IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 2 (Jan. - Feb. 2013), PP 51-55
www.iosrjournals.org

A Novel Approach for Mining Relevant Frequent Patterns in an Incremental Database

J. Mercy Geraldine[1], Anu Disney D[2]
[1] *Head of the Department, Department of CSE, Srinivasan Engineering College, Perambalur, Tamilnadu, India*
[2] *M.E-(Final Year), Department of CSE, Srinivasan Engineering College, Perambalur, Tamilnadu, India*

Abstract: *Frequent pattern mining is emerging as a powerful tool for many business applications such as e-commerce, recommender system and the group decision support system. Many techniques have been developed to mine the frequent patterns. However, it would be the centre of attraction if the degree of importance of each item is taken into consideration.. For this reason, weighted frequent pattern mining algorithms have been suggested. But these algorithms deal only with the static databases, whereas, in reality most databases are interactive and dynamic in nature. Incremental Weighted Frequent Pattern Mining based on Frequency Descending Order ($IWFP_{FD}$) tree is used to deal with the dynamic nature of the databases while pushing the weight constraints into frequent pattern mining. Branch Sorting Method (BSM) with merge sort is used to restructure the $IWFP_{FD}$ tree. This makes it more convenient for mining the patterns from the tree. It also makes the $IWFP_{FD}$ highly compact to save memory space. This tree allows mining of frequent patterns through a single pass over the database.*

Keywords - *Frequent Pattern Mining, Incremental frequent pattern mining, Incremental mining, utility mining, Weighted Frequent Patterns*

I. Introduction

Data mining discovers patterns hidden in data, and associations between the patterns. Data mining tasks such as mining association rules[1], mining correlations, mining closed patterns[2], mining sequential patterns[3] use frequent pattern mining techniques. Frequent pattern mining has been one of the hottest issues in the field of data mining. It is not an easy task to mine the data from the larger datasets. Larger databases require quite a large amount of time to mine the data from the databases[1][3]. It is also well known that frequent pattern mining generates a very large number of frequent itemsets and association rules[1].

Several frequent pattern mining algorithms which run fast than traditional algorithms and create fewer yet more important patterns have been developed. Support measure is used by most of the algorithms, to prune the combinatorial search space. However, support-based pruning is not sufficient while considering the characteristics of real datasets[4]. In addition, there is no way to adjust the number of frequent patterns after mining the data from the datasets. Alternative measures for mining frequent patterns have been suggested to address these issues. The major limitation of the traditional approach for mining frequent patterns is that all items are treated uniformly without considering the actual degree of importance of each item[4]. For example, in an industrial application an expensive item can contribute more to the total revenue even though it does not appear frequently in the transactions. For this reason, weighted frequent pattern mining algorithms are suggested that give different weights to items according to their significance.

Weighted frequent pattern mining concerns satisfying the downward closure property of the databases[5]. It involves defining a weight range and items are given different weights within the weight range. The support and weight of each item are considered for pruning the search space. An ascending weight order tree was used and the tree is traversed in a bottom-up strategy. By setting a weight range and a minimum weight and allowing the user to balance support and weight of itemsets, the number of weighted frequent itemsets can be reduced. In this research, scalable and efficient frequent pattern mining approaches with various weight constraints for the incremental databases has been suggested. Our main approach is to add the weight constraints into the pattern growth algorithm for an incremental database, while maintaining the downward closure property.

Moreover, the databases in reality are not always static. They grow rapidly either by the insertion of new data or by the deletion of the unnecessary data. This does not provide a static nature to the databases. Mining the frequent pattern or the weighted frequent pattern in a incremental/dynamic databases with a considerable time complexity is more desirable.

$IWFP_{FD}$ (Incremental Weighted Frequent Pattern Mining based on Frequency Descending Order) has been proposed for mining incremental patterns. This frequent pattern growth trees require a single database scan. $IWFP_{FD}$ tree makes use of restructuring techniques to make it highly compact to achieve higher space

efficiency. Use of this tree makes it convenient to create the prefix tree and the conditional tree for mining the most important patterns/ frequent patterns. This tree can reuse the mined data for later purpose. This property speeds up the process of mining the frequent/important patterns.

1.1 Paper Organization

This paper is organized as follows: Section 2 narrates the related work. Section 3 discusses about the proposed method.. Section 4 presents experiments of efficiency evaluation. Section 5 gives conclusions and directions for future work.

II. Related Work

Although there has been a notion of weight in the existing algorithms, the combination of weights with the incremental FPM would possibly suit the real time databases. Here we discuss about the existing methods/algorithms for mining frequent patterns, incremental frequent patterns and the weighted FPM.

2.1 Frequent Pattern Mining

Frequent Pattern mining has its origin from the early proposal of the Apriori algorithm[1] by Agarwal, et al. The Apriori algorithm propose the generation of the candidate keys and then find the most frequent patterns in the database[1].However, this suffered from the major disadvantage of candidate generation. In order to overcome this, the FP growth algorithm has been proposed as in [6].This reduces the number of candidates that are generated[6]. It scans the database once and inserts the data into a header table. The further processing are done on the header table rather than on the database. This requires just two scans of the database. The FP growth stores the data in a compact data structure called the FP Growth tree[6][7].These algorithms however apply to the databases that are static in nature, however most of the databases in reality are dynamic in nature.

2.2 Incremental Database Frequent Pattern Mining

The databases are usually updated and grow rapidly. Hence it requires that the updations in the database have to be taken into consideration while mining the data .There has been a number of algorithms that mine the data from the incremental databases.

The Fast Update Algorithm(FUP) as described in[8], scans the original database separately while the incremental datasets are scanned and processed as they arrive. However, this requires multiple scanning when the incremental data is frequent and the same pattern in the original database is infrequent. This lead to the development of New Fast Update algorithm(NFUP)[9], that considers the frequent items in the incremental dataset too. This requires a single scan of the database. However, this does not provide a compact data structure. A CAN tree is a tree that arranges the data in some canonical order[10]. It is a tree based incremental frequent pattern mining approach. It searches for the appropriate position of the data to be inserted and then inserts the data into the position. Another tree based mining techniques called the CAT tree. This does not require to be reconstructed while modifying. It makes use of the already existing and available tree structure[11].

These algorithms are useful in mining the databases that are dynamic in nature. However, these algorithms do not consider the weight i.e, the importance of the item.

2.3 Weighted Frequent Pattern Mining

The weight of the items has to be taken into consideration so that the algorithm can be more effective in real world applications. The weight of the pattern is the average of the weight of the itemsets that constitute the pattern. i.e., the weighted support of the pattern P=(X_1,X_2…..X_n) is defined as,

$$\boldsymbol{Weight(P)} = \frac{\sum_{q=1}^{length(P)} Weight(Xq)}{length(P)} \qquad \ldots \quad (2.1)$$

For example consider Table2.1 and Table 2.2, then the weight of the pattern “ab” is

Weight (ab)= (weight(A) * Weight(B))/ 2 = 0.55

A weighted support of a pattern is defined as the value that results from multiplying the pattern’s support by the weight of the pattern. The weighted support of the pattern, *P*, is given as follows,

$$\mathbf{WSupport_{(P)}} = \mathbf{Support(P) * weight(P)} \quad \ldots (2.2)$$

Table2.1 Transaction Table

TID	
T1	a,b,c,d,g,h
T2	a,e,f
T3	b,e,f,g,h
T4	a,b,c,d
T5	a,b,g,h
T6	a,b,d,e

Table2.2 Header Table

Item	W	F
A	0.6	7
B	0.5	6
C	0.2	3
D	0.3	4
E	0.5	3
F	0.3	2
G	0.8	5
H	0.38	2

There are a number of algorithms that consider the Weight constraints. Different weights are given to items according to their importance or intensity. In contrast to previous frequent pattern mining approaches which are mainly based on support constraints, weighted frequent mining approaches consider not only the frequency but also the importance of patterns. The main focus of weighted frequent itemset mining concerns the downward closure property. The downward closure property is usually broken when different weights are applied to the items. All weighted association rule mining algorithms suggested so far have been based on the Apriori algorithm.

The algorithms like LPMiner[12] mines the frequent patterns based on some support constraints. However, this does not consider the weight of the items. The WLPMiner considers the notion of weight while mining the frequent patterns. The WFIM [13] is the most widely used weighted frequent pattern mining approach. WFIM maintains the downward closure property thus avoiding the least frequent patterns. But WFIM requires two scans of the database to discover the frequent patterns. These Weighted Frequent Pattern Mining algorithms apply only for the static databases. Hence, an efficient algorithm for mining the weighted frequent pattern in an incremental database is essential.

III. The Proposed Method

$IWFP_{FD}$ uses the prefix tree structure for storing the data. A prefix tree is an ordered tree with any predefined order such as the ascending or descending order or any lexicographic order. The items from the transactions are scanned and then inserted into the prefix tree in some predefined order, thus making the storage structure more compact. This helps in reducing the time of mining the most relevant data.

In the case of the $IWFP_{FD}$ tree the data or the transactions are arranged in the descending order of the frequency of the items. Fig 1 depicts that the $IWFP_{FD}$ mining algorithm involves arranging the transactions in a frequency descending order and then creating the tree structure with the ordered data. This is followed by mining the required data from the $IWFP_{FD}$ tree.

3.1 $IWFP_{FD}$ Tree Construction

Creation of the $IWFP_{FD}$ tree marks the beginning of the mining process using the $IWFP_{FD}$ algorithm. $IWFP_{FD}$ is a variant of the enhanced FP tree. The transactions are scanned from the dataset and a header is created to maintain the count or the frequency of each item in the database, which maintains the items in the descending order of their frequency. The data is stored in a max heap with a null node as the root. $IWFP_{FD}$ tree is created in a way similar to creating the enhanced FP growth tree. The creation of the $IWFP_{FD}$ tree involves the following steps,

STEP 1: Scan the transactions in the dataset or the
STEP 2: Increment the count of the item in the header table H and sort the records in frequency descending order
STEP 3:Arrange the items of the transactions in the frequency descending order
STEP 3:Insert the items of the transaction into a prefix tree and ensure that it maintains the downward closure property.

The downward closure property has to be maintained while constructing the $IWFP_{FD}$ tree. Maintaining this property helps to prune the infrequent data or the items .The downward closure property states that if a pattern is infrequent then, all its super patterns are also infrequent. The $IWFP_{FD}$ tree performs the insertion and deletion operations

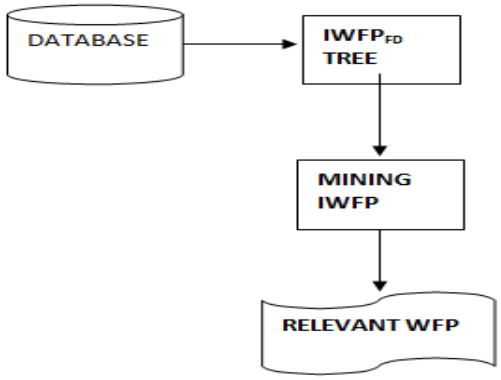

Figure 1 Block diagram of the IWFP$_{FD}$ based mining

3.2 Handling Incremental Data

The incremental insertion is done when a db+ arrives for insertion. The patterns that are arranged in the descending order of the frequencies are added as the leaf nodes of any super pattern that is available in the existing IWFP$_{FD}$ tree. However, before inserting the data or the transactions into the prefix tree or IWFP$_{FD}$ tree, it is required to increase the count of the item in the header table, H. The tree sometimes becomes longer so that the tree looks more complex and it may also require quite a large amount of memory for storing the data, i.e, the tree becomes less compact. Hence a restructuring technique is to be suggested to regain the compactness of the tree structure.

The incremental deletion is performed when a database, db$^-$ arrives for the deletion of the data. The deletion of the data is simple when compared to that of the insertion procedure.

3.3 Restructuring Using Branch Sorting Method(BSM)

Items can be retrieved from prefix tree with an Incremental Weighted Frequent pattern Mining algorithm. The trees grown by IWFP$_{FD}$ sometimes have some branches longer than others; hence, it is possible to reduce the mean retrieval time by restructuring the prefix tree to make the branches as uniform in length as possible. Branch Sorting Method is used to perform the restructuring of the modified IWFP$_{FD}$ tree.

The trees needed to be ordered so that the mining process is simplified and the expected frequent patterns are obtained in an average retrieval time. BSM also makes the structure of the tree compact, thus helping to reduce the amount of memory required for storing the items of the transaction. BSM makes use of the merge sort method to restructure the tree in descending order.

For simplicity of discussion, assume that the initial and restructured trees are termed as T_{init} and T_{sort} respectively. The BSM involves an array-based technique that restructures all branches, one-by-one, from the root of T_{init} . Each sub-tree is treated as a separate branch. Hence, T_{init} contains as many branches as the number of children it has. Each branch may consist of several paths and several branching nodes. While restructuring a branch, BSM sorts each and every path in the branch of the tree, according to the order by removing it from the tree, sorting it into a temporary array, and then merging it to form a tree in the sorted order. When all the branches in T_{init} are processed, the restructuring process gets terminated resulting in the final T_{sort}.

3.4 Mining Using IWFP$_{FD}$ Algorithm

The mining of the data using the IWFP$_{FD}$ tree requires the IWFP$_{FD}$ tree as the input. The IWFP$_{FD}$ algorithm makes use of two other manual inputs: the user defined threshold value and the pattern to be mined. The mining process in the IWFP$_{FD}$ algorithm commences with getting the user threshold along with the pattern from the user and the IWFP$_{FD}$ created. The entire IWFP$_{FD}$ mining algorithm is performed on the IWFP$_{FD}$ tree rather than mining the database. Initially IWFP$_{FD}$ generates a prefix tree for the item or the itemset to be mined. The items in the prefix tree are tested against the threshold using a candidate test. The weighted support of the pattern to be mined is compared against the threshold value provided by the user. This paves way to mine the relevant pattern and to prune the irrelevant pattern from being mined. The items that pass the candidate test generation is form the nodes of the conditional tree. The frequent patterns in the frequency descending order are obtained by investigating the conditional tree.

V. Experimental Evaluation

The IWFP$_{FD}$ using BSM makes use of the concept of enhanced FP and BSM for restructuring the data. This algorithm can considerably reduce the memory required for storing the data and can be more efficient in retrieving the relevant patterns as it considers the notion of weight. The performance of this algorithm can be evaluated based on the memory and the runtime efficiency.

4.1 Memory Usage

The $IWFP_{FD}$ normally uses a tree based data structure for storing the data. This makes the storage size to be more compact. The $IWFP_{FD}$ tree is created over a single scan of the database. The incremental database is scanned and added at the appropriate position using the tree restructuring, thus making the structure compact and reducing the time of merging each and every data at its position.

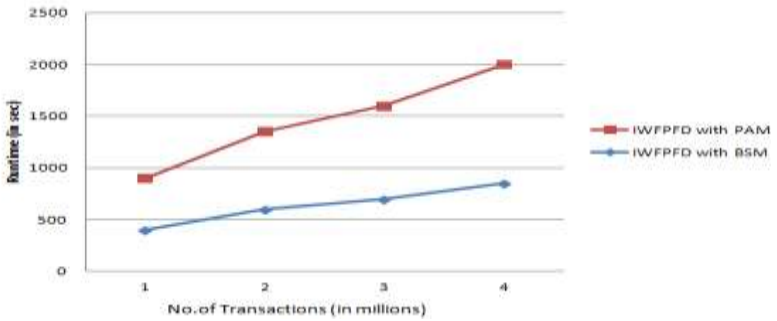

Figure 2 Database is increasing db^+=0.2 million transaction

4.2 Runtime Efficiency

The $IWFP_{FD}$ tree is based on the descending order of the frequency. The runtime of the $IWFP_{FD}$ tree considers the time for the creation of the tree, time for restructuring of the tree on the arrival of the incremental datasets and the time for mining the patterns from the tree. In this case considered here the incremental database db^+ contains 0.2 million data and it is seen from Fig 2 that the $IWFP_{FD}$ takes considerably lesser time than that of the $IWFP_{WA}$(based on Weight Ascending order). Fig 2 shows the runtime required for mining the data using the $IWFP_{FD}$. Moreover, using BSM for restructuring reduces the time of restructuring.

VI. Conclusion

In this paper, it is studied that the $IWFP_{FD}$ with BSM restructuring allows the user to mine the frequent patterns over a single scan, while the earlier existing algorithms requires more number of scans. Moreover, this tree throws a limelight over the weight constraint, which can be more beneficial when compared to the simple frequent pattern mining algorithm or techniques. It is seen that using BSM for restructuring the $IWFP_{FD}$ tree is more advantageous than using the PAM. The implementation of the IWFP tree has shown that it can be more time efficient in mining the data. It is also seen that the IWFP requires a considerably reduced amount of storage than the existing tree structure. The efficiency can be further improved by taking the weight of the item into consideration even while storing the data i.e., considering the weight of the item at the time of creating the tree.

References

[1] Aditya Telang, Chengkai Li, and Sharma . Agrawal, T. Imielinski, and A. Swami, (1993), "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.
[2] R. Agrawal, and R. Srikant,(March 1995), "Mining Sequential Patterns," Proceedings of the Eleventh International Conference on Data Engineering, p 3-14.
[3] R. Agrawal, and R. Srikant,(1994), "Fast Algorithms for Mining Association Rules in Large Databases," Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB'94), pp. 487-499.
[4] C.F. Ahmed, S.K.Tanbeer, Y.K.Lee, (2012), "Single Pass Incremental and Interactive Mining For Weighted Patterns",Expert System with applications.
[5] C. H. Cai, A. W. Chee Fu, C. H. Cheng, and W. W. Kwong, (July 1998), "Mining Association Rules with Weighted Items," Proceedings of the Sixth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2005).
[6] J. Han, J. Pei, and Y. Yin,(May 20000, "Mining Frequent Patterns without Candidate Generation," Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 1-12.
[7] J. Han, J. Pei, Y. Yin, and R. Mao,(Jan 2004), "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," Data Mining and Knowledge Discovery, vol. 8, pp. 53-87.
[8] W. Cheung , "Frequent Pattern mining without candidate generation or support constraint." Master's thesis, University of Alberta, SPRING '03, doi.ieeecomputersociety.org/10.1109/IDEAS.2003.1214917.
[9] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, (Feb 1996), "Maintenance of discovered association rules in large databases: an incremental updating technique," In Proc. 12th Intl. Conf. on Data Engineering, New Orleans, LA, pp. 106-114.
[10] Carson Kai-Sang Leung, Quamrul I. Khan, Tariqul Hoque,(2005), "CanTree: a tree structure for efficient incremental mining of frequent patterns", ICDM'05
[11] F Gharib Tarek, Hamed Nassar, Mohamed Taha , Ajith Abraham, (2010) "AN EFFICIENT ALGORITHM FOR INCREMENTAL MINING OF TEMPORAL ASSOCIATION RULES", IEEE Transaction on Data and knowledge Engineering.
[12] M. Seno, and G. Karypis, (Nov/Dec 2001) "LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint," *Proceedings of the First IEEE International Conference on Data Mining (ICDM 2001)*, pp. 505-512.
[13] U. Yun, and J. J. Leggett, (April 2005), "WFIM: Weighted Frequent Itemset Mining with a Weight Range and a Minimum Weight," Proceedings of the Fourth SIAM International Conference on Data Mining, pp. 636-643.