

ASSOCIATION RULE MINING ON STREAMS

Philip S. Yu

IBM T. J. Watson Research Center
<http://www.research.ibm.com/people/p/psyu/>

Yun Chi

NEC Laboratories America
<http://www.nec-labs.com/~ychi/>

DEFINITION

Let $I = \{i_1, \dots, i_m\}$ be a set of items. Let S be a stream of transactions in a sequential order, where each transaction is a subset of I . For an *itemset* X , which is a subset of I , a transaction T in S is said to *contain* the itemset X if $X \subseteq T$. The *support* of X is defined as the fraction of transactions in S that contain X . For a given support threshold s , X is *frequent* if the support of X is greater than or equal to $s\%$, i.e., if at least $s\%$ transactions in S contain X . For a given *confidence* threshold c , an *association rule* $X \Rightarrow Y$ holds if $X \cup Y$ is frequent and at least $c\%$ of transactions in S that contain X also contain Y . The problem of association rule mining on streams is to discover all association rules that hold in a stream of transactions.

HISTORICAL BACKGROUND

In 1993, Rakesh Agrawal et al. [1] proposed the framework for association rule mining. Since this seminal work, a lot of researches have been done to improve the efficiency of association rule mining algorithms, to extend the definition of associations rule, and to apply association rule mining to other types of data such as time sequence data and structured data such as graphs. On the other hand, research on data streams started in around 2000 when several data stream management systems were originated (e.g., the Brandeis AURORA project, the Cornell COUGAR project, and the

Stanford STREAM project) to solve new challenges in applications such as network traffic monitoring, transaction data management, Web click streams monitoring, sensor networks, etc. Because association rule mining plays an important role in these data stream applications (e.g., the motivation in the first association rule paper [1] is to mine patterns from retail store transaction data), along with the development of data stream management systems, developing association rule mining algorithms for data streams has become an important research topic.

SCIENTIFIC FUNDAMENTALS

Two Sub-problems

Algorithms for association rule mining usually consist of two steps. The first step is to discover frequent itemsets. In this step, all frequent itemsets that meet the support threshold are discovered. The second step is to derive association rules. In this step, based on the frequent itemsets discovered in the first step, the association rules that meet the confidence criterion are derived. Because the second step, deriving association rules, can be solved efficiently in a straightforward manner, most of researches focus mainly on the first step, i.e., how to efficiently discover all frequent itemsets in data streams. Therefore, in the rest of this article, the focus will be on frequent itemset mining in data streams.

Key Challenges

Frequent itemset mining in general is already a challenging problem. For example, due to combinatorial explosion, there may be huge number of frequent itemsets, and a main challenge is how to efficiently enumerate, discover, and store frequent itemsets. Data streams, because of their unique features, have further posed many new challenges to frequent itemset mining. Some of these new challenges are described as the following.

Single access of data In data streams, data are arriving continuously with high speed and in large volume. As a consequence, in many cases it is impractical to store all data in persistent media and in other cases, it is too expensive to (randomly) access data multiple times. The challenge is to discover frequent itemsets while the data can only be assessed once.

Unbounded data Another feature of data streams is that data are unbounded. In comparison, storage that can be used to discover or maintain the frequent itemsets is limited. Another consequence of unbounded data is that whether an itemset is frequent depends on time. The challenge is to use limited storage to discover dynamic frequent itemsets from unbounded data.

Real-time response Because data stream applications are usually time-critical, there are requirements on response time. For some restricted scenarios, algorithms that are slower than the data arriving rate are useless. The challenge is therefore to efficiently mine frequent itemsets in real-time.

Data Models

Compared with finite data in traditional databases, data streams are unbounded and the number of transactions increase with time. Due to this characteristic, different data models have been proposed in different mining algorithms.

The first data model is an accumulative model. In this model, all data in the range from the beginning of time to the current time are considered in an equal fashion. Therefore, frequent itemsets are defined on the accumulated data where new data are appended continuously as time grows.

The second data model is based on a sliding window. That is, although the whole data stream is unbounded, the frequent itemsets are defined based on the most recent data that fall within a temporal sliding window whose ending time is the current time. One justification for such a sliding-window model is that due to concept drifts, the data distribution in streams is usually changing with time, and very often people are interested in the most recent patterns.

The third data model falls in between the first two models — while all data are considered in frequent itemset mining, they are weighted differently according to a predefined weighting function. A very commonly used weighting function is the exponentially decaying function, that is, for a transaction at time τ , its weight is $\alpha(\tau) = \exp(\tau - t)$ where t is the current time.

Algorithm Types

Based on the mining results, existing frequent itemsets mining algorithms on data streams can be roughly divided into two categories: the exact mining algorithms and the approximate mining algorithms.

Exact mining algorithms provide as results all the frequent itemsets in the data streams together with their accurate supports. Usually the focus of exact algorithms is to efficiently update frequent itemsets when new transactions arrive and (in the sliding-window data model) when old transactions expire.

Approximate mining algorithms, on the other hand, focus more on finite memory usage and single access of data, at the cost of the accuracy of the mining results. Approximate algorithms target low false positive rate, zero or low false negative rate, and tight error bounds on the estimation for the supports of the frequent itemsets.

Representative Algorithms

Cheung et al. [5, 6] proposed algorithms *FUP* and *FUP₂* for incrementally updating frequent itemsets. Thomas et al. [13] presented a similar algorithm. Both Cheung's and Thomas's algorithms assume batch updates and take advantage of the relationship between the original database (*DB*) and the incrementally changed transactions (*db*). *FUP* is similar to the well-known Apriori Algorithm [1], which is a multiple-step algorithm. The key observation of *FUP* is that by adding *db* to *DB*, some previously frequent itemsets will remain frequent and some previously infrequent itemsets will become frequent (these itemsets are called *winners*); at the same time, some previously frequent itemsets will become infrequent (these itemsets are called *losers*). The key technique of *FUP* is to use information in *db* to filter out some winners and losers, and therefore reduce the size of candidate set in the Apriori algorithm. Because the performance of the Apriori algorithm relies heavily on the size of candidate set, *FUP* improves the performance of Apriori greatly. *FUP₂* extended *FUP* by allowing deleting old transactions from a database as well. Therefore *FUP* is restricted to the accumulative data model while *FUP₂* can be used on the sliding-window data model as well. The algorithm proposed by Thomas et al. is similar to *FUP₂* except that in addition to frequent itemsets, a negative border is maintained. In the algorithm, the frequent itemsets in *db* are mined first. At the same time, the counts of frequent itemsets (and itemsets on the negative border) in *DB* are updated. Then based on the change of the frequent itemsets in *DB*, the negative border in *DB*, and the frequent itemsets in *db*, the frequent itemsets in the updated database are computed with a possible scan of the updated database. Because the updated database is scanned at most once, Thomas's algorithm has very good performance. Thomas's algorithm can be used for both the accumulative and the sliding-window data models. In

addition, *FUP*, *FUP*₂, and Thomas’s algorithm all fall into the category of exact mining algorithms.

Veloso et al. [14] proposed an algorithm *ZIGZAG* for mining frequent itemsets in evolving databases. Later, Otey et al. [11] extended *ZIGZAG* into parallel and distributed algorithms. *ZIGZAG* is similar to Cheung’s and Thomas’s algorithms in that it achieves its speedup by using the relationship between *DB* and *db*. However, *ZIGZAG* has many distinct features. First, *ZIGZAG* mainly used *db* to speedup the support counting of frequent itemsets in the updated database and it does not discover the frequent itemsets in *db* itself. As a result, for a given minimum support, *ZIGZAG* can handle batch update with arbitrary block size. Second, *ZIGZAG* adapts the techniques proposed in the *GENMAX* algorithm [9] and in each update only maintains *maximal* frequent itemsets. Because the information on maximal frequent itemsets and their supports is not enough to generate association rules (because the support information of some non-maximal frequent itemsets may be missing), a second step is used in *ZIGZAG* in which the updated database is scanned to discover all frequent itemsets and their supports.

Chi et al. [7] developed an algorithm, *Moment*, to mine *closed* frequent itemsets over data stream sliding windows. In this work the authors introduced a compact data structure, the *closed enumeration tree* (CET), to maintain a dynamically selected set of itemsets over a sliding window. The selected itemsets contain a boundary between closed frequent itemsets and the rest of the itemsets. Concept drifts in a data stream are reflected by boundary movements in the CET, and can be efficiently captured. Both *ZIGZAG* and *Moment* are exact mining algorithms that use the sliding-window data model.

Charikar et al. [3] presented a 1-pass algorithm, *Count Sketch*, that returns most frequent *items* whose frequencies satisfy a threshold with high probabilities. Manku et al. [10] developed a randomized algorithm, the *Sticky Sampling* Algorithm, and a deterministic algorithm, the *Lossy Counting* Algorithm, for maintaining frequent *items* over a data stream where for a given time *t*, the frequent items are defined over the *entire* data stream up to *t*. The algorithms guarantee no false negative and a bound on the error of estimated frequency (the guarantees are in a probabilistic sense for the randomized algorithm). The *Lossy Counting* Algorithm is extended to handle frequent *itemsets*, where a trie is used to maintain all frequent itemsets and the trie is updated by batches of transactions in the data stream. The algorithms of Manku et al. strive for a tunable compromise between memory usage and error bounds. *Count Sketch*, *Sticky Sampling*, and *Lossy Counting* are all approximate mining algorithms that use the accumulative

Table 1: The categorization of the representative algorithms according to their data models and algorithm types

	Accumulative Data Model	Sliding-window Data Model	Weighted Data Model
Exact Mining Algorithm	<i>FUP</i> [5]	<i>FUP</i> ₂ [6] Thomas’s [13] <i>ZIGZAG</i> [14, 11] <i>Moment</i> [7]	
Approximate Mining Algorithm	<i>Count Sketch</i> [3] <i>Sticky Sampling</i> [10] <i>Lossy Counting</i> [10]	<i>FTP-DS</i> [12]	<i>estDec</i> [2] Giannella’s [8]

data model.

Teng et al. [12] presented an algorithm, *FTP-DS*, that mines frequent temporal patterns from data streams of itemsets. *FTP-DS* is an approximate mining algorithm that uses the sliding-window data model. Chang et al. [2] presented an algorithm, *estDec*, that mines recent frequent itemsets where the frequency is defined by an aging function. Giannella et al. [8] proposed an approximate algorithm for mining frequent itemsets in data streams during arbitrary time intervals. An in-memory data structure, *FP-stream*, is used to store and update historic information about frequent itemsets and their frequency over time and an aging function is used to update the entries so that more recent entries are weighted more. Both *estDec* and Giannella’s algorithm are approximate mining algorithms on weighted transactions. However, Giannella’s algorithm is a little different in that it can provide different error levels for data at multiple time granularities.

The above representative algorithms are summarized in Table 1. For a more detailed survey on algorithms for frequent itemset mining over data streams, the readers are referred to a recent survey by Cheng et al. [4].

KEY APPLICATIONS

For most data stream applications, there are needs for mining frequent patterns and association rules from data streams. Some key applications in various areas are listed in the following.

Performance monitoring Monitor network traffic and performance, detect abnormality and intrusion

Transaction monitoring Monitor transactions in retail stores, ATM machines, and financial markets

Log record mining Mine patterns from telecommunication calling records, Web server log, etc.

Sensor network mining Mine patterns in streams coming from sensor networks or surveillance cameras

EXPERIMENTAL RESULTS

In general, for each of the presented algorithms, there are supporting experimental studies in the corresponding references. The commonly compared performance metrics include memory usage, speed of the mining process, and (for the approximate algorithms) errors such as false positive rate, false negative rate, and support errors for the frequent itemsets.

DATA SETS

A Linux version of the synthetic data generator originally developed by Agrawal et al. [1] is available at

<http://miles.cnuce.cnr.it/~palmeri/datam/DCI/datasets.php>

Some other synthetic and real-life data sets are available from the Frequent Itemset Mining Dataset Repository at <http://fimi.cs.helsinki.fi/data/>

Furthermore, pointers to some related data sets are available at the KD-nuggets Web site <http://www.kdnuggets.com/datasets/>

Cross References

Some related entries in this encyclopedia are: Approximation of frequent itemsets, Association rule mining, Change detection on streams, Closed itemset mining and non-redundant association rule mining, Continuous queries in sensor networks, Data aggregation in sensor networks, Data estimation in sensor networks, Data mining, Data sketch/synopsis, Data streams, Frequent items on streams, Frequent itemsets and association rules, Incremental computation of queries, Internet and Web transactions, One-pass algorithm, Pattern-growth methods, Randomization methods, Real-time transactions, Sensor network, Stream mining, Stream models, Streaming applications, Tries, Web services.

RECOMMENDED READING

References

- [1] Agrawal R., Imielinski T., Swami A. (1993): *Mining Association Rules between Sets of Items in Large Databases*. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., USA, May 26–28, 1993.
- [2] Chang J. H., Lee W. S. (2003): *Finding Recent Frequent Itemsets Adaptively over Online Data Streams*. Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24–27, 2003.
- [3] Charikar M., Chen K., Farach-Colton M. (2002): *Finding Frequent Items in Data Streams*. Proceedings of the 2002 International Colloquium on Automata, Languages and Programming, Malaga, Spain, July 8–13, 2002.
- [4] Cheng J., Ke Y., Ng W. (2008): *A Survey on Algorithms for Mining Frequent Itemsets over Data Streams*. Knowledge and Information Systems, Volume 16, Number 1, July, 2008.
- [5] Cheung D. W., Han J., Ng V., Wong C. Y. (1996): *Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique*. Proceedings of the Twelfth International Conference on Data Engineering, New Orleans, Louisiana, USA, February 26 – March 1, 1996.
- [6] Cheung D. W., Lee S. D., Kao B. (1997): *A General Incremental Technique for Maintaining Discovered Association Rules*. Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA), Melbourne, Australia, April 1–4, 1997.
- [7] Chi Y., Wang H., Yu P. S., Muntz R. R. (2006): *Catch the Moment: Maintaining Closed Frequent Itemsets in a Data Stream Sliding Window*. Knowledge and Information Systems, 10(3): 265-294.
- [8] Giannella C., Han J., Pei J., Yan X., Yu P. S. (2004): *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. Data Mining: Next Generation Challenges and Future Directions, edited by H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, AAAI press, 2004.

- [9] Gouda K., Zaki M. J. (2001): *Efficiently Mining Maximal Frequent Itemsets*. Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, California, USA, November 29 – December 2, 2001.
- [10] Manku G., Motwani R. (2002): *Approximate Frequency Counts over Data Streams*. Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China, August 20–23, 2002.
- [11] Otey M. E., Parthasarathy S., Wang C., Veloso A., Meira Jr. W. (2004): *Parallel and Distributed Methods for Incremental Frequent Itemset Mining*. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 34(6):2439–2450.
- [12] Teng W-G, Chen M-S, Yu P. S. (2003): *A Regression-based Temporal Pattern Mining Scheme for Data Streams*. Proceedings of 29th International Conference on Very Large Data Bases, Berlin, Germany, September 9–12, 2003.
- [13] Thomas S., Bodagala S., Alsabti K., Ranka S. (1997): *An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases*. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14–17, 1997.
- [14] Veloso A., Meira Jr. W., de Carvalho M., Pôssas B., Parthasarathy S., Zaki M. J. (2002): *Mining Frequent Itemsets in Evolving Databases*. Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11–13, 2002.