Supervised Dynamic and Adaptive Discretization for Rule Mining

Rohan Khade∗ rkhade@gmu.edu
Nital Patel† nital.s.patel@intel.com
Jessica Lin∗ jessica@gmu.edu

Abstract
Association rule mining is a well-researched topic in data mining. However, a common limitation with existing algorithms is that they mainly deal with categorical data. In this work we propose a methodology that allows adaptive discretization and quantitative rule discovery in large mixed databases. More specifically, we propose a top-down, recursive approach to find ranges of values for continuous attributes that result in high confidence rules. Our approach allows any given continuous attribute to be discretized in multiple ways. Compared to a global discretization scheme, our approach makes it possible to capture different inter-variable interactions. We applied our algorithm to various synthetic and real datasets, including Intel manufacturing data that motivated our research. The experimental results and analysis indicate that our algorithm is capable of finding more meaningful rules for multivariate data, in addition to being more efficient than the state-of-the-art techniques.

1 Introduction
Data generated during the manufacture of semiconductor products continues to grow exponentially due to increased part and package complexity (e.g. integration of more functions within the same package) while the time window for data analysis and yield learning continues to decrease in order to support faster time to market. In order to overcome these conflicting requirements we have been working on developing scalable exploratory data mining solutions that are amenable to the increasing volumes of data coming out of Intel factories. The intent is to quickly and automatically find complex patterns in data that used to previously require a person to know what to look for, and even after that needed considerable amount of analysis. Some of this motivated us to focus on a general framework for association rule mining on discrete and continuous variables with very high order data dimensions.

Association rule mining [1, 2, 4, 16, 21, 23, 24, 25, 26, 27, 28, 29, 30] has received a lot of attention in the past two decades. Many existing rule mining algorithms consist of two important steps: the discovery of frequent itemsets, and generation of rules from these frequent itemsets. Frequent itemsets refer to itemsets whose support measures exceed some user-defined \( \text{minsup} \), and rules are considered interesting if their confidence measures exceed some user-defined \( \text{minconf} \). The rules are in the form of \( Y \Rightarrow Z \), where \( Y \) (antecedent) and \( Z \) (consequent) denote sets of items. Most rule mining algorithms, including those that follow this support-confidence framework, work best with categorical attributes. However, many real world applications contain continuous attributes. The mining of rules on such datasets is called quantitative association rule mining [4, 6, 11, 12, 13, 20, 21, 22, 23, 25, 26, 27]. A common technique for handling continuous attributes in rule mining is discretization. Also referred to as binning, discretization is the process of converting a numeric value into one of the (pre-defined) intervals, categories, or bins.

We discovered that applying association rule mining to continuous data resulted in a non-trivial problem fundamentally tied to developing an effective discretization of the data. For example, most quantitative association rule mining algorithms employ a one-time discretization strategy [4, 6, 11, 12, 13, 17, 19, 20, 21, 22, 23, 25, 26, 27]. That is, as a pre-processing step, continuous attributes are discretized; these new discretized values then replace the original numeric values and are used to find rules. While such discretization strategy [13, 18, 21] can be efficient, there exist some limitations. More specifically, each value is typically binned into only one category according to some discretization policy. In some cases, however, an attribute may benefit from different binning schemes depending on the attribute(s) it interacts with. To understand the need for adaptive discretization, consider a (tiny) subset of the Intel manufacturing data shown in Figure 1a. Note the attribute names have been removed due to proprietary reason.

The example shows the histograms of a continuous attribute \( X \) combined with two different categorical attributes, say \( C_1 \) and \( C_2 \), of varying cardinalities. In Figure 1a the categorical attribute \( C_1 \) has cardinality of 2 (i.e. there are 2 distinct values denoted by different shades of grey), whereas in Figure 1b, the categorical
attribute (C2) has cardinality of 4. We notice that the “best” number of bins for X is 2 when it is combined with attribute C1, and 4 when it is combined with attribute C2. A global discretization strategy may produce too many or too few bins.

In addition, imagine a scenario where the distribution changes for some attribute-pairs in a data stream. For example, suppose the boundary between the two classes in Figure 1a shifts to the left, but the boundaries between the four classes in Figure 1b remain the same. With a global discretization scheme that discretizes an attribute X regardless of other attributes, we would not be able to capture such a change in its relationship with C1 while retaining its relationship with C2.

This example demonstrates a likely situation with quantitative association rule mining. For clarity of presentation, let’s consider the simplest rule format for now: we are interested in finding rules such that the consequent contains a single categorical attribute, and the antecedent consists of one or more continuous attributes. Since the continuous attributes “interact” with different categorical attributes in various rules, there may not be a universally best discretization for a continuous attribute. If we consider the categorical attribute in the consequent as the “class label”, we can perform supervised discretization and use the categorical attribute to guide the discretization process.

Another common limitation for existing quantitative association rule mining (or more generally, discretization algorithms) is that they typically discretize one attribute at a time. This often produces suboptimal results, as this approach does not consider possible interactions between multiple attributes. To demonstrate the need for multivariate discretization [6] in the supervised manner, consider the XOR dataset shown in Figure 2a. This dataset contains two continuous attributes, Attribute1 and Attribute2, and one categorical (class) attribute with values {A, B}. The class attribute is treated as the consequent for the task of rule mining, e.g. \{Attribute1 < 0.5, Attribute2 < 0.5\} => A. If we examine only one attribute at a time, the two classes are indistinguishable. To find rules that distinguish between the two classes, we need to examine both attributes simultaneously and consider the interaction between them.

In this work, we propose a novel Supervised, Dynamic and Adaptive Discretization (SDAD) technique for quantitative association rule mining that addresses the aforementioned limitations. Our approach results in automatic rule discovery, hence eliminating the need for a separate rule mining step except for rules consisting of strictly categorical attributes. More specifically, our contributions are described as follows:

1. We propose a general framework to find association rules for mixed continuous and categorical data.
2. Our supervised algorithm takes attribute relationship into account and allows adaptive discretization. We consider two types of attribute relationship: antecedent-consequent relationship, and multivariate relationship. While the discovery of such attribute relationships is the goal for rule mining, we argue that without proper discretization, some rules might be missed or invalid.
3. Our approach automatically determines the number and sizes of the bins for continuous attributes.
4. Our approach combines discretization and rule generation into a single process. We do not generate frequent itemsets.
5. Our algorithm is able to find low-support, high-confidence rules.
6. We implement our algorithm in a Hadoop cluster, which shows the parallelizability of the algorithm and its potential for rule mining in big data.

The remaining of the paper is organized as follows. Section 2 discusses related work and background on discretization and quantitative association rule mining. We describe our algorithm in Section 3. In Section 4, we show experimental results on our discretization algorithm, as well as its impact on quantitative association rule mining using synthetic and real data. We conclude and discuss future directions in Section 5.
Figure 1: Histograms for the continuous attribute X showing interactions with two different categorical attributes. (a) The categorical attribute C1 has 2 distinct values. (b) The categorical attribute C2 has 4 distinct values.

Figure 2: Two non-trivial discretization examples. (a) Discretization for XOR data requires us to consider multiple attributes simultaneously. (b) The attribute has a bi-modal distribution. Each color represents a class, or equivalently, a consequent of a rule.

would likely be large since the support for the middle region is lower than that of the rest of the regions. It is desirable to have an algorithm that identifies the exact boundaries that separate the two distributions, which in turn would produce more meaningful rules.

Clustering-based discretization [3, 15] may be used to address the problems of attribute relationship and multi-modal distributions. Though such techniques are more versatile, they often suffer from the following shortcomings: (1) they are often more expensive computationally; (2) results might be suboptimal if the clusters are not well separated; and (3) depending on the algorithm, the number of clusters (i.e. bins) may need to be known in advance.

The discretization techniques we discussed so far are unsupervised — that is, data are partitioned based solely on the values. Most existing work on quantitative association rule mining uses some version of unsupervised discretization algorithm. Our initial attempt to find meaningful rules on the Intel data using various unsupervised binning strategies result in poor and unintuitive partitions of data, which translate to rules that inaccurately describe the data.

An alternative is supervised discretization, which uses additional information (class labels) to improve the quality of binning. Data are partitioned such that the “purity” of the intervals is maximized.

Quantitative association rule mining aims to find rules for datasets containing both categorical and continuous data. Srikant and Agarwal proposed a discretization technique that partitions the range of the continuous attribute into $n$ equal-frequency partitions, and assign the partitions to consecutive integers [26]. If the supports for any consecutive partitions fall below the $\text{minsup}$ threshold, they are merged. The final bins are then used as frequent items to find rules. The problem, however, is that it is unclear how to set $n$, the initial number of partitions. If $n$ is too small, it results in large partitions and potential information loss since elements in the same partition are indistinguishable. On the other hand, if $n$ is too large, the algorithm becomes computationally expensive since there are more merging
candidates to consider.

To account for attribute interactions, Bay introduced a supervised discretization scheme called Multivariate Discretization (MVD) [6]. Similar to the algorithm proposed by [26], MVD also splits the continuous attributes into \( n \) initial, fine partitions. It then merges the partitions if differences between their distributions are insignificant. There are two main advantages to this technique: the numbers and sizes of the partitions are automatically determined, and it handles discretization on multivariate data. Similar to the problem faced by the method proposed by [26], it is unclear how to set the initial \( n \).

Subgroup discovery is an application of association rule mining where rules are found based on a target attribute in contrast to traditional rule mining which is more exploratory. It differs from classification by being both predictive and descriptive of the target variable. A good survey of subgroup discovery algorithms and applications can be found in [14]. Although there are specialized algorithms to find subgroups, any association rule algorithm is capable of finding subgroups by keeping a fixed consequent. Our main goal for developing this algorithm is to find quantitative association rules but we note it could also be used to find subgroups for mixed type datasets.

Other approaches include fuzzy association rule mining [12] which eliminates the sharp boundary problem where elements near the boundaries are either ignored or overemphasized. RUDE [20] projects the class attributes onto the other attributes and cluster them. However, it is difficult to set the stopping criteria as the dimensionality increases. Some algorithms do not discretize but directly find rules based on statistical measures such as mean and variance [4]. Salleb-Aouissi et al. proposed a genetic algorithm which finds discretization boundaries by optimizing support and confidence [25]. The drawback of this algorithm is that there are many parameters to set.

3 Supervised Dynamic and Adaptive Binning

We will start the discussion of the algorithm by the simplest case of discretizing one continuous variable \( X \) against one categorical variable \( C \), but note that it can be generalized to multivariate discretization. For each \( X \) considered, we examine it with different categorical attributes \( C \), and use \( C \) to perform supervised discretization. Therefore, the categorical attribute holds the “class” information for our supervised discretization of \( X \). The goal is to find rules in the form \( X \Rightarrow C \) (e.g. \( 44 < \text{Age} < 54 \Rightarrow \text{Salary} > 50K = \text{Yes} \)).

We define a pure or high-conf interval of \( X \) as a region in \( X \) for which the conditional probability of any value in the range, given a particular categorical attribute \( C \), exceeds some user-defined \( \text{minconf} \), i.e. \( \text{conf}(X \Rightarrow C) > \text{minconf} \). SDAD uses a divide-and-conquer, binary split approach to find the high-conf sections. At each iteration, it splits a region in half, and computes the confidence for each partition. The algorithm recursively splits a region until its confidence exceeds \( \text{minconf} \). The recursive procedure also merges contiguous high-conf regions if their respective confidence measures differ no more than 10%. At the end, the algorithm checks to see whether the high-conf regions can be further expanded by linearly scanning the regions with mixed labels.

In a sense, our approach is similar to decision trees on the handling of continuous attributes. However, decision trees have some limitations that make them unsuitable for quantitative rule mining. First, the end goal of decision trees is inherently different. Our goal is not to classify a particular instance, or a combination of attribute values, but rather, find rules corresponding to some or any target attribute values. These target attributes may not be pre-defined, as association rule mining is an exploratory process in which we typically do not know what is interesting in advance. A natural question to ask is, could we have built a series of decision trees, one for each candidate rule, using the consequent as the class label? In practice, decision trees for quantitative rule mining does not work well since we would likely generate a massive rule set that could have been better aggregated or condensed. Repeated decision tree construction is also highly inefficient since there is no mechanism to prune the candidate attribute sets using this approach (similar to enumerating all itemsets and constructing a decision tree for each itemset). The overall cost can escalate, especially when considering the multivariate cases.

Figure 3(Left) shows the histogram of a continuous attribute \( X \) with respect to a categorical attribute \( C \). \( C \) has two values, \( \{A, B\} \) which are used as class labels and denoted by different shades of grey. The vertical lines denote all the splits performed by the algorithm. Ideally, we would like the algorithm to identify the point

Figure 3: (Left) Vertical lines denote all the splits before merging. (Right) Final result after merging.
that separates the two classes. The first step is to split the range of \( X \) into two halves of equal frequency (the left-most vertical line shows the first split). The region on the left is a pure region; the region on the right is an impure (low-conf) region containing values associated with both classes. We do not need to split the pure partition further since every value in this region is associated with the same category. The algorithm further splits the right partition into 2 regions and so on. The procedure continues until the confidence measures of all regions exceed \( \text{minconf} \). Finally, SDAD generalizes the rules by merging adjacent regions if appropriate. After merging, only one split remains for this example, and the final rules are \( \{ X > 13.74 \} \Rightarrow A \) and \( \{ X < 13.74 \} \Rightarrow B \), both with \( \text{conf} = 1 \) (Figure 3(Right)).

Unlike most quantitative rule mining algorithms, our approach integrates rule discovery with discretization. In our experiments we limit the number of items on the antecedent to 3, since rules involving a large number of items may be too specific and uninteresting. Also, since the main objective is to find meaningful rules, our algorithm may not discretize the entire space but only find rules in particular sections. Since it does not require a \( \text{minsup} \) requirement like Apriori-based algorithms, we are able to find low-support, high-confidence rules which represent non-trivial information. For example, the first rule found in Figure 3 has a support of 0.015.

3.1 Handling Multiple Continuous Attributes
Multivariate discretization techniques can be quite expensive, since one has to potentially try all combinations of attributes. Our algorithm prunes the search space in the following ways. First, if we find a good rule, that is, a rule satisfying \( \text{minconf} \), we do not need to continue expanding the rule. While it is possible to get a rule with higher confidence by adding more attribute(s) to the antecedent (since confidence measure does not have the anti-monotone property), here our goal is to find the most general rules that meet the \( \text{minconf} \) requirement. This is to avoid overfitting and finding rules that are too specific. To avoid finding trivial and uninteresting rules, we can set a high \( \text{minconf} \) value. Take the XOR dataset in Figure 2a as an example, no matter how we try to partition the data using just one attribute, we would not find a region with high enough confidence. We would then need to add another attribute to identify high-conf regions. We would not need to add more attributes to the combination since we are able to find good rules using only two attributes.

In addition to using confidence, our algorithm ranks the rules using various interest measures such as lift and leverage. Lift and leverage are objective measures that compare the observed support to that expected if the attributes were statistically independent and dependent, respectively. Lift measures “how many times more often the antecedent and consequent occur together than expected if they were statistically independent” [10]. Leverage measures “the difference of the antecedent and consequent appearing together in the data set and what would be expected if \( X \) and \( Y \) were statistically dependent” [24]. They are not monotonically decreasing and hence cannot be used for pruning the search space. However, they provide insight on the usefulness of a rule. Lift is useful for identifying low-support, high-confidence rules. If we contend to find top-\( k \) rules, Webb [30] introduced \( k \)-Optimal Rule Discovery (KORD). KORD uses these interest measures to prune the search space. Our algorithm works nicely to extend KORD to handle mixed datasets. Using the KORD approach rather than the Apriori approach to build the search tree improves the efficiency of our algorithm.

4 Experimental Validation
In this section, we show the utility of our algorithm on synthetic and real data.

4.1 Synthetic Data
We start our experiments by demonstrating the algorithm’s ability to find meaningful partitions. We created two synthetic datasets similar to those presented by Bay [6] (since their datasets are not publicly available). The first dataset consists of two multivariate Gaussians in the shape of an “X” as shown in Figure 4a. Each Gaussian belongs to a separate class. The bins and the rules discovered are denoted by the four red, dashed boxes. More specifically, the rules with \( \text{minconf} = 1 \) are \( \{ X > 0.7125 \}, \{ X < 0.6445 \} \Rightarrow \text{Class}1, \{ X > 0.7482 \}, \{ X < 0.6814 \} \Rightarrow \text{Class}2, \{ X > 0.7154 \}, \{ X < 0.6946 \} \Rightarrow \text{Class}1, \{ X > 0.6327 \} \Rightarrow \text{Class}2 \).

In the same figure, denoted by solid lines, we also show the partitions found by the decision tree. The 7 partitions enclosed by solid lines represent all the leaf nodes in the decision tree. While the decision tree could identify many “pure” regions except for the middle section, they are not as compact as the ones found by our algorithm. MVD [6] is able to detect the relationship of the two attributes with a tuning of the parameter \( \delta \) that controls the number of bins allowed; however, it also finds a bucket in the center. Our algorithm does not find any strong rule in that region because it contains items from both classes with about the same frequency.

The second simulated dataset is shown in Figure 4b. It consists of 2 multivariate Gaussians which belong to
different classes. Our algorithm finds the rules, \( \{X_1 > 3.5875\} \Rightarrow \text{Class}1 \) and \( \{X_1 < 3.5875\} \Rightarrow \text{Class}2 \), using only attribute \( X_1 \) and then terminates. MVD does not recognize this and combines attributes even when not needed [6].

![Figure 4: (a) Dynamic binning vs. decision tree on Synthetic Dataset 1. (b) Discretization result on Synthetic Dataset 2.](image)

### 4.2 Real Data
In this section, we conducted experiments on a real, Adult Census dataset from the UCI Machine Learning repository [5]. It has 14 attributes and 48,842 instances. There are 4 continuous attributes including age, capital-gain, capital-loss and hours-per-week. Our goal is to discretize the continuous attributes with respect to the class attribute (i.e. \( \text{Salary} > 50K \)). Table 1 shows the rules found by our algorithm, with \( \text{minconf} \) set to 0.35.

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
<th>Confidence</th>
<th>Lift</th>
<th>Leverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>age (44-54)</td>
<td>Sal&gt;50K</td>
<td>0.39</td>
<td>1.64</td>
<td>0.060</td>
</tr>
<tr>
<td>hours/week (53-97)</td>
<td>Sal&gt;50K</td>
<td>0.43</td>
<td>1.82</td>
<td>0.013</td>
</tr>
<tr>
<td>cap-gain (3103-99999)</td>
<td>Sal&gt;50K</td>
<td>0.75</td>
<td>2.58</td>
<td>0.035</td>
</tr>
<tr>
<td>cap-loss (625-4356)</td>
<td>Sal&gt;50K</td>
<td>0.50</td>
<td>2.11</td>
<td>0.122</td>
</tr>
<tr>
<td>age (30-80) and education=Doctorate</td>
<td>Sal&gt;50K</td>
<td>0.74</td>
<td>3.10</td>
<td>0.05</td>
</tr>
</tbody>
</table>

We compared our algorithm with MVD, and used the same parameter setting as in [6] where \( \delta = 0.01 \) and the number of items in each initial partition is 10,000. Table 2 shows the results by MVD as reported in [6]. The comparison shows that the cut points found by the 2 algorithms are similar, although the boundaries found by SDAD are more refined. In this experiment, we let the algorithm try different combinations of attributes without pruning, to obtain a complete picture of the attribute relationships. We were able to find rules at a higher level like \( \{\text{age} = (30-80), \text{education} = \text{Doctorate}\} \Rightarrow \{\text{Salary} > 50K\} \) which is not discovered by MVD. This suggests that \( \{\text{age}\} \) alone is not as indicative as \( \{\text{age}, \text{education}\} \) with respect to salary level, especially with specialized degree. We also note that our algorithm may not cover the entire range of the continuous attribute but only find rules in interesting regions.

### 4.3 Case Study: Semiconductor Manufacturing Data
A large amount of information is collected on a per package level as material moves through the packaging and test process. The segment of processing in the manufacture of CPUs of interest to us is that lies between the wafer test and final test operations. Wafer test is the test performed on an entire wafer before it gets singulated and packaged. Final test occurs after the packaging process, and is used to ensure the product is going to perform as designed under specified operating conditions. The data collected is tied to the part identifier and can consist of variables that have continuous, as well as, discrete values. One has parameters that correspond to contextual information related to, for example, the sequence of equipment that processed the part, including relevant subentities (e.g. test heads), material information, along with parametric measurement information and categorical data related to device performance. For the specific dataset considered in this example, the data contains information related to a few tens of thousands parts. Each part consists of a single row in the data table and includes 157 variables (note that this is a very small subset of all the variables collected during the process). The dataset does consist of missing values, which are ignored for the purposes of analysis (note that only the specific missing value is ignored in the calculations the rest of the table row is still used for analysis purposes). Some of the data contained in this table corresponds to the fab wafer id, the X and Y position of the die on the wafer, data collected
at wafer test, and data collected at final test. In addition, the equipment and subentity data is also recorded. What is of interest in this dataset is to understand potential cause of a specific final test failure outcome for a subset of the parts. Identifying attributes that are associated with a specific outcome allows analysts to further investigate and determine the cause(s) of the outcome.

In addition to ranking the rules using interest measures like confidence, lift and leverage, to validate the significance of the rules we discussed our findings with the Intel engineers. The engineers verified that a set of rules describing the association between the presence of a test failure (“Response=1”) and the X and Y locations of the die on the wafer are particularly interesting. To better understand the rules, consider the wafer diagram shown in Figure 5. The number on each shaded square in Figure 5 indicates the number of records with a Response = 1 outcome. If we look closely, we notice that only few locations on the X and Y axes have a (relatively) high number of failures. Note this diagram only shows the numbers of records with the Response = 1 outcome, which sums up to a tiny fraction of the data. Each square in the diagram has a much larger number of records with the Response = 0 outcome (not shown).

In the dataset the X and Y attributes are numeric and ordinal and hence can be grouped into regions. While most rule mining algorithms including our approach would find rules describing the “normal” behaviors (with outcome “Response=0”, or no failure), our algorithm also find rules pointing to the anomalies (“Response=1”). See Table 3 for such rules. While these rules may seem trivial, we want to emphasize that such information is buried among the 157 variables in the data.

As shown in the table, the algorithm finds all the interesting locations. The rules show that the method can quickly discern that the signal of interest is tied to the outer edge of a specific quadrant of the wafer.

We compared our results with other discretization techniques including MVD and decision trees. For MVD, we experimented with different values of δ and chose the value of 0.1 since lower values allow large numbers of bins. We also set the number of initial intervals to be equal to the number of unique elements in each column (the finest possible partition for this case). A smaller number of intervals would miss the anomalies. Hence the quality of MVD results depend on the parameter value for δ and the number of initial intervals. The results for MVD are shown in Table 4. Although MVD finds most of the rules, it fails to identify \{X=2, Y=14\}. In fact, any support-based algorithms would fail to identify any of these rules due to the low support measures. Both algorithms find rules describing the normal outcomes (Response=0). However, such rules found by SDAD describe large “regions” on the wafer, whereas MVD generates a rule for each individual cell.

Next we tried various versions of decision tree implementations available on Weka, and they all fail to identify any useful rules containing X, Y, with Response as the class label. The tree contains only one node — it “classifies” everything into the Response = 0 class. We notice that the confidence is less than 0.5 in each of the rules shown above. Since this is a two class problem, a classifier should predict everything as the majority class i.e Response = 0. For example, given location \{X, Y\} = \{2, 14\}, only 11 parts (records) have Response = 1 outcome, whereas the majority of parts have test outcome “Response = 0.” Naturally, given a specific location like \{2, 14\}, the decision tree would classify it as “normal” (Response = 0). On the other hand, as an association rule problem, these rules are

<table>
<thead>
<tr>
<th>Table 3: Rules found by our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LHS</strong></td>
</tr>
<tr>
<td>{X=7, Y=10}</td>
</tr>
<tr>
<td>{X=4, Y=13}</td>
</tr>
<tr>
<td>{X=2, Y=14}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4: Rules found by MVD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LHS</strong></td>
</tr>
<tr>
<td>{X=7, Y=10}</td>
</tr>
<tr>
<td>{X=4, Y=13}</td>
</tr>
</tbody>
</table>
interesting since we do not see Response = 1 in many locations. When we do see Response = 1, they are likely to occur at the locations shown in Table 3. Hence, these attributes taken together are statistically dependent, as indicated by the high lift values.

In Section 3, we used a small (subset of) Intel dataset to describe the algorithm, and showed that the correct split point was successfully found (see Figure 3). For comparison, decision trees were able to find the correct split location since this subset of data contains only one continuous attribute, while MVD with the standard input parameters as specified in [6] could not due to the locations of the initial bins. The final rules found by our algorithm and decision tree are \{X \geq 13.74\} \rightarrow A and \{X < 13.74\} \rightarrow B, both with confidence of 1. MVD found the rules \{X \geq 14.128\} \rightarrow A and \{X < 14.128\} \rightarrow B if we use 100 basic intervals, and \{X \geq 11.552\} \rightarrow A and \{X < 11.552\} \rightarrow B if we use 10 basic intervals.

4.4 Efficiency We compare the efficiency of our algorithm to that of MVD, both using KORD as the rule mining framework. Both are implemented in Matlab and evaluated using the same dataset (the Intel dataset described in the case study). We randomly picked 20, 40, 60 and 80 continuous attributes to be considered in the antecedent, with Response as the consequent. We repeated each experiment 5 times and recorded the average run time. Since both algorithms use supervised approach, we feel that this is a fair comparison. The goal is to compare the times for the algorithms to find rules with up to 3 attributes in the antecedents with respect to a fixed consequent. See Figure 6 for the comparison.

We note that in some cases SDAD might be slower than MVD. More specifically, consider a dataset with no or little associations between attributes. In such a case, SDAD would not be able to find high-confidence rules at a low dimension, so it would try to combine attributes to find better rules. MVD, however, will likely terminate earlier since its underlying contrast-set algorithm may find significant differences between groups at a low dimension from the initial binning. This gain in efficiency for MVD, however, may not be useful in discovering interesting rules.

4.5 Analysis In the worst case, SDAD keeps splitting until it reaches a single value of the attribute. The maximum number of split is \(m - 1\), where \(m\) is the number of unique values for the attribute. At each iteration, since we only perform counting to calculate the interest measures, the time it requires is \(O(cn)\) where \(n\) is the number of instances. In the worst case \(m = n\), i.e. no two records have the same value for the attribute. The complexity of splitting is thus \(O(n\log_2n)\). Merging has the worst case of \(O(n)\). The initial sorting of the data takes \(O(n\log_2n)\). Therefore, the complexity of the algorithm is \(O(n\log_2n)\). This analysis can be extended to multiple continuous attributes and the complexity will remain the same.

4.6 Parallel Implementation We have shown that SDAD is able to detect better quality rules, but a bottleneck that the algorithm faces is the computation of bins for every combination of attributes. To overcome this, we consider parallelizing the algorithm using the Hadoop framework. The main idea behind Hadoop is to break a problem into smaller and unrelated subproblems. This works perfectly for SDAD since our approach determines appropriate bins for every combination of attributes. Every computation of these bins is independent of each other, unlike MVD, and hence we can exploit this framework. Initial experiments show that we can achieve a large improvement in speed with our parallel implementation.

5 Conclusion

In this work, we present preliminary results on a general framework for quantitative association rule mining. Preliminary results show that our approach discovers more meaningful rules than the state-of-the-art while maintaining competitive time efficiency. The dynamic and adaptive nature of our algorithm makes it possible to capture different inter-variable interactions that may be missed by less sophisticated preprocessing/discretization techniques.

While the methodology and experiments presented in this work focus on static databases, we are currently working on extending the framework to handle continuous monitoring of variable relationships, updating of bins and rules, and detection of anomalies in data streams. We believe that the progress so far indicates the benefits and potential of developing an online, adaptive quantitative rule mining strategy.

References
[16] Han, J., Pei J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. Technical Report CMPT99-12, Simon Fraser University.(1999)
[27] Tong, Q., Yan, B. and Zhou, Y.: Mining quantitative association rules on overlapped intervals, In proceedings of the First international conference on Advanced Data Mining and Applications, Wuhan, China (2005)