Mining Quantitative Association Rules in Large Relational Tables

Ramakrishnan Srikant*
IBM Almaden Research Center
San Jose, CA 95120

Rakesh Agrawal
IBM Almaden Research Center
San Jose, CA 95120

Abstract

We introduce the problem of mining association rules in large relational tables containing both quantitative and categorical attributes. An example of such an association might be "10% of married people between age 50 and 60 have at least 2 cars". We deal with quantitative attributes by fine-partitioning the values of the attribute and then combining adjacent partitions as necessary. We introduce measures of partial completeness which quantify the information lost due to partitioning. A direct application of this technique can generate too many similar rules. We tackle this problem by using a "greater-than-expected-value" interest measure to identify the interesting rules in the output. We give an algorithm for mining such quantitative association rules. Finally, we describe the results of using this approach on a real-life dataset.

1 Introduction

Data mining, also known as knowledge discovery in databases, has been recognized as a new area for database research. The problem of discovering *association rules* was introduced in [AIS93]. Given a set of transactions, where each transaction is a set of items, an association rule is an expression of the from $X \Rightarrow Y$, where X and Y are sets of items. An example of an association rule is: "30% of transactions that contain beer also contain diapers; 2% of all transactions contain both of these items". Here 30% is called the *confidence* of the rule, and 2% the *support* of the rule. The problem is to find all association rules that satisfy user-specified minimum support and minimum confidence constraints.

Conceptually, this problem can be viewed as finding associations between the "1" values in a relational table where all the attributes are boolean. The table has an attribute corresponding to each item and a record corresponding to each transaction. The value of an attribute for a given record is "1" if the item corresponding to the attribute is present in the transaction corresponding to the record, "0" else. In the rest of the paper, we refer to this problem as the **Boolean Association Rules** problem.

Relational tables in most business and scientific domains have richer attribute types. Attributes can be quantitative (e.g. age, income) or categorical (e.g. zip code, make of car). Boolean attributes can be considered a special case of categorical attributes.

In this paper, we define the problem of mining association rules over quantitative and categorical attributes in large relational tables and present techniques for discovering such rules. We refer to this mining problem as the **Quantitative Association Rules** problem. We give a formal statement of the problem in Section 2. For illustration, Figure 1 shows a People table with three non-key attributes. Age and NumCars are quantitative attributes, whereas Married is a categorical attribute. A quantitative association rule present in this table is: $\langle$Age: 30..39$\rangle$ and $\langle$Married: Yes$\rangle$ $\Rightarrow$ $\langle$NumCars: 2$\rangle$.

1.1 Mapping the Quantitative Association Rules Problem into the Boolean Association Rules Problem

Let us examine whether the Quantitative Association Rules problem can be mapped to the Boolean Association Rules problem. If all attributes are categorical or the quantitative attributes have only a few values, this mapping is straightforward. Conceptually, instead of having just one field in the table for each attribute, we have as many fields as the number of attribute values. The value of a boolean field corresponding to $\langle$*attribute1, value1*$\rangle$ would be "1" if *attribute1* had *value1* in the original record, and "0" otherwise. If the domain of values for a quantitative approach is large, an obvious approach will be to first partition the values into intervals and then map each $\langle$attribute, interval$\rangle$ pair to a boolean attribute. We can now use any algorithm for finding Boolean Association Rules (e.g. [AS94]) to find

*Also, Department of Computer Science, University of Wisconsin, Madison.

People

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

(minimum support = 40%, minimum confidence = 50%)

Rules (Sample)	Support	Confidence
$\langle$Age: 30..39$\rangle$ and $\langle$Married: Yes$\rangle$ $\Rightarrow$ $\langle$NumCars: 2$\rangle$	40%	100%
$\langle$NumCars: 0..1$\rangle$ $\Rightarrow$ $\langle$Married: No$\rangle$	40%	66.6%

Figure 1: Example of Quantitative Association Rules

quantitative association rules.

Figure 2 shows this mapping for the non-key attributes of the People table given in Figure 1. Age is partitioned into two intervals: 20..29 and 30..39. The categorical attribute, Married, has two boolean attributes "Married: Yes" and "Married: No". Since the number of values for NumCars is small, NumCars is not partitioned into intervals; each value is mapped to a boolean field. Record 100, which had $\langle$Age: 23$\rangle$ now has "Age: 20..29" equal to "1", "Age: 30..39" equal to "0", etc.

Mapping Woes. There are two problems with this simple approach when applied to quantitative attributes:

- *"MinSup".* If the number of intervals for a quantitative attribute (or values, if the attribute is not partitioned) is large, the support for any single interval can be low. Hence, without using larger intervals, some rules involving this attribute may not be found because they lack minimum support.

- *"MinConf".* There is some information lost whenever we partition values into intervals. Some rules may have minimum confidence only when an item in the antecedent consists of a single value (or a small interval). This information loss increases as the interval sizes become larger.

  For example, in Figure 2, the rule "$\langle$NumCars: 0$\rangle$ $\Rightarrow$ $\langle$Married: No$\rangle$" has 100% confidence. But if we had partitioned the attribute NumCars into intervals such that 0 and 1 cars end up in the same partition, then the closest rule is "$\langle$NumCars: 0..1$\rangle$ $\Rightarrow$ $\langle$Married: No$\rangle$", which only has 66.6% confidence.

There is a "catch-22" situation created by these two problems – if the intervals are too large, some rules may not have minimum confidence; if they are too small, some rules may not have minimum support.

Breaking the logjam. To break the above catch-22 situation, we can consider all possible continuous ranges over the values of the quantitative attribute, or over the partitioned intervals. The ""MinSup" problem now disappears since we can combine adjacent intervals/values. The "MinConf" problem is still present; however, the information loss can be reduced by increasing the number of intervals, without encountering the "MinSup" problem.

Unfortunately, increasing the number of intervals while simultaneously combining adjacent intervals introduces two new problems:

- *"ExecTime".* If a quantitative attribute has n values (or intervals), there are on average $O(n^2)$ ranges that include a specific value or interval. Hence the number of items per record blows up, which will blow up the execution time.

- *"ManyRules".* If a value (or interval) of a quantitative attribute has minimum support, so will any range containing this value/interval. Thus, the number of rules blows up. Many of these rules will not be interesting (as we will see later).

There is a tradeoff between faster execution time with fewer intervals (mitigating "ExecTime") and reducing information loss with more intervals (mitigating "MinConf"). We can reduce the information loss by increasing the number of intervals, at the cost of increasing the execution time and potentially generating many uninteresting rules ("ManyRules" problem).

It is not meaningful to combine categorical attribute values unless unless a taxonomy (*is-a* hierarchy) is present on the attribute. In this case, the taxonomy can be used to implicitly combine values of a categorical attribute (see [SA95], [HF95]). Using a taxonomy in this manner is somewhat similar to considering ranges over quantitative attributes.

RecID	Age: 20..29	Age: 30..39	Married: Yes	Married: No	NumCars: 0	NumCars: 1	NumCars: 2
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

Figure 2: Mapping to Boolean Association Rules Problem

1.2 Our Approach

We consider ranges over adjacent values/intervals of quantitative attributes to avoid the "MinSup" problem. To mitigate the "ExecTime" problem, we restrict the extent to which adjacent values/intervals may be combined by introducing a user-specified "maximum support" parameter; we stop combining intervals if their combined support exceeds this value. However, any single interval/value whose support exceeds maximum support is still considered.

But how do we decide whether to partition a quantitative attribute or not? And how many partitions should there be in case we do decide to partition? We introduce a *partial completeness measure* in Section 3 that gives a handle on the information lost by partitioning and helps make these decisions.

To address the "ManyRules" problem, we give an *interest measure* in Section 4. The interest measure is based on deviation from expectation and helps prune out uninteresting rules. This measure is an extension of the interest-measure introduced in [SA95].

We give the algorithm for discovering quantitative association rules in Section 5. This algorithm shares the basic structure of the algorithm for finding boolean association rules given in [AS94]. However, to yield a fast implementation, the computational details of how candidates are generated and how their supports are counted are new.

We present our experience with this solution on a real-life dataset in Section 6.

1.3 Related Work

Since the introduction of the (Boolean) Association Rules problem in [AIS93], there has been considerable work on designing algorithms for mining such rules [AS94] [HS95] [MTV94] [SON95] [PCY95]. This work was subsequently extended to finding association rules when there is a taxonomy on the items in [SA95] [HF95].

Related work also includes [PS91], where quantitative rules of the from $x = q_x \Rightarrow y = q_y$ are discovered. However, the antecedent and consequent are constrained to be a single $\langle$attribute,value$\rangle$ pair. There are suggestions about extending this to rules where the antecedent is of the from $l \leq x \leq u$. This is done by partitioning the quantitative attributes into intervals; however, the intervals are not combined. The algorithm in [PS91] is fairly straightforward. To find the rules comprising $(A = a)$ as the antecedent, where a is a specific value of the attribute A, one pass over the data is made and each record is hashed by values of A. Each hash cell keeps a running summary of values of other attributes for the records with the same A value. The summary for $(A = a)$ is used to derive rules implied by $(A = a)$ at the end of the pass. To find rules for different attributes, the algorithm is run once on each attribute. Thus if we are interested in finding all rules, we must find these summaries for all combinations of attributes, which is exponentially large.

2 Problem Statement and Decomposition

We now give a formal statement of the problem of mining Quantitative Association Rules and introduce some terminology.

We use a simple device to treat categorical and quantitative attributes uniformly. For categorical attributes, the values of the attribute are mapped to a set of consecutive integers. For quantitative attributes that are not partitioned into intervals, the values are mapped to consecutive integers such that the order of the values is preserved. If a quantitative attribute is partitioned into intervals, the intervals are mapped to consecutive integers, such that the order of the intervals is preserved. These mappings let us treat a database record as a set of $\langle$attribute, integer value$\rangle$ pairs, without loss of generality.

Now, let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called attributes. Let $\mathcal{P}$ denote the set of positive integers. Let $\mathcal{I}_\mathcal{V}$ denote the set $\mathcal{I} \times \mathcal{P}$. A pair $\langle x, v\rangle \in \mathcal{I}_\mathcal{V}$ denotes the attribute x, with the associated value v. Let $\mathcal{I}_\mathcal{R}$ denote the set $\{\langle x, l, u\rangle \in \mathcal{I} \times \mathcal{P} \times \mathcal{P} \mid l \leq u$, if x is quantitative; $l = u$, if x is categorical $\}$. Thus, a triple $\langle x, l, u\rangle \in \mathcal{I}_\mathcal{R}$ denotes either a quantitative attribute x with a value in the interval $[l, u]$, or a categorical attribute x with a value l. We will refer to this triple as an *item*. For any $X \subseteq \mathcal{I}_\mathcal{R}$, let *attributes*$(X)$ denote the set $\{x \mid \langle x, l, u\rangle \in X\}$.

Note that with the above definition, only values are associated with categorical attributes, while both values and ranges may be associated with quantitative attributes. In other words, values of categorical attributes are not combined.

Let $\mathcal{D}$ be a set of records, where each record R is a set of attribute values such that $R \subseteq \mathcal{I}_{\mathcal{V}}$. We assume that each attribute occurs at most once in a record. We say that a record R *supports* $X \subseteq \mathcal{I}_{\mathcal{R}}$, if $\forall \langle x, l, u \rangle \in X$ $(\exists \langle x, q \rangle \in R$ such that $l \leq q \leq u)$.

A *quantitative association rule* is an implication of the form $X \Rightarrow Y$, where $X \subset \mathcal{I}_{\mathcal{R}}$, $Y \subset \mathcal{I}_{\mathcal{R}}$, and $\text{attributes}(X) \cap \text{attributes}(Y) = \emptyset$. The rule $X \Rightarrow Y$ holds in the record set $\mathcal{D}$ with *confidence* c if c% of records in $\mathcal{D}$ that support X also support Y. The rule $X \Rightarrow Y$ has *support* s in the record set $\mathcal{D}$ if s% of records in $\mathcal{D}$ support $X \cup Y$.

Given a set of records $\mathcal{D}$, the problem of mining quantitative association rules is to find all quantitative association rules that have support and confidence greater than the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*) respectively. Note that the fact that items in a rule can be categorical or quantitative has been hidden in the definition of an association rule.

Notation Recall that an *item* is a triple that represents either a categorical attribute with its value, or a quantitative attribute with its range. (The value of a quantitative attribute can be represented as a range where the upper and lower limits are the same.) We use the term *itemset* to represent a set of items. The support of an itemset $X \subset \mathcal{I}_{\mathcal{R}}$ is simply the percentage of records in $\mathcal{D}$ that support X. We use the term *frequent itemset* to represent an itemset with minimum support.

Let $\Pr(X)$ denote the probability that *all* the items in $X \subseteq \mathcal{I}_{\mathcal{R}}$ are supported by a given record. Then $support(X \Rightarrow Y) = \Pr(X \cup Y)$ and $confidence(X \Rightarrow Y) = \Pr(Y \mid X)$. (Note that $\Pr(X \cup Y)$ is the probability that all the items in $X \cup Y$ are present in the record.) We call $\widehat{X}$ a *generalization* of X (and X a *specialization* of $\widehat{X}$) if $\text{attributes}(X) = \text{attributes}(\widehat{X})$ and $\forall x \in \text{attributes}(X)$ $[\langle x, l, u \rangle \in X \wedge \langle x, l', u' \rangle \in \widehat{X} \Rightarrow l' \leq l \leq u \leq u']$. For example, the itemset { ⟨Age: 30..39⟩, ⟨Married: Yes⟩ } is a generalization of { ⟨Age: 30..35⟩, ⟨Married: Yes⟩ }.

2.1 Problem Decomposition

We solve the problem of discovering quantitative association rules in five steps:

1. Determine the number of partitions for each quantitative attribute. (See Section 3.)

2. For categorical attributes, map the values of the attribute to a set of consecutive integers. For quantitative attributes that are not partitioned into intervals, the values are mapped to consecutive integers such that the order of the values is preserved. If a quantitative attribute is partitioned into intervals, the intervals are mapped to consecutive integers, such that the order of the intervals is preserved. From this point, the algorithm only sees values (or ranges over values) for quantitative attributes. That these values may represent intervals is transparent to the algorithm.

3. Find the support for each value of both quantitative and categorical attributes. Additionally, for quantitative attributes, adjacent values are combined as long as their support is less than the user-specified max support. We now know all ranges and values with minimum support for each quantitative attribute, as well as all values with minimum support for each categorical attribute. These form the set of all *frequent* items.

   Next, find all sets of items whose support is greater than the user-specified minimum support. These are the *frequent itemsets*. (See Section 5.)

4. Use the frequent itemsets to generate association rules. The general idea is that if, say, $ABCD$ and AB are frequent itemsets, then we can determine if the rule $AB \Rightarrow CD$ holds by computing the ratio $conf = \text{support}(ABCD)/\text{support}(AB)$. If $conf \geq minconf$, then the rule holds. (The rule will have minimum support because $ABCD$ is frequent.) We use the algorithm in [AS94] to generate rules.

5. Determine the interesting rules in the output. (See Section 4.)

Example Consider the "People" table shown in Figure 3a. There are two quantitative attributes, Age and NumCars. Assume that in Step 1, we decided to partition Age into 4 intervals, as shown in Figure 3b. Conceptually, the table now looks as shown in Figure 3c. After mapping the intervals to consecutive integers, using the mapping in Figure 3d, the table looks as shown in Figure 3e. Assuming minimum support of 40% and minimum confidence of 50%, Figure 3f shows some of the frequent itemsets, and Figure 3g some of the rules. We have replaced mapping numbers with the values in the original table in these two figures. Notice that the item ⟨Age: 20..29⟩ corresponds to a combination of the intervals 20..24 and 25..29, etc. We have not shown the step of determining the interesting rules in this example.

3 Partitioning Quantitative Attributes

In this section, we consider when we should partition the values of quantitative attributes into intervals, and how many partitions there should be. First, we present a measure of partial completeness which gives a handle on the amount of information lost by partitioning. We then show that equi-depth partitioning minimizes the number of intervals required to satisfy this partial

Minimum Support = 40% = 2 records
Minimum Confidence = 50%

People

RecordID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

(a)

Partitions for Age

Interval
20..24
25..29
30..34
35..39

(b)

After partitioning Age

RecordID	Age	Married	NumCars
100	20..24	No	0
200	25..29	Yes	1
300	25..29	No	1
400	30..34	Yes	2
500	35..39	Yes	2

(c)

Mapping Age

Interval	Integer
20..24	1
25..29	2
30..34	3
35..39	4

Mapping Married

Value	Integer
Yes	1
No	2

(d)

After mapping attributes

RecordID	Age	Married	NumCars
100	1	2	0
200	2	1	1
300	2	2	1
400	3	1	2
500	4	1	2

(e)

Frequent Itemsets: Sample

Itemset	Support
{ ⟨Age: 20..29⟩ }	3
{ ⟨Age: 30..39⟩ }	2
{ ⟨Married: Yes⟩ }	3
{ ⟨Married: No⟩ }	2
{ ⟨NumCars: 0..1⟩ }	3
{ ⟨Age: 30..39⟩, ⟨Married: Yes⟩ }	2

(f)

Rules: Sample

Rule	Support	Confidence
⟨Age: 30..39⟩ and ⟨Married: Yes⟩ ⇒ ⟨NumCars: 2⟩	40%	100%
⟨Age: 20..29⟩ ⇒ ⟨NumCars: 0..1⟩	60%	66.6%

(g)

Figure 3: Example of Problem Decomposition

completeness level. Thus equi-depth partitioning is, in some sense, optimal for this measure of partial completeness.

The intuition behind the partial completeness measure is as follows. Let R be the set of rules obtained by considering all ranges over the raw values of quantitative attributes. Let R' be the set of rules obtained by considering all ranges over the partitions of quantitative attributes. One way to measure the information loss when we go from R to R' is to see for each rule in R, how "far" the "closest" rule in R' is. The further away the closest rule, the greater the loss. By defining "close" rules to be generalizations, and using the ratio of the support of the rules as a measure of how far apart the rules are, we derive the measure of partial completeness given below.

3.1 Partial Completeness

We first define partial completeness over itemsets rather than rules, since we can guarantee that a close itemset will be found whereas we cannot guarantee that a close rule will be found. We then show that we can guarantee that a close rule will be found if the minimum confidence level for R' is less than that for R by a certain (computable) amount.

Let $\mathcal{C}$ denote the set of all frequent itemsets in $\mathcal{D}$. For any $K \geq 1$, we call $\mathcal{P}$ K-complete with respect to $\mathcal{C}$ if

- $\mathcal{P} \subseteq \mathcal{C}$,
- $X \in \mathcal{P}$ and $X' \subseteq X \Rightarrow X' \in \mathcal{P}$, and

- $\forall X \in \mathcal{C}$ $[\exists \widehat{X} \in \mathcal{P}$ such that
  (i) $\widehat{X}$ is a generalization of X and support$(\widehat{X}) \leq K \times$ support(X), and
  (ii) $\forall Y \subseteq X$ $\exists \widehat{Y} \subseteq \widehat{X}$ such that $\widehat{Y}$ is a generalization of Y and support$(\widehat{Y}) \leq K \times$ support$(Y)]$.

The first two conditions ensure that $\mathcal{P}$ only contains frequent itemsets and that we can generate rules from $\mathcal{P}$. The first part of the third condition says that for any itemset in $\mathcal{C}$, there is a generalization of that itemset with at most K times the support in $\mathcal{P}$. The second part says that the property that the generalization has at most K times the support also holds for corresponding subsets of attributes in the itemset and its generalization. Notice that if $K = 1$, $\mathcal{P}$ becomes identical to $\mathcal{C}$.

For example, assume that in some table, the following are the frequent itemsets $\mathcal{C}$:

Number	Itemset	Support
1	{ ⟨Age: 20..30⟩ }	5%
2	{ ⟨Age: 20..40⟩ }	6%
3	{ ⟨Age: 20..50⟩ }	8%
4	{ ⟨Cars: 1..2⟩ }	5%
5	{ ⟨Cars: 1..3⟩ }	6%
6	{ ⟨Age: 20..30⟩, ⟨Cars: 1..2⟩ }	4%
7	{ ⟨Age: 20..40⟩, ⟨Cars: 1..3⟩ }	5%

The itemsets 2, 3, 5 and 7 would from a 1.5-complete set, since for any itemset X, either 2, 3, 5 or 7 is a generalization whose support is at most 1.5 times the support of X. For instance, itemset 2 is a generalization of itemset 1, and the support of itemset 2 is 1.2 times the support of itemset 1. Itemsets 3, 5 and 7 do not form a 1.5-complete set because for itemset 1, the only generalization among 3, 5 and 7 is itemset 3, and the support of 3 is more than 1.5 times the support of 1.

Lemma 1 *Let $\mathcal{P}$ be a K-complete set w.r.t. $\mathcal{C}$, the set of all frequent itemsets. Let $\mathcal{R_C}$ be the set of rules generated from $\mathcal{C}$, for a minimum confidence level* minconf. *Let $\mathcal{R_P}$ be the set of rules generated from $\mathcal{P}$ with the minimum confidence set to* minconf$/K$. *Then for any rule $A \Rightarrow B$ in $\mathcal{R_C}$, there is a rule $\widehat{A} \Rightarrow \widehat{B}$ in $\mathcal{R_P}$ such that*

- *$\widehat{A}$ is a generalization of A, $\widehat{B}$ is a generalization of B,*
- *the support of $\widehat{A} \Rightarrow \widehat{B}$ is at most K times the support of $A \Rightarrow B$, and*
- *the confidence of $\widehat{A} \Rightarrow \widehat{B}$ is at least $1/K$ times, and at most K times the confidence of $A \Rightarrow B$.*

Proof: Parts 1 and 2 follow directly from the definition of K-completeness. We now prove Part 3. Let $A \Rightarrow B$ be a rule in $\mathcal{R_C}$. Then there is an itemset $A \cup B$ in $\mathcal{C}$. By definition of a K-complete set, there is an itemset $\widehat{A} \cup \widehat{B}$ in $\mathcal{P}$ such that (i) support$(\widehat{A} \cup \widehat{B}) \leq K \times$support$(A \cup B)$, and (ii) support$(\widehat{A}) \leq K \times$ support(A). The confidence of the rule $\widehat{A} \Rightarrow \widehat{B}$ (generated from $\widehat{A} \cup \widehat{B}$) is given by support$(\widehat{A} \cup \widehat{B})$/support$(\widehat{A})$. Hence

$$\frac{\text{confidence}(\widehat{A} \Rightarrow \widehat{B})}{\text{confidence}(A \Rightarrow B)} = \frac{\frac{\text{support}(\widehat{A} \cup \widehat{B})}{\text{support}(\widehat{A})}}{\frac{\text{support}(A \cup B)}{\text{support}(A)}} = \frac{\frac{\text{support}(\widehat{A} \cup \widehat{B})}{\text{support}(A \cup B)}}{\frac{\text{support}(\widehat{A})}{\text{support}(A)}}$$

Since both $\frac{\text{support}(\widehat{A} \cup \widehat{B})}{\text{support}(A \cup B)}$ and $\frac{\text{support}(\widehat{A})}{\text{support}(A)}$ lie between 1 and K (inclusive), the confidence of $\widehat{A} \Rightarrow \widehat{B}$ must be between $1/K$ and K times the confidence of $A \Rightarrow B$. □

Thus, given a set of frequent itemsets $\mathcal{P}$ which is K-complete w.r.t. the set of all frequent itemsets, the minimum confidence when generating rules from $\mathcal{P}$ must be set to $1/K$ times the desired level to guarantee that a close rule will be generated.

In the example given earlier, itemsets 2, 3 and 5 form a 1.5-complete set. The rule "⟨Age: 20..30⟩ ⇒ ⟨Cars: 1..2⟩" has 80% confidence, while the corresponding generalized rule "⟨Age: 20..40⟩ ⇒ ⟨Cars: 1..3⟩" has 83.3% confidence.

3.2 Determining the number of Partitions

We first prove some properties of partitioned attributes (w.r.t. partial completeness), and then use these properties to decide the number of intervals given the partial completeness level.

Lemma 2 *Consider a quantitative attribute x, and some real $K > 1$. Assume we partition x into intervals (called* base *intervals) such that for any base interval B, either the support of B is less than* minsup $\times (K-1)/2$ *or B consists of a single value. Let $\mathcal{P}$ denote the set of all combinations of base intervals that have minimum support. Then $\mathcal{P}$ is K-complete w.r.t. the set of all ranges over x with minimum support.*

Proof: Let X be any interval with minimum support, and $\widehat{X}$ the smallest combination of base intervals which is a generalization of X (see Figure 4). There are at most two base intervals, one at each end, which are only partially spanned by X. Consider either of these intervals. If X only partially spans this interval, the interval cannot be just a single value. Hence the support of this interval, as well as the support of the portion of the interval not spanned by X, must be less than *minsup* $\times (K-1)/2$. Thus

$$\begin{aligned}
\text{support}(\widehat{X}) &\leq \text{support}(X) + 2 \times \textit{minsup} \times (K-1)/2 \\
&\leq \text{support}(X) + \text{support}(X) \times (K-1) \\
&\quad \text{(since support}(X) > \textit{minsup}) \\
&\leq \text{support}(X) \times K
\end{aligned}$$

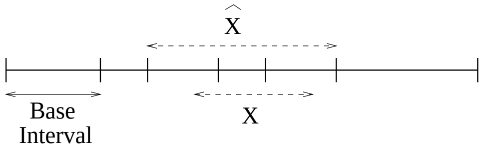

Figure 4: Illustration for Lemma 2

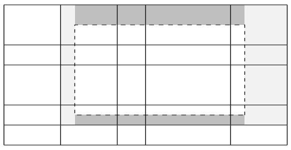

Figure 5: Example for Lemma 3

□

Lemma 3 *Consider a set of n quantitative attributes, and some real $K > 1$. Assume each quantitative attribute is partitioned such that for any base interval B, either the support of B is less than* $\text{minsup} \times (K-1)/(2 \times n)$ *or B consists of a single value. Let $\mathcal{P}$ denote the set of all frequent itemsets over the partitioned attributes. Then $\mathcal{P}$ is K-complete w.r.t the set of all frequent itemsets (obtained without partitioning).*

Proof: The proof is similar to that for Lemma 2. However, the difference in support between an itemset X and its generalization $\widehat{X}$ may be $2m$ times the support of a single base interval for a single attribute, where m is the number of quantitative attributes in X. Since X may have upto n attributes, the support of each base interval must be at most $\textit{minsup} \times (K-1)/(2 \times n)$, rather than just $\textit{minsup} \times (K-1)/2$ for $\mathcal{P}$ to be K-complete. A similar argument applies to subsets of X.

An illustration of this proof for 2 quantitative attributes is shown in Figure 5. The solid lines correspond to partitions of the attributes, and the dashed rectangle corresponds to an itemset X. The shaded areas show the extra area that must be covered to get its generalization $\widehat{X}$ using partitioned attributes. Each of the 4 shaded areas spans less than a single partition of a single attribute. (One partition of one attribute corresponds to a band from one end of the rectangle to another.) □

For any given partitioning, we can use Lemma 3 to compute the level of partial completeness for that partitioning. We first illustrate the procedure for a single attribute. In this case, we simply find the partition with highest support among those with more than one value. Let the support of this partition be s. Then, to find the partial completeness level K, we use the formula $s = \textit{minsup} \times (K-1)/2$ from Lemma 2 to get $K = 1 + 2 \times s/\textit{minsup}$. With n attributes, the formula becomes

$$K = 1 + \frac{2 \times n \times s}{\textit{minsup}} \qquad (1)$$

where s is the maximum support for a partition with more than one value, among all the quantitative attributes. Recall that the lower the level of partial completeness, the less the information lost. The formula reflects this: as s decreases, implying more intervals, the partial completeness level decreases.

Lemma 4 *For any specified number of intervals, equi-depth partitioning minimizes the partial completeness level.*

Proof: From Lemma 3, if the support of each base interval is less than $\textit{minsup} \times (K-1)/(2 \times n)$, the partial completeness level is K. Since the maximum support of any base interval is minimized with equi-depth partitioning, equi-depth partitioning results in the lowest partial completeness level. □

Corollary 1 *For a given partial completeness level, equi-depth partitioning minimizes the number of intervals required to satisfy that partial completeness level.*

Given the level of partial completeness desired by the user, and the minimum support, we can calculate the number of partitions required (assuming equi-depth partitioning). From Lemma 3, we know that to get a partial completeness level K, the support of any partition with more than one value should be less than $\textit{minsup} * (K-1)/(2 \times n)$ where n is the number of quantitative attributes. Ignoring the special case of partitions that contain just one value[1], and assuming that equi-depth partitioning splits the support identically, there should be $1/s$ partitions in order to get the support of each partition to less than s. Thus we get

$$\text{Number of Intervals} = \frac{2 \times n}{m \times (K-1)} \qquad (2)$$

where

$$\begin{aligned} n &= \text{Number of Quantitative Attributes} \\ m &= \text{Minimum Support (as a fraction)} \\ K &= \text{Partial Completeness Level} \end{aligned}$$

If there are no rules with more than n' quantitative attributes, we can replace n with n' in the above formula (see proof of Lemma 3).

4 Interest

A potential problem with combining intervals for quantitative attributes is that the number of rules found may be very large. [ST95] looks at subjective measures of interestingness and suggests that a pattern is interesting if

[1] While this may overstate the number of partitions required, it will not increase the partial completeness level.

it is unexpected (surprising to the user) and/or actionable (the user can do something with it). [ST95] also distinguishes between subjective and objective interest measures. [PS91] discusses a class of objective interest measures based on how much the support of a rule deviates from what the support would be if the antecedent and consequent of the rule were independent.

In this section, we present a "greater-than-expected-value" interest measure to identify the interesting rules in the output. This interest measure looks at both generalizations and specializations of the rule to identify the interesting rules.

To motivate our interest measure, consider the following rules, where about a quarter of people in the age group 20..30 are in the age group 20..25.

⟨Age: 20..30⟩ ⇒ ⟨Cars: 1..2⟩ (8% sup., 70% conf.)
⟨Age: 20..25⟩ ⇒ ⟨Cars: 1..2⟩ (2% sup., 70% conf.)

The second rule can be considered redundant since it does not convey any additional information and is less general than the first rule. Given the first rule, we expect that the second rule would have the same confidence as the first and support equal to a quarter of the support for the first. Even if the confidence of the second rule was a little different, say 68% or 73%, it does not convey significantly more information than the first rule. We try to capture this notion of "interest" by saying that we only want to find rules whose support and/or confidence is greater than expected. (The user can specify whether it should be support *and* confidence, or support *or* confidence.) We now formalize this idea, after briefly describing related work.

Expected Values. Let $E_{\Pr(\widehat{Z})}[\Pr(Z)]$ denote the "expected" value of $\Pr(Z)$ (that is, the support of Z) based on $\Pr(\widehat{Z})$, where $\widehat{Z}$ is a generalization of Z. Let Z be the itemset $\{\langle z_1, l_1, u_1\rangle, \ldots, \langle z_n, l_n, u_n\rangle\}$ and $\widehat{Z}$ the set $\{\langle z_1, l'_1, u'_1\rangle, \ldots, \langle z_n, l'_n, u'_n\rangle\}$ (where $l'_i \le l_i \le u_i \le u'_i$). Then we define

$$E_{\Pr(\widehat{Z})}[\Pr(Z)] = \frac{\Pr(\langle z_1, l_1, u_1\rangle)}{\Pr(\langle z_1, l'_1, u'_1\rangle)} \times \cdots \times \frac{\Pr(\langle z_n, l_n, u_n\rangle)}{\Pr(\langle z_n, l'_n, u'_n\rangle)} \times \Pr(\widehat{Z})$$

Similarly, we $E_{\Pr(\widehat{Y} \mid \widehat{X})}[\Pr(Y \mid X)]$ denote the "expected" confidence of the rule $X \Rightarrow Y$ based on the rule $\widehat{X} \Rightarrow \widehat{Y}$, where $\widehat{X}$ and $\widehat{Y}$ are generalizations of X and Y respectively. Let Y be the itemset $\{\langle y_1, l_1, u_1\rangle, \ldots, \langle y_n, l_n, u_n\rangle\}$ and $\widehat{Y}$ the set $\{\langle y_1, l'_1, u'_1\rangle, \ldots, \langle y_n, l'_n, u'_n\rangle\}$. Then we define

$$E_{\Pr(\widehat{Y} \mid \widehat{X})}[\Pr(Y \mid X)] = \frac{\Pr(\langle y_1, l_1, u_1\rangle)}{\Pr(\langle y_1, l'_1, u'_1\rangle)} \times \cdots \times \frac{\Pr(\langle y_n, l_n, u_n\rangle)}{\Pr(\langle y_n, l'_n, u'_n\rangle)} \times \Pr(\widehat{Y} \mid \widehat{X})$$

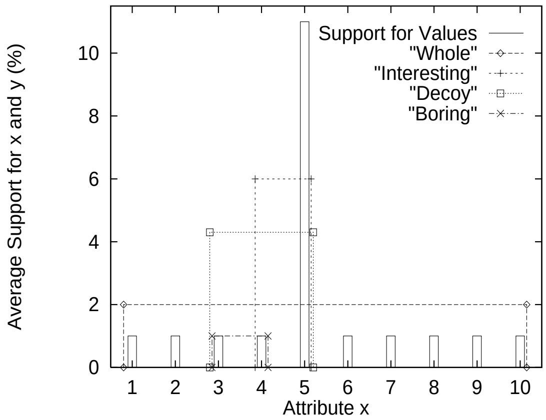

Figure 6: Example for Interest

A Tentative Interest Measure. We first introduce a measure similar to the one used in [SA95].

An itemset Z is *R-interesting* w.r.t an ancestor $\widehat{Z}$ if the support of Z is greater than or equal to R times the expected support based on $\widehat{Z}$. A rule $X \Rightarrow Y$ is *R-interesting* w.r.t an ancestor $\widehat{X} \Rightarrow \widehat{Y}$ if the support of the rule $X \Rightarrow Y$ is R times the expected support based on $\widehat{X} \Rightarrow \widehat{Y}$, or the confidence is R times the expected confidence based on $\widehat{X} \Rightarrow \widehat{Y}$.

Given a set of rules, we call $\widehat{X} \Rightarrow \widehat{Y}$ a *close ancestor* of $X \Rightarrow Y$ if there is no rule $X' \Rightarrow Y'$ such that $\widehat{X} \Rightarrow \widehat{Y}$ is an ancestor of $X' \Rightarrow Y'$ and $X' \Rightarrow Y'$ is an ancestor of $X \Rightarrow Y$. A similar definition holds for itemsets.

Given a set of rules S and a minimum interest R, a rule $X \Rightarrow Y$ is *interesting* (in S) if it has no ancestors or it is R-interesting with respect to its close ancestors among its interesting ancestors.

Why looking at generalizations is insufficient. The above definition of interest has the following problem. Consider a single attribute x with the range [1,10], and another categorical attribute y. Assume the support for the values of x are uniformly distributed. Let the support for values of x together with y be as shown in Figure 6. For instance, the support of $(\langle x, 5\rangle, y)$ = 11%, and the support for $(\langle x, 1\rangle, y)$ = 1%. This figure also shows the "average" support for the itemsets $(\langle x, 1, 10\rangle, y)$, $(\langle x, 3, 5\rangle, y)$, $(\langle x, 3, 4\rangle, y)$ and $(\langle x, 4, 5\rangle, y)$. Clearly, the only "interesting" set is $\{\langle x, 5, 5\rangle, y\}$. However, the interest measure given above may also find other itemsets "interesting". For instance, with an interest level of 2, interval "Decoy", $\{\langle x, 3, 5\rangle, y\}$ would also be considered interesting, as would $\{\langle x, 4, 6\rangle, y\}$ and $\{\langle x, 5, 7\rangle, y\}$.

If we had the support for each value of x along with y, it is easy to check that all specializations of an itemset are also interesting. However, in general, we will not have this information, since a single value of x together with y may not have minimum support. We will only have information about those specializations of x which (along with y) have minimum support. For instance, we may only have information about the support for the subinterval "Interesting" (for interval "Decoy").

An obvious way to use this information is to check whether there are any specializations with minimum support that are not interesting. However, there are two problem with this approach. First, there may not be any specializations with minimum support that are not interesting. This case is true in the example given above unless the minimum support is less than or equal to 2%. Second, even if there are such specializations, there may not be any specialization with minimum support that *are* interesting. We do not want to discard the current itemset unless there is a specialization with minimum support that is interesting and some part of the current itemset is not interesting.

An alternative approach is to check whether there are any specializations that are more interesting than the itemset, and then subtract the specialization from the current itemset to see whether or not the difference is interesting. Notice that the difference need not have minimum support. Further, if there are no such specializations, we would want to keep this itemset. Thus this approach is clearly preferred. We therefore change the definitions of interest given earlier to reflect these ideas.

Final Interest Measure. An itemset X is *R-interesting* with respect to $\widehat{X}$ if the support of X is greater than or equal to R times the expected support based on $\widehat{X}$ and for any specialization X' such that X' has minimum support and $X - X' \subseteq \mathcal{I}_{\mathcal{R}}$, $X - X'$ is R-interesting with respect to $\widehat{X}$.

Similarly, a rule $X \Rightarrow Y$ is *R-interesting* w.r.t an ancestor $\widehat{X} \Rightarrow \widehat{Y}$ if the support of the rule $X \Rightarrow Y$ is R times the expected support based on $\widehat{X} \Rightarrow \widehat{Y}$, or the confidence is R times the expected confidence based on $\widehat{X} \Rightarrow \widehat{Y}$, and the itemset $X \cup Y$ is R-interesting w.r.t $\widehat{X} \cup \widehat{Y}$.

Note that with the specification of the interest level, the specification of the minimum confidence parameter can optionally be dropped. The semantics in that case will be that we are interested in all those rules that have interest above the specified interest level.

5 Algorithm

In this section, we describe the algorithm for finding all frequent itemsets (Step 3 of the problem decomposition given in Section 2.1). At this stage, we have already partitioned quantitative attributes, and created combinations of intervals of the quantitative attributes that have minimum support. These combinations, along with those values of categorical attributes that have minimum support, form the frequent items.

Starting with the frequent items, we generate all frequent itemsets using an algorithm based on the Apriori algorithm for finding boolean association rules given in [AS94]. The proposed algorithm extends the candidate generation procedure to add pruning using the interest measure, and uses a different data structure for counting candidates.

Let k-itemset denote an itemset having k items. Let L_k represent the set of frequent k-itemsets, and C_k the set of candidate k-itemsets (potentially frequent itemsets). The algorithm makes multiple passes over the database. Each pass consists of two phases. First, the set of all frequent $(k{-}1)$-itemsets, L_{k-1}, found in the $(k{-}1)$th pass, is used to generate the candidate itemsets C_k. The candidate generation procedure ensures that C_k is a superset of the set of all frequent k-itemsets. The algorithm now scans the database. For each record, it determines which of the candidates in C_k are contained in the record and increments their support count. At the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding L_k. The algorithm terminates when L_k becomes empty.

We now discuss how to generate candidates and count their support.

5.1 Candidate Generation

Given L_{k-1}, the set of all frequent $k-1$-itemsets, the candidate generation procedure must return a superset of the set of all frequent k-itemsets. This procedure has three parts:

1. **Join Phase**. L_{k-1} is joined with itself, the join condition being that the lexicographically ordered first $k-2$ items are the same, and that the attributes of the last two items are different. For example, let L_2 consist of the following itemsets:

   { ⟨Married: Yes⟩ ⟨Age: 20..24⟩ }
   { ⟨Married: Yes⟩ ⟨Age: 20..29⟩ }
   { ⟨Married: Yes⟩ ⟨NumCars: 0..1⟩ }
   { ⟨Age: 20..29⟩ ⟨NumCars: 0..1⟩ }

   After the join step, C_3 will consist of the following itemsets:

   { ⟨Married: Yes⟩ ⟨Age: 20..24⟩ ⟨NumCars: 0..1⟩ }
   { ⟨Married: Yes⟩ ⟨Age: 20..29⟩ ⟨NumCars: 0..1⟩ }

2. **Subset Prune Phase** All itemsets from the join result which have some $(k-1)$-subset that is not in L_{k-1} are deleted. Continuing the earlier example, the prune step will delete the itemset { ⟨Married: Yes⟩ ⟨Age: 20..24⟩ ⟨NumCars: 0..1⟩ } since its subset { ⟨Age: 20..24⟩ ⟨NumCars: 0..1⟩ } is not in L_2.

3. **Interest Prune Phase**. If the user specifies an interest level, and wants only itemsets whose support

and confidence is greater than expected, the interest measure is used to prune the candidates further. Lemma 5, given below, says that we can delete any itemset that contains a quantitative item whose (fractional) support is greater than $1/R$, where R is the interest level. If we delete all items whose support is greater than $1/R$ at the end of the first pass, the candidate generation procedure will ensure that we never generate candidates that contain an item whose support is more than $1/R$.

Lemma 5 *Consider an itemset X, with a quantitative item x. Let $\widehat{X}$ be the generalization of X where x is replaced by the item corresponding to the full range of attribute(x). Let the user-specified interest level be R. If the support of x is greater than $1/R$, then the actual support of X cannot be more than R times the expected support based on $\widehat{X}$.*

Proof: The actual support of X cannot be greater than the actual support of $\widehat{X}$. The expected support of X w.r.t. $\widehat{X}$ is $\Pr(\widehat{X}) \times \Pr(x)$, since $\Pr(\widehat{x})$ equals 1. Thus the ratio of the actual to the expected support of X is $\Pr(X)/(\Pr(\widehat{X}) \times \Pr(x)) = (\Pr(X)/\Pr(\widehat{X})) \times (1/\Pr(x))$. The first ratio is less than or equal to 1, and the second ratio is less than R. Hence the ratio of the actual to the expected support is less than R. □

5.2 Counting Support of Candidates

While making a pass, we read one record at a time and increment the support count of candidates supported by the record. Thus, given a set of candidate itemsets C and a record t, we need to find all itemsets in C that are supported by t.

We partition candidates into groups such that candidates in each group have the same attributes and the same values for their categorical attributes. We replace each such group with a single "super-candidate". Each "super-candidate" has two parts: (i) the common categorical attribute values, and (ii) a data structure representing the set of values of the quantitative attributes.

For example, consider the candidates:

{ ⟨Married: Yes⟩ ⟨Age: 20..24⟩, ⟨NumCars: 0..1⟩ }
{ ⟨Married: Yes⟩ ⟨Age: 20..29⟩, ⟨NumCars: 1..2⟩ }
{ ⟨Married: Yes⟩ ⟨Age: 24..29⟩, ⟨NumCars: 2..2⟩ }

These candidates have one categorical attribute, "Married", whose value, "Yes" is the same for all three candidates. Their quantitative attributes, "Age" and "NumCars" are also the same. Hence these candidates can be grouped together into a super-candidate. The categorical part of the super-candidate contains the item ⟨Married: Yes⟩. The quantitative part contains the following information.

Age	NumCars
20..24	0..1
20..29	1..2
24..29	2..2

We can now split the problem into two parts:

1. We first find which "super-candidates" are supported by the categorical attributes in the record. We re-use a *hash-tree* data structure described in [AS94] to reduce the number of super-candidates that need to be checked for a given record.

2. Once we know that the categorical attributes of a "super-candidate" are supported by a given record, we need to find which of the candidates in the super-candidate are supported. (Recall that while all candidates in a super-candidate have the same values for their categorical values, they have different values for their quantitative attributes.) We discuss this issue in the rest of this section.

Let a "super-candidate" have n quantitative attributes. The quantitative attributes are fixed for a given "super-candidate". Hence the set of values for the quantitative attributes correspond to a set of n-dimensional rectangles (each rectangle corresponding to a candidate in the super-candidate). The values of the corresponding quantitative attributes in a database record correspond to a n-dimensional point. Thus the problem reduces to finding which n-dimensional rectangles contain a given n-dimensional point, for a set of n-dimensional points. The classic solution to this problem is to put the rectangles in a R*-tree [BKSS90].

If the number of dimensions is small, and the range of values in each dimension is also small, there is a faster solution. Namely, we use a n-dimensional array, where the number of array cells in the j-th dimension equals the number of partitions for the attribute corresponding to the j-th dimension. We use this array to get support counts for all possible combinations of values of the quantitative attributes in the super-candidate. The amount of work done per record is only O(number-of-dimensions), since we simply index into each dimension and increment the support count for a single cell. At the end of the pass over the database, we iterate over all the cells covered by each of the rectangles and sum up the support counts.

Using a multi-dimensional array is cheaper than using an R*-tree, in terms of CPU time. However, as the number of attributes (dimensions) in a super-candidate increases, the multi-dimensional array approach will need a huge amount of memory. Thus there is a tradeoff between less memory for the R*-tree versus less CPU time for the multi-dimensional array. We use a heuristic based on the ratio of the expected memory use of the R*-tree to that of the multi-dimensional array to decide which data structure to use.

6 Experience with a real-life dataset

We assessed the effectiveness of our approach by experimenting with a real-life dataset. The data had 7 attributes: 5 quantitative and 2 categorical. The quantitative attributes were monthly-income, credit-limit, current-balance, year-to-date balance, and year-to-date interest. The categorical attributes were employee-category and marital-status. There were 500,000 records in the data.

Our experiments were performed on an IBM RS/6000 250 workstation with 128 MB of main memory running AIX 3.2.5. The data resided in the AIX file system and was stored on a local 2GB SCSI 3.5" drive, with measured sequential throughput of about 2 MB/second.

Partial Completeness Level. Figure 7 shows the number of interesting rules, and the percent of rules found to be interesting, for different interest levels as the partial completeness level increases from 1.5 to 5. The minimum support was set to 20%, minimum confidence to 25%, and maximum support to 40%. As expected, the number of interesting rules decreases as the partial completeness level increases. The percentage of rules pruned also decreases, indicating that fewer similar rules are found as the partial completeness level increases and there are fewer intervals for the quantitative attributes.

Interest Measure. Figure 8 shows the fraction of rules identified as "interesting" as the interest level was increased from 0 (equivalent to not having an interest measure) to 2. As expected, the percentage of rules identified as interesting decreases as the interest level increases.

Scaleup. The running time for the algorithm can be split into two parts:

(i) *Candidate generation.* The time for this is independent of the number of records, assuming that the distribution of values in each record is similar.

(ii) *Counting support.* The time for this is directly proportional to the number of records, again assuming that the distribution of values in each record is similar. When the number of records is large, this time will dominate the total time.

Thus we would expect the algorithm to have near-linear scaleup. This is confirmed by Figure 9, which shows the relative execution time as we increase the number of input records 10-fold from 50,000 to 500,000, for three different levels of minimum support. The times have been normalized with respect to the times for 50,000 records. The graph shows that the algorithm scales quite linearly for this dataset.

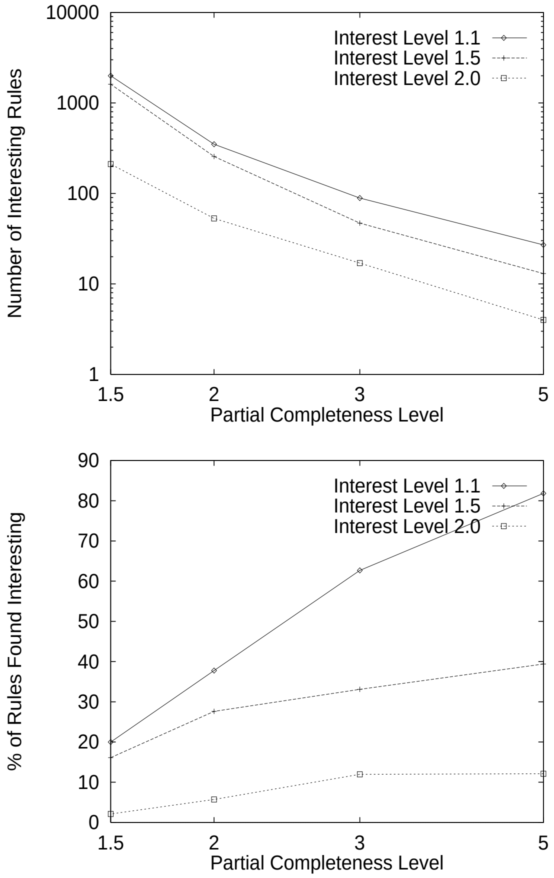

Figure 7: Changing the Partial Completeness Level

7 Conclusions

We introduced the problem of mining association rules in large relational tables containing both quantitative and categorical attributes. We dealt with quantitative attributes by fine-partitioning the values of the attribute and then combining adjacent partitions as necessary. We introduced a measure of partial completeness which quantifies the information lost due to partitioning. This measure is used to decide whether or not to partition a quantitative attribute, and the number of partitions.

A direct application of this technique may generate too many similar rules. We tackled this problem by using a "greater-than-expected-value" interest measure to identify the interesting rules in the output. This interest measure looks at both generalizations and specializations of the rule to identify the interesting rules.

We gave an algorithm for mining such quantitative association rules. Our experiments on a real-life dataset indicate that the algorithm scales linearly with the number of records. They also showed that the interest measure was effective in identifying the interesting rules.

Future Work:

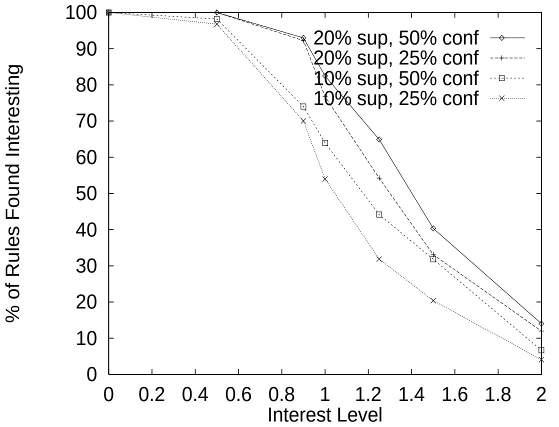

Figure 8: Interest Measure

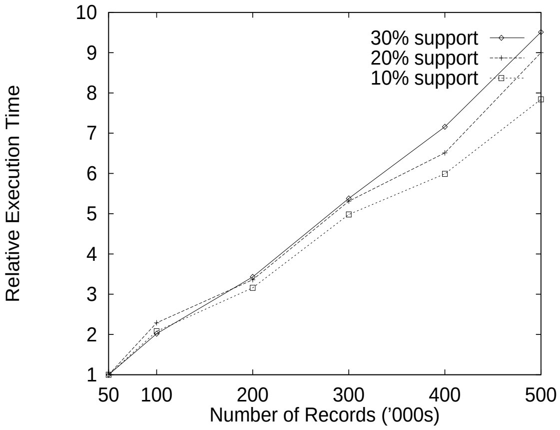

Figure 9: Scale-up : Number of records

- We presented a measure of partial completeness based on the support of the rules. Alternate measures may be useful for some applications. For instance, we may generate a partial completeness measure based on the range of the attributes in the rules. (For any rule, we will have a generalization such that the range of each attribute is at most K times the range of the corresponding attribute in the original rule.)

- 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. It may be worth exploring the use of clustering algorithms [JD88] for partitioning, and their relationship to partial completeness.

Acknowledgment We wish to thank Jeff Naughton for his comments and suggestions during the early stages of this work.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, September 1994.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R^*-tree: an efficient and robust access method for points and rectangles. In *Proc. of ACM SIGMOD*, pages 322–331, Atlantic City, NJ, May 1990.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, September 1995.

[HS95] Maurice Houtsma and Arun Swami. Set-oriented mining of association rules. In *Int'l Conference on Data Engineering*, Taipei, Taiwan, March 1995.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice Hall, 1988.

[MTV94] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, Seattle, Washington, July 1994.

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash based algorithm for mining association rules. In *Proc. of the ACM-SIGMOD Conference on Management of Data*, San Jose, California, May 1995.

[PS91] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press, Menlo Park, CA, 1991.

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. Mining Generalized Association Rules. In *Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, September 1995.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the VLDB Conference*, Zurich, Switzerland, September 1995.

[ST95] Avi Silberschatz and Alexander Tuzhilin. On Subjective Measures of Interestingness in Knowledge Discovery. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, August 1995.