International Journal of Advance Engineering and Research Development (IJAERD)
Volume 1,Issue 5,May 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

Fequent Pattern Recognization From Stream Data Using Compact Data Structure

Fabin M Christian[1], Narendra C.Chauhan[2], Nilesh B. Prajapati[3]

[1]*PG Scholar, CE Department, BVM Engg. College, V.V.Nagar*, fabin.christian@gmail.com

[2]*Assoc. Prof. , IT Department, A.D. Patel Engg. College, V.V.Nagar*, narendracchauhan@gmail.com

[3]*Assoc. Prof., IT Department, BVM Engg. College,V.V.Nagar*, nilesh.prajapati@bvmengineering.co.in

Abstract— Mining frequent pattern from stream data is a challenging task. Finding frequent pattern from data streams have been found to be useful in many application such as stock market prediction, sensor data analysis, network traffic analysis, e-business and telecommunication data analysis. Frequent Pattern Stream tree [1] is used for maintaining frequent pattern over a period of time using modified FP tree algorithm. This approach maintains tilted time window at each node which consumes larger space. Compact Pattern Stream Tree [2] assumes that only current patterns are of importance and uses sliding window protocol for maintaining it. This approach does not give importance to past frequent patterns.

The aim of this research paper is to combine the features of frequent pattern stream tree and Compact pattern stream tree to find the frequent pattern over a period of time. The research assumes the patterns that are infrequent for a period can become frequent in the future. It is proposed to generate a Compact Frequent Pattern Stream tree that maintains tilted time window at the tail node and at partition node that keep track of the pattern over a period of time. The pattern whose support count is greater than predefined threshold is considered to be frequent. The proposed approach has proved advantage of reduction in space utilization, compaction in space utilization and consideration of past patterns that may become frequent. The system will generate frequent pattern with compact data structure and in a time efficient manner.

Index Terms—Frequent pattern stream tree, Compact pattern stream tree, Dynamic stream tree tilted time window, sliding window

1. INTRODUCTION

Frequent pattern has been studied widely over a period of time but extending it to data stream is a challenging task. Data items are continuously flowing through the internet or sensor networks in applications like network monitoring and message dissemination [5]. Frequent pattern mining and its associated methods are widely used in association rule mining tasks, sequential pattern mining, sequential pattern mining, structured pattern mining, associative classification and so on[1].

Recently, the increasing prominence of data streams has led to the study of online mining of frequent Itemsets, which is an important technique that is essential to a wide range of emerging applications, such as web log and click-stream mining, network traffic analysis, trend analysis and fraud detection in telecommunications data, e-business and stock market analysis, and sensor networks. With the rapid emergence of these new application domains, it has become increasingly difficult to conduct advanced analysis and data mining over fast-arriving and large data streams in order to capture interesting trends, patterns and exceptions [5]. Mining frequent pattern from continuous data stream is more difficult than finding frequent pattern from static database. The basic properties of data stream are that they are continuous as they arrive at fast rate. Secondly, data can only be read only once because of continuity. Thirdly the total amount of data that are arriving is unbounded. Mining streams require fast processing so that fast data arrival rate can be maintained and results of mining can be achieved in shorter response time.

The mining of continuously arriving data can be done through time base sliding window approach. There are three important windows models can be used based on feature of the application landmark window model, damped window model or sliding window model. Landmark window performs mining from data from a particular point called landmark to the current time. In the damped window model arriving data is assigned various weights based on their arrival, newly arriving data are assigned higher weights than older data. In sliding window fixed window is maintained and mining is applied within this window.

2. STREAM MINING MODELS

Some paper that describes various methods to find frequent pattern identification from data streams are described and discussed here.

2.1 Frequent pattern stream tree model

In the paper [1] a method for finding frequent pattern using FP stream tree is proposed. Compared to mining from a static transaction data set, the streaming case has far more information to track and far greater complexity to manage. Infrequent items can become frequent later on and hence cannot be ignored. The storage structure needs to be dynamically adjusted to reflect the evolution of Itemsets frequencies over time. In this paper, computing and maintaining all the frequent patterns (which is usually more stable and smaller than the streaming data) and dynamically updating them with the incoming data streams. They extended the framework to mine time-sensitive patterns with approximate support guarantee. They incrementally maintain -time windows for each pattern at multiple time granularities. Interesting queries can be constructed and answered under this framework. Moreover, inspired by the fact that the FP-tree provides an effective data structure for frequent pattern mining, we develop FP-stream, an effective FP-tree-based model for mining frequent patterns from data streams. An FP-stream structure consists of (a) an in-memory frequent pattern-tree to capture the frequent and sub-frequent Itemsets information, and (b) a tilted-time window table for each frequent pattern. Efficient algorithms for constructing, maintaining and updating an FP-stream structure over data streams are explored.

2.2 Compact pattern stream model

In this paper [2] a method for finding pattern from data stream using Compact pattern stream tree. An efficient technique to discover the complete set of recent frequent patterns from a high-speed data stream over a sliding window has been proposed. A Compact Pattern Stream tree (CPS-tree) to capture the recent stream data content and efficiently remove the obsolete, old stream data has been developed to provide compaction. This paper also introduces the concept of dynamic tree restructuring in our CPS-tree to produce a highly compact frequency-descending tree structure at runtime. This causes the high frequency pattern to be maintained at the higher level and low frequency components are mainly maintained at the leaf nodes. The complete set of recent frequent patterns is obtained from the CPS-tree of the current window using an FP-growth mining technique.

2.3 Classification and clustering model

In this paper [3] a method for stream classification is proposed. A new ensemble model which combines both classifiers and clusters together for mining data streams is proposed. The main challenges of this new ensemble model include (1) clusters formulated from data streams only carry cluster IDs, with no genuine class label information, and (2) concept drifting underlying data streams makes it even harder to combine clusters and classifiers into one ensemble framework. To handle challenge (1) a label propagation method to infer each cluster's class label by making full use of both class label information from classifiers, and internal structure information from clusters is proposed. To handle challenge (2), we present a new weighting schema to weight all base models according to their consistencies with the up-to-date base model. As a result, all classifiers and clusters can be combined together, through a weighted average mechanism, for prediction. Due to its inherent merits in handling drifting concepts and large data volumes, ensemble learning has traditionally attracted many attentions in stream data mining research. Nevertheless, as labeling training samples is a labor intensive and expensive process, in a practical stream data mining scenario, it is often the case that we may have a very few labeled training samples (to build classifiers), but a large number of unlabeled samples are available to build clusters. It would be a waste to only consider classifiers in an ensemble model, like most existing solutions do. Accordingly, in this paper, a new ensemble learning method which relaxes the original ensemble models to accommodate both classifiers and clusters through a weighted average mechanism. In order to handle concept drifting problem, we also propose a new consistency-based weighting schema to weight all base models, according to their consistencies with respect to the up-to-date model.

2.4 Sliding window model

In this paper [4] they developed a novel approach for mining frequent Itemsets from data streams based on a time-sensitive sliding window model. The approach consists of a storage structure that captures all possible frequent Itemsets and a table providing approximate counts of the expired data items, whose size can be adjusted by the available storage space. . In addition, the approach guarantees no false alarm or no false dismissal to the results yielded. Self adjusting discounting table (SDT) for the limited memory is proposed. The SDT performs well when the available memory is limited.

2.5 Posterior probability based classification model

In this paper [5] a method is proposed method for classification of stream data In this paper a new approach to mine data streams by estimating reliable posterior probabilities using an ensemble of models to match the distribution over under-samples of negatives and repeated samples of positives has been proposed. They formally show some interesting and important properties of the proposed framework, e.g., reliability of estimated probabilities on skewed positive class, accuracy of estimated probabilities, efficiency and scalability.

2.6 Frequent pattern identification model based on dynamic load balancing

In this paper [6] a method is proposed for frequent pattern discovery from data streams In this paper, the practical problem of frequent-Itemsets discovery in data-stream environments which may suffer from data overload. The main issues include frequent-pattern mining and data-overload handling. Therefore, a mining algorithm together with two dedicated overload-handling mechanisms is proposed. The algorithm extracts basic information from streaming data and keeps the information in its data structure. The mining task is accomplished when requested by calculating the approximate counts of Itemsets and then returning the frequent ones. When there exists data overload, one of the two mechanisms is executed to settle the overload by either improving system throughput or shedding data load.

2.7 Dynamic stream tree model

DSTree [8], which is a prefix-tree that is built based on a canonical item order, has recently been proposed for mining an exact set of recent frequent patterns over a data stream using a sliding window technique. In DSTree, the sliding window consists of several batches of transactions, and the transaction information for each batch is explicitly maintained at each node in the tree structure. To discover the complete set of recent frequent patterns, the FP-growth mining technique is applied to the DSTree of the current window. Even though the construction of the DSTree requires only one scan of the data stream, it cannot guarantee that a high level of prefix sharing (like in FP-tree) will be achieved in the tree structure because the items are inserted into the tree in a frequency-independent canonical order. Moreover, upon sliding of window the tree update mechanism of DSTree may leave some 'garbage' nodes in the tree structure when the mining request is delayed.

3. PROPOSED APPROACH

Assuming that our proposed system architecture is already present, we carry out research regarding frequent pattern from data stream.

It combines the features of frequent pattern stream tree and Compact pattern stream tree. The frequent pattern stream tree maintains tilted time window at each node showing support count of the particular item over a period [1]. The FP stream tree also maintains a pattern tree showing support count of the item in the current window. The compact pattern stream tree uses window based approach in which the window is divided into pane and each pane constitutes the transaction [2]. The nodes are divided into two types 1) Ordinary node storing the item and the support count of the item and 2) Tail node and partition node storing the support count of the item as the information of the pane. The support count of the item is maintained at each node. The pane information is maintained at the tail node and partition node.

We could propose the working of our proposed model in the following manner:

The incoming data stream is divided into sliding windows. The size of the sliding window needs to be specified by the user. The data stream is equally divided according to the sliding window size specified. The compact FP Stream tree is constructed by maintaining tilted time window at the tail node and at the partition node. The tilted time window consists of the support count of the pattern as well as the sliding windows in which the pattern is occurred. The tilted time window consists of this information over a period of time. The entire pattern whose support count is above the threshold value is considered to be frequent. Depth first search is carried out starting from the root node to find the frequent pattern. Each tilted time window is checked and the final output will contain the frequent pattern, the transaction window in which the pattern is frequent as well as the support count of the frequent pattern. The structure of Compact FP stream tree is shown in Figure 3.1. The size of the sliding window is assumed to be three. Thus three words form a single transaction. The root of the tree is a always a null node. The given stream is paper, puddle and polythene. During the initial step the count of each and every word is recorded. The words are arranged in descending order in the Table 3.1. Thus the maximum occuring is entered in the table first.

Table 3.1 Table maintaining word count

Word	Count
P	4
E	4
D	2
L	2
A	1
R	1
U	1
O	1
Y	1
T	1
H	1
N	1

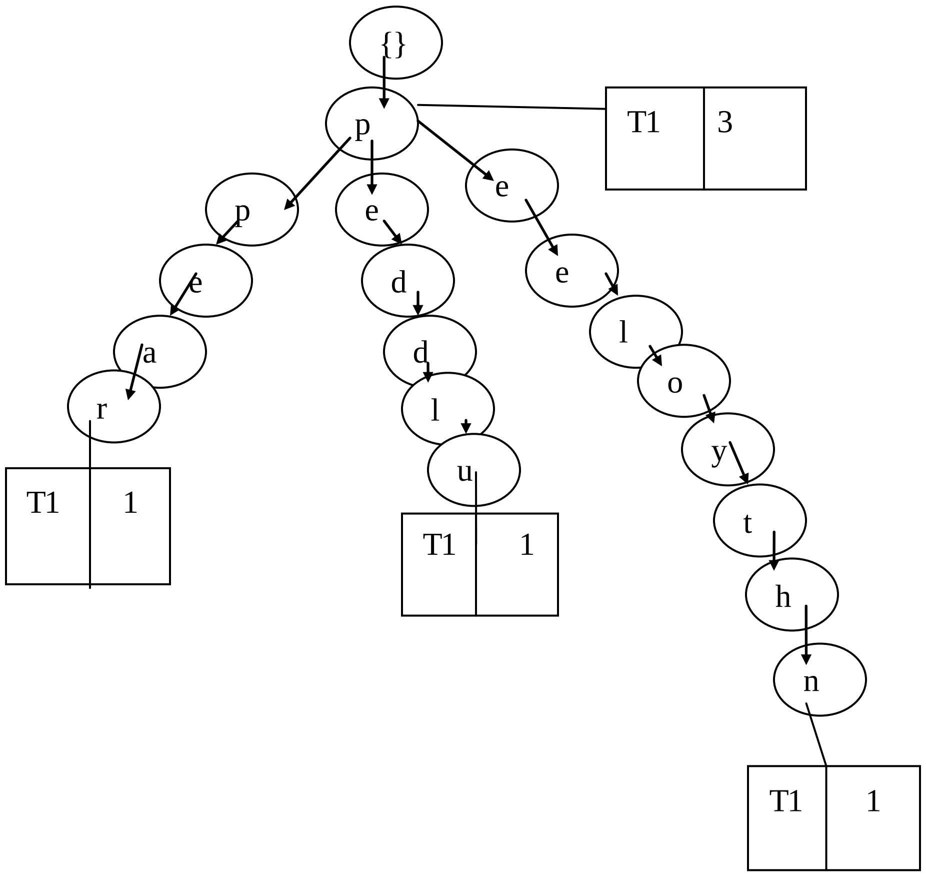


Figure 3.1 Compact FP stream tree

For same pattern paper, puddle and polythene the FP stream implementation is as shown in Figure 3.4. The sliding window size is assumed to three. In FP stream tree the tilted time window is to be maintained at each and evey node. It is also necessary to specify the number of transaction in each tilted time window. The table showing word count in window for FP stream tree is shown in Table 3.1.

As shown in Figure 3.2 the tilted time window in FP stream tree has to be maintained at each node making the data structure complex. Moreover the size of the TTW has to be speified in advance making the data structure static. Here the TTW size is assumed to be one. Threshold is provided by the user. The pattern whose support count is greater than threshold is said to be frequent.

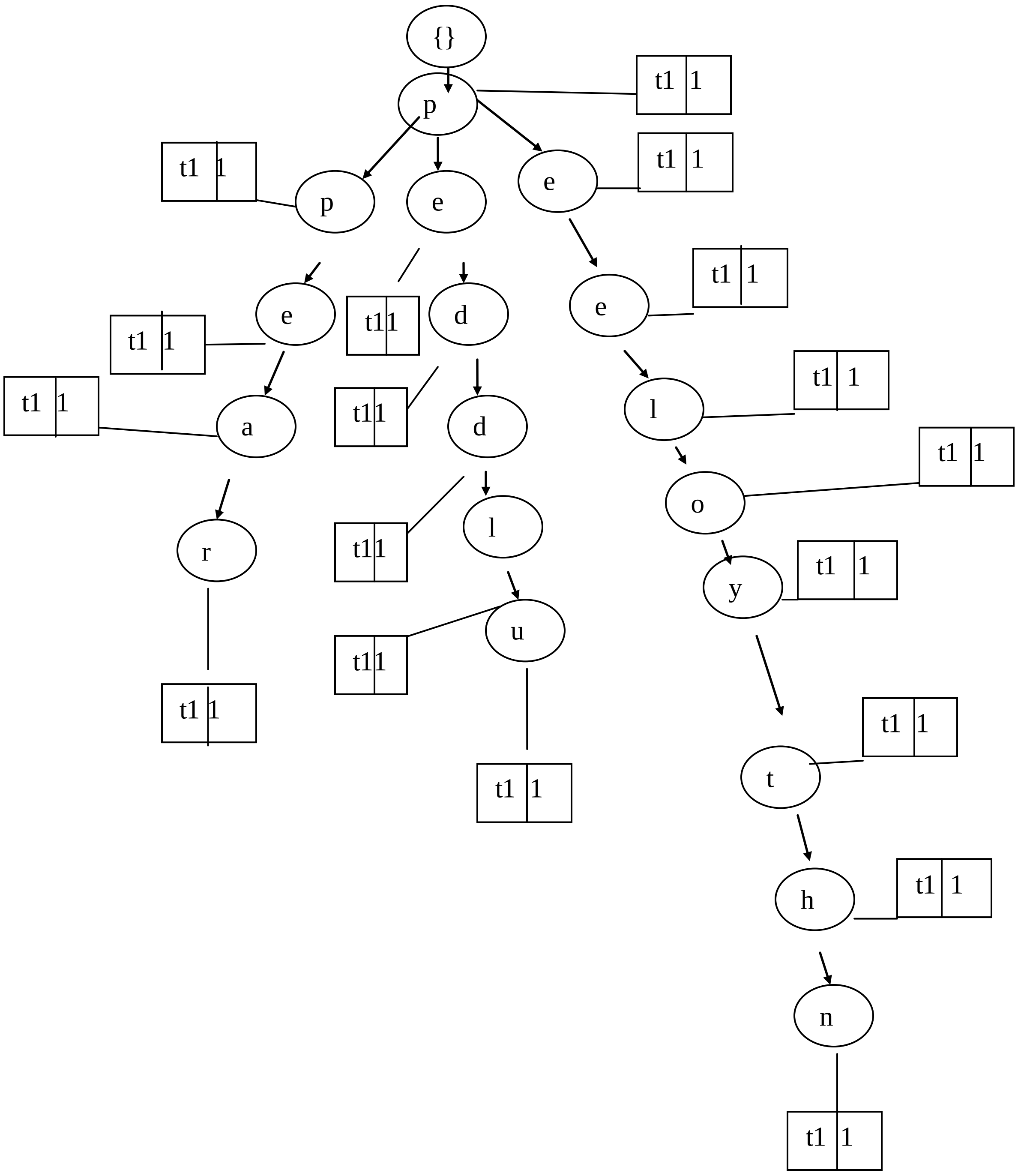


Figure 3.2 FP stream tree [1]

4. PROPOSED ALGORITHM

The proposed Compact FP stream algorithm is as follows:

Step 1: Initialize Compact FP stream tree to be empty.

Step 2: When the incoming stream are accumulated:

(i) The incoming stream data is divided into equal sized sliding window specified by the user.

(ii) The table containing pattern and suppot count is created. The pattern is arranged in the table in descending order of its support count.

(iii) The compact FP stream tree is constructed by maintaining linked list based tilted time window at partition and tail nodes.

(iv) The patterns are arranged in the Compact FP stream tree according to its occurance in the table.

Step3: Threshold σ is taken from the user.

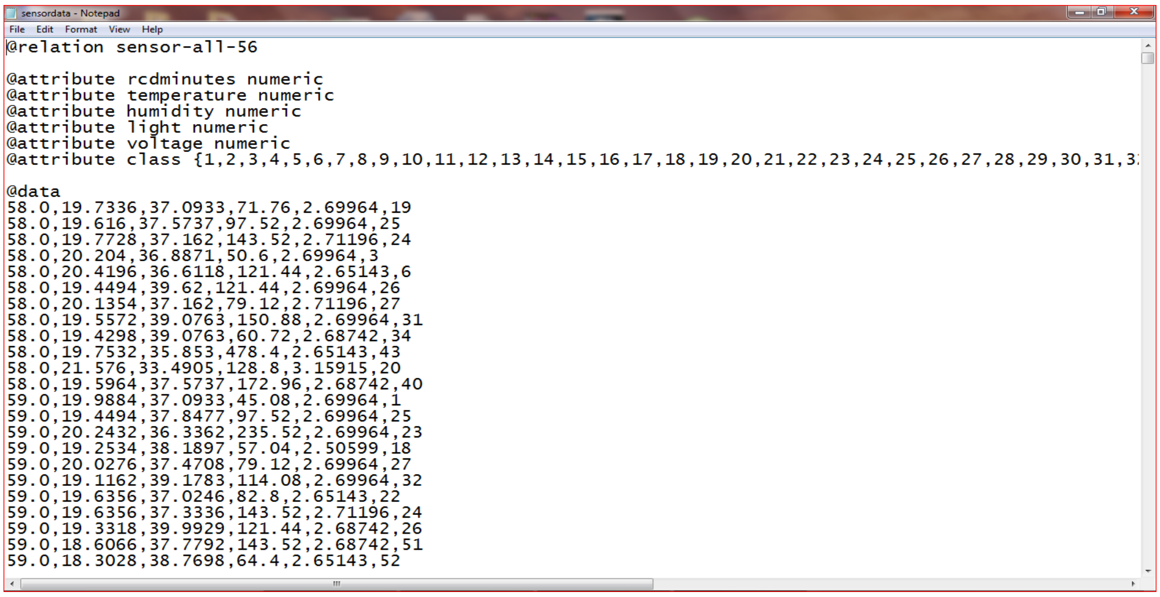
Step 4: Depth First traversal is done starting from the root node and the support count in each linked list based tilted time window maintained at tail node and partition node is checked against the threshold value σ.

Step 5: All those pattern whose support count is greater than threshold σ is displayed along the window number and support count.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 Data Set Used

Sensor stream contains information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed in Intel Berkeley research lab. The whole stream contains consecutive information recorded over a 2 months period. The sensor ID is the class label, so the learning task of the stream is to correctly identify most frequently occurring sensor ID purely based on the sensor data and the corresponding recording time. The data set is in standard Comma Separate Value(CSV) format. The aim is to find which class of Sensor ID is occurring most frequently and controlling the temperature and humidity corresponding to that region. Figure 5.1 shows the Sensor stream dataset.

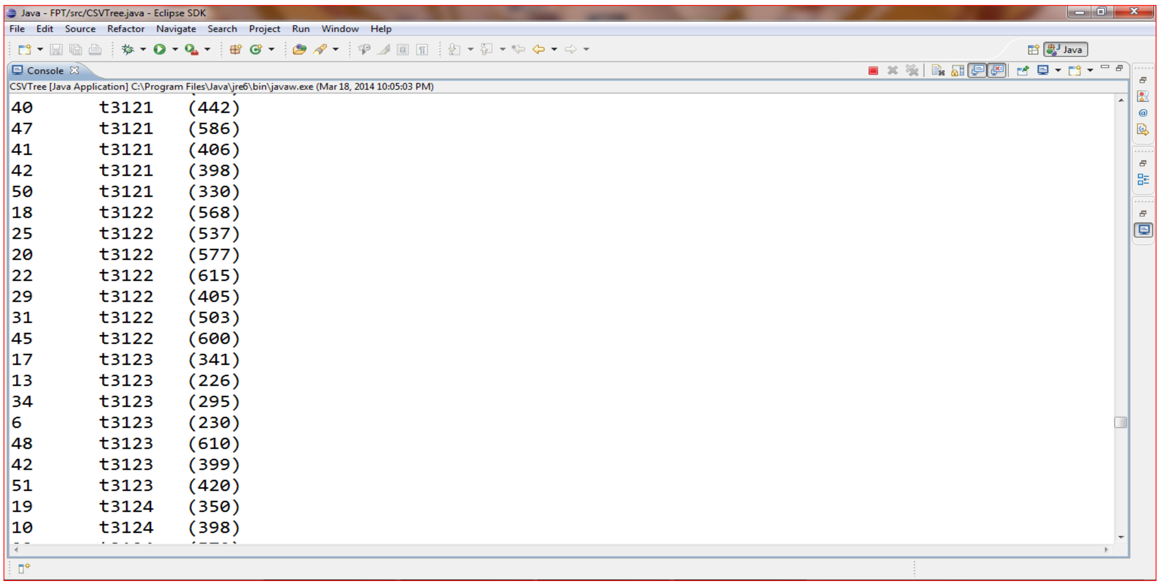


```
@relation sensor-all-56

@attribute rcdminutes numeric
@attribute temperature numeric
@attribute humidity numeric
@attribute light numeric
@attribute voltage numeric
@attribute class {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,3

@data
58.0,19.7336,37.0933,71.76,2.69964,19
58.0,19.616,37.5737,97.52,2.69964,25
58.0,19.7728,37.162,143.52,2.71196,24
58.0,20.204,36.8871,50.6,2.69964,3
58.0,20.4196,36.6118,121.44,2.65143,6
58.0,19.4494,39.62,121.44,2.69964,26
58.0,20.1354,37.162,79.12,2.71196,27
58.0,19.5572,39.0763,150.88,2.69964,31
58.0,19.4298,39.0763,60.72,2.68742,34
58.0,19.7532,35.853,478.4,2.65143,43
58.0,21.576,33.4905,128.8,3.15915,20
58.0,19.5964,37.5737,172.96,2.68742,40
59.0,19.9884,37.0933,45.08,2.69964,1
59.0,19.4494,37.8477,97.52,2.69964,25
59.0,20.2432,36.3362,235.52,2.69964,23
59.0,19.2534,38.1897,57.04,2.50599,18
59.0,20.0276,37.4708,79.12,2.69964,27
59.0,19.1162,39.1783,114.08,2.69964,32
59.0,19.6356,37.0246,82.8,2.65143,22
59.0,19.6356,37.3336,143.52,2.71196,24
59.0,19.3318,39.9929,121.44,2.68742,26
59.0,18.6066,37.7792,143.52,2.68742,51
59.0,18.3028,38.7698,64.4,2.65143,52
```

Figure 5.1: Sensor Data set [13]

5.2 Implementation

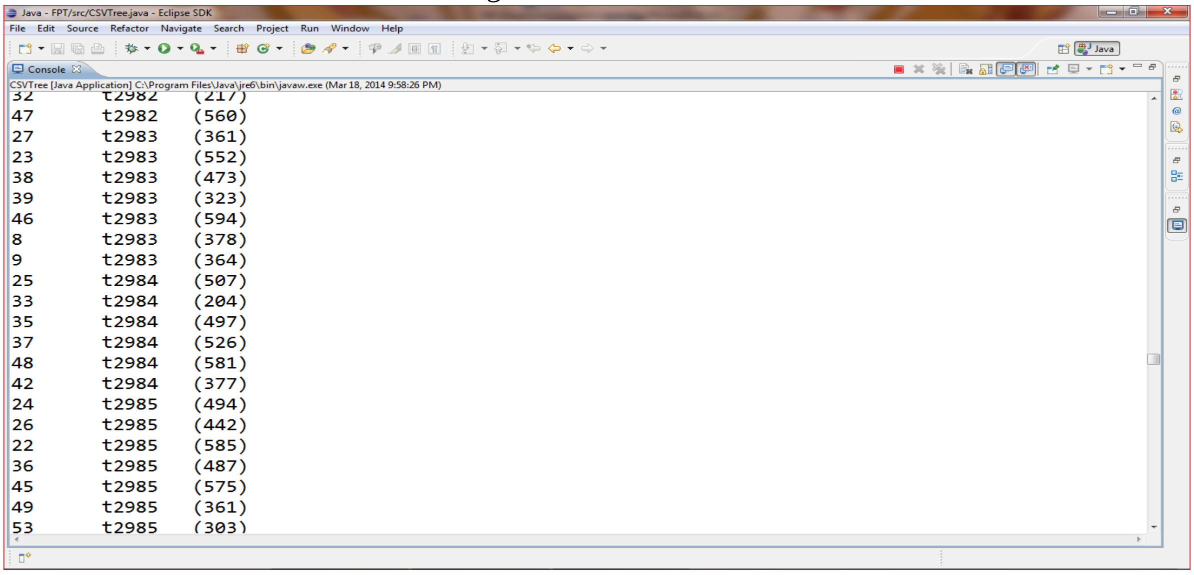
The results of applying the FP stream tree implementation on Sensor data set is shown in Figure 5.2. The threshold is taken to be four and the sliding window size is taken to be seven. The first field of the output provides the class names which are frequent, the second field gives the transaction window number in which the items is frequent and the third field gives the frequency of the corresponding frequent item.


Java - FPT/src/CSVTree.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Console

```
CSVTree [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Mar 18, 2014 10:05:03 PM)
40      t3121   (442)
47      t3121   (586)
41      t3121   (406)
42      t3121   (398)
50      t3121   (330)
18      t3122   (568)
25      t3122   (537)
20      t3122   (577)
22      t3122   (615)
29      t3122   (405)
31      t3122   (503)
45      t3122   (600)
17      t3123   (341)
13      t3123   (226)
34      t3123   (295)
6       t3123   (230)
48      t3123   (610)
42      t3123   (399)
51      t3123   (420)
19      t3124   (350)
10      t3124   (398)
```

Figure 5.2: Frequent Pattern Stream tree implementation on sensor data.

After implementation of the sensor data on Compact Frequent pattern stream tree with threshold value four and sliding window size seven the result is shown in figure 5.3. The first column in Figure 5.3 shows the class which is frequent, the second column shows the transaction in which the pattern is frequent and the thid column shows the support count of each frequent pattern. The threshold is taken to be four and the sliding window size is taken to be seven.


Java - FPT/src/CSVTree.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Console

```
CSVTree [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Mar 18, 2014 9:58:26 PM)
32      t2982   (217)
47      t2982   (560)
27      t2983   (361)
23      t2983   (552)
38      t2983   (473)
39      t2983   (323)
46      t2983   (594)
8       t2983   (378)
9       t2983   (364)
25      t2984   (507)
33      t2984   (204)
35      t2984   (497)
37      t2984   (526)
48      t2984   (581)
42      t2984   (377)
24      t2985   (494)
26      t2985   (442)
22      t2985   (585)
36      t2985   (487)
45      t2985   (575)
49      t2985   (361)
53      t2985   (303)
```

Figure 5.3: Compact FP Stream Tree implementation on sensor data.

5.3 Experimental Results

5.3.1 Comparison of FP Stream tree and Compact FP Stream tree for tilted time window requirement.

The Comparison table for the amount of the tilted time window to be maintained at the nodes of FP stream tree and Compact FP stream tree is shown in Table 5.1

Table: 5.1 Comparison FP stream tree and Compact FP stream tree based on tilted time window

Size of the sliding window	Number of the windows to be maintained for FP stream tree	Number of windows to be maintained for Compact FP stream tree	Percentage reduction in number of windows in Compact FP stream tree with

			respect to FP stream tree
7	183600	175867	5%
9	183600	136803	36%
11	183600	111938	39%
13	183600	94679	48%
15	183600	82060	55%
17	183600	72397	60%

The window size is varied and the threshold is taken to be 10 in Table 5.1. The above result on sensor data shows that by increasing the size of the sliding window there is reduction in the number of tilted time window to be maintained in the Compact FP stream tree but the size of the windows in FP stream tree remains the same because the size of the minimum number of windows needed have to be specified before the FP stram tree is constructed. Thus a compact structure is obtained by Compact FP stream tree.

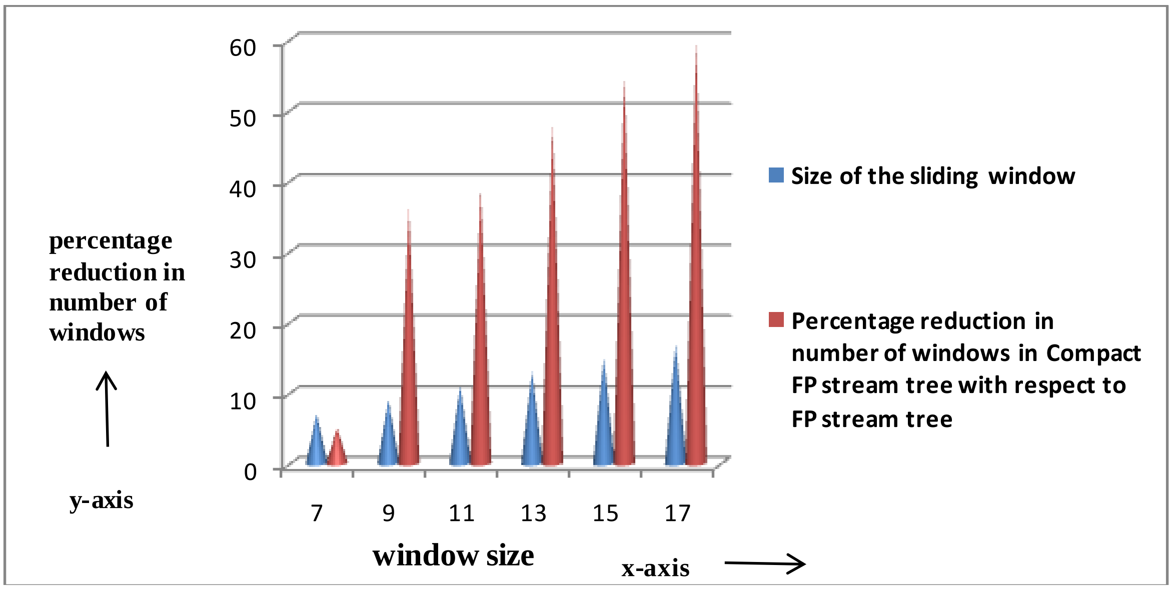


Figure 5.4: Chart showing percentage reduction in size of tilted time window to be maintained by Compact FP stream tree.

The graph given in Figure 5.4 shows the percentage reduction in amount of the tilted time window to be maintained at nodes of Compact FP stream tree with respect to FP stream tree. The graph is constructed by varying the window size.

5.3.2 Comparison of space reqirement for FP stream tree and Compact FP stream tree for the sensor based data.

Table 5.2 Space requirement by FP stream tree and Compact FP stream tree

Window	**Size of FP Stream Tree(bytes)**	**Size of compact FP stream tree(bytes)**	**Percentage Reduction in Size**
9	8815392	6569136	25%
11	8815392	5375616	39%
13	8815392	4547184	49%
15	8815392	3941472	55%
17	8815392	3477648	60%

The Table 5.2 shows the comparison of space requirement between FP stream tree and compact FP stream tree. The comparison is carried out by varying the window size and keeping the threshold to be 10. The above table shows that with the increasing the size of the window the performance of Compact FP stream tree gradually increases. Thus Compact FP stream tree requires less space as compared to FP stream tree.

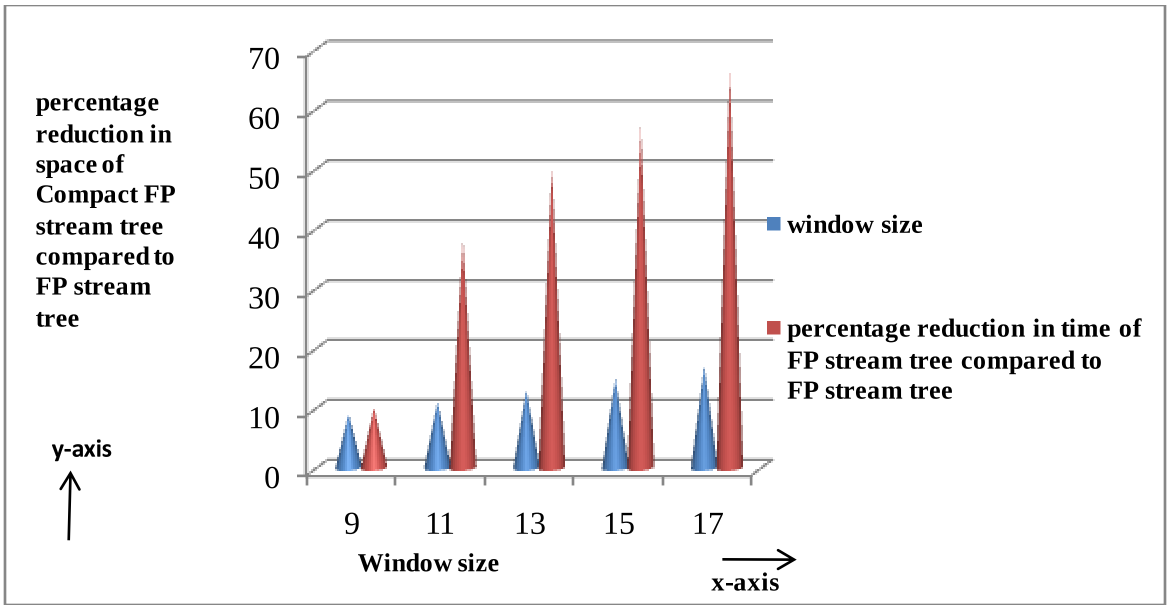


Figure 5.5: Chart showing percentage reduction in space requirement between FP stream tree and Compact FP stream tree.

The graph in Figure 5.5 shows the reduction in space requirement of Compact FP stream tree compared to FP stream tree. The graph is constructed by varying window size.

5.3.3Comparison of time requirement by FP stream tree and Compact FP stream tree for the sensor based data.

The Table 5.3 shows the comparison of time requirement between FP stream tree and Compact FP stream tree. The comparison is carried out by varying the window size and keeping the threshold to be 10. The above table shows that with the increasing the size of the window the performance of Compact FP stream tree gradually increases. Thus Compact FP stream tree requires less time as compared to FP stream tree.

Table 5.3 Time requirement by FP stream tree and Compact FP stream tree

Window	Time requirement for FP stream tree(in nanoseconds)	Time requiement for compact FP stream tree(in nanoseconds)	Percentage Reduction in Time
9	7098533081	6411111222	10%
11	7188275420	4506723126	38%
13	6972364112	3472862153	50%
15	6862267104	2994140361	57%
17	6998243084	2395539701	66%

The Table 5.3 shows the comparison of time requirement between FP stream tree and Compact FP stream tree. The comparison is carried out by varying the window size and keeping the threshold to be 10. The above table shows that with the increasing the size of the window the performance of Compact FP stream tree gradually increases. Thus Compact FP stream tree requires less time as compared to FP stream tree.

The graph in Figure 5.6 shows the reduction in time requirement of Compact FP stream tree compared to FP stream tree. The graph is constructed by varying window size.

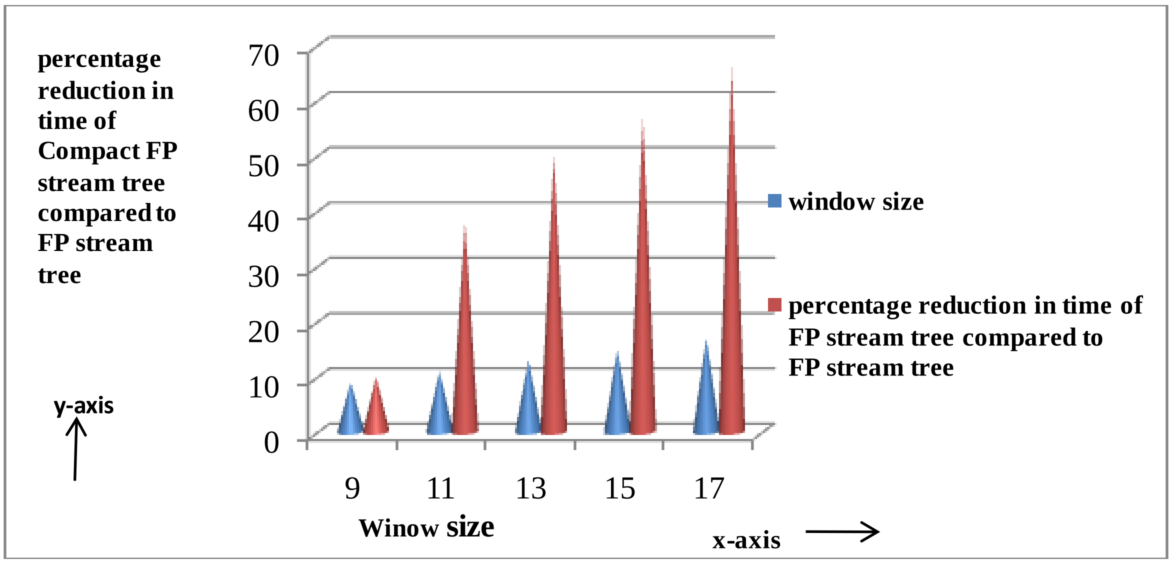


Figure 5.6: Chart showing percentage reduction in time requirement between FP stream tree and Compact FP stream tree.

6. CONCLUSION

The proposed Compact FP Stream Tree maintains the tilted time windows at the tail node and at the partition window. It uses linked list based tilted time window thus the amount of tilted time window to be maintained at the nodes need not be specified in advance. It maintains history of information over a period of time. Experimental results on Sensor data set shows that the amount of windows needed to be maintained in Compact FP stream tree is much less as compared to FP stream tree. The time and space reqirement of Compact FP stream tree is less as compared to FP stream tree. Thus Compact FP stream finds frequent pattern over a period of time with compact data structure and in a time efficient manner.

ACKNOWLEDGEMENT

We would like to thank reference authors and also like to thank the anonymous reviewers, whose comments and suggestions have helped them to improve the quality of the original manuscript.

REFERENCES

1. C. Giannella; J. Han, J. Pei; X. Yan; P.S. Yu, "Mining frequent patterns in data streams at multiple time granularities, in: Data Mining: Next Generation Challenges and Future Directions" , AAAI/MIT Press, 2004 (Chapter 6).
2. S.K Tanbeer; C.F. Ahmed; B. Jeong, Y. Lee; "Sliding window-based frequent pattern mining over data streams", Information Sciences Elsevier, Vol. 179, Issue 22, pp. 3843 - 3865, 2009.
3. P. Zhang; X. Zhu; J. Tan; L. Guo, "Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams", IEEE International Conference on Data Mining, Sydney,13-17 Dec, 2010
4. C. Lin; D.Y Chiu; Y.H. Wu; A. Chen, "Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window", In Proc. Society of industrial and applied mathematics(SIAM) International Conference on Data Mining ,California,21-23 April,2005.
5. J. Gao; W. Fan; J. Han; P. S. Yu , "A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions", Society for industrial and applied mathematics(SIAM) International Conference on Data Mining, Minneapolis, April 2007
6. C. Li; K.F. Jea; R.P. Lin; S.F. Yen; C.W. Hsu, "Mining frequent patterns from dynamic data streams with data load Management" , Journal of Systems and Software ,vol 85,Issue 6, pp 1346-1362, 2012
7. M. kholghi; M. keyvanpour,"An analytical framework for Data stream mining techniques Based on challenges and Requirements", International journal of engineering science and technology (IJEST), vol. 3, no. 3, march 2011
8. 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.[12] F.-Y. Ye, J.-D. Wang, B.-L. Shao, New algorithm for mining frequent itemsets in sparse database, in: Proc. the Fourth International Conference on Machine Learning and Cybernetics, 2005, pp. 1554–1558.
9. S.K. Tanbeer, C.F. Ahmed, B.-S. Jeong, Y.-K. Lee, "CP-tree: a tree structure for single-pass frequent pattern mining", in: T. Washio et al. (Eds.), Proc. PAKDD, 2008, pp. 1022–1027.

10. P. Zhang, X. Zhu, Y. Shi, L. Guo, and X. Wu, "Robust Ensemble Learning for Mining Noisy Data Streams*",Decision Support Systems,* Vol. 50, Issue 2, pp: 469-479,2011.
11. C.K.-S Leung; Q.I. Khan; "Efficient mining of constrained frequent patterns from streams", in: Proc. 10th International Database Engineering and Applications Symposium, 2006.
12. X. Zhi-Jun, C. Hong, C. Li, "An efficient algorithm for frequent itemset mining on data streams", in: Proc. ICDM, 2006, pp. 474–491.
13. http://www.cse.fau.edu/~xqzhu/Stream/sensor.arff
14. G.K Gupta, "Introduction to Data Mining with case studies", PHI publication, 2nd edition, 2006
15. J. Han; M. kamber; J. Pei, "Data mining concepts and techniques" -The Morgan Kaufmann series in Data Management Systems, 3rd edition, 2012
16. V. pudi; P. R. Krishna, "Data Mining", Oxford University Press, 1st edition, 2009