Discovering associations with numeric variables

Geoffrey I. Webb
School of Computing and Mathematics
Deakin University
Geelong, Vic. 3217, Australia
webb@deakin.edu.au

ABSTRACT

This paper further develops Aumann and Lindell's [3] proposal for a variant of association rules for which the consequent is a numeric variable. It is argued that these rules can discover useful interactions with numeric data that cannot be discovered directly using traditional association rules with discretization. Alternative measures for identifying interesting rules are proposed. Efficient algorithms are presented that enable these rules to be discovered for dense data sets for which application of Auman and Lindell's algorithm is infeasible.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; I.2.6 [**Artificial Intelligence**]: Learning; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Impact Rule, Association Rule, Numeric Data, Search

1. INTRODUCTION

Association rules [1] have demonstrated the ability to detect interesting associations between fields in a database. However, they utilize frequency statistics and hence have limited utility for quantitative analyses. In particular, they cannot directly segment data to optimize a numeric target. Aumann and Lindell [3] propose rule structures that associate conditions (an antecedent) with an impact upon a target numeric variable. Somewhat confusingly, they use the name *quantitative association rules*, previously used by Srikant and Agrawal [17] to describe techniques for automatic discretization for association rules. This paper presents extensions to Aumann and Lindell's [3] proposal, which is renamed *impact rules* in order to distinguish it from the work of Srikant and Agrawal. Impact rules provide a form of data mining analysis for detecting useful interactions between combinations of data selectors and a numeric variable.

I characterize impact rules as follows. A *training set* is a finite set of *records*, where each record is an element to which we apply Boolean predicates called *conditions*, and which is associated with a numeric value called the *target*. An *impact rule* consists of a conjunction of conditions, called the *antecedent*, and one or more statistics, called the *consequent*, describing the impact on the target of selecting the training set records that satisfy the antecedent.

Impact rule analysis may seek a finite number n of impact rules that individually optimize some function of quality, usually one of the rule statistics. Alternatively, as proposed by Aumann and Lindell, it may seek all impact rules that satisfy specified constraints.

The following provides a simple example of a hypothetical impact rule.

```
prev-response>0.2 & region=E & socio-class=F
→ profitability:  count=105211, mean=12.51,
sum=1316190.61, max=205.31, min=-1.05
```

In this example, the antecedent is the three conditions that precede the arrow ($\rightarrow$), the target is *profitability*, and the consequent comprises statistics summarizing the impact on the target of selecting the records that satisfy the antecedent. Such a rule might be valuable for identifying classes of customers from a mailing list that might be targeted most profitably in a mailing campaign.

Aumann and Lindell [3] propose that such rules be found by identifying frequent itemsets [1] and then, treating each as an antecedent, calculating the appropriate statistics for the target. An *itemset* is a set of conditions. A *frequent itemset* is an itemset that covers at least a predefined minimum number of training set records. The primary difficulty with this approach is that frequent itemset generation is only feasible for sparse data [5]. This paper presents an alternative strategy for impact rule discovery, utilizing the OPUS search algorithm [19], that has lower computational requirements than a frequent itemset approach and which avoids in many cases the requirement that arbitrary constraints (namely, minimum cover) be imposed on the rules that be considered.

Aumann and Lindell [3] suggested that distribution based measures of interestingness be utilized, such as the deviation of the mean or variance of the target for the records selected by the antecedent from the mean or variance of the training set. I propose alternative measures of interestingness that are likely to be useful in a wide range of practical applications.

To appear in *KDD-2001* San Francisco, CA, August, 2001.

2. ALTERNATIVE NUMERIC DATA MINING TECHNIQUES

It is valuable to contrast impact rules to alternative data mining algorithms that address numeric data. These fall into two categories, regression techniques and association rules using discretization of numeric values.

Regression techniques can be used to predict a numeric value for a given set of input parameters. Examples of such techniques include linear regression, regression trees [9], and neural networks. The relationship between regression techniques and impact rules is like the relationship between classification rules and association rules. Regression techniques allow a prediction to be made about a target value for a given data point. However, they do not provide an explicit segmentation of the data on the basis of performance on the target value. For example, regression techniques may allow the user to predict for a given customer the likely profitability of that customer, but they will not provide an explicit description of a segment of customers for whom profitability is particularly high. In contrast, impact rules provide a characterization of a segment of the data which has a particular impact on the target variable, but do not support prediction of numeric values.

One approach that has been employed for segmentation with numeric data is association rule discovery with discretization. Under this strategy, numeric data fields are converted to discrete valued data fields by aggregating values within set ranges. A particularly sophisticated approach to this is proposed by Srikant and Agrawal [17]. Their quantitative association rule discovery algorithm performs dynamic discretization, forming appropriate ranges on numeric attributes during the association discovery process.

However, even such a sophisticated approach is limited in its utility with respect to segmenting data with respect to a numeric value. Provision of statistics relating to the frequency of one sub-range of values of a target variable does not directly address the issue of the impact on the general distribution of values for that variable. For example, a data segment that has increased frequency for `profitability = high`, irrespective of the definition of `high`, could have a decreased mean value for `high`, due to a corresponding increased frequency of extremely low values for the target. Likewise, two associations with identical frequencies of some specific sub-range of profitability may have dramatically different mean values of profitability. Similar comments apply to statistics other than the mean, such as the sum, mode, or median, that might be of interest. There are clear advantages to segmenting on the basis of the precise distribution of a numeric value rather than a distribution distorted by discretization.

3. THE OPUS SEARCH ALGORITHM

The OPUS search algorithm [19] provides a framework for efficient search for impact rules. It enables systematic search through the combinations of conditions that may appear in an antecedent, pruning the search space according to the requirements of a particular search. OPUS was explicitly designed to provide efficient search through such search spaces. It improves upon previous algorithms for rule search [10, 14, 15, 16] by substantially increasing the amount of the search space that is pruned by each pruning action. It has been successfully applied to learning both classification rules [18] and association rules [20].

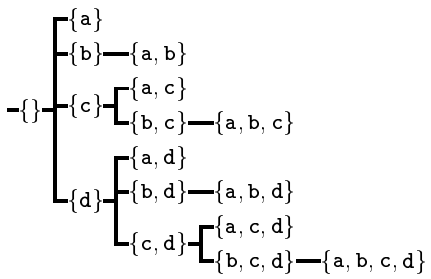

Figure 1: A fixed-structure search space

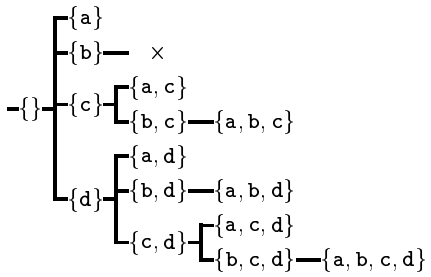

Figure 2: Pruning a branch from a fixed-structure search space

OPUS was developed to provide efficient search for combinations of elements that optimize some given metric. Previous algorithms [10, 13, 14, 15, 16] relied upon assigning an arbitrary order to the elements that was then used to structure the search space so that each combination of elements was considered just once. A search space with four elements (a, b, c, and d) structured in this manner is presented in Fig. 1.

Such a search space is exponential in size. For m elements the search space is 2^m. In many data mining contexts where the elements are the conditions that may appear in the antecedent of a rule, m may exceed 10,000. Clearly it will be possible to explore such a search space only if it can be pruned. Under fixed structure search, algorithms typically seek to identify branches that cannot contain a solution[1], and prune those branches. Under this search strategy, pruning the nodes below $\{b\}$ removes only one node from the search space, as illustrated in Fig. 2.

The identification of branches to be pruned requires pruning rules. These identify regions of the search space that cannot contain a solution. In rule discovery search, many pruning rules consider for a given node N whether any search node in the space below N that contains a given condition C can be a solution. Thus, a pruning rule identifying that no node containing b may contain a solution would enable pruning the nodes below $\{b\}$. This is the pruning achieved by the previous search algorithms. In this case, the ideal outcome would be the removal from the search space of all nodes containing b, as illustrated in Fig 3. This approxi-

[1]What constitutes a solution will depend upon the search objective. For example, in association rule discovery, a solution might be any set of conditions that is a frequent itemset.

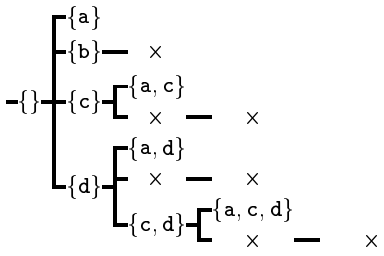

Figure 3: Pruning all nodes containing a single operator from a fixed-structure search space

mately halves the remaining search space[2].

The OPUS^S search strategy [19] achieves this with minimal overheads by maintaining at each node a set of search operators[3] that may be applied below that node, and deleting from that set any operator that cannot lead to a solution below the current node. This algorithm guarantees that every pruning action approximately halves the remaining search space. This algorithm has been demonstrated to achieve dramatic reductions in compute time in comparison to previous search algorithms for these search spaces and to support efficient complete search of a number of standard rule discovery search tasks [19].

4. SEARCH FOR IMPACT RULES

Using OPUS for impact rule discovery, search may be constrained to find the top n impact rules on some measure, and pruning may remove branches that cannot lead to an impact rule that satisfies that constraint. For many measures, this constraint is sufficient to provide efficient search. Where the measure does not facilitate effective pruning, additional constraints, such as minimum cover, can be employed.

Table 1 presents OPUS_IR, an impact rule discovery algorithm based on OPUS. *Current* is the set of conditions in the antecedent of the rule currently being considered. *Available* is the set of conditions that may be added to the antecedent of rules to be explored below this point. The algorithm is called with initial arguments $Current = \{\}$ and $Available =$ *the set of all available conditions*. The predicate $solution(X)$ is true if and only if X is an antecedent of one of the impact rules sought in the search. For example, if the algorithm is invoked to discover the 1000 impact rules with the highest impact, then only the 1000 impact rules with the highest impact are solutions. The algorithm records at Step 2b all impact rules encountered that might be solutions. As rules are recorded at Step 2b, only the top n so far are retained. When the search terminates, those retained will be the n best impact rules from the search space.

The pruning rules used at Step 2b will vary depending on the objectives of the search. The pruning rules below

[2]It does not exactly halve the remaining search space as the root node has already been visited, as, depending upon the search technique, may have the node containing c, and hence these nodes should not be counted as part of the remaining search space.

[3]In rule search each condition can be considered a search operator. Formally, the search operator is the inclusion of the condition in the set of conditions associated with a node.

Table 1: The OPUS_IR algorithm

```
Algorithm:  OPUS_IR(Current, Available)

  1. SoFar := {}

  2. FOR EACH P in Available

     (a) New := Current ∪ {P}

     (b) IF pruning rules cannot determine that ∀x
         ⊆ Available:  ¬solution(x ∪ New) THEN

             record New → relevant stats

             OPUS_IR(New, SoFar)

             SoFar := SoFar ∪ {P}

         END IF

     END FOR
```

are utilized in the current work. The following definitions are used: $\overline{targ}$ is the mean of the target for all records; $value(x)$ is the value of the target for a record x; $cover(X)$ is the set of records for which the condition X is true; $ante(ir)$ is the antecedent of impact rule ir; $mean(ir)$ is the mean value of the target for records covered by $ante(ir)$; $impact(ir) = (mean(ir) - \overline{targ}) \times |cover(ante(ir))|$; and $IRule_n$ is the impact rule with the n^{th} highest mean (or impact if searching by impact) of those recorded so far at Step 2b. Note that for a superset C' of set of conditions C, $cover(C') \subseteq cover(C)$.

- For all searches, if New covers no records then no superset of New may be a solution as no superset of New may cover any records.

- When searching for the n impact rules with highest impact, after n rules have been recorded at Step 2b, if $\sum_{x \in cover(New):value(x) \geq \overline{targ}}(value(x) - \overline{targ}) < impact(IRule_n)$ then no superset of New may be a solution because the greatest possible impact of a subset of $cover(New)$ will be for the subset that excludes all records with value less than $\overline{targ}$. Thus $\sum_{x \in cover(New):value(x) \geq \overline{targ}}(value(x) - \overline{targ})$ sets an upper limit for the impact of a superset of New and no superset of New may be a solution if this value is less than the impact of the n^{th} best rule discovered so far.

- When searching for the n impact rules with the highest sum, after n rules have been recorded at Step 2b, if $\sum_{x \in cover(New):value(x) \geq 0} value(x) < sum(IRule_n)$ then no superset of New may be a solution because the greatest possible sum for a subset of $cover(New)$ will be for the subset that excludes all records with value less than 0. Thus $\sum_{x \in cover(New):value(x) \geq 0} value(x))$ sets an upper limit for the sum for a superset of New and no superset of New may be a solution if this value is less than the sum for the n^{th} best rule discovered so far.

- If a constraint, $minsup$, has been set on the minimum number of records that the antecedent of an impact

rule may cover, if $|cover(New)| < minsup$ then no superset of New may be a solution as for every $S \supset New$, $|cover(S)| \leq |cover(New)|$.

4.1 Completeness

THEOREM 1 (CORRECTNESS). *OPUS_IR records all solution impact rules.*

PROOF. That OPUS_IR records all solution impact rules follows from the completeness of the underlying OPUS algorithm. Without pruning OPUS will enumerate all combinations of conditions. Pruning removes from the available conditions only those that cannot participate in the antecedent of a solution impact rule in the search space below the current node. Hence, all solutions must be recorded. □

4.2 Relative complexity

The impact rule search space is exponential on the number of conditions and requires very effective pruning for efficient exploration. The dominant costs are scanning the data to determine whether a potential itemset is frequent or to accumulate its impact rule statistics. In the current work we assume that the data is loaded into the host computer's memory. We must either iterate through the records and for each record iterate through each candidate itemset, or iterate through the candidate itemsets and for each itemset iterate through each record. The former is the frequent itemset approach and the latter is employed by OPUS_IR. The frequent itemset approach was developed for processing data using database accesses. Hence minimizing the number of data accesses was important. When the data is loaded in memory, however, the computational costs of each strategy will be equivalent. The frequent itemset approach involves many itemset lookup actions whereas the OPUS_IR approach involves many record lookup actions.

Aumann and Lindell [3] propose that frequent itemsets be generated using the Apriori algorithm [2], and then impact rules be generated for the frequent itemsets. The frequent itemset approach first generates all itemsets $\{x_1, ..., x_{n-2}, x_{n-1}, x_n\}$ of size n such that both $\{x_1, ..., x_{n-1}\}$ and $\{x_1, ..., x_{n-2}, x_n\}$ are frequent itemsets. In a second step it then prunes from the resulting any itemset for which a subset is not a frequent itemset. This requires $n-2$ lookups as it is already known that $\{x_1, ..., x_{n-1}\}$ and $\{x_1, ..., x_{n-2}, x_n\}$ are frequent itemsets, and hence it is only necessary to check the subsets formed by deleting x_1 to x_{n-2}. The main computational burden imposed by this approach is not the lookup, but rather the memory requirement of storing all frequent itemsets of size n and size $n-1$. As we will see, the numbers of such itemsets can be prohibitive for many real world data sets.

THEOREM 2 (ITEMSET GENERATION). *With pruning on minimum cover only and assuming equivalent ordering of conditions, OPUS_IR also generates all itemsets $\{x_1, ..., x_{n-2}, x_{n-1}, x_n\}$ of size n such that both $\{x_1, ..., x_{n-2}, x_{n-1}\}$ and $\{x_1, ..., x_{n-2}, x_n\}$ are frequent itemsets.*

PROOF. Itemset $\{x_1, ..., x_{n-2}, x_{n-1}, x_n\}$ will be generated as a child of $\{x_1, ..., x_{n-2}, x_{n-1}\}$[4]. It will only be generated if $x_n \in Available$ for the call to OPUS_IR with $Current = \{x_1, ..., x_{n-2}, x_{n-1}\}$. This call is made by the activation of OPUS_IR with $Current = \{x_1, ..., x_{n-2}\}$. This activation of OPUS_IR will generate both $\{x_1, ..., x_{n-2}, x_{n-1}\}$ and $\{x_1, ..., x_{n-2}, x_n\}$. If a set of attribute values New is not a frequent itemset then the pruning rules can determine that no specialization of New can be a frequent itemset, and hence that $\forall x \subseteq Available : \neg solution(x \cup New)$. As a result, if $\{x_1, ..., x_{n-2}, x_n\}$ is not a frequent itemset then x_n will not be added to $SoFar$ and hence will not be passed as a member of $Available$ to the activation with $Current = \{x_1, ..., x_{n-2}, x_{n-1}\}$. If $\{x_1, ..., x_{n-2}, x_{n-1}\}$ is not a frequent itemset then there will be no call to OPUS_IR with $Current = \{x_1, ..., x_{n-2}, x_{n-1}\}$. Either way, $\{x_1, ..., x_{n-2}, x_{n-1}, x_n\}$ will not be generated. If both are frequent itemsets then the call will be made and x_{n-1} will be passed as a value in $Available$ and hence $\{x_1, ..., x_{n-2}, x_{n-1}, x_n\}$ will be generated. □

[4]See Figure 1. Note that the order is arbitrary under each search strategy, so equivalent orderings must be assumed, in this case: $n = 4$, $x_1 = d$, $x_{n-2} = c$, $x_{n-1} = b$, and $x_n = a$)

If other pruning rules are used in addition to pruning by minimum cover, OPUS_IR may consider even fewer itemsets. Irrespective of this issue, OPUS_IR avoids the large memory requirements of a frequent itemset approach by avoiding the need to store all frequent itemsets. Where the other constraints are sufficiently powerful, OPUS_IR further allows search without the need to specify a minimum cover, thus avoiding the risk that useful impact rules will fail to be discovered because they fall outside the specified arbitrary constraint on cover.

5. INTERESTINGNESS MEASURES

Association rule discovery will often generate very large numbers of associations. This imposes a large burden on the data analyst who must determine manually which of these associations are of interest. An active area of research is the identification of suitable measures of interestingness that might be applied to automatically filter associations [6, 11, 12]. This is also of importance for impact rule discovery.

Aumann and Lindell [3] suggested that distribution measures be used to measure interestingness for impact rules. Their examples include the deviation from that of the training set as a whole of the mean or variance of the target for the records selected by the antecedent. These measures will tend to identify groups of 'scientific' interest, in that the groups differ from the norm. In commercial applications, however, identifying groups that are different will often not be of primary importance. Rather, the primary objective will be to identify groups that contribute most (or least) to some outcome, such as profit or cost. Identifying a group with a high mean value does not equate to identifying a group that contributes a large amount to the total, as the group may be small. In consequence, I argue that aggregate measures will often be of greater interest, such as the sum or the impact (as defined in Section 2). These measures will evaluate the total contribution of the group selected by the antecedent. The sum will tend to be of interest where the target directly measures the end objective, such as the true profit from a transaction. The impact will tend to be of interest where the target is an intermediate variable, such as the income from a transaction. This is because it favors large groups for which the individuals each contribute more than average. While this paper considers search for impact rules that maximize these measures it is trivial to recast it

Table 2: Data sets

name	records	attributes	target
abalone	4177	8	Rings
covtype	581012	54	Elevation
housing	506	13	MEDV
ipums.la.99	88443	60	inctot
kddcup98	52256	480	TargetD
ticdata2000	5822	85	CARAVAN

Table 3: Results

data set	CPU time	item sets
abalone	0:0:1	2667
covtype	17:5:55	1158671
housing	0:0:1	3797
ipums.la.99	0:12:15	120787
kddcup98	0:17:46	958160
ticdata2000	0:13:7	2508494

to seek impact rules that minimize the measures.

6. EVALUATION

To demonstrate the applicability of the proposed algorithm, data sets with appropriate numeric targets were identified from the UCI knowledge discovery and machine learning repositories [4, 7]. Table 2 lists for these data sets the name, the number of records, the number of attributes (excluding the target), and the name of the target attribute. As the conditions in the antecedent must be Boolean, numeric attributes other than the target were discretized into three categorical values applying to as close as possible to equal numbers of records each. Each condition tested whether an attribute took a specific value.

OPUS_IR was implemented as a DOS program and run on an 800MHz PIII Windows PC with 256Mb RAM and approximately 650Mb virtual memory. This implementation was applied to each data set to find the 1000 impact rules with the highest impact on the target. Impact was chosen as the metric of interestingness as only for the *ipums.la.99* data set did the target have negative values and hence for the other data sets the sum would tend to be maximized by the impact rules with the greatest cover. Therefore, search by impact is the more challenging of the two search tasks. The number of conditions in the antecedent was limited to a maximum of 5. This was achieved by suppressing recursive calls to OPUS_IR below depth 5. Table 3 lists the CPU time (hours:minutes:seconds) and the number of item sets for which statistics were evaluated by the system for each of these tasks.

6.1 Example rules

To illustrate the technique the highest value impact rule for each data set is presented. Each rule is preceded by the summary population statistics for the data set. The results for the data sets from the KDD Repository are particularly notable. The rule for *ipums.la.99* selects a segment containing 5% of the data that accounts for over 25% of the target (total income). The rule for *kddcup98* also selects a segment containing 5% of the data. However, this rule accounts for over 99% of the target (amount donated). The rule for *ticdata2000* selects a segment containing 38% of the data. This segment accounts for 64% of the target (caravan insurance). Each of these rules identifies a relatively small group that contributes a disproportionately large amount of the total for the target variable.

Abalone: Rings: mean=9.9; min=1; max=29; sum=41493

Shell weight $> 0.2940 \rightarrow$ Rings: mean=12.0; min=6; max=29; sum=16680; coverage=0.333 (1390); impact=2927.2

Covtype: Elevation: mean=2959.4; min=0; max=3858; sum=1719423451

WA4=0 & ST04=0 & ST10=0 & ST11=0 & ST12=0 $\rightarrow$ Elevation: mean=3042.1; min=0; max=3858; sum=1449198640; coverage=0.820 (476381); impact=39430149.3

Housing: MEDV: mean=22.5; min=5; max=50; sum=11402

LSTAT<8.43 $\rightarrow$ MEDV: mean=31.0; min=12; max=50; sum=5239; coverage=0.334 (169); impact=1427.0

ipums.la.99: inctot: mean = 15121.7; min=-16451; max=383762; sum=1337406371

ftotinc>50000 & momloc=00 & nsibs=0 & relateg=01 & incwage>36000 $\rightarrow$ inctot: mean=79964.0; min=30001; max=383762; sum=357119224; coverage=0.050 (4466); impact=289585673.0

kddcup98: TARGET_D: mean = 0.8; min=0; max=200; sum=41018

0≤POP90C1≤99 & TARGET_B>0 & 0≤HPHONE_D≤1 & RFA_2R=L $\rightarrow$ TARGET_D: mean=15.5; min=1; max=200; sum=40889; coverage=0.050 (2638); impact=38947.7

ticdata2000: CARAVAN: mean=0.1; min=0; max=1; sum=348

PWALAND Contribution third party insurance (agriculture)=0 & PPERSAUT Contribution car policies=6 & PVRAAUT Contribution lorry policies=0 & PWERKT Contribution agricultural machines policies=0 & 0 ≤ AWAPART Number of private third party insurance ≤ 1 $\rightarrow$ CARAVAN: mean=0.1; min=0; max=1; sum=223; coverage=0.383 (2229); impact=125.8

6.2 The frequent itemset approach

Aumann and Lindell's [3] algorithm finds frequent itemsets and then calculates the desired statistics for the target with respect to each frequent itemset. This approach is limited by the requirement to store all frequent itemsets in memory during frequent itemset generation. Where the data is not sparse, the number of frequent itemsets will be very large and frequent itemset storage and access will dominate the computation. To compare the feasibility of this approach with that of OPUS_IR, Christian Borgelt's [8] Apriori implementation was compiled using the same compiler as used

Table 4: Frequent Itemset Generation

data set	CPU time	item sets	not found
abalone	0:0:1	2399	374
covtype	67:0:43	16451508	0
housing	0:0:1	24020	4
ipums.la.99		—	0
kddcup98		—	498
ticdata2000		—	0

in the experiments above and run on the same computer to generate the frequent itemsets of size no more than five with cover no less than 0.05. Note that this latter constraint means that only a subset of the search space explored by OPUS_IR is being explored by Apriori. The results are presented in Table 4. For each dataset this table presents the time taken (hours:mins:secs), the number of frequent itemsets discovered, and the number of impact rules discovered by OPUS_IR that are not included within the search space Apriori is exploring. These rules are not included in the search space because their antecedents cover less than 0.05 of the training data.

Note that this evaluation could not be completed for half the data sets either due to the computer running out of virtual memory (*kddcup98* and *ticdata2000*) or the excessive compute time required (*ipums.la.99* was halted after 148 hours). For the *abalone* data, approximately one third of the 1000 most interesting impact rules cover less than 0.05 of the data and hence are not found by the frequent itemset approach. For *kddcup98*, with minimum cover set to 0.05, even if the frequent itemset approach had not run out of memory it would not have found 498 of the 1000 most interesting impact rules.

This illustrates the dilemma that faces any attempt to apply the frequent itemset approach. A minimum cover must be specified. Setting it too low can make computation infeasible. Setting it too high can lead to missing important itemsets. As the *kddcup98* data illustrates, there may be no minimum cover value that satisfies both objectives.

7. CONCLUSIONS

Impact rules provide association rule like knowledge discovery for numeric data. This technique directly evaluates the impact of conditions on a numeric variable in a manner that cannot be satisfactorily emulated by association rules with discretization. This paper presents efficient techniques for impact rule discovery. It shows that these techniques can discover impact rules in contexts for which application of frequent itemset techniques is infeasible. It further demonstrates that these techniques avoid the need to set arbitrary constraints on the cover of the antecedent of an impact rule, hence avoiding the risk of failing to detect interesting rules.

8. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *SIGMOD-93*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB-94*, Santiago, Chile, 1994.

[3] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *KDD-99*, pages 261–270, 1999.

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

[5] R. J. Bayardo. Brute-force mining of high-confidence classification rules. In *KDD-97*, pages 123–126. AAAI Press, 1997.

[6] R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In *KDD-99*, pages 145–154, 1999.

[7] C. Blake and C. J. Merz. UCI repository of machine learning databases. [Machine-readable data repository]. University of California, Department of Information and Computer Science, Irvine, CA., 2001.

[8] C. Borgelt. *Apriori*. [Software]. School of Computer Science Otto-von-Guericke-University of Magdeburg, Magdeburg, Germany, 2001.

[9] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International, Belmont, CA, 1984.

[10] S. H. Clearwater and F. J. Provost. RL4: A tool for knowledge-based induction. In *TAI-90*, pages 24–30, Los Alamitos, CA, 1990. IEEE Computer Society Press.

[11] M. Kamber and R. Shinghal. Evaluating the interestingness of characteristic rules. In *KDD-95*, pages 263–266, 1995.

[12] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. Verkamo. Finding interesting rules from large sets of discovered associationd rules. In *Proc. 3rd Int. Conf. Information and Knowledge Management*, pages 401–407, 1999.

[13] S. Morishita and A. Nakaya. Parallel branch-and-bound graph search for correlated association rules. In *Proc. ACM SIGKDD Workshop on Large-Scale Parallel KDD Systems*, volume LNAI 1759, pages 127–144. Springer, Berlin, 2000.

[14] F. Provost, J. Aronis, and B. Buchanan. Rule-space search for knowledge-based discovery. CIIO Working Paper IS 99-012, Stern School of Business, New York University, NY, NY 10012, 1999.

[15] R. Rymon. Search through systematic set enumeration. In *KR-92*, pages 268–275, Cambridge, MA, 1992.

[16] R. Segal and O. Etzioni. Learning decision lists using homogeneous rules. In *AAAI-94*, Seattle, WA, 1994. AAAI press.

[17] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *SIGMOD-96*, pages 1–12, 1996.

[18] G. I. Webb. Recent progress in learning decision lists by prepending inferred rules. In *SPICIS'94*, pages B280–B285, Singapore, November 1994.

[19] G. I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

[20] G. I. Webb. Efficient search for association rules. In *KDD-2000*, pages 99–107, Boston, MA, 2000. The Association for Computing Machinery.