Threshold Based Association Rule Mining Algorithm for Dynamic Content

Prof. Jyoti Golakia[1] Prof. Trupti Shah[2] Prof. Krishanjali Shinde[3]
[1,2,3]Assistant Professor
[1,2,3]Department of Computer Engineering
[1,2,3]Atharva College of Engineering, Mumbai University,Mumbai, India

Abstract— Today, data mining has become a very vast area of research. A data mining technique, Association Rule Mining (ARM), is also a vast area of research. Association rules identify relationships among data items and were introduced in 1993 by Agarwal et al. With the increasing use record-based databases whose data is being continuously added, recentimportant applications have called for the needof incremental mining. In dynamic transactiondatabases, new transactions are appended andobsolete transactions are discarded as timeadvances. Several research works havedeveloped feasible algorithms for derivingprecise association rules efficiently andeffectively in such dynamic databases. Also,sometimes the itemsets are not as frequent asdefined by the threshold, but the associationrules generated from them are still important.Such items are called rare items. Classical ARMframework assumes that all items have the samesignificance or importance which is not alwaysthe case. In this paper, an algorithm – EnhancedNew Fast Update, is proposed for miningassociation rules from dynamic databaseconsidering rare interesting items also.

Key words: Data Mining, KDD, ARM

I. INTRODUCTION

Researchers are drowning in data, but starving for knowledge. Since the dawn of the Internet era in 1994, electronic commerce and e-data are growing at, such an astonishing rate and the companies around the world race to move their business online in order to position them in the Internet dominated worldwide trading [9]. This technology elevation leads to store tremendous variety of data in Information repositories like data warehouse, XML repository, relational database etc. The interesting, useful (potentially useful and previously unknown rules and patterns) information can be extracted from this large information repository. With the enormous amount of data stored in files, databases, and other repositories, it is increasingly important, if not necessary, to develop powerful means for analysis and perhaps interpretation of such data and for the extraction of interesting knowledge that could help in decision-making. Data Mining, also popularly known as Knowledge Discovery in Databases (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. While data mining and knowledge discovery in databases (or KDD) are frequently treated as synonyms, data mining is actually part of the knowledge discovery process. One major application area of data mining is association rule mining was first introduced in Agrawal et al. 1993[1]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories.

II. ASSOCIATION RULE MINING

Association rule mining discovers the frequent patterns among the itemsets. It aims to extract interesting associations, frequent patterns, and correlations among sets of items in the data repositories [10]. An association rule is an expression of the form A -> B, where X, Y are itemsets. It reveals the relationship between the itemsets X and Y. The two significant basic measures of association rules are support(s) and confidence(c). The fraction of transactions containing X also containing Y, i.e., $P(Y|X) = P(X\cup Y)/P(X)$ is called the confidence (conf) of the rule. The support (sup) of the rule is the fraction that contain all items both in X and Y, i.e., $sup(X\rightarrow Y) = P(X\cup Y)$. To generate an interesting association rule, the support and confidence should satisfy a user-specified minimum support (minsup) and minimum confidence (minconf) respectively.

III. INCREMENTAL ASSOCIATION RULE MINING

One general assumption is that database is static. However, in reality, most of the databases are dynamic and are updated frequently i.e. some new transactions are added, some old transactions are deleted and existing transactions are modified frequently. So the itemsets which are frequent may no longer remain infrequent when the database is updated and the itemsets which were infrequent may be no longer frequent when the database is updated. Moreover, some new interesting rules which were not present in the old database may be added in new database.
Considering an original database and newly inserted transactions, the following four cases may arise:
– Case 1: An itemset is large in the original database and in the newly inserted transactions.
– Case 2: An itemset is large in the original database, but is not large in the newly inserted transactions.
– Case 3: An itemset is not large in the original database, but is large in the newly inserted transactions.
– Case 4: An itemset is not large in the original database and in the newly inserted transactions.

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 each update of 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.

Sometimes even the itemsets are not as frequent as defined by the threshold, but the association rules generated from them are still important. Such items are called rare items. Classical ARM framework assumes that all items have the same significance or importance i.e. their weight within a transaction or record is the same (weight=1 per item) which is not always the case. For example, in the supermarket some items are very expensive, consequently they are not purchased so often as the threshold required, but association rules between those expensive items are as important as other frequently bought items to the retailer. To consider such items for generating association rules, weights are assigned to reflect the importance of items. Rare items which are important are assigned high weight so that they exceed the threshold.

IV. EXISTING ALGORITHMS

A. Automated Threshold Based Association Rule Mining

In order to find important associations, an appropriate support threshold has to be specified. The support threshold plays a key role in deciding the interesting itemsets. The rare itemsets may not found if a high threshold is set. Some uninteresting itemsets may appear if a low threshold is set. This approach involves rare items to obtain the frequent itemsets by setting the support thresholds automatically [7].

1) Apriori algorithm

Apriori algorithm has been proposed for finding frequent itemsets. This algorithm is based on iterative level-wise search for frequent itemset generation. It uses a single minsup value at all levels to extract frequent itemsets. Prior to the generation of frequent itemsets, the algorithm generates all candidate itemsets in that level. A candidate k-itemset is an itemset having 'k' number of items. A candidate k- itemset is said to be frequent if the support of the subset of candidate k-itemsets is greater than or equal to the user-specified minsup threshold. This algorithm is suitable for finding the frequent itemsets and not the rare itemsets. Unless the minsup value is fixed at a low value, the rare itemsets could not be found which leads to the explosion of frequent itemset generation.

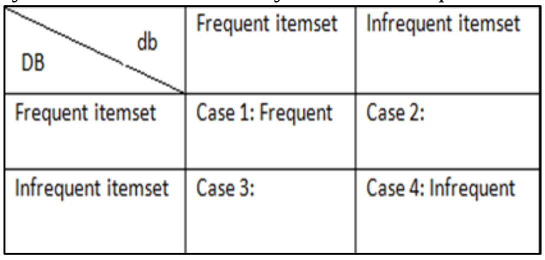
2) MSApriori algorithm

An extension of Apriori algorithm called MSApriori algorithm has been proposed in the literature (Liu et al., 1999) which attempts to discover frequent itemsets involving rare items. This algorithm assigns a minsup value known as MIS for each item and frequent itemsets are generated if an itemset satisfies the lowest MIS value among the respective items. This method derives the MIS values for items based on their support percentage. Here the frequent items are assigned with a higher MIS value whereas rare items are assigned with a lower MIS value. So, the MSApriori algorithm addressed the rare itemset problem and improves the performance over single minsup based algorithms.

In automated threshold based association rule mining algorithm, items are grouped into Most_interesting_Group(MiG), Somewhat_interesting_Group(SiG), Rare_interesting_Group(Ri). The itemset which has more support than average support of items in that level are known as Most_interesting_(MiG) in that level. The portion of itemsets which has low support than AvgSup has to be filtered as Somewhat_interesting_Items. The threshold used for filtering the SiG is known as MedianSup. The itemsets which have less support than AvgSup and MedianSup should be considered in Rare_interesting_Group. The rare items are separated in group RiT and MiG and SiG are separated in group iT. Then Apriori is applied to iT and MSApriori is applied to RiT.

B. New Fast Update (NFUP) 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.



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

Table – 1: Four Scenarios Associated with an Item Set in Db+

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 [8].

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 n∈N.

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

– ∝set: frequent itemsets in DB+,
– β set: frequent itemsets in$db^{m,n}$ (m ≤ n), but infrequent in $db^{m-1,n}$
– r 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 $\propto, \beta$ and r.

Fig. 1 shows the working of NFUP algorithm.

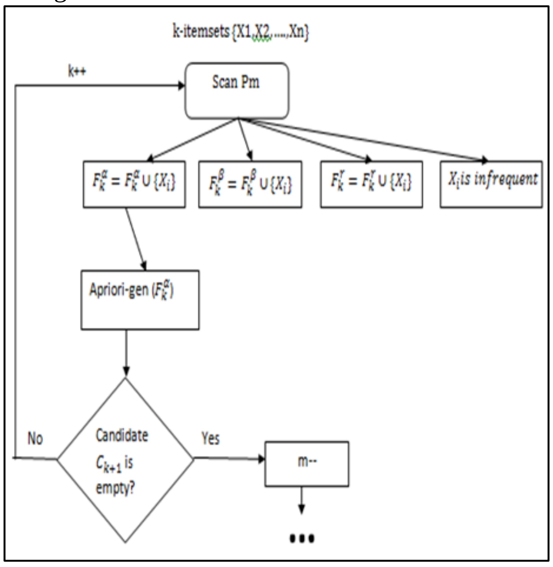

k-itemsets{X1,X2,....,Xn}
k++
Scan Pm
$F_k^\alpha = F_k^\alpha \cup \{X_i\}$
$F_k^\beta = F_k^\beta \cup \{X_i\}$
$F_k^\gamma = F_k^\gamma \cup \{X_i\}$
X_i is infrequent
Apriori-gen (F_k^α)
Candidate C_{k+1} is empty?
No
Yes
m--
...

Fig. 1: Process of NFUP for Pm

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 α,β 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.

Fig. 2 below shows the comparison of running time for different number of partitions created and different minimum support values taken. Here only two partitions are considered. It can be observed that as the numbers of partitions are more, the running time is less.

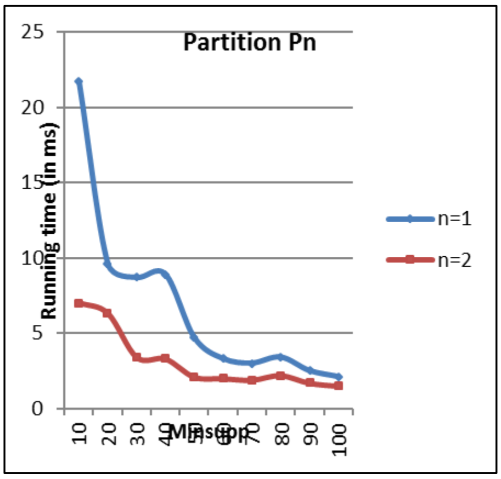


Fig. 2: Running time of NFUP for n=1, 2

V. Comparison of Existing Algorithms

Fig. 3 shows the comparison of automated threshold algorithm and NFUP algorithm on the basis of rules generated and the number of transactions in the database. The simulation is done on mushrooms dataset which is a dense dataset. It is observed that number of rules generated in automated threshold algorithm is more as compared to NFUP algorithm as the automated threshold algorithm contains rare items also.

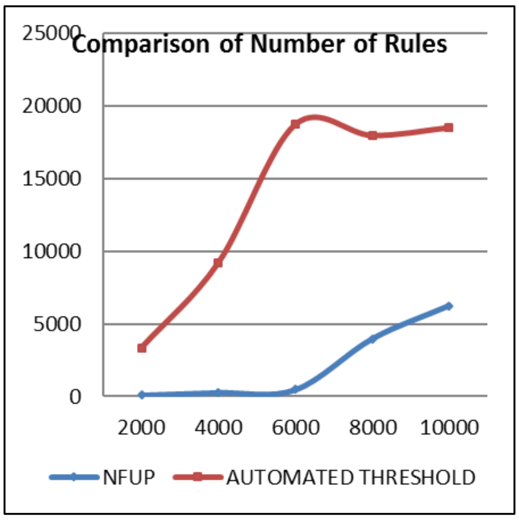


Fig. 3: Comparison of NFUP and Automated threshold algorithm on the basis of rules count and no. of transactions

VI. Proposed Algorithm

In the proposed scheme, the two above algorithms are combined by enhancing each of them.

A. Enhanced Automated Threshold based Association Rule Mining

The itemsets in iT and RiT are merged in Final and then candidate itemsets are generated and Apriori is applied.

B. Enhanced NFUP

Here the partitioning of the incremental database will be done on the basis of number of items and time interval. For e.g. if the threshold for number of items is 100, then counting from the latest transaction, the transactions containing total of 100 or near to 100 are consider as one partition. When the number of items in the transaction exceeds the threshold, partition is done. Timing constraint is kept to allow mining when it takes long for new transactions to be added to the database.

C. Steps of Proposed Algorithm

The steps of the proposed algorithm are as follows:

1) Partition the incremental database on the basis of number of items and time threshold.
2) Scan latest partition and find level 1 itemsets.
3) Group the items in MiG, SiG and RiG.
4) MiG and SiG are grouped into iT for frequent items and RiG into RiT for rare items.
5) Candidate itemsets are generated in group RiT and pruning is done based on MIS values and stored in RI. MIS are assigned as minimum of support of items in itemset. Assign the items to α, β and r as per the characteristic.
6) For frequent items, iT and RiT are merged, candidate sets are generated and store in FI. Assign the items to α, β and r as per the characteristic. Calculate AvgSup and MedianSup and goto step 3.
7) Scan all partitions and repeat step 3 to step 6.

VII. Conclusion

Association rule mining is an important area of data mining research and a comparatively a younger member of data mining community. The real-world databases, from which useful patterns and rules are mined, are dynamic in nature. It may become necessary to carry out the mining process again on the updated database. Mining is a costly activity, typically requiring multiple database scans. The proposed scheme which partitions the database on the basis of number of items, however, is efficient to mine association rules. In addition to this, rare itemset problem is also solved by setting support threshold automatically according to characteristics of rare items.

From the implementation results, it is concluded that the running time of NFUP algorithm depends on the number of partitions made in the incremental database. As the number of partitions increases, the execution time also increases. The two existing algorithms were compared on the basis of number of rules generated. The number of rules in automated threshold algorithm is more as compared to NFUP algorithm. And hence the proposed algorithm will generate more efficient rules as compared to existing NFUP as the rare interesting itemsets are also considered in the proposed approach.

ACKNOWLEDGEMENT

This paper is prepared for National Conferenceon Technological Advancement and Automatization in Engineering which is to be held in AET"s Atharva College of Engineering, Mumbai, India. I am grateful for the kind support and lab facility provided by principal, management and head of department of Computer Engineering.

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 Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA,May 1993.

[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] Kanimozhi Selvi Chenniangirivalsu Sadhasivam and Tamilarasi Angamuthu “Mining Rare Itemset with Automated Support Thresholds” in Journal of Computer Science 7 (3): 394-399, 2011

[8] 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

[9] Amit A. Nanavati, Krishna, I. and Chitrapura Sachindra Joshi Raghu Krishnapuram. Mining Generalised Disjunctive Association rules. ACM 1-581 13-436-3/01/0011, 2001.

[10] Dhanabhakyam, M and Punithavalli, M. A Survey on Data Mining Algorithm for Market Basket Analysis. Global Journal of Computer Science and Technology, Vol. 11 issue 11, version 1.0, 2011.