Abstract

Objectives: This paper focuses on various frequent pattern mining techniques, their challenges involved in static as well as stream data environment. Analysis: Information, the most precious asset is most of the times hidden inside heaps of raw data. It is required to be polished to retrieve information, converted into a form which can be analyzed for making decisions. Many research papers were read for the understanding of various techniques that mine frequent item sets either in static or stream data environments. Findings: This paper summarizes all the popular techniques available with us for frequent item set mining and provides some suggestions for optimizing them. The issues for static mining just take into account the time and space complexities but stream data mining is much more complex and challenging as compared to static. The problems like concept drifting, nature of data, its processing model, inadequacy, size, its retention in warehouses etc. are also required to be taken into account while working with real time data. Improvements: The algorithms for stream data mining should be incremental and resource adaptive in nature so that they can handle the change and adjust their processing parameters according to the availability of resources respectively. The available resources which will be very helpful in shared environments (where resources are shared by multiple processes).

Keywords: Data Mining Techniques, Frequent Patterns, Issues in Data Mining, Static Data Mining, Stream Data Mining

1. Introduction

Data mining is one of the important phases of KDD process (a procedure for acquiring knowledge out of data). It is the process where we analyze huge data from store house to extract useful information with the help of intelligent techniques that aid in making decisions. The paper will discuss two techniques of data mining: frequent patterns and association rules under static as well as stream data environment.

2. Data Mining Techniques: Frequent Patterns, Association Rules

Before explaining frequent patterns and association rules, one should have idea about two terminologies: Support and Confidence.

Support is the count of the number of transactions in which item appears in a dataset. Minimum Support is a value specified for finding frequent patterns. If itemsets are occurring together in a database greater than or equal to the specified support value, then those itemsets are called frequent itemsets.

\[\text{Support} (M \Rightarrow N) = P(M \cup N)\]  \hspace{1cm} (2.1)

A confidence defines if a person purchases any item \(M\), how many times he purchases item \(N\) with it.

Minimum confidence value is set to validate associations.

\[\text{Confidence}(M \Rightarrow N) = \frac{\text{Support} (M \cup N)}{\text{Support} (M)} \]  \hspace{1cm} (2.2)

Frequent patterns, as the term describes, are those patterns that occur together time and again in dataset. The patterns can be subsets of item sets, sequences, graphs, structures. When transactional database is discussed, frequent item sets are discovered depending upon the value
of support. If the count of occurrence of an item set is greater than or equal to the value of minimum threshold (support), that item set will be considered as frequent itemset.4

Association rule mining is done to discover hidden relationships among data items which can help in making decisions and predictions.5 These are of the form A => B(S,C), here A,B represents frequent item sets where A∩B=Ø and S and C represent minimum support and confidence values respectively. Associations are mined based upon confidence value. First, produce all non-empty subsets of a frequent item set. Then, for each subset, if \[
\frac{\text{Support (frequent set)}}{\text{Support (subset)}} > \text{confidence,}
\]
apply rule,

“Subset → frequent set – subset” (2.3)

The most famous application of association rule mining is Market Basket Analysis, a technique of analyzing customer’s behavior by discovering relations among various items they place in their shopping baskets.

3. Static and Stream Data Mining

Data streams are of two types: In Online data streams, we deal with real time data that needs to be updated regularly. This data comes one by one continuously in a sequence. For example: sensor data, stock tickers. In Offline data streams, data is collected at regular intervals in large amounts and then processed afterwards. For example, in web logs, data at certain intervals is logged and then processed offline. All the data mining techniques (data clustering, data classification, association rules and frequent patterns) are applicable to stream data but the process of extracting meaningful information from stream data is very complex. This paper reviews various approaches to find frequent patterns both in static as well as stream data environment.

4. Challenges in Stream and Static Data Mining

Authors focused on different factors needed to be taken care while performing association rule mining on data streams. Due to the different nature of data stream, conventional algorithms like Apriori and FP-Growth cannot be used as these require more than one scan of database which is extremely undesirable case in stream data mining environment. Two type of issues, general and application dependent were discussed. General issues are applicable for all applications that deal with stream data.

Data Treatment Model: Data stream arises in limitless and uninterrupted manner and that too in large volumes. The issue is to draw out transactions from a large data stream that would help in association rule mining. Three frameworks were introduced for data treatment. In Landmark model, a point known as landmark is selected. All the transactions from that point to the current are used for finding frequent patterns. In Damped model, each transaction is assigned some value and this value reduces with their timestamp. Recent transaction is having more value as compared to older. In Sliding window model, a sliding window is maintained in which a portion of stream is stacked in and processed.

Memory Management: Sufficient space for accommodating itemsets and their frequencies when a large volume of data arrives at once is the biggest issue. Moreover, with the arrival of fresh stream, the frequencies of itemsets vary most of the times. So, it is essential to gather up least amount of information. But this information should be sufficient to yield association rules.

Choice of Algorithm: The algorithm should be chosen according to the requirement of results. Some algorithms give exact results and some give approximate results with false positives or false negatives.

Concept Drift Problem: The itemset which is frequent can become infrequent with the coming transactions and vice versa. Due to this varying nature of data, predictions of association rules can become inaccurate. This problem is known as Concept Drift and to handle it, incremental algorithms are required.

Resource aware algorithms: Resource aware algorithms are required which can adjust their processing rate according to the availability of resources. This thought will very helpful in the environment where resources are shared by multiple processes.

Every application has its own needs and issues. Users should be able to change the mining parameters according to their requirements even when the algorithm is running. Mining multidimensional data stream is another issue which increases the complexity. The applications need to generate responses in accordance with user’s queries. If data is arriving from more than one source, it leads to the increased communication cost. Integrating the frequency counts is also an issue.6

Authors discussed issues at all phases of KDD and in business environment. Data preprocessing is the most time consuming task and much complex in stream data as compared to static data because it is continuous, arrives
every second in large volumes. In requirement gathering, most of the times clients are not able to formulate their business problems and business questions. They are unable to specify what they want from developers. The preprocessing challenges include noisy, varying data and outliers. In real world scenario, data is mostly incomplete, biased and even sometimes is not readily available. Selecting important information from large set of data is also a challenge. Data transformation is the most time consuming phase. Problem of concept drift in data is very difficult to handle. It is very hard to summate the information every second. Performing various operations like aggregation, association is not an easy task. Unlike static data mining, there are very few tools available which can assess the quality of stream mining algorithms. The retention period of data in data warehouse and making updates without compromising its availability is very big challenge. The system should be able to handle missing and noisy data. Data should be mined at different levels. The system should be scalable and support data migration, should be capable of handling multiple data types. Complexity of algorithms is another important issue.

5. Static Data Mining Techniques for Frequent Patterns

5.1 Apriori Algorithm
It is a basic technique for extracting frequent patterns by generating candidates. As the name implies, it requires the prior knowledge of frequent itemset properties. It is an incremental approach where frequent k-item set is used to generate frequent (k+1)-itemset. Initially, the database is scanned for finding count of all 1-itemsets. Then based upon the threshold value; frequent 1-itemsets are extracted. A cross join on the resultant is applied to get all possible 2-itemsets combinations. Again database is scanned for the counts of those itemsets and the process repeats until there is no new frequent itemset. To reduce the number of candidates, algorithm uses apriori property, also called downward closure property, says,"If an itemset is not frequent, its supersets will never be frequent". Hence, the algorithm works in two steps: joining(cross join is performed on k-itemsets to generate k+1 itemsets) and pruning(casting out infrequent itemsets based upon apriori property). The disadvantage of using this algorithm is that database is required to be scanned multiple time which increases the execution time. The generation of large number of candidates increases the space complexity.

5.2 FP Growth
It is a method that extracts frequent itemsets based on divide and conquer technique. FP-Growth works in two steps: Creating and Mining FP tree. While creating tree, the database items are scanned and arranged as a branch of tree in the descending order of their counts for each transaction. Items are marked along with these counts. Root is always NULL. If some sequence of a transaction is already existing, then the remaining items are joined below it and the count of subset items is increased by one. Tree is mined by constructing its conditional pattern which includes the paths to reach the node through root. A sub tree is constructed and patterns are generated by concatenating the item with its path. Search space is reduced due to the generation of conditional patterns. It gives good results for even long patterns. Since there is no need of candidate generation, space complexity is reduced.

5.3 ECLAT
It is an improvement of apriori algorithm. It uses vertical data format (item:transaction id set). It is similar to apriori, just the table is reversed. The item sets having count less than minimum support threshold will be eliminated. 2-itemsets will be generated by the intersection of transaction id sets of 1-itemsets. Cross join is performed to generate three item sets. The 2-itemset subsets of 3-itemset are evaluated from previous table. From the downward closure property, 2-itemsets which are not frequent, their 3-itemset will also be infrequent. So, those 3-itemsets are casted out. Algorithm repeats till no new frequent itemset is generated. Due to this methodology, multiple scans of database are not required since transaction id set contains all the required information for counting supports. But length of TID-set requires large memory space. Computation time is also affected during intersection process.

Many improvements were made on these basic algorithms to mine frequent patterns efficiently. Another paper introduced a hybrid apriori algorithm for frequent itemset mining using both horizontal (original Apriori) and vertical apriori(Eclat). Both the approaches were used simultaneously using the concept of multithreading. The algorithm used thread based multitasking. As
threads are light weighted, they require less space and CPU. Context switching and communications are less expensive. So, multithreading approach is encouraged for optimized utilization of CPU by decreasing its free time. Hybrid of Apriori and FP-Growth was introduced. More improvements in Apriori were introduced.

6. Stream Data Mining Techniques

6.1 RAQ-FIG

RAQ-FIG(Resource Adaptive Quality Assuring Frequent Item Generation) by is an improvement of MFI_TRANSW developed. Here, data items in a particular transaction are presented in a bit sequence. If an item is in i-th transaction, i-th bit in the sequence is fixed to 1 and the rest to 0. After it, frequent ‘k’ itemsets are mined by executing logical AND operation on items. The RAQ-FIG algorithm includes the concept of resource adaptation and enhances the quality of results through its capability to adjust its processing parameters according to the availability of resources. The algorithm uses sliding window model to mine the most recent data. The model works by putting transactions into batches. A sliding window is composed of a number of basic windows. The main idea behind basic window is that rather than discarding single old transaction, whole batch can be discarded during window sliding. Figure 1 shows the concept of sliding window composed of various basic windows.

\[
\text{Size of sliding window} = x \cdot \text{number of basic windows allowed in a sliding window} \quad (6.1)
\]

Initially, all the items of a transaction are presented in a bit sequence. The number of bits in a sequence depends upon the number of transactions. The concept will be clear with example. Say the number of transactions be four (t1, t2, t3, t4), then the number of bits will be four. w, x, y and z be the items present in various transactions in Table 1.

Table 1. Bit representation of items

| Transaction | Items | Bit Sequence |
|-------------|-------|--------------|
| <t1,xyz>    | W     | 0110         |
| <t2,wxy>    | X     | 1110         |
| <t3,wxyz>   | Y     | 1111         |
| <t4,yz>     | Z     | 1011         |

Since ‘w’ is present only in t2 and t3, corresponding bits are set to 1 and rest to 0. Representation of xyz= 1110 & 1111 & 1011 = 1010, hence x, y and z are present in t1 and t3.In this way, other item sets can be generated. 3-itemsets can be used to generate 4-itemsets. Using bit representation; the computation of frequent item sets becomes easier. One needs to just calculate the number of 1’s after AND operation and then compare it with support value. Current resource usage is traced with reference to space used, processing speed and input frequency. The quality of results is analyzed with the ratio of error value to the minimal support value. Initially, the error is set by user and later on, the value varies depending upon resource availability. In this way, by controlling error, the quality of results is improved. All these measurements are made repeatedly. If it is felt that resources are not sufficient, adaptation parameter gets adjusted accordingly, for example, decreasing or increasing sliding windows size. Current resource usage is traced with reference to space used, processing speed and input frequency. The parameter which measures the current resource consumption is called as ‘Adaptation Factor’ (AF).

\[
\text{AF} = \frac{\text{processing time in seconds} \times \text{speed of stream} \times \text{memory allocated}}{180} \quad (6.2)
\]

The desired value is near 1. Modification in the size of sliding window depends upon this factor.

6.2 Mining Frequent Itemsets over Arbitrary Time Intervals

Another algorithm improved the temporal accuracy of subsisting algorithm (Mining frequent item sets over arbitrary time intervals) by ignoring outdated data and introducing a concept of ‘Shaking Point’. The base algorithm, FP streaming uses tilted time window treatment model and stocks the data in FP-tree which is updated with the arrival of new transaction. FP streaming tree consists of three components: A Transaction tree, which stocks the dealings for the current window; a Pattern tree, which stocks all the frequent patterns of previous windows and a tilted time...
window. Various improvements are made to FP-streaming algorithm. The concept of tail pruning is casted out for maintaining veracity of results. The input is read and filled in batch within a fixed time point after which algorithm discards the input and processes the batch. Initially, when an item arrives, its time stamp is set to current time. Each item is treated as node of a tree. With the arrival of each new transaction, if there is any new item, it is added to the tree. Frequencies of nodes are updated. The frequencies of items in the current item set increases and the remaining nodes in the tree are updated with 0 frequencies and this frequency is added to tilted time window. During processing, after some fixed number of batches, some nodes having continuously 0 frequencies will be encountered, these nodes are called obsolete items. A variable named as fading factor is assigned to each node. A user defined parameter called 'fading support' is passed. The timestamp of the node whose count is 0 will never be updated. After 'x' batches, Shaking Point will be performed where fading factor of every node is computed. It is equal to the difference between current timestamp and time-stamp of node. If this factor touches or is larger than the fading support, then that node and its full super tree will be wiped out. This helps to avoid useless procedures and saves time.

6.3 WMS (Weighted Minimum Support)
In WMS, algorithm, itemsets are assigned weights depending upon the time of their occurrence in the transactions. Database is divided into zones according to different time spans. Weights are assigned according to formula.

\[ WMS = \frac{Minimum \ support \ DB}{DB(t_1,t_2)} \]  

\( DB(t_1,t_2) \) is the number of transactions between the time \( t_1 \) and \( t_2 \). DB is the total number of transactions in DB and minimum support is the common support initially assigned for all transactions.

6.4 SWFP (Sliding Window Frequent Patterns)
SWFP used sliding window treatment model to find in-frequent patterns along with frequent item sets. To decrease the effect of old items, decomposition technique was used. The patterns which are not useful for final result are casted out which cuts down time and space complexity. The experimental results prove the efficiency and extensibility of SWFP.

6.5 More Algorithms
One more algorithm focused on the uncertain nature of stream data to improve the veracity of results. The items which are not frequent right now may become frequent in the coming transactions. Two tree based algorithms were used, first was capable of giving approximate results and the second one gave exact results. The concept of pre-minimum-support was introduced, where frequent patterns were found with support little less than the exact support threshold value. In this way, early casting out of items was avoided. Frequent item sets are stored in a tree structure called UF-stream. Then, the second algorithm helps in outputting exact results without any requirement of pre-minimum-support.

A new algorithm ‘output granularity’ was introduced. Stream data mining suffers from a problem that the rate of input stream is much greater than the processing and mining process of algorithms. Due to this, one is always left with estimated results. To solve this problem, author followed resource adaptation approach to maintain accuracy in results by reducing the rate of stream along with sampling and aggregation depending upon available resources. The main focus of algorithm was on the number of transactions that can reside in the primary memory before cumulative additions. The author explained the algorithm for clustering. The main component of algorithm was Resource Adaptation (RA) which tries to compute the minimal input rate and match the processing rate of algorithm with it. The algorithm can be applied for classification, clustering and frequent patterns. Another algorithm was introduced for incremental mining in distributed environment. Here support is calculated at local servers and then sent to the main (global) servers or to their above hierarchy.

7. Discussion
Various algorithms in both static and stream data environments are studied. All are having some advantages as well as shortcomings. Apriori, the basic algorithm for mining frequent patterns in static data environment suffers from space complexity due to large number of candidate generation. Multiple scans of database are required to find the support counts of item sets. Éclat emerged as an improvement of Apriori which uses vertical data format. FP-Growth uses a different approach of divide and rule. A tree is created and frequent item sets are mined. Due to limited number of database scans and zero candidates, it is efficient as compared to Apriori. Many improvements
were made to these algorithms. The algorithm for mining frequent patterns in time sensitive environment discussed in previous section helps in reducing the size of tree through the execution of shaking point but the problem is that data within a certain time span was taken for processing and rest was discarded. There is rise in execution time when shaking point was executed after some fixed number of batches. RAQ-FIG came out with a powerful approach of adjusting its processing parameters according to the availability of resources to save the system from being halted by getting stressed due to excess usage of resources. Many new algorithms emerged which introduced the use of sliding windows, weighted transactions, finding the frequent patterns at various granularities.

8. Conclusion

Mining frequent patterns in stream data environment is complex than that of static data environment due to their unbounded, continuous and varying nature. The data is generated every second in large amounts which makes its processing difficult. The algorithm should include the concept of resource adaptation. It should output the results or vary the processing speed according to the available resources. A smart approach is required to either reduce the execution time of algorithm or improve the processing of sliding window. It is very difficult to mine data streams. Concept drift is the biggest problem. Various algorithms for mining stream data like FP-streaming, RAQ-FIG, MFI-TRANSW, WMS are available but there is no single algorithm which deals with all the challenges of stream data environment. Most of the algorithms give approximate results and very few generate exact results. The algorithms for data stream mining should be incremental in nature and address the problem of concept drift. Algorithms should adjust its processing parameters like input size, threshold value to make it adaptable according to the available resources which will be very useful in shared environments where multiple processes work together. Their incremental nature will make them adaptable to the changes in the distribution of input. There are many more challenges which are required to be overcome to mine frequent patterns efficiently.

9. References

1. Han J, Kamber M, Pei J. Mining frequent patterns, associations, and correlations. Data Mining: Concepts and Techniques. 2nd edn., Morgan Kaufmann Publishers Inc.: San Francisco; 2006. p. 227–48.
2. Zarrouk M, Gouider MS. Frequent patterns mining in time-sensitive data stream. International Journal of Computer Science Issues. 2012 Jul; 9(4):117–24.
3. Aggarwal CC. Data streams: Models and algorithms. 1st edn., Springer US: US. 2007;1:761–100.
4. Gruenwald JN. Research issues in data stream association rule mining. ACM SIGMOD Record. 2006 Mar; 35(1):14–19.
5. Zhu Y, Shasha D. StatStream: Statistical monitoring of thousands of data streams in real time. ACM SIGMOD Record. 2005 Jun; 34(2):358–69.
6. Krempl G, Zliobaite I, Brzezinski D, Hullermeier E, Last M, Lemaire V, Noack T, Shaker A, Sievi S, Spiliopoulos M, Stefanowski J. Open challenges for data stream mining research. ACM SIGKDD Explorations Newsletter - Special issue on big data. 2014 Jun; 16(1):1–10.
7. Kohavi R, Mason L, Parekh R, Zheng Z. Lessons and challenges from mining retail e-commerce data. Kluwer Academic Publishers. 2004 Oct; 57(1):83–113.
8. Yassir A, Nayak S. Issues in data mining and information retrieval. International Journal of Computer Science and Communication Networks. 2012 Mar; 2(1):93–8.
9. Kanth MR, Loshma G. Parallel multithreaded apriori algorithm for vertical association rule mining. International Journal of Advanced Research in Computer and Communication Engineering. 2013 Dec; 2(12):4729–35.
10. Dandu S, Deekshatulu BL, Chandra P. Improved algorithm for frequent item sets mining based on apriori and FP-tree. Global Journal of Computer Science and Technology Software and Data Engineering. 2013; 13(2):13–16.
11. Maolegi M, Arkok B. An improved apriori algorithm for association rules. International Journal on Natural Language Computing. 2014 Feb; 3(1):21–9.
12. Yabing J. Research of an improved apriori algorithm in data mining association rules. International Journal of Computer and Communication Engineering. 2013 Jan; 2(1):25–7.
13. Chandrika J, Kumar KRA. Frequent itemset mining in transactional data streams on quality control and resource adaptation. International Journal of Data Mining & Knowledge Management Process. 2012 Nov; 2(6):1–12.
14. Li HF, Ho CC, Shan MK, Lee SY. Efficient maintenance and mining of frequent itemsets over online data streams with a sliding window. IEEE International Conference on Systems, Man, and Cybernetics. 2006 Oct; 3:2672–7.
15. Li Y, Chang C, Yeh J. An algorithm for mining association rules with weighted minimum supports. 1st edn., Artificial
Intelligence Applications and Innovations. Springer US: Beijing; 2005. p.291–300.

16. Feng W, Quanyuan W, Yan Z, Xin J. Mining frequent patterns in data stream over sliding windows. Proceedings of International Conference on Computational Intelligence and Software Engineering. 2009 Dec; p.1–4.

17. Leung CKS, Hao B. Mining of frequent itemsets from uncertain data. Proceedings of IEEE 25th International Conference on Data Engineering; Shanghai; 2009 Mar. p.1663–70.

18. Gaber MM, Krishnaswamy S, Zaslavsky A. Cost-efficient mining techniques for data streams. Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation, Australia; 2004 Jan. p.109–14.