Fast Algorithms for Online Generation of Profile Association Rules

Charu C. Aggarwal, *Member*, *IEEE*, Zheng Sun, and Philip S. Yu, *Fellow*, *IEEE*

Abstract—The results discussed in this paper are relevant to a large database consisting of consumer profile information together with behavioral (transaction) patterns. We introduce the concept of profile association rules, which discusses the problem of relating consumer buying behavior to profile information. The problem of online mining of profile association rules in this large database is discussed. We show how to use multidimensional indexing structures in order to actually 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.

Index Terms—Association rules, data mining.

1 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 [4]. 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. This paper deals primarily with finding associations between the profile and behavior attributes in these databases. We refer to this mining problem as the *profile association rules* problem.

The problem of finding association rules takes on different forms, depending upon the nature of the underlying database. In sales transaction databases, the nature of the information is in the form of basket data comprising the set of items which a consumer may buy in a transaction. In such cases, the association rule problem is that of finding how the presence of one item in a transaction may influence the presence of another. Consider the rule $X \Rightarrow Y$, where X and Y are two sets of items. The intuitive implication of the association rule is that a presence of the set of items X in a transaction set also indicates a possibility of the presence of the itemset Y. Two notions for establishing the strength of a rule are those of *minimum support* and *minimum confidence*, which were first introduced in [4].

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. Thus, if we say that a rule has 90 percent confidence, then it means that 90 percent of the tuples containing X also contain Y.

Starting with pioneering work in Agrawal et. al. [4], a host of work [5], [14], [16], [8], [9], [15] has been done in this area with a focus of finding association rules from large sets of transaction data. The primary idea proposed in [4] was an itemset approach in which first all large itemsets are generated, and then these large itemsets are used in order to determine data dependencies. Subsequent work has primarily concentrated on this approach.

The itemset method is as follows: Generate all combinations of items that have fractional transaction support above a certain user-defined threshold called *minsupport*. We call all such combinations *large itemsets*. Given an itemset S satisfying the support constraint, we can use it to generate rules of the type $S - X \Rightarrow X$ for each $X \subset S$. Once these rules have been generated, only those rules above a certain user-defined threshold, called *minconfidence*, need to be retained.

The quantitative association rule problem first proposed in [17] is a generalization of the problem of discovering association rules on sales transaction data. The conventional association rule problem considers only Boolean attributes, e.g., buying or not buying an item. In the quantitative association rule problem, the nature of the data is generalized such that categorical and quantitative data are allowed. The categorical data corresponds to the attributes like sex, region, or ethnic information. The quantitative data may consist of attributes such as age and salary. It is possible to convert the quantitative association rule problem into an instance of the Boolean association rule problem by discretizing the data appropriately [17]. An example of a quantitative association rule is as follows:

$$\mathrm{Age}[30-35], \mathrm{Sex} = \mathrm{Female} \Rightarrow \mathrm{FirstTimeHomeBuyer}.$$

In this case, the support of a rule corresponds to the percentage of the records which satisfy both the antecedent and the consequent conditions. Correspondingly, the confidence of the rule corresponds to the percentage of the records satisfying the antecedent condition which also satisfy the consequent condition. Such rules may often be useful in making business decisions in large demographic databases. The profile association rule problem is closely related to the quantitative association rule problem. Consequently, we shall discuss some features of this problem.

- *C.C. Aggarwal and P.S. Yu are with IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532. E-mail: {charu, psyu}@us.ibm.com.*
- *Z. Sun can be reached at 18837 Biarritz Ct., Saratoga, CA 95070. E-mail: sunz@cs.duke.edu.*

Manuscript received 23 July 1998; revised 3 July 2001; accepted 22 Oct. 2001. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 107028.

An important notion which has been introduced in [17] is that of *interesting* association rules. A rule $X \Rightarrow Y$ is said to be r-interesting with respect to a generalization of the rule $\overline{X} \Rightarrow \overline{Y}$ if and only if the support and confidence of the former rule is at least r times the expected support and confidence based upon the assumption of statistical independence and the strength of the latter rule.

The notion of clustered association rules was first developed by Lent et al. in [11] with reference to the problem of finding quantitative association rules in the data. The idea in clustering is to bunch together the rules corresponding to adjacent ranges. For example, consider the following two rules:

$$\text{Age}[30] \Rightarrow \text{FirstTimeHomeBuyer},$$
$$\text{Age}[31] \Rightarrow \text{FirstTimeHomeBuyer}.$$

The two rules may be clustered to make the following rule: $\text{Age}[30-31] \Rightarrow \text{FirstTimeHomeBuyer}$. This kind of clustering is useful in order to provide a compact representation. The form of the rules discussed in that paper is one in which the quantitative attributes are on the left-hand side, and a single categorical attribute on the right. For future reference, we shall refer to the problem of quantitative association rules being divided into too many smaller rules as the "rule fragmentation problem." Another related notion is that of "granularity:" How small should the divisions of the quantitative ranges be so as to not result in too many rules and yet provide an accurate description of the rules themselves. 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 customer profile information, such as age, salary, years of education, martial status, etc. The right-hand side of the rules consists of customer behavior information, such as buying milk, diaper, beer, etc. The rules need to satisfy a given support and confidence requirement as in conventional association rules. The objective of mining profile association rules is to identify customer profile for target marketing, where the support indicates the size of the potential target market and the confidence gives the effectiveness of the rule. In order to discuss the notion of profile association rules, we shall introduce the following new operator called "|:"

Definition 1.1. *Let C correspond to a customer profile. 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, in some sense, C represents the common profile of people who exhibit all of the behaviors $b_i = v_i$ for all $i \in \{1,\ldots k\}$. Such rules often have the following form:

$$C_1[l_1 \ldots u_1] \cap C_2[l_2 \ldots u_2] \ldots\ldots$$
$$C_j[l_j \ldots u_j] \Rightarrow b_1 = v_1|b_2 = v_2|\ldots|b_h = v_h.$$

Here, C_i is the ith profile attribute and $[l_i \ldots u_i]$ denotes the range for the ith attribute. It can either be a quantitative attribute, such as salary or age, or a categorical attribute, such as sex or marital status. b_j is the jth behavior attribute and v_j is its value. The behavior attribute can take a simple Boolean value to indicate the occurring or nonoccurring of an event (such as buying an item), or multiple categorical values. C_i is referred to as the antecedent item of the rules and b_j is referred to as the consequent item. The rule indicates that among people with $\{l_i \le C_i \le u_i$, for $1 \le i \le k\}$, those with attribute b_j equal to v_j, for any $j(1 \le j \le h)$, satisfy the confidence and support requirement. That is to say, the rule identifies the common profile showing the behaviors $b_1 = v_1$, $b_2 = v_2, \ldots, b_h = v_h$ as $\{l_i \le C_i \le u_i$, for $1 \le i \le k\}$. It may be noted that there may not be many *individual people* who show each and every one of these behaviors, but that, for each given behavior, a significantly large percent of the people within the profile set exhibit it. For example, consider the following rule:

$$\text{Age}[30-40] \cap \text{Salary}[80\text{K}-125\text{K}] \Rightarrow$$
$$(\text{buying projection TV})|(\text{buying high-end PC}).$$

This rule indicates that, among people between the ages of 30 and 40 and salaries between 80K and 125K, those buying a projection TV (respectively, high-end PC) will satisfy the confidence and support requirement. That is to say, the common profile or characteristics shared by a group of people buying either a projection TV or a high-end PC and satisfying the given support and confidence requirement are between the ages of 30 and 40 and salary between 80K and 125K. Note that the profile association rule does not indicate that, among people between the ages of 30 and 40 and salaries between 80K and 125K, those buying *both* a projection TV and a high-end PC will satisfy the confidence and support requirement. To find out whether people are buying both a projection TV and a high-end PC together can be done through conventional association rules mining. Profile association only determines the common profile characteristics related to a specific set of customer behavior under a given confidence and support requirement. When there is only one consequent item on the right-hand side, the rule is similar to the clustered association rule problem in [11] or a special case of the quantitative association rules in [17]. For the general case with multiple consequent items, the profile association rule is not a special case of either the quantitative association rules or clustered association rules. In this paper, we will develop a method to derive profile descriptions based on the set of customer behavior under a given support and confidence requirement. We note that, although the formalism allows for several behavioral attributes, the formalism proposed is such that each attribute-value pair must be interesting throughout a selected region. However, when a user tries to identify interesting regions for several attributes $(b_1 = v_1|\ldots|b_n = v_n)$, the system does *not* behave as the compound system corresponding to the consequent $(b_1 = v_1 \ldots b_n = v_n)$ since the support/confidence requirements of the two formalisms are different. In fact, if there are a large number of attributes in the consequent, the latter formalism is unlikely to meet the support requirement, whereas the former will usually meet the support

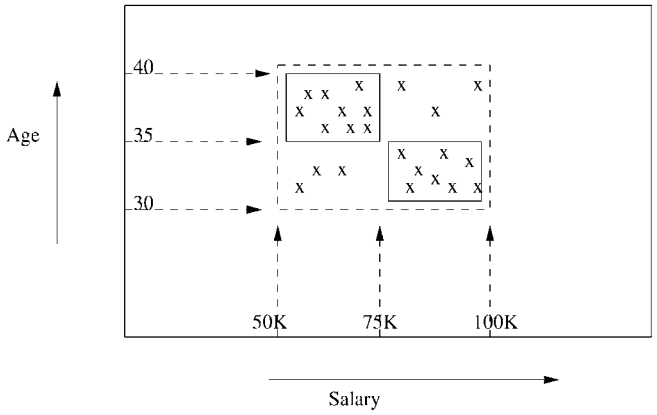


Fig. 1. Illustrating the Nature of Hiearchical Clustering.

requirement since it is judged separately for each attribute.

To present the profile association rules succinctly, clustering needs to be done as in [11], although the online approach proposed here based on multidimensional indexing is quite different from the approach adopted in [11] which performs the clustering on each consequent attribute separately. Here, we further introduce the notion of hierarchical clustering of profile association rules. Often, hierarchical clustering may be useful in providing a user with a better view of the rules.

Consider the example illustrated in Fig. 1. Suppose that the data is on the following dimensions: age and salary. The points marked "x" correspond to first-time home buyers. We can see from Fig. 1 that people who lie in the $[\text{Age}, \text{Salary}] = [30\text{-}40, 50\text{K} - 100\text{K}]$ have a greater propensity to be first-time home buyers. However, people in the $[\text{Age}, \text{Salary}]$ range of $[30\text{-}35, 75\text{K} - 100\text{K}]$ or $[35\text{-}40, 50\text{K} - 75\text{K}]$ have an even greater propensity to be homebuyers. We would like to be able to mine the hierarchical nature of this information generalizing the notion of interesting rules introduced in [17]. However, because of the online nature of the technique, the method is much faster than quantitative association rule algorithms from the user perspective which work in batch mode.

1.1 Contributions of this Paper

This paper discusses a model for online mining of profile association rules. This approach relies on indexing methods to actually perform the mining of the rules. Specifically, the contributions of this work are as follows:

1. **Framework**. We propose a notion, called profile association rules, and discuss how to use multidimensional indexing structures in order to generate such rules. A useful feature is that, because of the indexing structure, it is possible to find rules which are such that the quantitative profile attributes lie within certain user-specified ranges.
2. **Interesting Rule Determination.** We discuss the issue of finding interesting profile association rules with greater than expected confidence value based under the assumption of statistical independence *as well as* the other rules present in the output. For example, if a rule $\text{Age}[20-30] \Rightarrow \text{FirstTimeHomeBuyer}$ is present in the output, then a rule such as $\text{Age}[20-25] \Rightarrow \text{FirstTimeHomeBuyer}$ is reported only if it is r-interesting with respect to the former rule and other general rules in the output. In other words, the hierarchy is taken into account while generating the rules.
3. **Handling the granularity problem.** In order to develop rules from continuous data of profile attributes, it becomes necessary to discretize the data into several disjoint ranges. Traditional algorithms for quantitative association rule generation discretize the data based upon user-specified values of *minsupport*. Dividing into ranges which are too large may result in loss of useful information, while dividing into ranges which are too small may result in too many rules. The natural hierarchical organization of an indexing structure helps us in developing association rules at different levels of granularity.
4. **Finding association rules in online fashion.** We provide the ability to find the association rules in online fashion. It is possible to issue queries with different sets of parameters such as support/confidence levels, profile ranges, and consequent attributes. This is possible because a multidimensional index tree structure is used in order to support queries. Thus, this method also shows how to integrate an OLAP environment with the problem of finding profile association rules.

This paper is organized as follows: In the next section, we discuss the process of building a multidimensional index structure. In Section 3, we discuss the basic mining algorithm which uses a hierarchical traversal of the index tree in order to find the rules. In Section 4, we show how to use a merging algorithm in order to find the rules. In Section 5, the method is refined still further so as to provide rules with much better accuracy. This algorithm will be referred to as the "deeper mining algorithm" since it uses nodes at much lower levels of the index tree. The process of finding hierarchically interesting association rules is illustrated in Section 6. The algorithm is tested empirically in Section 7, while a conclusion and summary is presented in Section 8.

2 Building an Index Structure

In this section, we discuss an overview of the multidimensional index structure which is used to find profile association rules. This is necessary in order to provide the ability to support online queries. We first build a multidimensional index on the profile attributes of the data, so as to be able to answer queries within a specified query box easily. The required data structure for this purpose is the R-Tree [7].

R-trees are a multidimensional extension of B-Trees. The data structure is in the form of a height balanced tree in which each leaf node contains between m and M index records, and each internal node except for the root contains between m and M children. Each leaf node of the R-Tree contains record entries which are of the form

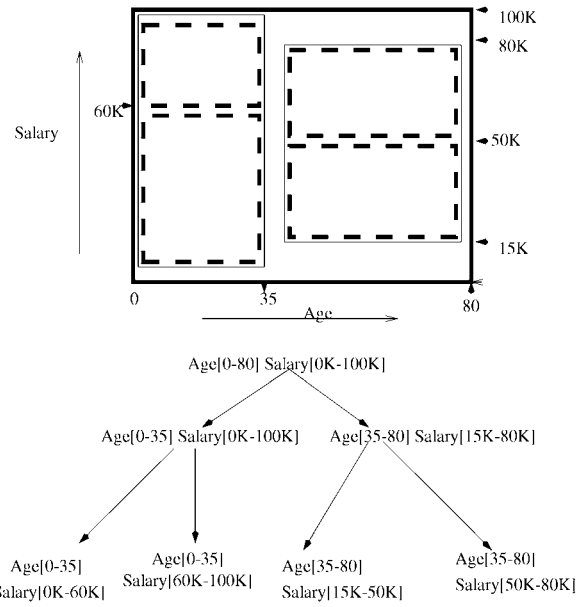


Fig. 2. An Example of the Quantitative Partitioning.

$(C, recordidentifier)$. In this case, C is the *coordinate* of the corresponding record. The structure of an internal node entry is of the form $(I, pointer)$. In this case, I is the minimum bounding rectangle, while $pointer$ points to a child node. For the purpose of our application, we will also need to store information about the consequent attributes in the nodes of the R-Tree.

Further, each node also carries information about the behavioral attributes for the data points enclosed. Specifically, we maintain the following information:

1. We maintain the number of data points within that bounding box.
2. For each of the behavioral attributes i, we maintain the fraction of the entries in that box which take on particular values for that behavioral attribute.

There is another important modification which we make to the traditional R-Tree structure for the purpose of this algorithm. The difference lies in the degree of the nodes in the tree. In the case of the R-Tree, it is necessary to pack the index nodes as compactly as possible for efficiency in representation. Thus, typical degrees of nodes may be of the order of 50. Unfortunately, this may result in considerable loss of information for the purpose of data mining. This is because the support values of the nodes at successive levels of the index tree are cut down by a factor of 50 as well. Thus, it is desirable to have a degree of 2 for the index tree which is generated. On the other hand, it may not be possible to store the entire data set as a binary tree because of space constraints. The solution is to have high degree for the lower-level nodes in order to improve the page utilization in the nodes which matter the most. An example of such a tree with two levels is illustrated in Fig. 3. An exact algorithm which successively collapses the lower-level nodes of a binary tree in order to convert it into a two-level tree may be found in [3]. The space requirements for this index structure are dominated by the number of leaf-level nodes and may be found in [3]. Note that the nodes

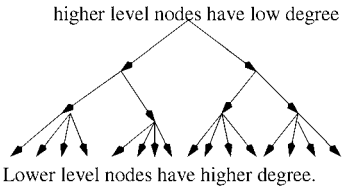


Fig. 3. A Two-Level Hierarchical Tree Structure.

which have a degree higher than two will have extremely low values of support. *Thus, for the purpose of the algorithms assume that the index tree which is generated is a binary tree.*

Consider an example in which there are two quantitative dimensions: age and salary. An example of the possible partition of the data is illustrated in Fig. 2. The age attribute ranges from 0 to 80, while the salary attribute ranges from 0 to 100K. The first partitioning of the data is along the age dimension, and it partitions the data into the following [Age, Salary] regions: [Age, Salary]=[0-35, 0K-100K] and [35-80, 15K-80K]. Note that, although the salary dimension lies in the range [0K-100K], the second component of the division shows salary in the range [15K-80K]. This is because there are no transaction data in the ranges [Age, Salary]=[35-80, 80K-100K] and [Age, Salary]=[35-80, 0K-15K]. This division takes place recursively along each dimension. In the diagram, we have not shown the entire binary tree down to the individual leaf nodes. We show a few of the upper levels of the tree.

3 The Basic Mining Algorithm

We shall first discuss the basic mining algorithm which uses the nodes of the multidimensional tree in order to generate the rules. As we shall see, this may result in a large and somewhat fragmented rule set. Consider the case when we generate all the rules corresponding to the attributes $\{b_1, \ldots b_k\}$ assuming specific values. More specifically, let us consider the case in which the consequent is $b_1 = v_1 | b_2 = v_2 | \ldots | b_k = v_k$. It is assumed that we wish to generate all the rules which have minimum support s and minimum confidence c.

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.

We define a node of the index tree 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. In other words, some index $i \in \{1, \ldots, k\}$ exists such that less than a fraction c of the points in that node satisfy $b_i = v_i$.

Once the above results have been established, it is relatively easy to describe the basic mining algorithm which traverses the index tree and finds the nodes which are both large and concentrated. The rules from the bounding rectangles of these nodes are generated.

Algorithm *BasicMining*$(T, (b_1, v_1), \ldots, (b_k, v_k))$
begin
{ The large nodes will be visited in breadth first search order }
$B(n)$ denotes the bounding box for node n
for 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

Fig. 4. The Pseudocode for the Basic Mining Algorithm.

The algorithm is illustrated in Fig. 4. The rules thus generated can also be presented to a user in their natural tree-like hierarchy. (For example, the rule $\text{Age}[10-50] \Rightarrow \text{FirstTimeHomeBuyer}$ bears a hierarchical relationship with the rule $\text{Age}[20-30] \Rightarrow \text{FirstTimeHomeBuyer}$.)

Consider the example illustrated in Fig. 6. 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 percent and minimum confidence of 50 percent. The behavioral attribute which is investigated is that for first-time home-buyers. (In other words, the numbers corresponding to the support and confidence are for those points in the database which correspond to first-time home-buyers.) In that case, the rules generated would be as follows:

$$B(2) \Rightarrow \text{FirstTimeHomeBuyer},$$
$$B(4) \Rightarrow \text{FirstTimeHomeBuyer},$$
$$B(5) \Rightarrow \text{FirstTimeHomeBuyer}.$$

As we can see, the contiguous shaded region of Fig. 6b 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 view of the association rules generated. We shall proceed to describe this algorithm in the next section.

4 Mining with Merging

The merging algorithm constructs a new rule tree by performing appropriate modifications on the index tree. The idea is to delete those nodes in the index tree which do not correspond to any rule and then restructure the remaining nodes in the tree to result in some new merged rules. A rule tree is specific to a given 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 use of rule trees provides the user with a natural and compact hierarchical representation of the user profiles

Algorithm *MiningWithMerging*$(IndexTree: T, (b_1, v_1) \ldots (b_k, v_k))$
begin
{ The large nodes will be visited in postorder }
{ Tree T is the subtree of the original index tree
containing only the large nodes }
$B(n)$ denotes the bounding box for node n
for each large node n **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 the 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 *minsupport* and
minconfidence
end;

Fig. 5. The Pseudocode for Generating Rule Trees.

corresponding to a given set of behaviors. In some sense, this can be considered somewhat similar to the clustering method discussed by Lent et al. in [11], except that in this case, we provide a hierarchical view of the association rules. Also, the method discussed in that work considers only the case for single items in the consequent.

The idea in merging is to convert fragmented rules into large rule sets. Often, it may be the case that by deleting some of the diluted nodes and recombining other more concentrated nodes, it may be possible to get a single large rule. The algorithm works in a bottom-up fashion by visiting the large nodes of the index tree in postorder. For the purpose of abstraction, we may assume that the tree T used in the description of the algorithm in Fig. 5 consists only of those nodes which are large. A node is modified or deleted only if it is a diluted node. Otherwise, it is left untouched. If a node is a diluted node with no large children, then it is automatically deleted. As a result of such deletions, some nodes will have only one child. If such a node n is diluted, then it is deleted and the single child of that node becomes the child of its parent p. As a result, the node p may no longer be the *minimum* bounding rectangle of its children. When the node 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. We note that this readjustment creates a new minimum bounding rectangle which is the union of the minimum bounding rectangles of its children nodes. This readjustment may sometimes result in such a node p changing from being diluted to being concentrated. These kinds of readjustments usually result in larger concentrated regions. Thus, the "merging" algorithm really proceeds by a series of deletions and readjustment operations of the minimum bounding rectangles of the nodes. The algorithm is illustrated in Fig. 5.

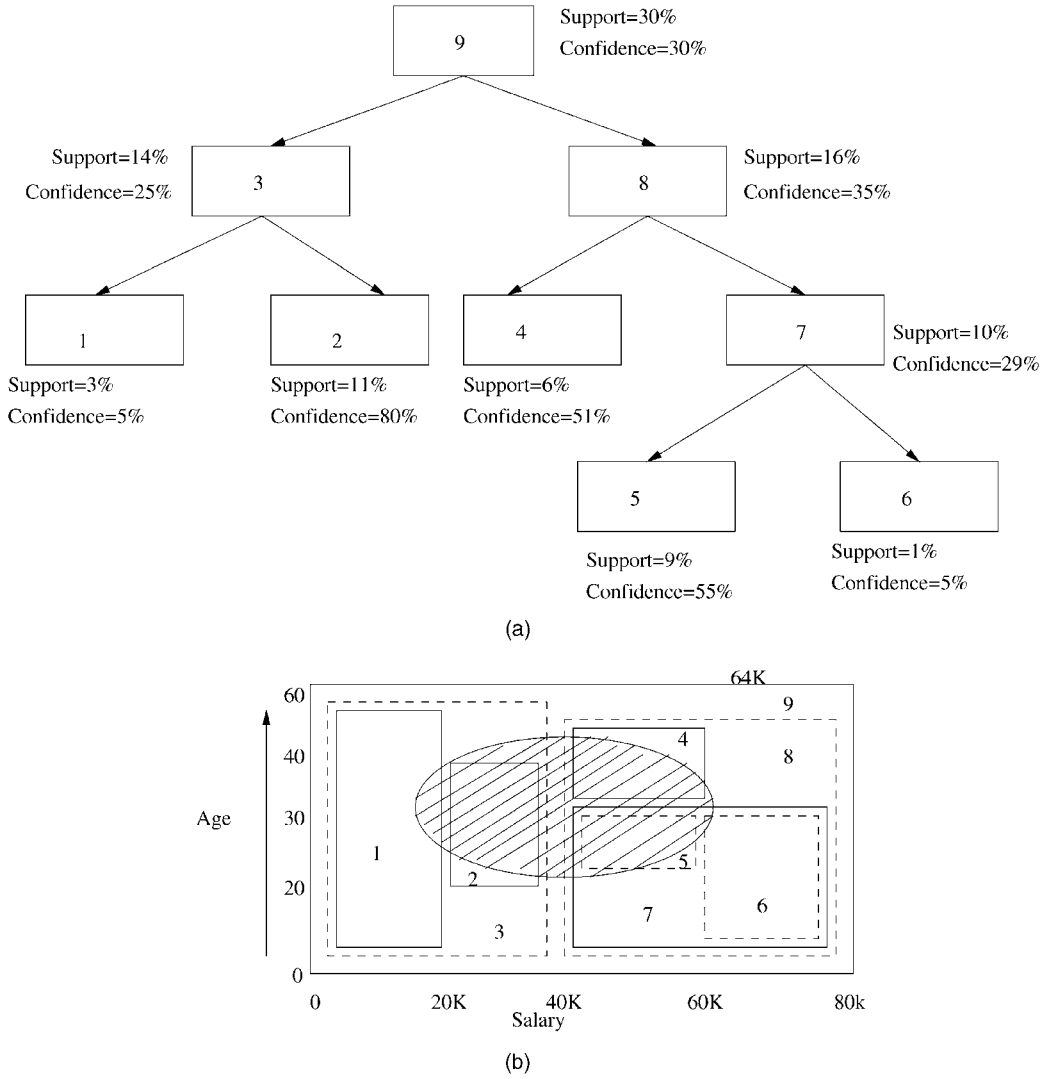


Fig. 6. An Example of the Rule Tree and the Corresponding Positions.

In order to illustrate the workings of the algorithm better, we shall explain with the help of an example.

For the purpose of the example, we shall again consider the tree in Fig. 6 and use a minimum support of 1 percent and a minimum confidence of 50 percent in order to generate the rules. We have already discussed in the last section that three fragmented rules were generated by directly applying the mining algorithm on the index tree.

As we have already indicated, the nodes of the tree are visited in postorder. The nodes of the tree in Figs. 6a and 6b are labeled corresponding to the order in which they are visited. The steps of the algorithm are as follows:

1. **Fig. 7a**. Node 1 is examined. Since it is diluted and has no child, it is deleted immediately. Node 2 is examined. Since it is concentrated, it is not disturbed. The rule corresponding to node 2 is generated.
2. **Fig. 7b**. Node 3 is examined. Since it is diluted and has at most one child, it is deleted, and node 2 is pushed up.
3. **Fig. 7c**. Nodes 4 and 5 are examined. Since they are concentrated, the rules corresponding to them are generated. Node 6 is examined and deleted since it is diluted with no child.
4. **Fig. 7d**. Node 7 is examined and deleted. Node 5 is pushed up.
5. **Fig. 7e**. Node 8 is examined. Since it is diluted and has two children, the bounding rectangle at that node is readjusted to become the bounding rectangle of nodes 4 and 5. This results in a new and concentrated node $8'$.
6. **Fig. 7f**. Node 9 is examined. Since it is diluted and has two children, the bounding rectangle of node 9 is readjusted to those of its children. This results in a new (and concentrated) node $9'$. The rule corresponding to node $9'$ is generated.

The final set of rules is illustrated in Fig. 8. As we see, node $9'$ tends to overlap the region corresponding to the rules quite well. This region $9'$ was obtained by a process of successive deletion and readjustment of the diluted nodes in the tree.

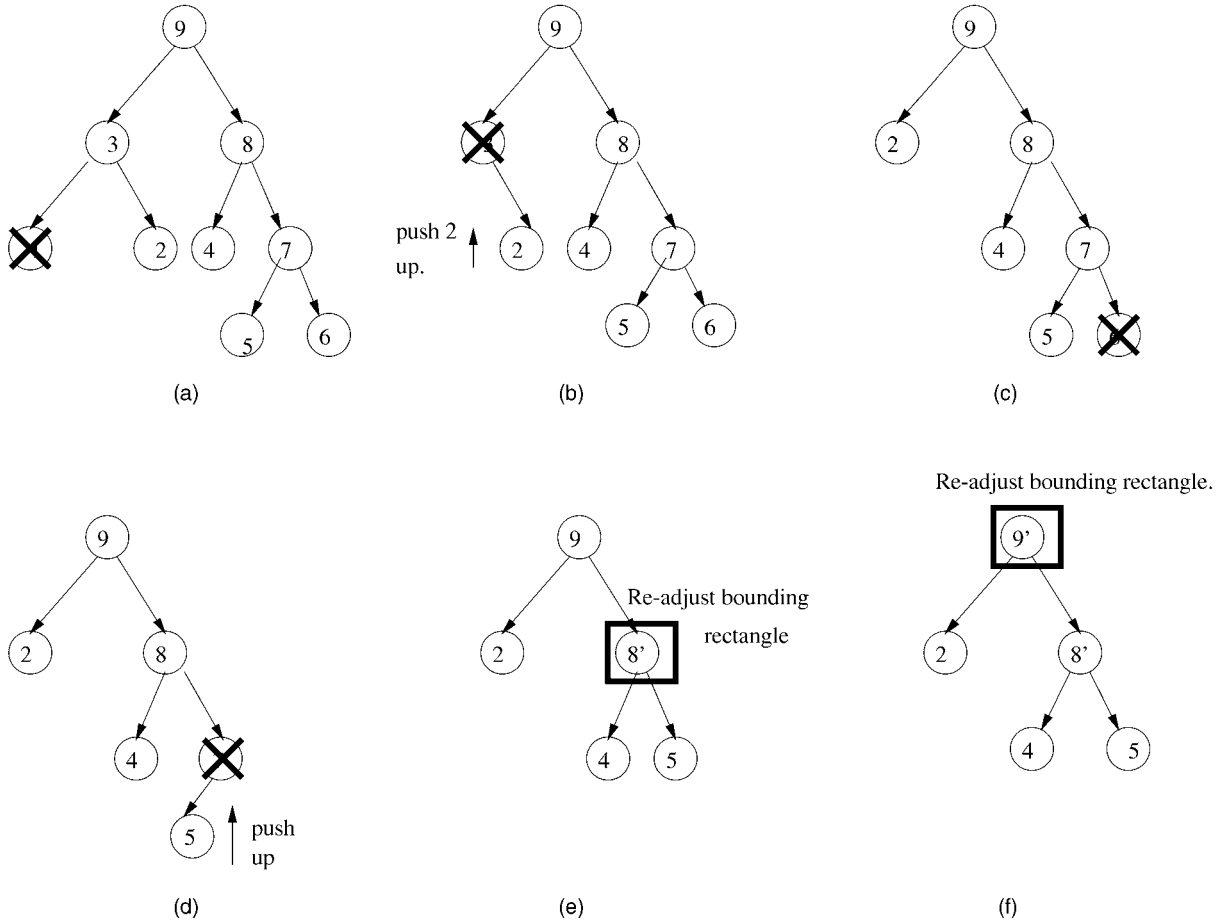


Fig. 7. Pictorial Representation of the Steps of the Algorithm.

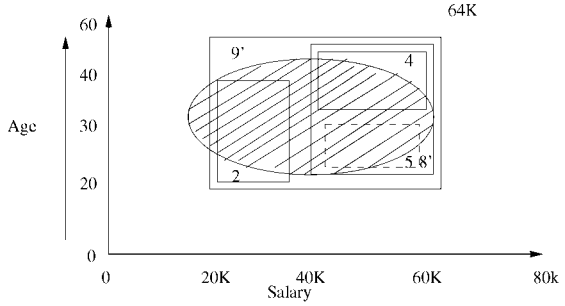
5 The Deeper Mining Algorithm

In this section, we shall discuss a heuristic improvement to the algorithm of the previous section. We shall call this the *deeper* improvement. The idea in the *deeper* improvement is that we search the index tree to a deeper level than the minimum support value indicates. The primary trade-off in the deeper improvement is as follows:

1. Finding the rules for a minimum support level which is too low may result in too many small rules in spite of the merging algorithm.
2. Finding the rules for a minimum support level which is too high may result in loss of information and granularity of bounding rectangles. The contours of the hyperrectangles found for rules may not correspond very well with the actual rules. For example, the final rectangular region denoted by $9'$ in Fig. 8 is slightly misaligned with the real region of concentration. This is because the degree of granularity at which the region may be found is restricted by user-specified values of the minimum support.

The idea of the deeper improvement is that we use the nodes in the index tree at a lower level of support than that indicated by the user. In order to explain this algorithm, we shall introduce some additional notations and terminology.

A node is defined to be *noticeable* with respect to the behavior-value pairs $(b_1, v_1), \ldots, (b_k, v_k)$, at minimum



Regions corresponding to final rule tree
Note readjusted nodes 8' and 9'
Rules are: Bounding(9')=>FirstTimeHomeBuyer
Bounding(8')=>FirstTimeHomeBuyer
Bounding(4)=>FirstTimeHomeBuyer
Bounding(5)=>FirstTimeHomeBuyer
Bounding(2)=>FirstTimeHomeBuyer

Fig. 8. The final regions corresponding to the rule tree.

```
Algorithm DeeperMining(IndexTree : T, (b_1, v_1) ... (b_k, v_k))
begin
  { The noticeable nodes will be visited in postorder }
  { Tree T is the subtree of the original index tree
  containing only the noticeable nodes }
  B(n) denotes the bounding box for node n
  for each noticeable node n do
    begin
    Delete all medium grandchildren of node n
    case
    (1) Node n is concentrated and large:
        Generate rule B(n) ⇒ b_1 = v_1 | ... | 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 the 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 large and concentrated then generate rule B(n') ⇒ b_1 = v_1 | ... | b_k = v_k
        else
        delete any of the children of n which are medium
        if n now has one or less child remaining then
              delete node n and let the parent of any remaining child of n be the
                    node which was the parent of n
    end;
end
```

Fig. 9. The Pseudocode for the Deeper Mining Algorithm.

support s and deeper level f 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 $f * s$ of the total number of points.

Thus, every large node is noticeable, but not every noticeable node is large. A node is defined to be *medium* if it is noticeable, but not large.

The idea in the deeper improvement is to use the medium noticeable nodes for the purpose of merging. The medium noticeable nodes provide a further refinement of the rectangular regions, thus improving the granularity. The deeper algorithm uses the improved granularity of the medium noticeable nodes in order to improve the contours of the rectangular regions obtained. Specifically, the *deeper-f* algorithm uses all noticeable nodes at deeper level f in order to perform the merging. All the final rules generated will correspond only to the large nodes. Thus, medium nodes are not individually used to generate rules, but they may improve the quality of the final rules generated because of being merged with other rules.

The *deeper-f* algorithm works by traversing all the noticeable nodes in the index tree in postorder. For the purpose of abstraction, we assume that the tree T in the description of the algorithm in Fig. 9 consists of only the noticeable nodes in the original index tree. As we shall see, the medium nodes in the tree will be either pruned by the algorithm or merged with other nodes, so that, in the end, the tree T will contain only the large nodes and the rules corresponding to them. If a node n is large and concentrated, then we immediately generate the rule corresponding to the bounding rectangle of that node. If the node is diluted, and has either zero or one children, then we perform exactly the same steps as in the previous merging algorithm discussed in Fig. 5. However, when the node n has two children p_1 and p_2, then the steps are somewhat different. We first check if the minimum bounding rectangle of a node obtained by encompassing p_1 and p_2 into one node n' is concentrated. If so, then we replace the node n by n'. Further, if n' is large, then a rule can also be generated corresponding to the node n'. On the other hand, if the node obtained by finding the minimum bounding rectangle of the nodes p_1 and p_2 is not concentrated, then we delete the nodes p_1 and/or p_2 if they are medium. If one or less child of n remains, then we delete the node n and set the parent of that remaining child of n to be the current parent of n.

Another difference from the merging algorithm is that, whenever we examine a node n, we delete all of its grandchildren if they are medium nodes. The modifications to node n depend only upon the characteristics of its immediate children, and the postorder traversal of the tree ensures that any other node examined in the future will also not be affected by those medium nodes. Since medium nodes are relevant only in obtaining contours of bounding rectangles which have better granularity, rather than generating rules, such nodes can be deleted.

5.1 A Note on Time Requirements

It is important to see how well the merging algorithm performs in terms of the time complexity. We note that the number of nodes visited by the merging algorithm will be of the order of $1/s$, where s is the minimum support. (This is because the total number of nodes visited will be about twice the number of lowest-level nodes visited.) Since each lowest-level node visited has support s and the nodes cover "approximately" disjoint regions, it means that the number of nodes visited is of the order of $1/s$. This is significant because it indicates that even if the size of the original database is large, the number of nodes actually visited is small. For example, when the value of the support in

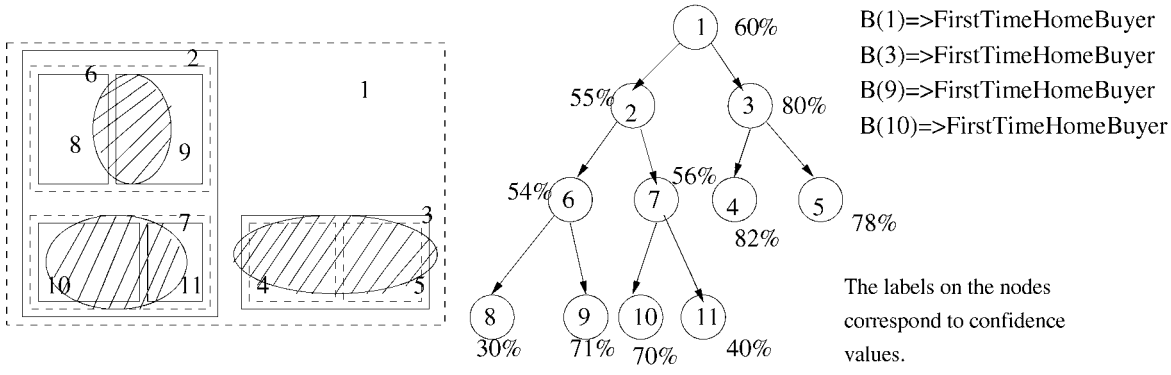


Fig. 10. Finding Interesting Rules.

Algorithm *FindInterestingRules(RuleTree: T, Minconfidence: c, Interest: r,* $(b_1, v_1), (b_2, v_2), \ldots, (b_k, v_k)$ *)*
begin
{ The nodes of the rule tree
will be visited in preorder}
for each node n **do**
begin
if the rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$ exists at node n
begin
find the closest ancestor n' of the node n such that
the rule $B(n') \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$ was generated at node n'.
for each $i \in \{1, \ldots, k\}$ **if** the confidence of the rule $B(n) \Rightarrow b_i = v_i$ is
at least a ratio r above that of the rule $B(n') \Rightarrow b_i = v_i$
generate the rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$ at node n
end
end
end

Fig. 11. Finding Interesting Rules.

percent is 0.5, then the number of nodes visited is of the order of $1/0.005 = 200$. For the case of the deeper improvement, it is easy to see by a similar argument that the number of nodes visited is of the order of $1/(f * s)$.

6 Finding Interesting Association Rules

We shall now discuss the notion of finding interesting association rules for the above algorithm. As has been discussed in earlier work, the notion of interesting association rules has to do with finding all rules which have greater than expected support/confidence values than what they would be based on statistical independence. Our emphasis is to find rules which are interesting in the *hierarchical sense*. In other words, if $\mathcal{S}$ is the set of rules constituting the final output, then, for any given rule $R \in \mathcal{S}$, we look at the nearest ancestor T of R in $\mathcal{S}$. The rule R should be interesting with respect to this ancestral rule T. This kind of hierarchical way of visualizing interesting rules is both intuitive to an end user and prunes a larger number of redundant rules.

The method in which the rules are generated is by using a search algorithm. The nodes of the rule tree are traversed in preorder. In order to test whether the rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$ at a given node n of the tree is interesting, we perform the following procedure. We find the nearest ancestral node n' of n for which the rule $B(n') \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$ was found to be interesting. In the event that there is no such node, then we automatically generate the rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$. Otherwise, we check if, for each $i \in \{1, \ldots, k\}$, the rule $B(n) \Rightarrow b_i = v_i$ has a confidence which is more than a ratio r above the confidence value of the rule $B(n') \Rightarrow b_i = v_i$. If such is indeed the case, then we generate the rule $B(n) \Rightarrow b_1 = v_1 | \ldots | b_k = v_k$. The pseudocode for the algorithm is illustrated in Fig. 11.

Consider the rule tree illustrated in Fig. 10. This tree has 11 nodes. Therefore, there are 11 relevant rules, but only four rules which are generated at an interest level of 1.1. These four rules are the following:

$$B(1) \Rightarrow \text{FirstTimeHomeBuyer},$$
$$B(2) \Rightarrow \text{FirstTimeHomeBuyer}$$
$$B(3) \Rightarrow \text{FirstTimeHomeBuyer},$$
$$B(9) \Rightarrow \text{FirstTimeHomeBuyer}.$$

This is because once the rule $B(1) \Rightarrow \text{FirstTimeHomeBuyer}$ has been generated, only rules which have confidence at least 1.1 times the confidence of this rule may be generated. This property is recursively true for the entire subtree below each rule which was generated. Thus, we see that by adding in an interest measure, we are able to prune away those rules which are not considered interesting, so that succint idea of the rule set may be obtained. Also, it is possible to present these four rules to the user in a hierarchical fashion so as to provide a better view of the generated rules.

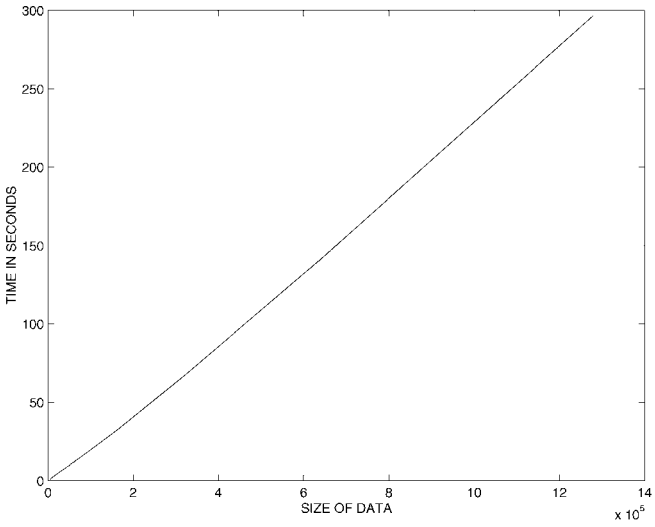


Fig. 12. Setup Time for a Multidimensional Index Tree.

7 Empirical Results

We wish to have a data generation method which would provide a precise method for determining the quality of the rules generated.

We performed the empirical testing by generating N data points, each having l profile 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 first 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 $frac$ 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 $frac$ 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 $frac$ 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 $5,000$ to $1,000,000$.

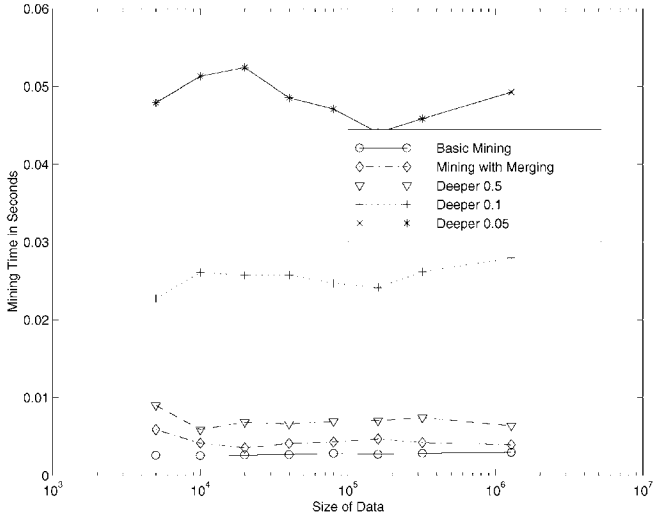


Fig. 13. Variation in Execution Time with Size of Data.

7.1 Performance Measures

The algorithm was tested on several performance measures:

1. **Speed of execution.** We tested the time it took both for the preprocessing as well as the online processing parts of the algorithm. As we see from Fig. 12, the time for setting up the multidimensional index tree is linear with the size of the data. However, the really interesting issue is the amount of time that the online phase of the mining process takes. We have illustrated this in Figs. 13 and 14 for the different algorithms including several values of f for the deeper mining algorithm. For the case of Fig. 13, we fixed the value of the minimum support to 0.8 percent. We see that these running times are so small as to be practically instantaneous. As we had indicated earlier, the reason for this is that the number of nodes in the multidimensional index tree which are large (or noticeable) is typically very small and is of the

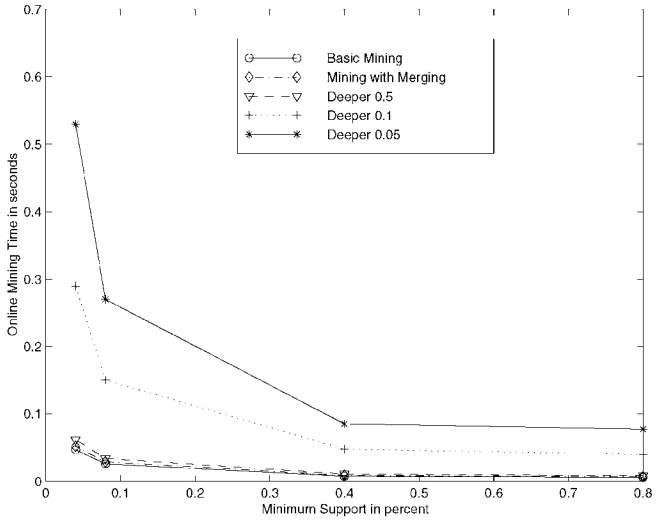


Fig. 14. Variation in Execution Time with Minimum Support.

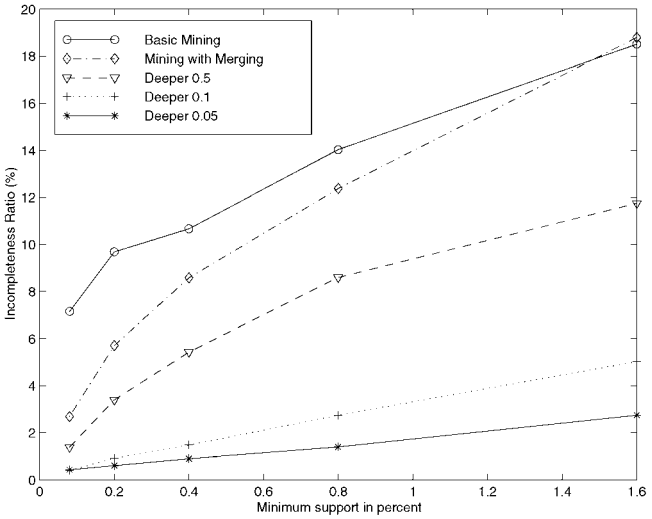


Fig. 15. Variation of an Incompleteness Ratio with Minimum Support.

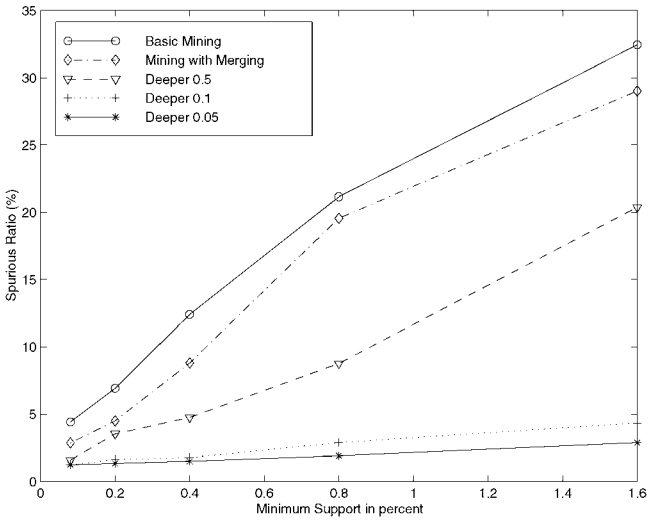


Fig. 16. Variation of a Spurious Ratio with Minimum Support.

order of $1/s$, where s is the value of the minimum support. The empirical illustration of the sensitivity of the online execution time with the minimum support is consistent with the above hypothesis. Further, as is evident from Fig. 13, the execution times are practically insensitive to data size.

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. Specifically, the nearest rule is defined as the percentage overlap of the rule with the given concentration in behavior. In order to test 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} to 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 by 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:

$$\text{Incompleteness Ratio} = 1 - N_{discovered}/N_{actual},$$
$$\text{Spurious Ratio} = N_{false}/N_{actual}.$$

The value of the incompleteness ratio lies between 0 and 1. A value of the incompleteness ratio which is close to 1 means that a considerable fraction of the points corresponding to the rule could not be discovered. A value of the incompleteness ratio which is close to 0 refers to the best possible scenario, where all the points in the rule rectangle could be discovered. We note that the incompleteness ratio is analogous to the concept of recall used in similarity retrieval applications. On the other hand, the analogous concept of precision is captured using the spurious ratio. As we see, the process of merging actually improves the incompletness ratio. However, the real improvement comes from use of the "deeper" mining algorithm. As we see from Fig. 15, the deeper algorithms perform much better in terms of the incompleteness ratio. The spurious ratio is illustrated in Fig. 16. This also shows a similar pattern as the incompleteness ratio. The deeper mining algorithm performs much better than the other mining algorithms. A point to be noted here is that both the spurious ratio and the incompleteness ratio improve by lowering the minimum support. This is because lower minimum support values provide better granularity.

7.2 Some Examples of Rules Generated

In order to illustrate some of the interesting results of the technique, we tested it on a two-dimensional synthetic data set containing 100,000 data points. The two dimensions were Age and Years of Education, and the probability that a given attribute was a first time home buyer was linearly proportional to each of these two attributes. The following rules were obtained:

$$\text{Age}[0-100]\ \text{Education}[1-12] \Rightarrow$$
$$\text{FirstTimeHomeBuyer}(\text{Confidence} = 10\%)$$
$$\text{Age}[55-100]\text{Education}[6-12] \Rightarrow$$
$$\text{FirstTimeHomeBuyer}(\text{Confidence} = 27\%)$$
$$\text{Age}[85-100]\text{Education}[9-12] \Rightarrow$$
$$\text{FirstTimeHomeBuyer}(\text{Confidence} = 40\%).$$

The above set of rules makes it succintly clear that, with increasing age and education, the confidence of being a first time home buyer also increases. The support used was 5 percent which pruned this rule set to a small number of hierarchically arranged rules. When a quantitative association algorithm [17] was used with a grid discretization value of 50, and a confidence of 20 percent, 876 rules were generated for different regions of the data. The quantitative association algorithm required two minutes to generate these rules, whereas the online indexing technique required only four seconds. Furthermore, it was very difficult to understand the overall meaning of the large number of rules generated by the quantitative technique, whereas it was relatively easy to assimilate the rule set obtained by using the online method.

8 Conclusions and Summary

In this paper, we discussed the concept of profile association rules. Such rules may be very useful in developing relationships between consumer profiles and behavioral information. We discussed how to use the multidimensional indexing structure to generate profile assocation rules in online fashion. An additional advantage of using indexing structures in generating the profile association rules is that it is possible to specify specific profile ranges and behavioral features interactively for which a user may wish to find rules.

References

[1] C.C Aggarwal, Z. Sun, and P.S. Yu, "Online Algorithms for Finding Quantitative Association Rules," *Proc. Conf. Information and Knowledge Management (CIKM),* 1998.

[2] C.C. Aggarwal, Z. Sun, and P.S. Yu, "Online Generation of Profile Association Rules," *Proc. Tnt'l Conf. Knowledge Discovery and Data Mining (KDD),* 1998.

[3] C.C. Aggarwal, J.L. Wolf, P.S. Yu, and M. Epelman, "The S-Tree: An Efficient Index for Multi-Dimensional Objects," *Proc. Int'l Symp. Spatial Databases,* pp. 350-370, July 1997.

[4] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Very Large Databases," *Proc. ACM SIGMOD Conf.,* pp. 207-216, 1993.

[5] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Proc. Very Large Data Bases Conf.,* pp. 478-499, 1994.

[6] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 3-14, 1995.

[7] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. SIGMOD Conf.,* pp. 47-57, 1984.

[8] J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," *Proc. Very Large Data Bases Conf. (VLDB),* pp. 420-431, 1995.

[9] M. Houtsma and A. Swami, "Set-Oriented Mining for Association Rules in Relational Databases," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 25-33, 1995.

[10] M. Klementtinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo, "Finding Interesting Rules from Large Sets of Discovered Association Rules," *Proc. Conf. Information and Knowledge Management (CIKM),* 1994.

[11] B. Lent, A. Swami, and J. Widom, "Clustering Association Rules," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 220-231, 1997.

[12] H. Mannila, H. Toivonen, and A.I. Verkamo, "Efficient Algorithms for Discovering Association Rules," *Proc. AAAI Workshop Knowledge Discovery in Databases,* pp. 181-192, 1994.

[13] R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. Very Large Data Bases (VLDB) Conf.,* pp. 144-155, 1994.

[14] J.S. Park, M.S. Chen, and P.S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules," *Proc. SIGMOD Conf.,* pp. 175-186, 1995.

[15] A. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Data Bases," *Proc. Very Large Data Bases (VLDB) Conf.,* pp. 432-444, 1995.

[16] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proc. Very Large Data Bases (VLDB) Conf.,* pp. 407-419, 1995.

[17] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Proc. SIGMOD Conf.,* 1996.

[18] H. Toivonen, "Sampling Large Databases for Association Rules," *Proc. Very Large Data Bases (VLDB) Conf.,* 1996.

[19] W. Ziarko, "The Discovery, Analysis, and Representation of Data Dependencies in Databases," *Knowledge Discovery in Databases,* 1991.

Charu C. Aggarwal received the BTech degree in computer science from the Indian Institute of Technology (1993) and the PhD degree in operations research from the Massachusetts Institute of Technology (1996). He has been a research staff member at the IBM T.J. Watson Research Center since June 1996. He has applied for or been granted 39 US patents, and has published in numerous international conferences and journals. He has been designated a master inventor at IBM Research. His current research interests include algorithms, data mining, and information retrieval. He is interested in the use of data mining techniques for Web and ecommerce applications. He is a member of the ACM and the IEEE.

Zheng Sun received the BS degree in computer science from Fudan University, China, in 1995 and the MS degree in computer science from Duke University in 1998. He was a summer intern at the IBM T.J. Watson Research Center in 1997. He is expected to receive his PhD degree in computer science from Duke University in 2002.

Philip S. Yu received the BS degree in electrical engineering from National Taiwan University, the MS and PhD degrees in electrical engineering from Stanford University, respectively, and the MBA degree from New York University. He is with the IBM T.J. Watson Research Center and is currently the manager of the Software Tools and Techniques group. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, disk arrays, computer architecture, performance modeling, and workload analysis. Dr. Yu has published more than 320 papers in refereed journals and conferences. He holds or has applied for 244 US patents. He is a fellow of the ACM and a fellow of the IEEE. He is the Editor-in-Chief of *IEEE Transactions on Knowledge and Data Engineering*. He is also an associate editor of *ACM Transactions on the Internet Technology* and that of *Knowledge and Information Systems*. He is a member of the IEEE Data Engineering steering committee and is also on the steering committee of IEEE Conference on Data Mining. In addition to serving as program committee member on various conferences, he was the program co-chair of the 11th International. Conference on Data Engineering and the program chairs of the Second International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems. He served as the general chair of the 14th International Conference on Data Engineering and will be serving as the general co-chair of the Second International Conference on Data Mining. He has received several IBM and external honors including the Best Paper Award, two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 70th plateau of Invention Achievement Awards. He also received an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. Dr. Yu is an IBM Master Inventor and was recognized as one of the IBM's 10 top leading inventors in 1999.