Online generation of profile association rules

Charu C. Aggarwal
T.J. Watson Research Center
Yorktown Heights, NY 10598
charu@watson.ibm.com

Zheng Sun
Duke University,
Durham, NC-27706
sunz@cs.duke.edu

Philip S. Yu
T.J. Watson Research Center
Yorktown Heights, NY 10598
psyu@watson.ibm.com

Abstract

The results discussed in this paper are relevant to a large database consisting of consumer profile information together with behavioral (transaction) patterns. The focus of this paper is on the problem of online mining of profile association rules in this large database. The profile association rule problem is closely related to the quantitative association rule problem. We show how to use multidimensional indexing structures in order to perform the mining. The use of multidimensional indexing structures to perform profile mining provides considerable advantages in terms of the ability to perform very generic range based online queries.

Introduction

Association rules have recently been recognized as an important tool for knowledge discovery in databases. The problem of discovering association rules was first investigated in pioneering work in (Agrawal, Imielinski, et. al. 1993). Here we examine a database of records which consist of both customer profile (such as salary and age) and behavior (such as buying decision) information.[1]

The association rule problem was originally proposed for the case of binary itemset data (Agrawal, Imielinski, et. al. 1993). The intuitive implication of the association rule $X \Rightarrow Y$ is that a presence of the set of items X in a transaction also indicates a strong possibility of the presence of the itemset Y. The measures used in order to evaluate the strength of the rule are support and confidence. The *support* of a rule $X \Rightarrow Y$ is the fraction of transactions which contain both X and Y. The *confidence* of a rule $X \Rightarrow Y$ is the fraction of transactions containing X which also contain Y.

A considerable amount of research effort (Agrawal and Srikant 1994) (Savasere et. al. 1995) has been devoted to the problem of speeding up the itemset method for finding association rules from very large sets of transaction data. Several variations of this problem have been discussed in (Han and Fu 1995) (Srikant and Agrawal 1995) (Srikant and Agrawal 1996). The quantitative association rule problem (Srikant and Agrawal 1996) is a generalization of the problem of discovering association rules on sales transaction data, in which both categorical and quantitative attributes are allowed.

In this paper, we examine association between customer profile information and behavior information, referred to as *profile association* rules. The left hand side of the rules consists of quantitative attributes corresponding to customer profile information. Examples of such attributes are age and salary. The right hand side of the rules consists of binary or categorical attributes corresponding to customer behavior information. Examples of such attributes are buying beer or diapers. The objective of mining profile association rules is to identify customer profile for target marketing. Thus, profile association rules are a special case of quantitative association rules. It is the object of the paper to use this special structure in order to provide online mining capabilities. Furthermore, we will also show how to provide merged rules for different combinations of attributes in the consequent. We shall first describe an operator called "|" which indicates a combination of the behavioral attributes.

Let C correspond to a customer profile of quantitative attributes and $b_1 \ldots b_k$ correspond to behavioral attributes assuming values $v_1 \ldots v_k$. The rule $C \Rightarrow b_1 = v_1 | b_2 = v_2 | \ldots | b_k = v_k$ is true at a given level of support and confidence if and only if all of the rules $C \Rightarrow b_i = v_i$ for all $i \in \{1, \ldots k\}$ are true at that level of support and confidence. Thus, C represents the common profile of people who exhibit all of the behaviors $b_i = v_i$ for all $i \in \{1, \ldots k\}$. Note that if the set of rules $C_i \Rightarrow b_i = v_i$ holds for all $i \in \{1, \ldots, k\}$, then the rule $\cap_{i=1}^{k} C_i \Rightarrow b_1 = v_1 | b_2 = v_2 | \ldots | b_k = v_k$ may *not* necessarily hold. Thus, a straightforward postprocessing after finding rules for individual behavior types does not solve the above problem.

The rule $C \Rightarrow b_1 = v_1 | b_2 = v_2 | \ldots | b_k = v_k$ is different from the rule $C \Rightarrow b_1 = v_1, b_2 = v_2, \ldots, b_k = v_k$ in that the former tries to find the common profile of people exhibiting a set of behaviors, while the latter tries to

find the profile of the people such that each of them exhibit this set of behaviors. Often, enough support for the latter rule may not be present for the case of sparse behavioral data, while in fact the profile set C may strongly imply that set of behaviors.

We discuss a framework for online mining of profile association rules which provides the capability to issue queries with different sets of parameters such as support/confidence levels, profile ranges, and combinations of behavioral attributes. Thus, this method also shows how to integrate an OLAP environment with the problem of finding profile association rules. We shall see in the empirical section that the times for query responses are almost instantaneous for large amounts of data.

Description of the algorithm

We create a multidimensional index on the profile attributes of the data. This data structure was proposed by Guttman (Guttman 1984) for efficient retrieval of multidimensional data. The particular variant of index trees which we shall use for the purpose of our algorithm is the S-Tree (Aggarwal, Wolf et. al. 1997). The support of a node is equal to the fraction of data points which are encapsulated by the minimum bounding rectangle of that node. We call a node of the index tree to be *primary* if its support is above a preprocessing parameter called the *primary threshold.* Otherwise the node is referred to as *secondary.* Thus, the upper levels of the tree contain primary nodes, while the lower levels contain secondary nodes. At the lowest level, the secondary nodes point to the raw transactions. This preprocessing structure is designed in order to answer queries for which the support is above the primary threshold.[2] We store the following data at each node:

(1) We store the support of each node.
(2) For each of the behavioral attributes i, we store the support of particular values of behavioral attributes in the corresponding minimum bounding rectangle of the node provided that they are above the primary threshold. If the data is skewed and sparse, then we expect to keep only a few entries for each node. No behavioral information is maintained in the secondary nodes except at the leaf level which contain the individual transactions.

[2]The issue of limitation in the number of addressable queries is a pervasive one for most OLAP methods using the preprocess-once-query-many paradigm. In this case, this is not necessarily a major constraint, since online queries are expected to generate only a small number of rules which can be assimilated by an end user. We shall see later that for a primary threshold of p_c, there are only at most $2/p_c$ primary nodes. Thus, low enough primary threshold values can be chosen so that enough storage space would be available to store the behavioral information for these nodes. The use of a primary threshold for online generation of association rules was first introduced in (Aggarwal & Yu 1998).

Algorithm *BasicMining*$(T, (b_1, v_1), \ldots, (b_k, v_k))$
begin
  while any unvisited large node n exists **do**
    begin
    pick the next unvisited large node n in breadth first search order
    if node n is concentrated
      Generate rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$
    end;
end

Figure 1: The basic mining algorithm

An important difference with the traditional multidimensional index is that the fanout of the nodes are not constant. All primary nodes have just two children. This ensures that a maximum amount of support information is preserved, since the support of each node reduces by a factor of two with depth of the nodes. For secondary nodes, the fanouts are chosen so that they are packed as compactly as possible.[3] The binary fanouts at the higher level nodes do not lead to storage inefficiency as in regular R-Trees, because of the fact that the overall compactness is decided by the fanouts at the lower level secondary nodes which are more numerous. Another issue is the impact of the binary nature of the higher level primary nodes on the depth of the tree. When the primary threshold is p_c, the disjoint nature of nodes for point data (Aggarwal, Wolf et. al. 1997) ensures that the total number of nodes at the lowest level of primary nodes is $1/p_c$. Thus, the total number of primary nodes in the binary levels of the index tree is $2/p_c$. The depth of the subtree comprising the primary nodes is $\log_2(1/p_c)$. Since most of the limited number of primary nodes can be main memory resident, this does not lead to inefficiency at the time of retrieval.

Consider a user query for a set of behavior-value pairs $(b_1, v_1), (b_2, v_2) \ldots (b_k, v_k)$. We define a node of the index tree to be *large* with respect to the behavior-value pairs $(b_1, v_1), \ldots (b_k, v_k)$ at minimum support s if for each $i \in \{1, \ldots, k\}$ the number of points satisfying $b_i = v_i$ in that node is at least a fraction s of the total number of points.

A node of the index tree is defined to be *concentrated* at minimum confidence c with respect to the behavior-value pairs $(b_1, v_1), (b_2, v_2), \ldots (b_k, v_k)$ if for each $i \in \{1, \ldots, k\}$ at least a fraction c of the points in that node satisfy $b_i = v_i$.
A node of the index tree is said to be *diluted* at minimum confidence c with respect to the behavior-value pairs $(b_1, v_1), \ldots (b_k, v_k)$ if and only if it is not concentrated.

The basic mining algorithm traverses the index tree,

[3]An exact algorithm which can construct an index structure with differing fanouts at upper and lower level nodes is discussed in (Aggarwal, Wolf et. al. 1997). The modification to the algorithm in order to incorporate the information about the behavioral attributes is relatively straightforward.

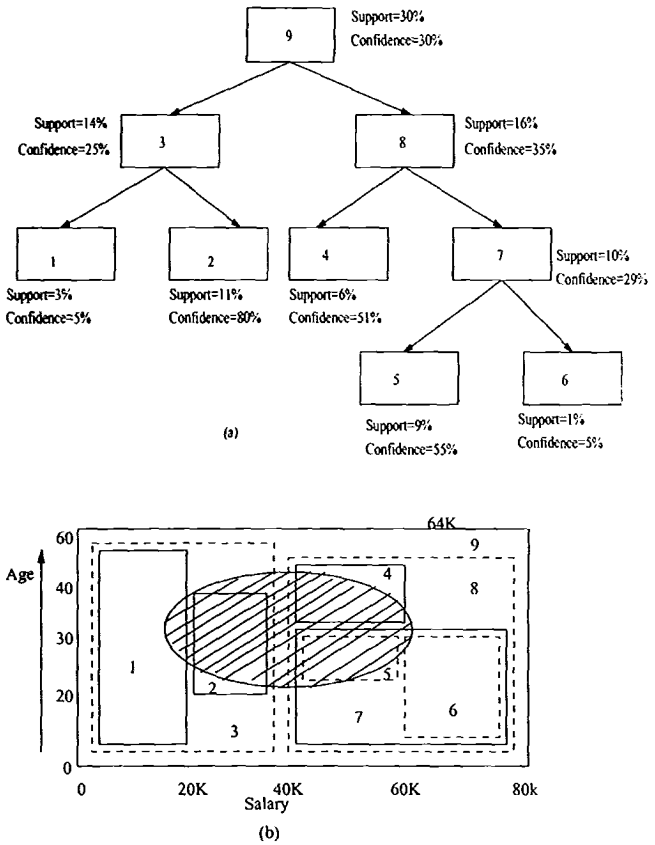

Figure 2: An example of the rule tree

and finds the nodes which are both large and concentrated. The rules from the bounding rectangles of these nodes are generated. Thus, the basic mining algorithm needs to use only the behavioral information at the primary nodes. The algorithm is illustrated in Figure 1. Consider the example illustrated in Figure 2. For the purpose of notation, we shall denote the bounding rectangle of node i by $B(i)$. We wish to find the profile association rules at a minimum support of 1% and minimum confidence of 50%. The behavioral attribute which is investigated is that for first-time home-buyers. In that case, the rules generated would be as follows:
$B(2) \Rightarrow$ FirstTimeHomeBuyer
$B(4) \Rightarrow$ FirstTimeHomeBuyer
$B(5) \Rightarrow$ FirstTimeHomeBuyer

As we can see, the contiguous shaded region of Figure 2(b) gets fragmented into a number of smaller rectangular regions, each of which corresponds to a generated rule. This is because there is no single rectangular region which approximately overlaps the entire shaded region. The purpose of the merging algorithm is to merge these fragmented regions into larger regions, and construct a rule-tree which provides the user with a better hierarchical tree-like view of the association rules generated. In this case, it will be necessary to use the the

Algorithm *MiningWithMerging*(*LargeNodesTree* : T, $(b_1, v_1) \ldots (b_k, v_k)$)
begin
{ Tree T is a copy of the subtree of the
original index tree containing only the large nodes }
for each large node n in postorder **do**
begin
case
(1) Node n is concentrated:
Generate rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$
(2) Node n has no children and is diluted:
Delete node n
(3) Node n has one child p and is diluted:
Delete node n. Set the parent of p to parent of n
(4) Node n has two children p_1 and p_2
and is diluted:
$n' =$*bounding*(p_1, p_2); replace node n by n'.
if n' is concentrated then
generate rule $B(n') \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$
end;
end

Algorithm *bounding*(p_1, p_2)
begin
return the minimum bounding rectangle of
all transactions in p_1 and p_2, with
updated values of support and confidence
end;

Figure 3: Generating rule trees

secondary nodes in order to recalculate the support of the merged regions.

The *merging algorithm* constructs a new rule tree by taking a copy of the index tree of large nodes and performing appropriate modifications on it. The idea is to delete those nodes which do not correspond to any rule, and then restructure the remaining nodes in the tree, to result in some new merged rules. This rule tree is specific to a user-specified set of behavior-value pairs $(b_1, v_1), (b_2, v_2), \ldots, (b_k, v_k)$. The value of k can vary from 1 to the total number of behavioral attributes. Thus, the rule tree can be built for any subset of behavioral attributes with corresponding values. As in the case of the basic mining algorithm, a node n in the rule tree contains the rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$ if and only if it is large and concentrated.

The algorithm works in a bottom-up fashion, by visiting the large nodes of the index tree in postorder. We assume that the tree T in the description of the algorithm in Figure 3 consists of only those nodes which are large. A node is modified or deleted only if it is diluted. If a node is diluted and has no large children then it is deleted. As a result of such deletions some node p will have only one child. In that case, the single child of that node replaces it. As a result, ths node will no longer be the *minimum* bounding rectangle of its children. When p is visited, (note that the postorder sequence of visiting the nodes ensures that

all descendants of p are visited before p is visited) its minimum bounding rectangle is readjusted to that of its children if it is diluted. This readjustment may result in p changing from being diluted to being concentrated. It is precisely these kinds of re-adjustments which result in larger concentrated regions. Thus, the "merging" algorithm proceeds by a series of deletions and re-adjustment operations on the minimum bounding rectangles of the nodes. The algorithm is illustrated in Figure 3. Note that the readjustment operation (indicated by the algorithm *bounding*(p_1, p_2)) requires a recalculation of the support and confidence of the behavioral attributes which are being queried. This requires a range query using the bounding rectangle of p_1 and p_2 on the original index tree. Typically, the support may be calculated using the entries for the behavioral attributes at the primary nodes. However, for some of the nodes intersected by this bounding rectangle, the corresponding entries are not present because of the primary threshold criterion. For those branches of the index tree, it may be necessary to access the secondary nodes up to the individual data points in order to recalculate support. The overhead for this operation is relatively small, since such branches are few in number if any, and most support information can be obtained directly from p_1 and p_2.

Empirical Results

We generated N data points, each having l antecedent and m behavioral attributes. For the purpose of the experiments, we use behavioral attributes, each of which may assume values from the set $\{1, \ldots, k\}$. We shall assume that the m behavioral attributes are denoted by $b_1, b_2, \ldots b_m$. Each of the l profile attributes are denoted by $C_1, C_2, \ldots, C_l$. We normalize the profile data in such a way that each of the attributes lies in the range $[0, 1]$. The individual data points are generated by first deciding the values of the profile attributes, and then deciding the values of the behavioral attributes.

The profile attribute values were generated by spreading the data points randomly over the profile subspace. Then, the value of the behavioral attributes are set. For each of the m behavioral attributes (say the ith behavioral attribute b_i), we pick a random number between 1 and k (say j). We generate a hypercube in the profile space, such that each side has a length which is distributed uniformly randomly in the range $[0, 0.2]$. For all points which lie inside this hypercube, a fraction f of points must satisfy the consequent condition $b_i = j$. The actual points which are chosen are arrived at by randomly picking off a fraction f of the points which lie inside the hypercube. Then we set the value of the attribute b_i to j for these points. The remaining points inside the hypercube may assume any of the values for b_i from 1 to k except j. The points *outside* the hypercube may assume any value from 1 to k. This method of generating hypercubic regions of concentration is useful in evaluating the quality of the rules generated.

For the purpose of this experiment, we choose $l = 2$, $m = 4$, and the value of f is chosen uniformly randomly in the range $[0.7, 0.95]$. We ran the experiments for cases when the number of data points N ranged from 5000 to $1,000,000$. We tested both the "basic mining" and the "mining with merging" versions of the algorithm. The following were the performance measures that we used:

(1) Speed of execution: We tested the time it took for the preprocessing and the online processing parts of the algorithm. As we see from Figure 4(a), the time for setting up the multidimensional index tree varies linearly with the size of the data. The online response times for a support of 0.8% on a database with $100,000$ points are illustrated in Figure 4(b). These times are so small as to be practically instantaneous, and are consistent with the hypothesis that the time required is of the order of $1/s$, where s is the minimum support.

(2) Rule Quality: For each behavioral concentration, let us define *nearest rule* as the rule in the final rule tree which matches most closely with the given concentration in behavior. In order to assess the quality of the rules generated, we tested how well this rule overlapped with the true regions of behavioral concentration. Let us consider N_{actual} be the number of data points in a rectangular concentration corresponding to the consequent of a given rule. Let $N_{discovered}$ be the number of data points in this rectangle which were actually found by the rule, and let N_{false} be the number of data points outside the rectangle which are mistakenly found be the rule. Note that $N_{discovered}$ is always bounded above by N_{actual}. Thus, we have two measures in order to decide the quality of a rule:

$$\text{Incompleteness Ratio} = 1 - \frac{N_{discovered}}{N_{actual}} \quad (1)$$

$$\text{Spurious Ratio} = \frac{N_{false}}{N_{actual}} \quad (2)$$

The incompleteness and spurious ratios are illustrated in Figures 4 (c) and (d). These ratios are small for reasonably low support values. Given the quick response times of the algorithm in the range of such support values, it is practically possible to get this high quality of performance for most online queries. The process of merging improved both the incompleteness and spurious ratios. This is because of the better bounding rectangles obtained after the process of deletions and readjustments.

Conclusions and Summary

In this paper, we discussed the problem of online mining of profile association rules. Such rules may be very useful in developing relationships between consumer profiles and behavioral information. We discussed how to use multidimensional indexing to generate profile association rules in online fashion. An additional advantage

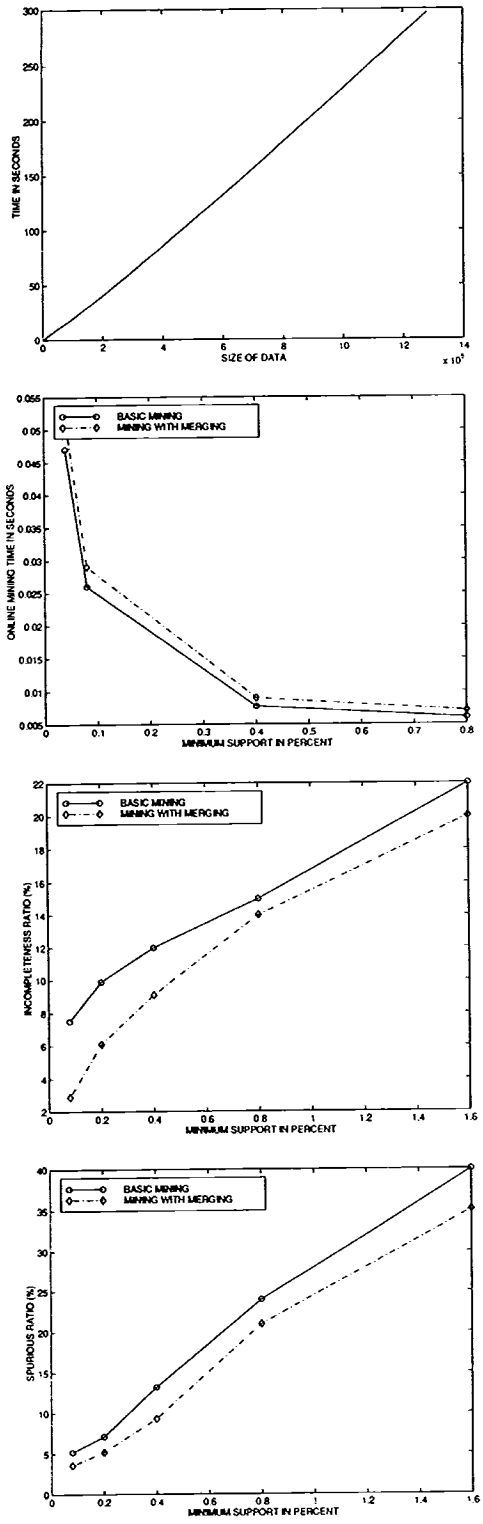

Figure 4: (a) Setup Time for Multidimensional Index Tree (b) Variation of online execution time (c) missing ratio (d) spurious ratio with minimum support

of using indexing structures is that it is possible to specify specific profile ranges and behavioral attributes for which it is desired to find the rules.

References

Aggarwal C. C., and Yu P. S. 1998. Online Generation of Association Rules. *Proceedings of the International Conference on Data Engineering*, Orlando, Florida.

Aggarwal C. C., Wolf J. L., Yu P. S., and Epelman M. 1997. The S-Tree: An efficient index for multi-dimensional objects. *Proceedings of the International Symposium on Spatial Databases.* pages 350–370, Berlin, Germany.

Agrawal R., Imielinski T., and Swami A. 1993. Mining association rules between sets of items in very large databases. *Proceedings of the ACM SIGMOD Conference on Management of data,* pages 207–216, Washington D. C.

Agrawal R., and Srikant R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases,* pages 478–499.

Guttman A. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. *Proceedings of the ACM SIGMOD Conference,* 47–57.

Han J. and Fu Y. 1995. Discovery of Multiple-Level Association Rules from Large Databases. *Proceedings of the 21st International Conference on Very Large Data Bases.* Zurich, Switzerland, pages 420–431.

Mannila H., Toivonen H., and Verkamo A. I. 1994. Efficient algorithms for discovering association rules. *AAAI Workshop on Knowledge Discovery in Databases,* pages 181–192, Seattle, Washington.

Savasere A., Omiecinski E., and Navathe S. 1995. An Efficient Algorithm for Mining Association Rules in Large Data Bases. *Proceedings of the 21st International Conference on Very Large Data Bases.* Zurich, Switzerland, pages 432–444.

Srikant R., and Agrawal R. 1995. Mining Generalized Association Rules. *Proceedings of the 21st International Conference on Very Large Data Bases,* pages 407–419.

Srikant R., and Agrawal R. Mining quantitative association rules in large relational tables. *Proceedings of the 1996 ACM SIGMOD Conference on Management of Data.* Montreal, Canada.