TAR: Temporal Association Rules on Evolving Numerical Attributes

Wei Wang, Jiong Yang
T.J. Watson Research Centers
IBM
{ww1, jiyang}@us.ibm.com

Richard Muntz
Department of Computer Science
UCLA
muntz@cs.ucla.edu

Abstract

Data mining has been an area of increasing interests 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 temporal association rule model for evolving numerical attributes. Metrics for qualifying a temporal association rule include the familiar measures of support and strength used in the traditional association rule mining and a new metric called density. *The density metric not only gives us a way to extract the rules that best represent the data, but also provides an effective mechanism to prune the search space. An efficient algorithm is devised for mining temporal association rules, which utilizes all three thresholds (especially the strength) to prune the search space drastically. Moreover, the resulting rules are represented in a concise manner via rule sets to reduce the output size. 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 demonstrated 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 [1, 2, 5, 7, 8] 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 cannot describe relationships such as

- *Supermarket.* If the price per item of A falls below \$1, then the monthly sales of item B rise by a margin between 10000 and 20000.

- *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 patterns involving numerical attributes and temporal evolution has significant practical applications (e.g., segmenting a large customer-oriented database for market targeting) and is also quite challenging. Recently there have been a number of advances in this area. We provide an overview of some recent results in the following section. In this paper, we aim at mining *temporal association rules* which capture the correlation among (numerical) attribute 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. 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. For example, in an employee database, each person is an object and age, salary, height, etc. are attributes associated with each object. A sequence of snapshots of objects and their attribute values are taken at some frequency. Here, *time* plays a role in synchronizing these numerical attributes, i.e., each attribute is recorded at the same sequence of time instants.

Due to the nature of numerical attributes, many valid but uninteresting rules may exist if support and strength are the only criteria employed. For example, the following two rules have the same value for support and strength (confidence/interest) assuming that there is no new employee younger than 25 in the database.

1. *If a new employee's age is between 25 and 30, then his (her) salary would start between \$40,000 and \$60,000, and increase by at least \$2000 each subsequent year.*

2. *If a new employee's age is between 20 and 30, then his (her) salary would start between \$40,000 and \$60,000, and increase by at least \$2000 each subsequent year.*

Certainly, the first rule conveys more precise information and is preferable. In order to filter out the second rule, we use **density** as an additional metric for determining whether a rule is valid. For a given object, the evolution of its attributes over a period of time can be mapped to a point in a high dimensional space. The distance between a pair of such points can be regarded as an indication of the disparity in the attribute evolution of corresponding objects. If a significant number of points are "close" to each other (i.e., a cluster is present), then their corresponding object evolutions must follow a similar pattern, which in turn suggests that some correlation may exist among the evolutions of involved attributes. A temporal association rule can be viewed as a hypercube in such a high dimensional space, which contains the set of objects that satisfy the rule. Thus, the clusters can serve as a natural foundation for generating temporal association rule. Much work has been done in the area of high dimensional clustering. We adopt a **density**-based approach [3, 11, 13, 14], in which a cluster is defined as a region where the density is above a user-specified threshold. This density threshold requirement provides the following benefits.

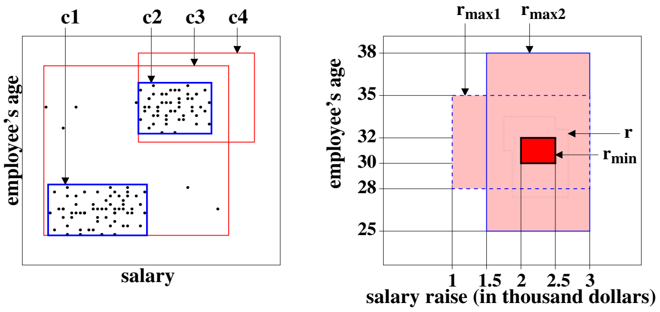
- Only hypercubes "fully occupied" by a cluster of points with some minimum density are thereby identified. For example, c_1 and c_2 in Figure 1(a) are reported while c_3 and c_4 are not qualified for generating temporal association rules. Intuitively, this coincides with the notion that there should be enough samples everywhere within the intervals specified in a rule to show that the rule is valid over the entire range. As a result, an interval would 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 $[20, 25]$ is excluded from any rule as the employee's age since no new employee is younger than 25.

- Since a temporal association rule may represent the correlation in the evolutions of any subset of attributes over a period of time, it is necessary to find clusters embedded in any subspace (i.e., a space consisting of a subset of dimensions of the original space). The density metric enables us to prune the search space dramatically in finding subspace clusters [3].

The density parameter establishes a natural connection between a temporal association rule and a density-based cluster. As a more restrictive model than clustering, the temporal association rule can be mined in two steps.

1. *Find density-based clusters.* A clustering algorithm is adopted to find clusters within the mapped high dimensional space. Since a temporal association rule represents the correlation in the evolutions of any subset of attributes over a period of time, it is necessary to find clusters embedded in a subspace. We employ a bottom-up subspace clustering algorithm similar to [3] to find all clusters with some minimum density and support.

2. *Rule generation.* For each identified cluster, we generate rules which have enough support and strength. Both the well-known Apriori property (with respect to the support metric) and a novel property based on the strength metric are used in prune the search during the rule generation process.



(a) clusters in domain space **(b) min-rules and max-rules**

Figure 1. Example of Employee Database

It is well known that the number of valid rules is often large in association rule mining applications and, without additional filtering, would be even 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* 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(b) represents the attribute values which satisfy the support, strength (confidence/interest), and density requirement. Any subregion of the light shaded region, which contains the dark shaded area, also satisfies the requirements (e.g., r in Figure 1(b) 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(b)) to represent these subregions. The formal definition of rule sets is presented in Section 3. This representation of large sets 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.

Contribution of the paper: In this paper, we propose a *temporal association rule* model to represent the correlation among numerical attribute evolutions. Besides the commonly used metrics such as *support* and *strength*, an additional metric *density* is introduced to qualify valid temporal association rules. A novel approach is proposed to mine temporal association rules by first discovering subspace clusters followed by efficiently constructing rules via special properties held by the support and strength metrics. Empirical evaluation shows that this approach outperforms alternative algorithms by a large margin in terms of execution time. Last but not least, the result is represented in a concise manner via *rule sets*.

The remainder of the paper is organized as follows. Section 2 presents some related work. The general model is presented in Section 3. Section 4 describes our approach to mining temporal association rules with numerical attributes in detail. Performance analysis is presented in Section 5. Finally, conclusions are drawn in Section 6.

2 Related Work

The model of quantitative association rules is introduced in [9]. The domain of each quantitative attribute is divided into a set of intervals and 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 to these items. Unfortunately, the Max-support threshold may exclude some strong and interesting rules from being discovered. In order to overcome this problem, in our model we will use a more meaningful metric, namely *density*, to restrict the search space.

[6] presented a geometric-based algorithm, BitOp, for performing clustering on association rules of the form $A \cap B \Longrightarrow C$ where the left hand side attributes (A and B) are quantitative and the right hand side attribute (C) is categorical. Given a rule format $A \cap B \Longrightarrow C$, the attribute domain is partitioned into two dimensional grids. For each specific value of C, data in each individual grid cell is examined to determine whether the association rule applies for that grid cell and the result is represented by a bit in a two dimensional bitmap. Then, a *clustered association rule* can be formed by combining a set of association rules with identical right hand side but "adjacent" left hand side attribute values. A smoothing technique is employed to *cover* "small holes" within a big cluster. The result is an approximation to the result a human might construct.

None of the above work addressed the temporal aspect of association rules. The authors of [16] consider the subject of mining co-evolving time sequences online. A method, namely MUSCLES, is proposed to use a multi-variate linear regression model based on the portions of given time sequences within a *tracking window* to detect correlations, outliers, and to predict missing data.

[10] proposed a scheme, called FITI, to mine inter-transaction association rules on categorical attributes. This algorithm was inspired by the observation that an inter-transaction frequent itemset contains only the frequent itemsets of its corresponding intra-transaction counterpart. A special data structure containing intra-transaction frequent itemsets is used to provide fast mining of inter-transaction frequent itemsets.

We also notice that many research has been conducted in the areas of mining "optimized association rules", mining association rules with constraints, mining sequential patterns in time series data, discovering periodic patterns, and similarity search in time sequence(s). Interested readers please refer to [13] for a detailed overview.

Alternative solutions to our problem

Inspired by [9] and [6], the following are two alternative solutions to our problem.

SR algorithm: One method is to map the numerical attribute evolutions to binary attributes and apply one of the existing algorithms for mining association rules [9]. 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 for each attribute. Therefore, if the data consists of t snapshots, $O(b^2 \times t)$ items are required to encode all possible evolutions of an attribute range. 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. However, this creates a huge number of items and hence makes the mining process very inefficient.

LE algorithm: In this method, after domain quantization, rules are first generated for each possible right hand side attribute and each possible value of this attribute. Then final rules are formed by combining "adjacent" association rules with identical right hand sides [6]. This is due to the fact that the original algorithm was designed for categorical right hand side attribute. Therefore, in order to solve the problem presented in this paper, each possible evolution of the right hand side attribute has to be mapped into a distinct (categorical) value. After generating the rules, the combining process has to be performed separately for each possible rule format and each possible evolution of the right hand side attribute. For example, for each right hand side attribute A, if its domain is partitioned into b intervals and we consider its evolution over t snapshots, then there could be b^{2t} distinct evolutions. This approach becomes inefficient, in the context we address, as the number of possible attribute evolutions (which can serve as the right hand side attribute of the rule) explodes exponentially with the total number of attributes and/or length of attribute evolution time series. In contrast, our proposed algorithm handles attributes on the right hand side of the rule in a similar manner as the attributes on the left hand side. Thus, the necessity of transforming the right hand side into categorical values can be avoided. This characteristic of our algorithm leads to a significant performance advantage over the previous approach. This observation is verified by the performance results in Section 5.

3 General Model

In this paper, we deal with objects. Each object has a unique ID and a set of numerical attributes $A_1, A_2, \ldots$. 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. We aim at finding correlations among evolutions of attributes. In contrast to the traditional association rule discovery, 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. 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. This evolution of attribute $salary$ involves 3 snapshots and can be represented as $(salary \in [40000, 45000]) \to (salary \in [47500, 55000]) \to (salary \in [60000, 70000])$. "3" is referred to as the *length* of the evolution. Formally, given an attribute A_i and m snapshots, an **evolution** $E(A_i)$ of length m describes the range of values of A_i at each snapshot, which is represented as $(A_i \in [l_1, u_1]) \to (A_i \in [l_2, u_2]) \to \ldots \to (A_i \in [l_m, u_m])$ where $[l_j, u_j]$ is the range of values of A_i at the jth snapshot. Conceptually, an m dimensional space can be constructed by associating the jth dimension to the value of A_i at the jth snapshot relative to the start of the evolution. This space is referred to as the *evolution space* of A_i for evolutions of length m. Any evolution (of A_i) of length m uniquely corresponds to an m dimensional hypercube (referred to as *evolution cube*) in the evolution space, and vice versa. For example, Figure 2 shows the evolution space for evolutions on $salary$ of length 3. The evolution $E_1 = (salary \in [40000, 45000]) \to (salary \in [47500, 55000]) \to (salary \in [60000, 70000])$ corresponds to the shaded cube.

For any two evolutions $E(A_i) = (A_i \in [l_1, u_1]) \to (A_i \in [l_2, u_2]) \to \ldots \to (A_i \in [l_m, u_m])$ and $E'(A_i) = (A_i \in [l'_1, u'_1]) \to (A_i \in [l'_2, u'_2]) \to \ldots \to (A_i \in [l'_m, u'_m])$ of the same length m, $E(A_i)$ is a **specialization** of $E'(A_i)$ iff, for every $j (1 \le j \le m)$, $[l_j, u_j]$ 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 $(salary \in [40000, 55000]) \to (salary \in [40000, 60000]) \to (salary \in [60000, 70000])$. However, it is not a specialization of $(salary \in [40000, 50000]) \to$

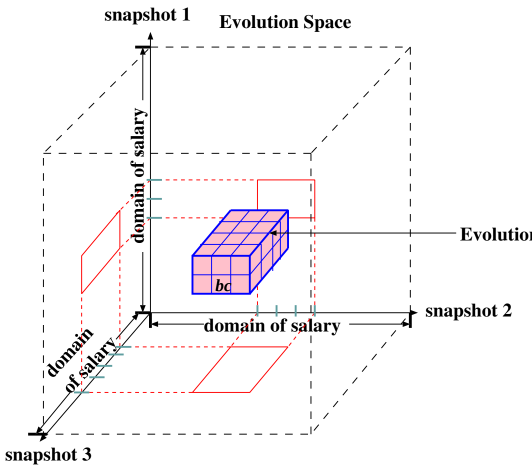


Figure 2. Evolution Space for $salary$

$(salary \in [40000, 50000]) \to (salary \in [60000, 65000])$. Note that an evolution is always a specialization and a generalization of itself. This specialization/generalization relationship forms a lattice among evolutions.

Multiple attribute evolutions

Evolutions can span any set of snapshots. In this paper, we focus on mining correlations of simultaneous evolutions (of different attributes), i.e., evolutions over the same set of snapshots. 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 $(1 \le j \le n)$, $E(A_i)$ is a specialization of $E'(A_i)$. In this case, EC' is also called a **generalization** of EC.

3.1 Temporal Association Rules

Our proposed temporal association rule has the form of $X \iff Y$ where X and Y are conjunctions of attribute evolutions. (We use $\iff$ instead of $\implies$ since correlation is indeed a symmetric relationship.) 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 3.1 *Let $A_1, A_2, \ldots, A_n$ be a subset of attributes. A* **temporal association rule** *of length m is a correlation $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \iff E(A_k)$ and $E(A_i)$ is an evolution (of length m) of attribute A_i $(1 \le i \le n)$.*

For any two temporal association rules $R = E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \Longleftrightarrow E(A_k)$ and $R' = E'(A_1) \cap E'(A_2) \cap \ldots \cap E'(A_{k-1}) \cap E'(A_{k+1}) \cap \ldots \cap E'(A_n) \Longleftrightarrow E'(A_k)$ of the same length m, 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. For example, $R_1 = (salary \in [40000, 55000]) \rightarrow (salary \in [40000, 60000]) \Longleftrightarrow (housing\_expense \in [10000, 15000]) \rightarrow (housing\_expense \in [10000, 17000])$ and $R_2 = (salary \in [40000, 55000]) \rightarrow (salary \in [40000, 50000])) \Longleftrightarrow (housing\_expense \in [10000, 15000]) \rightarrow (housing\_expense \in [10000, 12000])$ are two temporal association rules of length 2 and R_1 is a generalization of R_2. This specialization/generalization relationship also defines a lattice among temporal association rules.

Metrics for selecting interesting rules

In general, for a given set of attributes, many temporal association rules may exist simultaneously. We need to identify which of these rules are of interests. Criteria for assessing interest in a rule are elaborated in the following discussions.

For any rule, its **support**, **density**, and **strength** are indications of frequency of occurrence, concentration of population and the degree of non-independency 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 [1, 2]. In addition, due to the nature of numerical data, users need to specify one more parameter, *density*. Intuitively, this is motivated by the notion that clustering may provide an efficient means to segment objects naturally according to their behavior and, on the other hand, 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. The concepts of *window* and *object history* are then introduced to facilitate the mining of rules of any length. 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 sequence of temporal changes of o over $W(j, m)$ is called the **object history** of o within window $W(j, m)$. The *length* of this object history is m. An object history can be mapped to a point in the corresponding evolution space of the involved attributes. A temporal association rule can be viewed as a hypercube in this 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. These thresholds will be discussed in more detail shortly.

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 (i.e., $E(A_i)$) 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 (defined in the previous example). 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]. 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 $E(A_i)$ $(1 \leq i \leq n)$.

As mentioned before, clusters are generated as a first step in constructing temporal association rules. In this paper, we employ an efficient clustering algorithm based on space partitions similar to [3, 11]. The space is discretized into small base cubes and a cluster is thereby approximated by a set of adjacent base cubes which meet some density threshold. Each attribute domain is quantized into a set of disjoint equal-length intervals, referred as *base intervals*, and a certain amount of information loss is introduced. For simplicity of exposition, in this paper we assume that the 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 as non-distinguishable. As a result, the evolution space consists of b^m basic hypercubes, referred to as **base cubes**. Each evolution cube is then represented as the union of a set of adjacent base cubes. For example, the evolution cube of E_1 in Figure 2 consists of 24 base cubes. (The span of a base cube in each dimension in this example is 2,500.)

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 all windows (of length 3) are examined to aggregate the overall support, strength, and density. We now discuss each of these three measures in detail.

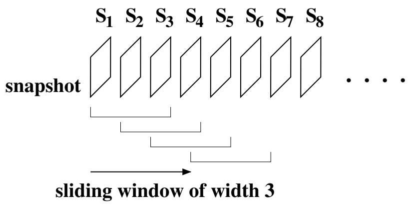


Figure 3. Sliding Window

3.1.1 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 $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \iff E(A_k)$ 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 3.2 *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* $W(j, m)$.

3.1.2 Strength

Different methods can be used to capture the degree of non-independence. In this paper, we use a metric that is similar to *interest* defined in [4] to measure the strength of a temporal association rule.

Definition 3.3 *Given a temporal association rule* $R : X \iff Y$ *and a sequence of snapshots* $\Omega : S_1, S_2, \ldots, S_t$, *the* **strength** *of the rule is* $\frac{Support(X \cap Y, \Omega)}{Support(X, \Omega) \times Support(Y, \Omega)}$.

3.1.3 Density

Since we are concerned with numerical attributes, there can be unoccupied regions in the evolution space, especially when the attribute domain is continuous. The density concept is introduced to enable a cluster oriented approach and to ensure that a region in the evolution space will not be included as part of a rule if there is no enough evidence to show that the rule holds for the region. More specifically, a valid rule corresponds to an evolution cube in which object history concentration within each base cube contained in the evolution cube is above the specified density threshold. Thus, we introduce the *density* parameter to capture the object history concentration of the sparsest base cube within the 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 falling into each base interval is $\frac{N}{b}$. $\frac{N}{b}$ can be viewed as the "average density". The density threshold may be specified as a ratio (denoted by ϵ) [1] of the number of object histories, which will make a base cube considered "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 $(salary \in [40000, 42500]) \rightarrow (salary \in [50000, 52500]) \rightarrow (salary \in [67500, 70000])$. 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 3.4 *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.

In other words, the density of an evolution conjunction is equal to the minimum density of any of its enclosed base cubes. The density of a rule $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \iff E(A_k)$ 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.

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

3.2 Rule Sets

Definition 3.5 *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.*

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

4 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. This problem can be transformed into a problem of identifying clusters in a high dimensional space.

Two phases involved are involved in finding the temporal association rules:

1. After partitioning each attribute domain into b base intervals, find all (subspace) clusters with respect to the density threshold.
2. Based on identified clusters, find all valid rule sets.

4.1 Cluster Discovery

We first find all dense base cubes in a bottom-up fashion and then generate clusters by coalescing adjacent dense base cubes [3, 14]. The Apriori property holds on the density metric [3, 14] and can be restated into the following properties that are directly used in pruning the evolution space.

Property 4.1 *Let $E = (A_i \in [l_1, u_1]) \rightarrow (A_i \in [l_2, u_2]) \rightarrow \ldots \rightarrow (A_i \in [l_m, u_m])$ and $E' = (A_i \in [l_j, u_j]) \rightarrow (A_i \in [l_{j+1}, u_{j+1}]) \rightarrow \ldots \rightarrow (A_i \in [l_k, u_k])$ 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 the 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 4.2 *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 4.1, Property 4.2 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) [7] 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 4.1, both of bc's projections in $BaseCube(i, m-1)$ have to satisfy the threshold as well. In addition, by Property 4.2, all bc's projections in $BaseCube(i-1, m)$ must also satisfy the density threshold.

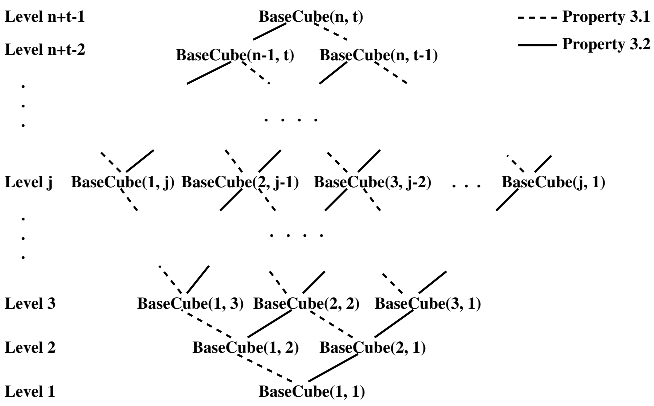


Figure 4. Base Cube Lattice

Therefore, beginning from the set of all base intervals, 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 set $BaseCube(i, m)$ is examined during the $(i+m-1)$th scan of the data. Let θ be the maximum value of $i+m-1$ for all dense base cubes where i and m are the number of involved attributes and the evolution length of a base cube, respectively. Then the algorithm makes θ passes over the database. Let $\gamma = i \times m$ be the highest dimensionality of any dense base cube[2]. Then the algorithm is exponential with respect to γ since all of its projections of this dense base cube onto a subset of these γ dimensions are also dense. It follows that the computational complexity of this step is $O(\theta \times \mid \Omega \mid + c^{\gamma})$ for a small constant c where $\mid \Omega \mid$ is the data size.

After we find all dense base cubes, a set of clusters can be formed by linking adjacent base cubes. This step is equivalent to finding connected component in a graph. Each dense base cube is mapped to a graph vertex, and there is an edge between two vertices if the corresponding dense base cubes are adjacent (i.e., they share a common face). A depth-first traversal through this graph would be able to find all clusters.

In fact, any algorithm proposed for efficiently computing subspace clusters can be adopted in this phase. We will not elaborate on them in this paper due to space limitations. Since every valid rule has to satisfy the density requirement,

[2]Note that $\gamma \neq \theta$ in general.

we only need to explore these clusters in the second phase. In addition, we will not examine a cluster if its support is less than the user specified threshold (because no rule derived from this cluster can meet the required support).

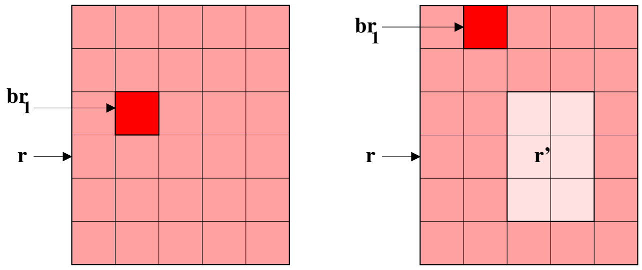
4.2 Temporal Association Rule Discovery

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 $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) \iff E(A_k)$ for each possible choice for the right hand side attribute A_k. Note that, in contrast to the previous algorithm proposed in [6], our approach treats the attribute on the right hand side of a rule in a similar manner to the handling of left hand side attributes. As a result, it avoids the necessity of examining the left hand side attributes repeatedly for each possible value of the right hand side. A significant performance advantage over the previous approach is then achieved as demonstrated in Section 5.

It is obvious that, given an attribute A_k 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 cluster. The finest granularity rule we consider is the rule that corresponds to only one dense base cube. We regard such rules as **base rules**. The following two properties (which are proven in [13]) are used to guide the rule discovery process in each cluster. The final result is the union of all valid rule sets discovered in each cluster.

Property 4.3 *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). Basically it says that any valid rule has to be a generalization of some base rule whose strength is at least $STRENGTH$. Then, for this cluster, we only need to examine rules which are generalizations of the set of base rule(s), BR, whose strengths are at least $STRENGTH$. Let R be this set of rules, each of which is a generalization of some subset of base rule(s) in BR and whose corresponding evolution cube is enclosed by the given cluster. R contains all potential valid rules which can be discovered from this cluster. The following property can be utilized to prune R further.



(a) Property 4.3 **(b) Property 4.4**

Figure 5. Properties for Strength

Property 4.4 *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.*

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' but which 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, in Figure 6 $BR = \{b_1, b_2, \ldots, b_9\}$ 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. (SEC stands for Set of Evolution Cubes.) Any evolution cube that 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.

Therefore, each cluster 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.

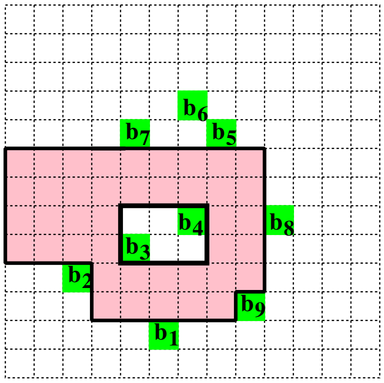


Figure 6. Rule Discovery

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 4.4, 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 4.4), 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 r_{max}, 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. The formal description of the rule sets discovery algorithm is in [13]. The complexity of our algorithm is $O(\lambda^2)$ for a cluster which contains λ dense base cubes. With a reasonable density threshold, λ is small compared to the total number of base cubes in the domain space.

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. Our rule detection algorithm consists of two phases, (1) cluster discovery, and (2) valid rule discovery. In this section, we investigate the advantages of our algorithm with two alternative algorithms on both synthetic and real data. In the remainder of this section, we use "TAR" to denote our algorithm.

5.1 Three Mining Algorithms

In this section, we compare our algorithm (TAR) with two alternative algorithms described in the related work section, and which are referred to as the SR algorithm and the LE 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[3]. Figure 7(a) describes the variation of the average overall response time versus the number of base intervals for each attribute domain. (Note that the y-axis in Figure 7(a) is in the log scale.) 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 1.3, respectively.

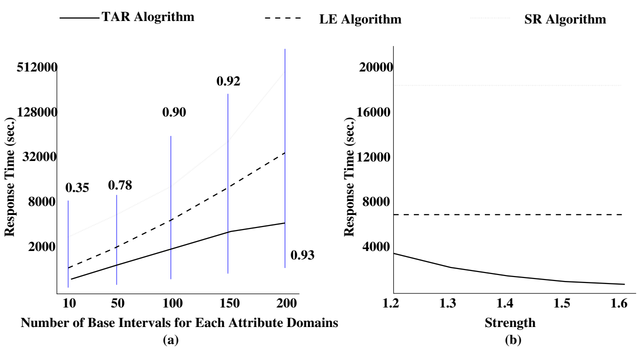


Figure 7. Response Time

It is obvious that the TAR algorithm has a much faster response time than the SR and LE algorithms as shown in Figure 7(a). Moreover, the response time of TAR increases

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

at a slower pace with respect to the number of base intervals. When we choose the number of base intervals to be 100 in the TAR algorithm, the response time is still within an acceptable length, but 90% recall can be achieved.

One of the reasons that TAR has a significant better performance is that the strength threshold is merely used to verify whether a rule is valid in the SR and LE algorithms, whereas the strength is used to prune the search space in the TAR algorithm. In turn, the set of candidate rules searched by the TAR algorithm is much smaller than others and hence the response time is improved. Figure 7(b) shows the response time of the three algorithms as a function of the strength. In this test, the support and density thresholds are set to 5% and 2, respectively and there are 100 base intervals on each domain. The response time of the SR and LE remain constant because they do not use strength as a tool to prune the search space. However, in the TAR algorithm, the strength threshold is utilized to prune the search space, thus the performance is improved.

5.2 Real Data Set

We also ran the TAR 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 raises tend to move further away from the city center." This rule might be explained by the price of a house in the suburbs being more expensive and the area being 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 Conclusions

In this paper, we studied the problem of discovering correlations among temporal evolutions numerical attributes. The temporal association rules are introduced to describe this type of relationships. Three parameters: support, strength, and density are employed to qualify the validity of a rule. We also devise an algorithm that can efficiently find the set of rules based on user specified thresholds. In particular, subspace clustering is employed and some special properties on strength metric are proven and utilized to prune the search space further. Experimental results show that this approach outperforms other alternatives greatly.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *Proc. ACM Conf. on Management of Data*, 207-216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. 20th Int. Conf. on Very Large Databases*, 487-499, 1994.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining application. *Proc. ACM Conf. on Management of Data*, 94-105, 1998.

[4] S. Brin, R. Motwani, C. Silverstein. Beyond market baskets: generalizing association rules to correlations. *Proc. ACM SIGMOD Conf. on Management of Data*, 265-276, 1997.

[5] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. *Proc. 21st Int. Conf. on Very Large Databases*, 420-431, 1995.

[6] B. Lent, A. Swami, and J. Widom. Clustering association rules. *Proc. 13th Int. Conf. on Data Engineering*, 220-231, 1997.

[7] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, vol. 1, no. 3, 241-258, 1997.

[8] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in a large databases. *Proc. 21st Int. Conf. on Very LArge Data Bases*, 432-444, 1995.

[9] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1-12, 1996.

[10] A. Tung, H. Lu, J. Han, and L. Feng, Breaking the barrier of transactions: mining inter-transaction association rules. *Proc. 1999 Int. Conf. on Knowledge Discovery and Data Mining*, 297-301, 1999.

[11] W. Wang, J. Yang, and R. Muntz. STING: a statistical information grid based approach to spatial data mining. *Proc. 23rd Int. Conf. on Very Large Data Base*, 186-195, 1997.

[12] W. Wang, J. Yang, and R. Muntz. STING+: an approach to active spatial data mining. *Proc. 15th Int. Conf. on Data Engineering*, 116-125, 1999.

[13] W. Wang, J. Yang, and R. Muntz. Temporal association rules with numerical attributes. *UCLA Tech. Report 990011*, 1999.

[14] W. Wang, J. Yang, and P. Yu. Efficient mining of weighted association rules (WAR). *Proc. ACM SIGKDD Conf. on Knowledge Discovery and Data mining*, 270-274, 2000.

[15] J. Yang, W. Wang, and P. Yu. Mining asynchronous periodic patterns in time series data. *Proc. ACM SIGKDD Conf. on Knowledge Discovery and Data mining*, 275-279, 2000.

[16] B. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. *Proc. 16th Int. Conf. on Data Engineering*, 13-22, 2000.