분산데이터베이스 환경하의 시간연관규칙 적용

조암, 김룡, 서성보, 류근호
충북대학교 데이터베이스 연구실
{Zhaoy, Kimlyong, sbseo, khryu}@dblab.chungbuk.ac.kr

Discovery Temporal Association Rules in Distributed Database

Yan Zhao, Long Jin, Sungbo Seo, Keun Ho Ryu
Database laboratory, Chungbuk National University

Abstract

Recently, mining for association rules in distributed database environments is a central problem in knowledge discovery area. While the data are located in different share-nothing machines, and each data site grows by time. Mining global frequent itemsets is hard and not efficient in large number of distributed servers. In many distributed databases, time component(which is usually attached to transactions in database), contains meaningful time-related rules. In this paper, we design a new DTA(distributed temporal association) algorithm that combines temporal concepts inside distributed association rules. The algorithm confirms the time interval for applying association rules in distributed databases. The experiment results show that DTA can generate interesting correlation frequent itemsets related with time periods.

1 Introduction

Recently, there has been considerable interest in extracting association rules from large number of distributed databases. DARM(Distributed Association Rules Mining) develops very fast in this area. There are many excellent algorithms to resolve the problem. However most of these kinds of algorithms all directly focus on source data.

DTA imports temporal features to DARM. The algorithm is based on Calendar-based Temporal Association rules[2]. Usually transaction databases contain temporal features. Suppose that M is one chain supermarket. Every branch shop of M sells the commodities every day and each shop has its own database to save the information. Applying temporal association rules in such environment can help us mine interesting temporal patterns to direct the marketing policy. In such environments, finding global association rules is so hard and costful. With respect to applying association rules over all the distributed data sites, these data sites include old supermarket(contains 10 years□ data) and new market (contains one year뭘 data). Since the combination is unbalanced (for the old data and new data mass together), we may have a wrong results. DTA imports the temporal features in this environment. It can filter unnecessary data and make the results more meaningful. Then, DTA uses one efficient DARM algorithm that is efficient and cheaper communication. DTA uses client-server model, and the algorithm deduces from FDM[1] and DDM[5].

Briefly, DARM needs algorithm to resolve the limitation of the network bandwidth, load-balancing problem and exact the rule patterns problem. DTA tries to resolve the problem from temporal point of view. There are two contributions in this paper. Firstly, we try to make a distributed temporal association rules that can fit for distributed data sites environment. Secondly, we make a system structure to manage tasks and results.

The paper is organized as follows: After the introduction we review the related research works and define system framework. Section 3 makes basic definitions and explains DTA algorithm briefly. Section 4 evaluates the performance and results. At last, section 5 concludes the paper.

2 General frameworks

2.1 Related work

There are some characteristics in DDM environment. Each data site has the ability to compute its own data. They are connected loosely by network, the network bandwidth is more expensive compared to CPU cost and each node뭘 ability and capacity is different. All the algorithms try to resolve three main problems. How to arrange distributed tasks and make them work balanced? How to minimize the communication costs? How to combine result and generate excellent useful rules? DDM algorithm tries to resolve the problems mentioned before from different point of view. For some special mining tasks and applications, different algorithms are designed upon their special characters.

FDM[1] takes advantage of the fact that ARM algorithms only look for rules, which are globally large. FDM adds pruning step before sending the frequent itemsets counting. And this step really reduces the communication cost. DDM[5] is more efficient and resilient to data skewness. It has better ability to overcome certain communication difficulties such as unordered messages. FDM and DDM try to improve the ARM algorithm to

minimize the communication cost. But the data in each site is unbalanced. Exp. old data mixes with new data together.

In this paper we import temporal association rules to distributed databases. It minimizes communication cost by adding temporal features. It can help the distributed association rules by mining more exact results and pruning unnecessary data. And it뭘easy to face incremental and unbalanced problem. It뭘a new idea in distributed data mining area. In the following section we will talk about the system structure.

2.2 System structure

Figure 1 shows us the system structure. There is one global server. It receives user뭘 temporal mining request, then separates them into tasks and arrange them to distributed data sites. Each distributed data site contains temporal datasets. We organize them into different pieces of basic temporal intervals. DTA depends on these temporal intervals.

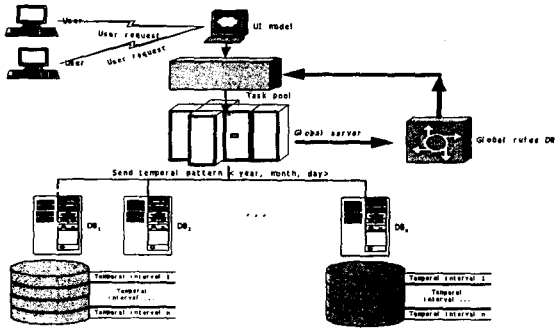

Figure 1 distributed system structure

Firstly, a user뭘 request comes to our system. UI model receives the request and shows the result. Then the task manager module explains the request to our global server. After that, task manage puts these calendar patterns into the task pool. DTA algorithm starts at this point. DTA receives the calendar patterns, runs on global server and contacts with distributed data site agent models to generate results. Right now all the tasks from task pool work in sequence. Whenever one task is finished, the global server puts the result into the UI model to show the result. After that, the result is saved in global rules database for future usage. After getting enough rules, we can generate new rules from exist rules.

3. DTA definition and algorithm

3.1 Simple calendar-based pattern

We present a class of calendar related temporal patterns called calendar patterns. Calendar pattern represents the sets of time intervals in terms of calendar schema specified by user.

Definition 1: *Calendar schema* is a relational schema $R = (f_n:D_n, f_{n-1}:D_{n-1}, ..., f_1:D_1)$, where each attribute f_i is a time granularity name like year, month, day etc. Each domain D_i is a domain value corresponds to f_i. "*" denotes all values corresponding to domain and means "every□

For example, given the calendar schema (week, day, hour), the calendar pattern <1,*,10> represents time intervals, which means the 10th hour of every day of week 1.

Definition 2: *k-star calendar pattern*, we call a calendar pattern with exactly k symbols a k-star calendar pattern (denoted e_k) and a calendar pattern with at least one symbol a star calendar pattern.

Let DB be a temporal database with D transactions. Each transaction contains temporal related information. Assume that there are n sites $S_1, S_2, □ , S_n$ in a distributed system and the database DB is partitioned over the n sites into { $DB_1, DB_2, ..., DB_n$} respectively.

Let the size of the partitions DB_i be D_i, for i=1, □ n. Let X.sup and $X.sup_i$ be the support counts of an itemset X in DB and DB_i, respectively. X.sup is called the global support count, and $X.sup_i$ the local support count of X at site S_i. For a given minimum support threshold s, X is globally large if X.sup ≥ s × D_i correspondingly, X is locally large at site S_i, if X.supi ≥ s × D_i. In the following, GL denotes the globally large itemsets in DB, and GL_k denotes the globally large k-itemsets in GL, and $GL_k(e)$ the time interval e globally large k-itemsets. The essential task of a temporal distributed association rules mining algorithm is to find the globally large itemsets GL(e). And other symbols we will use in algorithm are shown as Table 1.

Table 1 symbols and definitions

Symbol	Definitions
e	a star calendar pattern
DB_i	Distributed database site i
SDB	Searching database group, it is the DB_i group that intersect or contain time interval e.
$GL_k(e)$	Global k large itemsets of e
$DB_i\_LL_k$	contains DB_i Local Large itemsets and its local counting
$GC_k(e)$	Global k itemsets candidate of e
GC_table	contains the global frequent itemsets candidates and their support counting

3.2 Import temporal features in DDM

Mining temporal association rules can be decomposed into two steps: (1) finding all large itemsets for all star calendar patterns on the given calendar schema, and (2) generating temporal association rules using the large itemsets and their calendar patterns. The first step is costful of the discovery of temporal association rules; in the following, we will focus on this problem. The generation of temporal association rules from large itemsets and their calendar patterns is straightforward and can be resolved using the method discussed in [6]. The main problem of temporal association rules in distributed databases is how to generate large itemsets efficiently and make the communication between data sites minimized.

The central aim of importing temporal feature into DDM is to prune unnecessary data. It뭘the preprocessing step in temporal association rules. In our algorithm, firstly, global server collects the DBs, which contains the calendar pattern. Each DB_i only mines association rules in calendar pattern e of local data. If one DB_i does not contain data during e, we do not consider it in the mining task.

3.3 Algorithm overview

The main purpose of this algorithm is to apply temporal association rules in distributed environments. We use sever-

clients module in this algorithm. Global server collects information about distributed databases rules and manages different data sites. Each distributed data sites runs its own agent, to receive global server뭘tasks and send the results back.

Figure 2 shows the outline of our algorithms in global server side. (1)-(3) is to collect large 1 itemsets inside calendar pattern e from distributed databases. After that, all the DB_i are collected into SDB group, and then sent the require time interval e to distributed databases $DB_i \in SDB$ to collect necessary information.

```
(1)  for ( i=1; i<=n; i++ ) // n is distributed databases number
(2)        if (DBi∩ e ≠ ∅) insert(DBi, SDB);
(3)  Generate (GL1(e), SDB);
(4)  Generate GC2(e) from GL1(e);
(5)  for ( k=2; ∃ a star calender pattern e such that GCk(e)
     ≠∅; k++ ) do begin
(6)       forall DBi ∈ SDB do begin
(7)            Send_candidate_counting(GCk(e),DBi);
(8)            Receive_LL_itemsets(DBi_LLk);
(9)            Combine (DBi_LLk, GC_table) ;
(10)      end
(11)      Move Make_sure(GC_table) into GLk(e);
(12)      forall DBi ∈ SDB do begin
(13)           Send_unsure(GC_table,DBi);
(14)           Recive_count(GC_table,DBi);
(15)      end
(16)      Insert Make_sure(GC_table) into GLk(e);
(17)      Output <GLk(e),e> for star calendar pattern e;
(18)      Insert the pattern into rules Global_rules_DB;
(19)      generate candidates GCk+1(e) from GLk(e);
(20) end
```

Figure 2 DTA algorithm

(5)-(20) is the main part of the whole algorithm. The main idea is as follow. At first, we collect the $G_1(e)$, then can generate global large 2 itemsets candidates $GC_2(e)$ from $Gl_1(e)$ and send the candidates $GC_2(e)$ to distributed data sites. Then each data site DB_i scan the local database and count the local counting of $GC_2(e)$. After counting, we know which candidate itemset is local large and only send the $LL_2(e)$ back to global server. The global server collects SDB group for all local large itemsets and their counting. Here we know that, the $GL_2(e)$ must be contained in $DBi\_LL_2(e)$ (i = 1, □ n), and we apply make sure function step (Line 11) here to minimize the unsure candidate global itemsets, then send it back to DBi for checking. After checking, we can get the $GL_2(e)$. And then apply on k=3 if $GC_3(e)$ is not null. It loops until $GC_k(e)$ is NULL.

The client algorithm in each distributed sites can easily describe as 1) Receive the request from server; 2) generate LL and send back; 3) receive make_sure counting request; 4) send back the counting.

In make sure step (11, 16): Here we try to explain make sure and pruning step main ideas by example. Suppose that we have {AB, AC, BC, BD} large itemsets, and we have the counting of them, the support=δ and 3 distributed databases. Now if the AB_counting> GT×δ, that means AB is global large itemsets, else {AC, BC, BD} counting are grouped into un_sure group.

4 Performance and results

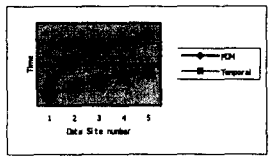

Figure 3 running time compare

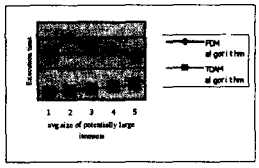

Figure 4 large itemsets generating

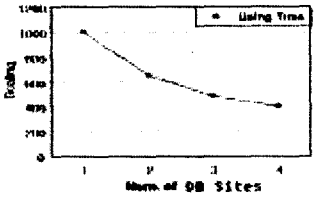

Figure 5 increment DB sites

We use synthetic data sets to evaluate DTA algorithm. Compared with FDM, the results show that DTA is less consuming. DTA can generate more frequent itemsets; its scaling ability is better and its communication cost lows down.

On the other hand, DTA generates temporal rules. The rules are more meaningful and accurate. For example, we can get <2001, 12, *> rules, in common, which shows out 2001 Christmas month뭘association pattern.

5 Conclusion and future works

This paper focuses on applying temporal association rules in distributed databases. We draw out the system structure and explain the algorithm briefly. DTA has following advantages, 1) low communication cost, 2) better balanced ability, 3) meaningful temporal rules, 4) easy to face incremental problem.

The algorithm is just a primary algorithm in the field of distributed temporal association rules. Future research will focus on how to improve distributed part and try to use temporal concepts in other algorithms such as classification.

References

[1] David Cheung, Jiawei Han, Vincent T. Ng, Ada W. Fu, Yongjian Fu, 밃 Fast Distributed Algorithm for Mining Association Rules,□ PDIS, 1996.

[2] Yingjiu Li, Peng Ning, X.Sean Wang, Sushil Jajodia, 밃 iscovering Calendar-based Temporal Association Rules,□ TIME, 2001.

[3] Ran Wolff, Assaf Schuster, Dan Trock, 밃 High-Performance Distributed Algorithm for Mining Association Rules,□ IEEE Conference on Data Mining, 2003.

[4] Chang-Hung Lee, Cheng-Ru Lin, Ming-Syan Chen, 밟n Mining General Temporal Association Rules in a Publication Database,□ ICDM, 2001.

[5] Assaf Schuster, Ran Wolff, 밃 ommunication-Efficient Distributed Mining of Association Rules,□ ACM SIGMOD Conference, 2001.

[6] Rakesh Agrawal, Ramakrishnan Srikant, "Fast algorithms for mining association rules in large databases,□ VLDB, 1994