Association Rule Mining for Dynamic Database Algorithms-a Survey

Jyoti Golakia
Department of Computer Engineering,
Atharva College of Engineering
Mumbai, Maharashtra

Trupti Shah
Department of Computer Engineering,
Atharva College of Engineering
Mumbai, Maharashtra

Snehal Kathale
Department of Information Technology,
Atharva College of Engineering
Mumbai, Maharashtra

Komal S. Mahajan
Department of Information Technology,
Atharva College of Engineering
Mumbai, Maharashtra

***Abstract*— Association rule discovery is an important area of data mining. Association rules identify associations among data items and were introduced in 1993 by Agarwal et al. Most of the algorithms for finding association rules deal with the static databases. In dynamic databases, new transactions are appended as time advances. This may introduce new association rules and some existing association rules would become invalid. Thus, the association rule mining for a dynamic database is an important problem. In this paper, we analyzed three algorithms-Modified Border, Promising frequent itemset and New Fast Update for finding association rules for dynamic databases and observed their pros and cons. These algorithms reduce rescanning of the original databases.**

***Keywords*— Borders, Promising frequent itemset, FUP, NFUP.**

I. INTRODUCTION

Following the explosive growth of data gathered by transactional systems, a challenge for finding new techniques to extract useful patterns from such a huge amount of data arose. Data mining emerged as a new research area to meet this
challenge. Data mining, or knowledge discovery, is the computer-assisted process of digging through and analyzing enormous sets of data and then extracting the meaning of the data. One major application area of data mining is association rule mining [1] was first introduced in [Agrawal et al. 1993]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. The association rule mining problem is to find out all the rules in the form of $X \Rightarrow Y$, where X and $Y \subset I$ are sets of items, called itemsets. The association rule discovery algorithm is usually decomposed into 2 major steps. The first step is find out all large itemsets that have support value greater than minimum support threshold and the second step is to find out all the association rules that have value greater than minimum confidence threshold.

One general assumption is that database is static. However in reality, most of the databases are dynamic and are updated frequently i.e. new transactions are added, old transactions are deleted and existing transactions are modified frequently. So the itemsets which are frequent may become infrequent when the database is updated and the itemsets which were infrequent may become frequent when the database is updated. Moreover, new database may contain some new interesting rules which were not present in the old database. One obvious technique is re-running association rule mining algorithms in the updated database to find the frequent itemsets in the updated database. However, this is not the optimal solution because of running the algorithms on adding, deleting or updating a small number of records. It will be time consuming to scan the same database repeatedly and generation of same itemsets repeatedly. As a brute force approach, apriori may be reapplied to mining the whole dynamic database when the database has been changed. However, this approach is very costly even if small amount of new transactions is inserted into a database. Thus, the association rule mining for a dynamic database is an important problem. Several research works [2, 3, 4, 5, 6] have proposed several incremental algorithms to deal with this problem.

Section II describes Modified Borders algorithm, Promising Frequent Itemset algorithm and New Fast Update algorithm. Section III includes analysis of the algorithm. Section IV includes conclusion and future work.

II. ASSOCIATION RULE MINING ALGORITHMS FOR DYNAMIC DATABASE

Many algorithms are available for generating association rules for dynamic database. In this section, some of the frequently used algorithms are reviewed and the problems associated with each algorithm are discussed.

A. Modified Borders Algorithm

This algorithm is the modified version of Borders algorithm. An itemset X is called a **border set** if X is not frequent, but all its proper subsets are frequent. So collection of border sets forms the border line between the frequent sets and non-frequent sets. An itemset that was a border set before the database was updated and has become frequent after the database has been updated is called a **promoted border set**. Borders algorithm also uses the same concept and maintains support counts for all the frequent sets as well as border sets. The algorithm scans the whole database, if there is some promoted border set. Otherwise, it does not require scanning the whole database. The Borders algorithm is robust enough to find the frequent itemsets in a dynamic database. However, the cost of the scanning of whole database in the every iteration becomes too expensive with the increase in the volume of the old database as well as new database. It also suffers from scalability problem.

Hence modified borders algorithm was proposed [7]. It include some of the borders which are likely to become promoted border to generate candidate sets in the old database so that if those borders become promoted, no new candidates will have to be generated. This reduces the rescanning of original database. Modified Borders work by considering two border sets. The first border set is B-old *I* and the second border set is B-old *II*. B-old *I* contain items of border set with support value greater than some β < α (min. support) and less than α which will take part in candidate generation and B-old *II* contains items of border set with support value less than β and will not take part in candidate generation. Thus, B-old *I* contain the itemsets with higher probability of becoming promoted when new transactions are added. New candidate sets will be generated only when any of the elements of the B-old *II* becomes promoted. If new candidate itemsets are generated, one scan over the whole database is required to find supports of the new candidates.

B. Promising Frequent Itemsets Algorithm

In this algorithm we compute not only frequent itemset but also compute itemset that may be potentially large in an incremental database called “Promising frequent Itemset” [8]. The algorithm uses maximum support count of 1-itemsets obtained from previous mining to estimate infrequent itemsets of an original database that will capable of being frequent itemsets when new transactions are inserted into the original database. With maximum support count and total size of new transactions that allow insertion into an original database, support count for promising frequent itemsets i.e. min_PL, is given by

$$\min\_supp_{DB} - \left(\left(\frac{\text{maxsupp}}{totalsize}\right) \times inc\_\text{size}\right) \leq \min\_\text{PL} \leq \min\_\text{supp}_{\text{DB}} \quad (1)$$

Where

$\min\_supp_{DB}$ is minimum support count for an original database,

maxsupp is maximum support count of itemsets,

totalsize is a number of transaction of an original database,

*inc*_size is a maximum number of new transactions.

Apriori algorithm scans all transactions of original database to find all possible frequent k-itemsets and promising frequent k-itemsets. Items in both frequent k-itemsets and promising frequent k-itemsets can be joined together in the join step. The support count of frequent itemset must be higher than user-specified minimum support count threshold and the support count of promising frequent item must be higher than min_PL but less than the user-specified minimum support count.

1) Updating of frequent and promising frequent itemsets:

The size of an updated database increases when new transactions are inserted into an original database. Thus, min_PL must be recalculated in order to associate with the new size of an updated database.

min_PL (update) is computed as follows:

$$\min\_PL_{DB\cup db} = \min\_supp_{DB\cup db} - \left(\left(\frac{maxsupp}{totalsize}\right) \times inc_{size}\right) \quad (2)$$

Any k-item has support count greater than or equal to min_sup (DBUdb), this itemset is moved to a frequent k-item of an updated database. In the other case, if any k-item has support count less than min_sup(DBUdb) but it is greater or equal to min_PL(update) , this k-item is moved to a promise frequent itemset of an updated database.

In addition, if any item is a new frequent item or a new promising frequent item, this item will be joined and pruned with both a promise frequent k-itemset and a frequent k-itemset. The k-itemsets are scanned in an incremental database and if they are frequent they can become a frequent itemset in an updated database. Thus, if that itemset has support count greater than or equal to min_sup(db), the item is moved to an estimated frequent k-itemset. Similarly, a k-itemset can become a promising frequent itemset in an updated database only if the k-itemset is a frequent itemset in an incremental database.

Thus, if a k-itemset has support count less than min_sup(db) but greater or equal to min_PL(update) or min_PL(DB), the k-itemset is moved to an estimated promising frequent k-itemset. Then, both the estimated promise frequent k-itemsets and the estimated frequent k-itemsets are scanned in original database to update their support count.

C. NFUP (New Fast Update) Algorithm

This algorithm is a modification of FUP (Fast Update) algorithm. In 1996, Cheung et al. proposed the FUP algorithm to efficiently generate associations in the updated database. The FUP algorithm relies on Apriori and considers only these newly added transactions. Let DB be the original database, db de the incremental database and DB+ be the updated database (including DB and db). An itemset X can have any one of this functionality. X can be frequent or infrequent in DB or db.

TABLE I
Four scenarios associated with an itemset in DB+

DB \ db	Frequent itemset	Infrequent itemset
Frequent itemset	Case 1: Frequent	Case 2:
Infrequent itemset	Case 3:	Case 4: Infrequent

In the first pass, FUP scans db to obtain the occurrence count of each 1-temset. Since the occurrence counts of frequent itemsets in DB are known in advance, the total occurrence count of arbitrary X is easily calculated if X is in Case 2. If X is unfortunately in Case 3, DB must be rescanned. Similarly, the next pass scans db to count the candidate 2-itemsets of db. The FUP algorithm is time consuming because of rescanning of the original database. Hence, Chin-Chen Chang, Yu-Chiang Li and Jung-San L proposed NFUP (New Fast Update) algorithm [9].

NFUP partitions the incremental database logically according to unit time interval to mine new interesting rules in updated database. NFUP progressively accumulates the occurrence count of each candidate according to the partitioning characteristics. NFUP scans each partition backward, namely, the last partition is scanned first and the first partition is scanned last as the last partition contains the latest information. The frequent set of itemsets of DB is known in advance. The new transaction database db can be divided into n partitions (db = P1 U P2 U, ...,U Pn where Pn denotes the partition n).

Let $db^{m,n}$ represent the continuous time interval from partition Pm to partition Pn, where n≥m≥1 and nƐN.

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

- α set: frequent itemsets in DB+,
- β set: frequent itemsets in $db^{m,n}$ (m≤n), but infrequent in $db^{m-1,n}$
- γ set:frequent itemsets in $db^{m,m}$ but infrequent in $db^{m+1,n}$.

For $db^{n,n}$ (Pn), the process starts at 1-itemsets. Each frequent itemset has three attributes.

- X.count: includes the occurrence count in current partition,
- X.start: includes the partition number of the corresponding starting partition when X becomes an element of frequent set,
- X.type: denotes one of the three types and r.

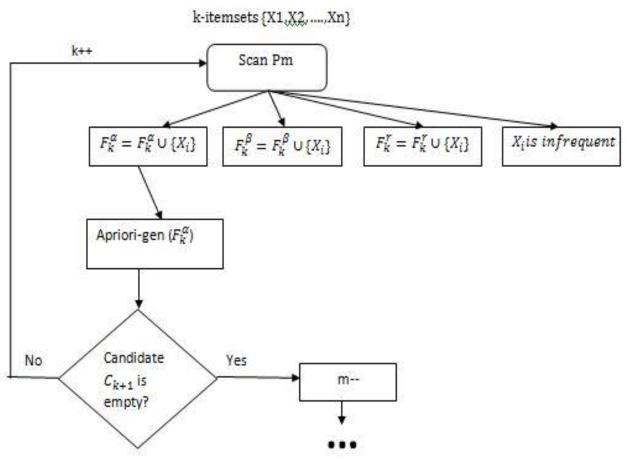
k-itemsets {X1,X2,....,Xn}
k++
Scan Pm
$F_k^\alpha = F_k^\alpha \cup \{X_i\}$
$F_k^\beta = F_k^\beta \cup \{X_i\}$
$F_k^\gamma = F_k^\gamma \cup \{X_i\}$
X_i is infrequent
Apriori-gen (F_k^α)
No
Candidate C_{k+1} is empty?
Yes
m--
•••

Fig 1. Process of NFUP for Pm

Fig. 1 shows the working of NFUP algorithm.

After Pn has been scanned, all frequent 1-itemsets are added into the set and are joined to form 2-itemset candidates. In Pn, the process is performed like that of Apriori. NFUP is applied to the next partition Pn-1 whenever no more candidate k-itemsets can be generated in Pn. In each partition, NFUP determines which candidate k-itemset will
become an element of or r set and identifies from which partition the k-itemset becomes frequent. After P1 is scanned, the occurrence count is accumulated with that of DB.

III. ANALYSIS OF ALGORITHMS

Modified Borders reduces rescanning of old database than Borders algorithm but still requires rescanning of database a few number of times. The value of β has a great effect on the performance of Modified Borders algorithm. As the value of β increases, the border sets which are likely to become promoted border sets will be less. Hence number of whole scan also increases. Also, Modified Borders tend to take little more time because number of full scan tends to increase with the increase in value of β. Thus, Modified Borders takes much less time than that of Borders when β is small.

Promising Frequent Itemset algorithm requires more computation because min_PL is calculated for each updation of database.

NFUP does not require rescanning of the original database and can determine frequent itemsets from recent transactions at the latest time intervals. But the running time of NFUP increases if the incremental database is large.

IV. CONCLUSION

In the real world, databases are periodically and frequently updated. Therefore, mining must be repeated on each update. In this paper, we analyzed various algorithms for generating association rules for dynamic database. Promising frequent itemset algorithm reduces the rescanning of old database as compared to Modified Borders algorithm. NFUP does not require rescanning of original database. However the execution time depends on the size of the database either original or incremental.

It can be observed that most of the data mining approaches discover the knowledge pertaining to frequently occurring entities. However, real-world datasets are mostly non-uniform in nature containing both frequent and relatively rare items. Some rare items are equally important for generating association rules. The rare cases are more difficult to detect because they contain fewer data. At high user specified minimum support value, rare itemsets are missed, and at low minimum support value, the number of frequent itemsets explodes.

REFERENCES

[1]. R. Agrawal., T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases,” In Proc. Of the ACM SIGMOD Intl Conf. on Management of Data (A CM SIGMOD ’93), Washington, USA, May1993

[2]. C. H. Lee, C. R. Lin , and M. S. Chen, “Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining ACM, 2000

[3].C.C. Chang, Y.C. Li and J.S. Lee, “An efficient algorithm for incremental mining of association rules,” Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA’05) ,IEEE, 2005

[4]. A. A. Veloso et al., “Mining frequent itemsets in evolving databases,” In Proc. 2nd SIAM Intl. Conf. on Data Mining, Arlington, VA, Apr. 2002

[5]. K.L. Lee, G. Lee and A. L.P. Chen, “Efficient Graph-based algorithm for discovering and maintaining knowledge in large database,” In Proc. third pacific-asia conference on methodologies for knowledge discovert and data mining, April 1999.

[6]. N. L. Sarda and N. V. Srinivas, “An adaptive algorithm for incremental. mining of association rules,” In Proc. 9th Intl. Workshop on Database and Expert System Applications,Vienna, Austria, pp. 240-245, Aug1998.

[7]. A Das & D K Bhattacharyya, Department of Information Technology, Tezpur University, Napaam 784 028, India “Rule mining for dynamic databases” In AJIS Vol 13, No. 1 September 2005.

[8]. Ratchadaporn Amornchewin & Worapoj Kreesuradej, “Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm” In ICICS 2007.

[9]. Chin-Chen Chang, Yu-Chiang Li and Jung-San Lee “An Efficient Algorithm for Incremental Mining of Association Rules” In Proc. 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications,2005