International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-2, Issue-4, September 2012

Generation of Frequent Patterns with Weights Over Continuous Flow of Data Efficiently

P.Satheesh, B.Srinivas, A.Satish Kumar

Abstract*— Mining data streams for knowledge discovery has been used in many applications like web click stream mining, network traffic monitoring, network intrusion detection, and dynamic tracing of financial transactions. In this paper, by analyzing characteristics of date stream, we propose an efficient algorithm weighted frequent pattern (WFP) mining that discovers more knowledge compared to traditional frequent pattern mining. The existing algorithms cannot apply for stream of data because those algorithms require multiple database scans. This technique uses a single database scan for mining stream of data. Our technique is efficient for web applications for mining web records and also discovers valuable knowledge compared to other techniques.

Index Terms*— Data stream, weight, weighted frequent pattern mining

I. INTRODUCTION

In recent years, data streams mining has been an important direction in data mining, a data stream is an ordered sequence of points that must be accessed in order and that can be read only small number of times or read only once. Unlike mining static databases, mining data streams poses many new challenges. First, it is unrealistic to keep the entire stream in the main memory or even in a secondary storage area, since a data stream comes continuously and the amount of data is unbounded. Second, traditional methods of mining on stored datasets by multiples scans are infeasible, since the streaming data is passed only once. Third, mining streams requires fast, real-time processing in order to keep up with the high data arrival rate and mining results are expected to be available within short response times[l].

Frequent patterns mining is an important data mining task with many real-world applications. By considering different weights of the items, weighted frequent pattern mining can discover more important knowledge compared to traditional frequent patterns mining. Take the shopping in supermarket for example, although the frequency of gold ring is very low compared to the frequency of pen sold, we would not treat them with the same criteria, because the gold ring is more important than pen due to its high weight. There is a great amount of work that studies mining frequent item-sets on static databases and many efficient algorithms have been proposed. Most of the existing weighted frequent pattern mining algorithms are devised for static databases that are not suitable for the data streams mining [2] [3] [4]. Motivated by these real world scenarios, in this paper, we propose a sliding window based novel technique WFPMDS (Weighted Frequent Pattern Mining over Data Streams). It can discover useful recent knowledge from a data stream by using a single scan. Our technique exploits a pattern growth mining approach to avoid the level-wise candidate generation-and-test problem. Besides retail market data, our technique can be well applied for mining weighted weighted web path traversal patterns. By considering different importance values for different websites, our algorithm can discover very important knowledge about weighted frequent web path traversals in real time using only one scan of data stream.

The remainder of this paper is organized as follows. In Section 2, we develop for weighted frequent pattern mining over data streams. In Section 3, our experimental results are presented and analyzed. Finally, in Section 4, conclusions are drawn.

Manuscript received on September 2012.

P.Satheesh, Associate Professor, CSE department, MVGR College of engineering, Chintalavalasa, Vizianagaram, Andhrapradesh, India.

B.Srinivas,CSE department, MVGR College of engineering, Chintalavalasa, Vizianagaram, Andhrapradesh, India.

A.Satish Kumar, CSE department, MVGR College of engineering, Chintalavalasa, Vizianagaram, Andhrapradesh, India.

II. MATHEMATICAL APPROACH

A. Frequent pattern mining

The support/frequency of a pattern is the number of transactions containing the pattern in the transaction database. The problem of frequent pattern mining is to find the complete set of patterns satisfying a minimum support in the transaction database. The downward closure property is used to prune the infrequent patterns. This property tells that if a pattern is infrequent then all of its super patterns must be infrequent. The Apriori algorithm is the initial solution of frequent pattern mining problem. But it suffers from the level-wise candidate generation-and-test problem and needs several database scans. The FP-growth algorithm solved this problem by using FP-tree based solution without any candidate generation and using only two database scans. Other research has been done to efficiently mine frequent patterns. However, this traditional frequent pattern mining considers equal profit/weight for all items.

B. Weighted frequent pattern mining

For example I = $\{i_1, i_2 \ldots\ldots i_j\}$ be a set of j items. A sequence is an ordered list of items from I denoted by $< s_1, s_2 \ldots s_k >$. A sequence ordered list if items S = $< a_1, a_2 \ldots\ldots a_p >$ is a subsequence of a sequence S ' = $< b_1, b_2, \ldots\ldots b_q >$ if there exist integers like $i_1 < i_2 < \ldots < i_p$ such that the ordered list of sequence items will be followed as $a_1 = b_{i1}$, $a_2 = b_{i2}$, .. , $a_p = b_{ip}$. A sequence s contains another sequence s' if s' is a subsequence of s. The count of a sequence S , denoted by count(s), is defined as the number of sequences that contain s. The support of a sequence s, denoted by support(s), is defined as count(s) divided by the total number of sequences seen. If support(s) $\geq . \sigma$, where (σ is a user-supplied minimum

support threshold) then we say that S is a frequent patterns.

Example 1: Suppose the length of our data stream is only 3 sequences:SI=(a,b,c),S2=(a,c), and S3=(b,c). Let us assume we are given that (σ =0.5. The set of sequential patterns and their corresponding counts are as follows: (a):2 , (b) :2, (c):3 ,(a,c): 2, and (b,c):2. The weight of an item is a non-negative real number which is assigned to reflect the importance of each item in the transaction database [3, 5]. For a set of items, I = {i1,i2,i3…..In},the weight of a pattern, F{x 1, x2, ...x m}, is given as follows

$$\text{Weight(F)} = \frac{\sum_{q=1}^{length(F)} Weight(x_q)}{length(F)} \quad (1)$$

A weighted support/frequency of a pattern is defined as the value that results from multiplying the pattern's support with the weight of the pattern [3, 5]. So, the weighted support of a frequent pattern F, is given as follows

Wsupport (F) = Weight (F) □ × Support (F) (2)

A pattern is called a weighted frequent pattern if the weighted support of the pattern is greater than or equal to the minimum threshold.

Example 2: take the example 1 for example, suppose that the weight of a, b, c is 0.5, 0.3, and 0.2, respectively, then Wsupport (< a, c>) is calculated based on equation (1) and (2) as follows:

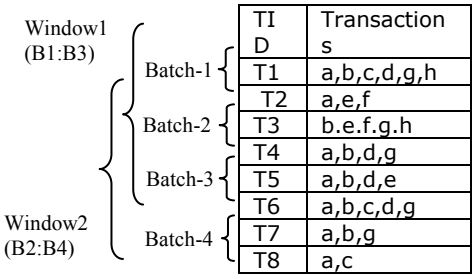
$$\text{Wsupport (a, c)} = \frac{0.5 + 0.2}{2} \text{ x } 2 = 0.7$$

So, we can say that sequential patter < a, c> is a weighted frequent pattern. The existing weighted frequent pattern mining methods need at least two database scans and therefore not suitable for stream data mining. Moreover, they cannot find important knowledge from the recent data. Hence, we propose a sliding window based novel technique for single-pass weighted frequent pattern mining over data streams.

C. Our Proposed Technique

A data stream may have infinite number of transactions. A batch of transactions contains a nonempty set of transactions. Figure 1 shows an example of transaction data stream which has divided into four batches with equal length. A window can be composed of 0xed number of non-overlapping batches. In our example data stream, we consider that one window contains three batches of transactions. Therefore, window1 contains batch1, batch2 and batch3. Similarly window2 contains batch2, batch3 and batch4. The weighted support of a pattern P can be calculated over a window W by multiplying its support in W with its weight. Therefore, pattern P is weighted frequent in W if its weighted support is greater than or equal to the minimum threshold. For example, if minimum weighted threshold is 2.0, “ab” is a weighted frequent pattern in window2. Its frequency in batch2, batch3 and batch4 are 1, 2, and 1 respectively. Accordingly, it has total frequency of 4 in window2. Its weighted support in window2 is 4×0.55 = 2.2, which is greater than the minimum weighted threshold.

1) Tree construction



TID	Transactions
T1	a,b,c,d,g,h
T2	a,e,f
T3	b.e.f.g.h
T4	a,b,d,g
T5	a,b,d,e
T6	a,b,c,d,g
T7	a,b,g
T8	a,c

(a) Transaction stream

Items	W
a	0.6
b	0.5
c	0.2
d	0.35
e	0.5
f	0.3
g	0.4
h	0.38

(b) Weight table

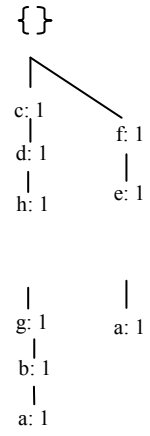
Figure 1. Example of transaction data stream and weight table

In this section, we describe the construction process of our tree structure to capture stream data using a single pass rotation. The header table is maintained to keep an item order in our tree structure. Each entry in a header table explicitly maintains item-id, frequency and weight information for each item. However, each node in a tree only maintains item-id and frequency information for each batch. To facilitate the tree traversals adjacent links are also maintained (not shown in the figures for simplicity) in our tree structure.

Consider the example data stream of Figure 1(a). At first we create the header table and keep all the items in weight ascending order. After that, we scan the transactions one by one, sort the items in a transaction according to header table sort order and then insert into the tree. The first transaction T1 has the items “a”, “b”, “c”, “d”, “g”, “h”. After sorting, we get the new order is “ c”, “ d”, “ h”, “g”, “ b” , “a”.

Header- Table

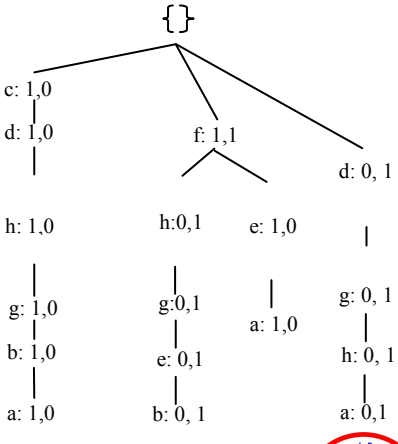
Item	W	F
c	0.2	1
f	0.3	1
d	0.35	1
h	0.38	1
g	0.4	1
e	0.5	1
b	0.5	1
a	0.6	2



(a) After inserting Batch-1

Header- Table

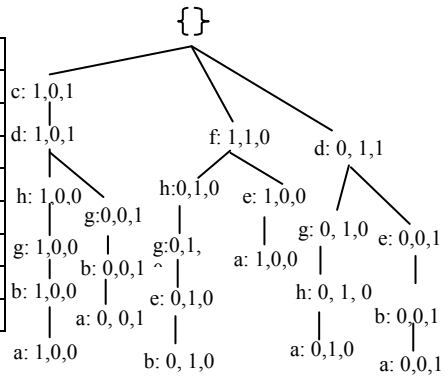
Item	W	F
c	0.2	1
f	0.3	2
d	0.35	2
h	0.38	2
g	0.4	3
e	0.5	2
b	0.5	3
a	0.6	3



(b) After inserting Batch-2

Header- Table

Item	W	F
c	0.2	2
f	0.3	2
d	0.35	4
h	0.38	2
g	0.4	4
e	0.5	3
b	0.5	5
a	0.6	5



(c) After inserting Batch-3

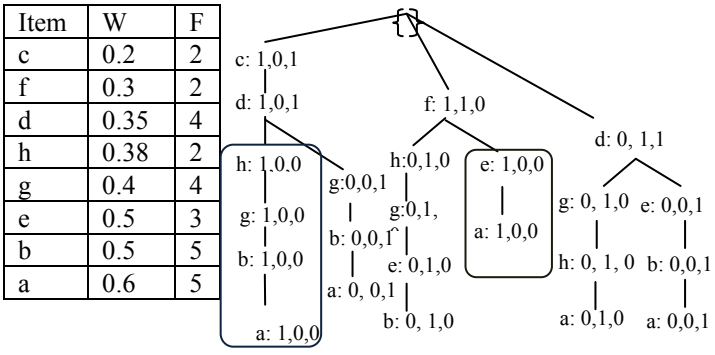
Figure 2. Tree construction for window1

Figure 2(a) shows the tree after inserting batch1. Figure 2(b) shows the tree after inserting batch2. In the same way batch3 is inserted into the tree. Figure 2(c) shows the final tree for window1.

When the data stream moves to batch4, it is necessary to delete the information of batch1 because window2 does not contain it. Therefore, information of batch1 is actually garbage information for window2. We delete the information. of batch1. Some nodes do not have any information for batch2 and batch3. As a result, they are deleted from the tree. Other nodes' frequency counters are shifted one bit left in order to remove the frequency information of batch1 and represent the last frequency information for batch4. As a consequence, now the three frequency information of any node represents batch2, batch3 and batch4. Figure 3(b) shows the tree after inserting batch4.

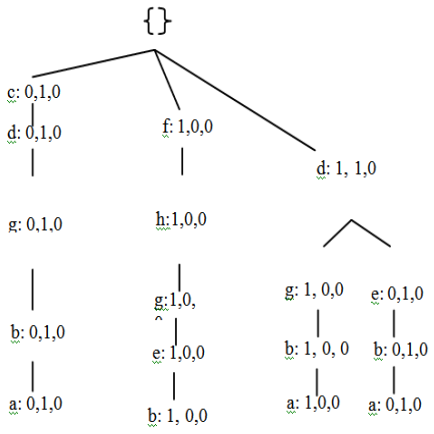
Header- Table

Item	W	F
c	0.2	2
f	0.3	2
d	0.35	4
h	0.38	2
g	0.4	4
e	0.5	3
b	0.5	5
a	0.6	5



Header- Table

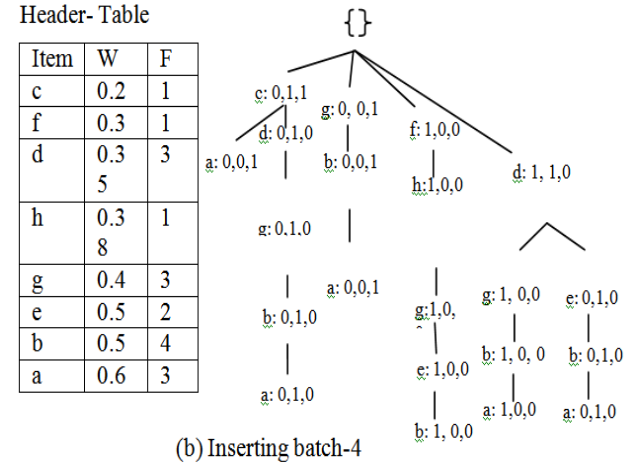
Item	W	F
c	0.2	1
f	0.3	1
d	0.35	3
h	0.38	1
g	0.4	3
e	0.5	2
b	0.5	4
a	0.6	3



(a) Deleting batch-1

Header- Table

Item	W	F
c	0.2	1
f	0.3	1
d	0.35	3
h	0.38	1
g	0.4	3
e	0.5	2
b	0.5	4
a	0.6	3



(b) Inserting batch-4

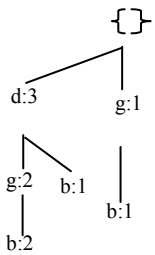
Figure 3. Tree construction for window2

2) *Mining process*

In this section, we describe the mining process of our proposed WFPMDS technique. As discussed in Section 2, the main challenge in this area is, the weighted frequency of an itemset does not have the downward closure property and to utilize this property we have to use the global maximum weight. The global maximum weight, denoted by GMAXW, is the maximum weight of all the items in the current window. For example, in Figure 1(b), the item "a" has the global maximum weight 0.6 for window1 and window2.

Header- Table

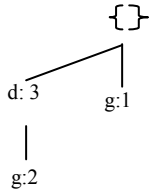
Item	W	F
d	0.35	3
g	0.4	3
b	0.5	4



(a). Conditional tree for item 'a'

Header- Table

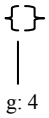
Item	W	F
d	0.35	3
g	0.4	3



(b). Conditional tree for item 'ab'

Header- Table

Item	W	F
g	0.4	4



(c). Conditional tree for item 'b'

Figure 4. Mining process

The local maximum weight, denoted by LMAXW, is needed when we are doing the mining operation for a particular item. As the tree is sorted in weight ascending order, we get advantage in the bottom up mining operation. For example, after mining the weighted frequent patterns prefixing the item "a", when we go for mining operation prefixing the item "b", then the item "a" will

never come in any conditional trees. As a result, now we can easily assume that the item "b" has the maximum weight. This type of maximum weight in mining process is known as LMAXW. As LMAXW is reducing from bottom to top, the probability of a pattern to be a candidate is also reduced.

Suppose we want to mine the recent weighted frequent patterns in the data stream presented at Figure 1. It means that we have to find out all the weighted frequent patterns in window2. Consider the minimum threshold = 1.8. Here the GMAXW = 0.6 and after multiplying the frequency of each item with GMAXW, the weighted frequency list is <c:1.2, f:0.6, d:1.8, h:0.6, g:2.4, e:1.2, b:3.0, a:3.0>. As a result, the candidate items are "d", "g", "b" and "a". Now we construct the conditional trees for these items in a bottom up fashion and mine the weighted frequent patterns. At first the conditional tree of the bottom-most item "a" (shown in Figure 4(a)) is created by taking all the branches prefixing the item "a" and deleting the nodes containing an item which cannot be a candidate pattern with the item "a". For item "a", LMAXW =0.6 and we can get the weighted frequency list for item "a" by multiplying the other item's frequency with LMAXW. Obviously this weighted frequency is the maximum possible weighted frequency of an itemset prefixing item "a". Hence, we have to take all the patterns as a candidate having maximum weighted frequency greater than or equal to minimum threshold. Accordingly, the weighted frequency list for the item "a" is <d: 1.8, g: 1.8 b: 2.4> (we should not consider the global non-candidate items "c", "f", "h" and "e".) and candidate patterns "ad", "ag", "ab" and "a" are generated here. In the similar fashion, conditional tree for the pattern "ab" is created in Figure 4(b) and candidate frequent patterns "abd " and "abg " are generated from the figure.

For item " b", the LMAXW= 0.5 as item " a" will not come out here. The weighted frequency list is <d: 1.5, g: 2.0 >. The key point is that, the maximum weighted frequency of item "d" with item "b" is 3□□× 0.5 = 1.5, as LMAXW reduces from 0.6 to 0.5. Now, without further calculation we can prune "d". But if LMAXW is 0.6 at this place, the weighted frequency of "d" is 3□□×□0.6 = 1.8 and as a result it becomes a candidate. This is one of the strength available in our tree structure. The conditional tree of item " b" contains only one conditional tree of item set " g" (shown in Figure 4(c)) and the candidate pattern "bg" is generated. For item set "g" the LMAXW= 0.4 and the weighted frequency list are <d: 0.8 >. As a result, we do not have to create any conditional tree for the item set "g". We have to test all the candidate patterns with their actual weights and the weighted frequency and mine the actual weighted frequent patterns. The actual weighted frequent patterns in window2 are <a: 3.0, b: 2.5, ab: 2.2, bg: 1.8 >.

III. RESULTS & DISCUSSION

To evaluate the performance of our proposed technique, we have performed several experiments on real-life kosarak dataset. Table shows the characteristics of these datasets. These datasets do not provide the weight values of each item. As like the performance evaluation of the previous weight based frequent pattern mining we have generated random numbers for the weight values of the items, ranging from 0.1 to 0.9. our programs written in Microsoft visual c++ 6.0 and run with the windows XP operating system the a Pentium dual core 2.13GHz cpu with 1Gb memory.

Table 1. Dataset characteristics

Dataset	Size(MB)	No.of trans	No.of distinct items	Avg.trans length
Kosarak	30.5	990,002	41,270	8.1

The kosarak dataset contains web click-stream of a Hungarian on-line news portal. It is a big dataset containing almost one million transactions and 41,270 distinct items. Here we have used the following parameters of size= 100k, 150k and 200k and window size= 3 batches and minimum threshold is from 2% to 6%.

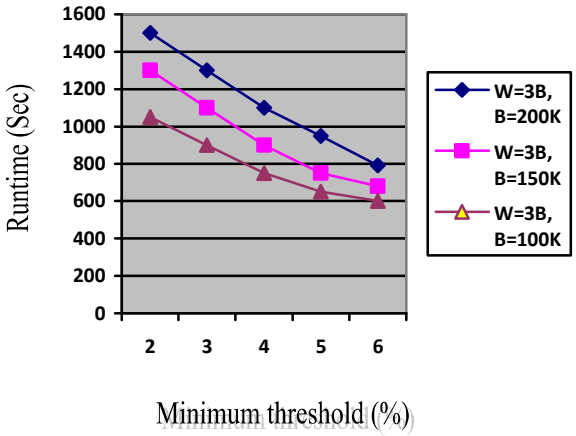


Figure 5. Performance on Kosarak dataset

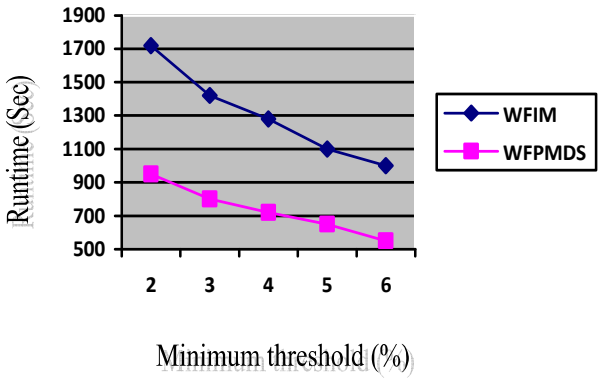


Figure 6. Run time comparison Kosarak dataset

We show the comparison of our technique with existing WFIM algorithm in Kosarak dataset. The existing WFIM algorithm is not suitable for stream data mining due to scanning a database at least twice. In the first scan, it finds all single-element candidate patterns and in the second scan it performs the tree creation and mining operation. Moreover, it cannot keep batch by batch information in the tree for sliding window-based stream data mining. The figure shows that WFPMDS out performs WFIM significantly with respect to execution time.

IV. CONCLUSION

In this paper, we propose a sliding window based novel technique for weighted frequent pattern mining over data streams. Our main goal is to mine recent weighted frequent patterns from stream data. By using an efficient tree structure, our proposed technique WFPMDS can capture

the recent data form a data stream. WFPMDS requires only a single-pass of data stream for tree construction and mining operations. Therefore, it is quite suitable to apply in real time data processing to discover valuable recent knowledge. Moreover, the tree structure used by our technique is easy to construct and handle. The performance analyses show that our technique is more efficient for weighted frequent pattern mining over continuous flow of data.

REFERENCES

[1] James Cheng, Yiping Ke, and Wilfred Ng, "A Survey on algorithms for mining frequent itemsets over data streams,"
[2] C.Raissi, P.Poncelet, and M.Teisseire, "Towards a new approach for mining frequent itemsets on data stream," Journal of Intelligent Information Systems, vol. 28, pp.23-36, 2007.
[3] A.Metwally, D.Agrawal, and A.E.Abbadi, "An integrated efficient solution for computing frequent and top-k elements in data streams," ACM Transactions on database systems, vol. 31, pp.l 095-1133, 2006.
[4] N.Jiang, L.Gruenwald, "Research issues in data stream association rule mining," SIGMOD Record, vol 35, pp. 14-19, 2006.
[5] C.K.-S.Leung, Q.I.Khan, "DSTree: A tree structure for the mining of frequent sets from data streams," Proc.Sixth IEEE Int'l Conf. on Data Mining, pp.928-932, 2006.
[6] U.Yun, J.J.Leggett, "WFIM: weighted frequent itemset mining with a weight range and a minimum weight," Proc.Fifth SIAM Int. Conf. on Data Mining, pp.630-640, 2005.
[7] U.Yun, "Efficient mining of weighted interesting patterns with a strong weight and/or support affinity," Information Sciences, vol. 177, pp.3477-3499, 2007.
[8] R.Agrawal, A. Swami, "Fast algorithm for mining association rules," In Proc. Of the 20th IntI. Conf. on Very Large Data Bases, September 1994.
[9] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, and Byeong Soo Jeong, "Efficient mining of weighted frequent patterns over data streams," 2009 11th IEEE International Conference on High Performance Computing and Communications.

P.Satheesh received M.Tech in computer Science and Technology in 2006 from Andhra University; he has ten years of teaching experience. He is currently employedas a an Associate professor in CSE department, MVGR College of Engineering. He has more than ten papers in journals.

B.Srinivas received M.Tech in computer Science and engineering in 2008 from Acharya Nagarjuna University; he has two and half years of industry and four years of teaching experience. he is currently employed as a an assistant professor in CSE department, MVGR College of Engineering. He has more than eight papers in journals.

A.Satish Kumar currently pursuing M.Tech in Computer Science Engineering from JNTU, Kakinada.