Efficient search for association rules

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

ABSTRACT

This paper argues that for some applications direct search for association rules can be more efficient than the two stage process of the Apriori algorithm which first finds large itemsets which are then used to identify associations. In particular, it is argued, Apriori can impose large computational overheads when the number of frequent itemsets is very large. This will often be the case when association rule analysis is performed on domains other than basket analysis or when it is performed for basket analysis with basket information augmented by other customer information. An algorithm is presented that is computationally efficient for association rule analyses during which the number of rules to be found can be constrained and all data can be maintained in memory.

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

Association Rule, Search

1. INTRODUCTION

The Apriori algorithm [2] and its derivatives [15, 11, 17] have become the de facto standard for discovering association rules. This paper presents an alternative approach to association rule discovery that may be more efficient when all data can be retained in memory and the number of candidate itemsets cannot be adequately constrained by considering individual itemsets in isolation. Given the current availability of very large memory machines, many potential applications of the new algorithm may satisfy the first constraint. Many data miners will consider their time more valuable than the cost of a few extra gigabytes of memory.

The Apriori algorithm relies on constraining the number of itemsets by considering features of itemsets in isolation, most commonly, by placing a lower limit on the frequency of an itemset, below which itemsets will not be considered. This is often feasible for simple basket analysis, as few combinations of products will be bought together in large quantities. However, even for basket analysis, the numbers of frequent itemsets may rapidly increase if simple basket analysis is augmented by considering socio-economic or other attributes of the customers. Augmenting simple basket analysis in this way can add much to the richness of the knowledge gained. However, if a customer description attribute is common to 50% of the customer base then that attribute will occur frequently with a large number of item combinations. Add a number of such attributes to the analysis and the number of frequent itemsets can rapidly expand to an extent where application of Apriori becomes infeasible.

The same problem occurs when association rule analysis is applied to domains other than basket analysis. Association rules can be a very valuable tool for discovering interesting inter-relationships between variables in many different types of domain, as they do not filter through a machine learning bias the rules that are presented to the user. This enables the user to identify the interesting rules rather than relying on a machine learning system to determine the rules of interest.

This paper describes how a search algorithm can take advantage of inter-association-rule constraints to find association rules efficiently.

2. BACKGROUND

Early approaches to identifying interesting rules from data were dominated by attempts to form small sets of rules for accurate classification of further previously unsighted data [9, 7, 13]. For the most part, borrowing from an elegant characterization of mining optimized rules by Bayardo and Agrawal [3], this activity can be characterized 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*.
- A *rule* consists of two conditions or combinations of conditions (typically conjunctions or, less frequently, disjunctions) called the *antecedent* and *consequent*. A rule with antecedent A and consequent C is denoted

To appear in *KDD-2000* Boston, MA, August, 2000.

as $A \rightarrow C$.

- The search is limited to exploring rules that have as consequent the values of a distinguished attribute, called the *class attribute*.
- The search seeks a set of rules that optimize some function of quality. The search is usually incremental, adding one rule at a time. The quality function usually attempts (often indirectly) to trade-off complexity against errors on the training set.
- In consequence, the rules selected tend to have antecedents that select subsets of the training set that are strongly dominated by a single class variable.

In the nineties, this research program took two divergent branches. On one hand, a number of researchers explored techniques for identifying large numbers of classification rules [4, 8, 10, 12, 14, 16]. This work was distinguished by the removal of the objective of using the rules for classification and hence of the requirement that a small number of rules be identified. Rather, all rules that satisfied some criterion of interestingness were sought. Interestingness was usually evaluated by some measure that led to identification of rules for which the antecedent identified subsets of the training set that were dominated by a single value of the class attribute, the intent being to predict the occurrence of that value.

The other branch was *association rule discovery* [1]. Association rule discovery differs in intent from most other rule discovery paradigms. While the other paradigms have concentrated on finding rules that are predictive of a single, preselected, class variable, association rule discovery has been motivated by finding rules that predict increased frequency of an attribute value, or collection of attribute values, without limitation on the values that may appear in the consequent of a rule. Association rule discovery can be distinguished by the aims of

- discovering all rules that satisfy a given set of constraints,
- an emphasis on processing large training sets, and
- allowing any available condition to appear as either an antecedent or consequent.

Due to its emphasis on analysis of large datasets, association rule discovery has concentrated on algorithms that process data via database access whereas the other branches of rule discovery have tended to concentrate on algorithms that retain all data in memory. This has led to the development of very different forms of algorithm. Association rule discovery algorithms have sought to minimize the number of passes through the data due to the very high time overheads that these imply when accessing a database. This is less of a concern when data is retained in memory.

Recent research has started to bring these two divergent branches of rule discovery research back together. Bayardo and Agrawal [3] present a variant of the OPUS search algorithm [18], developed in the context of classification rules research, to discover key rules of the type sought by association rule discovery. However, as is typical in classification rules research, their technique considers only the search space for a single consequent at a time, limiting its applicability in the most common association rule activity, market basket analysis, where it is often desirable to consider every product as a possible candidate rule consequent.

This paper presents techniques for employing the OPUS search algorithm for rule discovery where the search space encompasses rules for which the antecedent can contain any conjunction of available predicates and the consequent can be any single predicate. It is further distinguished by the ability to efficiently find a prespecified number of rules that maximize an arbitrary function measuring rule quality. This distinguishes the approach from typical association rule algorithms that explore all rules that satisfy prespecified constraints. This distinction is particularly significant. For dense search spaces, typical rule constraints may result in numbers of itemsets that make the Apriori approach infeasible. The ability to restrict search to a predefined number of target rules can allow the new algorithm to efficiently process such search spaces.

A major concern in developing association rule algorithms has been minimizing the number of database accesses that are required. I contend that the need to do this is reduced if the database is retained in main memory. I further contend that doing so is now feasible for a large range of data mining tasks due to the increase in the availability of very large memory computers. However, I recognize that there will always remain some tasks for which it is not feasible to retain a sufficient sample of cases in memory for acceptable association rule discovery. The techniques explored in this paper do not address that scenario.

2.1 The Apriori algorithm

The Apriori algorithm discovers association rules in two steps, utilizing the concept of an *itemset*. An itemset is a conjunction of conditions[1]. A *large itemset* is an itemset that occurs more frequently than a predefined minimum frequency. The Apriori algorithm exploits the observation that many common measures of the value of an association rule are functions of the frequency of LHS, RHS, and $LHS \wedge RHS$, where LHS and RHS represent, respectively, the itemsets for the antecedent and consequent of the association rule.

The two top-level steps of the Apriori algorithm are:

1. Find all large itemsets.
2. Generate association rules from the large itemsets.

The first stage plays two roles,

1. limiting the number of rules that need be explored to

[1] In basket analysis the relevant conditions are predicates, one for each of the available items, each of which is true iff the corresponding item was purchased by a customer, hence the name *itemset*.

those for which the union of the LHS and RHS occur with sufficient frequency, and

2. caching the relevant information about those itemsets, specifically their frequency, so that the search for association rules need not repeatedly access the database to compute them.

This strategy can be very successful at reducing the number of passes through the data base. Indeed, variants of the approach can reduce database access to two passes [15, 17]. However, where there are numerous large itemsets, the overheads of itemset maintenance and manipulation can severely impact upon the computational feasibility of the approach. A dramatic illustration of this is provided in Section 3, below. In this example, applying Apriori with standard settings to the Cover Type data set, with just 120 items, but with many items occurring very frequently, results in 14,567,892 large item sets. With so many large itemsets, management and manipulation of those itemsets creates a large computational burden.

2.2 The OPUS search algorithm

The move towards search for large numbers of classification rules resulted in the development of algorithms for efficient traversal of the search spaces involved. These initially relied upon assigning an arbitrary order to the conditions which was then used to structure the search space so that each combination of conditions was considered just once. A search space with four conditions (a, b, c, and d) structured in this manner is presented in Fig. 1.

Such a search space is exponential in size. If there are 10,000 conditions, a figure commonly exceeded in market basket analysis, the search space size is $2^{10,000}$. Clearly it will only be possible to explore such a search space if it can be pruned. Under fixed structure search, algorithms typically seek branches that cannot contain a solution[2], and prune those branches. Fig. 2 demonstrates the effect of pruning the branch for condition c from the fixed-structure search space illustrated in Fig. 1. As can be seen, this removes only one node from the search space.

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, the pruning illustrated in Fig. 1 may have resulted from a pruning rule identifying that no node containing c may contain a solution. In this case, the ideal outcome would be the removal from the search space of all nodes containing c, as illustrated in Fig 3. As can be seen, this approximately halves the remaining search space[3]

An elegant method of achieving this outcome is to reorder the search space so that any condition to be pruned at a node precedes all conditions not to be pruned. This is the OPUS$^{\mathrm{S}}$ strategy [18]. This algorithm guarantees that every pruning action approximately halves the remaining search space. The OPUS$^{\mathrm{O}}$ algorithm [18] extends OPUS$^{\mathrm{S}}$ for optimization search, using a heuristic that reorders the search space to maximize the amount of the space associated with the least promising search operator[4]. This is illustrated in Fig. 4. The OPUS algorithms have been demonstrated to support efficient complete search of a number of standard rule discovery search tasks[18].

A further approach to pruning is provided by *inclusive pruning* [19]. Whereas the (*exclusive*) pruning actions illustrated above involve excluding from the search space those nodes containing a particular condition, inclusive pruning results in the exclusion of all nodes that *do not* contain a given condition. Like exclusive pruning, each inclusive pruning action approximately halves the remaining search space.

2.3 Efficient search for association rules

Many frequent itemsets will relate to association rules that are not of interest. This might be addressed by placing additional constraints upon the itemsets that are considered. It is possible, although computationally expensive, to take account of the relationship between the antecedent and consequent of association rules that might be derived from an itemset, such as the potential *lift*[5]. However, this would require duplicating during the first stage much of the work of the second stage of the Apriori algorithm. More importantly, it is not possible to impose constraints that rely on relationships between association rules, such as only finding itemsets that could participate in the 1000 association rules with the highest lift. It will often be the case that the end users to receive the association rule reports will only be interested in considering a limited number of association rules. Selecting a prespecified number of those that maximize a particular measure will be desirable from the user's perspective and can be used to constrain a directed search for association rules.

Search for association rules can be tackled as a search process that starts with general rules (rules with one condition on the LHS) and searches through successive specializations (rules formed by adding additional conditions to the LHS). Such search is *unordered.* That is, the order in which successive specializations are added to a LHS is not significant. $A \wedge B \wedge C \rightarrow X$ is the same is $C \wedge B \wedge A \rightarrow X$. An important component of efficient search in this context is minimizing the number of association rules that need be considered. A key technique used to eliminate potential association rules from consideration is optimistic pruning. Optimistic pruning operates by forming an optimistic evaluation of the highest rule value that may occur in a region of the search space.

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

[3] 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.

[4] 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.

[5] Lift is a frequently utilized measure of association rule utility. The lift of an association rule $= \frac{|LHS \wedge RHS|}{|LHS|} / \frac{|RHS|}{n}$ where $|X|$ is the number of cases with conditions X and n is the total number of cases in the data set.

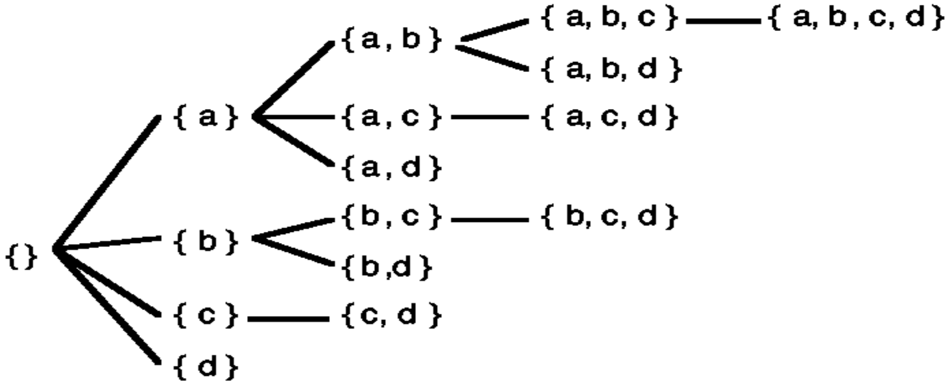

Figure 1: A fixed-structure search space

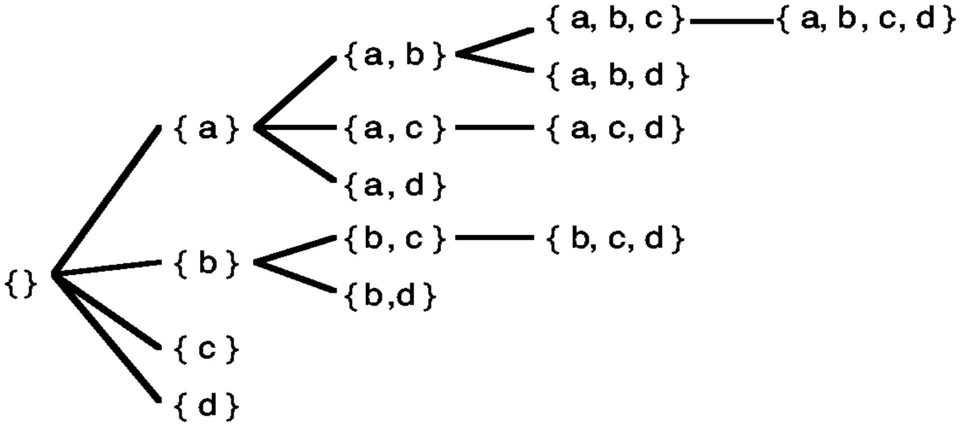

Figure 2: Pruning a branch from a fixed-structure search space

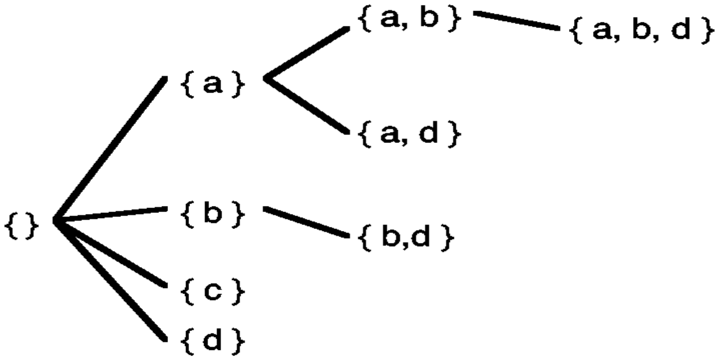

Figure 3: Pruning all nodes containing a single operator from a fixed-structure search space

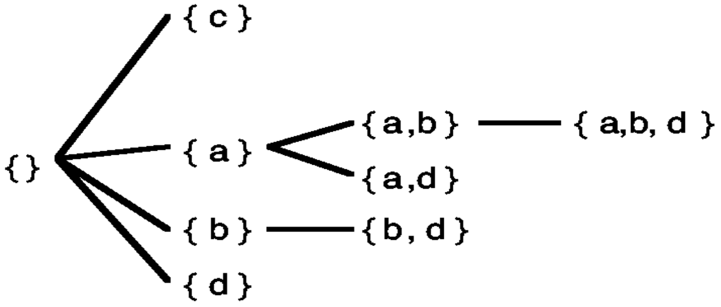

Figure 4: Pruning with a restructured search space

An optimistic evaluation is one that cannot be lower than the actual maximum value. If the optimistic value for a region is lower than the lowest value that can be of interest, then that region can be pruned. If search seeks the top m association rules, then it can maintain a list of the top m rules encountered so far during the search. If an optimistic evaluation is lower than the lowest value of a rule in the top m, then the corresponding region of the search space may be pruned. Other pruning rules may identify regions that can be pruned because they can contain only rules that fail to meet prespecified constraints such as:

- minimum support (the frequency in the data of the RHS or of the RHS and LHS in combination);
- minimum lift (as defined in footnote 5); or
- being one of the top m association rules on some specified criteria.

I use the term *credible rule* to denote association rules for which, at some given point in a search, it is possible that the rule will be of interest, using whatever criteria of interest apply for the given search.

If we restrict association rules to have a single condition on the RHS, two search strategies are plausible,

1. for each potential RHS condition explore the space of possible LHS conditions; or
2. for each potential LHS combination of conditions explore the space of possible RHS conditions.

The former strategy leads to the most straight-forward implementation as it involves a simple iteration through a straight-forward search for each potential RHS condition. However, this implies accessing the count of the number of cases covered by the LHS many times, once for each RHS condition for which an LHS is considered. At the very least this entails the computational overheads of caching information. At the worst it requires a pass through the data each time the value is to be utilized. While a pass through the data has lower overheads when the data is stored in memory rather than on disk, it is still a time consuming operation that must be avoided if computation is to be efficient.

These considerations mitigate in favor of the second strategy. We systematically explore the space of possible LHS condition combinations, searching from the general to the specific. During this process we track the set of conditions that can appear on the RHS of a credible rule in the search beyond the current point. We then organize the search to attempt to minimize the number of LHS condition combinations that are explored. A single pass through the data can be performed for every LHS combination during which all statistics are collected for both the LHS and each of the RHS conditions currently under consideration. We prune from the search space any regions of potential LHSs for which optimistic evaluation can ascertain no RHS can result in a credible rule. The relative efficiency of this approach against the Apriori approach will depend on the cost of a pass through the data (lower favoring the new direct search), the number of frequent itemsets (lower favoring Apriori), and the number of LHS combinations that must be explored (lower favoring direct search).

Table 1 displays the algorithm that results from applying the OPUS search algorithm [18] to obtain efficient search for this search task. The algorithm is presented as a recursive procedure with three arguments,

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

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

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

The initial call to the procedure sets CurrentLHS to {}, and AvailableLHS and AvailableRHS to the sets of conditions that are to be considered on the LHS and RHS of association rules, respectively.

Step 2(c)iiA records each credible association rule as it is evaluated. If the search seeks the m best rules on some metric, once m rules have been added at this step, as new rules are added, the rule with the lowest value on the metric can be removed from the table of best rules. A rule will not be credible if it fails other constraints, such as minimal strength, or, once the table is full, has lower value on the evaluation metric than the worst rule in the table of best rules.

Table 1: The OPUS search algorithm adjusted for search for association rules

```
Algorithm: OPUS_AR(CurrentLHS, AvailableLHS,
AvailableRHS)

 com CurrentLHS is the set of conditions in the LHS
     of the rule currently being considered.

 com AvailableLHS is the set of conditions that may
     be added to the LHS of rules to be explored
     below this point

 com AvailableRHS is the set of conditions that
     may appear on the RHS of a rule in the search
     space at this point and below

  1. SoFar := {}
  2. FOR EACH P in AvailableLHS
     (a) NewLHS := CurrentLHS $\cup$ {P}
     (b) AvailableLHS := AvailableLHS - P
     (c) IF pruning rules cannot determine that
         $\forall$x $\subseteq$ AvailableLHS: $\forall$y $\in$ AvailableRHS:
         $\neg$credible(x $\cup$ NewLHS $\rightarrow$ y) THEN
          i. NewAvailableRHS = AvailableRHS
         ii. FOR EACH Q in AvailableRHS
             A. IF credible(NewLHS $\rightarrow$ Q) THEN
                  record NewLHS $\rightarrow$ Q
             B. IF pruning rules determine that
                $\forall$x $\subseteq$ AvailableLHS: x = {} $\vee$
                $\neg$credible(x $\cup$ NewLHS $\rightarrow$ Q) THEN
                  NewAvailableRHS :=
                  NewAvailableRHS - Q
        iii. IF NewAvailableRHS $\neq$ {} THEN
               OPUS_AR(NewLHS, SoFar,
               NewAvailableRHS)
         iv. SoFar := SoFar $\cup$ {P}
```

Step 2c prunes conditions from the space of those explored on the LHS of a rule. Rather than exploring the space of possible LHS sets beyond the current one, optimistic techniques with low computational overheads should be employed. For example, if $|CurrentLHS \cup \{P\}|$ is less than minimum support then no rule in the relevant space of possible rules can achieve minimum support as all are specializations of $|CurrentLHS \cup \{P\}|$ and hence cannot have higher support.

Step 2(c)iiB prunes conditions from the space of those explored on the RHS of a rule. Optimistic rules with low computational overheads should be employed here also. For example, if $|CurrentLHS \cup \{P\}| = 0$ then no credible rule will exist in the relevant space of possible rules.

For both of the pruning steps, the exact pruning rules to be employed will depend upon the specific constraints for the search.

Without pruning this algorithm will systematically explore the entire search space. The pruning step removes from the search space below a node all and only those rules containing the identified condition. It follows, therefore, that the algorithm is complete, always finding the target association rules, so long as the pruning rules employed are correct.

This algorithm is based on OPUS$^{\mathrm{S}}$ rather than OPUS$^{\mathrm{O}}$. This is because the more efficient OPUS$^{\mathrm{O}}$ requires at least two passes through the available LHS conditions at each node of the search tree, one to select and sort the LHS conditions and the second to make the recursive call for each LHS with the appropriate second and third arguments. The overheads of doing this are excessive for this search task because an evaluation of which RHS conditions should be retained for each LHS would need to be performed in both loops. If there are a very large number of potential RHS conditions, either calculating this each time or caching the information between loops, will have very high overheads. For example, if there are 1,000 conditions then there might be 1,000 LHSs for each of which 1,000 potential RHS values need to be considered. Examining each of the resulting 1,000,000 possible combinations twice would clearly be undesirable as would caching such a large number of values. Thus, a single pass approach is employed that sacrifices the efficiencies to be gained from dynamic reordering on optimistic value but delivers far greater efficiency in processing a search node than would otherwise be possible.

3. AN EXAMPLE

The largest dataset in the UCI machine learning repository was subjected to association rule analysis using both the Apriori algorithm and the above OPUS search. The Cover Type data set was selected as the largest of the UCI machine learning repository datasets. A data set from the machine learning repository was used instead of one from the UCI KDD repository due to ease of access by the researcher. The Cover Type data set was already in a format that could be directly employed by both the Apriori and OPUS search software without further data manipulation.

The Cover Type data set was collected for the purpose of predicting forest cover type from cartographic variables only [5]. However, it is quite conceivable that association rule analysis might also detect interesting inter-relationships between those cartographic variables in addition to between them and the variable describing the forest cover. 581,012 cases are described by 55 attributes. The ten continuous valued attributes were discretized into three sub-ranges with as close as possible to equal numbers of cases within each subrange. The remaining 45 attributes were all binary. In consequence there were 120 attribute-values, each of which was treated as a separate condition for association rule analysis purposes. Note that this treatment results in many frequent items, as for each binary attribute at least one value must occur for $\geq 50\%$ of the cases.

The publicly available apriori system developed by Borgelt [6] was applied to the Cover Type dataset. This implementation of Apriori generates rules with a single RHS condition and multiple LHS conditions, thus exploring the same space of rules as the OPUS based algorithm. It generated 14,567,892 itemsets when employed with its default settings (maximum itemset size of 5; minimum coverage of 10% of the data for the LHS of a rule; minimum strength of 80%). The *coverage* of a set of conditions is the proportion of the training set for which the conditions are true. The *strength* of an association is the coverage of the union of the LHS and RHS divided by the coverage of the LHS. From the minimum LHS coverage and strength apriori can determine that only itemsets with coverage of 8% or higher need be generated. This required 96 hours and 44 minutes CPU time on a 350MHz PIII linux computer. It was not possible to complete the generation of all association rules as the file size limit was exhausted after 30,677,279 rules were generated.

The OPUS_AR algorithm was applied to the same data on the same computer. The same search space was explored to find the top 1000 associations on lift.

Four pruning rules were employed. To describe these we use the following abbreviations.

- $cover(s)$ is the coverage of the set of conditions s, the proportion of the training set for which the conditions in s are all true.
- $strength(LHS \rightarrow RHS)$ is the strength of association rule $LHS \rightarrow RHS$. $strength(LHS \rightarrow RHS) = cover(LHS \cup RHS)/cover(LHS)$.

The first pruning rule, used at step 2c, prunes any condition P for which $cover(NewLHS) < minLHScover$, where $minLHScover$ is the minimum allowed LHS coverage. No superset of such a LHS can exceed the minimum LHS coverage as the coverage of a superset of conditions must be no larger than the coverage of the original set of conditions.

The second pruning rule is used at step 2(c)iiB. It prunes any RHS condition Q for which $cover(NewLHS \cup \{Q\}) < minRHScover$, where $minRHScover = minLHScover \times min\_strength$ and $min\_strength$ is the minimum allowed value for association strength. This is the minimum allowed coverage for $LHS \cup RHS$ for any association. The justification for this rule mirrors that for the previous.

The next pruning rule is also used at step 2(c)iiB. This rules utilizes an optimistic assessment of the maximum value of association strength for a rule with Q as the consequent in the search space below the current node. First we determine the maximum number of specialization operations that may be applied to the current node to reach a node in the search space below the current node. $max\_spec = min(max\_LHS\_size - |NEWLHS|, |SOFAR|)$, where max_LHS_size is the maximum number of conditions allowed in a LHS. There may be no more specializations than there are conditions available to specialize by ($|SOFAR|$). Nor may there be more specializations than allowed by the constraint on the number of conditions permitted in a LHS.

Next we determine an upper limit on the maximum reduction in coverage that may result from the addition of any one condition to the LHS of an association in the search space below the current node. All associations in this search space cover subsets of the items covered by the association for the current node. Hence, no condition may remove more items from the cover of an association in that search space than it removes from the cover of the association for the current node. Hence $max\_cover\_reduction = max(cover(LHS) - cover(LHS \cup \{c\}) : c \in SOFAR)$.

The next step is to determine the minimum coverage for the LHS of a rule in the search space below the current node. It is not possible for the coverage to be reduced by more than $max\_spec * max\_cover\_reduction$. Nor is it possible for it to be reduced below the minimum allowed LHS coverage. Hence, $min\_cover = max(minLHScover, cover(LHS) - max\_spec \times max\_cover\_reduction)$.

If $min\_cover < cover(LHS \cup \{Q\})$ then the optimistic assessment of the maximum strength ($opt\_strength$) for an association with Q as consequent that may lie below the current node is 1.0 on the basis that the specializations may remove from the cover of LHS all cases that are not covered by Q.

Otherwise, $opt\_strength = cover(LHS \cup \{Q\})/min\_cover$, the result that would be obtained if all reduction in coverage removed cases covered by the LHS but not the RHS of the associations.

If $opt\_strength < min\_strength$, where $min\_strength$ is a constraint on the minimum allowed value for strength, then the RHS condition Q can be pruned.

The final pruning rule also applies at step 2(c)iiB. This rule determines an optimistic value for lift for associations in the search space below the current node that have Q as a consequent. Lift is maximized when strength is maximized. Thus, $opt\_lift = opt\_strength/cover(\{Q\})$. If $opt\_lift < min\_lift$, where min_lift is the minimum allowed lift, then the RHS condition Q can be pruned. Note that min_lift could be a global constraint on associations, but may also be determined dynamically. In the current application, min_lift was initialized to zero. However, once the target number of associations had been added to the table of best association rules, at step 2(c)iiA, *min_lift* was progressively updated to equal the minimum value of lift for a rule in the table. Hence, as the search progressed and the overall quality of the associations in the table improved, more stringent pruning could occur.

Using these pruning rules, a total of 384,312 association rules were evaluated and only 84,639 distinct antecedents considered. This took 48 minutes and 9 seconds CPU time. To find the top 100 associations on lift required the exploration of 204,264 association rules involving 51,678 distinct antecedents and took 26 minutes and 49 seconds CPU time.

4. DISCUSSION

The above example demonstrates that using OPUS search and pruning the search space on the basis of inter-relationships between itemsets, it can be feasible to perform efficient association rule analysis on data sets for which the Apriori approach is infeasible. Whether or not this is useful depends, of course, upon whether there are inter-itemset constraints that should be applied for the given association rule application. It seems plausible, however, that for many applications an upper-limit on the number of association rules to be generated will be appropriate, and this can be all that is required to enable efficient search.

Further search constraints, such as SC-Optimality [3], might usefully be employed to deliver even greater computational efficiency within the OPUS_AR framework.

That OPUS_AR has wider application than the single dataset examined herein is demonstrated by the commercial association rule discovery system *Magnum Opus*[6]. This system, that utilizes the OPUS_AR algorithm, is routinely employed for commercial association rule discovery from datasets containing millions of cases each described by tens of thousands of variables.

Association rule discovery has been firmly rooted in the domain of market basket analysis. However, prior to the popularization of market basket analysis, a number of machine learning researchers were exploring techniques with many similarities to association rule discovery. These researchers were exploring the use of complete or extensive search to form large rulesets in the belief that such rulesets could provide insight or other utility beyond that obtained from the small rulesets normally generated by machine learning systems [8, 12, 14, 16]. The current work can be viewed as a direct descendent of this research effort, extending it by utilizing the efficient OPUS search algorithm and by utilizing metrics of rule value developed within the field of basket analysis.

5. CONCLUSIONS

I have presented an algorithm for association rule analysis based on the efficient OPUS search algorithm. This approach is distinguished from the widely utilized Apriori algorithm by its ability to use inter-relationships between itemsets to constrain the number of itemsets that are considered. It is distinguished from a number of recent rule mining algorithms, that have been presented as alternatives to Apriori [4, 3, 10], by exploring associations containing all available conditions as consequents. However, the approach has the potential disadvantage, compared with Apriori, that it requires many more passes through the data. Where the data can be maintained in main memory this need not be a serious handicap. The availability of very large memory computers means that quite sizeable data sets can be retained in main memory. Where the data cannot be maintained in main memory, however, this approach to association rule discovery is unlikely to be feasible.

A simple example has been used to demonstrate the potential advantage of the new approach in some applications. Analysis of the Cover Type data set requires generation and analysis of 14,567,892 itemsets when the Apriori algorithm is utilized, even when the itemset size is restricted to five. In contrast, finding the 1000 association rules with the highest values of lift within the same constraints required evaluation of only 384,312 rules and 84,639 distinct antecedents. With the implementations employed, the OPUS search was completed with all 1000 rules identified in less than 50 CPU minutes while it took apriori more than 96 CPU hours just to generate the itemsets. This starkly illustrates the potential advantages of the new approach.

Acknowledgements

I am very grateful to Christian Borgelt for making his excellent implementation of the Apriori algorithm publicly available. I am also grateful to Jock A. Blackard and the UCI machine learning repository librarians Catherine Blake and Chris Mertz for providing access to the Cover Type data.

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA., 1996.

[3] R. J. Bayardo, and R. Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, 1999.

[4] R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4(2/3):217–240, 2000.

[5] J. A. Blackard. *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types.* PhD thesis, Colorado State University Department of Forest Sciences, Fort Collins, Colorado, 1998.

[6] Magnum Opus is distributed by Rulequest Pty Ltd, http://www.rulequest.com.

[6] C. Borgelt. apriori. (Computer Software) http://fuzzy.cs.Uni-Magdeburg.de/ borgelt/, February 2000.

[7] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–284, 1989.

[8] S. H. Clearwater and F. J. Provost. RL4: A tool for knowledge-based induction. In *Proceedings of Second Intl. IEEE Conf. on Tools for AI*, pages 24–30, Los Alamitos, CA, 1990. IEEE Computer Society Pres.

[9] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–129. Springer-Verlag, Berlin, 1983.

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

[11] J. S. Park, M.-S. Chen, and S. Y. Philip. An effective hash based algorithm for mining assiociation rules. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 175–186. ACM Press, 1995.

[12] 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.

[13] J. R. Quinlan. Generating production rules from decision trees. In *IJCAI 87: Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 304–307, Los Altos, 1987. Morgan Kaufmann.

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

[15] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases*, pages 432–444. Morgan Kaufmann, 1995.

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

[17] H. Toivonen. Sample large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pages 134–145. Morgan Kaufmann, 1996.

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

[19] G. I. Webb. Inclusive pruning: A new class of pruning rule for unordered search and its application to classification learning. In *Proceedings of the Nineteenth Australasian Computer Science Conference*, pages 1–10, Melbourne, January 1996.