An Efficient Algorithm for Mining Frequent Itemsets Within Large Windows Over Data Streams

Mahmood Deypir mdeypir@cse.shirazu.ac.ir
School of Engineering, Computer Science and Engineering
Shiraz University
Shiraz, 7134851154, Iran

Mohammad Hadi Sadreddini sadredin@shirazu.ac.ir
School of Engineering, Computer Science and Engineering
Shiraz University
Shiraz, 7134851154, Iran

Abstract

Sliding window is an interesting model for frequent pattern mining over data stream due to handling concept change by considering recent data. In this study, a novel approximate algorithm for frequent itemset mining is proposed which operates in both transactional and time sensitive sliding window model. This algorithm divides the current window into a set of partitions and estimates the support of newly appeared itemsets within the previous partitions of the window. By monitoring essential set of itemsets within incoming data, this algorithm does not waste processing power for itemsets which are not frequent in the current window. Experimental evaluations using both synthetic and real datasets shows the superiority of the proposed algorithm with respect to previously proposed algorithms.

Keywords: Data Stream Mining, Frequent Itemsets, Sliding Window, Support Estimation.

1. INTRODUCTION

A data stream is an infinite amount of data elements which receive at a rapid rate. By the emergence of the application of data stream in business, science and industry, mining this type of data becomes an attractive field in data mining community. Frequent patterns mining [1] over data streams is a challenging problem since it must be solved using minimum resources of main memory and processing power. In a data stream mining algorithm, data elements should be scanned only once due to the rapid data arrival rate [2]. Handling the concept change is another issue. Concept change in the frequent itemset mining problem is changes that occur in the set of frequent itemsets during a data stream mining. Although monitoring previous frequent itemsets in the newly arrived data is an easy task, it is hard to detect new frequent itemsets and computing their supports. Sliding window model is a widely used model to perform frequent itemset mining since it considers only recent transactions and forgets obsolete ones. Due to this reason, a large number of sliding window based algorithms have been devised [3-10]. However, a subset of these studies adaptively maintain and update the set of frequent itemsets [6-10] and others [3-5] only store sliding window transactions in an efficient way and perform the mining task when the user requests. In this study, a novel approach for mining frequent itemsets over data streams is proposed which operate under sliding window model. Experimental evaluations on real and synthetic datasets show the superiority of the proposed approach with respect to previous algorithms. The rest of the paper is organized as follows. The next section introduces some preliminaries and also states the problem. In section 3, some previous related studies are reviewed. Section 4 presents the proposed approach and section 5 empirically compares the approach to its competitors. Finally, section 6 concludes the paper.

2. PRELIMINARIES

Let I={i_1,i_2,…,i_m} be a set of items. Suppose that, DS be a stream of transactions received in sequential order. Each transaction of DS is a subset of I. For an itemset X, which is also a subset of I, a transaction T in DS is said to contain the itemset X if $X \subseteq T$. A transactional sliding window W over data stream DS contains |W| recent transactions in the stream, where |W| is the size of the window. The window slides forward by inserting a new transaction into the window and deleting the oldest transaction from the window. Due to efficiency issues, instead of a single transaction, the unit of insertion and deletion can be a partition (or batch) of transactions. In fact the window contains the n most recent partitions of transactions of the input stream. The first transaction id (Tid) of each partition is regarded as partition id (Pid) of that partition and first Pid of the window is named window id (Wid). An itemset X is said to be frequent in W if Freq(X) ≥ n×|P|×s, where Freq(X), n, |P| and s are frequency of X in W, number of the partitions in the window, partition size and the minimum support threshold, respectively. The number of transactions in each partition, i.e., partition size and number of partitions in each window are fixed during a data stream mining and are the parameters of the mining algorithm. Thus, having a partitioned transactional window W and a minimum support threshold s specified by the user, the problem is defined as mining all frequent itemsets that exists in window W. The results should be continuously updated when the window advances. Due to the rapid arrival rate of transactions, an approximate result of frequent itemsets is acceptable.

3. RELATED WORKS

There are a large number of studies related to frequent itemset mining over data streams. They are mainly belonging to different models of data stream processing including sliding window [3-10], landmark [11-13] and damped models [14, 15]. DSTree [3] and CPS-Tree [4] are two algorithms that use the prefix tree to store raw transactions of sliding window. DSTree uses a fixed tree structure in canonical order of branches while in CPS-Tree, the prefix tree structure is maintaining in support descending order of items to control the amount of memory requirement. Both of [3] and [4] perform the mining task using FP-Growth [16] algorithm that was proposed for static databases. In [5], an algorithm namely MFI-TransSW was proposed which is based on the Apriori algorithm [2]. MFI-TransSW uses a bit string to store the occurrence information of an item within sliding window. Moreover, it mines all frequent itemsets over recent window of transactions. All of [3-5] perform the mining task on the current window when a user requests and don’t adaptively maintain and update the mining result. Therefore, after the mining, when new transactions are arrived from the stream, obtained result becomes invalid for the user and thus the mining task need to be re-executed. Lin et al. [6] proposed a method for mining frequent patterns over time sensitive sliding window. In their method the window is divided into a number of batches for which itemset mining is performed separately. In this algorithm at each timestamp a couple of transactions namely a block are received from input stream. The sliding window contains fixed number of blocks. However, since each batch contains a different number of transactions, different windows over a data stream have various number of transactions. The Moment algorithm [7] finds closed frequent itemsets by maintaining a boundary between frequent closed itemset and other itemsets. In [9] the authors devised an algorithm for mining non-derivable frequent itemsets over data streams. This algorithm continuously maintains non-derivable frequent itemsets of the sliding window. Algorithm of [7] and [9] adaptively mine the concise representation of frequent patterns which are a subset of all set of frequent patterns. The SWIM [8] is a partition based algorithm in which frequent itemsets in one partition 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 partitions and incrementally updates their supports and prunes infrequent ones. Chang and Lee proposed the estWin algorithm [10] that finds recent frequent patterns adaptively over transactional data streams using sliding window model. It uses a reduced minimum support to early monitoring of new itemsets.

DSM-FI [11] is a landmark based algorithm. In this algorithm, every transaction is converted into smaller transactions and inserted into a summary data structure called item-suffix frequent itemset forest which is based on prefix-tree. In [12] the authors used the Chernoff Bound to produce an approximate result of frequent patterns over the landmark window. Zhi-Jun et al. [13] used a lattice structure, referred to as a frequent enumerate tree, which is divided into several equivalent classes of stored patterns with the same transaction-ids in a single class. Frequent patterns are divided into equivalent classes, and only those frequent patterns that represent the two borders of each class are maintained; other frequent patterns are pruned. Chang and Lee proposed an algorithm called estDec based on damped model in which each transaction has a weight decreasing with age [14]. In this method, in order to reduce the effect of old transactions in the set of frequent patterns, a decay rate is defined. In [15] an algorithm similar to the estMax is proposed for mining maximal frequent itemsets over data streams based on damped model.

4. THE PROPOSED ALGORITHM

In [6, 8], frequent itemsets of a new partition are mined using the FP-Growth [16] algorithm and since then the found frequent itemsets are checked against new partitions to update their support. The idea of these algorithms is based on the fact that each frequent itemset of the window are frequent in at least one partition of the window. The idea is inspired by the partitioning algorithm [17] for static databases in which, it is proved that a frequent itemset is frequent in at least one partition of a database. However, exploiting this criterion in data stream mining, increases the number of frequent itemsets that required to be monitored in the incoming transactions. The reason is that the reverse of this criterion is not correct. That is:

Theorem. An itemset that is frequent in a partition of the window might be infrequent in whole window.

Proof. An itemset which is frequent in a partition of the window might have low support in other partitions of the window. Therefore, its overall support in whole window might be smaller than the minimum support threshold. □

Based on the above statement, in our algorithm we don't monitor each frequent itemset of a new partition. Instead, for such frequent itemset we estimate their support in the previous partitions individually using their subsets. If the sum of estimated support and actual support of the itemset is greater than minimum support threshold, the itemset is inserted to the monitoring prefix tree and their support becomes verified in subsequent new partitions. Moreover, by expiring each partition of the window, support of itemsets in the prefix tree are updated. For each itemset, the process of updating continues until the itemset is frequent in the window. In our approach, the estimation of support is partition based estimation. That is, the support of an itemset is estimated in each partition of the window. Therefore, estimated support of an itemset is equal to sum of all estimated values. For an itemset, its actual support and estimated support in different partitions of the window are stored in the corresponding node of the prefix tree. When an itemset identified as frequent in a new partition, its support is estimated in previous partitions of the window. For an n-itemset its longest subsets have length of n-1 or smaller. For each previous partition first, among the supports of their subsets, minimum value is selected. Longer subsets are checked first since longer subsets have closer value to the actual support of the itemset. If actual values of long subsets are not contained in a partition, shorter ones are tested. Therefore, a high quality estimated value for the itemset in each partition is computed. An itemset in the new partition is inserted into the prefix tree if the following conditions are hold:

$$\begin{cases} F_n \geq Sup \\ \sum_{i=1}^{n-1} EF_i + F_n \geq Sup \end{cases} \quad (1)$$

Where EF_i and F_n are estimated support in each partition i and actual support of the itemset in the newly received partition. The window contains n partitions of transactions. Considering the estimated supports of previous partitions to insert and monitor the support of the itemset reduces the size of prefix tree and also enhances the processing time. Estimating support using actual counts of individual partitions improves the mining quality since more realistic value is obtained.

For a newly inserted node, actual support of the partition and estimated support of previous partitions are stored separately. For this node, its actual supports in subsequent partitions are stored since it is monitored in newly arrived partitions. This information of individual partitions of the window is used to remove the oldest partitions efficiently. Table 1 summarizes elements that are stored in the prefix tree nodes. In this prefix tree each node represents an itemset which can be induced in the path from the root to the node. Prefix sharing among the itemset in the tree, reduces the amount of memory requirement.

Element	Purpose
ID	Item ID
Cs	Actual Count of the itemset in partitions of the window
ECs	Estimated support of the itemset in partitions of the window
Children	Set of pointers to the children of the node

TABLE 1: Information contained in each node of prefix tree

After adding a newly arrived partition, to complete the window sliding phase, the oldest partition of transactions should be removed. For each node of the prefix tree, if corresponding itemset has estimated support in this partition, the value is removed. Otherwise, the actual support value in this partition is neglected. Therefore, the oldest partition removal process does not need FP-Tree of the oldest partition as in SWIM [8] or current transactions of the window as in estWin [10] algorithm. As result, removing obsolete information is performed using smaller memory and processing time. In the partition removal process, infrequent nodes and their descendents are also deleted. When information of the oldest partition is removed from a node, its support is reduced and if the support falls below the threshold the node and their descendent are removed recursively from the prefix tree. The reason for deletion of descendents is due to the Apriori principle which states that all supersets of an infrequent node are also infrequent.

A high level pseudo code of the proposed algorithm is shown in Figure 1. As shown in this Figure, for each itemset of the prefix tree (PT), its support is updated using the newly inserted partition (P). Frequent patterns of the new partition are found by applying the FP-Growth algorithm.

```
$\forall e \in PT\ update\ \ e.sup$
F = FP-Growth(P)
$\forall e \in F\ \ e.supp + estimated(e) \geq sup$
    $Insert\ e\ into\ PT$
$if\ |W| > n \times |P|$
    $\forall e \in PT\ remove\ oldest\ pane$
        If $e.supp < supp$
            $Erase\ e\ and\ all\ of\ its\ descendents\ recursively\ from\ PT$
```

FIGURE 1: The Proposed Algorithm

For each itemset of the new partition, if its support in the new partition in addition to its estimated support in the previous partitions is greater than or equal to minimum support threshold, it is inserted to the prefix tree. If the window size is greater than its specified number of the partition, the oldest partition must be removed from the window to preserve fixed size window. Hence, for each itemset in the prefix tree, its support information of the oldest partition containing estimated or actual support is deleted from the tree. By removing this information form corresponding arrays, if the itemset becomes infrequent, it and its subsets are erased from the tree.

In the proposed algorithm, information of equal sized partitions is separately stored in the prefix tree. The proposed approach can be also used in time sensitive window where at each timestamp a number of transactions are received from a stream. These transactions can be regarded as a new partition and processed according to the above described approach. However, since at each timestamp, different number of transactions is arrived, the partition size is not fixed during the stream mining process. Using the proposed approach, extracting old transactions from the window is performed efficiently. In [6], information of the partitions of the window are stored in tables which does not benefits from prefix sharing and requires large amount of memory and processing time. Moreover, all frequent itemsets of a new partition are monitored and their previous supports are estimated imprecisely.

5. EXPERIMENTAL EVALUATION

The proposed algorithm is experimentally evaluated with respect to previously proposed algorithms. The estWin and SWIM are selected for comparison since they similarly mine frequent itemsets adaptively over data streams. We have implemented all algorithm using C++ and STL template library. All experiments were conducted on P4 Intel CPU running Windows XP with 2 GB of RAM. We have compared the algorithms in terms of runtime since it is an important factor of every data stream mining algorithm. Two datasets are selected for experimentation. First dataset is a real dataset named BMS-POS and second dataset is a synthetic dataset generated using synthetic data generator [1]. Specifications of these datasets are summarized in the Table 1.

Dataset	#trans	#items	Max. length	Avg. length
BMS-POS	515,597	1657	164	6.53
T40I10D100K	100,000	942	77	39.61

TABLE 2: Datasets specifications

Since the value of minimum support threshold has direct effect on the runtime, the first experiment compares the algorithms using different values of this parameter on BMS-POS dataset. The results are shown in Figure 2.

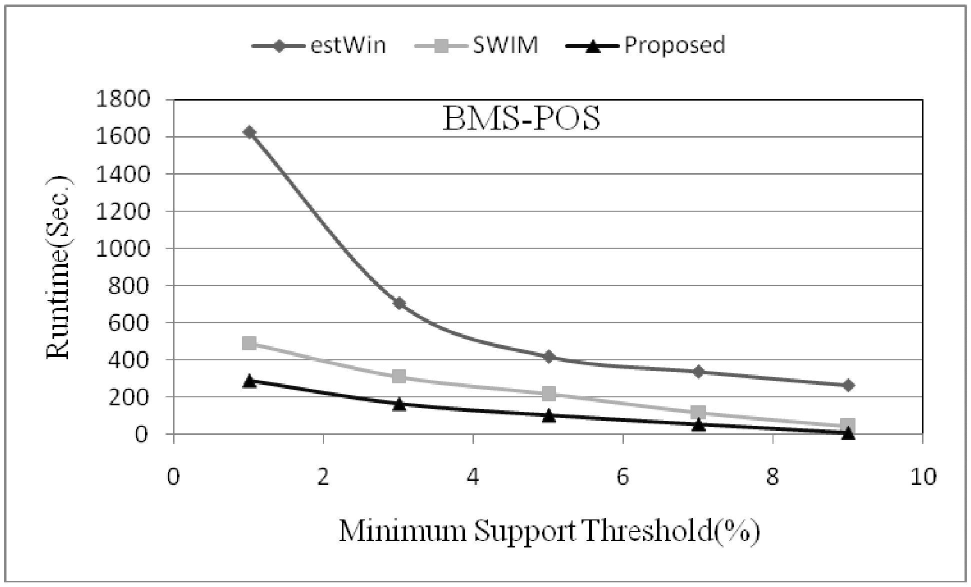

FIGURE 2: Runtime Comparison on BMS-POS

As shown in this figure, the proposed algorithm has the better runtime for different minimum support values. As the minimum support threshold decreases, the performance gap of our algorithm with respect to the other methods increases. The reason is, for lower minimum support thresholds, the number of frequent itemsets is increased. In this situation, SWIM requires to verify the support of a large number of patterns in different panes of the window. On the other hand, since the estWin uses a reduced minimum support value, i.e., significance instead of the actual threshold, the number of generated itemsets becomes prohibitively large. Therefore, the proposed algorithm operates better than both SWIM and estWin especially in lower minimum support thresholds.

The second experiment mesures the runtime for different values of minimum support thresholds on T40I10D100K synthetic dataset. The result is plotted on Figure 3.

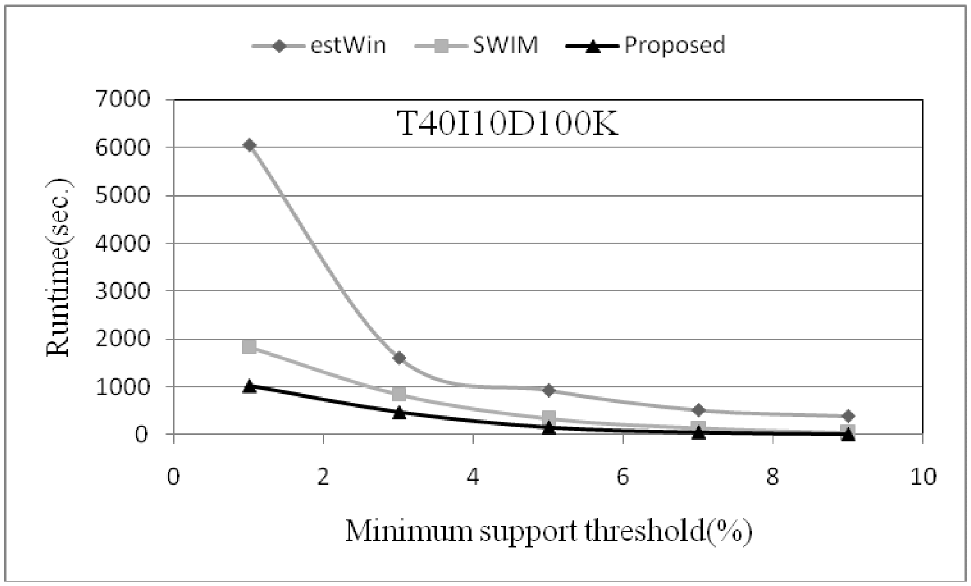

FIGURE 3: Runtime comparison on T40I10D100K

As shown in this figure, the proposed algorithm has lowest runtime. However, its runtime is closed to the SWIM algorithm. In addition to the above mentioned reasons, in both the SWIM and the proposed algorithms, the incoming transactions are batch processed while the estWin algorithm processes a single transaction at each sliding. Hence, both of them have better runtime. The SWIM stores transactional FP-Tree of each partition of the window to verify support value of new incoming partitions. On the other hand the proposed algorithm throws away the old transactions and estimates the support values of the new itemsets within previous partitions. Support estimation is faster than verifying and thus the proposed algorithm is faster than the SWIM.

6. CONCLUSION

In this study, a new algorithm for online frequent itemset mining over data streams is proposed. This algorithm has better runtime with respect to previously proposed estWin and SWIM algorithms. A prefix tree is only data structure used by the proposed algorithm while in the estWin and SWIM transactions of the window are also stored in addition to frequent itemsets of the current window. Therefore, the proposed algorithm has lower memory requirements. Although, the algorithm is proposed for operating in transactional window, it is also suitable for time sensitive window in which at any timestamp a different number of transactions are received from a stream.

7. RFERENCES

[1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules” in Proc. Int. Conf. on Very Large Databases, pp. 487–499, 1994.

[2] J. Han, H. Cheng, D. Xin, & X. Yan. “Frequent pattern mining: current status and future directions”, *Data Mining and Knowledge Discovery*,vol. 15(1), pp. 55–86, 2007.

[3] C.K.-S. Leung, & Q.I. Khan. "DSTree: a tree structure for the mining of frequent sets from data streams", Proc. ICDM, 928–932, 2006.

[4] S.K.Tanbeer, C. F. Ahmed, B.-S. Jeong, & Y.-K. Lee. "Sliding window-based frequent pattern mining over data streams", *Information Sciences*, vol. 179(22), pp. 3843-3865, 2009.

[5] H.-F. Li, S.-Y. Lee. "Mining frequent itemsets over data streams using efficient window sliding techniques", *Expert Systems with Applications*, vol. 36(2), pp. 1466–1477, 2009.

[6] C.-H. Lin, D.-Y. Chiu, Y.-H. Wu, & A.L.P. Chen. "Mining frequent itemsets from data streams with a time-sensitive sliding window", Proc. SIAM Int. Conf. Data Mining, 2005.

[7] Y. Chi, H. Wang, P.S. Yu, & R.R. Muntz. "Catch the moment: maintaining closed frequent itemsets over a data stream sliding window" *Knowledge and Information Systems*, 10(3), pp. 265–294, 2006.

[8] B. Mozafari, H. Thakkar, & C. Zaniolo. "Verifying and mining frequent patterns from large windows over data streams", Proc. Int. Conf. ICDE, pp.179–188, 2008.

[9] H. Li, & H. Chen. "Mining non-derivable frequent itemsets over data stream", *Data & Knowledge Engineering*, vol. 68(5), pp. 481-498, 2009.

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

[11] H. F. Li, S. Y. Lee, & M. K. Shan "An efficient algorithm for mining frequent itemsets over the entire history of data streams" Proc. Int. Workshop on Knowledge Discovery in Data Streams, 2004.

[12] J.X. Yu, Z. Chong, H. Lu, Z. Zhang, Z., & A. Zhou. "A false negative approach to mining frequent itemsets from high speed transactional data streams" *Information Sciences*, vol. 176(14), pp. 1986–2015, 2006.

[13] X. Zhi-Jun, C. Hong, & C. Li. "An efficient algorithm for frequent itemset mining on data streams" Proc. ICDM, 474–491, 2006.

[14] J. Chang, W. Lee, "Finding recently frequent itemsets adaptively over online transactional data streams", *Information Systems*, vol. 31 (8), pp. 849-869, 2006.

[15] J.H. Chang, W.S. Lee, "estMax: Tracing Maximal Frequent Itemsets Instantly over Online Transactional Data Streams", *IEEE Transactions on Knowledge and Data Engineering*, vol. 21 (10) pp.1418-1431, 2009.

[16] J. Han, J. Pei, Y. Yin, & R. Mao. "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach", *Data Mining and Knowledge Discovery*, vol. 8(1), pp. 53-87, 2004.

[17] A. Savasere, E. Omiecinski, and S. Navathe, "An efficient algorithm for mining association in large databases", in Proceeding of the VLDB International Conference on Very Large Databases, pp. 432–444, 1995.