Aldrich Conference 2018

Memorial University of Newfoundland

A Study on Frequent pattern mining

Thanjida Akhter

takhter@mun.ca

ABSTRACT

Pattern mining algorithms can be applied on various types of data such as sub graphs, associations, indirect associations, trends, periodic patterns, sequential rules, lattices, sequential patterns, high-utility patterns, etc. Example like frequent pattern mining is that of finding relationships among the items or pattern in a database those products that are frequently purchased together by customers. This paper introduces a detailed analysis of the algorithms focusing on frequent pattern mining also try to find suitable algorithm base on time complexity.

1. INTRODUCTION

The problem of frequent pattern mining is about finding relationships among the items in a database. It can be described as follows [1], given a database D with transactions $T_1 \ldots T_N$, determine all patterns P that are present in at least a fraction S of the transactions. The fraction S is called the minimum support and can be expressed as an absolute number or as a fraction of the total number of transactions in the database. Each transaction T_i can be considered a sparse binary vector, or as a set of discrete values representing the identifiers of the binary attributes that are instantiated to the value of 1 [1]. Frequent patterns mining has become an important data mining method since it was first proposed by Agrawal et al. for analysis of market basket [2].

It can be applied in various areas. Some important applications of frequent patterns mining are listed here:

1. Customer Transaction Analysis: In this case, frequent patterns of buying behavior, used for making decision about shelf stocking or recommendations, is desired to be determined.
2. Other Data Mining Problems: As a fundament in the analytical process for a host of data mining problems, frequent pattern mining is also used to enable other major data mining problems, such as classification clustering and outlier analysis.
3. Web Mining: For benefiting Web site design, recommendations, or outlier analysis, the browsing behavior patterns can be determined by processing the Web logs.
4. Software Bug Analysis: The logical errors in these bugs can be shown as specific kinds of patterns.
5. Chemical and Biological Analysis: As the data in chemistry and biology are usually represented as graphs and sequences, the frequent patterns can be used in such graphs.

Besides above applications, frequent pattern mining also relates to a wide variety of topics. Generally, the research in this area can be categorized in one of four different classes.

1. Technique-centered: In this area, more efficient algorithms for frequent pattern mining, using different enumeration tree exploration strategies and different data representation methods, are determined.
2. Scalability issues: When the data is distributed or very large, then parallel or big-data frameworks instead of multi-pass methods must be used.
3. Advanced data types: In this area, multiple variations of frequent pattern mining are proposed for advanced data type.

Moreover, different data domains require specific algorithms for frequent pattern mining as well.

2. FREQUENT PATTERN MINING ALGORITHMS

Most of the algorithms for frequent pattern mining were developed based on the traditional support-confidence framework or generated more specific patterns for specialized framework. The raw frequency is greater than a minimum threshold, which is the target of a support framework designed for. This simple way of defining frequent patterns has an algorithmically convenient property, the level-wise property, which enables the design of a bottom-up approach to be exploring the space of frequent patterns. When any of the subsets is not frequent, a (k+1)-pattern may not be frequent. The techniques for frequent pattern mining started with Apriori-like join-based methods. In 1994, Agrawal and Srikant observed an interesting downward closure property, called Apriori, among frequent k-itemsets [3]. In these algorithms, candidate itemsets are generated in increasing order of itemset size which is referred to as level-wise exploration. These Apriori-like methods could be systematically explored as enumeration and tree, which provides a more flexible framework for frequent itemset mining by exploring in different strategies, including depth-first, breadth-first, or other hybrid strategies. The level-wise pruning can be used in the breadth-first strategy. For depth-first strategy, it has the advantages in maximal pattern mining, because of the greater effectiveness of the pruning-based lookaheads in the depth-first strategy [4]. Join-based algorithms can be considered as an enumeration-tree based algorithm and Apriori as well, either of which belongs to level-wise algorithm. One of the main challenges of frequent pattern mining is that many extra patterns are usually calculated. To overcome this issue, closed frequent itemsets, defined as frequent patterns without superset having the same frequency as that itemset, are usually mined. By this way, the number of patterns found could be reduced without losing any information about the support level.

The efficiency in frequent pattern mining algorithms can be gained in following ways:

1. Reducing the size of the candidate search space using pruning methods;
2. Improving the efficiency of counting based on database projection;
3. Using more efficient data structures, such as vertical lists, or an FP-Tree for more compressed database representation.

When quantifications, like support and confidence, are used, the found rules are usually not interesting, because of the absent of the normalization for the original frequency of the underlying items. For this issue, numerous methods, like combination of random sampling and hashing techniques [5], have been proposed. In negative associative rule mining, the traditional support frameworks are not available for the cases where an item is presented in the data 98% of the time, also called “negative items”. As the similarities between finding negative patterns and finding interesting patterns, some frameworks are designed to address both issues simultaneously.

In the context of constrained applications, frequent pattern mining methods are usually useful. When constraints are considered in the mining process, the mining can be performed at lower support levels than that by using a two-phase approach, especially under the condition that a large number of intermediate candidates can be pruned by the constraint-based pattern mining algorithm. The main shortcoming of this method is the constraints may result in the violation of the downward closure property. However, this drawback can be overcome by developed specialized algorithms based on specialized properties of constraints.

The volume of the mining pattern is usually extremely large besides various kinds of redundancy. Therefore, the determination of concise representations can be useful. A particularly interesting form of concise representation is that of closed patterns, which means that an itemset X is set to be closed if none of its supersets have the same support as X. By determining all the closed frequent patterns, the exhaustive set of frequent itemsets and their supports can be derived.

Besides aforementioned techniques, due to the development of hardware and software technologies leading to data stream, the major constraint on data mining algorithms comes from execution of the algorithms in a single pass. Two variants of frequent pattern mining are there for data streams: frequent items and frequent itemsets. In the former case, frequent 1-itemsets are usually determined from a data stream in a single pass, which is necessary when the total number of distinct items is huge. In the latter case, due to multi-pass frequent pattern mining methods, the main challenge is computational.

Moreover, big data also brings many challenges for frequent pattern mining, especially when the data is stored in a distributed way. Both data shuffle and intermediate results across the distributed nodes could cause significant costs. Algorithms should be designed to consider

the disk access constraint and the data transfer costs. Moreover, specialized algorithms for frequent pattern mining are also required for many distributed frameworks.

Generally, the basic frequent pattern mining algorithms are required to be extended to the variations for the advanced data types. Sequential pattern mining is one of the earliest proposed extensions. The concern in sequential pattern mining is that sequences of events presenting in at least a fraction s of the underlying records [6]. Unlike frequent pattern mining, sequential pattern mining should be responsible for the presence of complex baskets of items in the database and the temporal ordering of the individual baskets. Further, frequent pattern mining method can also be used for spatiotemporal data collected from GPS-enabled mobile devices and wearable sensors. When using frequent pattern in graphs and structured data, one challenge is the problem of isomorphism in graphs which requires more complex algorithms. When dealing with uncertain data with the attribute of probability distributions, numerous algorithms, like FP-growth, have been proposed [7].

Generally, there are two types of applications for frequent pattern mining. One is for other major data mining problems, like clustering. One is for generic applications, like Web log analytics.

3. APPLICATION OF FREQUENT PATTERN MINING

Frequent patterns reflecting strong associations among multiple items or object, capture the underlying semantics in data. They were successfully applied to inter-disciplinary domains beyond data mining. Frequent pattern mining has huge application such as:

- **Supermarket (**Product placement, special promotions)
- **Web search (**Web mining, Software Bug mining and page-fetch)
- **Heath care (**frequent sets of symptoms for a disease, DNA sequence

4. FREQUENT PATTERN MINING ALGORITHMS

There are some algorithm names of frequent pattern mining are as follow:

- Apriori Algorithm
- FP Growth Algorithm
- Broglet’s FP-Growth
- Goethals FP-Growth
- Eclat
- SaM Algorithm
- PrefixSpan

From all those algorithms now, Now the researcher elaborates three frequent pattern mining algorithms.

5. APRIORI ALGORITHM [AGRAWAL & SRIKANT 1994]

Apriori is the very first algorithm for mining frequent patterns. It was given by R agarwal and R srikant in 1994 [5]. It works on horizontal layout based database. It is based on Boolean association rules which uses generate and test approach. It uses BFS (breadth first search). Apriori uses frequent k itemsets to find a bigger itemset of k+1 items. In Apriori support count for each item is given, the algorithm first scans the database to find out all frequent items based on support. The calculation of frequency of an item is done by counting its occurrence in all transactions [6]. All infrequent items are dropped. Apriori property: All subsets of a frequent itemsets which are non-empty are also frequent. Apriori follows two steps approach: In the first step it joins two itemsets which contain k-1 common items in kth pass. The first pass starts from from the single item, the resulting set is called the candidate set Ck. In the second step the algorithm counts the occurrence of each candidate set and prune all infrequent itemsets. The algorithm ends when no further extension found.

Algorithm:

1. C_k: Candidate itemset of size k
2. L_k: frequent itemset of size k
3. L_1 = {frequent items};
4. for (k = 1; L_k! =∅; k++) do begin
5. C_{k+1} = candidates generated from L_k;
6. for each transaction t in database do increment the count of all candidates in C_{k+1} that are contained in t
7. L_{k+1} = candidates in C_{k+1} with min_support
8. End
9. return $\cup_k L_k$;

Apriori Algorithm Performance: The Apriori algorithm is an important algorithm for historical reasons and because it is simple algorithm that is easy to learn. However, faster and more memory efficient algorithms have been proposed. If efficiency is required, it is recommended to use a more efficient algorithm like FP Growth

6. FP-GROWTH ALGORITHMS [HAN, PEI & YIN 2000]

Frequent pattern growth also labeled as FP growth is a tree-based algorithm to mine frequent patterns in database the idea was given by (han et. al. 2000) [10]. It is applicable to projected type database. It uses divide and conquer method [7]. In it no candidate frequent itemset is needed rather frequent patterns are mined from fp tree. In the first step a list of frequent itemset is generated and sorted in their decreasing support order. This list is represented by a structure called node. Each node in the fp tree, other than the root node, will contain the item name, support count, and a pointer to link to a node in the tree that has the same item name [6]. These nodes are used to create the fp tree. Common prefixes can be shared during fp tree construction. The paths from root to leaf nodes are arranged in non-increasing order of their support. Once the fp tree is constructed then frequent patterns are extracted from the fp tree starting from the leaf nodes. Each prefix path subtree is processed recursively to mine frequent itemsets. FP Growth takes least memory because of projected layout and is storage efficient. A variant of fp tree is conditional FP tree that would be built if we consider transactions containing a itemset and then removing that itemset from all transactions. Another variant is parallel fp growth (PFP) that is proposed to parallelize the fp tree on distributed machines [8]. FP Growth is improved using prefix-tree-structure, Grahne and Zhu [9].

Algorithm:

1. if Tree contains a single path P then
2. for each combination do generate pattern β α with support = minimum support of nodes in β.
3. Else For each header ai in the header of Tree do {
4. Generate pattern β = ai α with support = ai. support;
5. Construct β.s conditional pattern base and then β.s conditional FP-tree Tree β
6. If Tree β = null
7. Then call FP-growth (Tree β, β)}

FP-growth Algorithm Performance: FP-Growth is generally the fastest and the most memory efficient algorithm. FP-Growth algorithm is efficient and scalable for mining both long and short frequent patterns. It is faster than the Apriori algorithm

7. ECLAT [MJ ZAKI, S PARTHASARATHY, M OGIHARA, W LI - KDD, 1997]

Eclat is a vertical database layout algorithm used for mining frequent itemsets. It is based on depth first search algorithm. In the first step the data is represented in a bit matrix form. If the item is bought in a transaction the bit is set to one else to zero. After that a prefix tree needs to be constructed. To find the first item for the prefix tree the algorithm uses the intersection of the first row with all other rows, and to create the second child the intersection of the second row is taken with the rows following it [4]. In the similar way all other items are found, and the prefix tree get constructed. Infrequent rows are discarded from further calculations. To mine frequent itemsets the depth first search algorithm is applied to prefix tree with backtracking. Frequent patterns are stored in a bit matrix structure. Eclat is memory efficient because it uses prefix tree. The algorithm has good scalability due to the compact representation.

Algorithm:

1. INPUT: A file D consisting of baskets of items, a support threshold σ, and an item prefix I, such that I ⊆ J.
2. OUTPUT: A list of itemsets F[I] (D, σ) for the specified prefix.
1. METHOD:
2. F[i] ← {}
3. for all i ∈ J occurring in D do
4. F[I]: = F[I] ∪ {I ∪ {i}}
5. # Create Di
6. Di ← {}
7. for all j ∈ J occurring in D such that j > I do
8. C ← cover({i}) ∩ cover({j})
9. if |C| ≥ σ then
10. Di ← Di ∪ {j, C}
11. 1# Depth-first recursion
12. Compute F [I ∪ i] (Di , σ)
13. F[I]: = F[I] ∪ F [I ∪ i]

Eclat Algorithm Performance: EClat is one of the interesting algorithm because it uses a depth-first search. In case of mining high utility item sets, the search procedure of EClat works very well.

Time Complexity

Apriori Algorithm: Scan database at every level of breadth first search. Apriori's outer loops are bounded by the number of common prefixes.

Worst case: $O(N^2)$

FP Growth : The complexity depends on searching of paths in FP tree for each element of the header table, which depends on the depth of the tree. Maximum depth of the tree is upper-bounded by n for each of the conditional trees. Thus, the order is: O (No. of items in header table * maximum depth of tree)= O(N * N).

Best case: $O(N \log N)$
Worst case: $O(N^2)$

EClat : Worst case: O(k(k+1) [K is the number of items]

8. COMPARISON OF ALGORITHMS

On standard dataset FP Growth performs the best and Apriori takes the maximum time. When the number of transactions are made three times the original, there is more increment in time for both FP Growth and Eclat than Apriroi. When the number of attributes are made three time the original, FP Growth performs the best and Apriori shows a sharp increase while Eclat lies in the between the two. Also from figure 2 and figure 3 increasing the number of attributes affects more any individual algorithm than increasing the number of tuples by same factor. When the number of transactions are made three times and the number of tuples are also increased three times, there is sharp increase in time taken by all algorithms. At this stage FP Growth performs the best and Apriori performs the worst as shown in figure 4.

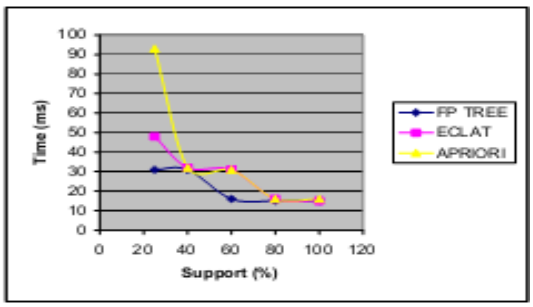

Figure 1: Comparison of Apriori, Eclat and FP Growth algorithm on artificial dataset.

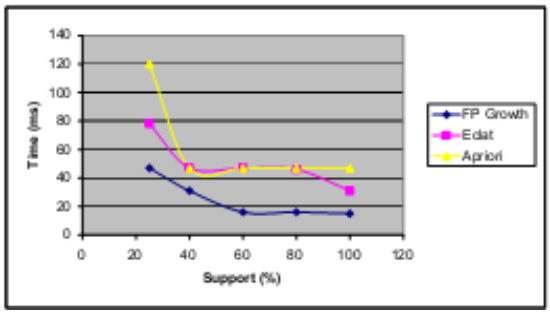

Figure 2: Comparison of Apriori, Eclat and FP Growth algorithm on artificial dataset when the number of transactions are made three times the original.

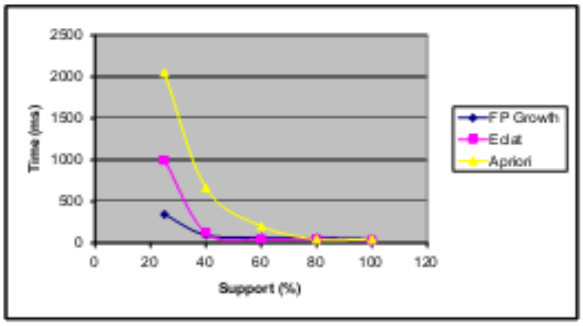

Figure 3: Comparison of apriori, éclat and FP Growth algorithm on artificial dataset when the number of attributes are made three times the

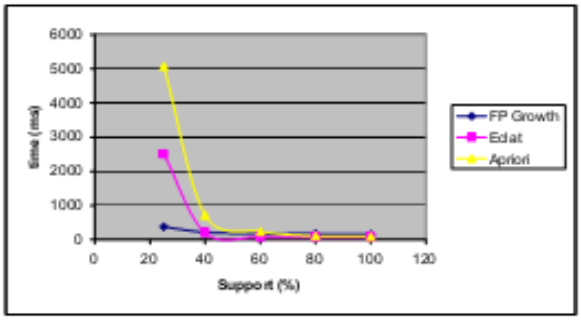

Figure 4: Comparison of Apriori, Eclat and FP Growth algorithm on a dataset with the number of transactions three times the original

9. CONCLUSION

Frequent pattern mining is the most important step in association rules which finally helps us in many applications like market basket analysis, clustering, series analysis, games, decision making, object mining, website navigation etc. In this paper the researcher surveyed the pattern mining algorithms namely apriori, Eclat and FP Growth. It is found that apriori uses join and prune method, Ecalt works on vertical datasets and FP Growth constructs the conditional frequent pattern tree which satisfy the minimum support. The major weakness of Apriori algorithm is producing large number of candidate itemsets and large number of database scans which is equal to maximum length of frequent itemset [5]. It is very much expensive to scan large database [11]. A true reason of apriori failure is it lacks efficient processing method on database [7]. FP Growth is the best among the three algorithms and is thus most scalable. Eclat performs poorer than FP Growth and the Apriori performs the worst. In the future improvements must be taken care to enhance the performance of Apriori and Eclat using a better layout to store the data.

10.REFEREANCE

1. Charu C. Aggarwal, Jiawei Han, Aggarwal Charu C., Han, Jiawei. An Introduction to Frequent Pattern Mining, Springer International Publishing: Cham 2014.
2. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM-SIGMOD international conference on management of data (SIGMOD'93), Washington, DC, pp 207–216.
3. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 1994 international conference on very large data bases (VLDB'94), Santiago, Chile, pp 487–499.
4. R. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth-first Generation of Long Patterns, ACM KDD Conference, 2000: Also appears as IBM Research Report, RC, 21538, 1999.
5. E. Cohen. M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding Interesting Associations without Support Pruning, IEEE TKDE, 13(1), pp. 64–78, 2001.
6. R. Agrawal, and R. Srikant. Mining Sequential Patterns, ICDE Conference, 1995.
7. Jiawei Han, Jian Pei, Yiwen Yin, Mining frequent patterns without candidate generation, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, p.1-12, May 15-18, 2000, Dallas, Texas, USA.
8. Agarwal, R.C., Agarwal, C.C. and Prasad, V.V.V. (2001) A tree projection algorithm for generation of frequent item sets. Journal of Parallel and Distributed Computing, 61(3), Pp. 350–371.
9. Bhadoria et. al. Analysis of Frequent Itemset Mining on Variant Datasets published in int.J.comp. Tech.appl., vol (2)5, ISSN:2229-6093, Pp. 1328-1333.
10. http://en.wikipedia.org/wiki/Database_transaction [on 11th nov 2012].
11. C.Borgelt. "Efficient Implementations of Apriori and Eclat". In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations, CEUR Workshop Proceedings 90, Aachen, Germany 2003.
12. Goswami D.N et. al. "An Algorithm for Frequent Pattern Mining Based On Apriori" (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, Pp. 942-947.
13. Rahul Mishra et. al. "Comparative Analysis of Apriori Algorithm and Frequent Pattern Algorithm for Frequent Pattern Mining in Web Log Data." (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (4), 2012, Pp. 4662 – 4665.