Concept Change Aware Dynamic Sliding Window Based Frequent Itemsets Mining Over Data Streams

K. Neeraja, V. Sireesha

M.Tech, Dept of Computer Science, Vasavi College of Engineering, Hyderabad, India.
Assistant Professor, Dept of Computer Science, Vasavi College of Engineering, Hyderabad, India.
Email: k.neeraja09@gmail.com, sireesha.vikkurty@gmail.com

ABSTRACT: Considering the continuity of a data stream, the accessed windows information of a data stream may not be useful as a concept change is effected on further data. In order to support frequent item mining over data stream, the interesting recent concept change of a data stream needs to be identified flexibly. Based on this, an algorithm can be able to identify the range of the further window. A method for finding frequent itemsets over a data stream based on a sliding window has been proposed here, which finds the interesting further range of frequent itemsets by the concept changes observed in recent windows.

Keywords: Data Mining, Data streaming, Frequent Itemsets, Concept Change.

1. INTRODUCTION

Data mining concepts and methods can be applied in various fields like marketing, medicine, real estate, customer relationship management, engineering, web mining etc. Frequent itemset mining has become one of the important subjects of data mining. Recently, there has been much interest in data arriving in the form of continuous and infinite data streams, which arise in several application domains like high-speed networking, financial services, e-commerce and sensor networks. Data-stream mining is a technique which can find valuable information or knowledge from primitive data. Unlike mining static databases, mining data streams poses many new challenges. First, each data element should be examined at most once. It is unrealistic to keep the entire stream in the main memory. Second, the memory usage for mining data streams should be bounded even though new data elements are continuously generated. Third, each data element in data streams should be processed as fast as possible. Finally, the results generated by the online algorithms should be instantly available when user requested.In this paper we consider mining recent frequent item sets in sliding windows over data streams and estimate their true frequencies, while making use of dynamic sliding windows according to the concept change occurred in the incoming data. Concept refers to the target variables, which the model is trying to predict or to describe. Concept change is the change of the underlying concept over time. The concept change is a known phenomenon in data stream processing due to dynamic nature of incoming data [1]. The concept change makes frequent itemset mining in data streams even more challenging than traditional static databases. In this study, the aim is to measure the amount of changes in the set of frequent patterns and to exploit it by dynamically adjusting the window size. The sliding window model is an interesting model to mine frequent patterns over data streams. This model tries to handle the concept change by considering only recent arrived data. The window is usually stored and maintained within the main memory for fast processing[1]. In order to overcome the problem of determing the window size, with motivation gained from the research article “Towards a variable size sliding window model for frequent itemset mining over data streams”, here we propose a frequent itemset mining model for Data streams, which can be referred as “concept change aware dynamic sliding window based frequent itemset mining over data steams”. The main objective the proposal is to investigate the problem of flexible size sliding window for frequent itemset mining over data streams in regard to justify the proposed model. In this proposed model, the process of continuous monitoring to observe the amount of change in the set of frequent patterns will be adopted. The window size should adaptively adjust, which is based on the observed amount of concept change within the incoming data stream and should expand or reduce according to the state of change. The rest of the paper is as follows. The next section presents a review on related works. Problem statement is described in Section 3 Sections 4 describe the most relevant features of the algorithm implementation, while the experimental results are reported and discussed in Section 5. Finally Section 6 concludes the paper.

2. LITERATURE SURVEY

Finding frequent item sets in a set of transactions is a popular method for so-called market basket analysis, which aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc. In particular, it is tried to identify sets of products that are frequently bought together. The first algorithm for frequent patterns mining over data streams was proposed by Manku and Motwani (2002) where the authors started with frequent items mining and then extended their idea to frequent itemset mining. Every single item coming from the data streams is considered as a transaction in frequent item mining. Sliding window is one of the useful and widely used model for datastream processing and mining. Chang et al. utilized an information decay model to differentiate the information of recent transactions from the information of old transactions. Based on the work by Karp et al on computing frequent items, Jin et al. proposed another one-pass algorithm, STREAM, to compute approximate frequent itemsets over entire data streams. Chang et al proposed a sliding window method of finding recent frequent itemsets

over a data stream based on the estimation mechanism of the Lossy Count-ing algorithm. Lin et al. introduced an efficient algorithm for mining frequent itemsets over data streams under the time-sensitive sliding-window model. The SWIM (Mozafari et al., 2008) is another pane based algorithm in which frequent itemsets in one pane of the window are considered for further analysis to find frequent itemsets in whole of the window. It keeps the union of frequent patterns of all panes and incrementally updates their supports and prunes infrequent ones. There are a number of studies related to diagnosing change in data streams (Aggarwal, 2003; Kifer, Ben-David, & Gehrke, 2004; Tao & Ozsu, 2009). These methods are mainly based on approximating underlying probability distribution that the incoming data stream is generated based on. Koh and Lin (2009) proposed a method to estimate and detect concept shift of frequent patterns over data streams. Ng and Dash (2008) proposed a test strategy to alarm the user about detecting change in data streams. In contrast to (Koh & Lin, 2009) and (Ng & Dash, 2008), an approach which dynamically updates the set of frequent itemsets over data streams and measures the precise value of concept change is more desirable and applicable to real data streams. Such an approach is exploited by (Mahmood Deypir & Sattar HAshemi, 2012) in a flexible size sliding window frequent itemset mining to adjust the window size dynamically according to concept changes within a data stream.

3. PROBLEM STATEMENT

The dynamic sliding window model is an interesting model to mine frequent patterns over data streams. This model tries to handle the concept change by considering only recent arrived data. For fast processing, the window is usually stored and maintained within the main memory. The context variation window is initialized with some percentage of the normal window and then the items coming from data streams are first initialized to normal window based on minimum window size. After that the incoming records are checked with the records present in the normal window and stored in the context window based on the similarity metric. Here we used Jaccard Similarity coefficient to check the similarity between the item sets. If the items are similar that is having less jaccard distance then those are placed in the normal window and the window is finalized. If the items are having more jaccard distance then it means the concept change occurred so with out inserting the new records it finalizes the normal window and it creates a new window and stores the records in the context window in next window and again the incoming records are calculated and inserted and so on. Like this the incoming items are stored according to the similarity at initial stages only. Then the frequent item sets are calculated for the finalized windows. Here we used Eclat algorithm for finding frequent itemsets and association rules are formed for the frequent itemsets to perform recommendations.

3.1 Jaccard Similarity:

The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

(If *A* and *B* are both empty, we define *J*(*A*,*B*)=1.) Clearly,

$$0 \leq J(A,B) \leq 1.$$

The Jaccard distance, which measures *dis*similarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1.

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

3.2 Eclat Algorithm:

The Eclat algorithm is used to perform itemset mining. Eclat [3] algorithm is basically a depth-first search algorithm using set intersection. It uses a vertical database layout i.e. instead of explicitly listing all transactions; each item is stored together with its cover (called tidlist) and uses the intersection based approach to compute the support of an itemset. In this way, the support of an itemset X can be easily computed by simply intersecting the covers of any two subsets Y, Z U X, such that Y U Z = X. The Eclat algorithm is as given below:

Input: D, K, i ⊆ I
Output: F[I](D, K)

1. F[I] :={}
2. for all I L I occurring in D do
3. F[I] := F[I] ∪ {I ∪ {i}}
4. // Create Di
5. Di: = {}
6. for all j L I occurring in D such that j>I do
7. C := cover({i}) O cover({j})
8. if |C| >= K then
9. Di: = Di ∪ {(j, C)}
10. end if
11. end for
12. //Depth-first recursion
13. Compute F[I ∪ {i}](Di, K)
14. F[I] := F[I] ∪ F[I ∪ {i}]
15. end for

In this algorithm each frequent item is added in the output set. After that, for every such frequent item i, the I projected database Di is created. This is done by first finding every item j that frequently occurs together with i. The support of this set {i, j} is computed by intersecting the covers of both items. If {i, j} is frequent, then j is inserted into Di together with its cover. The reordering is performed at every recursion step of the algorithm between line 10 and line 11. Then the algorithm is called recursively to find all frequent itemsets in the new database Di[3].

3.3 Association Rules:

An association is a rule of the format: LHS _ RHS**,** where *LHS* and *RHS* stand for Left Hand Side and Right Hand Side respectively. These are two sets of items and do not share common items. The rule can be read as “IF LHS THEN RHS”. A set of items is called an itemset [4]. The goal of association rule discovery is to find associations

among items from a set of transactions, each of which contains a set of items. Not all of the association rules discovered within a transaction set are useful. Generally the algorithm finds a subset of association rules that satisfy certain constraints. The most commonly used constraint is minimum support. The support of a rule is defined as the support of the itemset consisting of both the LHS and the RHS. The support of an itemset is the percentage of transactions in the transaction set that contain the itemset. An itemset with a support higher than a given minimum support is called frequent itemset. Similarly, a rule is frequent if its support is higher than the minimum support. Minimum confidence is another commonly used constraint for association rules. The confidence of a rule is defined as the ratio of the support of the rule and the support of the LHS. It is equivalent to the probability that a transaction contains the RHS if the transaction contains the LHS. A rule is confident if its confidence is higher than a given minimum confidence. Most association rule algorithms generate association rules in two steps:

1. Generate all frequent itemsets; and
2. Construct all rules using these itemsets.

The foundation of this type of algorithm is the fact that any subset of a frequent itemset must also be frequent, and that both the LHS and the RHS of a frequent rule must also be frequent. Therefore, every frequent itemset of size *n* can result in *n* association rules with a single item RHS. Then we compared the results of the proposed algorithm and the previous algorithms i.e., without context window checking[4].

4. PROPOSED ALGORITHM

In this study, a new dynamic sliding window algorithm for frequent itemset mining over data streams is proposed. In this algorithm, the user specifies initial window size, context window size and context variation . The algorithm adaptively adjusts the window length based on the concept changes that it detects during the stream data processing.The proposed algorithm is, Algorithm CDSW(mWSize, MWSize, CWSize, CV, ms)

1. W=WindowInit (mWSize, MWSize); //window is initialized with
2. C=ConceptChangeWindowInit(CWSize); //concept variation window is initialized
3. WT=Insert(T, mWSize);
4. CT=Insert(T, CWSize);
5. Forever
6. JSim = D_j (A,B); //jaccard distance between the itemsets is calculated
7. If JSim < CV then //concept variation checking
8. While WSize = (MWSize + CWSize)
9. WT = WT + CT; //window is dynamically adjusted
10. CT=Insert(T); //New transactions are inserted in concept change window
11. JSim(CT,WT); //New transactions are checked for similarity
12. End While;
13. FinalizeWindow(WSize,WT); //Window size and transactions are finalized
14. Else
15. FinalizeWindow(WSize,WT);
16. End if
17. FPSet = Eclat(WT,ms); //frequent items are calculated using éclat algorithm.
18. End for

In This Algorithm, mWSize, MWSize, CWSize, CV, ms are the parameters used to refer minimum window size, maximum window size, concept variation window size, concept variation and minimum support. In line 1 & 2, The window and the concept change window is initialized. In line 3, the transactions are inserted in to the widow up to minimum window size. After that, the next coming transactions are inserted in to the concept change window. Then Until all transactions are completed, the normal window transactions and the concept change window transactions are checked for similarity in line 6. In line 7, the jaccard distance and the given concept variation are checked. If the Jaccard similarity distance is less than the given concept variation threshold, that is, the incoming transactions are similar. So, the concept change window transactions are inserted in to the normal window in line 9. In line 10 & 11, the new incoming transactions are again inserted in to the concept change window and checked for similarity. This process goes until the normal window size is filled with the maximum window size and concept change window size. In line 13, If the window size is full, then the transaction of that window are finalized and the concept change window transactions are inserted in to the next window. Else if the similarity is greater than the concept variation given, that means the incoming transactions are not similar to the previous transactions. So In line 15, the window with minimum transactions are finalized. In line 17, the frequent itemsets are calculated for finalized window transactions using Eclat algorithm.

5. EXPERIMENTAL RESULTS:

We had performed experiments on the dataset generated by the synthetic dataset generator and the corresponding name convention to produce T40.I10.D100K dataset. Based on this name convention, the three numbers denote the average transaction size (T = 40), the average maximum potentially frequent itemset size (I = 10) and the total number of transactions (D = 100 K), respectively. Here we had taken 10000 transactions and performed the algorithm testing. We initially gave minimum window size as 10, maximum window size as 20, concept change window size is 60% of the normal window size i.e., 6 records will be stored in the concept change window and the concept variation threshold is given as 0.6. Then we had given 20% as support and 60% as confidence. Then with out selecting context variation checking, we got only 30 frequent item sets, as the window deletes the previous transactions and only stores the last coming transactions. When we selected concept variation checking, we got 2918 frequent itemsets. We had checked for several minimum supports and confidence, but when we took 20% as minimum support, then we got perfect differences. Thus the algorithm generated more and adequate frequent itemsets when the window is dynamically adjusted and concept change variation window is taken.

6. CONCLUSION

The results shown that the proposed concept change dynamic aware sliding window model gives more accurate frequent itemsets than the other algorithms. In this the amount of change in the incoming records is continuously monitored and window is dynamically updated. Experimental evaluations show that our algorithm effectively tracks the concept change during a data stream mining. Moreover, the length of window is reduced when the amount of change is greater than minimum change threshold given by the user. Though it is necessary to determine an initial window size for the algorithm, however it is adjusted during the data stream mining. Therefore, an improper value for this parameter without a prior knowledge does not affect the overall performance of the algorithm. Our algorithm requires from the user to input a minimum change threshold instead of a window size. It shows the amount of change that a user is interested to observe in the set of frequent patterns. In future, we are going to check this algorithm for different frequent itemset mining algorithms and check their performance comparisions.

REFERENCES:

[1]. Mahmood Deypir a, Mohammad Hadi Sadreddini, Sattar Hashemi, Towards a variable size sliding window model for frequent itemset mining over data streams.

[2]. Chandni Shah1 , Factors Influencing Frequent Pattern Mining on Stream Data.

[3]. Pramod S, O.P. Vyas, Survey on Frequent Item set Mining Algorithms.

[4]. Zijian Zheng, Ron Kohavi, Real World Performance of Association Rule Algorithms.

[5]. Zhen-Hui Song, Yi Li, Associative classification over Data Streams.

[6]. Mohamed Medhat Gaber, Arkady Zaslavsky and Shonali Krishnaswamy, Mining Data Streams: A Review.

[7]. Nan Jiang and Le Gruenwald, Research Issues in Data Stream Association Rule Mining.

[8]. Christian Borgelt, Efficient Implementations of Apriori and Eclat.

[9]. Lars Schmidt-Thieme, Computer-based New Media Group (CGNM), Algorithmic Features of Eclat.

[10]. Manku, G. S., & Motwani, R. (2002). Approximate frequency counts over data streams. In Proc. VLDB int. conf. very large databases (pp. 346–357).

[11]. Chang, J. H., & Lee, W. S. (2005). estWin: Online data stream mining of recent frequent itemsets by sliding window method. Journal of Information Science, 31(2), 76–90.

[12]. Mozafari, B., Thakkar, H., & Zaniolo, C. (2008). Verifying and mining frequent patterns from large windows over data streams. In Proc. int. conf. ICDE (pp. 179– 188).

[13]. Aggarwal, C. (2003). A framework for diagnosing changes in evolving data streams.

[14]. Koh, J.- L., & Lin, C.- Y. (2009). Concept shift detection for frequent itemsets from sliding window over data streams.

[15]. http://people.revoledu.com/kardi/tutorial/Similarity/Jaccard.html

[16]. Data stream mining - Wikipedia, the free encyclopedia

[17]. Jiawei Han (http://www.sal.cs.uiuc.edu/~hanj/DM_Book.html)

[18]. Vipin Kumar (http://www.users.cs.umn.edu/~kumar/csci5980/index.html)