Discovering Relationships Among Association Rules Over Time

Chen Chaohai

NATIONAL UNIVERSITY OF SINGAPROE

2008

Discovering Relationships Among Association Rules Over Time

Chen Chaohai
(B.Eng. Harbin Institute of Technology, China)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE
2008

Acknowledgements

I would like to express my sincere gratitude to all those who have shared the graduate life with me and helped me in all kinds of ways. Without their encouragement and support I would not be able to write this section.

Firstly, I would like to thank my supervisor Professor Wynne Hsu for her guidance, advice, patience and all kinds of help. Her kindness and supports are important to my work and her personality also gives me insights which are beneficial to my life and future career. I would also like to thank my co-supervisor Professor Mong Li Lee, who is nice and continuously help me throughout my postgraduate studies. Her guidance and help are really appreciated.

I would like to particularly thank Sheng Chang, Patel Dhaval, Zhu Huiquan and all the other previous and current database group members. Their academic and personal helps are of great value to me. I also feel the need to thank Sun Jun and Lin Yingshuai for their encouragement and support during the period of my thesis writing. They are such good and dedicated friends.

Finally, I would like to thank the National University of Singapore and Department of Computer Science, which give me the opportunity to pursue the advanced knowledge in this wonderful place. The period studying in NUS might be one of the most meaningful parts in my whole life. And I would also like to thank my family, who always trust me and support all of my decisions. They taught me to be thankful to life and made me understand that experience is much more important than the end-result.

Contents

Summary

Association rule mining aims to discover useful and meaningful rules which can be applied to the future data. Most existing works have focused on traditional association rule mining which mines the rules in the entire data, without considering time information. However, more often than not the data nowadays is subjected to change. The rules existing in the evolving data may have dynamic behaviors which might be useful to the user.

In this thesis, we investigate the association rules from temporal dimension. We analyze the dynamic behavior of association rule over time and propose to classify rules into different categories which can help the user to understand and use the rules better. We also define some interesting evolution relationships of association rules over time, which might be important and useful in real-world applications. The evolution relationships reveal the relationships about the effect of the conditions on the consequent over time, which reflect the change of the underlying data. Therefore they can give the domain expert a better idea about how and why the data changes.

To mine association rule in our problem, we partition the whole dataset into positive and negative sub-datasets, then mine the frequent itemsets from the positive

sub-dataset and count the support of the frequent itemsets from the negative sub-dataset. To analyze the dynamic behavior of the rule, we propose to find trend fragments and classify a rule based on the number of its trend fragments over time. To find evolution relationships among rules, we propose Group Based Finding (GBF) method and Rule Based Finding (RBF) method. GBF first groups the comparable trend fragments and then find relationships in each comparable group. RBF directly find relationships among rules.

The effectiveness and efficiency of our approaches are verified via comprehensive experiments on both synthetic and real-world datasets. Our approaches exhibit satisfying processing time on synthetic dataset and the experiments on real-world dataset show that our approaches are effective.

List of Figures

List of Tables

Chapter 1

Introduction

Association rule mining was first introduced to capture important and useful regularities that exist in the data [1]. Formally, association rule mining is stated as follows [2]: Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. An itemset X contains a set of items in I. A transaction T contains X if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I, Y \subset I$, and $X \cap Y = \phi$. X and Y are called the antecedent and consequent of the rule respectively. The rule $X \Rightarrow Y$ has support s in D if s% of the transactions in D contain $X \cup Y$. The rule $X \Rightarrow Y$ holds in the transaction dataset D with confidence c if c% of the transactions in D that contain X also contain Y. The

confidence of a rule is a measure to evaluate the accuracy of the antecedent implying the consequent and the support measures the generality of the rule. The task of association rule mining is to generate all the association rules whose supports and confidences exceed the user-specified minimum support (min_sup) and minimum confidence (min_conf) from the dataset *D*.

With the rapid proliferation of data, applying association rule mining to the huge dataset results in thousands of associations being discovered, many of them are non-interesting and non-actionable. In a dynamic environment where changes occur frequently in a short period of time, it is more important to discover evolving trends in the data. For example, suppose we have collected data of three years as shown in Table 1.1. Applying association rule mining to the entire data in Table 1.1 with a min_sup of 20% will result in association rules being discovered as shown in Table 1.2. None of these rules stands out. However, when we investigate the rules further, we realize that the confidence of the rule "beer $\Rightarrow$ chip" is 20% in 1997, 40% in 1998, and 80% in 1999. In other words, there is an increasing trend in the confidence values of "beer $\Rightarrow$ chip" from 1997 to 1999. This could be useful information to the user.

In addition, when we examine the rules "toothbrush A $\Rightarrow$ toothpaste C" and "toothbrush B $\Rightarrow$ toothpaste C" over each individual year, we observe that the confidence series of "toothbrush A $\Rightarrow$ toothpaste C" from 1997 to 1999 is [100%, 80%, 60%], while the confidence series of "toothbrush B $\Rightarrow$ toothpaste C" is [60%, 80%, 100%]. They have a negative correlation. This may indicate that the two rules

Id	Transaction	Time
1	beer, toothbrush A, toothpaste C	1997
2	beer, toothbrush A, toothpaste C	1997
3	beer, cake, toothbrush A, toothbrush B, toothpaste C	1997
4	beer, chip, toothbrush B	1997
5	chip, cake, toothbrush B, toothpaste C	1997
6	cake, beer, toothbrush B	1997
7	cake, toothbrush B, toothpaste C	1997
8	beer, chip, toothbrush A, toothpaste C	1998
9	beer, chip, toothbrush A, toothpaste C	1998
10	beer, toothbrush A, toothbrush B, toothpaste C	1998
11	chip, toothbrush B, toothbrush A	1998
12	beer, cake, toothbrush A, toothpaste C	1998
13	beer, cake, toothbrush B, toothpaste C	1998
14	chip, toothbrush B, toothpaste C	1998
15	toothbrush B, toothpaste C	1998
16	chip, toothbrush A, toothpaste C	1999
17	beer, chip, toothbrush A, toothpaste C	1999
18	cake, toothbrush A	1999
19	beer, chip, cake, toothbrush B, toothpaste C	1999
20	beer, chip, toothbrush A	1999
21	beer, cake, toothbrush B, toothpaste C	1999
22	beer, chip, toothbrush B, toothpaste C	1999
23	toothbrush A, toothpaste C	1999

Table 1.1: Sample Transactions

have a competing relationship：people who buy toothbrush A or B tend to buy toothpaste C but over the years people who buy toothbrush B are more and more likely to buy toothpaste C; whereas people who buy toothbrush A are less and less likely to buy toothpaste C. As such, if toothpaste C is the key product and the company wants to increase the sale of toothpaste C, it may produce more toothbrush B rather than A as a promotion for buying toothpaste C.

Id	Rule	Confidence
1	beer ⇒ chip	46%
2	chip ⇒ beer	63%
3	beer ⇒ toothpaste C	80%
4	cake ⇒ toothpaste C	77%
5	chip ⇒ toothpaste C	72%
6	toothbrush A ⇒ toothpaste C	76%
7	toothbrush B ⇒ toothpaste C	76%
8	toothpaste C ⇒ toothbrush A	55%
9	toothpaste C ⇒ toothbrush B	55%
10	toothbrush A, toothbrush B ⇒ toothpaste C	66%
….	….	….

Table 1.2 Discovered Association Rules

On the other hand, if the confidence series of "toothbrush A ⇒ toothpaste C" is [60%, 50%, 40%] and the confidence series of "toothbrush B ⇒ toothpaste C" is [70%, 60%, 50%], but the confidence series of "toothbrush A, toothbrush B ⇒ toothpaste C" is [50%, 70%，90%], the relationship between the three rules is interesting as it is counter-intuitive. It indicates that the combined effect of toothbrush A and toothbrush B is opposite to that of toothbrush A and B individually. As such, the company could sell toothbrush A and B together rather than individually if it wants to increase the sell of toothpaste C.

Based on above observations, we wish to investigate the dynamic aspects of association rule mining in this thesis. First, we find the evolving trends of each individual rule over time. In most of the time, it is important to know whether a rule is stable or whether it exhibits some systematic trends. Knowing such dynamic behavior of a rule will enable the user to make better decisions and to take appropriate actions. For example, if the rule exhibits trends, the user can exploit the

desirable trends, and take some preventive measures to delay or change the undesirable trends.

Second, we analyze the correlations among rules in the statistical properties over different time periods. Based on the correlations, we find some unexpected and interesting relationships among rules over time. In general, we are interested to find relationships among the association rules which have the same consequent but different antecedents. Suppose we have three association rules R_1: $\alpha \Rightarrow C$, R_2: $\beta \Rightarrow C$, R_3: $\alpha, \beta \Rightarrow C$, where C is the target item. We focus on the correlations among the confidence series of the rules. The correlations may reflect the change of the underlying data over time. They could help the user to understand the domain better.

There are some challenges in this work. First, since we investigate the association rules over time, the dataset is dynamic and may be huge. It needs an efficient algorithm to mine the association rules. Second, finding evolution relationships among rules is not straightforward. The rules might be of various forms. It is neither reasonable nor necessary to directly analyze the correlations among all rules. Instead we should analyze the dynamic behavior of the rules first and the correlation analysis should be done among the rules within the same category. Third, association rule mining tends to produce huge number of rules and each rule may have many trends. Pairwise way of directly finding relationships among rules might not be so efficient. Efficient algorithms and strategies need to be developed to improve efficiency.

1.1 Contributions

In this thesis, we investigate the trends and correlations in the statistical properties of association rules over time. We propose four categories of rules based on their trends over time and four interesting relationships among rules based on the correlations in their statistical properties. To our best knowledge, this is the first work to find such relationships among association rules over time. Our contributions are summarized as follows:

- Propose an efficient algorithm to mine the association rules with a known consequent
- Design novel algorithms and do some optimizations to discover relationships among the mined rules over time.
- Verify the efficiency and effectiveness of the proposed approaches with synthetic and real-world datasets.

1.2 Organization

This thesis is organized as follows. We introduce the related work in Chapter 2 and give some preliminary definitions about our work in Chapter 3. In Chapter 4, we propose our approaches and in Chapter 5 we evaluate the proposed approaches on both synthetic and real-world datasets. We conclude our work and identify the future research topics in Chapter 6.

Chapter 2

Related Work

Association rule mining was first proposed in R. Agrawal et al. [1]. Since then, many variants of association rule mining have been proposed and studied, such as efficient mining algorithms of traditional association rules [2,4], constraint association rule mining [5-7], incremental mining and updating [8-10], mining of generalized and multi-level rules [11-12], interestingness of association rules [3,13-18] and association rule mining related to time [19-32].

2.1 Association Rule Mining Algorithms

In this section, we briefly introduce two widely used association rule mining algorithms. In general, association rule mining includes two processes [1-2]. The first step is to generate all the frequent itemsets, whose support counts are at least as

large as the predetermined minimum support count. The second step is to generate association rules from the frequent itemsets; these association rules must satisfy the minimum support and minimum confidence. The major challenge is the first step.

Apriori algorithm [2] was first introduced to mine frequent itemsets. The basic idea is to employ the Apriori property of frequent itemsets: all nonempty subsets of a frequent itemset must also be frequent. Based on this property, Apriori algorithm uses a bottom-up strategy. To find frequent k-itemsets L_k, it first generate candidates of frequent k-itemsets C_k by joining L_{k-1} with itself. Since C_k is a superset of L_k, its members may or may not be frequent. According to Apriori, any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Therefore if any (k-1)-subset of a candidate frequent k-itemset is not in L_{k-1}, the candidate cannot be frequent and hence can be removed from C_k. In this way, the size of C_k can be significantly reduced.

J. Han et al. [4] introduces a more efficient algorithm (FP-growth) to mine frequent itemsets without candidate generation. FP-growth adopts a divide-and-conquer strategy. First, it compresses the database representing frequent items into a frequent pattern tree which retains the itemset association information. It then divides the compressed database into a set of conditional databases, each associated with one frequent item, and mines each such database separately. To find long frequent patterns FP-growth searches for shorter ones recursively and then

concatenates the suffix. It uses the least frequent items as a suffix, offering good selectivity. The method substantially reduces the search costs.

These two algorithms are widely used in tradition association rule mining which does not consider any time information.

2.2 Temporal Association Rule Mining

Recently, there have been interests in mining association rule which incorporates time information [19-22]. They consider lifespan of a rule or lifespan of items in the rule.

B. Ozden et al. [19] proposes to find cyclic association rules, where the rules satisfy the min_sup and min_conf at regular time intervals over time. Such a rule does not need to hold for the entire transaction database, but only for transaction data in a particular time interval. For example, we might find that beer and chip are sold together primarily between 6pm and 9pm. Therefore, if we partition the data over the intervals 6am-7am and 6pm-9pm, we may discover the rule "beer $\Rightarrow$ chip" in 6pm-9pm interval. On the other hand, if we mine the whole data directly, the rule could not be found.

However, B. Ozden et al. [19] can only find "cyclic association rules". B. Ozden et al. [20] generalizes the idea of B. Ozden et al. [19] to find calendar association rule, where the author introduces the notion of using a calendar algebra to describe the time period of interest in association rules. This calendar algebra is used to define

and manipulate groups of time intervals. The time intervals are specified by the user to divide the data into disjoint segments. An association rule will be mined if it satisfies the min_sup and min_conf during every time interval contained in a calendar.

In Y. Liu et al. [21], the authors further generalize the idea of S. Ramaswamy et al. [20] by using a calendar schema as a framework for temporal patterns, rather than user-defined calendar algebraic expression. As a result, the approach in Y. Liu et al. [21] requires less prior knowledge. In addition, the approach considers all possible temporal patterns in the calendar schema, thus can potentially discover more temporal association rules and unexpected rules. The main contribution of the work is to develop a novel representation mechanism for temporal association rules on the basis of calendars and identify two classes of interesting temporal association rules: temporal association rules with respect to the full match and temporal association rule with respect to the relaxed match. Association rules with respect to the full match refer to those rules that hold for each basic time interval covered by the calendar; while relaxed match association rules refer to those that hold for at least a certain percentage of time intervals covered by the calendar.

Similarly, J. Ale et al. [22] also incorporates time information in the frequent itemsets by taking into account the items' lifespan. An item's lifespan is the period between the first and the last time when the item appears in the transactions. They compute the support of an itemset in the interval defined by its lifespan and define temporal support as the minimum interval width. Because they limit the total number

of transactions to the items' lifetime, those associations with a high confidence level but with little support would be discovered. The approach differs from the works of [19-21] in that it is not necessary to define an interval or a calendar, since the lifespan is intrinsic to the data.

In another branch of research [23-25], the focus is on mining rules that express the association among items from different transaction records with certain time lag existing in the items of the antecedent and the consequent. Such rules reflect the delayed effect of the items on the others.

S. Harms et al. [23] and S. Harms et al. [24] model the association rule with a time lag between the occurrence of the antecedent and the consequent. The approach finds patterns in one or more sequences that precede the occurrence in other sequences, with respect to user-specified constraints. The approach is well suited for sequential data mining problems which have groupings of events that occur close together. The papers also show that the methods can efficiently find relationships between episodes and droughts by using constraints and time lags.

Similarly, H. Lu et al. [25] also finds association rules that have time lags. The difference is that H. Lu et al. [25] is more general in that the time lag not only exists between the antecedent and the consequent, it can also exist among the items in the antecedent or consequent. One rule they found is that "UOL(0),SIA(1) $\Rightarrow$ DBS(2)" with confidence of 99%, which means if the stock UOL goes down on the first day and SIA goes down the following day, DBS will go down the third day with probability of 99%.

To summarize, the works of [19-25] incorporate time information into association rule mining, either mining association rules in the time intervals where the items appear or association rules with a time lag existing in the items of the antecedent or consequent.

2.3 Association Rules Over Time

Another thread of association rule mining in recent years focus on analyzing the dynamic behavior of association rules over time [26-31] and detecting emerging pattern or deviation between two consecutive datasets [32].

S. Baron et al. [26] proposes to view a rule as a time object, and gives a generic rule model where each rule is recorded in terms of its content and statistics properties along with the time stamp of the mining session in which the rule is produced. In the follow-up papers, the works of [27-29] monitor statistics properties of a rule at different time points using the generic rule model. They further give some heuristics to detect interesting or abnormal changes about the discovered rule. One heuristic, for example, is to partition the range of values in the statistical property under observation into consecutive intervals and raises alerts when the value observed in an interval shifts to another interval. Other heuristics include significant test, corridor and occurrence based grouping heuristics. The basic idea is that concept drift as the initiator of pattern change often manifests itself gradually over a long time period where each of the changes may not be significant at all. Therefore the authors use different heuristics to take different aspects of pattern stability into account. For

example, the occurrence based grouping heuristic identifies the changes to the frequency of pattern appearance, while the corridor-based heuristic identifies the changes that differ from past values.

B. Liu et al. [30] also studies the temporal aspect of an association rule over time, but it focuses on discovering the overall trends of the rule rather than abnormal changes of the rule. It uses statistical methods to analyze interestingness of an association rule from temporal dimension, and classifies the rule into a stable rule, rule that exhibits increasing or decreasing trend and semi-stable rule. It employs Chi-square test to check whether the confidence (or support) of a rule over time is homogeneous. If it is homogeneous, the rule is classified as a stable rule. For an unstable rule, the authors use Run test to test whether the confidence or support of the rule exhibits trend.

In X. Chen et al. [31], the authors propose to identify two temporal features with the interesting rules. The motivation is that in real-world applications, the discovered knowledge is often time varying and people who expect to use the discovered knowledge may not know when it became valid, whether it is still valid at present, or if it will be valid sometime in the future. Therefore the paper focuses on mining two temporal features of some known association rules. The first one is to find all interesting contiguous intervals during which a specific association rule holds. And the second one is to find all interesting periodicities that a specific association rule has.

G. Dong et al. [32] finds the support differences of itemsets mined from two consecutive datasets and uses the differences to detect the emerging patterns (EP). In the paper, EPs are defined as itemsets whose supports increase significantly from one dataset to another. Because useful Apriori property no longer holds for EPs and there are usually too many candidates, the paper proposes the description of large collections of itemsets using their concise borders and design mining algorithms which manipulate only the borders of the collections to find EPs. Our work differs from this in that we analyze the relationships among rules over time rather than focus on emerging itemsets between two time points.

In summary, the works of [26-32] mine association rules in different time periods and investigate the behavior of the rule over time. The works of [26-29] detect interesting or abnormal changes about the discovered rule, the works of [30-31] discover the overall trend or pattern of the rule over time, and the work of [32] focus on the change of patterns in two consecutive datasets. However, all these works only consider the dynamic behavior of a single rule or pattern over time. To date, no work has been done to discover the relationships among the changes of the rules over time. We think in many cases the changes of the rules are correlated. Such correlations reflect the change of the underlying data. Therefore they may give the domain user a better idea about how and why the data changes. This is the main motivation of our work. In this thesis, we define some evolution relationships among rules over time and propose the corresponding approaches to find the relationships.

Chapter 3

Preliminary Definitions

In this chapter, we give some preliminary definitions used in this work before we introduce the details of the proposed approaches in Chapter 4. First, we define four types of rules according to their dynamic behavior over time. Second, we define four categories of evolution relationships among rules based on the correlations of their confidences.

3.1 Dynamic Behavior of a Rule

As mentioned in Chapter 1, we analyze the dynamic behavior of the rules and the correlations in their statistical properties. A rule's dynamic behavior is referred to as the changes in its statistical properties, i.e. confidence or support, over time. We

model a rule's confidence over time as a time series, denoted as $\{y_1, y_2, \ldots, y_n\}$. First, we introduce the terminology used in this thesis.

Definition 3.1.1 (Strict Monotonic Series): Given a time series $\{y_1, y_2, \ldots, y_n\}$. We say the time series is a strict monotonic series if

1) $y_i - y_{i+1} > 0 \ \forall \ i \in [1, n-1]$ (monotonic decreasing) or

2) $y_i - y_{i+1} < 0 \ \forall \ i \in [1, n-1]$ (monotonic increasing)

Definition 3.1.2 (Constant Series): Given a time series $\{y_1, y_2, \ldots, y_n\}$. We say the time series is constant if $y_i - y_{i+1} = 0 \ \forall \ i \in [1, n-1]$.

Definition 3.1.3 (Inconsistent Sub-Series): Given a time series $\{y_1, y_2, \ldots, y_n\}$, we say $\{y_i, \ldots, y_j\}$, $1 \leq i < j \leq n$, is an inconsistent sub-series in $\{y_1, y_2, \ldots, y_n\}$ if by removing $\{y_i, \ldots, y_j\}$, we can obtain the time series $\{y_1, \ldots, y_{i-1}, y_{j+1}, \ldots, y_n\}$ such that it is either a strict monotonic or constant series.

Definition 3.1.4 (Trend Fragment): Suppose $T = \{y_1, y_2, \ldots, y_n\}$ is a time series with k inconsistent sub-series $S_1, S_2, \ldots, S_k$. $|S_i|$ denotes the number of time points in sub-series S_i. T is said to be a trend fragment if

1) $|S_i| < \text{max\_inconsistentLen}, \ 1 \leq i \leq k$;

2) $n - \sum_i |S_i| > \text{min\_fragmentLen}$

where min_fragmentLen and max_inconsistentLen are the user-specified parameters denoting the minimum length of the trend fragment and the maximum length of inconsistent series.

A trend fragment is said to be stable/increasing/decreasing if the resultant series, after removing the inconsistent sub-series, is constant/monotonic increasing/monotonic decreasing.

Example 3.1.1

Suppose we are given the confidence values of a rule over 18 time points, CS = {0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.48, 0.6, 0.8, 0.8, 0.8, 0.8, 0.75, 0.68, 0.8, 0.8, 0.8, 0.8} with the user-specified parameters min_fragmentLen = 10 and max_inconsistentLen = 3.

Then, the sub-series S_1 = {0.48, 0.6} and S_2 = {0.75, 0.68} are inconsistent sub-series. Here, $|CS| = 18$, $|S_1| = 2 <$ max_inconsistentLen, $|S_2| = 2 <$ max_inconsistentLen, $18 - (|S_1| + |S_2|) = 18 - 4 = 14 >$ min_fragmentLen. We say CS is a stable trend fragment.

Based on the definition of stable/increasing/decreasing trend fragments, we classify a rule into the following categories:

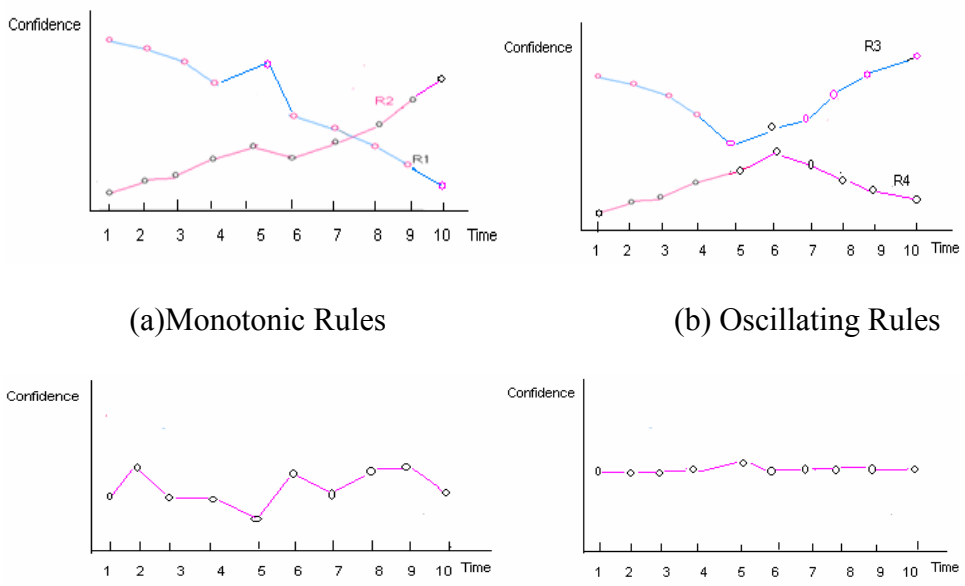
Definition 3.1.5 (Stable Rule): A rule r with confidence series CS is said to be a stable rule if CS is a stable trend fragment.

Definition 3.1.6 (Monotonic Rule): A rule r with confidence series CS is said to be a monotonic increasing/decreasing rule if CS is an increasing/decreasing trend fragment.

Definition 3.1.7 (Oscillating Rule): A rule r with confidence series CS is an oscillating rule if CS has more than one trend fragment or CS has only one trend fragment which is the sub-series of CS.

Definition 3.1.8 (Irregular Rule): A rule r with confidence series CS is an irregular rule if CS has no trend fragment.

Figure 3.1 illustrates the four different types of rules. Suppose min_fragmentLen = 5 and max_inconsistentLen = 2. The rules in Figure 3.1(a) are monotonic rules as their confidence series are increasing or decreasing trend fragments. The rules in Figure 3.1(b) are oscillating rules. There are two trend fragments in both rules. The confidence sub-series from time point 1 to 5 of R3 is a



(a)Monotonic Rules (b) Oscillating Rules

(c) Irregular Rule (d) Stable Rule

Figure 3.1 Rule Categories

decreasing trend fragment and the confidence sub-series from time point 5 to 10 is an increasing trend fragment. There is no trend fragment of the rule in Figure 3.1(c), so

it is an irregular rule. The confidence series of the rule in Figure 3.1(d) is a stable trend fragment, so the rule is a stable rule.

A stable rule is more reliable, so it can be used in real-world tasks. A monotonic rule has a systematic trend in the whole time period therefore is predictive. The confidence of an oscillating rule may increase in some time periods, and may decrease or stay unchanged in other time periods. An irregular rule is neither predictive nor reliable, so it may not be much useful in real-world applications.

In this thesis, we call a monotonic rule or stable rule a **trend rule** as it has a systematic trend in its entire confidence series, either increasing, decreasing or stable.

3.2 Evolution Relationships Among Rules

Besides analyzing the dynamic behavior of each association rule, we also wish to find the relationships among rules over time. These relationships are also called evolution relationships. They are based on the confidence correlations among rules. Here, to measure the confidence correlation, we use the Pearson correlation coefficient which is defined as follows [33]:

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}} \quad (2)$$

where X and Y are the vectors of the two confidence series and E is the expected value operator.

Our relationships are defined among the rules with the same consequent C. Suppose we have three rules: R_1: $\alpha \Rightarrow C$, R_2: $\beta \Rightarrow C$, R_3: $\gamma \Rightarrow C$ where C is the target value, $\alpha \cup \beta = \gamma$, $\alpha \not\subset \beta$ and $\beta \not\subset \alpha$. Let CS_1, CS_2, CS_3 be the confidence values of R_1, R_2, R_3 over the period $[t_1, t_2]$ in which CS_1, CS_2, CS_3 are trend fragments. ρ_{CS_1,CS_2} is the Pearson correlation coefficient between CS_1 and CS_2, and δ is a user-defined tolerance.

Definition 3.2.1 (Competing Relationship): Suppose CS_1 and CS_2 are monotonic trend fragments. We say $R_1 : \alpha \Rightarrow C$ and $R_2 : \beta \Rightarrow C$ ($\alpha \cap \beta = \varnothing$) have a competing relationship in [t1, t2] if $\rho_{CS_1,CS_2} < -1 + \delta$.

Competing relationship implies that the confidence of one rule increases as the confidence of the other rule decreases. It indicates that the antecedents of R1 and R2, i.e. α and β, are competing with each other over time in implying the consequent C.

Definition 3.2.2 (Diverging Relationship): Suppose CS_1, CS_2 and CS_3 are monotonic trend fragments. We say $R_1 : \alpha \Rightarrow C$ and $R_2 : \beta \Rightarrow C$ have a diverging relationship with R_3: $\alpha \cup \beta \Rightarrow C$ in [t1, t2] if

1) $\rho_{CS_1,CS_2} > 1 - \delta$,

2) $\rho_{CS_1,CS_3} < -1 + \delta$, or $\rho_{CS_2,CS_3} < -1 + \delta$

Diverging relationship indicates that the combined effect of α and β on implying the consequent C is opposite to that of α or β individually.

Definition 3.2.3 (Enhancing Relationship): Suppose CS_1 and CS_3 are monotonic trend fragments while CS_2 is a constant trend fragments. We say $R_1 : \alpha \Rightarrow C$ and $R_2 : \beta \Rightarrow C$ have an enhancing relationship with R_3: $\alpha \cup \beta \Rightarrow C$ in [t1, t2] if

1) $\rho_{CS_1, CS_3} < -1 + \delta$

2) CS_1 is monotonic decreasing and CS_3 is monotonic increasing

Enhancing relationship implies that the condition β enhances the effect of α on the consequent C.

Definition 3.2.4 (Alleviating Relationship): Suppose CS_1 and CS_3 are monotonic series while CS_2 is a constant series. We say $R_1 : \alpha \Rightarrow C$ and $R_2 : \beta \Rightarrow C$ have an alleviating relationship with R_3: $\alpha \cup \beta \Rightarrow C$ in [t1, t2] if

1) $\rho_{CS_1, CS_3} < -1 + \delta$

2) CS_1 is monotonic increasing and CS_3 is monotonic decreasing

Alleviating relationship implies that the condition β alleviates the effect of α on the consequent C.

These relationships are unexpected and counter-intuitive therefore could be important and useful in real-world applications. For example, consider the scenario that which type of qualifications may increase the chance of finding a job, competing relationship may indicate that the persons with qualification α are more and more likely to get the position over time, compared to the persons with qualification β. Enhancing relationships may imply that a person who have both qualifications α and β at the same time is more and more likely to get the position, compared with

the past time when only having qualification α can make a person to get the position. This might indicate the change of standards used in human resources department.

Chapter 4

Proposed Approaches

In this chapter, we introduce our proposed approaches. The overview of our work is shown in Figure 4.1. We have three tasks. First, partition the original dataset by time period and mine association rules over multiple time points; second, analyze the dynamic behavior of each individual rule over time and classify the rule by its dynamic behavior; third, find the evolution relationships among rules.

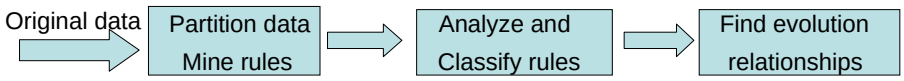


Figure 4.1: Work Overview

The following three sections give the details of our approaches.

4.1 Mine Association Rules over Time

To analyze the dynamic behavior of a rule and the relationships among rules over time, we first partition the available dataset into sub-datasets by year, month or day, depending on the applications. We then mine association rules from each sub-dataset and track the confidences of the rules over the different sub-datasets. One issue is immediately apparent: what happens if an association rule fails to meet the min_sup requirement in some sub-datasets but in other sub-datasets, the min_sup requirement is satisfied. This would imply that when we examine the time series of the confidence of this association rule, there will be missing confidences at those time points where the rule fails to satisfy the min_sup requirement. An association rule with too many missing confidences is said to be unstable. In this thesis, an unstable rule is one whose number of missing values exceeds the user defined maximum number of disappearance (max_disAppear). We filter these unstable rules from further considerations as they do not provide meaningful information in the evolution analysis process.

For those rules with only a few missing confidences, we perform additional database scans to compute the supports of the itemsets corresponding to these rules in the sub-datasets. With these supports, we can compute the missing confidences using the following formula [1, 2]:

$$Confidence(\alpha \Rightarrow C) = \frac{\sup(\alpha \cup \{C\})}{\sup(\alpha)} \qquad (3)$$

Where $\sup(\alpha \cup \{C\})$ and $\sup(\alpha)$ are the supports of $\alpha \cup \{C\}$ and α respectively. The procedure is summarized in Algorithm 4.1.1.

Algorithm 4.1.1 MineAssoRuleOverTime

Input: dataset in the whole time period, target value C

Output: association rules with its consequent as C over time

1. partition the dataset into sub-datasets by time period
2. mine association rules in each sub-dataset
3. for each rule r
4. If the number of missing confidences > max_disAppear
5. drop r
6. end if
7. end for
8. for each sub-dataset
9. for each of the remaining rules $\alpha \Rightarrow C$ which misses the confidence in this sub-dataset
10. put the itemsets α and $\alpha \cup \{C\}$ in I.
11. end for
12. scan the sub-dataset to get the supports of the itemsets in I
13. for each of the remaining rules $\alpha \Rightarrow C$ which misses the confidence in this sub-dataset
14. compute the missing confidence using $\sup(\alpha \cup \{C\})/\sup(\alpha)$
15. end for
16. end for

In Algorithm 4.1.1, line 1 partitions the dataset by time period and line 2 mines association rules in each sub-dataset. After that, lines 3-7 check the confidences of

the rules. If the number of missing confidences of a rule exceeds the max_disApppear, we drop the rule. For the remaining rules, lines 8-16 complete their missing confidences as follows. For each sub-dataset, lines 9-11 first collect the itemsets needed to compute the missing confidences. After that, line 12 scans the sub-dataset once to get the supports of the itemsets and lines 13-15 computes the missing confidences with the supports.

Another issue is the efficiency consideration of mining association rules in line 2. Traditionally, mining association rule is performed in two steps. The first step generates all the frequent itemsets in the dataset. The second step derives the association rules from the frequent itemsets. Generation of frequent itemsets is time consuming and there have been many algorithms proposed to mine the frequent itemsets efficiently such as Apriori [2] and FP-Growth [4]. In this thesis we make use of the constraint that the association rules we are interested in must have a target value, say C, as the consequent. This reduces the number of frequent itemsets generated as we only need to generate the frequent itemsets containing target value C. So we can reduce the time complexity of the frequent itemset generation as follows. First we partition the dataset into two parts, positive dataset (PD) and negative dataset (ND). PD consists of all instances with target value C. ND consists of all instances without target value C. To discover association rules with C as their consequents, we mine the frequent itemsets from PD, and count the frequencies of these itemsets in ND to compute the rules' confidences using the following formula.

$$confidence(\alpha => C) = \frac{\sup(\alpha \ in \ PD)}{\sup(\alpha \ in \ PD) + \sup(\alpha \ in \ ND)} \quad (4)$$

where α is a frequent itemset mined from PD, $\sup(\alpha \ in \ PD)$ is the support of α in PD and $\sup(\alpha \ in \ ND)$ is the support of α in ND. Note that Formula 4 is consistent to Formula 3 in that $\sup(\alpha \ in \ PD)$ is equal to $\sup(\alpha \cup \{C\})$ since every instance in PD contains target value C, and $\sup(\alpha \ in \ PD) + \sup(\alpha \ in \ ND)$ is equal to $\sup(\alpha)$ since both of them are the support of the instances that contain α in the whole dataset.

The algorithm is summarized in Algorithm 4.1.2. When size of PD is much smaller than that of the original dataset D, the resulting savings is substantial as compared to naively mining the association rules from the dataset directly.

Algorithm 4.1.2 MineAssoRule

Input: sub-dataset, target value C

Output: association rule with its consequent as C

1. partition the sub-dataset into two parts, PD and ND
2. mine the frequent itemsets from PD using FP-Growth algorithm. For each frequent itemset α, there will be a corresponding rule $\alpha \Rightarrow$ C
3. count each of the frequent itemsets in step 2 from ND
4. compute the confidence of each rule, using

$$confidence(\alpha => C) = \frac{\sup(\alpha \ in \ PD)}{\sup(\alpha \ in \ PD) + \sup(\alpha \ in \ ND)}$$

5. output rules whose confidences satisfy the min_conf

4.2 Dynamic Behavior of a Rule

Having mined all the association rules with target value C as the consequents, we proceed to analyze the dynamic behavior of these rules. Recall in Chapter 3, we have defined the concepts of stable, monotonic increasing, monotonic decreasing and irregular rules.

Given the confidence values of a rule over n time points $\{y_1,\ldots,y_n\}$, we scan the series from left to right, grouping the values into consistent sub-series such that all the values in each sub-series are either constant or monotonic increasing/decreasing (see Algorithm 4.2.1). Note that there are three fields in a consistent sub-series (CSS). A "begin" field is used to record the start point of the sub-series; An "end" field records the end point of the sub-series; and a "flag" indicates the trend of the sub-series, with value of -1 decreasing trend, value of 1 increasing trend, and value of 0 stable.

Algorithm 4.2.1 FindCSSs

```
Input: confidence series of a rule CS
Output: all consistent sub-series CSSArray
1.  if (CS[2]-CS[1] = = 0)
2.      initialFlag = 0
3.  else if (CS[2]-CS[1] >0)
4.      initialFlag = 1
5.  else
6.      initialFlag = –1
7.  end if
8.  k = 1, initialBegin = 1
9.  for i = 3 to |CS|
10.   if (CS[i]-CS[i-1] = = 0)
11.       newFlag = 0
```

```
12.    else if (CS[i]-CS[i-1] > 0)
13.        newFlag = 1
14.    else
15.        newFlag = -1
16.    end if
17.    if (newFlag ! = initialFlag)      // store the sub-series and find the next CSS
18.        CSSArray[k].begin = initialBegin
19.        CSSArray[k].end = i-1
20.        CSSArray[k].flag = initalFlag
21.        k = k+1
22.        initalFlag = newFlag
23.        initialBegin = i
24.    end if
25. end for
```

Example 4.1

Suppose the min_fragmentLen is 9 and max_inconsistentLen is 3. Figure 4.2 shows the confidence series of a rule over time. According to Algorithm 4.2.1, we find six consistent sub-series, namely CSS_1 = CS[1:3] (denoting the sub-series of confidence series from time point 1 to 3), CSS_2 = CS[4:5], CSS_3 = CS[6:9], CSS_4 = CS[10:14], CSS_5 = CS[15:16] and CSS_6 = CS[17:20].

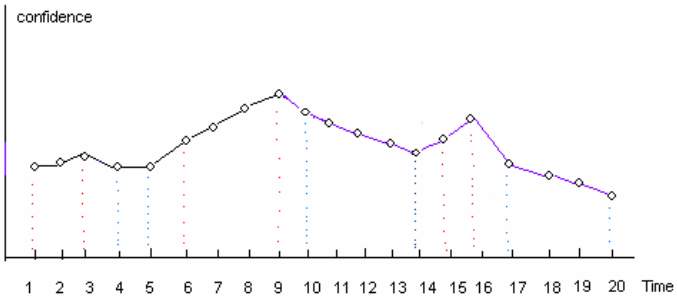


Figure 4.2 Example of Finding Trend Fragment

After all the sub-series have been formed, we proceed to merge the adjacent sub-series if the gap between the two series is less than max_inconsistentLen and the merged series is strictly monotonic or constant. The merged sub-series whose lengths are greater than min_fragmentLen are identified as trend fragments (see Algorithm 4.2.2 for details). Back to Example 4.1, CSS_1 and CSS_3 are merged as CS[1:9], CSS_4 and CSS_6 are merged as CS[10:20]. Since both the merged sub-series CS[1:9] and CS[10:20] are longer than 9, they are both trend fragments.

After all the trend fragments are found, we classify a rule based on the number of its trend fragments. If the number of trend fragment is zero (this implies that the confidences of the rule vary greatly with no specific trend), we classify the rule as an irregular rule. If the number of trend fragment is one, we classify the rule as a trend rule. Rules that do not fall into the above categories are classified as oscillating rules which means that their confidences may increase in some time periods, and decrease or remain stable in other time periods. Details of the steps can be found in Algorithm 4.2.2. Note that Algorithm 4.2.2 calls Function 4.2.1 which returns a value indicating whether two sub-series should be merged.

Algorithm 4.2.2 MergeCSSAndClassifyRules

Input: a rule's confidence series, CS
its consistent sub-series, CSSArray
Output: the rule's trend fragments, TFArray
the category of the rule, CR

```
1.  k = 1, mergedCSS = CSSArray[1]
2.  for i = 2 to |CSSArray|
3.      if (isMergeable(CS,CSSArray[i-1],CSSArray[i])
```

```
4.          mergedCSS.end = CSSArray[i].end
5.      else
6.          if (|mergedCSS| ≥ min_fragmentLen) // if yes, it is a trend fragment
7.              TFArray[k] = mergedCSS
8.              k = k+1
9.          end if
10.         mergedCSS = CSSArray[i]          // start to find a new merged sub-series
11.     end if
12. end for
13. if (|TFArray| = = 0)                     // classify the rule
14.     CR = irregular rule
15. else if ((|TFArray| = = 1)
16.     CR = trend rule (monotonic or stable)
17. else
18.     CR = oscillating rule
19. end if
```

In Algorithm 4.2.2, lines 2-12 merge adjacent consistent sub-series from left to right and find the trend fragments of the rule. In each iteration, we first check whether current sub-series should be merged with the previous one; if they can be merged, we merge the current sub-series and continue to check the next sub-series (lines 3-4); otherwise, we check whether the merged sub-series is a trend fragment and start to find another merged sub-series (lines 5-10). Lines 13-19 classify the rule based on the number of the trend fragments.

Function 4.2.1 isMergeable

input: a confidence series CS; its two consistent sub-series, CSS_i and CSS_j
output: a value indicating whether the two sub-series should be merged

```
1.  if (CSS_j.begin – CSS_i.end > max_inconsistentLen)
2.      return false
3.  end if
4.  result = false
5.  if(CSS_i.flag = =0)          // case 1: both sub-series stable
6.      if(CSS_j.flag = = 0)
7.          if(CS[CSS_i.end] = = CS[CSS_j.begin])
```

```
8.              result = true
9.          end if
10.     end if
11. else if (CSS_i.flag = = 1)    // case 2: both sub-series increasing
12.     if(CSS_j.flag = = 1)
13.         if(CS[CSS_i.end] < CS[CSS_j.begin])
14.             result = true
15.         end if
16.     end if
17.  else                         // case 3: both sub-series decreasing
18.     if(CSS_j.flag = =-1)
19.         if(CS[CSS_i.end] > CS[CSS_j.begin])
20.             result = true
21.         end if
22.     end if
23.  end if
24.  return result
```

In Function 4.2.1, lines 1-3 check whether the gap between the two sub-series is greater that max_inconsistentLen. If it is, the two sub-series cannot be merged and we return false. Lines 4-24 check whether the merge of the two sub-series is strictly monotonic or constant series. If it is, the two sub-series can be merged and the function returns true.

4.3 Find Evolution Relationships Among Rules

In this section, we introduce the approaches to find relationships among trend rules and oscillating rules. First we define the notion of a combined rule and sub-rule as follows:

Defintion 4.3.1 (Combined Rule): Suppose we have three rules $r_i : \alpha \Rightarrow$ C, $r_j : \beta \Rightarrow$ C , $r_k : \gamma \Rightarrow$ C. If $\alpha \cup \beta = \gamma$, $\alpha \not\subset \beta$ and $\beta \not\subset \alpha$, we say r_k is the combined rule of r_i and r_j.

Definition 4.3.2 (Sub-Rule): Given two rules $r_i : \alpha \Rightarrow$ C, $r_k : \gamma \Rightarrow$ C. If $\alpha \subset \gamma$, we say r_i is a sub-rule of r_k.

4.3.1 Find Combined Rules

From the definitions of diverging, enhancing and alleviating relationships discussed in Chapter 3, it is evident that we need to analyze the confidence correlations between a combined rule and its sub-rules. Repeated scanning of the rules to find the corresponding combined rule is inefficient and time consuming. Hence, in this thesis, we design a hash table structure that captures the implicit relationships between a combined rule and its sub-rules.

For each rule r of the form $a_1, a_2, \ldots, a_m \Rightarrow$ C, where $a_1, a_2, \ldots, a_m$ are the unique integer identifiers of the items, we add up these unique identifiers to form a hash key. A hash function is then applied to this key to obtain the location of the rule r. In this way, the rules are stored in a hash table indexed by the antecedents of the rules. The procedure is summarized in Algorithm 4.3.1.

Algorithm 4.3.1 StoreRuleUsingHash

Input: $a_1,a_2,\ldots,a_m \Rightarrow$ C where $a_1,a_2,\ldots,a_m$ are unique integer identifiers of items; number of buckets: Num

Output: bucket number: BNo
1. hashKey = $a_1 + a_2 + \ldots + a_m$
2. BNo = hashKey%Num
3. return BNo

Back to our running example, suppose the integer identifiers of the items are tabulated in Table 4.1 and the number of buckets is 20. Some of the rules in Table 1.2 are stored into the hash structure as shown in Table 4.2.

Item	Identifier
beer	101
chip	102
cake	103
toothbrush A	104
toothbrush B	105
toothpaste C	106

Table 4.1 Identifiers of Items

Bucket No	Rules
1	beer $\Rightarrow$ toothpaste C
2	chip $\Rightarrow$ toothpaste C
3	cake $\Rightarrow$ toothpaste C
4	toothbrush A $\Rightarrow$ toothpaste C
5	toothbrush B $\Rightarrow$ toothpaste C
6	
7	
8	
9	toothbrush A, toothbrush B $\Rightarrow$ toothpaste C
…	…

Table 4.2 Hash Table of Rules

With the hash structure, given any two rules, we can simply union and add the antecedents of the two rules to form a hash key that is used to access the location of

the combined rule. For example, given the rules "toothbrush A ⇒ toothpaste C" and "toothbrush B ⇒ toothpaste C", we add toothbrush A(104) and toothbrush B(105) to form a hashKey: 104+105 = 209, then we can get the bucketNo: 209%20 = 9. After that, we use the bucketNo to locate the combined rule "toothbrush A, toothbrush B ⇒ toothpaste C".

4.3.2 Find Relationships Among Trend Rules

In this section, we discuss how to discover interesting relationships among trend rules. Recall, a trend rule is one that exhibits a singular behavior over the whole time period. Hence, there is only one trend fragment associated with each trend rule. For such rules, we apply the definitions in Chapter 3 to find the relationships among each pair of rules. Algorithm 4.3.2 gives the details, where δ is the user-defined tolerance.

Algorithm 4.3.2 FindRelInTrendRules

Input: all trend rules

user-defined tolerance δ

Output: the relationships among trend rules

1. for each pair of trend rules r_i, r_j
2. if (both r_i and r_j are not stable) // **case 1**: r_i not stable, r_j not stable
3. corr = calculateCorrelation (r_i, r_j)
4. if (corr < -1 + δ)
5. if (r_i and r_j have no common items in the antecedent)
6. output: competing relationship(r_i, r_j)
7. end if
8. else if (corr > 1 - δ)

```
9.              if( r_k , the combined rule of r_i and r_j , is a trend rule and is not stable)
10.                 corr = calculateCorrelation ( r_k , r_i )
11.                 if(corr < -1 + δ)
12.                     output: diverging relationship( r_k , r_i , r_j )
13.                 end if
14.             end if
15.         else ;
16.         end if
17.     else if ( r_i is not stable and r_j is stable)   // case 2: r_i not stable, r_j stable
18.         if ( r_k , the combined rule of r_i and r_j , is a trend rule and is not stable)
19.             corr = calculateCorrelation( r_k , r_i )
20.             if(corr <-1 + δ)
21.                 if( r_k is increasing)   output: enhancing relationship( r_k , r_i , r_j )
22.                 else                     output: alleviating relationship( r_k , r_i , r_j )
23.                 end if
24.             end if
25.         end if
26.     else if( r_i is stable and r_j is not stable)   // case 3: r_i stable, r_j not stable
27.         /* similar process as 18-25 */
28.     else ;                                          // case 4: r_i stable, r_j stable
29.     end if
30. end for
```

In Algorithm 4.3.2, calculateCorrelation (r_i, r_j) computes the Pearson correlation coefficient between the confidence series of r_i and r_j. The algorithm performs pairwise comparisons of the rules. If both rules are not stable, we compute the Pearson correlation coefficient (line 3); if the correlation is less than -1 + δ (close to -1) and they have no common items in the antecedent, we say the two rules exhibit a competing relationship (lines 4-7). If the correlation is greater than 1- δ (close to 1), we obtain their combined rule to determine whether the rules exhibit a diverging

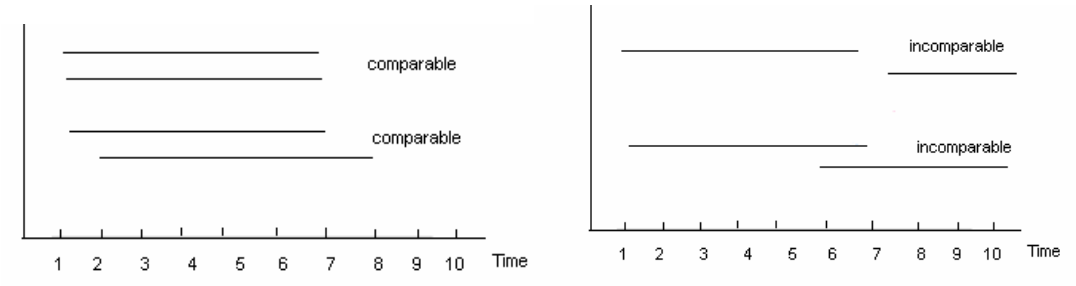
relationship (lines 8-14). If one of the rules is stable, we obtain their combined rule to determine whether the rules exhibit an enhancing or alleviating relationship (lines 17-27). If both rules are stable, there is no evolution relationship (line 28).

4.3.3 Find Relationships Among Oscillating Rules

Finding relationships among oscillating rules is more complex than finding relationships among trend rules. This is because the oscillating rules may increase in some time periods and decrease or remain stable in other time periods. The algorithm must automatically discover the overlapped time intervals of the trend fragments in which the oscillating rules exhibit trends and have the different types of relationships as discussed in Chapter 3. If the overlapped interval is too short, it is not significant. Hence, in this thesis we find the relationships only in the trend fragments that have a significant overlapped interval.

In general, we say a trend fragment is comparable to another trend fragment if they have a significant overlapped interval. More precisely, Suppose we have a trend fragment $TF_i = \{ y_{n_1}, y_{n_1+1}, \ldots y_{n_2} \}$. We say trend fragment $TF_j = \{ y_{m_1}, y_{m_1+1}, \ldots y_{m_2} \}$, where $n_1 \leq m_1$, is comparable to TF_i if $n_1 = m_1$ and $n_2 = m_2$; or $(\min(n_2,m_2) - m_1)/(\max(n_2,m_2) - \min(n_1,m_1)) > \text{min\_ratio}$, where min_ratio is the user-specified minimum ratio. Here, TF_i is called the seed fragment. In other words, a fragment is comparable to the seed fragment if the proportion of the overlap between the two fragments is greater than a user-specified ratio. Suppose min_ratio

is 0.7. Figure 4.3(a) shows examples of trend fragments that are comparable; and Figure 4.3(b) shows examples of trend fragments that are not comparable.



(a) comparable (b) incomparable

Figure 4.3: Example of Comparable and Incomparable Fragments

By the definition of comparable trend fragments, the task of finding relationships among oscillating rules is to find the relationships among rules in the overlapped time intervals of their comparable trend fragments.

A naïve approach is to perform pairwise comparisons of the rules and confine the computation of the correlation to the overlapped region of the comparable trend fragments in each pair of rules. Details are given in Algorithm 4.3.3 and Algorithm 4.3.5. Note that Algorithm 4.3.3 finds diverging, alleviating and enhancing relationships. Algorithm 4.3.5 finds competing relationship. The pseudocodes of findCombinedRel (f_i, f_j, f_k), findSeed(f_i, f_j, f_k) and isComparable(f_i, f_j) in Algorithm 4.3.3 are given in Algorithm 4.3.4, Function 4.3.1 and Function 4.3.2.

Algorithm 4.3.3 FindRelInOsciRules

Input: all oscillating rules

Output: the diverging, alleviating and enhancing relationships among rules

1. for each pair rules r_i and r_j
2. find the combined rule, r_k
3. if (r_k exists)
4. TFS_i = trend fragments of r_i ,TFS_j = trend fragments of r_j ,
   TFS_k = trend fragments of r_k , m = 0,n = 0, l = 0
5. while m < |TFS_i| and n < |TFS_j| and l < |TFS_k|
6. f_i = TFS_i[m], f_j = TFS_j[n], f_k = TFS_k[l]
7. seed = findSeed(f_i, f_j, f_k)
8. if(seed = = 1)
9. if(isComparable(f_i, f_j) and isComparable(f_i, f_k))
10. findCombinedRel (f_i, f_j, f_k) // call Algorithm 4.3.4
11. m = m+1
12. if(f_j .begin = = f_i .begin and f_j .end = = f_i .end)
13. n = n+1
14. if(f_k .begin = = f_i .begin and f_k .end = = f_i .end)
15. l = l+1
16. else if (seed = = 2)
17. /* similar process as seed = = 1, this time the seed fragment is f_j */
18. else
19. /* similar process as seed = = 1, this time the seed fragment is f_k */
20. end if
21. end while
22. end if
23. end for

Algorithm 4.3.3 works as follows. For each pair of rules, line 2 finds the combined rule of the two rules using Algorithm 4.3.1. If the combined rule exists, we find relationships among the combined rule and the sub-rules in each pair of comparable trend fragments. The finding proceeds in a left-to-right order. We view the trend fragments of the three rules as three queues individually, and scan them

from left to right until one of the rules runs out of its fragments (lines 5-21). In each iteration, findSeed(f_i, f_j, f_k) choose a seed fragment from the fragments of the three rules (line 7). The seed fragment to be chosen is the fragment that has the smallest start point and end point. After choosing the seed fragment, we check whether the other two fragments are comparable to the seed fragment (line 9). If they are comparable, we find relationships among them using Algorithm 4.3.4 (line 10). At the end of the iteration, the seed fragment is dropped (line 11), and the other two fragments are discarded if they have the same start point and end point as the seed fragment (lines 12-15).

Function 4.3.1 FindSeed

Input: three fragments f_i, f_j, f_k
Output: the seed fragment
1. sort f_i, f_j, f_k by their start point and end point in ascending order
2. if(f_i is the first fragment)
3. return 1
4. else if(f_j is the first fragment)
5. return 2
6. else
7. return 3

Function 4.3.2 IsComparable

Input: seed fragment: f_i, a fragment: f_j, user-defined min_ratio
Output: a value indicating whether f_j is comparable to f_i
1. overlapLen = min(f_i.end, f_j.end) – f_j.begin
2. wholeLen = max(f_i.end, f_j.end) – min(f_i.begin, f_j.begin)
3. if(overlapLen / wholeLen $\geqslant$ min_ratio)

```
4.     return true
5.  else
6.     return false
7.  end if
```

In Algorithm 4.3.3, the approach findCombinedRel(f_i , f_j , f_k) finds the diverging, alleviating and enhancing relationships of rules in the pair of three comparable trend fragments. It is summarized in Algorithm 4.3.4. Algorithm 4.3.4 is similar to Algorithm 4.3.2. The difference is that we need to compute the overlapped region (lines 1-2) and output the relationships in the overlapped region.

Algorithm 4.3.4 FindCombinedRel

Input: three fragments f_i, f_j, f_k, where f_k is the fragment of the combined rule, f_j and f_i are the fragments of the sub-rules

Output: the relationship among the rules of f_i, f_j, f_k

```
1.  begin = max( f_i.begin,  f_j.begin,  f_k.begin)
2.  end = min( f_i.end,  f_j.end,  f_k.end)
3.  r_i = the rule of f_i,  r_j = the rule of f_j,  r_k = the rule of f_k
4.  if ( both f_i and f_j are not stable)   // case 1: f_i not stable, f_j not stable
5.     corr = calculateCorrelation ( r_i , r_j , begin, end)
6.     if (corr > 1 - δ)
7.        corr = calculateCorrelation ( r_i , r_k , begin, end)
8.        if(corr < -1 + δ)
9.           output: diverging relationship( r_k , r_i , r_j , begin, end)
10.       end if
11.    end if
12. else if ( f_i is not stable and f_j is stable)   // case 2: f_i not stable, f_j stable
13.    corr = calculateCorrelation( r_i , r_k , begin, end)
14.    if(corr < -1 + δ)
15.       if( f_k is increasing)   output: enhancing relationship( r_k , r_i , r_j , begin, end)
16.       else                     output: alleviating relationship( r_k , r_i , r_j , begin, end)
```

17. end if
18. else if(f_i is stable and f_j is not stable) // **case 3**: f_i stable, f_j not stable
19. /* similar process as 12-17 */
20. else ; // **case 4**: f_i stable, f_j stable
21. end if

The algorithm to find competing relationships among oscillating rules is summarized in Algorithm 4.3.5. Similar to Algorithm 4.3.3, to find competing relationship, Algorithm 4.3.5 views the fragments of the two rules as two queues and proceeds in a left-to-right order (lines 4-28). In each iteration, if the two fragments have the same start point and end point, it is comparable (lines 7-8). Otherwise we choose the fragment that have the smaller start point and end point as the seed fragment and check whether the other fragment is comparable to it (lines 11-18). If they are comparable, find competing relationship in their overlapped region (lines 20-27). In each iteration, the seed fragment and the fragment that has the same start point and end point as the seed fragment are dropped (lines 9,10,14,18).

Algorithm 4.3.5 FindComRel

Input: all oscillating rules
Output: the competing relationship among rules
1. for each pair of rules r_i and r_j
2. TFS_i = trend fragments of r_i, TFS_j = trend fragments of r_j
3. m = 0,n = 0
4. while m < |TFS_i| and n < |TFS_j|
5. f_i = TFS_i [m], f_j = TFS_j [n]
6. flag = 0;
7. if(f_i.begin = = f_j.begin and f_i.end = = f_j.end)
8. flag = 1
9. m = m+1

```
10.            n = n+1
11.        else if( f_i.begin < f_j.begin or  f_i.begin = =  f_j.begin and  f_i.end <
                   f_j.end)
12.            if(isComparable( f_i, f_j )
13.               flag = 1
14.               m = m+1
15.        else
16.            if(isComparable( f_j, f_i )
17.               flag = 1
18.               n = n+1
19.        end if
20.        if(flag = = 1)
21.            begin = max( f_i.begin,  f_j.begin)
22.            end = min( f_i.end,  f_j.end)
23.            corr = calculateCorrelation( f_i, f_j ,begin,end)
24.            if(corr < -1 + δ)
25.               if( f_i  and  f_j  have no common items in the antecedent)
26.                  output: competing relationship( r_i , r_j ,begin,end)
27.        end if
28.      end while
29.   end for
```

Note that with the naive approach, all the rules are compared even when they do not have any comparable trend fragments. This observation leads to our optimized algorithm. Instead of focusing on the rules, we first examine all the trend fragments and group the trend fragments if they are comparable.

The grouping of trend fragments proceeds in a left-to-right order. First, the fragments are sorted by their start points in increasing order. Fragments that have the same start points but different end points are sorted by their end points in ascending order. After sorting, we start with the fragment with the smallest start point as a seed fragment and check whether the adjacent fragment is comparable to the seed

fragment. If it is comparable to the seed fragment, we place it in the group of the seed fragment and continue to find all the other comparable fragments; After all the comparable fragments of the seed fragment are found, we choose the next seed fragment and repeat the process to find another group of comparable fragments. Here the next seed fragment is the fragment that follows the current seed fragment and does not have the same start point and end point as the current seed fragment. Details are given in Algorithm 4.3.6.

Algorithm 4.3.6 FindComparableGroups

```
Input: trend fragments of the oscillating rules, TFs
Output: groups of comparable fragments, G
1.  sort the fragments in TFs by their start points and end points in
     increasing order                                          // left-to-right
2.  k = 1, i = 1
3.  while (i ≤ |TFs|)
4.        count = 0;
5.        for j = i to |TFs|
6.           if (TFs[j].begin > TFs[i].end)      // no overlap anymore
7.              break
8.           else if(TFs[j].begin = = TFs[i].begin and TFs[j].end = = TFs[i].end)
9.              put TFs [j] into G[k]
10.             count = count+1
11.          else if(isComparable(TFs[j],TFs[i])
12.             put TFs [j] into G[k]
13.          else ;
14.          end if
15.       end for
16.       i = i+count-1
17.       k = k+1
18.  end while
```

In Algorithm 4.3.6, line 1 sorts the fragments by their start points and end points in ascending order. After that, lines 3-18 find comparable groups from left to right.

In each iteration, a seed fragment is chosen (fragment i). All the adjacent fragments that are comparable to the seed fragment are added into its comparable group (lines 8-12). At the end of each iteration, the current seed fragment and all the fragments which have the same start point and end point as the seed fragment are dropped (line 16). We then continue to the next seed fragment and repeat the process.

Once all the groups of comparable trend fragments are found, we find the relationships only among the oscillating rules whose trend fragments are in the same comparable group. This strategy allows us to skip comparisons among rules that do not have any comparable trend fragments. Note that according to Algorithm 4.3.6, one fragment may belong to more than one group corresponding to different seed fragments. To avoid repeated comparisons, we further partition a comparable group G into G1 and G2. G1 includes the fragments that have the same start point and end point as the seed fragment. The remaining fragments in G are placed in G2. To find the relationships of rules among comparable trend fragments in G, we only perform pairwise comparisons within G1, and between G1 and G2. In other words, we skip the pairwise comparisons within G2. This is because the fragments in G2 will appear in the next group(s), and the pairwise comparisons among fragments in G2 can be done in the next group(s). Therefore there is no need to do the comparisons in current group. Algorithm 4.3.7 gives the details.

Algorithm 4.3.7 FindRelInGroup

Input: a group of comparable trend fragments, G

Output: the relationships of rules in the group

1. G1 = fragments in G that have the same start point and end point as the seed fragment; G2 = G – G1
2. for i = 1 to |G1|
3. r_i = the rule of G1[i]
4. for j = i+1 to |G1|
5. findCompRelInFrag(G1[i], G1[j]) // find competing relationship
6. r_j = the rule of G1[j]
7. r_k = the combined rule of r_i and r_j
8. f_k = the fragment of r_k
9. if(f_k is not stable)
10. findCombinedRel(G1[i], G1[j], f_k) // find diverging,…,relationships
11. end if
12. end for
13. for j = 1 to |G2|
14. findCompRelInFrag(G1[i], G2[j]) // find competing relationship
15. r_j = the rule of G2[j]
16. if (r_j is the sub-rule of r_i)
17. find the other sub-rule r_k such that r_i is the combined rule of r_k and r_j
18. f_k = the fragment of r_k
19. findCombinedRel(f_k ,G2[j],G1[i]) // find diverging,…,relationships
20. else
21. /* similar process as 6-11 */
22. end if
23. end for
24. end for

In Algorithm 4.3.7, lines 3-12 find relationships among the fragments within G1 and lines 13-23 find relationships among the fragments between G1 and G2. Note that findCompRelInFrag(G1[i], G1[j]) finds the competing relationship using Function 4.3.3, and findCombinedRel(G1[i], G1[j], f_k) finds the diverging, alleviating and enhancing relationships using Algorithm 4.3.4.

Function 4.3.3 FindCompRelInFrag

Input: two trend fragments f_i and f_j
user-defined tolerance δ
Output: the competing relationship between the two rules of f_i and f_j

1. begin = larger(f_i.begin, f_j.begin)
2. end = smaller(f_i.end, f_j.end)
3. corr = calculateCorrelation(f_i, f_j,begin,end)
4. if(corr < -1 + δ)
5. if(f_i and f_j have no common items in the antecedent)
6. output: competing relationship(f_i's rule, f_j's rule, begin, end)
7. end if
8. end if

In summary, to find relationships among oscillating rules, we focus on fragments. We first find groups of comparable trend fragments. Comparisons are done only among the fragments within each comparable group. In this way, we skip those rules that do not have any comparable trend fragments. In each group of comparable trend fragments, we further partition the fragments into sub-groups to avoid redundant comparisons. We call this method Group Based method of Finding relationships (GBF), and we call the naïve method Rule Based method of Finding relationships (RBF).

Chapter 5

Experiments

In this chapter, we carry out experiments to evaluate the proposed approaches on both synthetic and real-world datasets. All our approaches are implemented in C++. The experiments are run on a PC with 2.33 GHZ CPU and 3.25 GB RAM, running Windows XP.

5.1 Synthetic Data Generator

We design a synthetic data generator by extending the data generator in R. Agrawal et al. [2] to incorporate time and class information. The data generation includes two steps.

In the first step, we create a table of potential frequent itemsets. The size of each itemset is generated from a Poisson distribution with mean equal to parameter *I*. The

items in each itemset are randomly chosen from a set of *N* different items. Next, we generate M combined itemsets. Each combined itemset is generated by randomly selecting and combining two potential frequent itemsets. For each generated itemset, we assign a confidence value *c* which determines the probability that the itemset will appear in the transaction having target value C. The confidence value c is given by the following formula.

$$c = \begin{cases} r \quad if \ 0 \le r \le 1; \\ 0 \ \ if \ r < 0; \\ 1 \ \ if \ r > 1, \end{cases} \tag{5}$$

where *r* is a normal-distributed random number with mean = 0.5 and deviation = 0.1.

Each itemset is associated with two arrays which capture how the confidence c changes over time. The first array stores the change rates. Each change rate is randomly chosen from a normal distribution. The second array stores the change flags where each flag indicates whether the confidence increase, decrease, or remain unchanged for the corresponding time point. We generate n fragments for each itemset, where n is a random number from 1 to the maximum number of fragments (maxFrag). Each fragment has several time points. The change flags of the itemset at different time points in the same fragment could be the same (increases, decreases, stay unchanged) or different. In this way, the itemset will have a trend in the fragment if the change flags are the same, or change randomly if the change flags are different.

In the second step, we generate a dataset for each time point by generating its transactions as follows. We change the confidence of each itemset based on its

change flag and change rate at the time point. The dataset consists of two sub-sets: *PD* (which consists of transactions with target value *C*) and *ND* (which consists of transactions without target value *C*). The transactions of *PD* and *ND* are generated as follows. The size of a transaction is chosen from a Poisson distribution with mean equal to *T*. The content of the transaction is generated as follows. We randomly choose an itemset from a series of itemsets generated in the first step. If the confidence of a selected itemset is *c*, we append it to the transaction of *PD* with probability *c*, or append it to the transaction of *ND* with probability 1 – *c*. When a transaction reaches its size, we proceed to generate the next transaction.

Table 5.1 summarizes the main parameters in the data generator as well as the default values used in our experiments.

Parameter	Description	Default Value
$\mid D \mid$	Number of transactions	100 000
perc	Percentage of positive transactions in D	1/2
T	Average size of the transactions	10
I	Average size of itemsets	2
N	Number of items	10 000
maxFrag	Maximum number of fragments in each itemset	10

Table 5.1: Parameters of Data Generator

5.2 Experiments on Mining Association Rule

In this section, we compare the performance of the proposed partition-based approach of mining association rule (Algorithm 4.1.2) with the naïve approach which

directly utilizes existing frequent itemset mining algorithms such as FP-Growth. We call our method Partition based Association rule Mining (PAM) and the naïve method Direct Association rule Mining (DAM). Figure 5.1 shows the execution time when the number of transactions in the dataset increases from 50 000 to 300 000 with the average size of the transactions $T = 10$. Figure 5.2 shows the execution time when T ranges from 5 to 30 and the number of transactions in the dataset is 100 000.

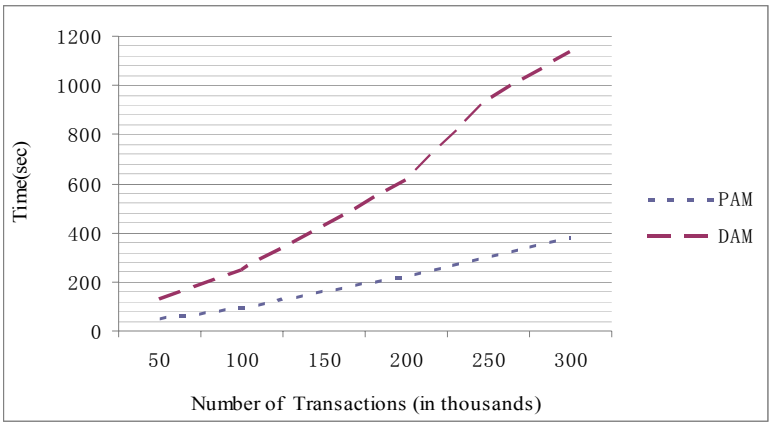


Figure 5.1: Running Time of Association Rule Mining

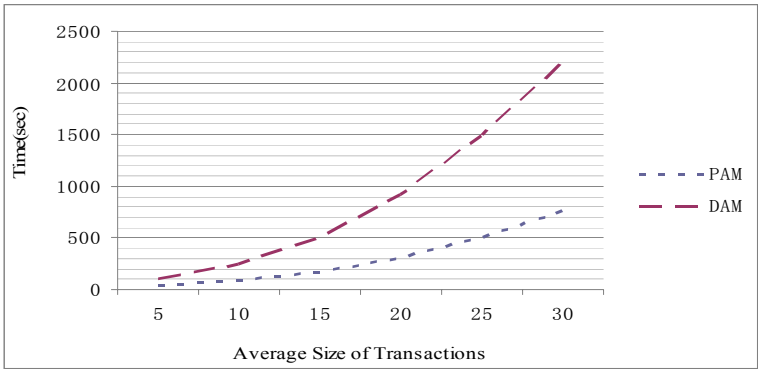


Figure 5.2: Running Time with Varying T

Both Figure 5.1 and Figure 5.2 show that our approach is better than the naïve approach. This is because the most time consuming part of association rule mining is the generation of frequent itemsets. In PAM we partition the dataset into positive sub-dataset and negative sub-dataset, and mine the frequent itemsets only in the positive sub-dataset; while in DAM we mine the frequent itemsets in the entire dataset, which may produce many redundant frequent itemsets.

We also evaluate the sensitivity of PAM and DAM to *perc* parameter. Figure 5.3 shows the running time of PAM and DAM when perc ranges from 1.0 to 0.1. We observe that PAM is better than DAM. As *perc* becomes smaller PAM becomes more efficient than DAM. The reason is that when *perc* is smaller we mine the frequent itemsets in a smaller positive sub-dataset while DAM still mine the frequent itemsets in the whole dataset.

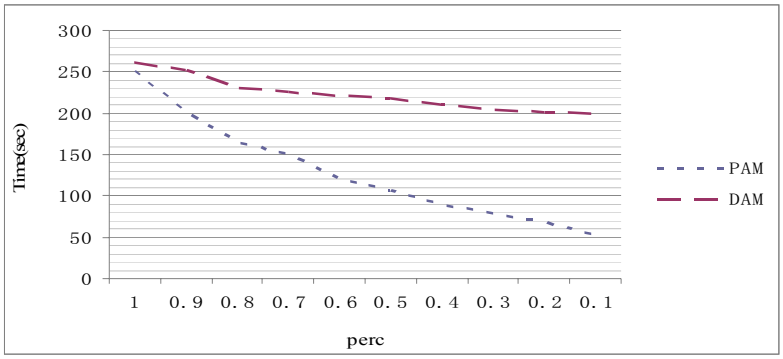


Figure 5.3 Running Time with Varying perc

5.3 Experiments on Finding Relationships Among Rules

Since a trend rule can be viewed as a single trend fragment spanning the whole time period, it can be regarded as a special case of the oscillating rule. Therefore we only evaluate the approaches (GBF and RBF) which are used to find the relationships in oscillating rules.

Figure 5.4 shows the running time of GBF and RBF when the number of rules increases from 1000 to 10 000 and parameter min_ratio is 0.85. We observe that GBF outperforms RBF. As the number of rules increases, the running time of RBF increases faster than GBF. In other words, GBF is more scalable than RBF. The reason for this is that RBF performs pairwise comparisons among rules, while GBF groups comparable fragments and performs pruning to avoid unnecessary comparisons.

Figure 5.4: Running Time of GBF and RBF

We also evaluate the sensitivity of GBF and RBF to min_ratio parameter. The rule number is set to be 5000. We vary min_ratio from 0.55 to 1 and evaluate the performance of GBF and RBF as shown in Figure 5.5.

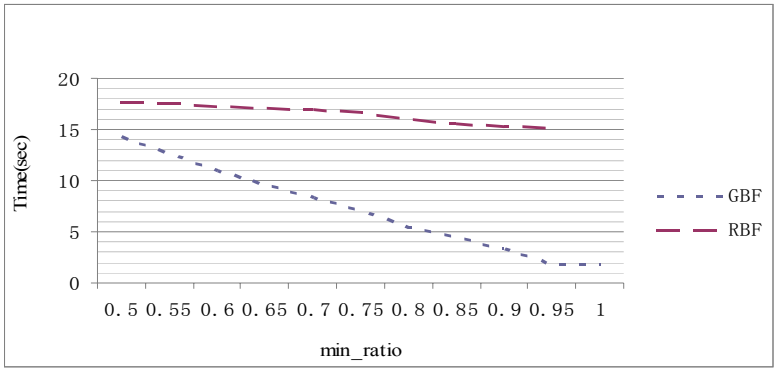


Figure 5.5: Varying min_ratio in GBF and RBF

We observe that GBF is faster than RBF. As the min_ratio increases from 0.55 to 1, the running time of GBF decreases rapidly, while the running time of RBF remains relatively constant. The reason is that when min_ratio is large, many combined rules do not have comparable fragments with the sub-rules and there is no relationship among them. GBF finds pairs of combined rule and its sub-rules only if they have fragments in the same group of comparable fragments. However, RBF finds each pair of combined rule and its sub-rules even when the rules do not have comparable fragments, and check whether their fragments have combined relationships (diverging, enhancing, and alleviating). As a result, GBF is more efficient when min_ratio is larger.

5.4 Experiments on Real World Dataset

Finally we use a real-world dataset to demonstrate the applicability of the algorithms in discovering meaningful relationships among rules. The dataset is the currency exchange rate dataset [34]. It contains the prices of 12 currencies relative to the US dollar from 10/9/1986 to 8/9/1996. The 12 currencies include AUD Australian Dollar (AUD), Belgian Franc (BEF), Canadian Dollar (CAD), French Franc (FRF), German Mark (DEM), Japanese Yen (JPY), Dutch Guilder (NLG), New Zealand Dollar (NZD), Spanish Peseta (ESP), Swedish Krone (SEK), Swiss Franc (CHF) and UK Pound (GBP). As discussed in the previous chapters, we mine association rules with a specific target. If we are interested in the conditions where the Japanese Yen will increase, then the target value is "Japanese Yen increase". One example of such rule is "Australian Dollar decrease, Canadian Dollar decrease $\Rightarrow$ Japanese Yen increase" with support of 0.5 and confidence of 0.9. This rule means that if we find that Australian Dollar decrease and Canadian Dollar decrease, we can predict that Japanese Yen will increase with a high accuracy of 0.9. To find such rules, we transform the changes of the prices on each day into a corresponding transaction as follows. For each day, the price of each currency is compared with its price of the previous day. Each increase or decrease of the price is associated with a corresponding Integer item in the transaction. If the target currency increases, the transaction will be put into the positive sub-dataset (*PD*). Otherwise the transaction

will be put into the negative sub-dataset (*ND*). After that we mine the association rules from *PD* and *ND* using Algorithm 4.1.2.

To analyze the dynamic behavior of the rules and the relationships among rules over time, we divide the dataset into 9 sub-datasets by year, excluding Year 1986 since its data is small. Then we mine each sub-dataset using the method discussed above and track the confidences of each rule. After that, we analyze the dynamic behavior of the rules and find evolution relationships among rules using the approaches proposed in Chapter 4.

Table 5.2 shows the number of relationships found when we target the increase of five different currencies and Table 5.3 shows some samples of the relationships. Each row corresponds to the number of different relationships found when the target currency is the entry of the first column. Note that in Table 5.3 " ↑ " denotes the confidence of the rule increases, " ↓ " denotes the confidence of the rule decreases and " – "denotes the confidence of the rule stays stable.

Target currency	Diverging	Enhancing	Alleviating	Competing
French Franc	31	0	31	755
German Mark	9	0	0	548
New Zealand	286	4	0	311
Spanish Peseta	107	1	78	807
Swedish Krone	319	0	28	317

Table 5.2: Number of Relationships With Different Categories

No	Relationship	Rules	Period
1	Competing	NLG +,DEM- => ESP+ ↑ NZD+,JPY- =>ESP+ ↓	1987-1991
2	Diverging 1	AUD-,CAD-,FRF-,GBP- => ESP+ ↑ AUD-, FRF-,GBP- => ESP+ ↓ CAD- => ESP+ ↓	1990-1992
3	Diverging 2	FRF+,ESP+,AUD-,CAD- => SEK+ ↓ AUD-,CAD- => SEK+ ↑ FRF+,ESP+ => SEK+ ↑	1991-1994
4	Enhancing	AUD-,FRF-,JPY-,SEK-, CHE- =>ESP+ ↑ AUD-,FRF-,CHE- =>ESP+ ↓ JPY-,SEK- =>ESP+ –	1990-1992

Table 5.3: Examples of Relationships

Following is the interpretation of the relationships in Table 5.3. For the first relationship, the rule "NLG +, DEM- => ESP+" means that if NLG increases and DEM increases, we can predict that ESP will increase, with some accuracy (the confidence of this rule). The competing relationship between the two rules means that from 1987-1994, the accuracy of the rule "NCG +, DEM- => ESP+" increases as the accuracy of the rule "NZD+,JPY- =>ESP+" decreases. As such, we have more confidence to judge whether ESP will increase based on the former rule than the latter rule because the former rule is more and more accurate. As for the second relationship, the diverging relationship among the three rules means that the accuracy of "AUD-, FRF-,GBP- => ESP+" and "CAD- => ESP+" decrease over

time while the accuracy of their combined rule “AUD-,CAD-,FRF-,GBP- => ESP+” increases. This is important information to the currency traders because they are aware that nowadays they cannot predict that ESP increases only based on the conditions {AUD decrease, RFF decrease and GBP decrease} or the condition {CAD decrease}. They are more confident to predict that ESP increases if all these conditions are satisfied. Similar interpretation can be applied to the other two relationships.

Chapter 6

Conclusion & Future Work

In this work, we have investigated the association rules from temporal dimension. We analyze the dynamic behavior of association rules over time and propose to classify the rules into different categories. By our definition, a stable rule is more reliable and can be trusted. A monotonic rule has a systematic trend in the whole time period and therefore it is predictive. An oscillating rule has several trends over time. An irregular rule has no trends and change irregularly which make it not so useful. Classifying rules into these categories can help the user to understand and use the rules better.

We also define some interesting evolution relationships of association rules, which might be important and useful in real-world applications. The evolution relationships reveal the correlations about the effect of the conditions on the consequent over time,

which reflect the change of the underlying data. Therefore they give the domain user a better idea about how and why the data changes.

In the last, we propose the corresponding approaches. To mine the association rule in our problem, we partition the whole dataset into positive and negative sub-datasets. Then we mine the frequent itemsets from the positive sub-dataset and count the support of the frequent itemsets from the negative sub-dataset. In this way, we only mine the frequent itemsets from part of the whole dataset, which make our approach more efficient. To analyze the dynamic behavior of the rule, we propose to find the trend fragments and classify a rule based on the number of its trend fragments over time. To find evolution relationships among rules we present a series of related methods such as GBF and RBF which are used to find the relationships among oscillating rules. Experiments on the synthetic and real-world datasets show that our approaches are efficient and effective.

In this work, we leave the task of partitioning the original dataset into sub-datasets by time period to the user. This requires the user possesses some prior knowledge of the domain. One of the possible future topics is to design a suitable method to automatically partition the dataset into sub-datasets, such partition should reflect the change of underlying data accurately. Another possible direction is to discover the relationships among rules by analyzing their content, rather than their statistics properties (support or confidence) as in this work, i.e. to discover whether a rule is the mutation of another rule. That is to identify the transformation of rules over time. For example, we might want to know whether one rule is changed from

another rule or several other rules. This can also give the user better insights into the dynamic behavior of the underlying data.

BIBLIOGRAPHY

[1] R. Agrawal, T. Imielinski and A. Swami. Mining association rules between sets of items in large databases. SIGMOD 93, pp 207-216.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB 94, pp 487- 499.

[3] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM 94, pp 401-408.

[4] J. Han, J. Pei, Y. Yin and R. Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery, 8, 53–87, 2004.

[5] J. Bayardo and R. Agrawal. Constraint-Based Rule Mining in Large, Dense Databases. Data Mining and Knowledge Discovery, 4, 217-240, 2000.

[6] R.T. Ng, S. Lakshmanan, A. Pang and J. Han. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD 98.

[7] R. Srikant, Q. Vu and R. Agrawal. Mining association rules with item constraints.

KDD 97, pp 67-73.

[8] R. Agrawal and J. C. Shafer. Parallel mining of association rules: Design, implementation, and experience. IEEE TKDE, 8. pp 962-969, Dee 1996.

[9] D.W. Cheung, J. Han, V. Ng and C.Y. Wang. Maintenance of discovered association rules in large databases:An incremental updating technique. ICDE 96, pp 106-114.

[10] E.H. Han, G. Karypis and V. Kumar. Scalable Parallel Data Mining for Association Rules. SIGMOD 97, pp 277-288.

[11] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB 95, pp 420-431.

[12] R. Srikant and R. Agrawal. Mining generalized association rules. VLDB 95, pp 407-419.

[13] B. Liu, W. Hsu, S. Chen and Y. Ma. Analyzing the subjective interestingness of association rules.Intelligent Systems and Their Applications.2000.

[14] B. Liu, W. Hsu, L.F. Mun and H.Y. Lee. Finding interesting patterns using user expectations.IEEE Trans.on Know. & Data Eng, vol: 11(6),1999.

[15] G. Piatetsky-Shapiro and C.J. Matheus. The interestingness of deviations. KDD-94, 1994.

[16] B. Liu, M. Hu and W. Hsu. Multi-level organization and summarization of the discovered rules. KDD-2000.

[17] B. Padmanabhan and A. Tuzhilin. A belief-driven method for discovering unexpected patterns. KDD-98.

[18] P.N. Tan and V. Kumar. Interestingness measures for association patterns: a perspective. KDD-2000 Workshop on Post-processing in Machine Learning and Data Mining, 2000.

[19] B. Ozden, S. Ramaswamy and A. Silberschatz. Cyclic association rules. In Proc. 1998 Int. Conf. Data Engineering (ICDE'98),1998.

[20] S. Ramaswamy, S. Mahajan and A. Silberschatz. On the Discovery of Interesting patterns in Association Rules. VLDB, 1998.

[21] Y. Li and P. Ning. Discovering Calendar-based Temporal Association Rules. Data & Knowledge Engineering 44 (2003).

[22] J. Ale and G. Rossi. An Approach to Discovering Temporal Association Rules. ASC'2000. Italy

[23] S.K. Harms, J. Deogun and T. Tadesse Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences. ISMIS 2002, LNAI 2366, pp. 432-441. 2002.

[24] S.K. Harms and J. Deogun. Sequential Association Rule Mining with Time Lags. Journal of Intelligent Information Systems, 22:1,7-22,2004.

[25] H. Lu, J. Han and L. Feng. Stock movement prediction and n-dimensional inter-transaction association rules. In Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pages 12:1--12:7, 1998.

[26] S. Baron and M. Spiliopoulou. Monitoring Change in Mining Results. Proceedings of the 3rd International Conference on Data. pp.51-60, 2001.

[27] S. Baron, M. Spiliopoulou and O. Gunther. Efficient Monitoring of Patterns in Data Mining Environments. ADBIS, pp 253-265, 2003.

[28] S. Baron and M. Spiliopoulou. Monitoring the Evolution of Web Usage Patterns. EWMF 2003, pp.181-200, 2004.

[29] M. Spiliopoulou and S. Baron. Temporal Evolution and Local Patterns. LNAI 3539, pp.190-206, 2005.

[30] B. Liu, R. Lee and Y. Ma. Analyzing the Interestingness of Association Rules from the Temporal Dimension. Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM), 2001.

[31] X. Chen and I. Petrounias. Mining Temporal Features in Association Rules. PKDD'99

[32] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. KDD 99, 1999.

[33] S. Mann. Introductory Statistics. John Wiley & Sons. INC.2003.

[34] http://www.stat.duke.edu/data-sets/mw/ts_data/all_exrates.html.