceedings of Florida Artificial Intelligence Research Symposium (FLAIRS-97)*.

Aronis, J., & Provost, F. (1994). Efficiently constructing relational features from background knowledge for inductive machine learning.. In *Working Notes of the AAAI-94 Workshop on Knowledge Discovery in Databases* Seattle WA.

Aronis, J., & Provost, F. (1997). Increasing the efficiency of data mining algorithms with breadth-first marker propagation.. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* Newport Beach, CA.

Aronis, J. M., Provost, F. J., & Buchanan, B. G. (1996). Exploiting background knowledge in automated discovery. In Simoudis, E., Han, J., & Fayyad, U. (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 355–358. AAAI Press.

Bisiani, R. (1987). Beam search. In Shapiro, S. (Ed.), *Encyclopedia of Artificial Intelligence*, pp. 56–58. John Wiley and Sons.

Blake, C., Keogh, E., & Merz, C. (1998). UCI repository of machine learning databases.. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Buchanan, B., & Mitchell, T. (1978). Model-directed learning of production rules. In Waterman, D., & Hayes-Roth, F. (Eds.), *Pattern Directed Inference Systems.* Academic Press., New York, NY.

Buchanan, B. G., & Feigenbaum, E. A. (1978). DENDRAL and META-DENDRAL: their applications dimension. *Artificial Intelligence*, *11*, 5–24.

Buchanan, B. G., Feigenbaum, E. A., & Lederberg, J. (1971). A heuristic programming study of theory formation in science. In *Proceedings of the Second International Joint Conference on Artificial Intelligence*, pp. 40–50.

Clearwater, S., & Provost, F. (1990). RL4: A tool for knowledge-based induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, pp. 24–30. IEEE C.S.Press.

Clearwater, S., & Stern, E. (1991). A rule-learning program in high energy physics event classification. *Comp Physics Comm*, *67*, 159–182.

Clearwater, S. H., & Lee, Y. (1993). Use of a learning program for trigger sensitivity studies. In *Proceedings of the Third International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics*, pp. 207–212.

Cooper, G., & al. (1997). An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial Intelligence in Medicine*, *9*, 107–138.

Danyluk, A., & Provost, F. (1993). Small disjuncts in action: Learning to diagnose errors in the telephone network local loop. In Utgoff, P. (Ed.), *Machine Learning: Proceedings of the Tenth International Conference*, pp. 81–88. Morgan Kaufmann Publishers, Inc.

Domingos, P. (1996). Linear time rule induction. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 96–101 Menlo Park, CA. AAAI Press.

Fahlman, S. (1979). *NETL: A System for Representing and Using Real-World Knowledge.* Cambridge, MA: MIT Press.

Fawcett, T., & Provost, F. (1996). Combining data mining and machine learning for effective user profiling. In Simoudis, E., Han, J., & Fayyad, U. (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 8–13 Menlo Park, CA. AAAI Press.

Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, *1*(3), 291–316.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13*(1), 3–54.

Gomez, J., Lee, Y., & Mattison, D. R. (1993). Identification of developmental toxicants using a rule learning expert system. In *Programs and Abstracts: The Fourteenth Annual Meetings of the American College of Toxicology.*

Gomez, J., Lee, Y., & Mattison, D. R. (1994). RL: An innovative tool for predicting developmental toxicity. *Toxicologist*, *14*(295).

Haussler, D. (1988). Quantifiying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, *36*, 177–221.

Hennessy, D., Gopalakrishnan, V., Buchanan, B. G., Rosenberg, J. M., & Subramanian, D. (1994). Induction of rules for biological macromolecule crystallization. In Altman, R., Brutlag, D., Karp, P., Lathrop, R., & Searls, D. (Eds.), *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pp. 179–187. AAAI Press.

Hirsh, H. (1989). *Incremental version-space merging: A general framework for concept learning.* Ph.D. thesis, Stanford University.

Kohavi, R., & Provost, F. (1998). Special issue on applications and the knowledge discovery process. *Machine Learning*, *30*(2/3).

Krenzelok, E., Jacobsen, T., & Aronis, J. (1995a). Jimsonweed (datura stramonium) poisoning and abuse .. an analysis of 1,458 cases.. In *Proceedings of North American Congress of Clinical Toxicology* Rochester NY.

Krenzelok, E. P., Jacobsen, T. D., & Aronis, J. M. (1995b). Botanical scoundrels and emergency department visits. *Journal of Toxicology—Clinical Toxicology*, *33*(5), 543. Abstract of presentation given at the 1995 North American Congress of Clinical Toxicology Annual Meeting.

Krenzelok, E. P., Jacobsen, T. D., & Aronis, J. M. (1995c). Jimsonweed (datura stramonium) poisoning and abuse... an analysis of 1,458 cases. *Journal of Toxicology—Clinical Toxicology*, *33*(5), 500. Abstract of presentation given at the 1995 North American Congress of Clinical Toxicology Annual Meeting.

Krenzelok, E. P., Jacobsen, T. D., & Aronis, J. M. (1996). Hemlock ingestions: the most deadly plant exposures. *Journal of Toxicology—Clinical Toxicology*, *34*, 601. Abstract of presentation given at the 1996 North American Congress of Clinical Toxicology Annual Meeting.

Lee, Y., Buchanan, B. G., & Aronis, J. M. (1998). Knowledge-based learning in exploratory science: Learning rules to predict rodent carcinogenicity. *Machine Learning*, *30*(2/3), 217–240.

Mehta, M., Agrawal, R., & Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT)* Avignon, France.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*, 203–226.

Núñez, M. (1991). The use of background knowledge in decision tree induction. *Machine Learning*, *6*, 231–250.

Oates, T., Schmill, M. D., & Cohen, P. R. (1999). Efficient mining of statistical dependencies. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pp. 794–799.

Padmanabhan, B., & Tuzhilin, A. (1999). Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems*, *27*.

Piatetsky-Shapiro, G. (1991). Discovery, analysis, and presentation of strong rules. In Piatetsky-Shapiro, G., & Frawley, W. J. (Eds.), *Knowledge Discovery in Databases.* AAAI Press.

Provost, F., & Aronis, J. (1996). Scaling up inductive learning with massive parallelism. *Machine Learning*, *23*, 33–46.

Provost, F., & Buchanan, B. (1995). Inductive policy: The pragmatics of bias selection. *Machine Learning*, *20*, 35–61.

Provost, F., Jensen, D., & Oates, T. (1999). Efficient progressive sampling. In *Proceedings of the SIGKDD Fifth International Conference on Knowledge Discovery and Data Mining.*

Provost, F., & Kohavi, R. (1998). Guest editors' introduction: On applied research in machine learning. *Machine Learning*, *30*(2/3), 127–132.

Provost, F., & Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, *3*(2), 131–169.

Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 304–307. Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California.

Quinlan, J. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27*, 221–234.

Riddle, P., Segal, R., & Etzioni, O. (1994). Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence*, *8*, 125–147.

Rymon, R. (1993). An SE-tree based characterization of the induction problem. In *Proceedings of the Tenth International Conference on Machine Learning.* Morgan Kaufmann.

Schlimmer, J. C. (1993). Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In Utgoff, P. (Ed.), *Proceedings of the Tenth International Conference on Machine Learning*, pp. 284–290. San Mateo, CA: Morgan Kaufmann.

Segal, R., & Etzioni, O. (1994). Learning decision lists using homogeneous rules. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 619–625 Menlo Park, CA. AAAI Press.

Shafer, J., Agrawal, R., & Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining.. In *Proceedings of the Twenty-Second International Conference on Very Large Data Bases* Mumbai, India.

Silberschatz, A., & Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, *8*(6).

Simon, H., & Lea, G. (1973). Problem solving and rule induction: A unified view. In Gregg (Ed.), *Knowledge and Cognition*, pp. 105–127. Lawrence Erlbaum Associates, New Jersey.

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

Webb, G. (1995). OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, *3*, 383–417.

Webb, G. (2000). Efficient search for association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Weiss, S. M., Galen, R. S., & Tadepalli, P. V. (1990). Maximizing the predictive value of production rules. *Artificial Intelligence*, *45*, 47–71.

Zhang, W. (1998). Complete anytime beam search. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 425–430.

Appendix A: Complexity analysis of the basic algorithm

The size of the rule space, roughly, is f^d, where f is the number of specialization operators and d is the depth of the search. We will consider d to be equivalent to the maximum acceptable rule complexity,[8] which in the simple case is the number of possible conditions in a pattern. Although the search space grows exponentially in rule complexity, in practice

8. This makes the assumption that each operator adds one unit of complexity to a rule. Such is the case with most GAT rule learning; for example, each operator adds one new conjunct to the rule's antecedent. However, measuring rule complexity more generally, taking problem semantics and pragmatics into account in addition to syntax, is an open problem.

GAT rule-space search can be surprisingly fast (cf. the demonstrations below). If w-beam heuristic pruning is used, rule-space search has run time linear in the number of data items (as well as other parameters). Almost all other data mining algorithms have superlinear run-time complexity (an exception being CWS (Domingos, 1996)).

Let w be the maximum size of the set of "most interesting" nodes that will be saved at each level; let n be the number of data items. At each level of the search, (at most) w nodes are expanded with f operators to produce wf new candidate rules. To gather statistics, each of these candidates is matched against n data items for a total of wfn constant-time operations. Maintaining a (fixed-size) heap of the w best of the wf new rules requires $O(wf \log w)$ time. All this must be done for each of d levels of the search space, so the run-time complexity of rule-space search is $O(d(wfn + wf \log w))$. In all reasonable cases $\log w$ is $O(n)$ (cf. (Haussler, 1988)), so the complexity can be stated more simply as $O(dwfn)$.

Appendix B: Example applications

We consider two knowledge discovery domains, fraud detection and public health research. Specifically, we mine cellular telephone calling records for indicators of fraud. The calling records contain about two dozen features, such as calling-from location, calling-to location, duration of the call, time of day, etc. We linked these data to a hierarchy of geographic domain knowledge. For example, where a call is placed from or to is helpful for detecting fraud, but the ideal granularity can not be specified a priori. The ultimate goal is to discover knowledge that could be used in the construction of fraud detection systems. The "users" were members of the IS department of the cellular company. This application is described in detail by Fawcett and Provost (Fawcett & Provost, 1997).

We also consider data on poisonous plant exposures, in order to determine characteristics of bad outcomes (Aronis, Provost, & Buchanan, 1996). The data include demographic and symptom information about the victims, information about the plant substances, recommended actions, actual actions, and final outcome. These data were linked to background knowledge hierarchies describing geographic regions, climate types, and botanical classifications (see below). The ultimate goal here is to improve public policy regarding posion information and poison center operations. The "users" were public health researchers and toxicologists.

We also include another very large public health data set, from a different project. The infant mortality data set comprises one million U.S. Department of Health birth records linked with records of infant deaths. The task is to mine rules to categorize infant mortality and survival. Each record has about twenty fields, including demographic factors, birthweight, etc. This task is similar to the task of mining the plant exposure data: the knowledge discovery system is used to support the work of public health researchers. The immediate goal is to identify subgroups of the population with unusually high and unusually low infant mortality rates, in order to direct further research. The long-term goal of such work is to formulate policies that will reduce the nation's infant mortality rate (Provost & Aronis, 1996). These data also were linked to geographic hierarchies (Aronis & Provost, 1994).

We should note that in both the fraud and the public health domains, *we* were the interface between the system and the users. Knowledge discovery systems are only just beginning to reach a maturity level where non-KDD users can interact with them directly. We believe that there is much research to be done regarding this interaction.

Our purpose here is to use data from these applications to demonstrate our claims of flexibility. However, readers may be interested in learning more about the particular applications and the associated use of knowledge discovery systems. The fraud detection problem, along with the specific uses of knowledge discovery methods, is discussed in detail in technical conference and technical journal papers (Fawcett & Provost, 1996, 1997). Unfortunately, because of the sensitive nature of fraud detection, we can not give many details of the knowledge discovered or its use.

The toxicology problem (and the KBRL architecture) has been discussed briefly (Aronis et al., 1996). Results of the KBRL discovery work, and subsequent followup analysis of potentially toxic exposures, with our botanical toxocology "users" has been published in their domain literature (Krenzelok, Jacobsen, & Aronis, 1995b, 1995c, 1996). Peripheral results from this work, such as the demonstration that poinsettia and mistletoe ingestions are not associated with bad outcomes, have received relatively widespread attention (including a joke by Jay Leno on the Tonight Show).

Summarizing briefly, using the KBRL representation, the exposure data were linked to a knowledge base of geographic areas and their climates constructed from several sources on the World Wide Web. We also linked a knowledge base of botanical species, genera, and families adapted from a U. S. Department of Agriculture database as well as several small hierarchies of demographic factors, treatment patterns, etc.

One area of investigation in which KBRL took part was a study of exposures to Datura species (Datura, or jimsonweed, exposures often are due to its abuse as a hallucinogen). Many of the rules KBRL found refined the existing model of the seasonal spread of Datura exposures, but were not surprising to our botanical and toxicology collaborators. Rules showing that Datura exposures peak later in colder areas than in warm areas are a reflection of the fact that plants take longer to mature in colder climates. Other rules, such as a surprising degree of Datura abuse in some states, were unexpected but did not rely on the background knowledge and therefore could have been found by other discovery methods. However, by utilizing the network of background knowledge, a new rule was found that characterizes an unexpected set of May exposures in terms of basic enviromental conditions. This new rule was judged significant by our collaborators in botany and toxicology (Krenzelok et al., 1995a).

Appendix C: Empirical demonstrations

Our argument for the flexibility of the rule-space search algorithm has been analytical. Because of its simple structure, rule-space search allows a great deal of flexibility in defining what rules are to be considered interesting. However, flexibility can be a drawback if the system behaves in an unpredictable manner.

In this appendix, we provide experimental evidence that rule-space search is *well behaved* as the interestingness criteria are varied. This is particularly important when viewed in the context of the knowledge discovery process: the mining system may be run dozens—or even hundreds—of times as search and interestingness criteria are modified incrementally, so it is important that results vary in a *smooth* and *predictable* manner as interestingness criteria change.

7.1 Results: rule-space search is well behaved

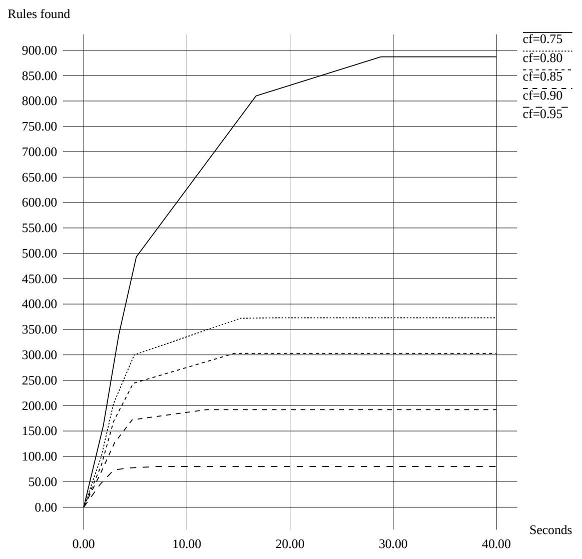
We examine the behavior of the basic rule-space search algorithm as four interestingness (pruning) criteria are varied: the confidence threshold, the support threshold, the threshold on the number of "most interesting" nodes kept, and the threshold on the number of conditions in a pattern. For simplicity we will call these *confidence*, *support*, *max_expanded*, and *max_conditions*. To make the presentation of results tractible, we treat these four criteria in two pairs. Our forays into different pairings all show similar results. We begin by investigating *confidence* and *max_expanded* for fixed values of *support* and *max_conditions*. We then fix *confidence* and *max_expanded*, and investigate *support* and *max_conditions*. The results of the study support our claim that the rule-space search behaves well. For the results presented in this section, the fraud data comprise 10,000 items described by 21 attributes; the poison data comprise 10,000 items with 20 attributes.

To begin, let us state our first two hypotheses regarding the expected behavior of this straightforward heuristic search algorithm.

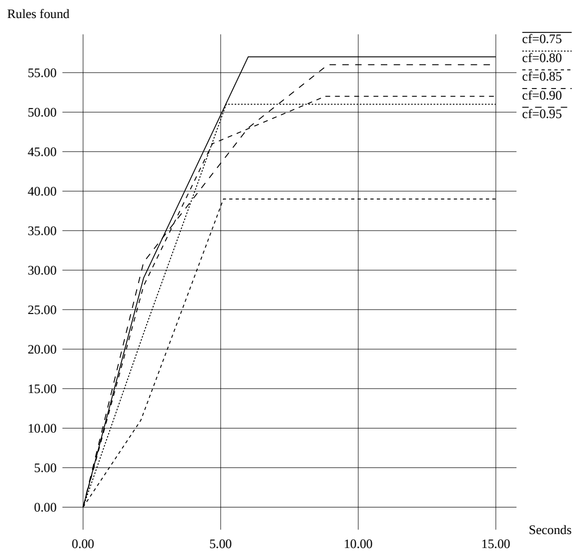
> Hypothesis 1: *As the required confidence of rules is raised or lowered, the number of rules found increases or decreases accordingly.*
>
> Hypothesis 2: *When max_expanded is small, the search will find a small number of interesting rules quickly; as max_expanded is increased the number of interesting rules found grows and the search time increases, but with diminishing returns.*

To investigate these hypotheses, we ran rule-space search on our two discovery tasks. We fixed *support* and *max_conditions* to what seemed reasonable levels to achieve tractible yet non-trivial search (below we show the effects of varying these criteria). We used the Laplace correction (Segal & Etzioni, 1994) to compute confidence, and varied *confidence* by 0.05 between 0.75 and 0.95. At each level, the search pruned all but the *max_expanded* patterns with the highest confidence (that were not yet satisfactory). We varied *max_expanded* from 10 to 10^4.

Standard learning curves, used for analyzing the performance of classifiers (Provost, Jensen, & Oates, 1999), plot the accuracy of the classifier as the number of data items increases. The typical behavior is that the accuracy rises quickly early, and as the amount



(a)



(b)

Figure 10: Number of rules found vs. time for different confidence (cf) thresholds: (a) Poison data, and (b) Fraud data.

of data increases the rate of improvement slows. Eventually, a plateau is reached and adding more data does not improve the accuracy of the learned models. Often most of the advantage is obtained surprisingly early in the process.

For our heuristic search of the rule space, we desire a similar rapid increase in the percentage of the set of interesting rules found (rather than in accuracy), as the amount of search is increased (by weakening the pruning). In particular, the rule-learning search algorithm would be particularly attractive if it could find most of the interesting rules with a small amount of search, with the rate of increase of the rule set tapering off as more search is performed. In figure 10 the number of rules found is plotted against the run time, creating a "rule-learning curve" analogous to a standard learning curve.

The results largely support Hypothesis 1. On the Poison data, as *confidence* is decreased the number of rules found increases. The results from the Fraud data are surprising and deserve further explanation. Note that, after the curves level off, decreasing *confidence* first *decreases* the number of rules, and then increases it. This is due to a subtlety of the interestingness criteria that at first may seem counterintuitive. Specifically, the set of interesting rules with confidence greater than 0.95 is not necessarily a subset of the set of interesting rules with confidence greater than 0.90. This is due to the fact that we are not considering all the rules in the version space to be interesting, but only the G-set—the (syntactically) most general rules. As the interestingness criterion becomes more liberal, the G-set may include a new member that subsumes several prior members (and thus, these prior members are no longer members of the G-set). As a specific example, consider a possible set of interesting fraud rules found with *confidence* = 0.95:

R1: $Cellsite = 123$ & $Night \rightarrow Fraud$ (confidence = 0.97)

R2: $Cellsite = 123$ & $Evening \rightarrow Fraud$ (confidence = 0.96)

R3: $Cellsite = 123$ & $Wee\text{-}hours \rightarrow Fraud$ (confidence = 0.99)

compared to this single rule found with *confidence* = 0.90:

R4: $Cellsite = 123 \ \rightarrow Fraud$ (confidence = 0.92)

Rule R4 subsumes the other three, and since we only want the G-set, the other three are not interesting. Thus, even though the experiments do not support Hypothesis 1 completely, the seemingly anomolous behavior on the Fraud data can be explained and actually is desirable.

The results support Hypothesis 2. In particular, the curves are steep early on and then level off. This indicates that the heuristic pruning mechanism does indeed focus the search on the most interesting patterns. For very strong pruning criteria, a little more search can give a lot more rules. However, the benefit of additional search decreases quickly.[9] Thus, the rule-learning curves do show behavior analogous to standard learning curves.

We now move on to investigate the other pair of criteria. For real-world problems the space of possible patterns may be very large (Webb, 1995). If the estimation of interestingness involves computing statistics over large training data sets, searching for all interesting patterns may become intractible without strong pruning criteria.[10]

Our second set of hypotheses for investigation follows.

Hypothesis 3: *Large values for support cause rule-space search to be very fast; search time increases smoothly as support is lowered.*

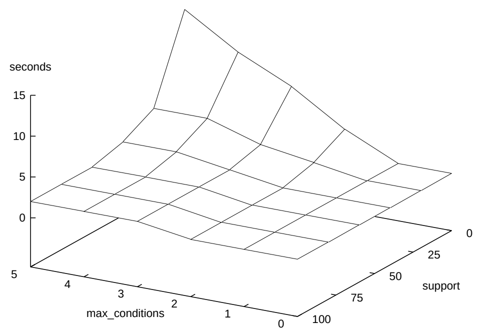
Hypothesis 4: *Small values for max_conditions cause rule-space search to be very fast; search time increases smoothly as max_conditions is increased.*

To investigate these hypotheses, we ran rule-space search on our two data sets, varying *support* and *max_conditions*, having fixed *confidence* = 0.90 and *max_expanded* = 50000 (the latter placing the searches well out on the plateaus shown in figure 10). In figure 11 search time is plotted against *support* and *max_conditions*.

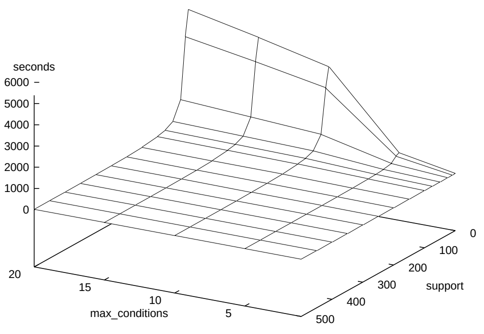
These results support Hypothesis 3 and Hypothesis 4. Specifically, for any given level of one criterion, as the other is weakened the search time increases gracefully. When both criteria are at their weakest levels, as expected, the run time is very large (since *max_expanded* is also very weak, there is relatively little search pruning at all).

9. We also performed experiments on data sets from the UCI repository of data sets (Blake, Keogh, & Merz, 1998) for which exhaustive search of the rule space is possible, and have verified that using a large *max_expanded* gives a very good approximation to an exhaustive search.

10. It should be noted that data mining algorithms can take advantage of particular, strong pruning criteria to achieve impressive efficiencies. As discussed further below, probably the most impressive case are the frequent itemset algorithms, which take advantage of support-based pruning (among others), and which are described in detail elsewhere (Agrawal et al., 1993).



(a)



(b)

Figure 11: Time vs. *support* and *max_conditions* for (a) Poison data, (b) Fraud data.

More significantly, these results show that a moderate pruning setting for either criterion *alone* controls the computational explosion well. For example, being able to constrain the search to rules with relatively few conjuncts eliminates the need for a support threshold. Being able to constrain the search to rules that cover a large number of examples eliminates the need to constrain rule complexity.

This general principle, that there are many alternatives for controlling the computational explosion, also is evident with other forms of pruning. For example, in the previous results we saw that *max_expanded* alone can control the computational explosion. The straightforward nature of rule-space search affords the data miner flexibility to incorporate different domain knowledge in different situations. For example, the utility of *support* for pruning is impressive. However, in some cases, there may not be a semantically based choice for bounding support. Fortunately, with rule-space search there is no need for one. In the worst case, the experimenter can fall back on using *max_expanded* to provide a fast, productive search. During the iterative knowledge discovery process, it is typical that interestingness criteria are refined. Pragmatically based support thresholds often emerge as the data miner and domain experts examine rules found in previous iterations. Later in the process, these can be used for pruning, and the other criteria can be relaxed (if desired).

We chose confidence and support as example interestingness criteria for our flexibility study in part because we have found them to be useful in our work, but also because they have become standard measures; frequent-itemset-based algorithms (e.g., for learning association rules (Agrawal et al., 1993)) find all the rules that satisfy support and confidence thresholds, and have been shown to be very efficient under some conditions. Our purpose is not to compete with these algorithms, but to offer an alternative. Rule-space search has some fundamental advantages. We argue mainly for its flexibility; for example, as we have shown above, support thresholds are not necessary, if other pruning heuristics take up the slack. Confidence could be replaced as the driving interestingness measure with a different criterion, such as the chi-squared statistic for measuring interestingness with respect to a rule's ability to separate probability distributions. Also, it should not be overlooked that, unlike many other algorithms, the rule-space search algorithm (with w-beam pruning) is a worst-case linear-time algorithm, and gives the user the option of very fast mining when little is known about the problem and immediate, preliminary results are desired. It would be interesting to experiment with "complete, anytime beam search" (Zhang, 1998), which iteratively weakens the pruning criteria, seamlessly scaling the algorithm from a fast heuristic search to a complete search of the space.