Compact Tree for Associative Classification of Data Stream Mining

K.Prasanna Lakshmi [1], Dr.C.R.K.Reddy[2]

[1] Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Andhra Pradesh 500 090, India

[2] Department of Computer Science, Chaitanya Bharathi Institute of Technology Hyderabad, Andhra Pradesh 500 075, India

Abstract

The *data streams* have recently emerged to address the problems of continuous data. Mining with data streams is the process of extracting knowledge structures from continuous, rapid data records [1]. An important goal in data stream mining is generation of compact representation of data. This helps in reducing time and space needed for further decision making process.
In this paper we propose a new scheme called Prefix Stream Tree (PST) for associative classification. This helps in compact storage of data streams. This PSTree is generated in a single scan. This tree efficiently discovers the exact set of patterns from data streams using sliding window.

Keywords: *Data Streams, Data Stream Mining, Association, Classification.*

1. Introduction

In recent years, data streams have become ubiquitous because of the large number of applications which generate huge volumes of data in an automated way [2]. Many existing data mining methods cannot be applied directly on data streams because of the fact that, the data needs to be mined in single scan. Data streams have following characteristics.

- The data arrives continuously.
- No assumptions on data stream ordering can be made.
- The memory usage during the mining process should be limited.
- Knowledge must be gained as quickly as possible.
- Each data element should be examined at most once and processed as fast as possible because of memory limitation.

Efficiently and effectively capturing knowledge from data streams has become very critical with wide spread of application areas likes network traffic monitoring, web click-stream analysis, market basket data mining, fraud detection etc.,

Therefore, the processing of data streams needs

i) To examine each data element of data stream at most one time as it is unrealistic to store data streams in main memory.
ii) Each element of it must be processed as fast as possible by consuming minimum memory.
iii) Finally, results generated should be available when ever user requests for them.

2. Preliminaries

In this section we will be discussing about data stream representation, various window models, association, classification, associative classification.

2.1 Data Stream

Let $\sum = \{x1,\ x2, \ldots ., \ xn\}$ be a set of items. A transaction $T = (TID, x1, x2, x3, \ldots xn)$, $xi \in \sum$, for $1 \leq i \leq n$, is a set of items, while n is called the size of the transaction, and TID is the unique identifier of the transaction. An itemset is a non-empty set of items. An itemset with size *m* is called an m-itemset. A data stream $D = \{B1,\ B2, \ldots \ldots,\ BN\}$ is an infinite sequence of batches where each batch B_i contains a set of transactions i.e. $Bi = \{T1, T2, \ldots .. Tk\}$ where

$k > 0$. Frequency of itemset X denoted by *freq(X)* is number of transactions that support X in a batch B.
Support of X denoted by *supp(X)* is *freq(X)/N*, where N is total number of transactions. X is called as frequent itemset if *supp(X) >=minSupp*, where *minSupp* is user defined minimum threshold support.

2.2 Windowing Model

According to the stream data processing model, the research of mining in data streams can be divided into three categories:

i) Landmark-window based mining [3]
ii) Damped-window based mining[4]
iii) Sliding-window based mining [5] as shown in *Fig.1*.

A window is a subsequence between *i-th* and *j-th* arrived transactions, denoted as

$$W[i,\ j\ ] = (ti,\ ti+1,\dots,\ tj),\ i \le j\ .$$

For the window-based approach, we can come up with two naive methods [6]:
1. Regenerate frequent itemsets from the entire window whenever a new transaction comes into or an old transaction leaves the window.
2. Store every itemset, frequent or not, in a traditional data structure such as the *prefix tree*, and update its support whenever a new transaction comes into or an old transaction leaves the window.

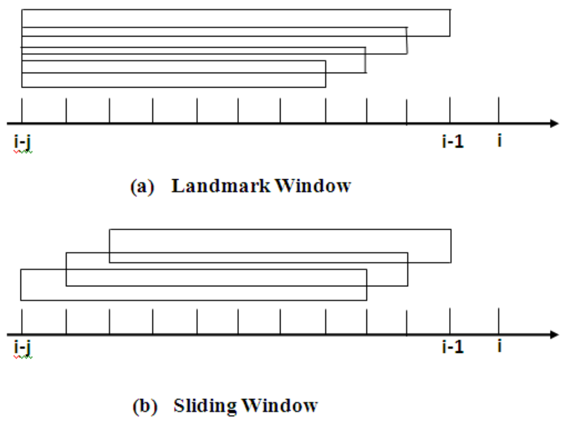


Fig.1. Various windows used for data streams

Method 2 is an incremental method, so in this paper, we focus on sliding window which is incremental in nature.

2.3 Association

Association deals with finding frequent patterns from unstructured, semi structured and structured datasets. It also helps in finding the relationships between itemsets. Patterns can be itemsets, sequences, sub trees and sub graphs depending on mining tasks.
The FP-growth mining technique [7] is one of the efficient mechanisms where the performance gain is based on highly compact FP tree structure. This tree is constructed by having two database scans and on prior threshold knowledge which restrict the usage of FP-tree on data streams. So in this paper we use a novel tree structure that constructs an FP–tree like compact prefix tree structure within a single pass.
A *Prefix-tree* [8] is an ordered tree which represents the transactions of the streams in a highly compressed form. Each read transaction is inserted into the tree in a path. Since different transactions can have several common items, their paths in the tree will be overlapped. The more the paths overlap with one another, the more compact tree is achieved.
For the association rule $(A \Rightarrow B)$ generation support, confidence and lift are used.

$$Support(A \Rightarrow B) = P(A \cup B).$$

$$Confidence(A \Rightarrow B) = P(B \mid A) = Support(A \cup B) / Support(A).$$

$$Lift(A \Rightarrow B) = Support(A \cup B) / Support(A) \times Support(B).$$

2.4 Classification

Classification techniques [9, 10] have attracted the attention of researchers due to significance of their applications in domains like statistics, pattern recognition, machine learning and data mining etc. The applications of data stream classification can vary from critical geophysical applications [11] to real–time decision support and industrial applications [12, 13]. There are several potential applications which do work on classification.

2.5 Using Association for Classification

Different from traditional algorithms, associative classification tries to find meaningful information to meet the needs of user association rules from data. In recent years, Classification of association rules are applied to obtain good results [14]. Classifying a data stream with an association classifier is a newly explored area of research. The Methodology is illustrated in the Fig.2. We present an approach to construct a compact tree for mining class association rules.

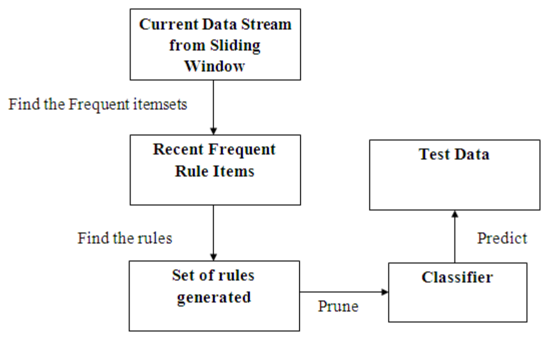

Current Data Stream from Sliding Window
Find the Frequent itemsets
Recent Frequent Rule Items
Find the rules
Set of rules generated
Prune
Classifier
Predict
Test Data

Fig.2. Mechanism of Associative Classification

3. Problem Statement

In data stream mining it is unrealistic to store all the data in limited memory or even in secondary storage. Processing recent data to mine complete set of exact frequent itemsets from data streams is another challenge. Efficiently updating old data with new data helps in reducing the memory usage which indirectly helps in increasing the performance. With a single scan of data a compact data structure must be built. This compact structure will help in having memory and time efficient mining. For processing recent data the compact tree constructed must be restructured. Motivated by these requirements, in this paper, we propose a memory-efficient summary data structure called PSTree which efficiently addresses all the above.

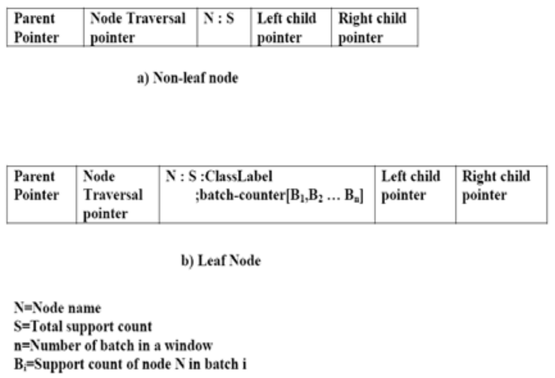
4. Construction of Compact tree

The PSTree, which we propose is based on prefix tree schema. It is an abstract and compact representation of data streams. For capturing the recent data stream contents the tree uses sliding window. As the window *W* slides the tree is updated. Each time the window *W* contains equal number of batches of transactions. Window slides batch-by-batch.

4.1 Structure of the Prefix Stream tree

Before discussing the tree construction, we briefly describe its structure. The PSTree is made up of nodes. First node of the tree is called root node which is referred as "*null*". Each subsequent node is called as ordinary node which represents the itemset and total number of passes (i.e., support) for that itemset in the path from the root up to that node in the current window. The end nodes of the tree are leaf nodes which contains the support, class label and batch counter.

Hence, two types of nodes are maintained in the tree. The structure of non-leaf and leaf nodes is shown in Fig.3.



Parent Pointer	Node Traversal pointer	N : S	Left child pointer	Right child pointer

a) Non-leaf node

Parent Pointer	Node Traversal pointer	N : S :ClassLabel ;batch-counter[B_1,B_2 … B_n]	Left child pointer	Right child pointer

b) Leaf Node

N=Node name
S=Total support count
n=Number of batch in a window
B_i=Support count of node N in batch i

Fig.3. Nodes of PSTree basic structure

A list called itemset list *I-list* is maintained along with tree which contains the support counts of all items and the items of tree.

4.2 Phases in construction of the tree

Construction of tree is done using two phases: *Insertion* and *Restructuring*. Insertion phase captures the stream contents into tree based on sorted order *I-list*.
Restructuring phase restructures the tree in descending order of frequency from *I-list*. Restructuring is done after inserting a batch of transaction using Insertion phase. These two phases are dynamically executed one after the other. We use an example to illustrate the tree construction in section 4.4.

4.3 Time Complexities

Complexity for construction of the tree using initial Insertion phase is always *O(nm)* where *n* is the number of transactions in a batch and *m* is the number of items in a transaction.
Irrespective of data distribution the complexity of restructuring is always $O(nlog_2n)$, where *n* is the total number of items in the list. In the worst case complexity of restructuring would be $O(mnlog_2n)$, where *m* is the number of paths in the tree and *n* is average length of transactions.

4.4 Construction of tree

Consider the data streams shown in Table 1 which contain three attributes, A_1 (a_1, b_1, c_1), A_2 (a_2, b_2, c_2) and A_3 (a_3, b_3, c_3) and two classes (y_1, y_2). Assuming *minsupp* = 20% and *minconf* = 80%. Taking the help of sliding window for two batches where each batch contains two tuples, the PSTree is constructed using the concept of prefix tree. Initially a batch of two tuples is inserted using Insertion phase along with maintenance of itemset list *I-list*. Restructuring the tree for Batch-1 according to sorted *I-list* is the next step as shown in Fig.4 and Fig.5. These insertion and restructuring phases are repeated one after the other for all consecutive batches. If all batches B_{j-1}, B_j in the current window W_i are inserted properly into the tree then the window will slide to the next batches B_j, B_{j+1}. While inserting the new batch B_{j+1}, the oldest batch B_j is deleted by changing the batch number as in Fig.6.

Table 1. Training Data Set

Tuple Id	A_1	A_2	A_3	Class
1	a_1	a_2	b_3	y_1
2	a_1	a_2	c_3	y_2
3	a_1	b_2	b_3	y_1
4	a_1	b_2	b_3	y_1
5	b_1	b_2	a_3	y_2
6	b_1	a_2	b_3	y_1
7	a_1	b_2	b_3	y_1
8	a_1	a_2	b_3	y_1
9	c_1	c_2	c_3	y_2
10	a_1	a_2	b_3	y_1
.				
.				

Window 1 Window 2 Stream Data

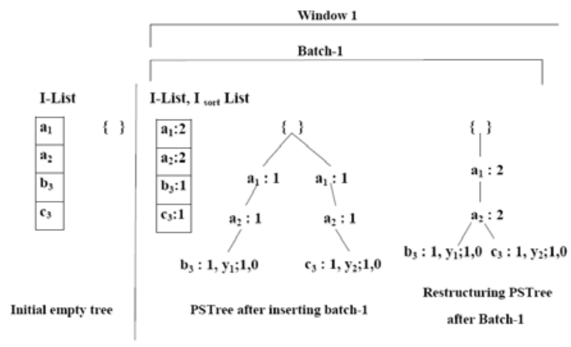
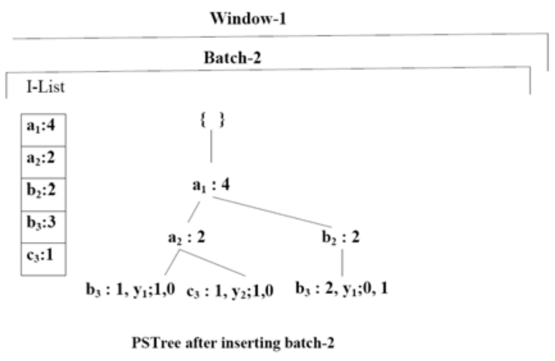


Fig.4. Insertion and Restructuring Phase for Batch-1



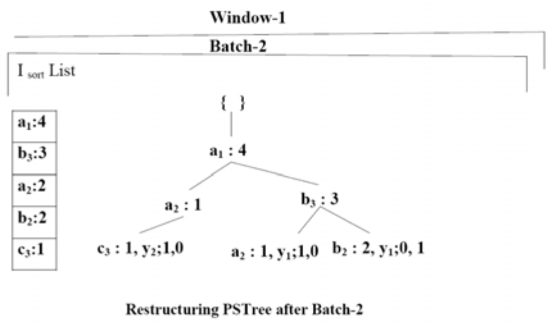


Fig.5. Insertion and Restructuring Phase for Batch-2

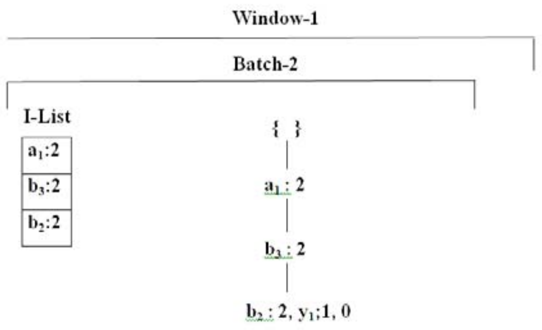


Fig.6. Methodology for extracting old Batch details

The tree is refreshed all the time as the window slides so that ready-to-mine platform with the exact information about frequent itemsets along with rules is provided for the current window. In cases where a rule item is associated with multiple classes, only the class with the largest frequency is considered.
Restructuring of the tree can be done using either Path Adjusting method or Branch Sorting method proposed by [15] [16].
When all the frequent itemsets are obtained, using bottom-up FP-Growth mining technique, confidence of the various rules is calculated and is sorted in the memory. For every request for classification the classifier will predict the class label of the tuple from the recent frequent itemsets.

5. Conclusions

In this paper we introduced a concept for classification based on association. We used a PSTree which was constructed using the concept of prefix tree and was restructured to handle the stream data. The constructed tree is a compact tree which reduces the memory consumption. It helps in finding exact set of recent frequent itemsets and predicts the class label for the requested tuple.

References

[1] K.Prasanna Lakshmi, Dr.C.R.K.Reddy, “A Survey on Different Trends in Data Streams “ pp.451-455, In Proc of 2010 IEEE International Conference on Networking and Information Technology, (ICNIT’10), 2010. ISBN : 978-1-4244-7577-3.

[2] Hua-Fu Li, Suh-Yin Lee, “Mining frequent itemsets over data streams using efficient window sliding techeniques”, Expert System with Applications, vol.36, no.2, pp. 1466-1477, 2009.

[3] Manku, G.S., & Motwani, R. (2002). Approximate frequency counts over data streams. In Proceedings of the 28th international conference on very large data bases, (pp. 346–357).

[4] J. H. Chang and W. S. Lee. Finding Recent Frequent Itemsets Adaptively over Online Data Streams. In Proc. of KDD, 2003.

[5] J. H. Chang and W. S. Lee. A Sliding Window method for Finding Recently Frequent Itemsets over Online Data Streams. In Journal of Information Science and Engineering, Vol. 20, No. 4, July, 2004.

[6]Yun Chi, Haixun Wang, Philip S. Yu, Richard R. Muntz, "Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window," icdm, pp.59-66, Fourth IEEE International Conference on Data Mining (ICDM'04), 2004

[7]Han, J., Pei, J., and Yin Y. 2000. Mining frequent patterns without candidate generation. In Proc. of Int. Conf. on Management of Data. 1-12.

[8] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing*, 61(3):350–371, 2001.

[9]Hand D. J. (1999) Statistics and Data Mining: Intersecting Disciplines ACM SIGKDD Explorations, 1, 1, pp. 16- 19.

[10]Hand D.J., Mannila H., and Smyth P. (2001) Principles of data mining, MIT Press.

[11] Burl M., Fowlkes C., Roden J., Stechert A., and Mukhtar S. (1999), Diamond Eye: A distributed architecture for image data mining, in *SPIE DMKD, Orlando.*

[12] Kargupta, H., Park, B., Pittie, S., Liu, L., Kushraj, D. and Sarkar, K. (2002). MobiMine: Monitoring the Stock Market from a PDA. ACM SIGKDD Explorations, Volume 3, Issue 2. Pages 37-46. ACM Press.

[13] Kargupta H., Bhargava R., Liu K., Powers M., Blair S., Bushra S., Dull J., Sarkar K., Klein M., Vasa M., and Handy D. (2004) VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. Proceedings of SIAMInternational Conference on Data Mining.

[14] Li Su, Hong-yan Liu, Zhen-Hui Song : A New Classification Algorithm for Data Streams, I.J. Modern Education and Computer Science 2011, 4, 32-39.

[15]Koh, J.-L., and Shieh, S.-F. 2004. An efficient approach for maintaining association rules based on adjusting FP-tree structures. In Lee Y-J, Li J, Whang K-Y, Lee D (eds) Proc. of DASFAA 2004. Springer-Verlag, Berlin Heidelberg New York, 417–424.

[16]Tanbeer, S. K., Ahmed, C. F., Jeong, B.-S., and Lee, Y.-K. 2008. CP-tree: a tree structure for single-pass frequent pattern mining. In Proc. of PAKDD, Lect Notes Artif Int, 1022-1027.

K.Prasanna Lakshmi received the Bachelor’s Degree in mechanical engineering(2000) and the Master’s Degree in computer science(2002). She is pursuing her part time Ph.D in computer science. She is currently working as Assoicate Professor in Department of Information Technology in Gokaraju Rangaraju Institute of Engineering and Technology. Her research interests include data mining, data stream mining, web mining.

Dr.C.R.K.Reddy received his doctorate from Central University, India in computer science. He is currently working as Professor in Department of Computer Science in Chaitanya Bharathi Institute of Technology. His research interests include Software Engineering, Data mining, Image processing.