Temporal Association Rules with Numerical Attributes[1]

Wei Wang, Jiong Yang, and Richard Muntz
Department of Computer Science
University of California, Los Angeles
{weiwang, jyang, muntz}@cs.ucla.edu

Abstract

Data mining has been an area of increasing interest during recent years. The association rule discovery problem in particular has been widely studied. However, there are still some unresolved problems. For example, research on mining patterns in the evolution of numerical attributes is still lacking. This is both a challenging problem and one with significant practical application in business, science, and medicine. In this paper, we present a parameterizable model for temporal sequences of numerical attributes and devise efficient ways to search for parameter values that will result in a good fit to (at least a significant portion of) the data. Metrics for how well instances of the model fit portion of the data include the familiar measures of support and strength used in association rule mining and a new metric called *density*. A user specifies thresholds for these metrics and, based on structural properties of the class of models we are attempting to fit to the data, the search space can be drastically pruned using these thresholds. Experimental results on real and synthetic data sets demonstrate the efficiency of our algorithm.

1 Introduction

Data mining, the discovery of "interesting" information from a given large data set, has become an active research area during recent years. The association rule discovery problem has been studied for several years. An association rule generally has the form $X \Longrightarrow Y$ where X and Y are two sets of items and the rule indicates that the occurrences of X and Y have high correlation. It has been proven that association rules are very useful tools for many purposes. For example, supermarket managers might use association rules among products to conduct their sale promotions to achieve greater profits. Earlier work on association rules [Agr93] [Agr94] [Han95] [Bri97b] focused on binary attributes and intra-transaction relationships. For example, an association rule $ham \Longrightarrow bread$ would be interpreted as "when a customer buys ham, he or she is likely to buy bread as well". However, the information conveyed by this rule is limited. This type of rule can not describe relationships such as

- *Supermarket.* If the price of item A increases by \$1, then the sales of item B increases by 1000.
- *Customer churning.* If a customer's phone bill increases by at least \$10 each month for six month, then he is likely to change his long distance telephone carrier.
- *Real estate.* People between 35 and 45 with salary between \$80,000 to \$120,000 are likely to buy a house whose price range is between \$300,000 and \$400,000 within two years of marriage.

In fact, mining association rules involving numerical attributes and temporal evolution has significant practical application and is also quite challenging. Recently, several efforts have been made to tackle this problem from different angles. [Sri96a]

[1] Wei Wang is the contact author. Her telephone number is (310)-206-1803 and her fax number is (310)-825-2273.

presented an approach to mine association rules involving quantitative attributes by partitioning the numerical domain into intervals, then mapping this problem to the traditional association rule mining problem. It only utilizes the support threshold to prune the search space. In contrast, our proposed algorithm utilizes all thresholds for pruning the search space, and the effects are shown in Section 5.2. [Fuk96a] [Fuk96b] [Ras98] [Ras99] focused on mining so called *optimized* association rules. One or more numerical attributes are allowed on the left hand side of the rule while the right hand side is restricted to categorical attributes. We allow categorical or numerical variables to be freely mixed[2]. Most research in time series has concentrated on pattern matching of attribute changes in a time sequence. Automatic pattern detection is a very difficult problem and research in this area has mainly focused on mining periodic patterns with predefined period [Ram98] [Han99].

In this paper, we generalize the association rule discovery problem to encompass numerical attributes without any restriction and also their temporal evolutions. Similar to traditional association rules, users are not required to provide a detailed road-map to guide the mining process even though temporal evolutions are involved. Parameters, such as support, strength (e.g., interest or confidence), and density (defined later), are necessary to qualify interesting rules. This is because the number of potential correlations among attributes and their evolutions are often huge and users usually are interested in mining out all existing strong correlations.

In this paper, we assume that the database consists of a set of objects, each of which has a unique ID and a set of time varying numerical attributes. The domain of a numerical attribute might contain a large number of distinct values (e.g., employee's salary) and might even be continuous (e.g., height of a person). Thus, the number of possible rules can be very large if not infinite. To address this problem, each attribute domain is quantized into a set of equal-length intervals and a certain amount of information loss is introduced. *Equi-depth partitioning* in [Sri96a] is an alternative way to partition the attribute domains. Equi-depth partitioning can minimize the information loss for a given fixed number of intervals. However, when the data is uniformly distributed over all attributes, the equi-depth partitioning will result in equi-length partitioning. On the other hand, when the distribution is skewed on some attribute, equi-depth tends to split adjacent values with high frequency of occurrences into separate intervals although their behaviors would typically be similar. Therefore, in the final result, these intervals would likely be merged together after all. The unnecessary partitioning and merging will increase the complexity. This is an effect which was observed in the experimental results in Section 5.1.

Another problem raised by numerical attributes is that many valid but noninteresting rules may exist. For example, the following two rules would have the same value of support and strength (confidence/interest) assuming that there is no employee younger than 25 in the database.

1. *If an employee's age is between 25 and 30, then his (her) salary is between $40,000 and $60,000.*

2. *If an employee's age is between 20 and 30, then his (her) salary is between $40,000 and $60,000.*

Certainly, the first rule conveys more precise information and is preferable to users. In order to eliminate the second rule, we use **density** as an additional metric for determining whether a rule is valid. The basic idea is that an interval should be included in a rule only if there are at least some minimum number of objects whose attribute values fall into that interval. For example, the interval $[20, 25]$ can not be included in any rule as the employee's age since no employee is younger than 25.

[2]For the categorical variable, if its domain can be totally ordered, then it can be treated in the same manner as a numerical attribute. Otherwise, categorical attribute values will not be combined unless a taxonomy (*is-a* hierarchy) is presented on that attribute. In this case, the taxonomy can be used to combine the categorical attribute values if necessary.

It is well known that the number of valid rules is often large in association rule mining applications and, without additional filtering, would be much larger in our proposed temporal association rule problem. To address this problem, we use a compact representation of the collection of all valid rules, namely, *rule sets*. Each rule set is represented by a pair of rules; one (called the min-rule) is a *specialization*[3] of the other (called the max-rule). A valid rule set is a <min-rule, max-rule> pair such that any rule which is a specialization of the max-rule and a generalization of the min-rule is also valid. For example, suppose the light shaded region in Figure 1 represents the attribute values which satisfy the support, strength (confidence/interest), and density requirement. Any subregion of the light shaded region, which is contains the dark shaded area, also satisfies the requirements (e.g., r in Figure 1 is an example of such region). There could be potentially many such subregions. Thus we use **min-rule** and **max-rule** boxes (as shown in Figure 1) to represent these subregions. The formal definition of rule sets is presented in Section 2.3. This representation of large set of rules by a <min-rule, max-rule> pair is both notationally efficient and also leads to algorithmic efficiencies by defining operations on rule sets.

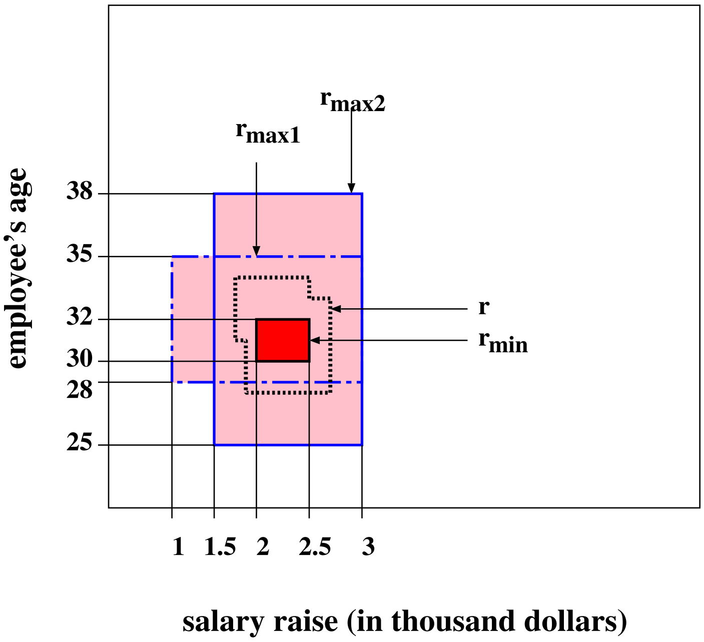


Figure 1: min rules and max rules

One method to mine temporal association rules is to map the numerical attributes to binary attributes and apply one of the existing algorithms for mining association rules [Sri96a]. For each numerical attribute A, its domain is quantized to b intervals. $O(b^2)$ items (i.e., binary attributes) are needed to represent all possible subranges. After the transformation, a traditional data mining algorithm can be used to mine the rules. Finally, the rules in terms of the binary attribute values can be mapped back to the numerical ranges. This would create a huge number of items and hence make the mining process very inefficient (shown in Section 5.2). Thus, a new, efficient algorithm for mining temporal association rules with numerical attributes is desirable. In our approach, some properties are utilized so that small neighboring intervals can be combined into larger intervals efficiently (if necessary) during the mining process.

The remainder of the paper is organized as follows. The general model is presented in Section 2. Section 3 describes our approach to mine temporal association rules with numerical attributes. We show how to extend the initial results to

[3] The formal definition of specialization is in Section 2.2.

temporal associations of higher order (e.g., involving rate of change of attribute values) in Section 4. Performance analysis and theoretical foundation of our approach are discussed in Section 5 and 6, respectively. Section 7 presents some related work. Finally, conclusions are drawn in Section 8.

2 General Model

In this paper, we deal with objects. Each object has a unique ID and a set of numerical attributes. The database is viewed as a sequence of snapshots $S_1, S_2, \ldots, S_t$ of objects and their attribute values taken at some frequency. A **window** of **width** m is a (sub)sequence of m consecutive snapshots $S_j, S_{j+1}, \ldots, S_{j+m-1}$ where $1 \leq j \leq t-m+1$ and S_j is called the *starting snapshot*. This window is denoted by $W(j,m)$. For example, S_3, S_4, S_5, S_6 and S_5, S_6, S_7, S_8, S_9 are two windows of width 4 and 5, and are denoted by $W(3,4)$ and $W(5,5)$, respectively. In general, given a sequence of t snapshots $S_1, \ldots, S_t$, there are a total of $t-m+1$ different windows of width m where $1 \leq m \leq t$. Given an object o and a window, $W(j,m) = S_j, S_{j+1}, \ldots, S_{j+m-1}$, the temporal changes of o over $W(j,m)$ is called an **object history** of o within window $W(j,m)$. The *length* of this object history is m. Looking ahead, an object history can be mapped to a point in a high dimensional space. A temporal association rule can be viewed as a hypercube in such high dimensional space, which contains the set of object histories satisfying the rule. The thresholds for support, strength, and density correspond to constraints on the number and distribution of object histories within the hypercube. This will be discussed in more detail shortly.

As mentioned before, we employ an *equi-length* partition to divide each attribute domain into a sequence of disjoint equal-length ranges, referred to as **base intervals**. Same number of base intervals are used for every attribute domain. (This can be easily generalized to different numbers of base intervals on different attribute domains.) Let b be the number of base intervals for each attribute domain. The values within each base interval are regarded non-distinguishable.

2.1 Evolution

In traditional association rule discovery, the building blocks are items, whereas the building blocks in our proposed temporal association rules are attribute **evolutions**. Intuitively, an evolution describes the temporal changes of attribute values of some object histories. For example, "employee's salary changes from an interval of $40,000 and $45,000 to an interval of $47,500 and $55,000, then to an interval of $60,000 and $70,000" is an evolution. As in this example, the attribute value is represented by intervals. Each interval in an evolution is formed by merging a sequence of adjacent base intervals.

Similar to the object history, an evolution also has its **length**, which is the number of snapshots involved. For example, the above evolution of employees' salaries involves three snapshots, therefore, its length is 3. Now we give the formal definition of evolutions.

Definition 2.1 *Given an attribute A_i, an* **evolution** $E(A_i)$ *of length m describes the change of values of A_i over m consecutive snapshots, which is represented as $(A_i \in [l_1, u_1]) \rightarrow (A_i \in [l_2, u_2]) \rightarrow \ldots \rightarrow (A_i \in [l_m, u_m])$ where $[l_j, u_j]$ is the interval of values of A_i at the jth snapshot, and is the union of a sequence of adjacent base intervals.*

For example, the evolution "the employee's salary changes from an interval of $40,000 and $45,000 to an interval of $47,500 and $55,000, then to an interval of $60,000 and $70,000" can be represented as $(salary \in [40000, 45000]) \rightarrow (salary \in [47500, 55000]) \rightarrow (salary \in [60000, 70000])$. Given an attribute A_i whose domain is D_i, its *evolution space* for evolutions of length m is an m dimensional space, where the jth dimension corresponds to the value of A_i at the jth snapshot relative to the start of the evolution. Since domain D_i is partitioned into b base intervals, the evolution space consists of b^m basic hypercubes, referred to as **base cubes**. Any evolution of length m uniquely corresponds to an m dimensional hypercube (referred to as *evolution cube*) in the evolution space, and vice versa. In fact, an evolution cube is the union of a set of adjacent base cubes. For example, Figure 2 shows the evolution space for evolutions on $salary$ of length 3. The evolution $E_1 = (salary \in [40000, 45000]) \rightarrow (salary \in [47500, 55000]) \rightarrow (salary \in [60000, 70000])$ corresponds to the shaded cube, which consists of 24 base cubes. (The span of a base cube in each dimension in this example is 2,500.)

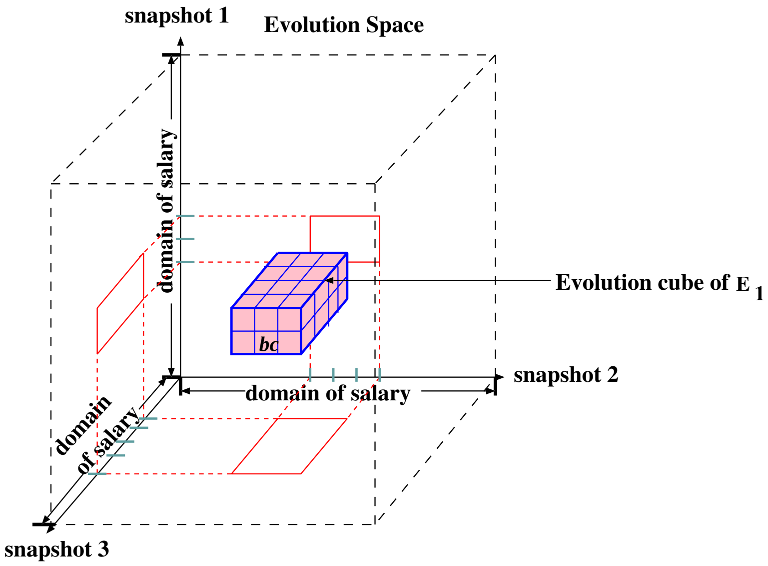


Figure 2: Evolution Space for $salary$

For a given object history of length m, the values of A_i define a point in such an evolution space. Given a sequence of m snapshots and an evolution on attribute A_i of length m, we say that an object history (of length m) within a window $W(j, m) = S_j, S_{j+1}, \ldots, S_{j+m-1}$ *follows* this evolution iff, for each snapshot in the window, the value of the attribute in the object history falls into the corresponding interval specified in $E(A_i)$. In other words, if the point representing the object history in the evolution space is enclosed by the box representing the given evolution, then we say that the object history follows this evolution. For example, if the salary of an object history of "Joe Smith" changes from \$44000, to \$50000, then to \$62000, then we say that this object history of "Joe Smith" *follows* evolution E_1. However, such an object history does not follow the evolution $(salary \in [40000, 50000]) \rightarrow (salary \in [55000, 57500]) \rightarrow (salary \in [60000, 67500])$ since \$50000 does not belong to the range [55000, 57500].

For any two evolutions

$$E(A_i) = (A_i \in [l_1, u_1]) \rightarrow (A_i \in [l_2, u_2]) \rightarrow \ldots \rightarrow (A_i \in [l_m, u_m])$$

and

$$E'(A_i) = (A_i \in [l'_1, u'_1]) \rightarrow (A_i \in [l'_2, u'_2]) \rightarrow \ldots \rightarrow (A_i \in [l'_m, u'_m])$$

of the same length m, $E(A_i)$ is a **specialization** of $E'(A_i)$ iff, for every $j (1 \leq j \leq m)$, $[l_j, u_j]$ is enclosed by $[l'_j, u'_j]$. In other words, the evolution cube corresponding to $E(A_i)$ (in the evolution space) is entirely subsumed by the evolution cube corresponding to $E'(A_i)$. In such a case, $E'(A_i)$ is a **generalization** of $E(A_i)$. For example, E_1 is a specialization of

$(salary \in [40000, 55000]) \rightarrow (salary \in [40000, 60000]) \rightarrow (salary \in [60000, 70000])$. However, it is not a specialization of $(salary \in [40000, 55000]) \rightarrow (salary \in [40000, 50000]) \rightarrow (salary \in [60000, 65000])$. Note that an evolution is always a specialization and a generalization of itself. This specialization/generalization relationship forms a lattice among evolutions.

In general, if n attributes $A_1, A_2, \ldots, A_n$ are considered, then the evolution space for the conjunction of evolutions $E(A_1), E(A_2), \ldots, E(A_n)$ of length m is an $n \times m$ space where each dimension represents the values of one attribute at one snapshot. For any two evolution conjunctions

$$EC = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$$

and

$$EC' = E'(A_1) \cap E'(A_2) \cap \ldots \cap E'(A_n)$$

EC is a **specialization** of EC' iff, for every attribute A_i $(1 \leq j \leq n)$, $E(A_i)$ is a specialization of $E'(A_i)$. In this case, EC' is also called a **generalization** of EC. An object history is said to *follow* an evolution conjunction $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$ iff this object history follows every evolution $E(A_i)$ $(1 \leq i \leq n)$.

2.2 Temporal Association Rules

Our proposed temporal association rule has the form of $X \Longrightarrow Y$ where X and Y are conjunctions of attribute evolutions. In this paper, we assume that Y only contains the evolution of one attribute for simplicity and clarity of exposition. All results, with minor modifications, can be applied to the case where evolution conjunctions are allowed for Y as well as X.

Definition 2.2 *Let $A_1, A_2, \ldots, A_n$ be a subset of attributes. A* **temporal association rule** *of length m is an implication $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ and $E(A_i)$ is an evolution (of length m) of attribute A_i $(1 \leq i \leq n)$.*

For any two temporal association rules

$$R = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$$

and

$$R' = E'(A_1) \cap E'(A_2) \cap \ldots \cap E'(A_{k-1}) \cap E'(A_{k+1}) \cap \ldots \cap E'(A_n) \Longrightarrow E'(A_k)$$

of the same length m and R is a *specialization* of R' iff, for every attribute A_i, $E(A_i)$ is a specialization of $E'(A_i)$. R' is also called a *generalization* of R. This specialization/generalization relationship also defines a lattice among temporal association rules.

In general, for a given set of attributes, many temporal association rules exist simultaneously. We need to identify which of these rules are of interest. This is addressed in the following definitions which introduce metrics useful in quantifying how potentially interesting a given rule is.

For any rule, its **support**, **density**, and **strength** are indications of frequency of occurrence, concentration of population and the degree of non-independence represented by this rule, respectively. The reasons for specifying support and strength thresholds are similar to the reasons for specifying support and confidence thresholds in traditional association rule discovery. However, due to the nature of numerical data, users need to specify one more parameter, *density*. Intuitively, this is motivated by the notion that there should be enough samples everywhere within the intervals specified in a rule to show that the rule is valid over the entire range. Thus, the density parameter ϵ is introduced for this purpose.

The length of a rule is usually smaller than the number of snapshots available. For a given evolution $E(A_i)$, object histories of different objects might follow $E(A_i)$ during different windows. For example, $E(A_i)$ may be followed by an object history of object o_1 at window $W(j, m)$ and an object history of object o_2 at window $W(k, m)$ where $j \neq k$. Therefore, we define support, strength, and density based on object histories. For an evolution of length m, a sliding window of width m is used to examine object histories of length m. For example, in order to evaluate evolutions of length 3, all object histories of length 3 are considered, as illustrated in Figure 3. All object histories within each window (of length 3) are examined to aggregate the overall support, strength, and density. We now discuss each of these three measures in detail.

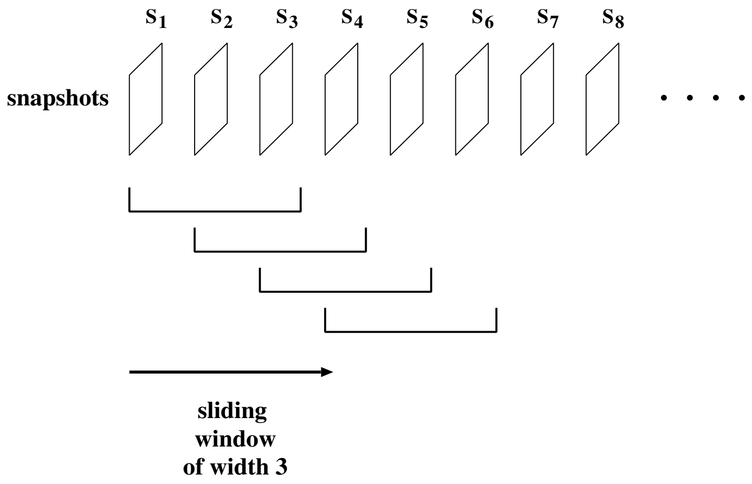


Figure 3: Sliding Window

2.2.1 Support

The *support* of an evolution conjunction $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$ of length m on a sequence of snapshots $S_1, S_2, \ldots, S_t$ is the number of object histories of length m which follow this evolution conjunction. For a given window $W(j, m)$, the support within this window is the number of object histories which follow the evolution conjunction. The overall support of this evolution conjunction is the sum of the supports within each window. Therefore, one object can contribute multiple times to the support if multiple object histories of this object follow the evolution conjunction. The support of a temporal association rule $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ over snapshots $S_1, S_2, \ldots, S_t$ is the support of $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{n-1}) \cap E(A_n)$ on the same sequence of snapshots.

Definition 2.3 *Given an evolution conjunction $\Pi = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$ of length m and a sequence of snapshots $\Omega : S_1, S_2, \ldots, S_t$, the* **support** *of the evolution conjunction, denoted by $Support(\Pi, \Omega)$, is*

$$\sum_{j=1}^{t-m+1} N(\Pi, W(j, m))$$

where $N(\Pi, W(j, m))$ is the number of object histories which follow Π on window $W(j, m)$.

2.2.2 Strength

Different methods can be used to capture the degree of non-independence. In this paper, we use a metric which is similar to *interest* defined in [Bri97b] to measure the strength of a temporal association rule.

Definition 2.4 *Given a temporal association rule* $R : E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{K+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ *of length* m *and a sequence of snapshots* $\Omega : S_1, S_2, \ldots, S_t$*, the* **strength** *of the rule is*

$$\frac{Support\left(E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{K+1}) \cap \ldots \cap E(A_n) \cap E(A_k), \Omega\right)}{Support\left(E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{K+1}) \cap \ldots \cap E(A_n), \Omega\right) \times Support\left(E(A_k), \Omega\right)}.$$

2.2.3 Density

Since we are concerned with numerical attributes, there can be unoccupied regions in the evolution space, especially when the attribute domain is continuous. In other words, there could exist two rules R_1 and R_2 where R_1 is a specialization of R_2 and $R_1 \neq R_2$, such that the support and the strength of R_1 and R_2 are the same. However, it is obvious that R_1 conveys more precise information than R_2, and is preferable to users. In order to eliminate rules such as R_2, we require that the object history concentration within each base cube of an evolution cube is above some threshold. If the concentration is too low, then this rule will be invalid because there is not enough evidence to show that the rule holds for this region. Thus, we use one additional parameter, called *density*, to capture the object history concentration of the most sparse base cube within an evolution cube.

Different base cubes will, in general, contain different number of object histories. Moreover, rules involving different attributes have different evolution spaces. In one snapshot, the average number of attribute values fall into each base interval is $\frac{N}{b}$. $\frac{N}{b}$ can be viewed as the "average density". The desirable density is specified as a ratio (denoted by ϵ)[4] of the number of object histories, which will make a base cube "dense". For example, let's assume that the domain of salary in Figure 2 is $[30000, 80000]$, there are a total of 10,000 employees, and b is chosen to be 20. In addition, bc is a base cube within E_1, which corresponds to the evolution $(salary \in [40000, 42500]) \to (salary \in [50000, 52500]) \to (salary \in [67500, 70000])$. If ϵ is specified as 2, then bc is regarded as dense when the number of object histories which follow bc is at least $\frac{10000}{20} \times \epsilon = 500 \times 2 = 1000$. The density of E_1 is the minimum of the normalized densities of its base cubes.

Definition 2.5 *Given an evolution conjunction* $\Pi = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$ *of length* m *and a sequence of snapshots* $\Omega : S_1, S_2, \ldots, S_t$*, the* **density** *of the evolution conjunction is* $\min_{\forall bc \in \Pi} Support(bc, \Omega)$ *where* bc *is a base cube which is a specialization of* Π.

The density of a rule $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{K+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ is the density of the evolution conjunction $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$. A user is required to specify thresholds for the minimum support, strength, and density. A temporal association rule is considered **valid** if it satisfies these three thresholds.

2.3 Rule Sets

Definition 2.6 *Given two rules* r_{min} *and* r_{max} *where* r_{min} *is a specialization of* r_{max}*, a* **rule set** $< r_{min}, r_{max} >$ *represents the set of all rules* r *such that* r *is a specialization of* r_{max} *and a generalization of* r_{min}*.* r_{min} *and* r_{max} *are called* **min-rule** *and* **max-rule** *of the rule set, respectively.*

[4] ϵ can be any positive real number. Usually ϵ is greater than 1.

Looking ahead, since the number of valid rules can be very large, we use a small number of rule sets to summarize all valid rules.

3 Our Approach

Given a sequence of snapshots, our objective is to find the set of temporal association rules that meet the user specified thresholds for support, density, and strength. Two phases involved are:

1. After partitioning each attribute domain into b base intervals, find all base cubes which satisfy the density threshold.
2. Based on dense base cubes, find all valid rule sets.

3.1 Dense Base Cube Discovery

The Apriori Property [Agr94] also holds on the density metric defined in previous section. The following properties are useful in pruning the evolution space.

Property 3.1 *Let* $E = (A_i \in [l_1, u_1]) \to (A_i \in [l_2, u_2]) \to \ldots \to (A_i \in [l_m, u_m])$ *and* $E' = (A_i \in [l_j, u_j]) \to (A_i \in [l_{j+1}, u_{j+1}]) \to \ldots \to (A_i \in [l_k, u_k])$ *be two evolutions where* $1 \leq j \leq k \leq m$*. For a given sequence of snapshots* Ω*, the density of* E *over* Ω *is less than or equal to that of* E'*.*

In above property, E' contains a subsequence of intervals specified in E. E' can be viewed as a *projection* of E onto an evolution space of lower dimensionality.

Property 3.2 *Given a sequence of snapshots* Ω*, the density of an evolution conjunction* $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$ *is less than or equal to the density of the conjunction of any subset of* $\{E(A_1), E(A_2), \ldots, E(A_n)\}$*.*

Similar to Property 3.1, Property 3.2 states that the density of an evolution conjunction is less than or equal to the density of any of its projections (onto an evolution space of lower dimensionality). Based on these two properties, a level-wise (dynamic programming) [Man97a] algorithm can be used to find all dense base cubes. In Figure 4, $BaseCube(i, m)$ denotes the set of base cubes of evolution conjunctions involving i distinct attributes and having length m. If the density of a base cube bc in $BaseCube(i, m)$ satisfies the density threshold, then, by Property 3.1, both of bc's projections in $BaseCube(i, m-1)$ have to satisfy the threshold as well. In addition, by Property 3.2, all bc's projections in $BaseCube(i-1, m)$ must also satisfy the density threshold.

Therefore, beginning from the set of all base intervals[5], at each successive level j, we compute the set of dense base cubes in $BaseCube(i, j-i+1)$ for all $1 \leq i \leq j$ until such a level that the set of dense base cubes at the previous level is empty. The formal description of the algorithm is in Appendix A.

In fact, any algorithm proposed for efficiently computing large itemsets in the traditional association rule problem can be adopted in this phase. We will not elaborate on them in this paper due to space limitations. After we find all dense base cubes, a set of dense regions can be formed by linking adjacent base cubes. Since every valid rule has to satisfy the density requirement, we only need to explore these dense regions in the second phase. This is described in the next subsection.

[5] A base interval can be viewed as a base cube in the evolution space for evolutions involving one attribute and of length 1.

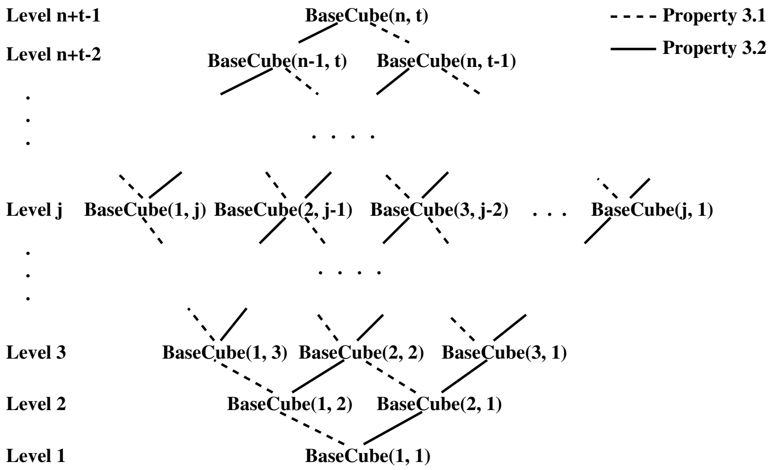


Figure 4: Base Cube Lattice

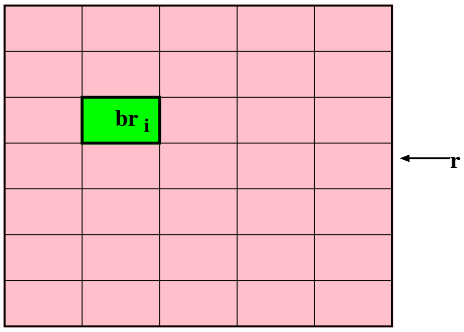
3.2 Temporal Association Rule Discovery

Let $SUPPORT$, $STRENGTH$, and ϵ be the user specified thresholds for support, strength, and density, respectively. The properties of strength will be shown to enable us to find all temporal association rules efficiently. Let $A_1, A_2, \ldots, A_n$ be a subset of attributes. The space of evolution conjunctions $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_n)$ of length m is an $m \times n$ dimensional space. Our objective is to find all valid rule sets $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ for each possible choice for the right hand side attribute A_k. It is obvious that, given such an attribute A_k serving as the right hand side, only one rule corresponds to each evolution cube, whose strength can be calculated uniquely. In the following discussion, without loss of generality, we assume that A_k is on the right hand side of the rule, and $rule(c)$ and $cube(r)$ are used to denote the corresponding rule of evolution cube c and the corresponding evolution cube of rule r, respectively. As we mentioned before, due to the density requirement, we only need to examine rules whose corresponding evaluation cubes are enclosed entirely by some dense region. The finest granularity rule we consider is the rule which corresponds to only one dense base cube. We regard such rules as base rules. The following two properties (which are proven in Section 6) are used to guide the rule discovery process in each dense region. The final result is the union of all valid rule sets discovered in each dense region.

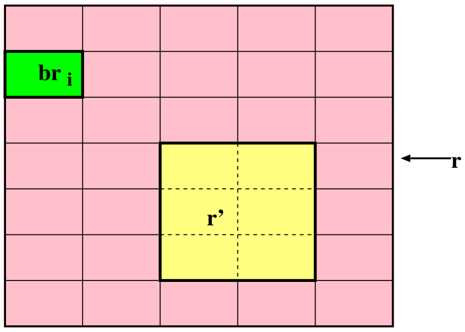
Property 3.3 *For any rule r, there exists a base rule br_i which is a specialization of r and whose strength is at least that of r.*

This property is illustrated in Figure 5(a). It implies that any valid rule has to be a generalization of some base rule whose strength is at least $STRENGTH$. In a given dense region, let $BR = \{br_1, br_2, \ldots, br_g\}$ be the set of base rules whose strengths are at least $STRENGTH$. Then, for this dense region, we only need to examine rules which are generalizations of some base rule(s) in BR. Let R be the set of rules, each of which is a generalization of some base rule(s) in BR and whose corresponding evolution cube is enclosed by the given dense region. R contains all potential valid rules which can be discovered from this dense region. The following property can be utilized to prune R further.

Property 3.4 *For any two rules r and r' where r' is a specialization of r, and the strength of r' is less than that of r, there exists another base rule br_i which is a specialization of r but not r', and whose strength is greater than that of r.*



(a) Property 3.3



(b) Property 3.4

Figure 5: Properties for Strength

As illustrated in Figure 5(b), this property implies that if the strength of a rule r' is below the threshold $STRENGTH$, then the strength of any generalization of r', denoted by r, is below $STRENGTH$ unless r contains some other base rule br_i whose strength is greater than $STRENGTH$. This suggests that we can prune R further by skipping those rules which are generalizations of r' and do not contain any other base rule in BR. Since each rule in R is a generalization of some base rule(s) in BR, for any two rules $r \in R$ and $br_i \in BR$, we say that r *contains* br_i iff the evolution cube of br_i is enclosed by that of r.

Rules in R can be divided into groups such that rules in the same group contain the same subset of base rules in BR. If there are g base rules in BR, then there are $2^g - 1$ distinct subsets of BR, and hence R consists of $2^g - 1$ groups, one for each subset. Moreover, the set of evolution cubes corresponding to rules in the same group occupy a continuous region in the search space. For example, $BR = \{b_1, b_2, \ldots, b_9\}$ are the set of base rules whose strengths are above the threshold. $BR' = \{b_3, b_4\}$ is a subset of BR. The light shaded area between those two solid contours in Figure 6 corresponds to the region for the set of evolution cubes, denoted by SEC_{34}, which are generalizations of b_3 and b_4 but not others in BR. Any evolution cube which contains the inner contour and is enclosed by the outer contour belongs to SEC_{34}. All rules whose corresponding evolution cubes are in SEC_{34} belong to the same group.

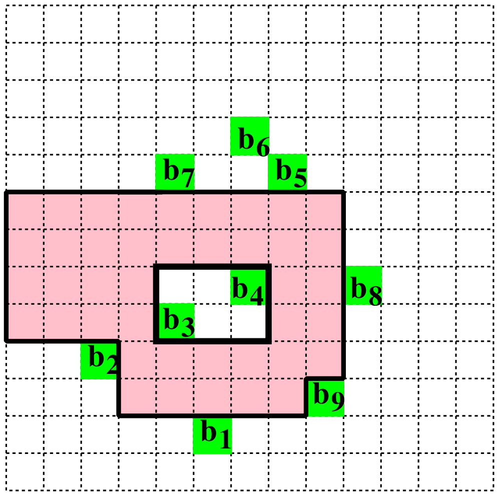


Figure 6: Rule Discovery

Therefore, each dense region can be viewed as the union of a set of regions, each of which contains the set of evolution cubes corresponding to rules belonging to the same group. In our algorithm, each such region is examined in turn so that the space can be pruned efficiently.

For each subset of base rules in BR, denoted by BR', we begin to explore the corresponding search region (e.g., the light shaded region in Figure 6) from the rule r which corresponds to the minimum bounding box of rules in BR' (e.g., the inner contour in Figure 6). If the strength of this rule is less than the threshold $STRENGTH$, then, by Property 3.4, no rule in this region can satisfy the required strength. Thus, we do not need to examine any other rule in this region. Otherwise (i.e., the strength of r is at least $STRENGTH$), if r's support satisfies the requirement, r is valid and is taken as the min-rule of some valid rule set. If the support of r is below the threshold $SUPPORT$, we search for its valid generalizations within the region, which can potentially serve as the min-rule of some valid rule set. Breadth first search is employed during this process. The span of one dimension of the evolution cube is expanded in one direction by one base interval at each step until the support is at least $SUPPORT$, while the strength is still at least $STRENGTH$. Then this rule, referred to as r_{min} is taken as the min-rule for the valid rule set. However, during the expansion process, if the strength of the candidate rule falls below the requirement before the required support is met (by Property 3.4), or the support is still less than $SUPPORT$ when another base rule in BR is encountered (i.e., the boarder of this search region is reached), then there is no valid rule in this region.

In order to find the max-rules, searching continues in the same manner until a rule r_{max} is found such that all of r_{max}'s generalizations either violate the strength requirement or another base rule in BR becomes included, then the rule r_{max} is taken as the max-rule of the valid rule set. Note that multiple max-rules might exist for the same min-rule (as in Figure 1). For each such max-rule, a valid rule set is constructed as $< r_{min}, r_{max} >$. Any rule which is a generalization of r_{min} and a specialization of r_{max} is guaranteed to be valid. This process repeats on every subset of base rules in BR. In Section 6, the correctness of our proposed algorithm will be proved. The formal description of the rule sets discovery algorithm is in Appendix B.

4 Rules of Higher Order

In previous sections, the attribute evolution in the temporal association rules is defined as a sequence of intervals, each of which corresponds to the attribute value at each snapshot. However, this can not describe all interesting relationships that may exist. For example, "if an employee received raise for at least $5,000 each year for 3 consecutive years, he is likely to move to a more expensive house" could be an interesting rule in which the amount of change to an attribute between consecutive snapshots plays an essential role. In order to accommodate this type of rule, we extend the definition of evolution. An evolution on attribute A_i of length m is represented as $I_1(A_i) \rightarrow I_2(A_i) \rightarrow \ldots \rightarrow I_m(A_i)$ where each $I_j(A_i)$ has the form of either $(A_i \in [l_j, u_j])$ or $(\delta A_i \in [l_j, u_j])$.

The algorithms presented in the previous section apply directly except for one change. The only modification is that for each attribute we take the difference between each pair of consecutive snapshots, and quantize the difference into b intervals as well. Then there are $2b$ intervals for each attribute: the first b intervals correspond to the attribute values at a snapshot and the last b intervals correspond to the differences of attribute value from previous snapshot[6]. When small adjacent base

[6]The relative differences (i.e., percentage-wise differences) can also be handled in a similar manner as the absolute differences described here. An example of interesting rules involving relative difference would be "A raise of greater than 20% of the salary is very likely accompanied by a change in title." Interested readers please refer to [Wan99] for details.

intervals are merged together, only base intervals of the same type are allowed to be merged.

5 Experimental Results

We have implemented a prototype of our rule discovery algorithm in C. The experiments were carried on an Ultra Sparc10 workstation with 128 MB main memory, one 300MHZ CPU, and running Solaris 2.5. After quantizing the attribute domains, our rule detection algorithm consists of two phases, (1) dense region discovery, and (2) valid rule discovery. We apply our algorithm on different data sets, synthetic and real.

5.1 Equi-length Partition versus Equi-depth Partition

An equi-length partition divides an attribute domain into a set of equal length intervals while an equi-depth partition divides an attribute domain into a set of intervals, each of which contains a similar number of objects. [Sri96a] proves that the equi-depth partition can minimize the information loss with the same of number of intervals. However, if attribute values have a skewed distribution over the domain, then in high density regions, objects with attribute values in adjacent intervals may have similar behaviors. In our application, these small intervals have to be merged together, this may increase the complexity of the algorithm.

In this section, we analyze the pros and cons of the equi-length and equi-depth partition. In [Sri96a], the *partial completeness level* K is used to measure the information loss, which is defined in the following manner. Consider two sets of rules: one set is obtained by considering all ranges over the raw values, denoted by Q, and the other is obtained by considering only ranges over partitions, denoted by Q'. For each rule q_i in Q, find the closest generalization q_i' in Q'. The *distance* between q_i and q_i' is measured as the ratio of supports of q_i' and q_i. The further away the closest generalization is, the greater the loss is. The value of K is the maximum of all such distances and is used to quantify the degree of information loss. To empirically analyze these two partitions, we synthetically generated a data set which contained 50,000 objects, with 7 attributes associated with each object, and 15 snapshots overall. The support, strength, and density thresholds were set to 2%, 1.2, and 2, respectively. Figure 7(a) shows the loss of information of the equi-length and equi-depth partition as a function of number of base intervals, whereas Figure 7(b) shows the overall computation time.

It is clear that less information is lost and more response time is required with the increase of the number of base intervals on each attribute domain. With a small number of intervals (e.g., less than 100), the equi-depth has a significant advantage on minimizing information loss while the penalty on response time is small. With a larger number of intervals (e.g., larger than 100), the equi-depth partition has much longer response time because a large number of small intervals have to be emerged at the end, while the advantage with respect to information loss is insignificant. Therefore, if high result quality is required, the equi-length partition with slightly more intervals on each attribute domain is preferable. For example, if K is required to be less than 1.5, then the equi-length partition with 120 intervals would be more cost effective than the equi-depth partition with 100 intervals because both partitions result in similar K value (1.4), but the equi-length partition has a significantly lower computation cost (about 20%). We therefore adopted the equi-length partition to quantize the attribute domains in the remaining experiments.

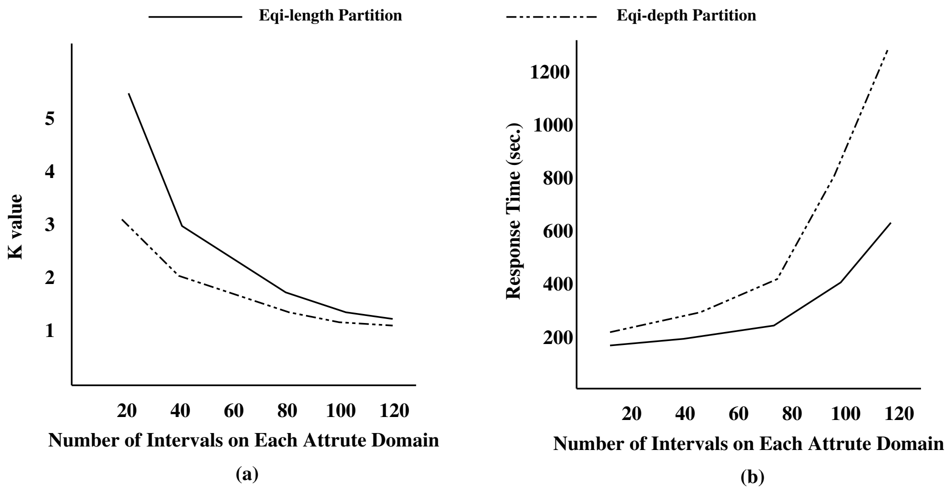


Figure 7: Equi-length Partition versus Equi-depth Partition

5.2 Two Mining Algorithms

In this section, we compare our algorithm with an alternative algorithm that is similar to that proposed by Srikant and Agrawal [Sri96a]. In the alternative algorithm, after each attribute domain is quantized to a sequence of base intervals, these base intervals are combined as long as their density[7] satisfies the threshold. The original intervals and the combined intervals (whose support satisfies the user specified threshold), are treated as a set of items. The algorithms for mining traditional association rules can be applied on these items. For the rest of this section, we refer this alternative algorithm to as *SR* algorithm.

Three synthetic data sets were generated, each of which consists of 100,000 objects and 100 snapshots. Each object has 5 attributes. We embedded 500 rules of length 5 or less in each data set[8]. Figure 8 describes the variation of the average overall response time versus the number of base intervals for each attribute domain. The number indicated on the curve represents the recall, which is the percentage of embedded rules that are reported by both algorithms. (The precision of the algorithms is 100%, i.e., all reported rules are valid.) In this experiment, the density, support, and strength were chosen as 2, 5%, and 1.4, respectively.

It is obvious that our algorithm has a much faster response time than the SR algorithm because the strength threshold is merely used to verify whether a rule is valid in the SR algorithm, whereas the strength is used to prune the search space in our algorithm. In turn, the set of candidate rules searched by the SR algorithm is much large than that in our algorithm, the response time, hence, is impacted. Moreover, response time increases much more quickly with an increase in the number of base intervals while recall improves slowly. However, when we choose the number of base intervals to be 100 in our algorithm, the response time is still within an acceptable length, but 90% recall can be achieved.

Figure 9 shows the ratio of the number of rules examined by our algorithm versus the number of rules examined by the SR algorithm as a function of strength. The same data set as the previous test was used. The support and density thresholds were 5% and 2, respectively. It is clear that with larger strength, a larger of number of possible rules can be pruned in our

[7] [Sri96a] did not have the notion density, it used max-support to prune the search space. In order to fairly compare these two algorithms, the density was used to prune the search space in the alternative algorithm.

[8] For each embedded rule, we calculate the number of object histories which is necessary to make the rule valid and generate object histories accordingly.

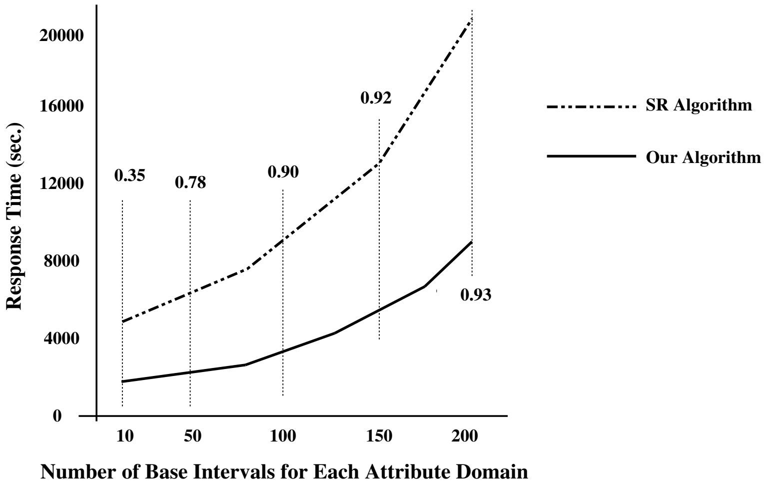


Figure 8: Response Time vs. Recall

algorithm, while the SR algorithm still examines these rules.

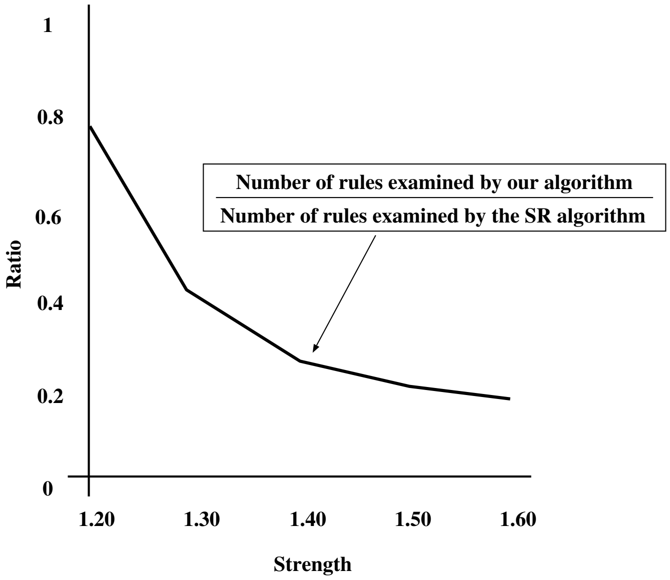


Figure 9: Number of Rules Examined

5.3 Real Data Set

We also ran our mining algorithm on a real data set. Each object represents a person. The attributes are the age, the title of that person, the salary of that person, family status (single, married, head of household), and the distance between the person's house and a major city. The distance is computed based on the zip codes. We selected all attributes as interesting. There are 20,000 objects and 10 snapshots. The snapshot was taken once a year, from 1986 to 1995. We set the number of base intervals to be 100 and the thresholds for support, density, and strength to be 3% (i.e., 600 objects), 2, and 1.3, respectively.

It took approximately 260 seconds to mine the rules. A total of 347 rule sets were discovered. Due to space limitation, we only can present a few of mined rules here. One rule discovered is that "People receiving a raise tends to move further away from the big city." This rule may be explained as the price of a house in the suburbs is more expensive and the area is more desirable to high income family. The other is that "People with a salary in the range between $70,000 to $100,000 get a raise, the range of the raise will likely be from $7,000 to $15,000."

6 Theoretical Foundation

In this section, we prove some important properties held by the temporal association rules. We mainly focus on the strength metric and the correctness of our rule set discovery algorithm due to space limitations. Assume that each attribute domain is partitioned into a set of disjoint intervals and only ranges formed by the union of consecutive partitions are considered.

Lemma 6.1 *Given two lists of positive real numbers* $x_1, x_2, \ldots, x_n$ *and* $y_1, y_2, \ldots, y_n$*, let* $x = \sum_{1 \leq j \leq n} x_j$ *and* $y = \sum_{1 \leq j \leq n} y_j$*. If there exists an integer* i *(*$1 \leq i \leq n$*) such that* $\frac{x_i}{y_i} < \frac{x}{y}$*, then there must exist another integer* k *(*$1 \leq k \leq n$*) such that* $\frac{x_k}{y_k} > \frac{x}{y}$*.*

Proof. The proof is by contradiction. Assume that $\frac{x_j}{y_j} \leq \frac{x}{y}$ for all $1 \leq j \leq n$. Then we have $xy_j - yx_j \geq 0$ for all $1 \leq j \leq n$. Since $\frac{x_i}{y_i} < \frac{x}{y}$, we have $xy_i - yx_i > 0$. Therefore, $\sum_{1 \leq j \leq n}(xy_j - yx_j) > 0$. On the other hand,

$$\frac{x}{y} = \frac{\sum_{1 \leq j \leq n} x_j}{\sum_{1 \leq j \leq n} y_j}$$

$$x \times \sum_{1 \leq j \leq n} y_j = y \times \sum_{1 \leq j \leq n} x_j$$

Then, $\sum_{1 \leq j \leq n}(xy_j - yx_j) = 0$, which contradicts $\sum_{1 \leq j \leq n}(xy_j - yx_j) > 0$. Thus, this lemma holds. □

The following corollary follows directly from Lemma 6.1.

Corollary 6.2 *Given two lists of positive real numbers* $x_1, x_2, \ldots, x_n$ *and* $y_1, y_2, \ldots, y_n$*, let* $x = \sum_{1 \leq j \leq n} x_j$ *and* $y = \sum_{1 \leq j \leq n} y_j$*. There must exist an integer* k *(*$1 \leq k \leq n$*) such that* $\frac{x_k}{y_k} \geq \frac{x}{y}$*.*

Lemma 6.3 *For any rule* $r = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$*, let* $br_1, br_2, \ldots, br_h$ *be the set of base rules which are specializations of* r*, where* $br_j = e_j(A_1) \cap e_j(A_2) \cap \ldots \cap e_j(A_{k-1}) \cap e_j(A_{k+1}) \cap \ldots \cap e_j(A_n) \Longrightarrow e_j(A_k)$*. Then the strength of* r *over a sequence snapshots* $\Omega = S_1, S_2, \ldots, S_t$ *is*

$$Strength(r, \Omega) = \frac{\sum_{1 \leq j \leq h} Support(v_j \cap e_j(A_k), \Omega)}{\sum_{1 \leq j \leq h} \left(Support(v_j, \Omega) \times Support(e_j(A_k), \Omega)\right)}$$

where $v_j = e_j(A_1) \cap e_j(A_2) \cap \ldots \cap e_j(A_{k-1}) \cap e_j(A_{k+1}) \cap \ldots \cap e_j(A_n)$*.*

Proof. Let h_l and h_r be the number of base cubes enclosed by the evolution cubes of $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n)$ and $E(A_k)$ respectively. According to the definition, the strength of br_j is

$$Strength(br_j, \Omega) = \frac{Support(v_j \cap e_j(A_k), \Omega)}{Support(v_j, \Omega) \times Support(e_j(A_k), \Omega)}$$

and the strength of r is

$$Strength(r, \Omega) = \frac{Support(V \cap E(A_k), \Omega)}{Support(V, \Omega) \times Support(E(A_k), \Omega)}$$

where

$$V = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n)$$

$$Support(V \cap E(A_k), \Omega) = \sum_{1 \le j \le h} Support(v_j \cap e_j(A_k), \Omega),$$

$$Support(V, \Omega) = \frac{1}{h_r} \sum_{1 \le j \le h} Support(v_j, \Omega),$$

$$Support(E(A_k), \Omega) = \frac{1}{h_l} \sum_{1 \le j \le h} Support(e_j(A_k), \Omega).$$

Thus,

$$Strength(r, \Omega) = \frac{\sum_{1 \le j \le h} Support(v_j \cap e_j(A_k), \Omega)}{\sum_{1 \le j \le h} \left(Support(v_j, \Omega) \times Support(e_j(A_k), \Omega)\right)}.$$

□

Property 3.3 *For any rule r, there exists a base rule br_i which is specialization of r but not r' and whose strength is at least that of r.*

Proof. Without loss of generality, assume that $r = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ and $\Omega = S_1, S_2, \ldots, S_t$ be the sequence of snapshots we consider. Let $br_1, br_2, \ldots, br_h$ be the set of base rules which are specializations of r, where $br_j = e_j(A_1) \cap e_j(A_2) \cap \ldots \cap e_j(A_{k-1}) \cap e_j(A_{k+1}) \cap \ldots \cap e_j(A_n) \Longrightarrow e_j(A_k)$. Then, by Lemma 6.3,

$$Strength(r, \Omega) = \frac{\sum_{1 \le j \le h} Support(v_j \cap e_j(A_k), \Omega)}{\sum_{1 \le j \le h} \left(Support(v_j, \Omega) \times Support(e_j(A_k), \Omega)\right)}$$

where $v_j = e_j(A_1) \cap e_j(A_2) \cap \ldots \cap e_j(A_{k-1}) \cap e_j(A_{k+1}) \cap \ldots \cap e_j(A_n)$. By Corollary 6.2, there must exist a base rule br_i $(1 \le i \le h)$ such that

$$Strength(r, \Omega) \le \frac{Support(v_i \cap e_i(A_k), \Omega)}{Support(v_i, \Omega) \times Support(e_i(A_k), \Omega)} = Strength(br_i, \Omega).$$

Therefore, this property holds. □

Property 3.4 *For any two rules r and r' where r' is a specialization of r, and the strength of r' is less than that of r, there exists another base rule br_i which is a specialization of r and whose strength is greater than that of r.*

Proof. Without loss of generality, assume that $r = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longrightarrow E(A_k)$ and $\Omega = S_1, S_2, \ldots, S_t$ be the sequence of snapshots we consider. Let $br'_1, br'_2, \ldots, br'_{h'}$ be the set of base rules which are specializations of r' and $br_1, br_2, \ldots, br_h$ be the set of base rules which are specializations of r but not r', where $br'_j = e'_j(A_1) \cap e'_j(A_2) \cap \ldots \cap e'_j(A_{k-1}) \cap e'_j(A_{k+1}) \cap \ldots \cap e'_j(A_n) \Longrightarrow e'_j(A_k)$ and $br_j = e_j(A_1) \cap e_j(A_2) \cap \ldots \cap e_j(A_{k-1}) \cap e_j(A_{k+1}) \cap \ldots \cap e_j(A_n) \Longrightarrow e_j(A_k)$. Then, by Lemma 6.3,

$$Strength(r, \Omega) = \frac{\sum_{1 \le j \le h} Support(v_j \cap e_j(A_k), \Omega) + \sum_{1 \le j \le h'} Support(v'_j \cap e'_j(A_k), \Omega)}{\sum_{1 \le j \le h} \left(Support(v_j, \Omega) \times Support(e_j(A_k), \Omega)\right) + \sum_{1 \le j \le h'} \left(Support(v'_j, \Omega) \times Support(e'_j(A_k), \Omega)\right)}$$

$$Strength(r', \Omega) = \frac{\sum_{1 \le j \le h'} Support(v'_j \cap e'_j(A_k), \Omega)}{\sum_{1 \le j \le h'} \left(Support(v'_j, \Omega) \times Support(e'_j(A_k), \Omega)\right)}$$

where $v'_j = e'_j(A_1) \cap e'_j(A_2) \cap \ldots \cap e'_j(A_{k-1}) \cap e'_j(A_{k+1}) \cap \ldots \cap e'_j(A_n)$ and $v_j = e_j(A_1) \cap e_j(A_2) \cap \ldots \cap e_j(A_{k-1}) \cap e_j(A_{k+1}) \cap \ldots \cap e_j(A_n)$. Since $Strength(r', \Omega) < Strength(r, \Omega)$, by Lemma 6.1, there must exist a base rule br_i which is specialization of r but not r' such that

$$Strength(r, \Omega) < \frac{Support(v_i \cap e_i(A_k), \Omega)}{Support(v_i, \Omega) \times Support(e_i(A_k), \Omega)} = Strength(br_i, \Omega).$$

Therefore, this property holds. □

Theorem 6.4 (Correctness of Rule Set Discovery Algorithm) *Given some thresholds for density, support, and strength, the rule set discovery algorithm returns the correct answer.*

Proof. This involves two aspects. (1) All rule sets returned by the algorithm satisfy the requirement for density, support, and strength. (2) Every valid rule is returned by the algorithm. The first aspect follows directly from the algorithm since the expansion stops whenever one of the requirement becomes violated. Thus, we focus on the second aspect and prove that for any rule r which satisfies the requirement for density, support, and strength, there must exist a rule set $< r_{min}, r_{max} >$ such that r is a specialization of r_{max} and a generalization of r_{min}. This is proved by contradiction. Assume that there exist a rule r such that r is valid but not included by any rule set returned by the algorithm. Let the thresholds for density, support, and strength be ϵ, $SUPPORT$, and $STRENGTH$, respectively. By Property 3.3, there must exist a base rule br_i which is a specialization of r and whose strength is at least $STRENGTH$. Let $BR = \{br_1, br_2, \ldots, br_g\}$ be the set of all base rules which are specializations of r and whose strengths are at least $STRENGTH$. Also let r' be the minimum common generalization of base rules in BR. Obviously, r' is also a specialization of r. Then, since r does not belong to any rule set in the result, the following scenario must happen. When the algorithm expands from r', the expansion stops before r is reached. The only reason could cause this is that, during the expansion, the strength goes below $STRENGTH$ before the required support is met. Let r'' be the rule corresponding to the evolution cube where the expansion stops. By Property 3.4, there must exist another base rule br' which is a specialization of r but not r'' such that the strength of br' is greater than $STRENGTH$. This contradicts with the fact that BR is the set of all base rules which are specialization of r and whose strengths are at least $STRENGTH$, and r'' is a generalization of all base rules in BR. Therefore, this theorem holds. □

Note that all results presented in this section apply regardless of whether equi-length or equi-depth partition is used.

7 Related Work

[Sri96a] introduced the problem of mining association rule involving quantitative attributes. The domain of each quantitative attribute is divided into a set of intervals by equi-depth partitioning. Intervals are combined as long as their support is less than the user specified max-support. The original intervals and the combined intervals are treated as a set of items. The algorithms for mining traditional association rules can be applied on these items. In addition, this paper also presented a measure called "partial completeness" to quantify the information loss due to partitioning and proves that the equi-depth partition minimizes the information loss. Max-support is used to restrict the combining of adjacent intervals, but it can also remove some strong and interesting rules, e.g., rules with very high support which is greater than the max-support threshold.

[Fuk96a] introduced so-called "optimized association rules" of form $U \cap C_1 \Longrightarrow C_2$, which contain a single uninstantiated condition $U = A \in [l, u]$ on the left hand side, and proposed schemes to determine values for variable l and u such that one of confidence or support of the rule is maximized while the other satisfies the requirement. [Fuk96b] extended the results in [Fuk96a] to the case in which rules contain two uninstantiated numerical attributes on the left hand side. [Ras98] again generalized the problem and proposed algorithms to handle more than two uninstantiated attributes on the left hand side which can be either categorical or numerical. [Ras99] further generalized optimized support association rule problem by permitting rules to contain disjunctions over uninstantiated numerical attributes. A dynamic programming algorithm is proposed to compute these rules. However, all of these approaches assume that a categorical constraint is prefixed on both sides and the attributes in U are known in advance.

None of the above work addressed the temporal aspect of association rules. A new data mining problem, discovering sequential patterns, was introduced in [Agr95] and [Sri96b]. The input data is a set of sequences, called data-sequences. Each data-sequence is a list of transactions. Typically there is a transaction time associated with each transaction. A sequential pattern also consists of a list of sets of items. The problem is to find all sequential patterns with a user-specified minimum support, where the support of a sequential pattern is the percentage of data-sequences that contain the pattern. However, the attribute values are binary and the sequential patterns are limited in the sense that the class of patterns allow the representing the appearance of items, but not their absence.

The inclusion of a user defined calendar is studied in [Ram98]. A user explicitly defines a calendar and interesting patterns are discovered based on the calendar. For example, if a user defines temporal subsequence to start on the days when the US government announces the unemployment rate as this calendar and the calendar is applied to the stock prices in New York Stock Exchange, then some interesting patterns can be discovered relating the reaction of stock prices to these announcements.

[Han99] presented algorithms for efficiently mining partial periodical patterns. By exploring some interesting properties related to partial periodicity such as the Apriori Property and the max-subpattern hit set property. However, this paper did not address the problem with numerical attributes and the period is predefined.

8 Conclusion

In this paper, we studied a interesting problem: discovery of correlations among temporal evolutions numerical attributes. The temporal association rules are introduced to describe this type of relationships. Three parameters: support, strength, and density to qualify the validity of a rule. We also devise an algorithm which can efficiently find the set of rules based on user specified thresholds. An efficient algorithm is also devised to mine the set of temporal association rules satisfying the thresholds for support, strength, and density. A number of interesting extensions of this work are currently being undertaken. This includes temporal association rules which are not constrained to consecutive snapshots and association rules in which more complex Boolean expressions are permitted, i.e., not only conjunctions.

References

[Agr93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *Proc. of the ACM conf. on Management of Data*, 1993.

[Agr94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. of the 20th Intern. Conf. on Very Large Databases*, 1994.

[Agr95] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proc. of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, March 1995.

[Bri97a] S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic Itemset counting and implication rules for market basket data. *Proc. ACM SIGMOD Conf. on Management of Data*, 1997.

[Bri97b] S. Brin, R. Motwani, C. Silverstein. Beyond market baskets: generalizing association rules to correlations. *Proc. ACM SIGMOD Conf. on Management of Data*, 1997.

[Fel97] Ronen Feldman, Yonatan Aumann, Amihood Amir, and Heikki Mannila. Efficient algorithms for discovering frequent sets in incremental databases. *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, 59-66, 1997.

[Fuk96a] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. *Proc. the 15th ACM Symposium on Principles of Database Systems*, 1996.

[Fuk96b] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two dimensional optimized association rules: scheme, algorithms, and visualization. *Proc. ACM Conf. on Management of Data*, 1996.

[Han95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. *Proc. of the 21st Intern. Conf. on Very Large Databases*, 1995.

[Han99] J. Han, G. Dong, and Y. Yin. Efficient mining partial periodic patterns in time series database. To appear in *Proc. of Intern. Conf. on Data Engineering*, 1999.

[Man97a] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Technical Report C-1997-8*, University of Helsinki, Department of Computer Science, January 1997.

[Man97b] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Technical Report C-1997-15*, University of Helsinki, Department of Computer Science, February 1997.

[Ram98] Sridhar Ramaswamy, Sameer Mahajan, and Abraham Silberschatz. On the discovery of interesting patterns in association rules. *Proc. 24th Intl. Conf. on Very Large Data Base (VLDB)* 1998.

[Ras98] R. Rastogi and K. Shim. Mining optimized association rules with categorical and numerical attributes. *Proc. 14th Intl. Conf. on Data Engineering (ICDE)*, 1998.

[Ras99] R. Rastogi and K. Shim. Mining optimized support rules for numerical attributes. *Proc. 15th Intl. Conf. on Data Engineering (ICDE)*, 1999.

[Sri96a] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 1996.

[Sri96b] R. Srikant and R. Agrawal. Mining sequential patterns: generalizations and performance improvements. *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, March 1996.

[Wan99] Wei Wang, Jiong Yang, and Richard Muntz. Temporal association rules with numerical attributes. *UCLA CSD Technical Report 990011*, 1999.

A Dense Region Discovery Algorithm

Algorithm A.1 *Find Dense Base Cubes*

DenseBaseCube(ϵ, N, b)

{

$Candidate_1 \leftarrow$ *the set of all base intervals (each of which is associated with a counter set to 0)*

$Dense_1 \leftarrow \emptyset$

$level \leftarrow 1$

while $Candidate_{level} \neq \emptyset$ **do**

Scan the data and increment corresponding counters in $Candidate_{level}$

for *each cube* $bc \in Candidate_{level}$ **do**

if $bc.counter \geq \epsilon \times \frac{N}{b}$

then $Dense_{level} \leftarrow Dense_{level} \cup bc$

if $Dense_{level} \neq \emptyset$

then

$Candidate_{level+1} \leftarrow Construct(Dense_{level})$

/ The function* $Construct(Dense_{level})$ *is to construct all candidate dense base cubes at the next level in Figure 4. */*

$Dense_{level+1} \leftarrow \emptyset$

$level \leftarrow level + 1$

else $Candidate_{level+1} \leftarrow \emptyset$

$level \leftarrow level + 1$

}

B Rule Set Discovery Algorithm

Algorithm B.1 *Find Valid Rule Sets*

Valid_Rule_Sets ($D_{i,m}$, $SUPPORT$, $STRENGTH$)

{

$result \leftarrow \emptyset$

for *each dense region* dr *in* $D_{i,m}$

for $k \leftarrow 1$ **to** i **do**

*/*mining pattern of* $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_i) \Longrightarrow E(A_k)$ **/*

$DC \leftarrow$ *set of base cubes whose corresponding strength are at least* $STRENGTH$ *in* dr

$DS \leftarrow$ *set of minimum generalizations of any subset of base cubes in* DC

for *each* $c \in DS$ **do**

if $rule(c).strength \geq STRENGTH$

then $result \leftarrow result \cup Exam(c, dr, SUPPORT, STRENGTH)$

return $result$

}

Exam (c, dr, $SUPPORT$, $STRENGTH$)

{

$result \leftarrow \emptyset$

$r_{min} \leftarrow Mine\_Min\_Rule((c, dr, SUPPORT, STRENGTH)$

for *each* $r \in r_{min}$ **do**

$r_{max} \leftarrow Mine\_Max\_Rule(c, dr, DC, SUPPORT, STRENGTH)$

for *each* $r' \in r_{max}$ **do**

$result \leftarrow < r, r' >$

return $result$

}

Mine_Min_Rule $(c, dr, SUPPORT, STRENGTH)$

{

$r_{min} \leftarrow \emptyset$

if $c.support < SUPPORT$

$Candidate \leftarrow Expand(c)$

for *each* $c' \in Candidate$ **do**

if $within(c', dr)$ **and** $rule(c').strength > STRENGTH$ **and** $rule(c').support > SUPPORT$

then $r_{min} \leftarrow r_{min} \cup Mine\_Min_{R}ule(c', dr, SUPPORT, STRENGTH)$

else if $rule(c').strength > STRENGTH$ **and** $c'.support > SUPPORT$

then $r_{min} \leftarrow r_{min} \cup rule(c')$

return r_{min}

}

Mine_Max_Rule $(c, dr, DC, SUPPORT, STRENGTH)$

{

$r_{max} \leftarrow \emptyset$

if $rule(c).strength > STRENGTH$

$Candidate \leftarrow Expand(c)$

for *each* $c' \in Candidate$ **do**

if $within(c', dr)$ **and** $c'.strength > STRENGTH$ **and** $exclude(c, c', DS)$

then $r_{max} \leftarrow r_{max} \cup Mine\_Max\_Rule(c', dr, SUPPORT, STRENGTH)$

if $r_{max} = \emptyset$

then $r_{max} = rule(c)$

return r_{max}

}