Random Sampling over Data Streams for Sequential Pattern Mining

Chedy Raïssi, Pascal Poncelet

Random Sampling over Data Streams for Sequential Pattern Mining

Chedy Raïssi
LIRMM, EMA-LGI2P/Site EERIE
161 rue Ada
34392 Montpellier Cedex 5, France
France
raissi@lirmm.fr

Pascal Poncelet
EMA-LGI2P/Site EERIE
Parc Scientifique Georges Besse
30035 Nîmes Cedex, France
Pascal.Poncelet@ema.fr

February 16, 2007

Abstract

In recent years the emergence of new real-world applications such as network traffic monitoring, intrusion detection systems, sensor network data analysis, click stream mining and dynamic tracing of financial transactions, calls for studying a new kind of model. Named *data stream*, this model is in fact a continuous and potentially infinite flow of information as opposed to finite and statically stored data sets. We study the problem of sequential pattern mining in data streams. This problem has been extensively studied for the conventional case of disk resident data sets. In the case of data streams, this problem becomes more challenging as the volume of data is usually too huge to be stored on permanent devices, main memory or to be scanned thoroughly more than once. In this case, it may be acceptable to generate approximable solutions for our mining problem. In this paper we introduce a new approach based on biased reservoir sampling to achieve a more efficient mining of sequential patterns. Furthermore, we theoretically prove that our biased reservoir size is always bounded whatever the size of the stream is. This property often allows us to keep the entire relevant reservoir in main memory. We also show a simple algorithm to build the biased reservoir for the special case of sequential pattern mining. Experimental evaluation supports the claim that sequential pattern mining based on biased reservoir sampling needs small memory requirements. Besides, we also propose an adapted approach to handle the case of mining sequential patterns in a sliding window model. The experiment show that the results are accurate.

1 Introduction

Recently, the data mining community has focused on a new challenging model where data arrives sequentially in the form of continuous rapid streams. It is often referred to as data streams or streaming data. Since data streams are continuous, high-speed and unbounded flow of informations, it is often impossible to mine patterns with classical algorithms that require multiple scans. As a consequence new approaches were proposed to mine itemsets [5, 3, 2, 4, 8] using different approaches based on the *landmark*, *sliding windows* or *time-fading* models. However, few researches focused on sequential patterns extraction over data streams. In this paper we consider that transactions are ordered into the streams and grouped under different identifiers. We propose a new approach to mine sequential patterns based on the maintenance of a synopsis of the data stream. This proposition is motivated by the fact that the volume of data in real-world data streams is usually too huge to be efficiently mined and that an approximate answer for mining tasks is largely acceptable. In other words, in the data stream model one has to trade off accuracy against efficiency. A number of synopsis structures have been developed in recent years like sketches, sampling, wavelets and histograms. Our method belongs to the class of reservoir sampling. The reservoir sampling method is very easy to understand as it generates a sample of the original data representation. However, the classical unbiased reservoir method is inaccurate for data streams, this is due to the fact that when the the stream length increases, the accuracy of the reservoir decreases as it will contain a large portion of points from the distant history of the stream (the probability of successive insertions of new points reduces with the progression of the stream) and in an evolving data stream only the more recent data may be relevant for many mining tasks. To overcome this problem we use a biased reservoir sampling based on a temporal bias function in order to regulate the choice of the stream sample.

2 Preliminary Concepts

2.1 Sequential Patterns

The traditional sequence mining problem was first introduced in [7] and extended in [6].

Let $\mathcal{D}$ be a database of customer transactions where each transaction T consists of :

1. a customer-id, denoted by C_{id}
2. a transaction time, denoted by *time*
3. a set of items (called *itemset*) involved in the transaction, denoted by *it*

Definition 1 (Sequence) *Let $\mathcal{I} = \{i_1, i_2 ... i_m\}$ be a finite set of literals called items. An itemset is a non-empty set of items. A sequence S is a set of itemsets ordered according to their timestamp. It is denoted by $< it_1\ it_2\ ... it_n >$, where it_j, $j \in 1...n$, is an itemset. A k-sequence is a sequence of k items (or of length k).*

Definition 2 (Support) *Let C_{trans} be the ordered list of transactions for a single customer C (the maximal sequence supported by C). The support of a sequence S in a transaction database $\mathcal{D}$, denoted by $Support(S, \mathcal{D})$, is defined as:*

$$Support(S, \mathcal{D}) = \frac{|\{C \in \mathcal{D} | S \preceq C_{trans}\}|}{|\{C \in \mathcal{D}|}$$

Given a minimal support threshold, the problem of sequential pattern mining is to extract all the sequences S in $\mathcal{D}$ such that $Support(S, \mathcal{D}) \geq \sigma$.

2.2 Biased reservoir sampling

Reservoir sampling was first introduced in [9], in this method the first n points in the data stream are stored at the initialization step, when the $(t+1)^{th}$ is received in the data stream it replaces randomly one of the points in the reservoir with probability $\frac{n}{t+1}$. As the stream length increases the probability of the insertion reduces. This is a clear disadvantage for mining tasks that consider that recent information provided by the stream is the most relevant. One solution proposed in [1] was to use an exponential bias function defined as follow : $f(r, t) = e^{-\lambda(t-r)}$

with parameter λ being the bias rate. The aim of this bias function is to regulate the choice of the stream sample. In other words, the bias function modulates the sample in order to focus on recent or old behaviors in the stream following application specific constraints. Moreover, the inclusion of an exponential bias function implies also an upper bound on the reservoir size which is independant of the stream length. For a stream of length t, let $R(t)$ be the maximal size of the reservoir which satisfies the exponential bias function, we have $R(t) \leq \frac{1}{\lambda}$.

In the next section we give results on the bounds for sample size, given the desired accuracy of the results in terms of support and errore rate.

3 Sampling analysis

The first question that one has to answer when sampling for mining tasks is : how accurate my sample is compared to my original data set? We answer to this question by giving exact bounds on the size of the sample w.r.t an error rate.

Definition 3 (Error rate) *Let $\mathcal{D}$ be a database of customer transactions and $\mathcal{S}_\mathcal{D}$ a random sample generated from $\mathcal{D}$. Let s be a sequence from $\mathcal{D}$. The absolute error rate in terms of support estimation, denoted ϵ, is defined as :*

$$e(s, \mathcal{S}_\mathcal{D}) = |Support(s, \mathcal{S}_\mathcal{D}) - Support(s, \mathcal{D})|$$

Let X_i be a random variable for the i^{th} customer with $Pr[X_i = 1] = p_i$ if the i^{th} customer supports the sequence s and $Pr[X_i = 0] = 1 - p_i$, if not. All the X_i are independant. Note that we are in presence of Poisson trials as the number t of trials in which the probability of success p_i varies from trial to trial. Let $X(s, \mathcal{S}_\mathcal{D}) = \sum_i X_i = Support(s, \mathcal{S}_\mathcal{D}) \times |\mathcal{S}_\mathcal{D}|$ be the number of customers in the sample that supports the sequence s. Then the expected number of customers that support the sequence s in the sample is $E[X(s, \mathcal{S}_\mathcal{D})] = Support(s, \mathcal{D}) \times |\mathcal{S}_\mathcal{D}|$. We would like to estimate the probability that our error rate gets higher than a user defined threshold ϵ, denoted $Pr[e(s, \mathcal{S}_\mathcal{D}) > \epsilon]$.

Using Chernoff bounds the following theorem gives us a lower bound on the size of the reservoir given ϵ and a maximum probability δ that the error rate exceeds ϵ:

Theorem 1 *Given a sequence s then $Pr[e(s, \mathcal{S}_\mathcal{D}) > \epsilon] \leq \delta$ iff the reservoir size is $|\mathcal{S}_\mathcal{D}| \geq \ln(\frac{2}{\delta})\frac{1}{2\epsilon^2}$*

As we are working on biased reservoir samples, the following corollary gives an upper bound on the bias rate :

Corollary 1 *Given an error bound ϵ and a maximum probability δ that $e(s, \mathcal{S}_\mathcal{D}) > \epsilon$ we get an upper bound on the bias rate:*

$$\lambda \leq \frac{2\epsilon^2}{ln(2/\delta)}$$

4 Algorithm

Based on the sampling analysis results we built an algorithm that achieve exponential bias with the λ parameter. Unlike the algorithms presented in [1], our approach need to take into account the constraint of the lower bound of the size of the reservoir. Note that the reservoir size is defined in term of customers number and not in term of transaction numbers. That means that insertion and delete operations must be done at the customers level *and* at the itemsets level.

Algorithm 1: Reservoir Sampling for Sequential Patterns algorithm

Data: Reservoir $\mathcal{S}_\mathcal{D}$;Bias rate λ;Transaction T
Result: Reservoir $\mathcal{S}_\mathcal{D}$ after insertion of the $(t+1)^{th}$ transaction

```
1  // $F(t) = \frac{q}{|\mathcal{S}_\mathcal{D}|} \in [0,1]$ is the fraction of the reservoir filled at the arrival of the $t^{th}$
   transaction
2  // $I(C_i,t) = \frac{i}{|itemsetList|} \in [0,1]$ is the fraction of the itemsets list for customer $C_i$ at the
   arrival of the $t^{th}$ transaction
3  if $T.C_i \notin \mathcal{S}_\mathcal{D}$ then
4      // Deterministic insertion of the transaction $T$ with its customer id $C_i$
5      $Coin \leftarrow Random(0,1)$;
6      if $Coin \leq F(t)$ then
7          // Success case: we replace one of the customers with all its itemsets with $T$.
8          $pos \leftarrow Random(0,q)$;
9          Replace($T.C_{pos}.it$,$T.C_i.it$);
10     else
11         //Failure case: we directly add the transaction $T$ without replacement.
12         Add($T.it$,$\mathcal{S}_\mathcal{D}$);
13         q++;
14 else
15     // A sample of customer $C_i$ transactions is already present in the reservoir
16     // Deterministic insertion of the transaction $T$ in $C_i$ itemsets list
17     $Coin \leftarrow Random(0,1)$;
18     if $Coin \leq F(t)$ then
19         $pos \leftarrow Random(0,i)$;
20         ReplaceItemset($it_{pos}$,$T.C_i.it$);
21     else
22         AddItemset($T.it$,$C_i.itemsetList$);
23         i++;
```

5 Experimentation

The experiments were performed on a Core-Duo 2.16 Ghz MacBook Pro with 1GB of main memory, running Mac OS X 10.4.6. We performed several tests with sythetic datasets that were generated with the IBM QUEST synthetic data generator, our data stream is divided into batches of period 25 seconds, each batch contains from 25k to 50k transactions. The memory management is the main focus of our performance study.

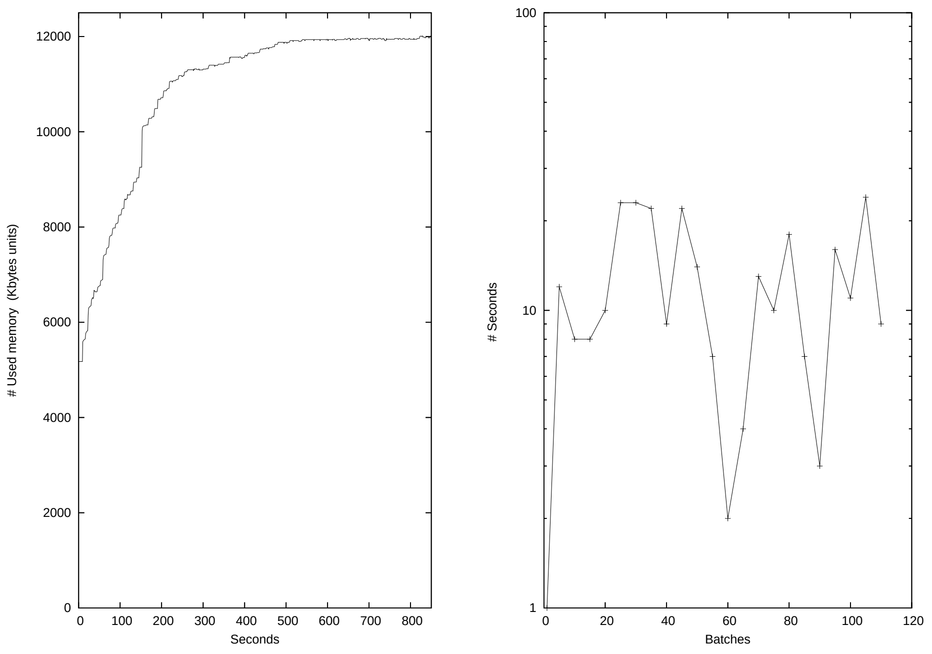

Figure 1: memory usage and time requirements for data set C1200I10K with reservoir sizes $\lambda = 2.10^{-5}$

6 Summary

In this presentation we introduced a new biased reservoir sampling algorithm for sequential pattern mining over data streams. The sampling analysis shows that we can efficiently bound our error rate to get approximate but acceptable results on our mining task. The experiments shows that our reservoir memory requirement are very low.

References

[1] Charu C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 607–618. ACM, 2006.

[2] Y. Chi, H. Wang, P.S. Yu, and R.R. Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 04)*, pages 59–66, Brighton, UK, 2004.

[3] G. Giannella, J. Han, J. Pei, X. Yan, and P. Yu. Mining frequent patterns in data streams at multiple time granularities. In *In H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha (Eds.), Next Generation Data Mining, MIT Press*, 2003.

[4] H.-F. Li, S.Y. Lee, and M.-K. Shan. An efficient algorithm for mining frequent itemsets over the entire history of data streams. In *Proceedings of the 1st International Workshop on Knowledge Discovery in Data Streams*, Pisa, Italy, 2004.

[5] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 02)*, pages 346–357, Hong Kong, China, 2002.

[6] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT 96)*, pages 3–17, Avignon, France, 1996.

[7] R. Agrawal R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE 95)*, pages 3–14, Tapei, Taiwan, 1995.

[8] W.-G. Teng, M.-S. Chen, and P.S. Yu. A regression-based temporal patterns mining schema for data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 03)*, pages 93–104, Berlin, Germany, 2003.

[9] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.