A New Approach for Mining Incrementally Closed Itemsets over Data Streams

Thanh-Trung Nguyen[1], Phong Le[2]
Dept. Information Technology
Hong Bang International University
Ho Chi Minh City, Vietnam
[1]trungnt@hiu.vn, [2]phonglh@hiu.vn

Sptisyn Vladimir Grigorievich[3]
Dept.Division for Information Technology
School of Computer Science & Robotics
National Research Tomsk Polytechnic University, Russian Redereation
[3]spvg@tpu.ru

Phan Ngoc Hoang[4]
Dept. School of Information Technology
Electrical & Electronic Engineering
Ba Ria-Vung Tau University, Vietnam
[4]hoangpn@bvu.edu.vn

***Abstract*—Incremental mining always requires an intermediate structure to store the results of the previous steps and update the results of the current step based on this structure. In particular, over data streams, the intermediate structure needs to be particularly effective because of the following characteristics of data streams: the size of input data is not limited; the use of main memory is limited; input data can only be processed once; the appearing speed of new data is fast; system can not control the appearing order of incoming data; analytical results generated by algorithms must be available immediately upon user request; errors of analysis results must be bounded in a range acceptable to users. In the previous study, the author proposed an intermediate structure called constructive set. In this paper, the author proposes applying the constructive set and two incremental algorithms to the problem of mining closed sets over data streams.**

Keywords—closed itemsets; constructive set; data mining; data streams; incremental mining

I. Introduction

In recent years, advances in hardware technology have facilitated the ability of continuous data collection. Popular everyday transactions such as using credit cards, making phone calls or browsing web have created the need for automated data storage. Likewise, advances in information technology have led to large amounts of data being transmitted across the Internet. The need to mine the information and knowledge latent from such data volumes is huge for many applications. However, when the volume of data is too large, there are some challenges in the mining process:

With the increasing volume of data, it is not possible to process data efficiently when browsing multiple times. More precisely: it can only process one data item once, at the very most. This leads to constraints in the execution of algorithms. Therefore, algorithms for mining data streams should generally be designed to only scan data once.

In most cases, time is an inherent component associated with the process of mining data streams, because data can develop over time. Consequently, algorithms for mining data streams need to be carefully designed with a clear goal focused on the development of data. This means that there is a need for incremental mining.

Another important feature of data streams is that they are often mined in a distributed environment.

Frequent itemsets mining is a core operation of data mining. Therefore, frequent itemsets mining over data streams has attracted a lot of research interest. Compared to other operations over data streams, frequent itemsets mining poses major challenges due to the computational cost and the large memory need, as well as the requirement for accuracy of mining results. The problem of frequent itemsets mining was first introduced in [1], and has widely analyzed for the usual cases of disk resident data sets. In the case of data streams, one might want to find frequent itemsets on sliding windows or entire data streams [4][6].

A data stream *D* is defined as a sequence of transactions, *D* = (t_1, t_2, ..., t_i, ...) where t_i is the transaction occurs at the ith point of time. To handle and mine data streams, there are three commonly used window models. A window is a sequence of transactions occurring from the ith to the jth, denoted W [i, j] = (t_i, t_{i+1}, ..., t_j).

Landmark window: In this model, frequent itemsets are found from a starting point of time i until the present time t. In other words, frequent itemsets are found over the window W [i, t]. A special case of the Landmark window is i = 1. In this case, frequent itemsets are mined over the entire data stream. One note in this model is that each time after the start of time is equally important. However, in many cases, recent times are of great interest. The next two models focus on this case.

Sliding window: Given the length of the sliding window is w and the current time is t. The frequent itemsets are mined in the window W [t - w + 1, t]. When the time changes, this window will remain the same size and move along with the current time. This model does not care about data that appears before t - w + 1.

Damped window: This model assigns a large weight to transactions that occur near the current time. To do this, the decay rate is defined and used to update (by multiplication) transactions that appear before a new transaction occurs. Correspondingly, the frequency of an itemset is also determined based on the weight of each transaction.

With the sliding window model, user can specify the window length not too large so that the number of transactions in the sliding window can be stored in the main memory. And obviously, with the goal of updating the results of the frequent itemsets in the window when new transactions occur (it means removing the old transactions), the sliding window model needs to be applied the technique of incremental mining in case of adding and removing transactions.

Closed itemsets mining is a general case of frequent itemsets mining. So, with the requirement of mining closed itemsets in the Sliding window model, the constructive set along with the algorithms introduced at the end of the next section can be applied.

II. Overview

Up to now, intermediate structures used for incremental mining over data streams are mostly a tree, as listed below:

Li et al. proposed prefix tree-based single-scan algorithms, called DSM-FI [8] and DSM-MFI [9], for mining the set of all frequent itemsets and maximal frequent itemsets over the entire history of offline data streams.

[4] proposed a FP-tree [5] based algorithm, called FP-stream, to mine frequent itemsets at multiple time granularities by a titled-time windows technique. FP-stream focuses on offline data streams.

[2] proposed a transaction-sensitive sliding window based algorithm, called Moment, which might be the first to find frequent closed itemsets from online data streams with a transaction-sensitive sliding window. A summary data structure, called CET (closed enumeration tree), is used in the Moment algorithm to maintain a dynamically selected set of itemsets over a transaction-sensitive sliding window.

[10] presented an algorithm, called FP-CDS that can capture all frequent closed itemsets and a storage structure, called FP-CDS tree that can be dynamically adjusted to reflect the evolution of frequencies of itemsets over time.

[3] proposed an incremental mining algorithm, called DSM-CITI (Data Stream Mining for Closed Inter-Transaction Itemsets), for discovering the set of all frequent inter-transaction itemsets from data streams. In the framework of DSM-CITI, an in-memory summary data structure, ITP-tree, is developed to maintain frequent inter-transaction itemsets.

Other studies have developed extensively based on the tree structures presented above:

[9] proposed a single-pass algorithm, called DSM-RMFI, based on DSM-MFI to find maximal frequent itemsets over offline data streams with a time-sensitive sliding window.

[7] Li et al. (2009) proposed an algorithm, called NewMoment to improve the efficiency of the algorithm Moment [2].

The purpose of [11] is mining closed frequent itemsets from transactional data streams using a sliding window model. An algorithm, called IMCFI is proposed for Incremental Mining of Closed Frequent Itemsets from a transactional data stream. The proposed algorithm IMCFI uses a data structure called INdexed Tree (INT) similar to NewCET used in NewMoment [7].

The author proposed an intermediate structure called constructive set to produce closed sets along with their occurrence frequencies [12]. The constructive set is constructed from a set of group patterns – an extended form of bit chains. The author also proposed algorithms based on the constructive set for mining incrementally closed sets when adding and removing transactions. Background knowledge is briefly outlined in the next section.

III. Background

Transaction database is $\mathcal{T}= (\mathcal{O}, \mathcal{I}, \mathcal{R})$ – a trio, with a set $\mathcal{O} \neq \varnothing$ consisting of transaction objects $o_1, o_2,.., o_n$, $|\mathcal{O}| = n$; a set $\mathcal{I} \neq \varnothing$ of transaction items $i_1, i_2,.., i_m$, $|\mathcal{I}| = m$ and $\mathcal{R}$ is a binary relation on $\mathcal{O} \times \mathcal{I}$.

The transaction set of $\mathcal{T}$ has a representation of n×m bit matrix $R = (\rho_{pq})$, $\rho_{pq} = \mathcal{R}(o_p,i_q) \in \{0,1\}$, $o_p \in \mathcal{O}$, $i_q \in \mathcal{I}$, $p = 1,2,\dots,n$, $q = 1,2,\dots,m$, and $\rho_{pq} = 1$ if o_p deals with iq, $\rho_{pq} = 0$ otherwise.

For a transaction set $\mathcal{T}= (\mathcal{O}, \mathcal{I}, \mathcal{R})$, each row of transaction matrix R is described by a m bit-chain, called *bit pattern*, namely bit pattern with size m or m-bit pattern: $b = b_1b_2b_3\dots b_{m-1}b_m$, $b_k \in \{0,1\}$, $k = 1,2,..,m$.

Given two m-bit patterns $a = a_1a_2a_3\dots a_{m-1}a_m$ and $b = b_1b_2b_3\dots b_{m-1}b_m$, then: $a = b \Leftrightarrow a_k = b_k, \forall k \in \{1,..,m\}$.

Composition pattern of a, b is established by the & (AND) operation on bits of a, b: $a\&b = c = c_1c_2c_3\dots c_{m-1}c_m \Leftrightarrow c_k = a_k \times b_k$, $\forall k \in \{1,..,m\}$.

When a & b = b, pattern a has more bits 1 than that of pattern b, in other words b *covers* a or a *is covered by* b, denoted $a \propto b$, thus: $a \propto b \Leftrightarrow a \,\&\, b = b$. The negation operator is $a\, !\!\propto b$.

The number of appearances of a bit pattern a in $\mathcal{T}$ is the *frequency* of a, denoted f_a. To describe a bit pattern with its frequency, we may use a dot as the delimitation.

If there are some bits whose values are not specified, the character * is used to indicate the ‘aggregation’ of these possible values. Since then, the *group patterns* should be identified. A bit pattern is a specific case of a group pattern when its all bits have a definite value of 0 or 1.

If u, v are group patterns with size m, the composition of u and v, also denoted u & v, is a pattern: $w = w_1w_2w_3\dots w_{m-1}w_m = u \,\&\, v$ with $w_k = 1$ if $u_k \times v_k = 1$ and $w_k = *$ otherwise, for $k = 1,..,m$.

If u & v = v, the group pattern u is called ‘*is covered by* v’, also denoted $u \propto v$.

The number of appearances of a group pattern u in the transaction set $\mathcal{O}$ is *frequency* of u corresponding with $\mathcal{O}$, also denoted f_u.

Composition group pattern of group patterns $u.f_u$ and $v.f_v$ is a group pattern $w.f_w$, denoted $w.f_w = u.f_u \,\&\, v.f_v$ with: $w = u \,\&\, v$ and $f_w = f_u + f_v$.

A group pattern $u.f_u$ is called *private* group pattern of a group pattern $v.f_v$, denoted $u.f_u \,\underline{\ll}\, v.f_v$, if it happens $u = v$ and $f_u \leq f_v$.

On the other hand, with two group patterns u, v, if $u \neq v$, $u \propto v$ and $f_u \leq f_v$ then v is *wide* group pattern of u, denoted $u.f_u \ll v.f_v$. The relation $\underline{\ll}$ is considered to be a specific case of $\ll$.

Let $\mathcal{T}= (\mathcal{O}, \mathcal{I}, \mathcal{R})$, $O \subseteq \mathcal{O}$, $I \subseteq \mathcal{I}$, *rectangle* (rct) $R = \langle O,I \rangle$ in $\mathcal{R}$ is the set of elements of $O \times I \subseteq \mathcal{R}$. In order, $|O|$, $|I|$ are the vertical dimension, the horizontal dimension of R, the size of R is $|O| \times |I|$. With rct $R = \langle O,I \rangle$, the projections R on the set of objects, the set of items are defined by $Pr_o(R) = O$, $Pr_i(R) = I$.

Rct $\langle O',I' \rangle$ is called *be contained* by rct $\langle O,I \rangle$, denoted $\langle O',I' \rangle \sqsubseteq \langle O,I \rangle$, if $O' \subseteq O$, $I' \subseteq I$. When O′, I′ are strictly contained by O, I, it is denoted $\langle O',I' \rangle \sqsubset \langle O,I \rangle$.

A rct is *maximal* if it is not contained by any other rct of $\mathcal{R}$.

Let the transaction database $\mathcal{T}= (\mathcal{O}, \mathcal{I}, \mathcal{R})$, the set of maximal group patterns is defined as the *constructive set* P of

$\mathcal{T}$, each maximal group pattern is called a *constructive pattern*. So: $\forall p.f_p \in P, \nexists u.f_u \in P, u \neq p : u \propto p$ and $f_u \geq f_p$

```
ALGORITHM IncPatSet(o,P,nP)
// Function for updating constructive pattern set when increasing a new object
// in: new object o, current constructive set P
// out: P new constructive set
 1. if all bits in o equal * then return P; // o∉P
 2. if all bits in o equal 1 then {
 3.    for p∈P do fp := fp+1;          // statements in A-block
 4.    if o∉P then append o to P; return P; }
 5. Q := {o};                          // statements in B-block
 6. for p∈P do {
 7.   q := o&p;
 8.   if all bits in q equal * then continue // q∉P
 9.   else Q:=Q∪{q}; }
10. S := Q; R := ∅;  // statements in C-block: Filtered-PatSet(Q)
11. for q∈Q do {
12.   s := q;
13.   for r∈S do if q«r then s := r;
14.   R := R∪{s}; nP:=nP+1; }  // set of new creative patterns
15. Q := ∅;                    // statements in D-block: Filtered-PatSet(P)
16. for p∈P do {
17.   q := p;
18.   for r∈R do if p«r then q := r;
19.   Q := Q ∪ {q}; nP:=nP+1;}  //updated old creative patterns
20. Return P := Q ∪ R;  // creative pattern set with incremental object
```

```
ALGORITHM DesPatSet(o,P)
//Updating a Constructive Pattern Set P when descending an object
//in: Descended object o, Current set P
//out: Updated constructive set P
1. Q := ∅;
2. for p∈P do
3.   if o ∝ p then {
4.     fp := fp-1; Q := Q∪{p}; }
5. for q∈Q do {
6.   for p∈P do {
7.     if p«q then {
8.     P := P\{q}; break; } } }
9. Return P.
```

IV. Example

The following example illustrates in detail the process of applying two algorithms to mining incrementally closed itemsets in the sliding window model over data streams.

Assume that there are 12 transactions as in Table I appearing in data stream model with sliding window having the length of 4.

TABLE I. The transaction set $\mathcal{T}_1$

	a	b	c	d
1	1	1	1	0
2	0	1	1	1
3	1	1	1	0
4	0	1	1	0
5	0	1	1	1
6	1	1	1	0
7	0	1	1	0
8	0	1	0	1
9	1	1	1	0
10	0	1	1	0
11	0	1	0	1
12	0	0	1	1

The data stream model with sliding window having the length of 4. Initially four transactions occurs (Table II). Currently, the transaction o_5 appears (Table III).

TABLE II. Four transactions appearing

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
a	1	0	1	0	0	1	0	0	1	0	0	0
b	1	1	1	1	1	1	1	1	1	1	1	0
c	1	1	1	1	1	1	1	0	1	1	0	1
d	0	1	0	0	1	0	0	1	0	0	1	1

TABLE III. The transactions o_5 appearing

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
a	1	0	1	0	0	1	0	0	1	0	0	0
b	1	1	1	1	1	1	1	1	1	1	1	0
c	1	1	1	1	1	1	1	0	1	1	0	1
d	0	1	0	0	1	0	0	1	0	0	1	1

Initially, with 4 transactions o_1, o_2, o_3, o_4, the constructive set P has the closed sets: P_{1234} = {*111.1, 111*.2, *11*.4}.

The transaction o_5 appears, it needs to remove o_1. Update the constructive set for the window by first removing o_1 = 1110, `DesPatSet(`o_1`, `P_{1234}`)`, P_{234} = {*111.1, 111*.1, *11*.3}.

Then add o_5, `IncPatSet(`o_5`, `P_{234}`,3)`. Now, the constructive set of the window has the following closed sets: P_{2345} = {111*.1, *111.2, *11*.4}.

Next, two transactions o_6 and o_7 appear.

TABLE IV. Transactions o_6 and o_7 appearing

	2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
a	0	1	0	0	1	0	0	1	0	0	0
b	1	1	1	1	1	1	1	1	1	1	0
c	1	1	1	1	1	1	0	1	1	0	1
d	1	0	0	1	0	0	1	0	0	1	1

Remove o_2 = 0111, `DesPatSet(`o_2`, `P_{2345}`)`, P_{345} = {111*.1, *111.1, *11*.3}.

Remove o_3 = 1110, `DesPatSet(`o_3`, `P_{345}`)`, P_{45} = {*111.1, *11*.2}.

Add o_6 = 1110, `IncPatSet(`o_6`, `P_{45}`,2)`, P_{456} = {*111.1, *11*.3, 111*.1}

Add o_7 = 0110, `IncPatSet(`o_7`, `P_{456}`,3)`, P_{4567} =

{*111.1, 111*.1, *11*.4}

V. Conclusion

This paper introduced an intermediate structure, called the constructive set, for incremental mining closed sets. In addition, two incremental algorithms, corresponding to two processes of adding and removing transactions, are also introduced for applying to incremental mining closed sets over data streams.

In the future, this process of applying will be realized in the environment Hadoop-Spark.

References

[1] Agrawal R., Imielinski T., and Swami A., "Mining Association Rules between Sets of items in Large Databases," in SIGMOD '93 Proceedings of the 1993 ACM SIGMOD international conference on Management of data, Pages 207-216, May 25 - 28, 1993.

[2] Chi Y., Wang H., Yu P., and Muntz R., "MOMENT: Maintaining closed frequent itemsets over a stream sliding window," in Proceedings of the 4th IEEE international conference on data mining, (pp. 59–66), 2004.

[3] Chiu S.-C., Li H.-F., Huang J.-L., and You H.-H., "Incremental mining of closed inter-transaction itemsets over data stream sliding windows," Journal of Information Science, Volume: 37 issue: 2, page(s): 208-220, April 2011.

[4] Giannella C., Han J., Pei J., Yan X., and Yu, P.S., "Mining frequent patterns in data streams at multiple time granularities," in Kargupta H., Joshi A., Sivakumar K., and Yesha Y. (Eds.), Data mining: Next generation challenges and future directions. AAAI/MIT, 2003.

[5] Han J., Pei J., and Yin Y., "Mining frequent patterns without candidate generation," in Proceedings of the 2000 international conference on management of data, (pp. 1–12), 2000.

[6] Jin R. and Agrawal G., "An algorithm for in-core frequent itemset mining on streaming data," in ICDM '05 Proceedings of the Fifth IEEE International Conference on Data Mining, Pages 210-217, November 27 - 30, 2005.

[7] Li H.-F., Ho C.-C., and Lee S.-Y., "Incremental updates of closed frequent itemsets over continuous data streams," Expert Systems with Applications 36(2):2451-2458,·March 2009.

[8] Li H.-F., Lee S.-Y., and Shan M.-K., "An efficient algorithm for mining frequent itemsets over the entire history of data streams," in Proceedings of the first international workshop on knowledge discovery in data streams, 2004.

[9] Li H.-F., Lee S.-Y., and Shan M.-K., "Online mining (recently) maximal frequent itemsets over data streams" in Proceedings of the 15th IEEE international workshop on research issues on data engineering, (pp. 11–18), 2005.

[10] Liu X., Guan J., and Hu P., "Mining frequent closed itemsets from a landmark window over online data streams," Computers & Mathematics with Applications, Volume 57, Issue 6, Pages 927-936, March 2009.

[11] Naik S.B. and Pawar J.D., "An Efficient Incremental Algorithm to Mine Closed Frequent Itemsets over Data Streams," in Proceeding of The 19th International Conference on Management of Data (COMAD), 19th - 21st Dec, 2013.

[12] Nguyen T.-T., "Mining Incrementally Closed Item Sets with Constructive Pattern Set," Expert Systems With Applications, Vol.100, page(s): 41-67, June 2018.