Information Sciences 257 (2014) 54–69

Contents lists available at ScienceDirect

Information Sciences

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

Mining frequent items in data stream using time fading model

Ling Chen [a,b,*], Qingling Mei [a]

[a] Department of Computer Science, Yangzhou University, Yangzhou 225127, China
[b] State Key Lab of Novel Software Tech, Nanjing University, Nanjing 210093, China

ARTICLE INFO

Article history:
Received 24 March 2012
Received in revised form 26 August 2013
Accepted 1 September 2013
Available online 10 September 2013

Keywords:
Stream data mining
Frequent data item
Fading factor
Hash function

ABSTRACT

We investigate the problem of finding frequent items in a continuous data stream, and present an algorithm named λ-*HCount* for computing frequency counts of stream data based on a time fading model. The algorithm uses r hash functions to estimate the density values of stream data items. To emphasize the importance of recent data items, a time fading factor is used. For a given error bound, our algorithm can detect approximate frequent items under a certain probability using limited number of memory space. The memory requirement only depends on the number of different data items and the number of hash functions used. Experimental results on synthetic and real data sets show that our algorithm outperforms other methods in terms of accuracy, memory requirement, and processing speed.

1. Introduction

In recent years, more attentions have been paid to stream data mining. Detecting frequent data items is an important task in stream data analysis. Frequency is a fundamental characteristic in many data mining tasks such as association rule mining and iceberg queries. It has applications in many areas such as sensor data mining, business decision support, analysis of web query logs, direct marketing, network measurement, and internet traffic analysis. Correspondingly, the stream data could be stock tickers, bandwidth statistics for billing purposes, network traffic measurements, web-server click streams, and data from sensor networks.

Traditional mining algorithms assume a finite dataset and multiple scans on the data. For the stream data applications, the volume of data is usually too large to be stored in memory or to be scanned for more than once. For data streams, data items can only be sequentially accessed, and random access is prohibited. Therefore, traditional frequent item mining algorithms are not applicable to stream data. Furthermore, because of the high throughput of the data streams, possibly in the speed of gigabytes per second, any feasible algorithm for detecting frequent data item must perform data processing and query fast enough so as to match the speed of arriving data in the stream. In addition, the algorithm can use only limited memory space and store only the sketch or synopsis of the data items in the stream.

Solutions for finding frequent items in stream data have been proposed recently. Several algorithms use random sampling [10,12,13,16,17,20,25,32,34,39,45] to estimate the frequencies of the data items. For example, the *Sticky Sampling* [34] algorithm presented by Manku and Motwani is a sampling based algorithm for computing an ε-deficient synopsis over a data stream. It is a probabilistic one-pass algorithm that provides an accuracy guarantee on the set of frequent data items and their frequencies reported. The second class of such algorithms are deterministic algorithms [1,5,11,19,27,31,37,38]. The *MG* algorithm by Misra and Gries [37] is a well-known deterministic algorithm to detect frequent stream data.

* Corresponding author at: Department of Computer Science, Yangzhou University, Yangzhou 225127, China. Tel.: +86 514 87870026; fax: +86 514 87887937.
E-mail addresses: yzulchen@gmail.com (L. Chen), mql859@163.com (Q. Mei).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.09.007

In many applications, recent data in the stream is more meaningful. For instance, in an athlete ranking system, more recent records typically should carry more weight. One way to handle such problem is to use a sliding window model [2,4,18,22,30,42]. In this model, only the most recent data items in a time period of a fixed length are stored and processed, and only the frequent data items in this period are detected. The advantage of this method is that it can get rid of the stale data and only consider the fresh data, which are more meaningful in many cases. To emphasize the importance of the recent data, time fading model [8,35,43,44] also can be used for frequency measures in data stream. In this model, data items in the entire stream are taken into account to compute the frequency of each data item, but more recent data items contribute more to the frequency than the older ones. This is achieved by introducing a fading factor λ ($0 < \lambda < 1$). A data item which arrived n time points in the past is weighted λ^n. Thus, the weight is exponentially decreasing. In general, the closer to 1 the fading factor λ is, the more important the history is taken into account. There are two advantages of the time fading model over the sliding window model. One is that in the time fading model, frequency takes into account the old data items in the history, while the sliding window model only observes within a limited time window and entirely ignores all the data items outside the window. This is undesirable in many real applications. The second is that in the time fading model, when more data arrive continuously, the frequency changes smoothly without a sudden jump which may occur in the sliding window model.

In real world applications, since there could be a huge number of different data items in the stream, it is impracticable to set a counter for each data item. But when using limited memory space, we cannot keep exact frequency counts for all possible items. Usually, an error bound is predefined by the user, and we can obtain an approximate result using less memory space. Therefore, it is necessary to tackle the problem of finding the right form of data structure and related construction algorithm so that the required frequency counts can be obtained with a bounded error for unbounded input data and limited memory. In solving this problem, we may end up facing a dilemma. That is, by setting a small error bound, we achieve high accuracy but suffer in terms of efficiency. On the contrary, a bigger error bound improves the efficiency but seriously degrades the mining accuracy. Therefore, we need to achieve good balance between the accuracy of the results and memory requirement. In this work, we investigate the problem of detecting approximate frequent items using limited number of memory space under a certain probability.

In this paper, we present an algorithm called λ-*HCount* for computing frequency counts over a user specified threshold on a data stream based on the time fading model. A fading factor λ is used to emphasize the importance of the more recent data items. For a given error ε and a threshold s of density, our algorithm can detect ε-approximate frequent items using $\frac{e(1-\lambda)}{\varepsilon^2}\ln\left(-\frac{M}{\ln p}\right)+\frac{r}{s-\varepsilon}$ memory space under the probability of p, here$M$ is the number of different data items, r is the number of hash functions used. Experimental results on synthetic and real data sets show that our algorithm λ-*HCount* outperforms other methods in terms of accuracy, memory requirement, and processing speed.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 formally defines the problem and describes the time fading model. Section 4 describes the framework of our algorithm λ-*HCount* while Section 5 analyzes its space and time complexity. Section 6 reports and analyzes our experimental results and Section 7 gives conclusions.

2. Related work

Problems related to frequency estimate have been actively studied. Many algorithms for identifying frequent items and other statistics in the entire data stream have been proposed.

Lossy Counting [34] was among the first algorithms for finding frequent items from a data stream. *Lossy Counting* is a one-pass algorithm that provides an accuracy guarantee on the set of frequent data items and their frequencies reported. Given a user-specified support threshold s, and an error threshold ε, *Lossy Counting* guarantees that: (1) All items whose true frequency exceeds sn are detected, where n is the total number of data items processed. Namely, there are no false negatives. (2) No item whose true frequency is less than $(s\text{-}\varepsilon)n$ is output. (3) The estimated frequency of any item is at most εn less than its true frequency. Homem and Carvalho [21] presented an algorithm for identifying the k most frequent elements by merging the commonly used counter-based and sketch-based techniques. The algorithm also provides guarantees on the expected error estimate, order of elements and the stochastic bounds on the error. Karp et al. [27], and Demaine et al. [11] applied a deterministic *MG* algorithm [37] to detect frequent stream data. They reduced the time for processing one data item in *MG* algorithm to $O(1)$ by managing all counters in a hash table. The algorithm can easily be adapted to find ε-approximate frequent items in the entire data stream without making any assumption on the distribution of the item frequencies. This algorithm needs $1/\varepsilon$ counters for the most frequent data items in the stream. Processing the arrival data items entails incrementing or decrementing some counters.

Many algorithms for frequent item detecting use random sampling. They make assumptions on the distribution of the item frequencies, and the quality of their results is guaranteed probabilistically. Whang et al. [39] proposed a probabilistic algorithm to estimate the number of distinct items in a large collection of data in a single pass. Golab et al. [17] gave an algorithm for the case when the item frequencies are multinomial-distributed. Gibbons and Matias [16] presented sampling algorithms to recognize top-k queries. Liu et al. [32] presented an error-adaptive and time-aware maintenance algorithm for frequency counts over data streams. Manku and Motwani [34] advanced a sampling based algorithm called *sticky sampling* for computing an ε-deficient synopsis over a data stream of singleton items. The algorithm scans the stream data and randomly samples the data items based on three user-specified parameters: support s, error bound ε, and probability of