A Review of Incremental Association Rule Mining Techniques

Pooja Dubey[1], Dr. R. K. Gupta[2]

(Department of CSE & IT)
[1]Madhav Institute of Technology & Science, Gwalior (India)

Abstract- *Applications of Association rule mining are market basket analysis, customer's purchase pattern and web data accessing patterns. However, if new transactions are added time to time to the database means if the datasets are incremental in nature, frequent itemsets and association rules may change. Some of the new itemsets may become frequent, while some previously existing frequent set may become infrequent. Due to updated database some rules that are already derived may become obsolete and some new rules may be generated. For the new consistent rules over the updated dataset, if the association mining technique redo the rule generation process for the whole dataset, based on the frequent itemsets, simply by discarding the earlier computed results, it will be inefficient. It is mainly because of the multiple scanning over the older dataset. Various approaches have been proposed for incremental association rule mining as apriori based techniques, tree based and hashing based. In this paper, these types of incremental association techniques are reviewed.*

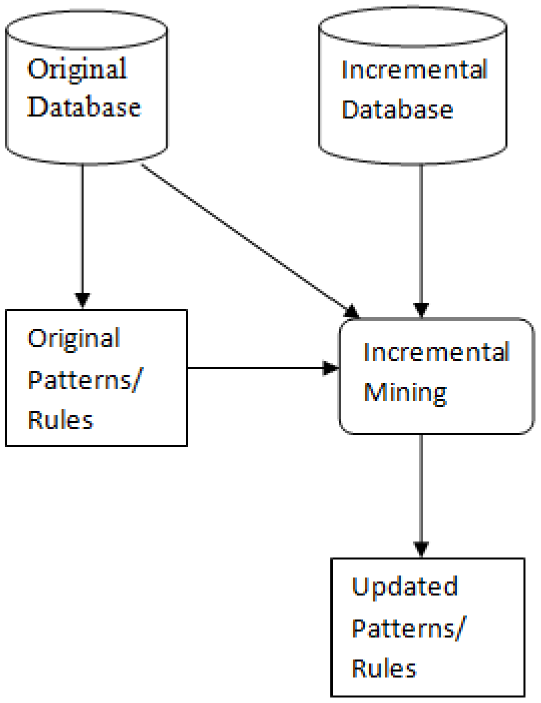
Index Terms - Incremental Mining, Support, Confidence, Frequent Itemset, Strong Association Rule.

I. INTRODUCTION

Due to the increasing use of large data with high computation required for various applications, the importance of data mining has grown rapidly. From the point of view of business application, analysis of previous transaction data can provide valuable information on behavior of customer, and thus help in making business decisions. Thus it is necessary to collect and analyze a sufficient data properly before making any decisions. Since the amount of data being processed is large, it is important for the mining algorithms to be very computationally efficient. Various data mining algorithms have been proposed in the literature to explore knowledge. Recently many important applications have created the need of incremental mining.

Mining association rules is the core task of numerous data mining techniques. As the amount of data increases, designing an efficient mining algorithm becomes increasingly important; accordingly, two of the main issues concerning data mining are therefore studied extensively herein. One is the development of algorithms for mining rules or patterns. The second is the design of algorithms to update and maintain rules, called incremental mining.

Association rule mining is a promising data mining technique which discovers strong associations or correlation relationships among data. Given a set of transactions (similar to database records in this context), where each transaction consists of items (or attributes), an association rule is an implication of the form X ->Y , where X and Y are sets of items and X ∩Y = Ǿ. The



support of this rule is defined as the percentage of transactions that contain the set X, while its confidence is the percentage of these "X" transactions that also contain Y. In association rule mining, all items with support higher than a specified minimum support are called large or frequent itemsets. An itemset X is called an i-itemset if it contains i items.

The very first association rule mining algorithm is the Apriori algorithm. The Apriori algorithm gives solution for two sub-tasks (1) to find all frequent itemsets, and (2) to use these frequent itemsets to generate association rules. The first task importantly governs the overall performance of the mining process. After frequent itemsets have been determined, the corresponding association rules may be derived easily. One has to first find out the frequent itemset using Apriori algorithm, Then, Association rules will be generated using min. support & min. confidence. The set of frequent 1-itemsets, L_1, consists of the candidate 1- itemsets satisfying minimum support. In the first iteration of the algorithm, each item is a member of the set of candidate. To discover the set of frequent 2-itemsets, L_2, the algorithm uses L_1 Join L1 to generate a candidate set of 2-itemsets, C_2. Next, the transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated (as shown in the middle table). The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support. The generation of the set of candidate 3-itemsets, C_3 , involves use of the Apriori Property. In order to find C_3, we compute L_2 Join L_2. $C_3 = L_2$ Join L_2. Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k. The algorithm uses L3 Join L3 to generate a candidate set of 4-itemsets, C4, this itemset is pruned which are not frequent. Suppose, C4 = φ, then algorithm terminates, having found all of the frequent items. This completes Apriori Algorithm. These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

II. TECHNIQUES OF INCREMENTAL ASSOCIATION RULE MINING

1. FAST UPDATE (FUP)

The first technique [2] was proposed by Cheung, Han et al. The FUP algorithm is based on Apriori and takes into account only newly added transactions. Let db be a set of new transactions and DB+ be the new updated database (including all transactions of DB and db). An itemset X is either frequent or infrequent in DB or db. Therefore, *X* has four possibilities, as shown in Table 1. In the possibility 1, FUP scans db to obtain the occurrence count of each 1-itemset. Since the support counts of F_k in older database are known by previous mining, the total frequency of *X* is easily calculated if X is in 2. If X is in 3, DB i.e. older database must be rescanned. Similarly, the next pass scans db to count the support of candidate 2-itemsets of db i.e. incremental database. The process is reiterated until all frequent itemsets have been detrermined. In the worst case, FUP does not minimize the number of the original database scan passes.

DB ↓	db →	Frequent itemset	Infrequent itemset
Frequent itemset		1: Frequent	2:
Infrequent itemset		3:	4: Infrequent

Table 1: Four cases associated with FUP

2. FUP2

In 1997, Cheung *et al.* [3] described the FUP2 algorithm, which is a more generalized incremental technique than FUP. An advanced version of FUP algorithm FUP2 is introduced for updating the existing frequent items and association rules when transactions are added to and deleted from the original database. It can deal with insertion as well as deletion. FUP2 performs efficiently not only on growing database but also on trimming data. The difference between FUP2 and Apriori is that FUP2 separates the candidate item-sets in the new database into two subsets in each pass of the algorithm. That is, in kth iteration, candidate itemsets C_k is divided into A_k and B_k, where A_k is the intersection of C_k and L_k, L_k is the the previous frequent itemsets of size k in the old database. B_k is the remaining part of C_k not included in the set L_k, that is, $B_k= C_k - (C_k \cap L_k)$. FUP2 algorithm also has poor performance.

3. UPDATE WITH EARLY PRUNING (UWEP)

Ayan et al. proposed an algorithm [4] to update large itemsets with early pruning. The benefit with UWEP are that it scans the existing database at most once and new database exactly once, and it generates and counts the minimum number of candidate itemsets in order to determine the updated set of large itemsets. Moreover, it prunes an itemset that will become small from the set of generated candidates as soon as possible by a look-ahead pruning technique. Thus, look-ahead pruning results in a much smaller number of candidates in the computation of new large itemsets in the updated database. Now suppose that X is small in the updated database. So, any superset of X must also be small in the updated database. UWEP differs from the previous algorithms [4, 5] at this point, by pruning all supersets of an itemset from the set of large itemsets in DB i.e. older database as soon as it is known to be small. In the previous algorithms, a k-itemset is only checked in the kth iteration, but UWEP does not wait until the kth iteration in order to prune the supersets of an itemset in LDB (large itemsets in older database) that are small in LDB+db (large itemsets in updated database).
Let X be a k-itemset which contains items I1, . , Ik. An immediate superset of X is a (k + l)-itemset which contains the k items in X and an additional item Ik+1. If X ¢ LDB, then X ϵLDB+db only if X ϵ Ldb. If X is large in DB, then X is also large in DB + db. In this case, we put X into LDB+& with the total support. If X is small in DB, we have to check whether it is large in DB+db or not. However, we do not know supportDB(X). We can obtain it by scanning DB. Thus this is more efficient technique than FUP2.

4. A LOW-SCAN INCREMENTAL ASSOCIATION RULE MAINTENANCE METHOD (MAAP)

This algorithm[6] makes use of an Apriori property, and starting with the high level large itemsets in the older database mining result, it computes the equivalent high level large item- sets in the updated database as well as infers some low level large itemsets in this updated database. Thus, this algorithm eliminates the requirement of computing some low level large itemsets and save on rule maintenance time. It is advantageous when high level large itemsets generate a high percentage of low level large itemsets. All non- empty subsets of a items frequentet must be frequent. For example, if a frequent 3- itemset is L3={123}, we can immediately infer that the following itemsets are frequent as well: {12}, {13}, {23}, {1}, {2}, {3}. Based on this principle, when association rules are to be maintained in the updated new database, the frequent itemsets can be computed from the highest level frequent itemsets, that is, from Lk. If any itemset in Lk is still frequent in the updated new database, its lower level subset itemsets are included to their appropriate level frequent itemsets in Lk-1, Lk-2, . . . , L1.

For example, since L3= {123} is confirmed to be still frequent in the new database, this algorithm includes {12}, {13}, {23} to L2 and {1}, {2}, {3} to L1. By doing so, some computation time is saved, the MAAP algorithm continues by checking if each itemset in L3 = { ABC, ACD, BCD } is still frequent in the updated new database. Since ABC is frequent in the new database, AB, AC, BC, A, B, C are also frequent itemsets. Thus, ABC is included in L3' (Li' denotes the large i-itemsets in the updated new database), include AB, AC, BC to L2', include A, B, C to L1'.

The procedure of this algorithm starts by computing parts of new large itemsets using only itemsets that were frequent in the older database, and are guaranteed to still be frequent in the new database because of a superset itemset in a higher level new frequent itemset. Assume for each pass in the older database, we divide the candidate itemsets into two parts. One part consists of the frequent itemsets, another part is the small itemsets.Then in second step, Compute for each updated new frequent itemset, additional frequent item- sets that were frequent in the old database but not computed in the first step because their superset higher level itemset is small in the new updated database, but these older lower level frequent itemsets may still be frequent in the newly grown database. In third step, It Computes the rest of the itemsets in the candidate itemsets that may be frequent itemsets in the new updated database. Since by the end of step 2 above, we reduced the sizes of all level infrequent itemsets and candidate sets, the algorithm now takes each infrequent itemset Si = Ci- Li' and scans the updated new database to determine if these itemsets are frequent in the new updated database. If they are frequent, they are included in the appropriate level new frequent itemset Li'. In the last step, all level i candidate sets are adjusted to include the new frequent itemsets previously small in the older database at level (i-1). This accomplishes the set computed above in Step 3 by including all candidate sets that arise from these new frequent itemsets.

5. COMPRESSED AND ARRANGED TRANSACTION SEQUENCE (CATS) TREE AND CANONICAL- ORDER (CAN) TREE

The FELINE Algorithm with the CATS Tree [7] extends the idea of the FP-tree to improve storage compression, and allows frequent-pattern mining without the generation of candidate itemsets. The goal is to build a CATS tree as compact as possible. All the items are arranged in descending local frequency order in the CATS tree So, during the mining process, the FELINE algorithm needs to traverse both upwards and downwards to include frequent items. Extra cost is required for the swapping or merging of nodes to make it compact.

CanTree [8] only requires one database scan. Items are arranged according to some canonical order, which can be defined by the user prior to the mining process or at runtime during the mining process. Specially, all the items can be consistently arranged in lexicographic order or alphabetical order. Once the ordering is defined (say, for *DB*), items will follow this ordering in the CanTrees for subsequently updated databases (e.g. $DB \cup db1$, $DB \cup db1 \cup db2$, ...) even the frequency ordering of items in these updated databases is different from *DB*. The ordering of items is unaffected by the changes in frequency caused by incremental updates. The frequency of a node in the CanTree is at least as high as the sum of frequencies of its children. CanTree may not be as compact as the corresponding CATS tree. However, it is important to note that CATS trees do not necessarily minimize computation or time because a lot of computation spent on finding mergeable paths as well as traversing paths upwards and downwards, while CanTrees significantly minimize computation and traversal time, because they easily find mergeable paths and require only upward path traversals.

6. ASSOCIATION RULE MINING BY MODIFIED APPROACH OF PROMISING FREQUENT ITEM-SET ALGORITHM BASED ON BUCKET SORT APPROACH

A new idea of incremental association rule mining which does not scan original database. i.e. without scanning original database it will scan only updated database. This incremental association approach [9] for mining streamed data applies on the dense data efficiently. This covered the old existed algorithm and minimized the execution time and space complexity is also reduced, and developed a new approach that directly compares the updated as well as old record of the database and there is no need to include the older database with new coming data and the used buckets will give the accurate result.

The itemset which are not frequent in original database but it could be frequent when updated transaction are added to database is called promising frequent itemset. This algorithm uses maximum support count of 1-itemset which is determined before and this will estimate infrequent itemset of original database which is going to be frequent when new transaction are added to the older database.

Based on maximum support count of 1-itemset, promising itemset, Support is calculated using follows :

$\text{Min\_PI}_{DB \cup db} = \text{min\_supp}_{DB \cup db} - \left(\frac{\max\ supp}{total\ \ size} * inc\_size\right).$

It checks which item are frequent and which are promising frequent item set and make list of both type. Then, adding incremented database, calculating frequent itemset and promising frequent itemset of incremented database. Now apply bucket sort on updated promising frequent itemset list, then calculating how many buckets are going to be frequent from promising list, now adding support of needed bucket 's itemset to the support of old promising itemset . Finally, move new frequent itemset into old frequent itemset list and remove from promising itemset list.

7. MINIMAL PERFECT HASHING AND PRUNING BASED INCREMENTAL MINING ALGORITHM

DHP (Direct Hashing & Pruning) and MPIP (Multi-Phase Indexing and Pruning) [10] algorithms employ hash tables to reduce the database access times. They are for deavery beneficial while dealing with the candidates of 2-itemsets i.e. *C2*, which is the most time-consuming step in association rules mining. Consequently, hashing-based association rule mining algorithms are more time saving than Apriori- based algorithms. IMPHP (Incremental Minimum Perfect Hashing and Pruning) algorithm has following advantages : 1)All the candidate itemset will be addressed by hashing into a hash table without collisions and their minimum support can be calculated by a hash function for latter process. 2.) When the database is updated, the arrangement of hash table will not be re- constructed. There is only a need to scan the updated parts and add new items into the end of the original hash table. Hence, the efficiency can be improved significantly. DHP employ hash functions to generate candidate itemsets efficiently, and DHP also employs effective pruning techniques to reduce the size of database. The potential-less itemsets will be filteredout in early stage of candidate generation, and the scanning of database will be avoided. Since DHP does not scan over the database all the time, the performance is also enhanced. DHP is particularly powerful for finding the frequent itemsets in starting stage. It finds 1-itemsets and makes a hash table for *C2*, and then determines *L2* based on the hash table generated in previous stage.
In MPIP algorithm, a unique address will be assigned to each itemset. It also enhances the accuracy of the hash table. All the entries in the hash table is used for determining the support of corresponding itemset. Because of such type of structure, the repeated scanning of database can be avoided. Besides, the Bit Vector in the hash table can filter out the candidate itemset and directly indicate the large itemsets. Hence, once we construct the hashing table, the frequent itemsets are also found.

In this algorithm, hashing address can be determined by a minimum perfect hashing function and mining with incremental transaction and item is also supported. In this section, we will analyze the regularity in 2-itemsets first and then advanced it to the case of 3-item and k-itemsets. Finally, the minimal perfect hashing function is obtained. The minimum perfect hashing function of 2-itemset in list as following:

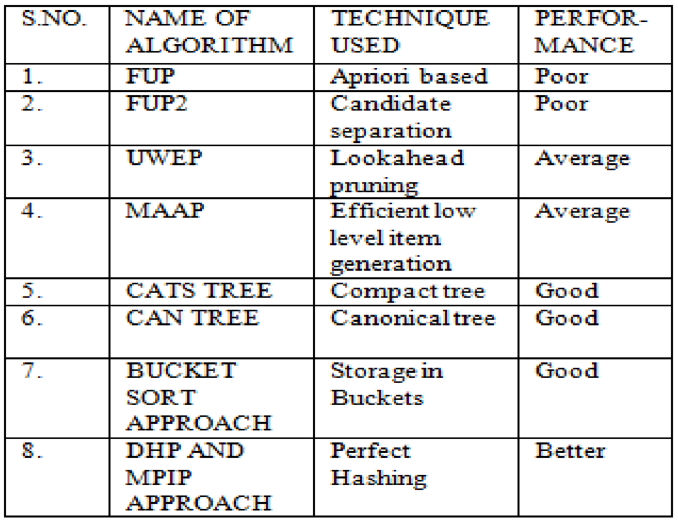
$$F_n\ (j_1, j_2) = 1\ ,\ \text{for } j_1 = 1 \text{ and } j_2 = 2$$

Otherwise,

$$F_n\ (j_1, j_2) = C_2^{j-1} + j_1\ .$$

Minimum perfect hashing function is employed in IMPHP to avoid collisions and enhanced the incremental mining efficiency. In addition, the arrangement of proposed hash table structure does not need to be reconstructed again when new items are added. All newly added items will be arranged at the end of the original hash table. Hence, the efficiency can be enhanced significantly when it applies for incremental association rule mining.

III. COMPARATIVE STUDY OF EXISTING RESEARCHES



S.NO.	NAME OF ALGORITHM	TECHNIQUE USED	PERFOR-MANCE
1.	FUP	Apriori based	Poor
2.	FUP2	Candidate separation	Poor
3.	UWEP	Lookahead pruning	Average
4.	MAAP	Efficient low level item generation	Average
5.	CATS TREE	Compact tree	Good
6.	CAN TREE	Canonical tree	Good
7.	BUCKET SORT APPROACH	Storage in Buckets	Good
8.	DHP AND MPIP APPROACH	Perfect Hashing	Better

IV.CONCLUSION AND FUTURE WORK

Association rule mining can give very useful and beneficial information, and improve the quality of business decisions in market, web based survey and in multinational companies. A number of incremental mining algorithms have been developed by different researchers in need of applications which uses record based database and where database increments rapidly. In todays scenario, a static approach can't persist for a long. The whole approach towards incremental mining is to make use of previously mined knowledge and scan the database so as to improve efficiency in terms of time and space . Most of the algorithms try to reduce the number of scans of database and maintain the association rules. Apriori based techniques like FUP requires more than two scans in worst case and FUP2 requires two complete scans of database, which are computationally less efficient in terms of time. Tree based algorithms, like CAN tree and FELINE, need only single scan of database. CATS tree requires swapping, merging and splitting of tree nodes to make it as compact as possible, since it uses frequency dependent ordering and this drawback has been overcome in CAN tree. CAN tree uses user defined ordering before or after the mining process. CATS tree also takes large computation time in finding merge-able paths and needs downward traversals during mining. DHP and MPIP algorithms employ hash table structures to minimize the database access times. In MPIP algorithm, a unique address will be assigned to each itemset. It also enhances the accuracy of the hash table. All the entries in the hash table are used for determining the support of corresponding itemset. Due to such type of structure, the repeated scanning of database can be avoided. Summarily, many algorithms have contributed having goal of incremental mining, however still there are scopes to enhance the efficiency of algorithms, generation of new techniques.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. "Mining Association Rules between Sets of Items in Large Databases", Proceedings

of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207—216, May 1993.

[2] D. Cheung, J. Han, V. Ng, and C. Y. Wong. Large Databases: An Incremental Updating Technique. Proceedings of the 12th International Conference on Data Engineering, pp.106—114, February 1996.

[3] D. Cheung, S. D. Lee, and B. Kao, "A General Incremental Technique for Updating Discovered Association Rules", Proceedings of the Fifth International Conference On Database Systems for Advanced Applications, pp. 185—194, April 1997.

[4] N. F. Ayan, A. U. Tansel, and M. E. Arkun, "An Efficient Algorithm to Update Large Itemsets with Early Pruning", Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 287 291, August 1999.

[5] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases", Proceeding of the 3rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 263- 266, August 1997.

[6] Z. Zhou and C. I. Ezeife. "A Low-Scan Incremental Association Rule Maintenance Method",Proceedings of the 14th Canadian Conference on Artificial Intelligence, June 2001.

[7] W. Cheung and O. R. Zaiane, "Incremental Mining of Frequent Patterns without Candidate Generation or Support Constraint", Proceedings of the 7th International Database Engineering and Application Symposium, July 2003.

[8] C. K. Leung, Q. I. Khan and T. Hoque, "CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns", Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), 2005.

[9] Ms. Anju k.kakkad1, Ms. Anita Zal, "Incremental Association Rule Mining by Modified Approach of Promising Frequent Itemset Algorithm Based on Bucket Sort Approach", International Journal of Advanced Research in Computer and Communication EngineeringVol. 2, Issue 11, November 2013.

[10] Chuang-Kai Chiou, Judy C. R. Tseng, "An Incremental Mining Algorithm for Association Rules based on Minimal Perfect Hashing and Pruning", APWeb'12 Proceedings of the 14th International conference on Web Technologies and Applications pp. 106-113, Springer-Verlag Berlin, Heidelberg, 2012.

[11] Chin-Chen Chang, Yu-Chiang Li, Jung-San Lee, "An Efficient Algorithm for Incremental Mining of Association Rules", Proceedings of the 15th IEEE International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE-SDMA'05), 2005.