Efficient Classifier over Stream Sliding Window using Associative Classification

K. Prasanna Lakshmi¹, Dr. C. Ramesh Kumar Reddy²

¹Department of Information Technology, GRIET, Hyderabad, Telangana, India
²Department of Computer Science, CBIT, Hyderabad, Telangana, India

Abstract- Prominence of data streams has dragged the interest of many researchers in the recent past. Research is going in the direction of formulating association rules on data streams for the purpose of prediction. From among the classification techniques the associative classification mining stands out with better performance over former classification techniques. A new technique is introduced through this paper, which takes the advantage of the associative classification for mining of data streams. To the best of our knowledge there are only a few techniques existing in the domain of data streams. We have designed a compact data structure to maintain data stream efficiently without losing important information. We present a PSToS for mining rules from the tree. Subsequently, an optimized algorithm called PSToSWMine is proposed for mining a classifier which contains set of high qualified classification rules. We have conducted experiments both on synthetic and real data sets for the purpose of assessing the performance of our approach. The results that we arrived at prove that our approach is superior to existing algorithms in terms of accuracy of prediction and run time efficiency.

Keywords: Data Streams, Associative Classification, Frequent Item sets, Prediction

1 Introduction

Data stream mining deals with gaining knowledge from the stream of data. Most of the recent applications involve processing of large volumes of data, flowing in continuously [11]. To take a few examples, web click streams, financial transaction, science surveillance data etc., Given the nature of data, mining of these data streams necessitate a real time response after analysis. This also means that the technique applied should be capable enough to process the data quickly as it should be read in a single pass and produce the results [11]. Sequential access methods for stream mining are cost effective and better than random access methods. As mentioned earlier, stream mining is applicable for applications of large data sets, this would be impractical to store on main memory and hence stored on secondary storage devices. Data sets such as sensor data, router packet statistics are temporal and need not be stored in disk; these must be processed and discarded. And as the size of these data sets increases far beyond the space available to an algorithm, it is not possible for the streaming algorithm to remember too much of data scanned in the past. In order to mitigate the challenges posed by the situation mentioned above, there is a need to design algorithms that store summary of past data, so as to make memory available for processing future data. The quintessence of the algorithm for data streams would be to examine each data element at most once in least possible time and occupy minimum memory space for storage.

Association and Classification are two useful and ubiquitous tools in data analysis. Mining based on association is concerned with extracting correlated features shared among transactions of data streams. These algorithms give the statistical relationship between items without giving significance of items [4]. On the other hand, classification uses class attribute in construction of classifier. The classifier needs the significance of items for predicting the class label. Integration of these two methods will provide efficient associative classifier [13], [1]. We study the associative classification in the stream context and provide a streaming algorithm with performance guarantees. Associative classifier predicts class from rules generated using association for unseen stream of data. Compared with existing classification techniques, classification based on association gives more accurate results due to better classifier. Rules containing class information are stored in classifier. These rules are generated from frequent pattern mining concept of association. So, frequent pattern mining plays an important role in associative classification. Many frequent pattern mining techniques exist currently and many more efficient techniques will evolve in future as these have direct impact on performance of associative classification.

Moreover, classifying data streams using this technique is a newly explored area of research [12]. Due to inimitable features of streaming data, it is not possible to simply apply the algorithms designed for static datasets to data streams. Challenges posed on associative classification of data streams include working with limited memory, processing
data at a glance, concept drift and improving accuracy of classification. Many researchers have devoted their efforts to frequent item set mining on data streams as this method is used for feature selection and classifier construction. Time windows are commonly used for handling data streams [6], [2], [7]. Based on application, landmark, sliding and damped windows can be used. A landmark window is divided into many windows and the data in these windows is used as updating units. As the name suggests, in sliding window only a fixed number of data elements present in recent window can be used for mining. In applications where all historical data is needed with more weight age on recent data than older data then damped windows are preferred.

Algorithm for mining classifier containing associative rules over sliding window for data streams will be very useful for classifying unseen data. In this paper, we propose a new algorithm PSToSWMine, for associative classification mining over data streams from sliding window. A new storage structure called PSTree [10] which is already proposed by us is adopted. It dynamically restructures to reflect the growth of item sets frequencies over time. Intensive study shows that our proposal is efficient and attains high classification accuracy.

The work we carried out is briefed out below:

- We created a compact data structure called PSTree [10] to maintain the relevant and current information.
- We devise an algorithm for mining frequent item sets with class labels from a streaming data within sliding window.
- We defined an algorithm PSToSW, to directly extract rules for classification on sliding window.
- Performed experiments and found PSToSWMine to have achieved better accuracy than other algorithms designed for the similar task.

Rest of paper is organized as follows. We discuss related work in the next section and give problem definition in Problem Statement Section. Proposed algorithm PSToSWMine along with PSToSW is discussed in next section. The empirical results are shown in Experimental Analysis section and finally we conclude in last section.

2 Related Work

The problem of Associative classification is to find a subset of rules which satisfy supports and confidence. An Associative Classification approach called HARMONY algorithm [8] directly mines k best rules for each transaction and uses these for building a classifier. HARMONY uses an instance-centric rule generation to discover the highest confidence discovering rules. Another algorithm called DDPMine [5] uses sequential covering paradigm for constructing classifier. DDPMine tries to find the best discriminative rules from those transactions which have not been covered and removed and finds locally optimal rules.

STREAMGEN algorithm [3] constructs an enumeration tree for each sliding window and mines a set of item set generators for classification. This algorithm directly mine a set of high quality classification rules over stream sliding windows while keeping high performance. When compared to DDPMine and Moment, the accuracy of prediction of StreamGen algorithm is on the higher side. Classifying a data stream with an associative classifier is a newly explored area of research. There is no algorithm which accurately mines a set of frequently generated rules for classification by taking less amount of time.

Recently another algorithm called AC-DS [12] is proposed as an associative classification algorithm for data streams which works by using support threshold and land mark window model. AC-DS uses single rule for predicting a new data stream. This is biased on general rule, and not appropriate for streams that are slowly changing from time to time. This algorithm works well with single concept. If the concept function is a concept drift one then the algorithm will not output an accurate result.

Motivated by these, we proposed PSToSWMine which improves the efficiency of mining in terms of accuracy of prediction and time consumed for prediction. A new data structure called PSTDTree is developed for online incremental maintenance of data. Because the focus of the paper is on building classifier using associative classification over data streams with sliding window, we mainly compare our approach with StreamGen and DDPMine algorithms.

3 Problem statement

Let data stream $D_s$ be a set of instances $I$ which are grouped under batches. Each batch contains equal number of instances. Each instance in a batch $B$ contains set of values for attributes and class label value. Instance $i$ is represented as $<i_d, A, y>$ where $i \in I$, $i_d$ being instance identification number, $A$ is set of normal attributes present in instance $i$ and $y$ is class label. An item set $S$ is present in $I$ if $S \subseteq I$ holds. The number of instances containing item set $S$ is support count of $S$ denoted as $supCountS$. A common rule is shown as $A \rightarrow y$ where $A$ is set of normal attributes and $y$ is class label attribute. The quality of a rule generated is measured using minimum support denoted as $sup_{min}$ and $conf_{min}$. The rules which do not satisfy these thresholds are called infrequent rules which are rejected and rules satisfying these are used for constructing classifier.

Given $sup_{min}$ we have following definitions.

Definition 1. Current length of data stream is given as $DSL = |B_i| + |B_{i-1}| + \ldots + |B_0|$ where $B_j = |I|$ in which $I$ is set of instances of stream and $j$ is batch number.
**Definition 2.** Item set $S$ is frequent if it satisfies minimum support. That is, $\text{supCount}_S \geq \text{sup}_{\text{min}}$.

**Definition 3.** A rule is of form $A \rightarrow y$ where $A \subseteq I$ and $A \cap \{y\} = \emptyset$.

<table>
<thead>
<tr>
<th>ID</th>
<th>Attribute</th>
<th>Attribute</th>
<th>Attribute</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>2</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$c_1$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>3</td>
<td>$a_1$</td>
<td>$b_2$</td>
<td>$b_3$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>4</td>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$b_3$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>5</td>
<td>$b_1$</td>
<td>$b_1$</td>
<td>$a_3$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>6</td>
<td>$b_1$</td>
<td>$a_2$</td>
<td>$b_3$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>7</td>
<td>$a_3$</td>
<td>$b_2$</td>
<td>$b_3$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>8</td>
<td>$e_4$</td>
<td>$a_3$</td>
<td>$b_3$</td>
<td>$y_2$</td>
</tr>
<tr>
<td>9</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$e_3$</td>
<td>$y_2$</td>
</tr>
<tr>
<td>10</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$e_3$</td>
<td>$y_2$</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Compared with traditional associative classification, associative classification mining over data streams must be an incremental task. The important task of our work is to find complete set of frequent rules from most current sliding window of $I$ instances of data stream. Algorithm for gaining knowledge quickly and accurately is also needed. Table 1 depicts an example of data stream with sliding window size of 2 batches where each batch contains 2 instances of data stream for associative classification.

## 4 Sliding Window

Stream data processing is done using landmark window [11], damped window [11] or with sliding window [11]. In a Sliding window model, discovery of knowledge is achieved using a fixed number of recently generated data stream. For example, given a window of size $N$ over data streams, only latest $|N|$ transactions of stream or all transactions in the last $|N|$ time are used for knowledge gaining. As a new transaction arrives the oldest transaction in the window expires [16]. Representation of sliding window is shown in Fig. 1.

![Fig. 1. Sliding Window.](image1)

Advantages of using sliding window model are:
- It is well defined and easily understood
- It is deterministic
- It emphasizes on recent data

There are two types of sliding windows:
1. Transaction sensitive sliding window
2. Time sensitive sliding window

The sliding window model is therefore widely used to find recent frequent patterns in data streams [3], [6], [7].

The proposed PSToSWMine works over transaction sensitive sliding window.

## 5 Prefix streaming tree over sliding window mining framework

PSToSWMine is a learning classification model based on frequent pattern mining.

The framework for PSToSWMine contains three phases:
1. Representation of stream in a compact data structure called PSTree [10].
2. Frequent item set mining and feature selection called frequent rules using PSToSW and

Framework for associative classification is built in two phases:
- In the first phase, classification rules are discovered from training dataset using frequent item set concept of association. The right-hand-side of the rules is restricted to class label. Rules are represented as $X \rightarrow C$ where $X$ is an item set and $C$ is a class label.
- In the second phase, pruning techniques are applied for generating high quality rules for building accurate classifier. Pruned association rules were used to form classifier based on confidence.

The methodology used is illustrated in Fig. 2.

![Fig. 2. Stream Mining approach using Associative classification over sliding window.](image2)

First, we present some common properties which are used in algorithm design. Then, we introduce the compact tree PSTree. Later, we show the construction of model for
learning called classifier. This is build based on conditional pattern base used in FP-Growth mining.

Some properties used in this paper are

**Property 1.** A frequent item set S is used in classifier if it meets the minimum confidence threshold.

**Property 2.** Given a classifier M, any subset of M would also be a classifier.

**Property 3.** Given an unpromising item set S, any superset of S must be either unpromising or infrequent.

**Property 4.** For a new instance of data stream the state of set of frequent item set S many change depending on frequency of new item sets in the instance.

### 5.1 PSTree

PSTree, the structure used to store data, is based on principles of prefix tree [16] which is an ordered tree. It represents instances flowing through sliding window in a highly compact form. Each read instance is inserted into the tree in a path. As it is possible for multiple instances to have the same items, their path in the tree is overlapped. Due to the overlapping, the compactness of the tree is enhanced. To facilitate the concept of sliding window and tree updating with new instances, each window \( W \) is decomposed into number of fixed size batches of instances called a batch \( B \). Window slides batch by batch.

PSTree is constructed using FP-tree concept for inserting instance into the tree. Creation of this compact tree happens with the help of three stages:

1. Insertion stage
2. Restructuring stage
3. Refreshing stage

Initially the PSTree is empty. After receiving a new instance from a batch of data stream it is inserted into PSTree according to an order which is maintained in I-List. The order is based on support count of items. Later, after complete insertion of instances present in current window, the tree is restructured to maintain compactness. It is done based on sorted list called \( I_{sort}\)-List. This \( I_{sort}\)-List is created by sorting the items present in I-List based on their support count. For restructuring PSTree, we used Branch sorting method [9], [10], [16].

The window slides if the size of current window exceeds the user specified value for window size. Before sliding, algorithm performs refreshing stage by extraction of older batch information to maintain current information of data streams. During next insertion of instances of second window, the item details are maintained using \( I_{sort}\)-List. Batch information of instances is maintained in tree by using batch-counter. This information is stored in leaf node of every path along with class label information of the tree.

The methodology of constructing PSTree for data streams through sliding window is illustrated in Fig.3. Fig.3 (a) shows the initial tree which is empty. Fig.3 (b) depicts the insertion of two instances represented as first batch in data streams. Fig.3 (c) illustrates the restructuring step using \( I_{sort}\)-List. Fig.3 (d), (e) shows the same for second batch of instances. This is repeated till the last stream of data.

The algorithm used for constructing and maintaining PSTree is depicted below along with two methods used for insertion and restructuring the tree.

---

**ALGORITHM 1. Construction and Maintenance of PSTree**

**Algorithm Construct**

**Input:** Data Stream \( DS \) where each record contains \( N \) items, \( W- \)window-size, \( B-Batch \) Size, \( I-List \)

**Output:** PSTree for the window

```
Begin
    \( P \leftarrow 0 \);
    \( T \leftarrow \text{tree with null as initial value} \);
    \( CI-List \leftarrow I-List \);
    \text{while}(P \neq W) \text{ do}
        \text{call} Insert\_Batch(T) \quad \text{// Insertion stage}
        \( CI-List \leftarrow Sort\_Order \);
        \text{call} Restructure(T, CI-List) \quad \text{// Restructuring stage}
        \( P=P+1 \);
    \text{end while}
End
```

Algorithms for insertion and restructuring are presented in [17]. Insertion and Restructuring steps are repeated sequentially for all successive batches till the end of data stream. If the batches \( B_i, B_j \) are currently present in window \( W \), then first insertion step followed with restructuring step for these batches is performed. Later, when window \( W \) slides to \( W_{j+1} \) containing batches \( B_i, B_{i+1} \) the same two steps are repeated. While inserting the new batch \( B_{i+1} \), the oldest batch \( B_i \) is deleted by changing the batch number. Time
complexity of insertion step is $O(mn)$ and for restructuring step it would be $O(n \log m)$ where $m$ is number of items in a transaction and $n$ is number of transactions. PSTree is refreshed before every slide of window in order to provide an environment which helps to mine exact content from the current window. Upon sliding a window the first value in Batch-counter in each leaf node and same value from support count value of each node up to the root in the path are removed, and the remaining values in the list are moved left by one position which shows that the earlier batch is expired.

5.2 Generation of Classifier

The technique used for generation of classifier uses two thresholds given by user as input called minimum support $sup_{min}$ and minimum confidence $conf_{min}$. The generated rules contain item sets and class label which are denoted as $X \rightarrow c$, where $X \subseteq$ frequent item sets and $c \subseteq$ class label. The generated rules are first arranged in an order based on confidence, support and length of generated rule. Ordered rules are pruned using statistical method called chi-square testing ($\chi^2$). This measure helps in testing correlation among rules [15]. The rules which satisfy this testing are used in construction of classifier.

5.3 Learning Model

For the purpose of predicting unknown data, the classifier acts as a model. The mining operation is efficient due to frequency descending prefix structure. The rules found in model are globally optimal. The classifier build using PSToSW tend to have better accuracy in classification. For predicting test data $t$ exactly only those rules $X \rightarrow c$ matching $t$ i.e., $X \subseteq t$ are selected. This algorithm maintains recent information from the data streams. For predicting a new tuple for class label recent information is not sufficient. For doing this the PSToSW must be converted into an incremental algorithm. As the insertion and refreshing stages are independent it is very easy to convert the PSToSW into an incremental algorithm. As these two stages are not related, they can be easily combined depending on the specific type of application in PSToSW.

Incremental PSToSW contains only two stages.
1. Insertion stage
2. Restructuring stage

PSToSW without refreshing stage generates all frequent rules of recent window for classifier. PSToSW with refreshing stage generates a classifier containing all frequent rules collected from entire data stream. The algorithm used for mining data streams using PSToSW is shown below

Input: $min_{\text{sup}}, min_{\text{conf}}, \text{Data Stream DS}$ where each record contains $N$ items, $W$-window-size, $B$-Batch Size, $L$-List

Output: Classifier for the window

Begin
    call Construct for constructing and restructuring PSTree
    Generate frequentPatterns containing Class label which satisfy min_sup
    Build classifier with Rules satisfying min_conf
End

6 Experimental Analysis

In this section we compare the performance and classification accuracy of our incremental PSToSWMine algorithm against several existing algorithms. Incremental PSToSWMine mines rules from real datasets and synthetic datasets by considering window size as two batches where each batch is half the size of data stream. All programs are written in Java and run on windows XP on a 2.53GHz Intel PC with 1.0GB of main memory.

Real Datasets. Real datasets are from UCI Machine Learning Repository [14] and Intel Berkeley Research Lab. The important characteristics of these datasets are listed in Table 2.

Sensor Stream: The data set contains information collected from 54 sensors. It contains information about temperature, humidity, light and sensor voltage. Sensor ID is used as class label, so the task of mining this stream is to correctly identify the sensor ID.

Synthetic Data Streams. We generated synthetic data streams using MOA (Massive Online Analysis) whose characteristics are listed in Table 3. These streams are approximately 80MB in size, consisting of 1 Lakh to 100 Lakhs transactions. All these datasets are very widely used for evaluation of associative classification.

<table>
<thead>
<tr>
<th>Table II</th>
<th>REAL DATA SETS CHARACTERISTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>Number of Transactions</td>
</tr>
<tr>
<td>adult</td>
<td>48842</td>
</tr>
<tr>
<td>breast-w</td>
<td>699</td>
</tr>
<tr>
<td>horse</td>
<td>368</td>
</tr>
<tr>
<td>hepatitis</td>
<td>155</td>
</tr>
<tr>
<td>mushroom</td>
<td>8124</td>
</tr>
<tr>
<td>pima</td>
<td>768</td>
</tr>
<tr>
<td>Sensor</td>
<td>2,219,803</td>
</tr>
</tbody>
</table>

| Stream   |                               |

Synthetic Data Streams. We generated synthetic data streams using MOA (Massive Online Analysis) whose characteristics are listed in Table 3. These streams are approximately 80MB in size, consisting of 1 Lakh to 100 Lakhs transactions. All these datasets are very widely used for evaluation of associative classification.
6.1 Accuracy

To our knowledge, currently there are only few existing algorithms which mine classifier for classification over a data stream using sliding window. Table 4 shows the accuracy comparison of PSToSWMine with StreamGen Rules [3] and DDPMine [5] with minimum support threshold of 1 percent, minimum confidence threshold of 50 percent. These two methodologies perform similar tasks as PSToSWMine does. Comparison was done using six datasets. It is seen that the PSToSWMine gives better accuracy than StreamGen by an average percent of 5.63. It even excels DDPMine by an average accuracy of 7.11. Methodology which attains highest accuracy is shown in bold font. Fig.4 depicts the accuracy comparisons of these algorithms for various datasets. The entire study shows that PSToSWMine outperforms both the classifiers in terms of accuracy.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Transactions</th>
<th>Number of Attributes</th>
<th>Number of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stagger Generator</td>
<td>100,000,000</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Stream</td>
<td>1,000,000</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Hyper Plane Generator Stream</td>
<td>1,000,000</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Agrawal Generator</td>
<td>1,000,000</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Random Tree</td>
<td>1,000,000</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

6.2 Runtime Efficiency

We have conducted experiments for evaluating the runtime efficiency of PSToSWMine with STREAMGEN [3] and with DDPMine [5]. Table 5 shows the time taken for construction, restructuring and prediction in PSToSWMine. Fig.5 (a) depicts the plot between training time and number of transactions for Stagger generator. Fig.5 (b) plots the prediction time against number of transactions for Stagger data stream.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Transactions</th>
<th>Tree Construction Time</th>
<th>Tree Restructuring Time</th>
<th>Prediction Time</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>48842</td>
<td>49</td>
<td>9</td>
<td>54</td>
<td>112</td>
</tr>
<tr>
<td>mushroom</td>
<td>8124</td>
<td>770</td>
<td>769</td>
<td>61</td>
<td>1600</td>
</tr>
</tbody>
</table>

Fig.4. Accuracy Comparison

Fig.5 (a) Training Time when varying number of transactions
Fig.5 (b) Prediction Time when varying number of transactions.
Fig. 6 shows that PSToSW takes less time for generating frequent item sets when compared with StreamGen. The plot is between various support thresholds and time consumption.

7 Conclusion and Future Work

In this paper we introduced an associative classification algorithm called PSToSWMine for data streams. Dynamic tree restructuring is a novel concept that we used to handle streaming data. PSTree construction algorithm uses this technique for achieving a highly compact prefix structure within a single pass on a sliding stream. As it is fast in updating, PSToSWMine is the apt algorithm for mining data streams. Despite of restructuring cost, PSToSWMine’s overall runtime cost is much less than any one of the existing algorithms. Experimental results show that the proposed mining technique increases the classification accuracy due to the availability of large rule sets. By implementing a statistical technique, chi-square testing, the process of rule generation for classifier has been greatly enhanced. This technique shuns information loss and generates the complete non-redundant rule set needed by the classifier. As a future work, we plan to improve the performance of PSToSWMine by reducing the number of rules generated without affecting the accuracy of mining.

8 Acknowledgment

The authors would like to thank the reviewers for helpful comments

9 References