Incremental Learning Algorithm for association rule Mining

Mrs. Prajakta Vispute, Prof. Dr. S. S. Sane

Abstract— These Association rule mining is to find association rules that satisfy the predefined minimum support and confidence from a given database. The Apriori and FP-tree algorithms are the most common and existing frequent itemsets mining algorithm, but these algorithms lack incremental learning ability. Incremental learning ability is desirable to solve the temporal dynamic property of knowledge and improve the performance of the mining process as the incremental data is available with the passage of time. Currently FUFP, pre-FUFP and IMBT algorithms have been developed that support incremental learning. The IMBT uses a binary tree data structure called an Incremental mining binary tree. This work proposes a novel incremental learning algorithm that makes use of a data structure called Item-Itemset(I-Is) tree that is a variation of B+ tree. Initially I-Is tree is created from the original data to allow searching of frequent items based on the threshold values. The created I-Is tree is updated incrementally.

Index Terms— : Association rule mining, B+ Trees, Data Mining,Frequent pattern tree, IMBT, Incremental mining,Support.

1 Introduction

THIS Data mining is one of the fastest growing fields in the computer industry. Data mining refers to extracting or mining knowledge from large amounts of data. Frequent pattern, as the name suggests, are patterns (a set of items, subsequences, substructures, etc.) that occurs frequently in a data. Association rule mining [1] [2] [3] is one method of finding frequent itemsets in which, it finds interesting associations and/or correlation relationships among a large set of data items. Association rules show attributes value conditions that occur frequently together in a given dataset. There are different association rule mining algorithms [1] [2] [6] [7] [8] in data mining and Incremental data mining. In incremental data mining [9] [10] [11] [12] speeding up the process of mining the frequent itemsets and proper utilization of memory is necessary.

Existing incremental data mining method uses a tree structure called IMBT (Incremental Mining Binary Tree) [16] to enumerate the support of each itemset in an efficient way after the transactions are added or deleted. Instead of rescanning the database many times to enumerate the support of the itemsets after the database update, it processes a transaction at a time and records the possible itemsets in a data structure that can reduce the processing and IO time.

In this work, conceptually suggested technique tries to use previous results as the basis to incrementally mine the database with new transactions. There are some performance issues such as a need to rescan the original database to enumerate the support when a database is updated, and issue to deal with the threshold changes during the lifetime of the database. To resolve these issues and to satisfy the speed and memory requirements the proposed work recommends the use of data structure called Item-Itemset (I-Is) Trees which is based on the concept of B+ Trees.

2 Literature Review

Mining frequent itemsets is a fundamental requirement for mining association rules. It plays an important role in many other data mining tasks such as sequential patterns, multi-dimensional patterns. In this section most popular and widely used association rule mining algorithms are discussed.

2.1 APRIORI ALGORITHM

Apriori algorithm [1] [2] works with Candidate Generation-and-Test Approach. Apriori algorithm computes the frequent itemsets in the database through several iterations. Iteration i computes all i-frequent itemset (itemset with i-elements).Each iteration has two steps:

1. Candidate generation
2. Candidate counting & selection

In the candidate generation phase of the first iteration, set of candidate itemsets containing all 1-itemset is generated and in the counting phase support for all 1-itemset is calculated for the whole database. Finally, only 1-itemsets with support above required threshold is selected as frequent itemset. The next iteration is based on the property that if a pattern with k items is not frequent, any of its super-patterns with (k +1) or more items can never be frequent.

A candidate-generation-and-test approach iteratively generates the set of candidate patterns of length $(k + 1)$ from the set of frequent patterns of length k $(k \geq 1)$ and check their corresponding occurrence frequencies in the database.

The Apriori algorithm achieves good reduction in the size of candidate sets. However, when there exist a large number of frequent patterns and/or long patterns, candidate generation-and-test methods may still suffer from generating huge numbers of candidates and taking many scans of large databases for frequency checking. Since, the number of database accesses of the Apriori algorithm has equaled to the size of the maximal frequent itemset. It accesses the database k times even when only one k-frequent itemset exists. If the dataset is huge, the multiple database scans can be one of the drawbacks of the Apriori algorithm. Moreover it does not have incremental learning ability.

2.2 FREQUENT PATTERN TREE (FP Tree)

The frequent pattern (FP-tree) [6] [7] is used for efficiently mining association rules without generation of candidate itemsets. The FP-tree mining algorithm consists of two phases. The first phase focuses on constructing the FP-tree from the database, and the second phase focuses on deriving frequent patterns from the FP-tree. The FP-tree is used to compress a database into a tree structure by storing only large items. It is condensed and complete for finding all the frequent patterns. Three steps are involved in FP-tree construction phase. The database is first scanned to find all items with their frequency. The items with their supports larger than a predefined minimum support are selected as large 1-itemsets (items). Next, the large items are sorted in descending frequency. At last, the database is scanned again to construct the FP-tree according to the sorted order of large items. The construction process is executed tuple by tuple, from the first transaction to the last one. To facilitate tree traversal, an item header table is built in which each item points to its first occurrence in the tree. The Header_Table includes the sorted large items and their pointers (called frequency head) linking to their first occurrence nodes in the FP-tree. If more than one node has the same item name, they are also linked in sequence. Links between nodes are single-directional from parents to children. After all transactions are processed, the FP-tree is completely constructed. This approach constructs a highly compact FP-tree, which is usually substantially smaller than the original database and thus saves the costly database scans in the subsequent mining processes. Moreover it also does not have incremental learning ability.

2.3 THE FUFP ALGORITHM

In real-world applications, transaction databases usually grow over time and the association rules mined from them must be re-evaluated. Some new association rules may be generated and some old ones may become invalid. Conventional batch-mining algorithms solve this problem by reprocessing the entire new databases when new transactions are inserted into original databases. They, however, require lots of computational time and waste existing mined knowledge. The FUP algorithm effectively handles new transactions for maintaining association rules. A fast updated FP-tree (FUFP-tree) [13] structure is proposed, which make the tree update easier as compared to a FP tree algorithm.

FUFP tree construction

An FUFP-tree [13] [14] must be built in advance from the original database before new transactions arrive. The FUFP tree construction algorithm is the same as the FP-tree except that the links between parent nodes and their child nodes are bi-directional. Bi-directional linking will help fasten the process of item deletion in the maintenance process. Besides, the counts of the sorted frequent items are also kept in the Header_Table as like a FP - tree.

Incremental FUFP tree maintenance approach

When new transactions are added, the proposed incremental maintenance algorithm will process them to maintain the FUFP-tree. It first partitions items into four parts according to whether they are large or small in the original database and in the new transactions. Each part is then processed in its own way. The Header_Table and the FUFP-tree are correspondingly updated whenever necessary.

In the process for updating the FUFP-tree [13], item deletion is done before item insertion. When an originally large item becomes smaller, it is directly removed from the FUFP-tree and its parent and child nodes are then linked together. On the contrary, when an originally small item becomes large, it is added to the end of the Header_Table and then inserted into the leaf nodes of the FUFP-tree. It is reasonable to insert the item at the end of the Header_Table because when an originally small item becomes large due to the new transactions; its updated support is usually only a little larger than the minimum support. The FUFP-tree can thus be least updated in this way, and the performance of the proposed maintenance algorithm can be greatly improved. The entire FUFP-tree can be re-constructed in a batch way when a sufficiently large number of transactions are inserted.

2.4 INCREMENTAL MINING TECHNIQUE WITH IMBT STRUCTURE

Incremental Mining Binary Tree (IMBT) [16] is used to record the itemsets in an efficient way. In traditional approaches, the lexicographic tree is used to store the generated candidates and frequent itemsets. In order to efficiently enumerate the support of each itemset, binary tree structure i.e. IMBT is used. Furthermore, this approach does not require to predetermine the minimum support threshold and scans the database only once. This method not only performs incremental data mining more efficiently, but also finds frequent itemsets faster than the Apriori and FP-growth algorithms.

IMBT structure enumerates the support of each itemset in an efficient way after the transactions are added or deleted. Instead of rescanning the database many times to enumerate the support of the itemsets after the database update, it processes a transaction at a time and record the possible itemsets in a data structure that can reduce the processing and IO time.

As already discussed, Apriori and FP-tree algorithm are batch mining algorithms. Apriori algorithm is costly as it needs to handle a huge number of candidate sets, as well as it is tedious to repeatedly scan the database and check a large set of candidates. The FP - tree algorithm is compact and avoids candidate generation-and-test approach. FP-tree is less costly than candidate generation and pattern matching operations performed in most Apriori-like algorithms. With FP-tree search technique is partitioning-based, divide-and conquer method rather than Apriori-like level-wise generation of the combinations of frequent itemsets.

A fast updated FP-tree (FUFP-tree) structure modifies FP-tree construction algorithm for efficiently handling new transactions. An incremental FUFP-tree maintenance algorithm is proposed for reducing the execution time in reconstructing the tree when new transactions are inserted. FUFP-tree maintenance algorithm runs faster than the batch FP-tree construction algorithm for handling new transactions and generates nearly the same tree structure as the FP-tree algorithm.

Incremental Mining Binary Tree (IMBT) is used to record the itemsets in an efficient way. It is not required to predetermine the minimum support threshold and scans the database only once. It also finds frequent itemsets faster than the Apriori and FPgrowth.

The proposed work will mine the incremental data using a data structure called Item-Itemset (I-Is) Tree which is based on the concepts of B+ tree. The proposed work emphasizes on reducing the searching time and space requirement with respect to IMBT.

All tables and figures will be processed as images. You need to embed the images in the paper itself. Please don't send the images as separate files.

3 Proposed System Architecture

In the proposed system, to enumerate the support of each item and itemset, a data structure called Item-Itemset (I-Is) tree is used.

3.1 Incremental Mining Technique with Item-Itemset (I-Is) Tree Structure

***Definition 1* (*I-Is-tree*):** An Item-Itemset tree (or *I-Is-tree* in short) is a tree structure defined below.

1. Root node consists of frequency range defined by the user.
2. All other nodes in the item-itemset tree consists of items or itemsets

The items or itemsets are placed in the tree depending upon its frequency. With this technique, existing dataset is scanned only once, frequency of each item and itemset is calculated and I-Is tree is constructed. When incremental data arrives, the I-Is tree is updated.

The proposed technique executes in two steps. Initially the existing dataset is scanned once, frequency of each item and itemset is calculated and I-Is tree is constructed. This process is shown in Figure 1. When incremental data arrives, already created I-Is tree is updated, which is shown in Figure 2

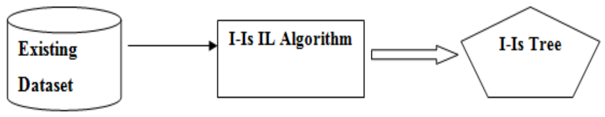


Fig. 1: I-Is tree generated for static data

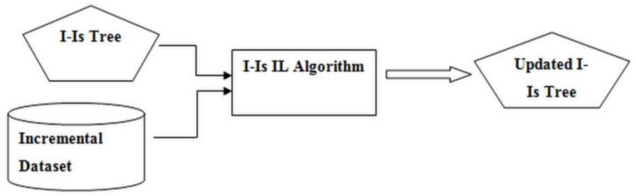


Fig. 2: I-Is tree generated for incremental data

In the process of I-Is tree creation and updation, predetermination of minimum support threshold is not required. I-Is tree helps to find frequent itemsets for specified threshold values after tree creation.

After finding the frequent itemsets, association rule can be enumerated.

4 Detailed Design

In this section the algorithm along with the example is discussed.

I-Is tree is created for the given dataset depending upon the algorithm and the required frequent itemset is searched by traversing the tree. Along with that the incremented data will be inserted in the tree according to the calculated frequency and tree will be updated.

The tree creation function will be dependent upon the frequency calculation. And the frequent itemset searching function will be dependent on tree function whereas incremental data mining will be dependent on the original data mining results.

Algorithm:

STEP 1: Scan the input dataset once.
STEP 2: Calculate the frequency of each item & itemsets
STEP 3: Generate root node of I-Is tree.
STEP 4: Generate remaining I-Is tree from items & itemsets
STEP 5: Take threshold value for support from user and search respected items and itemsets in the tree
STEP 6: For incremental data repeat step 2 and update tree accordingly.
STEP 7: Update I-Is tree generated in STEP 4.
STEP 8: Repeat step 5.

I-Is incremental learning algorithm is used to mine the frequent itemsets form static as well as incremental data. This algorithm scans the original dataset once. The dataset is scanned line by line. All possible items and itemsets in each transaction along with the frequency count are computed.

After processing of all transactions in the original dataset, the actual tree is constructed. Initially the root node is created. The size of the root node depends upon the user. Thus it is dynamic in nature. Depending upon the size of the root node all other nodes in I-Is tree is created with the same node size as that of the root node. As the root node is created, the remaining nodes are created depending upon the frequency count of the computed items and itemsets and frequency range specified in the root node. Once the complete tree for the original dataset is created, depending upon the support value given by the user all frequent items and/or itemsets are searched from the tree.

In incremental data, the original dataset is normally much larger than the incremental database. For incremental data instead of reconstruction of I-Is tree, the existing I-Is tree is updated. The incremental dataset is scanned once and all possible items or itemsets with their frequency count from incremental dataset are generated. If the new item or itemset is already exists in the original I-Is tree, its frequency count is incremented in the tree. Otherwise it is inserted into the tree. In this manner, the tree is updated for incremental data. After tree updation, depending upon the support value provided by the user all frequent items or itemsets are generated.
Pseudo code for the algorithm is as follows:

EXAMPLE:

Step 1: Insert given dataset as input

Tid	Item
001	1 3 4
002	2 3 5
003	1 2 3 5
004	2 5

Step 2: Calculate frequency of each item

Item	Frequency
1	2
2	3
3	3
4	1
5	3

As well as itemset as follows:

Itemset	Frequency
1 2	1
1 3	2
1 4	1
1 5	1
2 3	2
2 4	0
2 5	3
3 4	1
3 5	2
4 5	0

Itemset	Frequency
1 2 3	1
1 2 4	0
1 2 5	1
1 3 4	1
1 3 5	1
1 4 5	0
2 3 4	0
2 3 5	2
3 4 5	0
1 2 3 5	1

Step 3: According to the frequencies create the I-Is tree

Here, root node indicates the range of frequency and the child node contains the items and itemsets according to their frequencies where the items with zero frequency are discarded.

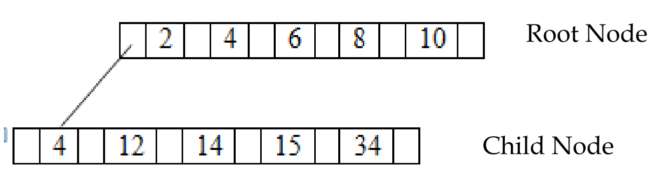
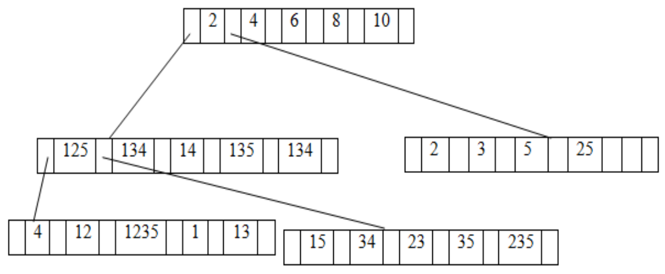


Fig. 3: Initial phase of I-Is Tree for items with frequency up to 20%.

Fig. 4: Final I-Is Tree for the given example



For incremental data step 2 is repeated and according to the calculated frequency of item & itemsets the tree is updated.

5 Expected Results

The proposed work will mine the incremental data using a data structure called Item-Itemset (I-Is) Tree Frequent items or itemsets with user specified support values searched. As this algorithm is incremental in nature new data will be inserted at proper position depending upon its frequency. This technique will minimize searching time and the required space.

5 Conclusion

In this paper, I-Is tree structure is proposed to efficiently and effectively handle the existing dataset and new transaction insertion in data mining. I-Is tree structure is based on the concepts of B+ tree. For existing dataset I-Is tree is created and when new transactions are added, the proposed algorithm processes them to update I-Is tree. It is shown with an example that by applying a proposed technique searching time can be reduced and memory requirement can be handled.

References

[1] Agrawal, R., Imielinksi T., & Swami, A., “Mining association rules between sets of items in large database”, *In the ACM SIGMOD conference, Washington* pp. 207–216, 1993.

[2] Agrawal, R., Imielinksi, T., & Swami, A., “Database mining: A performance perspective”, *IEEE Transactions on Knowledge and Data Engineering*, 5(6), 914–925.1993.

[3] Chen, M. S., Han, J., & Yu, P. S., “Data mining: An overview from a database perspective”, *IEEE Transactions on Knowledge and Data Engineering*, 8(6), 866–883, 1996

[4] Raudel Hernandez Leon, Airel Perez Suarez, Claudia Feregrino Uribe, Zobeida Jezabel Guzman Zavaleta, “An Algorithm for Mining Frequent Itemsets”, *5th International Conference on Electrical Engineering, Computing Science and Automatic Control*, 2008

[5] Zijian Zheng, Ron Kohavi, Llew Mason “Real World Performance of Association Rule Algorithms”, *ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[6] J. Han, J. Pei, Y. Yin, “Mining Frequent Itemsets without Candidate Generation,” *ACM SIGMOD International Conference on Management of Data, 2000*.

[7] Gosta Grahne, Jianfei Zhu, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees", *IEEE Transactions on Knowledge and Data Engineering*, 17(10), October 2005

[8] Qihua Lan, Defu Zhang, “A New Algorithm for Frequent Itemsets Mining Based On Apriori And FP-Tree”, *Global Congress on Intelligent Systems*, 978-0-7695-3571-5/09, 2009

[9] Cheung, D. W., Lee, S. D., & Kao, B., “A general incremental technique for maintaining discovered association rules”. *In Proceedings of database systems for advanced applications*, pp. 185–194, 1997.

[10] Chin-Chen Chang, Yu-Chiang Li Jung-San 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),5,pp.1097-8585, 2005

[11] D. W. Cheung, J. Han, V. T. Ng, C. Y. Wong, “Maintenance of discovered association rules in large databases: An incremental updating approach,” *In The twelfth IEEE international conference on data engineering*, pp. 106–114, 1996.

[12] T. P. Hong, C. Y. Wang and Y. H. Tao, "A new incremental data mining algorithm using pre-large itemsets," *Intelligent Data Analysis*,5(2),pp. 111-129,2001.

[13] T. P. Hong, J. W. Lin and Y. L. Wu, "A fast updated frequent pattern tree", *The IEEE International Conference on Systems, Man, and Cybernetics, pp.2167-2172, 2006.*

[14] Tzung-Pei Hong, Chun-Wei Lin, Yu-Lung Wu, “Incrementally fast updated frequent pattern trees”, *Expert Systems with Applications* 34, pp.2424–2435, 2008

[15] C. W. Lin, T. P. Hong, W. H. Lu, “The Pre-FUFP algorithm for incremental mining”, *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining*,2007

[16] Chia-Han Yang and Don-Lin Yang, “IMBT - A Binary Tree for Efficient Support Counting of Incremental Data Mining”, *Proceedings* 2009 *International Conference on Computational Science and Engineering*,1,pp.324-29,August 2009.

[17] T.SathishKumar, V.Kavitha, Dr.T.Ravichandran “Efficient Tree Based Distributed Data Mining Algorithms for mining Frequent Patterns”, *International Journal of Computer Applications,* 10(1), pp.0975 – 8887, November 2010