Volume No.03, Special Issue No. 02, February 2015 ISSN (online): 2348 – 7550

CLUSTERING BASED INFREQUENT WEIGHTED ITEMSET MINING

Kalaiyarasi. P[1], Manikandan. M[2]

[1]PG Scholar, [2]Assistant Professor, Department of Computer Science and Engineering,

Adhiyamaan College of Engineering, Hosur, (India).

ABSTRACT

Frequent item set mining is one of the popular data mining technique and it can be used in many data mining fields for finding highly correlated itemsets. Infrequent itemset mining finds rarely occurring itemsets in the database. Most of the Existing Infrequent itemset mining techniques finds infrequent weighted itemsets with high computing time and are less scalable when the database size increases. The proposed system uses clustering or logical grouping concepts for finding infrequent weighted itemsets. The proposed algorithm works well with real-time databases and is highly scalable which suits for real-time applications.

Keywords : Clustering, FP-Growth, Infrequent itemsets, Performance, Scalability.

I. INTRODUCTION

In Data Mining, Association Rule mining(ARM)[1] is one of the popular technique used to find the correlation between the data items in the database based on some statistical measures but not considering the interesting of the business users. ARM is one of the oldest technique in data mining. The goal of ARM is to find the relationship, correlation among different data sets in the database. Frequent itemset mining is an exploratory data mining technique widely used for discovering valuable correlations among data. Frequent itemsets mining is a core component of data mining and variations of association analysis, like association rule mining. Infrequent itemsets are produced from very big or huge data sets by applying some rules or association rule mining algorithms like Apriori technique, that take larger computing time to compute all the frequent itemsets. Extraction of frequent itemsets is a core step in many association analysis techniques. The frequent occurrence of item is expressed in terms of the support count. However, significantly less attention has been paid to mining of infrequent itemsets, but it has acquired significant usage in mining of negative association rules from infrequent itemset, fraud detection where rare patterns in financial or tax data may suggest unusual activity associated with fraudulent behavior, market basket analysis and in bioinformatics where rare patterns in microarray data may suggest genetic disorders. Several frequent item set mining including Apriori, FP-Growth algorithm, AFOPT algorithm, NONORDFP algorithm, FP_Growth* algorithm, Broglet's FP-Growth, DynFP-Growth algorithm, Enhanced FP-Growth algorithm, IFP_min Algorithm and Transaction mapping algorithm were proposed. Clustering involves logical grouping of similar items into a single group. The proposed work mainly focus on incorporating clustering concepts and some strategies for finding infrequent itemsets. The proposed algorithm works in two phases, first it groups the data items then it prunes the frequent itemsets in

order to obtain infrequent itemsets.

II. RELATED WORK

The frequent pattern mining problem is to discover the complete set of all patterns contained in at least a specified support threshold λ, of transactions in the transaction database. FP-Growth-like algorithms adopt divide-and-conquer method which can be stated as follow: First, it compresses the database representing frequent items into a frequent-pattern tree, or FP-tree, which retains the itemset association information. It then divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or "pattern fragment," and mines each such database separately[5]. FP-Growth opened up a new way to efficiently mine frequent pattern. However, its low time and space utilization, still need to improve. Many variants of FP-Growth appeared recently. The representations include FP-Growth*[6] proposed by G.Grahne et al. in 2003 and AFOPT[7][8] proposed by G.Liu et al. at the same year. FP-Growth* adopts a new array technique to enhance the operation capability. It constructs a two-dimensional array at the same time of building FP-Tree to preserve the support counts of all 2-itemsets. By using this array FP-Growth* only need scan FP-Tree once at the time of each recursion, which improve the efficiency of FP-Growth. AFOPT construct a sample and compact data structure Ascending Frequency Ordered Prefix-Tree (AFOPT) based on FP-Tree. The algorithm adopts top-down scan to reduce the number of conditional databases and depress the overhead of scanning each conditional database. We have studied on the FP-Growth-like algorithm for a long time and already get some achievements, such as F-Miner in [9] and LPS-Miner in [10].

III. EXISTING SYSTEM

1.1 Apriori Algorithm

Apriori[2] was the first proposed algorithm in association rule mining, to identify the frequent itemsets in the large transactional database. Apriori works in two phases. During the first phase it generates all possible Itemsets combinations. These combinations will act as possible candidates. The candidates will be used in subsequent phases. In Apriori algorithm, first the minimum support is applied to find all frequent itemsets in a database and Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

Apriori Algorithm:

```
procedure Apriori (T, minSupport)
{
L1= {frequent items};
for (k= 2; Lk-1 !=∅; k++) {
Ck= candidates generated from Lk-1 for each transaction t in database
do
{
Lk = candidates in Ck with minSupport
}
}
```

return Uk Lk ;

}

The main drawback of Apriori is the generation of large number of candidate sets. The efficiency of apriori can be improved by Monotonicity property, hash based technique, Partioning methods.

1.2 FP-Growth Algorithm

The drawback of Apriori can be improved by Frequent pattern Growth algorithm[3].This algorithm is implemented without generating the candidate sets. This algorithm proposes a tree structure called FP tree structure, going to collect information from the database and creates an optimized data structure as Conditional pattern. Initially it Scans the transaction database DB once and Collects the set of frequent items F and their supports and then Sort the frequent itemsets in descending order as L, based on the support count. This algorithm reduces the number of candidate set generation, number of transactions, number of comparisons.

Algorithm:

Input:

-A transactional database *DB* and a minimum support threshold ξ.

Output:

- frequent pattern tree, FP-tree /*phase1: */

[1] Scan the transactional database.

[2] Collect the set of frequent items *F* and their supports. Sort *F* in support descending order as *L*.

/* the *list* of frequent items*/

[3] Create the root of an FP-tree, *T*, and label it as "root" /* for each transaction do */

[4] Select and sort the frequent items in *Trans* according to the order of *L*.

[5] perform the insert_tree function

/* call insert_tree function recursively */ /*phase2: */

Input:

An FP-tree constructed in the above algorithm, *D* – transaction database;

s – minimum support threshold. Output:

The complete set of frequent patterns.

1.call the FP_Growth function

2. check if the tree has a sigle path,

3. then for each combination (denoted as *B*) of the nodes in the path *P* do

4. generate pattern B ∪ A with support=minimum support of nodes in B

5. else

6. construct B's conditional pattern base and B's conditional FP-tree.

7. call the FP_Growth function

8. check if the tree has a sigle path,

9. then for each combination (denoted as *B*) of the nodes in the path *P* do

10. generate pattern B ∪ A with support=minimum support of nodes in B

11. else
12. construct FP-Tree
13. construct B's conditional pattern base and B's conditional FP-tree.

1.3 AFOPT Algorithm

Liu et al[4] investigated the algorithmic performance space of the Fpgrowth algorithm. AFOPT algorithm uses dynamic ascending frequency order for both the search space exploration and prefix-tree construction, it uses the top-down traversal strategy. AFOPT algorithm utilizes dynamic ascending frequency for the item search space ,adaptive representation for the conditional database format, physical construction for the conditional database construction, and top-down traversal strategy for the tree traversal. The dynamic ascending frequency search order can make the subsequent conditional databases shrink rapidly. As a result, it is useful to use the physical construction strategy with the dynamic ascending frequency order.

1.4 Transaction Mapping Algorithm

The transaction tree is similar to FP-tree but there is no header table or node link. The transaction tree has compact representation of all the transactions in the database. Each node in tree has an id corresponding to an item and a counter

that keeps the number of transactions that contain this item in this path. Here we can compress transaction for each itemset to continuous intervals by mapping transaction ids into a different space to a transaction tree. Advantage of this algorithm is the performance can be improved compared to FP-Growth, FP-Growth* algorithms.

Algorithm:

Input: -Databse DB Output:

-all infrequent item sets

[1] scan the database and identify the infrequent item sets.

[2] construct the transaction tree with the count for each node.

[3] Construct the transaction interval lists.

[4] Construct the lexicographic tree in a depth first order keeping only the minimum amount of information necessary to complete the search.

1.5 The Infrequent Weighted Itemset Miner Algorithm

IWI Miner is a FP-growth-like mining algorithm that performs projection-based itemset mining. Hence, it performs the main FP-growth mining steps: (a) FP-tree creation and (b) recursive itemset mining from the FPtree index. Unlike FP-Growth, IWI Miner discovers infrequent weighted itemsets instead of frequent (unweighted) ones. To accomplish this task, the following main modifications with respect to FP-growth have been introduced: (i) A novel pruning strategy for pruning part of the search space early and (ii) a slightly modified FP-tree structure, which allows storing the IWI-support value associated with each node.

Algorithm (IWI Miner(T,E))
Input:
-T, a weighted transactional dataset Input:
-E, a maximum IWI-support threshold Output:
-F, the set of IWI satisfying E 1. F=0 /*Initialization*/
/*scan T and count the IWI-support of each item */

1. count the infrequent weighted item sets with the support value.
2. create header table which is a data structure which holds information about total weight values.
3. for each transaction, create equivalent transaction.
4. create an FP-Tree, for each transaction.
5. Iterate the process until all transactions are traced.
6. create conditional pattern base calculate weight value.
7. obtain the infrequent item sets.

To reduce the complexity of the mining process, IWI Miner adopts an FP-tree node pruning strategy to early discard items (nodes) that could never belong to any itemset satisfying the IWI-support threshold. Hence, an item(i.e., its associated nodes) is pruned if it appears only in tree paths from the root to a leaf node characterized by IWI-support value greater than E.

1.6 HPFP Miner Algorithm

HPFP Miner algorithm is implemented by creating a tree structure where the infrequent itemsets are mined and compared with the threshold value and a tree is constructed, where it uses pruning techniques which reduces the communication overheads.

Input: Mininmum threshold value MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid); numprocs-=1;
MPI_Recv(message,length,MPI_INT,0,99,MPI_COMM_WO RLD,&status);
insertNode(message,length,numprocs);
sign=numItem/numprocs;

for(k=0;k<sign;k+=1)

{

i=k*numprocs+myid; for(j=numItem-i-1;j>=0;j-=1) prune(root+k,j,numItem-i-1);
/*prune infrequent nodes in HPFP-Tree*/ merge(root+k,numItem-i-1);
/*merge pruned HPFP-Tree */ delete_tree(k,numItem-1); /*release memory*/
}

Volume No.03, Special Issue No. 02, February 2015 ISSN (online): 2348 – 7550

IV. PROPOSED SYSTEM

The proposed Algorithm is based on the concept of clusters to find infrequent itemsets. A cluster is a logical grouping of similar or closely resembling itemsets into a single group.

The proposed Algorithm is given as follows:

C1= promising infrequent items

C2= non-promising infrequent items

C3=frequent itemsets.

Step 1: initialize three clusters

Step 2: calculate candidate-1 itemsets and their correlation Values.

Step 3: if (Candidate(i).value>support value)

{

Add Candidate(i) to C3

}

Step 4: else if(Candidate(i).value lies between {supp,supp/2}

{

Add Candidate(i) to C2

}

Step 5: else

{

Add Candidate(i) to C1

}

Step 6: members in C3 are pruned.

Step 7: members in C1 are infrequent itemsets of first iteration Step 8: copy C1 to C2

Step 9: copy C2 to C3

Step 10: generate next level candidates and goto Step 3.

Cluster level determines the promising and non-promising.

V. CONCLUSION

In this paper a novel algorithm is presented for mining infrequent weighted itemsets form large real-time databases. The proposed algorithm uses simple logical grouping of data items using some strategies in order to prune frequent itemsets and to find infrequent weighted itemsets. our proposed algorithm scales well and forms a non-linear curve i.e., even when the number of transactions or number of distinct items increases the computing time varies slightly, unlike previous algorithms which shows a linear variation between computing time and performance parameters copmapred to the existing work which involves complex tree data structure which has been overcome by use of clustering techniques which scales well and performance has been improved with time and space complexity.

REFERENCES

[1] Agrawal , R. , Imieliński , T. , & Swami , A.”Mining association rules between sets of items in large databases”.In proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington, DC, 1993.

[2] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” Proc. 20th Int’l Conf. Very Large Data Bases (VLDB ’94), pp. 487-499, 1994.

[3] Luca Cagliero and Paolo Garza “Infrequent Weighted Itemset Mining using Frequent Pattern Growth”, IEEE Transactions on Knowledge and Data Engineering, pp. 1- 14, 2013.

[4] Liu,G. , Lu ,H. , Yu ,J. X., Wang, W., & Xiao, X.. ”AFOPT:An Efficient Implementation of Pattern Growth Approach”, In Proc. IEEE ICDM'03 Workshop FIMI'03, 2003.

[5] J.Han, M.Kamber. Data Ming Concepts and Techniques, Second Edition. Morgan Kaufmann Publisher, Aug. 2000.

[6] G.Grahne, J.Zhu. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In ICDM'03, 2003.

[7] G.Liu, H.Lu, W.Lou, et al. Efficient Mining of Frequent Patterns Using Ascending Frequency Ordered Prefix-Tree. In DASFAA, 2003.

[8] G.Liu, H.Lu, J.X.Yu, et al. AFOPT: An Efficient Implementation of Pattern Growth Approach. In ICDM'03, 2003.

[9] X. Chen, L. Li, Z. Ma, et al. F-Miner: A New Frequent Itemsets Mining Algorithm. In ICEBE06, pp446-472, 2006

[10] X.Chen, H.Liu, et al. A High Performance Algorithm for Mining Frequent Patterns: LPS-Miner. In Information Science and Engieering, 2008. ISISE '08, 20-22 Dec. 2008: pp7-11

[11] Cornelia Gyorodi, Robert Gyorodi, T. Cofeey & S. Holban – ”Mining association rules using Dynamic FP-trees” – în Proceedings of The Irish Signal and Systems Conference, University of Limerick, Limerick, Ireland, 30th June-2nd July 2003, ISBN 0-9542973-1-8, pag. 76-82.

[12] Grahne G. and Zhu J., “Efficiently Using Prefix-Trees in mining Frequent Item sets,” Proc. ICDM 2003Workshop Frequent Item set Mining Implementations, (2003).

[13] A.Gupta, A. Mittal, and A. Bhattacharya, “Minimally Infrequent Itemset Mining Using Pattern-Growth Paradigm and Residual Trees,” Proc. Int’l Conf. Management of Data (COMAD), pp. 57-68, 2011.

[14] J. Han, J. Pei, and Y. Yin, ”Mining frequent patterns without candidate generation,” Procedings of ACM SIGMOD Intnational Conference on Management of Data, ACM Press, Dallas, Texas, pp. 1-12, May 2000.

[15] Han, J. , Pei, J. , & Yin, Y. “Mining frequent patterns without candidate generation”. In Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD ’96), Page 205-216,2000.