Effect of Count Estimation in Finding Frequent Itemsets over Online Transactional Data Streams

Joong Hyuk Chang and Won Suk Lee

Department of Computer Science, Yonsei University, Seoul 120-749, Korea
E-mail: jhchang@amadeus.yonsei.ac.kr; leewo@amadeus.yonsei.ac.kr
Received November 14, 2003; revised August 12, 2004.

Abstract A data stream is a massive unbounded sequence of data elements continuously generated at a rapid rate. Due to this reason, most algorithms for data streams sacrifice the correctness of their results for fast processing time. The processing time is greatly influenced by the amount of information that should be maintained. This issue becomes more serious in finding frequent itemsets or frequency counting over an online transactional data stream since there can be a large number of itemsets to be monitored. We have proposed a method called the estDec method for finding frequent itemsets over an online data stream. In order to reduce the number of monitored itemsets in this method, monitoring the count of an itemset is delayed until its support is large enough to become a frequent itemset in the near future. For this purpose, the count of an itemset should be estimated. Consequently, how to estimate the count of an itemset is a critical issue in minimizing memory usage as well as processing time. In this paper, the effects of various count estimation methods for finding frequent itemsets are analyzed in terms of mining accuracy, memory usage and processing time.

Keywords count estimation, frequent itemsets, transactional data streams

1 Introduction

A data stream is a massive unbounded sequence of data elements continuously generated at a rapid rate. Due to this reason, it is impossible to maintain all elements of a data stream. Consequently, online data stream processing should satisfy the following requirements[1]. First, each data element should be examined at most once to analyze a data stream. Second, memory usage for data stream analysis should be restricted finitely although new data elements are continuously generated in a data stream. Third, newly generated data elements should be processed as fast as possible to produce the up-to-date analysis result of a data stream, so that it can be instantly utilized upon request. To satisfy these requirements, data stream processing sacrifices the correctness of its analysis result by allowing some error.

In order to find frequent itemsets[2–4] accurately, the occurrence of every itemset in a data stream should be carefully monitored. When a new itemset appears in a newly generated transaction, it cannot be simply ignored just because it is not a frequent itemset currently. If it is ignored, there is no frequent itemset forever since every frequent itemset was a new one at its first appearance. In spite of this, it is almost impossible to monitor the actual count of every itemset that appears in a data stream. Such monitoring not only requires a large amount of main memory but also increases processing time. The actual count of an itemset whose current support is much less than a minimum support is not necessarily monitored since it cannot be a frequent itemset in the near future.

The Carma algorithm[5] introduces how to estimate the count of an itemset in a finite data set. It analyzes transactions in a data set one by one and has two phases. In the first phase, a candidate itemset is inserted into a lattice located in main memory if it is estimated as a potentially frequent itemset and all of its subsets are already maintained in the lattice. The appearance count of a newly inserted itemset is set to one initially and is incremented whenever it appears in the subsequent transactions. In the second phase, the correct count of each itemset in the lattice is computed by looking up the previous transactions of its insertion. Moreover, if any itemset is confirmed as an infrequent itemset, it is removed from the lattice. Consequently, a data set is scanned up to twice at most.

Recently, several algorithms[6,7] for frequency counting over a data stream are proposed. Especially, in the Lossy Counting algorithm[6], the set of frequent itemsets in a data stream is found when an error parameter ε as well as a minimum support is given. A set of newly generated transactions in a data stream is loaded together into a fixed-size buffer in main memory and they are batch-processed. The actual counts of all single items in the data stream are maintained separately in the buffer. A set of local itemsets for the new transactions in the buffer is generated by those single items whose current supports are greater than or equal to ε. The local count of every local itemset is identified by scanning the new transactions in the buffer. The information about the previous mining result up to the latest batch operation is maintained in a data structure called D containing a set of entries of a form (e,f,Δ), where e is an itemset, f is the count of the itemset e, and Δ is the maximum possible error count of the itemset e. In order to update the information of the data structure D, all of its entries are looked up in sequence. For the entry (e,f,Δ) of an itemset e in D,
if the itemset $e$ is one of the local itemsets identified by the new transactions in the buffer, its previous count $f$ is incremented by its local count. Subsequently, when $N$ denotes the total number of transactions so far including the newly generated transactions in the buffer, if its estimated count, i.e., $f+\Delta$ is less than $\varepsilon \times N$, it is pruned from $D$. On the other hand, when there is no entry in $D$ for a local itemset $e$, a new entry $(e, f, \Delta)$ is inserted to $D$. Its maximum possible count $\Delta$ is set to $[\varepsilon \times N^2]$, where $N^2$ denotes the number of transactions that were processed up to the latest batch operation. This is because $[\varepsilon \times N^2]$ is the maximum possible count that could be missed for the itemset under the given error parameter $\varepsilon$.

In [8], we have proposed an algorithm called the estDec method for finding frequent itemsets over an online data stream adaptively in order to support flexible trade-off between memory usage and mining accuracy. In this algorithm, the current set of monitored itemsets in an online data stream is minimized by two major operations: delayed-insertion and pruning. The former is delaying the insertion of a new itemset in new transactions until the itemset becomes significant enough to be monitored. The count of an itemset that is delayed-inserted is estimated by the counts of its subsets. The latter is pruning a monitored itemset when the itemset turns out to be insignificant. The number of monitored itemsets can be flexibly controlled by the thresholds of these two operations.

This paper introduces several count estimation methods that can be employed to estimate the count of a new itemset in the estDec method. In addition, the effects of the count estimation methods on the performance of finding frequent itemsets are analyzed in terms of mining accuracy, memory usage and processing time. Consequently, given the specific requirements of an individual application domain, the most appropriate one among the proposed count estimation methods can be chosen based on this analysis.

The rest of this paper is organized as follows. Section 2 overviews a method of finding frequent itemsets over an online data stream. Section 3 introduces several count estimation methods. In Section 4, the performance of each count estimation method is comparatively analyzed by a series of experiments in terms of mining accuracy, memory usage and processing time. Finally, Section 5 concludes this paper.

2 Preliminaries

For finding frequent itemsets, a data stream can be viewed as an infinite set of continuously generated transactions as follows.

- Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of current items that have ever been used as a unit information of an application domain.
- An itemset $e$ is a set of items such that $e \in (2^I - \{\emptyset\})$, where $2^I$ is the power set of $I$. The length $|e|$ of an itemset $e$ is the number of items that form the itemset and an itemset with $m$ items is called an $m$-itemset. An itemset $\{a, b, c\}$ is denoted by $abc$.
- A transaction is a non-empty subset of $I$ and each transaction has a unique transaction identifier $TID$. A transaction generated at the $k$-th turn is denoted by $T_k$ and its transaction identifier $TID$ is $k$.
- When a new transaction $T_k$ is generated, the current data stream $D_k$ is composed of all transactions that have ever been generated so far, i.e., $D_k = (T_1, T_2, \ldots, T_k)$.

When a transactions $T_k$ is generated currently, the current count $C_k(e)$ of an itemset $e$ is the number of transactions that contain the itemset among the $k$ transactions in the current data $D_k$. Likewise, the current support $S_k(e)$ of an itemset $e$ is the ratio of its current count $C_k(e)$ over $k$. For the current data stream $D_k$, the count $C_k(e)$ and support $S_k(e)$ of an itemset $e$ are defined respectively as follows:

\[
C_k(e) = \sum_{i=1}^k W_i(e) \\
S_k(e) = \frac{1}{k} \sum_{i=1}^k W_i(e) \\
W_i(e) = \begin{cases} 
1, & \text{if } e \subseteq T_i; \\
0, & \text{otherwise.}
\end{cases}
\]

Given a predefined minimum support $S_{\min} \in (0, 1)$, an itemset $e$ whose current support $S_k(e)$ is greater than or equal to $S_{\min}$ is a frequent itemset in the current data stream $D_k$. In this paper, an itemset is called a significant itemset if its current support $S_k(e)$ is greater than or equal to a predefined significant support $S_{\text{sig}} \in (0, S_{\min})$.

The estDec method[8] examines each transaction in a data stream one-by-one without any candidate generation. Among all the itemsets in the transactions of a data stream, only those itemsets that should be monitored closely are maintained in main memory by a lexicographic tree structure[3,4] called a monitoring tree. Every node in a monitoring tree contains an item and it denotes an itemset composed of the items that are in the nodes of its path from the root. In addition, it maintains the count of the itemset in the current data stream $D_k$. The effect of the information in an old transaction on the current mining result is diminished by decaying the old occurrence count of an itemset as time goes by. The target of the estDec method is finding recent frequent itemsets in a data stream and the weight of information in each transaction is differentiated. However, in order to concentrate on how to estimate the count of an itemset in a data stream, the weight of all information in transactions of a data stream is regarded as the same in this paper like most conventional approaches to finding frequent itemsets in a data stream.

When a new transaction $T_k$ is generated in the current data stream $D_k$, the following steps except Step 4 are performed in sequence to reflect the information of