Mining Weighted Association Rules in Data Streams with a Sliding Window

Weimin Ouyang, Qinhua Huang
Department of Computer Teaching,
Shanghai University of Political Science and Law
Shanghai, China
{oywm,hqh}@shupl.edu.cn

***Abstract*—Association rule mining is one of the most important data mining techniques. Typical association rules consider each item in transactions with the same significance.In order to represent significances of items, every item has be assigned with a weight, and the mining weighted association rules have been proposed. All of the literature on weighted association rules mining, to our best knowledge, is confined to the traditional, relatively static database environment; no research work has been conducted on mining weighted association rules over data streams. In this paper, we propose an algorithm for mining weighted association rules over data streams. Experiments on the synthetic data stream are made to show the effectiveness and efficiency of the proposed approach.**

Keywords—association rules; weighted association rules; data streams

I. INTRODUCTION

A data stream is an ordered sequence of elements which arrives one by one with positive real time intervals [1]. It is often refer to as streaming data. Different from data in traditional static datasets, a data stream is continuous, huge, fast changing, rapid and infinite. Many applications generate large amount of data streams in real time, such as sensor data generated from sensor networks, online transaction flows in retail chains, Web log and click-streams in Web applications, call records in telecommunications, etc. The nature of streaming data makes it essential to use online algorithms which require only one scan over the data for knowledge discovery [2].

Association rule is one of the most important data mining topics [3,4]. Typical association rules consider each item in transactions with the same significance. However, it is not ture in the real world. In order to represent significances of items, every item has be assigned with a weight, and algorithms for mining weighted association rules have been proposed in literature [5,6].

All of the literature on weighted association rules mining, to our best knowledge, is confined to the traditional, relatively static database environment; no research work has been conducted on mining weighted association rules over data streams. In this paper, we propose an algorithm for mining weighted association rules over data streams. Experiments on the synthetic data stream are made to show the effectiveness and efficiency of the proposed approach.

The rest of the paper is organized as follows. The definitions for weighted association rules in data stream with sliding window are given in section 2. In section 3, we describe the algorithm to find weighted association rules in data stream with sliding window, and Section 4 presents our experimental results. The conclusion and future works are made in the last section.

II. PROBLEM DEFINITIONS

A. Weighted Association Rule

In order to express the significance of items, a weight value is assigned for each item x denoted as ω(x). We arrange itemset according to weight of items in descending order. Let X be an itemset $\{x_1,x_2, \dots , x_k\}$, where $w(x_1)\geq w(x_2)\geq \dots \geq w(x_k)$. Using this itemset arrangement, we can keep the downword closure property hold which will be proved late.

The weighted support of an itemset $X = \{ x_1, x_2, \dots , x_k \}$ in which $w(x_1) \geq w(x_2) \geq \dots \geq w(x_k)$ is defined as $w\sup(X) = (\sum_{i=1}^{k} w(x_i))\sup(X)$ .The downward closure property of frequent itemset will hold true.

Let $X = \{ x_1, x_2, \dots , x_{k-1} \}$ and $Y = \{ y_1, y_2, \dots , y_{k-1}\}$, k>=2. if $x_i=y_i$, i =1,2,3,…, k-2, and $w(x_{k-1})>w(y_{k-1})$, the X and Y is called as joinable.

If $X = \{ x_1, x_2, \dots , x_{k-1} \}$ and $Y = \{ y_1, y_2, \dots , y_{k-1}\}$(k>=2) is joinable denoted as $X\bullet Y= \{ x_1, x_2, \dots , x_{k-1} , y_{k-1}\}$.

Theorem 1: Let itemset $X = \{ x_1, x_2, \dots , x_k \}$ and $Y = \{ x_1, x_2, \dots , x_{k-2}, x_{k-1} \}$, $Z = \{ x_1, x_2, \dots , x_{k-2},x_k \}$and $w(x_1)\geq w(x_2)\geq \dots \geq w(x_k)$. If X is a weighted frequent itemset, then Y and Z will be both weighted frequent itemset.

Prove:

Because $x_i = 0$ or 1 for I = 1,2, 3,…, k, $\prod_{i=1}^{k-1} x_i \geq \prod_{i=1}^{k} x_i$, and

$$\sup(X) = \prod_{i=1}^{k} x_i \ , \ \sup(Y) = \prod_{i=1}^{k-1} x_i \ .$$

Hence, sup(Y)≥sup(X).

Because $w(x_1) \geq w(x_2) \geq \ldots \geq w(x_k)$, so

$$w(X) = \max_{i=1}^{k} w(x_i) = w(x_1),$$

$$w(Y) = \max_{i=1}^{k-1} w(x_i) = w(x_1)$$

Hence, $w(X) = w(Y)$

We have, $w\sup(X) = w(X)\sup(X), w\sup(Y) = w(Y)\sup(Y)$.

So, $w\sup(Y) \geq w\sup(X)$.

With the same way, $w\sup(Z) \geq w\sup(X)$ is also true.

Therefore, if X is a weighted frequent itemset, then Y and Z will be both weighted frequent itemset

Theorem **2:** For k>2, any weighted frequent k-itemset X can be obtained by joining two weighted frequent (k-1)-itemset.

Prove: For any weighted frequent k-itemset X = { x_1, x_2, … , x_k }, let Y = { x_1, x_2, … , x_{k-2},x_{k-1} }, Z = { x_1, x_2, … , x_{k-2},x_k }.

According to theorem 1, Y and Z are both weighted frequent (k-1)-itemset.

Hence, any weighted frequent k-itemset X can be obtained by joining two weighted frequent (k-1)-itemset.

Clearly, we can directly apply algorithm Apriori to mining weighted association rules only if using weighted support instead of support.

Definition 1: Given a transaction database, and a user-defined minimum support threshold *s*, The form (X, Y) is a weighted association rule, if and only if $wsup(X \cup Y) \geq s$ and $wconf(X, Y) = wsup(X \cup Y)/wsup(X) \geq c$.

B. Weighted Association Rule in Data Stream

Let I = {i_1, i_2, …, i_m} be a set of items. A transaction T = (tid, x_1, x_2, 。。。, x_n), $x_i \in$ I, for $1 \leq i \leq n$, is a subset of I, while n is called the size of the transaction, and tid is the unique identifier of the transaction. A non-empty subset of I is called itemset. An itemset containing k items is called k-itemset.

Definition 2: A transaction data stream TDS = T_1, T_2, …, T_N is a continuous sequence of transactions, where N is the tid of latest incoming transaction T_N.

A transaction-sensitive sliding window in the transaction data stream is a window that slides forward for every transaction. The window at each slide has a fixed number, w, of transactions, and w is called the size of the window. Hence, the current transaction-sensitive sliding *window* is $TransSW_{N-w+1} = [T_{N-w+1}, T_{N-w+2}, \ldots, T_N]$, where $N-w+1$ is the index of the first transaction of the current window SW.

Definition 3: The **support** of an itemset *X* in *SW*, denoted as $\mathbf{sup}(X)^{SW}$, is the number of transactions in *SW* containing *X* as a *subset*.

Definition 4: An itemset *X* is called a **frequent** if $\mathbf{sup}(X)^{SW} \geq s \cdot w$, where *s* is a user-defined minimum support threshold, c is a user-defined minimum confidence. The value $s \cdot w$ is called the **frequent threshold** of *SW***.**

Definition 5: Given a transaction-sensitive sliding window *SW*, and a user-defined minimum support threshold *s*, The form (X, Y) is a weighted association rules in window *SW*, if and only if $wsup(X \cup Y)^{SW} \geq s \cdot w$ and $wconf(X, Y)^{SW} = wsup(X \cup Y)^{SW} / wsup(X)^{SW} \geq c$.

III. MINING WEIGHTED ASSOCIATION RULES IN DATA STREAM WITH SLIDING WINDOW

According to the definitions of positive and negative association rules in last section, we propose an algorithm to discover both positive and negative association rules in data stream called *MPNAR-SW* (Mining Positive and Negative Association Rules in a Sliding Window). In the proposed algorithm, for each item X in the current sliding window SW, we construct a bit-sequence with w bits denoted as Bit(X). If an item X is in the i-th transaction of current window SW, the i-th bit of Bit(X) is set to be 1; otherwise, it is set to be 0. The process is called bit-sequence transform.

For example, in Table 1, the first sliding window SW_1 consists of three transactions: <*Tid1*, (*abd*) >, <*Tid2*, (*bcd*) >, <*Tid3*, (*be*) > and <*Tid4, (bde)*>, the *window* SW_2 consists of transactions: <*Tid2*, (*bcd*) >, <*Tid3*, (*be*) >, <*Tid4, (bde)*> *and*<*Tid5, (bd)*> . Because item *a* only appears in the 1st transactions of *window* SW_1, the bit-sequence of *a*, **Bit**(*a*), is 1000. Similarly, **Bit**(*b*) = 1111, **Bit**(*c*) = 0100, **Bit**(*d*) = 1101, and **Bit**(*e*) = 0011.

TABLE I. TABLE 1: BIT-SEQUENCES OF ITEMS IN WINDOW INITIALIZATION PHASE OF SW

Window-id	Transactions	Bit-Sequences of items
SW_1	<*Tid1*, (*abd*) >	Bit(a)=1000, Bit(b)=1111
	<*Tid2*, (*bcd*) >	Bit(c)=0100,Bit(d)=1101
	<*Tid3*, (*be*) >	Bit(e)=0011
	<*Tid4, (bde)*>	

The proposed algorithm MWAR-SW is described as follows:

Algorithm *MWAR-SW*

Input: *TDS* (a transaction data stream), minimum support threshold: s ; the minimum confidence threshold: c; minimum interesting threshold mininterest; the user-specified sliding window size *w*.

Output: Set of weighted association patterns: WAR ;

```
Begin
  SW = Null; /* Window SW consists of w transactions */
  Repeat:
    For each incoming transaction Ti in SW
      do
        If SW = Full then
            Do bitwise-shift on bit-sequences of all items in
        SW;
        else
              For each item X in Ti do
                  Do bit-sequence
          transform(X);
        EndIf
    ;
    For each bit-sequence Bit(X) in SW do
       If sup(X) = 0 then Drop X from SW;
  EndRepeat
/* The following is the frequent and infrequent itemsets
generation phase. */
  Fres=∅;
  S= ∅;
  F1 = {frequent 1-itemsets};
  For (k=2; Fk-1≠ Null; k++) do {
       Ck = Candiate_Gen(Fk-1);
       Do bitwise AND to find the supports of Ck;
       Fk = { ck ∈Ck | wsup(ck)^SW ≥s·w };
                              Sk = Ck − Fk;
    FreS = FreS ∪ Fk;
    S=S ∪ Sk;
    }
/* The following is the positive association rules generation
phase. */
  WAR = ∅;
  For each itemset i in FreS Do {
    For any X∪Y=i and X ∩Y = ∅ Do {
          If wsup(X∪Y)^SW − wsup(X)^SW × sup(Y)^SW >=
                          mininterest∧
       wsup(X∪Y)^SW >= s ∧ wconf(X, Y)^SW ≧ c
     Then WAR = WAR ∪ { X⇒Y };
    }
     }
End
```

The proposed MWAR-SW algorithm consists of four phases, window initialization phase, window sliding phase, and weighted-frequent itemsets generation phase and weighted association rules generation phase. Since the approach to generate weighted association rules is the same with the well-known Apriori algorithm, and the method to generate weighted association rules is straightforward, the description of weighted association rules generation phase is omitted in this paper.

(1) Window Initialization Phase

The phase is processed when the number of transactions come into the current window so far is less than or equal to a user-predefined sliding window size w. In this phase, each item in the new incoming transaction is transformed into its bit-sequence representation. Before this phase, for each item X in I, the bit-sequence Bit(X) is initialized with 0.

For example, in Tab. 1, the first sliding window SW_1 contains four transactions: Tid1, Tid2, Tid3 and Tid4. The bit-sequences of items of SW1 in the window initialization phase are shown in Table 2.

TABLE II. BIT-SEQUENCES OF ITEMS IN WINDOW INITIALIZATION PHASE OF SW1

Tid	Items	bit-sequence transformation in SW1
Tid1	(*abd*)	Bit(a)=1000, Bit(b)=1000, Bit(c)=0000,Bit(d)=1000,Bit(e)=0000
Tid2	(*bcd*)	Bit(a)=1000, Bit(b)=1100, Bit(c)=0100,Bit(d)=1100,Bit(e)=0000
Tid3	(be)	Bit(a)=1000, Bit(b)=1110, Bit(c)=0100,Bit(d)=1100,Bit(e)=0010
Tid4	(bde)	Bit(a)=1000, Bit(b)=1111, Bit(c)=0100,Bit(d)=1101,Bit(e)=0011

(2) window sliding phase

The phase is activated after the number of transactions in the sliding window SW is w. Before a new incoming transaction is appended to the sliding window, the oldest transaction is removed from the window.

For removing the oldest transaction, a simple method is used in the proposed algorithm. Since the MDIAR-SW algorithm use bit-sequence representation, we can uses the bitwise left shift operation to remove the oldest transaction from the current sliding window.

For appending a new transaction, the same as Window Initialization Phase, bit-sequence transformation is processed.

After sliding the window phase, an effective pruning method, called Item-Prune, is used to improve the memory usage. The pruning method is that an item X in the current sliding window is dropped if and only if $wsup(X)^{SW} = 0$.

For example, in Figure 1, before the fifth transaction <Tid5, (bd)> is processed, the first transaction Tid1 must be removed from the current window using bitwise left shift on the set of items. Hence, Bit(a) is modified from 1000 to 0000. Similarly, Bit(b)= 1110,Bit(c)= 1000, Bit(d)= 1010, and Bit(e)= 0110. Then, the new transaction <T4, (be)> is processed by bit-sequence transform. The result is shown in Table 3.

TABLE III. BIT-SEQUENCES OF ITEMS IN WINDOW SLIDING PHASE OF SW2

Window-id	Transactions	Bit-Sequences of items
SW_2	<*Tid2*, (*bcd*) >	Bit(a)=0000, Bit(b)=1111
	<*Tid3*, (*be*) >	Bit(c)=1000,Bit(d)=1011
	<*Tid4, (bde)*>	Bit(e)=0110
	<*Tid5, (bd)*>	

Note that item a is dropped since Bit(a)=0000, i.e., $sup(a)^{SW} = 0$.

(3) frequent and infrequent itemsets generation phase

In this phase, MWAR-SW algorithm uses a level-wise method to generate the set of candidate itemsets C_k from the frequent itemsets F_{k-1} according to the Apriori. The step is called Candidate_Gen. Then, the proposed algorithm uses the bitwise AND operation to count the support of these candidates in order to find the frequent and infrequent k-itemsets F_k and S_k. The process is stopped until no new candidates are generated.

For instance, consider the bit-sequences of SW_2 in Figure 3, and let the minimum support threshold s to be 0.5. Hence, an itemset X is frequent if $wsup(X)^{SW} \geq 0.5*4 = 2$. In the following, we discuss the step of frequent and infrequent itemset mining of $TransSW_2$.

Firstly, MWAR-SW algorithm find out frequent 1-itemset F_1={(b),(d),(e)}, then generates three candidate 2-itemsets, (bd), (be) and (de), by combining frequent 1-itemsets: (b), (d) and (e), where Bit(b) = 1111, i.e., sup(b) = 4, Bit(d) = 1011, i.e., sup(d) = 3, and Bit(e) = 0110, i.e., sup(e) = 2. 1-itemset (c) is an infrequent itemset, since its Bit(c) = 1000, i.e., sup(c) = 1. After using bitwise AND operations to count the supports of these candidates, F_2={(bd), (be)}, S_2 =(de), because the Bit(bd) =1011, sup(bd) = 3, Bit(be)=0110, sup(be) = 2, Bit(de) =0010, sup(de) = 1. Secondly, MPNAR-SW generates one candidate 3-itemset (bde) and uses bitwise AND operation to count the sup(bde) = 1, i.e., Bit(bd) AND Bit(be) = 0010. The 3-itemset (bde) is infrequent. Because no new candidates are generated, the generation of frequent and infrequent itemset process is stopped. Hence, there are five frequent itemsets, (b), (d), (e), (bd), (be), infrequent itemsets, (c),(de),(bde), generated by MWAR-SW algorithm in $TransSW_2$.

IV. Experiment

In this section, we evaluate the performance of our proposed algorithm for mining indirect temporal sequential patterns. The computation environments are i5-3470, 4G RAM, Windows 7 operating system. The algorithm is implemented with C++. The synthetic experiment data set is generated by Assocgen [4].

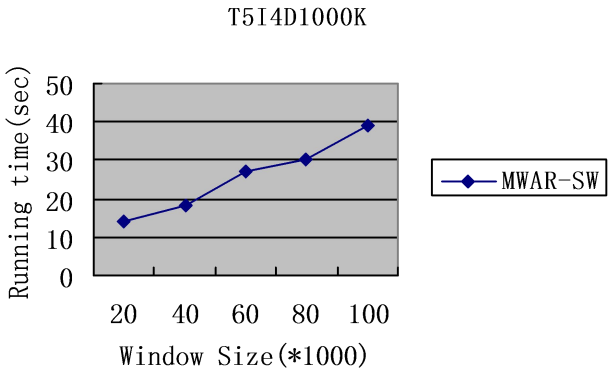

Fig. 1. Running time in window initialization phases of algorithm MWAR-SW under different window size.

The synthetic data stream, denoted as T5I4D1000K, of size 1 million transactions (D1000K) has an average transaction size of 5 items (T5) with average maximal frequent itemset size of 4 items (I4). In the experiments, the transactions of T5I4D1000K are looked up in sequence to simulate the environment of an online data stream.

Figure 1 shows the processing time of window initialization phase under different window sizes from 20,000 (200K) transactions to 100,000 (1,000K) transactions. Figure 2 shows the total time of window sliding time and weighted associatiuon rules mining time at each 100K transactions using various window sizes from 200K transactions to 1000K transactions. As shown in Figure 1 and 2, MWAR-SW algorithm is efficient and scalable.

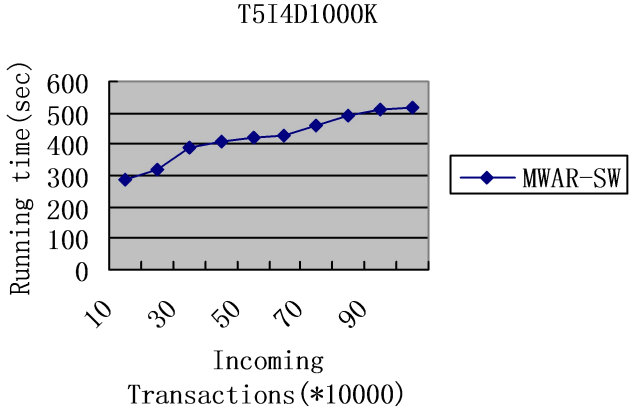

Fig. 2. Running time including window sliding time and weighted association rules generation time of algorithm MWAR-SW under different window size.

V. Conclusions

In this paper, we proposed an efficient one-pass algorithm, called MWAR-SW, for mining weighted association rules over online data streams with a sliding window. Experiments show that the proposed algorithm is efficient and scalable.

References

[1] N. Jiang, and L. Gruenwald. Research Issues in Data Stream Association Rule Mining. In SIGMOD Record, Vol. 35, No. 1, Mar. 2006.

[2] Babcock B, Babu S, Datar M, et al. Models and issues in data stream systems. In Proc. of the 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Wisconsin: Madison, 2002:1-16.

[3] Agrawal R,Srikant R. Fast Algorithms for Mining Association Rules In the Proc. of the 20th International Conference on VLDB. Santiago, 1994. pp.487~499.

[4] Brin,S., Motwani,R. and Silverstein,C., Beyond Market Baskets: Generalizing Association Rules to Correlations, in Proc. of ACM SIGMOD Conf., pp.265-276, May 1997.

[5] Cai Chun Hing, Fu Ada Wai-Chee, Cheng Chun-hung, et al. MiningAssociation Rules with Weighted Items, in Proc. of International Database Engineering and Applications Symposium. pp.68-77, 1998.

[6] R.Srikant, Q.Vu,R.Agrawal, Mining association rules with item constraints, in Proc. of the Third Int'l Conf. in Knowledge Discovery in Databases and Data Mining. pp.67-73,1997.