Interestingness-Based Interval Merger for Numeric Association Rules

Ke Wang and **Soon Hock William Tay** and **Bing Liu**
Department of Information Systems and Computer Science
National University of Singapore
{wangk,taysoon1,liub}@iscs.nus.edu.sg

Abstract

We present an algorithm for mining association rules from relational tables containing numeric and categorical attributes. The approach is to merge adjacent intervals of numeric values, in a bottom-up manner, on the basis of maximizing the interestingness of a set of association rules. A modification of the B-tree is adopted for performing this task efficiently. The algorithm takes $O(kN)$ I/O time, where k is the number of attributes and N is the number of rows in the table. We evaluate the effectiveness of producing good intervals.

Introduction

An association rule (Agrawal, Imielinski, and Swami 1993a) $Y \rightarrow X$ (e.g. $\{tea, coffee\} \rightarrow \{sugar\}$) represents the knowledge that customers buying all items in Y also buy all items in X. The applicability of the rule is measured by the proportion of transactions buying all items in $Y \cup X$, called the *support*, and the association level is measured by the proportion of transactions buying all items in $Y \cup X$ out of those buying all items in Y, called the *confidence*. The user will specify the minimum support and minimum confidence. Association rules have applications in areas such as analyzing purchase patterns, marketing promotion, document clustering, catalog design, predicting telecommunications order failures and medical test results.

In this paper we consider association rules in a relational table. A transaction is a row in the table and an item corresponds to an attribute/value pair. The antecedent Y and consequent X in association rule $Y \rightarrow X$ are conjunctions of attribute/value pairs (e.g. $Education = high \wedge Bankrupt = no \rightarrow Loan = yes$). Very often, an attribute is *numeric* instead of *categorical*, such as *Age* and *Salary*. For a numeric attribute, it is certain intervals of the attribute, not individual values, that have strong associations with other attributes. For example, the ownership of driving liences is strongly associated with the age 18 and above (the driving age in Singapore). Without priori knowledge, however, determining the "right" intervals can be a tricky and difficult task due to the following "catch-22" situation, as called in (Srikant and Agrawal 1996):

- *Small support*: if an interval is too small, a rule containing this interval may not have the minimum support. As a result, either very few rules are generated or rules are nearly as specific as the data itself.
- *Small confidence*: if an interval is too large, a rule containing this interval in the antecedent may not have the minimum confidence. As a result, many rules with little information are generated.

The interval finding task is further challenged by several other issues. *First*, numeric attributes may interact with other attributes, in which case it does not work to search for intervals one attribute at a time. For example, the normal weight for taller persons is larger than that for shorter persons, thus, the definition of being normal depends on both height and weight. *Second*, most algorithms are exponential in the dimensionality of data, due to the explosion of the number of hyper-rectangles. It is too expensive to extend such algorithms to high dimensions that are very common in real-world applications. *Third*, the antecedent and consequent of an association rule can be any combination of items, thus, often sparsely distributed in an interval. As a result, statistical methods, such as the χ^2 test (Brin, Matwani, Silverstein 1997; Kerber 1992), that often require a dense distribution of data are inapplicable. In this paper, we will address these issues.

A motivating example

Every time two adjacent intervals are merged into one interval, some information may be lost. The idea of our approach is to merge "good" intervals so that the information loss is justified by the improved interestingness of association rules.

Example 1 Consider the simplified data in Table 1. *Age* is a numeric attribute. Initially, every age forms an interval of its own and each row maps to an association rule. See Table 2. If we merge ages [40, 40] and [45, 45] into interval [40,45], there is an information loss on how personnel from this age group are associated

$$\begin{aligned}
rule1) &: \quad Age \in [35,35] \wedge FastTrack = Yes \rightarrow Married = No \\
rule2) &: \quad Age \in [40,40] \wedge FastTrack = No \rightarrow Married = No \\
rule3) &: \quad Age \in [45,45] \wedge FastTrack = No \rightarrow Married = Yes \\
rule4) &: \quad Age \in [50,50] \wedge FastTrack = No \rightarrow Married = Yes.
\end{aligned}$$

Table 2: Initial rules

Age	FastTrack	...	Married
35	Yes	...	No
40	No	...	No
45	No	...	Yes
50	No	...	Yes

Table 1: A simplified dataset

with the marital status. This is reflected by the decrease in confidence of rule 2) and rule 3) from 100% down to 50% in the new rules

$$Age \in [40,45] \wedge FastTrack = No \rightarrow Married = No$$

$$Age \in [40,45] \wedge FastTrack = No \rightarrow Married = Yes$$

On the other hand, if we merge ages [45, 45] and [50, 50] into interval [45,50], no information is lost because all personnel in this interval are married. In fact, the use of interval [45, 50] improves the interestingness by having a larger support while maintaining the same confidence in the new rule

$$Age \in [45,50] \wedge FastTrack = No \rightarrow Married = Yes.$$

If we merge ages [35, 35] and [40, 40] instead, there will be no change in information and interestingness because there is no change on the confidence and support. Therefore, if we are to minimize the loss of information, the second and third mergings are better than the first merging. □

The above information-based merging criterion forms the basis of the work in this paper. We will formally define this criterion by some interestingness measures of association rules in Section 3.

Our approach

Assume that there is a way to evaluate a given interval partitioning, the brute-force method for finding the optimal interval partitioning is in $O(N^{p_1-1} * \ldots * N^{p_k-1})$, where N is the number of rows in the database and p_i is the number of intervals for the ith numeric attribute. This is prohibitively expensive for large databases. Our approach is greedily merging two adjacent intervals at a time, initially one value per interval, by locally optimizing some interestingness-based merging criterion. Several questions need to be answered.

First, what association rules will be used for evaluating the merging criterion. Using all potential association rules will diverse the optimization goal and is too expensive as well. We adopt the template approach in (Rastogi and Shim 1995) where only the instantiations of a specified template are considered. This also addresses the impact of multiple attributes through considering association rules rather than single attributes. *Second*, what is the stopping criterion of the merging process. To avoid stopping at a local optimal, we propose that the merging process stop when the specified number of intervals for each numeric attribute is obtained. The number of intervals for a numeric attribute can be either specified by the user or estimated such as in (Srikant and Agrawal 1996). *Third*, how to find the best merging at each step without scanning all intervals. We adopt a modification of the B-tree for this purpose. Using the modified B-tree, the merging process takes time $O(k * N)$ I/O time, where k is the number of attributes and N is the number of rows in the database.

The paper is organized as follows. Section 2 gives an overview of the approach. Section 3 defines merging criterion. Section 4 discusses the implementation of the interval merging algorithm and presents a cost analysis. Section 5 presents the experimental result. Section 6 reviews related works. Section 7 concludes the paper.

The overview

In our framework, the user specifies a *template* of the form $C1 \rightarrow C2$, where $C1$ and $C2$ are conjunctions of *uninstantiated* and *instantiated* attributes, such that (i) each attribute appears at most once in the template, (ii) $C2$ contains only categorical attributes, and (iii) all numeric attributes are uninstantiated. Each conjunct has the form $A \in [l, u]$ (for numeric attributes) or $A = v$ (for categorical attributes). For an uninstantiated attribute, l, u, v are variables; for an instantiated attribute, l, u, v are values. The template can be instantiated by replacing all variables with values. Each *numeric association rule* is an instantiation of the template. We consider only partitionings of intervals $[l, u]$ that are not overlapping. The problem of mining numeric association rules is finding a "good" interval partitioning of numeric attributes with respect to some evaluation criterion. This problem is decomposed into four steps:

Step 1. Specify the template $C1 \rightarrow C2$, the minimum support, and the minimum confidence by the user. All attributes not mentioned in the template and all rows not matching the instantiated attributes in the template can be removed from the database.

Step 2. Find all instantiations of the categorical attributes in the template $C1 \rightarrow C2$ that satisfy the minimum support. These instantiations are used to initialize the association rules used by the interval merging algorithm. The idea is that if a rule has the minimum support, so does the set of categorical values in the rule. Since the categorical attributes in the template are fixed, we can find all these instantiations in one scan of the database.

Step 3. Generate intervals for numeric attributes in the template. This step greedily merges adjacent intervals by evaluating some merging criterion on a set of association rules. See Section 3.

Step 4. Return all numeric association rules that use the intervals generated in Step 3 and have the minimum support and minimum confidence. This is done by scanning the final set of association rules maintained in Step 3.

For the rest of the paper, we focus on the major step, Step 3.

The merging criterion

We define the merging criterion based on the change of interestingness of association rules.

The interestingness measures

For association rule $Y \rightarrow X$, let $p(X), p(Y), p(XY)$ (XY means the conjunction of X and Y) denote the fraction of rows satisfying condition X, Y, XY, respectively, and $p(X|Y) = p(XY)/p(Y)$. Note that $p(XY)$ and $p(X|Y)$ are the support and confidence of $Y \rightarrow X$. We will consider two interestingness measures, which can be easily substituted by other interestingness measures without changing the framework. The first is the well known *J-measure* used in (Smyth and Goodman 1992) defined as

$$\begin{aligned} J(X,Y) &= p(Y)[p(X|Y)log_2\tfrac{p(X|Y)}{p(X)} \\ &\quad +(1-p(X|Y))log_2\tfrac{1-p(X|Y)}{1-p(X)}]. \end{aligned}$$

The first term $p(Y)$ measures the generality of the rule. The term inside the square bracket measures the "discrimination power" of Y on X, i.e., how dissimilar the priori $p(X)$ and the posteriori $p(X|Y)$ are about X. The dissimilarity is "two-sided" in that $p(X|Y)$ could be either larger or smaller than $p(X)$, due to $p(X|Y)log_2\frac{p(X|Y)}{p(X)}$ and $(1-p(X|Y))log_2\frac{1-p(X|Y)}{1-p(X)}$, respectively. A useful rule implies a high degree of dissimilarity. In our framework of association rules $Y \rightarrow X$, however, the larger the confidence $p(X|Y)$ is, the more interesting the rule is. Therefore, for association rules we will use the "one-sided" J-measure:

$$J_1(X,Y) = p(Y)p(X|Y)log_2\frac{p(X|Y)}{p(X)}.$$

The second interestingness measure is

$$J_2(X,Y) = p(Y)(p(X|Y) - miniconf),$$

where *miniconf* is the minimum confidence in the association rule problem. In words, $J_2(X,Y)$ is the "surplus" of the association in the rule $Y \rightarrow X$ relative to the association level specified by the minimum confidence. For both J_1 and J_2, the larger the value is, the more interesting the rule is. Note that both J_1 and J_2 can be negative, which indicates a negative association in the case of J_1, or an association level below the minimum confidence in the case of J_2.

The merging criterion

Naturally, we can define the usefulness of merging two adjacent intervals as the change of interestingness for each rule in the two intervals. However, our experiments show that this does not give a good result. This is because if two original rules become the same rule after the merging, the interestingness of the result rule will be counted twice. To alleviate this problem, we measure the change of interestingness for *each pair* of rules that are merged into the same rule. Let us formalize this idea.

Consider two adjacent intervals I_1 and I_2. Let $I_1 \cup I_2$ denote the merged interval of I_1 and I_2. Let $rules(I_i)$ be a set of association rules involving I_i. $rules(I_i)$ will be defined in Section 4. Consider a pair k of rules $A \in I_1 \wedge Y_k \rightarrow X_k$ and $A \in I_2 \wedge Y_k \rightarrow X_k$, where Y_k is the rest of the antecedant that does not involve attribute A. After the merging, the new rule is $A \in I_1 \cup I_2 \wedge Y_k \rightarrow X_k$, in place of the pair k.

We define the *loss* of the pair k by merging I_1 and I_2 as:

$$\begin{aligned} Loss_k(I_1,I_2) &= \sigma_1 J_i(X_k, A \in I_1 \wedge Y_k) \\ &\quad +\sigma_2 J_i(X_k, A \in I_2 \wedge Y_k) \\ &\quad -J_i(X_k, A \in I_1 \cup I_2 \wedge Y_k) \end{aligned}$$

where J_i is one of the interestingness measures defined above. $\sigma_i = 1$ if $A \in I_i \wedge Y_k \rightarrow X_k$ is in $rules(I_i)$, and $\sigma_i = 0$, otherwise. In words, $Loss_k(I_1,I_2)$ measures the interestingness loss of the rules in pair k caused by the merging. From the definition of J_i, we can see that, as expected, $Loss_k = 0$ if the merging does not reduce the confidence of the involved rules. The *loss* of merging I_1 and I_2 is defined as

$$Loss(I_1,I_2) = \Sigma_k Loss_k(I_1,I_2)$$

for all such pairs k of rules in $rules(I_i)$.

The merging criterion: merge the two adjacent intervals I_1 and I_2 such that $Loss(I_1,I_2)$ is minimum across all numeric attributes in the template.

Example 2

Consider Example 1. Assume $miniconf = 80\%$. For the merging of $I_1 = [45,45]$ and $I_2 = [50,50]$: $rules(I_1)$ contains rule 3) and $rules(I_2)$ contains rule 4). Rule 3) and rule 4) form a pair of rules that only differ in the interval. The rule after the merging is $rules(I_1 \cup I_2)$:

$Age \in [45,50] \wedge FastTrack = No \rightarrow Married = Yes.$

$p(Y)$ of rules 3) and 4) increases from 25% to 50%, and $p(X|Y)$ remains 100%. Therefore,

$$\begin{aligned} J_1:\ Loss(I_1,I_2) &= 2*0.25*1*log_2(1/0.5) \\ &\quad -0.5*1*log_2(1/0.5) = 0 \\ J_2:\ Loss(I_1,I_2) &= 2*0.25(1-0.8) \\ &\quad -0.5(1-0.8) = 0. \end{aligned}$$

For the merging of $I_1 = [40, 40]$ and $I_2 = [45, 45]$: $rules(I_1)$ contains rule 2) and $rules(I_3)$ contains rule 3). $p(Y)$ of rules 2) and 3) increases from 25% to 50%, and $p(X|Y)$ decreases from 100% to 50%. Therefore,

$$J_1: \ Loss(I_1, I_2) = 2((0.25 * 1 * log_2(1/0.5) \\ -0.5 * 0.5 * log_2(0.5/0.5)) = 0.5$$
$$J_2: \ Loss(I_1, I_2) = 2(0.25(1 - 0.8) \\ -0.5(0.5 - 0.8)) = 0.4.$$

For the merging of $I_1 = [35, 35]$ and $I_2 = [40, 40]$: $p(Y)$ and $p(X|Y)$ remain unchanged, so $Loss(I_1, I_2) = 0$. Therefore, the merging criterion will equally favor the merging of $[45, 45]$ and $[50, 50]$, and the merging of $[35, 35]$ and $[40, 40]$. □

The interval merging algorithm

The interval merging algorithm consists of an initialization phase and a bottom-up merging phase. Let $A_1, \ldots, A_m$ be the numeric attributes and $B_1, \ldots, B_k$ be the categorical attributes in the template $C1 \rightarrow C2$. Let $rows(I)$ be the set of rows that use interval I.

Initialization phase

Initially, each interval has the form $[v, v]$, where v is a numeric value in the database. For each such interval I, we initialize $rules(I)$ to the set of instantiations of template $C1 \rightarrow C2$ of the form $B_1 = b_1, \ldots, B_k = b_k$ and $A_1 \in I_1, \ldots, A_m \in I_m$, where

- $B_1 = b_1, \ldots, B_k = b_k$ is an instantiation found in Step 2 (Section 2) and is true for some row in $rows(I)$,
- $A_1 \in I_1, \ldots, A_m \in I_m$ is true for some row in $rows(I)$, where I_j are initial intervals.

Note that $rules(I)$ are not necessarily disjoint for intervals I of different attributes. Let *Rules* denote the union of $rules(I)$ for all intervals I. Each rule $Y \rightarrow X$ in *Rules*, together with its statistics $p(X), p(Y), p(XY)$, will be stored only once. Thus, in implementation $rules(I)$ is a set of pointers to the rules involved. Without confusion, we continue to talk $rules(I)$ as a set of rules.

Bottom-up merging phase

At each step, we merge the best pair of adjacent intervals chosen from all mergeable attributes according to the merging criterion defined in Section 3. If the number of intervals of the considered attribute has reached the number specified by the user, the attribute becomes non-mergeable. The merging process is repeated until all numeric attributes become non-mergeable. See Figure 1.

As two intervals I_1 and I_2 are merged into $I_1 \cup I_2$, the statistics of rules in $rules(I_1 \cup I_2)$ are obtained from those of rules in $rules(I_1)$ and $rules(I_2)$ as follows. For each rule $A \in I_1 \cup I_2 \wedge Y \rightarrow X$ in $rules(I_1 \cup I_2)$,

$$p(A \in I_1 \cup I_2 \wedge Y) = p(A \in I_1 \wedge Y) \\ +p(A \in I_2 \wedge Y)$$
$$p(A \in I_1 \cup I_2 \wedge YX) = p(A \in I_1 \wedge YX) \\ +p(A \in I_2 \wedge YX).$$

The statistics on the right side of these equalities are stored with rules in $rules(I_1)$ and $rules(I_2)$. At the end of the merging phase, rules in *Rules* satisfying the minimum support and minimum confidence are returned as the solution.

Finding the best merging

In the merging phase, line (1) for finding the best pair of intervals is performed for each merging. The implementation of this operation will significantly affect the efficiency of the algorithm. Simply sorting all potential mergings and processing them in the sorted order does not work because an early merging will affect the ranking of later mergings. Scanning all pairs of adjacent intervals at each merging leads to the quadratic complexity because there are as many intervals as the number of numeric values. For a large database, a reasonable requirement is that finding the best merging takes a time independent of the number of intervals. We adopt a modification of the B-tree (Elmasri and Navathe 1994), called the *M-tree*, to meet this requirement.

The M-tree. The idea is to index all potential mergings by the *Loss* value so that it takes only a few B-tree like operations to find the best merging and maintain the index. Each entry indexed in the M-tree, denoted by (I_1, I_2), represents a potential merging of two adjacent intervals I_1 and I_2. The search key value consists of the *Loss* value of the merging. Like the B-tree, all leaf entries are chained in the ascending order of *Loss* values. This is called the *goodness chain*. The best merging therefore is represented by the left-most leaf entry in the goodness chain. Unlike the B-tree, all leaf entries for the same numeric attribute are chained by the adjacency of intervals. These chains are called *adjacency chains*. After performing the best merging (I_1, I_2), by following the adjacency chain we can find the affected entries of the form (I_0, I_1) and (I_2, I_3), in which intervals I_1 and I_2 must be replaced with the new interval $I_1 \cup I_2$. This is done by deleting the affected entries and inserting entries for $(I_0, I_1 \cup I_2)$ and $(I_1 \cup I_2, I_3)$ using new *Loss* values. To compute the new *Loss* values, each leaf entry (I_1, I_2) contains the list of pointers to the rules in $rules(I_1)$.

Example 3 Consider template

$$Age \in [l_1, u_1] \wedge Salary \in [l_2, u_2] \rightarrow FastTrack = v$$

for a mining task. Figure 2 shows three leaf nodes of the initial M-tree for numeric attributes *Age* and *Salary* (non-leaf nodes are omitted for simplicity). For clarity, we have displayed only entries for *Age*; entries for *Salary* are left blank. Each node contains five entries, though typically more. Each entry contains only the first interval to be merged. For example, the first entry in node 1 represents the merging $(I_1 = [30, 30], I_2 = [40, 40])$ of *Age*, where the second interval $[40, 40]$ can be found by following the adjacency chain in that entry. All entries (including those for *Salary*) are sorted according to *Loss* values.

Bottom-up merging phase:

```
    while there are mergeable attributes do
(1)     select the best pair of adjacent intervals $I_1$ and $I_2$ from all mergeable attributes;
(2)     merge $I_1$ and $I_2$ into one interval $I_1 \cup I_2$:
(3)         replace $I_1$ and $I_2$ with interval $I_1 \cup I_2$ in all rules in $rules(I_1)$ and $rules(I_2)$;
(4)         create $rules(I_1 \cup I_2)$ as the union of $rules(I_1)$ and $rules(I_2)$;
(5)         update the statistics of rules in $rules(I_1 \cup I_2)$;
(6)     if the number of intervals in the considered attribute reaches the threshold
            then mark the attribute non-mergeable;
```

Figure 1: Bottom-up merging phase

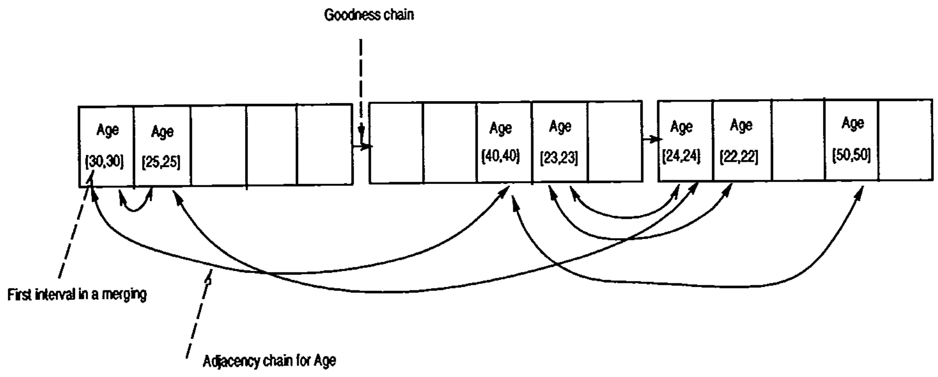

Figure 2: The leaf level of the M-tree

Suppose that the best merging ($I_1 = [30,30], I_2 = [40,40]$) represented by the first entry is performed. We need to delete this entry and update entries affected by the merging. By following the adjacency chain in the first entry, we can find the affected entries ($I_0 = [25,25], I_1 = [30,30]$) and ($I_2 = [40,40], I_3 = [45,45]$). We delete these entries and insert two new entries, i.e., ($I_0, I_1 \cup I_2$) and ($I_1 \cup I_2, I_3$). The insertions are guided by the *Loss* values of these new entries. Therefore, finding the best merging and maintaining the index takes only a few B-tree operations. □

Initialization of the M-tree. This is similar to the loading of the B-tree. The main difference is creating adjacency chains for numeric attributes. We focus on this part only. First, for each numeric attribute A, sort all values $v_1, \ldots, v_k$ and create an entry $([v_i, v_i], [v_{i+1}, v_{i+1}])$ for merging every two adjacent values v_i, v_{i+1}. This entry is assigned identifier i, which indicates the position of the interval in the adjacency chain. Second, sort all entries of all numeric attributes in the ascending order of *Loss* values and store them in a number of leaf nodes (determined by the degree of the B-tree). Third, for each entry of attribute A with identifier i, store its leaf-node address into $A[i]$. Therefore, array $A[\,]$ contains the addresses of entries in the adjacency of intervals of attribute A. Fourth, for each entry of attribute A with identifier i, the two pointers on the adjacency chain are found in $A[i-1]$ and $A[i+1]$.

The cost of the interval merging algorithm consists of the cost of loading the M-tree and the cost of the bottom-up merging phase. Assume that there are N rows and k attributes in the database. The number of entries or mergings is bounded by the maximum number of intervals, which is bounded by kN. The loading of the M-tree is dominated by the work of sorting all entries, thus, has the cost $O(kNlog(kN))$. Each merging needs a fixed number of insertions and deletions on the M-tree. The height of the M-tree is usually no more than 3 or 4 (like the B-tree), so these operations take a constant time. Each merging also needs to access the rules involving the intervals to be merged. Thus, the total cost is $O(kNlog(kN) + krN)$, where r is the average number of rules involving an interval. For most databases in real applications, the (external) sorting takes no more than 3 I/O scans of the database (i.e., $log(kN) \leq 3$). Therefore, assuming that the M-tree is stored on the disk and rules are stored in the memory, the I/O cost is dominated by $O(kN)$.

Experiments

To test if the algorithm can produce right intervals, we used synthetic data generated by Functions 1,2,3 described in (Agrawal, Imielinski, and Swami 1993b). There are six numeric attributes: *Salary*, *Commission*, *Age*, *Hvalue*, *Hyears*, *Loan*, and three categorical attributes: *Elevel*, *Car*, *Zip_code*. We set the number of rows at 10,000. For each function, the number of intervals appearing in the function (for a numeric attribute) is used as the number of intervals in our algorithm, and the minimum support and minimum confidence are chosen as the smallest for all the rules in the function. This is merely for the experimental purpose; in the real case, these thresholds are specified by the user. We list out the intervals generated by our interval merging algorithm.

Function 1. This is a simple case with only one numeric attribute. The rules used to generate the data are:

$$\begin{aligned}&Age < 40 \rightarrow Group = A,\\ &40 \leq Age < 60 \rightarrow Group = B,\\ &Age \geq 60 \rightarrow Group = A.\end{aligned}$$

These rules have minimum support 25% and minimum confidence 100% (which can be easily found out by scanning the database once). There are three intervals for *Age* in these rules, so we specify 3 as the number of intervals for *Age* in running our algorithm. The template for these rules is $Age \in [l, u] \rightarrow Group = g$. For both J_1 and J_2 interestingness measures, our algorithm produces the following intervals:

$$I_1 = [20, 39], I_2 = [40, 59], I_3 = [60, 80].$$

After extending the upper bound u_i of the ith interval $[l_i, u_i]$ to cover values between u_i and l_{i+1} and removing the two extreme values, i.e., 20 and 80, we obtain exactly the intervals in the above rules: $I_1 = (-, 40)$, $I_2 = [40, 60)$, $I_3 = [60, -)$.

Function 2. This function involves two numeric attributes (*Age*) and (*Salary*) through rules:

$$\begin{aligned}&Age < 40 \wedge 60K \leq Salary < 95K \rightarrow Group = A,\\ &40 \leq Age < 60 \wedge 95K \leq Salary < 125K \rightarrow Group = A,\\ &Age \geq 60 \wedge 25K \leq Salary < 60K \rightarrow Group = A.\end{aligned}$$

Note that we have modified the original Function 2 in (Agrawal, Imielinski, and Swami 1993b) (by modifying the data generation program) so that the Salary intervals are non-overlapping. These rules have minimum support 15% and minimum confidence 75%. The number of intervals is 3 for *Age* and 3 for *Salary*. The template used is

$$Age \in [l_1, u_1] \wedge Salary \in [l_2, u_2] \rightarrow Group = g.$$

The intervals generated are

$$\begin{aligned}J_1:&\quad Salary : (-, 61K), [61K, 96K), [96K, -)\\ &\quad Age : (-, 40), [40, 60), [60, -)\\ J_2:&\quad Salary : (-, 61K),\ [61K, 95K), [95K, -)\\ &\quad Age : (-, 40),\ [40, 60),\ [60, -).\end{aligned}$$

These are very good approximation of those in the given function.

Function 3. This function involves one numeric attribute (*Age*) and one categorical attribute (*Elevel*) in the antecedent, given by:

$$\begin{aligned}&Age < 40 \wedge Elevel \in [0..1] \rightarrow Group = A,\\ &40 \leq Age < 60 \wedge Elevel \in [1..3] \rightarrow Group = A,\\ &Age \geq 60 \wedge Elevel \in [2..4] \rightarrow Group = A.\end{aligned}$$

These rules have minimum support 3% and minimum confidence 95%. The number of intervals is 3 for *Age*. The template used is $Age \in [l, u] \wedge Elevel = e \rightarrow Group = g$. For both J_1 and J_2, the intervals generated are identical to those in the function:

$$(-, 40),\ [40, 60),\ [60, -).$$

To summarize, the proposed interestingness-based interval merging does produce the good intervals as expected.

Related work

Most works on association rules have focused on categorical values. Mining numeric association rules was considered recently in a few papers. (Srikant and Agrawal 1996) introduces the partial completeness as a measure of information loss when going from values to intervals, and proposes that the equi-depth partitioning minimizes the number of intervals required to satisfy a partial completeness level. Our work was motivated by the remark in (Srikant and Agrawal 1996): "equi-depth partitioning may not work very well on highly skewed data. It tends to split adjacent values with high support into separate intervals though their behavior would typically be similar". (Fukuda et al. 1996) and (Lent, Swami, and Widom 1995) consider association rules in which two numeric attributes appear in the antecedent and one categorical attribute appears in the consequent. In our framework, any number of numeric attributes and categorical attributes can appear in the antecedent, and more than one categorical attributes can appear in the consequent. More recently, (Rastogi and Shim 1995) studies the problem of finding non-overlapping instantiations of a template association rule so that the combined support (confidence, resp.) meets some threshold and the combined confidence (support, resp.) is maximized. In (Rastogi and Shim 1995), a heuristic is used to guide the search, but the worst-case is exponential. We also use a template to focus the rules to be optimized, but with a different goal: find the best intervals that maximize the interestingness gain of association rules. Our algorithm does not suffer from the exponential worst-case.

Less related are the work on classification rules with numeric attributes, e.g., (Agrawal et al. 1992; Kerber 1992; Wang and Goh 1997). In these works, the intervals of an attribute are produced by examining only the current attribute (and the class attribute), therefore, interaction with other attributes is not addressed. We address the interaction by considering association rules (which involve the attributes of interest),rather than individual attributes. A more fundamental difference between these works and ours is that of the objective: for classification rules, the task of interval generation aims at minimizing the predictive error; for association rules, it aims at improving the interestingness of rules.

Conclusion

Relational tables containing both numeric and categorical values are very common in real-life applications. We have proposed a new algorithm for mining numeric association rules from relational tables. There are several contributions in the development of the algorithm. We proposed an interestingness-based criterion for determining intervals to be merged. To address the impact of multiple attributes on interval generation, the interestingness-based criterion was defined with respect to association rules to be discovered, rather than a single attribute being considered. We dealt with the complexity explosion due to high dimensionality of data by considering mergings of intervals rather than hyperrectangles. We adopted a modification of the B-tree, the M-tree, for performing the merging process efficiently for large databases. Though we have considered two specific interestingness measures, the M-tree by no means depends on the interestingness measures used, and can be used as a general data structure for merging intervals using any interestingness measure. The experiments show that our interestingness-based interval merger is highly effective in discovering right intervals. Our algorithm is scalable for very large databases because it is linear in the database size and in the dimensionality.

Further researches are needed in several areas. The current method requires the number of intervals for a numeric attribute to be specified by the user. It is desirable to automatically determine the number of intervals. Also, our method produces only non-overlapping intervals. Thus, rules like

$$40 \leq Age \leq 60 \wedge Salary \leq 95K \rightarrow Group = A,$$
$$50 \leq Age \leq 70 \wedge Elevel \in [1..3] \rightarrow Group = B$$

cannot be generated. Finally, more datasets and more interestingness measures can be tested to characterize the applicability of different interestingness measures to different data characteristics.

References

Agrawal, R.; Ghosh, S.; Imielinski, T.; Iyer, B.; & Swami, A. 1992. An interval classifier for database mining applications. In Proceedings of VLDB.

Agrawal, R.; Imielinski, T.; & Swami, A. 1993a. Mining association rules between sets of items in large databases. In Proceedings of SIGMOD, 207-216.

Agrawal,
R.; Imielinski, T.; & Swami, A. 1993b. Database mining: a performance perspective. IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6, Dec 1993 (http://www.almaden.ibm.com/cs/quest/).

Brin, S.; Matwani, R.; & Silverstein, C. 1997. Beyond market baskets: generalizing association rules to correlations. In Proceedings of SIGMOD.

Kerber, T. 1992. ChiMerge: discretization of numeric attributes. In Proceedings of Ninth National Conference on Artificial Intelligence, 123-128.

Elmasri, R.; & Navathe, S.B. 1994. Fundamentals of Database Systems. The Benjamin/Cummings Publishing Company, Inc.

Fukuda, T.; Morimoto, Y.; Morishita, S.; & Tokuyama, T. 1996. Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization. In Proceedings of SIGMOD, 13-23.

Lent, B.; Swami, A.; & Widom, J. 1995. Clustering association rules. In Proceedings of ICDE.

Rastogi, R.; & Shim, K. 1995. Mining optimised association rules with categorical and numeric attributes". In Proceedings of ICDE.

Srikant, R.; & Agrawal, R. 1996. Mining quantitative association rules in large relational tables. In Proceedings of SIGMOD, 1-12.

Smyth, P.; & Goodman, R. 1992. An information theoretic approach to rule induction from databases. IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 4, Aug 1992.

Wang, K. and Goh, H.C. 1997. Minimum splits based discretization for continuous features. In Proceedings of IJCAI, 942-947