AN OVERVIEW OF ALGORITHMS USED FOR MINING FREQUENT PATTERNS IN DATA STREAMS

A.V.Srinivas

Assistant Professor (Sr.grade), Department of IT, Vardhaman College of Engg, Hyderabad, India-501218

Abstract—Data streams are an ordered sequence of items that arrives in timely order. It is impossible to store the data in which item arrives. To apply data mining algorithm directly to streams instead of storing them before in a database. Real time surveillances system, telecommunication system, sensor network, financial applications, transactional data are some of the examples of the data stream systems. These types of streams produced millions or billions of updates every hour. In this paper, we have studied the concept of data streams and how the frequent patterns are mined from data streams. We analyzed the existing algorithms used for mining frequent patterns in data streams.

Keywords—Data Streams, Frequent Patterns, Data Mining

I. INTRODUCTION

Recently, database and data mining communities have focused on a new data model, where data arrives in the form of continuous streams. It is often referred to data streams or streaming data. Many applications generate large amount of data streams in real time, such as sensor data generated from sensor networks, transaction flows in retail chains, Web record and click streams in Web applications, performance measurement in network monitoring and traffic management, call records in telecommunications, etc.

Mining such streaming data differs from traditional data mining in following aspects:

- Each data element in streaming data should be examined at most once.
- Memory usage for mining data streams should be bounded even though new data elements are continuously generated from the data stream.
- Each data element in data streams should be processed as fast as possible.
- The results generated by the online algorithms should be instantly available when user requested.
- The frequency errors of the outputs generated by the online algorithms should be constricted as small as possible.

Hence, the nature of streaming data makes it essential to use online algorithms which require only one scan over the data for knowledge discovery. It is not possible to store all the data in main memory or even in secondary storage. This motivates the design for in-memory summary data structure with small memory footprints that can support both one-time and continuous queries.

This paper will focus on the following sections.
Section 2 gives the overview of frequent pattern mining.
Section 3 gives the overview of various existing algorithms for mining frequent patterns in data streams by various papers.
Conclusion and future work of this paper are discussed in section 4.
II. FREQUENT PATTERN MINING

Frequent-pattern mining has been studied extensively in data mining, with many algorithms proposed and implemented (for example, Apriori [Agrawal & Srikant1994], FP-growth [Han, Pei, & Yin2000], CLOSEST [Pei, Han, & Mao2000], and CHARM [Zaki & Hsiao2002]). Frequent pattern mining and its associated methods have been popularly used in association rule mining [Agrawal & Srikant1994], sequential pattern mining [Agrawal & Srikant1995], structured pattern mining [Kuramochi & Karypis2001], iceberg cube computation [Beyer & Ramakrishnan1999], cube gradient analysis [Imielinski, Khachiyan, & Abdulghani2002], associative classification [Liu, Hsu, & Ma1998], frequent pattern-based clustering [Wang et al.2002], and so on.

Recent emerging applications, such as network traffic analysis, Web click stream mining, power consumption measurement, sensor network data analysis, and dynamic tracing of stock fluctuation, call for study of a new kind of data, called stream data, where data takes the form of continuous, potentially infinite data streams, as opposed to finite, statically stored data sets. Stream data management systems and continuous stream query processors are under popular investigation and development. Besides querying data streams, another important task is to mine data streams for interesting patterns.

III. OVER VIEW OF ALGORITHMS

In 2008, Syed Khairuzzaman Tanbeer, Choudary, Farhan Ahmad, Byeong-Soo Jeong, Young-Koo Lee had proposed a prefix-tree structure called CPS-tree (Compact Pattern Stream tree) in their paper “Efficient frequent pattern mining over data streams”. The Compact Pattern Stream tree uses dynamic tree restructuring technique for handling data streams. With Single pass scanning this technique constructs a compact-pattern tree. For each window we restructuring the Compact Pattern Stream tree (CPS-tree). For restructuring, they proposed one efficient restructuring method called Branch Sorting Method by using the existing method called Path Adjusting Method.

Path Adjusting Method Algorithm

Let X, Y and Z be three nodes in a path in a prefix-tree, where X is the parent of Y, Y is the same of Z and nodes Y and Z are required to be exchanged to adjust the path. Consider node_name.name, node_name.count and node_name.child refers to the name, the support count (in the referred path) and a child of a node. Therefore, the path adjusting is performed according to the following algorithm.

1. If(Y.count==Z.count) goto step 5

   Insertion:

   2. Insert Y’ (such that Y’.name=Y.name) to X as a new child node and
      Set Y’.count=Y.count-Z.count
   3. Assign all children of Y except Z to Y’
   4. Set Y.count=Z.count

   Exchange:

   5. Exchange parent and children links of Y and Z

   Merge:

   6. If C is another child node of X such that C.name=Z.name then Merge_Node(C,Z)
   7. Delete C and its sub-tree
   8. Repeat with next two nodes of Y and Z in another path to be exchanged and terminate when no further node exchange is required.
   9. Merge_Node(P,Q){
10. Set Q.count=Q.count+P.count
11. For each child of Q
12. For each child of P
13. If(Q.child==P.child)
14. Merge_Node(Q.child, P.child)
15. Else add P.child and its sub-tree to Q children list
16. }

Branch Sorting Method Algorithm

**Input:** CPS tree (T) and Item set (I)

**Output:** T_{sort} and I_{sort}

1. Compute I_{sort} from I in frequency-descending order using merge sort technique
2. For each branch B_i in T
3. For each unprocessed path P_j in B_i
4. If P_j is a sorted path
5. Process_Branch(P_j)
6. Else Sort_Path(P_j)
7. Terminate when all the branches are sorted and output T_{sort} and I_{sort}
8. **Process_Branch(P){**
9. For each branching node n_b in P from the leaf_p node
10. For each sub-path from n_b to leaf_k with k≠p
11. If item ranks of all nodes between n_b and leaf_k are greater than that of n_b
12. P=sub-path from n_b to leaf_k
13. If P is a sorted path
14. Process_Branch(P)
15. Else P=path from the root to leaf_k
16. Sort_Path(P)
17. **}**
18. **Sort_Path(Q){**
19. Reduce the count of all nodes of Q by the value of leaf_Q count
20. Using merge sort, sort Q in an array according to I_{sort} order
21. Delete all nodes having count zero from Q
22. Insert sorted Q into T at the location from where it was taken
23. **}**

**Merits and Demerits:**

The main advantage of CPS-tree structure is, by dynamically applying a tree restructuring technique achieves a frequency-descending prefix-tree structure with a single-pass and considerably reduces the mining time to discover frequent patterns from a dataset.

The main disadvantage of this algorithm is every time a new item is arrives, it reconstructs the tree. So it causes more memory space as well as time.

In the year 2009 Pauray S.M. Tsai proposed a new technique called Weighted Sliding Window (WSW) algorithm for mining frequent item sets in data streams using the weighted sliding window model. For this algorithm, the user has to supply the number of windows, the minimum support, and the size of a window and the weight of a window. Using this algorithm, calculate the weighted support count of each item in each window. If the weighted support count of an item set is greater
than or equal to the minimum weighted support count, it is called a frequent item set. This algorithm can generate candidate item set also. After generating candidate item set, we can determine whether it is frequent item set or not by using this algorithm.

**Weighted Sliding Window (WSW) Algorithm:**

Assume the number of windows is $n$, the size of a window is $t$, the current time point is $T_i$, and $x_1,x_2,...,x_k$ are items. The transactions considered are those within the time range of $T_i$ and $(T_i - nX_t)$. Let $SPij({x_1,x_2,...,x_k})$ be the set of identifiers of transactions containing itemset $\{x_1,x_2,...,x_k\}$ in window $Wij$ ($1 \leq j \leq n$) and $a_j$ the weight of $Wij$. $|SPij({x_1,x_2,...,x_k})|$ represents the number of transaction identifiers in $SPij({x_1,x_2,...,x_k})$.

**Input:**
- the number of windows: $n$
- The minimum support: $S$
- The size of a window: $t$
- The weight of window $Wij(1 \leq j \leq n)$: $a_j$

**Step 1:**
Assume the current time point is $T_i$. $i=1$;

**Scan window $Wij(1 \leq j \leq n)$ once and evaluate $SPij({x})$ for each item $x$.**

**Step 2:**
Assume the number of transactions contained in $Wij$ is $Nij$. The minimum weighted support count is $S x \sum_{j=1}^{n} (a_j x N_{ij})$

**Step 3:**
$L1=\{\{x\} | \text{the weighted support count of item } x \geq \text{the minimum weighted support count}\}$

**Step 4:**
for $(k=2;|Lk-1| > 1;k++)$ do begin

**Step 5:**
Generate candidate $k$-itemset $Ck$ by $Lk-1$

**Step 6:**
for each candidate itemset $c \in Ck$
do begin

**Step 7:**
Assume $c$ is generated by $Xp$ and $Xq$.

$SPij(c)=SPij(Xp) \cap SPij(Xq)$

$(1 \leq j \leq n)$

Step 8: If the weighted support count of $c \geq \text{the minimum weighted support count}$

Step 9: then $Lk= Lk \cup \{c\}$

Step 10: end

Step 11: end

Step 12: $i=i+1$ ; $T_i = T_{i-1} + t$

Step 13: for $(j=1;j\leq n-1;j++)$ do begin

$SPi(j+1)(\{x\})=SP(i-1)j(\{x\})$ for each item $x$

end

**Step 14:**
Scan $W1$ once.
Evaluate $SP1(\{x\})$ for each item $x$.

**Step 15:** Go to Step 2

**Merits and Demerits:**

The main advantage of this algorithm is, it scans the database only once to find out the frequent item sets. The main disadvantage of this algorithm is, it may take more time and memory for generating candidate item set.

In the year 2010, Yo Unghee Kim, Won young Kim and Ungmo Kim had proposed an efficient algorithm “Weighted Support Frequent Item sets mining (WSFI mine)” in their research paper “Mining frequent item sets with normalized weight in continuous data streams”.
The algorithm is described as follows:
Initially, the WSFI-Mine reads a stream database TDS from the current window. Then, the WSFI-Mine processes the weight of each item and sorts the weighted support item sets in each sliding window into descending order. Next, a WSFP-Tree is constructed from the descending order list. Finally, frequent item sets mining is usually performed.

Input: (1) A Stream Database (TDS)  
(2) Normalized minimum weighted support ($\phi$)  
(3) Normalized minimum weighted error support ($\epsilon$)  
(4) Weighted Range  
Output: WSFP-Tree, A set of weighted support frequent item sets.

1. Scan a Stream Database and count support for each item.  
2. Multiply item support by the weight of each item.  
3. Sort them into a descending order list in a sliding window.  
4. Create the root of a WSFP-Tree. Next, each transaction performs as follows:  
   4-1. Select the descending order frequent item and call:  
      insert_wsfp_tree (dsitem_list, T).  
   4-2. The function insert_wsfp_tree (dsitem_list, T) is performed as follows:  
      (1) If T has a child node such that node.item = dsitem_list.item then increment the node’s count by 1 or create a new node with its count initialized to 1.  
      (2) Link its parent to T and link its node-link to the nodes with the same item name via the node-link.  
      (3) If $\epsilon \leq$ weighted support of node.item $\leq \phi$ then do not remove it from WSFP-Tree (latent pattern) or If the weighted support of node.item $\leq \epsilon$ then remove it from the WSFP-Tree in the next phase (infrequent pattern).  
5. The construction process of WSFP-Tree with respect to previous sliding window tree result is the same as in step 4 recursively.  
6. End.

**Merits and Demerits:**

WSFI-Mine algorithm can mine all frequent item sets in one scan from the data stream. The WSFI-Mine algorithm effectively executes frequent by generating constraint candidate item sets. This is the main advantage of this algorithm. At the time of pruning there are lot of problems arises. This is the disadvantage of this algorithm.

In the year 2011 Jing Guo, Peng Zhang, Jianlong Tan and Li Guo had proposed a new algorithm H-STREAM to mine frequent patterns across multiple data streams in their research paper “Mining frequent patterns across multiple data streams”. This algorithm built a new hybrid frequent tree to maintain historical frequent and potential frequent item sets for efficient collaborative and comparative pattern mining.

**H-stream algorithm:**

H-Stream algorithm mainly consists of three steps. In the first step, it builds an H-tree structure to maintain the frequent and potential frequent item sets from all streams. In the second step, H-Stream continuously updates H-tree by inserting new emerging frequent and potential frequent item sets,
meanwhile deleting item sets that have become infrequent. In the last step, H-Stream traverses the
tree to retrieve collaborative and comparative frequent patterns.

Merits and Demerits:
Using this algorithm, the user can easily retrieve the comparative and collaborative frequent patterns
from multiple data streams.

In 2012, Yong-gong Ren proposed an algorithm in order to predict the future data based on the new
method called AMFP-Stream (Associated Matrix Frequent Pattern-Stream), it predicts the frequently
occurred item sets over data streams efficiently. It also has a capability to predict that which item set
will be frequent with high potential. It takes the data in the form of 0-1 matrix and then it updates the
values by doing logical bit operations. Based on this it will find out the item sets that will frequently
occur in the future. It uses the associated matrix for the further manipulation.
This algorithm involves four different algorithms.
The first algorithm discuss about the generation of the associated matrix by giving the transaction
dataset as an input.

First Algorithm:

Step-1: Store the incoming data in a separate memory;
Step-2: Initialize the Mij to be 0;
Step-3: Evaluate each value;
Step-4: Loop verify till the last element
Step-5: If item sets are repeated then
Step-6: Record the statistical information;
Step-7: Store the support number;
Step-8: End If.
Step-9: End loop.
The second algorithm involves in updating the associated matrix values with frequent item sets. It removes the item sets that are infrequently occurred.

**Second algorithm:**

Step-1: Consider all the items.
Step-2: Check for the minimum support
Step-3: If the transaction value not repeated then
Step-4: Reduce the frequency of that item.
Step-5: Delete the items which are infrequent.

The third algorithm takes the last updated associated matrix as an input and produces the frequent item sets as the result.

**Third Algorithm:**

Step-1: Consider all the elements from the associated matrix.
Step-2: If the occurrences is 0 then do nothing
Step-3: Else item sets and supporting data are determined.
Step-4: End if
Step-5: If more items then continue from step-1 again.
Step-6: Else exit the routine after updating the frequent item sets.

The last fourth algorithm predicts the item sets that will occur frequently occur in the future. It takes the current sliding window as an input and produces the next sliding with maximum possible frequent itemsets.

**Fourth Algorithm:**

Step-1: Consider the data of the current sliding window.
Step-2: Consider the already exiting frequently occurred itemsets.
Step-3: Find out the common itemsets.
Step-4: This results in the itemsets frequency measure in upcoming sliding window.

**Merits and Demerits:**

This algorithm fails to explain about the space and time complexity even though it poses full accuracy in the frequent itemsets.

**IV. CONCLUSION**

In this paper, we provide a survey of research on mining data streams. We focus on frequent item set mining and has tried to cover both early and recent literature related to mining Frequent Item sets. Moreover, we have addressed the merits and demerits and presented an overall analysis of the algorithms, which can provide insights for end-users in applying or developing an appropriate algorithm for different streaming environments and various applications.

More high-speed data streams are generated in different application domains, such as millions of transactions generated from retail chains, millions of calls from telecommunication companies, millions of ATM and credit card operations processed by large banks, and millions of hits logged by popular Web sites. Mining techniques will then be very significant in order to conduct advanced
analysis, such as determining trends and finding interesting patterns, on streaming data. It is our intention to present this survey to simulate interests in utilizing and developing the previous studies into emerging applications.

REFERENCES


[2] “Data Mining: Introductory and Advanced Topics” Margaret H. Dunham


[5] “Mining frequent item sets over data streams using efficient window sliding technique” Hue-Fu Li, Suh-Li, 2009, Elsevier publication.


