DATAMINING IN DATA STREAMS

Jakub Malý

Doctoral Degree Programme (1), FIT BUT

E-mail: xmalyj07@stud.fit.vutbr.cz

Supervised by: Jitka Kreslíková

E-mail: kreslika@fit.vutbr.cz

Abstract: Process of data mining consists from many steps. Successful and fast data mining challenges us with techniques for data reduction without substantial loss of information, structuring data by indexing methods for faster retrieving and final data mining. In this paper several final data mining algorithms are overviewed.

Keywords: Data mining, Lossy counting algorithm, DSM-FI, FTP-DS, estDec

1. INTRODUCTION

Data mining is process of finding relevant information in data. This relevant information can be frequent patterns. The pattern can be a set of items, subsequence, substructure. To consider pattern as frequent it must comply with condition of minimum support. There are methods for data mining in static data sets, but e.g. financial data streams – which is never ending stream of data - different approaches. This is because we cannot scan whole history of database, so we cannot easily determine which patterns are frequent.

There are two approaches to overcome difficulty with determination of frequent and infrequent patterns – which frequency of occurrence can vary during the time:

1. It is possible to keep track just for predefined, limited set of items.
2. Or it is possible to derive approximate set of answers, which approximates the frequency of items within user defined error bound ε.

The first approach has very limited usage because it requires the system to confine the scope of predefined patterns beforehand. The second approach is more common in data mining; therefore this chapter will cover some interesting data mining methods from this area.

2. ALGHORITHMS

2.1. LOSSY COUNTING ALGORITHM

Simple but very efficient algorithm. User defines two input parameters – minimal support threshold σ, error ε bound. Incoming data stream is divided into groups of same width $w = \frac{1}{\varepsilon}$. Let N be the count of items seen so far. Frequency list data structure is used by this algorithm to store all items with frequency >0. For each of stored items the structure contains information about approximate frequency count f and Δ for maximum possible error of f. The algorithm is processing groups of same length and frequency list is continuously updated. If a given item already exists in the list, its frequency count is increased. If item does not exist in list it is added into and its f value is set to default value of 1. If the new item is from bth group, we set Δ, the maximum possible error on the frequency count of the item to be $b - 1$. Whenever a last item in group is processed the frequency structure list is revaluated. Let b be current group number. An item from frequency list structure is removed when condition $f + \Delta \leq b$ is satisfied. Thanks to this revaluation frequency list remains small in size and can be stored in limited computer memory. The frequency count stored for each item is true frequency of an item or an underestimate of it.

Approximation ratio (or error bound) is one of the most important factors. Item from frequency structure list is deleted in case $f + \Delta \leq b$ condition is satisfied for an item. $f + \Delta$ stands for actual maximal frequency. Following conditions are valid $b \leq \frac{N}{w}$, $b \leq \varepsilon N$. From previous equation results that maximal value of b for item to be underestimated is equal to εN.

Features of Lossy counting algorithm

- No false negatives
- Result can contains false positives – but their minimal frequency still satisfies condition $f \geq \sigma N - \varepsilon N$
- Frequency of a frequent item can be underestimated by at most εN
- Needed space for items in computer memory to handle Lossy counting algorithm should not exceed $\frac{7}{\varepsilon}$.

Looking for frequent itemsets is more difficult that looking for frequent items. Number of possibilities grows exponentially with count of items in set. Processing group after group can lead to out of memory problems very soon. It is possible to process as many groups as possible (related to memory issues) to partially avoid this problem. With consideration to loading as many groups as possible to memory frequency list structure oriented on frequent itemsets can be created. In case β groups can be read into computer memory following operation can be done. We update *f* by counting the occurrences of an itemset among the current batch of β groups if this itemset already exists in frequency list structure. In case updated itemset satisfies $f + \Delta \leq b$ where b stands for current group number, itemset is removed from frequency list structure. In case when itemset has frequency $f \geq \beta$, and it is not currently presented in frequency list, it is inserted with $\Delta = b - \beta$ where Δ represents maximum error of *f*.

It is very common in real applications to set β to larger numbers – greater than 30. This setting has significant influence to memory consumption. The bigger the β the lesser the count of stored item sets. As the count of stored itemsets rise the algorithm is slower and much greedier for free memory. Lossy counting algorithm has following limitations:

- Space bound is insufficient due to frequency list structure, which can grow as far as data stream continue
- Content of frequency list structure is based on all previews processed data, although user preferences can be different – he can be more interested in recent data than historical ones

2.2. DSM-FI

Single-pass algorithm for mining of all frequent itemsets in data streams based on a landmark windows model with defined minimum support threshold $ms \in (0, 1)$ and user-defined maximal estimated support error threshold $\varepsilon \in (0, ms)$ with a goal to minimize memory usage as possible (Hua-Fu, Suh-Yin, & Man-Kwan, An Efficient Algorithm for Mining Frequent Itemests over the Entire History of Data Streams).

DSM-FI algorithm reads a basic window of transactions from the buffer in main memory and sorts them in lexicographical order. It constructs and maintains an in-memory prefix-tree based summary data structure called SFI-Forest (Summary Frequent Itemset Forest). It prunes the infrequent information from the current SFI-Forest. It finds the frequent itemsets from the SFI-Forest when it is needed.

Algorithm to search frequent (Top Down Frequent Itemset selection) for identifying frequent items from SFI-Forest can be found as part of (Hua-Fu, Man-Kwan, & Suh-Yin, DSM-FI: An efficient algorithm fo mining frequent itemsets in data streams, 2007) paper.

Comparision to Lossy counting alghoritm (LCA)

In practice, execution of DSM-FI is steadier and shorter than LCA. Memory usage of DSM-FI is more stable. This is mainly because DSM-FI does not need to enumerate all subsets of each incoming transaction. The amount of all subsets is an enormous exponential number for long transaction. Hence, it shows that DSM-FI is more suitable for mining long frequent itemsets in data streams (Hua-Fu, Suh-Yin, & Man-Kwan, An Efficient Algorithm for Mining Frequent Itemests over the Entire History of Data Streams).

2.3. FTP-DS

FTP-DS (Frequent Temporal Patterns of Data Streams) has two major features. The first one is one scan for online statistics collection and regression-based compact pattern representation. This information is used for better addressing time and the space constraints in a data stream environment. To attain the feature of one data scan, the occurrence frequency of a temporal pattern is first defined in accordance with the time constraint of sliding windows. For frequency counting purpose the data segmentation and pattern growth scenarios are explored. In next step FTP-DS scans online transaction flows and generates candidate frequent patterns as data arrives. Thanks to downward closure property, longer patterns are gradually formed from their subsets as time advances. The result of this is that frequent patterns are incrementally discovered and recorded by only one database scan. The second major feature of FTP-DS is based on the regression-based compact pattern representation which is designed to address the space constraint of a data stream environment. Data stream of incoming data is segmented. Regression process is used for identifying each segment of a time series. Regression analysis is used to capture the trends of frequent patterns. Algorithm works with implements segmentation tuning which aims to reduce the error of constructing fit lines. Segmentation tuning can result in more precise segments, which leads to more accurate mining results. The second improving technique is the segment relaxation which refers to the adjustment of segment periods along the time dimension in accordance with their corresponding importance to save the storage required. By experimental tests it was proved that algorithm FTP-DS can meet both time and space limitations in a data stream environment (Wei-Guang, Ming-Syan, & Philip S., 2003).

Algorithm FTP-DS: Frequent Temporal Patterns of Data Streams can be found as part of (Wei-Guang, Ming-Syan, & Philip S., 2003) paper.

2.4. ESTDEC

Algorithm estDec is designed to work with most recent data, which has the biggest impact on itemset frequency. Algorithm examines each transaction in a data stream one by one without any candidate generation and keeps track of the occurrence count of an itemset in the transactions generated so far by a prefix tree structure. estDec maintains a lattice for recording the potentially frequent itemsets and their counts, and updates the lattice for each new transaction correspondingly. By recording additional information for each itemset *p*, the time-point of the most recent transaction that contains *p*, the algorithm only needs to update the counts for the itemsets which are the subsets of newly arrived transactions. It reduces their counts using the constant factor *d* and then increases them by one (Mala & Dhanaseelan, 2011).

Algorithm estDec can be found as part of (Joong Hyuk & Won Suk, 2003) paper.

3. FUTURE RESEARCH

In future I will try to learn more about other algorithms for more accurate selection of set of algorithms for my next research work related to price stream indication system.

As a part of my future research, I would like to test the efficiency of alternative algorithms, namely IDSM-MFI [6]. Live data streams data mining is generally posing more challenges for the data processing subsystems than mining static database data set collections because of the continuous nature of the streaming data. IDSM-MFI algorithm implements a synopsis data structure concept which stores the transactions progressively found in the data streams that are being mined. IDSM-MFI algorithm uses a top-bottom and bottom-top models to construct the set of all maximum frequent itemsets in landmark windows that are contained in the examined data stream. The data mining results can be displayed in real time based on the operator's preferences and expressed thresholds. The creators of IDSM-MFI algorithm claim that their theoretical analysis and experimental results prove the efficiency of this algorithm, as well as its robustness and scalability for mining the data set for all maximum frequent itemsets in the history of the examined data stream.

I would also like to analyse the Probabilistic Lossy Counting (PLC) algorithm, introduced by the authors Dimitropoulos, Hurley and Kind from IBM Zurich Research Lab. PLC algorithm focuses on processing real time data traffic flows in network management applications in order to pin point the largest flows. The central issue related to finding these resource-consuming network flows is known as the “heavy-hitter problem”. Numerous attempts at solving the heavy-hitter problem have been made in the past years. The suggested Probabilistic lossy counting algorithm enhances the above discussed Lossy counting algorithm and is geared towards the applications in the domain of network traffic to uncover the heavy-hitters. The accuracy of PLC derives more from probabilistic rather than deterministic guarantees. The main advantage of PLC over LC is seen in lower memory consumption. Moreover, PLC generates lower rate of false positives compared to LC as well as low estimation error, even though the estimation error is slightly higher than the same measure for LC.

4. CONCLUSION

Enormous effort is invested into development of new data mining algorithms, which are faster, less greedy for computer resources and can successfully filter insignificant information. Data streams are a computational challenge to data mining problems because of the additional algorithmic constraints created by the large volume of data.

ACKNOWLEDGEMENT

This work was partially supported by the BUT FIT grant FIT-S-11-1 and the research plan MSM0021630528.

REFERENCES

[1] Hua-Fu, Li, Suh-Yin, Lee a Man-Kwan, Shan: *An Efficient Algorithm for Mining Frequent Itemests over the Entire History of Data Streams*, National Chiao-Tung University, Taiwan, [online, cit. 2012-03-05], WWW: <http://selab.iecs.fcu.edu.tw/wiki/images/4/46/An_efficient_algorithm_for_mining_frequent_itemsets_over_the_entire_history_of_data_streams.pdf>

[2] Hua-Fu, Li, Man-Kwan, Shan a Suh-Yin, Lee: *DSM-FI: An efficient algorithm for mining frequent itemsets in data streams*, London: Springer-Verlag, 2007, [online, cit. 2012-03-05], WWW: <http://nccur.lib.nccu.edu.tw/bitstream/140.119/29576/1/12.pdf>

[3] Wei-Guang, Teng, Ming-Syan, Chen a Philip S., Yu: *A Regression-Based Temporal Pattern Mining Scheme for Data Streams*, National Taiwan University, Taiwan, 2003, [online, cit. 2012-03-05], WWW: <http://www.vldb.org/conf/2003/papers/S04P03.pdf>

[4] Mala, A. a Dhanaseelan, F.Ramesh: *Data streamming algorithms - a review of issues and existing approaches*, KNSK College of Engineering, Nagercoil, India, 2011,

[online, cit. 2012-03-05],
WWW: <http://www.enggjournals.com/ijcse/doc/IJCSE11-03-07-064.pdf>

[5] Joong Hyuk, Chang a Won Suk, Lee: *Finding Recent Frequent Itemsets Adaptively over Online Data Streams*, Yonsei University, Seoul, Korea, 2003,
[online, cit. 2012-03-05],
WWW: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.1264>

[6] Yinmin Mao, Hong Li, Lumin Yang, Zhigang Chen, Lixin Liu, "A Mining Maximal Frequent Itemsets over the Entire History of Data Streams," dbta, pp.413-417, 2009 First International Workshop on Database Technology and Applications, 2009