International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

Incremental Frequent Pattern Mining using Graph based approach

Sanjay Patel* and Dr. Ketan Kotecha**
*: Vishwakarma Government Engineering College, Ahmedabad, Gujarat, India.
Email: sanjaypatel.ce@gmail.com
**: Institute of Technology, Nirma University, Ahmedabad, Gujarat, India.
Email : drketankotecha@gmail.com

ABSRACT

Extracting useful information from huge amount of data is known as Data Mining. It happens at the intersection of artificial intelligence and statistics. It is also defined as the use of computer algorithms to discover hidden patterns and interesting relationships between items in large datasets. Candidate generation and test, Pattern Growth etc. are the common approaches to find frequent patterns from the database. Incremental mining is a crucial requirement for the industries nowadays. Many tree based approaches have tried to extend the frequent pattern mining as an incremental approach, but most of the research was limited to interactive mining only. Here, instead of tree based approach, graph based approach is presented which also gives good results for incremental mining.

KEYWORDS

Data Mining, Frequent Pattern Mining, Knowledge Discovery, Artificial Intelligence, Ant Colony Optimization

I. INTRODUCTION

Industries suffer the problem of huge amount of data. It is very difficult to find out which information is important and useful in day to day life. Researchers are trying to solve the above said problem by the method known as data mining. Real world databases are highly susceptible to noise, incomplete and inconsistent data. These types of databases result in low quality mining results. To improve the quality of the databases, some preprocessing is required. This includes Data Cleaning, Data Integration, Data Selection and Data Transformation. Data cleaning is just to detect errors and rectify it. In data integration multiple data sources may be combined. Data selection is just to select the data which is required for the analysis task. To transform the data into the appropriate form of the mining procedure is known as data transformation. Data Mining can be applied on different kinds of databases like relational databases, data warehouses, transactional databases, object relational databases, temporal databases, sequence databases, time series databases, spatial databases, spatiotemporal databases etc. There are main three data mining techniques: (1) association Rule Mining (2) classification and (3) clustering. A procedure so called mining frequent patterns is a foundation step of association rules discovery. Most famous example of Association Rule Mining is Market basket analysis, which shows association of items purchased by the customer. Such information can be used to make projections, for example designing weekly catalog, customer segmentation, cross-marketing etc. Frequent pattern mining for large databases of business data, such as transactional records, is of great interest. So many algorithms have been proposed in this area. All of the algorithms are having some limitations that will be discussed later.

Well known algorithm for the data mining is the Apriori Algorithm. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. It is based on the downward closure property of itemset that if an itemset of length k is not frequent, none of its superset patterns can be frequent. So many variations proposed to improve the efficiency of the Apriori algorithm; they are as 1) Sampling, 2) Transaction reduction, 3) Hash-based techniques, 4) Dynamic Itemset Counting.

The Candidate Generation and Test is costlier approach as growth in the size of the database.

Pattern growth is the concept on which all the second class algorithms are designed. This concept is based on tree structure. The first algorithm based on the pattern growth approach is the FP-Growth algorithm which generates frequent patterns without candidate generation. FP-growth algorithm has an extended prefix-tree structure for storing compressed, crucial information about frequent patterns for mining the complete set of frequent patterns by pattern fragment growth. It requires two database scan. When there is a change in the database, the mining procedures have to be started from the scratch. Association rule mining works on the trial and error basis. One cannot predict the required minimum support directly. If the minimum support is changed than also, the mining procedure has to be started from the scratch. This is a time consuming procedure.

It is possible that non frequent items become frequent items only due to addition of some transaction in the database. At that time the mining procedure has to be started from the scratch in the FP-growth algorithm. To avoid this problem, William C. et al. (2003), proposed a new tree structure which is known as CATS (Compressed Arranged Transaction Sequence) tree and the algorithm for the mining is known as the FELINE Algorithm. CATS Tree allows all the items to be stored in the structure either it is frequent or non frequent. This will occupy more amount of memory because all the items need to be stored. FELINE algorithm is useful for interactive mining. For the same database, if the values of the minimum support are changed, then also it must give the result without starting the whole procedure from scratch, this is known as interactive mining. The detail algorithm for the tree construction and the mining algorithm are available in Cheung W. et al., 2002.

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

The compactness of CATS tree and FELINE algorithm mainly depends on the sequence of the items. The CATS tree and FELINE algorithm stores all the items in the tree as the order is available in the transaction. No specific ordering is required. Considering CATS Tree as the base tree Q.I. Khan et al. (2005) proposed CanTree (Canonical order Tree) which describes a specific ordering of items. The ordering can be based on lexicographic or number. The description of the algorithm as well as the mining procedure in detail will be discussed in the related work section. Whatever different methods that have been discussed are tree based. H.D.K. Moonesinghe et al. (2006) proposed graph based approach which is known as PGMiner. This graph based concept works basically on the FP-Tree and FP-Growth approach. In this approach the first scan of the database finds the frequent items. On the base of the first scan of the database, graph will be formed which includes only frequent items. Again it creates the same problem as FP-growth, means the mining procedure has to be started from the scratch. After that Vivek Tiwari et al. (2010) proposed another graph based approach for association rule mining. It works mainly on three phases: The first is to generate the graph including all items. The next phase is for finding the frequent items. In the last phase, to remove the items which are not frequent and readjust all the links. It faces the same problem as PGMiner.

The contribution of this work is the representation of the database in the form of graph which also works for incremental mining. Suppose already analyzed data for one year is available with results. After 6 months if there is a need to combine the results of these 6 months with the previous 1 year data than also it is possible without starting the procedure from the scratch.

This paper is organized as follows. Section II discusses about related work for the frequent pattern mining. Section III discusses the representation of the database in the form of graph for incremental mining. Section IV shows the Conclusion with future scope.

II. RELATED WORK

Association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. It is required to find the frequent patterns for the association rule mining. Here in this section some of the interesting methods to find the frequent pattern mining are discussed.

A. Apriori based Algorithms

The first ever known algorithm for frequent pattern mining is Apriori. The algorithm works on the principle of anti monotone property means if a set cannot pass a test, all its super set will fail the same test as well. The main principle of Apriori algorithm is the candidate generation and test. The drawbacks of the algorithm are identified as follows (Jiawei Han et al., 2008).

1. It is costly to handle huge number of candidate sets. For, example if there are 10^5 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^9 length-2 candidates and test their occurrence frequencies.
2. There is a need to scan the database frequently which is a difficult task.

In the past several years, many improvements over the Apriori algorithm have been proposed. Here, we review some proposals (Jiawei H. et al., 2008).

1. **Sampling:** Basic idea is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D. The sample size of S is such that the search for frequent itemsets in S can be done in main memory.
2. **Transaction reduction:** A transaction that does not contain any frequent k-itemsets can not contain any frequent (k+1) itemsets. Therefore, such a transaction can be removed from further consideration.
3. **Hash-based technique:** A hash-based technique can be used to reduce the size of the candidate k-itemsets, C_k, for k>1.

   For example, when scanning each transaction in the database to generate the frequent 1-itemsets, L_1, from the candidate 1-itemsets in C_1, we can generate all of the 2-itemsets for each transaction, hash them into the different buckets of a hash table structure and increase the corresponding bucket counts.
4. **Dynamic itemset counting**: To divide the database into blocks marked by start points. New candidate itemsets can be added at any start point, which determines new candidate itemsets only immediately before each complete database scan. The resulting algorithm requires fewer database scan than Apriori.

B. Pattern Growth Methods

In this method, the database is represented in the form of the tree which retains the itemset associate information and after that divides the database into a set of conditional databases. There are several pattern growth methods for efficient frequent pattern mining like FP-Growth, H-Mine, FreeSpan, PrefixSpan etc., In the FP-Growth approach; there is a need of an effective data structure called FP-tree to store the complete but no redundant information for frequent pattern mining. Since only the frequent items play an important role in the frequent pattern mining, it is necessary to perform one scan of the transaction database to identify the set of frequent items. After the frequent patterns found, the tree will be formed containing only the frequent patterns. The detailed procedure for the tree construction is available in Jiawei Han et al.,

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

(2008). When the FP-tree formed then mining procedure starts. It is required to construct conditional pattern base then small FP-tree for the same. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree. The detailed mining procedure with example is also available in Jiawei Han et al., (2008).

Although FP-growth is more efficient than Apriori in many cases, it may still face problems in some cases as mentioned below (Jien Pei, 2002):

1. Large applications need more scalability. Many existing methods are efficient when the data set is not very large.
2. Not suitable for interactive as well for incremental mining.
3. Database contains all the cases. Real data sets can be sparse or dense in different applications.

The next pattern growth method is H-mine where space requirement is small even for very large databases. One major feature of H-mine is that it is moderate in memory usage. It can fully utilize all available memory space as well as it performs well even with small main memory. Are all the patterns generated are interesting? It does not provide any facility like constraint based mining. Details regarding H-mine algorithm with example is given in Jian Pei et al.(2002). Other approaches for pattern growth methods are FreeSpan (Frequent Pattern – Projected Sequential Pattern Mining) and PrefixSpan (Prefix Projected Sequential Patterns Mining). FreeSpan creates projected databases based on the current set of frequent patterns, whereas PrefixSpan does so based on frequent prefixes only. FreeSpan adopts the heuristic to mine sequential patterns by partitioning search space and projecting sequence sub databases recursively. PrefixSpan partitions the problem recursively. That is, each subset of sequential patterns can be further divided when necessary. To mine the subsets of sequential patterns, PrefixSpan constructs the corresponding projected databases.

C. CATS Tree and FELINE Algorithm

The basic idea behind the CATS Tree is to store all the items such that for different values of minimum support, no need to start the mining procedure from scratch. CATS Tree extends the idea of FP-Tree to improve the storage compression and allow frequent pattern mining without generation of candidate itemsets. The differences between FP-tree and Compressed Arranged Transaction Sequences (CATS) Tree are discussed here.

Compressed Arranged Transaction Sequences (CATS) Tree contains all items from every transaction. FP-Tree contains only frequent items which are based upon the minimum support value. Single scan of the database is supported by CATS Tree, as compared to FP-Tree which supports two scan of the database. In CATS Tree sub-trees are locally optimized to improve compression, while sub-trees are locally optimized in FP-Tree. Items within a transaction do not need to be sorted in CATS Tree while items within a transaction are sorted in FP-Tree.

A CATS tree is a prefix tree that contains all components of the FP-Tree that include a header and item links. Each item in the dataset has a node in the header and each of them consists of the total frequency of the item in the dataset. In addition, each header node contains a pointer that points to the first node in the CATS tree having the same label as that of the header node.

The CATS Tree satisfies the following properties (Q. I. Khan et al., 2005).

1. No item of the same kind, i.e., nodes containing the same item label, could appear on the lower right hand side of that level item.
2. The compactness of CATS Tree measures how many transactions are compressed at a node. The compactness of CATS Tree is the highest at the root and the compactness decreases as a node is further away from the root.

The algorithm for the CATS Tree builder is briefly discussed in (William C. et al., 2003).The FELINE algorithm takes the CATS Tree as an input and mines the frequent items according to the algorithm given in (Cheung W. et al, 2002). However the CATS Tree and FELINE algorithm suffers from the following problems (William C. et al., 2003):

1. Too much expensive to build the tree with all items.
2. Swapping will take more times than the normal FP-Tree nodes.

D. CanTree and Its Variants

CanTree (Canonical –order tree) arranges tree nodes according to some canonical order, which can be determined by the user prior to the mining process or at runtime during the mining process. Items in the CanTree can be consistently arranged in lexicographic order. It satisfies the following properties (Carson Kai-Sang Leung et al. ,2007).

1. Items are arranged according to a canonical order, which is a fixed global ordering.
2. The ordering of the items is unaffected by the changes in frequency caused by incremental updating.
3. The frequency of a node in the CanTree is at least as high as the sum of the frequencies of all its children.

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

Although items are arranged according to a fixed global ordering, CanTree maintains its structural integrity. Due to this property, transactions can be easily added to the CanTree without any extensive searches for mergeble paths. Once the CanTree is constructed, frequent patterns from the treecan be mined in the similar fashion to the FP-growth algorithm. It is advantageous in the case of handling deletions of transactions. The basic goal of CanTree is to provide incremental mining whereas CATS Tree is for interactive mining. Memory and time complexity is the main issues in the case of CanTree.

To reduce the memory requirements, one of the variant proposed of CanTree is CanTries. The structure of the CanTrie is quite similar to that of the CanTree , except that nodes along the same path are combined into a mega node if they have the same frequency. Experiences show that it helps to save the memory up to 50%.

E. FP_Graph Based Approach

This method can be divided in three parts: To make a graph for the given database. In the second part, remove all the non frequent nodes and readjust all the links, and in the third part frequent itemsets are mined from the pruned graph (Vivek Tiwari et al., 2010). The detailed algorithm and the mining procedure with example is available in Vivek Tiwari et al., 2010. Experiments on different datasets show that under large minimum supports, FP-Growth runs faster than FP-Graph while running slower under large minimum supports (Vivek Tiwari et al., 2010). For many real world applications however, utility of itemsets based on cost, profit or revenue is of importance. The utility mining problem is to find itemsets that have higher utility than a user specified minimum. The details of utility itemset mining are available in Alwa Erwin et al., 2007.

III. CONTRIBUTION

Observations on different representation of the databases show that, in the tree structure the same nodes are repeated on the different branches. It will occupy more amount of memory. Second, it is not possible to apply incremental mining in so many algorithms because of the different structures of the algorithms. So to remove the problem of memory, graph structure is one of the options. In graph structure duplication of the node is avoided. Incremental mining is also possible in the graph structure. Consider the following database given in table 1 for construction and working of the graph for frequent itemset mining.

TID	Items
1	B, C, D, F, G
2	D, F, G
3	D, F, G, A, C
4	B, D, F, G, E
5	A, G, F, E
6	E, F, G

Table 1: Transaction Database

Based on the data given in the table 1, the complete graph will be formed as shown in the figure 1. In the graph, the items are represented as a node and the numbers on the link between the two nodes represents the transaction(s) number(s). For example, the items C and D simultaneously appear in transaction number 1 and 3, which is clearly shown in the graph.

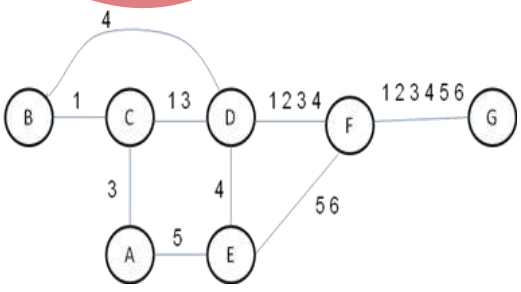

Figure 1: Database representation in the form of the graph

Now let's see how the frequent patterns will be formed based on the graph represented in figure 1. Suppose that minimum support (Min_Sup) threshold is 2. Select any node from the graph, for example the first selected node is B. This node will be stored in one array. Now all the links from B will be checked. The first link is between B and C via '1', which shows that support of items BC=1 which is less than the Min_Sup. So the pair BC will not be included in frequent patterns. At the same time C is the new node so it will also be saved in array where B was stored. Now because of '14', the nodes B and D are connected. Here '14' means because of transaction number '1' and '4', so total transaction becomes 2 which satisfy the minimum support requirement. The pair BD with support equal to 2 will be saved in the frequent pattern list as well as

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

the node D will be saved in the array where the total nodes will become 3, e.g. B, C and D. When the procedure of B with all the links will be completed, then from the array the next node will be picked up, here it is C. All the nodes reachable by C will be determined. When the procedure of node C will be completed the frequent pattern pair CDFG with support equal to 2 will be saved. In the same way when the D will be completed the frequent pattern pair DFG with support equal to 4 (i.e. '1234') will be saved. In the same way the procedure will be completed for all the nodes. When the frequent pattern mining procedure completed, the whole graph and results saved in one file. When some new transactions are added to the database then only that new record and the saved file needed to complete the frequent mining procedure. To understand the incremental mining concept assumes that two new transactions are added in the given database of Table 1, which is shown in Table 2.

TID	Items
1	B, C, D, F, G
2	D, F, G
3	D, F, G, A, C
4	B, D, F, G, E
5	A, G, F, E
6	E, F, G
7	D, C, F
8	A, E, C

Table 2: Updated Database

The changes occurred in the graph due to insertion of two transactions are shown in the figure 2 below.

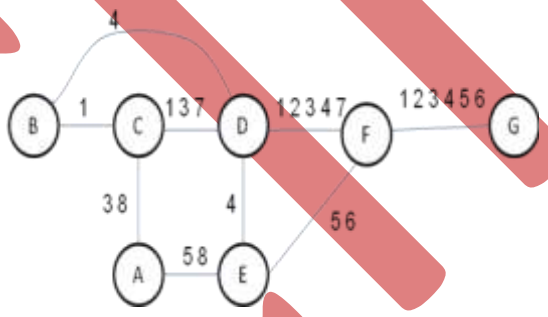

Figure 2 : Database representation in the form of graph after additional transaction

Already the detail of the previous graph has been saved in one of the file. In that saved file only need to update the new transactions only. No need to start the whole mining procedure from the scratch. If the results are saved again, they can be useful for incremental mining.

IV. CONCLUSION AND FUTURE SCOPE

Nowadays incremental mining is in high demand because of the continuously increasing size of the database. After day by day updation of the database if the mining procedure starts from the scratch, it is useless. It is too much costlier and time consuming that each and every time the mining procedure starts as a new database. All the tree based concepts are having some pros and cons related to frequent pattern mining. For the incremental approach, the tree based approach may not be suitable. In the graph based approach it seems to be easier to apply the incremental concepts. It gives good results in a reasonable time. Observation shows that the time taken by the graph based approach is more than the tree based approach. Ant Colony Optimization is a meta heuristic which has shown its capability to solve combinatorial optimization problems like Travelling Sales man Problem, Graph Coloring, etc. The results of the graph based approach may be improved with the use of the Ant Colony Optimization meta heuristic which can be considered as a further expansion of the problem. Also once ACO is applied for the frequent itemset mining, after that it can also be applied for the incremental mining.

REFERENCES

1) Alva Erwin, Raj P.Gopalan and N.R.Achuthan, "CTU-Mine: An efficient High Utility Itemset Mining Algorithm using Pattern growth approach" Seventh International conference on Computer and Information Technology,2007.

International Journal of Computers & Technology
Volume 4 No. 2, March-April, 2013, ISSN 2277-3061

2) Carson Kai-Sang Leung et al. (2007), "CanTree: a canonical- order tree for incremental frequent-pattern mining", Knowledge and Information Systems, 2007, 11(3), Page 287-311.

3) Cheung W., "Frequent Pattern mining without candidate generation or support constraint." Master's thesis, University of Alberta, 2002, SPRING '03, doi.ieeecomputersociety.org/10.1109/IDEAS.2003.1214917.

4) Christian Borgelt, " An Implementation of the FP-growth Algorithm" OSDM'05, August 21, 2005, Chicago, Illinois, USA, www.cs.rpi.edu/~zaki/OSDM05/papers/p1-borgelt.pdf.

5) http://fimi.cs.helsinki.fi/data

6) http://www.almaden.ibm.com/cs/quest//syndata.html#assocSynData

7) Jiawei Han and Micheline Kamber, Book :"Data Mining, Concept and Techniques", 578 pages, books.google.co.in.

8) Jien Pei, "Pattern-Growth methods for frequent pattern mining " Ph. D. Thesis, Simon Fraser University, 2002, ftp://fas.sfu.ca/pub/cs/theses/2002/JianPeiPhD.pdf.

9) Q. I. Khan, T. Hoque and C.K. Leung, " CanTree : A Tree structure for Efficient Incremental mining of Frequent Patterns". Proceeding of the Fifth International conference on Data Mining (ICDM'05), csdl2.computer.org/.../&toc=comp/proceedings/icdm/2005/2278/00/2278toc.xml&DOI=10.1109/ICDM.2005.

10) Rajnish Dass and Ambuj Mahanti, " An efficient heuristic search for Real-Time frequent pattern mining". International Conference on System Sciences – 2006, ieeexplore.ieee.org/iel5/10548/33362/01579371.pdf

11) Vivek Tiwari et al., (2010) , Association rule mining: A Graph Based Approach for mining Frequent Itemsets, 2010 International Conference on Networking and Information Technology, 978-1-4244-7578-0 © 2010 IEEE

12) William Cheung and Osmar R. Zaiane, " Incremental Mining of Frequent Patterns without candidate Generation or Support Constraint", IDEAS'03, doi.ieeecomputersociety.org/10.1109/IDEAS.2003.1214917.

13) www.cse.cuhk.edu.hk/~kdd/program.

Author's Biography

Sanjay Patel - B.E. in Computer Engineering from North Gujarat University, Gujarat, India in 2002 and M.E. (Computer Engineering) from Sardar Patel University, Vallabh Vidyanagar, Gujarat in 2008. He is currently working as an Assistant Professor in Computer Engineering Department at Vishwakarma Government engineering college, Chandkheda, Ahmedabad, Gujarat. Before that he has also worked as a lecturer in Computer Engineering department at Sankalchand Patel college of Engineering, Visnagar, Gujarat. His research interests are Data Mining and Artificial Intelligence.

Dr. Ketan Kotecha – B.E. in Electronics from Sardar Patel University and M.Tech. and Ph.D. from IIT, Bombay. He has over 15 years of experience in teaching UG students of Computer Engineering and over 9 years teaching PG students. He has published many International Journal papers. He visited several countries like USA, Canada, Europe, Singapore, Hongkong and South Africa to present papers in International Conferences. He has guided several masters' thesis in Computer Engineering and guiding 6 Ph. D. students. He served as the Principal of G H Patel College of Engineering and Technology, Vallabh Vidyanagar for three years before joining as a Director of Institute of Technology, Nirma University. His research interests include Artificial Intelligence & Computer Algorithms.