Current State of Data Mining

Darren T. Drewry, Lin Gu, A. Benjamin Hocking, Kyoung-Don Kang,
Robert C. Schutt, III, Christopher M. Taylor, John L. Pfaltz
Dept. of Computer Science, Univ. of Virginia
{dtd3r,lg6e,hocking,kk7v,rcs5y,cmt5n,jlp}@virginia.edu

CS-2002-15
9 May 2002

Abstract

This report is a compendium of results uncovered in CS 851, spring semester 2002. Our goal is to provide direction to other graduate students who want to explore the fascinating field of "data mining".

Be aware that this represents the topics and the papers that we found most interesting. We made no attempt to be either "fair" or to be "comprehensive" as in a *Computing Reviews* article. Instead we tried to pick just one or two references in each of our interest areas that were both relatively recent and, in our opinion, worth reading. Their bibliographies can direct a reader to other sources.

1 Overview

The term "data mining" means many different things to different people. Originally, the term was attached to blind, undirected search for probabilistic associations within large data sets. Rakesh Agrawal, together with his co-authors [3, 32] are generally considered to be the originators of "data mining". Since then, the term has come to be attached to "unsupervised machine learning", and the derivation of almost any statistic from a data set. For example, Dzeroski [10] says that

> "The output of a data mining algorithm is typically a pattern or a set of patterns that are valid in the given data. A pattern is defined as a statement (expression) in a given language, that describes (relationships among) the facts in a subset of the given data, and is in some sense simpler than the enumeration of all the facts in the subset. Different classes of pattern languages are considered in data mining: they depend on the data mining task at hand. Typical representations are equations; classification and regression trees; and association, classification, and regression rules. A given data mining algorithm will typically have a built-in class of patterns that it considers: the particular language of patterns considered will depend on the given data (the attributes and their values).
>
> Many data mining algorithms come from the fields of machine learning and statistics. A common view in machine learning is that machine learning algorithms perform a search (typically heuristic) through a space of hypotheses (patterns) that explain (are valid in) the data at hand. Similarly, we can view data mining algorithms as searching, exhaustively or heuristically, a space of patterns in order to find interesting patterns that are valid in the given data."

In this report, we lean to the former definition. We seek associations of the form A $\Rightarrow$ B which can be interpreted "if an object has the set A of attributes then it will often/always have those of B as well". Our concentration will be on those methods that depend on the *apriori* algorithm, or *apriori*-like algorithms to find such associations.

Moreover, we will assume that we are mining a binary relation such as Figure 1. In this figure we have labeled the rows as O (for object) and the columns as A (for attribute). But, there is no standard nomenclature for this. Applications arising from "market basket" analysis usually label the rows as T (for transaction, or tuple) and the columns as I (for item) because the row denotes the items involved in that transaction.

Many relations of interest are not binary. Many are numeric. If one is mining medical data, it is expected that many of the attributes will be numeric. Mechanisms for mining numeric data are of great interest; but as we will see in Section 5 they are at best rudimentary.

The main threads that we will be considering in this report are all related to discovering the existence of associations, or implications, in a data table such as Figure 1. They are:

A

O	a	b	c	d	e	f	g	h	i
1	×	×					×		
2	×	×					×	×	
3	×	×	×				×	×	
4	×		×				×	×	×
5	×	×		×		×			
6	×	×	×	×		×			
7	×		×	×	×				
8	×		×	×		×			

Figure 1: A binary relation, R, between two sets O and A.

- **Association Rule Mining:** Finding rules, such as $A \Rightarrow B$, where A probably implies B, is a fundamental goal of "data mining". The *apriori* procedure is central to this approach and is discussed in Section 2.

- **Categorization Rules:** Often the consequent B of the implication $A \Rightarrow B$ is a categorical property such as "the patient survives at least 1 year" or "the mushroom is poisonous". When the consequent B is fixed to be a few distinguished attribute combinations, special techniques such as those discussed in Section 2.5 can be used.

- **Neural Network Methods:** Neural networks create relationships between inputs that are not always easy to comprehend. Neural network methods investigate ways to train neural networks on data sets as well as ways to understand what the neural networks have learned. These methods are examined in Section 3.

- **Rough Set Theory:** Rough set data analysis uses partitions and the corresponding equivalence relations to describe the granularity of the data. It differs from most other data analysis methods in that it provides a strict mathematical model and relies mostly on the data itself to perform various data mining tasks. Section 4 discusses this topic.

- **Quantitative Rule Mining:** The development of data mining algorithms that incorporate numeric data is extremely important as a large portion of real-world data sets contain quantitative attributes. Section 5 presents two attempts to create association rules composed entirely, or in part, of quantitative data.

- **Deterministic Implications:** Deterministic data mining looks for rules that are *always* true as opposed to associative rule mining which looks for rules $A \Rightarrow B$ that are often true. These techniques are discussed in Section 6.

2 Association Rule Mining

Association rule mining is a fundamental technique in data mining. In some real-life applications, *e.g.*, market basket analysis in K-Mart, data sets can be too large for manual analysis, and potentially valuable relations among attributes may not be evident at a glance. An association rule mining algorithm can find frequent patterns (sets of database attributes) in a given data set and generate association rules among database attributes. For example, some items can be frequently sold together, *e.g.*, milk and cereal. Such items can be displayed together to improve the convenience of shopping. Association rule mining is generally applicable to those applications in which the data set is large and it is useful to find frequent patterns and their associations, *e.g.*, market basket analysis, medical research, and intrusion detection. In this section, we give an overview of association rule mining and describe the well-known *apriori* algorithm [1, 3] as follows.

2.1 Frequent Sets

It is the first step in (probabilistic) association rule mining to find all frequent sets. Let D be a database, which consists of tuples or records, called (sales) *transactions* in data mining. Let I be a set of all database attributes, often called *items* in data mining literature. A set of items is called an *itemset*. Given a threshold, called *min_sup* (minimum support), an itemset is called *frequent* if its frequency is equal to or greater than *min_sup*. Hence, the main objective of this step is finding all frequent itemsets whose support, *i.e.*, frequency, is higher than the threshold, *min_sup*. For example, consider Figure 1. Assume that $min\_sup = 0.375$, that is, a frequent itemset should appear at least three times in the database shown in Figure 1. In the given database, following itemsets are frequent: a (count = 8), b (5), c (5), d (4), f (3), g (4), h (3), ab (5), ac (5), ad (4), af (3), ag (4), ah (3), bg (3), cd (3), df (3), gh (3), abg (3), acd (3), adf (3), agh (3).

This step is computationally expensive, especially when *min_sup* is low. To find all frequent itemsets, it might be necessary to consider all possible combinations of items. Also, multiple database scans might be necessary. Therefore, the size of the problem space is exponential in terms of the number of items. Multiple database scans and the exponential explosion of search space can significantly affect the performance of data mining. The notion of support is also debatable. More detailed discussions will be given after describing the *apriori* algorithm.

2.2 Frequent Associations

Once all frequent sets are identified, association rules can be found. An association rule $A \Rightarrow B$ holds in a given database with a certain confidence γ $(0 \leq \gamma \leq 1)$ if:

$$\frac{sup(A \cup B)}{sup(A)} = \gamma \tag{1}$$

For example, in Figure 1 $conf(abg) = sup(abg)/sup(b) = (3/8)/(4/8) = 0.75$. Hence, the association rule $b \Rightarrow ag$ is confident with confidence 0.75 in the given data set. Note that this step is only applied to frequent itemsets. Infrequent itemsets are generally considered not important in probabilistic rule mining approaches. To filter out rules with low confidence values, a threshold called min_conf, is used in probabilistic rule mining. A rule is called confident if its confidence Eq. (1) is above min_conf. In this way, every rule with certain probabilistic relation between the antecedent and consequent can be found.

2.3 Basic *apriori* Search

The *apriori* algorithm consists of two main steps: (1) find all frequent itemsets and (2) generate confident rules as discussed before. Given a frequent itemset I, find all rules $A \Rightarrow I - A$ $(A \subset I)$ whose confidence is higher than min_conf.

The intuition behind the frequent set search is very simple: all $(k-1)$-itemsets (itemsets of length $k-1$) in a k-itemset must be frequent for the corresponding k-itemset to be frequent. Therefore, if an $(k-1)$-itemset is infrequent, no k-itemset containing it can be frequent, and need not be considered. A high level description of the frequent set search is as follows.

1. Find F[1] = {Frequent 1-itemsets}.
2. for (k=2; F[k-1] $\neq \emptyset$; k++)
   (a) Generate length k candidates from F[k-1].
   (b) Scan the database to count the frequency of candidates.
   (c) Insert each candidate of length k into F[k], if its frequency $\geq min\_sup$.

For example, F[1] $= \{a, b, c, d, f, g, h\}$ in Figure 1. From this, we can generate candidates of length 2: C[2] $= \{ab, ac, ad, af, ag, ah, bc, bd, bf, bg, bh, cd, cf, cg, ch, df, dg, dh, fg, fh, gh\}$. Observe than no 2-itemset containing e could be frequent. From the database scanning, we get F[2] $= \{ab, ac, ad, af, ag, ah, bg, cd, df, gh\}$. This iterates until there is no new frequent itemset of a larger cardinality.

Note that this algorithm has several possible performance drawbacks. Multiple database scanning can be very expensive in terms of response time. Also, too many infrequent candidates can be generated. This can cost both storage and time to save candidates and scan the database for counting purposes. (To alleviate the performance problem, a hash tree is used to save the storage and time for counting [1]. However, this approach does not directly handle the possible disadvantages discussed before.)

To address this problem, Han *etal.* [16] presented a novel approach, in which frequent sets can be found without generating candidates. In their performance evaluation, they showed a significant performance (*i.e.*, response time) improvement compared to the *apriori* algorithm.

2.4 Discussions

Probabilistic data mining approaches have several drawbacks. First, frequent itemsets may not mean anything. For example, many transactions in a census database may share a same gender, female or male. Given dense data sets such as a census database, it is highly likely that *apriori-like* algorithm may suffer poor performance. More importantly, it might generate many "frequent" but "unimportant" itemsets and rules. As a result, it may be almost impossible for human operators to interpret the association rules. We have observed that the *apriori* algorithm generates over three million rules for the MUSHROOM database described in section 9.2, which is generally considered dense, when $min\_sup = 0.01$ and $min\_conf = 0.5$. (Given $min\_sup = 0.2$ and $min\_conf = 0.9$, it generates over 340,000 rules.) Furthermore, it may not capture an important but infrequent rule. It was shown in [27] that *apriori* algorithm misses some rules detecting poisonous mushrooms. See Section 9.2. In summary, probabilistic rule mining may not represent/capture *causal* relations between antecedents and consequents. This limits the applicability of probabilistic rule mining approaches to some data sets, *e.g.*, MUSHROOM database and system call sequences for the detection of malicious attacks. For this reason, many alternative approaches have been considered recently. More detailed discussions are provided in the following sections.

2.5 Constraining Association Rules

Frequent set data mining techniques generate as their output a set of association rules that represent probabilistic relationships between data items. When using frequent set data mining for knowledge discovery, a problem that quickly arises is that the number of association rules generated overwhelm the researcher. Ordonez *etal.* [20] present a case study of using data mining to discover interesting association rules in medical data and demonstrate how the basic *apriori* algorithm can be extended to constrain the generated association rules. In this study they use frequent set data mining to predict a patient's risk of heart disease.

When data mining is used in a knowledge discovery endeavor, the purpose is to identify interesting relationships between data items. The problem with the typical support / confidence technique for constraining the generated rules is that by setting very low support and confidence thresholds many association rules are generated that are not significant. Ordonez *etal*. present a technique for constraining association rules in the following ways:

- **Support / Confidence:** The standard statistical notion of support and confidence, rules that do not meet the minimum support and confidence levels are not generated.

- **Antecedent / Consequent:** In many applications of data mining it is known which items will be in either the antecedent or consequent of the association rule (where rules are of the form $antecedent(s) \Rightarrow consequent(s)$). This is often true for medical data mining where some of the items are a set of symptoms or diagnostic tests (*antecedents*) and the researcher wants to determine how these symptoms relate to outcomes or disease processes (*consequents*). By constraining the algorithm to only generate rules that have well formed antecedent/consequent pairs the search space is greatly reduced and fewer but more meaningful association rules are generated. For example, when trying to discover rules that predict heart disease, rules that relate smoking to patient weight are less relevant than rules that relate smoking to coronary artery blockage (a measure of heart disease).

- **Grouping:** Attributes can be members of a group. Association rules are group constrained by the following: the antecedent or consequent of an association rule can have no more then 1 member of a group. In a shopping cart example if the following are antecedent constrained items and are also members of a group "whiskeys": *rye whiskey, scotch whiskey, and bourbon*, possible generated rules include *rye whiskey* $\Rightarrow$ *purchase* $\geq$ \$100, *bourbon* $\Rightarrow$ *purchase* $\geq$ \$100, and *scotch whiskey* $\Rightarrow$ *purchase* $\geq$ \$100. An example of a rule that will be disallowed by the group constraining criteria is *rye whiskey AND bourbon* $\Rightarrow$ *purchase* $\geq$ \$100 because *rye whiskey* and *bourbon* are members of the same group and both contained in the antecedent.

Even with antecedent/consequent constraining and group constraining the number of rules generated in the data mining process can be large. The following are the results presented by Ordonez *etal*. [20]:

1. 25 possible attributes, 655 database transactions (patient records).
2. 4 of the attributes were consequent constrained, 21 were antecedent constrained.
3. 19 of the 21 antecedent constrained attributes were put into 3 different groups. Group 1 had 9 members, Group 2 had 5 members, and group 3 had 4 members.

The data mining produced 2987 rules using support of 0.2% and confidence of 70%. Ordonez *etal*. claim that using only support/confidence frequent set data mining, the same support and confidence generated over 1 million rules from the same data set. The 2987 rules were reduced to 850 by a domain expert identifying rules that are "counter-intuitive to medical knowledge." This aspect of the result is very interesting but unfortunately the authors do not elaborate on these rules or present examples to illustrate why they are arising. The 850 rules were divided into categories where similar antecedents implied similar consequents and subsequently were sorted by support and confidence for analysis by a domain expert.

From the results presented by Ordonez *etal.* it is clear that constraining association rules is an important part of any data mining investigation. With basic support/confidence frequent set data mining far too many rules are generated when low support/confidence levels are desired. The drawback of association rule constraining is that the investigator must be able to constrain their search prior to running the algorithm since all possible results will not be enumerated.

3 Neural Network Methods

Several techniques use neural networks to mine associations from data sets. Two prominent techniques include training multi-layer perceptrons on samples of the data and evolving Kohonen feature maps on the data. As opposed to most other data mining techniques, these neural network data mining techniques are not constrained to input attributes that are categorized or even discrete. Removing this constraint means that neural networks can be used on a variety of inputs including boolean, real, complex, or even graphical.

3.1 Multi-Layer Perceptrons

When using multi-layer perceptrons, there are typically two main steps required. The first step is to train the multi-layer perceptron. Typically, a fraction of the database to be mined is presented multiple times to the network with back-propagation used to adjust the weights of the network. A crucial part of this step is deciding on the topology of the network to be trained. If the network contains too many hidden neurons, then the neural network will be over-fitted to the training data and have an insufficient ability to generalize. If the network contains too few hidden neurons, then the neural network may be unable to learn the problem. And although

> "It is known that a feed-forward network with one hidden layer can approximate any continuous function of the inputs, and a network with two hidden layers can approximate any function at all."

it is possible that "the number of units in each layer may grow exponentially with the number of inputs" [?]. The choice of an appropriate topology will be influenced by the nature of the inputs. In training a perceptron on the MUSHROOM database containing only categorical inputs, no hidden neurons were required to learn a perfect classification of poisonous or edible. Such a result implies the MUSHROOM database is linearly separable. This should be expected given the high dimensionality of the input and the sparseness of the database. Further results from our work with neural networks and the MUSHROOM database are discussed later in this section.

The second step in using multi-layer perceptrons to mine association rules is to analyze the trained neural network. This step is optional if one is only interested in predicting the correct class of an object but is necessary if one wants to generate concrete association rules. There are several methods for analyzing the neural network. One method is to reverse engineer the network to determine exactly which combinations of inputs contribute to the desired categorizations. Reverse engineering a neural network is intractable in the worst-case scenario. However, we were able to reverse engineer a neural network trained on the MUSHROOM database of Section 9.2 in a short amount of time. Another method that we studied involves the use of genetic algorithms.

Genetic algorithms reduce the complex problem of finding the best association rules into a simpler problem of finding good association rules. Two decisions involved when using genetic algorithms is to determine the genome encoding and the fitness function. The choice for fitness function might affect how you choose to encode the genome. The choice for genome encoding will affect your choice of fitness function.

As an example of some of the challenges, in most applications to data mining one will want the ability to represent a selection from a category as "don't care". This representation will require a special encoding in the genome. If the possible values for a category are represented by more than one gene in the genome, a possible mistake is to encode the "don't care" value identically to all other values in the category. This choice will cause some values to be more likely to be transitioned to from the "don't care" value than other values, giving them a greater probability of existing in the population than would naturally arise from their actual fitness. One possible solution is to add additional "don't care" values to the category such that every value has an equal probability of arising from a "don't care" value. Although this solution can alleviate this particular problem, a more complicated problem arises when all the other possible transitions are considered. For example, if one particular value in the category contributes significantly to the fitness of a genome, values that have a greater probability of being transitioned to from the fit value will be represented in the population disproportionally to their actual contribution to fitness. We did not find a satisfactory solution to this problem.

Our experience with the MUSHROOM database led to several practical discoveries in creating both the neural net and in determining the proper encoding and fitness function in the genetic algorithm. When creating the neural net we discovered that no hidden neurons are required to encode the MUSHROOM database. In general, we expect that most databases will benefit from no more than 5 hidden neurons, but more work is required in this area to be certain. Another discovery is that without sufficient redundancy in the network, the output can be fully determined by a small subset of possible inputs when those inputs perform exceptionally well at predicting the correct classification. This subset's ability to perfectly classify becomes problematic because it prevents the discovery of valid association rules dealing with those values that have a smaller impact on correct classification.

The biggest problem we encountered with applying genetic algorithms to the MUSHROOM database was identifying a proper choice of fitness function. With our initial choice of a fitness function, every single rule that was generated had no matching items in the database. This should be expected when one considers that there are approximately 1.22×10^{14} possible combinations of the attributes in the MUSHROOM database, but only 8,124 entries. Since we're encouraging "don't care" values the actual probability will be higher than the 6.7×10^{-11} suggested by these numbers, but the probability of reaching attributes that are in the MUSHROOM database is still so small as to make the odds indistinguishable from zero, unless a more directed approach is taken. One possible directed approach is to have the fitness function use the support as one of the criteria of fitness.

The best results we obtained were when we had the fitness function be a multiple of the logarithm of the support, and took the additional step of removing all members of the population that had zero support in the database.

3.2 Kohonen Maps

Another technique for mining association rules from databases is to use Kohonen (or self-organizing) maps [7]. Kohonen maps are self-supervised neural networks that impose a distance metric on the neurons in the map. This metric can be embedded in any n-dimensional space that is desired, although 2-dimensional spaces are the most frequently used because of the ease with which they can be navigated. An advantage of Kohonen maps is that they do not require any a priori knowledge of categorizations of input or classifications of output. Unlike other types of self-supervised neural networks, the learning rule in Kohonen maps is influenced not just by the inputs and outputs of the individual neurons, but also by the outputs of the neurons nearest neighbors. This rule ensures that clusters of neurons form that all tend to fire for similar inputs. These clusters therefore define the classifications that are recognized by the maps, where the center of the cluster is said to define the prototype of the classification. One technique for automatically identifying these clusters is given in [7].

4 Rough Set Theory

Rough set theory provides a neat methodology to formalize and calculate the results for data mining problems.

In the early 1980s Z. Pawlak, cooperating with other researchers, developed the rough set data analysis (RSDA) [23]. As suggested by its main motto – "let the data speak for themselves", RSDA tries to discern internal characteristics of a data set, such as categorization, dependency, and association rules, without invoking external metrics and judgment.

This approach has produced some useful results, but it also showed some drawbacks or insufficiency when dealing with complex data mining problems. More specifically, it is successful in the following respects:

1. it provides a clean mathematical model for data mining tasks; and
2. it provides a group of well-defined algorithms to solve the defined problems.

However, it is insufficient in the following aspects:

3. it lacks a systematic mechanism to incorporate domain specific knowledge or human interaction into the model, algorithm and the solution; and
4. it does not reduce the computation complexity of the tasks.

The rest of this section is organized as follows. The subsection 4.1 introduces the context of this methodology, some basic concepts, and some historic notes. Those three topics are combined together because the ideas in the concepts can be better understood when we know the context and history where they were developed. The subsection 4.2 deals with the problem modeling and the corresponding algorithms for data mining tasks. The subsection 4.3 analyzes the pros and cons of rough set theory in the context of data mining and in comparison with other theoretical and practical approaches.

4.1 Problem context, basic concepts and history

In the data mining area, the applications of rough set theory include attribute reduction, rule generation and prediction. With the motto "Let the data speak for themselves", RSDA distinguishes from other approaches with its "non-invasive" property. It uses only the information from the data without other model assumptions.

In order to discern and extract information from the data, RSDA utilizes a special *partition* relation of the data. Partitions are used to express properties and equivalence relations, with different granularity achieved by different partitions.

If we denote the universal set as U, a partition is a family P of non-empty disjoint subsets of U such that the union of all the subsets in P is exactly U. Observe that a partition can be regarded as a subdivision of the data set into categories; hence this work can be related to that of the preceding Section 2.5.

Given any subset X of U, with the concept of partition, rough set theory further defines the lower approximation (positive region) $\underline{X}$, upper approximation (possible region) $\bar{X}$ and area of certainty/uncertainty. Consequently, $U - \bar{X}$ is the impossible region.

A rough subset is defined as a pair $< \underline{X}, \bar{X} >$. These simple concepts form the basis of a theory that has demonstrated a variety of relations with other theoretical results.The collection of all rough subsets of U is shown to form a regular double Stone algebra, which serves as a model for three-valued Lukasiewicz logic [9]. In Shafer's evidence theory, the beliefs and plausibility can be expressed by lower approximation and upper approximation, respectively [29, 30].

With those concepts defined and calculi formed, RSDA further provides a group of definitions that are directly related with data mining problems. It defines an "information system" as follows $I =< U, \Omega, \{V_q\}, \{f_q\} > q \in \Omega$ where U is a finite set of objects, Ω is a finite set of attributes, $\{V_q\}$ is a class of set of attribute values, corresponding to each attribute q in Ω, and f_q is a set of functions that map a value from U to a value in V_q, corresponding to each q in Ω.

This theoretical definition differs from practical application in that f_q is a well-defined function so that multiple values or missing values are not allowed. [21] presented an extension and generalization of the information system.

4.2 Modeling and algorithms

Both the data and the problems involved in data mining can be described in an information system.

One of the most important and fruitful area of data mining is rule generation. Rough sets can be used to reduce attributes and generate rules in a structural way.

By comparing equivalence relations, RSDA is able to reduce the number of attributes. An equivalence relation with regard to a set $Q \subseteq \Omega$ is defined as $x\ \theta_Q\ y$ iff $\forall x : f_q(x) = f_q(y)$. Intuitively, $x\ \theta_Q\ y$ if and only if x and y cannot be separated by observing their values on Q.

With the equivalence relations thus defined for arbitrary Q, RSDA describes the internal structure of a data set with terms that characterize the relations between various subsets in the data set. First of all, in a data set, a group of attributes, say, Q, may dominate the values of another group of attributes, say, P. In this situation we say P is dependent on Q, written as $Q \Rightarrow P$. Formally, $Q \Rightarrow P$ iff $\theta_Q \subseteq \theta_P$

A set $P \subseteq Q \subseteq \Omega$ is called a "reduct" of Q when $\theta_P = \theta_Q$ and P is the smallest set that has this property. Reducts can be used to generate deterministic rules. The reducts for a given Q may not be unique. So we have a family of reducts for any given Q. We define the intersection of all those reducts in this family the "core" of Q. The core of a set can be calculated by a matrix of discernibility, which also appears in other contexts that involves equivalence related applications. All the elements in the core of Q are said to be "indispensable" for Q. On the other hand, if an attribute q does not belong to any reducts in the family, that attribute is said to be "redundant" for Q.

To generate association rules, we just notice that a fine partition often preserve all the information of a coarse partition if the latter, represented as a relation, includes most of the elements of the former. Actually this is also the idea behind the notion of dependence aforementioned.

Formally, and following the notation in [9], we can define the association rules as follows.

Denote θ_Q as $\{X_1, X_2, \ldots, X_s\}$, and θ_d as $\{Y_1, Y_2, \ldots, Y_t\}$ (d can be a single attribute or a composite attribute which is a combination of a number of attributes). For each X_i in θ_Q, we define a set $M_i = \{Y_j | X_j \cup Y_j \neq \emptyset\}$. Denote M_i as $\{Y_{j_1}, \ldots, Y_{j_m}\}$. We know that

$$x \in X_i \Rightarrow x \in Y_{j_1} \text{ or } x \in Y_{j_2} \text{ or } x \in Y_{j_m}$$

Supposing Q has n attributes, denoted as $q_{k_1}, \ldots, q_{k_n}$, each class X_i corresponds to a feature vector $< a_1, a_2, \ldots, a_n >$ where $a_i \in V_{q_{k_i}} (1 \leq i \leq n)$. Similarly, each class $Y_{j_l} (1 \leq l \leq m)$ corresponds to an attribute value $b_{j_l} \in V_d$. This information can be used to generate the following rule

$$f_{q_{k_1}}(x) = a_1 \wedge \ldots \wedge f_{q_{k_n}}(x) \Rightarrow f_d(x) = b_{j_1} \vee \ldots \vee f_d(x) = b_{j_m}$$

If $m = 1$, the rule becomes *deterministic*. Otherwise, it is *indeterministic*.

Another contribution of RSDA is that it defined a strict metric for the power of a classification system – approximation functions. An approximation function evaluates the classification power for a given class X by calculating the ratio of objects which can be correctly classified. In RSDA, partitions are used to describe, discern and classify the classes. So its approximation function uses the partition based *upper* and *lower approximation* to calculate the approximation functions. Formally, RSDA defines

$\gamma_\theta(X) = |\underline{X}| + |\underline{-X}|/|U|$

to express the percentage of elements in U that can be correctly classified by the relation θ, and defines

$\gamma_\theta(x) = |\underline{X}|/|\bar{X}|(X \neq \emptyset)$

to express the portion of all the possible elements that could belong to X that θ is able to discern as being definitely in X. Here an underlying uniform distribution of primitive elements is assumed. This is called the *principle of indifference*.

With the approximation function defined, RSDA gives formulae to calculate the quality of association rules, the impact of an attribute q on the power of a classification system, and the significance of rules. Those all gracefully model the corresponding problems and interests in the field of data mining. With those definition, and given the fact that all the notions are based on partition, one can easily construct algorithms to solve those problems. However, as described in subsection 4.3, the complexity of those algorithms could be formidable.

The evaluation metrics described above enables RSDA to evaluate a model for a given application. Specifically, we can evaluate the equivalent, or almost nearly expressive, models to find reducts of a model. With large data set, and algorithms that can often grow to be intractable, the reduction of models have practical significance in data mining. However, we need to be careful to trade off between the cost of model reduction and the performance gain.

4.3 Discussion of rough sets

It is easy to see the models that rough set theory gives matches the tasks researchers are dealing with in the area of data mining. However, an expressive model does not necessarily imply efficient algorithms. It has been shown that finding a smallest reduct of an arbitrary set Q is NP-hard [?]. Obviously, finding all reducts can have exponential time.

As we have seen RSDA has successfully modeled deterministic and indeterministic rules. A usual interest on indeterministic rules is its confidence. Because rough set uses partitions as a basic mechanism to describe problems and calculate the results, the underlying distribution (or measure) of the partitions can serve as a basis to calculate and estimate the confidence of a rule.

There have been many extensions to RSDA. Among them, *variable precision rough set model*(VPRS) [38], *rough mereology* [?], and similarity or indiscernibility based RSDA [17].

5 Quantitative Rule Mining

Numeric data comprises a significant proportion of real world databases. Despite this fact, the majority of data mining approaches have focused solely on the extraction of categorical association rules. The nature of the problem of mining quantitative association rules presents challenges distinct from standard categorical rule mining, and the approaches developed for the categorical case do not directly apply to this problem. This section concentrates on presenting two major approaches to mining association rules from databases containing numeric attributes. The approaches presented here provide differing viewpoints on the structure and content of quantitative association rules, as well as the way in which a rule is determined to be "interesting". These distinct approaches provide insight into the necessity of conceptualizing the properties of association rules that make them important to retain. This allows for maximal rule pruning and the reduction of the final set of association rules presented to an end-user, as well as the presentation of a set of rules that can be easily understood and acted upon. Examples in this section will be based on the database presented in Figure 2.

Age	Married	NumCars
23	No	1
25	Yes	1
29	No	0
34	Yes	2
38	Yes	2

Figure 2: A sample numeric database.

5.1 Partitioning

The first attempt at quantitative rule extraction was proposed by Srikant and Agrawal [32]. Their approach is an extension of previous work associated with categorical rule mining. They advocate partitioning the entire range of values of each quantitative attribute into a set of ranges, or intervals, which can then be mapped onto a set of integers, effectively creating categories from the numeric data. These numeric categories are then treated in a similar manner to standard categorical data. Association rules can be produced amongst binary, categorical and categorized numeric attributes using algorithms developed solely for binary and categorical data.

Srikant and Agrawal recognize two problems associated with the partitioning of quantitative data. The first problem is that as the number of intervals grows, or equivalently the attribute is more finely partitioned, the support for any single interval may decrease. This problem can lead to particular rules not being discovered with a fine partitioning because they lack the necessary support. A coarser partitioning of numeric data will act to increase support for those intervals that grow in size. As an example, consider the rule "(Age = 20..26) ⇒ (Married = No)" generated from the data displayed in Figure 2, where the attribute "Age" is partitioned into the intervals 20..26, 27..30, 30..35, and 36..40. With this partitioning the above rule has a support of 40%. If, however, the partitioning is made coarser so that the "Age" data is aggregated into the two partitions 20..30 and 30..40, then the above rule would be stated as "(Age = 20..30) ⇒ (Married = No)" with a support of 60%. The aggregation of data into fewer intervals causes the support of these larger intervals to rise.

The second problem is that as numeric data is partitioned into coarser intervals, information is lost and minimum confidence levels may not be attained for certain rules. As an example, consider the rule "(NumCars = 0) ⇒ (Married = No)", which has 100% confidence if "NumCars" is partitioned such that each object in the database in Figure 2 is in its own interval. If, however, the "NumCars" attribute is partitioned into the two intervals 0..1 and 2..3, the above rule would be stated "(NumCars = 0..1) ⇒ (Married = No)" and would have a confidence of 66%. Here the aggregation of numeric intervals decreases confidence in a particular rule.

The two problems described above act against each other in such a way that as one is ameliorated, the other is made worse. In recognizing this "catch-22" situation, Srikant and Agrawal propose the consideration of all possible continuous ranges over the intervals into which the numeric data has been partitioned. The partitioning is performed using a "partial completeness" measure aimed at producing optimal intervals which reduce the amount of information lost in the partitioning while aggregating enough to maintain support levels. The authors also utilize a "maximum support parameter" that prevents adjacent intervals from being combined when support goes beyond this value. This parameter acts to reduce the exponential growth in redundant rules that could occur by combining intervals that contain rules that already have the minimum necessary support.

5.2 A Statistical Theory

Aumann and Lindell [4] present an approach to numeric data mining that provides a unique definition of quantitative association rules, as well as a statistically-based measure of "interest" used in the generation of the rules. Their definition of quantitative association rules is based on the demonstration of a significant statistical difference in an attribute or set of attributes between a subset of the population and the remaining elements of that population. An example of a rule generated by the method of Aumann and Lindell is: "(Married = No) ⇒ (Age: mean = 26) (overall mean age = 29.8)". This rule can be interpreted to mean that the average age of all people in the database who are not married is 26 years, as opposed to all of the people in the database (married and not married), who have an average age of 29.8 years. (There is some degree of uncertainty as to whether Aumann and Lindell intend for the "overall mean age" to actually be the age of all those not contained in the antecedent, which for this case would be the married population contained in the database.) Rules of this type provide information regarding a statistical deviation of a subset of the population represented in the database. It should be noted that while the statistical measure used in this section will be the mean, any other measure of statistical distribution can be used to create rules of this type. The measure of "interest" used in the production and selection of these rules is one of statistically significant deviations in an attribute(s) of a subgroup.

Algorithms are presented for the production of two distinct types of association rule: $Categorical \Rightarrow Quantitative$ and $Quantitative \Rightarrow Quantitative$. A detailed algorithm is presented for the $Quantitative \Rightarrow Quantitative$ case, which produces rules of the form: "(NumCars [0, 1]) $\Rightarrow$ (Age: mean = 25.67) (overall mean age = 29.8)". In this type of rule the antecedent is always a range of a numeric attribute representing a subgroup and the consequent represents the statistically "interesting" behavior as a statistical measure of another, single numeric attribute. The $Categorical \Rightarrow Quantitative$ case produces rules in which the antecedent is a set of categorical attributes and their particular values, and the consequent contains the statistically "interesting" measures of one or more numeric attributes. This algorithm is much less developed in the paper.

Advantages of this statistical approach include the ease with which rules of both types mentioned above can be understood by an end-user. These rules seem to have a more intuitive appeal than those produced by the interval approach of Srikant and Agrawal. Also, this statistical approach apparently produces fewer rules than the interval approach, a major advantage when the large number of rules produced by typical data mining techniques begs the question of the utility of these methods.

6 Deterministic Implications

All of the preceding approaches to data mining are probabilistic in nature. They yield statements of the nature "a transaction involving item a and b, often also involve items c and d" or "any object with attributes a, b and c, most likely belongs to category d". Frequent sets are used to establish both the support and confidence for such statements.

Discrete, deterministic data mining (or DDDM) seeks deterministic logical implications that are true independent of frequency. This approach can be applicable in certain kinds of deterministic, causal environments where we are seeking "cause and effect" associations. It is totally inappropriate for market basket analysis or other non-deterministic environments.

6.1 Closure Theory

A collection $\mathcal{C}$ of subsets of some universe U is a closure system, if for all $S_1, S_2 \in \mathcal{C}$, $S_1 \cap S_2 \neq \emptyset$ implies $S_1 \cap S_2 \in \mathcal{C}$. That is $\mathcal{C}$ contains all its intersections. Alternatively, one can define $\mathcal{C}$ by a closure operator φ which is monotone, inclusion preserving, and idempotent. That is, $\forall X, Y \subseteq \mathbf{U}$, $X \subseteq X.\varphi$, $X \subseteq Y$ implies $X.\varphi \subseteq Y.\varphi$, and $X.\varphi.\varphi = X.\varphi$. The correspondence between these two ideas of closure is quite straight forward. Given the collection $\mathcal{C}$, $X.\varphi = \bigcap_{X \subset Y} Y \in \mathcal{C}$. And conversely, $\mathcal{C}$ consists just of those *closed* sets Z where $Z.\varphi = Z$.

The key aspect of closure theory for data mining lies in the concept of a *minimal generator* of a closed set Z, denoted $Z.\gamma$. This is a minimal set X (by inclusion) whose closure will be Z. See [24, 26] for background here.

Frequent set analysis yields associations of the form

$$ab \rightarrow abcde.$$

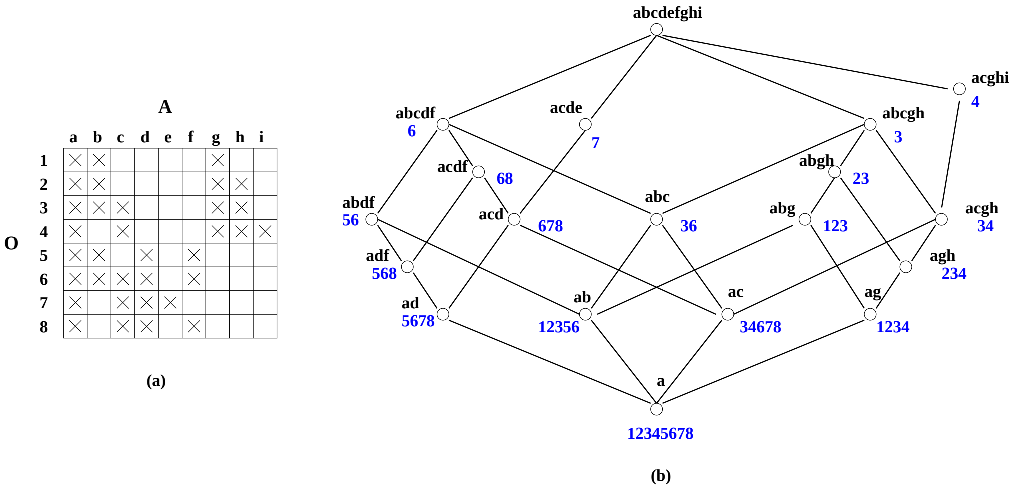
We have maximal information content when ab is minimal and $abcde$ is maximal. This will occur if the consequent $abcde$ is closed and its antecedent ab is its generator. This is one way that closure theory has entered into the mainstream of data mining.

6.2 Formal Concept Analysis

Concept lattices and formal concept analysis, as developed by Ganter and Wille in [11], have gotten a lot more play in data mining than closure theory. For example, it is central in papers by Godin, Missaoui, and Zaki [13, 14, 35]. Yet, it appears that much of the formal machinery is quite unnecessary. A major reason for involving concept lattices is to reduce the huge number of rules that *apriori*, frequent set data mining yields. For this, we suspect the concept of closed sets and their generators is sufficient.

Nevertheless, formal concept analysis does provide a mechanism for generating closed sets and their generators that can be interpreted as discrete logical implications. For example, Figure 3 shows the concept lattice that would be generated by the relation of Figure 1.

But, as seen this figure, formal concept lattices by themselves do not indicate the generators



	a	b	c	d	e	f	g	h	i
1	×	×					×		
2	×	×					×	×	
3	×	×	×				×	×	
4	×		×				×	×	×
5	×	×		×		×			
6	×	×	×	×		×			
7	×		×	×	×				
8	×		×	×		×			

Figure 3: The formal concept lattice $\mathcal{L}$ generated by the relation of Figure 1

of their closed sets.

We have written code that finds all closed sets and their generators using an algorithm similar to that in [12]. It is slow! A more powerful way is to use the faces of the closed sets to determine the generators according to the following theorem found in [26].

Theorem 6.1 *Let Z be closed and let $Z.\Gamma = \{Z.\gamma_i\}$ be its family of minimal generators. Then Z covers X in $\mathcal{L}$ iff $Z - X$ is a minimal blocker of $Z.\Gamma$.*

Using either method to determine the generators, we illustrate the generators of some of the closed sets of Figure 3 in Figure 4.

6.3 The Taylor/Pfaltz Algorithm

A capability missing in almost all frequent set data mining algorithms is the ability to continuously update R, that is treat R as a stream of tuples. Because *apriori*, and similar algorithms, must make more than one sweep over R, it must be fixed.[1] But, as Godin and Missaoui have shown, incremental creation and update of concept lattices is much faster [13, 14]. The problem is that their approach to incremental update does not update the generators as well. In [27] Theorem 6.1 is used to determine the generators of each closed set based on the faces of (sets covered by) the set.

[1] A major problem is that as more data is read, attribute combinations that originally passed the "min_sup" test may fail in the expanded data set — and vice versa. This creates non-trivial problems.

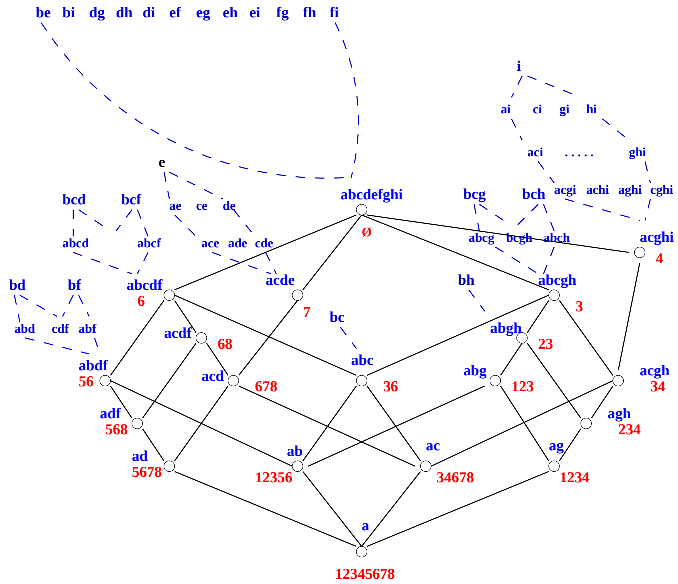


Figure 4: The concept lattice $\mathcal{L}$ of Figure 3 with some generators indicated.

The basic approach treats the attributes of each new row, tuple, or observation, as a closed set Y, which is either already in the lattice $\mathcal{L}$ or not. If it is new, there is some closed set Z that covers it in $\mathcal{L}$. Y is then inserted under Z, thereby creating a new face of Z and forcing the update of the generators of Z. This is embodied in a process called UPDATE_GENERATORS, which given a "new" closed set *new_c* updates the generators of the closed set *cov_c* immediately covering it in the lattice $\mathcal{L}$.

But, because $\mathcal{L}$ is a lattice of closed sets, any non-empty intersection $Y \cap X$, $X \in \mathcal{L}$ must also be in the lattice. Each such non-empty intersection must then be recursively entered into $\mathcal{L}$. Consequently UPDATE_GENERATORS is actually a subprocess of the driver routine INSERT_CLOSED_SET which given a new closed concept Y, inserts it in the lattice $\mathcal{L}$, updates the generators of the set Z that covers Y, then determines the closed sets that it covers, that is its non-empty intersections with sibling sets (or concepts). Those non-empty intersections which are not already in $\mathcal{L}$ are recursively entered. Pseudo-code for both these procedures is given in Appendix A.

Experiments with various data sets have shown (a) that updates are "local", in the sense that they are confined to a relatively small portion of the lattice [25, 28]; (b) that the number of closed concepts is typically one to two orders of magnitude fewer than frequent sets [36]; (c) that extraction of specific rules of interest is fairly straight forward [28]; and (d) that the current implementation is relatively slow. Clearly, an improved implementation of this process is needed.

6.4 Output from MUSHROOM Data Set

We exercised these procedures on the MUSHROOM data set (Section 9.2), because the properties of plant life tend to be deterministic. As described in that section, there were 3,773 deterministic rules generated. The most interesting rules tend to be those with only one or two antecedent attributes, because they are the easiest to apply. Those rules with only a single antecedent attribute are given in Figure 10.

Because of a natural desire to avoid poisonous mushrooms, we extracted in a categorical search all those rules with `p0` in the consequent. This is categorical data mining like that described in Section 2.5. Those with `p0` in the antecedent are clearly uninteresting, as are those whose antecedents contained one of the unique generators of `p0` as shown in Figure 10. These are easily filtered out. Finally, in Figure 11 we illustrate only those rules consisting of two attribute antecedents which denote poisonous mushrooms.

Similarly, in Figure 12, we display the same kind of rules that imply edible mushrooms.

7 Key Papers and Players

There is an immense body of literature *about* data mining; but surprisingly few actually describe data mining processes in sufficient detail to be able to replicate them and thus evaluate their performance. Although it has some rather idiosyncratic notation to over come, we have found that the book "Data Mining for Association Rules and Sequential Patterns" [1] by Jean-Mark Adamo is well worth reading. It lays out is considerable detail ordered search in an attribute space, followed by the *apriori* algorithm and a few of its variations.

To investigate in a quantitative way the key players and important publications in data mining we used the web-based citation database system called ResearchIndex [18]. ResearchIndex was queried for the top 25 papers with "data mining" in the bibliography entry and sorted by the number of times the publication had been cited by other publications. This is by no means meant as the most correct or only measure of the importance of an author or publication, but, it is however an easily obtainable quantitative measure of the impact of a researcher's work on the community. One limitation to this result is that although the ResearchIndex database is large, is not complete and hence may be missing some important work. With this in mind, it is the intention of this list is to provide a starting point for a new researcher to learn about publications and authors that are frequently cited.

7.1 Important Players

The following Table 1 is a list of all authors that published more then one paper that was listed in the top 25 by number of citations in ResearchIndex:

Author	number of publications cited
Rakesh Agrawal	4
Usama Fayyad	4
Heikki Mannila	4
Jiawei Han	3
Gregory Piatetsky-Shapiro	3
Padhraic Smyth	3
Hannu Toivonen	3
Peter Cheeseman	2
Marcel Holsheimer	2
Manish Mehta	2
Inkeri Verkamo	2

Table 1: Most frequently cited authors and publications

7.2 Author Biographies

- **Rakesh Agrawal**

  From author's website: "Rakesh Agrawal is the Project Leader and Manager of the Quest project on Data Mining and Decision Support Technologies at the IBM Almaden Research Center, San Jose, California. IBM is making the Quest technologies commercially available through its data mining product, IBM Intelligent Miner." [2]

- **Usama Fayyad**

  From digiMine website: "Dr. Fayyad is a co-founder of digiMine and has served as President and CEO since the company's inception in March 2000.

  Prior to digiMine, Dr. Fayyad founded and led the Data Mining & Exploration (DMX) Group at Microsoft Research from 1996 to 2000. His work there included the invention and development of scalable algorithms for mining large databases and customizing them for server products such as Microsoft SQL Server and OLAP Services. These components shipped in Microsoft SQL Server 2000 as part of the new industry standard in data mining, Microsoft's OLE DB API, which Dr. Fayyad also helped establish and promote. He also led the development of predictive data mining components for Microsoft Site Server (Commerce Server 3.0 and 4.0).

  From 1989 to 1995, Dr. Fayyad founded and headed the Machine Learning Systems Group at the Jet Propulsion Laboratory (JPL), California Institute of Technology, leading the development of data mining systems for the analysis of large scientific databases. During that time he received the most distinguished excellence award from Cal Tech/JPL and a U.S. Government Medal from NASA. He remains affiliated with JPL as Distinguished Visiting Scientist after joining Microsoft."[8]

- **Jiawei Han**

  From author's website: "Jiawei Han, Professor of the School of Computing Science and Director of Database Systems Research Laboratory, Simon Fraser University, Canada. He obtained his Ph.D. in Computer Sciences at the University of Wisconsin - Madison in 1985. Prior to joining SFU, he was an assistant professor in Northwestern University from 1986 to 1987. He has conducted research in the areas of data mining (knowledge discovery in databases), data warehousing, spatial databases, multimedia databases, deductive and object-oriented databases, and logic programming, with over 100 journal and conference publications. He is known for his work on data mining and has been invited to give talks or tutorials in international conferences, universities, and industry firms in many countries. His research has been supported by Natural Sciences and Engineering Research Council (NSERC) of Canada (1988–present), Network of Centres of Excellence of Canada (IRIS-3 Project Leader for the project "Building, Querying, Analyzing, and Mining Data Warehouses on the Internet", 1998-2002), IBM Canada, HP Lab, B.C. Science Council, B.C. Advanced Systems Institute, Seagate Software, and some other funding agencies."[15]

- **Padhraic Smyth**

  Padhraic Smyth is an Associate Professor at University of California, Irvine. [31]

- **Hannu Toivonen**

  Hannu Toivonen is a PhD, Professor of Computer Science University of Helsinki and a Principal Scientist Nokia Research Center, Helsinki. His research interests include knowledge discovery, data mining, computational methods for data analysis, analysis of scientific data, with applications in genetics, ecology, telecommunications, and information systems.[33]

- **Peter Cheeseman**

  From author's website: "Peter Cheeseman is a senior research scientist [at NASA Ames Research Center], whose specialization is in Artificial Intelligence, and Bayesian Inference Methods. He received his B.S. degree in Physics and Mathematics with honors from Melbourne University (Australia) in 1971, his M.Phil. in Applied Mathematics from Waikato University (New Zealand) in 1973, and his Ph.D. degree in Artificial Intelligence from Monash University (Australia) in 1979. His Ph.D thesis title was "A Problem Solving System with Learning". Dr. Cheeseman was a lecturer in Computer Science at the University of Technology, Sydney, Australia from 1978 to 1981. From 1981 to 1985, Peter performed research at SRI International in the areas of production planning, probabilistic methods for combining information, induction of probabilistic rules from data, development of a representation and procedure for spatial uncertainty estimation, and development of an information theoretic version of Bayesian estimation and its applications. In 1985 Peter began research at NASA Ames Research Center in the Artificial Intelligence Research Branch. He now manages a small group (4-6 people) who apply Bayesian inference methods to data analysis problems. This research has concentrated on the development of a practical general purpose automatic classification system, whose implementations are: AutoClass III; AutoClass X; and AutoClass C." [6]

- **Inkeri Verkamo**

  Inkeri Verkamo is a Ph.D., Docent, Senior Lecturer, Industry Professor (acting) at University of Helsinki / Nokia Research Center. His research include "software performance (including building metrics tools for software architectures in the MAISA project), software engineering (including building interfaces between software development environments in the VITAL project), data mining (including developing a Knowledge Extraction System for Statistical Offices in the KESO project)" [34]

No biographies were readily available for the following: Heikki Mannila, Gregory Piatetsky-Shapiro, Marcel Holsheimer (appears to be the founder of Data Distilleries but no further information available) and Manish Mehta

7.3 Important Publications

The top 10 publications sorted by number of citations in ResearchIndex on May 01, 2002:

351 citations: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. Advances in Knowledge Discovery and Data Mining. MIT Press, Cambridge, MA, 1996.

253 citations: R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and I. Verkamo, "Fast Discovery of Association Rules," in Advances in Knowledge Discovery and Data Mining, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI/MIT Press, 1995.

251 citations: J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In Proc. of the 12th ICDE, pages 152–159, New Orleans, February 1996. IEEE.

184 citations: U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusuamy, editors, Advances in Knowlede Discovery and Data Mining, chapter 1, pages 1-34, AAAI/MIT Press, Cambridge, MA, 1996.

173 citations: R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Proc. VLDB, pages 144–155, 1994.

115 citations: M. Ester, M.-P. Kriegel, J. Sander, and X. Xu (1996), "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 2-4 August, 1996, Portland, Oregon, 226-231.

108 citations: H. Mannila, H. Toivonen, A.I. Verkamo, "Discovering frequent episodes in sequences," Proc. KDD Conf., 1995.

103 citations: R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining application. Proc. ACM SIGMOD, 1998.

97 citations: M. Mehta J. Shafer, R. Agrawal. Sprint: A scalable parallel classifier for data mining. In Proceedings of the 22nd VLDB Conference, 1996.

92 citations: P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining. 1995.

8 Performance Issues

The process of data mining can be a very expensive task. Indeed, the data we are mining is usually orders of magnitude larger than anything which a human being can comprehend. In such circumstances, even an algorithm with quadratic complexity can be too expensive. Consequently, we usually seek algorithms with linear or log-linear complexity to perform our data mining tasks. However, this is another area where the intention of the data miner is crucial.

Information theory tells us that there is a certain limit to which a particular large body of data can be condensed without incurring loss of information. This limit is the entropy or information content of the data. Even if we could in practice reach this theoretical limit of compression, more often than not the resulting size of the data would still be far too large for a human being to examine. Hence, the effective mining of large data sets must permit and live with loss of information. The particular way this choice is handled can have a large impact on data mining performance.

The most commonly used approach to this issue is to set a frequency threshold and mine only rules which have a frequency of occurrence above this threshold. Such an approach arises out of the belief that if we must sacrifice some understanding of the domain, it would be best to sacrifice understanding of the least frequently occurring aspects. Indeed, this is the very approach that data mining was initially married to and is lovingly referred to as statistical data mining.

Recently, however, there has been interest in mining some of the less frequent aspects of a data set. Certain things, such as the threat of a terrorist attack or the existence of a rare breed of poisonous mushroom, seem worthy of our attention even if they occur only once and are buried inside of a large body of data. However, to mine such infrequent patterns places a large burden on the performance of traditional statistical data mining techniques. To address this issue, a number of data mining algorithms which are not statistical in nature are needed. But this then begs the question of where and how do we permit the necessary data loss to perform effective data mining if we want to mine the infrequent rules from a data set.

One approach that shows some promise is the use of closed sets. But closed sets by themselves do not afford such loss of information as is necessary and indeed on certain data sets could actually cause an increase in size. The notion of closed sets have also been used in tandem with the notion of frequent sets. This approach, pioneered by Pasquier, et al. [22] and by Zaki [35] is based on concept lattices and formal concept analysis which we discussed in Section 6.2.

8.1 Scalability

In light of the above discussion of the enormity of the data sets with which we work, the scalability of an algorithm is essential to its successful application in the data mining domain. There are two essential issues with regard to the ability of an algorithm to scale well as the data size increases. The first regards the complexity of the algorithm used (*e.g.* is it linear in the input, log-linear, quadratic, *etc.*). The second issue is whether the algorithm can break up the data, process the smaller chunks more efficiently, then merge the results. This issue essentially leads to the notion of parallelizing the processing of the chunks and can lead to enormous gains in efficiency if the needed hardware is available.

As discussed in Section 6.2, through the use of concept lattice theory and the notion of generators, we can extract all valid logical implications from a data set. Consequently, I will use this theory as a vehicle for discussing the complexity of rule extraction.

First we note that the rule extraction problem has worst-case exponential complexity (in the size of the given relation). Our relations are given by a set of rows and columns. Each row is an observation or object and the columns represent attributes or properties. The fact that the rule extraction problem has exponential complexity follows from the fact that the number of rules present in a data set can be exponential in the size of the relation for certain data sets. So if we are speaking of extracting all of the valid rules from a data set, then any algorithm must necessarily perform an exponential amount of work on such data sets because it must produce an exponential amount of output. This is a typical worst case argument to show that a problem has exponential complexity, yet it does not mean that an algorithm will always perform exponentially. It means simply that any algorithm that solves this problem can have no better than exponential complexity because there are some data sets for which an exponential amount of work is required.

This is best illustrated by an example. We will take as our example the simplest type of relation available, a binary relation. Consider the relation shown in Figure 5. The presence

	a	b	c	d
1		X	X	X
2	X		X	X
3	X	X		X
4	X	X	X	

Figure 5: A complete binary relation with the diagonal removed.

of an X in the figure indicates that the corresponding row has the corresponding attribute. For example, row 1 has attributes b, c, and d. You will readily notice that the relation given is simply the complete relation with the diagonal removed. This relation gives rise to the maximal number of concepts possible when the concept lattice is generated. Every member of the power set of A (the attribute set) is a closed set, and hence a concept. Indeed, it can be shown that for any sets O and A there exists a binary relation R(O,A) that gives rise to $min(2^{card(O)}, 2^{card(A)})$ concepts. This produces a number of concepts exponential in the size of the relation.

Now that we have come to terms with the fact that mining all of the valid rules from a data set is an exponential problem, we can see why this presents such an issue for data miners. If we are working with data sets so huge that even quadratic performance may be unacceptable, then certainly exponential performance won't do. So once again the issue arises that we must sacrifice some understanding. We can not extract and present all of the valid rules that a relation gives rise to so we must be more selective. This is the justification for moving towards the area of heuristic methods. The complexity of a heuristic method is almost entirely dependent on the heurism employed so we will not say more about such complexity. But before we do leave the issue of complexity, it is worthwhile considering the rate at which concepts are produced in an example. Though we have said that the number of concepts produced from a relation can be exponential, in practice we may not encounter such complexity.

In Figure 6 we have plotted the total number of concepts generated after adding each row of the MUSHROOM relation (discussed in Section 9.2). In Figure 7 we have plotted the

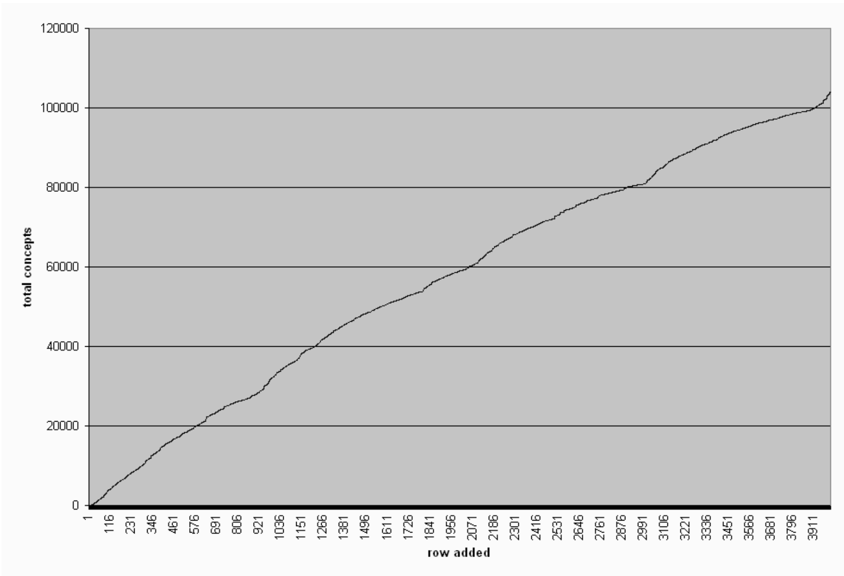


Figure 6: The total number of concepts for the MUSHROOM relation after adding each row.

number of concepts added for each row in the same relation. The plots in figures 6 and 7

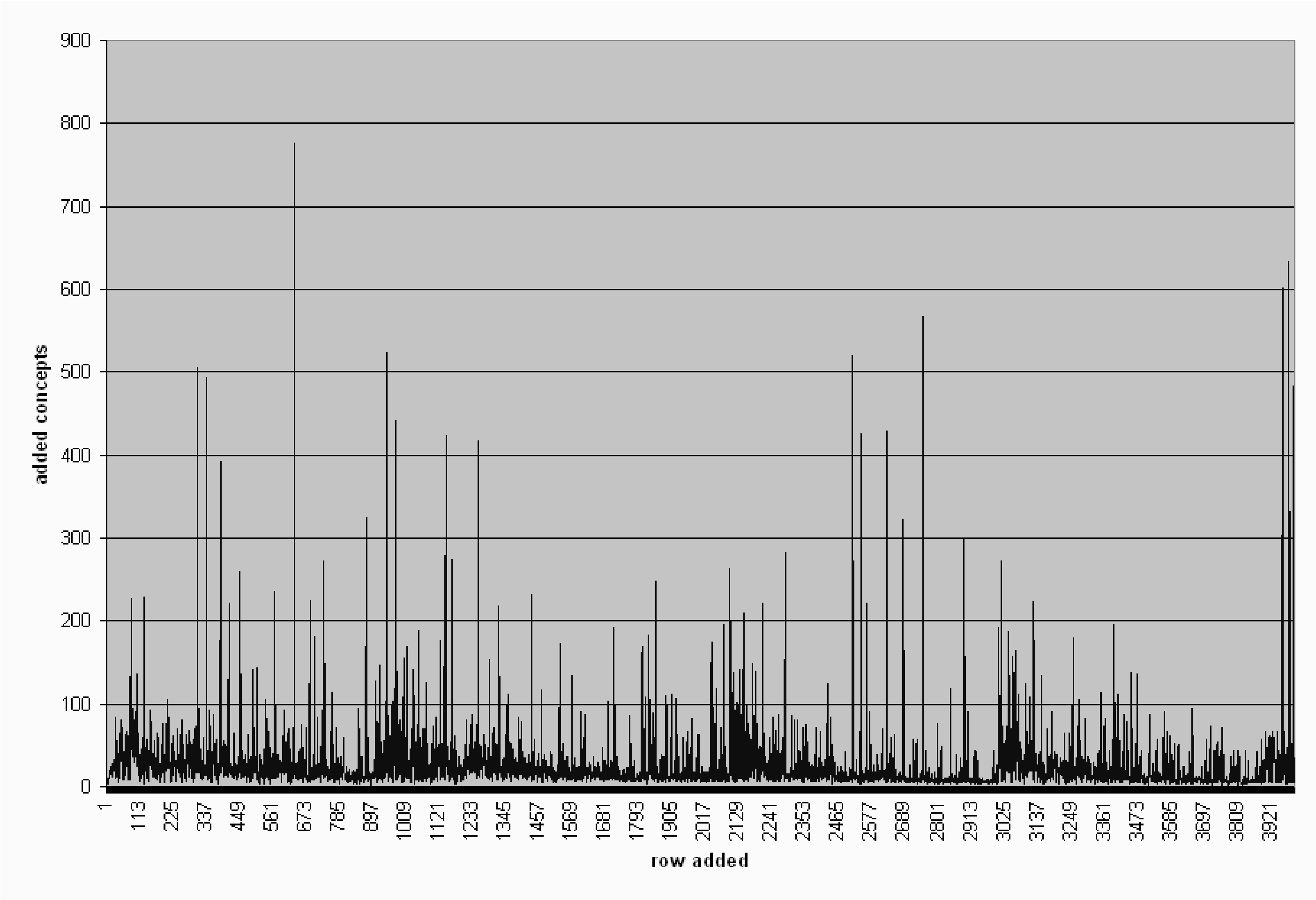


Figure 7: The number of concepts added for the MUSHROOM relation after adding each row.

are given for the first 4000 rows of the mushroom relation in which 86 distinct attributes are encountered. Figure 6 seems to indicate that the total number of concepts increases linearly as we add rows to the relation.

In Figure 7 we can see that the number of concepts added for each row is quite erratic. Most of the rows add under 100 concepts. However there are various spikes scattered about, some reaching as high as 600 to 800 concepts added. We have noticed that a great majority of these spikes correspond to a row in which we first encounter a new attribute. For example, the highest spike of 776, encountered upon adding row 633, corresponds to such a new attribute. That is, in row 633 there is an attribute which does not appear in any of the preceding rows of the relation.

It is not difficult to explain why encountering a new attribute can lead to such an explosion of concepts added. The algorithm presented in [27] for inserting a closed set into the lattice recurses whenever the intersection of a sibling concept with the closed set is non-empty. However, if this non-empty intersection is already in the lattice, no further work will be done. The insertion of a closed set (row) which contains a previously unseen attribute may exercise this recursion greatly. This is due to the fact that such a closed set must necessarily be inserted at the top of the lattice (directly under the lattice supremum). All of the concepts inserted as a result must reside below the new concept. This leaves the entire lattice open to new insertions.

It is quite remarkable that such an erratic seeming process as pictured in Figure 7 can somehow lead to such a smooth overall effect as seen in Figure 6. But this is what we have observed and it seems that many data sets exhibit a more linear relationship between total concepts and rows added. However, the worst-case exponential relationship should not be forgotten. Indeed, Figure 8 shows the same relationship between total concepts and rows added but for a randomly generated relation consisting of 40 distinct attributes. This figure

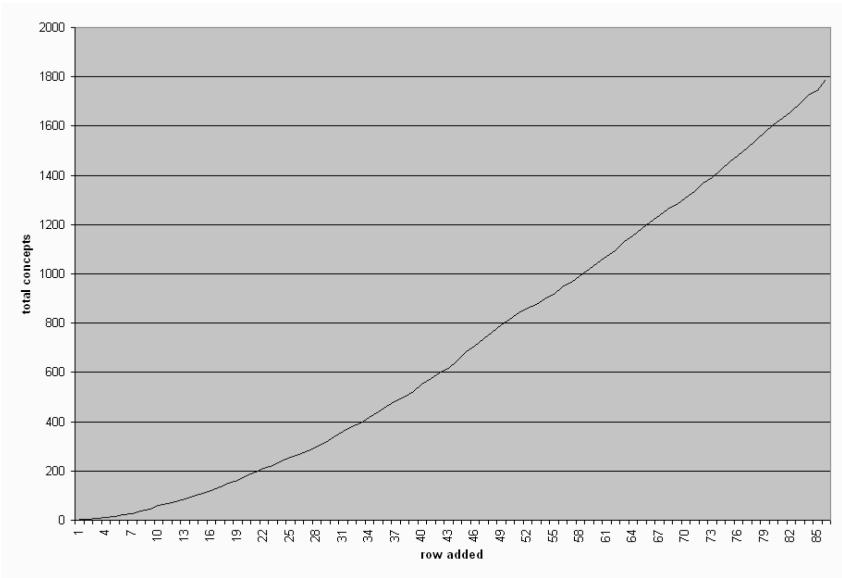


Figure 8: The total number of concepts for a random relation after adding each row.

exhibits a much more exponential relationship than that of the MUSHROOM data.

This is not an accidental phenomenon it seems. In a sense, the growth of total concepts as rows are added is reflective of the cohesion of the relation. Indeed, if we look at the extremes we can get an intuitive idea for why this might be so. If the relation is entirely uniform (i.e. all rows are the same) then there will be only one concept. This is the minimal number of concepts that can be present in the lattice. On the other hand, if the relation is perfectly uniformly random (i.e. every member of the power set of A is equally likely to appear) then we would expect the concept lattice to approach a complete lattice quickly. This remains to be seen.

The issue of complexity has been discussed for closed sets in order to gain some understanding of the complexity of data mining. The complexity with which the closed sets grow as rows are added to the relation is a good approximation for the complexity with which the number of rules generated grows. However, this only addresses rules of 100% confidence. If we lower our confidence floor to consider rules which may not always be true but are true often enough, then we will obviously get a superset of the rules we get with 100% confidence. Hence the above discussion still applies as a lower bound on the complexity of the rule mining problem.

9 Links to Resources

9.1 Data Sets

One of the most valuable resources is the UCI Data Repository [5] which is accessible at:

`http://www1.ics.uci.edu/~mlearn/MLRepository.html`

This repository contains over 123 data sets of various sizes and compositions. It is the source of the MUSHROOM data set which we used extensively in our exploration, and which is described more fully below.

Several databases from the "Wisconsin Breast Cancer Database" are available at:

`ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn`

or linked from Olvi Mangasarian's site at University of Wisconsin's Department of Computer Science:

`http://www.cs.wisc.edu/~olvi`

These datasets contain tumor characteristics, including benign or malignant, obtained from fine needle aspirates. They been used in several pattern recognition and linear programming studies.

In performance evaluation, a significant body of data mining literature used the C++ synthetic workload generator available from IBM:

`http://www.almaden.ibm.com/cs/quest/DEMOS.html`

It might deserve some review considering its popularity. However, note that synthetic workloads may not capture all natures of real workloads. In [37], Zheng *etal.* compared the performance of well known algorithms, i.e., *apriori*, *Charm*, *FP-growth*, *Closet*, and *MagnumOpus*, using both synthetic and real workloads such as retail and e-commerce transactions. For fair comparisons, they also acquired the corresponding software from the original authors. Zheng *etal.* claim these retail and e-commerce data are available in the public domain. Interested readers may contact the authors.

9.2 The Mushroom Database

Several sections refer to the MUSHROOM data set found in this UCI Data Repository. This data set consists of 8,124 records describing various kinds of mushrooms as found in the "The Audubon Society Field Guide to North American Mushrooms" [19] has been used in many data mining studies.

Each record consists of 22 attributes, all of which have nominal values. Because each of these attributes can have multiple values, the number of binary attributes is effectively over 100. Consequently, for many of our studies we used only the first 9 attributes which are enumerated in Figure 9. These were converted to binary attributes by appending to letter "values" on the right, the attribute number on the left. Thus, for example, a mushroom with attributes `g2` and `p5` has a "grooved cap surface" and a "pungent odor". A single mushroom will exhibit precisely 9 of these 42 possible binary attributes. The first two rows

Attr-0	**edibility:**	**e=edible, p=poisonous**
attr-1	**cap shape:**	**b=bell, c=conical, f=flat, k=knobed, s=sunken, x=convex**
attr-2	**cap surface:**	**f=fibrous, g=grooved, s=smooth, y=scaly**
attr-3	**cap color:**	**b=buff, c=cinnamon, e=red, g=gray, n=brown, p=pink, r=green, u=purple, w=white, y=yellow**
attr-4	**bruises?:**	**t=bruises, f=doesn't bruise**
attr-5	**odor:**	**a=almond, c=creosote, f=foul, l=anise, m=musty, n=none, p=pungent, s=spicy, y=fishy**
attr-6	**gill attachment:**	**a=attached, d=descending, f=free, n=notched**
attr-7	**gill spacing:**	**c=close, d=distant, w=crowded**
attr-8	**gill size:**	**b=broad, n=narrow**

Figure 9: The first 9 attributes of the MUSHROOM data set, with values.

of this reduced data set are

```
p0 x1 s2 n3 t4 p5 f6 c7 n8
e0 x1 s2 y3 t4 a5 f6 c7 b8
```

Readily, the attributes `e0` (edible) and `p0` (poisonous) are of considerable interest. Many of the data mining experiments of the literature have treated this data set as categorical data mining, that is discovering which of the other attributes can be used to assign a specific mushroom to one of these two categories.

Frequent set data mining using *apriori* yields 25,210 rules when we set $min\_sup = 1\%$ and $min\_conf = 90\%$. The discrete data mining technique described in Section 6 yields 2,641 concepts. But since some concepts have several generators, this translates into 3,773 distinct rules. We observe the nearly 10-fold reduction described by Zaki [35].

To provide some sense of this data set we list in Figure 10 those rules $A \Rightarrow B$ which have a singleton attribute for A. We have added the concept number to the left to indicate where this rule was uncovered in the discrete data mining process (Section 6) and its support to indicate the `min_sup` necessary to consider it frequent (Section 2.1). If $\sigma = 1\%$, `min_sup` $=$ 81.

The discrete data mining of Section 6 describes how other implications can be obtained from this data set in Section 6.4. Figure 11 illustrates those non-trivial implications $A \Rightarrow B$ for which $|A| = 2$ and `p0` $\in B$. Again recall that if $\sigma = 1\%$, $min\_sup = 81$, so only one of these rules would have been discovered by frequent set analysis.

Figure 12 illustrates the same kind of non-trivial implication, but with `e0` $\in B$.

9.3 Open Source Code

An open source *apriori* implementation is available at:

`http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html`

This a very efficient implementation using C, which was the fastest among several tested

CONCEPT	IMPLICATION	SUPPORT
60	w3 -> f6	1040
105	t4 -> f6	3376
109	n8 -> f6	2512
117	a5 -> e0, t4, f6	400
144	l5 -> e0, t4, f6	400
313	s1 -> e0, f2, f4, n5, f6, c7, n8	32
556	f2 -> f6	2320
604	w7 -> f6	1312
668	c5 -> p0, x1, f4, f6, n8	192
720	y3 -> f6	1072
898	b3 -> t4, f6, c7, b8	168
924	f5 -> p0, f6, c7	2160
1007	p3 -> f6	144
1081	u3 -> e0, y2, f4, n5, f6, c7, n8	16
1401	g2 -> p0, w3, t4, n5, f6, w7, n8	4
1553	r3 -> e0, y2, f4, n5, f6, c7, n8	16
1597	s5 -> p0, f4, f6, c7, n8	576
1687	y5 -> p0, f4, f6, c7, n8	576
2019	a6 -> f4, c7, b8	210
2022	m5 -> p0, y2, f4, c7, b8	36
2162	e3 -> c7	1500
2562	c1 -> p0, n5, f6, w7, n8	4

Figure 10: All implications in MUSHROOM with a single antecedent.

implementations of *apriori* available in the public domain.

Open source code for analyzing the MUSHROOM database using neural networks and genetic algorithms can be found at:

`http://www.cs.virginia.edu/~abh2n/mushroom.html`

This source code is implemented using Matlab, but is basic enough that it should not be difficult to rewrite in C or LISP.

CONCEPT	IMPLICATION	SUPPORT
666	p3, w7 -> p0, x1, f4, c5, f6	32
667	p3, f4 -> p0, x1, c5, f6, n8	64
667	p3, n8 -> p0, x1, f4, c5, f6	64
696	f2, p3 -> p0, x1, f4, c5, f6, n8	32
1184	b1, n8 -> p0, n5, f6	12
1495	b1, b3 -> p0, t4, n5, f6, c7, b8	12
1567	b1, p3 -> p0, t4, n5, f6, c7, b8	12
2081	y3, n5 -> p0, f4, f6, n8	24
2177	e3, f4 -> p0, c7	876
2181	y2, a6 -> p0, f4, m5, c7, b8	18
2372	c3, a6 -> p0, y2, f4, m5, c7, b8	6
2470	e3, a6 -> p0, y2, f4, m5, c7, b8	6
2561	c1, y3 -> p0, y2, f4, n5, f6, w7, n8	2
2561	c1, f4 -> p0, y2, y3, n5, f6, w7, n8	2
2563	c1, y2 -> p0, n5, f6, w7, n8	3

Figure 11: Rules with two attribute antecedents that denote poisonous mushrooms.

CONCEPT	IMPLICATION	SUPPORT
61	w7, b8 -> e0, f4, n5, f6	1056
119	y3, t4 -> e0, f6	400
131	s2, y3 -> e0, t4, f6	152
452	f2, t4 -> e0, f6	912
586	f2, e3 -> e0, t4, n5, f6, c7	288
1015	x1, r3 -> e0, y2, f4, n5, f6, c7, n8	8
1261	k1, b3 -> e0, t4, n5, f6, c7, b8	16
1347	f2, c3 -> e0, f4, n5, f6, w7, n8	12
1944	b1, g3 -> e0, f4, n5, f6, w7, b8	48
1966	k1, g3 -> e0, f4, m5, f6, w7, b8	48
2342	s2, c3 -> e0, t4, m5, f6, c7, b8	4
2343	c3, t4 -> e0, n5, f6, c7, b8	8

Figure 12: Rules with two attribute antecedents that denote edible mushrooms.

10 Open Problems

10.1 Negative Information

In a binary relation, it is customary to indicate those attributes which are present. But, sometimes the absence of an attribute is itself an important characteristic. This is particularly true in the kind of categorization process found in Section 2.5.

In theory, if the rules discovered by the data mining procedure are deterministic, as in Section 6, then one should be able to involve the negation of attributes through logical manipulation. But, the underlying rule based nature of data mining makes this difficult, at best. Suppose we want to derive a rule such as $\neg a \rightarrow bc$. This is easily transformed into $a \vee bc$, or $a \vee (b \wedge c)$. But, now it is not clear how to transform either of these expression back into an implicative rule involving a (not negated).

Readily this problem is compounded when the implication is only probabilistic.

So, the usual solution is to simply add another attribute, $\neg a$, to the base data relation R. This can effectively double the number of attributes; and since most procedures appear to be *exponential in the number of attributes*, it is prohibitive. (Can we justify this assertion?)

10.2 Numeric Attributes

As observed in Section 1, many relations R are numeric, not binary. This is particularly true of scientific relations which record measurements. Figure 2 in Section 5 is representative. The two methods presented in Section 5 represent only the first steps in the development of ideas for the efficient and productive mining of quantitative association rules.

10.3 Rule Reduction

As discussed in Section 2, the number of rules generated can frequently be prohibitive with respect to human interpretation. Therefore, additional methods are required to reduce these rules to a manageable level. Although work towards this goal was presented in Section 2.5, it is our belief that a more objective technique can and should be developed. We feel that this technique should use some form of information content as a criteria, perhaps using maximal relative entropy reduction as a guideline.

References

[1] Jean-Mark Adamo. *Data Mining for Association Rules and Sequential Patterns.* Springer Verlag, New York, 2000.

[2] Rakesh Agrawal. Rakesh Agrawals's short biography. http://www.almaden.ibm.com/u/ragrawal/bio.html, 2002.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining Association Rules between Sets of Items in Large Databases . In *Proc. 1993 ACM SIGMOD Conf.*, pages 207–216, Washington, DC, May 1993.

[4] Yonaton Aumann and Yehuda Lindell. A Statistical Theory for Quantitative Association Rules. In *5th ACM SIGKDD Conf. on Knowledge Discovery and Datamining*, pages 261–270, San Diego, CA, Aug. 1999.

[5] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[6] Peter Cheeseman. Peter Cheeseman's homepage. http://ic.arc.nasa.gov/ic/projects/bayes-group/people/cheeseman/, 2002.

[7] Jose' Costa and Marcio Netto. Clustering of Complex Shaped Data Sets via Kohonen Maps and Mathematical Morphology. In B. Dasarathy, editor, *Proc. of the SPIE, Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, volume 4384, 2001.

[8] digiMine. About digiMine. http://www.digimine.com/about/fayyad.asp, 2002.

[9] I. Duntsch. A logic for rough sets. *Theoretical Computer Science*, 179:427–436, 1997.

[10] Saso Dzeroski. Data Mining in a Nutshell. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining.* Springer Verlag, 2001.

[11] Bernard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations.* Springer Verlag, Heidelberg, 1999.

[12] Bernhard Ganter and Klaus Reuter. Finding All Closed Sets: A General Approach. *Order*, 8(3):283–290, 1991.

[13] Robert Godin and Rokia Missaoui. An incremental concept formation approach for learning from databases. In *Theoretical Comp. Sci.*, volume 133, pages 387–419, 1994.

[14] Robert Godin, Rokia Missaoui, and Hassan Alaoui. Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[15] Jiawei Han. Jiawei Han - current research. http://www.cs.sfu.ca/~han/research.html, 2002.

[16] Jiawei Han, Jian Pei, and Yiwen Yin. Mining Frequent Patterns without Candidate Generation. In W. Chen, J. Naughton, and P. Bernstein, editors, *Proc. of 2000 SIGMOD Conf. on Management of Data*, volume 29, pages 1–12, Dallas, TX, June 2000.

[17] X. Hu and N. Cercone. Rough Set Similarity Based Learning from Databases. In *Proc. of The first International Conference on Knowledge Discovery and Data Mining*, pages 162–167, 1995.

[18] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.

[19] G. H. Lincoff. *The Audubon Society Field Guide to North American Mushrooms.* Alfred A. Knopf, New York, 1981.

[20] Carlos Ordonez, Edward Omiecinski, Levien de Braal, Cesar A. Santana, Norberto Ezquerra, Jose A. Taboada, David Cooke, Elizabeth Krawczynska, and Ernst V. Garcia. Mining Constrained Association Rules to Predict Heart Disease. In *IEEE International Conf. on Data Mining, ICDM*, pages 433–440, 2001.

[21] E. Orlowska and Z. Pawlak. Representation of nondeterministic information. *Theoretical Comptuer Science*, 29:27–39, 1987.

[22] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lofti Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *Proc. 7th International Conf. on Database Theory (ICDT)*, pages 398–416, Jan. 1999.

[23] Z. Pawlak. Rough Sets. *Internat. J. Comput. Inform. Sci.*, 1:341–356, 1982.

[24] John L. Pfaltz. Closure Lattices. *Discrete Mathematics*, 154:217–236, 1996.

[25] John L. Pfaltz. Transformations of Concept Graphs: An Approach to Empirical Induction . In *2nd International Workshop on Graph Transformation and Visual Modeling Techniques*, pages 320–326, Crete, Greece, July 2001. Satellite Workshop of ICALP 2001.

[26] John L. Pfaltz and Robert E. Jamison. Closure Systems and their Structure. *Information Sciences*, 139:275–286, 2001.

[27] John L. Pfaltz and Christopher M. Taylor. Concept Lattices as a Scientific Knowledge Discovery Technique. In *2nd SIAM International Conference on Data Mining*, pages 65–74, Arlington, VA, Apr. 2002.

[28] John L. Pfaltz and Christopher M. Taylor. Uncovering Logical Implications in Scientific Databases through Empirical Induction. In *ACM SIGMOD Conference*, page (in review), Madison, WI, 2002.

[29] G. Shafer. *A Mathematical Theory of Evidence.* Princeton Univ. Press, 1976.

[30] A. Skowron and J. W. Grzymala-Busse. From rough set theory to evidence theory. In *Advances in the Dempster-Shafer Theory of Evidence*, pages 193–236. 1993.

[31] Padhraic Smyth. Padhraic Smyth. http://www.ics.uci.edu/~smyth/, 2002.

[32] Ramakrishnan Srikant and Rakesh Agrawal. Mining Quantitative Association Rules in Large Relational Tables . In *Proc. of 1996 SIGMOD Conference on Management of Data*, pages 1–12, Montreal, Quebec, June 1996.

[33] Hannu Toivonen. Hannu T.T. Toivonen. http://www.cs.helsinki.fi/u/htoivone/, 2002.

[34] Inkeri Verkamo. Inkeri Verkamo. http://www.cs.helsinki.fi/u/verkamo/, 2002.

[35] Mohammed J. Zaki. Generating Non-Redundant Association Rules. In *6th ACM SIGKDD Intern'l Conf. on Knowledge Discovery and Data Mining*, pages 34–43, Boston, MA, Aug. 2000.

[36] Mohammed J. Zaki and Ching-Jui Hsiao. CHARM: An Efficient Algorithm for Closed Association Rule Mining. In Robert Grossman, editor, *2nd SIAM International Conf. on Data Mining*, pages 457–473, Arlington, VA, April 2002.

[37] Zijian Zheng, Ron Kohavi, and Llew Mason. Real World Performance of Association Rule Algorithms. In Foster Provost and Ramakrishnan Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406, 2001.

[38] W. Ziarko. Variable precision rough set model. *Journal of Computer and System Science.*, 46, 1993.

A Pseudo Code

```
void update_generators( concept cov_c, concept new_c )
     // "cov_c" will cover the new concept "new_c"
     // update cov_c.gens to reflect the face
     // cov_c.atts - new_c.atts
     {
     element elem;
     set face, gsup;
     col new_GEN, keep_GEN, diff_GEN;
     int keep;

     face = cov_c.atts - new_c.atts;
     new_GEN = EMPTY_COL;
     for each gen_set in cov_c.gens
          if( (gen_set INTERSECT face) != EMPTY_SET )
               add_to_col(gen_set, new_GEN);

     keep_GEN = new_GEN;
     diff_GEN = cov_c.gens - keep_GEN;
     for each gen_set in diff_GEN
          {
          for each elem in face
               {
               keep = 1;
               gsup = gen_set UNION {elem};
               for each keep_set in keep_GEN
                    if( keep_set IS_SUBSET_OF gsup )
                         {
                         keep = 0;
                         break;
                         }
               if( keep )
                    add_to_col(gsup, new_GEN);
               }
          }
     cov_c.gens = new_GEN;
     }
```

```
insert_closed_set (SET Z, SET Y, LATTICE L)
        // Insert the closed set Y into L so that
        // it is covered by Z
        {
        SET Y[];

        update_gen (Z, Y, L);
        for_each Y[i] covered by Z do
               {       // Y[i] will be a sibling of Y
               if (not empty(Y meet Y[i]) )
                      {
                      update_gen (Y, Y meet Y[i], L);
                      if (Y meet Y[i] in L)
                             continue;
                      X = new SET (Y meet Y[i]);
                      insert_closed_set (Y[i], X, L);
                      }
               }
        }
```