Mining Frequent Pattern Form Large Dynamic Database with Time Granularities to Improve Efficiency

Pradnya A. Shirsath, Vijay Kumar Verma

Abstract— Incremental algorithms can manipulate the results of earlier mining to derive the final mining output in various businesses [1, 2, 3]. This study proposes a new algorithm, called the new approach for efficiently incrementally mining frequent pattern from large Dynamic database. Proposed approach is a backward method that only requires scanning incremental database. Rather than rescanning the original database for some new generated frequent itemsets in the incremental database, we add the occurrence counts of newly generated frequent itemsets and delete infrequent itemsets obviously. Thus, new proposed approach need not rescan the original database and to discover newly generated frequent itemsets. Proposed approach generates fewer candidates, reduces complex calculation and has good scalability as compared to the previous methods.

I. INTRODUCTION

In real-world applications, the target data is changed with time in association rules mining, and then existed association rules will also be changed, so the incremental mining algorithm should be developed [3,4]. Recent important applications have called for the need of incremental mining. This is due to the increasing use of the record-based databases whose data is being continuously added. Examples of such applications include Web log records, stock market data, sales data, transactions in electronic commerce, and daily weather or traffic records, to name a few. Incremental mining is not only including new data (i.e., data in the new month) into, but also remove the old data (i.e., data in the most obsolete month) from the mining process [4, 5, 6].

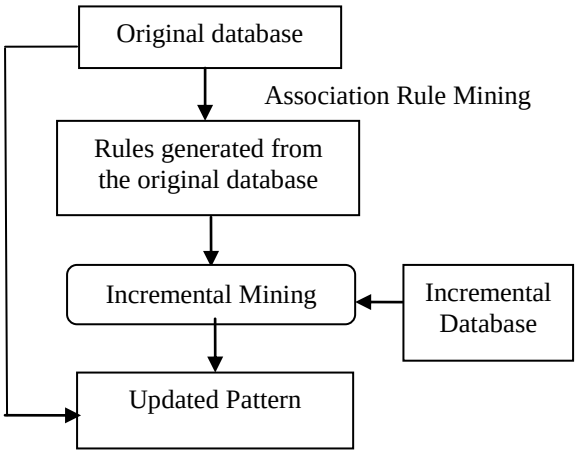

Fig -1 Process of Incremental Mining

Manuscript received October, 2013.

Pradnya A. Shirsath, Department of Computer Science and Engineering, Lord Krishna College of Technology, Indore, MP, India.

Prof. Vijay Kumar Verma, Department of Computer Science and Engineering, Lord Krishna College of Technology, Indore, MP, India.

Discovering the frequent item sets becomes more time consuming if the dataset is incremental in nature. In the incremental dataset, new records are added time to time. Generally the size of the increments, i.e. the number of records added to the dataset, is very small in comparison to the whole dataset [7, 8]. But due to the addition of these new records, scenario of the rules in the updated dataset may be changed. Some of the new item sets may become frequent, while some previously derived frequent set may become infrequent. For example, say X and Y are two distinct item sets of a dataset having 500 records with support count 200. Subject to the minimum support of 30%, X is frequent item set but Y is not.

II. TEMPORAL DATA MINING

The algorithms proposed for incremental mining were unable to handle data which include time expression. So to solve this problem new concept TAR (Temporal association rule mining) was introduced.

Temporal association rule mining was first introduced by Wang, Yang and Muntz in years 1999-2001. TAR (Temporal Association Rule) algorithm was introduced first to solve the problem on handling time-series by including time expression into association rules. It helps to find the valuable relationship among the different item sets, in temporal database. Temporal association rules are different from traditional association rules. Temporal association rules attempt to model temporal relationships in the data [7, 8, 9].

There are different types of temporal association rules defined in the literature such as intertransaction rules, episode rules, trend dependencies, sequence association rules and calendric association rules. Several algorithms have been proposed for mining the temporal association rules in temporal database.

Among these algorithms, ITARM algorithm was introduced by Tarek et al which discovers the temporal frequent item set after the temporal transaction database has been updated. The basic idea of ITARM algorithm depends on previously generated 2-candidate item set with their supports. ITRAM works as it checks first the extension of the pervious partition and attempts to find 2-candidate item set from the new partition; if it succeeds then it merges the current partition with the pervious partition, and from there it finds the 2-candidate item set. This approach is basically introduced to facilitate incremental mining techniques over an ever updating transaction database [8, 9, 10].

III. METHODOLOGY

No matter what kinds of tools or algorithms you select, strong temporal association rule mining can be divided into 4 steps. Following figure shows methodology of Temporal Association Rule Mining.

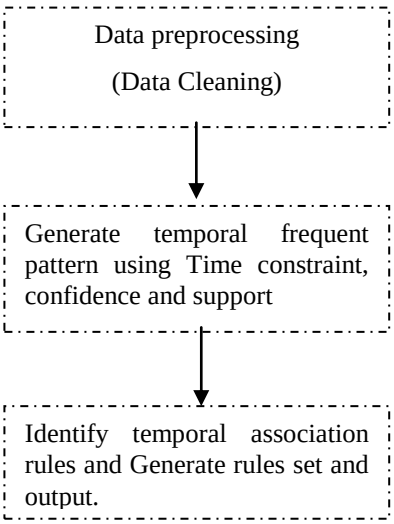

Fig 2- Working process of mining temporal association rule

First step is Data pre-processing. Data pre-processing performs several steps to improve the quality of the input dataset. This is an important step of data mining. Data pre-processing include removal of unwanted or irrelevant data, integration of databases and data exchange and data reduction. By processing, we can get high quality data mining objects.

Seconds step uses time constraints on the two parameters support and confidence to generate temporal frequent itemsets. All the generated temporal frequent itemsets have the support no less that min support.

Third step generate association rules with frequent itemsets. Here the association rules are temporal ones. It is different to generate association rules without time, because it adds time information on frequent itemsets.

IV. RELATED WORKS

Different methodologies were proposed to explore the problem of discovering temporal association rules. In the previous works, discovering association rules was performed on a given subset of a database specified by time. However, these works did not consider the individual exhibition period of each item. The exhibition period of an item is the time duration from the partition when this item appears in the transaction database to the partition when this item no longer exists [9, 10, 11]. That is, the exhibition period is the time duration when the item is available to be purchased.

Hence, these works cannot be effectively applied to a temporal transaction database, such as a publication database, where the exhibition periods of the items are different from one another. As a result, the concept of general temporal association rules has been proposed where the items are allowed to have different exhibition periods, and their supports are made in accordance with their exhibition periods. The new mining algorithms have been presented for the general temporal association rules in transaction databases such as Progressive Partition Miner (PPM), and Segmented Progressive Filter (SPF). On the other hand, other algorithms have been proposed for mining temporal association rules with numerical attributes such as the TAR algorithm [10, 11, 12].

As a matter of fact, temporal databases are often appended by adding new transactions. Hence, the previously discovered rules have to be maintained by discarding the rules that become insignificant and including new valid ones. Currently, some algorithms are proposed for the incremental mining of temporal association rules with numerical attributes. However, the incremental mining of temporal transaction databases is still at its infancy. Moreover, the incremental temporal mining algorithms with numerical attributes cannot be easily adapted to the transaction database. In the case of numerical attributes, we deal with objects. Each object has a unique ID and a set of numerical attributes. The database is viewed as sequence of snapshots of objects. The interestingness measures used are density and strength instead of confidence.

A temporal association rule is discovered from what is called base cubes instead of itemsets as long as it achieves the minimum support and strength. Moreover, many algorithms have been proposed for the incremental mining of association rules such as Fast Update algorithm (FUP), The Update Large Itemsets algorithm (ULI), Negative Border with Partitioning (NBP), Update with Early Pruning algorithm (UWEP), Fast Incremental Mining (FIM) algorithm and Pre-FUFP Algorithm[9,12,13,14],.

Some of these algorithms address the problem of determining when to update, while the others simply treat arbitrary insertions and deletions of transactions. Unfortunately, none of these algorithms address the incremental mining of temporal association rules. Following table shows comparative analysis of previously proposed algorithm.

Table I Comparison between various methods

Name of Method	Scanning Method/ Concept used	Nature of dataset	Performance
FUP	Multiple Scanning and Apriori based	It works on Incremented dataset only	Poor
FUP2	Multiple Scanning	It works on incremented as well as decremented dataset	Poor
NFUP	It scans only incremented dataset not original dataset	It works on incremented datasets only and it can also works with temporal dataset	Average
PPM	Multiple Scanning	It works on incremented datasets only	Good
SPF	Scanning by maximal candidate itemset	It works on incremented, decremented datasets and also temporal datasets	Good
ITARM	Based on FP growth tree	It works on incremented datasets and temporal datasets	Average

In this paper, the Recurrent Frequent Pattern Mining (RFPM) is proposed. It is used to maintain temporal frequent itemsets after the temporal transaction database has been updated.

V. PROBLEM STATEMENT

Recent researches in the field of temporal association rule mining are using Apriori based approach. These proposed approaches may still encounter some difficulties for different datasets such as Sparse or dense dataset. The limitations in these approaches are twofold [16, 17, 18]

1. Huge space is required to perform the mining in Apriori based temporal association rule. It generates a huge number of candidates in case of a dataset, which is large and/or sparse. Space requirement reducing is a challenge. Thus necessitates improvement in the existing approach.
2. The mining approach should ideally have more scalability. The existing Apriori based temporal association rule, may easily cause thrashing when dataset become large and sparse.

VI. PROPOSED APPROACH

The proposed Approach partitions the incremental database logically according to unit time interval (month, quarter or year, for example). For each item, assume that the ending time of exhibition period is identical. Proposed Approach progressively accumulates the occurrence count of each candidate according to the partitioning characteristics. The latest information is at the last partition of incremental database. Therefore, proposed approach scans each partition backward, namely, the last partition is scanned first and the first partition is scanned last. As in the preceding section, the original transaction database is denoted as DB, where db indicates the incremental portion, and DB+ signifies the updated database. The frequent set of itemsets of DB is known in advance. The new transaction database db includes n unit time intervals. Logically, db can be divided into n portions and each portion is called a partition (db = D_1 U D2 U... U Dn where D_n denotes the partition n).

Let $db^{m, n}$ represent the continuous time interval from partition P_m to partition P_n, where $n \geq m \geq 1$.

The final set of frequent itemsets consists of the three following types.

1. S1 set: frequent itemsets in *DB+*,
2. S2 set: frequent itemsets in $db^{m, n}$ ($m \leq n$), but infrequent in $db^{m-1, n}$, and
3. S3 set: frequent itemsets in $db^{m, n,}$ but infrequent in $db^{m+1,n.}$

Each candidate or frequent itemset has three attributes.

- Count: includes the occurrence count in current partition
- Start: includes the partition number of the corresponding starting partition when *X* becomes an element of frequent set, and
- Type: denotes one of the three types S1, S2 and S3. In the beginning, set S1, set S2, and set S3 are empty.

Consider a simple example of temporal database

Table II Simple transaction database

Date	Partition	Transaction	
		TID	Itemset
DB (D1)	2009	001	A,C,D,E
		002	A,C,D
		003	B,C,E
		004	A,B,C,E
		005	A,B,E
D2	2010	006	B,C,E
		007	A,B,E
		008	B,C,D,E
D3	2011	009	A,B,C,D
		010	C,E,F
		011	A,B,C,F

Consider the transaction database presented in Table 3 with a minimum support requirement is 50%. The original database contains five transactions and that six incremental transactions are divided into two portions {D1, D2}. D corresponds to the time granularity from 2009 to 2010. D1 and D2 correspond to the time granularities 2002 and 2003, respectively. All frequent itemsets of *D* are known in advance, and presented in Table II. Now Frequent Itemset in each partition shown in Table III.

Table III Frequent Itemsets in each partition

D3		D2		D1	
Itemset	Count	Itemset	Count	Itemset	Count
{A}	2	{B}	3	{A}	4
{B}	2	{C}	2	{B}	3
{C}	3	{E}	3	{C}	4
{F}	2	{BC}	2	{E}	4
{AB}	2	{BE}	3	{AC}	3
{AC}	2	{CE}	2	{AE}	3
{BC}	2	{BCE}	2	{BE}	3
{CF}	2			CE	3
{ABC}	2				

After scanning D3 table IV shows frequent item sets generated incrementally with time variant by proposed approach.

Table IV frequent item set in D3

S1 set	Start	Count
{A}	3	2
{B}	3	2
{C}	3	3
{F}	3	2
{AB}	3	2
{AC}	3	2
{BC}	3	2
{CF}	3	2
{ABC}	3	2

As shown in Table V from 2009 to 2011 three products ({*A*}, {*B*} and {*C*}) are very popular because their start partitions are one. {*AB*} and {*BC*} are the two elements of the S2 set and 2 is their start partition. Thus, from 2010 to 2011, the two combinations of products are interesting. {*AE*} is in the S3 set and the start partition is zero. Hence, {*AE*} is very popular from 2009. However, {*AE*} is no longer interesting from 2010 to 2011. The itemset {*E*} is frequent in traditional Incremental rules mining such as FUP, while {*E*} is in the S3 set of the mining result of proposed approach.

Table V frequent item set in D3 and D2

After Scan D2		
S1	Start	Count
{A}	2	3
{B}	2	5
{C}	2	5
{AB}	2	3
{BC}	2	4
S2	Start	Count
{F}	3	2
{AC}	3	2
{CF}	3	2
{ABC}	3	2
S3	Start	Count
{E}	2	3
{BE}	2	3
{CE}	2	2

The entire frequent patterns are shown in the table VI.

Table VI Frequent Itemset in all Partitions

Final Frequent Item set		
S1	Start	Count
{A}	1	7
{B}	1	8
{C}	1	9
S2	Start	Count
{F}	3	2
{AB}	2	3
{AC}	3	2
{BC}	2	4
CF	3	2
ABC	3	2
S3	Start	Count
E	2	3
AE	1	3
BE	2	3
CE	2	2

Proposed approach needs not rescan the original database since the S1 set is frequent in the updated database DB+, and the S2 set provides important rules in $db^{m, n}$ .Determining all frequent itemsets in DB+ as Apriori or FUP, it requires the rescanning of the original database only once to check the S2 set and S3 set. However, this rescanning phase is unnecessary because all frequent itemsets in DB+ are the subset of S1 U S2 U S3. Furthermore, all the itemsets contained in S2 set or S3 set are interesting.

VI. EXPERIMENTAL RESULTS

All the experiments were performed on a 2.0 GHz. Pentium 3i PC with 2GB MB of main memory, running under Windows XP. The algorithm was coded in Visual Basic Dot Net 2010 version. The real life datasets employed in our experiments. We are using SQL server for string database .We generate the transaction database *DB*+ with size |*DB*+|, where the first |*DB*| is the original database and the next is the incremental portion.

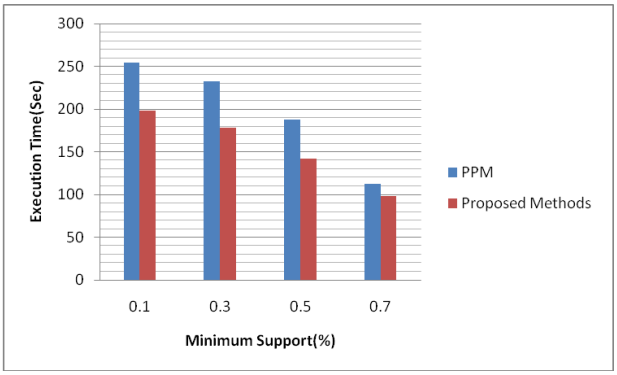

Fig 2- Execution Time and minimum support Graph

To test the scalability with the number of transactions of db, the number of partitions of db is set to 2. Figure presents the results. Consider the two minimum support thresholds 30% and 40%, the running time slightly increases with the growth of db's size. Thus, proposed approach shows good scalability.

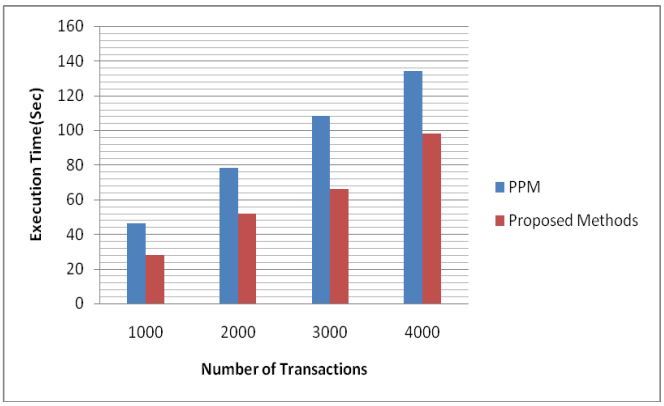

Fig 3- Execution Time and Number of Transactions Graph

In the real world, databases are periodically and continually updated. Therefore, mining must be repeated. Valid patterns and rules must to be efficiently generated. Incremental mining must usually involve the original database and the new added transactions. Scanning the original database is very expensive; figure 1 show that proposed method outperforms others by avoiding the rescanning of the original databases.

This investigation has presented a new method, proposed approach, for incremental mining. Proposed approach does not require the rescanning of the original database and can determine new frequent itemsets at the latest time intervals. The proposed method uses information available from a following partition to avoid the rescanning of the original database; it requires only the incremental database to be scanned. In reality, the transaction number of the incremental database is very small in contrast to the original database. The running time of proposed approach rises almost in direct proportion with the transaction number of the incremental database. Accordingly, proposed approach is suited frequently updated databases.

REFERENCES

[1] J. Han, M. Kamber, "Data mining, Concepts and techniques", Academic Press, pp 21-34, 2003.
[2] David Hand, Heikki Mannila and Padhraic Smyth, " Principles of Data Mining and Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm"
[3] Arun K. Pujari, "Data mining Techniques", University Press (India) Private Limited, pp 2-6, 2006
[4] D. Hand, H. Mannila, P. Smyth, "Principles of Data Mining", Prentice Hall of India, pp 141-150, 2004
[5] Pauray S.M. Tsai , Chih-Chong Lee , and Arbee L.P. Chen An "Efficient Approach for Incremental Association Rule Mining", Department of Information Management, Ming Hsin Institute of Technology, Hsin-Feng, Hsinchu 304, Taiwan, R.O.C
[6] William Cheung and Osmar R. Zaïane, "Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint", University of Alberta, Edmonton, Canada
[7] Rahman, Mohammad.M AL-Widyan Philadelphia, "Reduce Scanning Time Incremental Algorithm (RSTIA) of Association rules", Academic Research International2, September Volume 1, Issue 2, September 2011 University, Amman, JORDAN.
[8] Wei-Guang Teng and Ming-Syan Chen, " Incremental Mining on Association Rules", Department of Electrical Engineering National Taiwan University Taipei, Taiwan, ROC.
[9] Anour F.A. Dafa-Alla, Ho Sun Shon, "Incremental Mining of General Temporal Association Rules", Journal of Information Processing Systems, Vol.6, No.2, June 2010.
[10] Mohammed J. Zaki , Karam Gouda, "GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets".
[11] Weiqiang Lin Mehmet A. Orgun, "An Overview of Temporal Data Mining", The Australasian Data Mining Workshop
[12] Wei Wang, Jiong Yang Richard Muntz, " TAR: Temporal Association Rules on Evolving Numerical Attributes", T.J. Watson Research Centers IBM
[13] Keshri Verma O P Vyas, "Temporal Association Rule Using Without Candidate Generation", 3rd International caliber - 2005, cochin, 2-4 february, 2005, © inflibnet centre, ahmedabad
[14] Litvak Marina Temporal Mining Algorithms: Generalization and Performance Improvements The research work for this thesis has been carried out at Ben-Gurion University of the Negev under the supervision of Prof. Ehud Gudes November 2004
[15] Yingjiu Li Peng Ning X. Sean Wang Sushil Jajodia, "Discovering Calendar-based Temporal Association Rules", the work was partially supported by ARO under contract number DAAG-55-98-1-0302. Work of Wang was also partially supported by the NSF Career award 9875114.
[16] Chelliah Balasubramanian, Karuppaswamy Duraiswamy, "A mining method for tracking changes in temporal association rules from an encoded database", Chelliah Balasubramanian et al International Journal on Computer Science and Engineering Vol.1(1), 2009, 1-8
[17] Mohsin Naqvi, Kashif Hussain, Sohail Asghar, Simon Fong, "Mining Temporal Association Rules with Incremental Standing for Segment Progressive Filter 1", Center of Research in Data Engineering (CORDE), Mohammad Ali Jinnah University, Islamabad, Pakistan