An Adaptive Method of Numerical Attribute Merging for Quantitative Association Rule Mining

Jiuyong Li, Hong Shen and Rodney Topor
School of Computing and Information Technology
Griffith University
Nathan Qld 4111 Australia
Email: {jiuyong,hong,rwt}@cit.gu.edu.au

Abstract

Mining quantitative association rules is an important topic of data mining since most real world databases have both numerical and categorical attributes. Typical solutions involve partitioning each numerical attribute into a set of disjoint intervals, interpreting each interval as an item, and applying standard boolean association rule mining. Commonly used partitioning methods construct set of intervals that either have equal width or equal cardinality. We introduce an adaptive partitioning method based on repeatedly merging smaller intervals into larger ones. This method provides an effective compromise between the equal width and equal cardinality criteria. Experimental results show that the proposed method is an effective method and improves on both equal-width partitioning and equal-cardinality partitioning.

keywords: Data mining, association rule, continuous attribute discretization.

1 Introduction

1.1 Association rule mining

Association rule mining was first studied by Agrawal et al. [1], and can be formally stated as follows. Let $I = \{i_1, i_2, \cdots, i_m\}$ be a set of symbols that are called *items*. The items may be goods, attributes, or events. A *transaction* T is a set of items, $T \subset I$, and a collection of transactions form a database $\mathcal{D}$. A set of items is called an *itemset*, and an itemset with k items is called a k-itemset. The support of itemset X, denoted as $\sigma(X)$, is the percentage of $\mathcal{D}$ in which every transaction contains X. An itemset is called a *frequent itemset* if its support is not less than a user specified *minimum support* ζ, or $\sigma(X) \geq \zeta$. An implication in the form of $X \to Y$ is an association rule, where $X, Y \subset I$ and $X \cap Y = \emptyset$. The *confidence* of rule $X \to Y$ is the ratio of $\sigma(X \cup Y)$ to $\sigma(X)$, denoted by $\kappa(X \to Y)$. If $\kappa(X \to Y) \geq \eta$, where η is a user specified *minimum confidence*, the rule is called a *strong rule*. For example, suppose 30% of students have taken both Computer Programming and Word Processing, and 80% of the students who have taken Computer Programming also have taken Word Processing. Then the rule "(Computer Programming) $\to$ (Word Processing)" has support of 30% and confidence of 80%. The goal of association rule mining is to find strong rules in databases, which is normally achieved by first finding frequent itemsets and then forming association rules.

1.2 Quantitative association rule mining

Mining association rules in databases of numerical and categorical attributes rather than boolean attributes is called mining quantitative association rules [13]. An example of a quantitative association rule is Age $\in [40, 50] \wedge$ Married $\to$ Cars $= 2$.

There are several possible methods for quantitative association rules mining. One is to consider individual categorical states and separate numerical values as items, and then use a boolean association rule mining algorithm. After all rules are obtained, the rules with adjacent numerical values are grouped [9, 14]. For example rule $Age = 50 \wedge Married \rightarrow house$ and rule $Age = 55 \wedge Married \rightarrow house$ are combined as $Age \in [50, 55] \wedge Married \rightarrow house$. This method suffers from difficulties of mining on certain numerical values such as insufficient support from the database to some individual numerical values and too many numerical values on a particular attribute.

A second method of quantitative association rule mining is to map quantitative attributes into boolean attributes, then use algorithms of boolean association rule mining. Mapping a categorical attribute to a boolean attribute is straightforward: it can be realized by enumerating categorical states to a set of positive integers (items). As to numerical attributes, the common way is to cut a continuous attribute to some intervals then map all values in an interval to an item. Converting a continuous attribute to a set of discrete values is called *discretization*.

There are several methods to do this [2, 6, 5, 3]. Many methods are supervised discretization methods in which optimal classes are known beforehand. However, in many data mining problems, the optimal classes are not known, so unsupervised discretization is more suitable. There are few unsupervised discretization methods. The two most frequently used methods are the following.

1. Equal-width discretization divides the continuous attribute range into N intervals of equal width. For example, ages from 20 to 60 can be divided into four intervals of width 10 years. This method can be easily implemented, but has the clear drawback that there may be too few instances in some intervals and too many in other intervals, and both cases hinder mining high quality association rules.

2. Equal-depth (or equal-cardinality) discretization divides the continuous attribute range into N intervals so that there are $1/N$ of the total instances in each interval. This method avoids the possible imbalance inherent in the equal-width discretization method, but it may separate similar attribute values into different intervals and group dissimilar attributes into the same interval.

Equal-depth discretization method is preferred in general [7, 11, 13]. A key problem in using this method is choosing the number of intervals. Small interval size tends to result in the loss of interesting rules, and large interval size tends to reduce the accuracy of rules. To deal with this problem, a measure of K-partial completeness is defined to decide how many intervals should be cut in a continuous attribute [13]. However, even with this measure, this method may not work very well on highly skewed data as stated by Srikant and Agrawal [13]. A main cause of these problems is that equal-width discretization method and equal-depth discretizatione method do not consider both value densities and value distances at the same time.

In addition to the above, methods based on clustering have also been proposed for unsupervised discretization [4, 11]. In [11], a method for clustering numeric atttributes to mine distance based association rule was proposed, where an association rule is represented by $C_y \rightarrow C_x$, where C_x and C_y are density clusters.

Our work is motivated by the concept of clustering, but does not form association rules directly from clusters like [11]. We discretize numerical attributes and then convert quantitative association rule mining to boolean association rule mining. Our proposed discretization method is an unsupervised discretization method. It initially places each numerical attribute value in a separate interval, and then selectively merges similar adjacent intervals. It uses a merging criterion that considers both value densities and value distances of numerical attributes, and produces proper value density and suitable interval width so that association rules can be easily found from them. After numerical attributes are discretized as a set of disjoint intervals, each interval can be interpreted as a boolean attribute, thus transforming quantitative association rule mining into boolean association rule mining.

In this paper, we first define a criterion for merging adjacent intervals and develop a numerical attribute merging algorithm. Next we present a quantitative association rule mining algorithm. Finally, we implement both algorithms and compare our proposed discretization method with equal-width and equal-depth discretization methods.

2 Numerical attribute merging algorithm

The goal of our work is to partition a numerical attribute into a set of disjoint intervals by minimizing differences within intervals and maximizing differences between intervals in light of clustering[10]. Intervals with too few attribute occurrences in the database would prevent itemsets from having sufficient support and intervals with too many attribute values would fail to discriminate between attribute values and would hence fail to lead to useful association rules. Hence, we present a merging algorithm to produce a set of intervals with suitable attribute values.

Initially, suppose that a numerical attribute has m distinct values, $\{x_1, x_2, \ldots, x_m\}$. Without loss of generality we further assume that $x_i < x_{i+1}$ for all $1 \leq x \leq m-1$ (we can simply sort these values otherwise). Define m intervals $I_1, \ldots, I_m$ such that each I_i includes x_i, $1 \leq i \leq m$. Let each interval I_i have a *representative centre* c_i initially defined to be x_i, $1 \leq i \leq m$.

In general, suppose that an interval I contains attribute values $\{x_1, x_2, \ldots x_k\}$ and has representative center c. Define the average intra-interval distance of I with respect to c to be

$$\text{Dist}(I, c) = \frac{1}{k}\sum_{i=1}^{k} w_i D_d(x_i, c), \tag{1}$$

where w_i is the weight of value x_i, and $D_d = \sum_{i=1}^{k} \mid x_i - c \mid$ is the Manhattan distance since all values in I are one-dimensional. Let the weight w_i of x_i to be the number of occurrences of value x_i in the database.

Assume that two adjacent intervals $I_i = \{x_1, \ldots, x_i\}$ and $I_j = \{x_{i+1}, \ldots, x_{i+j}\}$ have representative centres $c_i = \frac{\sum_{p=1}^{i} x_p n_p}{\sum_{p=1}^{i} n_p}$ and $c_j = \frac{\sum_{p=i+1}^{i+j} x_p n_p}{\sum_{p=i+1}^{i+j} n_p}$, where attribute value x_p contains n_p occurrences (in the database), $\leq 1p \leq j \leq m$. Clearly, the number of attribute occurrences in I_i is $N_i = \sum_{p=1}^{i} n_p$. and in I_j is $N_j = \sum_{p=i+1}^{i+j} n_p$. The union, $I = I_i \cup I_j$, of the two intervals containing j attrubute values and $N_i + N_j = \sum_{p=1}^{i+j} n_p$ attrubute occurences in total thus has its representative centre given by the average weighted value of (c_i, n_i) and (c_j, n_j):

$$c \quad = \quad \frac{c_i \sum_{p=1}^{i} n_p + c_j \sum_{p=i+1}^{i+j} n_p}{\sum_{p=1}^{i+j} n_p} = \frac{c_i N_i + c_j N_j}{N_i + N_j}. \tag{2}$$

The intra-interval distance of I with respect to c calculated by Equation (1) is then uniquely defined by c_i and c_j as follows. When $x_i \leq c \leq x_{i+1}$ which holds in our algorithm, noting that $\sum_{p=1}^{i} x_p n_p = c_i N_i$ and $\sum_{p=i+1}^{i+j} x_p n_p = c_j N_j$ we can derive this distance as follows:

$$\begin{aligned}
\text{Dist}(I, c) &= \frac{1}{i+j}\sum_{p=1}^{i+j} n_p \mid x_p - c \mid \\
&= \frac{1}{i+j}\left(\sum_{p=1}^{i} n_p(c - x_p) + \sum_{p=i+1}^{i+j} n_p(x_p - c)\right) \\
&= \frac{1}{i+j}\left(\sum_{p=i+1}^{i+j} n_p x_p - \sum_{p=1}^{i} n_p x_p + \left(\sum_{p=1}^{i} n_p - \sum_{p=i+1}^{i+j} n_p\right)c\right) \\
&= \frac{1}{i+j}\left(c_j N_j - c_i N_i + (N_i - N_j)\frac{c_i N_i + c_j N_j}{N_i + N_j}\right) \\
&= \frac{2 N_i N_j}{(i+j)(N_i + N_j)}(c_j - c_i).
\end{aligned} \tag{3}$$

We may now define the difference between I_i and I_j, denoted by Diff(c_i, c_j), to be propotional to their intra-interval distances given by the above equation. That is,

$$\text{Diff}(c_i, c_j) = \frac{N_i N_j}{N_i + N_j}(c_j - c_i). \tag{4}$$

We can see that the difference between two intervals is determined not only by the distance between their representative centers but also by the number of database occurrences of values in each interval. If two pairs of intervals are the same distance apart, the pair with smaller number of occurrences in each interval has a smaller difference than the pair with larger number of occurrences in each interval.

Next we consider how to choose a pair of adjacent intervals to merge. Given m consecutive intervals $I_1, \ldots, I_m$, whose representative centers are $c_1, \ldots, c_m$, there are $m-1$ pairs of adjacent intervals. We test every pair of adjacent intervals, and then merge the two with the smallest Diff, say I_k and I_{k+1}. The merged interval $I_k = I_k \cup I_{k+1}$ has representative centre c_k and number of values n_k updated as follows.

$$\begin{aligned} c_k = & \quad (n_k c_k + n_{k+1} c_{k+1})/(n_k + n_{k+1}); \\ n_k = & \quad n_k + n_{k+1}; \end{aligned}$$

We repeatedly merge the pair of adjacent intervals with minimum difference in this way.

We now propose a criterion for terminating this merging process before we have reduced the data to a single large interval. If the density of each interval is large enough to form a rule, namely $n_i/N > \zeta$, or the representative centers of each pair of adjacent intervals are so far apart that they are unlikely to be in one group, for example $(c_{i+1} - c_i) > 3\overline{d}$, where $\overline{d}$ is the average distance of adjacent values of a numerical attribute, the numerical attribute merging procedure stops.

After executing the above procedure, the whole range of numerical attribute values is partitioned into a set of adjacent intervals, where the number of values in each interval is large enough to reach the user specified minimum support or each interval is too isolated to be merged into an adjacent interval.

The above procedure may be summarised as follows.

Algorithm 1 *Numerical attribute merging algorithm*
Input: *An ordered sequence of numeric attributes* $\{x_1, \ldots, x_m\}$.
Output: *An ordered sequence of disjoint intervals* $I_1, \ldots, I_{m'}$ *covering* $x_1, \ldots, x_m$, $m' < m$.

```
For each $x_i$ $(1 \leq i \leq m)$ do
    Let $I_i$ contain $x_i$;
    Let $c_i = x_i$ be the representative centre of $I_i$;
End
Let $m' = m$;
For each interval pair $(I_i, I_{i+1})$ $(1 \leq i < m')$ do
    Let Diff $(c_i, c_{i+1}) = \frac{n_i n_{i+1}}{n_i + n_{i+1}}(c_{i+1} - c_i)$;
End
While (termination condition is not satisfied) do
    Let $k$ be such that Diff $(c_k, c_{k+1})$ is minimal;
    Let $c_k = (n_k c_k + n_{k+1} c_{k+1})/(n_k + n_{k+1})$;
    Let $n_k = n_k + n_{k+1}$;
    Merge $I_k$ and $I_{k+1}$ into a new interval $I_k$;
    Let $m' = m' - 1$;
    Recompute Diff$(c_{k-1}, c_k)$ and Diff$(c_k, c_{k+1})$;
End;
Output intervals $I_1, \ldots, I_{m'}$;
```

3 Quantitative association rule mining

Quantitative association rule mining is the process of mining association rules for databases with both categorical attributes and numerical attributes. In a quantitative association rule $X \to Y$, X and Y may be the combination of boolean values, categorical states and numerical intervals. In the previous section, we have discussed how to find suitable intervals of numerical attributes. In this section we give a brief description of our algorithm for quantitative association rule mining. It consists of the following four steps.

3.1 Pre-processing

In this stage, the goal is to convert categorical and numerical attributes to boolean attributes on which boolean association rule mining algorithms can be applied. This is achieved by enumerating the values of categorical attributes and mapping interval sets of numerical attributes to a set of items. The core of this stage is to find suitable intervals for each numerical attribute, which affects the mining performance greatly. In our algorithm, interval finding is realized by Algorithm 1. After sets of cut-points of numerical attributes are found, all values in one interval are interpreted as an item. For example, different temperature ranges represent different items in Table 1. After that we can use boolean association rule mining on these items.

sex	code	cough	code	temperature	code	⋯
male	1	bad	3	35.0 – 36.9	6	⋯
female	2	slight	4	36.9 – 37.1	7	⋯
—	—	no	5	37.1 – 42.0	8	⋯

Table 1: An example of converting quantitative attributes to a set of items

3.2 Frequent itemset finding

The following efficient algorithm for frequent itemset finding was developed in [12]. The data structure *set trie* is important in the algorithm, since it allows efficient generation of candidates and verification and storage of frequent itemsets. Some similar data structures have also been used for association rule mining.

The set trie we use is an ordered and labeled root tree that can store a set and all its subsets conveniently. Its main characters are listed below.

Given a sorted set of positive integers $L = \{l_1, l_2, \dots, l_n\}$, where $l_i < l_{i+1}$ if $i < j$.

1. Each node in set trie is labeled by an element in L, and more than one node (or leaf) in a set trie can have the same labels.

2. Labels of son nodes are ranked higher than their parents by the order in set L.

3. Each node stores all nodes on the path from the root to it.

An example of a set trie that stores set {1, 3, 5, 6} and all its subsets is depicted in Figure 1.

Frequent itemset finding involves four steps:

1. Initiating a set trie.

   We first find all frequent 1-itemsets and frequent 2-itemsets, and then initiate a set trie. Nodes in the first layer of set trie are added directly from all frequent items. Nodes in the second layer are those items having *frequent links* (two items that form a frequent 2-itemset are called having frequent link between them) with nodes in the first layer.

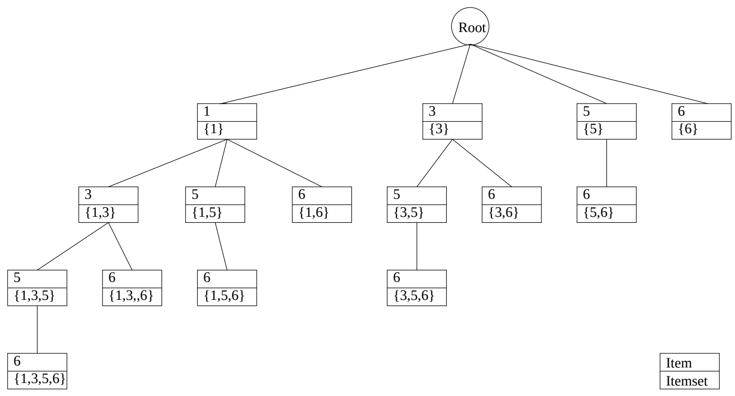

Figure 1: An example of set trie

2. Generating candidates for frequent itemsets

   Our candidate generation is based on the following observation: If an itemset is frequent, all its subsets are frequent itemsets and links among all items contained in it are frequent links. Candidates are generated as follows. Suppose that a node has m frequent itemsets, $\{i_1, i_2, \ldots, i_{k_1}\}, \{i_1, i_2, \ldots, i_{k_2}\}, \ldots, \{i_1, i_2, \ldots, i_{k_m}\}$, as its sons in the k-th layer of a set trie. Candidates in the $(k+1)$-th layer under the node i_{k_1} of k-th layer are $\{i_1, i_2, \ldots, i_{k_1}, i_{k_q}\}$, where $\sigma(i_{k_1}, i_{k_q}) \geq \zeta$ and $k_1 < k_q \leq k_m$. We can generate all candidates in the $(k+1)$-layer of the set trie by this way. The candidate generator based on set triee is very efficient since candidates only relate to siblings under a same parent rather than all k-itemsets so that much searching time is saved. A newly generated candidate is added in the tree as a new leaf.

3. Counting support of candidates

   A tree structure can be easily searched in breadth-first or depth-first. However in a set trie, we do not search the whole trie but *trace* it by a transaction. We illustrate tracing procedure by the following example. Suppose that a transaction has k frequent items, $\{i_1, i_2, \ldots, i_k\}$. In the first layer we only go to nodes $i_1, i_2, \ldots, i_k$, and mark them. Then we go into subtrees rooted by the marked nodes, and mark nodes $i_2, \ldots, i_k$ in the second layer of the subtrees. And so on. Support of each itemset under marked nodes is incremented by 1. As the length of a transaction is far smaller than the size of set I, tracing set trie only searches a small part of the whole and set trie hence is very fast.

4. Deleting infrequent itemsets

   If an itemset is proved to be infrequent, then all its supersets can not be frequent itemsets. In a set trie, supersets are stored in subtrees rooted by nodes storing the prefix subsets of them, as illustrated in Figure 1. If a node stores an infrequent itemset, the whole subtree including itself can be removed from the set trie. When there is no infrequent itemset to be removed, all frequent itemsets are stored in the set trie and are easily located since themselves are paths.

3.3 Rule forming

After all frequent itemsets are obtained, the task here is to test whether two disjoint subsets of an frequent itemset can form a rule. If there are the user specified targets, this step will be very simple. Otherwise we have to test all subsets as consequences in turn. For a consequence Y, which is an item or an itemset, we first search set trie for frequent itemsets $\mathcal{Z} = \{S_i | Y \subset S_i\}$. Then we obtain a set of rule $\{X_i \to Y, i = 1, 2, \ldots, m\}$, where $\kappa(X_i \to Y) > \eta$. For the convenience of rule pruning, rules with a same consequence are stored together and storage structure is also the set trie, since it is convenient for searching and counting in the post processing procedure.

The rule forming by the criterion of the minimum support and the minimum confidence may produce many uninteresting redundant rules. Rule pruning in the following section can resolve the problem of redundancy. Measures of interestingness are very application oriented.

3.4 Post-processing

There may be some redundant rules in the result of mining, if some of them account for some similar facts, such as $A \to C$ and $A \wedge B \to C$. The goal of rule pruning is to select a set of minimally overlapping rules from a raw rule set without losing information. Given support and confidence, an interesting rule is the one with the highest confidence, hence the pruning method we propose is to choose the rule with the highest confidence and delete other similar rules.

For a rule $X \to Y$, if $(X \cup Y) \subset T$, then we say that rule $X \to Y$ *covers* transaction T. On the other hand, if $(X \cup \neg Y) \subset T$, then we say that rule $X \to Y$ *uncovers* transaction T.

For a set of rules $\mathcal{R} = \{\, X_i \to Y \mid 1 \leq i \leq m \ ,\}$ and a database $\mathcal{D} = \{T_1, \ldots, T_n\}$, *the total covered transactions* of rule set $\mathcal{R}$ over $\mathcal{D}$ is

$$Cov(\mathcal{R}) = \bigcup_{i=1}^{n} T_i, \qquad \text{where } (X_k \cup Y) \subset T_i \text{ for any } (X_k \to Y) \subset \mathcal{R}.$$

The *coverage* of rule set $\mathcal{R}$ is the ratio of the size of $Cov(\mathcal{R})$ to the number of all transactions.

Consider two rules $X_1 \to Y$ and $X_2 \to Y$ where $\kappa(X_1 \to Y) > \kappa(X_2 \to Y)$. If both of them cover the same data set, rule $X_2 \to Y$ loses support when transactions covered by rule $X_1 \to Y$ are removed from database. As a result, the confidence of rule $X_1 \to Y$ is reduced as well.

If we choose a rule, and then remove all transactions covered by the rule from database, supports and confidences of the rest rules will change. For rule $X \to Y$, there are $\sigma(X) = \sigma(X \cup Y) + \sigma(X \cup \neg Y)$ and $\kappa(X \to Y) = \sigma(X \cup Y)/(\sigma(X \cup Y) + \sigma(X \cup \neg Y))$. Once we store a rule covering transactions in cov_i and uncovering transactions in $uncov_i$, and update them after each rule is selected, supports and confidences of rest rules can be updated easily. The ultimate goal is to simplify the raw rule set while maintaining total coverage unchanged.

A pruning algorithm selecting high confidence rules and keeping the same total coverage as the raw rule set is given below.

Algorithm 2 *Pruning the rule set*

```
Input: Database $\mathcal{D} = \{T_1, \ldots, T_n\}$, itemset $Y$, and rule set $\mathcal{R} = \{\, X_i \to Y \mid 1 \leq i \leq m \,\}$.
Output: Pruned rule set $\mathcal{R}' \subseteq \mathcal{R}$.
    Let $\mathcal{R}' = \emptyset$;
    For each rule $(X_i \to Y) \subset \mathcal{R}$ do
        Let $cov_i = \bigcup_{j=1}^{n} T_j$, where $(X_i \cup Y) \subset T_j$;
        Let $uncov_i = \bigcup_{j=1}^{n} T_j$, where $(X_i \cup \neg Y) \subset T_j$;
    End
    Let $Cov = \bigcup_{i=1}^{m} cov_i$;
    Let $m' = m$;
    While $(Cov \neq \emptyset)$ do
        Let $d$ be such that $\kappa(X_d \to Y)$ is minimal;
        Let $\mathcal{R} = \mathcal{R} \setminus (X_d \to Y)$;
        Let $\mathcal{R}' = \mathcal{R}' \bigcup (X_d \to Y)$;
        Let $m' = m' - 1$ ;
        Let $Cov = Cov \setminus \{\, T_k \mid (X_d \cup Y) \subset T_k \,\}$ ;
        For each $(X_i \to Y) \subset \mathcal{R}$ do
            Let $cov_i = cov_i \setminus \{T_k | (X_d \cup Y) \subset T_k\}$;
            Let $\kappa(X_i \to Y) = |cov_i|/(|cov_i| + |uncov_i|)$;
        End;
    End;
```

Output rule set $\mathcal{R}'$.

4 Performance results

Quantitative association rule mining algorithm depends critically on the discretization of continuous attributes. If discretization produces a set of suitable intervals, the computed association rules will have a large total coverage. The method proposed in this paper has been evaluated by comparing its performance with equal-width and equal-depth discretization methods.

We implemented our quantitative association rule mining algorithm and tested it on some databases from the Machine Learning Database Repository at the University of California at Irvine. A brief description of the database is given in Table 2.

Database name	Records	Attributes	Classes
Glass Identification	214	9 Numerical	3
Heart Disease	270	7 Numerical + 6 Categorical	2
Iris Plant	150	4 Numerical	3
Wisconsin Breast Cancer (original)	699	10 Numerical	2

Table 2: Brief description of databases

The experiment was conducted by comparing the proposed algorithm methods with equal-width and equal-depth discretization methods, in which numerical attributes are partitioned into 5 or 10 equal value intervals or equal density intervals respectively. They are denoted as equal-width 5, equal-width 10, equal-depth 5 and equal-depth 10.

In the experiment, we do not find all possible rules but some interesting rules with consequences as the labeled classes. To avoid finding too few rules from the small distributed items and too many rules from the large distributed items at a fixed minimum support, we use local support instead of support. The local support of a rule $X \to Y$ is the support of $X \cup Y$ in the sub-database of transactions that include Y.

The experiment results are displayed in Figure 2.

From Figure 2, we can see that the overall results of our numerical attribute merging method is better than others. In 24 trials, only 1 trial is worse, 4 trials are marginally lower, and the other 19 trials are better than or equal to equal-width and equal-depth methods. We observed that no other method in the experiment achieves this performance.

It is reported [5, 8] that performances of all discretization methods have little difference except that supervised discretization methods are better than unsupervised discretization methods. In the case that labelled classes are known, the supervised discretization methods are good choices. When the class information is unknown, the method proposed in this paper is a good one. Since it considers both instance densities and value distances, it has the merits of both equal-depth and equal-width methods. Let us consider two extreme cases. If the termination condition of Algorithm 1 does not include the restriction of the density in an interval, the merging result will become equal-width partition. On the other hand, if the termination criterion has no distance restriction, then the merging result will be equal-depth partition. Therefore our proposed method is an adaptive method that generalizes the equal-depth and equal-width methods.

5 Conclusion

In this paper, we have introduced an unsupervised discretization method that uses a clustering based criterion to merge adjacent intervals until some termination criterion is reached. The numerical attribute merging algorithm is evaluated in comparison with equal-width and equal-depth discretization methods.

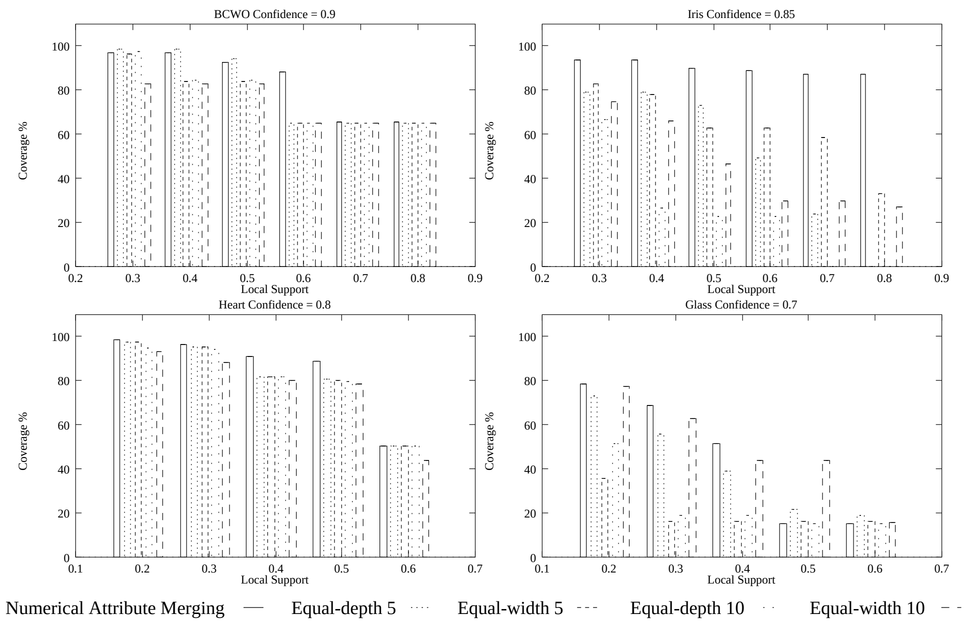

Figure 2: Comparison of mining result of three class blind methods

Experiment results show that the proposed method gains certain improvement in quantitative association rule mining over these existing methods. This improvement comes from the fact that our method is an adaptive method that generalises both equal-width and equal-depth methods.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Fast algorithms for mining association rules in large database s. In *20th Int'l Conf. on Very Large Databases (VLDA)*, june 1994.

[2] J. Catlett. On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Learning : Machine Learning (EWSL-91)*, volume 482 of *LNAI*, pages 164–178, Porto, Portugal, March 1991. Springer Verlag.

[3] Jesús Cerquides and Ramon López de Màntaras. Proposal and empirical comparison of a parallelizable distance-based discretization method. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 139. AAAI Press, 1997.

[4] Thierry Van de Merckt. Decision trees in numerical attributes spaces. In *IJCAI-93*, 1993.

[5] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proc. 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.

[6] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI-93*, 1993.

[7] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In ACM, editor, *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1996, Montréal, Canada, June 3–5, 1996*, volume 15 of *Proceedings of the ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems*, pages 182–191, New York, NY 10036, USA, 1996. ACM Press.

[8] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In Evangelos Simoudis, Jia Wei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119. AAAI Press, 1996.

[9] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *Proceedings of the 13th International Conference on Data Engineering (ICDE'97)*, pages 220–231, Washington - Brussels - Tokyo, April 1997. IEEE.

[10] Kaufman Leonard and Rouseeuw Peter. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley-Interscience Publication, 1990.

[11] R. J. Miller and Y. Yang. Association rules over interval data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):452–??, ???? 1997.

[12] Jiuyong Li Hong Shen and Paul Pritchard. Knowledge network based association discovery. In *Proc. of 1999 International Conference on Parallel and Distributed Processing Techniques and Applications, (PDPTA'99)*, 1999.

[13] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2):1–??, ???? 1996.

[14] Ke Wang, Soon Hock William Tay, and Bing Liu. Interestingness-based interval merger for numeric association rules. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 121–128. AAAI Press, 27–31 August 1998.