An Efficient Approach for Mining Frequent Itemsets with Large Windows

K Jothimani[1], S. Antony Selvadoss Thanmani [2]

[1] Research Scholar, Research Department of Computer Science,
NGM College, 90, Palghat Road, Pollachi - 642 001
Coimbatore District, Tamilnadu, INDIA

[2] Professor and Head, Research Department of Computer Science,
NGM College, 90, Palghat Road, Pollachi - 642 001
Coimbatore District, Tamilnadu, INDIA

Abstract— The problem of mining frequent itemsets in streaming data has attracted a lot of attention lately. Even though numerous frequent itemsets mining algorithms have been developed over the past decade, new solutions for handling stream data are still required due to the continuous, unbounded, and ordered sequence of data elements generated at a rapid rate in a data stream. The main challenge in data streams will be constrained by limited resources of time, memory, and sample size. Data mining has traditionally been performed over static datasets, where data mining algorithms can afford to read the input data several times. The goal of this article analysing the mining frequent itemsets in theoretical manner in the large windows. By comparing previous algorithms we propose new method using analytical modelling to determine the factors over data streams.

Keywords— Data Mining, Data Streams Frequent Itemset Mining, Large Sliding Window.

I. Introduction

Recently the data stream, which is an unbounded sequence of data elements generated at a rapid rate, provides a dynamic environment for collecting data sources. It is likely that the embedded knowledge in a data stream will change quickly as time goes by. Therefore, catching the recent trend of data is an important issue when mining frequent itemsets from data streams. Although the sliding window model proposed a good solution for this problem, the appearing information of the itemsets within the sliding window has to be maintained completely in the traditional approach. In this paper, for estimating the approximate supports of patterns within the current sliding window, two data structures are proposed to maintain the average time stamps and frequency Changing points of patterns, respectively.

The main challenge is that 'data-intensive' mining is constrained by limited resources of time, memory, and sample size. Data mining has traditionally been performed over static datasets, where data mining algorithms can afford to read the input data several times. When the source of data items is an open-ended data stream, not all data can be loaded into the memory and off-line mining with a fixed size dataset is no longer technically feasible due to the unique features of streaming data [2].

Data from sensors like weather stations is an example of fixed-sized data, whereas again, market basket data are an example of variable size data, because each basket contains a different number of items. By contrast, sensor measurements have a fixed size, as each set of measurements contains a fixed set of dimensions, like temperature, precipitation, etc.

We classify the stream-mining techniques into two categories based on the window model that they adopt in order to provide insights into how and why the techniques are useful. Then, we further analyze the algorithms according to whether they are exact or approximate and, for approximate approaches, whether they are false-positive or false-negative. First, each element in the datastream can be examined only once or twice, making traditional multiple-scan approaches infeasible. Second, the consumption of memory space should be confined in a range, despite that data elements are continuously streaming into the local site. Third, not with standing the data characteristics of incoming stream may be unpredictable; the mining task should proceed normally and offer acceptable quality of results. Fourth, the latest analysis result of the data stream should be available as soon as possible when the user invokes a query

II .Related Work

There are a number of research works which study the problem of data-stream mining in the first decade of 21st century. Many previous studies contributed to the efficient mining of frequent itemsets (FI) in streaming data [4, 5]. According to the stream processing model [20], the research of mining frequent itemsets in data streams can be divided into three categories: *landmark windows* [15, 12, 19, 11, 13], *sliding windows* [5, 6, 14, 16, 17, 18], and *damped windows* [7, 4], as described briefly as follows. In the landmark window model, knowledge discovery is performed based on the values between a specific timestamp called landmark and the present. In the sliding window model, knowledge discovery is performed over a fixed number of recently generated data elements which is the target of data mining.

Two types of sliding widow, i.e., *transaction- sensitive sliding window* and *time-sensitive sliding window*, are used in mining data streams. The basic processing unit of window sliding of first type is an expired transaction while the basic unit of window sliding of second one is a time unit, such as a minute or a hour. In the damped window model, recent sliding windows are more important than previous ones.

Besides, there are still some interesting research works [9] [10] [11] on the sliding window model. In [9] a false-negative approach named *LSWIM* was proposed. By employing a progressively increasing function of ms, *LSWIM* greatly reduces the number of potential itemsets and would approximate the set of FIs over a sliding window. According to this definition, each transaction could be represented by the product of the corresponding prime numbers of individual items into the transaction. As the product of the prime numbers unique we can easily check the inclusion of two itemsets by performing a modulo division on itemsets (*Y* MOD *X*). If the remainder is 0 then $X \subseteq Y$, otherwise *X* is not included in *Y*.

LSWIM is an approximate approach based on the application of the *Principle of Inclusion and Exclusion* in *Combinatorial Mathematics* [7]. One of the most notable features of *LSWIM* is that it would approximate the count of an arbitrary itemset, through an equation (i.e., Equation (4) in [12]), by only the sum of counts of the first few orders of its subsets over the data stream. There are also two techniques named *counts bounding* and *correction*, respectively, integrated within *LSWIM*. Both techniques are of the purpose to improve the quality of *LSWIM*'s approximation, while they adopt different means to achieve the purpose. By working together with these original techniques, the mining result of DSCA reaches good accuracy. The concept of *Inclusion and Exclusion Principle* [7] is valuable that it may also be applied in mining FIs under different window models other than the landmark window. Based on the theory of *Approximate Inclusion–Exclusion* [8], we devise and propose a novel algorithm, called *SWIM*, to approximate dynamically and discover FIs over the sliding window in a transactional data stream

III. MINING LARGE SLIDING WINDOWS

LSWIM (Large Sliding Window Incremental Miner) algorithm relies on a verifier function and it is an exact and efficient algorithm for mining very large sliding windows over data streams. The performance of LSWIM improves when small delays are allowed in reporting new frequent itemsets, however this delay can be set to 0 with a small performance overhead.

A. Problem Statement and Notations

Let *D* be the dataset to be mined (a window in our case); *D* contains several transactions (baskets), where each basket contains one or more items. Let $I = i1, i2, \cdots, in$ be the set of all such distinct items in *D*. Each subset of *I* is called an itemset, and by *k*-itemset we mean an itemset containing *k* different items. The *frequency* of an itemset *p* is the number of transactions in *D* that contain itemset *p*, and is denoted as Count(*p*,*D*). The support of *p*, sup(*p*,*D*), is defined as its frequency divided by the total number of transactions in *D*. Therefore, $0 \leq sup(p,D) \leq 1$ for each itemset *p*. The goal of frequent itemsets mining2 is to find all such itemsets *p*, whose support is greater than (or equal to) some given minimum support threshold α. The set of frequent itemsets in *D* is denoted as $\sigma\alpha(D)$.

Here we consider frequent itemsets mining over a data stream, thus *D* is defined as a sliding window over the continuous stream. *D* moves forward by a certain amount3 by adding the new slide ($\delta+$) and dropping the expired one ($\delta-$). Therefore, the successive instances of *D* are shown as $W1, W2, \cdots$. The number of transactions that are added to (and removed from) each window is called its slide size. In this paper, for the purpose of simplicity, we assume that all slides have the same size, and also each window consists of the same number of slides. Thus, $n = |W|/|S|$ is the number of slides (a.k.a. panes) in each window, where $|W|$ denotes the window size and $|S|$ denotes the size of the slides.

B. The LSWIM Algorithm

Large Sliding Window Incremental Miner (LSWIM) always maintains a union of the frequent itemsets of all slides in the current window *W*, called Pattern Tree (*PT*), which is guaranteed to be a superset of the frequent itemsets over *W*. Upon arrival of a new slide and expiration of an old one, we update the true count of each pattern in *PT*, by considering its frequency in both the expired slide and the new slide. To assure that *PT* contains all itemsets that are frequent in at least one of the slides of the current window $\cup i(\sigma\alpha(Si))$, we must also mine the new slide and add its frequent itemsets to *PT*. The difficulty is that when a new pattern is added to *PT* for the first time, its true frequency in the whole window is not known, since this pattern wasn't frequent in the previous $n-1$ slides. To address this problem, LSWIM uses an auxiliary array, *aux array*, for each new pattern in the new slide.

The *aux array* stores the frequency of a pattern in each window starting at a particular slide in the current window. In other words, the *aux array* stores frequency of a pattern for each window, for which the frequency is not known. The key point is that this counting can either be done eagerly (i.e., immediately) or lazily. Under the laziest approach, we wait until a slide expires and then compute the frequency of such new itemsets over this slide and update the *aux array*s accordingly. This saves many additional passes through the window. The pseudo code for the LSWIM algorithm is given in Figure 1. At the end of each slide, LSWIM outputs all itemsets in *PT* whose frequency

at that time is $\geq \alpha \cdot n \cdot |S|$. However we may miss a few itemsets due to lack of knowledge at the time of output, but we will report them as delayed when other slides expire. The following mini-example shows how LSWIM works.

For Each New Slide *S*
1: For each pattern $p \in PT$
update *p.freq* over *S*
2: Mine *S* to compute $\sigma\alpha(S)$
3: For each existing pattern $p \in \sigma\alpha(S) \cap PT$
remember *S* as the last slide in which *p* is frequent
4: For each new pattern $p \in \sigma\alpha(S) \backslash PT$
$PT \leftarrow PT \in \{p\}$
remember *S* as the first slide in which *p* is frequent
create auxiliary array for *p* and start monitoring it
For Each Expiring Slide *S*
5: For each pattern $p \in PT$
update *p.freq*, if *S* has been counted in
update *p.aux array*, if applicable
report *p* as delayed, if frequent but not reported
at query time
delete *p.aux array*, if *p* has existed since arrival of *S*
delete *p*, if *p* no longer frequent in any of the current
slides
Fig. 1. LSWIM pseudo code.

Max Delay. The maximum delay allowed by LSWIM is $n-1$ slides. Indeed, after expiration of $n-1$ slides, LSWIM will have a complete history of the frequency of all frequent itemsets of *W* and can report them. Moreover, the case in which a pattern is reported after $(n-1)$ slides of time, is rare. For this to happen, pattern's support in all previous $n-1$ slides must be less than α but very close to it, say $\alpha \cdot |S|-1$, and suddenly its occurrence goes up in the next slide to say β, causing the total frequency over the whole window to be greater than the support threshold. Formally, this requires that $(n-1) \cdot (\alpha \cdot |S| - 1) + \beta \geq \alpha \cdot n \cdot |S|$, which implies $\beta \geq n+\alpha \cdot |S|-1$. This is not impossible, but in real-world such events are very rare, especially when *n* is a large number (i.e., a large window spanning many slides). While LSWIM(Delay=L) represents an efficient incremental mining algorithm, counting frequencies of itemsets over a given dataset ($n - L + 1$ slides in our case) remains a bottleneck. Therefore, faster algorithms are required to compute these counts efficiently.

IV. CONCLUSION

Mining data streams for association rules has proven to be a difficult problem, since techniques developed to mine frequent itemsets on stored data result in excessive costs and time delays. This paper has made two important contributions to the solution of this problem. The first is the introduction of a very fast algorithm to verify the frequency of a given set of itemsets. In fact, our algorithm outperforms the existing state of- the-art counting algorithms by an order of magnitude. The second contribution is to use our fast verifier to solve the association-rule mining problem under the realistic assumption that we are mostly interested in the new/expiring itemsets.

This delta-maintenance approach effectively mines very large windows with slides, which was not possible before. However, we also explored a second approach that further improves the performance by simply allowing a small reporting delay. Clearly this approach would become desirable in situations where modest delays in reporting new itemsets are acceptable. Such delays are negligible when compared to the time needed for the experts to validate the new rules before they are actually put into use. In summary we have proposed an approach of great efficiency, flexibility, and scalability to solve the frequent pattern mining problem on data streams with very large windows.

REFERENCES

[1] M.N. Garofalakis, J. Gehrke, & R. Rastogi, Querying and mining data streams: you only get one look (A Tutorial), Proc. 2002 ACM SIGMOD Conf. on Management of Data, Madison, Wisconsin, 2002, p. 635.
[2] Y. Zhu & D. Shasha, *"Stat Stream: statistical monitoring of thousands of data streams in real time"*, Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 358–369.
[3] G.S. Manku & R. Motwani, *"Approximate frequency counts over data streams"*, Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 346–357.

[4] J.H. Chang & W.S. Lee, *"A sliding window method for finding recently frequent itemsets over online data streams"*, Journal of Information science and Engineering, 20(4), 2004, pp. 753–762.

[5] J. Cheng, Y. Ke, & W. Ng*," Maintaining frequent itemsets over high-speed data streams"*, Proc. 10th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Singapore, 2006, pp.462–467.

[6] C.K.-S. Leung & Q.I. Khan, *"DSTree: a tree structure for the mining of frequent sets from data streams,"* Proc. 6th IEEE Conf. on Data Mining, Hong Kong, China, 2006, pp. 928–932.

[7] B. Mozafari, H. Thakkar, & C. Zaniolo, "*Verifying and mining frequent patterns from large windows over data streams",* Proc. 24th Conf. on Data Engineering, Mexico, 2008, pp. 179–188.

[8] K.-F. Jea & C.-W. Li, "*Discovering frequent itemsets over transactional data streams through an efficient and stable approximate approach, Expert Systems with Applications*", 36(10), 2009, pp. 12323–12331.

[9] F. Bodon, *"A fast APRIORI implementation"*, Proc. ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), 2003.

[10] N. Jiang and L. Gruenwald, *"Research Issues in Data Stream Association Rule Mining*". In SIGMOD Record, Vol. 35, No. 1, Mar. 2006.

[11] Frequent Itemset Mining Implementations Repository (FIMI). Available: http://fimi.cs.helsinki.fi/

[12] Y. Chi, H. Wang, P.S. Yu, & R.R. Muntz, *"Moment: maintaining closed frequent itemsets over a stream sliding window",* Proc. 4th IEEE Conf. on Data Mining, Brighton, UK, 2004, pp. 59–66.

[13] N. Jiang & L. Gruenwald, *"CFI-Stream: mining closed frequent Itemsets in data streams"*, Proc. 12th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, Philadelphia, PA,USA, 2006, pp. 592–597.

[14] Quest Data Mining Synthetic Data Generation Code. Available: http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data _mining/datasets/syndata.html

[15] H.F Li, S.Y. Lee, M.K. Shan, *"An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams*", In Proceedings of First International Workshop on Knowledge Discovery in Data Streams 9IWKDDS, 2004.

[16] H.F Li, S.Y. Lee, M.K. Shan, *"Online Mining (Recently) Maximal Frequent Itemsets over Data Streams*", In Proceedings of the 15th IEEE International Workshop on Research Issues on Data Engineering (RIDE), 2005.

[17] P. Indyk, D. Woodruff, *"Optimal approximations of the frequency moments of data streams"*, Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pp.202–208, 2005.

Smt. K Jothimani received her Bachelor degree in Computer Science from Bharathidasan University, Trichy in 2003. She received her Master degree in Computer Applications in 2008. She pursued her Master of Philosophy in Computer Science in the year 2009 from Vinayaka Missions University, Salem. Currently she is a research scholar of the Department of Computer Science, NGM College under Bharathiyar University, Coimbatore. She had three years of experience in the computer field in technical as well as non technical. She is a life member of Indian Society for Technical Education from the year 2009. Also she is a member of Computer Society of India(CSI). She has published more than ten papers in international and national conferences including standard journals. Her area of Interests Data mining, Knowledge Engineering and Image Processing.

Dr. Antony Selvadoss Thanamani is presently working as Professor and Head, Dept of Computer Science, NGM College, Coimbatore, India (affiliated to Bharathiar University, Coimbatore). He has published more than 100 papers in international/ national journals and conferences. He has authored many books on recent trends in Information Technology. His areas of interest include E-Learning, Knowledge Management, Data Mining, Networking, Parallel and Distributed Computing. He has to his credit 24 years of teaching and research experience. He is a senior member of International Association of Computer Science and Information Technology, Singapore and Active member of Computer Science Society of India, Computer Science Teachers Association, New York.