International Journal of Computer Science and Informatics

Volume 1 | Issue 1 Article 5

July 2011

An Efficient Algorithm for Mining Of frequent items using incremental model

Nibedita Panigrahi
Konark Institute of Science and Technology Mtech CSE, nibedita.kuni@gmail.com

P.K. Pattnaik
Department of Computer Science and Engineering Konark Institute of Science and Technology Bhubaneswar, Orissa, India, patnaikprasant@gmail.com

S.K. Padhi
Department of Computer Science and Engineering; Konark Institute of Science and Technology; Bhubaneswar, Orissa, India, Sanjaya2004@yahoo.com

An Efficient Algorithm for Mining Of frequent items using incremental model

Nibedita Panigrahi
Konark Institute of Science and Technology
Mtech CSE 4th semester
nibedita.kuni@gmail.com

Prof. Dr.Prashant Patnaik
HOD , CSE Dept.
KIST , BBSR
patnaikprasant@gmail.com

Mr. Sanjay Padhi
Asst. Prof. ,CSE Dept
KIST , BBSR
sanjaya2004@yahoo.com

***Abstract**: Data mining is a part of know ledge Discovery in database process (KDD). As technology advances, floods of data can be produced and shared in many appliances such as wireless Sensor networks or Web click streams. This calls for extracting useful information and knowledge from streams of data. In this paper, We have proposed an efficient algorithm, where, at any time the current frequencies of all frequent item sets can be immediately produced. The current frequency of an item set in a stream is defined as its maximal frequency over all possible windows in the stream from any point in the past until the current state. The experimental result shows the proposed algorithm not only maintains a small summery of information for one item set but also consumes less memory then existing algorithms for mining frequent item sets over recent data streams.*

***Key words:** Data mining, frequent items, borders, incremental updates.*

1. Introduction

Mining frequent item sets has been studied extensively in data mining, with many algorithms proposed and implemented. Most previous work on mining frequently occurring item sets over data streams focuses on (1) Landmark model[1,8,9,12 [1,4,9,12],(2) the time fading model[1,5,6,12], (3)] sliding window model.

In Landmark model, a particular time points called landmark are fixed .The analysis of the stream is performed for only the part of the stream between the landmarks and the current time instance . The landmark usually refers to the time when the system starts. Moreover, the support count of an item set in this model is the number of transactions containing it between the landmark and the current time. To find frequent items under this model, the support count of each incoming item is accumulated on a counter. Since the number of distinct items is often more than available counters, sampling techniques are employed to assign items to counters and then estimate the support counts of all the items. But here the size of the window varies; it starts with size zero and grows until the next occurrence of the landmark, at which point it is reset to zero and this model is not aware of time and therefore cannot distinguish between new data and old ones.

The time-fading model assigns different weights to transactions such that new ones have higher weights than old ones. The weights are decreasing here as time passes by. A decay factor d >1 is introduced here[10]. By using this decay factor, the effects of old transactions diminish as time goes by.FP-stream is a approach which provides a way to mine frequent item sets under the time-fading model. This model satisfies approximation and adjustability. But , In certain applications, users can only be interested in the data recently arriving within a fixed time period which this model is unable to satisfy the need.

Two types of sliding window methods i.e. transaction-sensitive and time-sensitive sliding window methods are used for mining frequent items. The basic processing unit of the transaction sensitive sliding window is an expired transaction and the basic processing unit of the time sensitive sliding window is fixed time period. Two different points of view are considered here, the count-based view, in which the number of transactions in the window is fixed despite the generating speed of the transactions and the time-based windows, in which the number of time units in the window is fixed, causing the number of transactions to be different. Moreover, all the previous works consider a fixed number of transactions as the basic unit for mining, which is not easy for people to specify. By contrast, it is natural for people to specify a time period as the basic unit. Therefore we propose the time-sensitive sliding-window model [12], which regards a fixed time period as the basic unit for mining.

Time-sensitive sliding-window model [12]: given a time point *t* and a time period *p*, the set of all the transactions arriving in [t-p+1, t] will form a basic block. A data stream is decomposed into a sequence of basic blocks, which are assigned with serial numbers starting at 1. Given a window with length $|W|$, I slide it over this sequence to see a set of overlapping sequences, where each sequence is called the time-sensitive sliding-window abbreviated as TS.

The remainder of the paper is organized as follows. The problem is defined in section-2.section-3 represents the summary. The Algorithm is given in section-4. The Experiments are discussed in section-5.I conclude this work in section-6.Finally I have given the references in section-7.

2. PROBLEM DEFINITION:

Let a Stream S is an unbounded sequence. S_t will denote the stream S up to timestamp t; that is, the part of the stream that already passed at timestamp t. In the examples, the streams will grow from left to right. This means that the older item sets are on the left of the stream, and the newest, or more recent item sets on the right hand side. For simplicity we assume that the first item set arrived at timestamp 1, and since then, at every timestamp a new item set was inserted.

So here the problem is, for an evolving stream S and a fixed item a, maintain a small summary of the stream in time, such that, at any time point t, mfreq (a, St) can be produced instantly from the summary.

More formally, We introduce a summary of a stream summary(S), an update procedure Update, and a procedure Get_mfreq, such that, if on timestamp t+1 an item set containing item i is added to the stream, Update(summary(St),i) equals to summary(St.<i>) equals to summary(St+1), and Get mfreq(summary(St+1)) equals mfreq(a, St+1). Because Update has to be executed every time a new item set arrives, it has to be very efficient, in order to be finished before the next item set arrives. Similarly, because the stream continuously grows, the summary must be independent of the number of item sets seen so far, or, at least grow very slowly as the stream evolves.

Based on the results of chapter-4, we present an incremental algorithm to efficiently maintain the summary for one item set A allowing us to produce the current maxfrequency (without minimal window length constraint) of an item set instantly at any time.

Stream	**time**	***mfreq*(a,S_t)**
< | a >	1	max (1/1) = 1
< | a a >	2	max (1/1, 2/2) = 2/2=1
< | a a a >	3	max (1/1, 2/2, 3/3) = 3/3=1
< | a a a b >	4	max (0/1, ½, 2/3, 3/4) = ¾

Figure 1: Max-frequency of a stream at every time point

In Fig. 1, the max-frequency has been given for an example evolving stream.

DEFINITION-1: A stream is defined as a statically object .In reality, a stream is an evolving object that is essentially unbounded .At every time point, a new item set will be inserted at the end of the stream.

DEFINITION-2: Many applications involve the generation and analysis of new kind of data, called "*stream data"*

In stream data, the data flow in and out of an observation window dynamically. These data are massive (e.g. terabyte in volume), temporarily order, fast changing and potentially infinite. Learning from data streams is an incremental task that requires *incremental algorithms* that take drift into account.

Example A satellite- mounted remote sensor that is constantly generating data is a bright example of stream data .Other examples are: telecommunications data, customer click, large sets of web pages, multimedia data, transaction data from the retail industry, scientific and engineering experiments and data from electric grids etc. These sources are called data streams.

In general, a stream $<I_1,I_2,I_3,\ldots\ldots I_n>$ is a sequence of item sets, denoted S ,where n is the length of the stream, denoted |S| . The stream of length 3, having on the first timestamp the singleton {a}, on the second timestamp the singleton {b} and on the third timestamp the singleton {c}, is denoted <a b c>, and the stream of length 3, with on the first timestamp the set {a, b}, on the second timestamp the singleton {c} and on the third timestamp the set {a, d, f}, is denoted <ab c adf>.

DEFINITION 3: The number of sets in a stream S that contain item i will be denoted count (i, S).

Example Count (a, <a b c>) is 1, since a occurs in exactly one set .For the other stream we have count (a, <ab c adf>) = 2.

DEFINITION 4:

The frequency of i in S, is defined
As: freq(i, S) := count(i, S)/ |S|

Example

freq (a, <a b c>) = 1/3 and
freq (a, <ab c adf>) = 2/3.

DEFINITION 5: The minimal frequency can sometimes reveal interesting knowledge. The min-frequency of item a is the minimum of the frequency of a over all windows extending from the end of the stream. Therefore, in order to find the minimal frequency of $a^|$, I can monitor the imaginary item a, and when we need the minimal frequency of a, it is easily found as 1: − *mfreq* ($a^|$, S).

DEFINITION.6:

Timestamp q is called a boarder for a set A in S if there exist a stream B such that q-*startmax*(A,S.B). Thus a boarder is a point in the stream that can still become a starting point of the maximal window.

DEFINITION 7:

The max-frequency of an item in a stream is defined as the maximum of the frequencies of over all windows extending from the end of the stream; that is:

The longest window in which the max-frequency is reached called the maximal window for in. And its starting point is denoted. That is, is the smallest index such that

.

The measure is used as a new frequency measure for stream mining. For a given item, its current frequency in the stream is defined as the maximal frequency over all evolving windows from a point in the stream until the end.

Example We focus on target item

Mfreq(a,<a b a a a b>)=

$\max_{k=1\ldots.6}$(freq(a. last(k,<a b a a a b>)))

= $\max_{k=1\ldots.6}$(0/1,1/2,2/3,3/4,3/5,4/6)= ¾

Mfreq(a,<b c d a b c d a>)=max(1/1,…….)=1

Mfreq(a,<x a a x a a x>)=

max (0/1,1/2,2/3,2/4,3/5,4/6,4/7)= 2/3

DEFINITION 8:

Frequent item sets refers to a set of items that frequently appear together in a transactional data set . e.g. a set of items such as milk and bread .

DEFINITION 9:
In this section we show some properties of max-frequency that are crucial for the incremental algorithm that maintains the summary of the stream. Obviously, checking all possible windows to find the maximal one is infeasible. Fortunately, not every point in the stream needs to be checked. The theoretical results from this section show exactly which points need to be inspected. These points will be called the borders.

The summary of the stream will consist exactly of the recording of these borders, and the frequency of the target item up to the most recent time point.

Definition 10:

The position q in S is called a border for a in S if there exists another stream B such that q = maxwin (a, S.B).
Let S be a stream, and let q = 1. Position q is a border for item a if and only if the first item set in the stream contains the target item a.
Let S be a stream, and let $2 \leq q \leq |S|$. Position q is a border for item a in S if and only if for all indices j, k with $1 \leq j < q$ and $q \leq k \leq |S|$, it holds that freq (a, S [j, q − 1]) < freq(a, S[q, k]).

3: THE SUMMARY

The summary for an item a in the stream S_t is the array that contains a pair (p, x/y) for every border at position p, with x the number of occurrences of a since p, i.e., count(a, S_t[p, t]), and y the length of the block from p until the end of the stream S_t, i.e., t − p + 1. The output of the algorithm in the case of r borders is written as an array of the form [(p_1, x_1/y_1),……….,(p_r, x_r/y_r)], visualized by

$T_t(a)$ =

p_1	…………	p_r
x_1/y_1	…………	x_r/y_r

This array is in fact *summary* (S_t) for item a, and is denoted by $T_t(a)$. If the target item is clear from the context, I use the abbreviation T_t. In this array, the border positions are ordered from old to most recent, reflecting in $p_1 < …… < p_r$.

4:THE INCREMENTAL ALGORITHM

1. Before the first target item set enters the stream, the array will remain empty. The pseudo-code of the algorithm to create T_{t+1}, based on T_t and the item set *I* that enters the stream at time t + 1 is given below.

2. In short, when a new item set I enter the stream; the frequencies are updated by increasing the nominators if *I* contain the target item, and always increasing the denominators.

3. If the item set *I* contains the target item, a new border will be added only if the frequency of the last block in T_t was not equal to 1.

4. Furthermore, I have to take into account that some of the borders of S_t might no longer be borders in S_{t+1}. This can only happen if the item set that enters the stream is a non-target item set (does not contain the target item).

Algorithm

Update (T_t, I),for target item a on time t + 1
Require: T_t = summary (S_t) = [(p_1, x_1/y_1), . . . , (p_r, x_r/y_r)]
Ensure: T_{t+1} = summary (S_{t+1}) = summary (S_t.<i>)

```
1.   Set $T_{t+1}$ := [ ]
2.   if ($T_t$ is empty) then
3.      if (target item a € I ) then
4.   $T_{t+1}$ := [(t + 1, 1/1)]
5.   else
6.        if (target item a  €  I) then
7.        for $1 \leq j \leq r$ do
8.        $T_{t+1}$  := $T_{t+1}$ + ($p_j$ , ($x_j$ + 1)/($y_j$ + 1))
9.        if $x_r$ != $y_r$ then
10.       Tt+1 := $T_{t+1}$ + (t + 1, 1/1)
11.  else
12.       high := 0
13.        for all j := 1 . . . r do
14.         if ($x_j$ )/($y_j$ + 1) > high then
15.         $T_{t+1}$ := $T_{t+1}$ + ($p_j$ , ($x_j$)/($y_j$ + 1))
16.         high := ($x_j$)/($y_j$ + 1)
```

5: EXPERIMENTS

From the description of the algorithm it is clear that the update procedure is very efficient, given

the summaries remain small. Producing the current frequency of the target item is obviously very efficient, as it amounts to a simple lookup in the most recent entry.
Hence, the complete approach will be feasible if and only if the summaries remain small.

6:CONCLUSION

In this paper We presented an algorithm which gives a new frequency measure for items in streams that does not rely on a fixed window length or a time-decaying factor. An experimental evaluation supported the claim that the new measure can be computed from a summary with extremely small memory requirements that can be maintained and updated efficiently so far.

7: REFERENCES

1.T.calders,N.Dexters, and B.Goethals .*Mining frequent items in a stream using flexible windows.Intelligen data analysis*.May 2008 .

2 .Ruoming J. and Agrawal G.: *An Algorithm for In-Core Frequent ItemsetMining on Streaming Data*. In Proc. 5th IEEE Int. Conf. on Data Mining (ICDM'05), pp 210–217. (2005)

3 .Agrawal R., Imielinski T. and Swami A.: *Mining Association Rules between Sets of Items in Large Databases.* In Proc. ACM SIGMOD Int. Conf. on Management of Data, pp 207–216. (1993)

4.Cormode G. andMuthukrishnan S.: *What's Hot andWhat's Not: Tracking Most Frequent Items Dynamically.* In Proc. PODS Int. Conf. Principles of Database Systems (2003)

5.Demaine E.D., Lopez-Ortiz A. and Munro, J.I.: *Frequency Estimation of Internet Packet Streams with Limited Space*. In Proc. of the 10th Annual European Symposium on Algorithms, pp 348–360. (2002)

6.Giannella C., Han J., Robertson E. and Liu C.: *Mining Frequent Item- sets Over Arbitrary Time Intervals in Data Streams*. In Technical Report TR587, Indiana University, Bloomington. (2003)

7.Giannella C., Han J., Pei J., Yan X. and Yu P.S.: *Mining Frequent Patterns In Data Streams at Multiple Time Granularities.* In H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha (eds), Next Generation Data Mining, pp 191– 212. (2003) .

8.Golab L., DeHaan D.,Demaine E.D., Lopez-Ortiz A. and Munro J.I.: *Identifying Frequent Items in Sliding Windows over On-Line Packet Streams*. In Proc. of the 1st ACM SIGCOMM Internet Measurement Conference, pp 173–178. (2003).

9.B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, *"Models and Issues in Data Stream Systems," Proc. of ACM PODS Symp.*, 2002.

10..Karp, R. M., Papadimitriou, C. H. and Shenker, S.: *A Simple Algorithm for Finding Frequent Elements in Streams and Bags*. In ACM Trans. on Database Systems 28, pp 51–55. (2003) .

11.Lee, D. and Lee, W.: *Finding Maximal Frequent Itemsets over Online Data Streams adaptively*. In Proc. of the 5th IEEE International Conference on Data Mining (ICDM), pp 266–273. (2005) .

12.Lee, L.K. and Ting, H.F.: *A Simpler and More Efficient Deterministic Scheme for Finding Frequent Items over Sliding Windows. In Proc. PODS Int. Conf. Principles of Database Systems.* (2006)

13.Lin C.-H., Chiu D.-Y., Wu Y.-H. and Chen A.L.P.: *Mining Frequent Item-sets from Data Streams with a Time-Sensitive Sliding Window.* In Proc. SIAM International Conference on Data Mining. (2005)

14 isra J. and Gries, D.: *Finding Repeated Elements. In Science of Computer Programming*, 2(2), pp 143–152. (1982).

15.Chi-Wing Wong R. and Wai-Chee Fu A.: *Mining Top-K Frequent itemsets from Data Streams. In Data Mining and Knowledge Discovery.*

16.Yu J.X., Chong Z., Lu H. and Zhou A.: *False Positive or False Negative:Mining Frequent Items from High Speed Transactional Data Streams.* In Proc. of the 30th International Conference on Very Large Databases, pp 204–215. (2004) .

17.L. Golab and M. Ozsu, "*Issues in Data Stream Management,*" *ACM SIGMOD Record*, 32(2): 5-14, 2003.