Knowledge-Based Systems 56 (2014) 281–298

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Efficient frequent itemset mining methods over time-sensitive streams ☆

Haifeng Li *, Ning Zhang, Jianming Zhu, Huaihu Cao, Yue Wang

School of Information, Central University of Finance and Economics, Beijing 100081, China

ARTICLE INFO

Article history:
Received 11 May 2013
Received in revised form 28 October 2013
Accepted 2 December 2013
Available online 13 December 2013

Keywords:
Stream
Frequent itemset
Data mining
Association rules
Time-sensitive

ABSTRACT

Stream data arrives dynamically and rapidly, and the characteristics cannot be reflected by the traditional transaction-based sliding window; thus, the mining results are inaccurate. This paper focuses on this problem and constructs a timestamp-based sliding window model, which can be further converted into a transaction-based sliding window. Based on this model, an extended enumeration tree is developed to incrementally maintain the essential information. In our proposed frequent itemset mining algorithm, we introduce the type transforming bound to dynamically classify the itemsets into categories; thus, certain itemset processing can be deferred or ignored, that is, an itemset will not be handled unless its type transforming bounds reach a threshold; as a result, the computational pruning can be conducted. Nevertheless, it only guarantees the conditions to obtain accurate results, and thus cannot achieve the best performance. This problem is further improved in our approximate mining algorithm, in which we propose a heuristic rule-based strategy. Additionally, it can save more computational cost with a tolerable mining error. Theoretical analysis and experimental studies demonstrate that our proposed algorithms have high accuracy, spend less computational time and memory, and significantly outperform the baseline method and state-of-the-art algorithms.

1. Introduction

Frequent itemset mining is a traditional and important problem in data mining. An itemset is frequent if its support is not less than a threshold specified by the users. Traditional frequent itemset mining approaches primarily consider this problem over static transaction databases. In these methods, transactions are stored in secondary storages such that multiple scans over the data can be performed. Three types of frequent itemset mining approaches over static databases have been proposed: reading-based approaches [1], writing-based approaches [2], and pointer-based approaches [3]. Ref. [4] presented a comprehensive survey of frequent itemset mining and discussed the research directions.

With the growing number of data stream applications [5], such as stock analysis, wireless sensor network management, web log analysis, and network monitoring, more studies have begun to focus on stream mining. Data streams have posed new challenges: A data stream is continuous, unlimited, and high speed, which means that data cannot be stored in secondary storages. Additionally, only one scan or limited scans can be performed when the data arrives; therefore, the stream mining algorithms must run in an incremental manner. Furthermore, the data distribution may dynamically change with time; thus, a concept-drift problem [6] must be solved.

1.1. Motivation

A stream is a type of data that is high speed, dynamic, unlimited and continuous. Most of the recent proposed stream mining methods use the sliding window model or the damping model, both of which have sizes that are generally based on the transaction numbers because this is a convenient way to achieve the frequent itemset with regard to the minimum support; nevertheless, these two models are based on the assumption that the transactions arrive at a constant speed, which is clearly not suited for most real world situations. In both commerce and in network applications, the stream data generated from large number of applications are time relative, that is, the number of arriving transactions changes at different time points.

For example, in a supermarket, the passenger flows are different at different time segments. The transactions are dense when the passenger flow reaches its peak; otherwise, the transactions are sparse. To obtain the frequent itemsets from the continuous arrived stream, a sliding window that covers a fixed number of

☆ This research is supported by the National Natural Science Foundation of China (61100112,61309030), National Social Science Foundation of China(13AXW010), Beijing Natural Foundation (4112053), Beijing Philosophy and Social Science Foundation (11JGC136), MOE Project of Humanities and Social Sciences (11YJCZH006), Discipline Construction Foundation of Central University of Finance and Economics.

* Corresponding author. Tel.: +86 01062288897.
E-mail addresses: mydlhf@cufe.edu.cn (H. Li), zhangning75@sina.com (N. Zhang), tyzjm65@163.com (J. Zhu), caohhu@163.com (H. Cao), yuelwang@163.com (Y. Wang).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.12.001

recent arrived transactions is usually used in traditional stream mining methods, which we called the transaction-based sliding window model. Nevertheless, this model has drawbacks. As shown in Fig. 1(a), assuming that each time segment $[T_i, T_{i+1})$ represents a half hour and that the administrator wants to know the 1-h sale analysis at T_6, i.e., the sales between T_4 and T_6, the analyzed data should be $\{ac, cd\}$. However, if we use a transaction-based sliding window model (as shown in Fig. 1(b)), and assume that the window size is 5, then the actual analyzed data are $\{bcd, ad, bcd, ac, cd\}$, which contains the sale data of two hours. Consequently, the mining results are not the ones the administrator needs. Thus, in such an application, we should determine the sliding window size by a time factor to satisfy the requirements of administrator. In other words, the sliding window that will be processed covers a fixed number of timestamps rather than a fixed number of transactions, which we call the timestamp-based sliding window model, as shown in Fig. 1(c), and it can be converted into the transaction-based sliding window model shown in Fig. 1(d). In this model, the absolute minimum support cannot be regarded as the threshold because the transactions covered by the sliding window change dynamically, meaning that the transactions number is not well distributed at different timestamps, which will result in inaccurate mining results. For instance, in Fig. 1(d), assuming that an itemset is frequent if half of the transactions in a sliding window cover it. There are 7 transactions in $[T_0, T_5)$, and the absolute minimum support is 4; thus, the frequent itemsets are $\{b : 4, c : 5, d : 5\}$. However, when the sliding window continues to $[T_1, T_6)$, if the absolute minimum support is still 4, then the frequent itemsets change to $\{c : 4, d : 4\}$. Nevertheless, there are only 5 transactions in this time segment, and 3 is a more reasonable absolute minimum support. That is, $\{cd : 3\}$ should be a frequent itemset, but it is ignored if we use the absolute minimum support to perform mining. Consequently, if we want to mine a stream based on the time-sensitive sliding window, a relative minimum support needs to be used as the threshold.

1.2. Problem definition

We use $T_I (I = 0 \cdots P)$ to denote a timestamp and $[T_{I-1}, T_I)$ to represent a time segment. Moreover, the stream transactions arrived within a time segment are regarded as a basic block. Accordingly, we present the problem addressed in this paper, which is, given a threshold λ, to generate frequent itemsets from the transactions within $[T_{C-N}, T_C)$ in a stream, where T_C denotes the current timestamp and T_{C-N} denotes the Nth timestamp before T_C. This problem is called the time-sensitive stream mining problem, and the transactions within $[T_{C-N}, T_C)$ form a timestamp-based sliding window. In such a model, an intuitive mining consideration to define the frequent itemsets is based on the timestamp: Given a threshold λ within (0,1), an itemset appearing at least λN times is frequent. Nevertheless, this definition cannot satisfy the real mining request. In the example shown in Fig. 1(a), if the administrator wants know the goods that sold well during $[T_3, T_5)$; given $\lambda = 0.8, a$ is frequent according to this definition, but in fact, only 2 in 5 transactions covered a, which is obviously not the results that the administrator wants. Although the stream is time-relative, the definition of a frequent itemset is time-irrelative once the time segments are specified. Thus, the traditional definition is more reasonable.

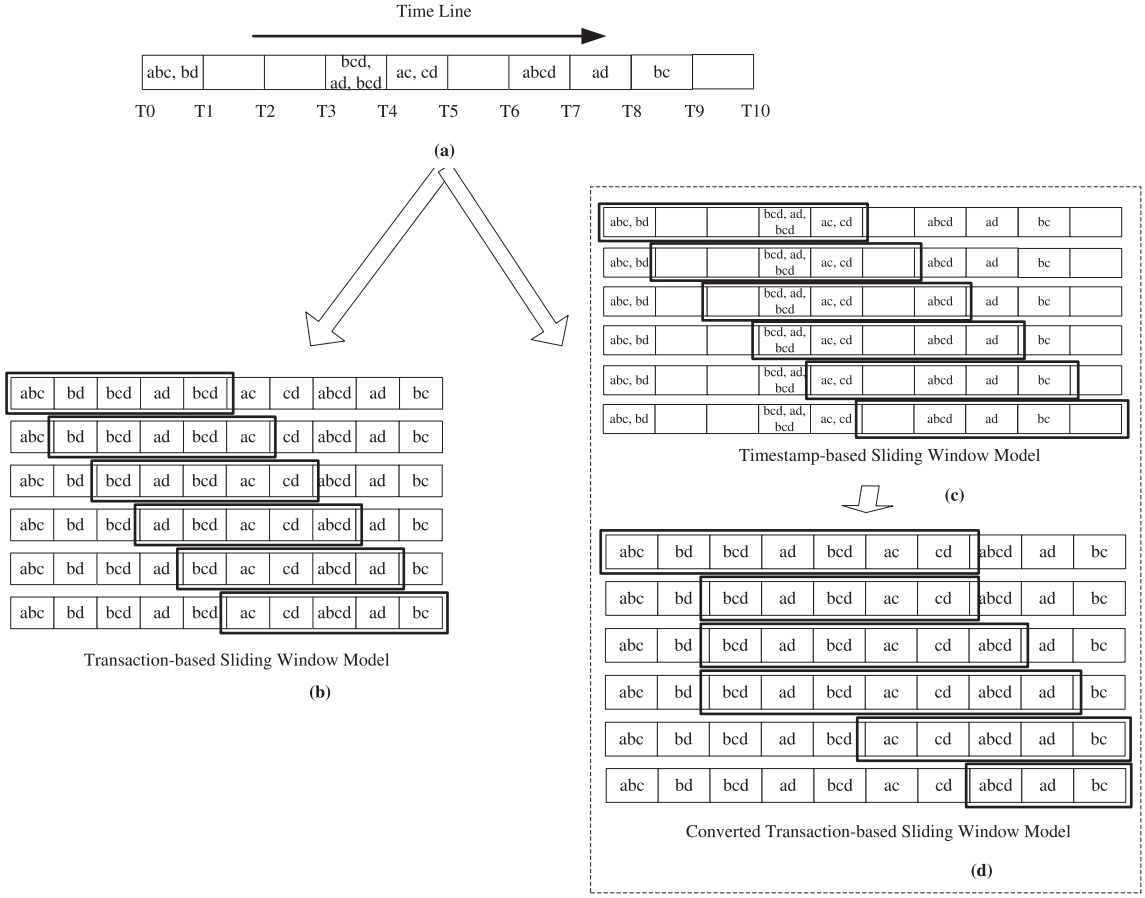

Fig. 1. Transaction-based sliding window vs. timestamp-based sliding window.