SS-ZG548: ADVANCED DATA MINING

Lecture-08: Stream Mining

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems Engineering,
BITS Pilani, Rajasthan-333031 INDIA

Feb 04, 2018 (WILP @ BITS-Pilani Jan-Apr 2018)

Correction: [Lecture-3] Association Rules Generation

If $X \subset W$ & $support(X)/support(W) \geq C_{min}$, then $X \Rightarrow W - X$

Transactions
T_1=(A,B,C)
T_2=(A,F)
T_3=(A,B,C,E)
T_4=(A,B,D,F)
T_5=(C,F)
T_6=(A,B,C)
T_7=(A,B,C,E)
T_8=(C,D,E)
T_9=(B,D,E)

- Our frequent item contains
  - $\{A\}_6$ $\{B\}_6$ $\{C\}_6$ $\{E\}_6$ $\{A,B\}_5$ $\{A,C\}_4$ $\{B,C\}_4$ $\{A,B,C\}_4$
- Possibilities are
  $A \Rightarrow B$, $B \Rightarrow A$, $A \Rightarrow C$, $C \Rightarrow A$, , $B \Rightarrow C$, $C \Rightarrow B$, $A \Rightarrow \{B, C\}$, $B \Rightarrow \{A, C\}$, $C \Rightarrow \{B, A\}$, $\{A, B\} \Rightarrow C$, $\{A, C\} \Rightarrow B$, $\{B, C\} \Rightarrow A$
- Let's take $C_{min} = 1.22$
- Association rules that qualifies as valid rule are shown green
  $A \Rightarrow B$, $B \Rightarrow A$, $A \Rightarrow C$, $C \Rightarrow A$, $B \Rightarrow C$, $C \Rightarrow B$, $A \Rightarrow \{B, C\}$, $B \Rightarrow \{A, C\}$, $C \Rightarrow \{B, A\}$, $\{A, B\} \Rightarrow C$, $\{A, C\} \Rightarrow B$, $\{B, C\} \Rightarrow A$

Recap: Data streams

Consider, stream of data. Where data in arriving in rapid succession. Re-scan is NOT possible. Even storage space is insufficient to accommodate all data points.

Without storing all the data one wish to estimate

- Set of frequent items
- Number of distinct items
- Frequent itemsets
- *etc*

Example: Assume continuous update in model. I would tell you n-1 numbers from first n Natural numbers, without repetition, in an arbitrary order. Can you report the missing one?
[Constraints are on memory and processing]

Frequent items over data stream

- Let identity of items is drawn from the set $\{1, 2, 3, ..., n\}$.
- Frequency of item i be f_i
- Assume general arrival model, (i, v), $v > 0$ represents arrival and $v < 0$ is departure.
- Sum of frequencies $m = \sum_i f_i$ represent size of data stream
- Item i is **frequent** if $f_i > m/(k+1)$ for some fixed k.

Observations

- There could be at most k frequent items
  (why ? proof?) $m > k(k+1)$
- Any algorithm that finds all frequent and only frequent items requires at least $\log\binom{n}{k}$ bits
  (how? $2^s \geq \binom{n}{k}$)

Approx frequent items setting

Wish to output a list of items such that

- Every item in the list has frequency $f_i > (1-\epsilon)\frac{m}{k+1}$
- All the items having frequency at least $(1+\epsilon)\frac{m}{k+1}$ is in the list

Output should satisfy above two properties with probability $(1-\delta)$

Algorithm maintains a data structure A over the stream. Step to update an item x are explained below

1. **IF** (A.ismember(x)) A[x]$++$
2. **ELSE** A.insert(x)
3. **IF** (A.size == k+1) **THEN** $\forall\ y \in A$
4. A[y]$--$,
5. **IF** (A[y] ==0) A.delete(y);

In action: Frequent items

Take $k = 4$, and consider following data stream

5	8	4	5	4	12	$\rightarrow$	
	6	5	2	8	3	5	$\rightarrow$
		4	5	4	12	6	13

Insert x in data structure

1. **IF** (A.ismember(x)) A[x]$++$
2. **ELSE** A.insert(x)
3. **IF** (A.size == k+1) **THEN** $\forall\ y \in A$
4. A[y]$--$,
5. **IF** (A[y] ==0) A.delete(y);

Let us step by step execute the algorithm:

Index	Item	Frequency
1		
2		
3		
4		
5		

Count distinct over data streams (FM sketch)

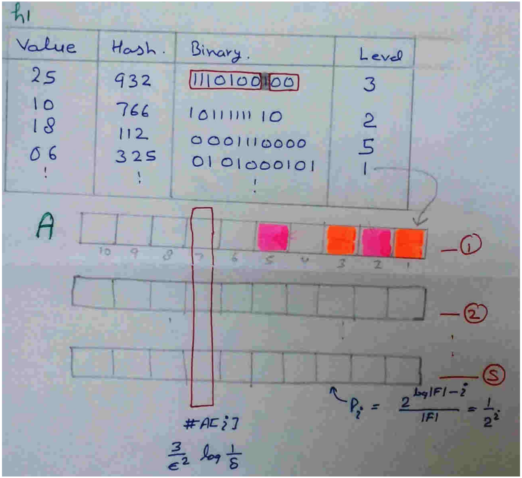
Estimate number of distinct items in data stream

- If $x = ???..??1\overbrace{000...0}^{i-1}$ then L[x]=i
- Probability of L[x]=i is $p_i = \frac{2^{\log|F|-i}}{|F|} = 1/2^i$ when $x \in \{1, 2, .., F\}$
- FM sketch is a bitmap A of size $\log|F|$ with hash a function h
- Arrival of an item x, sets bit $A[L[h(x)]] \leftarrow 1$.
  Probability that $A[i] = 1$ after seeing n items is $1 - (1 - p_i)^n$
- With s independent copies of FM sketch, let $\#A[i]$ represent count of 1's at level i and $\hat{q}_i = \frac{\#A[i]}{s}$. Then choose i, such that $\hat{q}_i \geq \frac{3}{\epsilon^2}\log\frac{1}{\delta}$. By Chernoff's bound $x \in [(1-\epsilon)E[x], (1+\epsilon)E[x]]$ with probability $(1-\delta)$

$$\hat{n} = \frac{\log(1-\hat{q}_i)}{\log(1-p_i)}$$

In action: Count distinct over data streams

Consider following data stream: 25, 10 ,18 ,25, 06, 03, 10, 8, 2, 5, 18, 12, 9, 6, 12, 6, 11, 15, 5, 6, 13, 6, 8, ... $\rightarrow$



$$\hat{n} = \frac{\log(1 - \hat{q}_i)}{\log(1 - p_i)}$$

Frequent pattern mining over data streams

- Applications involves retail market data analysis, network monitoring, web usage mining, and stock market prediction.
- Using sliding window
- Efficiently remove the obsolete, old stream data
- Compact Pattern Stream tree (CPS-tree) [1]
- Highly compact frequency-descending tree structure at runtime
- Efficient in terms of memory and time complexity
- Pane and window
- Insertion and restructuring

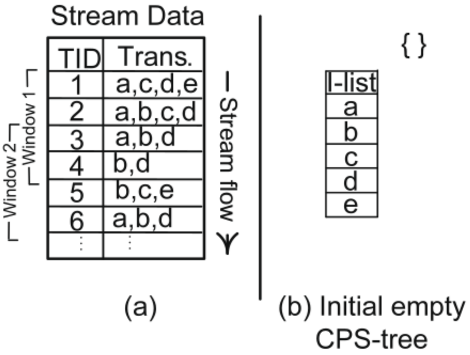
[1] Tanbeer, Syed Khairuzzaman and Ahmed, Chowdhury Farhan and Jeong, Byeong-Soo and Lee, Young-Koo, “Sliding window-based frequent pattern mining over data streams”, in Information sciences, 179(22) pages 3843–3865, Elsevier, 2009

CPS-tree construction

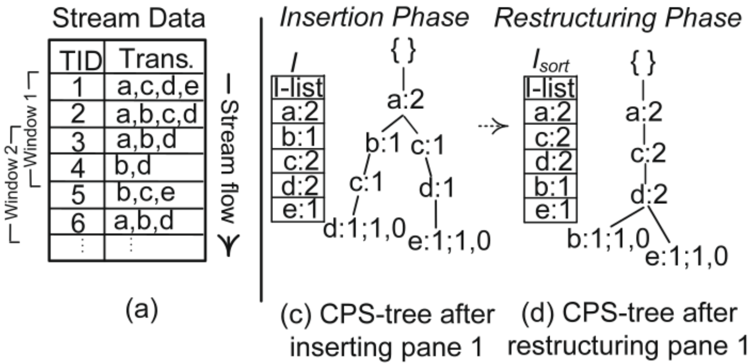
```
Algorithm 1 (Construction of a CPS-tree for a data stream)
Input: Stream_data, Pane_size, Window_size, Initial_Sort_Order
Output: T_sort: a CPS-tree for the current window
Method:
    Begin
        w ← φ;
        T ← a prefix-tree with null initialization;
        Current_Sort_Order ← Initial_Sort_Order;
    //For the first Window
            While (w ≠ Window_size) do
                Call Insert_Pane(T);                                  // Insertion Phase
                Current_Sort_Order ← Frequency-descending sort order; // Restructuring Phase
                Restructure T;
                w = w+1;
            End While
    //At each slide of Window
            Repeat
                Delete the oldest pane information from T;            // Extracting the old pane
                Call Insert_Pane(T);                                  // Insertion Phase
                Current_Sort_Order ← Frequency-descending sort order; // Restructuring Phase
                Restructure T;
            End
    End

Insert_Pane(Tr)
    Begin
        p ← φ;
        While (p ≠ Pane_size) do
            Scan transaction from the current location in Stream_data;
            Insert the scanned transaction into Tr according to Current_Sort_Order;
            p = p+1;
        End While
    End
```

Fig. 3. The CPS-tree construction algorithm.

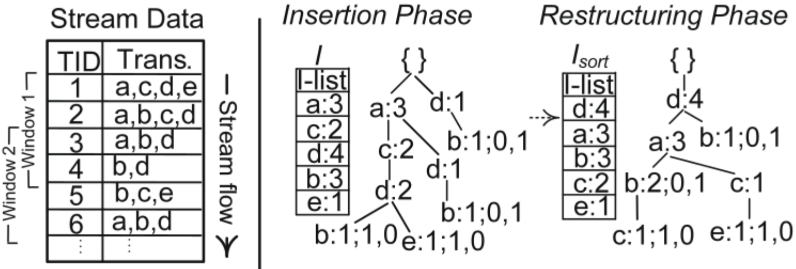
CPS-tree construction



CPS-tree construction



CPS-tree construction

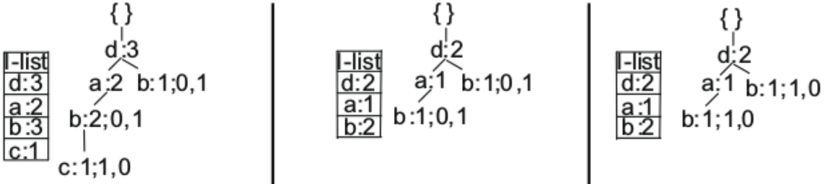


(a)

(e) CPS-tree after inserting pane 2

(f) CPS-tree after restructuring pane 1 & 2, i.e., at Window 1

CPS-tree construction

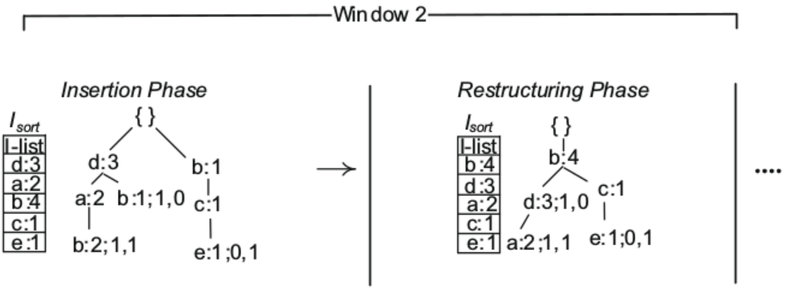


(a) After processing for '*e*'

(b) After processing for '*c*'

(c) After processing for '*b*', '*a*', and '*d*'

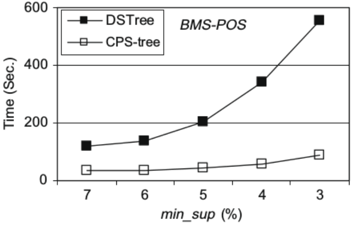
CPS-tree construction

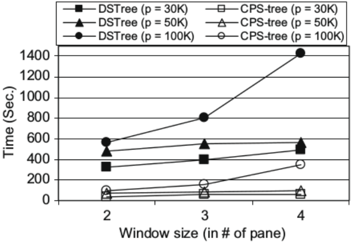


(a) CPS-tree after inserting new pane (i.e., *tids* 5 and 6) at Window 2

(b) CPS-tree after restructuring at Window 2

CPS-tree Performance





Thank You!

Thank you very much for your attention!

Queries ?