Efficient Classifier Generation over Stream Sliding Window using Associative Classification Approach

K.Prasanna Lakshmi
Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Andhra Pradesh 500 090, India

C.R.K.Reddy, Ph.D
Department of Computer Science, Chaitanya Bharathi Institute of Technology
Hyderabad, Andhra Pradesh 500 075, India

ABSTRACT
Prominence of data streams has dragged the interest of many researchers in the recent past. Mining associative rules generated on data streams for prediction has raised greater research interest in recent years. Associative classification mining has shown better performance over many former classification techniques in Data Mining and Data Stream Mining domains. This paper introduces a new technique for mining data streams using associative classification. To the best of our knowledge there are only few techniques existing. We designed a compact data structure to efficiently maintain data streams without losing any important information. We present a PSToSW 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 then conduct experiments using synthetic and real data sets to assess the performance of our approach. The experimental results show that our technique is superior to existing algorithms which perform similar tasks in terms of accuracy of prediction and run time efficiency.

General Terms
Data Stream Classification, Algorithms et. al.

Keywords
Data Streams, Sliding window, Associative Classification, Frequent item sets.

1. INTRODUCTION
Data stream mining deals with gaining knowledge from the stream of data. Data streams have recently emerged to address the problems of continuous data [11]. Due to its huge size and continuous nature data stream mining needs a real time response after analysis. Mining data streams is not an easy task as mining technique must have ability to produce results quickly by processing data after reading it in a single pass [11]. Sequential access methods for stream mining are cost effective and better than random access methods. Stream mining have emerged due to various applications involving massive data sets; for example, web click streams, financial transactions, science surveillance data etc., These data sets are very big to fit in main memory and are stored in secondary storage devices. Data sets like sensor data, router packet statistics are temporal and need not be stored in disk; it 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. This motivates for a design of algorithms that store summary of past data; leaving memory for dealing out with future data. So, mining algorithm for data streams must examine each data element at most once as fast as possible by taking 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.

A short briefing of our work in the paper is summarized below:

- We used our own created compact data structure called *PSTree* [10] for maintaining the relevant and current information.
- We devise the algorithm for mining frequent item sets containing class labels over sliding window on streaming data.
- Next we create an algorithm *PSToSW*, to directly mine rules for classification on sliding window.
- Performance study shows that *PSToSWMine* achieves better accuracy than algorithms doing 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 The accuracy of prediction by this algorithm is higher when compared with DDPMine and Moment.

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 *PSTree* 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 [3] 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 $< id, A, y >$ where $i \in I$, *id* 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 $supCount_S$. 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 $DSL = |B_1| + |B_2| + \ldots\ldots + |B_m|$ where $Bj = \{I\}$ 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, $supCount_S \geq sup_{min}$.

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

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.

Table 1. Training Dataset

Window 2 Window 1 Data Stream

ID	Attribute A_1	Attribute A_2	Attribute 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
. .	. .	. .	. .	. .

4. 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
3. Model learning phase.

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. 1.

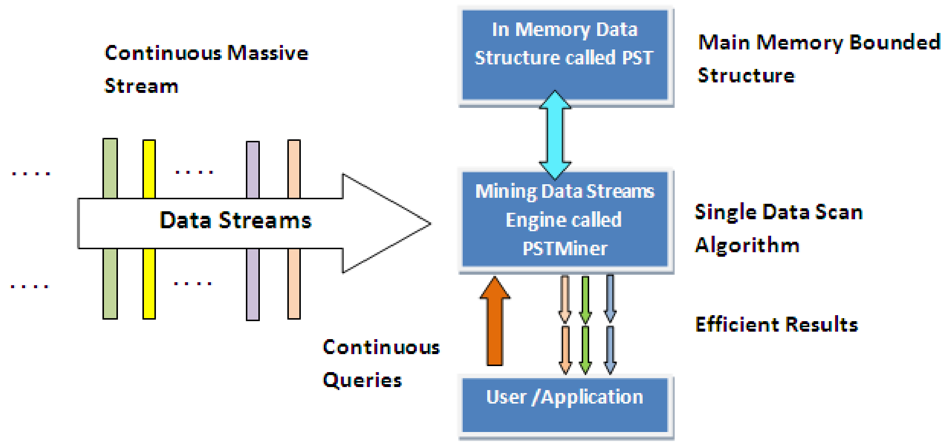

Fig. 1: **Stream mining approach using Associative classification over sliding window**.

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.

4.1 PSTree

PSTree is based on 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 many instances have same items, their path in the tree will be overlapped. The compactness of tree depends on the amount of overlapping. 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.2. Fig.2 (a) shows the initial tree which is empty. Fig.2 (b) depicts the insertion of two instances represented as first batch in data streams. Fig.2 (c) illustrates the restructuring step using I_{sort}*-List*. Fig.2 (d), (e) shows the same for second batch of instances. This is repeated till the last stream of data.

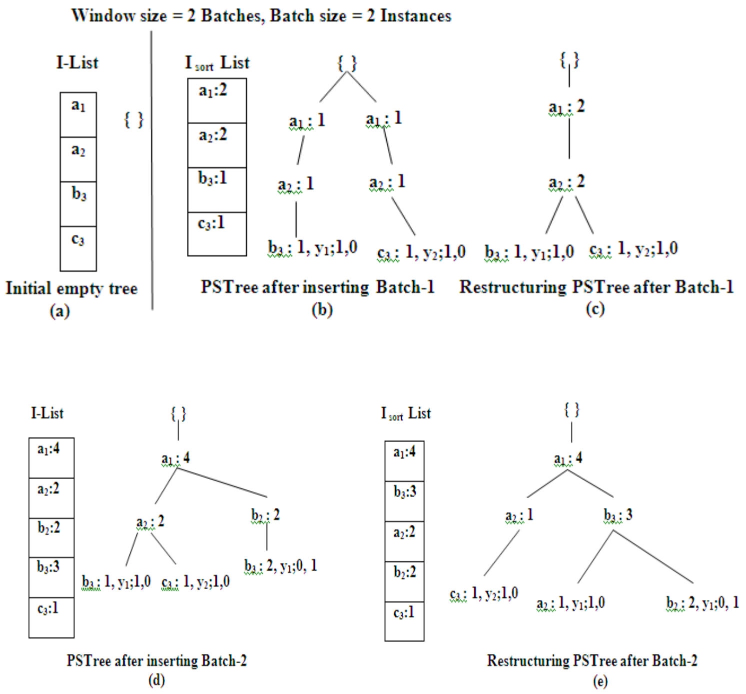

Fig. 2: PSTree Construction

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 $\leftarrow$ *tree with null as initial value;*
CI-List $\leftarrow$ *I-List;*
while*(P ≠ W)* **do**
call *Insert_Batch(T);* // *Insertion stage*
CI-List $\leftarrow$ *Sort_Order;*
call *Restructure(T, CI-List);* // *Restructuring stage*
P=P+1;
end while
end

Insert _Batch(T_s)
Output : Tree T_s
Begin
p $\leftarrow$ *0;*
while*(p ≠ B)* **do**
read transaction tr from DS;
insert t_r into T_s according to CI-List;
p=p+1;
end while
end

Restructure(T, I)

Output: T_{sort}
Begin
for *each Branch* B_i *in T*
for *each unprocessed path* P_j *in* B_i
if P_j *is unsorted path*
sortPath(P_j);
else *processBranch*(P_j);
end if
end for
end for
Stop when all branches are sorted and output T_{sort}
end

sortPath (P)
Begin
Reduce the support count of all nodes of P in T by total support count of $leaf_P$;
Delete all nodes having support count 0 from P;
Using merge sort technique, sort items in P in an array according to CI-List;
Insert sorted P into T at the location from where it was retrieved;
Assign the batch-counter and Class label of P to the new leaf node;
end
processBranch (P)
Begin
for *each branching node* n_p *in P from* $leaf_l$ *node*
for *each sub-path from* n_p *to* $leaf_k$ *with* $k \neq l$
if *items of all nodes between* n_p *and* $leaf_k$ *are at below of* n_p *in the CI-List*
P=sub-path from n_b *to* $leaf_k$;
if *P is a sorted path*
processBranch(P);
end if
else *P= path from he root to* $leaf_k$;
end if
sortPath(P);
end for
end for
end

Insertion and Restructuring steps are repeated sequentially for all successive batches till the end of data stream. If the batches B_{i-1}, B_i are currently present in window W_j then first insertion step followed with restructuring step for these batches is performed. Later, when window W_j 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-1} is deleted by changing the batch number. Time complexity of insertion step is $O(mn)$ and for restructuring step it would be $O(n\log_2 m)$ 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.

4.2 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 $c \subset$ 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 (χ^2). This measure helps in testing correlation among rules [15]. The rules which satisfy this testing are used in construction of classifier.

4.3 Learning Model

The classifier is used as a model for predicting unknown data. 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.

ALGORITHM 2. PSToSW mining for a window

Input: *min_sup, min_conf, Data Stream DS where each record contains N items, W-window-size, B-Batch Size, I-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

5. EXPERIMENTAL RESULTS

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.

Table 2. Real Data sets Characteristics

Dataset	Number of Transactions	Number of Attributes	Number of Classes	Number of Items
adult	48842	14	2	128
breast-w	699	10	2	29
horse	368	28	2	61
hepatitis	155	19	2	33
mushroom	8124	22	2	116
pima	768	8	2	15
Sensor stream	2,219,803	5	58	--

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 3. Synthetic Data sets Characteristics

Dataset	Number of Transactions	Number of Attributes	Number of Classes
Stagger Generator Stream	100,00,000	4	2
Hyper Plane Generator Stream	1,00,000	10	5
Agarwal Generator Stream	1,00,000	10	2
Random Tree Generator Stream	1,00,000	5	2
Sea Generator Stream	1,00,000	4	2

5.1 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* [3] 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.3. depicts the accuarcy comparisons of these algorithms for various datasets. The entire study shows that *PSToSWMine* outperforms both the classifiers in terms of accuracy.

Table 4. Accuracy Comparison

Dataset	StreamGen	DDPMine	PSToSW
Adult	82.1	81.29	79.31
breast-w	96.7	95.28	97.85
Horse	81.51	81.24	92.31
Hepatitis	82.0	76.98	100
mushroom	98.91	97.18	100
Pima	74.81	75.12	80.31
sensor stream	---	---	100
Average Accuracy	86.0	84.51	91.63

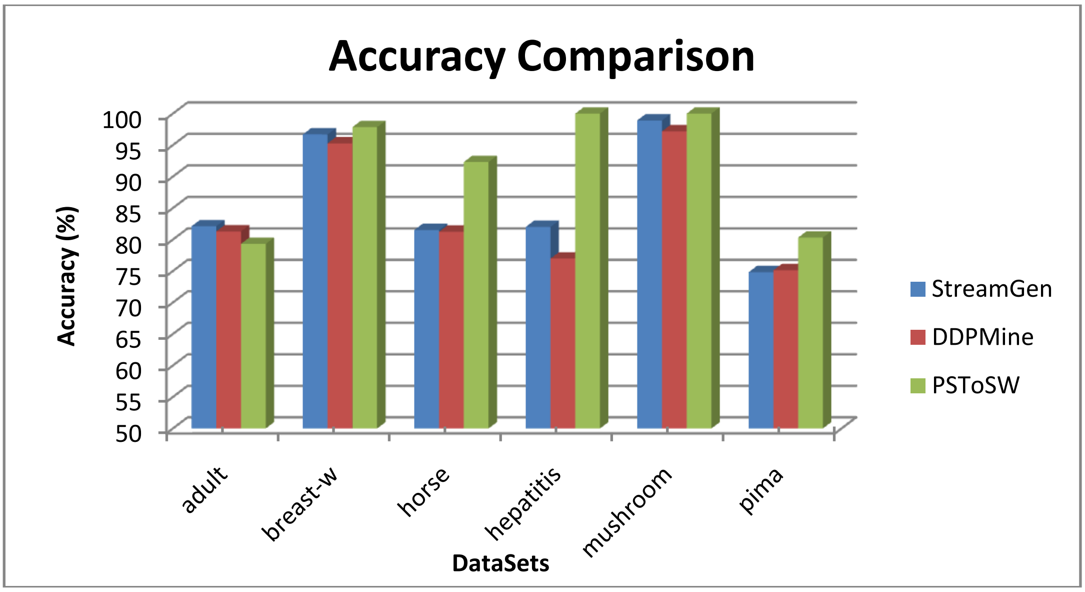

Fig.3: Accuracy Comparison

5.2 Runtime Efficiency

We have conducted experiments for evaluating the runtime efficiency of *PSToSW* with *STREAMGEN* [3] and with *DPMine* [5]. Table 5 shows the time taken for construction, restructuring and prediction in *PSToSWMine*. Fig.4 (a) depicts the plot between training time and number of transactions for Stagger generator. Fig.4 (b) plots the prediction time against number of transactions for Stagger data stream.

Table 5. Runtime Distribution in seconds

Data streams with minimum support and confidence	Number of Transactions	Tree Construction Time	Tree Restructuring Time	Prediction Time	Total Time
Real Time Datasets					
Adult min_sup=1, min_conf=50%	48842	49	9	54	112
mushroom min_sup=10, min_conf=50%	8124	770	769	61	1600
Sensor Stream min_sup=0.1, min_conf=50%	1,00,000	20	11	62	93
Synthetic Datasets					
StaggerGenerator min_sup=1, min_conf=50%	100,00,000	60	0.001	16	76.001
Hyper Plane Generator min_sup=1, min_conf=50%	1,00,000	15	6	43	64
Agarwal Generator min_sup=1 , min_conf=50%	1,00,000	4795	42	26	4863
Random Tree Generator min_sup= 1, min_conf=50%	1,00,000	714	1.9	0.8	716.7
Sea Generator min_sup= 1, min_conf=50%	1,00,000	99	50	12	161

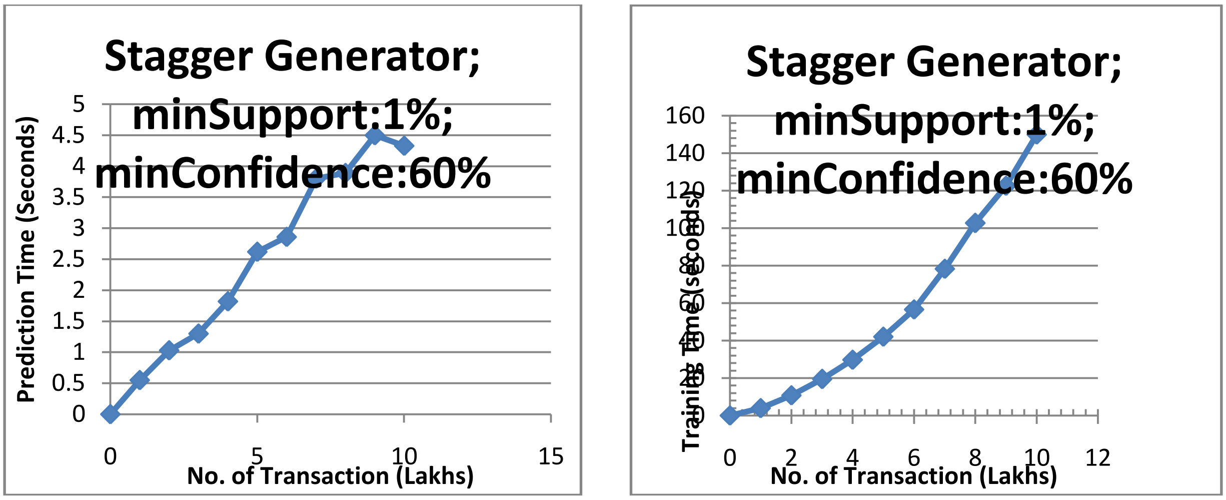

Fig.4 : (a) Training Time when varying number of transactions (b) Prediction Time when varying number of transactions.

Fig.5. shows that *PSToSW* takes less time for generating frequent item sets when compared with StreamGen [3]. The plot is between various support thresholds and time consumption.

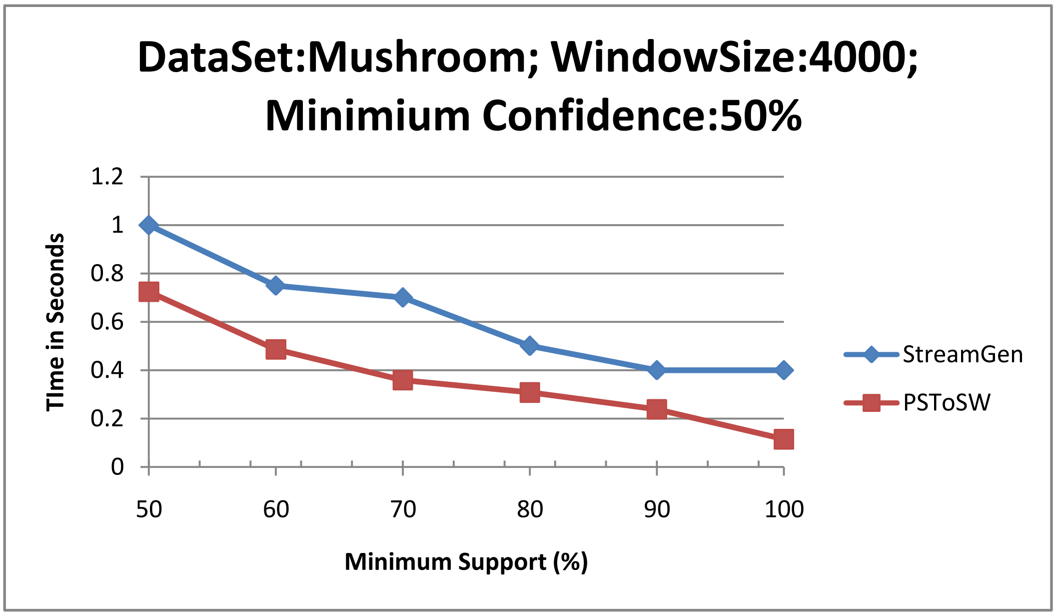

Fig.5 : Runtime Comparison of PSToSW with StreamGen

6. CONCLUSIONAND FUTURE WORK

In this paper we introduced an associative classification algorithm called *PSToSWMine* for data streams. We used a novel concept of dynamic tree restructuring 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. *PSToSWMine* is very suitable for mining data streams as it is fast updating. 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. The process of rule generation for classifier was further enhanced by implementing a statistical technique called chi-square testing. 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.

7. ACKNOWLEDGMENTS

The authors would like to thank the reviewers for helpful comments

8. REFERENCES

[1] B. Liu, W. Hsu, and Y. Ma, "Integrating Classification and Association Rule Mining," Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD '98), Aug. 1998.

[2] C.K.S. Leung, Q.I. Khan, DSTree: a tree structure for the mining of frequent sets from data streams, in: Proc. ICDM, 2006, pp. 928– 932.

[3] Chuancong Gao, Jianyong Wang, "Efficient item set generator discovery over a stream sliding window" in C

IKM'09, November 2009, Hong Kong, China, ACM 978-1-60558-512-3/09/11

[4] Hong Yao, H.J Hamilton (2006)," Mining item set utilities from transaction data bases", IEEE Transactions on Data and Knowledge Engineering, volume 59, issue 3, pp.603-626.

[5] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In Proceedings of the 24th International Conference on Data Engineering, pages 169–178, Cancun, Mexico, 2008. IEEE.

[6] J.H. Chang, W.S. Lee, estWin: Online data stream mining of recent frequent item sets by sliding window method, Journal of Information Science 31 (2) (2005) 76–90.

[7] J. Li, D. Maier, K. Tuftel, V. Papadimos, P.A. Tucker, No pane, no gain: efficient evaluation of sliding-window aggregates over data streams, SIGMOD Record 34 (1) (2005) 39–44.

[8] J. Wang and G. Karypis. On mining instance-centric classification rules. IEEE Trans. Knowledge Data Engineering 18(11):1497–1511, 2006

[9] Koh, and Shieh, 2004. An efficient approach for maintaining association rules based on adjusting FP-tree structures. In Proc. of DASFAA 2004. Springer-Verlag, Berlin Heidelberg New York, 417–424.

[10] K.Prasanna Lakshmi, Dr.C.R.K.Reddy, "Compact Tree for Associative Classification of Data Stream Mining", IJCSI International Journal of Computer Science Issues, Vol 9, Issue 2, No 2, March 2012, ISSN(online) : 1694-0814

[11] 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.

[12] L. Su, H. Liu and Z. Song, "A New Classification Algorithm for data stream". I.J.Modern Education and Computer Science, 4, 32-39, 2011.

[13] W. Li, J. Han, and J. Pei, "CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules," Proc. IEEE Int'l Conf. Data Mining (ICDM '01), Nov. 2001.

[14] R. C. Agarwal, C. C. Agarwal, 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.

[15] Snedecor. W, and Cochran. W(1989) Statistical Methods, Eighth Edition, Iowa State University Press.

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