Discovery of both Direct and Indirect Association Rules in Data Streams with a Sliding Window

Weimin Ouyang, Qinhua Huang

Modern Education Technology Center, Shanghai University of Political Science and Law, Shanghai 200438, China
{oywm; hqh}@shupl.edu.cn

Abstract - Association rules mining is a most of important tasks in data mining research. While most of the existing discovery algorithms are dedicated to efficiently mining of frequent patterns, it has been noted recently that some of the infrequent patterns can provide useful insight view into the data. As a result, indirect association rules have been put forward. All the existing algorithms for mining indirect association rules need get all frequent itemsets, and confined to the traditional static database. Instead of this method, we put forward an approach to discover both direct and indirect association rules in data streams with a sliding window. Experiments on the synthetic data stream are made to show the effectiveness and efficiency of the proposed approach.

Index Terms - Direct Association Pattern, Indirect Association Pattern, 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 rules mining is a most of important tasks in data mining research. A number of algorithms have been proposed to improve the running time for generating frequent itemsets and association rules since the problem was pointed out by R.Agrawal in 1993[3,4].

With the further research on the mining of frequent patterns, it has been recognized that some infrequent patterns can provide very useful insight view into the data set, and a new kind of knowledge discovery problems called as indirect associations has been proposed[5]. Consider a pair of item x and y, which are rarely present together in the same transaction. If both items are highly dependent on the presence of another itemsets M, then the pair of x and y is said to be indirectly associated by M called as mediator.

II . Related Work

A . Data Stream Mining

Due to the characteristics of data streams, the algorithms for mining data streams require only one scan over the data stream. Consequently, previous multiple-pass data mining algorithms studied for static datasets are not feasible for mining data streams.

According to the data stream processing model [2,6,7], the research of mining data streams can be divided into three categories: landmark windows, sliding windows, and damped windows, as described briefly as follows. In the landmark window model, knowledge discovery is performed based on the values between a specific timestamp called landmark and the present. In the sliding window model, knowledge discovery is performed over a fixed number of recently generated data elements which is the target of data mining. Two types of sliding widow, i.e., transaction-sensitive sliding window and time-sensitive sliding window, are used in mining data streams. The basic processing unit of sliding window of first type is an expired transaction while the basic unit of sliding window of second one is a time unit, such as a minute or an hour. In the damped windows model, knowledge discovery is performed over the data set between the beginning of data stream and the present.

B . Indirect Association Mining

The original indirect association mining approach [5] is shown as follows.

Algorithm: INDIRECT for Mining indirect associations

```
1. Discover all frequent itemsets $L_1$ , $L_2$ ,… , $L_n$ using
   Apriori, $L_k$ (k=1,2, … ,n)where is the set of all
   frequent i-itemsets.
2. SIA =     ; //set of indirect associations
3. for k = 2 to n {
4. $C_{k+1}$ = join($L_k$, $L_k$);
5. for each <x, y, M>     $C_{k+1}$ {
   if (sup(x, y)<$t_s$ AND dep({x},M)>=$t_d$ AND
   dep({x},M)>=$t_d$)
6. SIA = SIA     {<x, y, M>};
7.     }
8.   }
```

The algorithm is divided into two major phases: (1)get all frequent itemsets using Apriori (step 1); (2) discover all indirect associations by candidate generation (step 4) and candidate pruning (step 5~8).During the candidate generation step, frequent itemset L_k is used to generate candidate indirect associations C_{k+1} for pass k+1.Each candidate in C_{k+1} is a triplet <x,y,M>, where x and y are the items which are indirectly associated by mediator M. C_{k+1} is generated joining the frequent itemsets in L_k. During the join, a pair of frequent itemsets $\{x_1,x_2, \dots , x_k\}$ and $\{ y_1,y_2, \dots , y_k \}$ are joinable if the two itemsets have exactly k-1 items in common.If so,they generate a candidate indirect association <x, y, M>, where x and y are the different items,one from each k-itemset, and M is the set of common items. For example, two itemsets {a,b,c,d} and {a,b,d,e} can be joined together to generate a candidate indirect association <c,e,{a,b,d}>. Since the candidate indirect associations are generated by joining two frequent itemsets, they certainly satisfy the mediator support condition. Therefore, in the candidate pruning step, only the itempair support condition and mediator dependence condition are needed to be checked.

III . Problem definition

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items. A transaction $T = (\text{tid}, x_1 \cdot x_2 \cdot \circ \circ \circ \cdot x_n)$, x_i 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 1: A transaction data stream $TDS = T_1, T_2, \dots, T_N$ 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 $TransSW_{N-w+1} = [T_{N-w+1}, T_{N-w+2}, \dots, T_N]$, where $N-w+1$ is the id of current window *SW*.

Definition 2: 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*. **Itemset.**

Definition 3: An itemset *X* is called a **frequent** if $\mathbf{sup}(X)^{SW} \geq s \quad w$, where *s* is a user-defined minimum support threshold, c is a user-defined minimum confidence. The value $s \quad w$ is called the **frequent threshold** of *SW***.**

Definition 4: Given a transaction-sensitive sliding window *SW*, and a user-defined minimum support threshold *s*, The form (X,Y) is a association rules in window *SW*,, if and only if $\sup(X \quad Y)^{SW} \geq s \quad w$ and $\text{Conf}(X,Y) = \sup(X \quad Y)^{SW} / \sup(X)^{SW} \geq c$.

Definition 5: Given a transaction-sensitive sliding window *SW*, the form ({x, y} | M) is an indirect association rule in window *SW* if the following conditions are satisfied:

(1) $\sup(\{x,y\})^{SW} < t_s$;

(2) There exists a non-empty set M such that:

  (a) $\sup(\{x\} \quad M)^{SW} >= t_f$, $\sup(\{y\} \quad M)^{SW} >= t_f$,

  (b) $\text{dep}(\{x\},M)^{SW} >= t_d$, $\text{dep}(\{x\},M)^{SW} >= t_d$.

Where the thresholds t_s 、t_f and t_d are called itempair support threshold, mediator support threshold, and dependence threshold, respectively. We usually set $t_f >= t_s$ in practice.

IV . Mining both direct and indirect association rules

According to the definitions of direct association and indirect association rules in last section, we propose a algorithm to discover both direct and indirect association rules in data stream called *MDIAR-SW* (Mining Direct and Indirect 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 I, 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 Bit-sequences of items in window initialization phase of SW

Window-id	Transactions	Bit-Sequences of items
SW1	<*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 MDIAR-SW is described as follows:

Algorithm *MDIAR-SW*

Input: *TDS* (a transaction data stream), minimum support threshold: s;the minimum confidence threshold: c; itempair support threshold: ts ; mediator support threshold: t_f ; dependence threshold and t_d; the user-specified sliding window size *w*.

```
Output: Set of direct temporal association patterns: DAP;
        Set of indirect temporal association patterns: IAP;
Begin
  SW = Null; /* Window SW consists of w transactions */
  Repeat:
    for each incoming transaction $T_i$ in SW
      do
        if SW = Full then
            Do bitwise-shift on bit-sequences of all items in
            SW;
        else
            for each item X in $T_i$ do
                Do bit-sequence transform(X);
            Endfor
        Endif
      Endfor
    for each bit-sequence Bit(X) in SW do
      if sup(X) = 0 then
        Drop X from SW;
      Endif
    Endfor
```

```
/* The following is the frequent and infrequent itemsets
generation phase. */
  $F_1$ = {frequent 1-itemsets};
  for (k=2; FIk−1≠ Null; k++) do
      $C_k$ = Candiate_Gen($F_{k-1}$)
      Do bitwise AND to find the supports of $C_k$;
      $F_k$ = { $c_k$    $C_k$ | sup($c_k$)$^{SW}$ ≥ w    s }
      $S_k$ = $C_k$     $F_k$;
      for each S     $S_k$ do
          x = last_item(S); y = secondlast_item(S ); M = S – {x,y};
          if (sup(x, y) $^{SW}$ <$t_s$ AND dep({x},M) $^{SW}$ >=$t_d$ AND
            dep({x},M) $^{SW}$ >=$t_d$)
              IAP = IAP     {<{x, y}|M>};
      Endfor
      FreS = FreS     $F_k$;
  Endfor
  DAP = Construct(FreS);
End
```

The proposed MFI-TransSW algorithm consists of four phases, window initialization phase, window sliding phase, and frequent, infrequent itemsets generation phase and direct and indirect association rules generation phase. Since the approach to generate direct association rules is the same with well known algorithm Apriori, and the method to generate indirect association rules is straightforward, the description of direct and indirect 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 TABLE I, the first sliding window SW1 contains four transactions: Tid1, Tid2, Tid3 and Tid4. The bit-sequences of items of SW1 in the window initialization phase are shown in TABLE II.

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 sup(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 III .

TABLE III Bit-sequences of items in window sliding phase of SW2

Window-id	Transactions	Bit-Sequences of items
SW2	<*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, MDIAR-SW algorithm uses a level-wise method to generate the set of candidate itemsets Ck from the frequent itemsets Fk−1 according to the Apriori [1]. The step is called Candiate_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 Fk and Sk. The process is stopped until no new candidates are generated.

For instance, consider the bit-sequences of SW2 in Figure 3, and let the minimum support threshold s, itempair support threshold ts and mediator support threshold tf to be 0.5，05 and 0.6 respectively. Hence, an itemset X is frequent if sup(X)SW ≥ 0.5*4 = 2. In the following, we discuss the step of frequent and infrequent itemset mining of TransSW2.

Firstly, MDIAR-SW algorithm find out frequent 1-itemset F1={(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, (bd) and (be) are frequent, (de) is infrequent, because the Bit(bd) =1011, sup(bd) = 3, Bit(be)=0110, sup(be) = 2, Bit(de) =0010, sup(de) = 1. Secondly, MDIAR-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), (bde), generated by MDIAR-SW algorithm in TransSW2.

V . 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].

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.

The size of a sliding window w, the minimum support threshold s itempair support threshold ts , mediator support threshold tf, dependence threshold and td are set to 20,000, 0.1% ,0.1%, 0.2% and 50%, respectively. As shown in these experiments, the processing times of MDIAR-SW algorithm are shown in Figures 1 and 2.

Fig.1 shows the processing time of window initialization phase under different window sizes from 20,000 (200K) transactions to 100,000 (1,000K) transactions. Fig.2 shows the total time of window sliding time and pattern mining time at each 100K transactions using various window sizes from 200K transactions to 1000K transactions. As shown in Figures 1 and 2, MDIAR-SW algorithm.

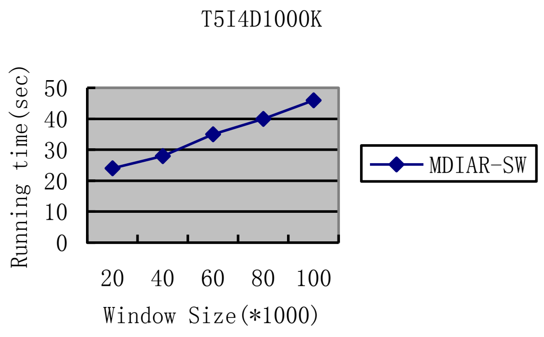

Fig.1 Running time in window initialization phases of algorithm MDIAR-SW under different window size.

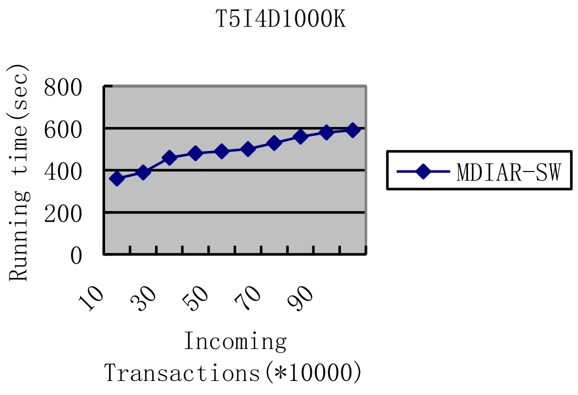

Fig.2 Running time including window sliding time and rule generation time of algorithm MDIAR-SW under different window size 200K transactions.

VI . Conclusions and Future Works

In this paper, we proposed an efficient one-pass algorithm, called MDIAR-SW, for mining direct and indirect 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] Agrawal R,Srikant R. Mining sequential patterns. In the Proc.1995 Int Conf. on Data Engineering, Taibei,Taiwan,March 1995,pp3-14.

[5] P.N.Tan and V.Kumar. Indirect Association: Mining Higher Order Dependences in Data. Proc. Of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases,pp632-737,Lyon,France(2000).

[6] Y. Chi, H. Wang, P. Yu, and R. Muntz. 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.

[7] C.H. Lin, D.Y. Chiu, Y.H. Wu and A.L.P. Chen. Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window. In Proceedings of 2005 SIAM International Conference on Data Mining, 2005.